2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS) | 978-1-6654-7177-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDCS54860.2022.00077

CODE: Compact IoT Data Collection with Precise Matrix Sampling and Efficient Inference

Huali Lu[1], Feng Lyu[1]*, Ju Ren[2], Jiadi Yu[3], Fan Wu[2], Yaoxue Zhang[2], and Xuemin (Sherman) Shen[4]
[1]School of Computer Science and Engineering, Central South University, Changsha, China
[2]Department of Computer Science and Technology, BNRist, Tsinghua University, Beijing, China
[3]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[4]Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada
*Corresponding Author
{huali_lu, fenglyu}@csu.edu.cn, {renju, wufancs, zhangyx}@tsinghua.edu.cn, jdyu@cs.sjtu.edu.cn, sshen@uwaterloo.ca

***Abstract*—It is impractical to conduct full-size data collection in ubiquitous IoT data systems due to the energy constraints of IoT sensors and large system scales. Although sparse sensing technologies have been proposed to infer missing data based on partial sampled data, they usually focus on data inference while neglecting the sampling process, restraining the inference efficiency. In addition, their inferring methods highly depend on data linearity correlations, which become less effective when data are not linearly correlated. In this paper, we propose, Compact IOT Data CollEction, namely *CODE*, to conduct precise data matrix sampling and efficient inference. Particularly, *CODE* integrates two major components, i.e., cluster-based matrix sampling and Generative Adversarial Networks (GAN)-based matrix inference, to reduce the data collection cost and guarantee the data benefits, respectively. In the sampling component, a cluster-based sampling approach is devised, in which data clustering is first conducted and then a two-step sampling is performed in accordance with the number of clusters and clustering errors. For the inference component, a GAN-based model is developed to estimate the full matrix, which consists of a generator network that learns to generate a fake matrix, and a discriminator network that learns to discriminate the fake matrix from the real one. A reference implementation of *CODE* is conducted under three operational large-scale IoT systems, and extensive data-driven experiment results are provided to demonstrate its efficiency and robustness.**

***Index Terms*—Compact Data Collection, Sparse Sensing, Data Inference, Generative Adversarial Networks**

I. Introduction

As one of the most important impetuses to push forward the construction of smart city, large-scale Internet of Things (IoT) systems have been widely deployed permeating the fields of environment monitoring, transportation, communication, etc. [1]–[4]. Over time, the IoT systems can collect extensive data in terms of system operation, user profiles, targeted status, etc., based on which a lot of intelligent services to government, business, and users, can be provisioned by leveraging the advanced technologies of big data and deep learning [5]–[8]. However, collecting and storing the large-scale IoT data is costly and usually prohibited when considering the system efficiency and robustness. On the one hand, as the system scale increases, the data size grows explosively, posing significant communication and storage overhead for data collection. On the other hand, the IoT sensors are usually power- and communication-restrained, being prohibited to support full-size data collection in the long run [9], [10]. Besides, the IoT data can have underlying spatio-temporal correlations, and the full-size data collection can result in information redundancy, restraining the system efficiency. Therefore, how to reduce the data collection cost without losing data benefits has become an urgent yet challenging problem for ubiquitous data systems.

To deal with the issue, sparse sensing can be leveraged to reduce the data collection cost, via which the system can infer the missing data based on partial sampled data [11]. As only partial data are collected, the communication and storage cost can be largely reduced. However, for most IoT applications, such as trajectory prediction [12], anomaly detection [13], and evolution analysis [14], they are highly dependent on the full-size data, and thus the accuracy of matrix completion/inference[1] becomes crucial for the system. Therefore, in recent years, the technologies of compressive sensing (CS) [15], matrix completion (MC) [16], and tensor completion (TC) [17], have received extensive attentions, which can be applied to infer the missing data from the low-rank feature data. However, as the existing sparse sensing technologies mainly focus on designing efficient algorithms to infer the missing data, they usually assume that the incomplete data already exists, or the random/uniform sampling approach is used, which cannot work well in general since the sampling process can affect the inference performance significantly. Particularly, for the random/uniform sampling approaches, they can perform well for Gaussian distribution data since each matrix location carries the same amount of information. Unfortunately, they can lose efficiency rapidly when there are information differences among matrix locations for non-Gaussian distribution data, which has been verified by our pilot experiments. On the other hand, the conventional sparse sensing technologies mainly rely on the data linearity correlations to infer the missing data, which restrains the inference performance without capturing the non-linearity features for complicated IoT data structures.

To bridge this gap, in this paper, we take the lead to

[1]In this paper, the words of completion and inference are used alternately.

2575-8411/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDCS54860.2022.00077

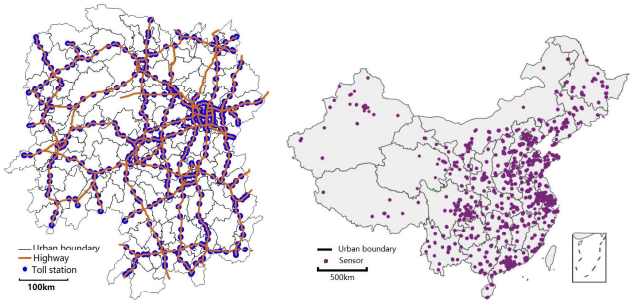
consider the deeply coupled problems of matrix[2] sampling and inference, to enable compact IoT data collection without losing the data benefits, which, however, is challenging for the following reasons. First, given a target matrix, to sample how much data is unknown ahead for the system. On the one hand, our goal is to sample as less data as possible in order to save the data collection cost. On the other hand, without sufficient sampling ratio, no matter how to design the sampling strategy, the required data information cannot be guaranteed, leading to the matrix recovering failures. Second, given the amount of samples, how to determine their locations within the matrix is another challenge since the sampling quality can affect the data inference performance directly, especially for the non-Gaussian distribution data. Third, with the sampled incomplete matrix, how to design the matrix inference model that can capture both the linearity and non-linearity correlations becomes crucial.

To tackle the above challenges, we first conduct an empirical study on typical IoT data systems by disclosing their low-rank features, based on which the matrix completion theory can be leveraged to determine the minimum amount of samples. In accordance with the principle, we then propose a systematical framework, named *CODE*, i.e., Compact IOT Data CollEction, which consists of two major components, i.e., matrix sampling and matrix inference, to determine sampling locations and conduct matrix inference, respectively. Particularly, in the matrix sampling component, given the amount of samples $|\Omega|$ that to be collected, we devise a cluster-based sampling approach. Particularly, for training matrix samples, the sensing data of each row (in spatial domain) are first clustered based on data values, resulting in a total of K clusters, and then the two-step scheduling process is performed to determine the respective K and $|\Omega| - K$ locations according to the number of clusters and clustering errors. In the matrix inference component, we devise a Generative Adversarial Networks (GAN) model to output the estimated matrix by learning the spatio-temporal correlations of data. Particularly, in the GAN-based model, there is a generator neural network that learns to generate a fake matrix based on the sampled matrix, and a discriminator neural network that learns to judge whether the generated matrix is true or fake. Those two neural networks are trained alternately to minimize both the generating and discriminating loss, which can be used to conduct matrix inference after offline training. The merits of *CODE* are two-folds: 1) the cluster-based sampling approach can keep pace with the underlying data distribution by achieving much more data information; 2) the GAN-based model can unleash the full potential of spatio-temporal correlations among data for matrix inference, collectively contributing to a superior performance.

The main contributions are summarized as follows.

- We take the lead to consider the deeply coupled problems of matrix sampling and inference, both of which are valuable to data collection performance in large-scale IoT systems, while existing research works usually lack one dimension control and suffer from approach defects.
- We conduct an empirical study on typical IoT datasets, and disclose insightful observations, such as long-tailed distribution and low-rank feature, which direct our solutions to address the considered data collection problem.
- We propose *CODE* to determine sampling locations and conduct matrix inference, which includes two technical components: 1) cluster-based matrix sampling; and 2) GAN-based matrix inference. The reference implementation of *CODE* is developed under three operational large-scale IoT data systems, and extensive data-driven experiments demonstrate its efficiency and robustness.

The remainder of this paper is organized as follows. The system description and problem definition are given in Section II. We conduct an empirical data analytics in Section III, and elaborate on the design of *CODE* in Section IV. Extensive experiments are conducted in Section V, and the related work is reviewed in Section VI. Finally, we conclude the paper.



(a) Provincial ETC system (b) National air quality sensors

Fig. 1. Typical large-scale IoT data systems.

II. System Description and Problem Definition

A. Large-Scale IoT Data Systems

In the big data era, extensive physical entities are managed by information systems, where large-scale IoT sensors are deployed to collect entity information, monitor the system environment, and act as server providers for nearby users, such as transportation ETC (Electronic Toll Collection) systems, air quality sensor networks, cellular communication networks, IoT monitoring systems, etc. By collecting the system data, extensive intelligent services can be enabled based on advanced techniques of big data and deep learning, where the government can achieve precise governance capability with system overview, the service providers can deliver cost-efficient services with efficient resource utilization, and the users can enjoy high-quality services with personalization [18]. Fig. 1 shows two typical large-scale IoT data systems, i.e., provincial highway ETC system and national air quality sensor networks, which are two fundamental building blocks for smart city[3]. Particularly, Fig. 1(a) shows a provincial

[2]For IoT systems, the data is usually managed in matrix format with temporal and spatial domains.

[3]Note that, the considered problem and proposed solutions in this paper can be readily applied in general large-scale IoT data systems, where the exemplary scenarios are used to carry out specific verification.

highway ETC system, in which there are 465 toll stations across 6,800 km highway roads covering 90 districts in the province. Normally, the daily transition traffic in the system can reach 1.6 millions, and the moving states of these vehicles have to be collected by the system. Fig. 1(b) shows a national air quality sensor network[4], which is used to monitor the air quality around the whole country, and there are about 1,518 sensors located in 375 cities. Each sensor needs to report the air quality value to the sink node on an hourly basis. Considering the system scales and energy constraints of sensors, full-size data collection becomes the significant bottleneck for system efficiency and robustness.

B. Problem Definition

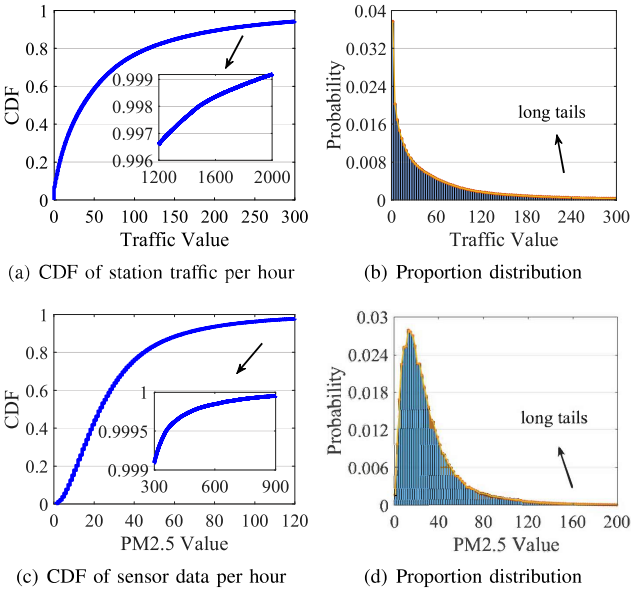
In particular, we consider a general data collection system, where each sensor has to collect the sensing data and deliver the data to one sink every time slot. Suppose there are a set of n sensors, $\{s_1, s_2, \ldots, s_n\}$, and m time slots, $\{t_1, t_2, \ldots, t_m\}$, where the length of each time slot can be set in accordance with the system management/granularity requirements. Then, the collected data can be represented by an $n \times m$ matrix $X \in \mathbb{R}^{n \times m}$, and each entry x_{ij} indicates the monitoring data from the sensor i at the time slot j. For n sensors in the IoT data system, instead of making each sensor periodically collect and report data to the sink, in each time slot, only a subset of sensors are scheduled to perform the sensing data collection. We define a matrix $X_{n \times m}$ to hold the data collected from n sensors within the m time slots. In the matrix, the row corresponds to different sensing locations and the column corresponds to different time slots. For instance, in the above mentioned highway ETC system, the row indicates different toll stations and the column indicates different time slots, where each entry in the matrix is a recorded number of traffic flow passing through the station within the time slot.

As the spatio-temporal data usually has similarity among neighboring locations and periodicity among time slots, which has been verified by our pilot experiments, we can sample partial data intelligently to reduce the data collection cost, and infer other empty data based on them to recover the data information. If there is no measurement data for a location at one time slot, the corresponding entry in X is set to be empty. All observed entries are denoted by $\Omega = \{i, j \mid x_{ij} \text{ is known}\}$, which forms an incomplete data sample matrix $S_{n \times m}$. If the set Ω contains enough information, we can use the incomplete data sample matrix $S_{n \times m}$ to reconstruct the complete data matrix $\hat{X}$. Therefore, we can cast the data collection problem as the following minimization problem:

$$\begin{aligned} &\min_{\{\Omega\}} \ \ \text{cost}(S) \\ &\text{subject to } \sum \left| (X - \hat{X}) \star X \right| \leq \theta, \end{aligned} \tag{1}$$

where $\text{cost}(.)$ denotes the cost of sampling the set Ω to form the incomplete data sample matrix S. Here, we assume that the sampling cost at each location is the same, so minimizing $\text{cost}(.)$ equals to minimize the number of samples. In addition, $\star$ represents a scalar division of two matrices, and θ is the predetermined relative error ratio that can be accepted by the data collection system.

[4]https://quotsoft.net/air



(a) CDF of station traffic per hour (b) Proportion distribution

(c) CDF of sensor data per hour (d) Proportion distribution

Fig. 2. Long-tailed distribution of observed values.

C. Problem Challenges

To tackle the above problem, the following technical challenges have to be carefully addressed:

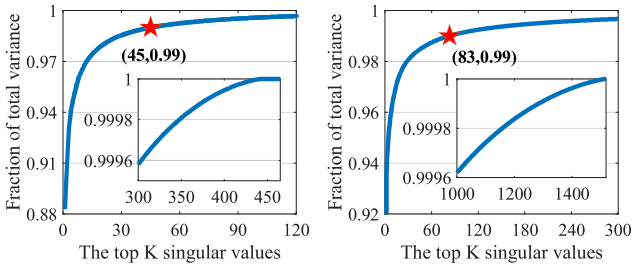
1) **Determining the Amount of Sampling Data.** To reduce the sampling cost, the first issue is to consider how much data are sufficient to recover the required data information, that is, how to determine the minimum amount of data (i.e., $|\Omega|$) to be sampled. Based on $|\Omega|$, the system should be able to infer the missing data with an acceptable inference error.
2) **Determining the Locations of Sampling Data.** The amount of samples determines whether the unsampled data can be accurately reconstructed, while the sampling locations (i.e., Ω) determine the quality of sampled data, which can also affect the inference accuracy significantly.
3) **Reconstructing the Unsampled Data.** Sampling can reduce the data collection cost, but data-based services usually rely on the entire data information (i.e., $\hat{X}$). Therefore, how to design the inference model to recover the entire data is another pivotal building block.

III. Empirical Study on IoT System Data

In this section, we conduct an empirical study on typical IoT system datasets to characterize the data distribution and low-rank feature, which are valuable to efficiently solving the above problem.

A. Long-Tailed Distribution of Observed Values

In actual IoT data systems, the distributions of data often differ from ideal assumptions. We reveal this phenomenon by



(a) SVD result of station traffic (b) SVD result of sensor data

Fig. 3. Low-rank feature of IoT system data.

analyzing the data of two typical IoT data systems. In the highway ETC system, each station will record the number of transition vehicles within each time slot. We first investigate the distribution of traffic value, where the time slot is set to be one hour and we adopt the one-year data dating from Jan. 1st, 2019 to Dec. 31th, 2019. Fig. 2(a) shows the cumulative distribution function (CDF) of station traffic, and we can achieve the following two major observations. First, the traffic value can vary widely, e.g., being as small as 0 and as large as $2,000$, which can enlarge the information space and pose challenges to data sampling and inference. Second, the observed traffic values follows a clear long-tailed distribution. Particularly, the 80 and 90 percents of traffic values are limited within about 110, 200, respectively, while for the remaining 10 percents, the traffic values can span to $2,000$. Fig. 2(b) shows the proportion of station traffic, which cross-verifies the long-tailed phenomenon. For instance, when the traffic value is larger than 180, the proportion becomes quite small, composing the long tails. In addition, we can observe that the proportions of traffic values are uneven, which generally decreases with the traffic value.

The same phenomenon appears in air quality monitoring data. For the air quality sensor network, each sensor needs to report data every one hour. The data contains the concentration of many different pollutants (e.g., PM2.5, PM10, SO_2, and NO_2), together with some meteorological logs (e.g., rainfall, wind speed, and temperature) collected within the country. Among them, the primary pollutant of air quality is PM2.5, and thus we employ its values as the target data. Particularly, we adopt nearly one-year data dating from Jan. 11th to Dec. 11th, 2021 to observe the overall distribution of the sensor data. As shown in Fig. 2(c) and Fig. 2(d), a widely and long-tailed distribution is also followed by wireless sensor data. Both experiments reveal that in real IoT data systems, the data distribution usually cannot conform to a perfect Gaussian distribution. Therefore, when sampling the same amount of data, the traditional random/uniform approaches cannot well capture the data diversity information.

B. Low-Rank Feature

The low-rank feature is the prerequisite to adopt the sparse sampling approach, otherwise, the unsampled data cannot be accurately recovered. In this subsection, we examine the low-rank features of the previous two typical IoT datasets. Particularly, we apply the singular value decomposition (SVD) approach to strictly examine whether the data matrix has a low-rank structure. Particularly, the traffic matrix $X_{n\times m}$ can be decomposed as

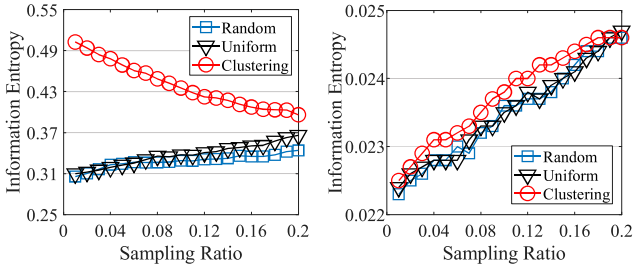
$$X = U\Sigma V^T, \tag{2}$$

where $U = \begin{bmatrix} u_{11} & \cdots & u_{1n} \\ \vdots & \vdots & \vdots \\ u_{n1} & \cdots & u_{nn} \end{bmatrix}, V = \begin{bmatrix} v_{11} & \cdots & v_{1m} \\ \vdots & \vdots & \vdots \\ v_{m1} & \cdots & v_{mm} \end{bmatrix}$ is the $n \times n$ and $m \times m$ unitary matrix, respectively. In addition, Σ is an $n \times m$ diagonal matrix with the diagonal elements (i.e., the singular values) in descending order, i.e., $\Sigma = \mathrm{diag}\,(\sigma_1, \sigma_2, \ldots, \sigma_r, 0, \ldots, 0)$. Denote by r the rank of matrix X, which is equivalent to the number of its non-zero singular values. A matrix is low-rank if it holds that $r \ll \min\{n, m\}$. Note that, the above calculation is defined for precise rank, which can be ill-posed for practical data when there exists arbitrary and small perturbations of matrix elements, containing limited data information but affecting the rank calculation significantly. Therefore, instead of calculating the precise rank, we adopt the approximate rank, which is widely used in the literature [19]. Particularly, the matrix X is claimed to have ω-rank k if

$$\inf\{\|X - Y\| : Y \quad \text{has} \quad \text{rank} \quad k\} \leq \omega. \tag{3}$$

where $\|\cdot\|$ represents norm operation used to calculate the distance of X and Y (Y is any matrix with rank k), $\inf$ represents the operation of calculating the lower bound, and ω is the lower bound.

The problem to find an approximate rank k can be formulated in a definite manner, i.e., finding one matrix X_k with rank k, such that there is no other matrix of rank k whose distance from X is less than the distance from X_k to X. A theorem provided by Eckart and Young [20] proves that the error of approximating a matrix X by X_k satisfies $\|X - X_k\|_F^2 \leq \|X - Y\|_F^2$, where Y is any matrix with rank k, and X_k is the truncated SVD of the matrix X with the ω-rank k, i.e., $X_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$. The ratio $g(k) = \sum_{i=1}^{k} \sigma_i^2 / \sum_{i=1}^{r} \sigma_i^2$, represents the fraction of total variance (Frobenius norm) in X that is explained by its approximated matrix with ω-rank k, i.e., X_k. According to the Principal Components Analysis (PCA), if a matrix has low-rank, its top k singular values should occupy the nearly total variance, i.e., $\sum_{i=1}^{k} \sigma_i^2 \approx \sum_{i=1}^{r} \sigma_i^2$. Fig. 3(a) shows the fraction of total variance captured by the top k singular values for traffic data, where we adopt the one year data, i.e., $n = 465$ and $m = 8760$. Fig. 3(b) also shows the fraction of total variance captured by the top k singular values for air quality monitoring data, where $n = 1518$ and $m = 8040$. We can observe that large fractions of total variance can be covered by a few top singular values. Particularly, the respective top 45 and 83 singular values can capture 99% of total variance for the traffic and sensor datasets, indicating that the data matrix usually has a approximate low-rank structure in practice.



(a) Long-tailed distribution data (b) Gaussian distribution data

Fig. 4. Pilot sampling experiments.

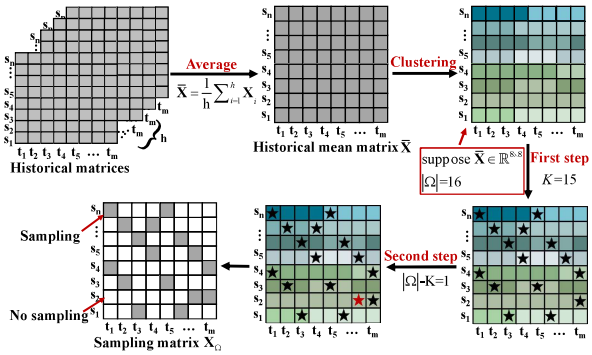


Fig. 5. Sampling scheduling process.

According to the matrix completion theory [21], for most of matrix $X \in \mathbb{R}^{n\times m}$ with low rank r, if a subset of its entries $X_{ij}, (i,j) \in \Omega$ are known ahead, it can be perfectly recovered by solving the following convex optimization problem, i.e.,

$$\begin{aligned} &\min_{\{\Omega\}} \ \|\hat{X}\|_* \\ &\text{subject to } \hat{X}_{ij} = X_{ij}, \ (i,j) \in \Omega, \end{aligned} \tag{4}$$

where $\|\hat{X}\|_*$ is the nuclear norm of the matrix $\hat{X}$. It is the sum of its singular values, i.e., $\|\hat{X}\|_* = \sum_{i=1}^{\min\{n,m\}} \sigma_i$, where $\sigma_i \geqslant 0$ are the singular values of $\hat{X}$. However, there is a prerequisite for $|\Omega|$ (the amount of sampling data), which should satisfies

$$|\Omega| \geq C g^{6/5} r \log g, \tag{5}$$

where C is a numerical constant and $g = \max\{n,m\}$. Therefore, the principle of $|\Omega| \geq C g^{6/5} r \log g$ can be adopted to guide the determination of the amount of sampling data.

C. Takeaways

With the empirical study, we can conclude that: 1) the IoT data matrix usually has low-rank feature, which is the fundamental for sampling; 2) when the IoT data has uneven distribution, the traditional random/uniform sampling approach can lose efficiency, calling for more efficient method; and 3) for low-rank matrix, the matrix completion theory can provide the principle for $|\Omega|$, but the matrix factorization based inference approach can only extract the linear features among data, leaving optimization room for complicated data structure with non-linear features. Inspired by these insights, in what follows, we propose *CODE*, which consists of sampling and inference components, to determine the locations of $|\Omega|$ and estimate the matrix $\hat{X}$, collectively optimizing the data collection problem.

IV. Design of *CODE*

In this section, we elaborate on the design of *CODE* by detailing its two major components, i.e., cluster-based matrix sampling and GAN-based matrix inference.

A. Cluster-Based Matrix Sampling

In this subsection, we focus on determining the specific locations of samples, the goal of which is to achieve as much data information as possible.

Cluster-Based Sampling Matters. In IoT systems, when the observed data values are similar, their locations usually have the underlying correlations, resulting in data redundancy. To deal with it, we propose to cluster the observed values, since data values in the same cluster are more likely to be correlated, which can be leveraged to guide the location determination. To verify the proposal, we then conduct pilot experiments on both Gaussian and non-Gaussian distribution data. Fig. 4 shows the information entropy achieved by different sampling approaches, where the long-tailed distribution data mentioned above, and one Gaussian distribution data in [22] are used. In the clustering approach, the samples are distributed evenly for each cluster. We can observe that for non-Gaussian distribution data, there is a significant performance gap between the clustering and uniform/random approach, while for Gaussian distribution data, their corresponding entropy are quite close. However, in most IoT data systems, the observed data values do not follow perfect uniform or Gaussian distributions. Therefore, in what follows, we cast our cluster-based sampling approach.

The Two-step Sampling Approach. As shown in Fig. 5, the sampling approach works as follows. Given a set of historical matrices $X_1, X_2, ..., X_h$ with $(n \times m)$, we first calculate the historical mean matrix $\bar{X} = \frac{1}{h}\sum_{i=1}^{h} X_i$. For the matrix $\bar{X}$, we then conduct the k-means clustering for each row values, i.e., $\{\bar{X}_{i,1}, \bar{X}_{i,2}, ..., \bar{X}_{i,m}\}$. Let k_i denote the number of clusters which is independently learned by each row according to its own data distribution, and $g^i = \{g_1^i, g_2^i, ..., g_p^i, ..., g_{k_i}^i\}$ denote the set of clusters for the i-th row. The $|\Omega|$ locations are determined by the following two-step sampling. In the first step, for each cluster in each row, we choose one sample, and thus a total $K = \sum_{i=1}^{n} k_i$ samples are selected. For the remaining $|\Omega| - K$ samples, we distribute them according to the cluster error of each cluster, since the larger the clustering error is, the higher diversity of the data within the cluster, calling for more samples. Particularly, for all clusters (i.e., $\forall g_p^i$) in the list, they are ranked by their clustering errors in descending order. Then, one sample is distributed to the first cluster with the largest error, and the cluster is then deleted from the list. The process repeats until the $|\Omega|$ samples are used up. Note that, during the two-step sampling, when determining

the specific (i, j) for one sample in the cluster, the sampling frequency of the time slot index j is used. To be specific, we maintain a list $\{q_1, q_2, ..., q_j, ..., q_m\}$ to record how frequency that the time slot j has been sampled. If one element $\bar{X}_{i,j}$ in g_p^i is sampled, then the count of q_j adds one. After that, when to determine a new sample (i, j) in the cluster g_p^i, the element (i, j') is selected if q'_j is the smallest count in the candidate elements. Fig. 5 shows an example of two-step sampling approach with $|\Omega| = 16$ and $K = 15$, where the first step is to determine the first 15 locations in accordance with each cluster, and the second step is to determine the remaining 1 location based on the clustering errors.

B. GAN-Based Matrix Inference

In this subsection, we devise a GAN model to learn the spatio-temporal data correlations for matrix inference, where the generator network $\mathcal{G}$ attempts to approximate the distribution of true samples, while the discriminator network $\mathcal{D}$ distinguishes the real samples from the fake ones to update the model weights of $\mathcal{G}$ and $\mathcal{D}$.

Pre-Completion Matrix Input. For general GAN models, random noises are usually input into the generator, based on which the generator can extract the underlying features of noise for plausible data generation. The noise randomness for data sampling can make the GAN model act like an "artist" with creativity, where the plausible data (e.g., pictures) looks real but can be utilized without restrictions. However, the high variance of the generated outputs is not suitable for matrix inference, since its goal is the reconstruction accuracy rather than the originality and creativity. Therefore, instead of using the random noise as input, our GAN model takes the incomplete sampling matrix (i.e., X_Ω) as the generators input so as to reduce the predictive uncertainty. In addition, for empty locations, we conduct the pre-completion operation based on the clustering results. Particularly, within each cluster, the empty data is replaced by the empirical mean value of the sampled data instead of zero, and the pre-completion matrix is denoted by X_s $(n \times m)$. Although the mean value is not the same as the true value, it can reflect the basic feature of original value to some extent, which is more effective than starting training from the value of zero. The pre-completion operation can reduce the training time effectively and improve the inference accuracy accordingly.

Generator ($\mathcal{G}$). As shown in Fig. 6, the pre-completion matrix is first fed into a convolutional neural network (CNN), which includes multiple convolutional, batch normalization and activation layers to extract the features of the pre-completion matrix input. After the l-th convolutional layer, the l-th extracted feature map X_s^l can be expressed as

$$\begin{aligned} X_s^l &= \mathcal{F}\left(X_s^{l-1} * k^l + b_k^l\right), \quad l = 2, 3, \ldots, L, \\ X_s^1 &= \mathcal{F}\left(X_s * k^1 + b_k^1\right), \end{aligned} \tag{6}$$

where k^l denotes the convolution filter, $*$ represents the convolution operator, and b_k^l is a bias. Besides, $\mathcal{F}(\cdot)$ is a non-linear activation function ReLU, which can be mathematically

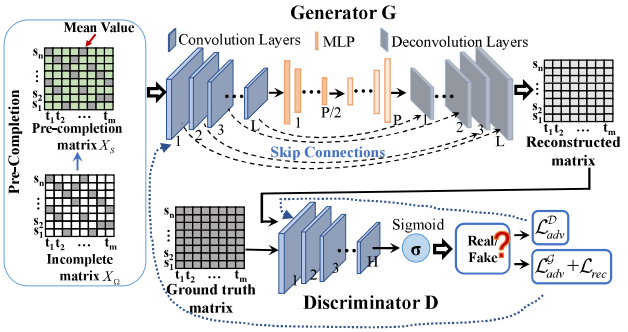


Fig. 6. Architecture of the GAN model.

represented by $\mathcal{F}(x) = \max(0, x)$ [23], and L is the total number of convolutional layers.

The extracted convolutional feature maps, i.e., X_s^l are then processed by a multi-layer perceptron (MLP) component [24]. The goal is to achieve their respective condensed feature vectors, V_s^p, which can be calculated by

$$\begin{aligned} V_s^p &= \mathcal{F}\left(\left\langle V_s^{p-1}, W_s^p \right\rangle + b_s^p\right), \quad p = 2, 3, \ldots, P, \\ V_s^1 &= \mathcal{F}\left(\left\langle X_s^L, W_s^1 \right\rangle + b_s^1\right), \end{aligned} \tag{7}$$

where P is the number of layers in MLP, (W_s^p, b_s^p) are the weight and bias parameters, respectively, $\langle \cdot, \cdot \rangle$ represents the inner product operator, and $\mathcal{F}(\cdot)$ means the ReLU non-linear activation function.

To recover the matrix, the deeply merged feature vector V_s^P should be expanded to contain the feature maps in their original sizes, which can be accomplished by the deconvolution operation (inverse to the convolution), mapping one single input activation to multiple outputs [25]. In deconvolution operation, a special designed filter is used to convolve with the single input. Specifically, we can convolve the acquired $(l-1)$-th layer feature map with a special filter d^l to restore the l-th layer feature map [26], i.e.,

$$\begin{aligned} \widehat{X}^l &= \mathcal{F}\left(\widehat{X}^{l-1} * d^l + b_d^l\right), \quad l = 2, 3, \ldots, L, \\ \widehat{X}^1 &= \mathcal{F}\left(V_s^P * d^1 + b_d^1\right), \end{aligned} \tag{8}$$

where L is the total number of deconvolution layers, being equivalent to the number of convolution layers to make the convolution and deconvolution structure symmetric. In the deconvolution operation, d^n is the duplicate filter of k^{L+1-l} in the $(L+1-n)$-th convolutional layer, and b_d^l is a bias term.

For the learning model, both the convolution and deconvolution structures are made up of multiple layers, which are valuable to feature learning for deep neural networks. However, when the network has more deeper layers, vanishing and exploding gradients are more likely to happen, posing training difficulties [27]. The problem happens since there are deep residual networks. To deal with it, skipping connections between mirrored layers are proposed in the autoencoder framework, which have achieved encouraging results in numerous applications, such as translation and image restoration [28], [29]. Inspired by the success, we propose

to link the corresponding convolutional and deconvolutional layers with a new skip-connection module. For instance, the l-th convolutional feature map X_s^l is directly linked to its mirrored deconvolutional feature map $\widehat{X}^{L+1-l}$ via skipped connections. Afterwards, they are concatenated together and delivered to the next deconvolutional layer. Therefore, formula (8) can be rewritten as:

$$\begin{aligned}\widehat{X}^n &= \mathcal{F}\left(\left[X_s^{L+1-l}, \left(\widehat{X}^{l-1} * d^l + b_d^l\right)\right]\right), l = 2, 3, \ldots, L,\\ \widehat{X}^1 &= \mathcal{F}\left(\left[X_s^L, \left(V_s^P * d^1 + b_d^1\right)\right]\right).\end{aligned} \tag{9}$$

With skipped shortcuts connecting the symmetrical convolution and deconvolution structures, hierarchical feature representations can be passed from the convolution side to their counter deconvolution side directly, and for back propagation, gradients can be directly propagated to the bottom layers.

Finally, the last output of deconvolutional layers, i.e., $\widehat{X}^L$ is used as the generators output $\mathcal{G}_{\text{out}}$, i.e.,

$$\mathcal{G}_{\text{out}} = \widehat{X}^L = \hat{X}. \tag{10}$$

Discriminator ($\mathcal{D}$). The discriminator is designed to determine whether the input matrix is a ground-truth matrix or a generated matrix. It works like a binary classifier: $y \to \{0, 1\}$, where 0 and 1 are the classification labels indicating the input matrix is fake or real [30]. The design of discriminator structure is similar as generator with empirical principles, which consists of multiple convolutional layers as well, and a sigmoid function $\sigma(\cdot)$ is followed to calculate the discriminator result. The training process can be benefited from such design in stabilization and acceleration [31]. Particularly, the convolution output of the l-th layer can be achieved by

$$\begin{aligned}D^l &= \mathcal{F}'\left(D^{l-1} * f^l + b_f^l\right), \quad l = 2, 3, \ldots, H,\\ D^1 &= \mathcal{F}'\left(D * f^1 + b_f^1\right),\end{aligned} \tag{11}$$

where f^l and b_f^l denotes the convolution filter and bias, respectively, and H is the convolutional layers number. In addition, $\mathcal{F}'(\cdot)$ is the Leaky ReLU activation function [32], i.e., $\mathcal{F}'(x) = x$ if $x \geq 0$, otherwise $\mathcal{F}'(x) = px$, where $p \in [0, 1)$. D is the input of discriminator, which can be either the generator output $\mathcal{G}_{\text{out}}$ (i.e., $\widehat{X}^L$) or ground-truth matrix X.

Finally, passing the last-layer feature vector D^H through a sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$, we can get the output of discriminator, i.e.,

$$\mathcal{D}_{\text{out}} = \sigma\left(D^H\right), \tag{12}$$

which can be utilized to perform the classification.

Model Training. When training the GAN model, the min-max game between the generator and discriminator can be characterized as follows [33],

$$\begin{aligned}\min_{\{\theta_\mathcal{G}\}}\max_{\{\theta_\mathcal{D}\}} V(\mathcal{D}, \mathcal{G}) = &\min_{\{\theta_\mathcal{G}\}}\max_{\{\theta_\mathcal{D}\}} \left[\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log \mathcal{D}(x)]\right.\\ &\left.+\mathbb{E}_{z\sim p_z(z)}[\log(1-\mathcal{D}(\mathcal{G}(z)))]\right],\end{aligned} \tag{13}$$

where $\theta_\mathcal{G}$ and $\theta_\mathcal{D}$ are the parameters of the generator $\mathcal{G}$ and discriminator $\mathcal{D}$, respectively. The first part $\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log \mathcal{D}(x)]$ represents the probability that classifying the real data as true, and the second part $\mathbb{E}_{z\sim p_z(z)}[\log(1-\mathcal{D}(\mathcal{G}(z)))]$ represents the probability that classifying the generated data as fake. Instead of solving this intractable min-max optimization problem, we split (13) into two parts, i.e., $V_{\text{GAN}}(\mathcal{G})$ and $V_{\text{GAN}}(\mathcal{D})$, and then attempt to minimize both of them alternately. Hence, the objective function can be transformed by

$$\begin{cases}\min\limits_{\{\theta_\mathcal{D}\}} V_{\text{GAN}}(\mathcal{D}) = \min\limits_{\{\theta_\mathcal{D}\}} \mathcal{L}_{adv}^\mathcal{D}\\ \min\limits_{\{\theta_\mathcal{G}\}} V_{\text{GAN}}(\mathcal{G}) = \min\limits_{\{\theta_\mathcal{G}\}} \left(\lambda_\mathcal{G}\mathcal{L}_{adv}^\mathcal{G} + \lambda_r\mathcal{L}_{rec}\right)\end{cases} \tag{14}$$

with

$$\begin{cases}\mathcal{L}_{adv}^\mathcal{D} = &\mathbb{E}_X\left[\log\left(1-\mathcal{D}\left(X\right)\right)\right]\\ &+\mathbb{E}_{X_s}\left[\log\left(0-\mathcal{D}\left(\mathcal{G}\left(X_s\right)\right)\right)\right]\\ \mathcal{L}_{adv}^\mathcal{G} = &\mathbb{E}_{X_s}\left[\log\left(1-\mathcal{D}\left(\mathcal{G}\left(X_s\right)\right)\right)\right]\\ \mathcal{L}_{\text{rec}} = &\frac{1}{N}\left\|X-\hat{X}\right\|_2\end{cases} \tag{15}$$

where $\mathcal{L}_{adv}^\mathcal{D}$ and $\mathcal{L}_{adv}^\mathcal{G}$ are the adversarial loss for the discriminator and generator, respectively, and $\mathcal{L}_{\text{rec}}$ is the reconstruction loss term. Besides, $\lambda_\mathcal{G}$ and λ_r are the corresponding weighting constants, and N is the total number of samples in input matrix, i.e., $n \times m$. By this means, the gradients of generator for both adversarial loss functions can be optimized towards the same direction.

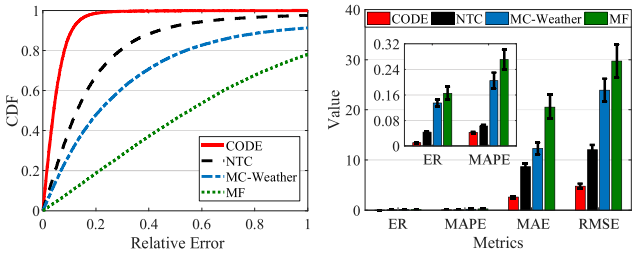
When training the GAN model, the adversarial generator $\mathcal{G}$ and discriminator $\mathcal{D}$ are jointly optimized using alternated stochastic gradient descent (SGD). The gradients are calculated by the Adam optimizer [34], and then backpropagated from the last layer to the input layer. In the adversarial game of GAN model, $\mathcal{G}$ and $\mathcal{D}$ are trained alternately (i.e., when one network is trained, another one is fixed), such that $\mathcal{G}$ can be stronger in generating more precise data, and $\mathcal{D}$ can also be enhanced in differentiating the real and fake samples.

V. Performance Evaluation

In this section, we conduct extensive data-driven experiments to evaluate the performance of *CODE*, check the impact of sampling ratio, and examine the robustness of *CODE* when deploying in different IoT data systems.

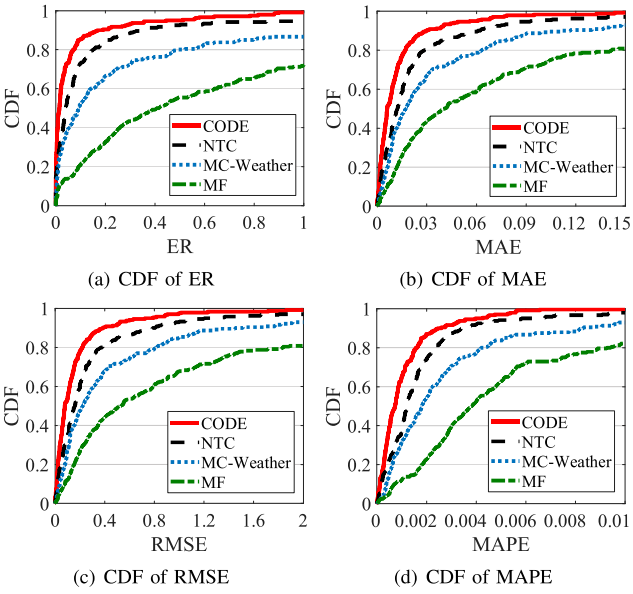
A. Evaluation Methodology

Experiment Setup. We develop *CODE* in Python, and implement the core functions based on PyTorch, which is an open-source machine learning framework. The experiments are carried out on a server with 4 CPUs each containing 192 Intel(R) Xeon(R) Platinum 8260 CPU @ 2.40GHz with 24 cores, and one graphics processing unit card (NVIDIA Tian X) is used to accelerate the training process. For exemplary data system, we adopt the provincial highway traffic data to conduct the performance comparison, while other dataset evaluations are carried out at the last subsection for robustness evaluation. Particularly, the highway traffic system contains 465 stations with about 1.6 millions daily transitions, and we use a subset of four-month data, i.e., 16 weeks data. The traffic data is calculated on an hourly basis, and the traffic matrix



(a) CDFs of RE (b) Average performance

Fig. 7. System overall performance.



(a) CDF of ER (b) CDF of MAE

(c) CDF of RMSE (d) CDF of MAPE

Fig. 8. CDFs of performance at different stations.

can be the 465×2688 scale. In addition, for performance comparison, the scale of X in *CODE* is set to be 465×168 (i.e., $m = 465, n = 168$), covering one week in temporal. Under the set, the first 4 weeks data is used for training while the remaining 12 weeks data is used for testing.

Metrics. Denote by X_{ij} and $\hat{X}_{ij}$ the raw data and inferred data, at (i, j)-th element of the traffic matrix X and estimated matrix $\hat{X}$, respectively, where $1 \leq i \leq n$ and $1 \leq j \leq m$. Ω is the sampled entries, $\bar{\Omega}$ is the unsampled entries, and $\Omega + \bar{\Omega}$ represents the total data entries. Denote by N the total number of data entries, i.e., $N = |\Omega + \bar{\Omega}|$. For performance comparison, the following five metrics are adopted, which are the most popular metrics used to evaluate prediction performance.

- **Relative Error (RE):** refers to the relative error between each raw value and the inferred one, calculated by $\frac{|X_{ij} - \hat{X}_{ij}|}{X_{ij}}$ $((i, j) \in \bar{\Omega})$.
- **Error Ratio (ER):** refers to the error ratio of total data, calculated by $\frac{\sqrt{\sum_{i,j \in \bar{\Omega}} \left(X_{ij} - \hat{X}_{ij}\right)^2}}{\sqrt{\sum_{i,j \in \bar{\Omega}} X_{ij}^2}}$.
- **Mean Absolute Error (MAE):** refers to the average absolute error, calculated by $\frac{1}{N} \sum_{i,j \in \bar{\Omega}} \left| X_{ij} - \hat{X}_{ij} \right|$.
- **Root Mean Square Error (RMSE):** refers to the standard deviation of the differences between the raw data values and recovered ones, calculated by $\sqrt{\frac{1}{N} \sum_{i,j \in \bar{\Omega}} \left(X_{ij} - \hat{X}_{ij} \right)^2}$.
- **Mean Absolute Percentage Error (MAPE):** refers to the mean absolute percentage error, calculated by $\frac{1}{N} \sum_{i,j \in \bar{\Omega}} \left| \frac{X_{ij} - \hat{x}_{ij}}{X_{ij}} \right|$.

Benchmarks. To evaluate the performance of *CODE*, the following three benchmarks are adopted.

- **NTC [22]:** is usually adopted in the literature to evaluate the efficiency of matrix inference approach, which is the state-of-the-art data reconstruction algorithm based on deep learning, while for matrix sampling, the random sampling approach is adopted.
- **MF [35]:** is usually adopted in the literature to evaluate the efficiency of matrix inference approach, which is a conventional inference algorithm based on low-rank matrix factorization. Likewise, in the scheme of *MF*, the random sampling approach is adopted.
- **MC-Weather [36]:** is an online data collection scheme based on matrix completion theory, in which the UTSCS (i.e., Uniform Time-Slot and Cross Sample) model is designed for matrix sampling, i.e., sampling uniformly in time and crossly in location. For matrix inference, the above *MF* approach is adopted by inferring the unsampled data based on low-rank matrix factorization.

B. Performance Comparison

We first evaluate the system overall performance, where the sampling ratio is fixed to 20%, i.e., $|\Omega| = N * 0.2 = 15,624$. Fig. 7(a) shows the CDF of all relative errors (i.e., one location in the matrix is a sample), and we can observe that *CODE* can outperform other benchmarks significantly. Particularly, given the percentile of 80%, the ERs of *NTC*, *MC-Weather*, and *MF* reach about 0.35, 0.5, and 1, respectively, while the scheme of *CODE* only achieves a score of 0.08, improving the performance by 77.1%, 84%, and 92%, respectively. Fig. 7(b) shows the average performance of other four metrics with error bars (i.e., one-week matrix results are calculated as one sample), and we can make the following two statements. First, for all metrics, *CODE* can achieve the superior performance and the gaps are significant. For instance, for the metric of MAPE, the average score is about 0.04, 0.06, 0.2, and 0.27 in *CODE*, *NTC*, *MC-Weather*, and *MF*, respectively, improving the performance by 33.3%, 80%, and 85.2%, respectively. In addition, the four schemes can achieve the score of RMSE about 4.8, 12, 23.9, and 29.7, respectively, with the performance improvement ratio about 60%, 79.9%, and 83.8%, respectively. Note that, for the *NTC* scheme, the performance improvement ratio is enlarged from 33.3% to 60% for the MAPE and RMSE, which means that there are more large inference errors when compared to our proposed scheme. Second, when observing the error bars for all metrics, we can find that the deviation achieved by *CODE* is much

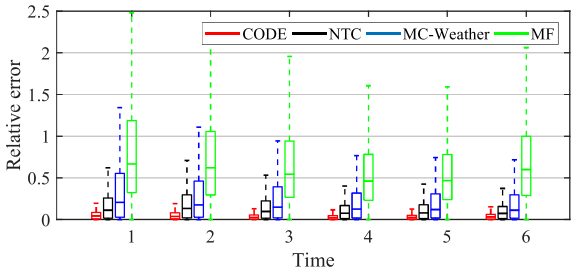


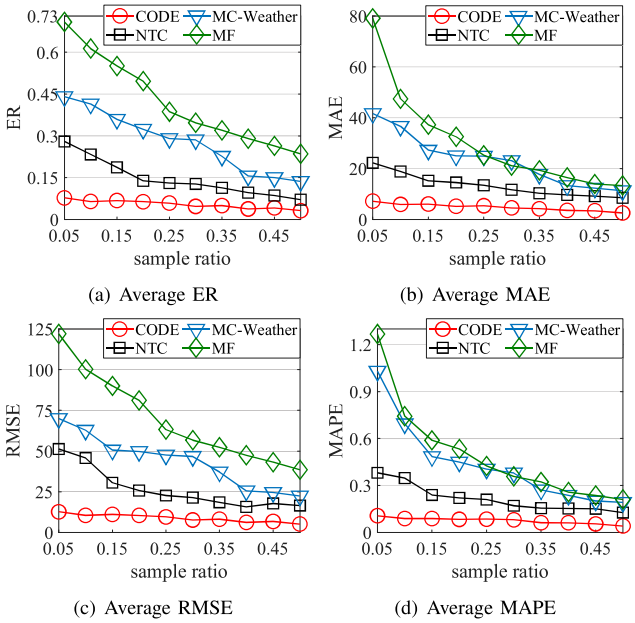
Fig. 9. The RE performance vs. time.

smaller when compared to other benchmarks, demonstrating its reliability to guarantee the performance.

With the overall performance guarantee, we then check its spatio-temporal performance, i.e., how it performs at different stations with time. Fig. 8 shows the CDF results at stations (i.e., the items at each station is calculated as one sample) for different metrics. It can be seen that *CODE* can outperform the other three benchmarks significantly for all metrics. For instance, for 80% stations, their ER, MAE, RMSE, and MAPE can be smaller than 0.09, 0.017, 0.2, and 0.0018 when adopting our proposed scheme of *CODE*, which is enlarged to 0.18, 0.029, 0.39, and 0.002 (when adopting *NTC*), 0.5, 0.06, 0.8, and 0.004 (when adopting *MC-Weather*), and 2.25, 0.135, 1.8, and 0.0095 (when adopting *MF*), respectively. Fig. 9 shows the temporal box-plots of RE, where one day is divided into six temporal zones. We can have the following three major observations. First, under all temporal zones, *CODE* can achieve the supreme performance with significant gaps. Second, when observing the 25^{th}, 50^{th}, 75^{th}, and 100^{th} percentiles, their gaps are quite small by adopting *CODE*. However, the percentiles can have a large deviations when adopting other benchmarks, demonstrating the stability of *CODE*. Finally, with the time evolving, the performance of *CODE* remains stable while other schemes can degrade or improve dramatically, further demonstrating its superiority.

C. Impact of Sampling Ratio

After guaranteeing the performance, we then examine the impact of sampling ratio. Particularly, we vary the sampling ratio from 0.05 to 0.5 with the step of 0.05, and plot the average metric performance by adopting the different schemes in Fig. 10. It is obvious that *CODE* can outperform all the benchmarks under all sampling ratio conditions. In addition, the inference error usually decreases with sampling ratio for all schemes. The difference is that when adopting *CODE*, the error can be bounded within a small range even with a quite small sampling ratio, while for other benchmarks, larger sampling ratio is required to reach a satisfied performance. Taking the metric of MAE as an example, when adopting *CODE*, the score can be smaller than 10 just with the sampling ratio of 0.05, but to reach the target inference accuracy, about 0.35, 0.5, and 0.5 sampling ratio is required for the scheme of *NTC*, *MC-Weather*, *MF*, respectively. It means that to reach the same inference accuracy, *CODE* can save the data collection cost by 85.7%, 90%, and 90%, respectively.



(a) Average ER (b) Average MAE

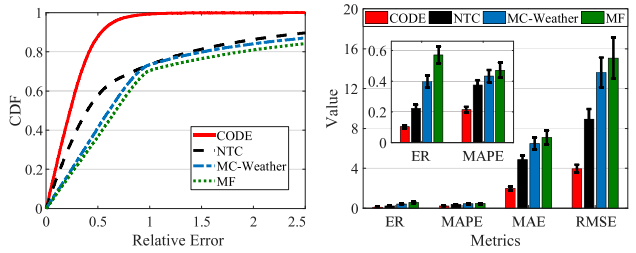
(c) Average RMSE (d) Average MAPE

Fig. 10. The average performance vs. sampling ratio.

D. Robustness Experiments

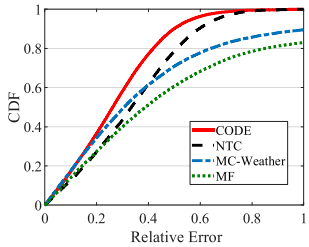
To verify the robustness of *CODE*, i.e., whether it can still work efficiently under other IoT data systems, we adopt another two public IoT datasets to evaluate its performance. One is public WiFi system data[5], and another one is public air quality sensor network which is mentioned above. Particularly, in the WiFi system, $3,600$ APs are used, and the observed value is the number of users associated to the AP within the time slot. In addition, the duration of each time slot is set to be one hour and the target matrix lasts one week in temporal, i.e., $m = 3600, n = 168$. We adopt six weeks data, where the first two weeks data are used for training while the remaining four weeks data are used for testing. Fig. 11(a) shows the CDFs of RE, and Fig. 11(b) shows the average performance of other four metrics achieved by different schemes in the WiFi system. Fig. 11(c) and Fig. 11(d) shows the performance in the air quality sensor network. Observing from these two public IoT data systems, we can achieve the similar observations, e.g., *CODE* outperforming the other benchmarks significantly. It is worth noting that the performance advantage of *CODE* drops when evaluating from highway ETC system to WiFi system with MAPE rising from 5% to 20% in Fig. 7 and Fig. 11. It is normal since different IoT datasets have diverse distribution features, and it means that 20% sampling ratio is not enough for WiFi system, whereas *CODE* can still perform the best in all benchmarks. During actual operation, the inference accuracy can be enhanced by improving the sampling ratio.

[5]https://github.com/Intelligent-WiFi/DataSet

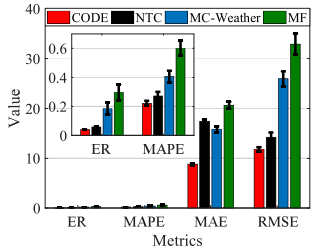


(a) CDFs of RE of WiFi system

(b) Average performance of WiFi system



(c) CDFs of RE of sensor network



(d) Average performance of sensor network

Fig. 11. Robustness experiments.

VI. Related Work

We review the related work by two categories, i.e., matrix sampling and matrix inference.

A. Matrix Sampling

The explosive growth of data has posed huge burden on the sensing and storage of IoT data system, which motivates new data sampling studies. There have been three kinds of sampling schemes in the literature. The most common approach is random sampling. In [21], Candès and Recht proved that for an $n_1 \times n_2$ matrix with rank r, it can be recovered by at least $Cn^{6/5}r\log n$ random samples, where $n = max(n_1, n_2)$. Whereafter, a large number of studies have attempted to recover the missing data based on random sampling approaches [22], [37]. The second type of approach is uniform sampling, which aims to select the same number of samples from temporal and spatial domains. In [36], Xie *et al.* proposed an on-line data collection scheme based on uniform sampling, to adaptively sample different locations according to the UTSCS model. The last one is active sampling strategy. Unlike random/uniform sampling approaches, there are quite few studies [38]–[40] that investigate active sampling strategies to determine the sampling locations. CCS-TA [41] designed a leave-one-out and re-sampling principle to reduce the total number of samples by taking multiple-steps sampling. Wang *et al.* [42] designed an SPACE-TA framework to actively determine sampling locations which have large estimated uncertainties or errors.

All those works focus on the numbers of samples rather than the sample qualities. Besides, they can work well on Gaussian data but perform poorly on non-Gaussian data. Our approach differs from them in that, we take advantage of the difference in the amount of information that each location contains, and then determine the sampling locations with a non-uniform manner to select a sample set with much more data information to benefit the inference.

B. Matrix Inference

Although there have been some studies on data inference based on sparse sensing technologies, including compressive sensing [15], matrix completion [16], and tensor completion [17], they mainly rely on the mathematical modeling methods based on the underlying linear features, but rarely leverage the deep learning models to extract non-linear features. In the literature, there are only a few works that have used deep learning models. He *et al.* [43] designed a general framework, named NCF, to express and generalize matrix factorization with a neural network. Xie *et al.* [22] proposed a scheme, named Neural Tensor Completion (NTC), which uses 3D CNN to extract the hidden features in order to improve the missing data inference accuracy.

In summary, conventional data completion methods usually ignore the extraction of non-linear features, and they aim to infer the missing data under the assumption that the sparse data is known ahead. Therefore, they pay more attention to improving the inference accuracy of missing data, while neglecting the data sampling process. However, the sample qualities can affect the data inference performance significantly, but there is limited study available to tackle the deep-coupled problem, which motivates our research in this paper.

VII. Conclusion

In this paper, we have proposed *CODE*, a systematical data collection framework for large-scale IoT data sytems, which can reduce the data collection cost significantly without affecting the data benefits. The reference implementation of *CODE* has been conducted under three operational IoT data systems, and extensive data-driven experiments have been conducted to demonstrate its efficiency and robustness. For the future work, considering the accumulations of inference error with time, we will investigate the model retraining problem to further enhance the temporal robustness of *CODE*.

Acknowledgement

This research was supported in part by National Natural Science Foundation of China under Grants 62002389, 62102223, 62122095, and 62072472, 111 project (No. B18059), China Postdoctoral Science Foundation under Grant 2021M691785, Young Elite Scientist Sponsorship Program by CAST under Grant YESS20200238, Natural Science Foundation of Hunan Province of China under Grant 2021JJ20079 and 2020JJ2050, the Key Research and Development Project of Hunan Province of China under Grant 2022GK2013, Young Talents Plan of Hunan Province of China under Grant 2021RC3004 and 2019RS2001, and Natural Sciences and Engineering Research Council (NSERC) of Canada.

REFERENCES

[1] Y. Ding, Y. Yang, W. Jiang, Y. Liu, T. He, and D. Zhang, "Nationwide Deployment and Operation of a Virtual Arrival Detection System in the Wild," in *Proc. ACM SIGCOMM'21*, 2021.

[2] D. Zhang, "Mobile Cyber-Physical Systems for Smart Cities," in *Proc. ACM WWW'20*, 2020.

[3] F. Lyu, J. Ren, N. Cheng, P. Yang, M. Li, Y. Zhang, and X. Shen, "LEAD: Large-Scale Edge Cache Deployment Based on Spatio-Temporal WiFi Traffic Statistics," *IEEE Trans. Mob. Comput.*, vol. 20, no. 8, pp. 2607–2623, 2021.

[4] N. Cheng, F. Lyu, J. Chen, W. Xu, H. Zhou, S. Zhang, and X. Shen, "Big Data Driven Vehicular Networks," *IEEE Netw.*, vol. 32, no. 6, pp. 160–167, 2018.

[5] Y. Yang, X. Xie, Z. Fang, F. Zhang, Y. Wang, and D. Zhang, "VeMo: Enabling Transparent Vehicular Mobility Modeling at Individual Levels with Full Penetration," in *Proc. ACM MobiCom'19*, 2019.

[6] C. Shi, J. Liu, H. Liu, and Y. Chen, "WiFi-Enabled User Authentication through Deep Learning in Daily Activities," *ACM Trans. Internet Things*, vol. 2, no. 2, pp. 1–25, 2021.

[7] Y. Deng, F. Lyu, J. Ren, H. Wu, Y. Zhou, Y. Zhang, and X. Shen, "AUCTION: Automated and Quality-Aware Client Selection Framework for Efficient Federated Learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1996–2009, 2022.

[8] Y. Deng, F. Lyu, J. Ren, Y.-C. Chen, P. Yang, Y. Zhou, and Y. Zhang, "FAIR: Quality-aware federated learning with precise user incentive and model aggregation," in *Proc. IEEE INFOCOM'21*. IEEE, 2021, pp. 1–10.

[9] K. Li, C. Yuen, B. Kusy, R. Jurdak, A. Ignjatovic, S. S. Kanhere, and S. Jha, "Fair Scheduling for Data Collection in Mobile Sensor Networks with Energy Harvesting," *IEEE Trans. Mob. Comput.*, vol. 18, no. 6, pp. 1274–1287, 2019.

[10] M. Ibrahim, H. Harb, A. Mansour, A. Nasser, and C. Osswald, "All-in-one: Toward hybrid data collection and energy saving mechanism in sensing-based IoT applications," *Peer-to-Peer Networking and Applications*, vol. 14, no. 3, pp. 1154–1173, 2021.

[11] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu, "Spatio-Temporal Compressive Sensing and Internet Traffic Matrices (Extended Version)," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 662–676, 2012.

[12] P. Tong, M. Li, M. Li, J. Huang, and X. Hua, "Large-Scale Vehicle Trajectory Reconstruction with Camera Sensing Network," in *Proc. ACM MobiCom'21*, 2021, pp. 188–200.

[13] J. Liu, H. Liu, Y. Chen, Y. Wang, and C. Wang, "Wireless Sensing for Human Activity: A Survey," *IEEE Commun. Surv. Tutorials*, vol. 22, no. 3, pp. 1629–1645, 2020.

[14] A. M. Avila and I. Mezi, "Data-driven analysis and forecasting of highway traffic dynamics," *Nat Commun*, vol. 11, no. 1, pp. 2090–2105, 2020.

[15] E. J. Candès *et al.*, "Compressive sampling," in *Proc. ICM'06*, vol. 3, 2006, pp. 1433–1452.

[16] R. H. Keshavan, S. Oh, and A. Montanari, "Matrix completion from a few entries," in *Proc. IEEE ISIT'09*, 2009, pp. 324–328.

[17] Y.-B. Zheng, T.-Z. Huang, X.-L. Zhao, Q. Zhao, and T.-X. Jiang, "Fully-Connected Tensor Network Decomposition and Its Application to Higher-Order Tensor Completion," in *Proc. AAAI'21*, vol. 35, no. 12, 2021, pp. 11 071–11 078.

[18] M. Babar and F. Arif, "Smart Urban Planning Using Big Data Analytics Based Internet of Things," in *Proc. ACM UbiComp '17*, 2017, pp. 397–402.

[19] I. Markovsky, *Low Rank Approximation - Algorithms, Implementation, Applications*. Springer, 2012.

[20] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[21] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *CoRR*, vol. abs/0805.4471, 2008.

[22] K. Xie, H. Lu, X. Wang, G. Xie, Y. Ding, D. Xie, J. Wen, and D. Zhang, "Neural Tensor Completion for Accurate Network Monitoring," in *Proc. IEEE INFOCOM' 20*, 2020, pp. 1688–1697.

[23] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proc. ICML'10*, 2010, pp. 807–814.

[24] T. Yu, X. Li, Y. Cai, M. Sun, and P. Li, "S2-MLP: Spatial-Shift MLP Architecture for Vision," in *Proc. IEEE/CVF WACV'22*, 2022, pp. 297–306.

[25] H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation," in *Proc. IEEE ICCV'15*, 2015, pp. 1520–1528.

[26] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE CVPR'10*, 2010, pp. 2528–2535.

[27] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS'10*, 2010, pp. 249–256.

[28] X.-J. Mao, C. Shen, and Y.-B. Yang, "Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections," in *Proc. NIPS'16*, 2016, pp. 2810–2818.

[29] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *Proc. IEEE CVPR'17*, 2017, pp. 5967–5976.

[30] Z. Li, J. Cao, H. Wang, and M. Zhao, "Sparsely Self-Supervised Generative Adversarial Nets for Radio Frequency Estimation," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2428–2442, 2019.

[31] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," in *Proc. ICLR'16*, 2016.

[32] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," in *Proc. ICML WorkshopDeep Learn.*, 2013.

[33] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Proc. NIPS'14*, 2014, pp. 2672–2680.

[34] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. ICLR'15*, 2015.

[35] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[36] K. Xie, L. Wang, X. Wang, G. Xie, and J. Wen, "Low Cost and High Accuracy Data Gathering in WSNs with Matrix Completion," *IEEE Trans. Mob. Comput.*, vol. 17, no. 7, pp. 1595–1608, 2018.

[37] Y.-C. Chen, L. Qiu, Y. Zhang, G. Xue, and Z. Hu, "Robust Network Compressive Sensing," in *Proc. MobiCom '14*, 2014, pp. 545–556.

[38] J. Silva and L. Carin, "Active Learning for Online Bayesian Matrix Factorization," in *Proc. ACM SIGKDD '12*, 2012, pp. 325–333.

[39] D. J. Sutherland, B. Póczos, and J. Schneider, "Active Learning and Search on Low-Rank Matrices," in *Proc. ACM SIGKDD '13*, 2013, pp. 212–220.

[40] S. Chakraborty, J. Zhou, V. Balasubramanian, S. Panchanathan, I. Davidson, and J. Ye, "Active Matrix Completion," in *Proc. IEEE ICDM'13*, 2013, pp. 81–90.

[41] 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 *Proc. ACM UbiComp '15*, 2015, pp. 683–694.

[42] 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 Trans. Intell. Syst. Technol.*, vol. 9, no. 2, 2017.

[43] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural Collaborative Filtering," in *Proc. WWW'17*, 2017, pp. 173–182.