SATORIS: A Unified Framework for Singular vAlue and TensOR weIght regresSion to Perform Imputation on Spatio-Temporal Traffic Data

We introduce two imputation techniques for spatio-temporal traffic data: Singular Value Regression (SVR) based on a matrix representation and Tensor Weight Regression (TWR). We unify the two approaches under SATORIS, a novel low-rank parameter estimation framework. SATORIS explicitly capitalizes on the temporal invariance of singular vectors and the tensor factors across closely observed data points, a unique observation derived from empirical traffic data, to achieve unification. Using real-world traffic dataset, we show that in terms of accuracy, our techniques markedly surpass traditional baseline methods like mean/median imputation, k-nearest-neighbour imputation under various metrics like tensor completion score (TCS), root mean square error (RMSE) and mean absolute error (MAE). Compared to more competitive methods like *softimpute*, our methods are just as good for missing ratios under 70% and significantly better for higher missing ratios. While recent deep learning methods like Miracle and Gain perform great for low missing percentages, they are outperformed by our methods at higher missing percentages due to a lack of sufficient observations. We consider two versions of both the SVR and TWR techniques, first in which the factors are estimated from completely observed days temporally close to the target day to be imputed - we call this the known factor regime and second in which they are estimated from partial observations of target and its temporally neighboring days, which we call the unknown factor regime. Our results indicate that when the missing data percentage is low, unknown factors regime performs better while for higher missing percentages the known factors regime performs better. Our techniques are the best choice for higher missing percentages and we provide a general guideline for picking the better out of our proposed TWR and SVR technique - TWR is recommended when partial observations are available from temporally proximal days, else SVR should be used.

1 INTRODUCTION

The transportation infrastructure of a city serves as its backbone, providing essential support to various economic sectors and ensuring the well-being of its citizens. It is, therefore, crucial to enhance the efficiency of this system in order to sustain the vitality of the city, not only by reducing commute times for citizens and minimizing freight costs [18], but also by prioritizing safety to prevent any loss of lives. Furthermore, addressing the growing concerns of climate change necessitates improvements in the effectiveness of the transportation sector. Given that road transport is a significant contributor to urban air pollution, modern smart cities also need to implement traffic-shaping measures to maintain good air quality.

To tackle the aforementioned challenges, city administrations worldwide have been extensively deploying traffic sensors that gather real-time and historical traffic data, enabling a comprehensive understanding of the distinct traffic patterns within a city. By leveraging traffic sensors and their data, city authorities can effectively manage traffic and address the various challenges associated with transportation. Due to budget constraints, nonetheless, costly traffic sensors may only be deployed

ACM SIGSMETRICS '24, June 10 - 14, 2024, Venice, Italy

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

	Unknown factors regime	Known factors regime
Matrix (SVR)	Unknown Singular Vector (USV)	Known Singular Vector (KSV)
Tensor (TWR)	Unknown Tensor Factor (UTF)	Known Tensor Factor (KTF)

Table 1. Proposed methods

sparsely. Even in cases where sensors are densely deployed, it is a common occurrence to encounter missing or incomplete traffic data caused by sensor or network failures. From prior literature[29], it is known that a mere 2% of missing data may account for up to 18% of information loss, with information loss reaching 35%-98% for missing data percentages of 10%-35%. Hence it is critical to accurately estimate the missing data, not only for generating informative visualizations that can provide valuable insights but also for ensuring the availability of reliable data for downstream models to utilize.

Motivated by these concerns, we study the problem of interpolating missing spatio-temporal traffic data in a given urban environment. The main contribution of our work is the design of two simple and interpretable low-rank imputation techniques based on matrix and tensor representations of traffic data that we refer to as *singular value regression (SVR)* and *tensor weight regression (TWR)*, respectively. We present these techniques under SATORIS, a unifying framework that includes both techniques. While prior works (including some based on matrix and tensor factorization techniques [1, 2, 8, 9, 22, 25, 27]) have considered the imputation of missing traffic data using particular snapshots of time, a key novelty of our imputation framework is that it explicitly leverages the slow temporal changes (or, temporal invariance) of singular vectors and tensor factors across days of the week. The relative consistency of singular vectors and tensor factors across adjoining weekdays is something we have observed empirically to be a notable characteristic of urban traffic data. It exploits an intuitive aspect of real traffic in that we would naturally expect vehicular densities at a city scale to be similar on adjoining weekdays. Our methods are simple, fast, and efficient in terms of sample complexity, leading to a better imputation from very little data. These stand in contrast to deep learning-based models which have higher complexity, have high compute costs, and require large amounts of data to be trained successfully. While our SVR and TWR techniques lend themselves to a certain level of interpretation in terms of eigenvectors and tensor factors, deep learning methods are often difficult to interpret.

The SVR technique is based on a two-dimensional matrix representation, while TWR is based on higher-dimensional representations. To our knowledge, no prior work has explicitly exploited the temporal invariance of singular vectors and tensor factors over temporally proximate days for imputation. In our study, we examine two variations of both SVR and TWR techniques. The first version assumes that the underlying singular vectors or tensor factors are estimated from the data of adjoining weekdays and hence *known* a priori with respect to the target day's partially observed matrix/tensor. We call this the known factor regime and it consists of two proposed methods called known singular vectors (KSV) method and known tensor factors (KTF) method. The second version considers the scenario where these singular vectors or tensor factors are *unknown*, requiring estimation from scratch based on partial observations from different adjacent days. We call this the unknown factor regime which consists of two methods - unknown singular vectors (USV) and unknown tensor factors (UTF). See Table 1 for an easy overview of the four methods.

Methods in the unknown factor regime are implemented through a novel variation on the alternating least square (ALS) algorithm, and are more sophisticated compared to the known factor regime, but quite useful in practical situations when complete observations are not available and the partial observations are too sparse (high missing ratio).

We evaluate on real traffic data-sets obtained from the cities of Beijing and Shanghai. Our analysis allows us to draw insightful conclusions regarding the relative performance of the techniques under various conditions. The key contributions of this work are as follows:

(i) We propose two novel techniques for spatio-temporal traffic data, one based on singular value decomposition of a 2-D matrix representation (SVR) and one based on a tensor representation (TWR). We further unify both techniques under a common framework called SATORIS, which exploits the observation that the singular vectors and tensor factors are relatively consistent and stable across adjoining days.

(ii) We systematically evaluate the proposed techniques on real urban spatio-temporal traffic data from Beijing and Shanghai and highlight the observed patterns of our algorithms common to both datasets. We show that the proposed techniques substantially outperform baseline methods such as row mean, row median, k-nearest neighbor, Softimpute[22], and two recent deep learning imputation techniques - Gain [34] and Miracle [16] with respect to metrics such as the tensor completion score (TCS)[1, 2, 27], root mean square error (RMSE) and mean absolute error (MAE).

(iii) We find that when the missing data percentage is low, somewhat to our surprise, methods in unknown factor regime perform better than the methods in the known factor regime; otherwise, when complete data is available for temporally proximal days, known factor regime perform better. Additionally, we show that neither SVR nor TWR is a clear winner – interestingly, for low-to-medium missing data percentage, matrix-based SVR performs best, while tensor-based TWR is better for the cases with high missing data percentage, suggesting that the tensor data representation utilized in TWR exhibits greater resilience in scenarios with a high amount of missing data.

2 PRELIMINARIES

2.1 Matrix and Tensor Factorization

We briefly discuss matrix decomposition and focus on tensor decomposition techniques while comprehensive surveys on low-rank tensor decomposition can be found in [14, 26] and see [11] for matrix factorization.

A rank one matrix $\mathbf{H} \in \mathbb{R}^{m \times n}$ is one which can be written as the outer product of two vectors $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^n$. We call a rank one matrix an atom (or more specifically a "matrix atom", to avoid confusion with rank one "tensor atoms" to be introduced, when it is not clear from the context).

$$\mathbf{H} = \mathbf{a} \otimes \mathbf{b}$$

The rank of a matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ is defined as the minimum number of atoms that sum up to it. Equivalently, a rank r matrix can be written as a linear combination of r atoms. Again, equivalently, a rank r matrix can be written as a product of three matrices: $\mathbf{A} \in \mathbb{R}^{m \times r}$, a diagonal matrix $\mathbf{diag}(\boldsymbol{\alpha}) \in \mathbb{R}^{r \times r}$, and $\mathbf{B} \in \mathbb{R}^{r \times n}$ where columns of $\mathbf{A}$ and $\mathbf{B}$ are set to $\mathbf{a_i}$ and $\mathbf{b_i}$ respectively for $i \in [r]$.

$$\mathbf{M} = \sum_{i=1}^{r} \alpha_i \mathbf{H_i} = \sum_{i=1}^{r} \alpha_i (\mathbf{a_i} \otimes \mathbf{b_i}) = \mathbf{A}\ \mathrm{diag}(\boldsymbol{\alpha})\ \mathbf{B}^{\mathbf{T}}$$

Tensor decomposition has recently been popular in this community. [14] and [26] are two excellent expositions on tensor decomposition and their applications. In this paper we discuss only about tensors of order three to build intuition, unless stated otherwise. A rank one tensor, or a "tensor atom" (or simply an atom, if no ambiguity) $\mathbf{H} \in \mathbb{R}^{m \times n \times p}$ is the generalized outer product $\mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c}$ of three vectors $\mathbf{a} \in \mathbb{R}^m$, $\mathbf{b} \in \mathbb{R}^n$ and $\mathbf{c} \in \mathbb{R}^p$.

$$\mathbf{H} = \mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c}$$

Similar to the rank of a matrix, the rank of a tensor is defined as the minimum number of tensor atoms whose linear combination gives us the original tensor. This is called the Canonical Polyadic

(CP) decomposition of a tensor.

$$\mathbf{T} = \sum_{i}^{r} \alpha_i \mathbf{H_i} = \sum_{i}^{r} \alpha_i (\mathbf{a_i} \otimes \mathbf{b_i} \otimes \mathbf{c_i})$$

Just as in the case of matrices, we can store the factors $\mathbf{a_i}$, $\mathbf{b_i}$, $\mathbf{c_i}$ of the tensor $\mathbf{T}$ in the matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, columns of which contain $\mathbf{a_i}$, $\mathbf{b_i}$ and $\mathbf{c_i}$ respectively for $i \in [r]$, as shown in Figure 1.

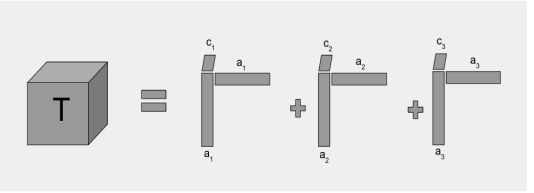

Fig. 1. Tensor Decomposition

2.2 Factorization Algorithm

We describe numerical techniques and algorithms used to decompose matrices and tensors into linear combinations of respective atoms. Since we want to discuss the decomposition of matrices/tensors with missing entries, it is useful to talk about a common factorization algorithm called the Alternating Least Squares (ALS) algorithm [14]. For factorizing a matrix $\mathbf{M}$ as $\mathbf{AB}$, the ALS algorithm fixes $\mathbf{A}$ to estimate $\mathbf{B}$ and then fixes $\mathbf{B}$ to estimate $\mathbf{A}$, alternatively. Estimating the factors simultaneously is a non-convex problem whereas the problem reduces to a least squares fitting when all but one of the factors is fixed. It accomplishes in finding a rank-k approximation of a matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ as given by

$$\mathbf{M} \approx \mathbf{AB^T}$$

where $\mathbf{A} \in \mathbb{R}^{m \times k}$ and $\mathbf{B} \in \mathbb{R}^{n \times k}$ by trying to minimize the objective

$$\underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} ||\mathbf{M} - \mathbf{AB^T}||_F$$

where $||.||_F$ represents the Frobenius norm of a matrix.

The ALS algorithm requires the following parameters:

M: a completely known matrix
k: the rank to be used for decomposition

The ALS algorithm needs to be modified to handle the case when we do have missing entries in the matrix. To that end, we capture the observed entries in the matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ using the binary mask $\mathbf{\Omega} \in \{0, 1\}^{m \times n}$ and define the masked norm of a matrix as

$$\text{maskednorm}(\mathbf{M}, \mathbf{\Omega}) = ||\mathbf{M}||_\Omega = ||\mathbf{M} \circ \mathbf{\Omega}||_F$$

where $\circ$ represents the element-wise multiplication or the Hadamard product. In the case of missing entries in matrix, to find a rank-r approximation to $\mathbf{M}$, we minimize the masked norm which measures the error only on the observed entries,

$$\underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} ||\mathbf{M} - \mathbf{AB^T}||_\Omega$$

The ALS algorithm for a matrix can be extended to third and higher order tensors. The masked tensor norm given a binary mask $\mathbf{\Omega} \in \{0, 1\}^{m \times n \times p}$ is

$$\text{maskednorm } (\mathbf{T}, \mathbf{\Omega}) = ||\mathbf{T}||_\Omega = ||\mathbf{T} \circ \mathbf{\Omega}||_F$$

3 PROBLEM STATEMENT

3.1 Data Representation

Our dataset consists of GPS logs from two different taxi datasets, Shanghai and Beijing, taken over 3 weeks. We discretize the latitude and longitude of the Shanghai region into 40x40 grid cells. Similarly, the Beijing region is discretized into 56x56 grid cells. Time is discretized by hour into 24 slots. For each grid cell and each time slot, we aggregate the density (count) of taxis. Naturally, for each day we get a 3rd order tensor of shape 56x56x24 for Beijing and 40x40x24 for Shanghai where the first two dimensions are associated with the space (latitude and longitude) and the third with time. We also express the same data in matrix representation by zipping the 1st and 2nd spatial dimensions which gives us matrices of shapes 1600x24 for Shanghai and 3136x24 for Beijing. Using both datasets, we would be comparing the performance of the proposed SVR and TWR low-rank imputation methods for matrix and tensor representations.

3.2 Problem Formulation

Given a set of incomplete observations of traffic density, we want to estimate the missing entries as best as possible. Since we are representing the incomplete data in the form of matrices and tensors, we want to complete the missing entries in the matrix/tensor representation. The missing indices in the matrix/tensor are captured by the binary mask $\boldsymbol{\Omega}$, which has the same shape as that of the matrix/tensor. Formally, given a tensor $\mathbf{T}$ (including one of order 2, i.e a matrix) and a binary mask $\boldsymbol{\Omega}$, we predict the entries in $\mathbf{T}$ corresponding to the 0 entries in $\boldsymbol{\Omega}$.

Performance Metrics. We borrow the widely used tensor completion score (TCS) metric from [1, 2, 27]. Let the inverse mask of a binary mask $\boldsymbol{\Omega}$ be defined as $\bar{\boldsymbol{\Omega}} = \mathbf{1} - \boldsymbol{\Omega}$ where $\mathbf{1}$ is the all ones tensor of the same shape as $\boldsymbol{\Omega}$. Let $\tilde{\boldsymbol{T}}$ be the estimated tensor using any imputation algorithm. Then,

$$TCS(\mathbf{T}, \tilde{\mathbf{T}}, \boldsymbol{\Omega}) := \frac{||\mathbf{T} - \tilde{\mathbf{T}}||_{\bar{\Omega}}}{||\mathbf{T}||_{\bar{\Omega}}} = \frac{||(\mathbf{T} - \tilde{\mathbf{T}}) \circ (\mathbf{1} - \boldsymbol{\Omega})||_F}{||\mathbf{T} \circ (\mathbf{1} - \boldsymbol{\Omega})||_F}$$

$$TRE(\mathbf{T}, \tilde{\mathbf{T}}, \boldsymbol{\Omega}) := \frac{||(\mathbf{T} - \tilde{\mathbf{T}}) \circ \boldsymbol{\Omega}||_F}{||\mathbf{T} \circ \boldsymbol{\Omega}||_F}$$

The tensor reconstruction error (TRE) measures error on the observed entries. TRE and TCS can be thought of as the training error and the test error. In addition, we also test the performance of our techniques using mean absolute error (MAE) and root mean squared error (RMSE).

4 METHODOLOGY

We take advantage of the daily and weekly periodicity of traffic time series data in our methods. The daily pattern in the traffic density for 7 days in Beijing and Shanghai can be seen in Figures 2 and 3. The M-shaped daily pattern is commonly observed in traffic density. There is more variation from day to day in Beijing than in Shanghai due to a long weekend leading to lower traffic volume.

When estimating missing values under any framework, certain assumptions need to be made since there must exist pattern or structure in the data in the first place for imputation to be possible, without which the missing entries could be arbitrary. In our imputation methods, we hypothesize that the subspace formed by low-rank tensor (and matrix) atoms does not vary much over temporally close days. This is a reasonable hypothesis which needs to be verified with experiments, which we turn to next.

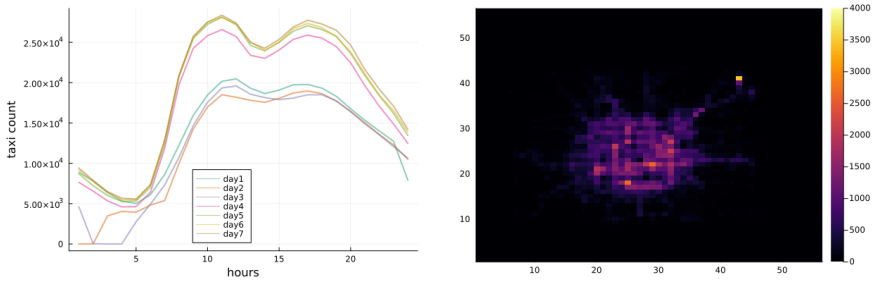

Fig. 2. Beijing traffic density temporal and spatial plot

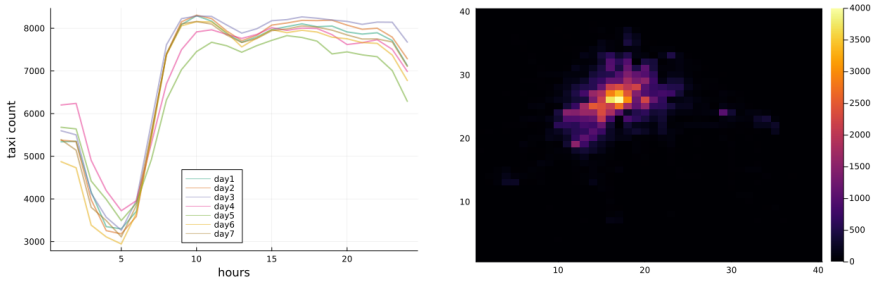

Fig. 3. Shanghai traffic density temporal and spatial plot

4.1 Invariance of Matrix and Tensor Atom Subspaces

We conduct an experiment to look at the amount of invariance in the matrix atoms and tensor atoms when we decompose the matrix $\in \mathbb{R}^{mn\times p}$ (using SVD) and tensor $\in \mathbb{R}^{m\times n\times p}$ (using CP) representations of the same traffic data. For a small k, to *k-atomize* a representation, we do a rank-k decomposition on it to get the k rank-one factors/atoms ($\in \mathbb{R}^{mn\times p}$ or $\mathbb{R}^{m\times n\times p}$) while throwing away the weights corresponding to them.

We pick three consecutive days (for temporal proximity) and k-atomize their tensor and matrix representations. We expect the subspace formed by the k-atoms from the decomposition of a single day to be of dimension k (unless the representation is already low-rank with rank < k). If we flattened the k atoms from the i^{th} day into vectors and put them into k columns of a tall matrix $\mathbf{D_i}$ $\in \mathbb{R}^{mnp\times k}$, we expect this matrix to have rank k. A fundamental feature that is invariant across temporally close days is something we are interested to find. If the atoms capture this fundamental structure, we expect not much of an increase in the subspace dimension when we put all the $3k$ atoms from the three days together as compared to the subspace formed from the k atoms of a single day. Hence a tall matrix $\mathbf{E}$ whose columns are the $3k$ flattened atoms (formed by concatenating the matrices $\mathbf{D_1}, \mathbf{D_2}$ and $\mathbf{D_3}$ containing the atoms as columns from the three consecutive days) must be approximately rank k instead of rank $3k$. We measure this phenomenon using the normalized k-variance (NKVAR) contained in the top k singular values of the matrix $\mathbf{E}$.

$$NKVAR(E) = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{3k} \sigma_i^2} = \frac{\sum_{i=1}^{k} \sigma_i^2}{Tr(\mathbf{E^T E})}$$

We take 7 sets of "consecutive 3 days" data for both Beijing and Shanghai and plot Figure 4 ($NKVAR$ wrt to k). Figure 4 shows that, for matrix representation, more than 75% of the total variance is captured in top-k atoms for up to $k = 5$. For tensor representation, we have more than 90% of the total variance in the top-k atoms even for $k = 30$. The observation of temporal invariance

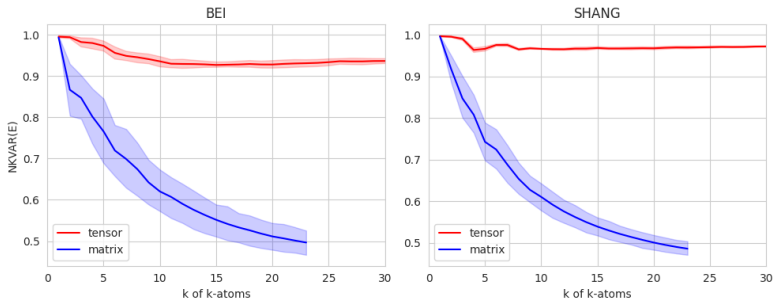

Fig. 4. Normalized variance in top-k singular values of $\mathbf{E}$ vs rank k (for 7 sets of 3 days)

and stability in the subspace formed by the matrix and tensor atoms across several consecutive days of a week provides substantial evidence to support our hypothesis.

We also examine the average angle between the subspaces formed by the tensor and matrix k-atoms of different days, which tells us whether the atom subspaces are well aligned in the case of matrix and tensor factorization over different adjacent days. For $\mathbf{A} \in \mathbb{R}^{y \times a}, \mathbf{B} \in \mathbb{R}^{y \times b}$, let $ssa(\mathbf{A}, \mathbf{B})$ define the average subspace angle formed between the column spaces of A and B. Then we define the average subspace angle between the columnspaces of $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ as

$$AKANG(D_1, D_2, D_2) = \frac{ssa(\mathbf{D_1}, \mathbf{D_2}) + ssa(\mathbf{D_2}, \mathbf{D_3}) + ssa(\mathbf{D_1}, \mathbf{D_3})}{3}$$

We find that, for low values of rank (k), the average angle between the subspaces remains small, further confirming the existence of temporal invariance in traffic datasets. Notably, we observe that this effect is more pronounced for tensor atoms compared to matrix atoms, underscoring that the tensor atoms subspace is more invariant and robust than matrix atom subspace. Figures supporting this can be found in A.1.

This unique finding of temporal invariance in this paper forms the empirical foundation of the SATORIS framework, in which the reconstruction of matrix and tensor structure can be achieved by using partial data across adjoining days. The unique characteristics of temporal invariance in traffic data further reinforce the efficacy and robustness of our proposed SATORIS framework, as later shown in Section 5.

4.2 The SATORIS Framework

We develop a general framework called SATORIS for the proposed methods. Let $\boldsymbol{\theta} \in \mathbb{R}^p$ be the parameters that drive the generating process and generate the data in our observations. These generating processes are specific to the particular data tensor and matrix representations. We partition the parameters into two sets $\boldsymbol{\theta_1} \in \mathbb{R}^{p_1}$ and $\boldsymbol{\theta_1} \in \mathbb{R}^{p_2}$ where $p_1 + p_2 = p$. We define the general estimator which takes in observation (possibly missing and noisy) along with the known part of the parameters $\boldsymbol{\theta_1}$ and estimates the unknown part $\boldsymbol{\theta_2}$. This is an inverse problem of estimating parameters from observations.

$$E(\mathbf{D}, \boldsymbol{\theta_1}) : \mathbb{R}^d \times \mathbb{R}^{p_1} \to \mathbb{R}^{p_2}.$$

We would like to point out that the partition $\boldsymbol{\theta_1}$ might be empty, in which case the estimator would be

$$E(\mathbf{D}) : \mathbb{R}^d \to \mathbb{R}^p.$$

Similarly, we define the data generator G

$$G(\boldsymbol{\theta}) : \mathbb{R}^p \to \mathbb{R}^d,$$

or equivalently

$$G(\boldsymbol{\theta_1}, \boldsymbol{\theta_2}) : \mathbb{R}^{p_1} \times \mathbb{R}^{p_2} \rightarrow \mathbb{R}^d.$$

If only we could estimate the parameters that generate the observations for a certain time period, e.g., a day, we could use the data-generating process G to get the observations. So our primary task boils down to estimating, using an estimator E, these parameters for a time period, given a possibly incomplete observation $\mathbf{D}$. The quality of estimation may vary from one time period to another based on the percentage of missing observations and the noise present. In our SATORIS framework, we hypothesize that certain parameters are invariant across temporally close days whereas other parameters are not. Accordingly, we partition the parameters $\boldsymbol{\theta}$ into the time varying and time invariant parts $\boldsymbol{v} \in \mathbb{R}^{p_1}$ and $\boldsymbol{c} \in \mathbb{R}^{p_2}$ respectively, where $p_1 + p_2 = p$. Let us see what this buys us through an example. Let there be two temporally close days for which we have observations. For the first day we have 95% of observations missing with noise and denoted by $\mathbf{D_1}$. For the 2nd day there is complete observation and denoted by $\mathbf{D_2}$. It is intuitive that the quality of parameters estimated for the first day with missing observations would be poor relative to the second day. However, since the set of invariant parameters $\boldsymbol{c}$ are same for the two days from our assumption, we can get a very good estimate of $\boldsymbol{c}$.

$$\boldsymbol{c}, \boldsymbol{v_2} \leftarrow E(\mathbf{D_2})$$

If $p_1 << p_2$, then even with the noisy and partial observation $\mathbf{D_1}$, we can get a good estimate of the time varying parameters $\boldsymbol{v_1}$ for the first day

$$\boldsymbol{v_1} \leftarrow E(\mathbf{D_1}, \boldsymbol{c})$$

And then we can generate the observations for day 1 with

$$\hat{\mathbf{D}}_\mathbf{1} \leftarrow G(\boldsymbol{c}, \boldsymbol{v_1})$$

We assume that the underlying subspace formed by the matrix/tensor atoms does not vary too much within temporally close days, as verified in Section 4.1. We thus need an estimator E which can take an observation $\mathbf{D}$ (possibly incomplete and noisy) and estimate the invariant subspace parameters $\boldsymbol{c}$. Once we have a good estimate of the parameters, we can generate an estimate of the observations using a forward data generating process $\hat{\mathbf{D}} = G(\boldsymbol{c}, \boldsymbol{v})$ which takes in the time-invariant and the time-varying parameters $\boldsymbol{c}$ and $\boldsymbol{v}$.

4.3 Singular Value Regression

In this method, we assume that the singular vectors are time-invariant while the singular values are time-varying – this essentially means that the matrix atoms are invariant. We describe two algorithms in this method.

4.3.1 Known Singular Vectors: If complete data is available for a temporally close day $\mathbf{D_2}$, we estimate the singular vectors (and hence the matrix atoms) from the matrix representation $\mathbf{D_2}$ (if multiple days are completely observed, we average the matrices and set it as $\mathbf{D_2}$).

$$\text{singularvalues}_2, \text{singularvectors} = \text{SVD}(\mathbf{D_2})$$

Then, using the partial observations for the target day $\mathbf{D_1}$ and the singular vectors (or matrix atoms) obtained from $\mathbf{D_2}$ (which are the same for $\mathbf{D_1}$ and $\mathbf{D_2}$ due to our temporal invariance assumption), we use an estimator of the following form to estimate the singular values for the incomplete observation $\mathbf{D_1}$ as

$$\text{singularvalues}_1 \leftarrow E(\mathbf{D_1}, \text{singularvectors})$$

We call this the known singular vectors (KSV) approach, since we know a priori the singular vectors for the target day (by estimating them from one or more neighboring days) and thus they are only

estimating the singular values using the incomplete and noisy data. In algorithm 1, U and V are the time-invariant parameters and *weights* is a time-variant parameter.

Algorithm 1 SVR: Known Singular Vectors (KSV)

Require: incompletematrix $\in \mathbb{R}^{q \times p}$, completematrices, mask $\in \mathbb{R}^{q \times p}$, rank
Ensure: $0 < rank < \min(p, q)$
  cmatrix ← mean (completematrices)
  U, S, V ← svd (cmatrix, rank)
  atoms ← getatoms(U,V)
  weights ← estimateweights (atoms,incompletematrix,mask)
  return $\sum_i^{\text{rank}}$ weights[i] × atoms[i]

4.3.2 Unknown Singular Vectors. In the above approach, the requirement of having access to a complete observation for a day temporally close to the target day to be imputed may be very limiting. Hence we devise a second approach that we call the unknown singular vectors (USV) approach that can be used to impute values without requiring a set of complete observations. In this approach, we simultaneously impute the missing entries in multiple temporally close days at once. We assume that the singular vectors are time invariant across the days and only the singular values are time-varying (need to be estimated separately for each day). This method fits into our SATORIS framework described in Section 4.2. We accomplish estimating the singular vectors **U** and **V** and the singular values **S** using the alternating least squares (ALS) approach implemented in the pseudo-code below. The incomplete matrix and masks are in $\mathbb{R}^{p \times q}$. For the sketch of our implementation of estimateU, estimateV and estimateS, please look at the file "*usv.py*" in the appendix.

Algorithm 2 SVR: Unknown Singular Vectors (USV)

Require: incmatrices, masks, rank
Ensure: $0 < rank < min(p, q)$
  n ← len(incmatrices)
  U, S, V ← rand(p,rank), rand(rank), rand(rank,q)
  converged ← False
  while !converged **do**
    U ← estimateU (incmatrices, masks, S, V)
    for i in [n] **do**
      S[i] ← estimateS (incmatrices[i],masks, U, V)
    end for
    V ← estimateV (incmatrices, masks, U, S)
    estmatrices = [U @ S[i] @ V for i in 1,2,...n]
    converged = TRE(incmatrices, estmatrices, masks) < 1e-5
  end while
  return estmatrices

4.4 Tensor Weight Regression

In this method, we represent the traffic data in the form of third-order tensors. As mentioned earlier, the first two axes correspond to the spatial parts (longitude and latitude), and the third axis spans the 24 hours of a day. Here, we assume that the underlying rank-one tensor atoms remain invariant across temporally close days. However, the weights that multiply with these atoms to give us the observations vary over the days.

Just as with matrix representations, we consider two scenarios. First, we devise a method for the case in which we have access to a day with complete observations that is temporally close to the target day we wish to impute. We call this method as Known Tensor Factors method or KTF in short.

4.4.1 Known Tensor Factors: Let $\mathbf{D_1}$ be the incomplete observation for a target day and $\mathbf{D_2}$ be the complete observation for one of the temporally close days. We use the non-negative Candecomp/Parafac factorization to estimate the weights $\boldsymbol{w}$ and the rank one factors/atoms $\boldsymbol{f}$ as follows

$$\boldsymbol{w_2}, \boldsymbol{f} \leftarrow \text{nnparafac}(\mathbf{D_2}, \text{rank})$$

Then we estimate the weights $\mathbf{w_1}$ for the incomplete $\mathbf{D_1}$ using an estimator that takes in both the factors $\boldsymbol{f}$ and the incomplete observations $\mathbf{D_1}$ as follows

$$\boldsymbol{w_1} \leftarrow \text{E}(\mathbf{D_1}, \boldsymbol{f})$$

Using the weights and the factors, we predict the observations as

$$\hat{\mathbf{D}}_1 = \sum_{i}^{\text{rank}} \boldsymbol{w}_1[i]\boldsymbol{f}[i]$$

We use the *Tensorly* python library to implement the tensor factor estimators and the generators [15]. The following is an implementation of our KTF algorithm.

Algorithm 3 TWR: Known Tensor Factors (KTF)

Require: incompletetensor, completetensors, mask, rank
Ensure: rank > 0
  ctensor ← mean(completetensors)
  oldweights, factors ← non_negative_cp(ctensor)
  atoms ← getatoms(factors)
  weights ← estimateweights(atoms,incompletetensor,mask)
  return $\sum_i^{\text{rank}}$ weights[i] × atoms[i]

4.4.2 Unknown Tensor Factors: Our second method for tensor representation deals with the case when a complete observation is not available for a temporally close day. Hence we are not able to get a good estimation of the time-invariant tensor-atom subspace. Also, estimating the invariant parameters with good quality is difficult when all we have is observations from a single day with missing and noisy entries. To remedy the situation, we propose the unknown tensor factor (UTF) approach to estimate the invariant factors/atoms simultaneously for target $\mathbf{D_1}$ and temporally close days $\mathbf{D_2}, \mathbf{D_3}, ..., \mathbf{D_n}$, all of which have missing and noisy entries. Estimating the factors simultaneously from multiple days enhances the quality of estimation due to information from the additional days. The weights corresponding to the factors/atoms are considered to vary over the days and are estimated individually for the different days.

$$\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_n}, \boldsymbol{f} \leftarrow E(\mathbf{D_1}, \mathbf{D_2}, ..., \mathbf{D_n})$$

The above formulation lends itself easily to be reformulated as the decomposition of a fourth-order tensor where the fourth axis corresponds to the days stacked together (to remind, the first two axes corresponded to the spatial location, and the third axis reflects the time of the day). For the third-order tensor decomposition, the factor matrices A, B correspond to the spatial factors, C corresponds to the temporal factors in the hours of a day. In the fourth-order tensor decomposition, the fourth-factor matrix corresponds to the weights of the third-order tensor atoms. To obtain this

estimation, inspired by ALS algorithm, we essentially convert a lower-order traffic data imputation problem into a higher-order tensor decomposition and regression problem that takes in the fourth-order tensor, a fourth-order mask tensor (by stacking the masks corresponding to the days) and the rank, as outlined in Algorithm 4. Through this formulation, we are able to use the *non_negative_cp* subroutine from [15] for performing weighted ALS on the incomplete fourth order tensor.

Algorithm 4 TWR: Unknown Tensor Factors (UTF)

Require: incompletetensors, masks, rank
Ensure: rank > 0
  tensor4d ← stack_along_4th_dimension(incompletetensors)
  mask4d ← stack_along_4th_dimension(masks)
  weights, factors ← non_negative_cp(tensor4d, mask4d)
  atoms ← getatoms(factors)
  reconstructed4d ← $\sum_i^{\text{rank}}$ weights[i] × atoms[i]
  imputedtensors ← unstack_4th_dimension(reconstructed4d)
  return imputedtensors

5 RESULTS

Comparison between the proposed SATORIS framework and state-of-the-art imputation algorithms is made in evaluating its performance. A systematic assessment, encompassing factors such as cities, temporal duration, and missing data percentages, enables a comprehensive understanding of our framework's behaviors across diverse scenarios. Furthermore, conducting a comparative analysis of the four different variants of the SATORIS framework reveals their inherent strengths and weaknesses.

5.1 Baseline Algorithms

We compare the proposed methods with the following baseline algorithms.
Row Mean[6, 19–21]: It is applied to the matrix representation. Missing values are set to the mean of the observed entries in the same row as the missing entry. This corresponds to filling the missing traffic density entry at time t by the average of the densities at other locations of the same period t.
Row Median[20, 21]: Row median is used instead of row mean.
KNN[5, 6, 20, 25, 28, 32]: Given an integer k and an incomplete matrix, k closest rows of the row to be imputed are identified according to a metric (l_2 distance in our study), by using the observed entries of the target row. Missing values are filled with the average (or weighted average where weights are proportional to the inverse of the distance) of the corresponding entries in the k rows. In our experiments, we use $k \in \{5, 10, 15, 20\}$ and observe that k=5 is the best performing across all settings. KNN performance for k=5 is shown in all figures to de-clutter the plots.
SoftImpute [22]: It is a semi-definite programming-based soft-thresholded SVD approach on matrix completion, which relies on the minimization of the nuclear norm (i.e., the tightest convex relaxation of the matrix rank).
Gain [34]: A deeplearning based generative adversarial network that uses the generator to impute the missing values and the discriminator to discriminate the missing components from the observed ones. Default hyper-parameters are used for this model.
Miracle [16]: This is state of the art deeplearning assisted imputation algorithm that uses causal structure in the observations to improve the imputation accuracy. Default hyper-parameters are used for this model.

Category	Description
1day	Just the target day.
2next	The target day and the corresponding day from the next week. For example, if the target day is Monday of week10, we consider the set Monday of week 10, Monday of week 11.
2prev	The target day and the corresponding day from the previous week. For example, if the target day is Thursday of week32, we consider the set Thursday of week 32, Thursday of week 31.
3adj	The target day, the day just before and just after the target day of the same week. Hence if the target day is Wednesday of week 15 then the set we consider is Tuesday of week 15, Wednesday of week 15, Thursday of week 15.
5adj	The target day, the two days before and the two days after the target day of the same week. Hence if the target day is Wednesday of week 15 then the set we consider is Monday, Tuesday, Wednesday, Thursday, Friday all from week 15.

Table 2. Dataset organisation wrt temporal proximity

5.2 Dataset and Experiment Setup

Experiments are conducted on the dataset with different amounts of temporal closeness and missing percentages for both cities. The rationale behind sets of data with varying temporal proximity is to compare the invariance of the atom subspaces not only in the neighboring days of the same week but also in corresponding days from temporally close weeks. This can highlight the extent to which the invariance holds true not only across the days of a week but also across weeks. Table 2 explains more details about the dataset organization. Our experiments also include a wide range of missing percentages (low $\leq$ 10%, medium 10% - 90%, and high $\geq$ 90%) to assess the impact on imputation efficacy. We particularly look into high percentages ($\geq$ 90%) to examine our SATORIS framework's ability to handle significant levels of data sparsity.

In our experiments, for the matrix representations, we try all the ranks from 1-23. Note that the maximum rank we can use is 24 as the matrices are of shape 1600x24 for Shanghai and 3136x24 for Beijing. For the tensor representations, we try tensor ranks from 1-50. For all the algorithms, we pick the rank which minimizes the error metric (TCS/RMSE/MAE) for that particular algorithm.

5.3 Tuning the Rank Hyperparameter

We plot the tensor reconstruction error (TRE) and the tensor completion score (TCS) vs rank for a few experiment settings in Figure 5. As expected, the complexity of the model increases with an increased rank, leading to better fit on the training data (decreasing TRE, training error). The TCS (test error), in contrast, decreases initially with rank but then increases rapidly due to over-fitting. Rank is a hyper-parameter in our model and the best rank for our algorithm is the one that minimizes the imputation error (TCS). In Figure 6, we have plotted the box-plot distribution of the rank which minimizes TCS for our four proposed algorithms over all the missing percentages. With an increasing missing percentage, we observe that the best rank keeps decreasing. This is expected as a lower rank corresponds to a simpler model that is not prone to over-fitting. It is interesting to note that for methods in the unknown factor regime, we get very low-rank (almost rank 1) models that perform the best at extremely high missing percentages.

5.4 Imputation Performance

Single Day: In Figure 7, we compare the performance of the baseline algorithms with our proposed method on a single day for 7 different days. We only compare unknown factor/atom methods (USV and UTF) here, since known factor/atom methods (KSV and KTF) are not applicable when we

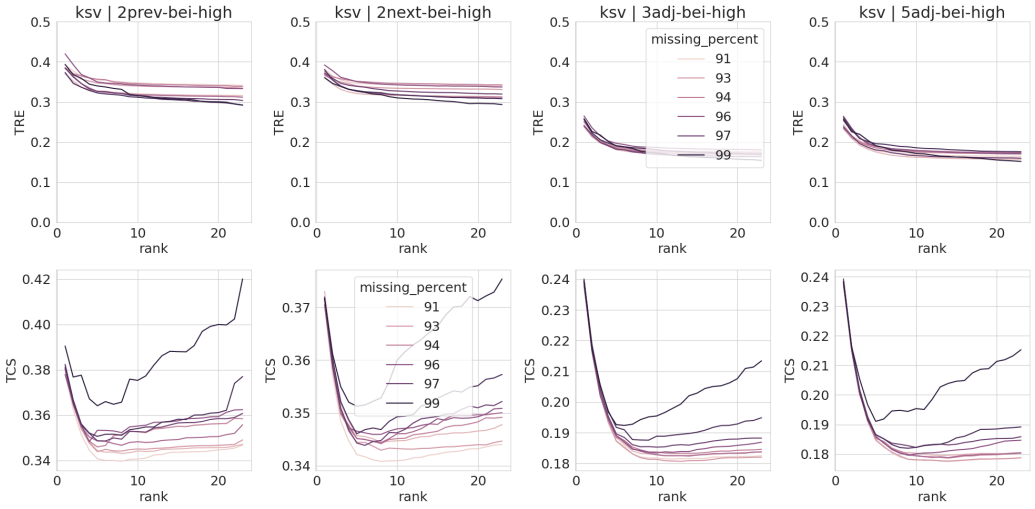

Fig. 5. Error vs rank for Beijing data using KSV

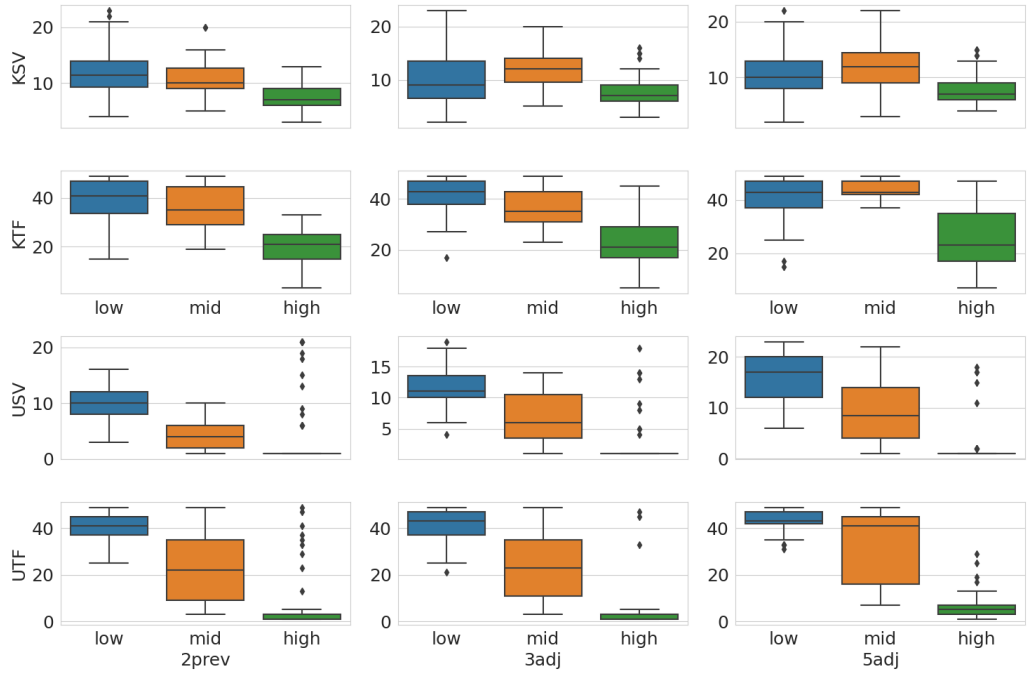

Fig. 6. Rank that minimizes imputation error using KSV-KTF-USV-UTF on Shanghai dataset

only have a single day's observation. From the figures, both USV and UTF outperform the KNN and row statistics baselines in all three missing percent regimes. Both USV/UTF are competitive with softimpute[22] for missing percentages under 70%, and outperform softimpute by a large margin for higher missing percentages. Although Miracle is marginally better for low missing percentages, USV and UTF start performing better as we approach 50% and 60% missing ratio

respectively. For low missing percentages, both the tensor method (UTF) and matrix method (USV) perform equally well; however, for large missing percentages, the tensor method significantly outperforms the matrix method. This can be attributed to the high-order tensor representation's inherent sophistication, which enables it to preserve and handle complex structures more effectively while avoiding over-fitting, making it more resilient to data loss. Please see appendix for RMSE and MAE error plots.

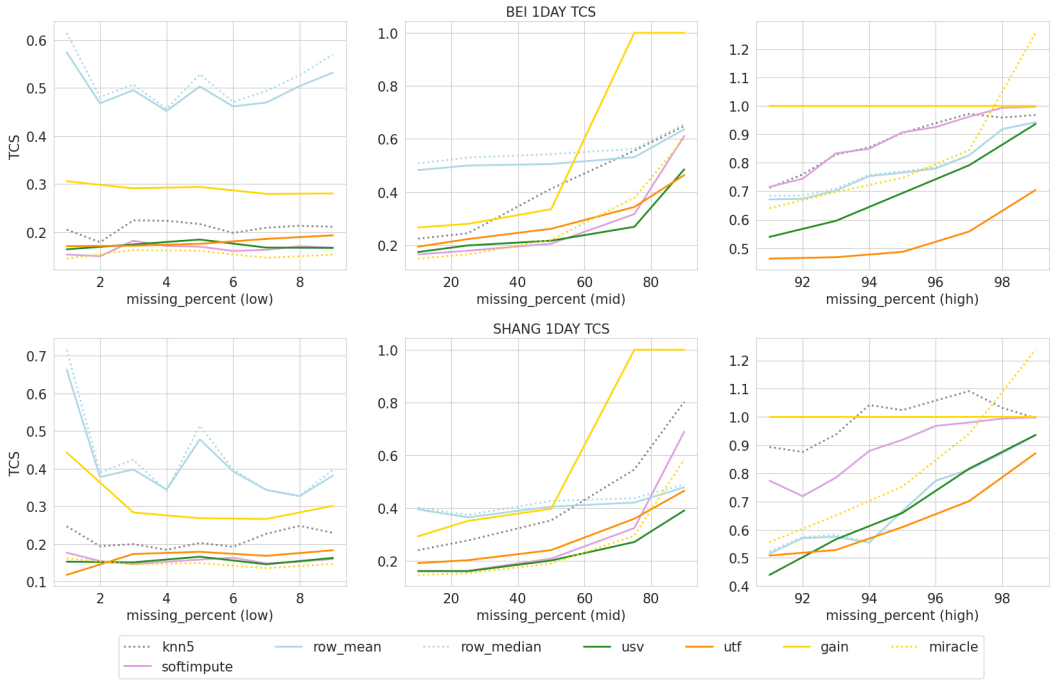

Fig. 7. 1 day performance

2 days (2next and 2prev): For Shanghai, our methods perform better than KNN, row statistic baselines on all missing percentages and are competitive with softimpute for missing ratios below 70% and outperform softimpute by a large margin at higher missing ratios, consistent with what was observed for single-day results above. Similarly, we observe that the proposed methods start outperforming Miracle starting at about 50% missing ratio which is lower than the case for 1 day.

The performance of KTF/KSV is stable across all the missing percentages. For Beijing dataset with low and medium missing percentages, KTF and KSV perform worse than the KNN due to a wide variation in traffic patterns across different days (see Figure 2 for reference). This wide variability exhibited in Beijing dataset undermines our assumption of temporal invariance, resulting in reduced imputation efficacy. Methods in the unknown factor regime, USV and UTF, both perform much better than the baselines for almost all the missing percentages, suggesting that the unknown factor-based estimation of the underlying invariant atoms subspace from multiple days are more robust to signal variations over temporally proximal days.

For higher missing percentages, the KSV/KTF methods outperform UTF/USV methods by a large margin for both datasets. Another trend that is consistent across both datasets is: while UTF and USV perform relatively equally for low and medium missing percentages, for missing percentages above 95%, UTF performs significantly better, suggesting that tensor methods are

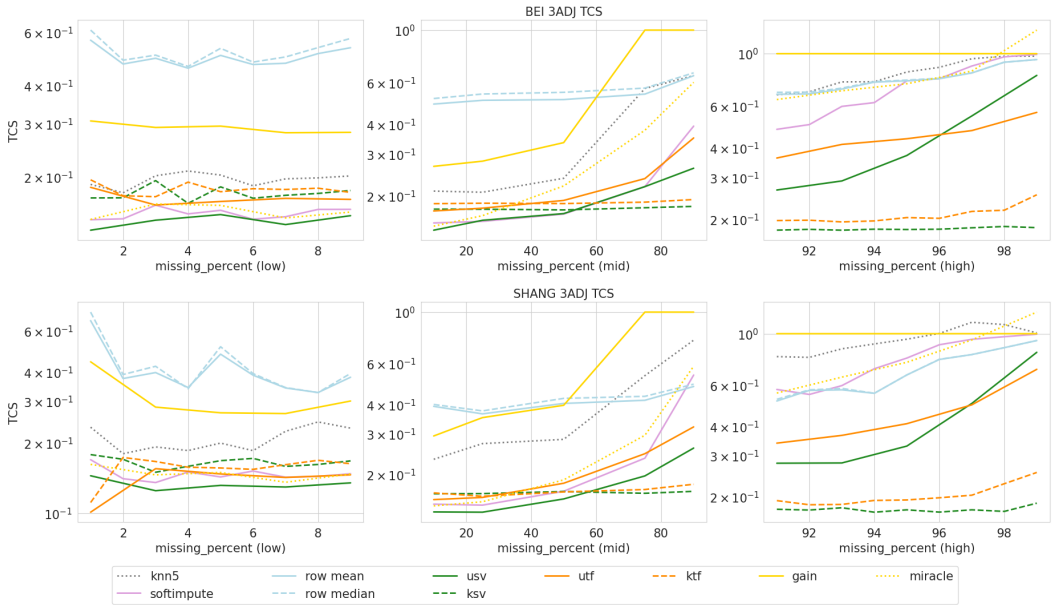

Fig. 8. 3 days performance

better for recovering structure in cases where data is sparse. This observation is consistent with what was observed for 1day and is going to remain consistent for 3adj and 5adj as shown later.

3 days (3adj): As shown in Figure 8, all our methods keep performing better than the KNN and row statistic baselines as in the previous section, and the same observation about softimpute also remains true. Compared to Miracle, the performance is similar to the case of 2days where our are better performing starting at missing ratio of about 50%. Please see appendix for supporting figures and RMSE and MAE error plots.

KSV and KTF perform equally well for low missing percentages but as the missing percentage increases, KTF outperforms KSV gradually and the trend can be seen clearly for the high missing percentages across all datasets. For USV and UTF, USV is slightly better than UTF for low and mid missing percentages. For missing percentages above 97%, UTF performs better, which is consistent with the observations from 1-day and 2-day performances. Also, as observed previously, the UTF/USV methods perform as good as KTF/KSV methods as long as the missing percentages are low. With increasing missing ratio, KTF/KSV slowly outperform UTF/USV approximately after 40%/60% missing percentage. Also note that, as observed previously, KTF/KSV performance is steady and stable across all the missing percentages.

Please see appendix for supporting figures and RMSE and MAE error plots.

5 days (5adj): The same observations hold here as in the case of 3 days. Compared to 3days where the proposed methods started outperforming Miracle at about 50%, this time they do so starting at about 30% which shows that the proposed methods are able to capitalize on the additional observations much better. We also observe that for the tensor method UTF to outperform the matrix method USV, it now takes missing percentages above 98% whereas for 3days it was 97% and for two days it was 95%. This hints that as we add more information (more days), the matrix method (USV) performs better for higher and higher missing percentages. The supporting graphs (Figure **??**) are in the appendix which also contain RMSE and MAE error plots.

Missing Percent (x)	Complete observation available for temporally close day(s)?	Recommended Algo
$x < 50\%$	-	USV
$50\% < x < 95\%$	YES	KSV or KTF
$95\% < x$	YES	KSV
$50\% < x < 95\%$	NO	USV
$95\% < x$	NO	UTF

Table 3. Selecting the best algorithm under different cases

5.5 Key Observations and Summary

We summarise the key observations from the above experiments that are consistent across the datasets from the two cities. We provide table 3 for an easy overview on selecting an imputation algorithm from the proposed ones in the various situations presented in this paper.

Baseline vs Proposed Methods: All of our methods outperform the baselines consistently as the missing ratio exceeds about 75%. This is because our methods are able to capture the inherent time-invariant and variant structures explicitly while being a simple in model complexity so as to not over-fit the small number of observations available at high missing ratios whereas the baselines either lack a mechanism for explicitly capture the above two mechanisms or are too complex of a model to start over-fitting the sparse observations leading to degraded performance at higher missing ratios.

Known Factor Regime vs Unknown Factor Regime: Unknown factor regime (USV/UTF) perform slightly better than the known factor regime for up to 50% missing in all experiments. At 50% or more missing values, the known factor regime start to outperform the unknown factor regime significantly. This is because *the known factor regime sees a complete dataset for estimating the factors and hence have access to more and better quality data than the methods in the unknown factor regime.* At low missing percentages, the observations are almost complete and this difference is not significant enough leading the unknown factor regime to perform at par or slightly better than the known factor regime. This could be attributed to the fact that the factors in the unknown factor regime are also able to learn from the observations of the target day where as in the known factor regime they can only see observations from the temporally proximate days. At high missing percentages, the lack of sufficient observations degrades the performance of the unknown factor regime significantly. However, we would like to point out that methods in the known factor regime are *much more sensitive to the assumption about invariability of the invariant temporal factors*; it won't work as well as the unknown factor regime when this assumption is violated. This can be seen in figures from A.2 in which the KSV/KTF perform worse on the Beijing dataset but do well on the Shanghai dataset, because there is more day-to-day variation in the Beijing dataset (as evident from the Figure 2).

Unknown Factor Regime: USV vs UTF: For almost all the experiments (Figures 7,**??**,8), the matrix method (USV) slightly outperforms the tensor method (UTF) for low and medium missing percentages. When the missing percentages are high (95% and above), nonetheless, the tensor method (UTF) outperforms the matrix counterpart (USV) significantly, which is highlighted by Figure 9. *We believe this is attributed to over-fitting by USV model caused by its high sample complexity.*

The number of parameters in k matrix-atoms is $k(mn + p)$ and in k tensor-atoms is $k(m + n + p)$. For example, we have $m = n = 40$ and $p = 24$ for a day's observation in Shanghai dataset. Hence at the same rank k, matrix based models are more expressive (1624 parameters for rank-1 matrix) as compared to tensor based models (104 parameters for rank-1 tensor). This explains the better performance of UTF as compared to USV for missing percentages above 95%, since the low number of parameters in the tensor model (only 104 parameters) prevent overfitting; in contrast, even a

simple rank-1 USV matrix-based model (with 1624 parameters) overfits the sparse observations, leading to degraded imputation performance.

Known Factor Regime: KSV vs KTF: Both of them perform almost equally. Unlike in the unknown regime where the performance suffers significantly with an increase in missing percentages, *there is not so much of an effect on KSV and KTF with an increase in missing data – the performance remains stable over all the missing percentages.* Also, KSV and KTF outperform all the other methods and give the best imputation results at high missing percentages. All this comes with a cost - *having access to complete observations from at least one temporally close day to the target day to be imputed.*

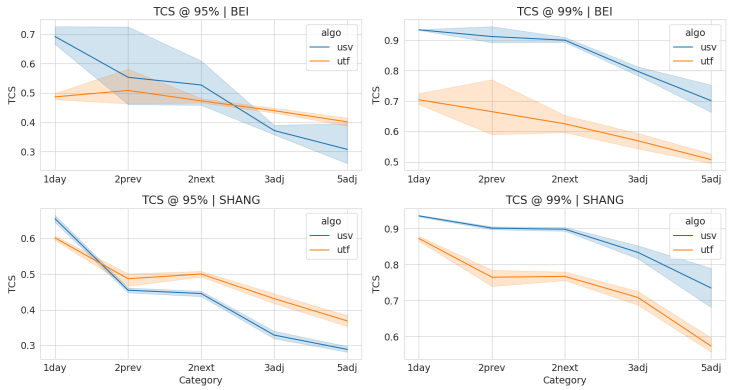

Fig. 9. Imputation error vs no. of days (for HMP)

Imputation Error vs Number of Days (at high missing percent): We observe from Figure 9 that the imputation performance improves (error goes down) as we provide more number of days to our algorithms. This is expected since more data helps with a better estimation of the time-invariant tensor/matrix factors (atoms) and the time-varying weights corresponding to the atoms. Here we have shown the plot for only the TCS error metric but the same is observed for RMSE and MAE as well.

6 RELATED WORK

Many techniques have been developed to tackle missing values in traffic data [5, 10, 12, 23, 29, 33]. Sun *et al.* [29] classify the techniques into simple and complex techniques. Simple techniques use basic mathematical operations, for example, replacing the missing values with the mean/median/mode of a subset of the non-missing data. They conclude that matrix completion techniques have turned out to outperform other techniques. Also from their experiments, we find k-nearest-neighbour(KNN) is competitive and has better performance than LSTMs and ARIMA-based models in many situations, thus KNN making it into our baselines. Chan *et al.* [5] split the missing data imputation techniques into two types — statistical and machine learning (ML) based methods. Tensor decomposition methods belongs to both the statistical and ML-based methods category simultaneously, and it is considered to be more interpretable compared to other ML models. They point out that although traffic volume gives a good idea of the state of traffic, traffic volume sees fewer missing data imputation studies, likely due to data availability and the imprecise nature of traffic volume. In their survey of more than a 10 tensor decomposition-based methods, none of them have applied imputation on traffic volume/count/density, which then places our work in a unique position. They stress on future imputation studies to take into account not only the accuracy but also interpretability and computational complexity which have been considered in this work. In the context of our work and space constraints, we discuss interpretable methods and a few recent deep learning imputation techniques.

Conventional methods: Tang *et al.* [31] present an approach based on fuzzy C-means (FCM) and use a genetic algorithm for optimization. Their results surpass ARIMA and multiple linear regression-based models but they only have results for up to 30% missing ratio, as compared to our extensive experiments on missing ratios from 1% to 99%. Wang *et al.* [32] propose an adaptive k-nearest neighbor method (with a multivariate linear regression model) to estimate weights for data smoothed by a wavelet technique. Their experiments compare missing ratios of 10-90% and show better performance than the standard KNN method.

Matrix methods: Huang *et al.* [13] present a model based on probabilistic PCA and show that missing data on links with higher Pearson scores can be better recovered. However, they compare their method to only another PPCA variant that uses EM algorithm and uses missing ratios of 25%,50% and 75% compared to our results which are extensive. Lei *et al.* [17] present a Bayesian kernelized matrix factorization (BKMF) model with MCMC sampling for traffic data imputation. Another promising category of approaches to handle missing traffic data has to do with singular value decomposition (SVD) which are related to PCA-based imputation methods. In [3], Candès *et al.* demonstrate that, under certain conditions, their SVD-based matrix completion approach recovers the missing values in a low-rank matrix exactly. Recent studies [4], [24] have also explored non-convex formulations for addressing the phase retrieval problem which has to do with matrix completion. These works focus on developing algorithms specifically designed for particular non-convex formulations and with predetermined initialization, providing theoretical guarantees.

Tensor methods: Various papers have explored the use of tensor representations for missing traffic data imputation in the past few years. Chen *et al.* [9] present a scheme whereby a truncated SVD first captures latent features along each dimension and these are combined into a tensor representation with an imputation technique based on the Tucker decomposition. Tucker decomposition of tensors are not as interpretable as CP decomposition or non-negative CP decomposition (used in our tensor-based methods). The authors of [8] present an approach that extends a Bayesian probabilistic matrix factorization model to higher-order tensors. Li *et al.* [19] compare four different tensor-based completion techniques for missing traffic data imputation. Their results show that Smooth PARAFAC/CP decomposition outperforms the other tensor-based methods and hence partially supports our choice of basing our tensor-based algorithms on the more interpretable CP decomposition rather than more complex tensor decomposition.

To the best of our knowledge, none of the prior literature on matrix and tensor based imputation methods have considered invariance and stability of singular vectors and tensor factors across temporally proximal observations. Our methods, in contrast, are explicitly designed to exploit this invariance phenomena which is the secret to the strong results shown in this paper.

Deep learning methods: [10] provides a comprehensive survey on deep learning-based imputation methods. [30] compares deep learning methods with more conventional imputation methods. They conclude that deeplearning methods such as Gain [34] and others do not do well as compard to conventional methods when there are very few samples/observations which is consistent with observations made earlier in this paper. Chen *et al.* [7] propose a deep learning-based approach that uses a self-attention-based temporal convolutional network (ATCN) and a model equipped with an encoder-decoder architecture. [34] proposes a generative adversarial network based imputation scheme where the generator is tasked with guessing the values of the missing observations and the discriminator tries to identify the missing components. [16] proposes a graph neural network based imputation technique that can be used to augment a given imputation scheme by exploiting causality in the observed part of the data.

7 CONCLUSION AND FUTURE WORK

In this paper, we presented a novel low-rank parameter estimation framework called SATORIS which explicitly use temporal invariance in the matrix/tensor factors across temporally close days. We propose four algorithms which are classified based on two different criteria: 1) whether they use matrix or tensor representation of the traffic data and 2) whether they have prior knowledge of the time-invariant structure for the day they are imputing. We evaluate all the algorithms on taxi dataset from two cities, Beijing and Shanghai, and compare their performances over varying missing percentages with various baseline methods including two recent deeplearning based imputation methods. We found our proposed algorithms to perform competitively with the state-of-the-art baselines at low and medium missing ratio and surpassing them for higher missing ratios (above 70%). We also compare the performances of proposed algorithms among themselves. Intriguingly, we find that, at low missing percentages, the algorithms in the unknown factors regime are slightly better than the known factor regime with the matrix method (USV) performing better than the tensor method (UTF). In contrast, at high missing percentages, the known factor algorithms (KSV/KTF) outperform the unknown factor algorithms (USV/UTF). We also observe that in the case of unknown factor algorithms, the tensor method tends to be more resilient to high missing data.

In the future, we plan to improve methods in the unknown factor regime to work well on high missing percentages using regularisation to further control model complexity. We also would like to explore the possibility of extending the idea of invariance in certain parameters techniques to deep tensor decomposition and graph neural networks for the traffic imputation problem.

REFERENCES

[1] Evrim Acar, Tamara G. Kolda, Daniel M. Dunlavy, and Morten Morup. 2010. Scalable Tensor Factorizations for Incomplete Data. (5 2010). https://doi.org/10.1016/j.chemolab.2010.08.004

[2] Muhammad Tayyab Asif, Nikola Mitrovic, Justin Dauwels, and Patrick Jaillet. 2016. Matrix and Tensor Based Methods for Missing Data Estimation in Large Traffic Networks. *IEEE Transactions on Intelligent Transportation Systems* 17 (7 2016), 1816–1825. Issue 7. https://doi.org/10.1109/TITS.2015.2507259

[3] Emmanuel Candès and Benjamin Recht. 2009. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics* 9, 6 (2009), 717–772. https://doi.org/10.1007/s10208-009-9045-5

[4] Emmanuel J. Candès, Xiaodong Li, and Mahdi Soltanolkotabi. 2015. Phase Retrieval via Wirtinger Flow: Theory and Algorithms. *IEEE Transactions on Information Theory* 61, 4 (2015), 1985–2007. https://doi.org/10.1109/TIT.2015.2399924

[5] Robin Kuok Cheong Chan, Joanne Mun-Yee Lim, and Rajendran Parthiban. 2023. Missing Traffic Data Imputation for Artificial Intelligence in Intelligent Transportation Systems: Review of Methods, Limitations, and Challenges. *IEEE Access* 11 (2023), 34080–34093. https://doi.org/10.1109/ACCESS.2023.3264216

[6] Gang Chang and Tongmin Ge. 2011. Comparison of missing data imputation methods for traffic flow. In *Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*. 639–642. https://doi.org/10.1109/TMEE.2011.6199284

[7] Weiqiang Chen, Jianlong Zhao, Wenwen Wang, and Huijun Dai. 2022. Towards missing traffic data imputation using attention-based temporal convolutional networks. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 3733–3739.

[8] Xinyu Chen, Zhaocheng He, and Lijun Sun. 2019. A Bayesian tensor decomposition approach for spatiotemporal traffic data imputation. *Transportation research part C: emerging technologies* 98 (2019), 73–84.

[9] Xinyu Chen, Zhaocheng He, and Jiawei Wang. 2018. Spatial-temporal traffic speed patterns discovery and incomplete data recovery via SVD-combined tensor decomposition. *Transportation research part C: emerging technologies* 86 (2018), 59–77.

[10] Chenguang Fang and Chen Wang. 2020. Time Series Data Imputation: A Survey on Deep Learning Approaches. arXiv:2011.11347 [cs.LG]

[11] K R Gabriel. 1978. Least Squares Approximation of Matrices by Additive and Multiplicative Models. , 186-196 pages. Issue 2. https://www.jstor.org/stable/2984755

[12] Vikesh Kumar Gond, Aditya Dubey, and Akhtar Rasool. 2021. A Survey of Machine Learning-Based Approaches for Missing Value Imputation. In *2021 Third International Conference on Inventive Research in Computing Applications*

(ICIRCA). 1–8. https://doi.org/10.1109/ICIRCA51532.2021.9544957

[13] Liping Huang, Zhenghuan Li, Ruikang Luo, and Rong Su. 2022. Missing traffic data imputation with a linear generative model based on probabilistic principal component analysis. *Sensors* 23, 1 (2022), 204.

[14] Tamara G Kolda and Brett W Bader. 2009. Tensor Review. *SIAM Rev.* 51 (2009), 455–500. Issue 3. http://link.aip.org/link/?SIR/51/455%5Cnhttp://epubs.siam.org/doi/abs/10.1137/07070111X%5Cnpapers3://publication/uuid/2CAD3CE8-5FD4-4EFB-87B5-E8EA5C3889C0

[15] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. 2019. TensorLy: Tensor Learning in Python. *Journal of Machine Learning Research* 20, 26 (2019), 1–6. http://jmlr.org/papers/v20/18-277.html

[16] Trent Kyono, Yao Zhang, Alexis Bellot, and Mihaela van der Schaar. 2021. MIRACLE: Causally-Aware Imputation via Learning Missing Data Mechanisms. *CoRR* abs/2111.03187 (2021). arXiv:2111.03187 https://arxiv.org/abs/2111.03187

[17] Mengying Lei, Aurelie Labbe, Yuankai Wu, and Lijun Sun. 2022. Bayesian kernelized matrix factorization for spatiotemporal traffic data imputation and kriging. *IEEE Transactions on Intelligent Transportation Systems* 23, 10 (2022), 18962–18974.

[18] Soumaya Ben Letaifa. 2015. How to strategize smart cities: Revealing the SMART model. *Journal of Business Research* 68 (7 2015), 1414–1419. Issue 7. https://doi.org/10.1016/j.jbusres.2015.01.024

[19] Qin Li, Huachun Tan, Yuankai Wu, Linhui Ye, and Fan Ding. 2020. Traffic flow prediction with missing data imputed by tensor completion methods. *IEEE Access* 8 (2020), 63188–63201.

[20] Sunghoon Lim, Sun Jun Kim, YoungJae Park, and Nahyun Kwon. 2021. A deep learning-based time series model with missing value handling techniques to predict various types of liquid cargo traffic. *Expert Systems with Applications* 184 (2021), 115532. https://doi.org/10.1016/j.eswa.2021.115532

[21] Tanwi Mallick, Prasanna Balaprakash, Eric Rask, and Jane Macfarlane. 2020. Graph-Partitioning-Based Diffusion Convolutional Recurrent Neural Network for Large-Scale Traffic Forecasting. *Transportation Research Record* 2674, 9 (2020), 473–488. https://doi.org/10.1177/0361198120930010 arXiv:https://doi.org/10.1177/0361198120930010

[22] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. 2010. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *Journal of Machine Learning Research* 11, 80 (2010), 2287–2322. http://jmlr.org/papers/v11/mazumder10a.html

[23] Xiaoye Miao, Yangyang Wu, Lu Chen, Yunjun Gao, and Jianwei Yin. 2023. An Experimental Survey of Missing Data Imputation Algorithms. *IEEE Transactions on Knowledge and Data Engineering* 35, 7 (2023), 6630–6650. https://doi.org/10.1109/TKDE.2022.3186498

[24] Praneeth Netrapalli, Prateek Jain, and Sujay Sanghavi. 2013. Phase Retrieval using Alternating Minimization. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2013/file/242c100dc94f871b6d7215b868a875f8-Paper.pdf

[25] Thu Nguyen, Hoang Thien Ly, Michael Alexander Riegler, Pål Halvorsen, and Hugo L. Hammer. 2023. Principal Component Analysis based frameworks for efficient missing data imputation algorithms. arXiv:2205.15150 [cs.LG]

[26] Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. 2017. Introduction to Tensor Decompositions and their Applications in Machine Learning. (11 2017). http://arxiv.org/abs/1711.10781

[27] Aude Sportisse, Claire Boyer, and Julie Josse. 2019. Estimation and imputation in Probabilistic Principal Component Analysis with Missing Not At Random data. https://doi.org/10.48550/ARXIV.1906.02493

[28] Bin Sun, Liyao Ma, Wei Cheng, Wei Wen, Prashant Goswami, and Guohua Bai. 2017. An improved k-nearest neighbours method for traffic time series imputation. In *2017 Chinese Automation Congress (CAC)*. 7346–7351. https://doi.org/10.1109/CAC.2017.8244105

[29] Tuo Sun, Shihao Zhu, Ruochen Hao, Bo Sun, and Jiemin Xie. 2022. Traffic Missing Data Imputation: A Selective Overview of Temporal Theories and Algorithms. *Sci. China Ser. A Math.* 10, 14 (July 2022), 2544.

[30] Yige Sun, Jing Li, Yifan Xu, Tingting Zhang, and Xiaofeng Wang. 2023. Deep learning versus conventional methods for missing data imputation: A review and comparative study. *Expert Systems with Applications* 227 (2023), 120201. https://doi.org/10.1016/j.eswa.2023.120201

[31] Jinjun Tang, Yinhai Wang, Shen Zhang, Hua Wang, Fang Liu, and Shaowei Yu. 2015. On missing traffic data imputation based on fuzzy C-means method by considering spatial–temporal correlation. *Transportation Research Record* 2528, 1 (2015), 86–95.

[32] Yang Wang, Yu Xiao, Jianhui Lai, and Yanyan Chen. 2020. An adaptive k nearest neighbour method for imputation of missing traffic data based on two similarity metrics. *Archives of Transport* 54 (2020).

[33] Pan Wu, Lunhui Xu, and Zilin Huang. 2020. Imputation methods used in missing traffic data: A literature review. In *Artificial Intelligence Algorithms and Applications: 11th International Symposium, ISICA 2019, Guangzhou, China, November 16–17, 2019, Revised Selected Papers 11*. Springer, 662–677.

[34] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2018. GAIN: Missing Data Imputation using Generative Adversarial Nets. *CoRR* abs/1806.02920 (2018). arXiv:1806.02920 http://arxiv.org/abs/1806.02920

A APPENDIX

A.1 Invariance of atom subspace: Average Subspace angles

In Figure 10, we support the relative invariance of the atomic subspaces of neighboring days using the average angle between them. For lower ranks, we have smaller average angle implying better subspace alignment. This alignment is more so for the tensor atoms than the matrix atoms.

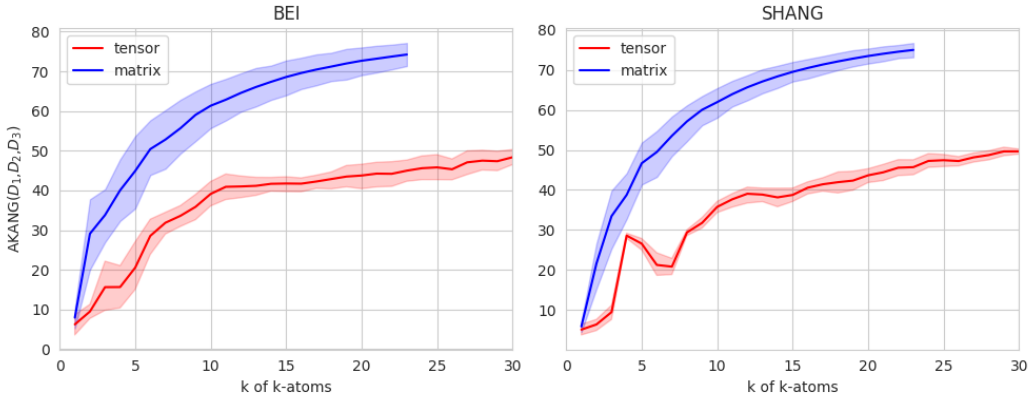

Fig. 10. Average subspace angles for matrix and tensor k-atoms over varying rank (k)

A.2 Performance comparison for 2day and 5day

Figures 14,15,12,13 and 16 support our claims for the dataset categories involving 2 and 5 days. The general performance of the algorithms for 2 and 5 days dataset categories are consistent with what is observed for the other dataset categories.

A.3 Number of parameters in the best model

Figure 18 plots the number of parameters of the best models instead of their rank (as in Figure 6). As can be seen, the matrix-based USV/KSV models use a lot more parameters than the tensor-based UTF/KTF models for all missing ratio categories.

A.4 Dealing with strict requirements of a complete observation in neighboring day for KSV and KTF

It is still possible to use KSV/KTF methods even if we do not have a complete observation from a neighboring day. If there are partial observations $\mathbf{O_1}, \mathbf{O_2}, ..., \mathbf{O_k}$ from k neighboring days which are associated with the observation masks $\mathbf{M_1}, \mathbf{M_2}, ..., \mathbf{M_k}$ (which are matrices/tensors) such that

$$\mathbf{M} = \mathbf{M_1} + \mathbf{M_2} + ... + \mathbf{M_k}$$

all entries of the combined mask $\mathbf{M}$ are non-zero (which implies we have atleast one observation for each index of the matrix/tensor representation of the observations), then we can use the following aggregate matrix/tensor as a complete observation $\mathbf{O}$ to be provided to the KSV/KTF algorithm

$$\mathbf{O} = \frac{1}{\mathbf{M}} \circ \sum_{i=1}^{k} \mathbf{O_i}$$

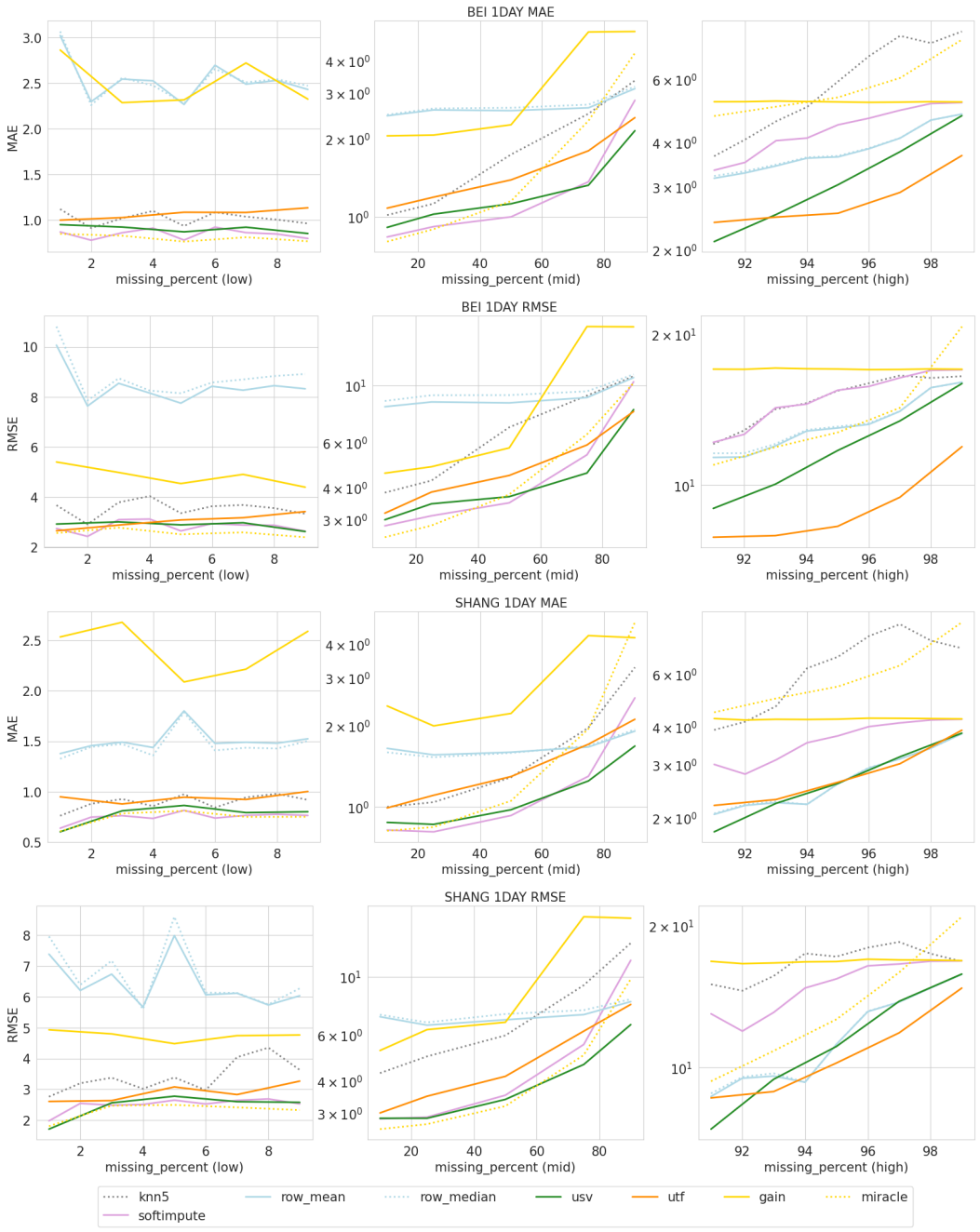

Fig. 11. 1 Day Performances using RMSE and MAE error metrics

where, by $\frac{1}{\mathbf{M}}$, we mean the elementwise inverse of the entries of $\mathbf{M}$ and $\circ$ is the elementwise product operation.

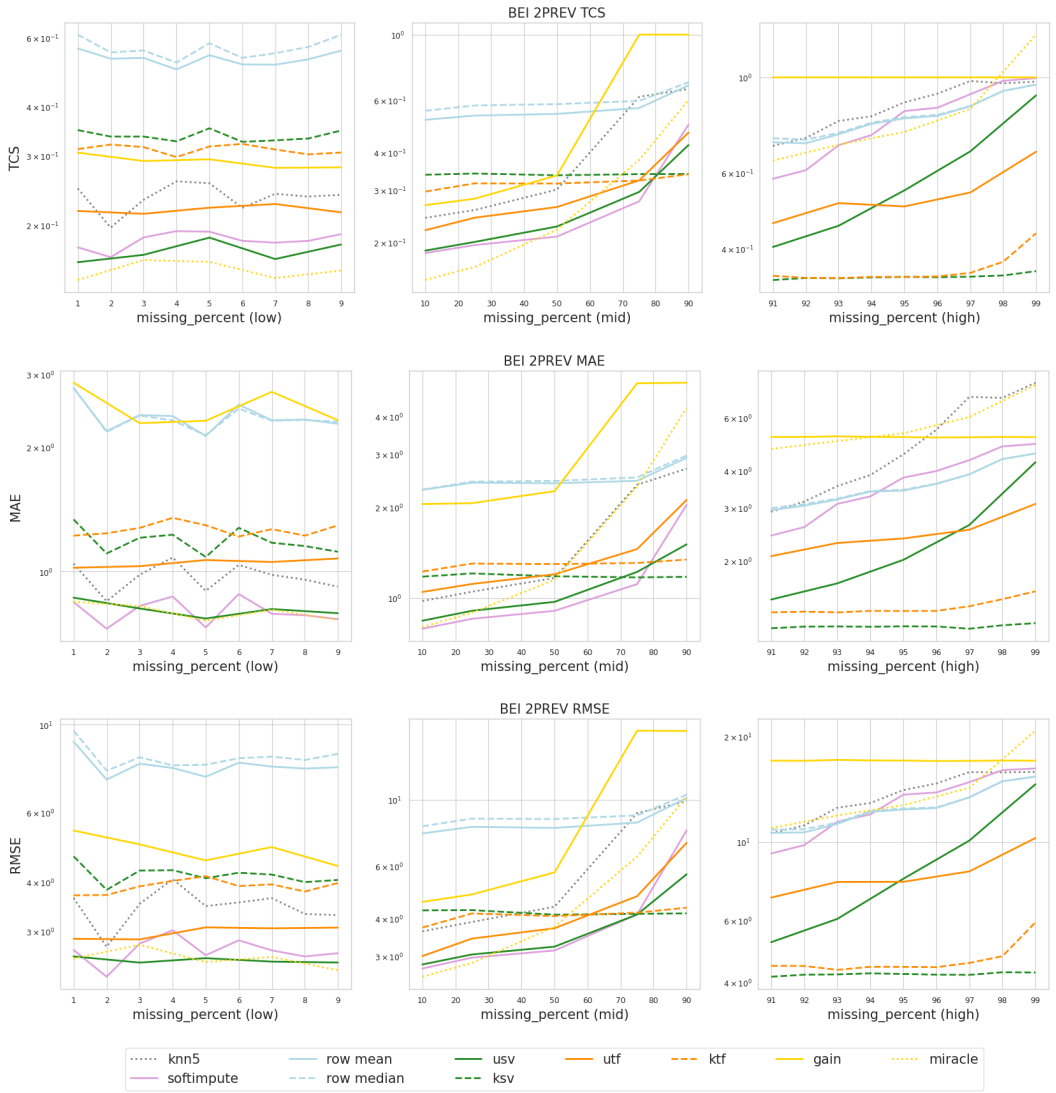

Fig. 12. 2 prev performance (Beijing)

A.5 Common imports

Importing modules common to the code that follows in sections describing KSV, USV, KTF, and UTF.

```
import tensorly.decomposition as td
import tensorly as tl
from tensorly import cp_to_tensors as cp2t
from pylab import *
```

A.6 KSV

```
def estimateAllParams(matrix, rank):
    U, S, V = svd(matrix)
    return U, S, V
```

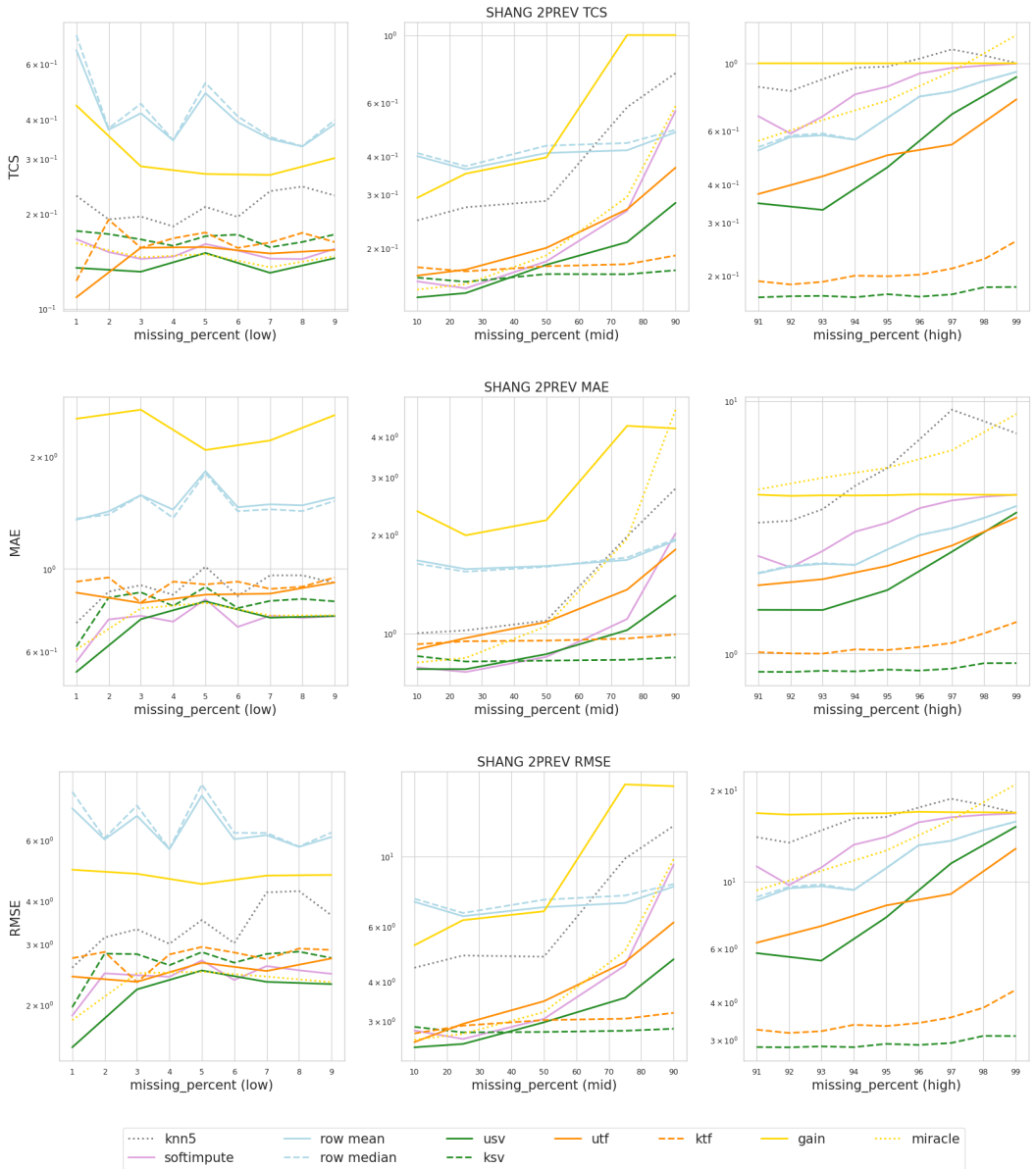

Fig. 13. 2 prev performance (Shanghai)

```
def estimateTimeVaryingParams(incompletematrix, mask, U, V):
    rank = U.shape[1]
    rankone_subspaces = \
    [outer(U[:, i], V[:, i]) for i in range(rank)]
    masked_subspaces = \
    [mask * ss for ss in rankone_subspaces]
    A = vstack(masked_subspaces)
    b = incompletematrix.flatten()
    singular_values = lstsq(A, b)
    return singular_values

def generate(U, S, V):
```

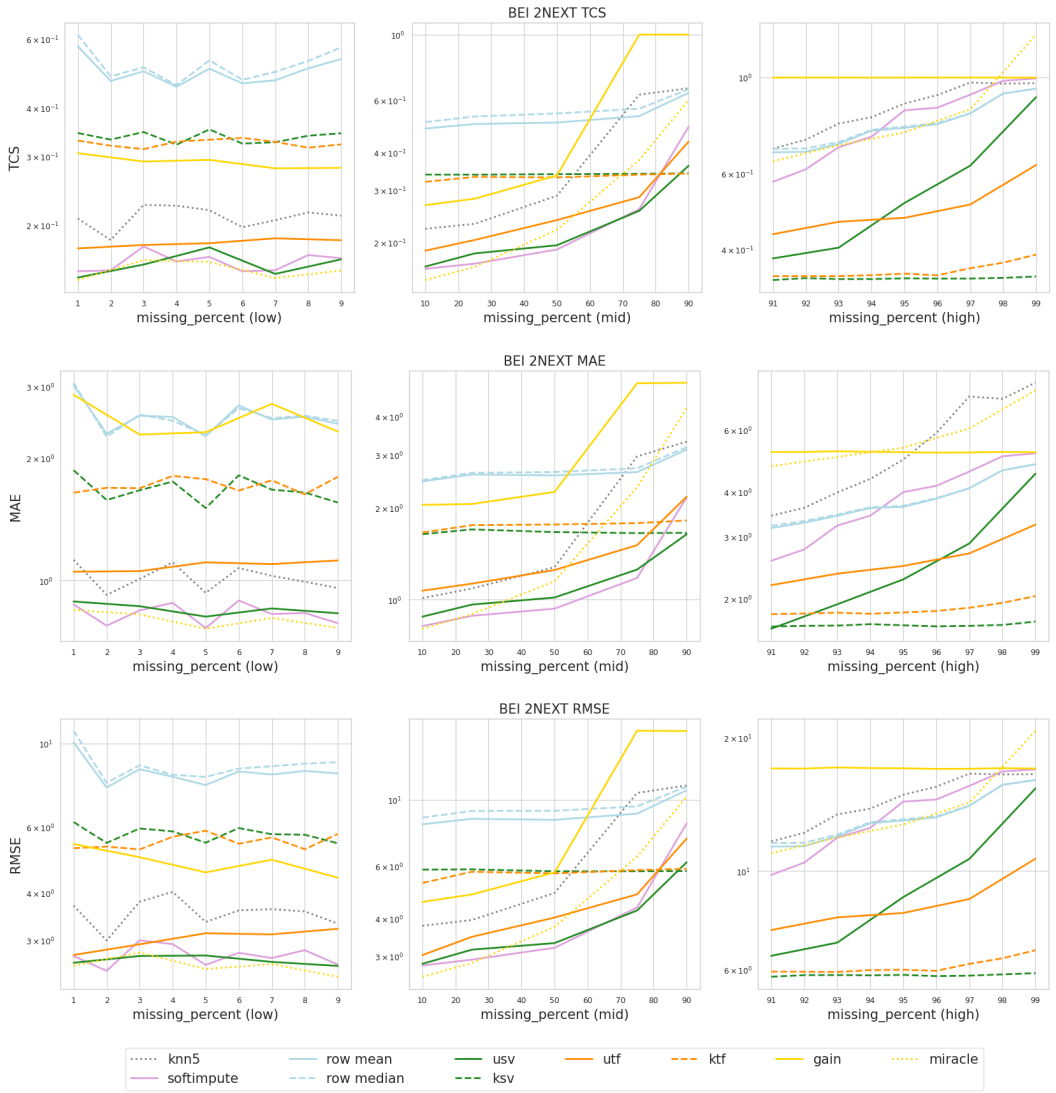

Fig. 14. 2 next performance (Beijing)

```
    return U @ diag(S) @ V

def ksv(incompletematrix, completematrices, mask, rank):
    complete = sum(completematrices)
    U, S, V = estimateAllParams(complete, rank)
    svalues = \
    estimateTimeVaryingParams(incompletematrix, mask, U, V)
    return generate(U, svalues, V)
```

A.7 USV

```
def estimateU(incmatrices, masks, S, V):
    nmats = len(incmatrices)
    rank = V.shape[1]
    m, n = incmatrices[0].shape
    U = rand(m, rank)
```

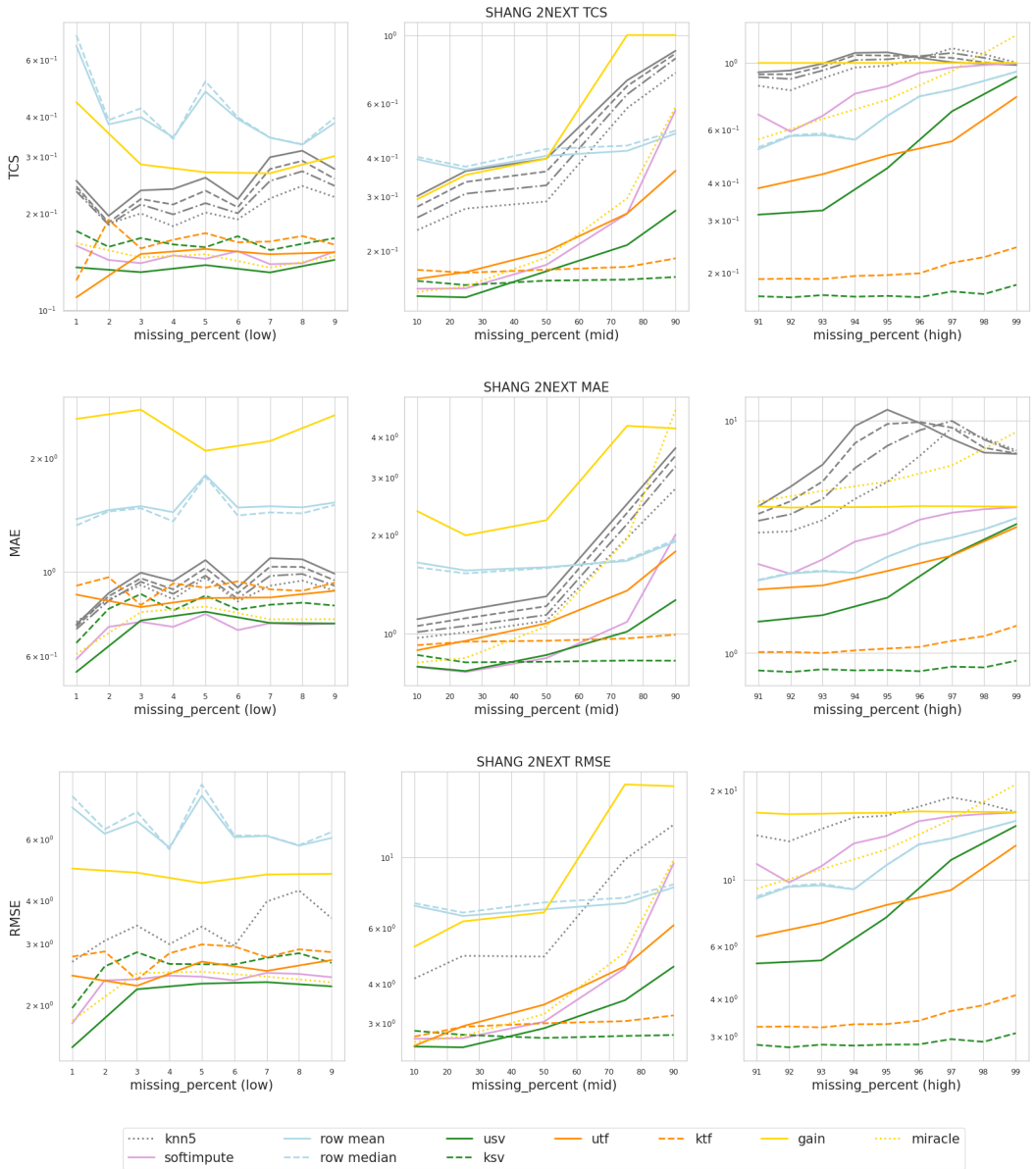

Fig. 15. 2 next performance (Shanghai)

```
    SV = [diag(S[i]) @ V.T for i in range(nmats)]
    for row in range(m):
        A, b = [], []
        for i in range(t):
            rowmask = outer(ones(rank), masks[i][row, :])
            A.append(SV[i] * rowmask)
            b.append(incmatrices[i][row, :] * masks[i][row, :])
        A = hstack(A)
        b = hstack(b)
        U[row, :], *_ = lstsq(A.T, b)

def estimateV(incmatrices, masks, S, U):
    nmats = len(incmatrices)
```

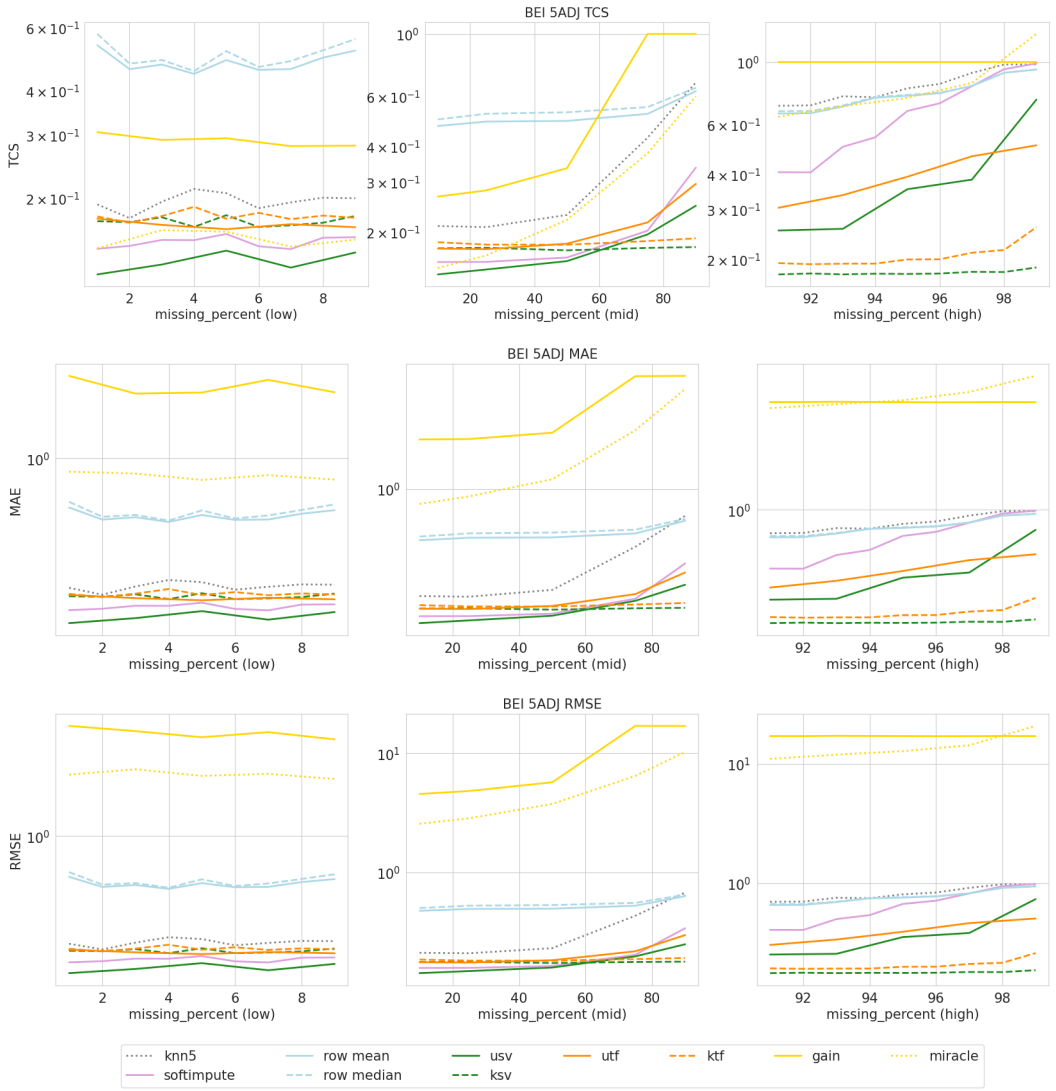

Fig. 16. 5 days performance (Beijing)

```
    rank = U.shape[1]
    m, n = incmatrices[0].shape
    US = [U @ diag(S[i]) for i in range(t)]
    for col in range(n):
        A, b = [], []
        for i in range(nmats):
            colmask = outer(masks[i][:, col], ones(rank))
            A.append(US[i] * colmask)
            b.append(incmatrices[i][:, col] * masks[i][:, col])
        A = vstack(A)
        b = hstack(b)
        V[col, :], *_ = lstsq(A, b)

def estimateS(incmatrices, masks, U, V):
    nmats = len(incmatrices)
    rank = U.shape[1]
```

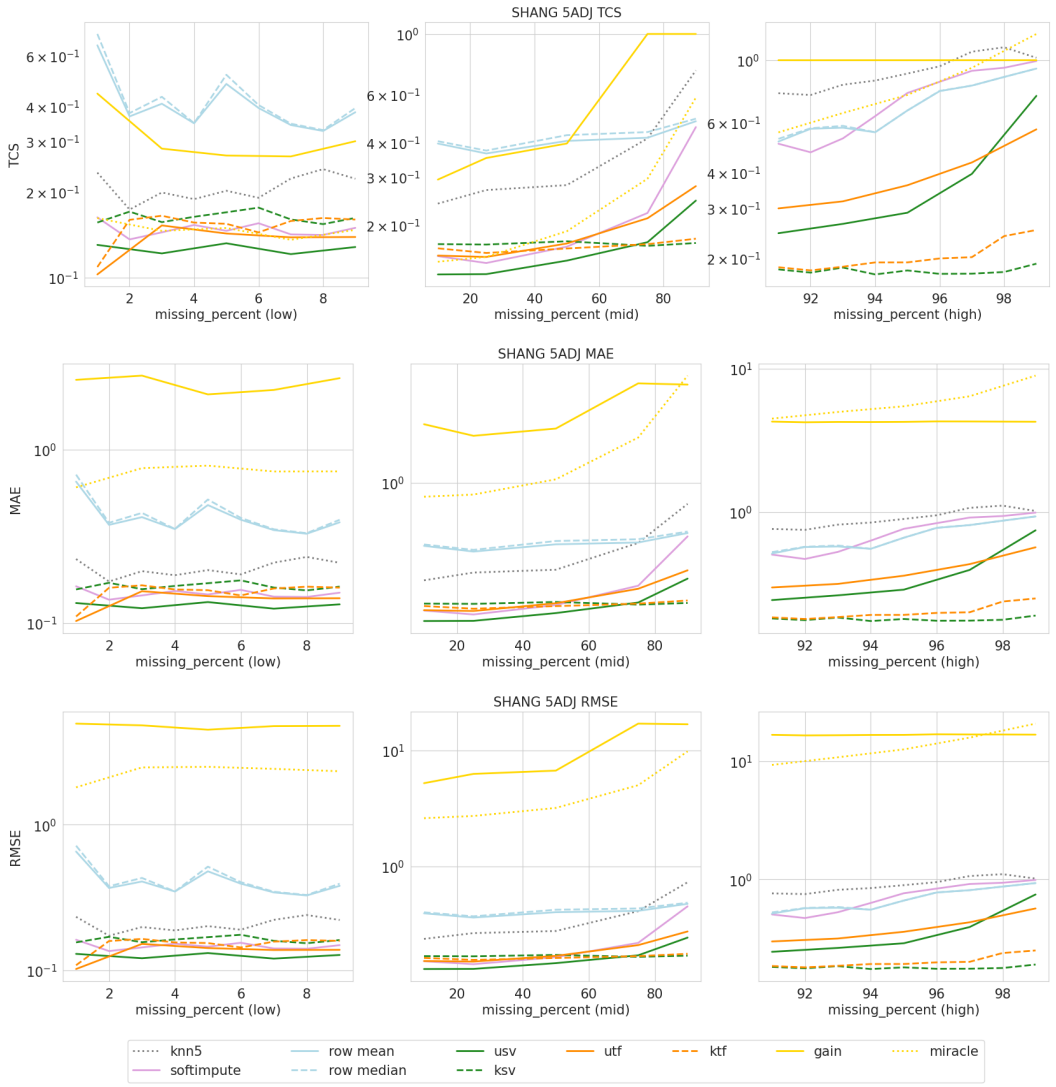

Fig. 17. 5 days performance (Shanghai)

```
    S = [zeros(rank) for inmat in incmatrices]
    for i in range(nmats):
        A = vstack([outer(U[:, r], V[:, r]) * \
        masks[i] for r in range(rank)])
        b = (incmatrices * masks[i]).flatten()
        S[i], *_ = lstsq(A, b)
    return S

def usv(incmatrices, masks, rank):
    m, n = incmatrices[0].shape
    nmats = len(incmatrices)
    U = rand(m, rank)
    V = rand(n, rank)
    S = [rand(rank) for inmat in incmatrices]
    tol = 1e-7
    converged = False
```

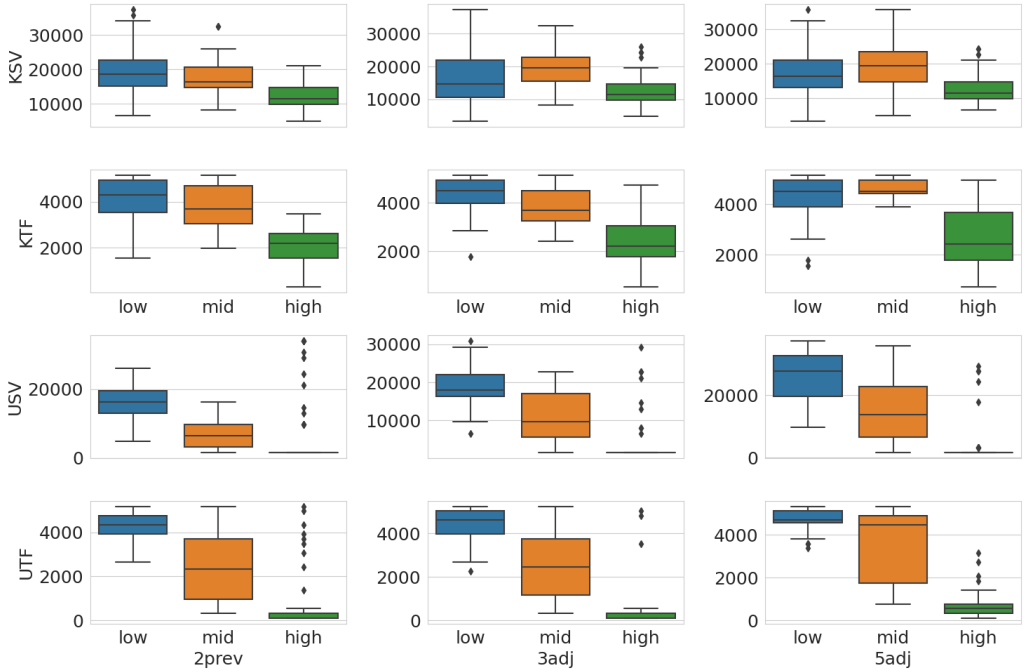

Fig. 18. No. of parameters required for best imputation error using KSV-KTF-USV-UTF on Shanghai dataset

```
    while not converged:
        estimateU(incmatrices, masks, S, V)
        estimateV(incmatrices, masks, S, V)
        estimateS(incmatrices, masks, U, V)
        imputedmatrices = [U@diag(S[i])@V.T for i in range(nmats)]
        diffstack = vstack(incmatrices)-vstack(imputedmatrices)
        maskstack = vstack(masks)
        nr = norm(diffstack * maskstack)
        dr = norm(vstack(incmatrices) * maskstack)
        converged = nr / dr < 1e-5

    return U @ diag(S) @ V
```

A.8 KTF

```
def estimateAllParams(completetensor, rank):
    weights, (A, B, C) = \
    td.non_negative_parafac(completetensor, rank)
    ones = ones(1)
    factors = []
    for i in range(rank):
        factors += [cp2t((ones, (A[:, i : i + 1],
                    B[:, i : i + 1],
                    C[:, i : i + 1])))]
    return weights, factors

def estimateTimeVaryingParams(incompletetensor, factors, mask):
    m = prod(incompletetensor.shape)
    n = len(incompletetensor)
    A = zeros((m, n))
    for i in range(n):
        A[:, i] = (factors[i] * mask).flatten()
    b = (incompletetensor * mask).flatten()
    weights, *_ = lstsq(A, b)
```

```
def generate(weights, factors):
    m,n,p = [factors[i].shape[0] for i in range(3)]
    rank = len(weights)
    tensor = zeros((m, n, p))
    for i in range(rank):
        tensor += weights[i] * factors[i]
    return tensor

def ktf(completetensor, incompletetensor, rank):
    weights, factors = estimateAllParams(completetensor, rank)
    weights = \
    estimateTimeVaryingParams(incompletetensor,factors)
    imputedtensor = generate(weights, factors)
    return imputedtensor
```

A.9 UTF

```
def tensorsarray2hdtensor(tensors):
    for i in range(len(tensors) - 1):
        assert tensors[i].shape == tensors[i + 1].shape

    oldshape = tensors[0].shape
    newshape = (*oldshape, len(tensors))
    hdtensor = zeros(newshape)

    for i in range(len(tensors)):
        hdtensor[..., i] = tensors[i]

    return tl.tensor(hdtensor)


def utf(inctensors, masks, rank):
    N = len(inctensors)
    m, n, p = inctensors[0].shape

    # create the higher-dimensional tensor
    hdtensor = tensorsarray2hdtensor(inctensors)
    hdmask = tensorsarray2hdtensor(masks)
    (weights, factors) = td.non_negative_parafac(
        hdtensor, rank=rank, init="random", return_errors=False,
        mask=hdmask, tol=1e-5
    )
    reconstructed = cp2t(weights, factors)
    return [reconstructed[..., i] for i in range(N)]
```