SeqST-ResNet: A Sequential Spatial Temporal ResNet for Task Prediction in Spatial Crowdsourcing

Dongjun Zhai[1,2,4], An Liu[1](✉), Shicheng Chen[3], Zhixu Li[1], and Xiangliang Zhang[2]

[1] Soochow University, Suzhou, China
{anliu,zhixuli}@suda.edu.cn
[2] King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
{Dongjun.Zhai,Xiangliang.Zhang}@kaust.edu.sa
[3] National University of Singapore, Singapore, Singapore
coder.chen.shi.cheng@gmail.com
[4] Blockshine Technology Corp., Shanghai, China

Abstract. Task appearance prediction has great potential to improve task assignment in spatial crowdsourcing platforms. The main challenge of this prediction problem is to model the spatial dependency among neighboring regions and the temporal dependency at different time scales (e.g., hourly, daily, and weekly). A recent model ST-ResNet predicts traffic flow by capturing the spatial and temporal dependencies in historical data. However, the data fragments are **concatenated** as one tensor fed to the deep neural networks, rather than learning the temporal dependencies in a sequential manner. We propose a novel deep learning model, called SeqST-ResNet, which well captures the temporal dependencies of historical task appearance in **sequences** at several time scales. We validate the effectiveness of our model via experiments on a real-world dataset. The experimental results show that our SeqST-ResNet model significantly outperforms ST-ResNet when predicting tasks at hourly intervals and also during weekday and weekends, more importantly, in regions with intensive task requests.

Keywords: Task prediction · Spatial crowdsourcing · Deep neural network

1 Introduction

Spatial Crowdsourcing (SC) [20] has attracted lots of attentions in recent years. A typical SC system consists of a platform that is responsible for releasing location-based tasks and a crowd of workers that can physically move to specified locations to perform tasks. Emerging SC applications include taxi-hailing service such as Didi and Uber, meal order and delivery service such as Eleme, and dynamic information collection service such as Gigwalk. In practice, hundreds

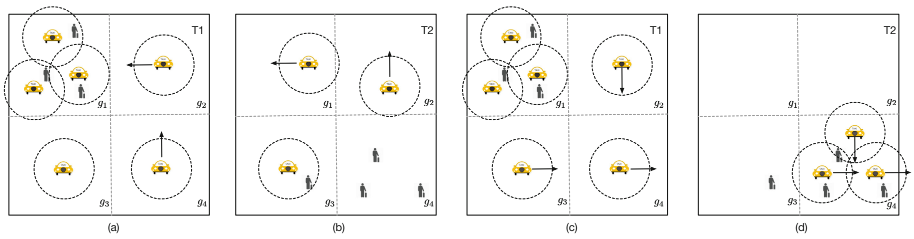

Fig. 1. Task assignment with/without task prediction in spatial crowdsourcing. Without task prediction, idle workers at T_1 in (a) make aimless movement, then at T_2 in (b) only one task-worker pair can be matched. With the guidance of task prediction, idle workers at T_1 in (c) move to g_4 which is predicted to have several task requests. Then at time T_2 in (d), three task-worker pairs are matched.

or thousands of workers and tasks could be online simultaneously, so an important problem in SC is to assign tasks to workers with the aim to maximize the number of total assigned task-worker pairs. In previous studies the online task assignment problem is usually decomposed into a series of offline task assignment problems, each of which is further reduced to be a maximum matching problem in a weighted bipartite graph [13,22]. While this reduction can generate feasible solutions, it does not consider the dynamic feature of workers and tasks. After one round of offline task assignment, idle workers are assumed to stay in place or just move around aimlessly, waiting for the next round of offline task assignment with new coming tasks. In fact, these idle workers can move towards some places or areas where new tasks are likely to arrive, so that more task-worker pairs can be assigned. This motivation can be further illustrated by the following example.

Figure 1 demonstrates a real time taxi-hailing service where tasks are passengers and workers are taxis. To facilitate later discussion, the whole area is divided into 4 grids and each taxi has a service range shown by dotted circles. At current time T_1, the platform can generate three task-worker pairs as three passengers are just in the service range of three taxis in g_1, as shown in Fig. 1(a). For taxis in other grids, if they stay at place (e.g., the taxi in g_3) or move around aimlessly (e.g., the taxis in g_2 and g_4), at next time T_2 shown in Fig. 1(b), only one task-worker pair can be made as most new tasks are appeared in g_4 but no workers are there. However, if the platform can guide idle workers to move to grid g_4 as shown in Fig. 1(c), then at time T_2 in Fig. 1(d), three task-worker pairs can be matched. From this toy example, we observe that it is beneficial to let idle workers move to areas that new tasks may appear, but the challenge here is how to predict accurately the time and the location of new coming tasks.

However, since tasks are always published individually, it is challenging to predict the specific position and time of the new coming tasks. This important problem recently has caused increasing attention. Several studies relax the problem to predict the future distribution of the new coming tasks using grid-based methods. Cheng et al. [4] used linear regression to predict the future number of workers/tasks of each grid cell. Tong et al. [23] compared the performance of several traditional statistical methods (e.g., ARIMA [3]) and traditional machine learning methods (e.g., GBRT [7]) on this problem. However, these methods are too simple and not good enough to model both spatial and temporal dependencies in task prediction problem, and the experimental results in this paper show that these methods have close performance and it is difficult to improve one over another.

Our study in this paper, for the first time, attempts to apply deep neural networks on addressing the task prediction problem. Deep neural networks have shown its success on diverse applications fields, and outperform traditional methods on modeling complex temporal and spatial feature dependency. For example, deep spatio-temporal residual network (ST-ResNet) in [29] is proposed to predict inflow and outflow of crowds in grid regions of a city by using convolution-based residual networks to model nearby and distant spatial dependencies between any two regions in a city, and the temporal properties of flows regarding temporal closeness, period, and trend.

Our problem of task appearance prediction shares similar challenges in traffic flow prediction on modeling the spatial dependency among regions and temporal dependency on dates and trend. However, the ST-ResNet model has a major limitation on learning the temporal dependency in historical sequences. It **concatenates data in one sequence as one tensor**, which weakens the sequential dependency in successive time units. Thus, we design a spatial-temporal residual network that learns from the **sequential historical data**, in a manner like Recurrent neural networks (RNN) and Long-short term memory (LSTM) learn from time series and word sequence.

The main contributions in this paper are summarized as follows:

– We analyze the spatial dependencies and temporal dependencies in task prediction problem, and targets on modeling the temporal dependencies in the sequences of historical task appearances at different scales, e.g., on the time interval level (e.g., half-hour) from $t-2$ to $t-1$ to t, on the daily level from one day to another, and on the weekly level from one week to another.
– We proposed a model called SeqST-ResNet, which well captures the temporal dependencies of historical task appearance in **sequences**, and the spatial dependencies among neighboring regions.
– We validate the effectiveness of our model on a real-world dataset. The experimental results show that our SeqST-ResNet model significantly outperforms the most competitive baseline ST-ResNet when predicting tasks at different time scales, and more importantly, in regions with intensive task requests.

The remainder of this paper is organized as follows. We give the definitions of task prediction problem and then analyze the spatial and temporal dependencies in Sect. 2. Section 3 presents the details of the proposed model. Experimental results are presented in Sect. 4 to demonstrate the performance of our methods. Related work and conclusion are discussed in Sects. 5 and 6, respectively.

2 Problem Definition and Analysis

2.1 Problem Definition

Following the previous work in [4,23], we address the task prediction problem by relaxing it to predict the number of task appearances in a specific spatio-temporal scope. In a city or region of interest, we divide its map into $M \times N$ grids, by partitioning along the longitude and latitude. The resolution of grid partition is controlled by M and N according to the application need. This practical grid-based map partition has been widely used in many spatio-temporal problems [4,23,29–31].

Based on the grid partition, at each time moment t, the task distribution can be considered as an image. We formally define it as:

Definition 1. *(Task image at time t) Task image of the whole area ($M \times N$ grids) at interval t can be defined as:*

$$\boldsymbol{X}_t = \begin{bmatrix} X_t^{(0,0)} & X_t^{(0,1)} & \cdots & X_t^{(0,N)} \\ X_t^{(1,0)} & X_t^{(1,1)} & \cdots & X_t^{(1,N)} \\ \vdots & \vdots & \ddots & \vdots \\ X_t^{(M,0)} & X_t^{(M,1)} & \cdots & X_t^{(M,N)} \end{bmatrix} \tag{1}$$

The element of task image matrix at the i-th row and the j-th column is

$$X_t^{(i,j)} = \sum_{P \in \mathbb{P}} |\{P_{loc} \in grid(i,j) \wedge P_{pt} = t\}| \tag{2}$$

which is the count of tasks P published at time P_{pt} at location P_{loc} belonging to grid cell $grid(i,j)$. Here, $\mathbb{P}$ denotes the task set in the spatial crowdsourcing system.

Then, our task appearance prediction problem is defined as

Definition 2. *(Task Prediction Problem) Given the sequence of historical task images $\{\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_{t-1}\}$, the goal is to predict $\boldsymbol{X}_t$ at next time t.*

2.2 Temporal and Spatial Dependency Analysis

To address the task appearance prediction problem, we discuss two important types of dependencies that should be included in prediction model design.

– **Temporal Dependencies:** at a given location, the task image value at a time t depends on the values before t. Such dependencies are often observed due to the time dependent properties of tasks. For example, at a central business district (CBD), the number of lunch orders on Grubhub or Eleme will have little differences between 11:30am–12:00pm and 12:00pm–12:30pm because these time intervals are both in lunch time. And the orders quantity

will be similar from Monday to Friday as they are workdays. Also, similar order numbers can be found from one weekend to another. More importantly, the dependency between t and $t-1$ is stronger than that between t and $t-2$. Therefore, the temporal dependencies should be modeled in the **sequence of task images at different scales**, e.g., on the time interval level (e.g., hourly or half-hourly) from $t-2$ to $t-1$ to t, on the daily level from $day1$ to $day2$ to $day3$, and on the weekly level from 1-st Monday to 2-nd Monday to 3-rd Monday and so on.
– **Spatial Dependencies:** the task image value at one location depends on the values at neighboring locations, because the neighboring grid cells may cover a same urban functional region. For example, at different grid cells around a commercial center. there will be similar number of taxi orders. Moreover, similar task appearance can even exist in distant grid cells when they cover the same type of functional region, e.g., train stations distributed in a big city.

3 The Proposed Approach, SeqST-ResNet

In this section, we will introduce our proposed model, SeqST-ResNet, which is a deep neural network model capturing the **sequential dependencies** in temporal features and spatial features of task appearance predication problem. We first present the network architecture and then discuss the network learning process, and its relevance to the ST-ResNet in [29].

3.1 SeqST-ResNet Architecture

As we discussed in the previous section, the sequential dependency on time-line is important for making correct prediction. Like in language models, video processing models and time-series models, the historical sequence is memorized for making predicting at the next time moment. Recurrent neural networks (RNN), Long-short term memory (LSTM), and all their variants have been widely used for this purpose. To incorporate the temporal dependency on different levels, we learn from the three types of historical sequences of task images:

$$\begin{array}{lr} \{\boldsymbol{X}_{t-n}, \boldsymbol{X}_{t-n+1}, \ldots, \boldsymbol{X}_{t-2}, \boldsymbol{X}_{t-1}\} & \text{interval-level} \\ \{\boldsymbol{X}_{t-n*p}, \boldsymbol{X}_{t-(n-1)*p}, \ldots, \boldsymbol{X}_{t-2*p}, \boldsymbol{X}_{t-p}\} & \text{day-level when } p = \text{one day} \\ \{\boldsymbol{X}_{t-n*q}, \boldsymbol{X}_{t-(n-1)*q}, \ldots, \boldsymbol{X}_{t-2*q}, \boldsymbol{X}_{t-q}\} & \text{week-level when } q = \text{one week} \end{array}$$

where p and q are parameters controlling the different level of temporal dependency to be modeled.

The architecture of our model, SeqST-ResNet, is shown in Fig. 2. The time axis on the top indicates the three types of dependency we modeled, denoting recent time (interval-level, in color red), near history (day-level, in color green) and distant history (week-level, in color blue). The sequence of each level (task images in the same color) goes through the same multi-layer deep network, including a convolution layer followed by several ResUnits, which are designed

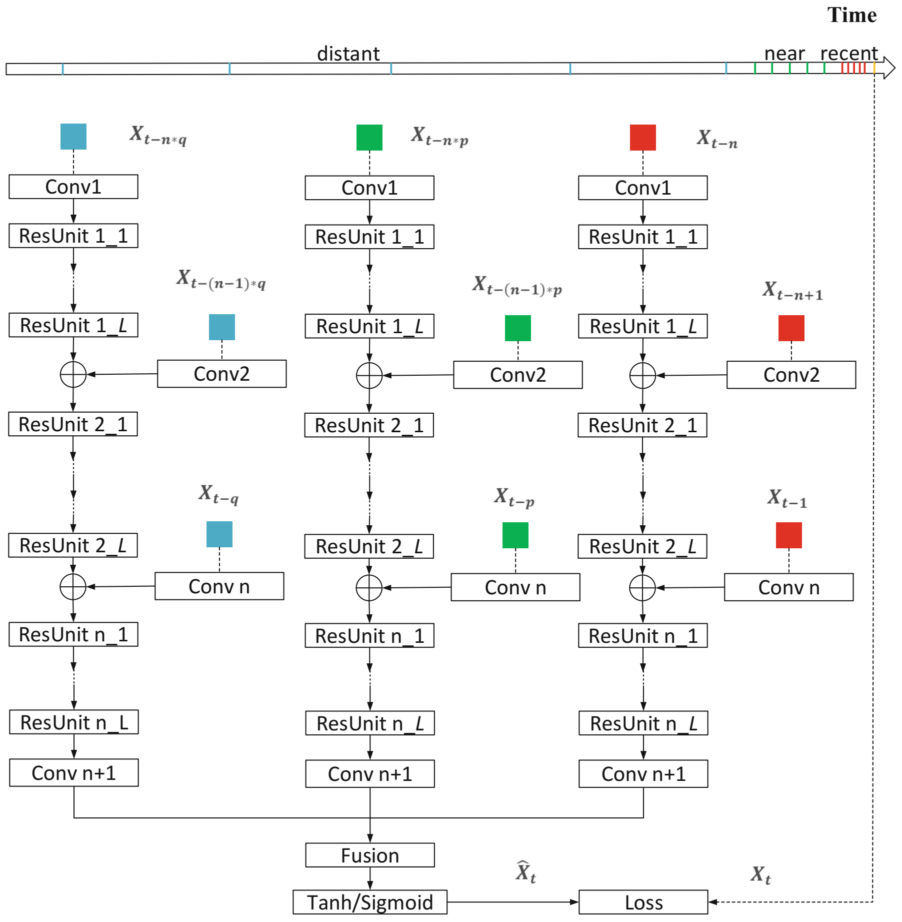

Fig. 2. SeqST-ResNet Architecture. The time axis on the top indicates three types of dependency, denoting recent time (interval-level, in color red), near history (day-level, in color green) and distant history (week-level, in color blue). The sequence of each level (task images in the same color) goes through the same multi-layer deep network, including a convolution layer followed by several ResUnits. The outputs from each sequence learning component are then combined by a fusion layer and followed by a final activation layer, as shown in bottom. (Color figure online)

following the Residual Network (ResNet) [9]. ResNet has been a great achievement in deep learning for addressing the gradients vanish problem and makes the deep network trainable even with over 1000 layers [10]. The outputs from each sequence learning component are then combined by a fusion layer and followed by a final activation layer, as shown in bottom of Fig. 2.

We next discuss the details of each sequence learning component, which consists of four parts: inputs, convolution layers, residual units, and addition layers.

Inputs. The input sequence of task images can be an interval-level sequence $\{\boldsymbol{X}_{t-n}, \boldsymbol{X}_{t-n+1}, \ldots, \boldsymbol{X}_{t-1}\}$, or a day-level sequence $\{\boldsymbol{X}_{t-n*p}, \boldsymbol{X}_{t-(n-1)*p}, \ldots, \boldsymbol{X}_{t-p}\}$, or a week-level sequence $\{\boldsymbol{X}_{t-n*q}, \boldsymbol{X}_{t-(n-1)*q}, \ldots, \boldsymbol{X}_{t-q}\}$, as we discussed previously.

Convolution Layers. The convolution operation captures the spatial dependency. Taking a task image, it outputs[1]:

$$\boldsymbol{X}_{t-n}^{(1)} = f(\boldsymbol{W}^{(1)} * \boldsymbol{X}_{t-n}^{(0)} + b^{(1)}) \tag{3}$$

where * denote the convolution operation and f is an activation function (e.g., ReLu, Tanh, or Sigmoid), $\boldsymbol{W}^{(1)}$ and $b^{(1)}$ are the learnable network parameters. With the increase of the number of the convolution layers, each unit of the feature map in the output of the component can cover more grids in input (task images). We use a 3×3 kernel as the filter. After one convolution layer, each unit of the output feature map can catch 9 pixels' information of the input. After two convolution layers, each unit can catch 27 pixels' information as the image is big enough. Thus, to capture the whole task image's information we use several convolution layers. To reduce training time, most convolution layers are contained in residual unit.

Residual Unit. The residual unit we use is shown in Fig. 3. Formally, a residual unit is defined as:

$$\boldsymbol{X}_l = \boldsymbol{X}_{l-1} + F(\boldsymbol{X}) \tag{4}$$

where $\boldsymbol{X}_l$ and $\boldsymbol{X}_{l-1}$ denote the input and output of a residual unit, respectively. Function F is a residual function which consists of multiple convolution layers and Batch Normalization [12] layers also with several ReLu transition function [17]. Batch Normalization is known by a lot of advantages, such as faster training, allowing, higher learning rate, easy to initialize network weights, regularization and improvement of network performance.

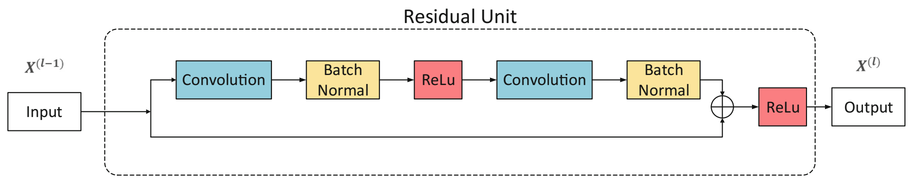

Fig. 3. Residual unit

Addition Layer. The addition layer, denoted by $\bigoplus$ serves for absorbing the task image into the sequence modeling component. Taking the first addition layer as an example, it operates

$$\boldsymbol{X}_{t-n+1}^{(1)} : \boldsymbol{X}_{t-n+1}^{(1)} + \boldsymbol{X}_{t-n}^{(l+1)} \tag{5}$$

[1] We show conv1 as an example. Other convolution layers follow the same function.

where $\boldsymbol{X}_{t-n+1}^{(1)}$ on the right is the output of conv2, and the second term $\boldsymbol{X}_{t-n}^{(l+1)}$ is the output of *ResUnit 1_L*. After merging $\boldsymbol{X}_{t-n}^{(l+1)}$, $\boldsymbol{X}_{t-n+1}^{(1)}$ is sent forward to next layers.

To demonstrate the end-to-end SeqST-ResNet model, we take a simple sequence of two task images, $\{\boldsymbol{X}_0, \boldsymbol{X}_1\}$, as input to one sequence modeling component (e.g., the interval-level model). The feed-forward procedure will be:

$$\begin{aligned}
\boldsymbol{X}_0^{(1)} &= f_0(\boldsymbol{W}_0^{(1)} * \boldsymbol{X}_0^{(0)} + b_0^{(1)}) \\
\boldsymbol{X}_1^{(1)} &= f_1(\boldsymbol{W}_1^{(1)} * \boldsymbol{X}_1^{(0)} + b_1^{(1)}) \\
\boldsymbol{X}_0^{(l+1)} &= \boldsymbol{X}_0^{l} + F_0^{(l)}(\boldsymbol{X}_0^{(l)}) \text{ (for } l \text{ from 1 to } L) \\
\boldsymbol{X}_1^{(1)} &= \boldsymbol{X}_1^{(1)} + \boldsymbol{X}_0^{(l+1)} \\
\boldsymbol{X}_1^{(l+1)} &= \boldsymbol{X}_1^{l} + F_1^{(l)}(\boldsymbol{X}_1^{(l)}) \text{ (for } l \text{ from 1 to } L)
\end{aligned}$$

The fusion layer will coalesce the output of three components with a simple parameter-matrix-based method. Suppose $\boldsymbol{X}_c$, $\boldsymbol{X}_p$, $\boldsymbol{X}_q$ are the output of each component respectively, the output of the fusion layer is:

$$\boldsymbol{X}_{out} = \boldsymbol{W}_c * \boldsymbol{X}_c + \boldsymbol{W}_p * \boldsymbol{X}_p + \boldsymbol{W}_q * \boldsymbol{X}_q \tag{6}$$

where the symbol $*$ is the hadamard product. The final activation function is Sigmoid, which predicts the output $\widehat{\boldsymbol{X}_t} = sigmoid(\boldsymbol{X}_{out})$. The loss function MSE (Mean Square Error) is defined to measure the difference between the predicted task image at t and the ground truth $\boldsymbol{X}_t$:

$$loss = ||\boldsymbol{X}_t - \widehat{\boldsymbol{X}_t}||_2^2 \tag{7}$$

3.2 Network Training

We use Adam algorithm as the optimizer. The learning rate is set as 0.003, and batch size is set as 16. The convolution kernels are with size of 3×3 both in convolution layers and residual blocks.

To reduce the influence of anomalies in training task image sequences, we apply moving average (e.g., with window length 3) on the input sequences. Min-max normalization is used for data pre-processing. The length of the interval-level and day-level sequences is set as 3, and the length of the week-level sequences is set to be 1 unless specified differently. The influence of these length parameters will be analyzed in the experimental result section.

3.3 Discussion

Our proposed model SeqST-ResNet shares the similar architecture of using three components of ResNet in ST-ResNet [29]. Our interval-level component corresponds to the closeness component in ST-ResNet, while day-level and week-level component corresponds to the period and trend component in ST-ResNet, respectively. The key difference lies in how to model the historical sequences. In each component of ST-ResNet, the fragments in one sequence are **concatenated as one tensor**, and then modeled by convolution layer and residual units. In our proposed SeqST-ResNet, **the task images in a sequence are modeled by considering their time order, and thus captures the temporal dependency in a more reasonable way than taking concatenation as a tensor**. The evaluation results in next section also verify that our sequential deep network architecture can better model the temporal dependency in task image sequences and achieves better prediction results than ST-ResNet in [29].

4 Experiments

In this section, we evaluate our proposed model SeqST-ResNet on a real-world dataset and compare it with the state-of-the-art models.

4.1 Dataset

The dataset we use is the taxi request data in Chengdu, China, provided by Didi GAIA Open Dataset [1]. The detail information of this dataset is shown in Table 1. The whole area covered by the dataset in Chengdu is divided by a 10×10 grid-based map partition. We set time interval as 30 min, and then get 2928 intervals totally from Oct. 1st to Nov. 30th, 2016.

Table 1. Dataset information

City	Chengdu, China
Time span	10/1/2016–11/30/2016
# taxi orders	11779076
Time interval	30 min
# intervals	2928
Grid-based map partition	10×10
Area of each grid	Nearly 800×800 m^2

4.2 Baselines and Settings

We compare our method with the following baselines including both traditional methods and deep learning methods:

- **HA:** Historical Average. This naive approach uses the historical average value in the same interval and same grid as the prediction.
- **ARIMA:** Auto-Regressive Integrated Moving Average [3]. It is a well-known time series prediction model.

Table 2. Prediction errors of different models

	Model	RMSE
Baselines	HA	22.99
	ARIMA	17.40
	LSTM-48	39.87
	LSTM-96	39.97
	DeepST	18.52
	ST-ResNet	16.28
Our methods	SeqST-ResNet-1AVG	14.00
	SeqST-ResNet-3AVG	**12.95**
	SeqST-ResNet-5AVG	12.97
	SeqST-ResNet-7AVG	13.61

- **LSTM:** Long-Short-Term-Memory Network [11]. Its chain like neural network structure is capable of learning long-term dependencies.
- **ST-ResNet:** Spatio-Temporal Residual Network [29]. The deep learning model is designed for predicting traffic flow based on historical data.
- **DeepST:** Deep Spatio-Temporal Network [30]. It is similar to ST-ResNet, but without using residual units.

For all methods, we select the data in the last 7 days (nearly 10%) for evaluating the prediction accuracy, while all data before that are used for training. The parameters of deep learning models (DeepST, ST-ResNet and our SeqST-ResNet) are set to the same values: $p = 48$ intervals (24 h) and $q = 48 \times 7$ intervals (one week), such that day-level and week-level sequences are fed with interval-level sequences to the corresponding components in the deep networks. The parameter settings in other baselines methods are tuned for achieving their best performance.

Evaluation Metrics. We use the most common metrics in this paper: Root Mean Squared Error (RMSE) for evaluation:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{X}_i - \widehat{\boldsymbol{X}_i}\right)} \tag{8}$$

The results reported next are the average of 5 independent runs.

4.3 Results

Overall Prediction Performance. Table 2 presents the prediction results of baseline methods and our proposed models with different smoothing window sizes (SeqST-ResNet-τAVG represents our model is trained by smoothed sequences with window size τ). The lower RMSE value indicates more accurate prediction. We can see that our SeqST-ResNet models consistently outperform the baseline methods. Especially SeqST-ResNet-3AVG shows the best performance with RMSE of 12.95. Obviously, smoothing of sequences can improve the results due to the elimination of anomalies (SeqST-ResNet-1AVG without smoothing is not as good as other SeqST-ResNet settings). One interesting observation is that SeqST-ResNet-5AVG has the closest RMSE (12.97) to the best result, and SeqST-ResNet-7AVG has higher RMSE (13.61). This is due to the over-smoothing of the sequences with a large window (length of 7 intervals means 3.5 h). Over all, the results in Table 2 verify that our proposed SeqST-ResNet model can capture well the sequential dependency in time order and among spatial locations, and thus make more accurate prediction of future task appearance than other models.

To demonstrate the model training and testing errors in the learning process, the training error curve and test curve of epochs are shown in Figs. 4 and 5. We can see clearly that SeqST-ResNet models always have lower error than DeepST and ST-ResNet in both training and testing. Moreover, SeqST-ResNet-7AVG is worse than SeqST-ResNet-5AVG and SeqST-ResNet-3AVG, but still better than SeqST-ResNet-1AVG without using smoothing. The training error curve and test error curve have the same tendency and similar value, which indicates that the parameters we use in these models are suitable and do not result in over-fitting or under-fitting issues.

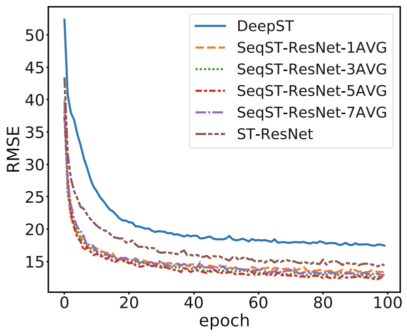

Fig. 4. Training error

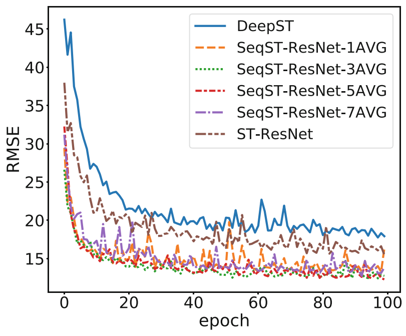

Fig. 5. Test error

Prediction Performance with Respect to Time. To evaluate the prediction accuracy at different time moments during a day, and on weekdays and weekends, we compare our best model SeqST-ResNet-3AVG and the most competitive baseline ST-ResNet. Figure 6(a) and (b) show that our model has lower prediction errors at the half-hour resolution prediction during a day, and also much lower prediction errors during weekends and weekdays. That is to say, our model can predict the task appearance at different time scales more accurately than ST-ResNet. This is mainly because our model learns from the sequences of task images at different time scales, rather than taking concatenation of tasks images as done in ST-ResNet.

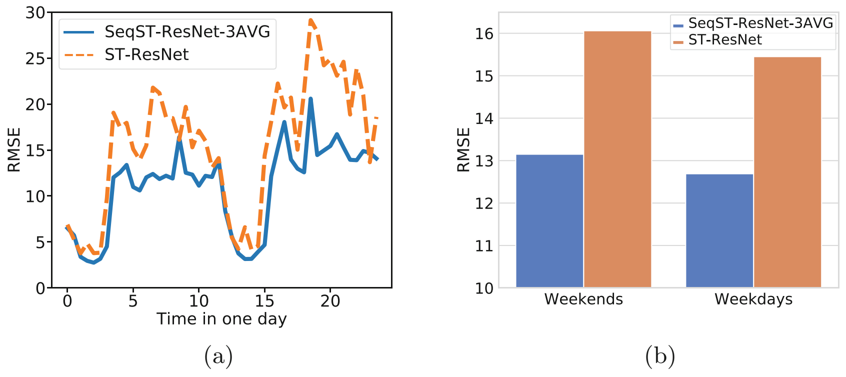

Fig. 6. Prediction at different time in one day (a) and at weekends and weekdays (b)

Prediction Performance on Regions with Different Request Intensity. We are also interested in evaluating the prediction performance of our model on regions with different true task frequency, e.g., request intensive regions vs mild regions. We calculate the average of ground truth task appearance frequency in each grids (10×10 grids), and then sort these grid cells by the average frequency with ascending order (from the most idle region to the most busy region). Figure 7 shows the prediction performance on these grid cells ordered on x-axis. In regions with intensive task requests, our SeqST-ResNet-3AVG model has much lower error than the baseline ST-ResNet model, while they have similar performance in mild regions. This is an important advantage, because correct prediction of tasks in request intensive regions (e.g., around commercial centers, central transportation stations) will highly improve the task assignment efficacy in spatial crowdsourcing platform.

Influence of the Sequence Length. Our model learns from three types of sequences. We also evaluate how the sequence length can affect the prediction performance. Let l_i, l_d, l_w denote the length of interval-level sequence, day-level sequence and week-level sequence, respectively. In Fig. 8(a), we show the prediction error when changing l_i from 1 to 5, while fixing $l_d = 1$, $l_w = 1$. The result shows that with l_i increasing, the RMSE decreases. This indicates longer interval-level sequences can help on improving prediction accuracy, but also takes more training time. An appropriate setting is $l_i = 3$ because the error decreases slowly after l_i is larger than 3. Figure 8(b) shows the influence of l_d when fixing $l_i = 3$, $l_w = 1$. The results show that the model performs best when $l_d = 3$. Neither too long nor too short day-level sequence is helpful. Then we set $l_i = 3$ and $l_d = 3$, and show the impact of l_w in Fig. 8(c) where we find $l_w = 1$ is the best setting.

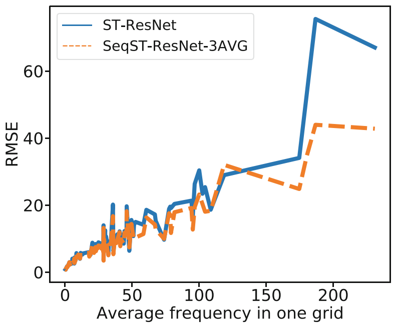

Fig. 7. Prediction on regions with different request intensity

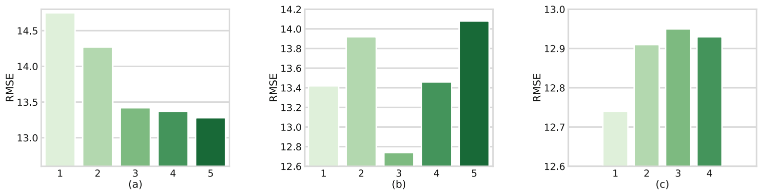

Fig. 8. Impact of sequence length at different levels. In (a), the length of interval-level sequence l_i varies when $l_d = 1$, $l_w = 1$. In (b), the length of day-level sequence l_d varies when $l_i = 3$, $l_w = 1$. In (c) the length of week-level sequence l_w varies when $l_i = 3$ and $l_d = 3$.

5 Related Work

In this section we briefly review some recent advances on task assignment in spatial crowdsourcing and deep learning since the focus of this paper is to apply deep neural networks on addressing the task prediction problem during task assignment in spatial crowdsourcing.

In [13], Kazemi and Shahabi propose several heuristics to maximize the number of assigned tasks in a given time interval while meeting the constraints specified by workers. In practice, tasks often arrives dynamically. This kind of online scenarios is more challenging and has been addressed in [20,21,23] where efficient algorithms with provable competitive ratio are proposed. Song et al. in [18] extend conventional task assignment from two objects matching problem to trichromatic matching problem. In [15,16,25,27,28], privacy of user or platform are considered when making the task assignment. In [4], spatial distribution of workers and tasks are taken into account when maximizing a global assignment quality score. In [14,24], travel time, as an important factor, the prediction of which has drawn some attentions recently. [32] tackles the problem of assigning tasks to workers such that mutual benefit are maximized. All these studies fail to model the spatial dependency among regions and temporal dependency on successive time units, which is the focus of our work in this paper. To the best of our knowledge, there is no deep learning based method to solve the task prediction problem. Thus, it is necessary to design a deep neural network based method.

On the other hand, recent years have witnessed the big success of deep learning in a variety of application domains. Specifically, there are lots of achievements in catching spatial or temporal properties. For temporal property, recurrent neural networks (RNN) [6] is designed to make use of sequential information, and has been shown great success in many NLP tasks [19]. However, vanishing gradient problem causes it to be difficult to capture the long-term dependency [2]. Long short term memory networks (LSTM) [11] improved RNN by using "gates" which control what to forget or remember. Gated Recurrent Unit (GRU) [5] simplifies LSTM by reducing the "gates" from 3 to 2. Bidirection LSTM (BiLSTM) [8] not only considers the forward data flow but also backward. For spatial property, convolution neural networks (CNN) can effectively capture the spatial property from near to distant with the depth deeper. Residual Networks (ResNet) [9] makes the deep network realize really "deep" even over 1000 layers. For combination of spatial and temporal properties, convolution LSTM (ConvLSTM) [26] and deep spatial temporal networks (DeepST) [29] are proposed to capture the two properties. However none of them can model long dependency as the training cost is really huge in practice.

6 Conclusion and Future Work

In this paper, we study the problem of predicting the task appearance in a spatial crowdsourcing platform. To take advantage of the temporal dependency of the historical sequential task request and the spatio-dependency in neighboring regions, we proposed a novel deep network model that learns from the sequences of task image data at different time scales. Experimental results on a real dataset demonstrated that our methods can significantly improve the prediction accuracy when comparing with the baselines. In future, we will consider to add the attention mechanism to further reduce the prediction error, and extend the application to other spatial crowdsourcing data such as meal order data.

Acknowledgements. This research was supported in part by National Natural Science Foundation of China (NSFC) (Grant No. 61572336, 61632016, 61572335, 61772356), and Natural Science Research Project of Jiangsu Higher Education Institution (No. 18KJA520010, 17KJA520003), and the King Abdullah University of Science and Technology (KAUST), and the Blockshine Technology corp. Data source: Didi Chuxing.

References

1. Didi Chuxing. https://gaia.didichuxing.com
2. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)

3. Box, G.E., Pierce, D.A.: Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. J. Am. Stat. Assoc. **65**(332), 1509–1526 (1970)
4. Cheng, P., Lian, X., Chen, L., Shahabi, C.: Prediction-based task assignment in spatial crowdsourcing. In: 33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, 19–22 April 2017, pp. 997–1008 (2017)
5. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
6. Elman, J.L.: Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990)
7. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**(5), 1189–1232 (2001)
8. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. **18**(5–6), 602–610 (2005)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
10. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 630–645. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_38
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
12. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
13. Kazemi, L., Shahabi, C.: GeoCrowd: enabling query answering with spatial crowdsourcing. In: Proceedings of the 20th International Conference on Advances in Geographic Information Systems, pp. 189–198. ACM (2012)
14. Li, Y., Fu, K., Wang, Z., Shahabi, C., Ye, J., Liu, Y.: Multi-task representation learning for travel time estimation. In: International Conference on Knowledge Discovery and Data Mining, KDD (2018)
15. Liu, A., et al.: Privacy-preserving task assignment in spatial crowdsourcing. J. Comput. Sci. Technol. **32**(5), 905–918 (2017)
16. Liu, A., Wang, W., Shang, S., Li, Q., Zhang, X.: Efficient task assignment in spatial crowdsourcing with worker and task privacy protection. GeoInformatica **22**(2), 335–362 (2018)
17. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning, ICML 2010, pp. 807–814 (2010)
18. Song, T., et al.: Trichromatic online matching in real-time spatial crowdsourcing. In: ICDE, pp. 1009–1020 (2017)
19. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
20. Tong, Y., Chen, L., Shahabi, C.: Spatial crowdsourcing: challenges, techniques, and applications. PVLDB **10**(12), 1988–1991 (2017)
21. Tong, Y., She, J., Ding, B., Chen, L., Wo, T., Xu, K.: Online minimum matching in real-time spatial data: experiments and analysis. PVLDB **9**(12), 1053–1064 (2016)
22. Tong, Y., She, J., Ding, B., Wang, L., Chen, L.: Online mobile micro-task allocation in spatial crowdsourcing. In: 32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, 16–20 May 2016, pp. 49–60 (2016)

23. Tong, Y., et al.: Flexible online task assignment in real-time spatial data. Proc. VLDB Endow. **10**(11), 1334–1345 (2017)
24. Wang, Z., Fu, K., Ye, J.: Learning to estimate the travel time. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 858–866. ACM (2018)
25. Xiao, M., et al.: SRA: secure reverse auction for task assignment in spatial crowdsourcing. IEEE Trans. Knowl. Data Eng. 1 (2019). https://doi.org/10.1109/TKDE.2019.2893240
26. Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W., Woo, W.: Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: Advances in Neural Information Processing Systems, pp. 802–810 (2015)
27. Zhai, D., et al.: Towards secure and truthful task assignment in spatial crowdsourcing. World Wide Web 1–24 (2018). https://doi.org/10.1007/s11280-018-0638-2
28. Zhang, D., Chow, C.Y., Liu, A., Zhang, X., Ding, Q., Li, Q.: Efficient evaluation of shortest travel-time path queries through spatial mashups. GeoInformatica **22**(1), 3–28 (2018)
29. Zhang, J., Zheng, Y., Qi, D.: Deep spatio-temporal residual networks for citywide crowd flows prediction. In: AAAI, pp. 1655–1661 (2017)
30. Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X.: DNN-based prediction model for spatio-temporal data. In: Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, p. 92. ACM (2016)
31. Zhao, J., Xu, J., Zhou, R., Zhao, P., Liu, C., Zhu, F.: On prediction of user destination by sub-trajectory understanding: a deep learning based approach. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, 22–26 October 2018, pp. 1413–1422 (2018)
32. Zheng, L., Chen, L.: Mutual benefit aware task assignment in a bipartite labor market. In: ICDE, pp. 73–84 (2016)