Spatio-Temporal Tensor Sketching via Adaptive Sampling

Jing Ma ✉, Qiuchen Zhang, Joyce C. Ho, and Li Xiong

Department of Computer Science
Emory University
{jing.ma, qzhan84, joyce.c.ho, lxiong}@emory.edu

Abstract. Mining massive spatio-temporal data can help a variety of real-world applications such as city capacity planning, event management, and social network analysis. The tensor representation can be used to capture the correlation between space and time and simultaneously exploit the latent structure of the spatial and temporal patterns in an unsupervised fashion. However, the increasing volume of spatio-temporal data has made it prohibitively expensive to store and analyze using tensor factorization.

In this paper, we propose SkeTenSmooth, a novel tensor factorization framework that uses adaptive sampling to compress the tensor in a temporally streaming fashion and preserves the underlying global structure. SkeTenSmooth adaptively samples incoming tensor slices according to the detected data dynamics. Thus, the sketches are more representative and informative of the tensor dynamic patterns. In addition, we propose a robust tensor factorization method that can deal with the sketched tensor and recover the original patterns. Experiments on the New York City Yellow Taxi data show that SkeTenSmooth greatly reduces the memory cost and outperforms random sampling and fixed rate sampling method in terms of retaining the underlying patterns.

Keywords: spatio-temporal data · tensor sketching · tensor completion.

1 Introduction

The increasing availability of spatio-temporal data has brought new opportunities in application domains including urban planning, informed driving, and infectious disease spread modeling [2, 8, 24]. Unfortunately, the rapid growth in these data streams can be prohibitively expensive to store, communicate and analyze. In addition, the high-dimensional, multi-aspect spatio-temporal data poses analytic challenges due to the correlations in the measurements from both time and space. Moreover, human-intensive and domain-specific supervised models are not tractable due to the constant and evolving deluge of measurements.

Given the high-dimensional, multi-aspect nature of spatio-temporal data, a tensor serves as an efficient way to represent and model such data [5, 8, 24]. As an example, each element of the tensor can represent the occurrences of an

event at a specific location (encoded as latitude and longitude) within a specific time interval. Compared with the low-dimensional matrix-based methods such as [15], tensors can capture the correlation between each mode. Furthermore, tensor factorization offers an unsupervised, data-driven approach to identify the global structure of the data via a high-order decomposition. It is also more interpretable compared to deep learning methods [30]. Unfortunately, existing models require the full tensor information (i.e., all the data samples must be stored) and do not readily scale to extremely large tensors.

The computation and storage limitations of large tensors motivate the need to approximate such tensors with relatively small "sketches" of the original tensors. Not only are these manageable-sized tensor sketches more readily stored on a single machine, the computationally expensive tasks such as tensor decomposition can be performed on the smaller tensors while still preserving the underlying structure. Although sketching has been proposed as a linear algebra tool for reducing the memory cost, traditional methods involve the linear transformation of the original tensor with a "fat" random projection matrix to reduce the dimensions [21,27]. However, the transformation has inherent shortcomings: 1) the design of the random projection matrix may not capture the evolving patterns without a priori information; and 2) continuous sketching is computationally expensive since it involves the inversion of a fat random projection matrix.

An alternative sketching approach is based on tensor sparsification – subsampling the original tensor while preserving the original tensor structure. Compared with the random projection method, sampling incurs negligible online complexity. Existing tensor sparsification methods include random sampling based on the tensor spectral norm [19], sampling according to the entry values [28], and sampling according to the pre-computed tensor distribution [4]. However, these tensor sparsification algorithms suffer from the following limitations: 1) they cannot deal with streaming data where measurements are not available a priori and the tensor is incrementally updated; 2) there is no formal mechanism to reconstruct the original streams [7]. While tensor reconstruction is often a byproduct of dealing with missing data, reconstruction of the original streams can help track dynamic and abrupt changes at particular locations.

In this paper, we propose SkeTenSmooth, a factorization framework that uses adaptive sampling to generate tensor sketches on-the-fly and preserves the underlying global structure. We explore the problem of serially acquired time slices (measurements appear once they are available). We introduce SkeTen, a tensor sketching method that adaptively adjusts the sampling intervals and samples time slices according to the feedback error between the prior estimate and the true value. Thus it can capture the time slices that are not well modeled by the prior forecasting model, while avoiding the storage of the time slices that contain redundant information (i.e., patterns that are already captured). Furthermore, we propose a novel method SkeSmooth to decompose the tensor sketches and reconstruct the underlying temporal trends. Unlike previous tensor factorization algorithms that deal with randomly missing entries, the sketched tensor has missing time slices (i.e., no data from a specific time point) which

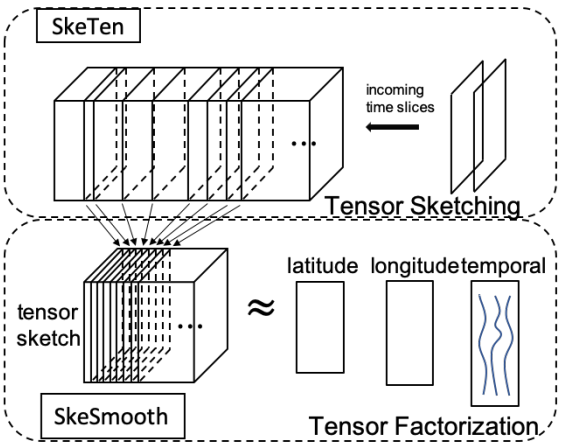


Fig. 1: The overall flow of SkeTenSmooth, including SkeTen and SkeSmooth.

poses additional challenges. Thus, we introduce a robust tensor factorization algorithm that incorporates temporal smoothness constraints using auxiliary information about the data gained from SkeTen. Figure 1 shows the overall flow of the proposed framework. We briefly summarize our contributions as:

1) **Tensor sketching with adaptive sampling.** We propose SkeTen, a tensor sketching technique that helps process large volumes of data with low memory requirements and preserves the underlying temporal dynamics.

2) **Tensor factorization with smoothness constraint.** We propose a tensor factorization framework, SkeSmooth, that decomposes the smaller tensor "sketches" with smoothness constraints to achieve robust recovery of the underlying latent structure and the missing entries.

3) **Case study on New York taxi data.** We illustrate the ability to produce small tensor "sketches" and generate smooth temporal factors on real data.

2 Preliminaries and notations

This section summarizes the notations used in this paper. Note that we use $\mathcal{X}$ to denote a tensor, $\mathbf{X}$ to denote a matrix, and $\mathbf{x}$ to denote a vector. The mode-d matricization of a tensor is denoted by $X_{(n)}$. The row and column vectors are represented by $\mathbf{x}_{i:}, \mathbf{x}_{:r}$ respectively.

2.1 Tensor Factorization

Definition 1. *(Khatri-Rao product). Khatri-Rao product is the "columnwise" Kronecker product of two matrices* $\mathbf{A} \in \mathbb{R}^{I \times R}$ *and* $\mathbf{B} \in \mathbb{R}^{J \times R}$*. The result is a matrix of size* $(IJ \times R)$ *and defined by* $\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1 \cdots \mathbf{a}_R \otimes \mathbf{b}_R]$*.* $\otimes$ *denotes*

the Kronecker product. The Kronecker product of two vectors $\mathbf{a} \in \mathbb{R}^I$, $\mathbf{b} \in \mathbb{R}^J$ *is*

$$\mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_1\mathbf{b} \\ \vdots \\ a_I\mathbf{b} \end{bmatrix}$$

Definition 2. *(CANDECOMP-PARAFAC Decomposition). The CANDECOMP-PARAFAC (CP) decomposition is to approximate the original tensor* $\mathcal{Y}$ *by the sum of* R *rank-one tensors where* R *is the rank of tensor* $\mathcal{Y}$*. For a three-mode tensor* $\mathcal{Y} \in \mathbb{R}^{I \times J \times K}$*, the CP decomposition can be represented as*

$$\mathcal{O} \approx \mathcal{X} = \sum_{r=1}^{R} \boldsymbol{a}_{:r} \circ \boldsymbol{b}_{:r} \circ \boldsymbol{c}_{:r}, \tag{1}$$

where $\boldsymbol{a}_{:r} \in \mathbb{R}^I$, $\boldsymbol{b}_{:r} \in \mathbb{R}^J$, $\boldsymbol{c}_{:r} \in \mathbb{R}^K$ *are the* r*-th column vectors within the three factor matrices* $\boldsymbol{A} \in \mathbb{R}^{I \times R}$, $\boldsymbol{B} \in \mathbb{R}^{J \times R}$, $\boldsymbol{C} \in \mathbb{R}^{K \times R}$, $\circ$ *denotes the outer product.*

In this paper, the spatio-temporal tensor has three dimensions. The first dimension is the temporal dimension, while the other two are longitude and latitude dimensions representing the spatial modes.

3 SkeTen: Tensor Sketching via Adaptive Sampling

The main idea of SkeTen is to discard slices along the temporal mode that are "predictable". Intuitively, this adaptive sampling strategy will capture sudden temporal changes of the incoming streams, which is difficult for other methods such as random sampling and fixed rate sampling to capture. Moreover, SkeTen does not require a priori knowledge of the data and can adjust the sampling interval in real-time. SkeTen consists of two main components. The adaptive sampling module uses a prediction model to measure the "predictability" of the data. The second component selects random time series projections to reduce the training time of the prediction models.

3.1 Adaptive Sampling

SkeTen introduces an adaptive sampling method that can adjust according to the detected temporal dynamics and thus can better capture the underlying patterns even with the same amount of data stored. In SkeTen, individual time-series prediction models are developed for each spatial location. The adaptive sampling strategy then analyzes the slice feedback error, or the feedback error across all the locations.

Definition 3. *(Slice Feedback Error). Each tensor fiber along the temporal mode,* $\mathcal{X}(:, j, k)$*, is denoted as a time-series stream,* $\boldsymbol{x}_{j,k}$*. If* $\boldsymbol{x}_{j,k}^{(t)}$ *represents the true value*

at a particular time t, where $0 \leq t \leq T$, then the slice feedback error at time t, E_t is defined as:

$$E_t = \sum_{k=1}^{K} \sum_{j=1}^{J} \left| \hat{\boldsymbol{x}}_{j,k}^{(t)} - \boldsymbol{x}_{j,k}^{(t)} \right| / \max\{\boldsymbol{x}_{j,k}^{(t)}, \delta\}, \tag{2}$$

where $\hat{\boldsymbol{x}}_{j,k}^{(t)}$ is the estimated value based on the prediction model, and δ is a small sanity bound.

The slice feedback error reflects how well the current time series models fit the current trends for each time series fiber. If there is a sudden increase across multiple spatial locations (i.e., an increase in the slice feedback error), then the sampling interval should be shortened to better capture the evolving trends.

Time Series Prediction Model SkeTen uses an Autoregressive Integrated Moving Average (ARIMA) model to predict the values of the temporal slices. Given a time-series sequence $\{x_i\}$, $i = 1, \cdots, t$, where i is the time index, ARIMA(p, d, q) is defined as

$$\nabla^d x_t = \sum_{j=1}^{p} \alpha_j x_{t-j} + \varepsilon_t + \sum_{j=1}^{q} \beta_j \varepsilon_{t-j} + c \tag{3}$$

where p is the order of lags of the autoregressive model, d is the degree of differencing, and q is the order of the moving average model. $\alpha_j, j = 1, \cdots, p$ and $\beta_j, j = 1, \cdots, q$ represent the coefficients of AR and MA, respectively, ε_t is the white noise at time t, and c is the bias term. An ARIMA model is trained on each tensor fiber for the initial sampled time slices.

ARIMA has several convenient properties that make it more suitable than other time series prediction models in our setting. (1) The auto-regressive part of ARIMA ($\{\alpha_1, \cdots, \alpha_p\}$) generates the coefficients that are later exploited in SkeSmooth for the smoothness constraint in Sec. 4.1. This is essential for recovering the original temporal patterns with missing time slices. (2) It doesn't require a large number of training samples so the compression can occur much earlier than more complex models. (3) The coefficients can be used to understand the adaptive sampling for the PID controller which we will explain next. In preliminary experiments, we tried several state-of-the-art deep learning models such as ConvLSTM [29], but the compression rates and sampled slices only improved marginally. Thus, ARIMA is an optimal choice for our setting.

Feedback Control SkeTen uses a controller system to detect rapid changes in the slice feedback error and adaptively adjusts the sampling rate. We adopt a PID controller similar to [6, 13] to change the sampling interval over time. The PID controls the *Proportional, Integral* and *Derivative* errors. *Proportional error* is defined as $\Gamma_p = \gamma_p E_t$ to control the sampling intervals by keeping the controller proportional to the current slice feedback error. *Integral error* is

Algorithm 1: SkeTen

Input: ARIMA models for each tensor fiber, incoming tensor slices $\mathcal{X}(:, j, k), (j \leq J, k \leq K)$, set of PID controller parameters $\{\gamma_p, \gamma_i, \gamma_d\}$, next sampling point ns

```
while not reach the end of data streams do
    if next time slice t == ns then
        sample the next time slice to the sketched tensor;
        compute slice feedback error according to eq. (2);
        compute the PID controller error Γ according to eq. (4);
        obtain the new interval according to eq. (5);
        update the next sampling point ns = ns + newInterval;
    else
        move on to the next time slice.
```

defined as $\Gamma_i = \frac{\gamma_i}{M_t} \sum_{m=0}^{M_t} E_t$, where M_t represents how many errors have been taken until time t. Thus the integral error considers the past errors to eliminate offset. *Derivative error* is defined as $\Gamma_d = \gamma_d \frac{E_t - E_{t-1}}{t_m - t_{m-1}}$ to prevent large errors in the future. Therefore, the full PID controller is defined as

$$\begin{aligned} \Gamma &= \Gamma_p + \Gamma_i + \Gamma_d, \\ \text{s.t. } &\gamma_p, \gamma_i, \gamma_d \geq 0, \ \gamma_p + \gamma_i + \gamma_d = 1. \end{aligned} \tag{4}$$

The PID errors can thus be interpreted as control actions based on the past, the present and the future [3]. We set the PID parameters according to the common practice that *proportional* > *integral* > *derivative* [6]. With the PID error, the sampling interval is adjusted according to

$$newInterval = \max\{1, \theta(1 - e^{\frac{\Gamma - \xi}{\xi}})\}, \tag{5}$$

where θ and ξ are predefined user-specific parameters. The adaptive sampling process is presented in Algorithm 1.

3.2 Random Projection of ARIMA Coefficients

For large scale spatio-temporal data with many spatial locations (countless values of longitude and latitude), training the prediction model (i.e., ARIMA) may be computationally infeasible. To reduce the computational cost and retain the temporal fidelity of the data, we propose a random projection algorithm to train a limited number of ARIMA models, while preserving the competitive performance of the trained models. Inspired by the Locality Sensitive Hashing (LSH) algorithm, which has demonstrated its success in Approximate Nearest Neighbor search, our proposed projection method has the following steps: (1) train L ARIMA models $g_i, i = 1, ..., L$ from L time series $ts_i, i = 1, ..., L$, where each of

Algorithm 2: Random Projection of ARIMA Coefficients

Input: Tensor fiber set $\mathcal{T} = \{ts_1, ..., ts_M\}$, number of models to train L

for *i=1,...,L* **do**
  train ARIMA model g_i for the randomly selected time series ts_i
for *j=1,...,M* **do**
  for *i=1,...,L* **do**
    map each ts_j into bucket i according to $arg\min_{g_i} ||g_i(ts_j) - ts_j||$

the L time series are chosen at random from the time series set $\mathcal{T}$ containing all tensor fibers $ts_j, j = 1, ..., M$; (2) construct L buckets from the time series set $\mathcal{T}$, each bucket i corresponds to an ARIMA model g_i; (3) map each time series $ts_j, j = 1, ..., M$ from $\mathcal{T}$ into bucket i according to the RMSE between the true and the predicted values using the ARIMA model g_i. In this way, we treat the fitting of each time series to the randomly selected time series as a special "random projection", and then map each time series according to the least RMSE to each bucket i. This process (detailed in Algorithm 2) allows locations that may not have been spatially close to be clustered to the same model and can decrease the training time, which is the bottleneck of the adaptive sampling process.

We compare our proposed random projection algorithm for the ARIMA coefficients with the k-means algorithm where similar time series can also be clustered and thereby reducing training time. Fig. 2 (c), (d) demonstrate that despite the high computation cost of the k-means algorithm, it provided limited improvement over the random projection algorithm.

4 SkeSmooth: Smooth Tensor Factorization

We then consider analyzing the sketched tensor using the CP decomposition to capture the underlying multi-linear latent structure of the data and reconstruct the unsampled time slices. Built on the success of the Quadratic Variation (QV) matrix constraint [30, 33], we consider the delay effect of the time series and utilize the coefficients from the pre-trained ARIMA model from SkeTen to make the temporal pattern smooth and robust to missing entire time slices.

4.1 Formulation

CP Decomposition with Missing Data. We consider the sketched tensor as an incomplete data problem where the goal is to both learn the underlying patterns of the data and reconstruct the unsampled values. Previous CP-based tensor completion algorithms [1,23] have formulated the completed tensor $\overline{\mathcal{X}}$ as:

$$\overline{\mathcal{X}} = \mathcal{W} * \mathcal{X} + (1 - \mathcal{W}) * [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!], \tag{6}$$

where $*$ denotes the element-wise product, the observed tensor is factorized as $\mathcal{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ and the binary weight tensor $\mathcal{W}$ is of the same size as $\mathcal{X}$ such that

$$w_{i,j,k} = \begin{cases} 1, & \text{if } w_{i,j,k} \text{ is known} \\ 0, & \text{if } w_{i,j,k} \text{ is missing.} \end{cases}$$

ARIMA-based Regularization Matrix. Standard CP decomposition with missing data assumes the entries are missing at random, whereas SkeTen yields a sketched tensor with entire time slices missing. Moreover, they do not account for temporal information which can improve interpretability and robustness. In spatio-temporal datasets, successive observations at the same spatial location (i.e., $x_{j,k}^{(t)}$ and $x_{j,k}^{(t+1)}$) are unlikely to change significantly. Thus, adjacent time slots are generally smooth. SkeSmooth incorporates the coefficients generated by the various ARIMA models as auxiliary information into the regularization matrix $\mathcal{L}$. Based on Eq. (3), we observe that $\mathbf{a}_i$ can be approximated as:

$$\mathbf{a}_i = \alpha_1 \mathbf{a}_{i-1} + \alpha_2 \mathbf{a}_{i-2} + \cdots + \alpha_p \mathbf{a}_{i-p}, \ p < i \tag{7}$$

Thus by utilizing the learned predictive models in SkeTen (the autoregressive coefficients and the order of the number of time lags, p), the algorithm can better reconstruct the missing time slices and learn the underlying patterns.

Although each ARIMA model may have different order of time lags, $p_{j,k}$, we use the largest order $p = \max p_{j,k}$ and set the coefficients $\alpha_{p_{j,k}+1}, \cdots, \alpha_p$, to zero if $p_{j,k} < p$. Since each tensor fiber is clustered to the ARIMA model that fits best, we can thus get the weight for each ARIMA model as the number of tensor fibers clustered to it. We then compute the weighted average of the autoregressive coefficients to produce the regularization matrix, $\mathcal{L}$, defined as:

$$\mathcal{L} = \begin{bmatrix} -\alpha_p & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -\alpha_1 & \cdots & -\alpha_p & 0 & \cdots & 0 \\ 1 & -\alpha_1 & \cdots & -\alpha_p & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & -\alpha_1 & \cdots & -\alpha_p \end{bmatrix} \tag{8}$$

SkeSmooth Optimization Problem. The objective function for SkeSmooth after adding the smoothness constraint is:

$$\min_{\mathcal{X}} \frac{1}{2} \|\mathcal{Y} - \mathcal{W} * [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F^2 + \frac{\rho}{2} \|\mathcal{L}\mathbf{A}\|_F^2, \quad \text{s.t. } \mathcal{Y} = \mathcal{W} * \mathcal{X}, \tag{9}$$

where $\mathcal{Y}$ denotes the sketched tensor with missing entries. Thus higher values of the regularization parameter, ρ, will yield smoother temporal factors. However, this may potentially reduce the overall fit of the tensor.

Algorithm 3: SkeSmooth

Input: Sparse tensor sketch $\mathcal{Y}$, weight tensor $\mathcal{W} \in \mathbb{R}^{I \times J \times K}$, smooth parameter ρ.

Randomly initialize the factor matrices $\mathbf{A} \in \mathbb{R}^{I \times R}, \mathbf{B} \in \mathbb{R}^{J \times R}, \mathbf{C} \in \mathbb{R}^{K \times R}$

while $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ *not converge* **do**

  compute tensor $\mathcal{Z} = \mathcal{W} * [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$;

  compute function value $f = \frac{1}{2}\|\mathcal{Y} - \mathcal{Z}\|_F^2$;

  update gradient G^n according to Eq. (10);

  update $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ with l-bfgs.

4.2 Algorithm

We use a first-order method to solve the optimization problem, similar to [1]. The gradient-based method has been shown to be robust to overspecification of the rank. As the regularization matrix is only enforced on the temporal mode (first mode), only the gradient for the temporal mode involves the gradient of the regularization matrix. Thus the updates for all three modes are:

$$\begin{aligned} \mathbf{G}^{(1)} &= -(\mathcal{Y}_{(1)} - \mathcal{Z}_{(1)})(\mathbf{C} \odot \mathbf{B}) + \rho \mathcal{L}^T \mathcal{L} \mathbf{A}; \\ \mathbf{G}^{(2)} &= -(\mathcal{Y}_{(2)} - \mathcal{Z}_{(2)})(\mathbf{C} \odot \mathbf{A}); \\ \mathbf{G}^{(3)} &= -(\mathcal{Y}_{(3)} - \mathcal{Z}_{(3)})(\mathbf{B} \odot \mathbf{A}); \\ \text{s.t. } \mathcal{Z} &= \mathcal{W} * [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!], \end{aligned} \tag{10}$$

where $\mathcal{Y}_{(n)}$ is the matricization on the n-th mode of the tensor sketch. Our gradient-based algorithm uses the limited-memory BFGS for the first order optimization. The optimization process is shown in Algorithm 3.

4.3 Complexity Analysis

The time complexity of SkeSmooth is dominated by the matricized tensor times Khatri-Rao product (MTTKRP) operations, which is computed in Eq. (10) as the matricized residual tensor $\mathcal{Y}_{(n)} - \mathcal{Z}_{(n)}$ times the Khatri-Rao product between two factor matrices. For simplicity purpose, we denote an N-th order tensor $\mathcal{X} \in \mathbb{R}^{I \times \cdots \times I}$ as the representation of the residual tensor $\mathcal{Y}_{(n)} - \mathcal{Z}_{(n)}$. For each mode, computing MTTKRP takes $O(nnz(\mathcal{X})R)$, where R is the approximate rank of the tensor, $nnz(\mathcal{X})$ represents the number of non-zero elements of $\mathcal{X}$. While computing the term $\rho \mathcal{L}^T \mathcal{L} \mathbf{A}$ for the first mode takes $O(I^2 R^2)$. Thus the overall time complexity for SkeSmooth is $O(nnz(\mathcal{X})NR + I^2 R^2)$.

5 Experiments

We evaluate SkeTenSmooth on a large real-world dataset, the New York City (NYC) Yellow Taxi data (see [18] for experiment settings). The goal of our evaluation is to assess both SkeTen and SkeSmooth from three aspects:

1. **Effectiveness of SkeTen:** Analyze memory compression without hurting the performance. Use SkeTen to evaluate the latent structure preservation.
2. **Effectiveness of SkeSmooth:** Evaluate with the sketched tensor on the effectiveness of the smoothness constraint.
3. **Sketching result on prediction tasks:** Analyze how the decomposed factor matrices perform on a downstream prediction task, to indirectly evaluate whether the sketching results are meaningful.

5.1 Dataset

We evaluate SkeTenSmooth on the NYC Yellow Taxi Data[1], which is collected and published by the New York City Taxi and Limousine Commission. To demonstrate the scalability of our methods, we collect data from January, 2009 to June, 2015, which has 1,090,939,222 trips in total, on a daily basis[2]. We investigate the NYC area using the latitude range of $[40.66, 40.86]$ and longitude range of $[-74.03, -73.91]$. We partition the NYC area with a 0.001×0.001 degree grid, where 0.001 degree is roughly 111.32 meters. Then we form a tensor $\mathcal{X}$ of size $2341 \times 120 \times 200$, for time, latitude and longitude modes respectively. The first 60% of the tensor is used for training and is thus considered as offline data, while the latter 40% is used to test SkeTen and SkeSmooth.

5.2 Evaluation Metrics

The tensor decomposition performance is evaluated by two criteria: Factor Match Scores (FMS) and Tensor Completion Scores (TCS) [1]. FMS is defined as:

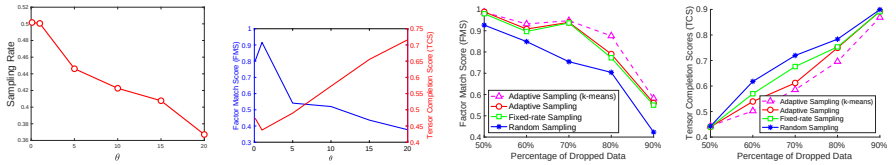
$$score(\bar{\mathcal{X}}) = \frac{1}{R} \sum_{r} \left(1 - \frac{\left|\xi_r - \bar{\xi_r}\right|}{max\left\{\xi_r, \bar{\xi_r}\right\}}\right) \prod_{\mathbf{x}=\mathbf{a},\mathbf{b},\mathbf{c}} \frac{\mathbf{x}_r^T \bar{\mathbf{x}}_r}{\|\mathbf{x}_r\| \|\bar{\mathbf{x}}_r\|},$$

$$\xi_r = \prod_{\mathbf{x}=\mathbf{a},\mathbf{b},\mathbf{c}} \|\mathbf{x}_r\|, \bar{\xi_r} = \prod_{\mathbf{x}=\mathbf{a},\mathbf{b},\mathbf{c}} \|\bar{\mathbf{x}}_r\|$$

where $\bar{\mathcal{X}} = [\![\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\mathbf{C}}]\!]$ is the estimated factor and $\mathcal{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is the true factor. $\mathbf{x}_r$ is the r^{th} column of factor matrices. FMS measures the similarity between two tensor decomposition solutions. It ranges from 0 to 1, and the best possible FMS is 1. We choose the factorization result of the complete tensor using CP-OPT [1] as a true factor to compare with.

[1] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[2] We also conducted experiments on hourly basis (a finer granularity), which shows consistent results with the daily basis. Results can be found in Supplement in [18].



(a) Sampling rate vs θ (b) FMS/TCS vs θ (c) FMS vs dropped rates (d) TCS vs dropped rates

Fig. 2: Overall sampling rates, FMS and TCS over the change of θ ((a) and (b)). FMS and TCS over dropped rates ((c) and (d)).

TCS indicates the relative error of the missing entries, and is defined as

$$TCS = \frac{\left\|(1-\mathcal{W}) * (\mathcal{X} - \bar{\mathcal{X}})\right\|}{\|(1-\mathcal{W}) * \mathcal{X}\|},$$

The best possible TCS is 0.

5.3 Parameter Selection

In SkeTen, parameters involved in achieving the adaptive sampling algorithm include: the PID controller parameters $\{\gamma_p, \gamma_i, \gamma_d\}$, θ and ξ for the interval adjustment. The PID parameters are set as $\{\gamma_p, \gamma_i, \gamma_d\} = \{0.7, 0.2, 0.1\}$[3]. Parameter θ determines the magnitude of the sampling interval adaptation, its impact is shown in fig. 2(a) and (b). From fig. 2(b), we observe that as θ increases, the reconstruction performance measured by FMS and the TCS decreases due to the enlarged average sampling intervals (shown in fig. 2(a)). The optimal choice shown in the figure is $\theta = 1$. When $\theta = 0.1$, the reconstruction performance (TCS and FMS) is worse than when $\theta = 1$ due to an insufficient interval adjustment. ξ is determined as 3.5 for the tolerance of PID error.

For SkeSmooth, the smoothness constraint ρ is chosen as 600 after grid search through {500, 600, 700, 800, 900, 1000}, and the rank of the tensor is determined as 5 after grid search in {5, 10, 15, 20, 25} (see Supplement in [18] for more details). The number of lags is determined as $p = 3$ based on the majority of all the trained ARIMA models. We use the number of tensor fibers mapped into the same bucket as weights, and compute the final coefficients for the regularization by weighted average of the coefficients of the randomly trained ARIMA models and is determined as $\boldsymbol{\alpha} = [0.55, -0.19, 0.04]$[4].

[3] We experimented with several PID settings, and it hardly affected the sampling rate as long as we set the controller ID according to *proportional* > *integral* > *derivative*. We thus consider our system robust to the control parameters and choose the optimal setting in empirical studies.

[4] The k-means based coefficients are computed the same way using the Dynamic Time Warping (DTW) distance and are determined as $\boldsymbol{\alpha}_{k-means} = [0.51, -0.07, 0.007]$.

5.4 Tensor Sketching

Here we consider a streaming setup where the tensor sketching will be applied to a succession of incoming time slices.

Baselines We compare SkeTen with two baseline sampling techniques:

- **Fixed rate sampling** samples the data with a predefined, fixed interval.
- **Random sampling** has been proposed as a way to sparsify the tensor [19].

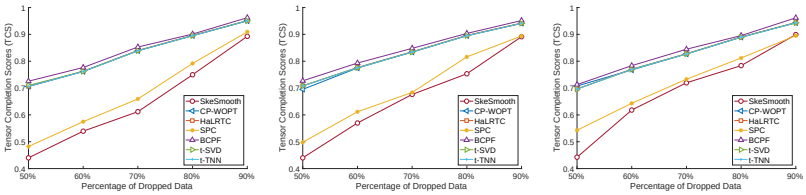
By evaluating the performance of fixed rate sampling and random sampling, we can gain a better understanding of the benefits of adaptive sampling under the setting of time-evolving streams and how it may provide more informative representations of the streams at the same storage cost.

Evaluation We evaluate the three sampling methods for different levels of dropped data by using the proposed SkeSmooth algorithm. The performance is measured by both FMS and TCS. We explore the sketching level from 50% to 90%. Fig. 2 (c) and (d) show the FMS and TCS of different sampling methods when applied to SkeSmooth algorithm. As more data are sampled, FMS gradually approaches 1, which means the extracted temporal patterns are equivalent to the true patterns. Notice that when the level of sampling is 50%, adaptive sampling and fixed-rate sampling method show FMS as high as 0.9886 and 0.9803, which means that with only 50% of the original data, SkeSmooth can successfully recover the original temporal pattern. Adaptive sampling achieves slightly better reconstruction performance than fixed-rate sampling, because the daily taxi data are fairly regular and periodical so the interval adjustments for adaptive sampling helps marginally. Both adaptive sampling and fixed-rate sampling are better than the random sampling, illustrating that carefully designing the sampling strategy is critical for pattern and data reconstruction.

5.5 Tensor Decomposition with Sketches

To evaluate the performance of tensor sketching, we first need to look into how it can preserve the temporal trend and the latent structures on the temporal dimension. We decompose the sketched results to generate the factor matrices as latent patterns, and use these factor matrices to reconstruct the original tensor. We compare SkeSmooth with the following state-of-the-art methods: CP-WOPT [1], HaLRTC [16], SPC [30], BCPF [32], t-SVD [31], t-TNN [12].

Evaluation Since not all baseline algorithms are CP-based tensor completions, it is infeasible to compare the FMS. Therefore, we use TCS as the evaluation metric. Fig. 3 illustrates the TCS as a function of the level of data dropped for different sketching methods. We note that SkeSmooth outperforms the baseline tensor completion algorithms in achieving a lower TCS (relative error), verifying



(a) Adaptive Sampling (b) Fixed rate Sampling (c) Random Sampling

Fig. 3: Tensor Completion Scores Comparison between SkeSmooth and the Tensor Completion baselines for different sampling mechanisms: (a) adaptive sampling; (b) fixed rate sampling; (c) random sampling.

its robustness in dealing with the missed time slices in the sketched tensor. In particular, we observe the noticeable improvement using the ARIMA-based coefficients in comparison with the SPC algorithm which adopts the QV smoothness constraint. This observation is consistent for all sampling mechanisms.

5.6 Prediction Task

It is also critical to analyze how meaningful the generated temporal patterns are by performing a downstream prediction task using the decomposed factor matrices. We fit a multivariate Long short-term memory (LSTM) model, which is well-suited for the traffic demand prediction task [26]. For the prediction task, the taxi demand is predicted on a daily basis where each day's number of pickups are treated as the daily demand with a train-test partition of 70-30. The LSTM model uses the factorized temporal mode factor matrix $\mathbf{A}$ of size $I \times R$, where each feature within $\mathbf{A}$ ($\mathbf{A}_i$; of size $1 \times R$) represents a latent transportation pattern. The LSTM model is configured as one hidden layer of 30 LSTM units with ReLu activation function, followed by an output dense layer[5].

Table 1 presents the root-mean-square error (RMSE) differences for adaptive sampling (SkeTen), fixed rate sampling, and random sampling with varying percentage of data dropped from 50% to 90%. Results demonstrate that adaptive sampling outperforms other two sampling methods in maintaining the predictive power with 60%-80% data dropped. As more data are dropped, the test error (RMSE) increases, indicating that it becomes harder to fit the LSTM model.

6 Related Works

We review the previous work regarding the tensor sketching and tensor completion approaches in this section.

[5] The model is trained by the RMSprop optimizer with a fixed learning rate (0.01) and a batch size of 8 for 50 epochs.

Table 1: Predictive performance (test RMSEs) for adaptive sampling, fixed rate sampling, and random sampling with 50% to 90% of data dropped.

Sampling techniques	**50%**	**60%**	**70%**	**80%**	**90%**
Adaptive sampling	0.0155	**0.0270**	**0.0303**	**0.02827**	0.0450
Fixed-rate sampling	0.0173	0.0278	0.0309	0.0427	0.0452
Random sampling	**0.0154**	0.0282	0.0317	0.0342	**0.0322**

6.1 Tensor Sketching

Sketching has been proposed and investigated as an indispensable numerical linear algebra tool to help process large volume of data [27]. Thus far, there have been two research directions. Random projection is the predominant form of tensor sketching. [20] developed an approach to randomly compress a big tensor into a much smaller tensor by multiplying the tensor with a random matrix on each mode. [21] extended this work to improve the permutation matching performances of the resulting factor matrices. [10] expanded the work further to the online setting. However, these compression methods cannot avoid the inherent bottleneck of the matrix-tensor multiplication operators.

An alternative for tensor sketching is tensor sparsification. The goal of tensor sparsification is to generate a sparse "sketch" of the large tensor. [19] proposed random sampling based on the tensor spectral norm. Entries are sampled with probability proportional to the magnitude of the entries. [28] extended this work with: 1) the extension from cubic tensors to general tensors; 2) the criteria and treatment of "small" entry values. They proposed to keep all large entries, sample proportionally to moderate entries, and uniformly sample small entries. [4] proposed a new sampling method that samples entries based on a pre-computed tensor distribution. These tensor sparsification algorithms do not consider the data streaming setting and require prior knowledge about tensors.

6.2 Tensor Factorization and Completion

Tensor completion is more considered as a byproduct when dealing with missing data in the tensor factorization process. As a result, CP decomposition and Tucker decomposition are two major types of tensor completion methods [23]. [1] proposed CP-WOPT, a tensor CP decomposition algorithm that can handle missing entries using a binary weight tensor. Tucker-based tensor completion including geomCG [14], HaLRTC/FaLRTC [16], and TNCP [17], etc. There are other approaches besides CP and Tucker decomposition, such as t-SVD [31]. [23] and [22] provide comprehensive surveys on the topic of tensor completion.

Tensor completion has also been widely applied to spatio-temporal data analysis for traffic prediction [11, 25], urban mobility pattern mining [24], and urban event detection [5]. [33] applied a similar tensor completion algorithm with smoothness constraints as in [30] and [33] to the imputation of missing internet traffic data, that tries to minimize the difference between adjacent time slots. [9]

explored the similarity for each element within one dimension and used this as auxiliary information to better recover the original tensor. Nevertheless, the above algorithms did not consider the tensor streaming problem. [25] proposed a dynamic tensor completion (DTC) algorithm, where traffic data are represented as a dynamic tensor pattern, but will still require the entire tensor.

7 Conclusions

SkeTenSmooth is a tensor factorization framework that generates tensor sketches in a streaming fashion using adaptive sampling mechanism and decomposes the sampled smaller tensor sketch with an ARIMA-based smoothness constraint. It is well suited for the incrementally increased spatio-temporal data analysis that can greatly reduce the storage cost and sample complexity, while preserving the latent global structure. SkeTenSmooth includes two main components: 1) SkeTen as a tensor sketching algorithm that tackles the memory efficiency issue by adaptively adjusting the sampling interval based on the model prediction error; and 2) SkeSmooth incorporates an ARIMA-based smoothness constraint, which better demonstrates the auto correlation between each time point of a time series. Experiments on the large-scale NYC taxi dataset illustrate that with adaptive sampling strategy, SkeTen can greatly reduce the memory cost by 80% and still preserve the latent temporal patterns and the predictive power. Future work will focus on the distributed setting to improve scalability and efficiency.

Acknowledgment. This work was supported by the National Science Foundation, award IIS-#1838200 and CNS-#2027783.

References

1. Acar, E., Dunlavy, D.M., Kolda, T.G., Mørup, M.: Scalable tensor factorizations for incomplete data. Chemometrics and Intelligent Laboratory Systems **106**(1), 41–56 (2011)
2. Angulo, J., Yu, H.L., Langousis, A., Kolovos, A., Wang, J., Madrid, A.E., Christakos, G.: Spatiotemporal infectious disease modeling: a bme-sir approach. PloS one **8**(9), e72168 (2013)
3. Åström, K.J., Wittenmark, B.: Computer-controlled systems: theory and design. Courier Corporation (2013)
4. Bhojanapalli, S., Sanghavi, S.: A new sampling technique for tensors. arXiv preprint arXiv:1502.05023 (2015)
5. Chen, L., Jakubowicz, J., Yang, D., Zhang, D., Pan, G.: Fine-grained urban event detection and characterization based on tensor cofactorization. IEEE Transactions on Human-Machine Systems **47**(3), 380–391 (2016)
6. Fan, L., Xiong, L.: An adaptive approach to real-time aggregate monitoring with differential privacy. IEEE TKDE **26**(9), 2094–2106 (2013)
7. Gama, F., Marques, A.G., Mateos, G., Ribeiro, A.: Rethinking sketching as sampling: Linear transforms of graph signals. In: 2016 50th Asilomar Conference on Signals, Systems and Computers. pp. 522–526. IEEE (2016)

8. Gauvin, L., Panisson, A., Cattuto, C.: Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach. PloS one **9**(1), e86028 (2014)
9. Ge, H., Caverlee, J., Zhang, N., Squicciarini, A.: Uncovering the spatio-temporal dynamics of memes in the presence of incomplete information. In: Proceedings of the 25th CIKM. pp. 1493–1502. ACM (2016)
10. Gujral, E., Pasricha, R., Yang, T., Papalexakis, E.E.: Octen: Online compression-based tensor decomposition. arXiv preprint arXiv:1807.01350 (2018)
11. He, L., Qin, Z.T., Bewli, J.: Low-rank tensor recovery for geo-demand estimation in online retailing. Procedia Computer Science **53**, 239–247 (2015)
12. Hu, W., Tao, D., Zhang, W., Xie, Y., Yang, Y.: A new low-rank tensor model for video completion. arXiv preprint arXiv:1509.02027 (2015)
13. King, M., et al.: Process control: a practical approach. Wiley Online Library (2011)
14. Kressner, D., Steinlechner, M., Vandereycken, B.: Low-rank tensor completion by riemannian optimization. BIT Numerical Mathematics **54**(2), 447–468 (2014)
15. Lakhina, A., Papagiannaki, K., Crovella, M., Diot, C., Kolaczyk, E.D., Taft, N.: Structural analysis of network traffic flows. In: ACM SIGMETRICS Performance evaluation review. vol. 32, pp. 61–72. ACM (2004)
16. Liu, J., Musialski, P., Wonka, P., Ye, J.: Tensor completion for estimating missing values in visual data. IEEE transactions on pattern analysis and machine intelligence **35**(1), 208–220 (2012)
17. Liu, Y., Shang, F., Jiao, L., Cheng, J., Cheng, H.: Trace norm regularized candecomp/parafac decomposition with missing data. IEEE transactions on cybernetics **45**(11), 2437–2448 (2014)
18. Ma, J., Zhang, Q., Ho, J.C., Xiong, L.: Spatio-temporal tensor sketching via adaptive sampling. arXiv preprint arXiv:2006.11943 (2020)
19. Nguyen, N.H., Drineas, P., Tran, T.D.: Tensor sparsification via a bound on the spectral norm of random tensors. arXiv preprint arXiv:1005.4732 (2010)
20. Sidiropoulos, N.D., Kyrillidis, A.: Multi-way compressed sensing for sparse low-rank tensors. IEEE Signal Processing Letters **19**(11), 757–760 (2012)
21. Sidiropoulos, N.D., Papalexakis, E.E., Faloutsos, C.: Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition. IEEE Signal Processing Magazine **31**(5), 57–70 (2014)
22. Sobral, A., Zahzah, E.: Matrix and tensor completion algorithms for background model initialization: A comparative evaluation. Pattern Recognition Letters (2016). https://doi.org/10.1016/j.patrec.2016.12.019
23. Song, Q., Ge, H., Caverlee, J., Hu, X.: Tensor completion algorithms in big data analytics. ACM TKDD **13**(1), 6 (2019)
24. Sun, L., Axhausen, K.W.: Understanding urban mobility patterns with a probabilistic tensor factorization framework. Transportation Research Part B: Methodological **91**, 511–524 (2016)
25. Tan, H., Wu, Y., Shen, B., Jin, P.J., Ran, B.: Short-term traffic prediction based on dynamic tensor completion. IEEE Transactions on Intelligent Transportation Systems **17**(8), 2123–2133 (2016)
26. Tsai, T.H., Lee, C.K., Wei, C.H.: Neural network based temporal feature models for short-term railway passenger demand forecasting. Expert Systems with Applications **36**(2), 3728–3736 (2009)
27. Woodruff, D.P., et al.: Sketching as a tool for numerical linear algebra. Foundations and Trends® in Theoretical Computer Science **10**(1–2), 1–157 (2014)
28. Xia, D., Yuan, M.: Effective tensor sketching via sparsification. arXiv preprint arXiv:1710.11298 (2017)

29. Xingjian, S., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.c.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: NIPS. pp. 802–810 (2015)
30. Yokota, T., Zhao, Q., Cichocki, A.: Smooth parafac decomposition for tensor completion. IEEE Transactions on Signal Processing **64**(20), 5423–5436 (2016)
31. Zhang, Z., Ely, G., Aeron, S., Hao, N., Kilmer, M.: Novel methods for multilinear data completion and de-noising based on tensor-svd. In: CVPR. pp. 3842–3849 (2014)
32. Zhao, Q., Zhang, L., Cichocki, A.: Bayesian cp factorization of incomplete tensors with automatic rank determination. IEEE transactions on pattern analysis and machine intelligence **37**(9), 1751–1763 (2015)
33. Zhou, H., Zhang, D., Xie, K., Chen, Y.: Spatio-temporal tensor completion for imputing missing internet traffic data. In: 2015 34th ipccc. pp. 1–7. IEEE (2015)