Reconstruction of Missing Trajectory Data: A Deep Learning Approach

Ziwei Wang, Shiyao Zhang, and James J.Q. Yu, *Member, IEEE*

***Abstract*— GPS trajectory data have become increasingly useful in traffic analysis and optimization. Nevertheless, due to sampling and communication-related issue, such trajectories suffer from data missing problems, and they further render a low quality of raw data for subsequent research. To address this problem, in this work, we propose a recurrent neural network based encoder-decoder deep learning approach. The head-direction information of trajectory, defined by the radius of curvature, is utilized together with the displacement attributed by an attention mechanism to learn from past trajectory points with different priority. Additionally, a smoothing data post-processor is adopted to make the reconstructed trajectories authentic. To evaluate the performance of the proposed reconstruction approach, a series of comprehensive case studies are conducted, which indicates that the proposed approach significantly outperforms baselines, such as the reduction of the missing impact to the original data and improvement in the prediction accuracy.**

I. Introduction

In the era of big data, with the development of mobile Internet and the popularity of smart phones, a massive volume of trajectory data of moving objects is generated by containing abundant spatio-temporal characteristics. Properly analyzing such data is of utmost importance in the modern smart city development. For example, trajectories processing in the urban transportation can provide great insights into optimizing traffic routes, such as personalized recommendation routes, road network prediction, urban planning, etc [1]–[3].

However, during the process of gathering such trajectories, the trajectory data may be potentially affected by uncertainties such as stochastic communication loss and positioning error, render missed data points. Such data loss introduces noises to the source of subsequent trajectory-related traffic and transportation services. In addition, trajectory data has the characteristic of great randomness, e.g., the life habits of users vary greatly and the traffic situation changes rapidly. These two cases raise the problem of direction rectification and missing location prediction.

In recent years, deep learning has become prevalent in Intelligent Transportation Systems (ITS) research, which is bolstered by the recent significant advance of storage and computing technologies. Transportation system, known as a special and complex system, consists of the multiple factors, including personal behaviours and vehicle types, time and weather, accidents and public construction, etc. Compared to traditional problem solving methods, deep learning techniques take advantage of better describing the randomness and variety of transportation, see [4]–[6] for examples.

This work is supported in part by the General Program of Guangdong Basic and Applied Basic Research Foundation No. 2019A1515011032 and in part by Guangdong Provincial Key Laboratory No. 2020B121201001. *(Corresponding author: James J.Q. Yu.)*

The authors are with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology.

A. Prior Art and Comparisons

There are two important steps in the trajectory data processing, i.e., data pre-processing and trajectory similarity measurement. Trajectory data pre-processing indeed includes noise filtering and trajectory segmentation. Generally speaking, there are three ways to filter noise points: mean filter [7], Kalman filter [7], and particle filter [8]. Several main strategies for trajectory segmentation are based on the time threshold, geometric topology and trajectory semantics [9] [10].

Determining the similarity among trajectories is a critical research problem to trajectory analysis applications such as trajectory clustering and relationship mining. For a pair of trajectories with the same length, the basic evaluating metrics are Mean Square Error (MSE) and Mean Absolute Error (MAE). Furthermore, Dynamic time warping (DTW), Longest Common Subsequence (LCSS) [11], and Edit Distance Real Sequence (EDR) [12] are the robust and precise techniques that can work with asymmetric sequence. However, the issue of high computational complexity occurs with the use of large datasets that cover thousands or even millions of trajectories. Therefore, it is worth exploring a new method to develop trajectory similarity scores with less computational burden.

At present, modeling methods can be roughly split into three categories: statistical, machine learning (non-deep learning), and deep learning methods. The majority of research effort in the past literature focuses on statistics-based models for prediction, such as Hidden Markov Model (HMM) [13] and tensor completion algorithm [14]. As for machine-learning methods, clustering is the most prevalent for missing spatio-temporal data completion [15]–[17]. In the meantime, a number of recent work shows that deep neural networks are able to recover the missing traffic data for prediction. Among the proposed approaches, Convolutional Neural Networks (CNNs) are widely adopted, whose architecture, however, requires that all input trajectories must be of the same length. For example, Lv et al. [18] used CNNs for taxi trajectory prediction and fused additional time and driver information in a fully-connected layer so as to further improve the system performance. Besides, generative adversarial network (GAN) is another widely-recognized deep learning technique in the context. For instance, Li et al. used a fractionally strided 3D convolutional neural network to construct the generator network and a 3D convolutional neural network as the discriminator network for missing

traffic data completion in [19]. However, these network structures require fix-length trajectory as input, which ignores the information contained in other parts of a trajectory.

To overcome the fixed-length trajectory issue in the above CNN-oriented research, recurrent neural networks (RNNs) are utilized due to their capability of handling data at variable lengths. Şahin [20] studied the regression of variable length missing sequential data and introduced a long short-term memory (LSTM) based algorithm for time-series data reconstruction. Liang et al. [21] introduced a random forest approach to identify the missing trajectory points first, and then applied LSTM as the second step for trajectory reconstruction. Wang et al. [22] proposed a hybrid model that integrates MLP to extract local features and LSTM to capture long term dependency, and then random forest for further prediction error minimization. However, in the previous studies, a single RNN structure is only deployed for the part of trajectory before prediction point. Hence, it is reasonable to apply a new method which uses the information of the whole trajectory for missing value prediction.

B. Contributions

To fill the research gaps shown above, the main contributions of this work is as follows.

- In this paper, we deploy recurrent sequence-to-sequence neural network to handle the missing trajectory recovery and demonstrate the superiority of the proposed approach.
- We propose a method of trajectory slice embedding with an iterative random matrix. The information of radius of curvature is included in this step for reconstruction robustness, which is novel as far as we are concerned.
- We propose a new trajectory similarity comparison function, which has a unified and finite scope. The function serves as the loss function of the proposed model and is easy to implement and fast to compute.

The remainder of this paper is organized as follows. Section II introduces the definition of missing trajectory data recovery problem and preliminaries. Section III presents the proposed model for data recovery. Section IV gives the results and analyses on a comprehensive set of case studies based on a real world dataset. Finally, this work is concluded in Section V.

II. Problem Formulation and Preliminaries

In this section, we formulate the missing trajectory recovery problem and present the pre-processing stages required in the subsequent section.

A. Missing Trajectory Data Recovery

A trajectory $\mathbf{L}$ is a sequence of timestamp locations. For each time t, point $\mathbf{L}_t = (x_t, y_t)$ represents the coordinates with longitude x_t and latitude y_t, $1 \leq t \leq n$. We use $\mathbf{L}$ to denote the ground truth trajectory, $\tilde{\mathbf{L}}$ to denote the sampled one corresponds to $\mathbf{L}$ with missing coordinates, and $\hat{\mathbf{L}}$ to denote an estimated one to resemble $\mathbf{L}$. We use I to mark the missing locations. An indicator series I produced by a characteristic function takes value of 0 or 1 and has the same length n with $\mathbf{L}$. The characteristic function is defined as follows: for time t, $I_t = 1$ if the t-th coordinate, i.e., $\mathbf{L}_t$ is missing, otherwise $I_t = 0$. The missing set $\mathcal{O} = \{\mathbf{L}_t | I_t = 1\}$ is the set of all missing point in $\mathbf{L}$. If $\mathbf{L}_t \in O$, then $I_t = 1$, which means $\mathbf{L}_t$ is missing. We aim to construct a neural network which can recover the missing data in the trajectories.

B. Encoder-Decoder Structure and Attention Mechanism

Encoder-Decoder structure is a type of deep learning architectures consist of two sub-neural networks called encoder and decoder [23]. The encoder reads an input data sequence and outputs an intermediate latent information tensor, and the decoder translate the tensor and produce an output sequence. The intermediate tensor that passes the input data characteristic information from encoder to decoder is typically called context vector, denoted by $\mathbf{c}$.

Attention mechanism is widely used in this structure due to the two major limitations of context vector decoding [24]. On the one hand, when the input sequence is excessively long, it is hard for the context vector to express the complete input information. In such cases, the information at the end of the sequence are more concentrated, rendering inferior model capacity. On the other hand, predicting target data sequence may put different emphases on various positions at the source sequence. Attention mechanism is thus introduced to deal with these issues. The fixed context $\mathbf{c}$ is replaced by $\mathbf{c}_t$, which is dynamically computed according to the current output to adjust the model.

Let the input and output sequences' length be n and m. Given a source sequence $\mathbf{x} = (x_1, \ldots, x_n)^\mathsf{T}$, $\mathbf{x}$ is encoded as $\mathbf{h} = (\mathbf{h}_1, \ldots, \mathbf{h}_n)^\mathsf{T}$by the encoder. When decoding the target sequence $\mathbf{y} = (y_1, \ldots, y_m)^\mathsf{T}$, a hidden vector $\mathbf{H} = (\mathbf{H}_1, \ldots, \mathbf{H}_m)^\mathsf{T}$ is created successively. Instead of directly using $\mathbf{h}$ as the context vector for decoding, an additional attention weight is adopted for context transformation by $\mathbf{c}_t = \mathbf{a}_t^\mathsf{T}\mathbf{h}$, where the attention $\mathbf{a}_t = (a_{1,t}, a_{2,t}, \cdots, a_{n,t})$ and $a_{i,t} = \exp(f(\mathbf{H}_{t-1}, \mathbf{h}_i)) / \sum_{j=1}^{n} \exp(f(\mathbf{H}_{t-1}, \mathbf{h}_j))$.

C. Curvature Radius

At a given point on a curve, let r be the radius of the osculating circle. If the curve is written in the form $y = f(x)$, the radius of curvature is given by

$$R(x) = \left[1 + f'(x)^2\right]^{3/2} / \left|f''(x)\right|. \tag{1}$$

For trajectory $\mathbf{L}$, we calculate the radius of curvature $\mathbf{R}$ with respect to the longitude and latitude values of each location. The two sequence are first fit with polynomials of order $p = 40$, denoted by f_x and f_y . Then, we use (1) to compute $\mathbf{R}_t = (f_x(x_t), f_y(y_t))$ in terms of the fitted curve. Intuitively, $\mathbf{R}$ has the same shape as $\mathbf{L}$, i.e. $[n, 2]$.

The idea of adding curvature information to the process is due to the fact that the major difficulty lies in the inflection point. If a starting point is given without any restriction to the result, the generated trajectory is usually overly smooth. We hypothesize that the inclusion of this additional information

can improve the model performance, which is supported by empirical studies presented in Section IV.

III. Proposed Model

In this section, we propose a new deep learning based technique to impute the missing data within trajectories.We first give the structure of the neural network and define a new similarity measurement criterion. Then, we provide the training details and the subsequent processing method.

A. Encoder Network

Gated Recurrent Unit (GRU) has the capability on handling time-series dataset, which is advantageous in capturing the long-term dependency. We use GRU as the encoder network. Let h be the hidden size of GRU and the initial hidden state $\mathbf{h}_0$ is zero. For each time t, a coordinate $\mathbf{L}_t = (x_t, y_t)$ is sent to the encoder and processed via the following two steps:

1) Embedding.

$$\mathbf{L}'_{et} = \tanh(\mathbf{W}_{ee}\mathbf{L}_t + \mathbf{b}_{ee} + \mathbf{W}_{er}\mathbf{R}_t + \mathbf{b}_{er}). \quad (2)$$

2) GRU.

$$r_t = \sigma(\mathbf{W}_{el}\mathbf{L}'_t + \mathbf{b}_{el} + \mathbf{W}_{eh}\mathbf{h}_{t-1} + \mathbf{b}_h), \quad (3a)$$

$$z_t = \sigma(\mathbf{W}_{el}\mathbf{L}'_t + \mathbf{b}_{el} + \mathbf{W}_{eh}\mathbf{h}_{t-1} + \mathbf{b}_{eh}), \quad (3b)$$

$$\tilde{\mathbf{h}_t} = \tanh(\mathbf{W}_{el}\mathbf{L}'_\mathbf{t} + \mathbf{b}_{el} + r_t \otimes (\mathbf{W}_{eh}\mathbf{h}_{t-1} + \mathbf{b}_{eh})), \quad (3c)$$

$$\mathbf{h}_t = (1 - z_t) \otimes \mathbf{h}_{t-1} + z_t \otimes \tilde{\mathbf{h}_t}, \quad (3d)$$

where $\mathbf{W}_{ee}, \mathbf{W}_{er} \in \mathbb{R}^{h\times 2}$, $\mathbf{W}_{el}, \mathbf{W}_{eh} \in \mathbb{R}^{h\times h}$ and $\mathbf{b}_{ee}$, $\mathbf{b}_{er}, \mathbf{b}_{el}, \mathbf{b}_{eh} \in \mathbb{R}^{h\times 1}$ are the neural network parameters to be learnt by training. In the formula, r_t and z_t are the reset and update gates respectively. $\tilde{\mathbf{h}_t}$ is the intermediate cell state, and $\otimes$ represent the Hadamard product. $\mathbf{h}_t$ is the hidden state vector at time step t. The current coordinates with its curvature are combined through the fully connected layer for embedding. It extracts the information of trajectory points from 2-dimensional coordinates to an h-dimensional latent space. Subsequently, a GRU layer is employed to further extract the data characteristics from raw trajectories for later reconstruction.

B. Decoder Network

The previous encoder network generates hidden states for every time step, $\mathbf{h} = (\mathbf{h}_1, \ldots, \mathbf{h}_n)$, which contains the whole information of a trajectory. In this work, the decoder is also constructed by GRU but with assistance from the attention mechanism. Given the encoded latent information $\mathbf{h}$, the decoding process first employs (2) to embed the input $\mathbf{L}_t$. Subsequently, the attention mechanism is employed to better extract local relationship among the latent information:

$$\mathbf{w}_t = \mathrm{softmax}(\mathbf{W}_w[\mathbf{L}'_t, \mathbf{H}_{t-1}] + \mathbf{b}_w) \quad (4a)$$

$$\mathbf{a}_t = \mathbf{w}_t^T\mathbf{h} \quad (4b)$$

$$\mathbf{l}_t = \mathrm{relu}(\mathbf{W}_a[\mathbf{L}'_t, \mathbf{a}_t] + \mathbf{b}_a) \quad (4c)$$

After the attention, the decoder re-employes GRU structure to capture the temporal data dependency as previous presented in (3). But the input of decoder GRU change to $\mathbf{l}_t$ and H_{t-1} , then the trainable parameters are different from those in the encoder. Finally, the ouput is projected using two layers of fully connected neurons:

$$\hat{\mathbf{l}}_t = \sigma(\mathbf{W_o}\mathbf{H}_t + \mathbf{b}_o) \quad (5a)$$

$$\hat{\mathbf{L}}_t = \mathbf{W}_m\hat{\mathbf{l}_t} + \mathbf{b}_m \quad (5b)$$

where $\mathbf{W}_w, \mathbf{W}_a \in \mathbb{R}^{2h\times h}$, $\mathbf{W}_o \in \mathbb{R}^{h\times h}$, $\mathbf{W}_m \in \mathbb{R}^{2\times h}$ and $\mathbf{b}_w, \mathbf{b}_a \in \mathbb{R}^{2h\times 1}$, $\mathbf{b}_o \in \mathbb{R}^{h\times 1}$, $\mathbf{b}_m \in \mathbb{R}^{2\times 1}$ are the network parameters. $\mathbf{l}_t$ uses the complete information obtained from encoder since $\mathbf{a}_t$ is the weighted sum of encoder hiddens. $\hat{\mathbf{l}}_t$ is the direct output of decoder GRU, and the final output $\hat{\mathbf{L}}_t$ is accordingly derived by a linear mapping.

C. Loss Function

Instead of using Mean Square Error (MSE) directly as the loss function for training the neural network, we propose a new loss indicator $\mathcal{R}^2$, which divides MSE by the variance of the true trajectory $\mathbf{L}$ to cancel the influence of some outliers and improve the robustness of the result. $\mathcal{R}^2$ is also called decision coefficient, which reflects the proportion of total variation of the dependent variable that can be explained by independent variable in regression analysis. Neural network can be seen as a generalized nonlinear regression composed of multiple linear regression. Accordingly, it is reasonable to measure the accuracy among trajectories by using $\mathcal{R}^2$:

$$\mathcal{R}^2 = 1 - \frac{\mathrm{MSE}(\hat{\mathbf{L}} - \mathbf{L})}{\mathrm{Var}(\mathbf{L})}, \text{ and } \mathrm{Loss}(\mathbf{L}, \hat{\mathbf{L}}) = -\mathcal{R}^2. \quad (6)$$

Specifically, MSE is calculated point-wisely between $\hat{\mathbf{L}}$ and $\mathbf{L}$. $\hat{\mathbf{L}}$ is the estimated trajectory and $\mathbf{L}$ is the ground truth one,respectively. A larger $\mathcal{R}^2 \in [0, 1]$ represents a smaller relative error, rendering a higher accuracy. The extreme cases occurs when $\mathcal{R}^2 = 0$ or 1. $\mathcal{R}^2 = 0$ indicates that it is better to take the mean value without any prediction, while $\mathcal{R}^2 = 1$ indicates that all predictions match the true results perfectly.

D. Training the Neural Network

The whole network trains one trajectory $\mathbf{L}$ randomly at one iteration and uses Adam optimizer [25] of learning rate 0.0001 for back propagation in both encoder and decoder, and gradient clipping is added to avoid gradients explosion [26]. At every time step t, the input of the encoder is $\mathbf{L}_t$ together with the hidden state generated in the previous step $\mathbf{h}_{t-1}$, and the output $\mathbf{h}_t$ is recorded. For the decoder, the input data is constructed considering the data missing situation. For each time step t, if $I_t = 0$, the input of the decoder is $\mathbf{L}_t$, and $\mathbf{H}_{t-1}$ is used as the latent cell information. Otherwise, the input is $\hat{\mathbf{L}}_t$ that was developed in the previous time step $t-1$. The outputs of the decoder is $\hat{\mathbf{L}}_{t+1}$ and $\mathbf{H}_t$. The final output trajectory $\hat{\mathbf{L}}$ is reconstructed as $\{\mathbf{L}_t \otimes (1 - I_t) + \hat{\mathbf{L}}_t \otimes I_t\}$.

E. Smoothing

As will be shown in the case studies, the restored trajectory of the network output is zigzag due to the fact that the restored location has random tiny offsets from the respective real location. For this reason, it is necessary to smooth the

new trajectory so that the outputs can be more authentic. A hybrid method of combining Moving Average (MA) with Savitzky-Golay (SG) filter [27] is proposed in this paper. For an estimated trajectory $\hat{\mathbf{L}}$, we first apply MA over the missing set $\mathcal{O} = \{\mathbf{L}_t | I_t = 1\}$ that are recovered by neural network. After that, SG filter will move through the whole sequence, i.e. every point in the trajectory.

Let the order of MA be $2m+1$, and the length of the trajectory $\mathbf{L}_t$ be n. In this work, $m = 7$. We apply MA on points that are missing, i.e. for all $\mathbf{L}_t \in \mathcal{O}$:

$$\mathbf{L}'_\mathbf{t} = (x'_t, y'_t) = \begin{cases} \left(\frac{\sum_{i=1}^{t+m} x_i}{t+m}, \frac{\sum_{i=1}^{t+m} y_i}{t+m}\right), & 1 \le t < m \\ \left(\frac{\sum_{i=t-m}^{t+m} x_i}{2m+1}, \frac{\sum_{i=t-m}^{t+m} y_i}{2m+1}\right), & m \le t \le n-m \\ \left(\frac{\sum_{i=t-m}^{n} x_i}{n-t-m+1}, \frac{\sum_{i=t-m}^{n} y_i}{n-t-m+1}\right), & n-m < t \le n \end{cases} \quad (7)$$

Subsequently, SG is employed to post-process the recovered trajectory, which is widely adopted in time series de-noising due to its simplicity and efficacy. The principle of SG filter is based on local polynomial least square estimation in time domain. Let the filtering window length be $2w+1$. A k^{th} order polynomial $y = \sum_{i=0}^{k-1} a_i x^i$ is applied on the subsequence $(x_{t-w}, \cdots, x_{t+w})$ for fitting. To make the equation solvable, $2w+1 > k$ must hold. Here we take $w = 4$ and $k = 5$. For t on head and tail, the subsequence will be extended by the nearest value. For example, the sequence is $(x_1, \cdots, x_n)$ and suppose that $w = 2$, then its extension becomes $(x_1, x_1 | x_1, x_2 \cdots, x_n | x_n, x_n)$. Write the $2w+1$ equations by matrix:

$$\mathbf{X} = \begin{bmatrix} 1 & x_{-w} & \cdots & {x_{-w}}^{k-1} \\ 1 & x_{-w+1} & \cdots & {x_{-w+1}}^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_w & \cdots & {x_w}^{k-1} \end{bmatrix} \quad (8a)$$

$$\mathbf{y} = \begin{bmatrix} y_{-w} \\ y_{-w+1} \\ \vdots \\ y_w \end{bmatrix}, \mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{bmatrix}, \mathbf{e} = \begin{bmatrix} e_{-w} \\ e_{-w+1} \\ \vdots \\ e_w \end{bmatrix} \quad (8b)$$

$$\mathbf{y} = \mathbf{X}\mathbf{a} + \mathbf{e} \quad (8c)$$

Therefore, the mean square solution of $\mathbf{a}$ is:

$$\hat{\mathbf{a}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}, \quad (9)$$

and

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{a}} = \mathbf{X}(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y} = \mathbf{B}\mathbf{y}, \quad (10)$$

where $\mathbf{B} = \mathbf{X}(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}$.

IV. Experimental Evaluation

In this section we examine the performance of the proposed model. First we introduce the dataset and the data preprocessing technique. Next, we analysis the data recovery accuracy with different missing rate, make comparisons of the proposed model with three baseline models as well as two traditional methods, and then test the effect of neural network hidden size on the results.

TABLE I
Rate of Missing Data with Different Parameter Configurations

Missing Rate (%)	λ_1 = 1	λ_2 = 5	λ_3 = 10	λ_4 = 15
p_2 = 0.05	6.34	19.71	32.62	41.55
p_3 = 0.10	11.97	32.97	49.25	58.68
p_4 = 0.15	16.92	42.45	59.24	68.02
p_5 = 0.20	21.37	49.60	66.00	73.90

A. Dataset and Simulation Configurations

The GPS trajectory data used in this work was collected by Geolife project from 182 users in a period of over five years [28]. This dataset contains 17621 trajectories with a total distance of 1292951 kilometers and duration of 50176 hours. These trajectories were recorded by different GPS loggers and have a variety of sampling rates. Each of the trajectory is a sequence of time-stamped points, which consists of latitude, longitude, and altitude. We segment the trajectories into time-series of 200 or less GPS points to make 145987 trajectories for case study. The fragments are continuous and of equal intervals $5s$. After segmenting the dataset, we construct the missing data within trajectories according to the following steps:

1) Ensure that $I_1 = 0$, i.e., the starting point of each trajectory L_1 is not missing.
2) At time $t \ge 1$, randomly draw b from $\mathrm{Bernoulli(p)}$ to judge whether the point is missing. Then a random integer k is sampled from the zero-truncated Poisson distribution, i.e. $K \sim \Pr(K = k) = \frac{\lambda^k}{(1-e^{-\lambda})k!}e^{-\lambda}$. It means that from time t, k steps will be continuously missing, i.e. $I_i = 1$, $t \le i \le t+k-1$. We also have a parameter k_{max} to control the maximum continuous missing length. If $k > k_{max}$, set $k = k_{max}$; At last, $I_{t+k} = 0$, which means no missing at time $t+k$.
3) Repeat Step 2 and stop until the array length exceeds the trajectory length.

This process of making missing data is mainly controlled by p and λ. As the expectation of the distribution is hard to derive, we compute this value numerically by randomly generating 100000 I's of length $n = 200$ and present the statistics in Table I. The training set and testing set contain 60000 and 10000 trajectories, respectively, which are randomly selected from the complete dataset without intersection after segmenting. The experiment is modeled with PyTorch, and eight nVidia RTX 2080 Ti GPUs are employed for neural network computing acceleration.

B. Data Recovery Accuracy

The accuracy of the estimated accuracy is among the most important metric in evaluating the efficacy of traffic estimation methods. We first investigate the average $\mathcal{R}^2$ with different missing rate configurations, namely, $p \in \{0.05, 0.10, 0.15, 0.20\}$ and $\lambda \in \{1, 5, 10, 15\}$. The dimension of hidden state h is set to be 128.

The simulation results are presented in Table II. In general, the model develops satisfactory result in all missing modes. $\mathcal{R}^2$ declines as p and λ increase and the decline is more drastic with larger p and λ. The variance also shows some

TABLE II

DATA RECOVERY ACCURACY OF THE PROPOSED APPROACH

	λ = 1		λ = 5		λ = 10		λ = 15	
	Accuracy (%)	Variance	Accuracy (%)	Variance	Accuracy (%)	Variance	Accuracy (%)	Variance
p = 0.05	99.32	0.0002	97.34	0.0026	93.68	0.0079	89.17	0.0178
p = 0.10	98.81	0.0003	96.15	0.0036	91.24	0.0131	87.30	0.0262
p = 0.15	98.44	0.0005	94.60	0.0049	90.70	0.0152	85.42	0.0310
p = 0.20	98.38	0.0008	93.34	0.0070	86.78	0.0210	84.43	0.0326

TABLE III

COMPARISON OF DATA RECOVERY ACCURACY

	p = 0.05, λ = 15		p = 0.10, λ = 15		p = 0.15, λ = 15		p = 0.20, λ = 15	
	Accuracy (%)	Variance	Accuracy (%)	Variance	Accuracy (%)	Variance	Accuracy (%)	Variance
Proposed	89.17	0.0178	87.30	0.0262	85.42	0.0310	84.43	0.0326
GRU	88.16	0.0262	82.11	0.0524	78.78	0.0854	72.56	0.1015
GRU-ED	84.18	0.0337	80.41	0.0621	75.22	0.0856	74.16	0.0572
GRU-Attn	85.04	0.0350	81.79	0.0431	79.99	0.0485	77.85	0.0688
ARIMA	88.64	0.0357	84.00	0.0573	80.87	0.0717	79.28	0.0814
Kalman Filter	88.08	0.0339	82.80	0.0603	79.61	0.0787	78.22	0.0833

regularity, a higher accuracy is usually paired with a smaller variance, indicating a higher reliability. Last but not least, the consecutive missing parameter λ has greater influence to variance than p. Fig. 1 shows an example of the recovered trajectories with $p = 0.10$ and $\lambda = 5$. The entity moves from the green point to the red one following the orange curve, and the black parts represent the missing data. The dash-dot blue curve is the re-constructed trajectory.

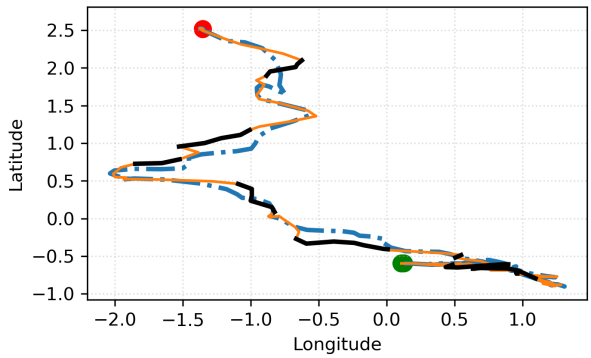

Fig. 1. Examples of trajectory recovery

In addition to the accuracy assessment, we also implete an ablation test of the proposed model with baseline approaches. Specifically, we construct four models of the components from the canonical GRU, the encoder-decoder structure GRU (denoted by GRU-ED), GRU-ED with attention mechanism (GRU-Attn), and finally the proposed model with radius of curvature added for mapping the original coordinates (Proposed). The learning rate and the hidden size are identical to the proposed approach. Additionally, we also compared the proposed approach with two traditional statistic methods, i.e., ARIMA and Kalman Filter (KF) [29]. In the test, AIC [30] is employed to determine the order of $\mathrm{ARIMA}(p, d, q)$ in ARIMA as a time-series analysis algorithm, which takes all combinations of $0 \leq p, d, q \leq 2$ and selects the optimal recovered trajectory as model output. KF adopt the current measured state and the estimated state of the previous sampling period to estimate the realistic state. Due to the space limit, we consider the missing data patterns when the longest consecutive missing happened — $\lambda = 15$ with $p = 0.05$, $p = 0.10$, $p = 0.15$, and $p = 0.20$ — to evaluate the performance of baselines and the proposed approach in handling hard trajectory reconstruction tasks. The simulation results are summarized in Table III.

From this table, it is evident that the proposed approach achieves the best performance in terms of recovery accuracy. Besides, the variance has a significantly decline from any compared approaches. This is because the added curvature radius provides additional information to the learning system, and thereby obtain stable results. Furthermore, the advantage of the proposed approach expands with the portion of missing data among all, indicating the superior data recovery capability of the proposed approach.

The traditional statistic methods are still competitive in comparison with deep learning approaches, the average accuracy of ARIMA is slightly superior than that of KF. But since ARIMA expends a long time duration of selecting the optimal order combination, its computation cost has no much difference between deep learning approaches. Among other deep learning methods, GRU is advantageous when the missing situation is mild but is inferior when p is large. Conversely, GRU-ED, as well as GRU-Attn, are weak when p is small but work better when p is large, which means that the encoder-decoder structure and attention mechanism can better capture the spatial-temporal information and recover the missing locations. This ablation test demonstrates that all major components in the proposed approach contributes to the final data recovery performance.

C. Parameter Test

The dimension of hidden state h in the proposed neural networks determines the size of representation space, which has crucial influence on the system performance. In this section, we examine the effect of h on the performances of the proposed approach by re-training multiple models with $h \in \{32, 64, 128, 256, 512\}$. All other settings are identical to those in the previous section, and the simulation results are presented in Table IV. From the simulation results, it can be observed that the accuracy increases with h for $h \leq 256$ and remains at a high level afterwards. Since the performance improvements for $h \geq 256$ is not as significant

TABLE IV
RESULTS OF PARAMETER SENSITIVITY

	p = 0.05, λ = 1		p = 0.10, λ = 5		p = 0.15, λ = 10		p = 0.20, λ = 15		Training Time
	Accuracy (%)	Variance	Accuracy (%)	Variance	Accuracy (%)	Variance	Accuracy (%)	Variance	(min / 10000 iter)
h = 32	97.96	0.0004	91.79	0.0071	83.85	0.0321	76.05	0.0739	84'06"
h = 64	98.80	0.0002	93.79	0.0050	87.26	0.0243	80.17	0.0444	83'56"
h = 128	99.32	0.0002	96.15	0.0036	90.70	0.0152	84.43	0.0326	83'39"
h = 256	99.42	0.0002	95.86	0.0036	90.93	0.0170	80.10	0.0430	83'44"
h = 512	99.47	0.0001	96.41	0.0031	90.03	0.0180	83.61	0.0476	85'08"

as the proportionally growing training time, $h = 128$ is a best performing network size parameters considering both the data recover accuracy and training time.

V. CONCLUSION

In this paper, we propose a new deep learning approach for missing trajectory reconstruction. We develop an iterative linear mapping between original trajectory attributes and its higher dimensional representation, and then apply a GRU encoder-decoder network with attention mechanism to predict the missing data entries. Besides, we use $\mathcal{R}^2$ as a new training objective in order to improve effectiveness and robustness of the neural network training process. To evaluate the proposed mechanism, we conducted a series of case studies on a practical dataset. The simulation results illustrate the advantage of the proposed approach in both data recovery accuracy and result stability compared with the baseline approaches. Additionally, all major data processing components in the proposed approach contribute to the final recovery performance. Lastly, we investigate the sensitivity of the neural network size.

REFERENCES

[1] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "T-drive: Enhancing driving directions with taxi drivers' intelligence," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 220–232, Jan 2013.

[2] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on GPS data," in *Proc. International Conference on Ubiquitous Computing*, Seoul, Korea, 2008, pp. 312–321.

[3] F. Lu, Y. Duan, and N. Zheng, "A practical route guidance approach based on historical and real-time traffic effects," in *Proc. 17th International Conference on Geoinformatics*, Aug. 2009, pp. 1–6.

[4] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, April 2015.

[5] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li, "Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction," in *AAAI*, 2018.

[6] J. J. Q. Yu and J. Gu, "Real-time traffic speed estimation with graph convolutional generative autoencoder," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3940–3951, Oct 2019.

[7] Y. Zheng, "Trajectory data mining: An overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, May 2015.

[8] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[9] E. Keogh, S. Chu, D. Hart, and M. Pazzani, *Segmenting time series: A survey and novel approach*. World Scientific, 2004.

[10] F. Moscheni, S. Bhattacharjee, and M. Kunt, "Spatio-temporal segmentation based on region merging," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 20, no. 9, pp. 897–915, 1998.

[11] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *International Conference on Data Engineering*, 2002.

[12] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proc. ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2005, p. 491–502.

[13] L. Qu, Y. Zhang, J. Hu, L. Jia, and L. Li, "A bpca based missing value imputing method for traffic flow volume data," in *Proc. IEEE Intelligent Vehicles Symposium*, June 2008, pp. 985–990.

[14] H. Tan, G. Feng, J. Feng, W. Wang, Y. J. Zhang, and F. Li, "A tensor-based method for missing traffic data completion," *Transportation Research Part C Emerging Technologies*, vol. 28, no. 3, pp. 15–27, 2013.

[15] Z. Liu, S. Sharma, and S. Datla, "Imputation of missing traffic data during holiday periods," *Transportation Planning & Technology*, vol. 31, no. 5, pp. 525–544, 2008.

[16] G. Shen, C. Zhang, and B. Tang, "Completing truck's missing trajectory based on the historical information," in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2015.

[17] X. C. Chen, J. H. Faghmous, A. Khandelwal, and V. Kumar, "Clustering dynamic spatio-temporal patterns in the presence of noise and missing data," in *International Conference on Artificial Intelligence*, 2015.

[18] J. Lv, Q. Li, Q. Sun, and X. Wang, "T-conv: A convolutional neural network for multi-scale taxi trajectory prediction," in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jan 2018, pp. 82–89.

[19] Z. Li, H. Zheng, and X. Feng, "3d convolutional generative adversarial networks for missing traffic data completion," in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, Oct 2018, pp. 1–6.

[20] S. O. Şahin, "Sequential regression with missing data using lstm networks," in *2019 27th Signal Processing and Communications Applications Conference (SIU)*, April 2019, pp. 1–4.

[21] M. Liang, R. W. Liu, Q. Zhong, J. Liu, and J. Zhang, "Neural network-based automatic reconstruction of missing vessel trajectory data," in *2019 IEEE 4th International Conference on Big Data Analytics (ICBDA)*, March 2019, pp. 426–430.

[22] Y. Wang, D. Zhang, Y. Liu, and K. Tan, "Trajectory forecasting with neural networks: An empirical evaluation and a new hybrid model," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2019.

[23] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014, `arXiv:1406.1078 [cs.CL]`.

[24] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, `arXiv:1409.0473 [cs.CL]`.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, `arXiv:1412.6980 [cs.LG]`.

[26] J. Zhang, T. He, S. Sra, and A. Jadbabaie, "Why gradient clipping accelerates training: A theoretical justification for adaptivity," in *Proc. International Conference on Learning Representations*, 2020.

[27] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[28] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W. Y. Ma, "Understanding mobility based on gps data," in *Proc. Ubiquitous Computing, 10th International Conference*, Seoul, Korea, 2008.

[29] A. C. Harvey, "Forecasting, structural time series models and the kalman filter," *Cambridge Books*, 1991.

[30] H. Akaike, *A New Look at the Statistical Model Identification*. New York, NY: Springer New York, 1998, pp. 215–222.