Toward Time-Continuous Data Inference in Sparse Urban CrowdSensing

Ziyu Sun, Haoyang Su, Hanqi Sun, En Wang, *Member, IEEE,* Wenbin Liu

***Abstract*—Mobile Crowd Sensing (MCS) is a promising paradigm that leverages mobile users and their smart portable devices to perform various real-world tasks. However, due to budget constraints and the inaccessibility of certain areas, Sparse MCS has emerged as a more practical alternative, collecting data from a limited number of target subareas and utilizing inference algorithms to complete the full sensing map. While existing approaches typically assume a time-discrete setting with data remaining constant within each sensing cycle, this simplification can introduce significant errors, especially when dealing with long cycles, as real-world sensing data often changes continuously. In this paper, we go from fine-grained completion, i.e., the subdivision of sensing cycles into minimal time units, towards a more accurate, time-continuous completion. We first introduce Deep Matrix Factorization (DMF) as a neural network-enabled framework and enhance it with a Recurrent Neural Network (RNN-DMF) to capture temporal correlations in these finer time slices. To further deal with the continuous data, we propose TIME-DMF, which captures temporal information across unequal intervals, enabling time-continuous completion. Additionally, we present the Query-Generate (Q-G) strategy within TIME-DMF to model the infinite states of continuous data. Extensive experiments across five types of sensing tasks demonstrate the effectiveness of our models and the advantages of time-continuous completion.**

***Index Terms*—Mobile CrowdSensing, data inference, fine-grained completion, continuous time.**

I. Introduction

With the evolution of information society and the increasing portability of wireless devices, Mobile CrowdSensing (MCS) [1], [2] has recently emerged as a promising paradigm of data collection. Typically, it recruits a large number of users equipped with mobile devices to collect data from specific sensing areas at particular time. Due to budget constraints and presence of unreachable sensing data, traditional MCS can only collect incomplete or even sparse data in most cases. To this end, a modified paradigm called Sparse MCS [3] is proposed, which introduces inference strategies to complete the full sensing data from the partial observations. Sparse MCS has already shown great advantages in some practical applications, such as the air quality monitoring [4], [5], traffic control [6], [7] and urban sensing [8], [9].

Ziyu Sun, Haoyang Su, Hanqi Sun, En Wang, Wenbin Liu are with the College of Computer Science and Technology, Jilin University, Changchun 130012, China, and also with the Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China (e-mail: sunzy2121@mails.jlu.edu.cn;suhy2121@mails.jlu.edu.cn;sunhq5521@mails.jlu.edu.cn; wangen@jlu.edu.cn; liuwenbin@jlu.edu.cn).

(Ziyu Sun and Haoyang Su contributed equally to this work.) (Corresponding author: Wenbin Liu.)

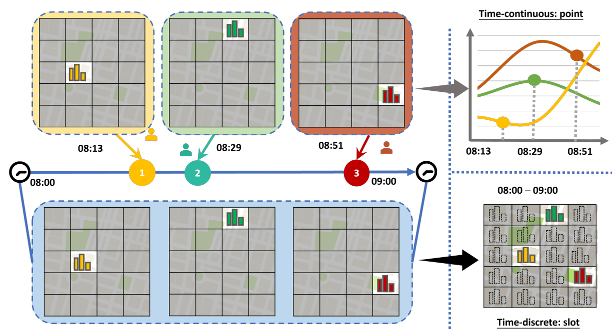


Fig. 1: Time-continuous and time-discrete sensing data in Sparse MCS.

In Sparse MCS, data inference is the most essential part and has therefore received considerable attention. To reduce cost and simplify inference process, most of existing works study the data inference problem from a time-discrete perspective [10]–[15]. For example, in Fig. 1, a requester would like to analyze urban traffic for a period. Upon receiving the task, existing works typically discretize the time period into units and aggregate the sensed data within each unit. Then, by assuming that the sensed data remains constant within each time unit, they use the data inference methods, such as compressive sensing [12], [16] or matrix completion [17], [18] to infer the missing data. However, in practical scenario, the sensing data changes continuously over time. Previous time-discrete approaches may cause significant errors on practical applications that are sensitive to change. For example, temperature or wind speed may fluctuates greatly within a short time due to severe weathers and the rough time-discrete method may fail to capture this dramatic local changes. Therefore, time-continuous data inference has become a crucial issue that urgently needs addressing for Sparse MCS.

In this paper, we adopt a time-continuous perspective, moving away from the traditional method of discretizing time into fixed units. This shift eliminates the assumption that data remains constant within a specific period, making it impossible to aggregate observed data within each time unit to reduce matrix sparsity. Consequently, our first challenge is to handle the *extremely sparse data matrix*. Additionally, in time-continuous scenarios, data is collected in real-time, leading to unequal lengths between sensed data intervals, which affects the relationships between consecutive data points. Thus, our second challenge is to model and utilize these *unequal intervals* effectively, maximizing the temporal information for accurate data inference. Finally, while time-discrete methods

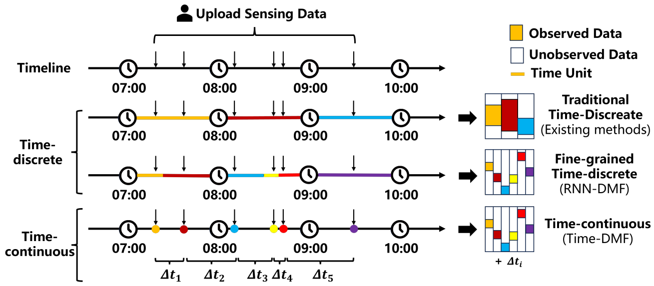


Fig. 2: Time-discrete and time-continuous formulation.

could represent the problem with a fixed-size matrix, they cannot infer the continuously changing data at every moment. This leads to our third challenge: how to *complete the data from a continuous perspective*, ensuring comprehensive data inference across all moments. To tackle the challenges inherent in time-continuous data completion, we introduce a comprehensive approach that begins by reformulating the problem and progresses to designing a method that addresses these challenges effectively. We start with fine-grained completion, which serves as an intermediary between traditional time-discrete methods and our final goal of continuous completion. Unlike traditional time-discrete completion, which uses predefined unit lengths to discretize the timeline, fine-grained inference divides the timeline based on the actual distribution of submissions, ensuring that each time unit contains only one submission. This adjustment eliminates the need to assume data remains constant within each time unit, thereby offering a more accurate model. However, this approach significantly reduces the data volume in each unit, leading to an extremely sparse spatiotemporal matrix (i.e., our first challenge). To address this, we introduce Recurrent Neural Network-enabled Deep Matrix Factorization (RNN-DMF), a neural network-based framework for data completion that also incorporates a temporal encoder to fully leverage previously hidden information, addressing the challenge of sparsity.

Moving beyond fine-grained completion, we recognize that it still does not fully capture the continuous nature of time. In time-continuous scenarios, we avoid discretizing the timeline entirely and handle each submission directly, preserving the precise arrival times and intervals between submissions—our second challenge. To leverage this additional temporal information, we introduce Time Gates-enhanced Deep Matrix Factorization (TIME-DMF), which further captures the temporal dynamics within intervals through the use of time gates and a more sophisticated propagation pattern. Finally, addressing the challenge of representing the infinite number of moments within a period, we propose the Query-Generate (Q-G) strategy, which works in conjunction with TIME-DMF to model any moment on the timeline, thus providing a comprehensive solution to time-continuous data completion.

Our work has the following contributions:

- We reformulate the problem of data inference from a time-continuous perspective. It pays attention to the continuity of data changes and is a much closer approximation of practical MCS problems.
- We introduce DMF which is neural network- enabled framework for data completion and extend it to RNN-DMF with a temporal encoder to handle the extreme matrix sparsity in fine-grained completion.
- We propose TIME-DMF based on RNN-DMF with time gates to capture temporal information within intervals and its accompanying Q-G strategy which allows users to make queries and dynamically generate responses to achieve time-continuous completion.
- Extensive experiments of five types are conducted step by step to validate the effectiveness of our methods.

The reminder of this paper is organized as follows. Section II reviews related works. Section III presents the system model and the problem formulation. In Section IV, we introduce the fine-grained completion, which comprises the DMF framework and RNN-DMF model. In Section V, TIME-DMF and its accompanying Q-G strategy for time-continuous completion are discussed. We evaluate the performance of our approaches through extensive experiments in Section VI, followed by the conclusion in Section VII.

II. Related Work

A. *Sparse MCS*

Mobile CrowdSensing [1], [19] is an emerging paradigm that leverages mobile device users to collect data, enabling a wide range of services within the Internet of Things ecosystem [20]–[22]. MCS has been widely used in domains such as traffic supervision [16], [23], pollution control [24], [25], and facility management [26], [27]. Initially, the mainstream algorithms for MCS were based on compressed sensing [16] and its various adaptations [12]. However, as these algorithms were implemented, it became apparent that data collection often exhibited sparsity [10] due to cost constraints and limitations of sensing devices. Consequently, algorithms used for inferring missing data [28] gained popularity. In 2016, Wang *et al.* [3] provided a comprehensive review of MCS methods based on sparse sensed data and systematically introduced frameworks [11], [13] for data collection and completion. Since then, Sparse MCS has emerged as an evolving paradigm, with many innovative algorithms being developed. Primary areas of work in this field include cell selection [29], data inference [17], [18] and user privacy protection [30].

In data inference, methods are generally categorized into two classes: dense-supervised [31], [32] and sparse-supervised [15], [17], [18]. Dense-supervised methods rely on large amounts of complete spatiotemporal data for training. Most Transformer models and their variants, which are powerful in handling time series, fall under this category. In contrast, sparse-supervised methods do not require complete spatiotemporal data for training and instead rely on capturing correlations within sparsely observed data. Despite their differences, neither approach considers the continuous nature of time.

B. *Spatiotemporal Granularity*

The goal of sensing technology is to capture more fine-grained spatiotemporal information. Initially, this was achieved by deploying additional sensors [33] to decrease granularity and improve accuracy at an expensive price. Subsequently, data inference algorithms [34] emerged as a more cost-effective alternative, finding widespread application in various domains such as air monitoring [35], [36].

In terms of time granularity, there is comparatively less research on fine-grained or even continuous timeline. Doya *et al.* [37] investigated reinforcement learning in continuous spatiotemporal domains, while Brockwell *et al.* [38] explored improvements to the ARMA algorithm on a continuous timeline. Higuchi *et al.* [39] and Kidger *et al.* [40] studied cases where events are unevenly distributed over time. In practical applications, Zhu *et al.* [41] examined the impact of non-uniform distribution of customer consumption from a time-continuous perspective in recommendation systems, introducing time gates into deep learning models for handling unequal time intervals between events. Drawing from these insights, in this paper, we conduct a comprehensive analysis on reducing the temporal granularity of sensing tasks and aim to achieve time-continuous completion in Sparse MCS.

III. System Model and Problem Formulation

A. System Model

Fig. 1 and Fig. 2 showcase different methods for formulating real-world spatiotemporal data. The majority of existing research has relied on the time-discrete approach depicted in the lower half of Fig. 1, which serves as a coarse-grained approximation of the real world. In this section, we will begin with the time-discrete formulation and subsequently introduce our novel time-continuous approach to provide a deeper understanding of our concept.

In Sparse MCS tasks, our sensing map is divided into N sub-regions. At moment t, a worker submits data of the $n-th$ sub-region. We use a position vector $c_{(t)} \in R^N$ and a data vector $y'_{(t)} \in R^N$ to represent the submission. The position vector $c_{(t)} = [0, 0, \dots, 1, 0 \dots 0]$ indicates the index of the subarea submitted by the worker. If the submitted data is from the $i-$th sub-region, the $i-$th element of $c_{(t)}$ is set to 1, and all other elements are set to 0. The data vector $y'_{(t)}$ indicates the value of the submitted subarea. The $i-$th element of $y'_{(i)}$ is the submitted value, and other elements of $y'_{(i)}$ are set to meaningless values like 0 or negative numbers.

Assuming that there are M submissions on the entire timeline, the results of M submissions are organized into position matrix and data matrix, we have $C \in R^{N\times M}$and $Y' \in R^{N\times M}$ by stacking all the submissions together:

$$C = [c_1^{\mathrm{T}}, c_2^{\mathrm{T}}, \cdots, c_M^{\mathrm{T}}], \tag{1}$$

$$Y' = [y_1'^{\mathrm{T}}, y_2'^{\mathrm{T}}, \cdots, y_M'^{\mathrm{T}}]. \tag{2}$$

Similarly, $Y = [y_1^{\mathrm{T}}, y_2^{\mathrm{T}}, \cdots, y_M^{\mathrm{T}}]$ represents the real values of each sub-region at each submission time. So we have:

$$Y' = Y \odot C, \tag{3}$$

where $\odot$ represents the Hadamard product.

In traditional Sparse MCS tasks, the timeline is evenly discretized into time units of predefined length, and we no more distinguish the difference of arriving time of submissions

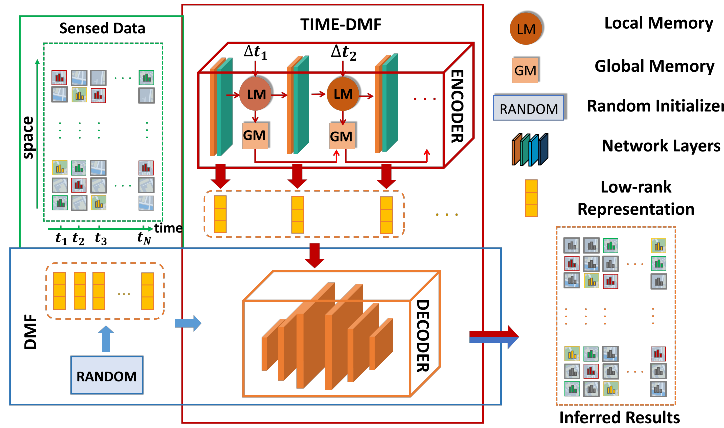


Fig. 3: Relationship between DMF and TIME-DMF.

within the same time unit. Suppose that we manually discretize the given timeline into P time units, we can have our new matrices $C^{(D)} \in R^{N\times P}$ and $Y^{(D)\prime} \in R^{N\times P}$ by putting each submission into the time unit they locate in and merge all the submissions within the same unit.

$$C^{(D)} = [c_1^{(D)\mathrm{T}}, c_2^{(D)\mathrm{T}}, \cdots, c_P^{(D)\mathrm{T}}], \tag{4}$$

$$Y^{(D)\prime} = [y_1^{(D)\prime\mathrm{T}}, c_2^{(D)\prime\mathrm{T}}, \cdots, c_P^{(D)\prime\mathrm{T}}], \tag{5}$$

$$c_p^{(D)}, y_p^{(D)\prime} = merge(c_{p_0}, c_{p_1}, \cdots, y'_{p_0}, y'_{p_1}, \cdots), \tag{6}$$

where c_{p_i} represents the $i-th$ submission arrived within the $p-th$ time unit and $merge(\cdot)$ represents the merging operation. Technically, for each subarea, if there is a submission within the given $p-th$ time unit, the corresponding position in $c_p^{(D)}$ will be set to one and the submitted value will be filled in $y_p^{(D)\prime}$. If there are multiple values observed for one location, the final value would be set to the average. By the merging operation, the sparsity of $C^{(D)}$ will be much less than that of matrix C. After the discretization and merging operation, we no longer keep the specific arrival time of all submissions and their intervals but use $p-index$ to represent their arrival time. Then our task is to infer the missing values in $Y^{(D)\prime}$ to know the value of each location on the map at each time unit.

$$\hat{Y} = f(Y^{(D)\prime}). \tag{7}$$

However, it is obvious that some temporal information has been ignored during the discretization and merging process such as the precise arrival time and the intervals. Inspired by this observation, we propose a new system model to reduce information loss during problem modeling phase. For the M submissions, we use an additional vector $T = [t_{(1)}, t_{(2)}, \cdots, t_{(M)}]$ to represent the accurate arrival time of each submission and no longer discretize C and Y' to $C^{(D)}$ and $Y^{(D)\prime}$. What's more, since in time-continuous completion each column in the observed matrix represents a specific moment instead of a period, we not only want to complete the matrix Y', but also the infinitely many moments that do not exist in Y'. In order to achieve this, we divide the challenging time-continuous completion task into to two subtasks.

Our first task is to achieve a mapping $f(\cdot)$ only to complete the matrix Y' which fully utilizes the temporal information in

T. This is similar to the time-discrete scenarios.

$$\hat{Y} = f(Y'). \tag{8}$$

After that, we want to provide accurate inference for any moment on the time line. Obviously there are countless values distributed along the time line, and our solution is to find a model that can provide accurate responses $\hat{y} \in R^N$ for any given time moment t.

$$\hat{y} = g(Y', t), t \in (t_0, t_M). \tag{9}$$

B. Problem Formulation

Problem [Completion and Generation on Continuous Timeline]: Given sparse sensed data Y' and time vector T, we aim to achieve the following two objectives:

- Identify a mapping $f(\cdot)$ to complete all the unsensed data in the matrix Y'. The mapping $f(\cdot)$ should adequately consider the high sparsity of Y' and the temporal information in T.
- Identify a model $G(\cdot)$ to accomplish the completion at any given time t'. $y^{(t')\prime}$ can be a column in Y' or not.

In this process, the mean square error is used to measure the quality of the completed and generated data. The following value should be minimized:

$$\epsilon(Y, Y') = \sum_{i}^{N} \sum_{j}^{M} |Y_{ij} - Y'_{ij}|. \tag{10}$$

IV. Fine-grained Completion with RNN-DMF

Fine-grained completion is the first step towards time-continuous completion. With the insight that we can construct our spatiotemporal matrix by including each submission in a unique time unit, we could align with the previous time-discrete problem setting but eliminate the assumption that data stays constant within each time unit. We propose Deep Matrix Factorization (DMF) [17] as a foundational framework for the following works. Due to its neural network-enabled structure, it can be updated by adding modules of different functions. In fine-grained completion, there is one submission within each time unit so that we only know the information of one spatial location, leading to the great reduction of spatial information within a unit and the extreme sparsity of observation matrix. In order to solve this problem, we further propose Recurrent Neural Network-enabled Deep Matrix Factorization (RNN-DMF) by introducing a temporal encoder into DMF. The encoder is able to utilize temporal information to compensate for the loss of spatial information.

A. Deep Matrix Factorization (DMF)

Given a sparse matrix $Y'_{m\times n}$, the traditional method is to take full rank decomposition of the estimated matrix $\hat{Y}_{m\times n}$ by using the property that the real matrix $Y_{m\times n}$ has the lower rank and can be decomposed. Assuming that $rank(Y) = r$, then the full rank decomposition can be expressed as:

$$Y_{m\times n} = P_{m\times r} Z_{r\times n}, \tag{11}$$

where P is a full rank matrix and Z is a row full rank matrix. Therefore, any column y_t of $Y_{m\times n}$ can be expressed as $y_t = Pz_t$. In this formula, z_t represents the low-rank vector that fully contains the information of the $t-$th column of $Y_{m\times n}$, which is also the $t-$th moment in real scenarios. P denotes the projection from the low-rank vectors to the inferred matrix. Note that Eq. (11) assumes that the spatiotemporal correlations between data are linear. To model the widely existing nonlinear spatiotemporal correlations, DMF was proposed. Similarly, we use the function $f(\cdot)$ to represent the nonlinear correlations between space and time.

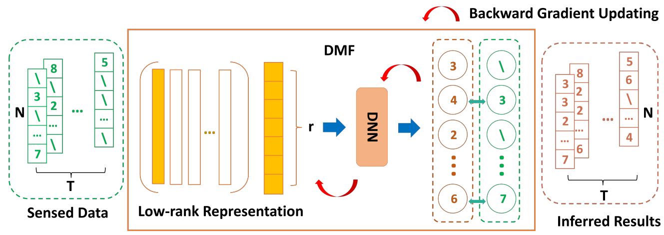


Fig. 4: The structure of DMF.

$$y = f(z), Y = f(Z). \tag{12}$$

We aim to obtain suitable z and $f(\cdot)$ by fitting them with a deep neural network (DNN). Specifically, assuming the neural network consists of K hidden layers, and their parameters are

$$W^* \triangleq \{W^{(1)}, W^{(2)}, \cdots, W^{(K)}, W^{(K+1)}\}, \tag{13}$$

$$b^* \triangleq \{b^{(1)}, b^{(2)}, \cdots, b^{(K)}, b^{(K+1)}\}. \tag{14}$$

The corresponding activation function set is

$$g^* \triangleq \{g^{(1)}(\cdot), g^{(2)}(\cdot), \cdots, g^{(K)}(\cdot), g^{(K+1)}(\cdot)\}. \tag{15}$$

In the set W^*, b^*, g^*, the $(K+1)th$ term represents the parameter or the activation function from the hidden layer K to its output layer. The nonlinear function is expressed as:

$$\begin{aligned} f(z) =& g^{(K+1)}(W^{(K+1)} g^{(K)}(W^{(K)}, \dots, \\ & g^{(1)}(W^{(1)} z + b^{(1)}) \cdots + b^{(K)}) + b^{(K+1)}). \end{aligned} \tag{16}$$

Different from traditional neural networks, the input z and the neural network parameters are both optimizable parameters in this context. The optimization object is as follows:

$$min \frac{1}{2n} \|(Y' - f(Z)) * C\|. \tag{17}$$

In this way, the inferred results can be obtained:

$$\hat{Y} = f(Z). \tag{18}$$

The detailed structure of DMF is shown in Fig. 4. DMF is a neural network framework that is already able to handle basic data completion tasks. However, it is not tailored for any specific scenarios. By inserting DNN modules of different functions into DMF, we can further enable DMF to deal with data of various specific properties.

B. Recurrent Neural Network-enabled Deep Matrix Factorization (RNN-DMF)

In each time step, the proposed DMF framework has low-rank vectors that can be concatenated to form the complete low-rank representation of spatiotemporal data. While in the

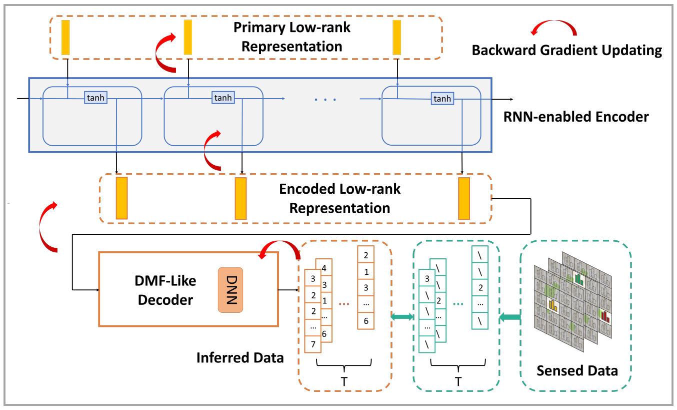


Fig. 5: The inner structure of RNN-DMF.

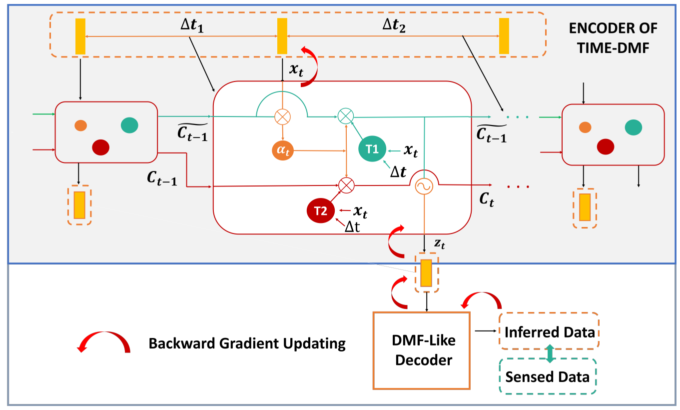


Fig. 6: The inner structure of TIME-DMF.

training process, each low-rank vector is learned separately and is not visible to each other. This limits the ability of DMF to fully capture temporal information. However, capturing and utilizing temporal information is an urgent in fine-grained completion due to the extreme sparsity of matrices. For this reason, we further propose Recurrent Neural Network-enabled Deep Matrix Factorization (RNN-DMF).

Fig. 6 shows that RNN-DMF is composed of two key modules: an upstream RNN-enabled encoder and a downstream DMF-like decoder. Unlike DMF, which initializes low-rank representations randomly, RNN-DMF accounts for relationships between low-rank vectors. The RNN structure enables the encoder to generate low-rank representations by incorporating temporal correlations, sharing parameters U, W, and V across time steps. The hidden state S_t is generated based on the previous state S_{t-1} and the primary low-rank vector X_t at each timestamp.

$$S_t = f(U \cdot X_t + W \cdot S_{t-1}). \tag{19}$$

As DMF, S_0 and the primary low-rank vector of each step is randomly initialized and serves as optimizable parameters during training. The encoded low-rank representation is then generated by a final projection. This process is similar to what traditional RNN does.

$$z_t = g(V \cdot S_t). \tag{20}$$

At this step, we finally have our encoded low-rank vectors with temporal information integrated. They will then be concatenated and decoded, serving as the final completion results. This is done by the downstream DMF-like decoder:

$$Z = [z_1^{\mathrm{T}}, z_2^{\mathrm{T}}, \cdots, z_M^{\mathrm{T}}], \tag{21}$$

$$\hat{Y} = f(Z). \tag{22}$$

It is obvious that RNN-DMF performs strictly superior to DMF theoretically. This is because DMF generates its low-rank vectors randomly and independently without considering temporal correlations. RNN-DMF considers the possible temporal correlations between low-rank vectors during the generating process, which is at least better than complete random. When sensed data is extremely sparse, there is very limited information for use within each time step, making it urgently necessary to share information between time steps. This explains why RNN-DMF performs significantly better than DMF, especially on extremely sparse matrices.

V. Time-continuous Completion with TIME-DMF

In the previous fine-grained completion, we address the challenges of the reduction of spatial information and the extreme sparsity of observation matrices. Based on that, we can finally introduce the time-continuous completion. In traditional works and fine-grained completion, the intervals between submissions are ignored during the discretization stage. However, in real-life scenarios, users submit data at real time and the interval between submissions are of great importance for inference. This oversimplification is also the reason why traditional time-discrete completion methods cannot fully capture temporal information.

Intuitively, it is not the absolute arrival time of submissions but their intervals that matter. The sensing data with a long period of time interval will vary significantly but the sensing data at adjacent time may be quite similar. This is the intuition of Time Gates-enhanced Deep Matrix Factorization (TIME-DMF) for our introducing time gates and a more complex propagation pattern to model the influence of intervals on correlations between time steps. Furthermore, in fine-grained scenarios, period is sliced into a finite number of units and the observation matrix is of fixed size. But from a time-continuous perspective, data changes continuously and there are infinitely many states of sensed data in a given period. In order to characterize data of all states at an acceptable price, we propose Q-G strategy. It allows users to query data of any time and leverages the generative ability of TIME-DMF to dynamically respond. By combining TIME-DMF and the Q-G strategy, we achieve the ultimate time-continuous completion. In this section, we will introduce the details of TIME-DMF and the Q-G strategy. Finally, we will conclude the complete flow of TIME-DMF algorithm for time-continuous data completion tasks.

A. Time Gates in TIME-DMF

The inner structure of TIME-DMF is shown in Fig. 6. Inspired by [41], we design two types time gates to manage a more complex pattern of information propagation. The first

Algorithm 1 Deep Matrix Factorization with Time Gates (TIME-DMF)

Require: $C_{N\times M},\ Y'_{N\times M},\ t,\ T = [t_1, t_2, \cdots, t_M],\ x = [x_1, x_2, \cdots, x_M]$
Ensure: $\hat{Y}$
Random Init x;
if $t \notin T$ **then**
  $T \leftarrow [t_1, t_2, \cdots, t_k, t, t_{k+1}, \cdots, t_M]$;
  $x \leftarrow [x_1, x_2, \cdots, x_k, x, x_{k+1}, \cdots, x_M]$;
  $C_{N\times M} \leftarrow C_{N\times(M+1)}, Y'_{N\times M} \leftarrow Y'_{N\times(M+1)}$;
end if
while not convergent **do**
  for $x^{(1)}$ to $x^{(M)}$ **do**
    $z_t, \tilde{C}_t, C_t \leftarrow encoder(x^{(t)}, \tilde{C}_{t-1}, C_{t-1})$;
  end for
  $\hat{Y} \leftarrow decoder([z_1^{\mathrm{T}}, z_2^{\mathrm{T}}, \cdots, z_M^{\mathrm{T}}])$;
  calculate and reduce $\|(Y' - \hat{Y}) * C\|$;
end while

time gate controls global memory. All data within the observed time period should follow a certain underlying distribution, reflecting the overall characteristics of the dataset, such as the mean or periodicity in real-life scenarios. The second time gate controls local memory. The distribution of data at adjacent time shows the local increase/decrease or other short-term trends. We combine these two memories to ensure the completion result is rational both globally and locally.

The two kinds of memories are controlled by different parameters and are updated independently. Intuitively, intervals determine the correlations between time steps. The shorter the interval between two time steps, the more similar they are expected to be. So we utilize intervals to control the update of memories. If the memories are slightly updated between the two adjacent time steps, their completion results will be very similar. Conversely, if the memories have a substantial update, the following results will greatly defer from the previous ones. We use the interval as a parameter within each time gates.

$$T1_m = \sigma(x_m W_{x1} + \sigma_{\Delta t}(\Delta t_m W_{t1}) + b_1), \tag{23}$$
$$T2_m = \sigma(x_m W_{x2} + \sigma_{\Delta t}(\Delta t_m W_{t2}) + b_2). \tag{24}$$

The function of two memories is similar to that of the hidden state in RNN, which is shown in Eq. (19).

In TIME-DMF, the global memories and local memories operate in different manners. The local memory has a more direct impact on the completion of each step:

$$a_t = f(U \cdot X_t + W \cdot \tilde{C}_{t-1}). \tag{25}$$

The global memory flows more consistently and has a broad effect on the update of memories. We use the two time gates calculated formerly to control the update of the two memories. In this way we make use of the information within intervals.

$$\tilde{C}_t = f[C_t \odot T1_m + a_t \odot (1 - T1_m)], \tag{26}$$
$$C_t = f[C_t \odot T2_m + a_t \odot (1 - T2_m)]. \tag{27}$$

Lastly, as the local memory can better assess the data status at the current moment, it is converted into an low-rank vector representing time t through the output gate.

$$z_t = f(V \cdot \tilde{C}_t). \tag{28}$$

At this point, the upstream encoder has been obtained. The encoder is capable of doing time-continuous completion as it can deal with submissions at any given moments. We further use the DMF-like decoder to map the low-rank representation to yield the final completion results. Similar to RNN-DMF, TIME-DMF adopts a joint training method for the upstream and downstream networks.

B. Q-G Strategy

Given a sparse observation matrix, RNN-DMF or TIME-DMF can infer the missing data. However, unlike previous methods relying on discrete slicing, in time-continuous scenarios, there are infinitely many possible moments on the timeline. Thus, it is impossible for the prior Submit-Complete strategy to infer data at all moments. To overcome this challenge, we propose the Query-Generate strategy which allows users to query data for any moment within the given period and dynamically generate the results. To use TIME-DMF for data generation, we leverage its property of completing data by first generating low-rank representation. We can insert a randomly initialized low-rank vector into the previous low-rank representation and then proceed with the usual completion process. The only difference is that the inserted vector does not affect the backward gradient propagation process.

C. The Complete Flow of TIME-DMF

For a time query provided by the user, we first check if there is an existing column in the given sparse matrix. If not, we insert an empty column into matrix Y' and update the intervals of its both sides. Concurrently, we insert a column with all zeros into matrix C to ensure the query won't affect existing completion process. We then deploy our encoder and decoder sequentially to complete the data matrix Y' and minimize the loss between the completed data and known data at observation positions. In this process, the encoder, decoder and the completed results are continuously updated by calculating and reducing the optimization objective. The complete algorithm flow is shown in Alg. 1.

VI. Experimental Validation

In this section, we first introduce the datasets and the comparison methods. Then we present performance evaluation of our proposed results. In particular, our experiments can be divided into the answers to the following research questions:

- **RQ1**: Does our RNN structure offer better completion performance for extremely sparse matrices?
- **RQ2**: Do our time gates truly leverage the time interval?
- **RQ3**: In time-continuous, the model needs to have additional generative capabilities. How effective is the generative capability of TIME-DMF?
- **RQ4**: Even our method achieves time-continuous completion, is time-continuous completion truly more effective than time-discrete completion?

TABLE I: Statistics of four evaluation datasets

	U-Air	Sensor-Scope	TaxiSpeed	Highway England
City/Country	Beijing (China)	Lausanne (Switzerland)	Beijing (China)	England
Data	PM2.5	Humidity	Traffic speed	People counting
Subarea	36 subareas	57 subareas	100 road segments	15∼25 subareas
Cycle & Duration	1h & 11d	0.5h & 7d	1h & 4d	0.25h & 0.5∼3months
Mean ± Std.	79.11 ± 81.21	84.52 ± 6.32	13.01 ± 6.97	112.42 ± 30.54

- **RQ5**: In the domain of spatiotemporal data, transformers seem to have become the mainstream approach. Why do we choose not to use transformer architectures?

A. Datasets

- **U-Air** [42] is utilized to gather significant air quality data, specifically PM2.5 and PM10 levels, via monitoring stations located in Beijing, China.
- **Sensor-Scope** [43] is employed to collect a diverse array of environmental readings through the deployment of numerous static sensors on the EPFL campus. A representative type of sensing, namely humidity, is selected for evaluation purposes.
- **TaxiSpeed** [44] gathers traffic speed data pertaining to road segments in Beijing, China by utilizing GPS devices installed on taxis.
- **Highways England (HE)** [45] serves as a resource in providing information pertaining to travel times, traffic flow rates, incidents, event data and camera imagery for England's major motorways. Due to its large space and time range, we manually selected spatiotemporal data from multiple adjacent regions and adjacent time periods, thus the data size is larger and more flexible.

B. Comparison Methods

1) Comparative models for completion task: Sparse-supervised methods which only rely on sparse observed data for training:

- **MC**, a classic linear matrix completion method, assumes a linear relationship $Y = PZ$.
- **KNN-S**, a variant of the K-Nearest Neighbors algorithm. KNN-S retrieves information from the K closest subregions to the region to be imputed and uses their average value as the imputation result.
- **GP** algorithm, a method that assumes the spatial distribution of data in the same cycle obeys the Gaussian distribution. The unknown data are inferred by calculating the expectation and variance of the known data.
- **DMF**, which has been introduced previously and also serves as a basic component of our method.
- **STformer** [46], a transformer-based model with multiple designed embedding and attention layers to capture spatiotemporal relationship. STformer is specially designed to be trained with only sparse observed data.

Dense-supervised methods which rely on complete observed data to train the model:

- **iTransforme**r [31], a variant which applies the attention and feed-forward network on the inverted dimensions. This is fine-tuned for completion tasks.
- **AutoFormer** [32], another variant of transformer which entangles different blocks in the same layers during supernet training.It is also fine-tuned for completion tasks.

2) Comparative Predictive models for generative task:

- **LINEAR**, which applies the linear regression model to predict the full map of the future cycles. It assumes that the sensed data varies linearly over time.
- **WNN**, which combines wavelet transform and neural network. WNN is good at extracting periodic features of time series for data prediction.
- **NAR**, which uses a nonlinear autoregressive neural network to predict the near future. NAR considers the nonlinear temporal correlations within data.

C. Completion on Extremely Sparse Data (RQ1)

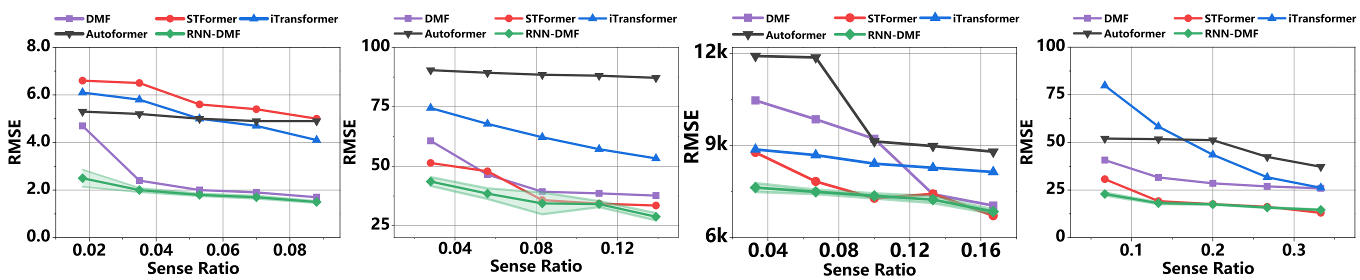
For most sparse data completion tasks, the sensing rate typically ranges from 20% to 80%, indicating that we can utilize abundant spatiotemporal information. However, in fine-grained and time-continuous completion, we have the matrices with sense ratio of $1/$n-columns which traditional methods may have difficulty handling. To show the effectiveness of RNN-DMF on extremely sparse data, we compare it with other existing completion methods on multiple datasets. Note that the sense ratio is not limited to $1/$n-columns to show the generalization performance of RNN-DMF.

As problem setting, each column of the data matrix represents a submission, and each row represents a subarea to sense. To build sparse datasets, we randomly mask the complete matrix and leave $1-5$ data points unmasked in each column to represent the sensed data. This process is entirely random, as cell selection strategy is not our focus. For fair comparison, we use a small amount of complete data to train dense-supervised methods to ensure that models function properly.

The results in Table II and Fig. 7 illustrate that RNN-DMF significantly outperforms existing works. The overall trend suggests that with the increase of sense ratio, the completion accuracy of all testing methods rises. This aligns with intuition as sensed data provides information for missing data inference. From Table II, we can clearly see that in scenarios where data matrix is extremely sparse, our method surpasses most existing methods. The green error bars in Fig. 7 also highlight the robustness of our method. This experiment forms the basis for our work as both fine-grained completion

TABLE II: Full RMSE results under different sparsity on four datasets.

Dataset	Sensed Ratio	Sparse-supervised method						Dense-supervised method	
		GP	knn-s	MC	DMF	RNN-DMF	STformer	iTransformer	Autoformer
U-AIR	1/57	4.3	4.2	3.9	4.7	**2.5**	6.6	6.1	5.3
	2/57	4.1	4.1	2.5	2.4	**2.0**	6.5	5.8	5.2
	3/57	3.6	3.2	2.5	2.0	**1.8**	5.6	5.0	5.0
	4/57	3.3	3.3	1.7	1.9	**1.7**	5.4	4.7	4.9
	5/57	3.2	3.2	1.6	1.7	**1.5**	5.0	4.1	4.9
Sensor-Scope	1/36	53.0	55.8	56.5	60.7	**43.6**	51.4	74.5	90.4
	2/36	54.2	44.6	44.4	46.5	**38.5**	47.9	67.8	89.3
	3/36	53.0	41.6	38.9	39.3	**34.4**	35.8	62.2	88.5
	4/36	52.5	39.4	36.9	38.6	**34.1**	34.3	57.2	88.1
	5/36	50.6	58.2	34.1	37.7	**28.8**	33.5	53.3	87.2
TaxiSpeed	1/30	39028.1	38252.8	31450.2	10474.5	**7632.0**	8778.2	8876.7	11921.1
	2/30	39990.3	32182.1	28295.3	9859.5	**7493.1**	7831.3	8691.6	11876.0
	3/30	38637.5	29491.2	24141.7	9220.7	7361.5	**7283.9**	8414.7	9139.1
	4/30	36617.3	28337.3	22709.9	7417.5	**7241.1**	7428.0	8277.5	8984.9
	5/30	35991.2	26787.2	21344.1	7012.0	6849.1	**6713.5**	8139.8	8803.8
Hishways England	1/15	66.5	70.5	66.1	40.7	**22.9**	30.7	79.8	52.0
	2/15	56.2	60.3	49.6	31.6	**18.1**	19.2	58.3	51.7
	3/15	50.7	53.8	41.6	28.5	**17.5**	17.6	43.5	51.2
	4/15	45.9	49.9	36.9	26.9	**15.8**	16.2	31.7	42.4
	5/15	42.5	52.4	29.8	25.9	14.6	**13.0**	26.2	37.2



(a) U-AIR (b) Sensor-Scope (c) TaxiSpeed (d) Highways England

Fig. 7: Partial RMSE results of different data sparsity under different datasets.

and time-completion models deal with scenarios where spatial information is lacking and the matrix is extremely sparse.

D. Ablation Study on Time Gates (RQ2)

Based on RNN-DMF, we aim to further demonstrate the effectiveness of time gates which are designed to capture information within unequal intervals between submissions. We design an ablation experiment between TIME-DMF, which incorporates time gates, and RNN-DMF, which does not incorporate time gates, on multiple datasets.

As most publicly available datasets consist of sensor data with equal time intervals, we artificially create datasets through random masking and deletion. In specific, for the complete data matrix, we randomly remove some columns from the matrix so that the left columns are of random intervals. For the columns left, we do random masking again like that in the last experiment. This preprocessing results in extremely sparse matrices where the intervals between columns vary in length. Because of the deletion, the datasets will be much smaller. So we need datasets of enough columns to make sure the experiments can have stable results. As Highways England is a dataset that collects road data per 15 minutes for years, it provides an ample number of columns for use. In this experiment we employ this typical dataset to test the effectiveness of time gates.

The experimental results are shown in Table III and Fig. 8. The x-axis represents the proportion of columns deleted from the original dataset. When the deletion proportion grows bigger, the interval between columns will be more uneven and there will be more temporal information within time intervals for time gates to capture. In all datasets, as the unevenness of the data increases, the inference performance of both RNN-DMF and TIME-DMF decreases. This is because when the data is uneven, the similarity between adjacent submissions decreases and it's hard for encoders to capture direct correlations between submissions. This trend is particularly evident when the deletion ratio is high. However, compared with RNN-DMF, TIME-DMF performs much more stably when the deletion rate and data unevenness increases. This is because RNN-DMF can only statically share parameters between time steps without considering the interval length. While in TIME-DMF, the global memory and local memory can cooperate to dynamically control the flow of temporal information. This propagation pattern better utilizes the information within

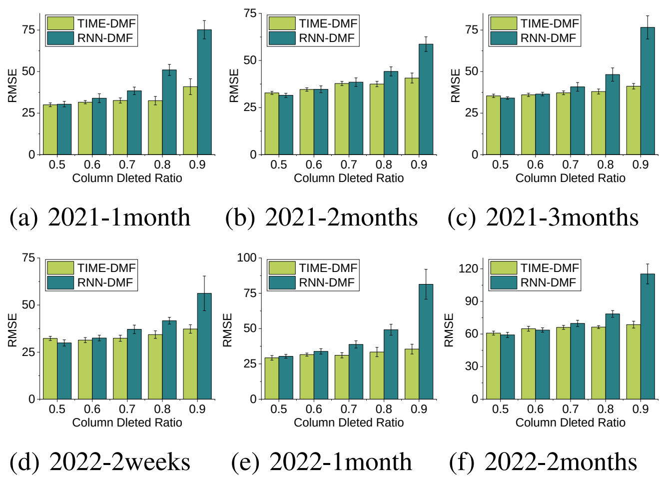
TABLE III: RMSE results of the ablation study of time gates.

Dataset	Method	Slices Deleted Ratio				
		0.5	0.6	0.7	0.8	0.9
1month	RNN-DMF	30.4	34.0	38.4	51.0	75.2
	TIME-DMF	**29.2**	**29.0**	**31.4**	**32.5**	**42.8**
2months	RNN-DMF	**31.5**	34.7	38.5	44.2	58.7
	TIME-DMF	32.8	**34.6**	**37.8**	**37.5**	**40.7**
3months	RNN-DMF	**34.1**	36.5	40.8	48.2	76.6
	TIME-DMF	35.4	**36.0**	**37.2**	**38.0**	**41.2**

Highways England for 2021

Dataset	Method	Slices Deleted Ratio				
		0.5	0.6	0.7	0.8	0.9
2weeks	RNN-DMF	**29.9**	32.5	37.1	41.7	56.2
	TIME-DMF	32.3	**31.4**	**32.4**	**34.3**	**37.3**
1month	RNN-DMF	30.4	33.9	38.8	49.2	81.4
	TIME-DMF	**29.3**	**31.6**	**31.1**	**33.4**	**35.5**
2months	RNN-DMF	**59.2**	**3.7**	69.8	78.5	115.3
	TIME-DMF	60.9	64.9	**66.1**	**66.4**	**68.7**

Highways England for 2022



(a) 2021-1month (b) 2021-2months (c) 2021-3months

(d) 2022-2weeks (e) 2022-1month (f) 2022-2months

Fig. 8: Completion results of two models with the only difference of time gates.

intervals of different lengths. If we consider RNN-DMF as the baseline, we observe that TIME-DMF performs better when other conditions remain unchanged. This directly proves the effectiveness of time gates in capturing information within unequal intervals.

E. Demonstration of Generative Capability (RQ3)

Since we account for data continuity, we cannot provide states for all moments. Instead, we enable our model to dynamically generate data states based on user requests. In this experiment, we test the generative capability of TIME-DMF. Note that existing time-discrete models don't have generative needs, so we compare TIME-DMF with series of predictive models. In order to make the comparison fairer, we just predict one step ahead in predictive models.

As we construct datasets via random masking and deletion, and each time we only generate data for a moment (a column in the matrix), the generation accuracy is highly unstable. For more persuasive results, we conduct extensive experiments on different randomly built datasets and depict the findings in a box plot. From the average height of the boxes in Fig. 9 we can tell that TIME-DMF provides superior overall generation accuracy. The comparison between the lengths of light green boxes and dark green boxes indicates that when the unevenness of data distribution rises, the overall performance becomes unstable. This corresponds to the conclusion that we draw in the second experiment. It should be noted that generation effect on U-AIR dataset fluctuates greatly under different deletion rates. This is because U-AIR is a rather tiny dataset, which becomes too small to contain ample temporal information as the deletion ratio increases. We do not use U-AIR in our second experiment for the same reason.

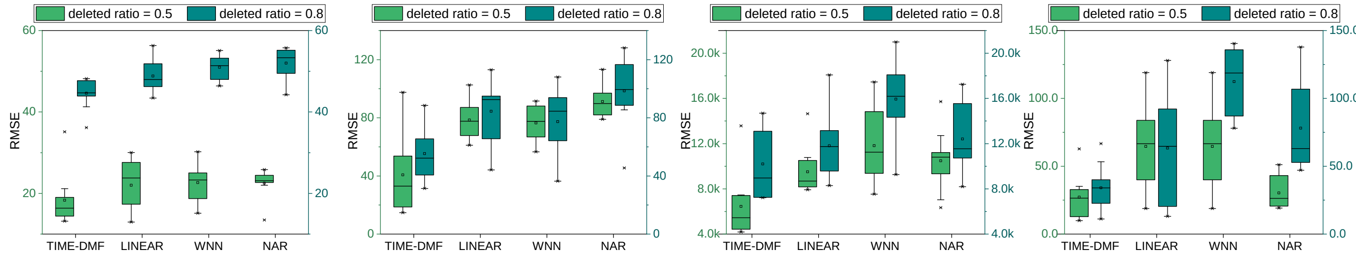
F. Comparison between Time-discrete and Time-continuous completion (RQ4)

In time-discrete scenarios, we reduce matrix sparsity by sacrificing accurate submission times, while in time-continuous scenarios, we use time intervals but face increased sparsity. After showing that our model can handle high sparsity and effectively use intervals, we need to prove that time-continuous completion truly outperforms time-discrete solutions. To test this, we compare TIME-DMF with time-discrete methods in their original problem setting. We simulate real MCS scenarios by constructing submissions with random masking and deletion from the original datasets. For time-discrete methods, we divide the timeline into equal slices and merge data within each slice, while TIME-DMF keeps submissions in their original state with recorded arrival times. We then apply various completion methods and compare their accuracy.

After random deletion, the size of datasets will reduce a lot and completion methods may yield different results on datasets of varying sizes. To show the universality, we conduct the experiments on both smaller datasets and larger datasets. In Fig. 10 and Fig. 11, moving from left to right, the number of columns in the datasets increases.

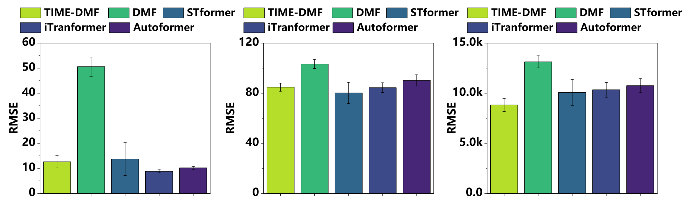
As shown in Table IV, Fig. 10 and Fig. 11, TIME-DMF (which is a time-continuous completion method) significantly outperforms the traditional methods (which are all time-discrete methods). While incorporating the continuous change of data does present new challenges such as extreme sparsity and interval handling, it is still a better model of the real scenarios if we could utilize temporal information to the maximum. When comparing the completion effects on smaller datasets and bigger ones, the performance on bigger datasets is better. This can be explained as the combination of global memories and local memories can be fully useful in the long term data distribution. While in smaller datasets, the design of global memory and local memory may seem to be unnecessary.

Previous studies used merged data as the ground truth for model training, leading to significant deviations between experimental and real results, which caused their models to be



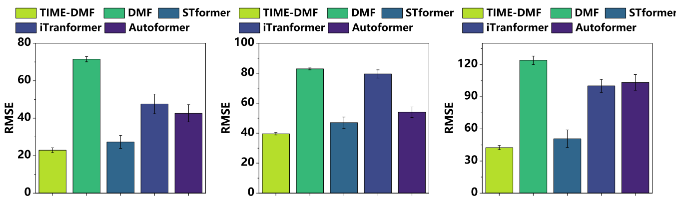
(a) U-AIR (b) Sensor-Scope (c) TaxiSpeed (d) Highways England

Fig. 9: RMSE of different data unevenness under different datasets.



(a) U-AIR (b) Sensor-Scope (c) TaxiSpeed

Fig. 10: RMSE for tiny datasets.



(a) HE-1month (b) HE-2months (c) HE-3months

Fig. 11: RMSE for large datasets.

TABLE IV: Direct comparison of time-continuous and time-discrete methods.

Dataset	Sparse-supervised Methods						Dense-supervised Methods	
	GP	KNN-S	MC	DMF	TIME-DMF	STformer	iTransformer	Autoformer
U-AIR	14.0	40.4	49.7	50.6	12.6	13.7	**8.8**	10.2
Sensor-Scope	105.7	93.0	96.9	103.3	84.8	**80.2**	84.4	90.2
TaxiSpeed	33213.7	26283.4	23232.2	13129.9	**8822.4**	10068.7	10337.1	10748.0
Hishways England A	66.5	70.5	66.1	71.5	**22.9**	27.3	47.6	42.6
Hishways England B	146.7	129.6	89.1	82.9	**39.6**	47.0	79.5	54.0
Hishways England C	207.7	182.0	109.6	124.1	**42.5**	50.8	100.2	103.4

TABLE V: Training condition, speed and parameter count.

	Time-DMF	iTransformer	Autoformer	STformer
Only Sparse-supervised	✓	×	×	✓
Parameter Count	**9k**	51k	56k	11M
Training Time / Epoch	**18ms**	58ms	66ms	400ms

misestimated. By evaluating the model in a way that is closer to real-world problems, we demonstrate that TIME-DMF's advanced time-continuous completion is a superior solution for real-world MCS applications.

G. Model Parameter Comparison (RQ5)

Transformer architecture has been a popular solution in time series modeling. We chose LSTM over the Transformer architecture by considering training conditions, parameter count, and speed. Fine-tuned Transformer models for completion tasks often require complete historical data for training, which is unavailable in typical MCS scenarios, making dense-supervised methods difficult to train effectively. Additionally, edge computing devices are sensitive to computational resources, so model size and training speed are critical. As shown in Table V, TIME-DMF does not need extensive historical data and has a more compact parameter setting. Given that MCS tasks focus on specific data resources and don't require broad generalization, we find LSTM to be a more efficient and suitable choice compared to Transformers.

VII. Conclusion

In this paper, we propose a time-continuous completion method called TIME-DMF to challenge a widely existing assumption in Sparse MCS that data stays constant within periods and finally increase data completion accuracy to a great extent. TIME-DMF is based on DMF which is a neural network-enabled framework for traditional data completion. Based on that, TIME-DMF is further inserted with a temporal encoder that has the function of passing temporal information between time steps and utilizing the intervals of different lengths. To pass temporal information, we imitate the structure of RNN to generate appropriate embedding vectors. For utilizing the temporal information within intervals, we design the global memory to control the overall property of data distribution and the global memory to control local trend. The length of interval serves as a parameter to monitor the memory updating process. TIME-DMF is used in cooperation with the Q-G strategy which allows users to query and then dynamically generate responses. Finally, we do extensive experiments on real-world datasets to prove the effectiveness of our models.

References

[1] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[2] J. Wu and E. Wang, *Mobile Crowdsourcing From Theory to Practice*. Springer, 2023.

[3] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, and A. M'hamed, "Sparse mobile crowdsensing: challenges and opportunities," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 161–167, 2016.

[4] W. Liu, Y. Yang, E. Wang, and J. Wu, "Fine-grained urban prediction via sparse mobile crowdsensing," in *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2020, pp. 265–273.

[5] C. Feng, Y. Tian, X. Gong, X. Que, and W. Wang, "Mcs-rf: mobile crowdsensing–based air quality estimation with random forest," *International Journal of Distributed Sensor Networks*, vol. 14, no. 10, p. 1550147718804702, 2018.

[6] J. Liu, G. P. Ong, and X. Chen, "Graphsage-based traffic speed forecasting for segment network with sparse data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1755–1766, 2020.

[7] A. Ali, M. A. Qureshi, M. Shiraz, and A. Shamim, "Mobile crowd sensing based dynamic traffic efficiency framework for urban traffic congestion control," *Sustainable Computing: Informatics and Systems*, vol. 32, p. 100608, 2021.

[8] F. Calabrese, L. Ferrari, and V. D. Blondel, "Urban sensing using mobile phone network data: a survey of research," *Acm computing surveys (csur)*, vol. 47, no. 2, pp. 1–20, 2014.

[9] W. Liu, Y. Yang, E. Wang, L. Wang, D. Zeghlache, and D. Zhang, "Multi-dimensional urban sensing in sparse mobile crowdsensing," *IEEE Access*, vol. 7, pp. 82 066–82 079, 2019.

[10] L. Wang, D. Zhang, A. Pathak, C. Chen, H. Xiong, D. Yang, and Y. Wang, "Ccs-ta: Quality-guaranteed online task allocation in compressive crowdsensing," in *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*, 2015, pp. 683–694.

[11] L. Wang, D. Zhang, D. Yang, A. Pathak, C. Chen, X. Han, H. Xiong, and Y. Wang, "Space-ta: Cost-effective task allocation exploiting intradata and interdata correlations in sparse crowdsensing," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 9, no. 2, pp. 1–28, 2017.

[12] S. He and K. G. Shin, "Steering crowdsourced signal map construction via bayesian compressive sensing," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1016–1024.

[13] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, and D. Zhang, "Active sparse mobile crowd sensing based on matrix completion," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 195–210.

[14] W. Liu, E. Wang, Y. Yang, and J. Wu, "Worker selection towards data completion for online sparse crowdsensing," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1509–1518.

[15] E. Wang, M. Zhang, W. Liu, H. Xiong, B. Yang, Y. Yang, and J. Wu, "Outlier-concerned data completion exploiting intra-and inter-data correlations in sparse crowdsensing," *IEEE/ACM Transactions on Networking*, 2022.

[16] H. Aly, A. Basalamah, and M. Youssef, "Automatic rich map semantics identification through smartphone-based crowd-sensing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2712–2725, 2016.

[17] J. Fan and J. Cheng, "Matrix completion by deep matrix factorization," *Neural Networks*, vol. 98, pp. 34–41, 2018.

[18] M. Zhang and Y. Chen, "Inductive matrix completion based on graph neural networks," *arXiv preprint arXiv:1904.12058*, 2019.

[19] D. Zhang, L. Wang, H. Xiong, and B. Guo, "4w1h in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 42–48, 2014.

[20] N. B. Truong, G. M. Lee, T.-W. Um, and M. Mackay, "Trust evaluation mechanism for user recruitment in mobile crowd-sensing in the internet of things," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 10, pp. 2705–2719, 2019.

[21] T. Luo, J. Huang, S. S. Kanhere, J. Zhang, and S. K. Das, "Improving iot data quality in mobile crowd sensing: A cross validation approach," *IEEE Internet of Things journal*, vol. 6, no. 3, pp. 5651–5664, 2019.

[22] Z. Sisi and A. Souri, "Blockchain technology for energy-aware mobile crowd sensing approaches in internet of things," *Transactions on Emerging Telecommunications Technologies*, vol. 35, no. 4, p. e4217, 2024.

[23] L. Hu, A. Liu, M. Xie, and T. Wang, "Uavs joint vehicles as data mules for fast codes dissemination for edge networking in smart city," *Peer-to-Peer Networking and Applications*, vol. 12, pp. 1550–1574, 2019.

[24] C. Xiang, P. Yang, C. Tian, H. Cai, and Y. Liu, "Calibrate without calibrating: An iterative approach in participatory sensing network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 351–361, 2014.

[25] L. Liu, W. Liu, Y. Zheng, H. Ma, and C. Zhang, "Third-eye: A mobilephone-enabled crowdsensing system for air quality monitoring," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, pp. 1–26, 2018.

[26] H. Aly, A. Basalamah, and M. Youssef, "Automatic rich map semantics identification through smartphone-based crowd-sensing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2712–2725, 2016.

[27] D. Suhag and V. Jha, "A comprehensive survey on mobile crowdsensing systems," *Journal of Systems Architecture*, p. 102952, 2023.

[28] Y. Zhu, Z. Li, H. Zhu, M. Li, and Q. Zhang, "A compressive sensing approach to urban traffic estimation with probe vehicles," *IEEE Transactions on Mobile Computing*, vol. 12, no. 11, pp. 2289–2302, 2012.

[29] W. Liu, L. Wang, E. Wang, Y. Yang, D. Zeghlache, and D. Zhang, "Reinforcement learning-based cell selection in sparse mobile crowdsensing," *Computer Networks*, vol. 161, pp. 102–114, 2019.

[30] L. Wang, D. Zhang, D. Yang, B. Y. Lim, X. Han, and X. Ma, "Sparse mobile crowdsensing with differential and distortion location privacy," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2735–2749, 2020.

[31] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "itransformer: Inverted transformers are effective for time series forecasting," *arXiv preprint arXiv:2310.06625*, 2023.

[32] M. Chen, H. Peng, J. Fu, and H. Ling, "Autoformer: Searching transformers for visual recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 270–12 280.

[33] N. Thepvilojanapong, T. Ono, and Y. Tobe, "A deployment of fine-grained sensor network and empirical analysis of urban temperature," *Sensors*, vol. 10, no. 3, pp. 2217–2241, 2010.

[34] Y. Hu, G. Dai, J. Fan, Y. Wu, and H. Zhang, "Blueaer: A fine-grained urban pm2. 5 3d monitoring system using mobile sensing," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[35] Y. Yang, Z. Zheng, K. Bian, L. Song, and Z. Han, "Real-time profiling of fine-grained air quality index distribution using uav sensing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 186–198, 2017.

[36] J. Huang, N. Duan, P. Ji, C. Ma, Y. Ding, Y. Yu, Q. Zhou, W. Sun *et al.*, "A crowdsource-based sensing system for monitoring fine-grained air quality in urban environments," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3240–3247, 2018.

[37] K. Doya, "Reinforcement learning in continuous time and space," *Neural computation*, vol. 12, no. 1, pp. 219–245, 2000.

[38] P. J. Brockwell, "Continuous-time arma processes," *Handbook of statistics*, vol. 19, pp. 249–276, 2001.

[39] T. Higuchi, "Approach to an irregular time series on the basis of the fractal theory," *Physica D: Nonlinear Phenomena*, vol. 31, no. 2, pp. 277–283, 1988.

[40] P. Kidger, J. Morrill, J. Foster, and T. Lyons, "Neural controlled differential equations for irregular time series," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6696–6707, 2020.

[41] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, "What to do next: Modeling user behaviors by time-lstm." in *IJCAI*, vol. 17, 2017, pp. 3602–3608.

[42] Y. Zheng, F. Liu, and H.-P. Hsieh, "U-air: When urban air quality inference meets big data," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 1436–1444.

[43] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: Application-specific sensor network for environmental monitoring," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 2, pp. 1–32, 2010.

[44] J. Shang, Y. Zheng, W. Tong, E. Chang, and Y. Yu, "Inferring gas consumption and pollution emission of vehicles throughout a city," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1027–1036.

[45] G. D. Service, "Highways england," data.gov.uk/dataset/highways-england-network-journey-time-and-traffic-flow-data, 2022.12.

[46] E. Wang, W. Liu, W. Liu, C. Xiang, B. Yang, and Y. Yang, "Spatiotemporal transformer for data inference and long prediction in sparse

mobile crowdsensing," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.