Exploring Trajectory Prediction through Machine Learning Methods

Chujie Wang, Lin Ma, Rongpeng Li, Tariq S. Durrani, and Honggang Zhang

Abstract—Human mobility prediction is of great importance in a wide range of modern applications in different fields such as personalized recommendation systems, the fifth-generation (5G) mobile communication systems, and etc. Generally, the prediction goal varies from different application scenarios. For the applications of 5G network including resource allocation and mobility management, it is essential to predict the positions of mobile users in the near future from dozens of seconds to a few minutes so as to make preparation in advance, which is actually a trajectory prediction problem. In this paper, with the particular focus on multi-user multi-step trajectory prediction, we first design a basic deep learning-based prediction framework where the Long Short-Term Memory (LSTM) network is directly applied as the most critical component to learn user-specific mobility pattern from the user's historical trajectories and predict his/her movement trends in the future. Motivated by the related findings after testing and analyzing this basic framework on a model-based dataset, we extend it to a region-oriented prediction scheme and propose a multi-user multi-step trajectory prediction framework by further incorporating the Sequence-to-Sequence (Seq2Seq) learning. Experimental results on a realistic dataset demonstrate that the proposed framework has significant improvements on generalization ability and reduces error-accumulation effect for multi-step prediction.

Index Terms—Trajectory Prediction, Multi-Step Prediction, Long Short-Term Memory, Sequence-to-Sequence, Machine Learning.

I. INTRODUCTION

Increasing pervasive usage of smart-phones and location-based services around the world has contributed to vast and rapid growth in mobility data. The large size of mobility data provides new opportunities for discovering the characteristics of human mobility patterns and making mobility predictions. Practically, human mobility prediction is of great importance in a wide range of modern applications, ranging from personalized recommendation systems to intelligent transportation, urban planning, and mobility management in the fifth-generation (5G) mobile communication system [1] [2]. Generally, the prediction goal varies from different application scenarios. For the case of 5G mobile communications, it is essential to predict the positions of mobile users in the near future from dozens of seconds to a few minutes so as to prepare for mobility management and resource allocation [2]. It is actually a trajectory prediction problem where the trajectory refers to a time series of positions with a fixed sampling time interval between each other.

Although researchers have proposed many mobility prediction methods, such as frequent patterns mining [3] [4], Markov-based models [5] [6] and other machine learning methods [7], most of these methods are dedicated to discrete location prediction, which is actually a multi-classification problem, and not suitable for predicting trajectories with fixed sampling time intervals. The reasons are as follows. On one hand, for trajectories composed of discrete location indexes, locations may keep same for several consecutive time-steps when the sampling time interval is small, while locations may have a mutation between two adjacent time-steps when the sampling time interval is large. Therefore, they can hardly reflect user movement trends effectively. On the other hand, for trajectories composed of continuous location coordinates, it is hard to specify the discretization granularity of coordinates. Generally, high discretization granularity benefits to reflect user movement trends. However, the prediction accuracy may decrease with increasing number of candidate locations under high discretization granularity.

In order to avoid the above problems, this paper takes comprehensive investigation for the approaches to predict trajectories composed of continuous coordinates. Since it is actually a time series regression prediction problem, conventional regression algorithms such as linear regression [8] and support vector regression (SVR) [9] are candidate solutions. Besides, autoregressive integrated moving average (ARIMA) is another regression algorithm. It is dedicated to processing prediction problems for long time series composed of numerical data with quantity relationship, such as stock prediction [10] and traffic prediction [11]. However, the mobility trajectories are usually short sequences composed of two-dimensional coordinates reflecting geographic locations, making ARIMA possibly not competent to the trajectory prediction problem. Fortunately, within the framework of deep learning, the Recurrent Neural Network (RNN) has proved its superiority in various time series problems not only in natural language processing field (i.e. machine translation [12], speech recognition [13]) but also some other fields (i.e. traffic prediction [14], precipitation prediction [15]). Therefore, as the improved versions of typical RNN, Long Term Short Term Memory (LSTM) [16] and Gate Recurrent Unit (GRU) [17] are promising algorithms for the trajectory prediction problem.

Benefiting from the latest advance in deep learning, this paper makes a detailed exploration of the trajectory prediction
problem from both the single-user perspective and multi-user perspective. The main contributions of this paper can be summarized as follows:

- We propose an LSTM-based single-user prediction framework and evaluate its performance on a model-based dataset. Experimental results demonstrate the capability of LSTM to predict user’s mobility based on pre-learning of the user’s mobility patterns. We also highlight some challenges (e.g., poor generalization ability, annoying error-accumulation effect) of this user-specific prediction scheme.

- To cope with these challenges, we further extend the user-specific prediction scheme to a region-oriented prediction scheme and put forward a multi-user multi-step trajectory prediction framework based on the Seq2Seq learning. Besides, we introduce a variable teacher ratio to control information transferring in the training process.

- Finally, we show empirically that the proposed multi-user multi-step trajectory prediction framework can effectively mitigate the error-accumulation effect and improve the generalization ability on a realistic dataset.

The rest of this paper is organized as follows. Section II presents some related works. Section III formulates the trajectory prediction problem, describes the dataset and introduces the fundamentals of neural networks. In Section IV, we give the experimental results of the basic single-user-specific LSTM framework and highlight its application challenges by some preliminary results. Then, we introduce a region-oriented multi-user prediction framework by further incorporating Seq2Seq technique in Section V. We finally conclude this paper in Section VI.

II. RELATED WORK

There have been some theoretical mobility models proposed to mimic the movements of mobile users and simulate their mobility patterns using parametric methods synthetically, such as Random Walk mobility model [18], Gauss-Markov mobility model [19], Levy-Walk mobility model [20], and so on. Although these models are relatively simple, they can hardly describe the movement of different users in a complex and volatile real environment, making it unreasonable to apply them in practice.

Besides, a number of previous efforts have attempted to model user mobility based on real-life movement trajectories. Early methods related to mobility prediction mainly focus on discovering frequent trajectories and then performing trajectory matching to predict the location of a moving object [3] [4]. However, these methods are computationally cost and suffer from the data sparsity problem. Another widely used mobility prediction methods fall into the scope of Markov-based models [5] [6]. The authors in [5] propose a hidden Markov model (HMM)-based trajectory prediction algorithm to discover transition rules from one location to another. Lv et al. [6] further combine the HMM model with user’s living habits for an individual to achieve effective location prediction. In addition, other conventional machine learning techniques such as K-nearest neighbor (KNN) and decision tree have also been applied for location recognition and prediction in [7]. However, these methods need the locations to be discrete, thus not applicable to trajectories composed of continuous coordinates with small sampling time intervals.

Within the framework of deep learning, the work in [21] applies LSTM to trajectory prediction for vehicles on highway. However, the proposed method is specifically designed for the highway scenario and requires complex external features, including position and velocity of surrounding vehicles, which restricts its general applicability. Alahi et al. propose a social LSTM network for pedestrian trajectory prediction [22]. However, it can only predict human trajectories through static-images under a specific small range scene such as hotels and intersections. Feng et al. [23] propose a DeepMove model which combines the GRU network with the attention mechanism to predict future discrete locations from long-range and sparse trajectories. However, its prediction accuracy can only reach 59.3% in cellular network scenarios since it is difficult to capture the trend of user movements in each cell from trajectories composed of discrete cells.

Trajectory prediction has a wide range of applications in 5G networks, such as radio resource pre-allocation [2], caching decision at the wireless edge [24], mobility management [25], and etc. For example, in order to mitigate the negative impact of frequent handovers in dense networks, our previous work in [25] proposes an intelligent dual connectivity mechanism for mobility management based on trajectory prediction, which improves the quality of service of mobile users in the handover process while guaranteeing the network energy efficiency. Moreover, driven by the stringent safety requirement of autonomous driving and advanced driver assistance systems, it is critical to understand the intentions of surrounding vehicles through trajectory prediction [21] [26]. Therefore, trajectory prediction is a problem worth well careful studying.

III. MATHEMATICAL BACKGROUND

In this section, the trajectory prediction problem is formulated first followed by the dataset description. Then, we introduce the fundamental concept of LSTM and Seq2Seq. For better understanding, Table I lists the main symbols in this paper.

A. Problem Formulation

**Definition 1: Trajectory.** We denote a trajectory as $T = \{p_1, p_2, \ldots\}$ where $p_i = (p_i.x, p_i.y) (i = 1, 2, \ldots)$ is a two-dimensional coordinate representing the position at time $i$. The sampling time interval between each two adjacent points is fixed and denoted as $\Delta$.

**Problem 1: Trajectory Prediction.** Given a trajectory $T = \{p_1, p_2, \ldots, p_T\}$ of length $T$, our objective is to predict the sequence of the next $K$ step location points. The problem can be represented as:

$$\hat{p}_T \cdots \hat{p}_{T+K} = \arg\max_{p_T \cdots p_{T+K}} P(p_T \cdots p_{T+K} | p_1, p_2, \ldots, p_T),$$

where $p_i = (p_i.x, p_i.y), (i = 1, \ldots, T + K)$.
B. Dataset Description

In order to evaluate the performance of the mobility prediction framework, we adopt two types of datasets (i.e., a model-based dataset and a realistic dataset) for the reasons as follows. Given the strong randomness of user mobility, it is necessary to testify the performance of a proposed algorithm in a realistic environment. But considering that a realistic dataset is often collected in a user-voluntary manner, the dataset usually consists of user mobility trajectories lasting short duration and possessing irregular starting and ending time. Hence, we generate a model-based dataset from well-known models to assist in finding some intuitive guidance.

1) Model-based Dataset: Based on the fundamental statistical properties of human mobility [27] [28], a number of mobility models have been proposed to generate human-like trajectories [20] [29]. Taking comprehensive consideration of both practicality and complexity of these models, we refer to the Self-Similar Least-Action Human Walk (SLAW) [20] and the SMOOTH model [29] to generate our mobility data. Specifically, we generate exclusive mobility pattern for each user and capture their location for six hours (360min) at one-minute granularity in a simulation area of 4000m × 4000m each day. Fig. 1 depicts the simulation area and one sample trajectory of a user.

2) Realistic Dataset: We utilize a large real-life GPS trajectory dataset from the Geolife project [30] of Microsoft Research Asia. The dataset was collected by 182 users, containing 18,670 trajectories with various sampling rates. Each trajectory is represented by a series of timestamp points with latitude and longitude coordinates recorded by GPS-functioned phones. As an essential work for a large and messy raw dataset, we take the following preprocessing steps. Firstly, we select the location records in Beijing and convert the geographic coordinates represented by latitude and longitude into two-dimensional plane coordinates by spatial coordinate projection. Secondly, to remove some noise points caused by the poor signal of location positioning systems, we adopt mean filtering by a sliding window covering $w$ temporally adjacent points. As suggested in [31], a sliding window of size 3 or 5 can meet the denoising requirement for individual noise points. However, for consecutive noise points, a larger size of sliding window is needed. Meanwhile, it should be noted that a large sliding window also leads to a big error between the estimated position and the true position. Therefore, after analyzing the realistic dataset, we select $w = 5$ to trade off between denoising and preserving valid information in trajectories. In this case, for a measured location point $p_i = (p_i.x, p_i.y)$, the estimation of its true value is calculated by $p_i.x = \frac{1}{5} \sum_{m=-2}^{2} p_{i+m}.x$ and $p_i.y = \frac{1}{5} \sum_{m=-2}^{2} p_{i+m}.y$. Thirdly, since most of the trajectories are sampled in high resolution, we compress the trajectories with fixed sampling time interval $\Delta$ to eliminate redundancy in raw data. Finally, we segment the trajectories with a fixed length (i.e. 15 in this paper). The preprocessed trajectories are shown in Fig. 2.

C. Preliminary

1) LSTM: As illustrated in Fig. 3, the LSTM neural network is composed of multiple copies of basic memory blocks and each memory block contains a memory cell and three types of gates (input gate, output gate, and forget gate). The memory cell is the key component of LSTM and responsible
Predicted position at time-step $t$.

The length of the predicted trajectory.

The sampling time interval of the trajectories.

For the information transfer at different time-steps. Meanwhile, the three gates, each of which contains a sigmoid layer to optionally pass information, are responsible for protecting and controlling the cell state. As its name implies, the forget gate controls which part of the input will be utilized to update the cell state. Similarly, the forget gate controls which part of the old cell state will be thrown away, while the output gate determines which part of the new cell state will be output.

The memory block at time-step $t$, we use $f_t$, $i_t$, and $o_t$ to represent the forget, input, and output gates respectively. Assume that $x_t$ and $h_{t-1}$ represent the input and output at the current time-step, $h_{t-1}$ is the output at the previous time-step, $\sigma$ represents the sigmoid activation function, and $\odot$ denotes the Hadamard product, the key equations of the LSTM scheme are given below [16]:

$$
\begin{align*}
  f_t &= \sigma(W_x f_t + W_h h_{t-1} + b_f) \\
  i_t &= \sigma(W_x i_t + W_h h_{t-1} + b_i) \\
  o_t &= \sigma(W_x o_t + W_h h_{t-1} + b_o) \\
  c_t &= \tanh(W_x c_t + W_h h_{t-1} + b_c) \\
  h_t &= o_t \odot \tanh(c_t)
\end{align*}
$$

where $W$ and $b$ are the corresponding weight matrices and biases of the three gates and the memory cell with subscripts $f$, $i$, and $o$ for the forget, input, and output gates respectively, while the subscript $c$ is used for the memory cell. $x_t$ and $h_{t-1}$ represent the input and output at time-step $t$, $\sigma$ represents the sigmoid activation function.

2) Sequence to Sequence (Seq2Seq): Seq2Seq is specifically designed for the learning and prediction of sequences [32] [33]. It maps input sequences with arbitrary length into variable-length output sequences, such as sentences in text or speech. It has been widely applied in the field of machine translation and question answering systems and has achieved good results. As shown in Fig. 4, a Seq2Seq framework consists of two different neural networks, an encoder network and a decoder network. They can be either simple single-layer of RNNs or LSTMs, or multi-layer stacks of them. The encoder is responsible for reading the input sequence and converting it into a fixed-length vector as an overall representation. For example, in the case of LSTM, the overall representation is the last hidden state vector $h_T$ and the memory cell state vector $c_T$. Then, the decoder uses the overall representation to initialize its own internal state and subsequently estimate the correct output sequence step by step during the iteration process. The output of each step represents the predicted result at that moment. Generally, the decoder part is designed as an auto-regressive model where the output of the previous step will be used as the input of the next step.

IV. LSTM-BASED SINGLE-USER PREDICTION FRAMEWORK AND SIMULATION RESULTS

In this section, we investigate the user-specific scheme for trajectory prediction problem. We put forward an LSTM-based trajectory prediction framework and evaluate it on a model-based dataset to assist in finding some intuitive guidance.

A. Prediction Framework Design

In the model-based dataset, different users typically have distinct mobility patterns, making the mobility prediction problem to be user-specific. Therefore, in order to make mobility predictions for a user, the most critical step is to establish a specific mobility model which fully represents the user’s mobile pattern from his/her historical trajectories. Fig. 5 presents the proposed LSTM-based single-user prediction framework. The prediction process involves three major steps. First, the given trajectory is processed by a fully connected (FC) input layer with 128 neurons so that each
two-dimensional coordinate is mapped to a 128-dimensional feature tensor. Then, the processed sequence is sent to the main part of the mobility model, a deep recurrent neural network formed by three stacked LSTM layers each with 128 neurons. Each LSTM layer takes the output of the previous layer as input and feeds its output to the next layer. Finally, an FC output layer with 2 neurons maps the output of the last LSTM layer to the predicted location of the next time-step, and thereby we get the prediction sequence \( \hat{p} = \{ \hat{p}_1, \hat{p}_2, \ldots, \hat{p}_T \} \). The training goal is to minimize the distance error between the predicted location and the actual location. Thus, we choose the Mean Square Error (MSE) as the loss function and adopt Backward Propagation Through Time (BPTT) algorithm [34] to update the network parameters. Ultimately, the user’s mobility pattern is saved in the mobility model as network parameters and the prediction of future trajectory can be completed based on the trained mobility model. The complete training algorithm is presented in Algorithm 1.

**B. Mobility Prediction Results**

In this part, we evaluate the prediction performance of the proposed framework on a user’s trajectories from the model-based dataset. The training settings are shown in Table II. The length of each trajectory is 360min at one-minute granularity. In order to fully learn the user’s mobile pattern, during the training process, we take the complete trajectory except the last minute (i.e. \( \{ p_1, p_2, \ldots, p_{359} \} \)) as input and push the time-series forward one minute as standard output (i.e. \( \{ p_2, p_3, \ldots, p_{360} \} \)). During the test process, after one-hour observation, we first make single-step predictions given the user’s real position at each time-step. Then, in order to evaluate the prediction performance comprehensively, the case of multi-step prediction where the real position becomes rapidly unavailable is also considered. In this case, we recursively reusing the recent prediction results as input for the following prediction step.

For comparison, we also use the conventional linear regression algorithm to fit the user’s movements and make predictions.
tion of the user’s trajectory while the LSTM model yields predictions with superior accuracy. As shown in Fig. 6(c), both methods perform well and the error remains below 20m for most of single-step prediction cases. However, for the multiple-step prediction case as shown in Fig. 6(d), LSTM model can make relatively more reliable predictions with error less than 200m, while the prediction error of linear regression continuously increases, resulting in a deviation of more than 1500m. Intuitively, one possible reason for this phenomenon is that the prediction model only works well when the training data and test data follow the same distributions. When the model operates based on its own predictions, any prediction error, even small, will lead to diverging distributions of input and real data. As a result, both methods fail to capture this initial trend and ultimately the error grows considerably with the prediction step. However, the scalability of LSTM in time series makes it possible to learn user’s complete mobility pattern in his/her movement period (i.e. 360min), thus perform much better than linear regression in multi-step prediction.

C. Further Analysis

Though the aforementioned experimental results have shown the superiority of basic LSTM framework for learning user-specific mobility pattern, there still exist several issues to address for practical applications.

First is the poor generalization ability of the proposed user-specific mobility model. Usually, it is necessary to predict the trajectories of multi-users simultaneously in practical applications. Therefore, we need to train specific prediction models for each user of interest, which is not a sensible approach. On one hand, it incurs large computation overhead. On the other hand, training such a model usually needs a lot of historical data of the user, leading to cold start problem for users with insufficient training data.

Second is the error-accumulation effect for multi-step prediction as shown in Fig. 6. When the position measurements are suspended, the prediction is rapidly unable to follow the actual evolution of trajectory accurately, resulting in negative impacts on practical applications.

V. MULTI-USER MULTI-STEP PREDICTION FRAMEWORK AND SIMULATION RESULTS

Multi-user multi-step prediction promises to bring lots of significant merits. Firstly, it allows for more practical near-real-time resource pre-allocation. But it has to deal with the annoying error-accumulation effect. Secondly, the generalization ability of the prediction model across users also makes it feasible to quickly perform trajectory prediction for any user. Thirdly, the computation overhead of training a model for each user separately can be significantly reduced. Therefore, we consider the real-world user movement scenario and propose a multi-user multi-step trajectory prediction framework. As shown in Fig. 2, it can be observed that though the trajectory comes from different users, most trajectories have similar short-term characteristics following geographical constraints in a small area given a limited trajectory duration (i.e. less than 10 minutes in this paper). This inspires us to focus on the shared movement patterns in a specific area (e.g., 3 to 5 macro base stations) rather than individual movement patterns when making predictions for multiple users. Therefore, we extend the user-specific prediction scheme for individual users to a region-oriented multi-user prediction scheme. Furthermore, in order to decrease the error-accumulation effect for multi-step prediction, we propose a Seq2Seq framework which can decouple the trajectory feature extraction process and the prediction process, thus making the decoder more focus on global information of the input sequence and ignore the local errors.
A. Prediction Framework

We establish a Seq2Seq framework based on the LSTM encoder-decoder architecture to capture the temporal association within the trajectory like speed or direction. All trajectories in the specific area are utilized for the network to acquire the shared short-term mobility patterns caused by geographical constraints. Specifically, the input sequence is the observation trajectory \( \{p_1, p_2, \ldots, p_T\} \) and the output sequence is the prediction of target trajectory \( \{p_{T+1}, p_{T+2}, \ldots, p_{T+K}\} \). Since the target trajectory also contains movement information and potential geographical characteristics, we mix two different methodologies as the final training strategy to make full use of the mobile information contained in the training data: (1) The first case is the same as the auto-regressive model where the inexact output of the previous step is served as the input of the next step. In this way, the decoder can be more focus on global information of the input trajectory and ignore the local prediction errors, thereby enhancing the coordination of the entire network. (2) For the other case, the target sequence shifted one step forward is served as the input of the decoder to learn more movement information and potential geographical characteristics. In order to maximize the prediction performance, we introduce a teacher ratio to balance the two cases.

The multi-user multi-step prediction framework consists of the following two neural networks as shown in Fig. 7.

1) Encoder Neural Network: It consists of one FC input layer with 128 neurons followed by two LSTM layers stacked each with 128 neurons. The input sequence is the given trajectory \( \{p_1, p_2, \ldots, p_T\} \). The input layer is responsible for transforming the 2-dimensional location \( p_i \) into a 128-dimensional feature tensor to capture the complex structure of the trajectory data. The output is then fed into the LSTM stack with two layers. After \( T \) recursive updates in the two LSTM layers, their latest cell states are determined and passed to the decoder.

2) Decoder Neural Network: It consists of one FC input layer with 128 neurons, two stacked LSTM layers each with 128 neurons, and one FC output layer with 2 neurons. The LSTM layers are initialized by the encoder state vectors \( (h_T, c_T) \). The first input of the decoder network is \( p_T \), the last value of the input sequence for the encoder network. For the training process, we use the teacher ratio to control the input of the next steps. Specifically, we generate a random number between 0 and 1. If the random number is larger than teacher ratio, the input of the next \( K - 1 \) steps will be \( \{p_{T+1}, p_{T+2}, \ldots, p_{T+K-1}\} \), the predicted value of the previous steps as shown by the red arrow in Fig. 7. Otherwise, the target output sequence shifted one step forward \( \{p_{T+1}, p_{T+2}, \ldots, p_{T+K-1}\} \) will be the input sequence for the next \( K - 1 \) steps as shown by the blue arrow in Fig. 7. For the test process, the predicted value of the previous step will be used as the input of the next step.

Same as the previous section, we choose MSE as the loss function. The complete training algorithm is presented in Algorithm 2.

Algorithm 2: Seq2Seq-based Multi-User Prediction

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>for e = 1 to Epoch do</td>
</tr>
<tr>
<td>2</td>
<td>Initialize loss ← 0;</td>
</tr>
<tr>
<td>3</td>
<td>Initialize encoder state ← 0;</td>
</tr>
<tr>
<td>4</td>
<td>for i = 1 to T do</td>
</tr>
<tr>
<td>5</td>
<td>Map ( p_i ) to feature tensors by Encoder-FC;</td>
</tr>
<tr>
<td>6</td>
<td>Input the tensors to Encoder-LSTMs;</td>
</tr>
<tr>
<td>7</td>
<td>Update encoder state;</td>
</tr>
<tr>
<td>8</td>
<td>end</td>
</tr>
<tr>
<td>9</td>
<td>Initialize decoder state ← encoder state;</td>
</tr>
<tr>
<td>10</td>
<td>Generate a random number ( n );</td>
</tr>
<tr>
<td>11</td>
<td>for ( i = 1 ) to ( K ) do</td>
</tr>
<tr>
<td>12</td>
<td>if ( n &gt; ) teacher ratio and ( i &gt; 1 ) then</td>
</tr>
<tr>
<td>13</td>
<td>Map ( \hat{p}_{T+i-1} ) to feature tensors by Decoder-input layer;</td>
</tr>
<tr>
<td>14</td>
<td>else</td>
</tr>
<tr>
<td>15</td>
<td>Map ( p_{T+i-1} ) to feature tensors by Decoder-input layer;</td>
</tr>
<tr>
<td>16</td>
<td>end</td>
</tr>
<tr>
<td>17</td>
<td>Input the tensors to Decoder-LSTMs;</td>
</tr>
<tr>
<td>18</td>
<td>Update decoder state;</td>
</tr>
<tr>
<td>19</td>
<td>Map the output of Decoder-LSTMs to ( \hat{p}_{T+i} ) by</td>
</tr>
<tr>
<td>20</td>
<td>Decoder-output layer;</td>
</tr>
<tr>
<td>21</td>
<td>Calculate loss ← ( \frac{\sum_{i=1}^{K}(\hat{p}<em>{T+i}-p</em>{T+i})^2}{K} );</td>
</tr>
<tr>
<td>22</td>
<td>Update model parameters through BPTT;</td>
</tr>
</tbody>
</table>

B. Mobility Prediction Results

In this part, we evaluate the performance of the proposed multi-user multi-step prediction framework on a realistic dataset. For the sake of simplicity, the trajectories are segmented into time series of 15 points with fixed sampling time interval \( \Delta \) where the first 10 points \( \{p_1, p_2, \ldots, p_{10}\} \) are used as the input sequence and the last 5 points are used as the target sequence. We divide the trajectory data by the ratio of 80%, 10% and 10% to generate the training set, validation set, and test set. The training settings are shown in Table II.

To prove the superiority of the proposed LSTM-based Seq2Seq prediction framework, we compare it with several baseline methods that are widely used in time series prediction including:

- **Linear Regression** [8]. Linear Regression is a conventional machine learning algorithm to discover linear relationships among data for regression problems.
- **Support Vector Regression (SVR)** [9]. SVR is another conventional machine learning algorithm which can cope with non-linear problem based on kernel method.
- **LSTM** [16]. LSTM is one of the recurrent neural networks with gate control mechanism and has shown its superiority in encoding long-term dependencies.
- **GRU** [17]. GRU is a simplified version of LSTM with only reset gate and update gate, which has less compu-
The geographic distance error of each step for different methods at \( \Delta = 30 \text{s} \) is presented in Fig. 9. It can be seen that the LSTM-based Seq2Seq achieves the best overall performance and all Seq2Seq frameworks exhibit apparently better performance than conventional regression methods (i.e., Linear Regression and SVR) and the basic RNNs (i.e., LSTM and GRU). Specifically, SVR achieves slightly better performance than Linear Regression since it can deal with the nonlinearities of the trajectory data. From basic RNN frameworks and Seq2Seq frameworks, it can be seen that LSTM and GRU achieve comparable performance while the LSTM is slightly superior. Moreover, the attention mechanism doesn’t show obvious superiority over the normal Seq2Seq framework. One possible reason is that unlike machine translation problem where there exists mismatch in the order of words between the input sentence and the output sentence, the trajectory is a time series of positions with an almost left-to-right sequential relationship, especially for the short-term trajectories with fine granularity in our problem. In this case, the global information such as the velocity, direction, and etc., is more important for trajectory prediction.

The performance comparison among different algorithms in terms of MSE is presented in Fig. 9. It can be seen that the LSTM-based Seq2Seq achieves the best overall performance and all Seq2Seq frameworks exhibit apparently better performance than conventional regression methods (i.e., Linear Regression and SVR) and the basic RNNs (i.e., LSTM and GRU). Specifically, SVR achieves slightly better performance than Linear Regression since it can deal with the nonlinearities of the trajectory data. From basic RNN frameworks and Seq2Seq frameworks, it can be seen that LSTM and GRU achieve comparable performance while the LSTM is slightly superior. Moreover, the attention mechanism doesn’t show obvious superiority over the normal Seq2Seq framework. One possible reason is that unlike machine translation problem where there exists mismatch in the order of words between the input sentence and the output sentence, the trajectory is a time series of positions with an almost left-to-right sequential relationship, especially for the short-term trajectories with fine granularity in our problem. In this case, the global information such as the velocity, direction, and etc., is more important for trajectory prediction.

The geographic distance error of each step for different methods at \( \Delta = 30 \text{s} \) is presented in Fig. 9. It can be seen that the LSTM-based Seq2Seq achieves the best overall performance and all Seq2Seq frameworks exhibit apparently better performance than conventional regression methods (i.e., Linear Regression and SVR) and the basic RNNs (i.e., LSTM and GRU). Specifically, SVR achieves slightly better performance than Linear Regression since it can deal with the nonlinearities of the trajectory data. From basic RNN frameworks and Seq2Seq frameworks, it can be seen that LSTM and GRU achieve comparable performance while the LSTM is slightly superior. Moreover, the attention mechanism doesn’t show obvious superiority over the normal Seq2Seq framework. One possible reason is that unlike machine translation problem where there exists mismatch in the order of words between the input sentence and the output sentence, the trajectory is a time series of positions with an almost left-to-right sequential relationship, especially for the short-term trajectories with fine granularity in our problem. In this case, the global information such as the velocity, direction, and etc., is more important for trajectory prediction.

The geographic distance error of each step for different methods at \( \Delta = 30 \text{s} \) is presented in Fig. 9. It can be seen that the LSTM-based Seq2Seq achieves the best overall performance and all Seq2Seq frameworks exhibit apparently better performance than conventional regression methods (i.e., Linear Regression and SVR) and the basic RNNs (i.e., LSTM and GRU). Specifically, SVR achieves slightly better performance than Linear Regression since it can deal with the nonlinearities of the trajectory data. From basic RNN frameworks and Seq2Seq frameworks, it can be seen that LSTM and GRU achieve comparable performance while the LSTM is slightly superior. Moreover, the attention mechanism doesn’t show obvious superiority over the normal Seq2Seq framework. One possible reason is that unlike machine translation problem where there exists mismatch in the order of words between the input sentence and the output sentence, the trajectory is a time series of positions with an almost left-to-right sequential relationship, especially for the short-term trajectories with fine granularity in our problem. In this case, the global information such as the velocity, direction, and etc., is more important for trajectory prediction.

The geographic distance error of each step for different methods at \( \Delta = 30 \text{s} \) is presented in Fig. 9. It can be seen that the LSTM-based Seq2Seq achieves the best overall performance and all Seq2Seq frameworks exhibit apparently better performance than conventional regression methods (i.e., Linear Regression and SVR) and the basic RNNs (i.e., LSTM and GRU). Specifically, SVR achieves slightly better performance than Linear Regression since it can deal with the nonlinearities of the trajectory data. From basic RNN frameworks and Seq2Seq frameworks, it can be seen that LSTM and GRU achieve comparable performance while the LSTM is slightly superior. Moreover, the attention mechanism doesn’t show obvious superiority over the normal Seq2Seq framework. One possible reason is that unlike machine translation problem where there exists mismatch in the order of words between the input sentence and the output sentence, the trajectory is a time series of positions with an almost left-to-right sequential relationship, especially for the short-term trajectories with fine granularity in our problem. In this case, the global information such as the velocity, direction, and etc., is more important for trajectory prediction.

The geographic distance error of each step for different methods at \( \Delta = 30 \text{s} \) is presented in Fig. 9. It can be seen that the LSTM-based Seq2Seq achieves the best overall performance and all Seq2Seq frameworks exhibit apparently better performance than conventional regression methods (i.e., Linear Regression and SVR) and the basic RNNs (i.e., LSTM and GRU). Specifically, SVR achieves slightly better performance than Linear Regression since it can deal with the nonlinearities of the trajectory data. From basic RNN frameworks and Seq2Seq frameworks, it can be seen that LSTM and GRU achieve comparable performance while the LSTM is slightly superior. Moreover, the attention mechanism doesn’t show obvious superiority over the normal Seq2Seq framework. One possible reason is that unlike machine translation problem where there exists mismatch in the order of words between the input sentence and the output sentence, the trajectory is a time series of positions with an almost left-to-right sequential relationship, especially for the short-term trajectories with fine granularity in our problem. In this case, the global information such as the velocity, direction, and etc., is more important for trajectory prediction.

The geographic distance error of each step for different methods at \( \Delta = 30 \text{s} \) is presented in Fig. 9. It can be seen that the LSTM-based Seq2Seq achieves the best overall performance and all Seq2Seq frameworks exhibit apparently better performance than conventional regression methods (i.e., Linear Regression and SVR) and the basic RNNs (i.e., LSTM and GRU). Specifically, SVR achieves slightly better performance than Linear Regression since it can deal with the nonlinearities of the trajectory data. From basic RNN frameworks and Seq2Seq frameworks, it can be seen that LSTM and GRU achieve comparable performance while the LSTM is slightly superior. Moreover, the attention mechanism doesn’t show obvious superiority over the normal Seq2Seq framework. One possible reason is that unlike machine translation problem where there exists mismatch in the order of words between the input sentence and the output sentence, the trajectory is a time series of positions with an almost left-to-right sequential relationship, especially for the short-term trajectories with fine granularity in our problem. In this case, the global information such as the velocity, direction, and etc., is more important for trajectory prediction.

The geographic distance error of each step for different methods at \( \Delta = 30 \text{s} \) is presented in Fig. 9. It can be seen that the LSTM-based Seq2Seq achieves the best overall performance and all Seq2Seq frameworks exhibit apparently better performance than conventional regression methods (i.e., Linear Regression and SVR) and the basic RNNs (i.e., LSTM and GRU). Specifically, SVR achieves slightly better performance than Linear Regression since it can deal with the nonlinearities of the trajectory data. From basic RNN frameworks and Seq2Seq frameworks, it can be seen that LSTM and GRU achieve comparable performance while the LSTM is slightly superior. Moreover, the attention mechanism doesn’t show obvious superiority over the normal Seq2Seq framework. One possible reason is that unlike machine translation problem where there exists mismatch in the order of words between the input sentence and the output sentence, the trajectory is a time series of positions with an almost left-to-right sequential relationship, especially for the short-term trajectories with fine granularity in our problem. In this case, the global information such as the velocity, direction, and etc., is more important for trajectory prediction.
from Step-1 to Step-5 and their average for SVR, LSTM, and Seq2Seq-LSTM, which are the representative of their own kinds, respectively. It can be seen that the Seq2Seq framework yields superior overall performance except for a slight weakness at the first step compared with the other two kinds of methods.

For the sake of a comprehensive evaluation of different types of algorithms, we also compare the prediction performance under various values of sampling time interval $\Delta$, including 5s, 10s, 15s, 20s, 25s, and 30s. The results in Fig. 11 shows the increasing distance error for all methods as the sampling time interval increases. This is consistent with our intuition since longer sampling time interval inevitably leads to larger distance between every two points, and the prediction error increases accordingly. Fig. 11 also demonstrates that the Seq2Seq framework yields the best performance throughout all sampling time intervals. Fig. 11(b)–Fig. 11(f) further present the geographic distance error from the first prediction step to the last prediction step. The similar conclusion with Fig. 10 can be drawn that except a slightly higher distance error (i.e. 0–2m) at the first time-step, the Seq2Seq framework outperforms the other two kinds of methods at subsequent prediction steps, especially for Step-3 to Step-5, by around 15m–20m. This can be explained that the Seq2Seq framework decouples trajectory feature extraction and prediction process, making the decoder more focus on global information of the input sequence and ignore the local errors, thereby enhancing the coherence and integrity of the entire network.

Finally, in order to highlight the advantage of the multi-user prediction scheme over the user-specific prediction scheme, we take an extensive experiment to compare their performance on a single user. Considering the fact that it is more likely to train a better model with more training data, we train the user-specific model for the user with the largest number of trajectories. Then we evaluate the prediction performance of different models on this user’s trajectory dataset. As shown in Fig. 12, the multi-user prediction models have a significant performance improvement compared with all single-user prediction models and the Seq2Seq multi-user framework achieves the best performance. This illustrates that 1) it is difficult to effectively learn the user’s movement pattern through incomplete trajectories lasting short duration collected in the real world; 2) the short-term trajectory prediction depends more on the sequential and geographic information contained in the trajectory rather than the user it belongs to; 3) a large number of data can better help exploit and reveal the relationship between the regional geographic characteristics and trajectories. Moreover, it can be noted that, among the single-user models, the regression model outperforms the Seq2Seq-LSTM framework because of the latter suffers a lot from underfitting caused by the lack of training data. However, this situation is absent in the multi-user models because of enough training data.

### C. Complexity Analysis and Discussion

In order to investigate the computational efficiency of different methods, we present their training time and test time in the case of $\Delta = 30s$ in Table IV. The training process of conventional machine learning algorithms like Linear Regression and SVR are performed on Intel Core i5 CPU, while the deep learning methods are all performed on GPU RTX2080. The test process of all methods is performed on the Intel Core i5 CPU. Obviously, the Linear Regression shows the highest training efficiency compared with the neural networks, especially those incorporated with the Seq2Seq technique. It is reasonable since a complex model with large number of parameters usually needs more training time to find the optimal solution. Although Seq2Seq framework has relatively larger training time, its test time is much shorter at around 0.27 seconds, which is acceptable for online prediction.

Therefore, we argue that the proposed Seq2Seq multi-user prediction framework proves its outstanding generalization ability for multi-user trajectory prediction as well as the superiority to mitigate the error-accumulation effect for multi-step prediction. However, its training efficiency is relatively lower than other methods, which can be eliminated by offline training. Among the Seq2Seq frameworks, the Seq2Seq-LSTM exhibit a slightly better performance but a little bit lower efficiency than the Seq2Seq-GRU. Therefore, the choice between Seq2Seq-LSTM or Seq2Seq-GRU can be determined based on the practical situation.
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2929430, IEEE Access.

Fig. 11. Comparisons of the average multi-step prediction performance as well as each step prediction performance for different methods versus sampling time interval $\Delta$.

Fig. 12. Comparison of the average multi-step distance error for the region-oriented multi-user prediction scheme and user-specific prediction scheme versus sampling time interval $\Delta$.

### Table IV

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Train (s)</th>
<th>Test (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Regression</td>
<td>0.01795</td>
<td>0.00199</td>
</tr>
<tr>
<td>SVR</td>
<td>84.77774</td>
<td>1.13208</td>
</tr>
<tr>
<td>LSTM</td>
<td>127.8675</td>
<td>0.24473</td>
</tr>
<tr>
<td>GRU</td>
<td>118.1579</td>
<td>0.20003</td>
</tr>
<tr>
<td>Seq2Seq-LSTM</td>
<td>170.4952</td>
<td>0.27762</td>
</tr>
<tr>
<td>Seq2Seq-GRU</td>
<td>164.3108</td>
<td>0.21221</td>
</tr>
<tr>
<td>Seq2Seq-Attention</td>
<td>207.0887</td>
<td>0.28294</td>
</tr>
</tbody>
</table>

### VI. Conclusion and Future Works

In this paper, we investigate the significance of trajectory prediction and explore feasible approaches from both the single-user perspective and multi-user perspective. For single-user trajectory prediction, we propose a basic LSTM framework and experimental results on a model-based mobility dataset illustrate the superiority of LSTM to make predictions based on pre-learning of user-specific mobility patterns. For multi-user multi-step prediction, we further propose a region-oriented prediction scheme and put forward an LSTM-based Seq2Seq framework. Experiments on a realistic dataset show that the proposed framework outperforms the other competing approaches, which demonstrate its outstanding generalization ability for multi-user prediction as well as robustness and stability for multi-step prediction.

Our current work does not consider the semantic context in the trajectory like the point of interests [37] because of the limitation of data. For future work, we plan to combine our algorithm with some semantic information to improve the prediction performance.

### Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61701439, 61731002), Zhejiang Key Research and Development Plan (No. 2019C01002), the Fundamental Research Funds for the Central Universities (2019QNA5010).

### References


