A Light-Weight and Robust Tensor Convolutional Autoencoder for Anomaly Detection

Xiaocan Li, Kun Xie, Xin Wang, *Senior Member, IEEE*, Gaogang Xie, *Member, IEEE*, Kenli Li, Jiannong Cao, *Fellow, IEEE*, Dafang Zhang, and Jigang Wen

***Abstract*—Robust PCA is a popular anomaly detection technique and has been widely used in many applications. Although Robust PCA is promising, it is usually designed in a two-order matrix form, which is inferior to the tensor that can capture multilinearity features of data. Moreover, the detection accuracy under Robust PCA further suffers due to its sensitivity to the rank parameter which is hard to set in practice and the limitation of PCA method in capturing the non-linear feature in the data. To address the issues, we propose a Robust Tensor Convolutional Autoencoder (RTCAE) where the autoencoder instead of SVD is exploited to recover the normal data from the corrupted measurement tensor data. However, directly exploiting deep autoencoder may suffer from the problem of high memory consumption and computation overhead due to the large number of parameters used in autoencoder. To make our anomaly detection lightweight, we further design a Light Convolutional Autoencoder (LightCAE) which contains a compressed autoencoder by exploiting tensor factorization to largely compress the parameters while significantly reducing the computation complexity. We conduct extensive experiments on three real data traces to compare the performance of our proposed schemes (RTCAE and lightCAE) with that of seven baseline algorithms. The experiment results demonstrate that our proposed RTCAE achieves the highest anomaly detection accuracy. Moreover, our LightCAE requires over 60 times smaller memory storage than that required in RTCAE while achieving the similar anomaly detection accuracy.**

***Index Terms*—Anomaly detection, compressed autoencoder, robust PCA.**

Manuscript received 13 December 2021; revised 19 November 2022; accepted 10 November 2023. Date of publication 15 November 2023; date of current version 7 August 2024. This work was supported in part by the National Science Foundation for Distinguished Young Scholars under Grant 62025201, in part by the National Natural Science Foundation of China under Grants 62102138, 61972144, and 61976087, in part by China National Postdoctoral Program for Innovative Talents under Grant BX20200120, in part by Hunan Provincial Natural Science Foundation of China under Grant 2021JJ40115, in part by the Funds for International Cooperation and Exchange of the National Natural Science Foundation of China under Grant 6231101260 and in part by the Key Research and Development Program of Hunan Province under Grant 2023GK2001. Recommended for acceptance by Petko Bogdanov. *(Corresponding author: Kun Xie.)*

Xiaocan Li is with the College of Computer Science and Electronics Engineering, Hunan University, Changsha 410012, China, and also with the State University of New York at Stony Brook, Stony Brook, NY 11794 USA (e-mail: hnulxc@hnu.edu.cn).

Kun Xie, Kenli Li, and Dafang Zhang are with the College of Computer Science and Electronics Engineering, Hunan University, Changsha 410012, China (e-mail: xiekun@hnu.edu.cn; lkl@hnu.edu.cn; dfzhang@hnu.edu.cn).

Xin Wang is with the Department of Electrical and Computer Engineering, State University of New York at Stony Brook, Stony Brook, NY 11794 USA (e-mail: x.wang@stonybrook.edu).

Gaogang Xie is with the Computer Network Information Center, Chinese Academy of Sciences, Beijing 100045, China (e-mail: xie@ict.ac.cn).

Jiannong Cao is with the Department of computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong (e-mail: csjcao@comp.polyu.edu.hk).

Jigang Wen is with the School of Computer Science, Engineering, Hunan University of Science, Technology, Hunan 411199, China (e-mail: wenjigang@hnust.edu.cn).

Digital Object Identifier 10.1109/TKDE.2023.3332784

I. Introduction

A. Background and Motivation

An ANOMALY in a data set is defined by Barnett and Lewis as "an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data" [1]. Anomaly detection aims to identify data that do not conform to the patterns exhibited by the data set.

In practice, high-dimensional measurement data are typically sampled from low-dimensional subspaces, but with intrusion of outliers and/or noises. As a result, given measurement data, the normal data will usually reside in a low-dimensional subspace and form a low-rank component, while the anomalies (outliers) will stay outside this subspace. This inspires the design of an unsupervised scheme to perform anomaly detection with the low-rank approximation.

PCA is the best known scheme that exploits the low-rank approximation to make the anomaly detection based on the above observations. As a dimensionality-reduction technique, PCA seeks to find the best (in the least square error sense) low-dimensional subspace approximation to high dimensional points. Some recent papers that apply PCA to the network traffic anomaly detection have shown some promising initial results [2], [3], [4]. However, it is well known (e.g., [5]) that a standard PCA is extremely fragile to the presence of outliers: even a single corrupted point can arbitrarily alter the quality of the approximation.

To make PCA more robust, Candes et.al. propose the Robust PCA [5] to decompose a corrupted observation (noisy) matrix $\mathbf{X}$ into a low-rank component $\mathbf{C}$ and a sparse outlier component $\mathbf{S}$. As Robust PCA can recover the normal data from the corrupted observation data even though the outlier values are large, it becomes the most famous low-rank approximation-based anomaly detection algorithm and has attracted lots of applications [6], [7].

Although Robust PCA is promising technique, it usually models the data in a matrix form. A two-order matrix cannot fully capture multilinearity features [8] in the data, and may impact the anomaly detection accuracy. With the rapid development of modern computing technology in recent years, tensor data such

as multichannel images, videos, and network traffic volume, are becoming increasingly popular [9], [10], [11].

Moreover, the normal data recovery directly impacts the outlier data separation thus the anomaly detection performance. Robust PCA usually applies truncated Singular Value Decomposition (SVD) to obtain the low rank normal data, which requires the rank parameter to truncate the data while such a rank parameter is difficult to set. Moreover, assuming the normal data are low rank thus linear while ignoring their possible non-linear features, the normal data extracted in Robust PCA are not accurate.

B. Contribution

As a higher-order generalization of vector and matrix, the tensor model can better exploit the inherent relationship among data, and take full advantage of the multilinear structures of data for better data understanding and information extraction [9], [10], [11], [12], [13]. Compared to matrix-based methods [14], tensor-based anomaly detection methods can also tolerate a higher level of data corruption. Moreover, as an unsupervised model in deep neural networks, the autoencoder can extract both linear and non-linear feature to transform the input data to a lower dimension and then reconstruct the input.

Enlightened by the tensor model's strong capability of information representation and the autoencoder's strong ability of data reconstruction, we propose a robust anomaly detection model based on autoencoder. It extends the traditional Robust PCA from a matrix form to a tensor form. Moreover, the autoencoder instead of SVD is exploited to recover the normal data from the corrupted measurement data. As a result, *our model is no longer sensitive to the rank parameter, and can well capture non-linearity feature of data for more accurate normal data recovery and thus more accurate anomaly detection.*

Despite the strong ability of data reconstruction, when the dimension of the input tensor is large to hold a big amount of data and the neural network is deep, the number of parameters of the autoencoder and the associated calculation will explode and result in high computation overhead. This makes the anomaly detection algorithm suffer from the scalability, adaptability, and speed problems, which further prevents the anomaly detection algorithm from executing in a normal server device without huge memory and processing power. To address the issue, we further propose a *compressed autoencoder* to solve the problems. In the compressed autoencoder, *we exploit tensor factorization to largely compress the parameters involved in the autoencoder and significantly reduce the computation complexity*. We also calculate the compression ratio and speed up ratio with the use of the compressed autoencoder.

We have done four groups of experiments to evaluate anomaly detection accuracy, normal data reconstruction, memory usage, and computation speed of the proposed algorithms. The experiment results demonstrate that our model can achieve very high anomaly detection accuracy with fast speed by consuming very small memory.

The rest of the paper is organized as follows. We introduce the related work in Section II, and give the problem and solution overview in Section III. We describe our compressed autoencoder model in Section IV, and anomaly data estimation method in Section V. We present the convergence analysis of our algorithm in Section VI. Finally, we evaluate our algorithm performance through extensive experiments in Section VII and conclude the work in Section VIII.

II. Related Work

Recently, the low-rank approximation-based anomaly detection algorithms have been proven as an efficient unsupervised scheme for anomaly deteciton. Among various low-rank approximation implementations, Robust PCA [5] and its variations [7], [15], [16] have become the most well utilized schemes because they can accurately separate the normal data and outliers from the corrupted observation data. These algorithms usually model the data as a matrix, but the two-order information in matrix can not fully capture the comprehensive correlations hidden in data. Although few literatures [17], [18] proposed tensor-based methods to increase the accuracy of the anomaly detection accuracy, the performance is limited due to the ignorance of non-linear features within the data.

Besides the low-rank approximation-based anomaly detection algorithms, some unsupervised neural network methods are also applied for anomaly detection. Noticing autoencoder's strong ability of reconstructing data, some literature studies [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30] propose to detect anomaly based on autoencoder, which can be mainly divided into two types.

The first type of algorithm determines whether a data item is anomalous by comparing the distance between the output data reconstructed through the autoencoder and the input data with a pre-defined threshold. In [23], a deep autoencoding gaussian mixture model is proposed, where autoencoder and gaussian mixture model are used together. A deep structured energy-based model [24] uses the density estimation tool (EBM: energy-based model) to connect with a regularized autoencoder. Additionally, in [31], a MemStream model uses a denoising autoencoder to extract features and a memory module with a FIFO replacement policy to learn the dynamically changing trends. Similarly, the autoencoder is exploited in [32] to extract features of input data. Different from [23], [24], studies in [31], [32] determine whether the input data are anomalous or not by comparing the distances between the extracted features of input data and normal data with a pre-defined threshold. These models are more suitable to process the data in the streaming fashion [32].

Different from above techniques, to detect the anomalies, the second type of algorithms decomposes the data into two parts: normal data and outlier data, and detect the anomalies using the outlier data. This type of algorithm can not only detect whether an data item is anomalous, but also identify which part in the data item is anomalous. Therefore, these models could detect and locate the anomalies at a finer grained level. In this paper, we design our scheme based on the data separation approach. Among current studies, denoising autoencoder models and maximum correntropy autoenccoder models may be the most known in this type. Denoising autoencoder models [25], [26], [27], [28], [33], [34] require the access to a source of clean,

noise-free data for model training, but such data are not always available in practical applications. The maximum correntropy autoencoder model [29] replaces the reconstruction cost with noise-resistant criteria correntropy. They still train the hidden layer of the autoencoder on corrupted data, and the quality of features extracted by the hidden layer may still be influenced by the training data with a large fraction of corruptions. As a result, the normal data recovered are still not accurate, which impacts the performance of anomaly detection.

To overcome the problem, the work in [30] propose a robust autoencoder method which trains the autoencoder model after the isolation of anomalies by representing the anomalies through L_1 norm. However, [30] is still not a efficient and effective model because of following reasons. First, as L_1 norm is the convex relaxation of L_0 norm to represent the sparse anomaly data, which may reduce the accuracy. Second, this model is designed based on the matrix form, and can not fully capture multilinearity features in the data thus still can not achieve good anomaly detection accuracy. Finally, the neural network based methods always suffer from the problem of large number of parameters and consequently the high computation complexity which is not suitable for running in normal computing units.

To conquer the problems in existing studies, we propose a novel Robust Tensor Convolutional Autoencoder model for anomaly detection. In our tensor model, to increase the anomaly detection accuracy, we utilize L_0 norm to represent the sparse anomaly data and exploit a convolutional autoencoder network to reconstruct the normal data by taking advantage of the strong capability of spatial feature extraction in the convolutional autoencoder network. Moreover, to compress the parameters involved in the autoencoder, we propose a compressed model (lightCAE), which can significantly reduce the computation and memory complexity.

III. Problem and Solution Overview

A. Robust Tensor PCA

Given the measurement data in the tensor form $\mathbb{M}$, the Robust Tensor PCA model for anomaly detection can be expressed as

$$\begin{aligned} &\min_{\mathbb{X},\mathbb{S}} \|\mathbb{M}-\mathbb{S}-\ell(\mathbb{X})\|_F^2 \\ &\text{s.t. } \mathbb{X} = \mathbb{M}-\mathbb{S} \\ &\|\mathbb{S}\|_0 \le \varepsilon, \end{aligned} \tag{1}$$

where $\mathbb{M}$, $\mathbb{X}$, and $\mathbb{S}$ are in the tensor form with their sizes being $T \times S \times I \times J$. By decomposing the corrupted measurement data $\mathbb{M}$ into two parts, normal data $\ell(\mathbb{X})$ and anomaly data $\mathbb{S}$, the anomalies can be detected.

As discussed in introduction, normal data usually reside in a low-dimensional subspace, in (1), we use $\ell(\mathbb{X})$ to denote the normal data. Given a low rank value g (i.e., the number of principal components in PCA), it can be obtained by applying the truncated Singular Value Decomposition (SVD) on $\mathbb{X}$.

Since it is very costly for an attacker to compromise a large number of measurement entries, the anomalous data form a sparse tensor. We use $\|\mathbb{S}\|_0 \le \varepsilon$ to represent such constraint in the problem (1). The outlier number constraint ε does not need to match the actual number of outliers, but is only used to prevent that too many data items from being classified as outliers.

The quality of normal data recovery directly impacts the outlier data separation thus the anomaly detection performance. However, the normal data recovered from the measurement data using the Robust Tensor PCA model in (1) are not accurate due to the following reasons:

- The parameter g is necessary for the construction and recovery of the normal data from the measurement data, but determining the rank of a practical data tensor is difficult.
- Although the model in (1) extends the traditional Robust PCA from a matrix form to the tensor form, it only considers the low rank thus the linearity features in the measurement data while ignoring their possible non-linearty features.

B. Robust Tensor Convolutional Autoencoder

To address these issues, without the requirement of low rank parameter g, we further formulate the anomaly detection problem based on autoencoder to well utilize both linear and non-linear features to recover the normal data from the corrupted measurement data.

In this section, we first present the Robust Tensor Convolutional Autoencoder model and then give an overview of the solution. However, in the following Section IV, we will show that, despite the strong ability of data reconstruction, directly exploiting autoencoder to reconstruct the normal data will introduce high overhead in storage and computation and should be relieved for the efficient operation of the algorithm.

1) Model: Universal approximation theorems imply that neural networks can represent a wide variety of functions when given appropriate weights [35], [36]. In neural networks, the autoencoder [37], [38] is a popular unsupervised deep learning algorithm that applies back propagation and has good data reconstruction ability by setting the target values to be equal to the inputs. The autoencoder includes two phases: (1) The encoder phase: encoding the input $\mathbf{X}$ into a lower dimensional representation ($\mathbf{h} = E(\mathbf{X})$). (2) The decoder phase: reconstructing the input by decoding the output of encoder phase($\hat{\mathbf{X}} = D(\mathbf{h})$). The autoencoder's main task is to train the encoder and decoder to minimize the difference between input $\mathbf{X}$ and reconstruction result $\hat{\mathbf{X}}$. In particular, an autoencoder can be viewed as a solution to the following optimization problems:

$$\min_{D,E} \|\mathbf{X}-D(E(\mathbf{X}))\|_F^2 \tag{2}$$

where D and E denote the neural network in encoder and decoder, respectively.

We extend the Robust Tensor PCA model in (1) to a new RTCAE model based on the autoencoder, expressed as follows:

$$\begin{aligned} &\min_{D,E,\mathbb{X},\mathbb{S}} \|\mathbb{M}-\mathbb{S}-D(E(\mathbb{X}))\|_F^2 \\ &\text{s.t. } \mathbb{X} = \mathbb{M}-\mathbb{S} \\ &\|\mathbb{S}\|_0 \le \varepsilon. \end{aligned} \tag{3}$$

Compared with (1), the normal data in (3) is reconstructed using $D(E(\mathbb{X}))$ though autoencoder. The model in (3) is not

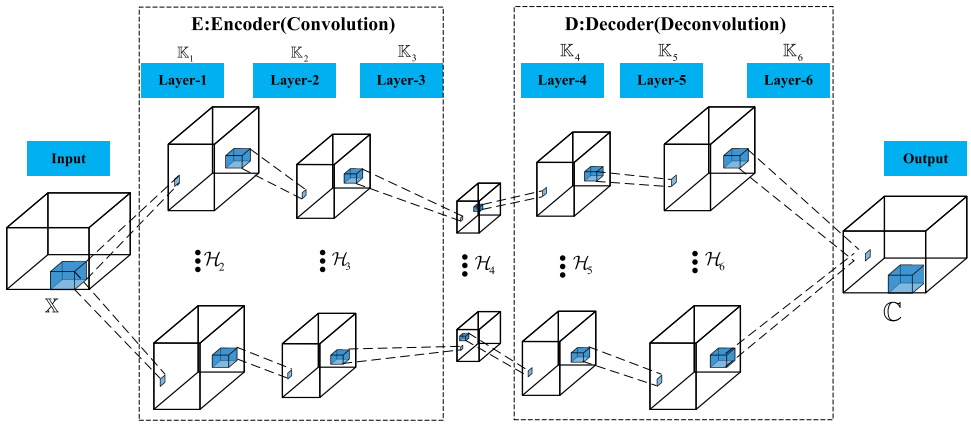


Fig. 1. Architecture of CAE.

sensitive to the rank parameter and can take both linear and non-linear features to more accurately recover the normal data.

2) Solution Overview: The problem in (3) is hard to solve because it uses the L_0-norm (i.e. set cardinality) to constrain the number of outliers. Relaxing the L_0-norm to the L_1-norm is usually adopted in Robust PCA for solving the problem. However, such a relaxation may impact the accuracy of anomaly detection. Instead, we adopt the block coordinate descent strategy to directly solve the problem (3) with L_0-norm iteratively. In each iteration step, the following two sub-problems need to be solved:

- Data reconstruction sub-problem

$$\mathbb{C} = \arg\min_{D,E,\mathbb{C}} \|\mathbb{X} - \mathbb{C}\|_F^2$$
$$\text{s.t. } \mathbb{X} = \mathbb{M} - \mathbb{S}$$
$$\mathbb{C} = D\left(E\left(\mathbb{X}\right)\right) \tag{4}$$

- Anomaly data estimation sub-problem

$$\mathbb{S} = \arg\min_{\mathbb{S}} \|\mathbb{E} - \mathbb{S}\|_F^2$$
$$\text{s.t. } \mathbb{E} = \mathbb{M} - \mathbb{C}$$
$$\|\mathbb{S}\|_0 \leq \varepsilon \tag{5}$$

In each iteration, we first fix the current estimate of anomalous data $\mathbb{S} \in R^{T\times S\times I\times J}$ and exclude them from the original data $\mathbb{M}$ to get the "clean" data $\mathbb{X}$, and then use the autoencoder to obtain the normal data $\mathbb{C}=D(E(\mathbb{X}))$. Next, we update the anomalous data $\mathbb{S}$ based on $\mathbb{E} = \mathbb{M} - \mathbb{C}$. In Sections IV and V, we will introduce our algorithms to solve above two sub-problems.

IV. Light-Weight Autoencoder to Reconstruct the Normal Data

A. Convolutional Autoencoder (CAE)

We design a convolutional autoencoder (CAE) model for the sub-problem in (4) based on convolutional neural network instead of fully connected neural network, because the convolutional neural network can better exploit the spatial-temporal correlation hidden in measurement data to enable more accurate normal recovery.

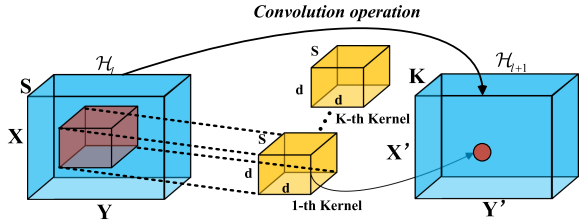


Fig. 2. Convolution Operation.

As shown in Fig. 1, the convolutional autoencoder (CAE) consists of multiple convolutional and deconvolutional layers. The convolutional layers take the role of encoder and perform the dimensionality reduction to down-sample the input tensor (i.e., to reduce the size of input), while the deconvolutional layers perform up-sample operations to reconstruct the tensor data. The number of convolutional layers/deconvolutional layers are all M. In our experiments, we set $M = 3$. The operations on the convolutional layer can be expressed as

$$\mathcal{H}_{l+1} = \sigma(\mathcal{H}_l \otimes \mathbb{K}_l + \mathcal{B}), \tag{6}$$

where $\mathcal{H}_l$ and $\mathcal{H}_{l+1}$ are the latent representation of feature map on layer l and layer $l+1$, respectively. $\mathbb{K}_l$ is the convolution kernel on layer l, σ is the non-linear activation function, $\mathcal{B}$ is the bias parameter, and $\otimes$ represents the convolution operation. When $l = 1$, $\mathcal{H}_l$ is the input tensor, that is, $\mathcal{H}_l = \mathbb{X}$, where $\mathbb{X}$ is defined in (4).

Following [39], the operation on deconvolutional layers is the transpose of convolution operation, and we omit the detailed descriptions to save the space.

We further utilize Fig. 2 to illustrate the convolution operation $\otimes$ in (6). With no zero-padding, given the input feature map $\mathcal{H}_l \in R^{S\times X\times Y}$ with convolution kernel ($\mathbb{K}_l$) of size $K \times S \times d \times d$, the output map $\mathcal{H}_{l+1} \in R^{K\times X'\times Y'}$ can be calculated as follows:

$$\mathcal{H}_{l+1}\left(k, x', y'\right) = \sum_{s=1}^{S}\sum_{i=1}^{d}\sum_{j=1}^{d} \mathcal{H}_l(s, x, y)\mathbb{K}_l(k, s, i, j)$$
$$\text{s.t. } x = (x'-1)stride + i$$

$$\text{and} \quad y = (y' - 1)stride + j \tag{7}$$

where $\mathcal{H}_{l+1}(k, x', y')$ is the (k, x', y')-th entry in the output map $\mathcal{H}_{l+1}$, $X' = (X - d + 1)$, $Y' = (Y - d + 1)$.

B. Memory and Computation Overhead of Using Autoencoder

The scalability, adaptability, and processing speed are very crucial for anomaly detection tasks. For example, in the network field, many network gateways, routers, or other small-size network devices do not have enough memory or processing power to train anomaly detection model or even execute such model trained.

Although promising, the convolution/deconvolution operation in CAE needs large amounts of memory and computational resources. The parameters of the CAE model are mainly from the convolution kernel. For layer l, the convolution kernel $\mathbb{K}_l$'s size is KSd^2, which leads to KSd^2 parameters. According to the convolution process in (7), to calculate each scalar value $\mathcal{H}_{l+1}(k, x', y')$ on the output map $\mathcal{H}_{l+1}$, it requires Sd^2 multiplication operations. Thus, obtaining the whole output map $\mathcal{H}_{l+1}$ requires $Sd^2 \times KX'Y'$ multiplication operations. This is the number of parameters and the computation complexity required on one layer. When the size of input tensor is large and CAE is deep with a large number of layers, the total number of parameters and computation complexity will be huge.

C. Compressed CAE

In this section, we propose a light-weight CAE though tensor factorization to compress the parameters and reduce the computation complexity.

As introduced in the CAE model, for the convolutional layer l, the convolution kernel ($\mathbb{K}_l$) is of size $K \times S \times d \times d$, which is actually a 4-way tensor, called the kernel tensor. In this paper, to compress our CAE model, we exploit CP-factorization to perform tensor factorization on the kernel tensor, which can expressed as follows.

$$\mathbb{K}_l(k, s, i, j) = \sum_{r=1}^{R} \mathbf{U}^{(1)}(k, r)\mathbf{U}^{(2)}(s, r)\mathbf{U}^{(3)}(i, r)\mathbf{U}^{(4)}(j, r) \tag{8}$$

where R is a hyperparameter in CP-factorization, $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, $\mathbf{U}^{(3)}$, $\mathbf{U}^{(4)}$ are four factor matrices whose sizes are $R \times K$, $R \times S$, $R \times d$, and $R \times d$, respectively.

Substituting $\mathbb{K}_l(k, s, i, j)$ in (8) into (7), (7) can be rewritten as follows.

$$\begin{aligned}\mathcal{H}_{l+1}(k, x', y') = \sum_{r=1}^{R} \mathbf{U}^{(1)}(k, r) \Bigg(\sum_{j=1}^{d} \mathbf{U}^{(4)}(j, r) \\ \times \Bigg(\sum_{i=1}^{d} \mathbf{U}^{(3)}(i, r) \Bigg(\sum_{s=1}^{S} \mathbf{U}^{(2)}(s, r)\mathcal{H}_l(s, x, y) \Bigg) \Bigg) \Bigg) \\ \text{s.t. } x' = {}^{(x-i)}/_{stride+1} \text{and} \quad y' = {}^{(y-i)}/_{stride+1}\end{aligned} \tag{9}$$

According to the (9), the above operation in (7) can be divided into four sequential sub-operations, each involving one factor matrix, expressed as follows

$$\mathcal{N}_l^1(r, x, y) = \sum_{s=1}^{S} \mathbf{U}^{(2)}(s, r)\mathcal{H}_l(s, x, y) \tag{10}$$

$$\mathcal{N}_l^2(r, x', y) = \sum_{i=1}^{d} \mathbf{U}^{(3)}(i, r)\mathcal{N}_l^1(r, x, y) \tag{11}$$

$$\mathcal{N}_l^3(r, x', y') = \sum_{j=1}^{d} \mathbf{U}^{(4)}(j, r)\mathcal{N}_l^2(r, x', y) \tag{12}$$

$$\mathcal{H}_{l+1}(k, x', y') = \sum_{r=1}^{R} \mathbf{U}^{(1)}(k, r)\mathcal{N}_l^3(r, x', y') \tag{13}$$

where $\mathcal{N}_l^1(r, x, y)$, $\mathcal{N}_l^2(r, x', y)$, and $\mathcal{N}_l^3(r, x', y')$ are intermediate maps of sizes $R \times X \times Y$, $R \times X' \times Y$, $R \times X' \times Y'$, respectively.

Obviously, the above four sequential sub-operations can be regarded as four convolution operations with the smaller kernels building from factor matrices with their sizes being $R \times S \times 1 \times 1$, $R \times 1 \times d \times 1$, $R \times 1 \times 1 \times d$, and $K \times R \times 1 \times 1$. We use Fig. 3(b) to illustrate these four convolution operations in four steps, each corresponding to one sub-operation above.

In the first step, we input a 3-way tensor $\mathcal{H}_l$, and obtain a 3-way output map $\mathcal{N}_l^1$ of size $R \times X \times Y$. Every red box of size $S \times 1 \times 1$ in $\mathcal{H}_l$ is mapped into R scalar values in $\mathcal{N}_l^1$ through a convolution kernel with the size of $S \times R \times 1 \times 1$.

In the second step, we map every red box of size $1 \times d \times 1$ into R scalar values in $\mathcal{N}_l^2$ through a convolution kernel with the size of $R \times 1 \times d \times 1$. Similarly, we map every red box of size $1 \times 1 \times d$ into R scalar values in $\mathcal{N}_l^3$ through the convolution kernel with the size of $R \times 1 \times 1 \times d$ in the third step.

In the fourth step, every red box of size $1 \times 1 \times R$ are mapped into K scalar values in $\mathcal{H}_{l+1}$ through the convolution kernel with the size of $K \times R \times 1 \times 1$.

For comparison, we also draw the convolution operation in (7) in Fig. 3(a). Instead of using the convolution operation in (7) with a large convolution kernel, after tensor factorization, four sequential convolution operations in Fig. 3(b) involve small convolution kernels and can still achieve the goal to output $\mathbb{H}_{l+1}$.

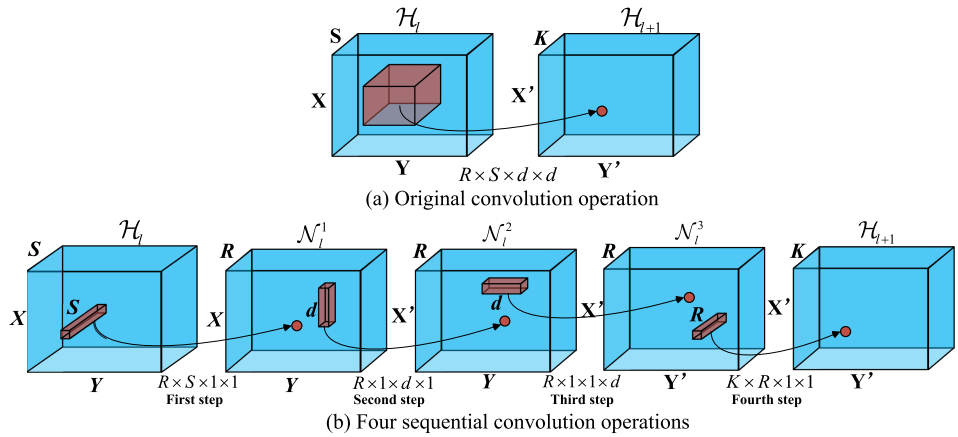
We design similar compression operations in deconvolutional layers, for the space limitation, we do not describe it.

D. Compression Ratio in Compressed CAE

After using the tensor factorization, it requires RS, Rd, Rd, and KR parameters in four sequential convolution operations, respectively. Compared with origin convolution operation in (7), the compression ratio (CR) of parameters can be expressed as follows:

$$CR = \frac{KSd^2}{RS + 2Rd + KR} \tag{14}$$

where KSd^2 is the number of parameters required on the original layer l in CAE calculated at the end of Section IV-B.



(a) Original convolution operation

(b) Four sequential convolution operations

Fig. 3. Convolution Operations After Tensor Factorization.

To perform four sequential convolution operations, it requires $S \times RXY, d \times RX'Y, d \times RX'Y'$, and $R \times KX'Y'$ multiplication operations. Thus, compared with CAE, the speed-up ratio SR under the compressed CAE can be expressed as follows:

$$SR = \frac{Sd^2 \times KX'Y'}{S \times RXY + d \times RX'Y + d \times RX'Y' + R \times KX'Y'} \tag{15}$$

where $KSd^2 \times X'Y'$ is the number of multiplication operations required on the original layer l in CAE calculated at the end of Section IV-B.

The good compression ability makes our anomaly detection model a light-weight design and can be executed even in a low cost device with limited memory and processing power. In the evaluation part, we will demonstrate that the effectiveness of large parameter compression and the speedup under our compressed CAE.

V. Anomaly Data Estimation Method

According to our solution overview in Section III-B2, after the normal data $\mathbb{C} \in R^{T\times S\times I\times J}$ is reconstructed through CAE, we need to estimate the outliers $\mathbb{S} \in R^{T\times S\times I\times J}$ by solving the sub-problem $\mathbb{S} = \arg\min_{\mathbb{S}} \|\mathbb{E} - \mathbb{S}\|_F^2$ defined in (5) where $\mathbb{E} = \mathbb{M} - \mathbb{C}$ and $\|\mathbb{S}\|_0 \leq \varepsilon$.

We have $\|\mathbb{E} - \mathbb{S}\|_F^2 = \sum_{t=1}^{T}\sum_{s=1}^{S}\sum_{i=1}^{I}\sum_{j=1}^{J}(e_{tsij} - s_{tsij})^2$, and $(e_{tsij} - s_{tsij})^2$ is positive. The original problem $\mathbb{S} = \arg\min_{\mathbb{S}} \|\mathbb{E} - \mathbb{S}\|_F^2$ is decomposable to look for each entry of $\mathbb{S}$, and we have $s^*_{tsij} = \arg\min_{s_{tsij}} (e_{tsij} - s_{tsij})^2$ to minimize $\|\mathbb{E} - \mathbb{S}\|_F^2$.

Since each entry s_{tsij} can have only two candidate values: s^*_{tsij} or 0, setting an item s^*_{tsij} to 0, the $\|\mathbb{E} - \mathbb{S}\|_F^2$ will increase, which is against the objective of minimizing $\|\mathbb{E} - \mathbb{S}\|_F^2$.

Therefore, the anomaly data estimation sub-problem described in (5) can be easily solved with the following steps: set all $s_{tsij} = 0$ and sort $(e_{tsij} - s_{tsij})^2$ in a descending order, then select the first ε entries to be e^*_{tsij} and set the remaining items to 0. That is:

$$s_{t,s,i,j} = \begin{cases} e_{t,s,i,j} & \beta_{t,s,i,j} > \beta(\varepsilon) \\ 0 & otherwise \end{cases} \tag{16}$$

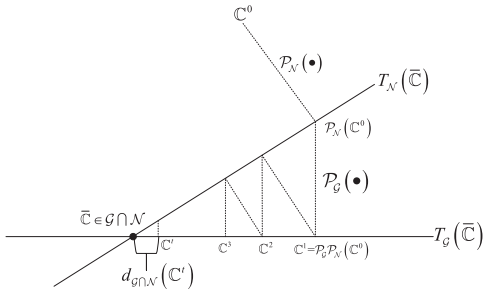


Fig. 4. Alternating projections.

where $\beta_{t,s,i,j} = (e_{t,s,i,j})^2$ and $\beta(\varepsilon)$ is the $\varepsilon - th$ largest value in $\beta_{t,s,i,j}$.

VI. Convergence Analysis

In this section, we prove the convergence of our whole anomaly detection algorithm in Section III-B2, which needs to iteratively solve two sub-problems. For these two sub-problems, only the first sub-problem in (4) requires iteration training with CAE model. As a neural network model optimized by SGD (Stochastic gradient descent), the CAE model has been proven to be converged in [40].

In the anomaly detection algorithm, we denote the normal data and outlier data obtained at each iteration step t as $\{\mathbb{C}^t, \mathbb{S}^t\}$. Next, we will demonstrate the convergence of the iterative sequence $\{\mathbb{C}^t, \mathbb{S}^t\}$.

Based on the framework proposed in [41], as shown in Fig. 4, the algorithm can be seen as a process of alternately projecting $\mathbb{C}$ onto one manifold $\mathcal{N}$ (its tangent space $T_{\mathcal{N}}(\bar{\mathbb{C}})$) and then onto another manifold $\mathcal{G}$ (its tangent space $T_{\mathcal{G}}(\bar{\mathbb{C}})$), where $\bar{\mathbb{C}} \in \mathcal{G} \cap \mathcal{N}$ is any local solution of problem (3), $\mathcal{N}$ and $\mathcal{G}$ will be defined in (17) and (18).

$$\mathcal{N} = \left\{\mathbb{M} - \mathcal{P}_{\mathcal{D}}(\mathbb{M} - \mathbb{C}) : \mathbb{C} \in R^{T\times S\times I\times J}\right\} \tag{17}$$

$$\mathcal{G} = \left\{\mathbb{C} \in R^{T\times S\times I\times J} : \mathbb{C} = \mathbf{CAE}(\mathbb{M} - \mathbb{S})\right\} \tag{18}$$

where $\mathcal{P}_{\mathcal{D}}$ is the process of solving sub-problem in (5). Both $\mathcal{N}$ and $\mathcal{G}$ are two C^k-manifolds around the point $\bar{\mathbb{C}} \in \mathcal{G} \cap \mathcal{N}$.

In the iterative step $t+1$, to obtain $\mathbb{C}^{t+1}$, we substitute $\mathbb{S}^t = \mathcal{P}_{\mathcal{D}}(\mathbb{M}-\mathbb{C}^t)$ into the problem (4), and we have

$$\mathbb{C}^{t+1} = \mathcal{P}_{\mathcal{G}}\left(\mathbb{M}-\mathcal{P}_{\mathcal{D}}\left(\mathbb{M}-\mathbb{C}^t\right)\right) = \left(\mathcal{P}_{\mathcal{G}}\mathcal{P}_{\mathcal{N}}\right)\left(\mathbb{C}^t\right) \tag{19}$$

where $\mathcal{P}_{\mathcal{G}}$ and $\mathcal{P}_{\mathcal{N}}$ are the projection process of (17) and (18).

Fig. 4 illustrates the iterative execution of our algorithm. In the first iterative step, $\mathbb{C}^0$ is projected onto the manifold $\mathcal{N}$ following the (17) with the projection result being $\mathcal{P}_{\mathcal{N}}(\mathbb{C}^0)$. Then $\mathcal{P}_{\mathcal{N}}(\mathbb{C}^0)$ is further projected onto the manifold $\mathcal{G}$ following (18), and we have $\mathbb{C}^1 = (\mathcal{P}_{\mathcal{G}}\mathcal{P}_{\mathcal{N}})(\mathbb{C}^0)$. After t iterations, we can obtain $\mathbb{C}^t$.

Any point $\mathbb{C} \in \mathcal{G}\cap\mathcal{N}$ is a local solution of problem (3). We define the angle between two manifolds $\mathcal{G}$ and $\mathcal{N}$ at point $\mathbb{C}$ as the angle between the corresponding tangent spaces $T_{\mathcal{G}}(\mathbb{C})$ and $T_{\mathcal{N}}(\mathbb{C})$. The angle is between 0 and $\frac{\pi}{2}$ with cosine:

$$\mathrm{c}\left(\mathcal{G},\mathcal{N},\mathbb{C}\right) = \mathrm{c}\left(T_{\mathcal{G}}\left(\mathbb{C}\right),T_{\mathcal{N}}\left(\mathbb{C}\right)\right) \tag{20}$$

Note that, the local solution of problem (3) is not unique, and we may have multiple points $\mathbb{C}$. Thus, we may have different $T_{\mathcal{G}}(\mathbb{C}), T_{\mathcal{N}}(\mathbb{C})$, and $\mathrm{c}(\mathcal{G},\mathcal{N},\mathbb{C})$.

We define $\mathbb{U}$ as the unit sphere in $R^{T\times S\times I\times J}$. The angle between $T_{\mathcal{G}}(\mathbb{C})$ and $T_{\mathcal{N}}(\mathbb{C})$ can also be defined as follows:

$$c\left(T_{\mathcal{G}}\left(\mathbb{C}\right),T_{\mathcal{N}}\left(\mathbb{C}\right)\right) = \max\left\{\begin{matrix}\langle x,y\rangle, x\in\mathbb{U}\cap T_{\mathcal{G}}\left(\mathbb{C}\right)\cap\left(T_{\mathcal{G}}\left(\mathbb{C}\right)\cap T_{\mathcal{N}}\left(\mathbb{C}\right)\right)^{\perp},\\ y\in\mathbb{U}\cap T_{\mathcal{N}}\left(\mathbb{C}\right)\cap\left(T_{\mathcal{G}}\left(\mathbb{C}\right)\cap T_{\mathcal{N}}\left(\mathbb{C}\right)\right)^{\perp}\end{matrix}\right\} \tag{21}$$

Moreover, according to [41], the quantity $c(T_{\mathcal{G}}(\mathbb{C}),T_{\mathcal{N}}(\mathbb{C}))$ is zero only if $T_{\mathcal{G}}(\mathbb{C})\subset T_{\mathcal{N}}(\mathbb{C})$ or $T_{\mathcal{G}}(\mathbb{C})\supset T_{\mathcal{N}}(\mathbb{C})$. And the compactness ensures that the maximum is always attained, and this easily yields $\mathrm{c}(\mathcal{G},\mathcal{N},\mathbb{C})<1$.

Let $\mathcal{G}$ and $\mathcal{N}$ be two transverse C^2-manifolds around the point $\overline{\mathbb{C}}\in\mathcal{G}\cap\mathcal{N}$, according to [41], we have:

$$\limsup_{\mathbb{C}\to\overline{\mathbb{C}},\mathbb{C}\notin\mathcal{G}\cap\mathcal{N}} \frac{\left\|\left(\mathcal{P}_{\mathcal{G}}\mathcal{P}_{\mathcal{N}}\right)^t\left(\mathbb{C}\right)-\mathcal{P}_{\mathcal{G}\cap\mathcal{N}}\left(\mathbb{C}\right)\right\|}{\left\|\mathbb{C}-\mathcal{P}_{\mathcal{G}\cap\mathcal{N}}\left(\mathbb{C}\right)\right\|} \le c^{2t-1} \tag{22}$$

where t is the iterative step and $\left(\mathcal{P}_{\mathcal{G}}\mathcal{P}_{\mathcal{N}}\right)^t(\mathbb{C})$ means the result of the t-th iteration.

According to (22), for all constants $\mathrm{c}>\mathrm{c}(\mathcal{G},\mathcal{N},\overline{\mathbb{C}})$, when $t=1$, there exists a radius $\eta>0$ such that

$$\forall\mathbb{C}\in B_\eta\left(\overline{\mathbb{C}}\right), \left\|\left(\mathcal{P}_{\mathcal{G}}\mathcal{P}_{\mathcal{N}}\right)\left(\mathbb{C}\right)-\mathcal{P}_{\mathcal{G}\cap\mathcal{N}}\left(\mathbb{C}\right)\right\| \le c\left\|\mathbb{C}-\mathcal{P}_{\mathcal{G}\cap\mathcal{N}}\left(\mathbb{C}\right)\right\| \tag{23}$$

where $B_\eta(\overline{\mathbb{C}})$ is the ball space with the radius equal to $\eta>0$, and the centre is $\overline{\mathbb{C}}$.

According to [41], the following theorem demonstrates the linear convergence of the variable $\mathbb{C}$.

Theorem 1: In $R^{T\times S\times I\times J}$, let $\mathcal{G}$ and $\mathcal{N}$ be two transverse manifolds around a point $\overline{\mathbb{C}}\in\mathcal{G}\cap\mathcal{N}$ (as shown in Fig. 4). If the initial point $\mathbb{C}^0\in R^{T\times S\times I\times J}$ is close to $\overline{\mathbb{C}}$, then the method of alternating projections

$$\mathbb{C}^{t+1} = \mathcal{P}_{\mathcal{G}}\mathcal{P}_{\mathcal{N}}\left(\mathbb{C}^t\right) \tag{24}$$

is well-defined, where t is the iterative step, and the distance $d_{\mathcal{G}\cap\mathcal{N}}(\mathbb{C}^t)$ (as shown in Fig. 4) from the iterate $\mathbb{C}^t$ to the intersection $\mathcal{G}\cap\mathcal{N}$ decreases Q-linearly to zero. More precisely, given any constant c strictly larger than the cosine of the angle of the intersection between the manifolds: $\mathrm{cosine}(\mathcal{G},\mathcal{N},\overline{\mathbb{C}})$, if $\mathbb{C}^0$ is close to $\overline{\mathbb{C}}$, then the iterations satisfy

$$d_{\mathcal{G}\cap\mathcal{N}}\left(\mathbb{C}^{t+1}\right) \le c\bullet d_{\mathcal{G}\cap\mathcal{N}}\left(\mathbb{C}^t\right) \tag{25}$$

Furthermore, $\mathbb{C}^t$ converges linearly to some point $\mathbb{C}^*\in\mathcal{G}\cap\mathcal{N}$, i.e., for some constant $\alpha>0$,

$$\left\|\mathbb{C}^t-\mathbb{C}^*\right\| \le \alpha c^t \tag{26}$$

Proof: We choose c such that $\cos(\mathcal{G},\mathcal{N},\overline{\mathbb{C}})<c<1$ and $\eta>0$, and therefore (23) is satisfied. Setting $\delta=\frac{(1-c)\eta}{4}$ and choosing any initial point $\mathbb{C}^0\in B_\delta(\overline{\mathbb{C}})$.

First, we prove by *induction* that the sequence of points $\mathbb{C}^t$ is well-defined, and that both $\mathbb{C}^t$ and its projection $\overline{\mathbb{C}^t}=\mathcal{P}_{\mathcal{G}\cap\mathcal{N}}(\mathbb{C}^t)$ belong to the neighborhood space $B_\eta(\overline{\mathbb{C}})$ and satisfy the following properties:

$$\left\|\mathbb{C}^t-\overline{\mathbb{C}^{t-1}}\right\| \le \delta c^t \tag{27}$$

$$\left\|\mathbb{C}^t-\overline{\mathbb{C}^t}\right\| \le \delta c^t \tag{28}$$

$$\left\|\overline{\mathbb{C}^t}-\overline{\mathbb{C}^{t-1}}\right\| \le 2\delta c^t \tag{29}$$

$$\left\|\overline{\mathbb{C}^t}-\overline{\mathbb{C}}\right\| \le 2\left(\sum_{\mathrm{i}=0}^{t}c^i\right)\delta \tag{30}$$

$$\left\|\mathbb{C}^t-\overline{\mathbb{C}}\right\| \le 2\left(\sum_{i=0}^{t}c^i\right)\delta \tag{31}$$

By setting $\overline{\mathbb{C}^{-1}}=\overline{\mathbb{C}^0}$, we have $\|\mathbb{C}^0-\overline{\mathbb{C}^0}\|\le\|\mathbb{C}^0-\overline{\mathbb{C}}\|\le\delta$. Therefore, it is easy to see that these inequalities (27)–(31) hold for $t=0$.

Assuming that these inequalities hold for some $t\ge0$, we need to prove that they hold with $t+1$. Note that if $\mathbb{C}^t$ belongs to $\mathcal{G}\cap\mathcal{N}$, there is no need to prove as $\mathbb{C}^t$ and $\overline{\mathbb{C}}$ are local solutions of $\mathbb{C}$. Otherwise, since $\mathbb{C}^t$ belongs to $\mathbb{B}_\eta(\overline{\mathbb{C}})$, the next iteration $\mathbb{C}^{t+1}$ is well-defined and the inequality (23) holds, so:

$$d_{\mathcal{G}\cap\mathcal{N}}\left(\mathbb{C}^{t+1}\right) \le \left\|\mathbb{C}^{t+1}-\overline{\mathbb{C}^t}\right\| \le c\left\|\mathbb{C}^t-\overline{\mathbb{C}^t}\right\| = cd_{\mathcal{G}\cap\mathcal{N}}\left(\mathbb{C}^t\right) \tag{32}$$

As the property (28) holds for t, according to (32), the inequality yields:

$$\left\|\mathbb{C}^{t+1}-\overline{\mathbb{C}^{\mathrm{t}}}\right\| \le \delta c^{t+1} \tag{33}$$

As $\|\mathbb{C}^{t+1}-\overline{\mathbb{C}^{t+1}}\|\le\|\mathbb{C}^{t+1}-\overline{\mathbb{C}^{\mathrm{t}}}\|$, with (33) , the inequality yields:

$$\left\|\mathbb{C}^{t+1}-\overline{\mathbb{C}^{\mathrm{t+1}}}\right\| \le \delta c^{t+1} \tag{34}$$

Based on (33) and (34), we have:

$$\left\|\overline{\mathbb{C}^{t+1}}-\overline{\mathbb{C}^t}\right\| \le \left\|\overline{\mathbb{C}^{t+1}}-\mathbb{C}^{\mathrm{t+1}}\right\|+\left\|\mathbb{C}^{t+1}-\overline{\mathbb{C}^t}\right\| \le 2\delta c^{t+1} \tag{35}$$

As $\|\overline{\mathbb{C}^{t+1}} - \overline{\mathbb{C}}\| \leq \|\overline{\mathbb{C}^{t+1}} - \overline{\mathbb{C}^t}\| + \|\overline{\mathbb{C}^t} - \overline{\mathbb{C}}\|$ and property (30) holds for t, with the help of (35), we have:

$$\left\|\overline{\mathbb{C}^{t+1}} - \overline{\mathbb{C}}\right\| \leq 2\delta c^{t+1} + 2\delta \sum_{i=0}^{t} c^i \leq 2\delta \sum_{i=0}^{t+1} c^i \tag{36}$$

Similarly, as $\|\mathbb{C}^{t+1} - \overline{\mathbb{C}}\| \leq \|\mathbb{C}^{t+1} - \overline{\mathbb{C}^t}\| + \|\overline{\mathbb{C}^t} - \overline{\mathbb{C}}\|$ and the property (30) holds for t, with the help of (33), we have:

$$\left\|\mathbb{C}^{t+1} - \overline{\mathbb{C}}\right\| \leq \delta c^{t+1} + 2\delta \sum_{i=0}^{t} c^i \leq 2\delta \sum_{i=0}^{t+1} c^i \tag{37}$$

As $\delta = \frac{(1-c)\eta}{4}$, we have (36) yields

$$\left\|\overline{\mathbb{C}^{t+1}} - \overline{\mathbb{C}}\right\| \leq \frac{2\delta}{(1-c)} \leq \frac{\eta}{2} \tag{38}$$

and (37) yields

$$\left\|\mathbb{C}^{t+1} - \overline{\mathbb{C}}\right\| \leq \frac{\eta}{2} \tag{39}$$

Therefore, $\overline{\mathbb{C}^{t+1}}$ and $\mathbb{C}^{t+1}$ belongs to $B_\eta(\overline{\mathbb{C}})$.

Second, we prove the convergence of the sequence of variable $\mathbb{C}$. Based on property in (29), for all indices $t, p \geq 0$ and $p \geq t$, we have:

$$\left\|\overline{\mathbb{C}^p} - \overline{\mathbb{C}^t}\right\| \leq \sum_{i=t+1}^{p} \left\|\overline{\mathbb{C}^i} - \overline{\mathbb{C}^{i-1}}\right\| \leq 2\delta \sum_{i=t+1}^{p} c^i \leq \frac{2\delta}{1-c} c^{t+1} \tag{40}$$

Therefore, the sequence of $\overline{\mathbb{C}^t}$ converges to an element $\overline{\mathbb{C}^*}$ in $B_\eta(\overline{\mathbb{C}})$. Passing to the limit in p and t in (40), we obtain:

$$\left\|\overline{\mathbb{C}^t} - \mathbb{C}^*\right\| \leq \frac{2\delta}{1-c} c^{t+1} \tag{41}$$

With the help of property (28), we have:

$$\|\mathbb{C}^t - \mathbb{C}^*\| \leq \left\|\mathbb{C}^t - \overline{\mathbb{C}^t}\right\| + \left\|\overline{\mathbb{C}^t} - \mathbb{C}^*\right\| \leq \left(1 + \frac{2c}{1-c}\right) \delta c^t \tag{42}$$

The proof completes. Then, the variable $\mathbb{C}$ in the proposed algorithm converges. Similarly, we can prove the convergence of $\mathbb{S}^t$, that is the iterative sequence $\{\mathbb{C}^t, \mathbb{S}^t\}$ is converged. Therefore, the convergence of algorithm in Section III-B2 has been proved.

VII. Performance Evaluation

A. Experiment Setting

To evaluate the performance of our proposed model, we apply our models to detect network anomalies using three network monitoring data traces:

1) The data trace (Abilene [42]) records the traffic between 12 nodes of U. S. Internet2 Network in 168 days, at an interval of 5 minutes.
2) The data trace (GÈANT [43]) records the traffic between 23 nodes of pan-European research backbone network in 112 days, at an interval of 15 minutes.
3) The data trace (Harvard [44]) records the application-level round trip delay between 226 Azureus clients in 72 hours, at an interval of 5 minutes.

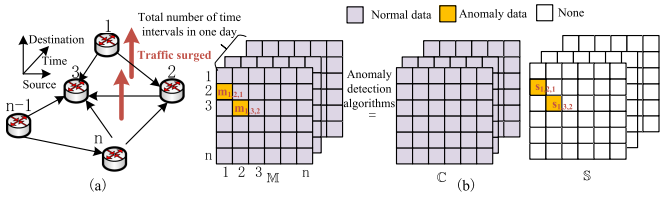


Fig. 5. Concrete example of network anomaly detection.

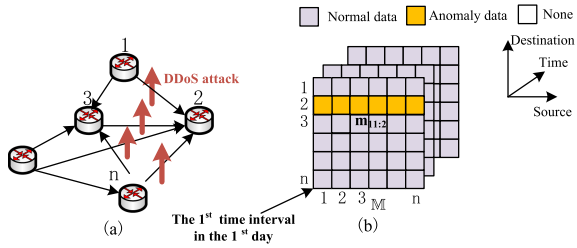


Fig. 6. Example of structured network anomalies.

As these three traces record the network monitoring data between all source and destination pairs, we model these data as a 4-way tensor $\mathbb{M} \in R^{T \times S \times I \times J}$, where I, J, T, and S correspond respectively to the number of source nodes, the number of destination nodes, total number of time intervals in one day, and the number of days that are covered by the data sets. Under the model, the network monitoring data of one day can be recorded by a 3-way tensor, as shown in Fig 5.

The network anomalies, such as flash crowds, denial-of-service attacks, and port scans, can have detrimental effects on network services. These anomalies often lead to unusual and significant changes of network traffic levels, and the changes can often span multiple links. Fig. 5 gives a concrete example to illustrate how to apply our algorithm to detect the traffic anomalies. In Fig. 5(b), we use a tensor to record the traffic volume data of the topology in Fig. 5(a). If the link between node 2 and node 3, and the link between node 1 and node 2 occur network anomalies (i.e., Traffic surged), which may make the entries $m_{1,3,2}$ and $m_{1,2,1}$ in tensor $\mathbb{M}$ changes. To detect these random element-wise anomalies, we input the corrupted monitoring tensor data $\mathbb{M}$ into anomaly detection algorithms to separate normal data $\mathbb{C}$ and outlier data $\mathbb{S}$ from $\mathbb{M}$. Using the sparse outlier data $\mathbb{S}$, we find that entries $s_{1,3,2}$ and $s_{1,2,1}$ are anomalies, thus the network anomalies happen at the two links that connect nodes 2 and 3 as well as nodes 1 and 2, respectively.

Besides the random element-wise anomalies, some structured anomalies may appear due to the DDoS attack, successive attacks, severe communication conditions, and the failure of network nodes. As shown in Fig. 6, when a DDoS attacker controls the multiple zombie hosts (i.e., source nodes 1, 2, ..., n) to inject traffic to a destination node 2, the attack introduces a structured corrupted monitoring data (i.e., $m_{1,1,:,2}$).

We apply $m_{t,s,i,j} = \frac{m_{t,s,i,j} - \min_{u,v,w,z}\{x_{u,v,w,z}\}}{\max_{u,v,w,z}\{x_{u,v,w,z}\} - \min_{u,v,w,z}\{x_{u,v,w,z}\}}$ to normalize each entries in $\mathbb{M}$ within the range [0, 1], where

TABLE I
KERNEL SIZES OF CAE MODELS

convolution	$layer-1$	$layer-2$	$layer-3$
Abilene	$Conv[512, 288, 2, 2]$	$Conv[256, 512, 2, 2]$	$Conv[128, 256, 2, 2]$
GÈANT	$Conv[512, 96, 2, 2]$	$Conv[256, 512, 2, 2]$	$Conv[128, 256, 2, 2]$
Harvard	$Conv[512, 16, 2, 2]$	$Conv[256, 512, 2, 2]$	$Conv[128, 256, 2, 2]$
deconvolution	$layer-4$	$layer-5$	$layer-6$
Abilene	$DeConv[128, 256, 3, 3]$	$DeConv[512, 256, 2, 2]$	$DeConv[288, 512, 2, 2]$
GÈANT	$DeConv[128, 256, 3, 3]$	$DeConv[512, 256, 3, 3]$	$DeConv[96, 512, 3, 3]$
Harvard	$DeConv[128, 256, 2, 2]$	$DeConv[512, 256, 3, 3]$	$DeConv[16, 512, 2, 2]$

$\max_{u,v,w,z}\{m_{u,v,w,z}\}$ and $\min_{u,v,w,z}\{m_{u,v,w,z}\}$ are the maximum value and minimum value of all the trace data, respectively.

We build our CAE model by using Pytorch 1.7.1, and run the anomaly detection model on a personal computer with Intel(R) Core(TM) i9-10850K CPU@3.60 GHz and NVIDIA GeForce RTX3090 GPU. Specially, the CAE model is built with 3 convolutional layers in the encoder part and 3 deconvolutional layers in the decoder part. The kernel size in each layer can be expressed as Table I. To train the CAE model, we view each day's traffic data $\mathcal{M} \in R^{1\times S\times I\times J}$ as an instance, and regard the number of days covered by the data set as the batch size. We compress the CAE model by CP decomposition implemented in the $tensorly$ package in python3.7. The hyper parameter R in the CP decomposition is set to be $R = 5$ as default.

To evaluate the performance of our scheme under the corruption of random element-wise anomalies, we synthetically generate anomalies by adding data outliers into the data sets following [2], [45]. 1) We randomly select $\gamma \times (T \times S \times I \times J)$ locations as the anomaly locations, where γ is the outlier ratio. 2) Following [45], we generate the values of anomaly data at these locations following the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with the mean μ and the variance σ^2. In this paper, we set μ=0, σ=1 as the default setting. Moreover, to evaluate the performance of our schemes under the structured anomaly detection, we randomly select a destination node (i.e., $m_{t,s,:,j}$) to be attacked in each time interval (i.e., $m_{t,s,:,:}$).

We implement eight methods for performance comparison, including four matrix-based anomaly detection methods and four tensor-based anomaly detection methods. RPCA [5], RobustAE [30], DRMF [15], [46], and AESc [28] are four matrix-based anomaly detection methods. RPCA is the typical Robust PCA method. RobustAE is designed based on Robust PCA and uses the autoencoder to replace the low rank constraint in robust PCA. DRMF is designed based on direct robust matrix factorization. AESc follows the PCA framework and uses the autoencoder to replace the low rank constraint in PCA.

Besides the matrix-based algorithms, four tensor-based anomaly detection methods are implemented. They are our Robust Tensor Convolutional Aotuencoder (termed RTCAE), the compressed version of our RTCAE model (termed LightCAE), FTF [17], TT-CAE [33], and Graph [47]. TT-CAE [33] follows the PCA framework and uses the convolutional autoencoder to replace the low rank constraint in PCA. Additionally, TT-CAE uses the tensor-train (TT) factorization technique to compress the convolutional autoencoder model. To make a fair comparison with our LightCAE, specially the comparisons in Section VII-D and Section VII-E, we apply the same CAE structure to TT-CAE as our LightCAE. FTF and Graph use the L_0-norm to constrain the anomaly data and the low rank constraint for the normal data. In Graph, the non-linear features are incorporated into the low rank constraint by exploiting manifold learning through the building of neighbor graph.

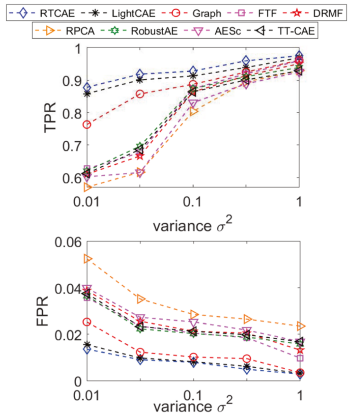


Fig. 7. Abilene: TPR and FPR under different σ^2.

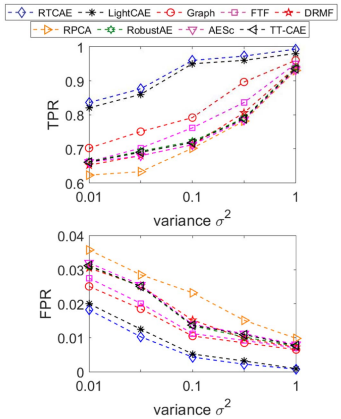


Fig. 8. GÈANT: TPR and FPR under different σ^2.

B. Anomaly Detection Accuracy

We use following two metrics to evaluate the accuracy performance.

- *False Positive Rate (FPR):* It measures the proportion of non-outliers that are wrongly identified as outliers.
- *True Positive Rate (TPR):* It measures the proportion of outliers that are correctly identified.

Smaller False Positive Rate and higher True Positive Rate mean better detection performance.

To compare the performance of different anomaly detection algorithms under the corruption from random element-wise anomalies, with other parameters fixed, we vary the variance σ^2, the mean value μ, and the outlier ratio γ of the outliers.

From Fig. 7–15, among all the algorithms compared, our proposed CAE model and LightCAE achieve the best performance. They achieve the lowest False Positive Rate and the highest True Positive Rate under all the experiment scenarios.

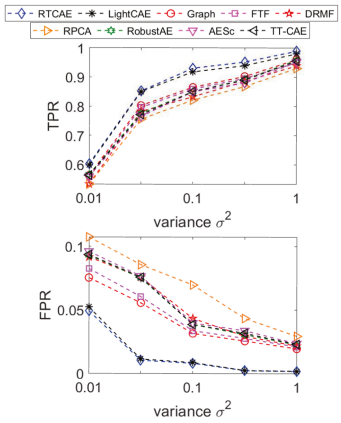


Fig. 9. Harvard: TPR and FPR under different σ^2.

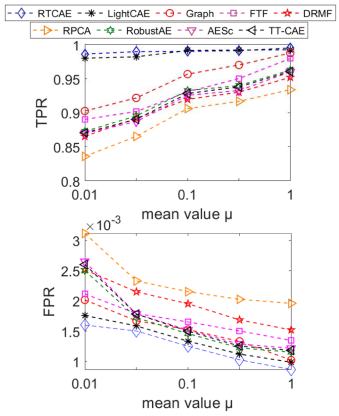


Fig. 12. Harvard: TPR and FPR under different μ.

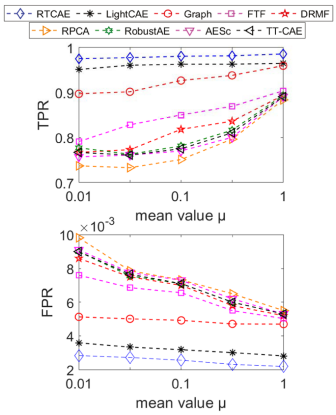


Fig. 10. Abilene: TPR and FPR under different μ.

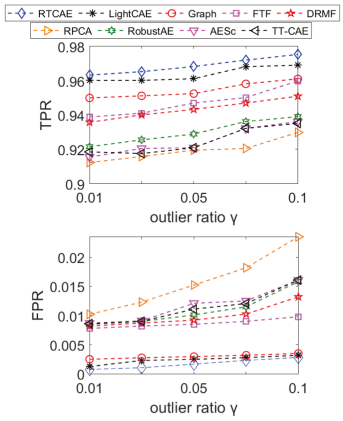


Fig. 13. Abilene: TPR and FPR under different γ.

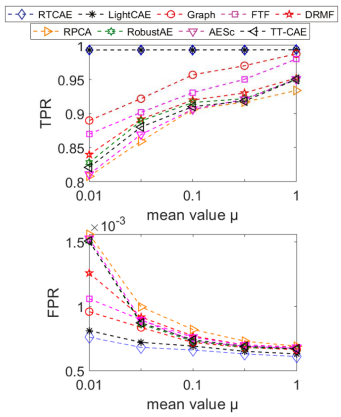


Fig. 11. GÈANT: TPR and FPR under different μ.

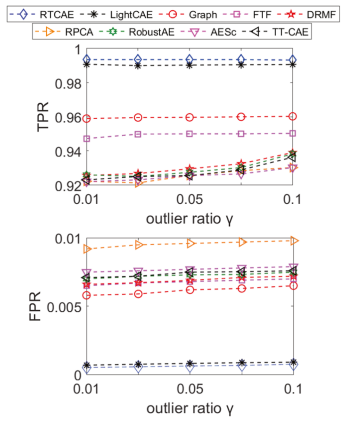


Fig. 14. GÈANT: TPR and FPR under different γ.

Similarly, we also compare the performance of all the algorithms under the structured anomalies in Table II. Our proposed CAE model and LightCAE still achieve the lowest False Positive Rate and the highest True Positive Rate, demonstrating that our models are effective in both element-wise and structured anomaly detection.

From the above results, we have following main observations.

TABLE II
COMPARING THE PERFORMANCE UNDER STRUCTURED ANOMALIES

TPR	RTCAE	LightCAE	Graph	FTF	DRMF	RPCA	RobustAE	AESc	TT-CAE
Abilene	0.9553	0.9491	0.9411	0.9398	0.9310	0.9098	0.9191	0.9161	0.9151
GÈANT	0.9732	0.9706	0.9393	0.9303	0.9190	0.9103	0.9183	0.9103	0.9163
Harvard	0.9692	0.9602	0.9393	0.9303	0.9190	0.9103	0.9383	0.9323	0.9353
FPR	CAE	LightCAE	Graph	FTF	DRMF	RPCA	RobustAE	AESc	TT-CAE
Abilene	0.0031	0.0035	0.0038	0.0101	0.0135	0.0238	0.0161	0.0163	0.0162
GÈANT	0.0011	0.0012	0.0068	0.0073	0.0075	0.0101	0.0078	0.0082	0.0079
Harvard	0.0505	0.0525	0.0756	0.0828	0.0918	0.1076	0.0933	0.0963	0.0938

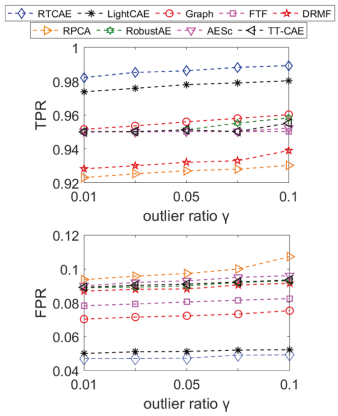


Fig. 15. Harvard: TPR and FPR under different γ.

TABLE III
RECONSTRUCTION ERROR OF DIFFERENT METHODS

RMSE	RTCAE	LightCAE	Graph	FTF	DRMF	RPCA	RobustAE	AESc	TT-CAE
Abilene	9.20e-4	9.20e-4	1.03e-3	1.29e-3	1.32e-3	1.38e-3	1.05e-3	1.15e-3	1.05e-3
GÈANT	1.85e-3	1.90e-3	2.52e-3	2.68e-3	2.72e-3	3.58e-3	1.98e-3	2.07e-3	2.0e-3
Harvard	2.57e-3	2.67e-3	5.05e-3	5.88e-3	7.02e-3	7.98e-3	3.58e-3	3.79e-3	2.83e-3

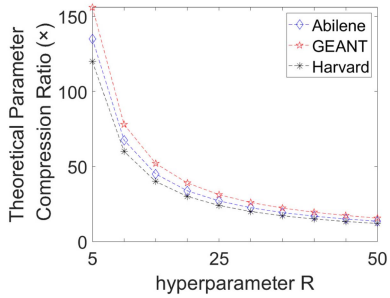


Fig. 16. Theoretical Compression Ratio of Parameters.

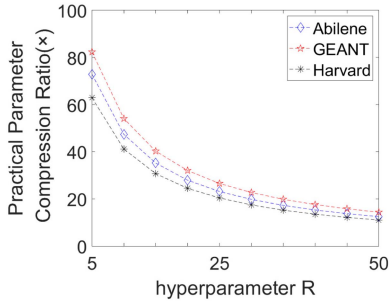


Fig. 17. Practical Compression Ratio of Parameters.

○ *The tensor model is more effective in increasing the accuracy of anomaly detection.* Compared with RobustAE which is the autoencoder based anomaly detection model, our proposed CAE and LightCAE model achieve better performance. This may because RobustAE is a matrix-based algorithm, while our RTCAE and LightCAE are tensor-based algorithms. According to [9], [10], [11], [12], [13], [14], the tensor model can learn more correlations in the data.

○ *Autoencoder can exploit nonlinear information to achieve better anomaly detection.* Compared with the other tensor-based anomaly detection methods FTF and Graph, our RTCAE and LightCAE achieve better overall performance. This is because of several reasons: 1) FTF uses the low rank constraint to reconstruct the normal data but ignores the non-linear features in data; 2) Although Graph incorporates the nonlinear proximity relation into the low rank constraint by building the neighboring graph, it still ignores a series of non-explicitly nonlinear relationships in data due to the limited ability of neighbor graph, as the low rank constraint and proximity relation are influenced by the anomalies in monitoring data and thus compromise the anomaly detection accuracy; 3) Differently, our RTCAE and LightCAE could inherit the non-linear representation capabilities [30], [48] of autoencoders that the anomalies cannot be effectively projected to a low-dimensional hidden layer.

As shown in Fig. 7–12, with the increase of variance σ^2 and mean μ of the outliers, the True Positive Rate increases while the False Positive Rate decreases for all algorithms implemented. Obviously, when the variance and mean of outliers are smaller, synthesized outlier data have closer and smaller values, and are more difficult to be differentiated from the normal data. Fig. 13–15 show the detection performance by varying the outlier ratio γ from 0.01 to 0.10. Even with a large outlier ratio at $\gamma = 0.1$, our RTCAE and LightCAE still achieve very large True Positive Rate and low False Positive Rate.

Note that, in Fig. 7–15, our proposed LightCAE model achieves similar performance with RTCAE model, which demonstrates that our compression approach is a good method that only leads to a slight drop in the accuracy performance.

C. Normal Data Reconstruction

In our algorithm, we decompose the corrupted observation data into two parts: normal data and outlier data. Normal data reconstruction directly impacts the anomaly detection. Table III lists the performance results of normal data reconstruction in term of RMSE under different methods. The error ratio $RMSE$ can be calculated as $RMSE = \frac{\sum_{t,s,i,j}^{T,S,I,J} x_{tsij} - \hat{x}_{tsij}}{T \times S \times I \times J}$ where x_{tsij} and $\hat{x}_{tsij}$ respectively denote the real data entry and reconstructed data entry. As shown in Table III, our proposed RTCAE and LightCAE achieve the lowest error of normal data reconstruction, which can help the improvement of accuracy of anomaly detection.

D. Memory Usage Under Compression

Fig. 16 shows the theoretical compression ratio of parameters calculated by (14). With the increase of hyperparameter R, compression ratio decreases as larger size factor matrices are generated in the CP decomposition. From Fig. 16, when the hyperparameter $R = 5$, the compression ratios under LightCAE are more than 135 (Abilene), 155 (GÈANT) and 120 (Harvard) respectively. Even though hyperparameter $R = 50$, a large value, the compression ratio under LightCAE is still more than 12.

TABLE IV
COMPARING THE MEMORY USAGE UNDER DIFFERENT TENSOR DECOMPOSITION TECHNIQUES

Hyperparameter R=		5	10	20(16)	30	40	50
Abilene	CP	144	220	374	527	681	834
	TT	146	229	413	615	839	1083
GÈANT	CP	132	202	340	479	618	756
	TT	135	212	385	581	801	1043
Harvard	CP	127	194	273	(Max TT-rank=16)		
	TT	135	212	385	(Max TT-rank=16)		

We further draw Fig. 17 to show the compression ratio by checking the memory usage in the implementation. The compression ratio is smaller than that under Fig. 16 for all parameter setting. This is because that except the convolution kernels, there still exist some other parameters that can not be compressed.

More specifically, when $R = 5$, the LightCAE model only needs 143 kb (Abilene), 132 kb (GÈANT) and 127 kb (Harvard) memory storage respectively, while the RTCAE needs 10.1 MB (Abilene), 10.6 MB (GÈANT) and 7.8 MB (Harvard) memory storage respectively. The reduction of memory storage allows our LightCAE model to run in smaller size and low cost processing devices.

Note that, as our LightCAE spends only small memory for one layer, for the device with fixed size memory, using our LightCAE, we can build a much deeper network that can achieve better anomaly detection performance.

We use the CP decomposition to compress the CAE model in our LightCAE, while TT-CAE exploits TT decomposition to compress the CAE model. Table IV lists the memory usage under these two decomposition techniques. We can find that, with the increase of R, the memory usage (in the unit of kb) under both compression techniques decrease as a large R leads to large decomposed factor matrices/tensors. Moreover, our LightCAE needs smaller memory usage than TT-CAE under the same R, and the memory usage gap becomes bigger with the increase of R, which demonstrates that CP decomposition can perform better parameter compression than TT decomposition. Moreover, when $R = 5$, we can find that our LightCAE model can achieve better anomaly detection accuracy in Figs. 7 to 15. These results demonstrate that our LightCAE model outperforms TT-CAE with higher anomaly detection accuracy while requiring smaller memory usage.

E. Computation Speed Under Compression

In order to evaluate the reduction of computation complexity, we use the following metric:

Speed-Up: Given the computation time under RTCAE model and LightCAE model, denoted as T_1 and T_2. The Speed-Up of LightCAE model in computation time is T_1/T_2.

Similarly, we first calculate the theoretical Speed-Up of LightCAE model by (15), as shown in Fig. 18. With the increase of hyperparameter R, the Speed-up decreases in Fig. 18.

We then run RTCAE and LightCAE on NVIDIA GeForce RTX3090 GPU. In Fig. 19, we show the Speed-Up under different R values in the model training process, with the time cost recorded directly by inserting the timer into the algorithms. We find that LightCAE can achieve more than twice the Speed-Up in both three trace data when $R = 5$. We can find that, the theoretical Speed-Up is much larger than that under practical implementation. This may because that we adopt a GPU processor to train the model, the parallelism in GPU reduces the speed gap between CAE and LightCAE.

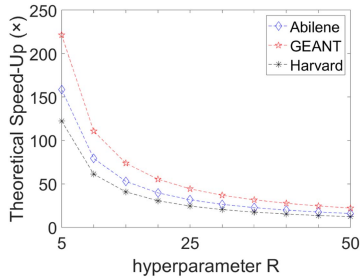


Fig. 18. Theoretical Speed-Up.

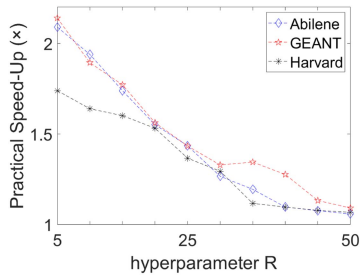


Fig. 19. Practical Speed-Up (run on NVIDIA GeForce RTX3090 GPU).

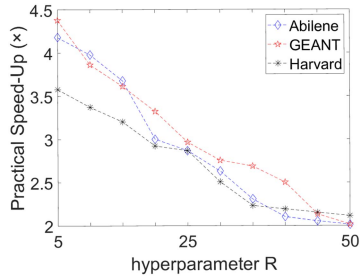


Fig. 20. Practical Speed-Up (run on Intel(R) Core(TM) i7-8750H CPU@2.2GHz).

We further run RTCAE and LightCAE on Intel(R) Core(TM) i7-8750H CPU@2.2 GHZ (a normal computer), and show the Speed-Up under different R values in Fig. 20. As the Intel CPU does not have the stronger parallelism in NVIDIA GPU, the speed gap between RTCAE and LightCAE increases. As expected, the Speed-Up in Fig. 20 is nearly twice that in Fig. 19.

We notice that the network devices usually have low computation ability because they will not be replaced and updated for a long time to ensure the stability of the network. The LightCAE model may be suitable for such devices and can achieve much higher Speed-Up in such network devices.

Similarly, we also compare the computation time (run on Intel(R) Core(TM) i7-8750H CPU@2.2 GHz) of our LightCAE model and TT-CAE model. As shown in Table V, we can find that 1) with the increase of R, the computation time (in the unit of

TABLE V
COMPARING THE COMPUTATION TIME WITH DIFFERENT TENSOR DECOMPOSITION TECHNIQUES

Hyperparameter R=		5	10	20(16)	30	40	50
Abilene	CP	0.0275	0.0262	0.0197	0.0173	0.0138	0.0132
	TT	0.0279	0.0269	0.0202	0.0178	0.0150	0.0139
GÈANT	CP	0.0251	0.0221	0.0191	0.0158	0.0143	0.0115
	TT	0.0263	0.0235	0.0207	0.0162	0.0147	0.0121
Harvard	CP	0.0223	0.0210	0.0201	(Max TT-rank=16)		
	TT	0.0232	0.0221	0.0210	(Max TT-rank=16)		

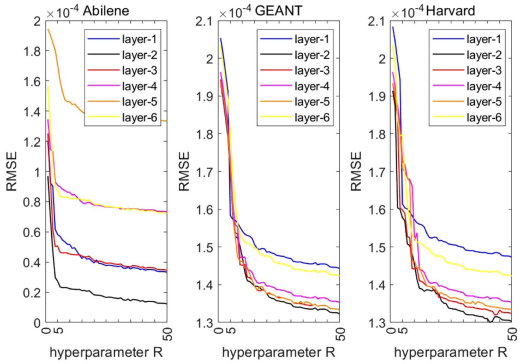


Fig. 21. Convolution kernel approximation accuracy of each layer under different hyperparameter R.

second) of CP decomposition and TT decomposition increases due to more training parameters; 2) the computation time of our LightCAE model is smaller when using the CP decomposition compared to the use of TT decomposition.

F. The Setting of Hyperparameter R

As the hyperparameter R could not only influence the memory usage and computation speed of our proposed LightCAE model but also influence the accuracy of the convolution kernel approximation and further anomaly detection accuracy.

To evaluate the influence of different hyperparameter R on the convolution kernel approximation, we first reconstruct the convolution kernels (e.g., $\hat{\mathcal{K}}$) of each layer with the corresponding four factor matrices (i.e., $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, $\mathbf{U}^{(3)}$, $\mathbf{U}^{(4)}$) by (8), and then use the $RMSE$ to calculate the error between the original convolution kernel (i.e., $\mathcal{K}$) and the reconstructed convolution kernel (i.e., $\hat{\mathcal{K}}$).

The Fig. 21 shows the $RMSE$ of convolution kernels approximation with different hyperparameter R under three data traces. With the increase of hyperparameter R, the $RMSE$ decreases as a larger R means a more accurate convolution kernel approximation.

Moreover, most curves in Fig. 21 may have an inflection point when the $R \leq 5$. As shown in Section VII-D and Section VII-E, with the increase of hyperparameter R, the memory usage compression ratio and the computation speed-up decrease. Therefore, we set the hyperparameter $R = 5$ for our proposed LightCAE, which can achieve good convolution kernel approximation accuracy, high memory usage compression ratio, and computation speed-up.

In practice, we could analyze the LightCAE during the model training process following the Section VII-D to Section VII-F, and make a trade-off based on the storage and computing power of the application device.

VIII. CONCLUSION

We propose a novel light-weight RTCAE in this paper. Different from traditional Robust PCA which is defined in a matrix form, our RTCAE extends to a tensor form which can exploit more hidden features in the data for more accurate anomaly detection. Moreover, it exploits the autoencoder instead of SVD to recover the normal data from the corrupted observed data, which can exploit both linear and non-linear features for anomaly detection. Compared with the traditional Robust PCA, the RTCAE model can achieve much higher accuracy in anomaly detection. However, directly exploiting deep autoencoder may suffer from the problem of high memory and computation overhead, which prevents our model from being used in normal computational devices. To solve the problem, we further design a compressed autoencoder to largely compress the parameters. The experiment results demonstrate that our RTCAE achieves the highest anomaly detection accuracy. Moreover, our LightCAE requires over 60 times smaller memory storage than that required in RTCAE while achieving the similar anomaly detection accuracy.

REFERENCES

[1] V. Barnett and T. Lewis, "Outliers in statistical data," New York: Wiley, 1994.
[2] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *Proc. ACM Annu. Conf. ACM Special Int. Group Data Commun. Comput. Commun. Rev.*, 2004, pp. 219–230.
[3] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *Proc. ACM Annu. Conf. ACM Special Int. Group Data Commun. Comput. Commun. Rev.*, 2005, pp. 217–228.
[4] C. Callegari, L. Gazzarrini, S. Giordano, M. Pagano, and T. Pepe, "A novel PCA-based network anomaly detection," in *Proc. IEEE Int. Conf. Commun.*, 2011, pp. 1–5.
[5] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *J. ACM*, vol. 58, no. 3, pp. 1–11, 2011.
[6] T. Bouwmans, S. Javed, H. Zhang, Z. Lin, and R. Otazo, "On the applications of robust PCA in image and video processing," in *Proc. IEEE*, vol. 106, no. 8, pp. 1427–1457, Aug. 2018.
[7] S. E. Ebadi and E. Izquierdo, "Foreground segmentation with tree-structured sparse RPCA," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 9, pp. 2273–2280, Sep. 2018.
[8] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.
[9] Y. Xu, Z. Wu, J. Chanussot, and Z. Wei, "Nonlocal patch tensor sparse representation for hyperspectral image super-resolution," *IEEE Trans. Image Process.*, vol. 28, no. 6, pp. 3034–3047, Jun. 2019.
[10] X. Li et al., "Tripartite graph aided tensor completion for sparse network measurement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 1, pp. 48–62, Jan. 2023.
[11] K. Xie et al., "Accurate recovery of internet traffic data: A sequential tensor completion approach," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 793–806, Apr. 2018.
[12] A. Baggag et al., "Learning spatiotemporal latent factors of traffic via regularized tensor factorization: Imputing missing values and forecasting," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 6, pp. 2573–2587, Jun. 2021.
[13] Y. Tang, Y. Xie, X. Yang, J. Niu, and W. Zhang, "Tensor multi-elastic kernel self-paced learning for time series clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 3, pp. 1223–1237, Mar. 2021.
[14] A. Anandkumar, P. Jain, Y. Shi, and U. N. Niranjan, "Tensor vs. matrix methods: Robust tensor decomposition under block sparse perturbations," in *Proc. Artif. Intell. Statist.*, PMLR, 2016, pp. 268–276.

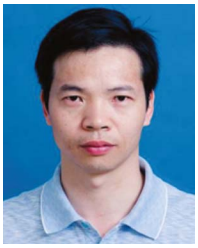
[15] L. Xiong, X. Chen, and J. Schneider, "Direct robust matrix factorizatoin for anomaly detection," in *Proc. Int. Conf. Data Mining*, 2011, pp. 844–853.
[16] S. Prativadibhayankaram, H. V. Luong, T. H. Le, and A. Kaup, "Compressive online video background–foreground separation using multiple prior information and optical flow," *J. Imag.*, vol. 4, no. 7, 2018, Art. no. 90.
[17] K. Xie et al., "Fast tensor factorization for accurate internet anomaly detection," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3794–3807, Dec. 2017.
[18] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, and D. Zhang, "Graph based tensor recovery for accurate internet anomaly detection," in *Proc. IEEE Conf. Comput. Commun.*2018, 2018, pp. 1502–1510.
[19] H. Xu et al., "Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications," in *Proc. World Wide Web Conf.*, 2018, pp. 187–196.
[20] N. Li, F. Chang, and C. Liu, "Spatial-temporal cascade autoencoder for video anomaly detection in crowded scenes," *IEEE Trans. Multimedia*, vol. 23, pp. 203–215, 2020.
[21] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Outlier detection for time series with recurrent autoencoder ensembles," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 2725–2732.
[22] F. Milković, B. Filipović, M. Subašić, T. Petković, S. Lončarić, and M. Budimir, "Ultrasound anomaly detection based on variational autoencoders," in *Proc. 12th Int. Symp. Image Signal Process. Anal.*, 2021, pp. 225–229.
[23] B. Zong et al., "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–19.
[24] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, "Deep structured energy based models for anomaly detection," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1100–1109.
[25] L. Meng, S. Ding, and Y. Xue, "Research on denoising sparse autoencoder," *Int. J. Mach. Learn. Cybern.*, vol. 8, no. 5, pp. 1719–1729, 2017.
[26] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 833–840.
[27] D. Gong et al., "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1705–1714.
[28] A.-S. Collin and C. De Vleeschouwer, "Improved anomaly detection by training an autoencoder with skip connections on images corrupted with stain-shaped noise," in *Proc. 25th Int. Conf. Pattern Recognit.*, 2021, pp. 7915–7922.
[29] Y. Qi, Y. Wang, X. Zheng, and Z. Wu, "Robust feature learning by stacked autoencoder with maximum correntropy criterion," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 6716–6720.
[30] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proc. 23rd ACM Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 665–674.
[31] S. Bhatia, A. Jain, S. Srivastava, K. Kawaguchi, and B. Hooi, "MemStream: Memory-based streaming anomaly detection," in *Proc. ACM Web Conf.*, 2022, pp. 610–621.
[32] S. Bhatia, A. Jain, P. Li, R. Kumar, and B. Hooi, "MSTREAM: Fast anomaly detection in multi-aspect streams," in *Proc. Web Conf.*, 2021, pp. 3371–3382.
[33] M. Sharma, P. P. Markopoulos, E. Saber, M. S. Asif, and A. Prater-Bennette, "Convolutional auto-encoder with tensor-train factorization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 198–206.
[34] S. Park and J. Lee, "Stability analysis of denoising autoencoders based on dynamical projection system," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 8, pp. 3155–3159, Aug. 2021.
[35] B. C. Csáji et al., "Approximation with artificial neural networks," *Fac. Sci., Etvs Lornd Univ., Hung.*, vol. 24, no. 48, 2001, Art. no. 7.
[36] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.
[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
[38] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
[39] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," 2016, *arXiv:1603.07285*.
[40] R. Ward, X. Wu, and L. Bottou, "Adagrad stepsizes: Sharp convergence over nonconvex landscapes," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2019, pp. 6677–6686.
[41] A. S. Lewis and J. Malick, "Alternating projections on manifolds," *Math. Operations Res.*, vol. 33, no. 1, pp. 216–234, 2008.
[42] "The abilene observatory data collections," 2003. [Online]. Available: http://abilene.internet2.edu/observatory/data-collections.html
[43] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM Annu. Conf. ACM Special Int. Group Data Commun. Comput. Commun. Rev.*, 2006, pp. 83–86.
[44] J. Ledlie, P. Gardner, and M. I. Seltzer, "Network coordinates in the wild," in *Proc. 4th Symp. Netw. Syst. Des. Implementation*, 2007, pp. 299–311.
[45] B. I. Rubinstein et al., "Compromising PCA-based anomaly detectors for network-wide traffic," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2008–73, 2008.
[46] X. Li et al., "Quick and accurate false data detection in mobile crowd sensing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1339–1352, Jun. 2020.
[47] D. Goldfarb and Z. Qin, "Robust low-rank tensor recovery: Models and algorithms," *SIAM J. Matrix Anal. Appl.*, vol. 35, no. 1, pp. 225–253, 2014.
[48] W. Wang, Y. Huang, Y. Wang, and L. Wang, "Generalized autoencoder: A neural network framework for dimensionality reduction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2014, pp. 490–497.

Xiaocan Li received the PhD degree from the College of Computer Science and Electronics Engineering, Hunan University, in 2020, and the exchange PhD degree from the Department of Electrical and Computer Engineering, State University of New York at Stony Brook, Stony Brook, New York. He is now an associate professor with the College of Computer Science and Electronics Engineering, Hunan University. His research interests include network measurement, network security, and matrix/tensor decomposition.

Kun Xie received the PhD degree in computer application from Hunan University, Changsha, China, in 2007. She worked as a postdoctoral fellow with the Department of Computing in Hong Kong Polytechnic University from 2007 to 2010. She worked as a visiting researcher with the Department of Electrical and Computer Engineering, State University of New York at Stony Brook from 2012 to 2013. She is currently a professor with Hunan University, Changsha, China. Her research interests include network monitoring, network security, Big Data, and AI.

Xin Wang (Senior Member, IEEE) received the PhD degree in electrical and computer engineering from Columbia University, New York, NY. She is currently an associate professor with the Department of Electrical and Computer Engineering, State University of New York at Stony Brook, Stony Brook, NY. Her research interests include algorithm and protocol design in wireless networks and communications, mobile and distributed computing, as well as networked sensing and detection.

Gaogang Xie (Member, IEEE) received the BS degree in physics, and the MS and PhD degrees in computer science all from Hunan University, in 1996, 1999, and 2002, respectively. He is currently a professor with the Computer Network Information Center(CNIC), Chinese Academy of Sciences(CAS) and the University of Chinese Academy of Sciences, and the vice president of CNIC. His research interests include Internet architecture, packet processing and forwarding, and Internet measurement.



Kenli Li received the PhD degree in computer science from the Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar with the University of Illinois at Urbana-Champaign from 2004 to 2005. He is currently the deean and a full professor of computer science and technology with Hunan Uni-versity and the director of National Supercom-puting Center in Changsha. His major research areas include parallel and distributed computing, edge computing, high-performance computing, and cloud computing.

Jiannong Cao (Fellow, IEEE) received the PhD degree in computer science from Washington State University, Pullman, WA, USA, in 1990. He is currently the Otto Poon Charitable Foundation professor in data science and the chair professor of distributed and mobile computing with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His research interests include parallel and distributed computing, wireless networking and mobile computing, Big Data and machine learning, and cloud and edge computing.

Dafang Zhang received the PhD degree in application mathematics from Hunan University, Changsha, China, in 1997. He is currently a professor with Hunan University, Changsha, China. His research interests include packet processing, Internet measurement, wireless network and mobile computing, and Big Data.

Jigang Wen received the PhD degrees in computer application from Hunan University, China, in 2011. He is now an associate professor with the School of Computer Science and Engineering, Hunan University of Science and Technology, China. His research interests include wireless network and mobile computing, high speed network measurement and management.