Federated Meta-Location Learning for Fine-Grained Location Prediction

Xiaopeng Jiang * Shuai Zhao † Guy Jacobson ‡ Rittwik Jana § Wen-Ling Hsu ‡
Manoop Talasila ‡ Syed Anwar Aftab ‡ Yi Chen † Cristian Borcea *
* Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA
† Martin Tuchman School of Management, New Jersey Institute of Technology, Newark, NJ, USA
‡ AT&T Labs, Bedminster, NJ, USA, § VMware, Florham Park, NJ, USA
Email:{xj8,sz255,yi.chen,borcea}@njit.edu,{guy,hsu,talasila,anwar}@research.att.com,rjana@vmware.com

***Abstract*—Fine-grained location prediction on smart phones can be used to improve app/system performance. Application scenarios include video quality adaptation as a function of the 5G network quality at predicted user locations, and augmented reality apps that speed up content rendering based on predicted user locations. Such use cases require prediction error in the same range as the GPS error, and no existing works on location prediction can achieve this level of accuracy. We propose Federated Meta-Location Learning (FMLL) on smart phones for fine-grained location prediction, based on GPS traces collected on the phones. FMLL has three components: a meta-location generation module, a prediction model, and a federated learning framework. The meta-location generation module represents the user location data as relative points in an abstract 2D space, which enables learning across different physical spaces. The model fuses Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Networks (CNN), where BiLSTM learns the speed and direction of the mobile users, and CNN learns information such as user movement preferences. The framework runs on the phones of the users and also on a server that coordinates learning from all users in the system. FMLL uses federated learning to protect user privacy and reduce bandwidth consumption. Our experimental results, using a dataset with over 600,000 users, demonstrate that FMLL outperforms baseline models in terms of prediction accuracy. We also demonstrate that FMLL works well in conjunction with transfer learning, which enables model reusability. Finally, benchmark results on Android phones demonstrate FMLL's feasibility in real life.**

***Index Terms*—location prediction, federated learning, deep learning, smart phones**

I. Introduction

A system that achieves high accuracy for fine-grained user location prediction on smart phones can improve system or app performance [34]. For example, accurate location prediction can be utilized in 5G networks at every time scale and across all layers of the protocol stack [5]. Since 5G performance is sensitive to small changes in location, the phone could use a map annotated with the location-based quality of wireless network to adapt video quality as a function of the predicted user locations. Augmented reality apps are delay-sensitive and can benefit from fine-grained location prediction to speed up content rendering. Yet another example is context-aware apps that need to adapt in advance based on where the user will move next, such as location-based gaming or advertising. For instance, location-based gaming could adapt in real-time based on the predicted user locations to be more interesting or challenging.

978-1-6654-3902-2/21/$31.00 2021 © IEEE

Existing location prediction systems cannot be used in such scenarios. Most location prediction research focuses on Place ID prediction. They work either at large spatial scales or at large time scales. For example, works for destination prediction [4], [11], [18], [27], place-label prediction [3], [19], [28], and Place of Interest (POI) prediction [6], [7], [14], [16], [17], [24], [25], [29] have poor spatial accuracy. We are aware of one study predicting location at small time-scale (e.g., predict where the user will be in several minutes), but the location error is in the order of hundreds of meters [22]. There are additional works that focus on small time-scale check-in POI prediction [30], [33], but they do not work for fine-grained locations or for every location in a road network.

The goal of this paper is to design a system for fine-grained location prediction from GPS traces that works on the users' phones. In our work, the term fine-grained refers to both spatial and temporal scales. Specifically, we aim to achieve high prediction accuracy, with prediction errors within the range of GPS errors. Furthermore, we want our system to be able to predict any potential locations of the users, not just important places identified by Place IDs as it is done by existing works. We focus on pedestrians and bicyclists, instead of users in cars or in public transportation systems, because their less predictable behavior makes the problem more difficult. In addition, their lower speeds and ability to stop whenever they want are expected to enable more applications of predictions. We also want to be able to predict at minute-scale (e.g., predict with a temporal step of one minute for $1, 2, ..., n$ minutes ahead). For example, we want to predict where a pedestrian will be in 5 minutes with a 10m spatial error.

Our system would have limited usability if it could predict only places that have been visited previously by the user. The system can be improved by training the prediction model with data collected by all the users who adopt the system. While sharing location data across users will improve prediction accuracy, a naive method using GPS traces directly in centralized training is unlikely to be accepted by users due to privacy concerns [15]. If users' location traces are disclosed, the identities of the users can be inferred even if pseudonyms

are used [9], [31]. This is due to the fact that location can contain identity information [20]. Thus, the system needs to also provide location privacy protection.

To summarize, we want to build a location prediction system that satisfies the following requirements: (R1) achieves high prediction accuracy at a fine-grained spatio-temporal scale; (R2) works well for pedestrians and bicyclists; (R3) works in places that have not been visited before by the owners of the phones invoking the prediction there; and (R4) protects user location privacy. To the best of our knowledge, there is no existing work that satisfies all these requirements.

This paper presents Federated Meta-Location Learning (FMLL) that satisfies all these requirements. FMLL has three main components: a meta-location generation module, a prediction model, and a federated learning framework. The meta-location generation module represents user location as relative points in an abstract 2D space, which is a grid with fixed-size cells. Meta-locations enable training on data received from all users, even when they visit different physical locations. Meta-locations also scale all data to the same range and avoid the bias introduced by data with high longitude and latitude values, which weigh more during the deep learning optimization. Our novel prediction model is trained on input derived from meta-locations. The model uses Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Networks (CNN), where BiLSTM learns the speed and direction of the mobile users, and CNN learns information such as user movement preferences. These two components are fused into a dense network with softmax activation. The federated learning framework allows the system to train on data from all users, while protecting user privacy.

Both the meta-location and the prediction model contribute to a superior prediction accuracy for pedestrians and bicyclists, satisfying requirements R1 and R2. The learning framework runs on the phones of the users and on a server that coordinates learning from all users in the system. It helps to satisfy R3 because a user can benefit from the learning on other users' phones with locations not visited by this user. Meta-location also contributes to R3 because it can extract repeated patterns, even when the physical locations are different. In FMLL, privacy is protected by combining federated learning (FL) [2] with our meta-location generation. This satisfies requirement R4. FL trains the models locally on each phone and then computes a global model at the server by aggregating the gradients of the local models. In this way, the server never gets access to the raw data. However, the gradients of the local models may still leak private location information if the FL model uses physical location data [26]. This problem is substantially mitigated by using meta-locations, because similar meta-locations may be generated from different physical locations, making the identification of physical locations at the server more difficult.

Although FMLL is designed to work on phones, its prediction model based on meta-location can also be used in the data center by network and service providers that already have user location data. For example, cellular network providers can employ the model to optimize handover in 5G. Similarly, they can use the location predictions to dynamically optimize the weights of antenna elements in 5G for best signal coverage while incurring minimum interference from other users.

Our experimental results, using a dataset with over 600,000 users, demonstrate that FMLL outperforms baseline models in terms of prediction accuracy for pedestrians and bicyclists respectively. We also demonstrate model reusability on another dataset, using FMLL with transfer learning [32]. We benchmarked the model on Android phones, and the results demonstrate that both training and inference are feasible in terms of execution time and battery consumption.

The rest of the paper is organized as follows. Section II discusses related work. Section III presents the meta-location generation. Section IV details the prediction model. Section V presents the federated, privacy-preserving learning framework. Section VI describes the datasets and the meta-location preprocessing. Section VII shows the experimental evaluation. The paper concludes in Section VIII.

II. Related Work

Early exploration of location prediction adapted Markov Chains and Hidden Markov Models [34]. Conventional machine learning (ML) methods, such as Bayesian networks [3], Support Vector Machines (SVM) [19], and tree-based models [11], were also applied for location prediction. Due to the limited information extraction capability of these models, the performance suffered.

More recently, researchers have started to exploit deep learning (DL) techniques for location prediction by treating it as a time series prediction problem. In a taxi destination prediction competition, de Brébisson et al. [4] tested several DL models, including MLP, LSTM, Bidirectional-RNN and Memory Network. Overall, the best model was Bidirectional-RNN with a time window covering 5 successive GPS points. In our case, given the need for fine-grained temporal scale, RNN-based methods alone cannot work well because they do not capture information such as road network characteristics and user preferences. Another obstacle to directly adopting RNN-based methods is that the transition from one location to another cannot happen between any two locations [27]. We overcome this by defining a reachable region centered at the current location and bounded by the traveling speed. We also differ in terms of privacy requirements. FMLL uses meta-locations instead of physical locations. In addition to improved privacy, this allows FMLL to easily scale uniformly among all users without losing the speed information, which is difficult when using physical locations.

The trajectory of movement on a map can be naturally processed with CNN-based methods. Lv et al. [18] proposed T-CONV, and performed better than Bidirectional-RNN [4] in the taxi destination prediction problem. The method uses trajectory data to mark the visited cells in a grid-like space, but does not incorporate the visit frequencies at specific locations. The CNN component of FMLL, on the other hand, incorporates visit frequency, which helps to improve prediction

accuracy. Zhang et al. [35] treated crowd inflow and outflow of grid cells in a city as a two-channel image-like matrix, and used CNNs for crowd flow prediction. This is a different problem from ours, but we share the ideas of visit frequencies for grid cells, and further extend the idea to represent the output as reachable grid cells.

Recent research [6], [17] applied state-of-art DL methods on POI IDs prediction. These works seem close to ours in terms of predicting human mobility. However, their problem definition is completely different, and their models use mechanisms that cannot work well for our problem. In Section VII, we adapted them for fine-grained location prediction and evaluated their performance. Section VII will further discuss the reasons for their inferior performance in fine-grained location prediction.

None of the studies discussed so far attempted to provide location privacy. Current privacy-preserving techniques in ML, such as differential privacy (DP), FL, and cryptographic methods could be applied for our problem, but have limitations. DP requires that computations be insensitive to changes in any particular individual's record, thereby restricting data leaks through the results. However, recent studies show record-level DP fails to address information leakage attacks [10]. A study by Graepel et al. [8] demonstrated machine learning on encrypted data using homomorphic encryption, but there are trade-offs regarding computational complexity and prediction accuracy. FL enables learning on the mobile devices without sending the raw data/features to the server. However, recent studies [26] showed user-level privacy leakage against FL by a malicious server, which can exploit the parameters received from the users. FMLL uses FL, but mitigates such attacks by using meta-location, which decouples the model input (relative points in a 2D space) from the physical locations.

III. Meta-Location Generation

The fundamental information to predict location is travel direction and speed. The user movement preferences and road characteristics also help the prediction. The GPS trajectories of each user contain this information. FMLL on the phones processes the raw location data to generate the meta-location, which represents trajectories as relative points in an abstract 2D space. This section presents the process of meta-location generation and its benefits.

A. Raw Location Data

The raw location data is recorded by each phone using the embedded GPS sensor. Let $\mathbf{L_t} =< \mathbf{lat_t}, \mathbf{lon_t} >$ denote the latitude and longitude of a user at time t. FMLL learns based on the transportation mode, such as walking or bicycling. Therefore, only the data specific to the desired transportation mode is selected for further processing. In real world, if the transportation mode is not explicitly known, it can be inferred from accelerometer data on mobile devices [21].

B. Meta-Location Input for Prediction Model

The raw location data of each user is processed on their phone to produce meta-location as two types of inputs for the prediction model: *fixed-length sequences of relative points* and *historic region occupancy matrices* of the space considered for prediction. The input sequences contain the speed and direction information of the user trajectories. The occupancy matrices record frequently visited places and the most likely trajectories between these places. The inputs are computed offline (e.g., when the phones are charging) and can be updated over time based on new data to enable re-training.

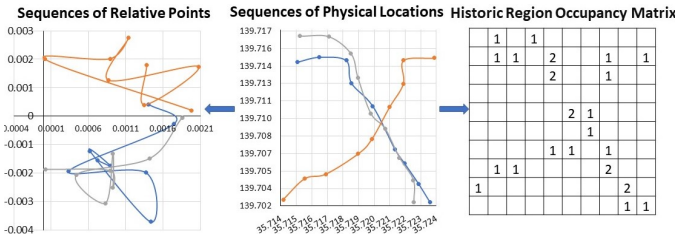

Fig. 1. Illustration of Meta-Location Generation

To generate the input sequences, FMLL splits the user trajectories into fixed-length sub-trajectories. The length in time of the trajectories is determined experimentally. Each sub-trajectory is transformed into a sequence of relative points in an abstract 2D space. The X and Y coordinates of relative points at time t are determined based on their offsets from the location at previous time step t-1. The location of the very first point in a trajectory session is excluded. A location offset is denoted as $\mathbf{\Delta L_t} =< \mathbf{lat_t} - \mathbf{lat_{t-1}}, \mathbf{lon_t} - \mathbf{lon_{t-1}} >$. An input sequence at time t that looks back k steps is denoted as $\mathbf{S_t} = (\mathbf{\Delta L_{t-k+1}}, \mathbf{\Delta L_{t-k+2}}, ..., \mathbf{\Delta L_{t-1}}, \mathbf{\Delta L_t})$. In its training, FMLL considers all possible k-length sequences, including overlapping sequences.

The historic region occupancy matrices are extracted from a historic occupancy matrix of the entire space (e.g., a city). FMLL divides the entire space into a grid of fixed-size cells, and each cell corresponds to an element in the historic occupancy matrix. Each element represents the number of visits of the user in its corresponding cell. The matrix represents the occupancy of a bounded region $\mathbf{R_t}$ with area A, which is centered at the physical location $\mathbf{L_t}$ at time t. $\mathbf{R_t}$ is divided into $M \times M$ fixed-size grid-cells, where A and M are predefined constants based on the maximum speed of users and the desired spatial granularity for the prediction. Each historic region occupancy matrix $\mathbf{H_t}$ is a $M \times M$ matrix, and it is extracted from the historic occupancy matrix for the entire space. Once extracted, this matrix is a meta-location input that does not maintain any relation with the physical locations that it represents. A matrix can implicitly tell if a road exists in a given cell (i.e., non-zero value for the corresponding matrix element) and can also tell if adjacent cells form routes taken frequently by the user.

Figure 1 illustrates how the meta-location input is generated from the sequences of physical locations. Let us note that in real world, the historic region occupancy matrix is generated from the numbers of visits to all grid cells which do not have to be temporal sequences of physical locations. We observe that the sequences of relative points do not resemble

the physical sequences, which helps with location privacy protection. Overall, different physical locations can be mapped to the same meta-locations. This not only helps repeated patterns to be extracted from different physical locations, but also makes it difficult for adversaries (i.e., the server in FL) to infer the physical locations.

C. Meta-Location Output for Prediction Model

The location to be predicted $\mathbf{L_{t+i}}$ is mapped into the region $\mathbf{R}$. FMLL builds a prediction matrix $\mathbf{Y_{t+i}}$, as shown below:

$$\mathbf{y_{i,j,t+i}} = \begin{cases} 1, & if\ \mathbf{L_{t+i}} \in \mathbf{R_{i,j}} \\ 0, & otherwise \end{cases} \quad (1)$$

where $\mathbf{y_{i,j,t+i}}$ is an element of $\mathbf{Y_{t+i}}$, and $\mathbf{R_{i,j}}$ $(\mathbf{1} \leq \mathbf{i,j} \leq \mathbf{M})$ is a cell in region $\mathbf{R}$. The meta-location output is formulated as a categorical class rather than a numerical value, so that we can set the spatial granularity of the prediction as a constant. Another reason for using categories is that the historic region occupancy matrix does not contain information to predict with spatial granularity beyond the grid-cell size. Overall, the output is a relative grid-cell, which is translated into a physical grid-cell on the user's phone.

D. Meta-Location Benefits

Using meta-location has four benefits. First, different meta-location sequences have the same magnitude, which is required for DL data. DL algorithms minimize the distance between two data points as a loss function. During this minimization, high-magnitude data weighs more than low-magnitude data, and it can lead to bias. For example, if physical location sequences are used directly, the training will focus on minimizing the loss for the data with high latitude and high longitude values. One way to avoid this problem is to scale every sequence to the same range [1]. However, scaling the location sequence will remove the traveling speed, which is necessary for location prediction. Since speed cannot be assumed constant for accurate prediction, there may not be an efficient mechanism to preserve it. With our meta-location generation, all meta-location sequences are in the same range and can be used directly in DL. This is especially important for FL training across all users.

The second benefit is the ability to change configuration parameters in data representation to perform prediction at different spatial granularity. For example, the grid-cell size can be 10m $\times$ 10m for pedestrians, and 40m $\times$ 40m for bicyclists. Instead of predicting the coordinates as arbitrary numerical values without a target granularity, with meta-location, we can formulate the output of the model categorically with specified granularity, and use accuracy to quantify the model performance. This also improves the model utility when an application requires certain spatial and temporal granularity from the model.

The third benefit is location privacy protection. This is achieved in conjunction with FL, which shares only the model gradients with the server. The data do not leave the phones because the learning happens on the phones. However, the gradients of the local models may still leak private location information if the FL model uses physical location data [26]. This problem is substantially mitigated by the use of meta-location. The meta-location input contains the essential information for location prediction, including speed, direction, and user movement preference, while not disclosing the physical location ($\mathbf{L_t} =< \mathbf{lat_t}, \mathbf{lon_t} >$) of the user to DL model.

The fourth benefit is the extraction of repeated patterns across different physical locations. When learning directly from different physical locations, the DL models encounter entirely different samples. However, because meta-location uses the same abstract 2D space, it may become similar for different physical locations. This speeds up learning because there will be more similar meta-location samples.

IV. FMLL Model

This section presents the formal problem definition for our model, and the description of the model architecture and its components.

A. Problem Definition

The problem is defined based on the meta-location input and output, defined in Section III. Let $\mathbf{S_t} \in \mathbf{R}^{2k}$ be the size-k sequence of relative points at time t for a given user. Let $\mathbf{H_t} \in \mathbf{Z}^{+M \times M}$ be the historic regional occupancy matrix of the same user, which is a square matrix of order M centered at the user location at time t. Our goal is to predict the relative location of this user $\hat{\mathbf{Y}}_{\mathbf{t+i}} \in \mathbf{Z_2}^{M \times M}$ for the future i^{th} timestamp.

$$\hat{\mathbf{Y}}_{\mathbf{t+i}} = \mathbf{F}(\mathbf{S_t}; \mathbf{H_t}) \quad (2)$$

where $\mathbf{F}$ is our DL model for location prediction. The predicted location is a cell in the $M \times M$ grid representation of the space surrounding the current location of the user.

B. Model Architecture

Meta-location provides a novel input and output formulation for the location prediction problem. Existing models cannot be used directly for three reasons. First, they [4], [6], [17], [18], [27] are designed to use either sequence input or matrix input, instead of both, so they are not able to take advantage of the benefits offered by the complementary meta-location inputs. Second, some problems, such as POI check-in prediction [6], [17], are naturally different from fine-grained location prediction. The user may check-in anywhere and anytime, and consequently their problem formulation does not bound the input and output spatially or temporarily. Their output candidates can be any POI IDs in the dataset, while fine-grained location prediction output can only be in the neighboring range of the current location. Third, the heuristics of designing some location prediction models is different from designing a fine-grained location prediction model. For example, the check-in location prediction [6], [17] should be designed to learn the popularity of POIs, while the fine-grained location prediction should learn the speed and direction of movement, road characteristics, etc.

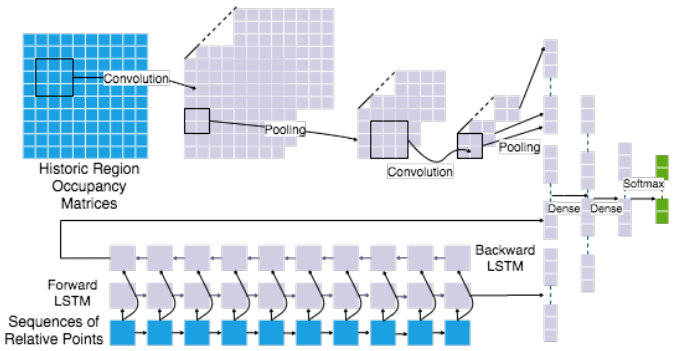

Fig. 2. Model Architecture

To learn effectively for fine-grained location prediction, we propose the FMLL model, whose architecture is shown in Figure 2. The model fuses BiLSTM and CNN, where BiLSTM learns the speed and direction of the user mobility from the sequences of relative points, and CNN learns user movement preferences and likely user routes from the historic region occupancy matrix. BiLSTM and CNN work in parallel. A densenet-type connection is used to fuse BiLSTM and CNN, and then softmax activation is adapted to output which grid-cell the user will be in. Batch normalization and dropout layers are also added in both BiLSTM and CNN to avoid overfitting, but for simplicity they are not shown in the figure. This architecture is designed to capture as much user-level information as possible.

For training, the phones use the meta-location input derived from physical locations, which can be pre-computed. For prediction, the sequence input is simple and can be generated in real-time, based on the last k recorded GPS locations. The historic region occupancy matrix, centered at the current location, is extracted in real-time from the pre-computed historic occupancy matrix for the entire space.

BiLSTM. Sequences of relative points contain the information of travel speed and direction. FMLL uses BiLSTM to learn them with a targeted spatial and temporal granularity. An LSTM unit is composed of a cell, an input gate, an output gate, and a forget gate. The cell remembers values over arbitrary time intervals, and the three gates regulate the flow of information into and out of the cell. In BiLSTM, one LSTM reads the relative location sequence forward, while a second LSTM reads it backward. The final two layers of hidden states are then concatenated, and the concatenation of these layers captures the speed and direction of users. To avoid overfitting, we perform both regular dropout and recurrent dropout. FMLL adopts BiLSTM for three reasons: (1) LSTM works well for sequence modeling. (2) BiLSTM augments data by using backward sequences in training. While backward sequences are not part of the dataset, they are real sequences that could occur. (3) Unlike unidirectional LSTM, which leads to a final internal state containing more information about the last points of a sequence (while the information about the first points is forgotten) [4], BiLSTM preserves the sequence information equally across the relative points.

CNN. Intuitively, knowing the exact speed and direction can determine the next location. However, the predicted speed and direction has to be tuned with other information for better learning. FMLL uses CNN on the historic region occupancy matrices, associated with the sequences fed into BiLSTM, to capture spatial features such as user movement preferences or the likelier route followed by a user between two points. CNN can learn this type of information because the historic region occupancy matrices contain information reflecting both occupancy frequency (explicit) and movement trajectory (implicit). Our CNN consists of batch normalization, convolution, max pooling, RELU activation, and dropout. With help from convolution and pooling, CNN is able to capture local connectivity and shift invariant. In our model, local connectivity can be the direction to which a user prefers to turn at a given intersection. The road characteristics are shift-invariant because the road network in a city usually follows the same urban design, and is similar in different areas.

Fusion. Although sequences of relative points and historic region occupancy matrices can be fit into next locations by BiLSTM and CNN, respectively, fusing the complementary information learnt from them can significantly improve the model performance. FMLL fuses the output layers from BiLSTM and CNN by concatenation, which allows for different-length outputs from BiLSTM and CNN. The concatenated output is fed into fully connected densenets, and the final output is computed by *softmax* activation, as shown in equation 3, where $\mathbf{k}$ corresponds to the $\mathbf{k_{th}}$ grid-cell, n = M $\times$ M is the total number of grid-cells, $\hat{\mathbf{y}}_\mathbf{k}$ is the $\mathbf{k_{th}}$ element of the output $\hat{\mathbf{Y}}$, and $\phi_\mathbf{k}$ is the $\mathbf{k_{th}}$ element of the final hidden layer before activation. The dense layers can gradually extract features of our desired length, and softmax converts them into probabilities. The output $\hat{\mathbf{Y}}$ contains the predicted probabilities of the future user location in each grid cell.

$$\hat{\mathbf{y}}_\mathbf{k} = \frac{\exp(\phi_\mathbf{k})}{\sum_{\mathbf{i=1}}^{\mathbf{n}} \exp(\phi_\mathbf{i})} \tag{3}$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} -\frac{1}{n} \sum_{i=1}^{n} \mathbf{y_i} \cdot \log(\hat{\mathbf{y}}_\mathbf{i}) \tag{4}$$

FMLL uses cross entropy loss for optimization, which is a standard function in multi-class classification. Therefore, the model learns the parameter $\mathbf{w}$ by minimizing the cross entropy loss measurement (equation 4). $\mathbf{y_i}$ is the $\mathbf{i_{th}}$ element of the ground truth $\mathbf{Y}$, where the grid cell of the user's future location is set to 1, and all others are 0.

V. FMLL Learning Framework

This section describes the FMLL framework that enables training and inference, while preserving the privacy of the user location. The section presents the system architecture for the framework, the operation stages of FMLL, and an enhanced training method.

A. System Architecture

The system architecture of our framework is shown in Figure 3. The framework software runs on a server and on the

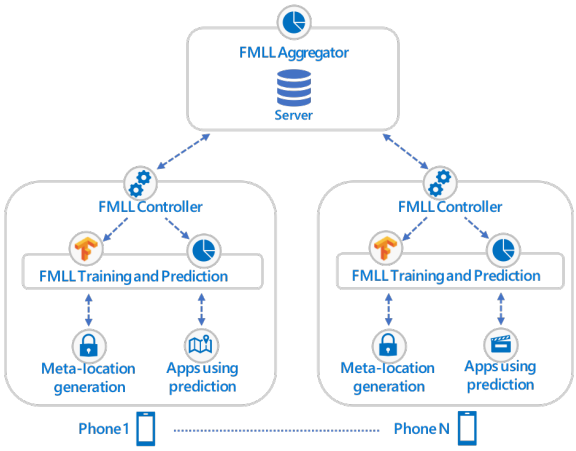

Fig. 3. FMLL System Architecture

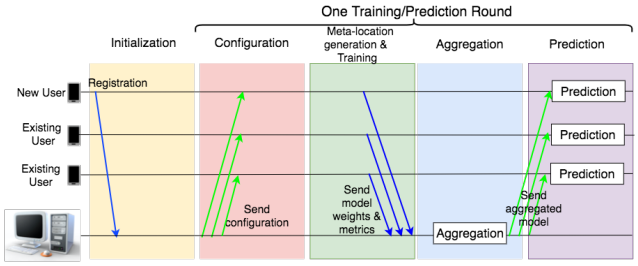

Fig. 4. Federated Learning Operation of FMLL

phones of the users, and it uses federated learning (FL) [2] for training across all users. The FMLL Controller on the phones mediates the communication between the server and the phones. The Meta-Location Generation module on the phones processes the physical location data and generates meta-location for training. The FMLL Training and Prediction module runs on the phones. This module performs local model training on the phones and then submits the model gradients to the server through the Controller. The FMLL Aggregator module at the server aggregates the gradients of the local models into a global model, and then distributes this model to the phones. When the OS or apps need a prediction, the Training and Prediction module is invoked. The output of the prediction is a meta-location, which is then converted into a physical location, with help from Meta-Location Generation module.

B. Operation Stages

To deploy the model in real-world and evolve it while the users collect location data over time, the FMLL learning framework has five computation and communication stages, illustrated in Figure 4. During these stages, the phones and the server jointly contribute to the model. The stages are executed periodically in rounds, similar to Google's FL framework [2]. In each round, the model is fine-tuned by re-training from the existing model. In the following, we detail each stage.

(1) *Initialization.* Newly participating phones are required to register with the server to ensure that the server knows when model gradients uploaded at different times come from the same user. This could further allow the server to remove potential malicious users who may inject fake data into the model. [1]

(2) *Configuration.* A training/prediction round starts with the configuration stage. The server informs the phones of the deadline to participate in training. This deadline is the end of stage 3. The server can select a subset of the connected phones based on the optimal number of participating phones in each round and the availability of training data. The server sends configuration parameters to the phones on how to generate meta-location for training. Parameters, such as sequence length, matrix size, grid size, may vary according to the desired spatial accuracy of the prediction. The server also sends the current global model parameters to each phone that did not participate in the previous training round, along with a training plan, such as gradient computation settings.

(3) *Meta-location Generation and Training.* Based on the configuration from the server, the phone performs meta-location generation. Then, it uses the global model received from the server to compute gradients based on its processed data. Finally, the phone sends the gradients back to the server.

(4) *Aggregation.* The server waits for the phones to report gradient updates, aggregates them using federated averaging, and updates its global model weights with aggregated gradients. Then, it deploys the model to the phones to ensure they have the latest model because they may not participate in a new round for a while.

(5) *Prediction.* The software on the phone can invoke the new FMLL model for predictions. Up to this stage, they use the old model from the previous round.

C. Training with Data Augmentation

A well-reported issue that restricts the performance of FL models is non Independent and Identically Distributed (IID) data distribution. FL trains on the dataset of each individual user. The datasets among different users may follow different distributions, due to user behavior differences, imbalanced class distribution, etc. While DL training can efficiently converge models with IID data, models trained in FL settings usually suffer from inferior performance [12]. To mitigate the non-IID issue, we leverage an advanced data augmentation mechanism inspired by Zhao et al. [37].

In this enhanced FL training, the data from a small percentage of users (e.g., less than 5%) are allocated as an augmentation dataset and made available to the aggregation server, and the server can sample and share it with the other users. This is usually the case when a small amount of users are willing to share their data with the server [36].

Training with data augmentation has three phases. First, the FMLL model is trained with the augmentation dataset at the server. This model is then distributed to the phones that will participate in FL training. Second, each phone selected in every round randomly picks a certain number of samples from the augmentation dataset and concatenates them with

[1]Protection against such malicious users is outside the scope of the paper.

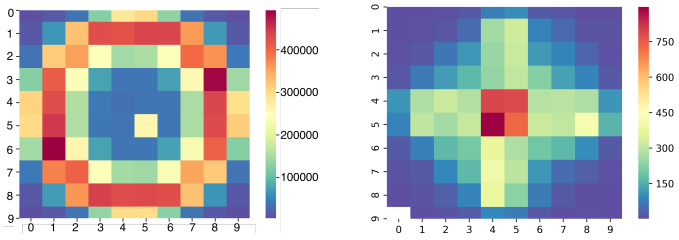

Fig. 5. Heatmap of possible next minute location in Open PFLOW (left) and Geolife (right) for 20m × 20m grid cells

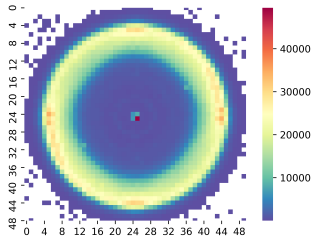

Fig. 6. Open PFLOW 5th min prediction heatmap of 20m × 20m grid cells

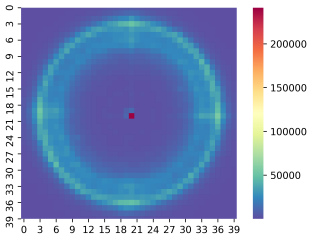

Fig. 7. Open PFLOW next minute prediction heatmap of 5m × 5m grid cells

its own dataset. Third, on-device training is conducted by initializing the model received from the server (trained with the augmentation dataset) and further training it with the augmented local data. The rest of the FL procedures are the same as in the basic FL technique. Thus, the local non-IID data of each user are augmented with IID data from the augmentation dataset, and the non-IID issue is mitigated.

VI. Dataset and Meta-location Preprocessing

This section describes the two real-world public datasets that we use in our evaluation, as well as the meta-location preprocessing to extract the inputs and outputs expected by our system.

A. Dataset Description

We use two datasets: Open PFLOW [13] and Geolife [38]. Most experiments use Open PFLOW, which is much larger. Geolife is used to demonstrate model reusability. Open PFLOW contains GPS trajectories that cover the Greater Tokyo area. It includes GPS data for 617,040 users, sampled every minute. The dataset covers typical movement patterns of people in the metropolitan area for one day, and it includes five transportation modes: stay, walk, vehicle, train, and bicycle. We select the data labeled "walk" and "bicyle", because their users' lower speeds and ability to stop whenever they want are expected to enable more applications of predictions. In real-world, the transportation mode can be inferred from accelerometer data on mobile devices [21]. Unless specified otherwise, the experiments are for pedestrian mobility. Bicycle data are used to demonstrate model reusability by using the pedestrian model to predict on bicycling data directly. Geolife [38] contains GPS trajectory data for 182 users over a five-year period. From this dataset, we selected the users (73) who labeled their trajectories with walking.

B. Meta-location Preprocessing

We choose a 200m × 200m region for both historic region occupancy matrices input and grid output, assuming a user can walk up to 100 meters in a minute. Let us recall that the region is centered at the current location of the user. For each experiment, we divide the region based on the accuracy we want to obtain. For example, we can divide the region into 100 cells of size 20m × 20m, and predict the location in one of these cells. The historic occupancy matrix is computed once and saved on mobile devices. The historic region occupancy matrix, centered at the current location, is extracted in real-time from the overall occupancy matrix.

For the fixed-length sequences of relative points, we choose a length of 9 (i.e., 9 locations, sampled one minute apart). Any walking session less than 9 minutes is dropped from the input. The sequence length should contain enough information to capture the direction and speed of movement, but it should not be too long, delaying the time when the first prediction can happen. We selected $length = 9$ as a good trade-off. In addition, most people in our datasets walk fewer than 15 minutes at once, so our length is practical from this point of view. To achieve quicker convergence, the relative points are scaled between -1 and 1.

Both datasets contain a large number of standing-still datapoints. We consider an input sequence to be standing-still if the user does not move at all in 9 minutes (the sequence length). Even though there is a "stay" transportation mode, about 50% of the walk data is standing-still. Our experiments presented in Section VII are without standing-still data. However, we also ran experiments with standing-still data (not included in the paper for the sake of brevity). These experiments proved the imbalance introduced by standing-still data is not a problem for large amounts of data.

Figure 5 shows the heatmap of the next minute possible location in Open PFLOW and Geolife after removal of standing-still data. The heatmap is generated by counting the number of samples in the dataset leading to each grid cell output. Since walking speed is similar for most people, in Open PFLOW, the location of next minute is most likely in the red/orange circle. In Geolife, because of fewer samples (three orders of magnitude fewer than in Open PFLOW), the circular pattern from typical walking speeds becomes less evident. The heatmap also illustrates the average number of samples per class in Open PFLOW is sufficient (i.e., 250,000). However, in Geolife, it is only 450 on average, and a lot of classes have less than 100 samples, which may be inadequate for training. Nevertheless, we keep this dataset to test if our model can be reused by leveraging transfer learning.

An accurate location prediction model requires large amounts of data. As we increase the region size to predict further into the future (Figure 6) or decrease the grid cell size for higher accuracy (Figure 7), the number of grid cells increases quadratically. Therefore, the number of cells

with insufficient samples increases correspondingly, and the prediction accuracy for both may suffer.

VII. Experimental Evaluation

Our evaluation has three goals. First, we assess the model's performance without FL. This allows a fair comparison between our model and the baseline models, which do not include privacy protection techniques such as FL. Second, we measure the performance of the federated solution of FMLL. Third, we test the feasibility of running FMLL on phones.

A. Model Performance Without FL

1) Implementation: FMLL is implemented using the Keras library. The output space of BiLSTM has dimensionality of 512 in each direction. Regular and recurrent dropout rates are set to 0.2 in BiLSTM. Two blocks of CNN are used sequentially with output space of 256 and 512, respectively. Both blocks include a convolutional layer with a 3×3 convolution window, a stride of 1, RELU activation, maximum pooling layer with size 2×2, batch normalization, and a dropout rate of 0.2. The outputs of BiLSTM and CNN are concatenated and fed into four densenet layers with output spaces of 1024, 512, 256, and 128, respectively. A dropout of 0.2 is added between the four dense layers. The final output is fitted by a densenet layer with softmax activation. We utilize the ADAM optimizer with learning rate of 0.001. The loss function is chosen as categorical cross entropy, which is commonly-used for multi-class classification.

2) Metrics: We choose three metrics for prediction performance: cross entropy loss, categorical accuracy, and weighted F1 score (i.e., the weighted average of F1 scores of each class divided by the number of samples in each class). A high F1 score indicates that both precision and recall are high.

3) Baseline models: Because the problem formulation of FMLL is unique and FMLL is designed to learn from meta-location instead of actual physical location, there are few comparable baseline candidates. We choose the baselines that can be adapted to learn from meta-location: Bidirectional RNN [4], T-CONV [18], and HO. Bidirectional RNN [4] is the best model in a taxi destination prediction competition. The state-of-the-art T-CONV model, used by Lv et al. [18], has recently outperformed the Bidirectional RNN in the taxi destination prediction. Highest Occupancy Model (HO) is a simple statistical prediction model. In HO, the location in the next minute is predicted as the most occupied grid cell in the historic region occupancy matrix. If there are multiple grid-cells with the same highest occupancy, HO will randomly select one of them.

To adapt the baseline models to be comparable with FMLL, we use the same meta-location input as FMLL, which are fixed-length sequences of relative points and historic region occupancy matrices. The dimensions of the hidden states and window sizes are chosen to be the same as in FMLL, and eventually fed into the same output layers. In this way, the baselines can also demonstrate the contributions of each individual component of FMLL, and the benefits of fusing them.

We considered and tested two other state-of-the-art POI ID prediction models for fine-grained location prediction, ST-RNN [17] and DeepMove [6]. These models use *recall@k* as metric, which is the the proportion of correct predictions found in the top-k recommendations. The values of recall@10 when applying these models to fine-grained location prediction are less than 5%.

The performance of these models is so poor in our settings because they are very different from FMLL in their problem formulation and design. First, these models use physical locations, instead of meta-locations. With physical locations, there are few similar trajectory samples across users. Without repeated patterns, the deep learning models cannot learn. Second, to predict the POI ID in a given time frame, these models formulate the problem as a recommender system that predicts a ranking of possible POI IDs a user will check in. On the other hand, our problem is formulated as a precise binary prediction of whether the user will be at a location cell in the grid or not. Third, these models do not consider spatial granularity, because a POI ID, such as a mall, typically covers a large area. They also do not consider temporal granularity, as they predict the next POI IDs without knowing when the user will visit that POI. Unlike these models, FMLL bounds the spatial and temporal granularity in the prediction and learns features such as speed and direction of travel, user preferences, and road characteristics. FMLL is unique in its fine-grained prediction across both spatial and temporal scales.

4) Experimental Settings: The experiments are conducted on a Ubuntu Linux cluster (CPU: Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz with 512GB memory, GPU: 4 NVIDIA P100-SXM2 with 64GB total memory), and the training and testing run with GPU acceleration. The ratio of training, validation, and testing datasets is 4:1:1. We choose a batch size of 256. Early stopping is used to stop the training earlier if the accuracy has not improved in the last 10 epochs for the validation dataset. The users are simulated in this system by replaying their location traces.

5) Results: Unless specified otherwise, the experiments are for pedestrian mobility, using Open PFLOW, and predicting one minute ahead for 20m $\times$ 20m grid-cells. As discussed in Section I, such short-term prediction can have significant benefits in real life.

Comparison with baseline models. Table I shows the prediction metrics for FMLL without FL and the baseline models. FMLL outperforms the baselines in both loss and accuracy. The weighted F1 score indicates that both precision and recall are high in FMLL. We notice that neither BiRNN nor T-CONV performs well. However, by fusing BiLSTM with CNN, FMLL leads to substantially better performance. This is because each of them captures a different type of information: speed and direction of traveling (BiLSTM) and movement trajectory and preferences (CNN).

Performance vs. spatial scale. Figure 8 shows how FMLL's accuracy varies with spatial scale (i.e., grid-cell size). Even

TABLE I
PERFORMANCE OF FMLL W/O FL AND BASELINES

Model	Metrics		
	Loss	Accuracy	Weighted F1
HO	NA	0.329	0.382
BiRNN	2.859	0.222	0.188
T-CONV	0.900	0.501	0.469
FMLL	**0.157**	**0.955**	**0.955**

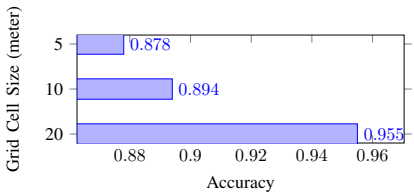

Fig. 8. Prediction accuracy as a function of grid-cell size

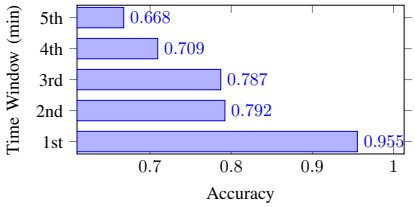

Fig. 9. Prediction accuracy as a function of time windows

though the accuracy decreases as the size of the cell decreases, FMLL can still achieve good performance (i.e., 87.8% accuracy) for 5m $\times$ 5m cells.

Performance vs. time scale. Figure 9 shows how FMLL's accuracy varies over time (i.e., predict x minutes ahead). The performance decreases over time, but we believe it is acceptable up to 5 minutes ahead. In addition to loss accumulation, there are two reasons for the decrease in accuracy over time. First, the user data is just for one day, and we cannot capture many recurring mobility patterns. Second, pedestrian mobility is inherently difficult to predict. Nevertheless, short-term prediction can be beneficial to many applications. For example, adjusting bit rate video streaming based on 5G coverage prediction can allow buffering up enough video while coverage is still good to avoid quality degradation due to predicted poor coverage in the next few minutes.

Model Reusability. A model is reusable when it can be used directly or in conjunction with transfer learning (TL) on another dataset. Currently, this property exists mainly in image recognition and natural language processing because few models can satisfy it.

We demonstrate that our model pre-trained on the pedestrian data from Open PFLOW works for bicyclist data from the same dataset, even without TL. We also demonstrate that TL works well in conjunction with FMLL when applied to the Geolife dataset. We believe that the main reason for FMLL's reusability is its meta-location, which makes it less location-specific.

TABLE II
FMLL W/O FL PRE-TRAINED ON PEDESTRIAN DATA AND USED TO PREDICT ON BICYCLING DATA W/O TL

Model	Metrics		
	Loss	Accuracy	Weighted F1
Walking	0.157	0.955	0.955
Bicycling	0.955	0.853	0.849

TABLE III
FMLL PERFORMANCE ON GEOLIFE DATASET: GEOLIFE ALONE VS. TL FROM OPEN PFLOW TO GEOLIFE

Model	Metrics		
	Loss	Accuracy	Weighted F1
Geolife alone	2.291	0.400	0.390
TL from Open PFLOW to Geolife	2.190	0.448	0.441

Table II shows the performance obtained when testing the pre-trained FMLL (i.e., trained only with pedestrian data) on bicycling data. For comparison, we also show the accuracy when testing on pedestrian data. Because bicycling speed is approximately four times walking speed, the sequence input is scaled down four times and the matrix input is scaled up four times to match the magnitude of pedestrian data. The results show good performance, with an accuracy of 85.3%, which demonstrates the model's reusability.

Table III shows the performance of FMLL over the Geolife dataset in two cases: a model trained directly on Geolife, and a TL model pre-trained with Open PFLOW and fine-tuned with Geolife. The results demonstrate the reusability of FMLL because the TL model from Open PFLOW to Geolife achieves 12.2% higher accuracy than the model trained on the Geolife dataset alone. This result is surprising, but it is explained by the fact that Open PFLOW is a much larger dataset than Geolife. We also observe, as expected, that training on Geolife leads to low accuracy. While the accuracy of the model pre-trained on Open PFLOW is significantly better, it is still not good in absolute terms. The reason is that the two datasets cover very different road networks. There are two types of urban planning for road networks, either as grid or circles, and these two types are quite different. Therefore, a model trained on one type cannot work very well on the other type.

B. Model Performance with FL

1) Implementation and Settings: FMLL with FL is implemented using TensorFlow Federated (TFF). For simplicity, we used the default federated averaging algorithm in TFF [2], instead of more sophisticated averaging mechanisms that may help FL's accuracy [23]. The same cluster is used for the experiments, but we could not use GPU acceleration because the current version of TFF (0.9.0) is not optimized for GPUs.

To simulate user participation in one FL round, we randomly sample 320 users every round for training. Before the training, we set aside some random users' data for testing, and these users are not selected for training. Because the number of samples per user is very limited (from 1 to 150, with an average of 60 in Open PFLOW), the batch size is set to 32.

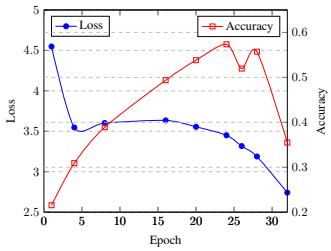

Fig. 10. Loss and accuracy over epochs with 40 rounds of training

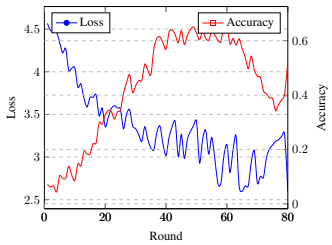

Fig. 11. Loss and accuracy over rounds with 24 epochs of training

TABLE IV
FMLL WITH FL PERFORMANCE

Training method	Metrics		
	Loss	Accuracy	Weighted F1
FMLL w/ FL without data augmentation	3.016	0.664	0.631
FMLL w/o FL on augmentation dataset alone	2.652	0.792	0.815
FMLL w/ FL with data augmentation	**2.432**	**0.842**	**0.853**

The number of users per round and the batch size are set experimentally for the best performance. All the other model settings are the same as in FMLL without FL.

2) Results: Due to the lack of a centralized validation dataset in FL, it is difficult to know when underfitting or overfitting occurs. Also, the lack of validation makes it impossible to determine the optimum number of epochs by early stopping. Therefore, in this experimental evaluation, we first show the model performance as the number of training epochs increases. Next, using the optimum number of epochs to train FMLL, we train the model with an increasing number of rounds for the best performance possible.

Performance over number of epochs. Figure 10 shows the testing performance of the models trained for 40 rounds and varying the number of epochs. Similar to the training for the model without FL, the loss decreases as the number of training epochs increase because the deep learning iteration process aims at minimizing the loss. Initially, the accuracy increases as the loss decreases. However, overfitting happens as the number of epochs keeps increasing, and accuracy decreases. The figure shows that the the best number of epochs is 24.

Performance over number of rounds. Figure 11 shows the testing performance of FMLL with FL models over rounds, with the number of training epochs fixed at 24. Unlike training over epochs, the performance fluctuates over training rounds. This is because every round is a sampling process based on the available phones, and there is no guarantee that the data quality from the sampled phones meets the demand to further improve the model. The fluctuation is minor at the beginning, but it increases over time. Similar with training over epochs, the loss decreases as the number of rounds increase due to re-training and the larger amount of sampled data. However, accuracy may not be further improved after a certain number of rounds because the model becomes overfitted. We observe that round 59 produces the model with the highest prediction accuracy.

Performance with data augmentation. Table IV shows the performance of FMLL with FL. This experiment uses the training with data augmentation described in Section V-C. The data from 5% of the users are allocated as an augmentation dataset. Let us note that the users who perform training and testing do not share any data with the aggregation server. During training, for each user selected in every round, we randomly pick 100 samples from the augmentation dataset, which are concatenated with the user dataset. The best performance is achieved in the 4th round, while training 20 epochs per user. FMLL with FL using data augmentation improves the accuracy by 20% over the original FMLL with FL. These results demonstrates that our data augmentation mechanism can effectively improve the performance of FMLL model in FL.

C. Model Benchmarks on Smart Phones

We implemented FMLL benchmark apps in Tensorflow and DL4J, and tested the feasibility of running FMLL on several smart phones for both training and inference in terms of latency, memory consumption, and battery consumption. The maximum RAM usage of FMLL is less than 200MB. The Google Pixel 3XL, a mid-range phone in 2021, can make a prediction in 46.41ms, and train 200 samples in 15min. With a full battery, the Google Pixel 3XL can execute 12.9 million predictions, or 203 rounds of training. The results show the system works well in real-life. More details are omitted due to space constraints.

VIII. CONCLUSION AND FUTURE WORK

This paper proposed FMLL, a novel system for fine-grained location prediction that protects user privacy. The main novelties of FMLL are its meta-location concept to represent physical data, its prediction model, and its privacy-preserving learning framework. Our experiments demonstrated good prediction accuracy, model reusability, and system feasibility on smart phones. While FMLL is designed for predictions on smart phones, its model can also be used by network/service providers in their data centers. In the future, FMLL could be incorporated directly into smart phone OSs to improve system and app performance using location prediction. In addition, its GPS-based location prediction can be fused with location prediction done by wireless network providers based on 5G signal fingerprinting techniques to further improve the end-user's experience in real-world applications such as augmented reality, mobile gaming, and video streaming.

ACKNOWLEDGEMENT

This research has been supported in part by the National Science Foundation (NSF) under Grants No. DGE 1565478 and DGE 2043104. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

REFERENCES

[1] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[2] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. *CoRR*, abs/1902.01046, 2019.

[3] M. Dash, K. K. Koo, J. B. Gomes, S. P. Krishnaswamy, D. Rugeles, and A. Shi-Nash. Next place prediction by understanding mobility patterns. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 469–474, March 2015.

[4] Alexandre De Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio. Artificial neural networks applied to taxi destination prediction. In *Proceedings of the 2015th International Conference on ECML PKDD Discovery Challenge - Volume 1526*, ECMLPKDDDC'15, pages 40–51, Aachen, Germany, Germany, 2015. CEUR-WS.org.

[5] R. Di Taranto, S. Muppirisetty, R. Raulefs, D. Slock, T. Svensson, and H. Wymeersch. Location-aware communications for 5g networks: How location information can improve scalability, latency, and robustness of 5g. *IEEE Signal Processing Magazine*, 31(6):102–112, Nov 2014.

[6] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. Deepmove: Predicting human mobility with attentional recurrent networks. In *Proceedings of the 2018 world wide web conference*, pages 1459–1468, 2018.

[7] Jie Feng, Can Rong, Funing Sun, Diansheng Guo, and Yong Li. Pmf: A privacy-preserving human mobility prediction framework via federated learning. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 4(1), March 2020.

[8] Thore Graepel, Kristin Lauter, and Michael Naehrig. Ml confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, pages 1–21. Springer, 2012.

[9] M. Gruteser and B. Hoh. On the anonymity of periodic location samples. In *Security in Pervasive Computing*, pages 179–192. Springer, 2005.

[10] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 603–618, 2017.

[11] Thomas Hoch. An ensemble learning approach for the kaggle taxi travel time prediction challenge. In *DC@ PKDD/ECML*, 2015.

[12] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[13] Takehiro Kashiyama, Yanbo Pang, and Yoshihide Sekimoto. Open pflow: Creation and evaluation of an open dataset for typical people mass movement in urban areas. *Transportation Research Part C: Emerging Technologies*, 85:249 – 267, 2017.

[14] Dejiang Kong and Fei Wu. Hst-lstm: A hierarchical spatial-temporal long-short term memory network for location prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2341–2347. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[15] John Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.

[16] C. H. Liu, Y. Wang, C. Piao, Z. Dai, Y. Yuan, G. Wang, and D. Wu. Time-aware location prediction by convolutional area-of-interest modeling and memory-augmented attentive lstm. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[17] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 194–200. AAAI Press, 2016.

[18] J. Lv, Q. Sun, Q. Li, and L. Moreira-Matias. Multi-scale and multi-scope convolutional neural networks for destination prediction of trajectories. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–12, 2019.

[19] B. T. Nguyen, N. V. Nguyen, N. T. Nguyen, and M. H. T. Tran. A potential approach for mobility prediction using gps data. In *2017 Seventh International Conference on Information Science and Technology (ICIST)*, pages 45–50, April 2017.

[20] R.A. Popa, A.J. Blumberg, H. Balakrishnan, and F.H. Li. Privacy and accountability for location-based aggregate statistics. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, pages 653–666. ACM, 2011.

[21] Muhammad Awais Shafique and Eiji Hato. Use of acceleration data for transportation mode prediction. *Transportation*, 42(1):163–188, 2015.

[22] R. Trasarti, R. Guidotti, A. Monreale, and F. Giannotti. Myway: Location prediction via mobility profiling. *Information Systems*, 64:350 – 367, 2017.

[23] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[24] P. Wang, H. Wang, H. Zhang, F. Lu, and S. Wu. A hybrid markov and lstm model for indoor location prediction. *IEEE Access*, 7:185928–185940, 2019.

[25] Yingzi Wang, Nicholas Jing Yuan, Defu Lian, Linli Xu, Xing Xie, Enhong Chen, and Yong Rui. Regularity and conformity: Location prediction using heterogeneous mobility data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1275–1284, New York, NY, USA, 2015. Association for Computing Machinery.

[26] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 2512–2520, April 2019.

[27] Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. Modeling trajectories with recurrent neural networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3083–3090, 2017.

[28] Q. Wu, X. Chen, Z. Zhou, and L. Chen. Mobile social data learning for user-centric location prediction with application in mobile edge service migration. *IEEE Internet of Things Journal*, 6(5):7737–7747, 2019.

[29] R. Wu, G. Luo, Q. Yang, and J. Shao. Learning individual moving preference and social interaction for location prediction. *IEEE Access*, 6:10675–10687, 2018.

[30] Shuai Xu, Jiuxin Cao, Phil Legg, Bo Liu, and Shancang Li. Venue2vec: An efficient embedding model for fine-grained user location prediction in geo-social networks. *IEEE Systems Journal*, 2019.

[31] T. Xu and Y. Cai. Feeling-based location privacy protection for location-based services. In *Proceedings of the 16th ACM conference on Computer and Communications Security (CCS)*, pages 348–357. ACM, 2009.

[32] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[33] Daniel Zhang, Yang Zhang, Qi Li, and Dong Wang. Sparse user check-in venue prediction by exploring latent decision contexts from location-based social networks. *IEEE transactions on Big Data*, 2019.

[34] H. Zhang and L. Dai. Mobility prediction: A survey on state-of-the-art schemes and future applications. *IEEE Access*, 7:802–822, 2019.

[35] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, and Xiuwen Yi. Dnn-based prediction model for spatio-temporal data. In *Proceedings of the 24th ACM SIGSPACIAL International Conference on Advances in Geographic Information Systems*, SIGSPACIAL '16, pages 92:1–92:4, New York, NY, USA, 2016. ACM.

[36] Shuai Zhao, Roshani Bharati, Cristian Borcea, and Yi Chen. Privacy-aware federated learning for page recommendation. In *Proceedings of 2020 IEEE International Conference on Big Data*, 2020.

[37] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[38] Y. Zheng, L. Wang, R. Zhang, X. Xie, and W. Ma. Geolife: Managing and understanding your past life over maps. In *The Ninth International Conference on Mobile Data Management (mdm 2008)*, pages 211–212, 2008.