Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 20(7): 11998–12023.
DOI: 10.3934/mbe.2023533
Received: 10 February 2023
Revised: 13 April 2023
Accepted: 24 April 2023
Published: 12 May 2023

Research article

Research on user recruitment algorithms based on user trajectory prediction with sparse mobile crowd sensing

Jing Zhang[1], Qianqian Wang[1,*], Ding Lang[2], Yuguang Xu[1], Hong-an Li[1] and Xuewen Li[3]

[1] School of Computer Science and Technology, Xi'an University of Science and Technology, Xian 710600, China

[2] School of Energy Enginnering, Xi'an University of Science and Technology, Xian 710600, China

[3] School of Safety Science and Engineering, Xi'an University of Science and Technology, Xian 710600, China

* **Correspondence:** Email: wangqianqian269@163.com.

Abstract: Sparse mobile crowd sensing saves perception cost by recruiting a small number of users to perceive data from a small number of sub-regions, and then inferring data from the remaining sub-regions. The data collected by different people on their respective trajectories have different values, and we can select participants who can collect high-value data based on their trajectory predictions. In this paper, we study two aspects of user trajectory prediction and user recruitment. First, we propose an STGCN-GRU user trajectory prediction algorithm, which uses the STGCN algorithm to extract features related to temporal and spatial information from the trajectory map, and then inputs the feature sequences into GRU for trajectory prediction, and this algorithm improves the accuracy of user trajectory prediction. Second, an ADQN (action DQN) user recruitment algorithm is proposed.The ADQN algorithm improves the objective function in DQN on the idea of reinforcement learning. The action with the maximum input value is found from the Q network, and then the output value of the objective function of the corresponding action Q network is found. This reduces the overestimation problem that occurs in Q networks and improves the accuracy of user recruitment. The experimental results show that the evaluation metrics FDE and ADE of the STGCN-GRU algorithm proposed in this paper are better than other representative algorithms. And the experiments on two real datasets verify the effectiveness of the ADQN user selection algorithm, which can effectively improve the accuracy of data inference under budget constraints.

Keywords: sparse mobile crowd sensing; STGCN-GRU; user selection; reinforcement learning

1. Introduction

Mobile swarm intelligence sensing is a promising sensing paradigm that allows a large number of mobile users with smart devices to perform a variety of sensing tasks, such as environmental monitoring [1], traffic congestion detection [2], urban infrastructure status inspection [3], and so on. In order to provide high-quality sensing services, traditional mobile swarm intelligence sensing applications need to recruit a large number of mobile users to cover most of the target sensing area. Obviously, this approach has a huge overhead and it is difficult to handle some sensing areas that lack mobile users. Therefore, some researchers have proposed a novel sensing paradigm that collects data from only a few sensing areas and then mines and exploits the spatio-temporal correlation of the sensed data to infer data from the remaining un-sensed areas using a data inference algorithm called sparse mobile crowd sensing [4]. In this way, sparse mobile crowd sensing can significantly reduce the number of required mobile users while still providing high quality sensing services.

In sparse mobile crowd sensing, a key issue is user recruitment. That is, the task initiator wants to recruit a limited number of users (under budget constraints) who are mobile enough to collect data from several important sensing areas they pass through to ensure a high degree of data inference accuracy and thus provide high quality sensing services.

The existing work of user recruitment mainly focuses on data inference, and ignores the mobility of users. Previous work often selects a large number of users to perceive data, which improves data accuracy but increases the perception cost, as people's daily life has a certain regularity, which leads to the daily trajectory of each person is basically stable. If we predict and select some effective users with known movement patterns, they can pass through some important sensing areas and collect data from them, and then use these more valuable sensing data to more accurately infer data from unperceived areas. Perception cost can be reduced.

We combine estimation and data inference in the hope of solving the sparse mobile crowd sensing user selection problem. The user's trajectory is first predicted, and then the user is selected by combining the user trajectory data.

The contributions of this paper are summarized as follows.

(1) We propose an STGCN-GRU user trajectory prediction algorithm using the STGCN (spatio-temporal graph convolution networks) algorithm for deep feature extraction in time and space. STGCN is used to solve the time series prediction problem in terms of trajectories, formalizing the problem on the graph and building a fully convolutional structure. It improves the efficiency of training prediction when dealing with large-scale data volume, and makes the model independent and parallel when testing. the GRU algorithm performs trajectory prediction on the input incoming feature sequences, and GRU merges cell states and hidden states, which is simpler than the LSTM network structure and also has good application performance.

(2) We propose an ADQN (action DQN) user recruitment algorithm. The ADQN algorithm improves the objective function in DQN on the idea of reinforcement learning. The action with the maximum input value is found from the Q network, and then the output value of the objective function of the Q network for the corresponding action is found. This prevents the Q network from over-predicting the value and speeds up the convergence. To a certain extent, it reduces the overestimation problem that occurs and improves the accuracy of user recruitment. The aim is to give a specific state to determine which users are the best choice to select in order to reduce the perceived

cost.

2. Related work

2.1. Sparse mobile crowd sensing

In swarm intelligence perception, we recruit users to collect information about the user's surroundings and other information using the smart devices they carry with them. In the physical world, most of the collected perceptual information usually exhibits strong spatio-temporal correlations. Therefore, from the data level, by mining and exploiting the spatio-temporal correlations of the sensed data, we can extend swarm intelligence perception to "sparse mobile crowd sensing", i.e., recruiting users to perceive data from some areas and using their spatio-temporal correlations to infer data from other unperceived areas, in order to significantly reduce perception consumption and ensure data accuracy.

In the practice of MCS (mobile crowd sensing), to ensure data quality while minimizing sensing cost, some MCS tasks involve inference algorithms to fill in the missing data of the sensing units, and compressed sensing has become the de facto choice of inference algorithms in MCS tasks. Wang et al. [5] formalized the sparse mobile crowd sensing model and proposed a framework with data inference, quality assessment and sub-region selection. Liu et al. [6] proposed to improve the accuracy of missing data inference in sparse mobile crowd sensing by using appropriate users. Tu et al. [7] proposed a participant recruitment strategy for sparse mobile crowd sensing based on reinforcement learning. They simulated the state, actions and rewards of a recruitment system and used a deep Q-network to determine which users are the best candidates for recruitment. Han et al. [8] modeled the spatio-temporal correlation in the collected sparse data, and performed a reasonable transformation and stitching of sparse data at different moments to keep the latest training data. Liu et al. [9] selected valuable locations to collect data based on active learning and used compressed sensing to infer data from unselected locations. Zhu et al. [10] collected data periodically by detecting vehicle to estimate urban traffic and design efficient data inference algorithms to achieve low inference error estimation under high data loss rate. Wang et al. [11] proposed an approximationally optimal greedy selection algorithm and a fast selection algorithm based on geographical location correlation, selecting and recruiting some users in the social network as initial seeds and using its influence to maximize the final coverage. Li et al. [12] proposed a dynamic user recruitment method aiming to maintain a high probability of perceived task completion with minimal perceptual cost. The incentive mechanism proposed by Gao et al. [13] allows perceived task initiators to provide additional bonuses to selected users based on the degree of task completion and prior performance to further motivate them. Pu et al. [14] proposed a self-organized mobile crowdsourcing system, "Crowdlet", where each user can actively post tasks to other users it encounters to obtain fast and high-quality feedback. Li et al. [15] started from the typical application of urban environmental monitoring, recruiting the optimal users under budget constraints to build an accurate monitoring map.

In addition to inference algorithms, sparse mobile crowd sensing abstracts other key research issues such as sub-region selection and quality assessment, etc. Later, privacy-preserving mechanisms were also added to sparse mobile crowd sensing [16]. In this paper, we focus more on the user recruitment problem with the aim of solving it using deep reinforcement learning techniques.

2.2. User trajectory prediction

With the continuous development of mobile positioning technology, its related application products, such as GPS (global positioning system) based devices and smartphones, help to obtain a large amount of trajectory information of mobile objects. The trajectory prediction of mobile objects is an important research direction in trajectory analysis mining, and the prediction of trajectories plays an increasingly important role in practical applications. By continuously improving the prediction methods to improve the prediction ability of trajectories, the traditional method is to extract frequent regions from the original trajectories and then can use trajectory clustering to do prediction analysis. For example, shen et al. [17] proposed a new trajectory clustering method based on submodular optimization for video motion segmentation; Yao et al. [18] proposed a low-dimensional representation method using learned trajectories to reexamine the trajectory clustering problem.

With the continuous research in deep learning, trajectory prediction is now more often done by feature extraction and model construction using neural networks. Jannik et al. [19] proposed a trajectory observation generation based on trajectories using image processing and map matching techniques; Xue et al. [20] proposed an LSTM-based network which considers the effects of social neighborhood and scene layout; Xie et al. [21] proposed an integrated approach for vehicle trajectory prediction; Fernando et al. [22] proposed a data association scheme based on predicted trajectories with minimal fragmentation to generate human-like trajectories; Woo et al. [23] proposed to predict vehicle trajectories by detecting the lane changes of surrounding vehicles; Deo et al. [24] investigated a unified framework for enclosing vehicle maneuver classification and motion prediction by utilizing multiple cues, i.e., estimated vehicle motion, understanding of typical movement patterns of highway traffic, and vehicle interactions. Yu et al. [25] proposed a spatio-temporal graph convolutional network (STGCN) model to solve the time-series prediction problem using a time- and space-domain convolutional structure with a smaller number of parameters. Guo et al. [26] proposed an attention mechanism-based spatio-temporal graph convolutional network (ASTGCN) model to solve the traffic data prediction problem and combined the idea of multiple components to capture the spatio-temporal correlation of data. Song et al. [27] effectively captured the spatio-temporal correlation of complex data by designing a spatio-temporal synchronous graph convolutional network (STSGCN) model using a synchronous modeling mechanism. Although these models achieved good results in spatio-temporal data mining, they did not fully consider the strong semantic nature of data in spatial dimension or the strong periodicity in temporal dimension. Luo et al. [28] proposed a semantic trajectory extraction method to demonstrate the effectiveness of semantic enhancement of spatio-temporal behavioral data.

2.3. Reinforcement learning

Reinforcement learning (RL) refers to learning a state-action mapping to maximize a numerically accumulated reward signal. Unlike dynamic programming, RL can be combined with model-free methods and is therefore particularly useful in observing long-term policy derivation in Markov decision process (MDP) settings. Value iteration computes the optimal state value function by iteratively improving the estimation of state values. Q-learning is a classical value iteration method that is widely used in MCS research to derive perceptual or pricing policies. In sparse mobile crowd sensing, we can consider the participant selection as a decision, by using reinforcement learning to

interact directly with the environment, ignore the complex data inference process and obtain the corresponding result, so it is more suitable to solve the continuous and complex perceptual region selection problem.

In recent years, many researchers have worked on combining deep learning with reinforcement learning to enhance its capabilities to solve specific problems in a variety of fields such as science and business [29]. Mnih et al. [30] proposed the first deep reinforcement learning model that successfully processed high-dimensional sensory inputs to directly learn control strategies. Silver et al. [31] applied deep reinforcement learning and Monte Carlo tree search and other techniques to propose the famous Go AI: Alpha Go. Hausknecht et al. [32] proposed to use a long short-term memory (LSTM) network to replace a fully connected layer in a deep reinforcement learning model, called deep recurrent reinforcement learning, and Lample et al. [33] proposed the first deep recurrent reinforcement learning architecture to deal with 3D environments in first-person shooter games and the partially observable states they involve. Li et al. [34] proposed a user selection method based on a combinatorial multi-armed gambling machine to maximize the overall benefit. The problem of perceptual tasks arriving at different time periods and uncertainty in the probability and time delay of users performing the tasks was addressed. Zhang et al. [35] proposed an online quality learning algorithm and task assignment based on the learned quality. Zhao et al. [36] proposed a privacy-preserving unknown user recruitment mechanism.They considered that the perceived quality of users is unknown and is sensitive information. The user's quality information is protected by differential privacy techniques, and a user recruitment mechanism based on the UCB algorithm and a user recruitment mechanism based on the -first algorithm are proposed to recruit users with higher perceived quality and lower perceived cost to perform tasks while protecting user privacy. Gao et al. [37] proposed two user recruitment algorithms based on combinatorial multi-armed gambling machines and analyzed the algorithms for regret values. The cases where the perceived quality of the user is unknown and the cases where both the perceived quality and cost are unknown are considered, respectively. Xiao et al. [38] defined the interaction between the server and the vehicle as a vehicle crowd-sensing game. Then, they proposed a q-learning based strategy to help servers and vehicles make optimal decisions for the dynamic game. In the three-stage user recruitment strategy of Liu et al. [6], they used reinforcement learning methods to select the most efficient sub-region for the current cycle in the sub-region selection stage.

3. Background

3.1. Compression perception

In sparse mobile crowd sensing, compressed perception has a wide range of applications in recovering spatio-temporal data. Using the low-rank property possessed by real matrices, we can infer and obtain the corresponding inference matrix based on the sparse collected data matrix: we will briefly introduce the principle of compressive perception below.

Any matrix can be decomposed into the form of Eq (1) by singular values, and we can use the r largest singular values to create an approximate matrix of rank r, as the following Eqs (2) and (3).

$$E_{m*t} = \sum_{i=1}^{min(m,t)} \sigma_i u_i v_i^t \tag{1}$$

$$\sum_{i=1}^{r} \sigma_i u_i v_i^t = \overline{E} \tag{2}$$

$$minrank(\overline{E}) s.t, \overline{E} \circ S = E^{'} \tag{3}$$

where $\circ$ denotes the matrix multiplication of the corresponding elements, $\overline{E}$ and $E^{'}$ are the data inference matrix, and the actual perception matrix, respectively, and r is the set matrix and S is the collection matrix. For subregion i, if the data of this subregion is collected in the perception cycle k of this recruitment cycle, then collect the elements of row i column k of the matrix r , $S_{i,k} = 1$ and 0 otherwise. Moreover, by singular value decomposition $E = LR^T$, we can transform the above problem from minimizing the rank to minimizing the F-parameterization of L and R. as the following Eq (4).

$$min\lambda(||L||_F^2 + ||R||_F^2) + ||LR^T \circ S - E^{'}||_F^2 \tag{4}$$

To better capture spatio-temporal correlations in perceptual data, we include temporal and spatial correlations in the optimization problem [35, 37], as the following Eq (5).

$$min\lambda_r(||L||_F^2 + ||R||_F^2) + ||LR^T \circ S - E^{'}||_F^2 + \lambda_s||\mathbb{S}(LR^T)||_F^2 + \lambda_t||(LR^T)\mathbb{T}^T||_F^2 \tag{5}$$

where λ_r, λ_s and λ_t are the weights for different correlations, $\mathbb{S}$ is the spatial-constrained matrix, and $\mathbb{T}$ is the time-constrained matrix. The $\mathbb{S}$ matrix controls the data correlation in different perceptual regions at the same time, and the $\mathbb{T}$ matrix controls the correlation at different times in the same perceptual region. During the computation of compressed sensing, we iterate on L the and R matrices using least squares to obtain $\overline{E} = LR^T$

3.2. Definition of the problem

(1) User trajectory prediction: After the coordinates of the trajectory of the pedestrians in the scene are known for a period of time in the past, the goal of the pedestrian trajectory prediction problem is to predict the trajectory paths of all pedestrians in a period of time in the future, and the multi-pedestrian trajectories in dynamic scenes have three kinds of interaction information, which are temporal interaction information, spatial interaction information and temporal association information. Spatial interaction information exists between different pedestrians at the same moment. There is temporal interaction information between the trajectories of different moments of the same pedestrian. Spatio-temporal association information exists between different pedestrians at different moments. It is known that there are N pedestrians in the scene and their trajectories at the past T_{obs} time points are obtained, and the trajectory data of individual pedestrian $i \in \{1, 2....., N\}$ at time $t \in \{1, 2....., T_{obs}\}$ are in the format$X_i^t = \{x_i^t, y_i^t\}$. In summary, for all trajectories of pedestrian i in the observation time is $X_I^{1:T_{obs}} = [X_i^1; X_i^2; ...X_i^{T_{obs}}]$, and the goal of this task is to predict the trajectory of pedestrian i in the future time T_{pred} in the premise of the trajectory in the observation time, denoted as $\hat{Y}_i^t = (\hat{x}_i^t, \hat{y}_i^t), t \in \{T_{obs+1}, ..., T_{obs+pred}\}$. X_i^t is the absolute position trajectory of pedestrians in the real scene, and due to the different sizes of different scenes, the absolute position trajectory is usually converted into the relative position trajectory ΔX_i^t as the following Eqs (6) and (7).

$$\Delta x_i^t = x_i^t - x_i^{t-1} \tag{6}$$

$$\Delta y_i^t = y_i^t - y_i^{t-1} \tag{7}$$

(2) User selection strategy: as shown in Figure 1, suppose we have n users moving in m regions. They can pass through multiple sub-regions in a perception cycle. There are w recruitment cycles in total. Our aim is to select k users among n users to reduce the perceptual cost. We represent the way users are selected as a vector. We assume that if they are selected, they can successfully perceive the covered sub-region data. Finally we infer the current and next sensed region data based on the data collected by the users from the previous sensing cycles. and minimize the data inference error. The process of selecting users to collect data is shown in Figure 2.

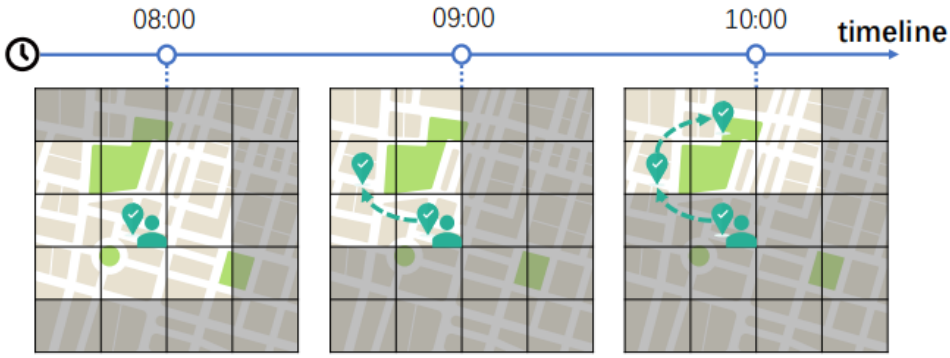

Figure 1. User activity trajectory.

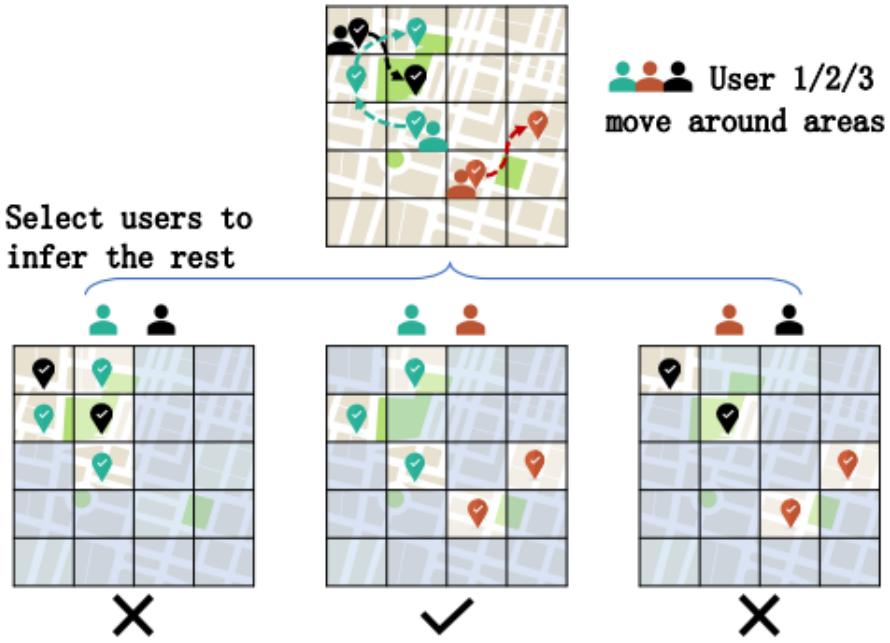

Figure 2. Selecting user data collection.

4. Our approach

In this section, first we propose the improved method STGCN-GRU algorithm for user trajectory prediction, followed by applying the predicted user trajectories to the user selection model. Next,

ADQN is proposed to solve the user selection problem in deep reinforcement learning. We first mathematically model the states, rewards and behaviors used in ADQN. Then, we train the model using historical data to guide the model can run effectively. The aim is to give a judgment on which users are the best choice to recruit in a given state. to reduce the perceived cost. The overall flow of the model is shown in Figure 3.

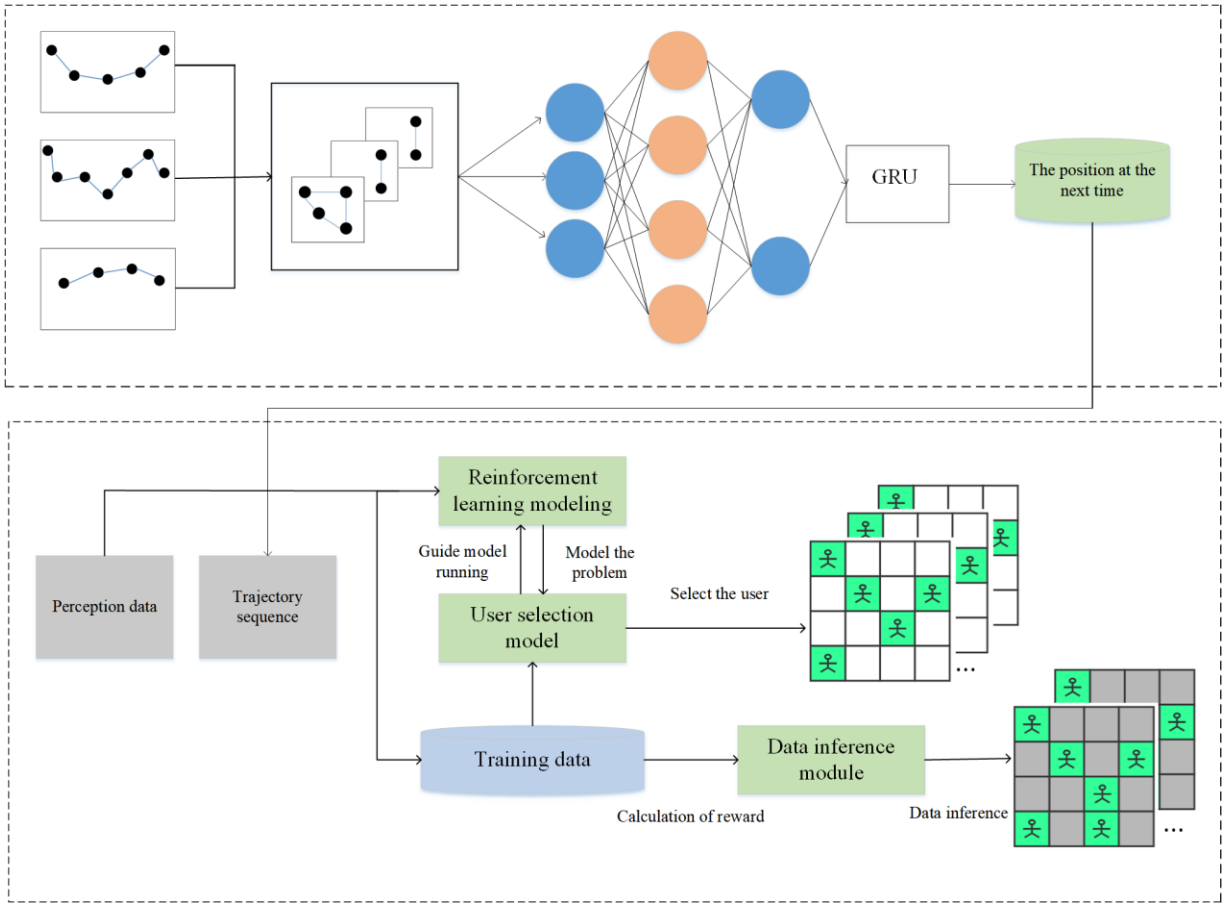

Figure 3. The process of user selection algorithm based on user trajectory prediction with sparse mobile crowd sensing.

4.1. User trajectory prediction

The process of user trajectory prediction is mainly divided into three parts: firstly, the original trajectory is converted into a trajectory sequence that can be accepted by the model; secondly, the spatio-temporal trajectory sequence is converted into a spatio-temporal trajectory map and feature extraction is performed by a graph convolutional network; thirdly, the extracted features are input into the already constructed model for training and prediction. The framework diagram of trajectory prediction by spatio-temporal graph convolutional network is shown in Figure 4.

Step 1: The accuracy of trajectory prediction of moving objects is influenced by the sampling rate. The continuous time information is converted into discrete time information by choosing a suitable time interval. In general, the time interval should be larger than the average sampling rate. Select representative points by calculating the linear centers of points with average longitude and latitude.

Transform all points in the trajectory by geographic feature mapping method.

Step 2: The spatio-temporal trajectories are transformed and a spatio-temporal trajectory map is created. The spatio-temporal semantic map is modeled by the spatio-temporal graph convolutional network (STGCN) algorithm for feature extraction.

Step 3: The trajectory sequence after depth feature extraction is input to GRU for GPS trajectory prediction.

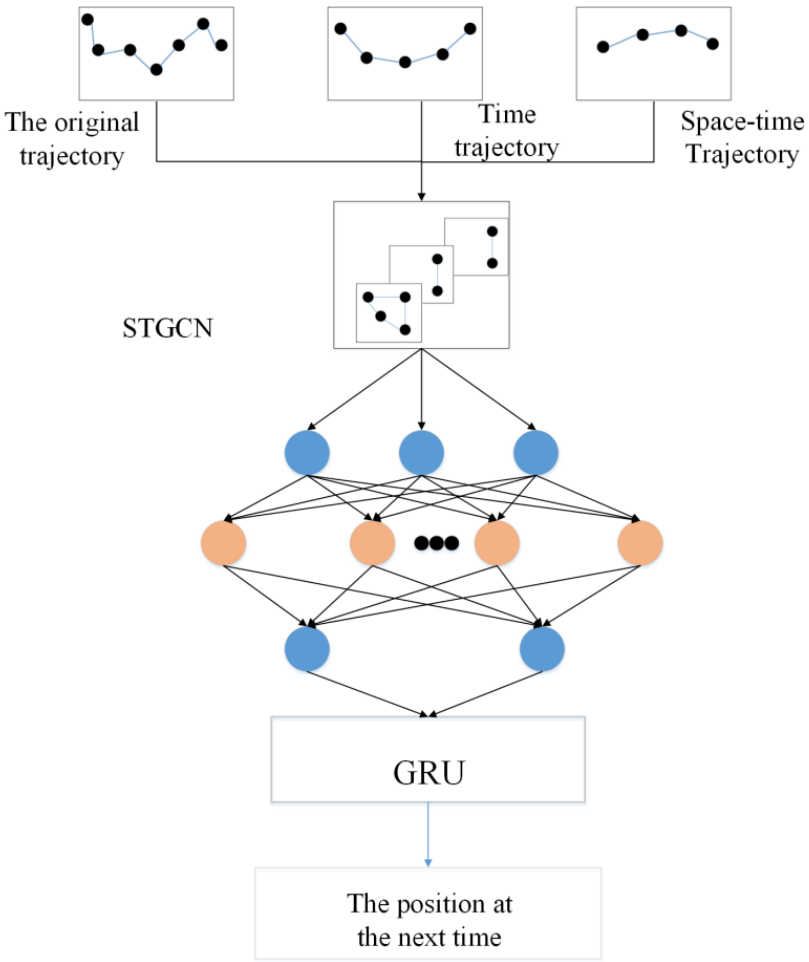

Figure 4. Framework diagram of spatio-temporal graph convolutional network for trajectory prediction.

The spatio-temporal graph network consists of two parts for graph feature extraction. The first part is the graph convolutional network (GCN) for extracting spatial features. The graph convolutional network extracts the neighborhood of each pedestrian in the graph by defining convolutional operators on the graph. The graph convolutional network is scalable and inter-row dependencies can be obtained by multiple iterations to obtain the global inter-row social dependencies. Another part is the temporal convolutional network (TCN) to extract temporal characteristics. The temporal dynamic behavior of the pedestrian flow is captured using a convolutional structure over the entire time axis.

Spatio-temporal graph convolutional network [33] (STGCN) is a classical graph spatio-temporal model and the pioneer of the spatio-temporal graph convolutional network framework. STGCN uses a

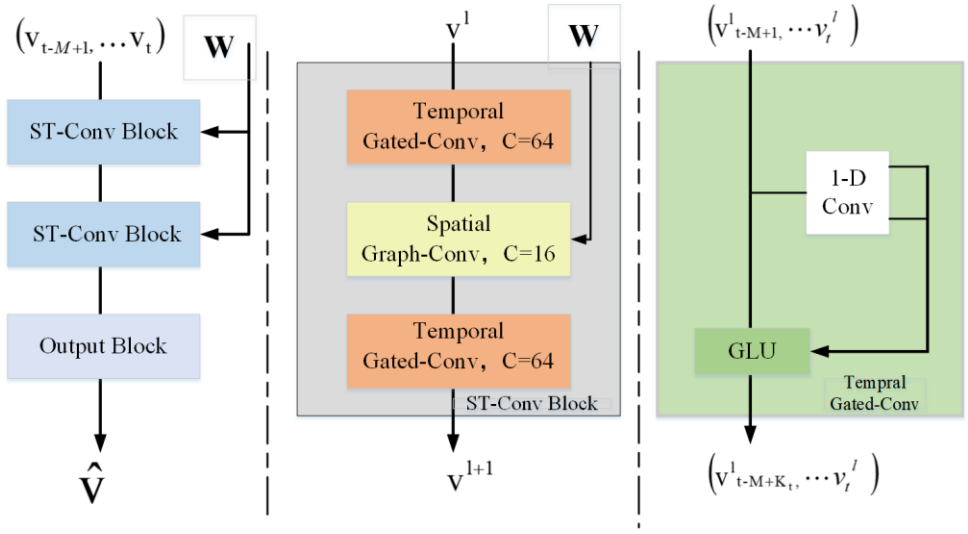

Figure 5. Architecture of spatio-temporal graph convolutional net-work.The network input is the feature vector $X \in R^{M\times n\times C_i}$ and the corresponding adjacenc matrix $W \in R^{n\times n}$, passing through two spatio-time convolution blocks and an output layer, and output $\hat{v} \in R^n$ to predict a certain time step feature after the t th time step.

one-dimensional temporal convolution in the temporal dimension and a graph convolutional structure in the spatial dimension to build a model with fewer parameters to speed up the training of the model, and experiments also show that STGCN can effectively capture spatio-temporal correlation. as shown in Figure 5, the key feature of STGCN is the uniquely designed spatio-temporal convolution block (ST-Conv Block), which can extract higher-order features by superimposing ST-Conv Block to increase the model expression capability.

The graph convolutional network we use is one of the neural networks on graphs, which has been successfully applied in problems such as network analysis and text classification. However, unlike images, on a normal graph if the adjacency matrix is used to define the neighborhood, the number of nodes in the neighborhood of each node is not fixed. Graph convolution is based on the concept of spectral graph convolution. In this paper, when using graph convolution network for feature extraction of temporal and spatial features of spatio-temporal graph, the graph convolution network consists of temporal convolution blocks and spatial convolution blocks, and the details of each block are shown in Figure 6.

The convolution operation on the graph is defined as follows, extending Eq (8) to the input feature graph located in the spatial graph, with a vector for each node on the graph, will redefine the sampling function p and the weighting function w.

$$(P_{sts_t})_{out} = \sum_{h=1}^{k} \sum_{w=1}^{k} (p(P_{sts_t}, h, w))_{in} W(h, w) \tag{8}$$

where p is the sampling function that enumerates the neighboring points at the location, and h,w are the height and altitude, respectively. W is the weight function that provides the weight vector for computing the feature vector of the sampled input. The convolution on the image domain is achieved by a p-matrix grid.

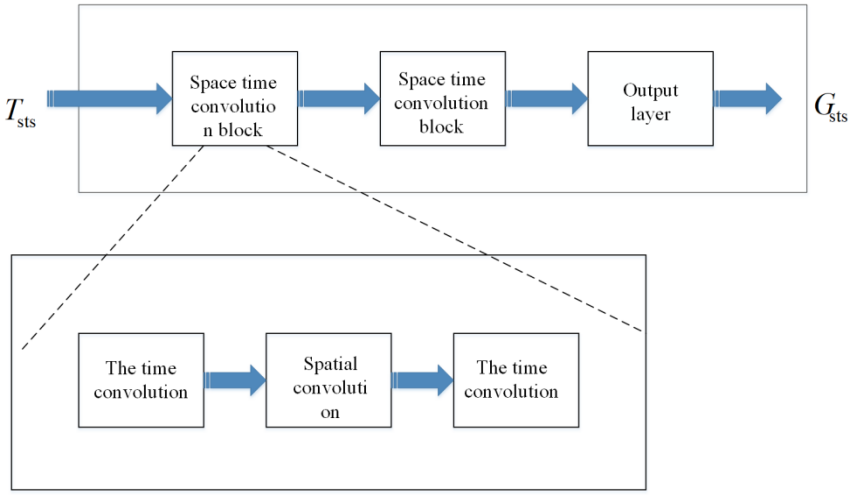

Figure 6. STGCN framework diagram.

On the image, the sampling function $p = (h, w)$ at the defined center of pixel adjacency P_{sts_t} . In the figure, the sampling function $B(P_{sts_{ti}}) = \{P_{sts_{tj}} | d(P_{sts_{tj}}, P_{sts_{ti}} < D)\}$ on the set of adjacencies where, $d(P_{sts_{tj}}, P_{sts_{ti}})$ is the minimum distance from $P_{sts_{tj}}$ to $P_{sts_{ti}}$, so the sampling function p is as in Eq (9).

$$p(p_{sts_{tj}}, p_{sts_{ti}}) = P_{sts_{tj}} \tag{9}$$

In two-dimensional convolution, there exists some fixed grid around the center position, so the neighboring pixels have a fixed spatial order. The weight function is implemented by indexing the multidimensional tensor according to the spatial order. By dividing the set $C(P_{sts_{ti}})$ of neighbors of a joint node $p_{sts_{ti}}$ into a fixed size set K , where each subset has a label. A mapping L is generated with a weight function as shown in Eq (10).

$$w((p_{sts_{tj}}, p_{sts_{ti}})) = W(L_{ti}(p_{sts_{tj}})) \tag{10}$$

Using the improved sampling and weighting functions, the graph convolution operation is shown in Eq (11).

$$(P_{sts_t})_{out} = \sum\nolimits_{P_{sts_{tj}} \in C(P_{sts_{ti}})} 1/Z_{ti}(P_{sts_{tj}})^{(p(P_{sts_{ti}}, P_{sts_{tj}}))_{in} W(P_{sts_{ti}}, P_{sts_{tj}})} \tag{11}$$

where $Z_{ti(P_{sts_{tj}})} = \{P_{sts_{tk}} | L_{ti}(P_{sts_{tk}}) = L_{ti}(P_{sts_{tj}})\}$ is equal to the base number of the corresponding subset. Adding this term in order to balance the contributions of the outputs of different subsets, the Eq (12) is obtained from Eqs (9)–(11).

$$(G_{sts_t})_{out} = \sum\nolimits_{P_{sts_{tj}} \in C(P_{sts_{ti}})} 1/Z_{ti}(P_{sts_{tj}})^{(P_{sts_{tj}})_{in} W(L_{ti}(P_{sts_{tj}}))} \tag{12}$$

In the construction of the graph, it is made by connecting the same moving object position points in the temporal aspect and connecting the neighboring position points in the spatial aspect.as the following Eq (13)

$$C(P_{sts_{ti}}) = \{P_{sts_{qj}} | d(P_{sts_{tj}}, P_{sts_{ti}}) \leq K, |q - t| \leq \frac{\Gamma}{2}\} \tag{13}$$

where the parameter Γ gives the time range in which the location points contained on each graph are located, so it can be considered as the size of the convolution kernel in time, when performing convolution operations for graph convolution networks, since the time axis is ordered.

4.2. User selection model

User selection refers to the selection of suitable perceptual users to perform the perceptual task under certain constraints. The user selection framework is shown in Figure 7, and we use a reinforcement learning model here. In reinforcement learning it is necessary to define the intelligent body, environment, and decision. In the second part we combine the user's trajectory and the characteristics of the data inference algorithm to first model the state, action, and reward, and use the user selection as an intelligent body for reinforcement learning. In the intelligences, performing an action leads to the next state and a reward. The value function is learned by state, action, and reward, and an action is preferentially selected if it will reap a larger Q value by executing it.

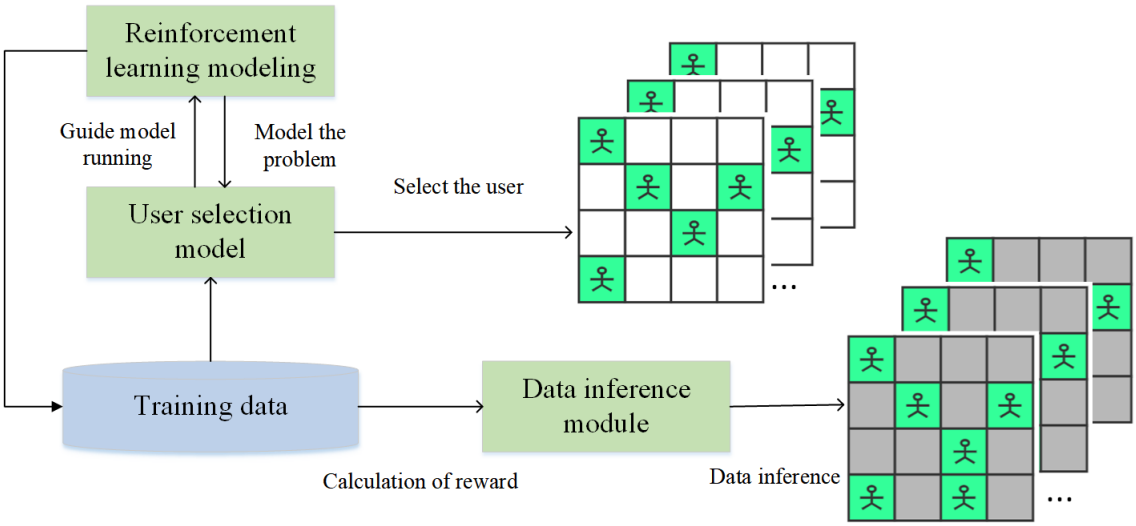

Figure 7. User selection framework.

4.2.1. Modeling the environment

(1) State: The state represents the current data collection condition, and the intelligence receives the information provided by the state and analyzes how good or bad each action is in the current state. The states contain the current and historical cycle selection situations, which influence the next user selection strategy. When collecting data, we consider the coverage of each perceptual cycle by the current selection situation. Data inference errors are reduced. The date is also a crucial factor that helps reinforcement learning to learn the temporal patterns in the data.

(2) Action: represents all possible cases of users that can be recruited in the current state. We set the action here to pick one user at a time instead of a group of users, because picking a group of users at a time would result in too much space for actions. Assuming that 3 out of 100 users are selected, there are C_{100}^{3} possibilities. In each selection, we select users one by one until the maximum number of recruits is reached. Because of the consideration of the order in which users are selected, we reorder the state for each user selected in the same recruitment cycle.

(3) Reward: indicates the score of the model feedback after selecting a user. In reinforcement learning, we set the initial state to the final state as a round, as in Eq (14), and when the round does not end, the intelligence obtains a reward with a fixed constant, the size of which is adjusted according to the size of the final accuracy during the actual training. When the round ends, we use the error of data inference to obtain the reward. error is set to [0,100], when the accuracy of data inference is higher, the smaller the error is, the larger the reward obtained by the intelligence.

$$reward = \begin{cases} 100 - Error[E, \overline{E},] & end\ of\ Recuitmentcycle \\ \text{c}, & else \end{cases} \tag{14}$$

4.2.2. ADQN

In traditional reinforcement learning, a common strategy for obtaining q-functions is the table learning approach (q-learning). In this approach, Q[S,A] represents the reward score of a particular action in a particular state. q function's goal is equivalent to filling all elements of the q table. But this traditional reinforcement learning algorithm cannot handle too large state spaces. Wang et al. [5] used DQN combined with neural networks, which is a reinforcement learning algorithm based on value function approximation, and this algorithm can be well suited for reinforcement learning models with too large state spaces. However, the traditional DQN usually overestimates the size of the Q-value, so we improve its objective function and propose ADQN. making the calculated Q-value must be less than or equal to the original Q-value, which reduces the overestimation problem to some extent and makes the Q-value closer to the true value.

The state-action value function is calculated as follows Eq (15), where R is the reward, which represents the reward that can be obtained by performing action a in state s. is the discount factor for calculating the cumulative reward, closer to 1 means that we are more farsighted, i.e., we place more importance on the possible rewards that can be obtained for future actions. In particular, the value of Q is then only the reward that can be obtained by performing action a, when the model focuses only on the immediate benefits. The Q-value calculated by the state-action value function represents our expectation of the cumulative reward that can be obtained by taking action a in state s.

$$Q[s, a] = E[\sum\nolimits_{k=0}^{\infty} r^k R_{t+k+1} | s = S_t, a = A_t] \tag{15}$$

From Eq (15) we can obtain the Bellman equation for the state-action value function, which shows that the Q value of each action is determined not only by the reward that can be obtained by the current action but also by the reward that can be obtained by subsequent actions. And its mathematical expression is shown in Eq (16)

$$Q[s, a] = E[R_{t+1} + \gamma Q[S_{t+1}, A_{t+1}] | s = S_t, a = A_t] \tag{16}$$

Two neural networks Q, $Q^{'}$ are used in ADQN. Q is used to calculate the Q value of the action, and the action with the largest Q value is selected, and $Q^{'}$ is used to calculate the $Q^{'}$ value of the selected action.and its mathematical expression is shown in Eq (17).

$$Q[s, a] = E[R_{t+1} + Q^{'}(s_{t+1}, \arg\max_{a} Q(S_{t+1}, A_{t+1})) | s = S_t, a = A_t] \tag{17}$$

Since the input state consists of multiple factors, the fully connected network can handle the heterogeneous input and capture the integrated correlation in our state, so the network with two fully connected layers is chosen as the hidden layer in this paper. The input of the neural network in ADQN is the current state, and the number of state nodes in the actual experiment is related to the maximum number of recruited users, when the maximum number of recruited users is larger the input layer needs. The number of nodes in the input layer is related to the maximum number of recruited users. The number of nodes in the hidden layer is usually set to twice the number of nodes in the input layer, and the activation function relu is used between the second hidden layer and the output layer, while the activation function tanh is used between the remaining layers. Each node in the output layer corresponds to the Q value of each action, so the number of actions determines the number of nodes in the output layer, which is equal to the number of users in the reinforcement learning model. We update the parameters of the neural network using a stochastic gradient descent method with a loss function,as shown in Eq (18).

$$L(\theta_t) = E_{[S_t,A_t,R_t,S_{t+1}]}[(R + \gamma max_{A_{t+1}} Q_{\theta_t}(S_{t+1}, A_{t+1}) - Q_{\theta_t}(S_t, A_t))^2] \tag{18}$$

Therefore, Eq (19) is obtained.

$$\nabla_{\theta_t} L(\theta_t) = E_{[S_t,A_t,R_t,S_{t+1}]}[(R + \gamma max_{A_{t+1}} Q_{\theta_t}(S_{t+1}, A_{t+1}) - Q_{\theta_t}(S_t, A_t))\nabla_{\theta_t} Q_{\theta_t}(S_t, A_t)] \tag{19}$$

5. Experiment

5.1. User trajectory prediction experiment

5.1.1. Data set and evaluation index

In this subsection, we present the experimental results of the proposed model and compare the prediction accuracy with some baseline, mainstream methods. In this paper, we use two mainstream datasets in the field of trajectory prediction, ETH and UCY, for training and validation of the proposed model. These two datasets contain real-world human trajectories and interaction information between pedestrians in various traffic scenarios. Among them, the two datasets mentioned above have five sub-datasets, namely ETH, HOTEL, UNIV, ZARA1 and ZARA2.

The format of each data set is the spatial coordinates of each pedestrian. By processing the data, we can obtain the relative trajectory coordinates of each pedestrian over time. A total of 1536 pedestrian data are available in both datasets and contain a variety of challenging interaction scenarios, such as pedestrians avoiding collisions, standing, pedestrians walking in groups, pedestrians behind chasing pedestrians in front, and so on. The number of pedestrians in a single scene in each traffic environment ranges from 0 to 51. The CPU used in this experiment is an Intel(R) Core(TM) i9-9820X CPU @ 3.30 GHz and the GPU is an NVDIA GeForce RTX 2080 Ti.

Two metrics can be used to evaluate the performance of the model: the average displacement error (ADE) defined in Eq (20) and the final displacement error (FDE) defined in Eq (21). Intuitively, the ADE measures the average prediction performance along the trajectory, while the FDE considers only the prediction accuracy at the endpoints. Since the Social-STGCNN generates a bivariate Gaussian distribution as a prediction, in order to compare the distribution with some target value, we followed the evaluation method used in the Social-LSTM, where 20 samples were generated based on the predicted

distribution. Then, the samples closest to the ground truth were used to calculate ADE and FDE. this evaluation method has been widely adopted.

$$ADE = \frac{\sum_{n \in N} \sum_{t \in T_p} \|\tilde{p}_t^n - p_t^n\|_2}{N \times T_p} \tag{20}$$

$$FDE = \frac{\sum_{n \in N} \|\tilde{p}_t^n - p_t^n\|_2}{N}, t = T_p \tag{21}$$

The model configuration and training setup consisted of a series of ST-GCN layers followed by GRU layers. We used relu as the activation function σ throughout the model. we set a training batch size of 128 and trained the model for 250 epochs using stochastic gradient descent (SGD). The initial learning rate is 0.01, which becomes 0.002 after 150 epochs. in addition, it is worth noting that the performance of the model decreases when the number of STGCN layers increases.

To evaluate the performance effectiveness of STGCN-GRU, STGCN-GRU is compared with Social-LSTM, Social-GAN, Social-STGCN and STGAT in terms of both performance and accuracy, respectively. Specifically, the performance is compared by the two aspects of average displacement error and final displacement error. Several baseline algorithms are presented as follows.

1) Social-LSTM: The method proposed by Alahi et al. [39]. Each person is modeled by an LSTM where hidden states are pooled at each time step using a social pooling layer.

2) Social-GAN:Generative adversarial networks are used to encourage the generation of multiple possible trajectories with randomness, and then the one with the least loss is selected. In contrast to Social-LSTM, the social relationships between all pedestrians are modeled at once, not only between neighboring pedestrians.

3) Social-STGCNN: Social spatio-temporal graph convolutional neural network (Social-STGCNN), which replaces the need for aggregation methods by modeling interactions as a graph.

4) STGAT: A spatio-temporal graph attention network (STGAT), which predicts the future trajectory of pedestrians based on a sequence-to-sequence architecture.

5) STGCN-GRU: This method is the proposed method in this paper, which combines GRU and STGCN algorithm. First, the difficult spatial data GPS trajectory is transformed into an easy-to-understand trajectory map, and second, the STGCN algorithm is used to extract the features related to temporal and spatial information from the trajectory map, and the feature sequence is input to GRU for GPS trajectory prediction.

5.1.2. Experimental results

Table 1 shows the prediction results of different prediction methods in terms of average displacement error, final displacement error, and two aspects as indicators for evaluating the model.

Table 1. Comparison of ADE / FDE metrics.

	ETH	HOTEL	UNIV	ZARA1	ZARA2	AVG
Social-LSTM	1.09/2.35	0.79/1.76	0.67/1.4	0.47/1.0	0.56/1.17	0.72/1.54
Social-GAN	1.13/2.21	1.01/2.18	0.60/1.28	0.42/0.91	0.52/1.11	0.74/1.54
STGAT	0.88/1.66	0.56/1.15	0.52/1.13	0.41/0.91	0.31/0.68	0.54/1.11
Social-STGCNN	0.64/1.11	0.49/0.85	0.44/0.79	0.34/0.53	0.30/0.48	0.44/0.75
STGCN-GRU	0.66/1.09	0.33/0.65	0.51/1.01	0.34/0.49	0.27/0.45	0.42/0.73

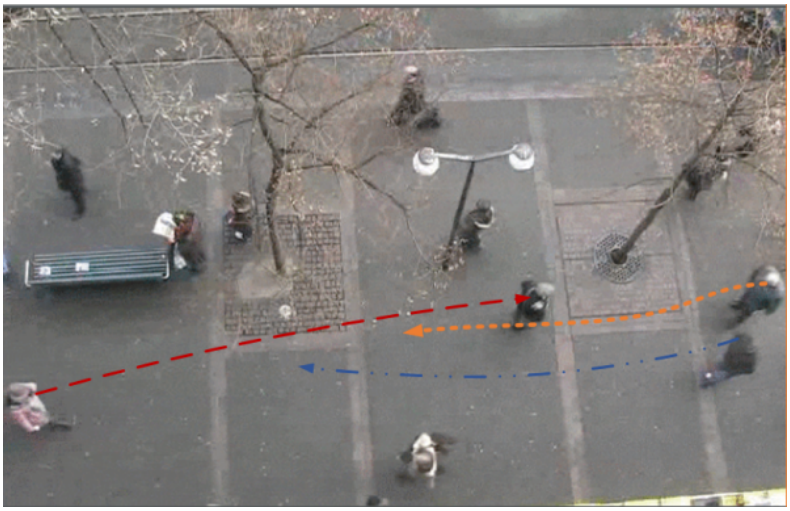

Figure 8. User trajectory prediction scenario 1.

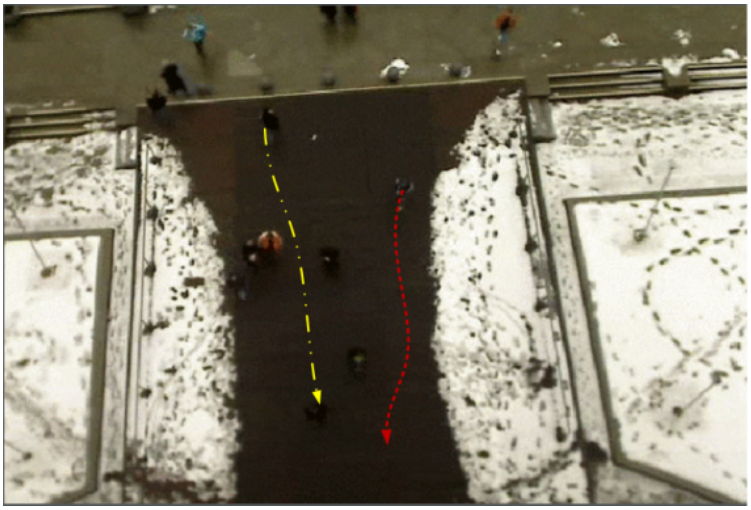

Figure 9. User trajectory prediction scenario 2.

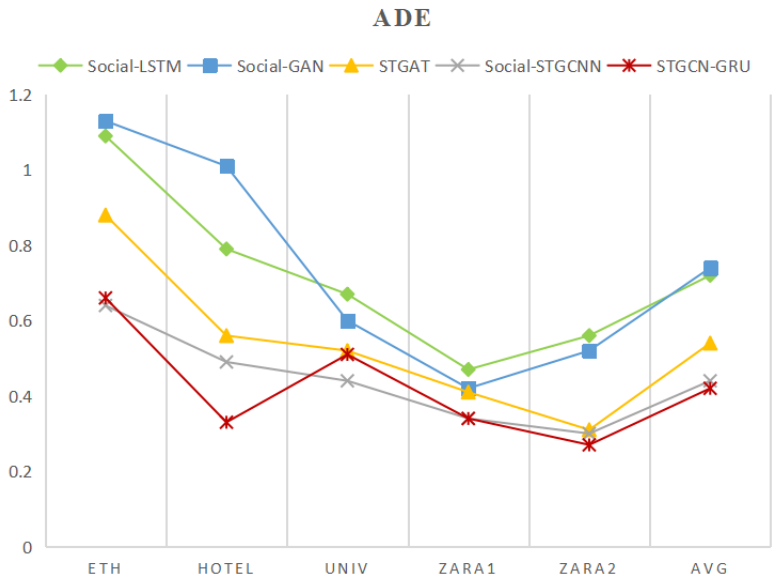

Figure 10. Comparison of ADE with different data sets.

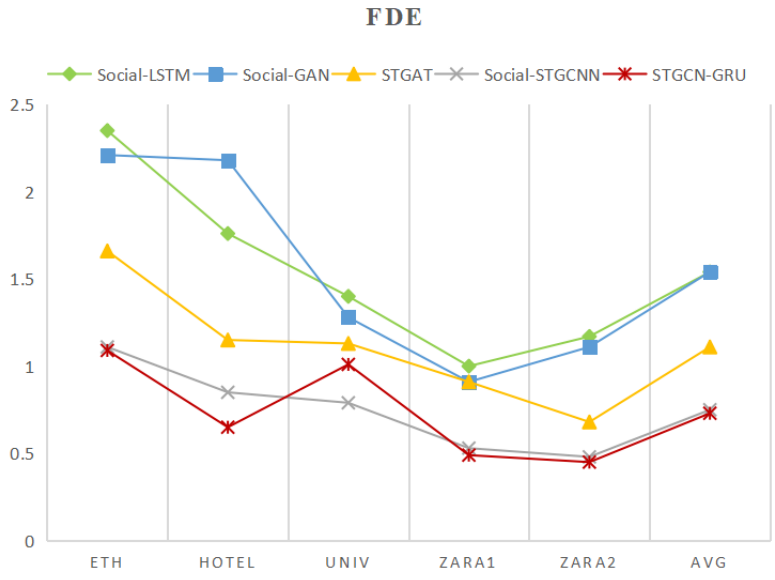

Figure 11. Comparison of FDE under different data sets.

5.2. User recruitment strategy experiment

5.2.1. Data set and evaluation index

In this paper, we use Beijing air quality monitoring data and the Geolife dataset to verify the validity of the user selection problem.

Perception dataset: In this paper, we used four datasets of AQI, PM2.5, CO and NO2 from 34 sites in Beijing for a large number of experiments. We viewed these 34 sites as 34 perception areas, and

divided 34 circular areas with a radius of 1000 m for the perception task with these sites as the center of the circle. Each user may not pass through the sensing area every cycle, but the user may move frequently in a certain location, so data are collected repeatedly for different periods at the same site.

User dataset: We use the "Geolife" dataset, where GPS trajectories are represented by a series of time-stamped points, each containing latitude, longitude and altitude information. In the "Geolife" dataset, we eliminate the user trajectories that infrequently pass through the sensing area we set, and finally generate 113 user trajectories within the sensing area. Each trajectory represents the travel of a user for a whole day (24 hours). In this experiment, we assume that users can successfully perceive the data when they pass through the perception area.

In this paper, the reinforcement learning model is trained using the perceptual data of May 2020, where the real perceptual data of 34 sites are known at the time of training, i.e., from 2020-5-1 00:00 to 2020-5-30 23:00, for 30 days. The perceptual data for June 2020, from 2020-6-1 00:00 to 2020-6-30 23:00, for 30 days, was used for testing. In the experiment, the recruitment period R' was set to 1 day and the task period R was set to 30 days, each period being 1 hour. We evaluated the experimental results by calculating MAPE on the data of all missing fractions, see Eq (22), where n is the number of missing subregions.

$$MAPE = \sum_{i=1}^{m} \sum_{j=1}^{t} |\frac{E[i, j] - \overline{E}[i, j]}{E[i, j]}| \times \frac{100}{n} \tag{22}$$

Several of the baseline algorithms used in this paper are described as follows.

1) ALL-RANDOM: Users are recruited randomly in each recruitment cycle, and then the data collected from the users are used to make inferences about the complete perceptual data. Randomly selected users are used to infer the complete perceptual map using their perceptual data covering sub-regions. Random can also provide more information for data inference when a larger number of users are allowed to be selected.

2) PSO: Meta-heuristic algorithms are commonly used to solve combinatorial optimization problems, among which the particle swarm optimization algorithm is considered as a meta-heuristic algorithm with wide applicability. We will use the particle swarm algorithm to solve this problem. A particle in the particle swarm algorithm is a solution.

3) DQN: A neural network with two fully connected layers is used to estimate the Q value and initialize the network with random weights. And empirical replay mechanism, fixed Q-network and -greedy algorithm are used to process.

4) ADQN: The difference with DQN is that the target function in DQN is modified. ADQN is selected each time according to the parameters of the current Q network, and not according to the parameters of target-Q as in DQN, so when calculating the target value is a little smaller than the original. This will solve the problem of overfitting and overestimation in DQN, and make the Q value closer to the real value.

5.2.2. Performance comparison of user-selected algorithms

We considered four factors in the reinforcement learning state, which are the coverage of the cycle, the historical recruitment cycle selection, the current selection, and the date. Figure 13 shows a comparison of the five states, where we remove one factor at a time and keep the other three to train the model. After each full cycle (30 days), the model was tested using the model and recorded. As can be seen from the graph, the selection of historical recruitment cycles helps the model more

significantly, and when the selection of historical recruitment cycles is not included in the states, the test results of the model fluctuate dramatically within a range. When all factors are taken into account, the model can be trained to be more stable and make better decisions.

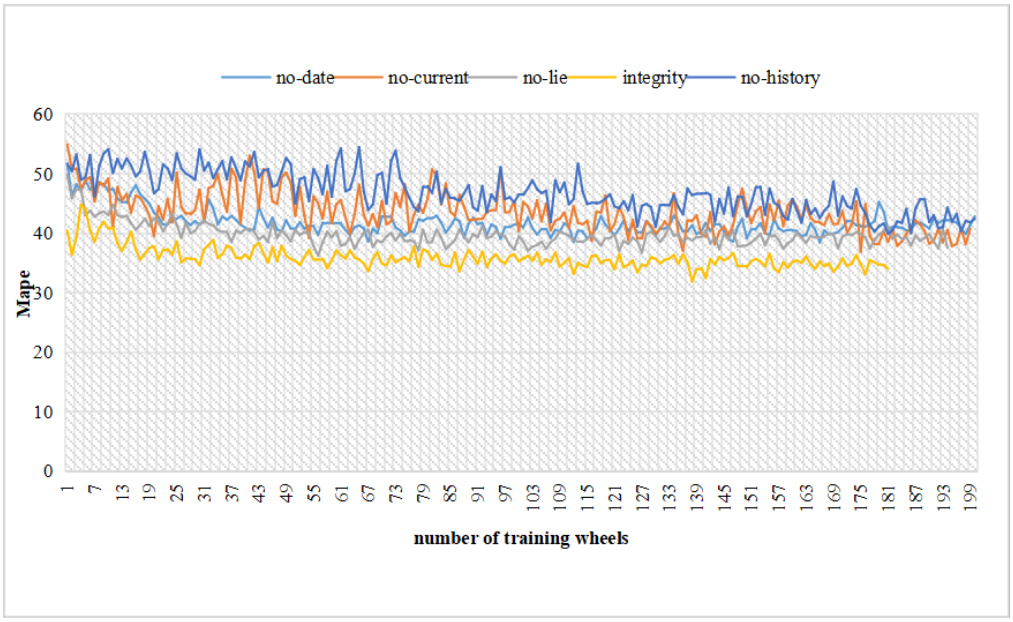

Figure 12. Comparison of mape in different states.

We verified the realism of our proposed algorithm on four real datasets, AQI, PM2.5, NO2 and CO, and selected 3, 5, 7 and 10 users from the alternative users to perform the perception task in each recruitment cycle, respectively, and the experimental results are shown in Table 2, as can be seen from the figure. Our proposed user recruitment strategy, ADQN, always outperforms the other three methods.

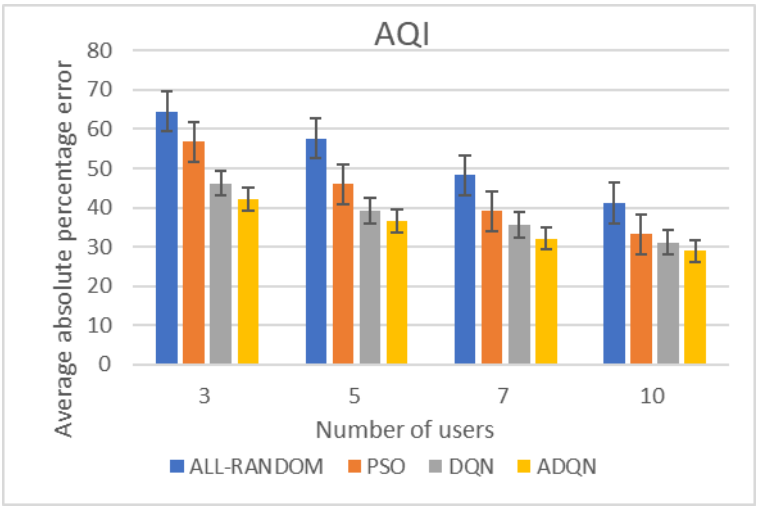

Figure 13. Comparison of mape values for different number of users in AQI dataset.

Table 2. MAPE for different number of users in the AQI dataset.

	ALL-RANDOM	PSO	DQN	ADQN
3	64.52	56.73	46.23	42.14
5	57.64	45.92	39.13	36.57
7	48.23	39.15	35.62	32.13
10	41.16	33.25	31.14	28.92

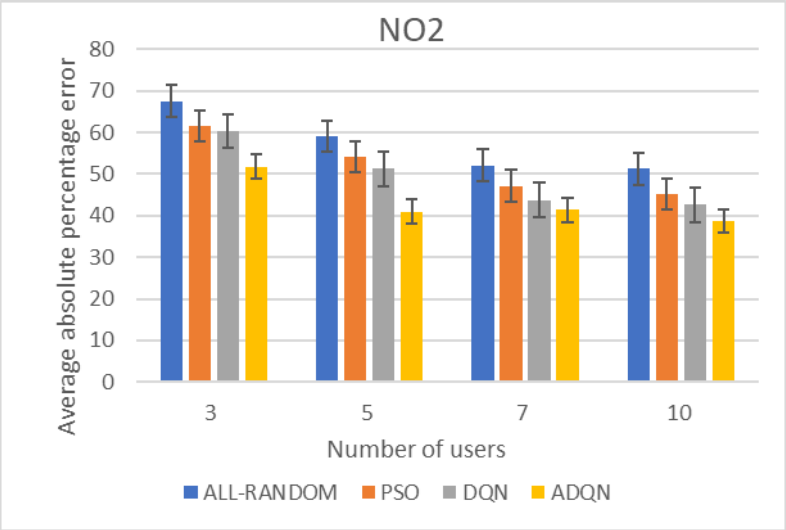

Figure 14. Comparison of mape values for different number of users in NO2 dataset.

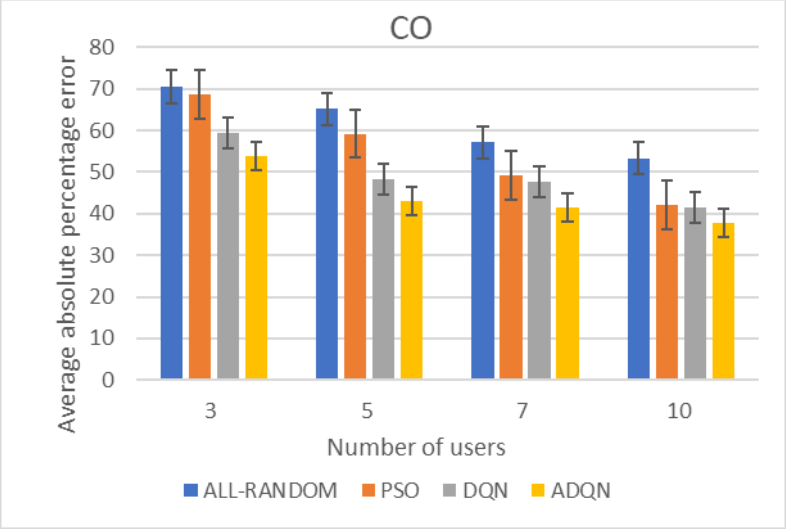

Figure 15. Comparison of mape values for different number of users in CO dataset.

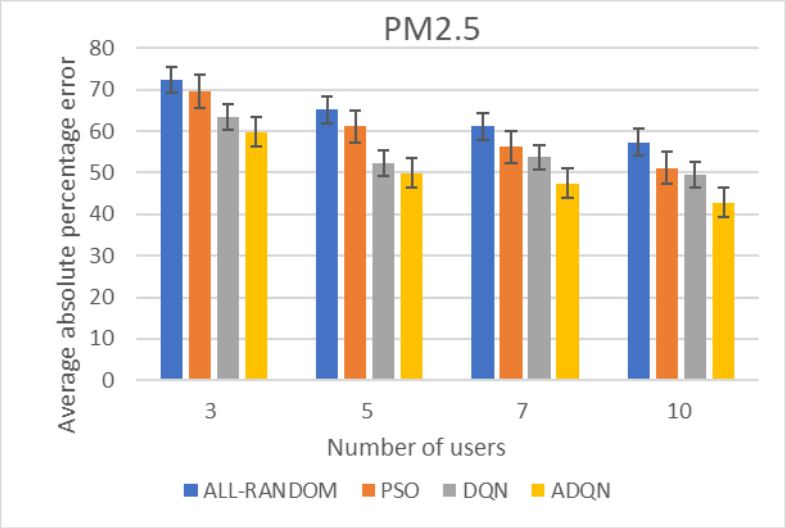

Figure 16. Comparison of mape values for different number of users in PM2.5 dataset.

Specifically, for the AQI dataset, ADQN can reduce the inference error by 34.69% / 36.55% / 33.38% / 29.73% (the number of users recruited per recruitment cycle is 3/ 5/ 7/ 10) compared to ALL-RANDOM. Compared with PSO and DQN, ADQN reduces the inference error by 25.72%/ 20.36% /17.93% /13.02% and 8.84% / 6.54%/ 9.79%/ 7.13%, respectively. Compared with DQN algorithm, ADQN algorithm makes improvements in the objective function. This reduces the problem of overestimation in Q network and improves the accuracy of user recruitment. In fact, smaller inference errors can be achieved when 10 users are recruited to perceive each recruitment cycle.

The runtimes of each method for recruiting 10 users per recruitment cycle at the time of testing are presented in Table 3. Our experimental platform was equipped with an Intel(R) Core(TM) i9-9820X CPU @ 3.30 GHz and an NVDIA GeForce RTX 2080 Ti GPU. time comparisons were not included as PSO is only used for comparative analysis, when the algorithm iterations are completed to obtain the optimal user mix, which is not possible in a practical context. For compression-awareness, it takes about 12.268–12.731 s to extrapolate one month of data on the test time. in determining the users to be recruited for the whole recruitment cycle, ALL-RANDOM only needs to generate user numbers randomly, without any computational process resulting in it being the fastest. ADQN takes 0.167–0.186 s, which is the fastest. In addition, the ADQN model is implemented in TensorFlow and can be trained offline, which takes about 200 minutes for a neural network with two fully connected layers.

Table 3. Run time (10 users recruited per recruitment cycle).

	AQI	PM2.5	NO2	CO
CS	12.731s	12.581 s	12.268s	12.547s
ALL-RANDOM	0.000734s	0.000734s	0.000734s	0.000734s
DQN	1.032s	1.051s	1.143s	1.281s
ADQN	0.186s	0.184s	0.169s	0.167s

6. Conclusions

In recent years, along with the explosive popularity of portable smart devices and the rapid development of wireless communication technology, sparse mobile crowd sensing has emerged as an emerging paradigm of perception and computation. Sparse mobile crowd sensing can greatly reduce perception consumption and can handle incomplete or randomly collected perceptual data well, which is extremely promising in practical applications in many fields, such as environment, transportation, smart cities, social services, etc.

In this paper, we investigate two aspects of user trajectory prediction and sparse mobile crowd sensing user selection, which directly selects to a high contribution group of users for multiple consecutive cycles, which is a near optimal combination within the cost budget, provided that the user trajectory is known. First, we propose an STGCN-GRU user trajectory prediction algorithm, which uses the STGCN algorithm to extract temporal and spatial information related features from the trajectory graph. STGCN uses fewer parameters to speed up the training of the model and can effectively capture spatio-temporal correlation. The feature sequences are then input into GRU for trajectory prediction, and the algorithm improves the accuracy of user trajectory prediction. Second, an ADQN (action DQN) user recruitment algorithm is proposed. the ADQN algorithm improves the objective function in DQN on the idea of reinforcement learning. The action with the maximum input value is found from the Q network, and then the output value of the objective function of the corresponding action Q network is found. This reduces the overestimation problem that occurs in Q networks and improves the accuracy of user recruitment.

Most of the existing studies on user trajectory prediction are developed based on the spatio-temporal characteristics of trajectories, and often ignore other additional information, such as the state of moving objects, driving direction and speed. In the future, we will add some semantic information to extract the high-level semantic features of trajectories, and we can generate semantic trajectories by combining trajectory sequences with related semantics. Reinforcement learning can continuously improve the accuracy of the algorithm through ”trial and error”. Therefore, how to better combine supervised learning and reinforcement learning to mimic experts' decisions in the real world and achieve automatic model adjustment will be another research direction in the future.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant No.61902311, 51904226, 52274137, 62172330), in part by the Post doctoral Research Foundation of China (Grant No. 2019M663801), the Key R & D Foundation of Shaanxi Province (Grant No.2021SF- 479), and the Scientific Research Project of Shaanxi Provincial Education Department (No.22JK0459).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. F. Abbondati, S. A. Biancardo, R. Veropalumbo, G. Dell'Acqua, Surface monitoring of road pavements using mobile crowdsensing technology, *Measurement*, **171** (2021), 108763. https://doi.org/10.1016/j.measurement.2020.108763
2. Y. Huang, H. Chen, G. Ma, K. Lin, Z. Ni, N. Yan, et al., OPAT: Optimized allocation of time-dependent tasks for mobile crowdsensing, *IEEE Trans. Ind. Inf.*, **18** (2021), 2476–2485. https://doi.org/10.1109/TII.2021.3094527
3. T. Tang, L. Cui, Z. Yin, S. Hu, L. Fu, Spatiotemporal characteristic aware task allocation strategy using sparse user data in mobile crowdsensing, *Wireless Networks*, **29** (2023), 459–474. https://doi.org/10.1007/s11276-022-03138-y
4. S. He, K. G. Shin, Steering crowdsourced signal map construction via Bayesian compressive sensing, in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, (2018), 1016–1024. https://doi.org/10.1109/INFOCOM.2018.8485972
5. L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, A. M'hamed, Sparse mobile crowdsensing: challenges and opportunities, *IEEE Commun. Mag.*, **54** (2016), 161–167. https://doi.org/10.1109/MCOM.2016.7509395
6. W. Liu, Y. Yang, E. Wang, J. Wu, User recruitment for enhancing data inference accuracy in sparse mobile crowdsensing, *IEEE Internet Things J.*, **7** (2019), 1802–1814. https://doi.org/10.1109/JIOT.2019.2957399
7. C. Tu, Z. Yu, L. Han, W. Zhu, F. Huang, L. Wang, Direct participant recruitment strategy in sparse mobile crowd sensing, *Chin. J. Comput.*, **45** (2022), 1539–1556. https://doi.org/10.11897/SP.J.1016.2022.01539
8. L. Han, Z. Yu, L. Wang, Z. Yu, B. Guo, Keeping cell selection model up-to-date to adapt to time-dependent environment in sparse mobile crowdsensing, *IEEE Internet Things J.*, **8** (2021), 13914–13925. https://doi.org/10.1109/JIOT.2021.3068415
9. T. Liu, Y. Zhu, Y. Yang, F. Ye, ALC^2: When active learning meets compressive crowdsensing for urban air pollution monitoring, *IEEE Internet Things J.*, **6** (2019), 9427–9438. https://doi.org/10.1109/JIOT.2019.2939552
10. Y. Zhu, Z. Li, H. Zhu, M. Li, Q. Zhang, A compressive sensing approach to urban traffic estimation with probe vehicles, *IEEE Trans. Mobile Comput.*, **12** (2012), 2289–2302. https://doi.org/10.1109/TMC.2012.205
11. J. Wang, F. Wang, Y. Wang, D. Zhang, L. Wang, Z. Qiu, et al., Social-network-assisted worker recruitment in mobile crowd sensing, *IEEE Trans. Mobile Comput.*, **18** (2018), 1661–1673. https://doi.org/10.1109/TMC.2018.2865355
12. H. Li, T. Li, W. Wang, Y. Wang, Dynamic participant selection for large-scale mobile crowd sensing, *IEEE Trans. Mobile Comput.*, **18** (2018), 2842–2855. https://doi.org/10.1109/TMC.2018.2884945

13. H. Gao, C. Liu, J. Tang, D. Yang, P. Hui, W. Wang, Online quality-aware incentive mechanism for mobile crowd sensing with extra bonus, *IEEE Trans. Mobile Comput.*, **18** (2018), 2589–2603. https://doi.org/10.1109/TMC.2018.2877459

14. L. Pu, X. Chen, J. Xu, X. Fu, Crowd foraging: A QoS-oriented self-organized mobile crowdsourcing framework over opportunistic networks, *IEEE J. Sel. Areas Commun.*, **35** (2017), 848–862. https://doi.org/10.1109/JSAC.2017.2679598

15. J. Li, J. Wu, Y. Zhu, Selecting optimal mobile users for long-term environmental monitoring by crowdsourcing, in *Proceedings of the International Symposium on Quality of Service*, (2019), 1–10. https://doi.org/10.1145/3326285.3329043

16. Q. Yang, Y. Chen, M. Guizani, G. M. Lee, Spatiotemporal location differential privacy for sparse mobile crowdsensing, in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, (2021), 1734–1741. https://doi.org/10.1109/IWCMC51323.2021.9498951

17. J. Shen, J. Peng, L. Shao, Submodular trajectories for better motion segmentation in videos, *IEEE Trans. Image Process.*, **27** (2018), 2688–2700. https://doi.org/10.1109/TIP.2018.2795740

18. D. Yao, C. Zhang, Z. Zhu, J. Huang, J. Bi, Trajectory clustering via deep representation learning, in *2017 international joint conference on neural networks (IJCNN)*, (2017), 3880–3887. https://doi.org/10.1109/IJCNN.2017.7966345

19. J. Quehl, H. Hu, S. Wirges, M. Lauer, An approach to vehicle trajectory prediction using automatically generated traffic maps, in *2018 IEEE Intelligent Vehicles Symposium (IV)*, (2018), 544–549. https://doi.org/10.1109/IVS.2018.8500535

20. H. Xue, D. Q. Huynh, M. Reynolds, SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction, in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, (2018), 1186–1194. https://doi.org/10.1109/WACV.2018.00135

21. G. Xie, H. Gao, L. Qian, B. Huang, K. Li, J. Wang, Vehicle trajectory prediction by integrating physics-and maneuver-based approaches using interactive multiple models, *IEEE Trans. Ind. Electron.*, **65** (2017), 5999–6008. https://doi.org/10.1109/TIE.2017.2782236

22. T. Fernando, S. Denman, S. Sridharan, C. Fookes, Tracking by prediction: A deep generative model for mutli-person localisation and tracking, in *2018 IEEE Winter conference on applications of computer vision (WACV)*, (2018), 1122–1132. https://doi.org/10.1109/WACV.2018.00128

23. H. Woo, Y. Ji, H. Kono, Y. Tamura, Y. Kuroda, T. Sugano, et al., Lane-change detection based on vehicle-trajectory prediction, *IEEE Rob. Autom. Lett.*, **2** (2017), 1109–1116. https://doi.org/10.1109/LRA.2017.2660543

24. N. Deo, A. Rangesh, M. M. Trivedi, How would surround vehicles move? A unified framework for maneuver classification and motion prediction, *IEEE Trans. Intell. Veh.*, **3** (2018), 129–140. https://doi.org/10.1109/TIV.2018.2804159

25. B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, preprint, arXiv:1709.04875. https://doi.org/10.48550/arXiv.1709.04875

26. S. Guo, Y. Lin, N. Feng, C. Song, H. Wan, Attention based spatial-temporal graph convolutional networks for traffic flow forecasting, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **33** (2019), 922–929. https://doi.org/10.1609/aaai.v33i01.3301922

27. C. Song, Y. Lin, S. Guo, H. Wan, Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **34** (2020), 914–921. https://doi.org/10.1609/aaai.v34i01.5438

28. X. Luo, X. Chen, L. Shou, K. Chen, Y. Wu, Semantic trajectory extraction framework for indoor space, *Tsinghua Sci. Technol.*, **59** (2019), 186–193. https://doi.org/10.16511/j.cnki.qhdxxb.2018.26.047

29. L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, et al., Model-based reinforcement learning for atari, preprint, arXiv:1903.00374. https://doi.org/10.48550/arXiv.1903.00374

30. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, et al., Playing atari with deep reinforcement learning, preprint, arXiv:1312.5602. https://doi.org/10.48550/arXiv.1312.5602

31. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, et al., Mastering the game of go with deep neural networks and tree search, *Nature*, **529** (2016), 484–489. https://doi.org/10.1038/nature16961

32. M. Hausknecht, P. Stone, Deep recurrent q-learning for partially observable mdps, in *2015 AAAI Fall Symposium Series*, **2015** (2015). https://doi.org/10.48550/arXiv.1507.06527

33. G. Lample, D. S. Chaplot, Playing fps games with deep reinforcement learning, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **31** (2017). https://doi.org/10.1609/aaai.v31i1.10827

34. H. Li, T. Li, F. Li, S. Yang, Y. Wang, Multi-expertise aware participant selection in mobile crowd sensing via online learning, in *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, (2018), 433–441. https://doi.org/10.1109/MASS.2018.00067

35. X. Zhang, X. Gong, Online data quality learning for quality-aware crowdsensing, in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, (2019), 1–9. https://doi.org/10.1109/SAHCN.2019.8824861

36. H. Zhao, M. Xiao, J. Wu, Y. Xu, H. Huang, S. Zhang, Differentially private unknown worker recruitment for mobile crowdsensing using multi-armed bandits, *IEEE Trans. Mobile Comput.*, **20** (2021), 2779–2794. https://doi.org/10.1109/TMC.2020.2990221

37. G. Gao, J. Wu, M. Xiao G. Chen, Combinatorial multi-armed bandit based unknown worker recruitment in heterogeneous crowdsensing, in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, (2020), 179–188. https://doi.org/10.1109/INFOCOM41043.2020.9155518

38. L. Xiao, T. Chen, C. Xie, H. Dai H. V. Poor, Mobile crowdsensing games in vehicular networks, *IEEE Trans. Veh. Technol.*, **67** (2018), 1535–1545. https://doi.org/10.1109/TVT.2016.2647624

39. A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, F. F. Li, S. Savarese, Social lstm: Human trajectory prediction in crowded spaces, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2016), 961–971. https://doi.org/10.1109/CVPR.2016.110

AIMS Press

© 2023 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)