IEEE INFOCOM 2022 - IEEE Conference on Computer Communications | 978-1-6654-5822-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INFOCOM48880.2022.9796720

NMMF-Stream: A Fast and Accurate Stream-Processing Scheme for Network Monitoring Data Recovery

Kun Xie [1], Ruotian Xie[1*], Xin Wang[2], Gaogang Xie[3], Dafang Zhang[1], Jigang Wen[1]
[1] College of Computer Science and Electronic Engineering , Hunan University, Changsha 410082, China
[2] Department of Electrical and Computer Engineering, State University of New York at Stony Brook, USA
[3] Computer Network lnformation Center, Chinese Academy of Sciences, Beijing, China
Corresponding author *

***Abstract*—Recovery of missing network monitoring data is of great significance for network operation and maintenance tasks such as anomaly detection and traffic prediction. To exploit historical data for more accurate missing data recovery, some recent studies combine the data together as a tensor to learn more features. However, the need of performing high cost data decomposition compromises their speed and accuracy, which makes them difficult to track dynamic features from streaming monitoring data. To ensure fast and accurate recovery of network monitoring data, this paper proposes NMMF-Stream, a stream-processing scheme with a context extraction module and a generation module. To achieve fast feature extraction and missing data filling with a low sampling rate, we propose several novel techniques, including the context extraction based on both positive and negative monitoring data, context validation via measuring the Pointwise Mutual Information, GRU-based temporal feature learning and memorization, and a new composite loss function to guide the fast and accurate data filling. We have done extensive experiments using two real network traffic monitoring data sets and one network latency data set. The experimental results demonstrate that, compared with three baselines, NMMF-Stream can fill the newly arrived monitoring data very quickly with much higher accuracy.**

***Index Terms*—Streaming-process, network monitoring data filling**

I. Introduction

A. Background and motivation

Network measurement [1], [2] is very important for network operation and maintenance, and their associated tasks such as anomaly detection, analysis of the root cause for an event, and traffic prediction. However, network-wide performance monitoring introduces very high measurement costs. For example, for a network consisting of n nodes, the measurement cost under one round of end-to-end network performance measurement (such as packet loss, flow traffic volume, delay, etc) is $O(n^2)$. The accumulated measurement cost of long time monitoring will be huge.

To reduce the measurement cost, network monitoring systems usually adopt a sampling mechanism where only a subset of Origin and Destination (OD) pairs are measured instead of taking full measurements upon all OD pairs. To record the end-to-end network performance data of the whole network in one time slot, a matrix is naturally built with its row denoting the origin node, column denoting the destination node. We call such a matrix Network Monitoring Matrix (NMM). Under the sampling-based mechanism, NMM is usually sparse with a small subset of entries observed from measurements, while the remaining large number of entries are unknown.

Some network applications such as network state tracking, forecasting, and failure recovery require complete network monitoring data and are highly sensitive to the missing of the data. Inferring unknown entries in the NMM from measurement samples becomes a very important problem. However, there are the need and challenge to design a high-precision and fast NMM filling algorithm because of the following reasons:

- **An NMM filling algorithm should fill the NMM as soon as possible to meet the speed requirements from its subsequent tasks**, such as anomaly detection and traffic management. To get the complete data in real time, the missing data filling should be accomplished upon the arrival of each new NMM.
- **An NMM filling algorithm should have the ability to update its model to learn the dynamic changes of network structure and features.** An algorithm should be able to handle the streaming monitoring data during the continuous arrivals of NMMs with the easy support of model update to learn the dynamically changed features.
- **An NMM filling algorithm should have the ability of feature learning from the monitoring data with a low sampling rate and irregular sampling pattern.** Due to the sampling-based mechanism, and further the measurement fault and unavoidable transmission loss, the sampling rate is usually low with the sampling locations randomly distributed as NMMs arrive in different time slots. Even though we can collect several recent NMMs together to learn the hidden correlations for data filling, since the number of observed data samples in the same positions is very small, it causes the difficulty in representation learning and feature extraction for the NMM filling.

B. Prior studies and limitations

With the rapid progress of sparse representation, matrix completion [3]–[9] techniques have attracted more and more recent research. Some studies [3], [5]–[9] design NMM filling algorithms based on matrix completion. According to the matrix completion theory, a matrix can be accurately recovered with a relatively small number of entries if the underlying matrix has a low-rank or approximate low-rank structure. However, these matrix completion-based algorithms only take action upon one NMM. Without considering the correlations of adjacent NMMs, the filling accuracy of this kind of algorithm is not high.

To conquer the above limitations, some recent studies combine multiple NMMs of different time slots to form a tensor and propose tensor completion algorithms [10]–[17] to fill the missing data. Tensor completion is a natural higher-order generalization of matrix completion which can recover a low-rank tensor from sparse observations of its entries. Compared with the two-dimensional matrix-based algorithms, tensor completion can achieve better filling accuracy as it can mine more correlations hidden in the data. However, the filling speed is low with the combining of multiple NMMs to learn the correlations through tensor factorizations such as CANDECOMP/PARAFAC (CP) decomposition [13]–[16] and tucker decomposition [17]. Moreover, when new NMM comes, the trained tensor factorization models for the history data are hardly updated under current tensor factorization techniques. As a result, current tensor completion algorithms can hardly handle streaming NMM data with dynamic and fast feature extraction requirements.

C. Contributions

To provide a fast and dynamic algorithm to accurately fill an NMM upon its arrival, we propose NMMF-Stream, an algorithm that can fill in the missing entries of a stream of network monitoring matrices. NMMF-Stream consists of two modules, a context extraction module for representation learning and a generation module for missing data estimation. We have made the following technique contributions.

In the context extraction module, to extract useful spatial-temporal correlations in historical monitoring data, we propose a training method using both positive and negative monitoring data with the measurement of Pointwise Mutual Information (PMI). Our context extraction module has the following three features:

- The context extraction accuracy will be compromised when the sampling rate is low in an NMM. To overcome the limitations, instead of using purely positive NMMs with partial data actually collected, we also propose the training using negative monitoring data mixed with random noise and historical data collected a long time ago with little relation to the current NMM.
- To ensure accurate data filling, our module extracts from the history data the common features that have higher spatial-temporal correlations with the current NMM by maximizing the PMI between the historical data and the current NMM and minimizing the PMI between the historical data and the negative monitoring data.
- To achieve the dynamic feature extraction for quick filling, instead of adopting a complex neural network structure to validate the effectiveness of the context information, our context extraction module performs the validation through the simple calculation of PMIs and applies GRU mechanism to reuse the module's to efficiently capture historical features to greatly speed up the learning.

In the generation module, we propose a new composite loss function to guide the module training for fast and highly accurate missing data filling. The new loss function has the following features.

- We discover that widely used Mean Square Error (MSE) has a fast convergence speed but low filling accuracy, while using relative error as loss in training can achieve high filling accuracy but suffer from slow convergence speed. To help the generation module achieve fast convergence with high filling accuracy, our proposed composite loss function takes advantage of the good features of both the loss functions while avoiding their drawbacks.

We have done extensive experiments using three real network monitoring data sets. The experiment results demonstrate that our NMMF-Stream can achieve very good performance with high filling accuracy and fast filling speed.

The rest of the paper is organized as follows. Section II presents the related work. The problem definition, symbol definition, and solution overview will be shown in Section III. We present the architecture, training process, and loss metric of the context extraction module in Section IV, and the details of the generation module in Section V. We describe the complete solution in Section VI. Finally, we evaluate the proposed NMMF-Stream in Section VII, and conclude the work in Section VIII.

II. Related work

As introduced in Section I, the task of network monitoring data filling is very important for some subsequent tasks, including network state tracking, forecasting, anomaly detection, and failure recovery. Taking advantage of the low-rank characteristics of the network monitoring data, some studies propose to fill the missing data based on matrix completion [3]–[9]. However, two-dimensional matrix-based algorithms only capture the information within a monitoring matrix while ignoring the correlations among different monitoring matrices.

To further utilize the correlations among different NMMs for more accurate missing data filling, some tensor completion based algorithms are proposed recently. For example, Reshape-Align [10] forms the regular tensor with data from variable-rate measurements and introduces user-domain and temporal-domain factor matrices to translate the matrix completion problem to the tensor completion problem. LTC [11] takes advantage of the stronger local correlation of data to form and recover sub-tensors each with a lower rank. Besides the few studies that apply tensor completion for NMM filling,

there also exist some results on tensor completion algorithms (i.e., BTMF [12], CPnmu [13], CPopt [14], CPwopt [13]) and applications including network imaging [18], signal processing [19], building knowledge base [20], transportation system [21], [22], etc.

Although these algorithms and applications demonstrate the ability of tensor model's feature extraction and representation, these algorithms are designed based on tensor decomposition methods such as high-order CP decomposition [15], [16] and Tucker decomposition [17], which handle the data matrices in a batch and look at the batch as the whole tensor to train the parameters. In this way, the training cost is high and can hardly support quick NMM filling when a new NMM arrives. Moreover, based on CP and Tucker decomposition, these tensor completion algorithms can only extract linear features while ignoring the non-linear features, the filling accuracy is still not high.

The neural network has a strong nonlinear learning ability. Although not targeting for network monitoring data, a few recent studies propose data filling algorithms based on neural networks, which can be classified into two categories. CoSTCo [23] and ONCF [24] are the first category. Like traditional tensor completion algorithms, they combine multiple matrix data together as a tensor and use the whole tensor sampling entries to train the neural network to extract the correlations [25]. However, using multiple matrix data in a batch as a large tensor results in a large training cost will be high and can not support quick data filling when new data are collected. As a second category, [26]–[30] propose to build a data filling network based on auto-encoder or GAN [31] by passing matrix data one by one to train the neural network and learn the common features among the matrices. However, once the network is trained using historical data, network parameters are fixed and can hardly capture new features from upcoming data.

III. Problem and Solution Overview

A. Problem

For a network consisting of N nodes, the end to end (called an OD-pair) network performance indicators (such as packet loss, traffic volume, RTT, etc.) can form an $N \times N$ matrix X. We call X the Network Monitoring Matrix (NMM). The entry $X_{i,j}(1 \leq i, j \leq N)$ denotes the performance indicator from the origin i to the destination j. Specially, for an NMM gathered in the time slot k, we denote the NMM as X^k, and the NMM with no missing value (Ground-Truth) as X'^k.

NMM is generally a sparse matrix because of some entry positions are missing. The reasons can be summarized as follows: 1) **Measurement overhead reduction.** Rather than taking measurements from all OD pairs, only a subset of pairs are measured. 2) **Measurement fault.** The measurement system suffers some faults due to hardware failures, environmental changes, etc. 3) **Transmission loss**. The measurement data may get lost under severe communications and system conditions, including network congestion, node misbehavior, and a transmission of measurement information through an unreliable transport protocol (i.e., UDP).

As the subsequent tasks of network monitoring, network state tracking and forecasting, anomaly detection and failure recovery require complete network monitoring data and are highly sensitive to the missing of data, either because of the un-measurement or the loss.

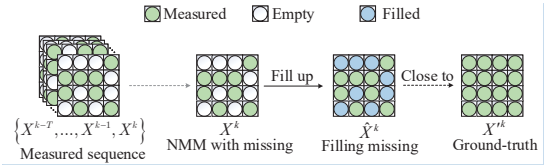

Fig. 1. Notations in this paper.

This paper studies an NMM filling problem which is described as follows: when a newly NMM X^k arrives, we want to quickly fill in the values of the missing positions in X^k to obtain the complete NMM (denoted by $\hat{X}^k$) by using a NMM sequence $\{X^{k-T}, X^{k-T+1}, ..., X^{k-1}, X^k\}$ so that the filled $\hat{X}^k$ can be close to the ground-truth X'^k, where the NMM sequence includes historical NMMs of recent T times slots $\{X^{k-T}, X^{k-T+1}, ..., X^{k-1}\}$ and the newly arrived NMM X^k. We use Fig.1 to illustrate the key concepts involved in the problem definition.

However, as discussed in Section I, due to the quick and dynamic model design requirements under stream processing as well as the low number of entries observed in NMM, achieving quick and accurate NMM filling is specially challenging.

B. Solution overview

We propose a matrix filling approach for the network monitoring based on stream-processing, called NMMF-Stream, with the architecture shown in Fig.2. It runs in the sliding window mode with the window size T on the time axis. Specifically, the inputs of NMMF-Stream (Fig.2 left side) contain two parts: the first part X^k at the left bottom is the current (newly arrived) NMM, the second part $\{X^{k-T}, X^{k-T+1}, ..., X^{k-1}\}$ on the left top is the historical NMM sequence gathered in recent T time slots. NMMF-Stream is mainly composed of two modules, the context extraction module in the blue dashed box, and the generation module in the yellow dashed box.

The context extraction module is used for representation learning whose main task is to extract from the historical sequence the shared (unchanged) information of NMMs, including the spatial and temporal correlations and their mixed relationships. We call these the context information, which is denoted by a low-dimensional vector $c^{k-1} \in \mathbb{R}^M$.

The context extraction module includes Extractor and an auto-regression model (i.e. GRU). It is known that there is a spatial correlation between the internal entries in the NMM. The Extractor can keenly capture the spatial correlation of each historical NMM in $\{X^{k-T}, X^{k-T+1}, ..., X^{k-1}\}$ and output the spatial feature vectors $\{z^{k-T}, z^{k-T+1}, ..., z^{k-1}\}$, where each $X^i \in \mathbb{R}^{N \times N}$ corresponds to spatial feature vector $z^i \in \mathbb{R}^M, k-T+1 \leq i \leq k-1$. After feeding $\{z^{k-T}, z^{k-T+1}, ..., z^{k-1}\}$ to GRU, the context information $c^{k-1} \in \mathbb{R}^M$ in the history data is extracted. c^{k-1} keeps both

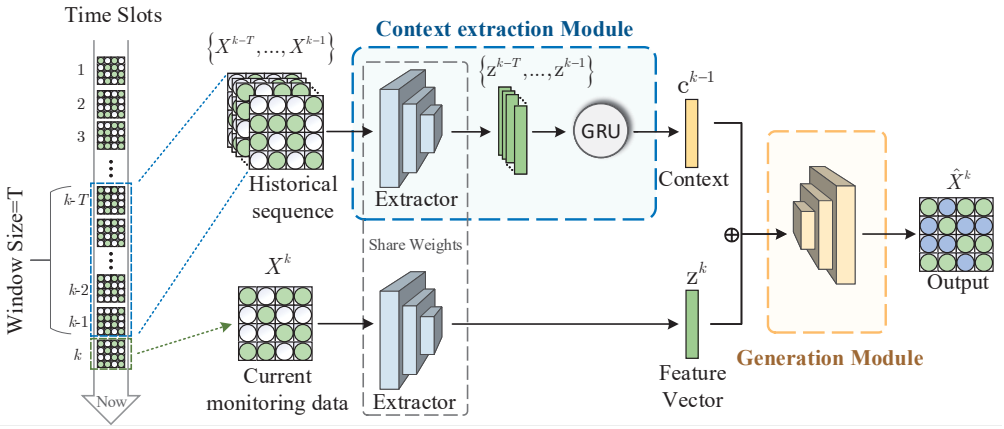

Fig. 2. NMMF-Stream's stream processing framework. The model is mainly composed of a context extraction module and a generation module. The input of the model is the historical measurement data in the past T time slots $\{X^{k-T}, X^{k-T+1}, ..., X^{k-1}\}$ and the measurement data at the time slot k X^k.

temporal and spatial correlations in history data. Compared with the high-dimensional NMM historical sequence, the context c^{k-1} discards the noisy and unrelated information in the historical data and is much smaller.

After the context extraction module is trained, we can exploit the Extractor trained to obtain the spatial feature vector z^k of the current NMM X^k.

The generation module is used to fill the current NMM X^k with the inputs including the spatial feature vector z^k of the current NMM X^k and the context information c^{k-1} of the history data, and output the filled NMM $\hat{X}^k$.

As a stream processing model, both the context extraction module and the generation module need to meet the two requirements, fast training speed and high filling accuracy. In the following Sections IV and V, we will introduce our techniques to achieve the goal.

IV. Context Extraction Module

A. Architecture

The context extraction module is composed with Extractor and GRU, and is responsible for acquiring both spatial and temporal correlations in historical data. The Extractor applies a deep neural network (DNN) that contains several fully connected layers to extract spatial correlations in historical NMMs. From the inputs of the historical monitoring data $\{X^{k-T}, X^{k-T+1}, ..., X^{k-1}\}$, the Extractor outputs a sequence of spatial feature vectors $\{z^{k-T}, z^{k-T+1}, ..., z^{k-1}\}$. Extractor's function can be expressed as follows.

$$z^i = f(W_{Extra}X^i + b_{Extra}) \quad (1)$$

where W_{Extra} denotes Extractor's weights, $f()$ denotes the activation function, and b_{Extra} denotes the bias of the Extractor. To make the output value fall into the range $(0, 1)$, the activation function of the last layer in Extractor is Sigmoid.

As network states generally do not change quickly, there is a strong temporal correlation between current NMM X^k and the historical sequence. To learn the temporal correlation between recent NMMs, NMMF-Stream adopts GRU (Gated Recurrent Unit), a powerful autoregressive model.

GRU is a kind of recurrent neural network (RNN). We adopt GRU in our NMMF-Stream because of the following reasons: The GRU's performance in prediction is more outstanding than the traditional RNN and Hidden Markov Model (HMM). More importantly, compared with existing neural network such as LSTM, GRU utilizes fewer parameters to make it possible to train quickly and adapt to the needs of stream processing.

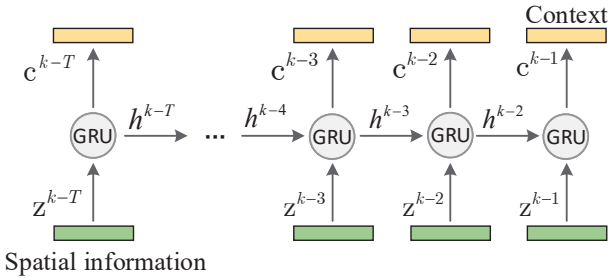

Fig. 3. Extract spatial and temporal context c^{k-1} using GRU.

As shown in Fig.3, multiple steps of processes are involved in GRU. In each step, a spatial feature vector $z^i (k - T \leq i \leq k - 1)$ and a hidden state h^{i-1} from the previous round serve as the input to generate a vector c^i and a hidden state h^i, and the later is further passed to the next step. The last output c^{k-1} is used as the context of the historical sequence to fill in the missing values in the newly arrived NMM in the generation module.

With the GRU's capability of temporal feature learning and memorization, our context extraction module can easily capture the dynamic information in the streaming monitoring data.

B. Training process and loss

1) Training with PMI based correlation measurement

As the context vector c^{k-1} extracted from the historical data will be further utilized in the generation module to infer the missing value in the newly arrived X^k, the accuracy of the information contained in c^{k-1} has a big impact on the quality of data filling.

To train the context extraction module to output more accurate context c^{k-1}, we use the Pointwise Mutual Information

(PMI) as a metric to measure the correlation between current data X^k and the historical context c^{k-1}.

$$\mathrm{PMI}(X^k; c^{k-1}) = log\frac{p(X^k|c^{k-1})}{p(X^k)} \tag{2}$$

The $p(X^k)$ is the probability of getting X^k from their data distribution when a random sampling is adopted, $p(X^k|c^{k-1})$ is the probability of getting X^k when c^{k-1} is known. $\mathrm{PMI}(X^k; c^{k-1})$ approaches the negative infinity when X^k and c^{k-1} are negatively correlated, that is, the relationship between c^{k-1} and X^k is inverse, and there is no possibility of inferring X^k given c^{k-1}. $\mathrm{PMI}(X^k; c^{k-1})$ approaches 0 when X^k and c^{k-1} are independent, and is maximized when X^k and c^{k-1} are perfectly associated. To accurately recover the missing matrix entries, it helps to make c^{k-1} more related to the current NMM X^k, and we can train the model to maximize the $\mathrm{PMI}(X^k; c^{k-1})$ value.

However, the prior probability $p(X^k)$ is unknown, thus, $\mathrm{PMI}(X^k; c^{k-1})$ cannot be directly calculated. In our design, we use a log-bilinear model [14] $f\left(z^k, c^{k-1}\right)$ to approximate $\mathrm{PMI}(X^k; c^{k-1})$, which can be expressed as follows:

$$f\left(z^k, c^{k-1}\right) = \exp\left(\left(z^k\right)^T\left(c^{k-1}\right)\right) \tag{3}$$

where the $\left(z^k\right)^T$ denotes the transpose of $\left(z^k\right)$.

Theorem 1 will show that we can achieve the goal of maximizing $\mathrm{PMI}(X^k; c^{k-1})$ by maximizing $f\left(z^k, c^{k-1}\right)$.

Theorem 1. *The log-bilinear model $f\left(z^k, c^{k-1}\right)$ is proportional to $\frac{p(X^k|c^{k-1})}{p(X^k)}$.*

Proof. Let $\mathbb{X}$ denote the set of all possible NMMs that can be inferred given the context information c^{k-1}. Let X^j be one NMM in $\mathbb{X}$, z^j be the spatial information extracted from X^j by Eq (1). Let $\mathbb{Z}$ denote the set of all z^j extracted from X^j in $\mathbb{X}$. Let $p(Y = X^j|c^{k-1}, \mathbb{X})$ denote the probability of getting X^j when c^{k-1} is given.

According to Eq (2) in [14], we can calculate $p(Y = X^j|c^{k-1}, \mathbb{X})$ as:

$$p(Y = X^j|c^{k-1}, \mathbb{X}) = \frac{\exp\left(\left(z^j\right)^T c^{k-1}\right)}{\sum\limits_{z^i\in\mathbb{Z}} \exp\left(\left(z^i\right)^T c^{k-1}\right)} = \frac{f\left(z^j, c^{k-1}\right)}{\sum\limits_{z^i\in\mathbb{Z}} f\left(z^i, c^{k-1}\right)} \tag{4}$$

Following the Bayes' theorem, the probability of $p(Y = X^j|c^{k-1}, \mathbb{X})$ can be calculated as

$$\begin{aligned} p(Y = X^j|c^{k-1}, \mathbb{X}) &= \frac{p(X^j c^{k-1}) \prod\limits_{l\neq j}^{\mathbb{X}} p(X^l)}{\sum\limits_{X^i\in\mathbb{X}} p(X^i c^{k-1}) \prod\limits_{l\neq i}^{\mathbb{X}} p(X^l)} = \frac{\frac{p(X^j c^{k-1})}{p(X^j)}}{\sum\limits_{X^i\in\mathbb{X}} \frac{p(X^i c^{k-1})}{p(X^i)}} \\ &= \frac{\frac{p(X^j|c^{k-1})p(c^{k-1})}{p(X^j)}}{\sum\limits_{X^i\in\mathbb{X}} \frac{p(X^i|c^{k-1})p(c^{k-1})}{p(X^i)}} = \frac{\frac{p(X^j|c^{k-1})}{p(X^j)}}{\sum\limits_{X^i\in\mathbb{X}} \frac{p(X^i|c^{k-1})}{p(X^i)}} \end{aligned} \tag{5}$$

where X^i and X^l denote the elements in $\mathbb{X}$, $p(X^i)$ denotes the prior probability of X^i, and $p(X^j c^{k-1})$ denotes the probability of obtaining both X^j and c^{k-1}.

Combining (4) and (5), we have

$$\frac{f\left(z^j, c^{k-1}\right)}{\sum\limits_{z^i\in\mathbb{X}} f\left(z^i, c^{k-1}\right)} = \frac{\frac{p(X^j|c^{k-1})}{p(X^j)}}{\sum\limits_{X^i\in\mathbb{X}} \frac{p(X^i|c^{k-1})}{p(X^i)}} \tag{6}$$

Let $A(X^j) = f\left(z^j, c^{k-1}\right)$ and $B(X^j) = \frac{p(X^j|c^{k-1})}{p(X^j)}$, the Eq (6) can be written as

$$\frac{A(X^j)}{\sum\limits_{z^i\in\mathbb{Z}} A(X^i)} = \frac{B(X^j)}{\sum\limits_{X^i\in\mathbb{X}} B(X^i)} \Rightarrow \frac{A(X^j)}{B(X^j)} = \frac{\sum\limits_{z^i\in\mathbb{Z}} A(X^i)}{\sum\limits_{X^i\in\mathbb{X}} B(X^i)} \tag{7}$$

For another $X^l \in \mathbb{X}$, repeating the steps from Eq (4) to Eq (7), we have

$$\frac{A(X^l)}{B(X^l)} = \frac{\sum\limits_{z^i\in\mathbb{X}} A(X^i)}{\sum\limits_{X^i\in\mathbb{X}} B(X^i)} \tag{8}$$

Combining (7) and (8), we can get:

$$\frac{A(X^j)}{B(X^j)} = \frac{A(X^l)}{B(X^l)} = M \Rightarrow \frac{f\left(z^j, c^{k-1}\right)}{\frac{p(X^j|c^{k-1})}{p(X^j)}} = M \tag{9}$$

where M denotes a multiplication constant. Substituting X^k into Eq (9), we have that $f\left(z^k, c^{k-1}\right)$ is proportional to the $\frac{p(X^k|c^{k-1})}{p(X^k)}$. The proof completes. □

2) Training with both positive and negative monitoring data

As discussed in Section I-A, the low sampling rate is one main difficult issue faced by representation learning for NMM filling. To solve the problem, we propose a training method that not only exploits the positive monitoring data, but also exploits some negative monitoring data.

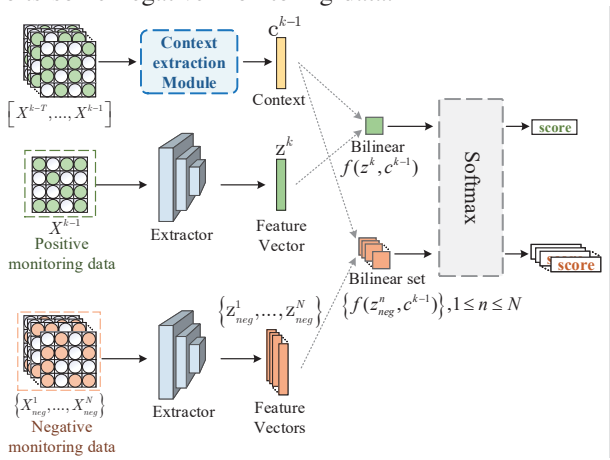

Fig. 4. PMI guided training with both positive and negative monitoring data.

Fig.4 shows our training process. To extract more accurate context c^{k-1} from the historical data, we set the current NMM X^k waiting to be filled as the positive data, the random noise or NMMs collected long time ago (so not closely related to the current moment) as the negative monitoring data, denoted by $\left\{X^1_{neg}, ..., X^N_{neg}\right\}$ in Fig.4.

Through contrasting both positive and negative monitoring data, such a training process can help the neural networks understand the difference between them.

To train c^{k-1} that its correlation with the positive monitoring data is high and with the negative monitoring data is low, two groups of log-bilinear models are adopted in our training process. The first group is $f(z^k, c^{k-1})$ and the second group is $\left\{f(z^n_{neg}, c^{k-1})\right\}, 1 \leq n \leq N$. To compare the relationship between c^{k-1} and the two groups of data z^k and $z^n_{neg}, (1 \leq n \leq N)$, a shared c^{k-1} is used in the training process.

Moreover, to compute the probability for c^{k-1} and z^k to be similar by the log-bilinear model $f(z^k, c^{k-1})$ (the same of $f(z^n_{neg}, c^{k-1})$), we design an auxiliary part softmax function (i.e., Fig.4 grey dotted box). After the final softmax operation, if the result of $f(z^k, c^{k-1})$ is close to 1 and $f(z^n_{neg}, c^{k-1})$ is close to 0, it shows that c^{k-1} is able to distinguish between positive and negative monitoring data. That is to say, c^{k-1} extracts the required contexts in the historical sequence to more accurately recover the missing entries in current data.

3) Loss function

To maximize PMI between c^{k-1} and the positive data and minimize PMI between c^{k-1} and the negative data, we design the loss function of the context extraction module as follows:

$$\mathcal{L} = -E_z \left[log \frac{f(z^k, c^{k-1})}{f(z^k, c^{k-1}) + \sum_{n=1}^{N} f(z^n_{neg}, c^{k-1})} \right] \quad (10)$$

where E_z means the expectation. z^k is the spatial information of the positive monitoring data, and $z^n_{neg}, (1 \le n \le N)$ is the spatial information of the negative monitoring data. Note that $f(z^n_{neg}, c^{k-1})$ and $f(z^k, c^{k-1})$ are both defined by Eq (3).

C. Good properties of the context extraction module

Accurate feature extraction for accurate missing data filling even with low sampling rate in current NMM. We exploit PMI theory to guide the training of the context c^{k-1} extracted from the history data that it is more similar to the positive monitoring data (current data) and less similar to the negative monitoring data. Moreover, even when the sampling rate in current monitoring data is low with random sample locations, as the negative monitoring data are infinity theoretically, our module can still approximate the target distribution of data to fill in missing data.

Low training complexity to meet the requirement in stream-processing. In some neural network models, such as ones based on encoder and decoder, context validation (e.g., decoding the features using a decoder) is needed to ensure the accuracy of context extraction. The additional network structure will add the computational complexity, while our module validates c^{k-1}'s effectiveness based on simple PMI calculation. The use of PMI validation in each training step to ensure an effective c^{k-1} to be extracted can significantly reduce the complexity of the whole training process. This further helps our context extraction module to easily capture the dynamic information.

V. Generation module

A. Design requirements in the generation module

The generation module will take the context information c^{k-1} learned from historical data and the spatial information z^k extracted from the current X^k to output the filled $\hat{X}^k$. Our goal is to fill the missing data accurately and quickly.

To ensure the accuracy of missing data inference, it requires training the neural network to minimize the difference between the expected value (the ground truth X'^k) and $\hat{X}^k$. This is done using gradient descent (back propagation) through the iteration process. Denoting $L(w)$ as the loss function between X'^k and $\hat{X}^k$ where w contains the parameters in the neural network, in each iteration, the neural network parameters are updated in the opposite direction of the gradient of the loss function $L(w)$. This iteration is repeated until reaching the minimum of the loss function. The learning process can be described by

$$w_{t+1} = w_t - \varepsilon_t \nabla_w L(w) \quad (11)$$

where w_{t+1} denotes parameter at the iteration $t+1$, w_t denotes the parameter at the iteration t, ε_t denotes the learning rate, and $\nabla_w L(w)$ denotes the gradient of loss $L(w)$.

Obviously, the loss function will impact the iteration process thus the performance of the generation module. Before we introduce our loss function, lets first see how the gradient descent of a loss function impacts the training process.

As illustrated in Fig 5, the gradient is the slope of the tangent of the loss function, and can either be positive or negative. If $\nabla_w L(w) > 0$, it means that the current weight is on the right of the optimal $w*$, and the update should be negative to get close to $w*$. On the other hand, if $\nabla_w L(w) < 0$, the update will be positive to increase the current w in order to converge to the optimal value $w*$.

The gradient together with the learning rate ε_t determines the length of the update step in each iteration. Given a learning rate ε_t, we find that:

1) If the function value is far from the minimum (i.e., the difference between the filled and ground-truth is large), a large gradient would speed up the training process to reduce the computational cost.

2) If the function value is close to the optimal value with a small loss (i.e., the difference between the filled and ground-truth is small), a large gradient may fail to drive the weight to converge and overshoot the minimum loss, and thus a small gradient is preferred.

Therefore, for our generation module to have an efficient training, the gradient of the loss function should satisfies above two properties.

B. Analysis of MSE and Relative Error

To design a proper loss function, we first examine two typical loss functions, Mean Square Error (MSE) and Relative Error.

As a widely used loss function, MSE measures the absolute distance between the ground-truth X'^k and the filled $\hat{X}^k$, and is expressed as follows:

$$MSE = \mathop{\mathrm{E}}_{i,j,k \in \Omega} \left(\hat{X}^k_{ij} - X'^k_{ij} \right)^2 \quad (12)$$

where E is the expectation, $\hat{X}^k_{ij}$ is the filled value, and X'^k_{ij} is the ground-truth at observed position (i, j) in time k. The Ω is the set of observed positions.

Relative error measures the relative distance between the ground-truth X'^k and the filled $\hat{X}^k$, and can be expressed as

$$Relative\ error = \mathop{\mathrm{E}}_{i,j,k \in \Omega} \frac{\left| \hat{X}^k_{ij} - X'^k_{ij} \right|}{\left| X'^k_{ij} \right|} \quad (13)$$

Fig 6 shows the filling errors resulted from different loss functions. We observe that

1) MSE helps the training process to quickly reduce the filling error in the early training stage, starts to rises slowly beyond a certain transition point, and even fluctuates with the number of iterations increases. Even after 50 iterations, we can hardly see it will converge. Obviously, MSE has a poor convergence trajectory.

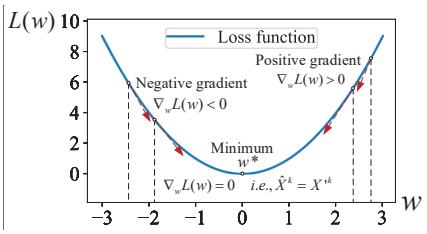

Fig. 5. The gradient descent process of the loss function.

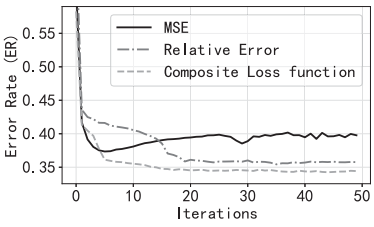

Fig. 6. The error decline curve in 50 iterations under different loss functions using network data set Abilene [32].

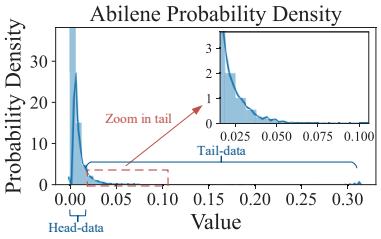

Fig. 7. Data distribution curve of the monitoring data. We also zoom the tail in the small figure.

2) For relative error, although its converge speed is slow in the early training stage, it can converge with higher accuracy compared with MSE after 20 iterations.

According to the MSE Eq in (12), the gradient at an observed position (i, j) can be calculated as $|2\left(\hat{X}_{ij}^{k} - X_{ij}^{'k}\right)|$, which is proportional to the difference between the filled item and the ground truth. In the early training process, the difference between $\hat{X}^k$ and X'^k is generally large, so the gradient under MSE is large, which explains the fast convergence speed observed.

However, because monitoring data follow a heavy-tailed distribution, MSE faces the fluctuation problem. As shown in Fig 7, most data (called the head-data) are observed to have relatively small values, while only a small amount of data (called the tail-data) have relatively large values. The small portion of large tail-data can cause large absolute error $|\left(\hat{X}_{ij}^{k} - X_{ij}^{'k}\right)|$, thus the large gradient $|2\left(\hat{X}_{ij}^{k} - X_{ij}^{'k}\right)|$. On the other hand, absolute errors of head-data are small, thus gradient values are also small. As a result, the training with MSE will pay more attention to the tail-data and ignore the head-data, which affects the improvement of accuracy and the convergence behavior. Therefore, MSE is not a good option for being the loss function.

For the relative error, from (13), the gradient of the relative error at the position (i, j) is $\frac{1}{X_{ij}^{'k}}$ if $\hat{X}_{ij}^{k} > X_{ij}^{'k}$, and is $-\frac{1}{X_{ij}^{'k}}$ if $\hat{X}_{ij}^{k} < X_{ij}^{'k}$. Even though the difference between the ground truth and a filled data item is large in the early training stage, under Relative Error, the magnitude of gradient is fixed to $|\frac{1}{X_{ij}^{'k}}|$. As the gradient of the tail-data is very small, the convergence speed is also low. As the gradients of head-data are often large, they may fail to converge and overshoot the optimal solution, which further compromises the accuracy.

C. Composite loss function

To take advantage of fast convergence speed of MSE in the early stage and higher accuracy in relative error while avoiding their drawbacks, we propose a new composite loss function in (14), where α represents the dynamic weight parameter that changes with the training iterations.

The composite loss function has two parts, MSE and our newly designed robust relative error (RRE). The range recommended for b is from 1 to 2, we set $b = 1.2$ in this paper. To calculate RRE, we add μ in the denominator to prevent small head-data from suffering the big gradient problem.

$$\mathcal{L}_{Generator} = \alpha * \mathop{\mathrm{E}}_{i,j,k\in\Omega}\left(\hat{X}_{ij}^{k} - X_{ij}^{'k}\right)^2 + (1-\alpha) * \mathop{\mathrm{E}}_{i,j,k\in\Omega}\left(\frac{\left|\hat{X}_{ij}^{k} - X_{ij}^{'k}\right|}{\left|\hat{X}_{ij}^{k} + X_{ij}^{'k} + \mu\right|}\right)^{b} \quad (14)$$
$$, 1 \le b \le 2, 0 < \mu < 1$$

As MSE has good and fast converge speed in the early stage, we set α large in the early training stage. Particularly, α is set following

$$\alpha = \left(-\frac{I_{current}}{I_{\max}} + 1\right)^2 \quad (15)$$

where $I_{current}$ denotes the number ID of the current iteration, and $I_{\max}$ denotes the total number of iterations. When $I_{current}$ is small, α will be large, and the training relies more on MSE. With the increase of $I_{current}$, α will become smaller, and the training more relies on the robust relative error. In this way, we can speed up the training while achieving high filling accuracy. For comparison, we also draw the curve under our composite loss function in Fig.6. Obviously, the curve converges quickly and can achieve much more accurate filling performance compared with MSE and relative errors.

VI. Complete Solution

The training process and online filling process of NMMF-Stream can be described as follows.

In the training process, we use the historical NMM data to train the NMMF-Stream model until it converges. Because the good properties owned by our context extraction module and generation module, our training process executes very quickly. After the training, the model parameters will be used for online filling. As another result of training, the missing values in historical data can be filled. In our evaluations in Section VII, we will show that our NMMF-Stream can accurately fill the historical missing data with much faster speed compared to the state of the art algorithms.

In the online filling process, when a new NMM arrives, if the filling loss value is large, we will update the model. With our simple structure in context extraction model and the new composite loss function adopted in generation model, the parameter update can be made very fast to capture the new feature in the newly arrived NMM. Experiments show that an update with only two iterations can make NMMF-Stream effectively capture the changed correlation. Our online execution performance can be found in Section VII-A2.

VII. Experiment

We evaluate the performance of our proposed algorithm using two public traffic data sets Abilene [32], GÉANT [33] and one network latency data set Seattle [34]. Abilene consists of 12 nodes collecting monitoring data every 5 minutes for six months from 2004-03-01. GÉANT consists of 23 nodes collecting monitoring data every 15 minutes for four months from 2005-01-01. The Seattle records the round trip times between 99 nodes in the Seattle network over 688 time slices. The NMM size of Abilene is 12×12, GÉANT is 23×23, and Seattle is 99×99. The metrics we consider include: Error Rate (ER), NMAE, Filling Computation Time, and Training Time.

The filling accuracy is evaluated by $ErrorRate(ER) = \frac{\sqrt{\sum_{i,j,k\in\bar{\Omega}} \left(X'^k_{ij} - \hat{X}^k_{ij}\right)^2}}{\sqrt{\sum_{i,j,k\in\bar{\Omega}} \left(X'^k_{ij}\right)^2}}$ and $NMAE = \frac{\sum_{i,j,k\in\bar{\Omega}} \left|X'^k_{ij} - \hat{X}^k_{ij}\right|}{\sum_{i,j,k\in\bar{\Omega}} \left|X'^k_{ij}\right|}$ where $\bar{\Omega}$ is the set of filling positions, $\hat{X}^k_{ij}$ is the filled value, and X'^k_{ij} is the ground-truth at the position (i, j) in the time slot k. To facilitate training, we use the global normalization for the data: $x^k_{i,j} = \frac{X^k_{i,j} - \min(\bar{\mathbb{X}})}{\max(\bar{\mathbb{X}})}$, where the $x^k_{i,j}$ is the normalized value of $X^k_{i,j}$, the $\min(\bar{\mathbb{X}})$ and $\max(\bar{\mathbb{X}})$ are the maximum and minimum values of the monitoring data respectively.

Definition 1. *(Filling Computation Time): a metric measuring the average time taken to fill the newly arrived NMM.*

Definition 2. *(Training Time): a metric measuring the average time taken to train the NMMF-Stream model using the historical NMM data until the model converges.*

All evaluations are run on a laptop equipped with Intel(R) Core(TM) i7-10750H CPU (2.60GHz) and 16GB RAM. To measure the time, we insert a timer into all methods.

As shown in Section VI, our NMMF-Stream should first execute a training process to obtain the basic parameters for online filling. In our experiment, we use the first 3000 NMMs of Abilene and GÉANT and the first 400 NMMs of Seattle for the training process. As discussed in Section I, in practical systems, we only have sparse monitoring data. In the training process, we use sparse monitoring data to train our model to satisfy the practical requirement. Even though we use sparse monitoring data in the training process, after training, our NMMF-Stream can fill the missing entries. After the training process, we use the following 1000, 500 and 250 NMMs of Abilene, GÉANT and Seattle to evaluate the performance of online filling.

A. Performance comparison

1) Performance under training process

Besides our NMMF-Stream, we implement other three data filling algorithms for performance comparison, including BTMF based on matrix factorization [12], DLMC exploiting neural network [26], and CoSTCo which integrates the use of tensor and neural network [23]. BTMF fills missing entries by integrating low-rank tensor factorization and vector autoregressive (VAR) process into a single probabilistic graphical model. DLMC utilizes the partially observed data to learn and construct a nonlinear latent variable model in the form of AutoEncoder. CoSTCo is a tensor completion method based on the convolutional neural network (CNN) that leverages the expressive power of CNN to model the complex interactions inside tensors.

TABLE I
Training Time in different sampling rates (seconds)

		0.03	0.06	0.1	0.2	0.3
Abilene Total 3000 NMMs in training	**BTMF**	108.6	112.6	122.3	162.8	181.2
	CoSTCo	110.1	166.2	233.3	297.2	419.7
	DLMC	108.5	115.2	115.6	116.3	126.9
	NMMF-Stream	**105.9**	**107.9**	**113.2**	**110.7**	**125.1**
GÉANT Total 3000 NMMs in training	**BTMF**	108.4	142.4	155.9	174.8	201.7
	CoSTCo	133.6	175.8	222.7	397.0	699.9
	DLMC	119.2	136.6	148.9	**168.8**	**172.3**
	NMMF-Stream	**106.2**	**126.1**	**146.4**	172.8	177.1
Seattle Total 400 NMMs in training	**BTMF**	183.3	251.3	283.8	330.7	409.6
	CoSTCo	81.8	116.7	159.7	251.6	396.8
	DLMC	86.4	105.1	114.2	125.4	137.7
	NMMF-Stream	**81.4**	**86.1**	**97.5**	**102.2**	**107.1**

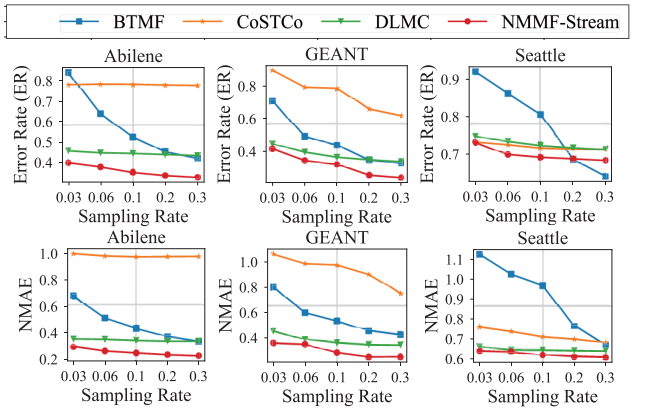

Fig. 8. ER and NMAE curves of different methods under different sampling rates of data sets.

To ensure a fair comparison, we use the same two stop conditions for all methods implemented: 1). The ER difference of 5 consecutive iterations is no more than $1e-3$; 2). The maximum number of iterations reaches, where we set the maximum number to 50.

We draw ER and NMAE under different methods of different sampling rates in Fig 8. Table I lists the training time. In Table I, the minimum training time in each sampling rate has been marked in bold. Fig.8 shows that NMMF-Stream is much more accurate than other methods, and has a stable recovery accuracy over all data sets with different data distributions. Table I shows that NMMF-Stream can complete the training process with the fastest speed in almost all scenarios with different data sets under different sampling rates. Combining Fig 8 and Table I, we conclude that our NMMF-Stream can achieve fast training speed with high filling accuracy. These benefits may mainly come from our simple but effective PMI-based context training and the novel composite loss function.

2) Performance under online filling process

As CoSTCo and BTMF use the whole tensor data as input to fill the tensor together, they can not run online to fill each incoming NMM.

TABLE II
FILLING COMPUTATION TIME(SECONDS)

	Abilene	GEANT	Seattle
BTMF	×	×	×
CoSTCo	×	×	×
DLMC	0.00306	0.00492	0.02028
Our method	0.00499	0.00685	0.02179

Original DLMC does not include a parameter update procedure for online execution, which may result in a large filling error in executing the online NMM filling task upon each NMM coming. For fair comparison, we adopt the same parameter update mechanism to capture the new features when a new NMM arrives: the model executes two iteration steps' parameter updating.

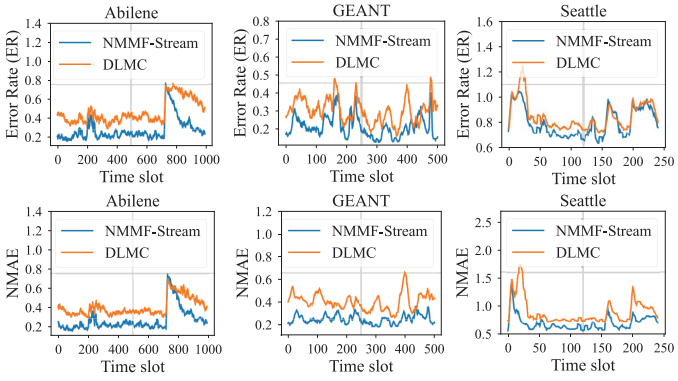

Fig. 9. Stream processing with NMMF-Stream and DLMC.

Fig 9 shows the filling performance of each time slot after the training data when sampling rate is 0.1. The average NMAE of DLMC in Abilene, GÉANT, Seattle are 0.41, 0.38, 0.89, respectively, while the average NMAE values under our NMMF-Stream are much lower, at 0.27, 0.25, 0.70, respectively. For Abilene, NMAE of NMMF-Stream is always below 0.3, while NMAE of DLMC fluctuates between 0.3 and 0.5. In addition, there is a sudden increase of NMAE near the time slot 200 and 700, which indicates that NMM correlations at that time have changed a lot. Compared with DLMC, ER of our NMMF-Stream decreases rapidly after that time, which proves that NMMF-Stream has a strong ability to capture dynamic correlations to achieve accurate filling performance.

Table II records average filling time to fill one NMM. NMMF-Stream can fill a newly arrived NMM from 5 ms to 21 ms. Although the time is slightly larger than that under DLMC, the accuracy under NMMF-Stream is much better, as shown in Fig. 9. This demonstrates that our techniques proposed in NMMF-Stream can run online with the accuracy largely raised while not introducing much computation to the model.

B. Impact of PMI

To evaluate how PMI impacts the training process, we implement two versions of our NMMF-Stream. UsePMI follows exactly our NMMF-Stream model which trains the context extraction module with PMI, while NoPMI follows all our mechanism but without this PMI-based training.

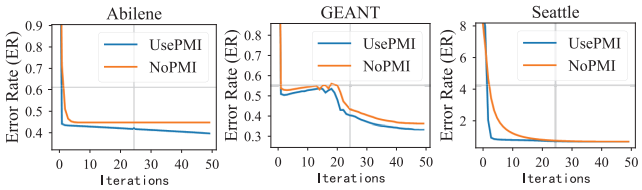

Fig. 10. Impact of PMI when other conditions are same.

In Fig 10, we compare ER of UsePMI and NoPMI under data sets Abilene, GÉANT and Seattle at a low sampling rate 0.1. As expected, UsePMI has a lower ER compared with NoPMI. Taking Seattle as an example, with the help of negative monitoring data, ER of the context extraction module trained with PMI converges rapidly after 4 iterations. Without PMI, ER of the model decreases slowly and converges until it reaches 20 iterations.

In conclusion, using PMI can significantly improve the feature extraction accuracy and convergence speed, even with a low sampling rate.

VIII. CONCLUSION

We propose NMMF-Stream, a learning-based stream-processing scheme for fast and accurate recovery of network monitoring data from sparsely observed samples. It consists of a context extraction module and a generation module. To enable accurate data filling even in the presence of low sampling rate, our context extraction module applies a simple neural network structure to extract rich and dynamic spatial-temporal correlations. It employs a few new designs to ensure the accurate and fast feature extractions, including the training based on both positive and negative monitoring data, simple validation of the quality of context extraction using the metric Pointwise Mutual Information, and exploiting GRU to simply track historical temporal features. Our generation module achieves both fast convergence speed and high filling accuracy with the guidance of a new composite loss function. Our extensive experiments using three real data sets demonstrate that even with the sampling rate as low as 0.1, our scheme can fill a newly arrived NMM less than 20ms by achieving the average NMAE 0.27 (Abilene), 0.25 (GÉANT), 0.70 (Seattle), while the NMAE under the best peer algorithm is 0.41 (Abilene), 0.38 (GÉANT), 0.89 (Seattle), respectively.

IX. ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grants 62025201, 61972144, and 61976087; in part by the NSF ECCS under Grants 1731238 and 2030063 and NSF CIF under Grant 2007313.

REFERENCES

[1] T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic sketch: Adaptive and fast network-wide measurements," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 561–575.

[2] Y. Zhou, T. Yang, J. Jiang, B. Cui, M. Yu, X. Li, and S. Uhlig, "Cold filter: A meta-framework for faster and more accurate stream processing," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 741–756.

[3] K. Xie, L. Wang, X. Wang, G. Xie, G. Zhang, D. Xie, and J. Wen, "Sequential and adaptive sampling for matrix completion in network monitoring systems," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 2443–2451.

[4] J. Fan, Y. Zhang, and M. Udell, "Polynomial matrix completion for missing data imputation and transductive learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, 2020, pp. 3842–3849.

[5] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and internet traffic matrices (extended version)," *IEEE ACM Transactions on Networking*, vol. 20, no. 3, pp. 662–676, 2012.

[6] M. Mardani and G. B. Giannakis, "Robust network traffic estimation via sparsity and low rank," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 4529–4533.

[7] R. Du, C. Chen, B. Yang, and X. Guan, "Vanet based traffic estimation: A matrix completion approach," in *2013 IEEE Global Communications Conference (GLOBECOM)*, 2013, pp. 30–35.

[8] G. Grsun and M. Crovella, "On traffic matrix completion in the internet," in *Proceedings of the 2012 Internet Measurement Conference on*, 2012, pp. 399–412.

[9] Y.-C. Chen, L. Qiu, Y. Zhang, G. Xue, and Z. Hu, "Robust network compressive sensing," in *Proceedings of the 20th annual international conference on Mobile computing and networking*, 2014, pp. 545–556.

[10] K. Xie, C. Peng, X. Wang, G. Xie, J. Wen, J. Cao, D. Zhang, and Z. Qin, "Accurate recovery of internet traffic data under variable rate measurements," *IEEE ACM Transactions on Networking*, vol. 26, no. 3, pp. 1137–1150, 2018.

[11] K. Xie, X. Wang, X. Wang, Y. Chen, G. Xie, Y. Ouyang, J. Wen, J. Cao, and D. Zhang, "Accurate recovery of missing network measurement data with localized tensor completion," *IEEE ACM Transactions on Networking*, vol. 27, no. 6, pp. 2222–2235, 2019.

[12] X. Chen and L. Sun, "Bayesian temporal factorization for multidimensional time series prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[13] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mrup, "Scalable tensor factorizations for incomplete data," *Chemometrics and Intelligent Laboratory Systems*, vol. 106, no. 1, pp. 41–56, 2011.

[14] E. Acar, D. M. Dunlavy, and T. G. Kolda, "A scalable optimization approach for fitting canonical tensor decompositions," *Journal of Chemometrics*, vol. 25, no. 2, pp. 67–86, 2011.

[15] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of 'eckart-young' decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[16] "Foundations of the parafac procedure: Models and conditions for an."

[17] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[18] T. Kolda and B. Bader, "The tophits model for higher-order web link analysis," 2006.

[19] A. Cichocki, D. Mandic, L. D. Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.

[20] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *AAAI'10 Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010, pp. 1306–1313.

[21] H. Tan, G. Feng, J. Feng, W. Wang, Y.-J. Zhang, and F. Li, "A tensor based method for missing traffic data completion," *Transportation Research Part C-emerging Technologies*, vol. 28, pp. 15–27, 2013.

[22] H. Tan, J. Feng, G. Feng, W. Wang, and Y.-J. Zhang, "Traffic volume data outlier recovery via tensor model," *Mathematical Problems in Engineering*, vol. 2013, no. 2013, pp. 1–8, 2013.

[23] H. Liu, Y. Li, M. Tsang, and Y. Liu, "Costco: A neural tensor completion model for sparse tensors," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining*, 2019, pp. 324–334.

[24] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua, "Outer product-based neural collaborative filtering," in *IJCAI'18 Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 2227–2233.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of The ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[26] J. Fan and T. W. S. Chow, "Deep learning based matrix completion," *Neurocomputing*, vol. 266, pp. 540–549, 2017.

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

[28] J. Guo and Y. Liu, "Image completion using structure and texture gan network," *Neurocomputing*, vol. 360, pp. 75–84, 2019.

[29] Q. Xing, C. Chen, and Z. Li, "Accelerated path tracing with gan and matrix completion," *IEEE Access*, vol. 9, pp. 39 055–39 066, 2021.

[30] L. Shen, W. Zhu, X. Wang, L. Xing, J. M. Pauly, B. Turkbey, S. A. Harmon, T. H. Sanford, S. Mehralivand, P. L. Choyke, B. J. Wood, and D. Xu, "Multi-domain image completion for random missing input data," *IEEE Transactions on Medical Imaging*, vol. 40, no. 4, pp. 1113–1122, 2021.

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of The ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[32] "The abilene observatory data collections. http://abilene. internet2.edu /observatory/data-collections.html."

[33] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *acm special interest group on data communication*, vol. 36, no. 1, pp. 83–86, 2006.

[34] R. Zhu, B. Liu, D. Niu, Z. Li, and H. V. Zhao, "Network latency estimation for personal devices: A matrix completion approach," *IEEE ACM Transactions on Networking*, vol. 25, no. 2, pp. 724–737, 2017.