Sparse Laplacian Component Analysis for Internet Traffic Anomalies Detection

Manas Khatua, *Member, IEEE,* Seyed Hamid Safavi, *Student Member, IEEE,* and Ngai-Man Cheung, *Senior Member, IEEE*

***Abstract*—We consider the problem of anomaly detection in network traffic. It is a challenging problem because of high-dimensional and noisy nature of network traffic. A popularly used technique is *subspace analysis*. In particular, subspace analysis aims to separate the high-dimensional space of traffic signals into disjoint subspaces corresponding to normal and anomalous network conditions. Principal component analysis (PCA) and its improvements have been applied for this analysis. In this work, we take a different approach to determine the subspaces, and propose to capture the essence of the data using the eigenvectors of graph Laplacian, which we refer as Laplacian components (LCs). Our main contribution is to propose a regression framework to compute LCs followed by its application in anomaly detection. This framework provides much flexibility in incorporating different properties into the LCs, notably LCs with sparse loadings, which we exploit in detail. In other words, our contribution is a new framework to compute the graph Fourier transform (GFT). The proposed framework enables sparse loadings and potentially other properties to be incorporated into the analysis components of GFT to suit different tasks. Furthermore, different from previous work that uses a sample graph to preserve local structure, we advocate modeling with a dual-input feature graph that encodes the correlation of the time series data and prior information. Therefore, the proposed model can readily incorporate the 'physics' of some applications as prior information to improve the analysis. We perform experiments on volume anomaly detection using three real datasets. We demonstrate that the proposed model can correctly uncover the essential low-dimensional principal subspace containing the normal Internet traffic and achieve outstanding detection performance.**

***Index Terms*—Network Anomaly Detection, Graph Signal Processing, Graph Fourier Transform, Dimensionality Reduction, Graph Laplacian, Sparse Loadings**

I. Introduction

Network traffic anomalies are considered as unusual but significant changes present in network traffic [5]. Example includes both the legitimate activities such as flash crowds, sudden changes in customer demand, and illegitimate activities such as port scans, distributed denial-of-service (DDoS), link flooding attack [6]. Very often, the collected data used for network anomaly detection is huge, high-dimensional, noisy and grossly distorted. Therefore, processing and analyzing such massive data in time-critical environment poses unprecedented challenges. We propose to address anomaly detection in massive data traffic by exploiting recent discoveries in high-dimensional graph signal analysis [7].

Determining anomalies in network data streams has attracted a significant amount of research efforts [8], [9], [10], [11], [12], [13], [14]. Mainly, there exist two paths of work for network anomaly detection: signature-based and non-signature-based detection. The second approach is useful for unknown threat and anomaly as it does not require any prior knowledge about the anomalies. In the domain of non-signature-based detection, *subspace analysis* is a popular approach that aims to partition the high-dimensional traffic signal space into disjoint subspaces corresponding to the normal and anomalous network conditions [8], [10]. One of the most critical requirements of subspace analysis is to uncover the essential low-dimensional normal traffic subspace from the noisy and high-dimensional traffic. Many spectral techniques such as PCA have been proposed to address this dimensionality analysis problem. Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p\times n}$ be the high-dimensional traffic measurement. In particular, $\mathbf{X}$ consists of n measurements of dimensionality p (In our experiment, $\mathbf{X}$ consists of measurements in n successive time intervals; each measurement consists of traffic statistics of p links of the network). The *classical PCA* (Model 1 in Table I) finds the projection $\mathbf{Q}^T \in \mathbb{R}^{k\times n}$ of $\mathbf{X}$ on a k-dimensional ($k \leq p$) linear space characterized by an orthogonal basis $\mathbf{V} \in \mathbb{R}^{p\times k}$. The product $\mathbf{VQ}^T$ is known as the low-rank approximation $\mathbf{L} \in \mathbb{R}^{p\times n}$ of $\mathbf{X}$. The clustering or subspace analysis can then be performed on the $\mathbf{L}$.

It is observed that, in many cases, low-dimensional data follows certain structures which are hidden in the original data. The performance of different applications such as dimensionality reduction, clustering, and anomaly detection could be improved if we leverage that structures in the models. Therefore, there is a trend to improve the performance of PCA by utilizing the hidden structure in a form of graph [15], [16], [17], [18], [19], [1], [2], [20], [21], [3], [22], [23]. These works mainly consider the graph structure based on *sample similarity*. Few of them such as [16] considered *feature similarity* along with sample similarity graph.

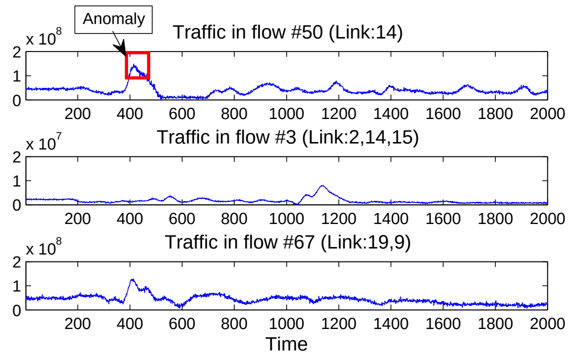
A. Focus of this work

The assumption in most of the graph-based models is that the data is "smooth" on the underlying graph, and they use the corresponding graph Laplacian to impose the

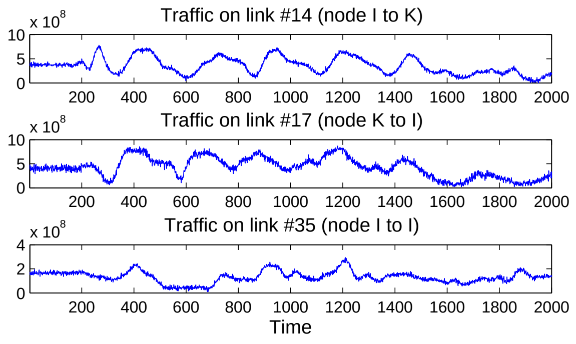
Manas Khatua is with the Indian Institute of Technology Jodhpur, Rajasthan, India-342037. E-mail: manaskhatua@gmail.com
Seyed Hamid Safavi is with the Shahid Beheshti University, Iran-1983969411. E-mail: hamid.safavy@gmail.com
Ngai-Man Cheung is with the Singapore University of Technology and Design (SUTD), Singapore-487372. E-mail: ngaiman_cheung@sutd.edu.sg
Most work was done while the authors were at SUTD, Singapore.

TABLE I: A comparison on the properties of classical PCA, RPCA, and various recent graph-based PCA models [1], [2], [3]. $\|.\|_1, \|.\|, \|.\|_F$ and $\|.\|_*$ denote the l_1, l_2, Frobenious, and nuclear norm, respectively. $\delta, \gamma, \gamma_1, \gamma_2$ are the weighting constants. $\mathbf{M}$ is the sparse matrix. In our proposed LCA and SLCA, $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_m\}^T$ is a $m \times p$ matrix where $\mathbf{s}_i = \{s_{i1}, \ldots, s_{ip}\}$, and $\mathbf{S}^T\mathbf{S}$ is the $p \times p$ Laplacian matrix of the *dual-input feature graph*. G is the graph structure representation of prior information. $\mathcal{F}_2$ represents the weight matrix computation function. $\mathbf{b}_j$ is the j^{th} column vector, and the Laplacian components $\mathbf{V} = \mathbf{B}_{p\times k} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}, k \leq p$.

#	Model	Objective	Constraints	Parameter	Graph on		
					samples	features	'physics'
1	PCA	$\min_{\mathbf{V},\mathbf{Q}} \|\mathbf{X} - \mathbf{V}\mathbf{Q}^T\|_F^2$	$\mathbf{V}^T\mathbf{V} = \mathbf{I}$	k	×	×	×
2	RPCA [4]	$\min_{\mathbf{L},\mathbf{M}} \|\mathbf{L}\|_* + \delta\|\mathbf{M}\|_1$	$\mathbf{X} = \mathbf{L} + \mathbf{M}$	δ	×	×	×
3	RPCAG [2]	$\min_{\mathbf{L},\mathbf{M}} \|\mathbf{L}\|_* + \delta\|\mathbf{M}\|_1 + \gamma_1\ tr(\mathbf{L}\Phi_s\mathbf{L}^T)$	$\mathbf{X} = \mathbf{L} + \mathbf{M}$	δ, γ_1	✓	×	×
4	FRPCAG [3]	$\min_{\mathbf{L}} \|\mathbf{X} - \mathbf{L}\|_1 + \gamma_1\ tr(\mathbf{L}\Phi_s\mathbf{L}^T) + \gamma_2\ tr(\mathbf{L}^T\Phi_f\mathbf{L})$		γ_1, γ_2	✓	✓	×
5	GLPCA [1]	$\min_{\mathbf{V},\mathbf{Q}} \|\mathbf{X} - \mathbf{V}\mathbf{Q}^T\|_F^2 + \gamma_1\ tr(\mathbf{Q}^T\Phi_s\mathbf{Q})$	$\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$	k, γ_1	✓	×	×
6	LCA (this work)	$\min_{\mathbf{A},\mathbf{B}} \sum_{i=1}^m \|\mathbf{s}_i - \mathbf{A}\mathbf{B}^T\mathbf{s}_i\|^2 + \gamma\sum_{j=1}^k \|\mathbf{b}_j\|^2;\ \mathcal{F}_2(\mathbf{X}, G) \Rightarrow \Phi_{df}$	$\mathbf{A}^T\mathbf{A} = \mathbf{I}_{k\times k}$	k, γ	×	✓	✓
7	SLCA(this work)	$\min_{\mathbf{A},\mathbf{B}} \sum_{i=1}^m \|\mathbf{s}_i - \mathbf{A}\mathbf{B}^T\mathbf{s}_i\|^2 + \gamma\sum_{j=1}^k \|\mathbf{b}_j\|^2 + \sum_{j=1}^k \delta_j\|\mathbf{b}_j\|_1$	$\mathbf{S}^T\mathbf{S} = \Phi_{df}$	k, γ, δ			



(a) Amount of traffic in an origin-destination flow.



(b) Amount of traffic in a link

Fig. 1: Flow and link level measurements of traffic in the Abilene network. Note that the vertical axes have different scales, and the flow measurements are small compared to link measurements.

smoothness constraints during the recovery of the low-rank approximation of data. Specifically, they use spectral graph regularization in the optimization problems to impose graph smoothness [1], [2], [3]. On the other hand, inspired by the recent graph signal processing [7], [24], [25], [26], our work takes a different approach and imposes graph smoothness using the first k eigenvectors of the *feature graph Laplacian*, where $k < p$ for p-dimensional input data. Note that, almost all state-of-the-art methods computed the eigenvectors from "sample graph Laplacian", whereas this paper compute the same from "feature graph Laplacian". As will be discussed, our approach takes the view of signal reconstruction of the normal traffic, which is assumed to be graph-smooth. Our main contribution is to propose a regression framework to compute these eigenvectors which allows us to introduce different properties on the eigenvectors, such as sparse loadings in the components, leading to improved detection accuracy. On the contrary, eigedecomposition-based design does not have such flexibility to add properties on eigenvectors. In other words, our contribution is an improved method for graph Fourier transform (GFT) that enables different properties to be included in the GFT analysis components. In addition, in many cases, the low-dimensional data follows the "physics" of the problem at hand which is known a priori. Examples of such prior knowledge could be the physical proximity of the sensors in wireless sensor networks (WSNs) [27] or the magnetometers in Magnetoencephalography (MEG) for brain imaging [28], [29], or the network routing topology in Internet traffic analysis [8]. A standard way to incorporate the knowledge of such information is by using a graph. Such prior information could be useful when the data samples are noisy. This knowledge motivated us to investigate a mechanism to incorporate prior information in the process of recovering low-rank data matrix. Specifically, we focus on the graph-based modified PCA with prior information for more accurate subspace identification followed by its application in detection of network traffic anomaly such as volume anomaly [8]. Note that the application of graph signal processing for network traffic analysis is new as per our knowledge. A preliminary version of this method has been published in [30]. In this version, we extended our analysis and added new experimental results with few more datasets.

At this juncture, it is pertinent to explain the volume anomaly briefly. The volume anomaly spans over multiple links in a network. It is referred to a sudden positive or negative change in amount of traffic of an origin-destination (OD) flow at some specific time while the other flows carry average amount of traffic throughout the period. Figure 1(a) shows an example of anomalous flow (flow #50) and two other normal flows (flow #3 and #67) captured from the Abilene network (more discussion about the Abilene network is given in Sections IV and V). The occurrence of such anomalous event is visible in flow-level data as shown in Figure 1(a). Our objective is to

TABLE II: Summary of notations

Notation	Description
$\|\|.\|\|_F$	Matrix Frobenious norm
$\|\|.\|\|_*$	Matrix nuclear norm
$\|\|.\|\|_1$	Matrix l_1 norm
$\|\|.\|\|$	Matrix l_2 norm
n	Number of data samples
p	Number of features in each sample
k	Reduced dimension ($k < p$)
$\mathbf{X} \in \mathbb{R}^{p \times n}$	Data matrix
$\mathbf{x}$	Denote a column vector in $\mathbf{X}$
$\mathbf{L} \in \mathbb{R}^{p \times n}$	Low-rank approximation of $\mathbf{X}$
δ	Sparsity penalty
γ	Ridge penalty
γ_1	Graph penalty on sample graph
γ_2	Graph penalty on feature graph
$\mathcal{F}_1$	Graph conversion function
$\mathcal{F}_2$	Weight matrix computation function for a graph
G_s	Sample graph
G_f	Feature graph
G_L	Network link graph
$\mathcal{G}$	Source graph or dual-input feature graph
$\Phi_s \in \mathbb{R}^{n \times n}$	Laplacian for sample graph
$\Phi_f \in \mathbb{R}^{p \times p}$	Laplacian for feature graph
$\Phi_{df} \in \mathbb{R}^{p \times p}$	Laplacian for dual-input feature graph (source graph)
$\mathbf{V}$	Eigenvectors of Graph Laplacian
$\mathbf{D}$	Degree matrix of an weighted graph
$\mathbf{W}$	Weight matrix of an weighted graph
$\mathcal{V}$	Set of vertices in graph
$\mathcal{E}$	Set of edges in a graph
$\mathbf{S}$	It is a matrix which satisfies $\mathbf{S}^T\mathbf{S} = \Phi_{df}$

identify the time-stamps of anomalous events using the *link-level measurements*. Note that link-level measurements can be easily captured from networks, and this approach is scalable for network-wide monitoring. We do not use flow-level statistics as in previous work [8]. On the other hand, from the Figure 1(b), we observe that the occurrence of anomalous event is not prominently visible using link-level measurements of the links #14, #17 and #35. The amount of traffic in those links at normal time are similar to that at abnormal time because of the superposition of normal and abnormal flows passing through the links, even though the link #14 is directly associated with the anomalous flow #50 according to the Abilene network. Therefore, it is challenging to determine the existence of network anomaly from link-level traffic measurements.

B. Contribution

Let $\Phi_s \in \mathbb{R}^{n \times n}$ and $\Phi_f \in \mathbb{R}^{p \times p}$ be the graph Laplacian on the sample graph G_s and the feature graph G_f of $\mathbf{X}$, respectively. The graph G_s connects the different samples of $\mathbf{X}$ (columns of $\mathbf{X}$) and the graph G_f connects the features of $\mathbf{X}$ (rows of $\mathbf{X}$). In this paper, we propose a subspace analysis for detecting volume anomaly. We apply the eigenvectors of the graph Laplacian $\Phi_{df} \in \mathbb{R}^{p \times p}$ of a *dual-input feature graph*. The dual-input feature graph incorporates both the data matrix and the prior information to encode the correlation between the features (we also name it as a *source graph* because it encodes the correlation between the data sources). As will be discussed, the prior information in our application is the network topology. Note that the subscripts 'f' and 'df' with Φ indicate default feature graph (incorporates only the information from data) and dual-input feature graph, respectively (more discussion in Sections III and IV). In particular, for p-dimensional data that can be defined on the vertices of a graph $\mathcal{G}$ ($|\mathcal{V}| = p$), we use the first k smooth eigenvectors $\{\mathbf{v}_1, ..., \mathbf{v}_k\}$ of Φ_{df} to define the intrinsic k-dimensional subspace corresponding to normal network conditions. Our approach uses a different mechanism to impose the graph smoothness constraint: instead of using spectral graph regularization and relying on γ_1, γ_2 to control the graph smoothness (as in Model 3 to 5 in Table I), we control the graph smoothness directly via the selection of k smooth eigenvectors for reconstruction of samples of $\mathbf{X}$.

Moreover, as our main contribution, we propose to use a regression-type optimization framework to compute the orthogonal bases $\{\mathbf{v}_l\}$ from the normalized graph Laplacian Φ_{df}. We name $\{\mathbf{v}_l\}$ as Laplacian components (LCs) and the regression-based approach as *Laplacian component analysis* (LCA). LCA produces the same resulting vectors as direct eigendecomposition of Φ_{df}. On the other hand, LCA provides flexibility to achieve different properties of the resulting LCs. In particular, in this work, we exploit the use of lasso penalty in the baseline LCA to obtain LCs with sparse loadings. We validate that this *sparse LCA* can better model the class of signals that are smooth with respect to the underlying graph (e.g., link-level traffic statistics in our application), leading to better detection performance.

We summarize our contributions as follows:

- We propose a new framework namely Laplacian Component Analysis (LCA). LCA performs regression on the structure of the graph Laplacian. It can be viewed as an extension of PCA with graph smoothness constraints.
- We present a method to include additional attributes (sparse loading) upon eigenvectors of the graph Laplacian using the LCA framework. We name this method as sparse LCA (SLCA).
- We design a network anomaly detection scheme by introducing dual-input feature graph (i.e. source graph) with (sparse) Laplacian component analysis.

C. Paper Organization

At the outset, we describe the notations used throughout the paper in Table II. The rest of the paper is organized as follows. In Section II, we present relevant related works reported in the existing literature. We describe the proposed model followed by its optimization solution in Section III. Section IV describes the graph construction method. The experimental setup for performing evaluation is described in Section V followed by performance evaluation and comparison with other benchmark models in Section VI. Finally, we conclude the paper in Section VII with discussions about how this work can be extended in the future.

II. Related Work

The aim of this work is to detect anomalous traffic volumes in a large network. The information sources are collected from the network and are focused on both the components - data and network, so that the robust and scalable detection of network traffic anomaly can be performed. There exist two major approaches for anomaly detection - signature-based and non-signature-based. We consider the non-signature-based approach as it is advantageous for unknown anomaly and zero-day attack [6]. We find a rich set of literature on network traffic anomaly detection [5], [31], [6], [32], [33], [34], [35], [36], [37], [38]. Brauckhoff *et al.* [39] applied association rule mining on the meta-data provided by histogram-based detectors for detecting anomalous flows. However, the work is based on the assumption that anomalies typically result in many flows with similar characteristics which is not true always [40]. Yut-Fong *et al.* [41] developed a non-parametric change point detection and localization method for high-dimensional network traffic data. However, their design is not robust with different data streams generated from a network [40]. Yang and Zhou [42] used the manifold learning technique called locally linear embedding (LLE) [43] with PCA for outlier detection. However, this method is not promising as the original data space is changed in the process and there is no mapping provided from the new space to the old space. URCA [44] searches for anomalous flows by iteratively eliminating subsets of normal flows. However, the URCA is very costly to execute in a network because of its repeated execution of anomaly detector on different subsets of flows.

The present network anomaly detection schemes have evolved to highly sophisticated levels and use many advanced signal processing techniques. The emerging signal processing techniques play a significant role to perform data mining on massive amounts of collected data, and, ultimately, detect the anomaly with much improved accuracy, reduced computational and storage overheads. One of the essential requirements for anomaly detection in high dimensional data space is to reduce the effective dimension. Many spectral techniques such as PCA have been proposed to address the dimensionality reduction problem [45]. The classical PCA suffers from several disadvantages, e.g., sensitive to outliers [16], [4], [27]. Therefore, many improvements over the classical PCA have been proposed. Candes *et al.* [4] proposed *Robust PCA* (RPCA, Model 2 in Table I) which is robust to outliers by directly recovering the low-rank matrix $\mathbf{L}$ from the grossly corrupted $\mathbf{X}$. In this model, the $\mathbf{M}$ represents sparse matrix containing error and $\|.\|_*$ denotes the nuclear norm. Recently, there is a trend to improve PCA by leveraging the hidden structure of data matrix in the form of a graph for improving the performance of various applications such as dimensionality reduction, clustering, anomaly detection. These works consider structure of feature similarity, sample similarity, and combination of both in the form of graphs. The *graph Laplacian PCA* (GLPCA)[1](Model 5 in Table I) considers implicit structure among data samples for improving the accuracy of clustering. The authors further proposed one robust version of the GLPCA. Shahid *et al.* [2] proposed *robust PCA on graph* (RPCAG) (Model 3 in Table I) which can accurately learn $\mathbf{L}$ in the presence of occlusion and missing pixel. Note that GLPCA [1] assumes the graph smoothness of the projected data $\mathbf{Q}$ while RPCAG [2] assumes the graph smoothness of the low-rank approximation $\mathbf{L}$. These smoothness constraints are used as the regularization. Considering the feature similarity graph, *Laplacian Lasso* (LLasso) [16] proposed a network-constrained regularization procedure in which the lasso penalty is combined with the network penalty induced by Laplacian matrix of the graph. Thereafter, different modifications of the LLasso has been proposed such as *graph-based elastic net* [46], and *graph OSCAR* [18]. Inspired by the two-way graph regularization scheme [47], Shahid *et al.* [3] proposed *Fast RPCAG* (FRPCAG) (Model 4 in Table I) for faster and better clustering. Several other techniques for dimensionality reduction, such as [15], [48], [49] and [50], involve eigen-decomposition of the graph Laplacian. In particular, LPP [15] is a linear mapping that involves graph Laplacian and can be seen as an alternative to PCA, similar to our work.

However, the framework and mechanism of our work are different from LPP and previous graph based works. Specifically, the LPP is geometrically motivated. Given a set of n p-dimensional points $\mathbf{x}_i$, LPP aims to preserve local structure of the n points. The structure is modelled by the *sample* graph with n nodes and edge weights measure the distance between $\mathbf{x}_i$. This is encoded by the $n \times n$ graph Laplacian Φ_s. As will be discussed more, our approach instead works with a different type of *feature* graph of p nodes namely *source-graph*. This is motivated from a different angle: we consider p correlated time series and $\mathbf{x}_i$ is the p-dimensional measurement at a time instant. The edge weights of the source-graph encode the cross-correlation of the time series as well as the relationship between nodes in the network structure which is available a priori. In brief, our approach focuses on the $p \times p$ graph Laplacian Φ_f, or Φ_{df} if prior information is incorporated. Besides, the mechanisms are different. For LPP, the solution is the set of eigenvectors of $X\Phi_s X^T$. For our work, LCA components are equivalent to the eigenvectors of Φ_{df}. We do not perform eigen-decomposition. Instead, we propose to regress on the structure of Φ_{df} to compute the components. Note that our work outperforms LPP, as will be discussed in our experiment. In addition, Spectral Regression [17] has been proposed as a two-step more efficient approach to solve the eigen-problem of a *specific* form. It is not applicable for our eigen-problem. Also, our algorithm performs regression directly on the structure of the Laplacian in a single step. Note that in our application the graph connectivity can be readily obtained (network topology). In other applications when connectivity is unknown, our proposed regression framework can be used with graph learning algorithms [51], [52].

In addition, our work is an improvement of the graph Fourier transform (GFT). The LCA components computed from our regression framework are the same as the analysis components of an ordinary GFT computed from eigendecomposition. Importantly, our regression framework enables other properties to be included in the analysis components by using different regularizations. In this work, we investigate in detail using the lasso penalty to obtain components with sparse loadings, which have been shown to be effective for anomaly detection [40]. Very recently, there are several works to improve GFT. However, the motivation and approach of these works are significantly different from our work: Sardellitti *et al.* [53] and Shafipour *et al.* [54] extend GFT for directed graphs. Magoarou *et al.* [55] investigate fast GFT. Girault *et al.* [56] consider irregular-aware GFT. On the other hand, our work investigates incorporating different properties such as sparse loading into the GFT analysis components, so that they are effective for specific analysis tasks.

III. Methodology

In network traffic anomaly detection, it is desirable to have the ability to distinguish abnormal traffic pattern from the normal pattern. Lakhina *et al.* [8] uses PCA for distinguishing normal and abnormal subspaces by constructing and separating the principal axes into two sets corresponding to the subspaces. The space spanned by the set of normal axes is considered as normal subspace. In this work, we propose a more accurate method for determining normal subspace, and, therefore, the performance of anomaly detection increases. This section describes the steps for finding out the principal axes followed by their sparse approximation for computing more accurate normal subspace.

A. Overview of Proposed Method

In brief, **our proposed method** is as follows:

$$G(\mathcal{V},\mathcal{E}) \Leftarrow \mathcal{F}_1(\text{Prior Information}) \tag{1}$$

$$\mathcal{G}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{D}) \Leftarrow \mathcal{F}_2(G,\mathbf{X});\ \mathbf{X}\in\mathbb{R}^{p\times n} \tag{2}$$

$$\Phi_{df} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}} \tag{3}$$

$$\min_{\mathbf{A},\mathbf{B}} \sum_{i=1}^{m} \|\mathbf{s}_i - \mathbf{A}\mathbf{B}^T\mathbf{s}_i\|^2 + \gamma\sum_{j=1}^{k}\|\mathbf{b}_j\|^2 + \sum_{j=1}^{k}\delta_j\|\mathbf{b}_j\|_1$$
$$s.t.\ \mathbf{A}^T\mathbf{A} = I_{k\times k},\ \mathbf{S}^T\mathbf{S} = \Phi_{df} \tag{4}$$

where $\mathbf{B}_{p\times k} = [\mathbf{b}_1,\ldots,\mathbf{b}_k]$, and γ and δ are the tuning parameters. At the outset, we describe the above method briefly followed by their detail explanation in the following subsections. The input to our method is the data $\mathbf{X}\in\mathbb{R}^{p\times n}$ and *prior information*. In our application, $\mathbf{X}$ consists of measurements in n successive time intervals; each measurement consists of traffic statistics of p links/traffic measuring points of the network. Alternatively, $\mathbf{X}$ can be viewed as p time series with n samples. The time series are correlated and the cross-correlation of the time series is encoded in the weights of the underlying graph model. The output of our method is the set of Laplacian Components obtained in $\mathbf{B}$ computed from (4).

The step in (1) shows that the available prior information needs to be converted to an undirected graph structure $G(\mathcal{V},\mathcal{E})$ where $\mathcal{V}$ and $\mathcal{E}$ are the set of vertices and edges, respectively. This is done by designing a conversion function $\mathcal{F}_1(.)$. The conversion process of prior information to a graph structure is specific to application domain, and, therefore, we discuss with graph construction method in Section IV. Let us assume, we have the graph G. We compute weight matrix $\mathbf{W}$ and degree matrix $\mathbf{D}$ of the graph using both the graph structure and the input data matrix, as mentioned in (2). Because of the use of prior information graph G in the conversion function $\mathcal{F}_2(.)$, we describe its operation with graph construction method in Section IV.

After (2), we have a simple, undirected, connected and weighted graph $\mathcal{G}(\mathcal{V},\mathcal{E},\mathbf{W},\mathbf{D})$. The step in (3) shows the computation method of normalized graph Laplacian of graph $\mathcal{G}$ followed by its conversion to $\mathbf{S}$ matrix which is, then, used as input in the optimization framework shown in (4). Finally, the computed k sparse LCs are $\{\mathbf{v}_1,...,\mathbf{v}_k\} = \mathbf{B}$, which are used to define the low-rank approximation of $\mathbf{X}$ corresponding to the normal subspace.

B. Regression Framework to Compute LCs

Unlike previous work using eigendecomposition [7], we propose to use a *regression-type optimization framework* to compute the eigenvectors from the graph Laplacian Φ_{df}. As will be discussed later, this allows us to impose different types of constraints to achieve different properties of the resulting LCs, e.g., sparse loadings. Specifically, since Φ_{df} is real, positive semi-definite and diagonalizable, we can write $\Phi_{df} = \mathbf{S}^T\mathbf{S}$ for some $m\times p$ matrix $\mathbf{S}$. Note that we introduce $\mathbf{S}$ to illustrate our mathematical formulation. In practice, our algorithm uses Φ_{df} directly, and there is no need to compute $\mathbf{S}$, as will be discussed. We apply the theorem from [57] to convert the computation of eigenvectors into a regression problem. Specifically, suppose we are considering the first k LCs $(\mathbf{v}_1,\ldots,\mathbf{v}_k)$ of $\Phi_{df} = \mathbf{S}^T\mathbf{S}$, with $\mathbf{S} = [\mathbf{s}_1,\ldots,\mathbf{s}_m]^T$. Let $\mathbf{A}_{p\times k} = [\mathbf{a}_1,\ldots,\mathbf{a}_k]$ and $\mathbf{B}_{p\times k} = [\mathbf{b}_1,\ldots,\mathbf{b}_k]$. For any $\gamma > 0$, let

$$(\hat{\mathbf{A}},\hat{\mathbf{B}}) = \arg\min_{\mathbf{A},\mathbf{B}} \sum_{i=1}^{m}\|\mathbf{s}_i - \mathbf{A}\mathbf{B}^T\mathbf{s}_i\|^2 + \gamma\sum_{j=1}^{k}\|\mathbf{b}_j\|^2$$
$$s.t.\ \mathbf{A}^T\mathbf{A} = I_{k\times k}, \mathbf{S}^T\mathbf{S} = \Phi_{df} \tag{5}$$

where γ is a ridge penalty factor for each principal component (PC). Using the result from [57], we have $\hat{\mathbf{b}_j} \propto \mathbf{v}_j$ for $j = 1,2,\ldots,k$. Note that we are interested in the first k principal components (i.e. LCs) of the Laplacian matrix, which are different from PCs in [57]. In particular, our interested Laplacian components correspond to the smaller eigenvalues of Φ_{df} which include the zero eigenvalue. Note that the principal components (PCs) corresponding to higher eigenvalues are used for subspace analysis in PCA based approaches.

To illustrate the link between (5) and regression analysis, note that when $\mathbf{A}$ is given, following [57], it can be shown that (5) becomes:

$$\arg\min_{\mathbf{B}} \sum_{j=1}^{k}\|\mathbf{S}\mathbf{a}_j - \mathbf{S}\mathbf{b}_j\|^2 + \gamma\sum_{j=1}^{k}\|\mathbf{b}_j\|^2. \tag{6}$$

Therefore, with $\mathbf{S}\mathbf{a}_j$ being viewed as the (known) response vector and $\mathbf{b}_j$ the regression coefficients, (6) is equivalent to k independent ridge regression problems. We named the formulation in (5) as Laplacian component analysis (LCA). It is pertinent to mention that, unlike Sparse PCA (SPCA) in which data matrix $\mathbf{X}$ is used in regression formulation, the LCA as well as the Sparce LCA (discussed in Section III-D) does not directly use the data matrix $\mathbf{X}$ in regression. Instead, the proposed models use the matrix $\mathbf{S}$ in optimization problem formulation where $\mathbf{S}^T\mathbf{S} = \Phi_{df}$, the graph Laplacian matrix. One example of $\mathbf{S}$ could be the incidence matrix which is easy to obtain, but with dimension $m\times p$, and m is the number of edges which would be very large if the graph is dense. Thus, it could be inefficient to work with $\mathbf{S}$. In Section III-D, we propose the solution method of (6) in which FISTA algorithm [58] is used to solve the "B given A" step. This allows reformulation so that we do not need to work with $\mathbf{S}$. Rather, in the solver, we can directly use the Laplacin matrix Φ_{df}.

C. Relation of Principal Subspace with Weight Matrix

Prior information such as network topology has influence on correlated measurements and it is encoded in Φ_{df} for extraction of smooth component $\mathbf{b}_j$ where $j \in [1, 2, \ldots, k]$. When $\mathbf{A}$ is given, we obtain $\mathbf{b}_j$ after solving (6). We can rewrite (6) for every $\mathbf{b}_j$ as follows:

$$\mathbf{b}_j = \arg\min_{\mathbf{b}_j} (\mathbf{a}_j - \mathbf{b}_j)^T \mathbf{S}^T \mathbf{S} (\mathbf{a}_j - \mathbf{b}_j) + \gamma \|\mathbf{b}_j\|^2 \tag{7}$$

As we know the $\mathbf{A}$ matrix, the modified equation shows that we need only $\mathbf{S}^T\mathbf{S}$ for solving (6). Let $\mathbf{b}_j^{'} = \mathbf{a}_j - \mathbf{b}_j$ and we know that $\mathbf{S}^T\mathbf{S} = \Phi_{df}$. Then, we further rewrite (7) as follows:

$$\begin{aligned}\mathbf{b}_j &= \arg\min_{\mathbf{b}_j} \mathbf{b}_j^{'T} \Phi_{df} \mathbf{b}_j^{'} + \gamma \|\mathbf{b}_j\|^2 \\ &= \arg\min_{\mathbf{b}_j} \sum_{r,s\in[1,2,\ldots,p]} w_{r,s} [b_j^{'}(r) - b_j^{'}(s)]^2 + \gamma \|\mathbf{b}_j\|^2\end{aligned} \tag{8}$$

Therefore, for a given $\mathbf{A}$, $\mathbf{b}_j^{'T} \Phi_{df} \mathbf{b}_j^{'}$ in (8) forces the r-th and s-th entries of $\mathbf{a}_j - \mathbf{b}_j$ to have have similar values if $w_{r,s}$ is large.

D. Sparse LCA (SLCA)

The regression formulation in (5) is a flexible framework that allows various enhancement of the baseline LCA. In particular, to achieve LCs with sparse loadings, we add the lasso penalty in (5):

$$\begin{aligned}(\hat{\mathbf{A}}, \hat{\mathbf{B}}) = \arg\min_{\mathbf{A},\mathbf{B}} & \sum\nolimits_{i=1}^{m} \|\mathbf{s}_i - \mathbf{A}\mathbf{B}^T\mathbf{s}_i\|^2 + \gamma \sum\nolimits_{j=1}^{k} \|\mathbf{b}_j\|^2 \\ & + \sum\nolimits_{j=1}^{k} \delta_j \|\mathbf{b}_j\|_1 \\ & s.t.\ \mathbf{A}^T\mathbf{A} = I_{k\times k}, \mathbf{S}^T\mathbf{S} = \Phi_{df}\end{aligned} \tag{9}$$

Note that different δ_j can be used for different LCs to promote sparse loadings. To solve (9), we use the alternating algorithm proposed in [57] and solve: (i) $\mathbf{B}$ given $\mathbf{A}$ and (ii) $\mathbf{A}$ given $\mathbf{B}$. On the other hand, we propose to re-formulate the problem to apply the popular fast iterative shrinkage thresholding algorithm (FISTA) [58] to attack $\mathbf{B}$ given $\mathbf{A}$ efficiently. In particular, following the method of conversion from (5) to (6), it can be shown that (9) is equivalent to:

$$\begin{aligned}\min_{\mathbf{A},\mathbf{B}} & \sum_{j=1}^{k} \|\mathbf{S}\mathbf{a}_j - \mathbf{S}\mathbf{b}_j\|^2 + \gamma \sum_{j=1}^{k} \|\mathbf{b}_j\|^2 + \sum_{j=1}^{k} \delta_j \|\mathbf{b}_j\|_1 \\ & s.t.\ \mathbf{A}^T\mathbf{A} = I_{k\times k}, \mathbf{S}^T\mathbf{S} = \Phi_{df}\end{aligned} \tag{10}$$

Moreover, if the first two terms are combined together, we reach a simple version:

$$\begin{aligned}\min_{\mathbf{A},\mathbf{B}} & \sum_{j=1}^{k} \|\tilde{\mathbf{S}}\mathbf{a}_j - \bar{\mathbf{S}}\mathbf{b}_j\|^2 + \sum_{j=1}^{k} \delta_j \|\mathbf{b}_j\|_1 \\ & s.t.\ \mathbf{A}^T\mathbf{A} = I_{k\times k}, \mathbf{S}^T\mathbf{S} = \Phi_{df}\end{aligned} \tag{11}$$

where $\bar{\mathbf{S}} = \begin{bmatrix} \mathbf{S} \\ \sqrt{\gamma_j}\mathbf{I}_{p\times p} \end{bmatrix}$, $\tilde{\mathbf{S}} = \begin{bmatrix} \mathbf{S} \\ \mathbf{0}_{p\times p} \end{bmatrix}$.

We describe our proposed algorithm for $\mathbf{B}$ given $\mathbf{A}$ step using FISTA in Algorithm 1, and the overall SLCA method in Algorithm 2.

Algorithm 1 FISTA Algorithm for SLCA: ($\mathbf{B}$ given $\mathbf{A}$)

```
1: Input: $\mathbf{b}_j^1$: initial random solution,  $t_0 = t_1 = 1$, $\mu$, $\epsilon$, $\eta_{max}$, $keep = 1$.
2: while $(\eta \le \eta_{max})$ and $(keep == 1)$ do
3:     Step $\eta$: $(\eta \ge 1)$ Compute
4:         $y^\eta = \mathbf{b}_j^\eta + \left(\frac{t_{\eta-1}-1}{t_\eta}\right)\left(\mathbf{b}_j^\eta - \mathbf{b}_j^{\eta-1}\right)$,
5:         $\mathbf{b}_j^{\eta+1} = \tau_{\delta_j\mu}\left(y^\eta - 2\mu\bar{\mathbf{S}}^T\left(\bar{\mathbf{S}}y^\eta - \tilde{\mathbf{S}}\mathbf{a}_j\right)\right)$
6:         $t_{\eta+1} = \frac{1+\sqrt{1+4t_\eta^2}}{2}$
7:     if $\frac{\|\mathbf{b}_j^{\eta+1} - \mathbf{b}_j^\eta\|}{\|\mathbf{b}_j^{\eta+1}\|} \le \epsilon$ then
8:         $keep = 0$
9: Output: $\mathbf{b}_j^{\eta+1}$
```

Algorithm 2 Regression Framework of SLCA

```
1: Input: $\mathbf{A}^1 = \left[\mathbf{a}_1^1, \ldots, \mathbf{a}_k^1\right]$: Ordinary principal components, $\epsilon$, $Itr_{max}$, $keep = 1$.
2: while $(\ell \le Itr_{max})$ and $(keep == 1)$ do
3:     Step $\ell$: $(\ell \ge 1)$ Compute
4:         $\mathbf{b}_j^\ell = FISTA(\mathbf{a}_j^\ell, \mathbf{b}_j^{\ell-1})$, $j = 1, \ldots, k$
5:         $\mathbf{B}^\ell = \left[\mathbf{b}_1^\ell, \ldots, \mathbf{b}_k^\ell\right]$
6:         $\mathbf{b}_j^\ell = \frac{\mathbf{b}_j^\ell}{\|\mathbf{b}_j^\ell\|}$, $j = 1, \ldots, k$
7:         Take SVD of $\mathbf{S}^T\mathbf{S}\mathbf{B}^\ell = \mathbf{U}\mathbf{D}\mathbf{V}^T$.
8:         Then $\mathbf{A}^{\ell+1} = \mathbf{U}\mathbf{V}^T$
9:         $\mathbf{a}_j^\ell = \frac{\mathbf{a}_j^\ell}{\|\mathbf{a}_j^\ell\|}$, $j = 1, \ldots, k$
10:    if $\left(\|\mathbf{B}^\ell - \mathbf{B}^{\ell-1}\| \le \epsilon\right)$ then
11:        $keep = 0$
12: Output: $\mathbf{B}$
```

In Algorithm 1, $\tau_\gamma(.)_i$ is a shrinkage operator which is defined for any $\mathbf{q} \in \mathbb{R}^n$ as follows: $\tau_\gamma(\mathbf{q})_i = (|q_i| - \gamma)_+ \operatorname{sgn}(q_i)$. The parameter ϵ is used as a convergence threshold and η_{max} is used as a maximum iteration number for convergence. Moreover, the parameter $\mu = \frac{1}{\psi}$ in which $\psi = 2\lambda_{max}\left(\bar{\mathbf{S}}^T\bar{\mathbf{S}}\right)$ is the Lipschitz constant of the first two terms in the objective function in (9), where $\lambda_{max}(.)$ denotes the maximum eigenvalue of a matrix. Note that, in the Step 5 of the Algorithm 1, we need the following parameters: $\bar{\mathbf{S}}^T\bar{\mathbf{S}} = \mathbf{S}^T\mathbf{S} + \gamma\mathbf{I}_{p\times p}$ and $\bar{\mathbf{S}}^T\tilde{\mathbf{S}} = \mathbf{S}^T\mathbf{S}$. Therefore, for solving the optimization problem, we just need the $\mathbf{S}^T\mathbf{S} = \Phi_{df}$. That is, *there is no need to decompose Φ_{df} in practice: Φ_{df} can be used directly in the optimization.* Note that the ridge penalty factor γ should be chosen as a small positive number and large values of δ_j gives sparser solution. Note that the principal components (PCs) corresponding to *higher eigenvalues* are used for subspace analysis in PCA based approaches. However, in this LCA based design, our interested Laplacian components (LCs) are correspond to the *smaller eigenvalues* of (normalized) graph Laplacian Φ_{df}. In this case we observed that, we need to handle the regression for LC corresponding to eigen value zero ($\mathbf{b}_1$) separately from the other LCs corresponding to eigenvalues greater than zero ($\mathbf{b}_j, j = 2, ..., k$). In particular, $\mathbf{b}_1$ has a much smaller magnitude during the iterations in Algo. 2 comparing to other LCs (note that normalization is performed in Algo.

2, Lines 6 and 9). If we use the same setup as in the 'Sparse PCA' work, the regression tends to converge to the trivial solution for $\mathbf{b}_1$: a zero vector (the zero vector is a solution to $\Phi_{df}\mathbf{b} = \lambda\mathbf{b}, \lambda = 0$, and is sparse), and we always end up with a zero vector for $\mathbf{b}_1$. After investigation, we resolve to use a separate and small threshold for $\mathbf{b}_1$ when sparsifying the LCs. This corresponds to using a separate (and small) lasso regularization parameter for $\mathbf{b}_1$. This modification allows us to regress a non-trivial sparse $\mathbf{b}_1$ (instead of a zero vector) which is much more useful for subspace analysis. Therefore, in the experiment part, we use two sparse penalty factor δ; one for the PC corresponding to the zero eigenvalue and one for the other PCs. Finally, the k sparse LCs are $\{\mathbf{v}_1, ..., \mathbf{v}_k\} = \mathbf{B}$.

In the following sections, we further discuss: i) the difference between previous work and our work, which imposes graph smoothness using eigenvector reconstruction; ii) the benefit to use dual-input feature graph (combined data and prior information) instead of the default sample graph and feature graph (without prior information); iii) the benefit of sparse loading in imposing graph smoothness.

E. Graph Smoothness using Reconstruction of Eigenvectors

Recent improvements of PCA use spectral graph regularization to enforce that the low-rank component is smooth on certain graph (sample or feature) [2], [3]. For example, [3] introduces $tr(\mathbf{L}^T\Phi_f\mathbf{L})$ in the minimization (Model 4 in Table I). Viewing the row vectors of $\mathbf{L}$ as n-dim signals that reside on the feature graph (with p vertices), this spectral graph regularization enforces the signals to change smoothly between the connected vertices.

In this work, instead of using spectral graph regularization to enforce smoothness, we directly impose the graph smoothness using the first k eigenvectors of the graph Laplacian ($k < p$ for p-dim data). Furthermore, we perform regression on the structure of the Laplacian to induce sparse loadings on the vectors.

Specifically, the eigenvectors with small eigenvalues are smooth with respect to the underlying graph, i.e., changes are small between the connected vertices. It is an application of Courant-Fischer Formula for Laplacian [59], [7]. In particular, for a graph Laplacian Φ, the k-th eigenvector is the minimizer of:

$$\min_f f^T\Phi f = \sum_{i,j} w_{i,j}[f(i) - f(j)]^2, \tag{12}$$

with $\|f\|_2 = 1$ and f being orthogonal to the first $k-1$ eigenvectors. Thus the first k eigenvectors have small $\sum_{i,j} w_{i,j}[f(i) - f(j)]^2$ and are graph smooth. Therefore, the first k eigenvectors model a class of signals that are smooth with respect to underlying graph. Reconstruction using the first k eigenvectors imposes graph smoothness constraint.

F. Feature Graph with Prior information

Unlike sample graph which is used in most of the prior works, we use feature graph with prior information, i.e., *dual-input feature graph*, as shown in Equation (2). We perform an experiment to quantify the improvement on our Abilene Network dataset. In particular, our dual-input feature graph uses two inputs - data and prior information, whereas the existing works use data matrix only. Specifically, the prior information is the known graph topology from which the data samples (i.e. graph signals) are generated. Note that, the proposed feature graph is not same as the standard feature graph used by Shahid *et al.* [3]. Therefore, we use the term '*source graph*' for denoting our *dual-input feature graph* which describes the correlation between data sources, i.e., features in $\mathbf{X}$.

From the Abilene Network dataset, we consider three graph Laplacian Φ_s, Φ_f, and Φ_{df} corresponding to sample, feature, and source graphs, respectively. We compute the graphs of Φ_s and Φ_f following the method in [3]: we connect each sample/feature to its K nearest neighbors and use a Gaussian function to compute the weight for connected vertices. We discuss the construction of source graph in the later section and compute Φ_{df} using Equation 3. We measure the smoothness of the dataset on these graphs.

To measure the global smoothness of signals on the intrinsic graphs, we compute the graph Laplacian quadratic value [7], [59]. For proper comparison, we propose to normalize them. In particular, given the input dataset $\mathbf{X}_{p\times n}$, for the sample graph with $\Phi_{n\times n}$, we compute:

$$quad(\mathbf{X}, \Phi) = \frac{tr(\mathbf{X}\Phi\mathbf{X}^T)}{p \times (\#W)} \tag{13}$$

where $\#W$ denotes the number of non-zero weight in $\mathbf{W}$. For the feature and source graphs with $\Phi_{p\times p}$, we compute:

$$quad(\mathbf{X}, \Phi) = \frac{tr(\mathbf{X}^T\Phi\mathbf{X})}{n \times (\#W)} \tag{14}$$

Note that it is important to normalize the quadratic value for proper comparison. In particular, for sample graph, $tr(\mathbf{X}\Phi\mathbf{X}^T) = \sum w_{i,j}\|\mathbf{x}_i - \mathbf{x}_j\|^2$ which is summation of $p\times(\#W)$ values. For feature and source graphs, $tr(\mathbf{X}^T\Phi\mathbf{X}) = \sum w_{i,j}\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|^2$ which is summation of $n \times (\#W)$ values. Here $\tilde{\mathbf{x}}$ and $\mathbf{x}$ represent the row vector and the column vector of $\mathbf{X} \in \mathbb{R}^{p\times n}$, respectively.

With the above set up, we observed the following values: $quad(\mathbf{X}, \Phi_s) = 0.0414$, $quad(\mathbf{X}, \Phi_f) = 0.1176$, $quad(\mathbf{X}, \Phi_{df}) = 0.0335$ for the Abilene Network dataset (more information about the dataset is provided in the experimental setup section). The results show that the input signals are smoother on the proposed source-graph, which motivates us to use source-graph in our proposed method. Note that the addition of prior information could improve the performance of a model as the dataset is smoother on the source-graph compared to the default feature graph, based on their Laplacian quadratic values. Further, there is no previous study in comparing these graph models quantitatively for a given problem or dataset. In our study, we use the graph smoothness concept, and specifically the graph Laplacian quadratic value to decide the model for the underlying graph (sample, feature or source graph model), before the signal analysis.

G. Usefulness of Sparse Loading

As discussed, the first k eigenvectors of the graph Laplacian model a class of signals that are smooth w.r.t. the underlying graph. In this section, we demonstrate the class of signals modeled by the first k sparse LCs is smoother than that by the normal LCs. Specifically, we compute the "zero crossings" of first k components as signals on the graph [7]. The count of zero crossing for a graph signal is defined as the number of edges connecting a vertex with a positive signal to another one with a negative signal. For the Abilene dataset, we computed the zero crossing for all the components generated by the LCA and SLCA methods. Figure 2 represents the cumulative number of zero crossing with increasing component index. It is observed that, the sparse components are much smoother on underlying graph compared to the normal components.

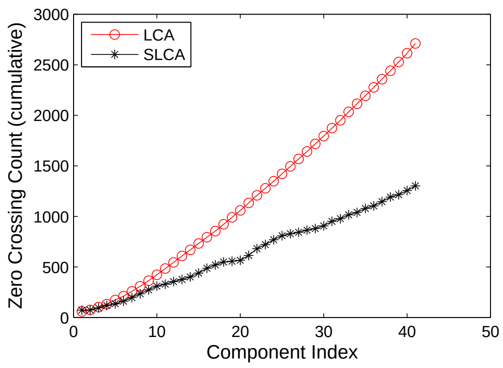


Fig. 2: Cumulative values of zero crossing as the number of component increases.

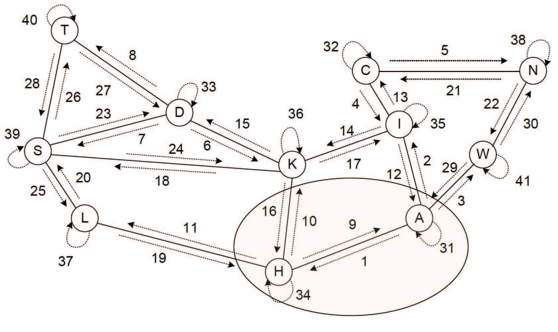
IV. Graph Construction

In this work, we use dual-input feature graph i.e. source-graph for graph embedding. In addition to dataset, we incorporate prior information about the sources of graph signals. In the following, we describe the graph construction method for the Abilene network dataset which is used extensively for network anomaly detection experiments, e.g., [8], [60] and [61].

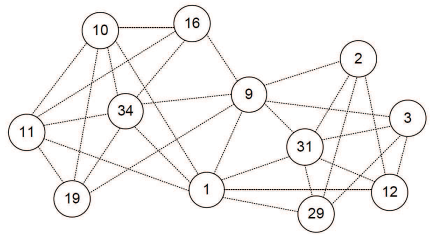
A. Graph Structure Representation of Prior Information

The proposed model leverages additional information available in the form of graph structure. For the Abilene dataset, we have topology information of the network. We consider a graph isomorphic to the Abilene network representing the network routers as vertices and the communication links as edges in the graph, respectively. Figure 3(a) shows the *network graph* G of the Abilene network. In this work with Abilene data, we need a graph describing the relationship among the links in G as the inputs for the SLCA method are the network's link-level measurements. Therefore, we compute another graph called *network link graph* from the directed *network graph* using the method described as follows.

For a directed and connected network graph $G = (\mathcal{V}, \mathcal{E})$ with self-loop, the corresponding link graph $G_L = (\mathcal{V}', \mathcal{E}')$ is defined such that $|\mathcal{V}'| = |\mathcal{E}|$, with the value at a vertex in $\mathcal{V}'$ representing the volume measurement of a network link in $\mathcal{E}$. Moreover, there exists an undirected edge in G_L for each pair of links in G that shares a common end point which makes it possible to flow data from one link to other; i.e.,



(a) Network graph G of Abilene Network.



(b) Link graph G_L for the highlighted part.

Fig. 3: Network graph G of the Abilene Network with link id (numbers written beside the links) and corresponding data direction, and Link graph G_L corresponding to the marked (by highlighted eclipse) portion of G only.

$\{(i,j) \in \mathcal{E}'\} \leftrightarrow \{\exists v \in \mathcal{V} | [v_s^i = v_e^j = v] \vee [v_s^j = v_e^i = v]\}$ where $i = (v_s^i, v_e^i) \in \mathcal{E}$, and $j = (v_s^j, v_e^j) \in \mathcal{E}$, $v_s^i \in \mathcal{V}$, $v_e^i \in \mathcal{V}$, $v_s^j \in \mathcal{V}$, $v_e^j \in \mathcal{V}$. An example of corresponding link graph for the specific portion marked by a highlighted eclipse in Figure 3(a) is shown in Figure 3(b). We have shown a portion of the link graph as the size of the full link graph is large. Note that, the full step described here corresponds to function $\mathcal{F}_1$ in Equation 1.

B. Dual-Input Feature Graph

In this paper, we construct the dual-input feature graph (source-graph $\mathcal{G}$) using two characteristic parameters.

- First one is related to the *Pearson correlation coefficient* (ρ) between the $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_j$, where $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_j$ are the row vectors of $\mathbf{X} \in \mathbb{R}^{p \times n}$.
- The second characteristic parameter is related to *minimum distance* (h) between two vertices in the graph G_L. For the network dataset, we consider 'hop count' as distance measure in G_L. For computing the shortest hop count from one vertex to other in G_L, we use the Bellman-Ford shortest path algorithm, and the weight of each edge in graph G_L is unity. It actually preserves the global structure of a network indirectly.

We compute the source-graph $\mathcal{G}$ as follows: (i) The set of vertices in source-graph is same as in G_L. (ii) Source-graph has all possible set of edges except self loop. (iii) The weight matrix $\mathbf{W}$ of the source-graph is computed following the equation of Gaussian kernel on both the parameters - data and hop count (i.e. prior information). With respect to our objective of finding anomaly in dataset, the correlation coefficient carries proportional relationship between the vertices whereas the distance metric carries reciprocal relationship. This is because an increase in minimum distance decreases the possibility of exposing similar behavior by the vertices in G_L. Therefore, the weight matrix of the source-graph is computed as follows:

$$[w_{i,j}]_{i\neq j} = \exp\Bigg(-\frac{(1-[\|\rho(i,j)\|_1]_+)^2}{\Delta_c^2}\Bigg) \times \exp\Bigg(-\frac{([\widehat{h}(i,j)]_+)^2}{\Delta_h^2}\Bigg) \tag{15}$$

where $[\|\rho(i,j)\|_1]_+$ equals $\|\rho(i,j)\|_1$ if $\|\rho(i,j)\|_1 \geq \theta_c$, and equals unity otherwise; $[\widehat{h}(i,j)]_+$ equals $\widehat{h}(i,j)$ if $h(i,j) \leq \theta_h$, and equals zero otherwise; if both the conditions are violated then we set the value of $w_{i,j}$ equals 0; θ_c and θ_h are constant thresholds; Δ_c and Δ_h are the parameters determining the rate of exponential decay; and $\widehat{h}$ represents the normalized h. We perform normalization by dividing with the maximum value of $h(i,j)$ in the distance matrix computed for all set of vertices in the graph G_L. We constrain the diagonal elements of $\mathbf{W}$ equals zero as the source-graph does not have self loop. Further, we compute the diagonal degree matrix $\mathbf{D}$ in which $d_{i,i} = \sum_{j\in\mathcal{V}} w(i,j)$ indicates the degree of i^{th} vertex in $\mathcal{V}$. Note that this step corresponds to function $\mathcal{F}_2$ in Equation (2).

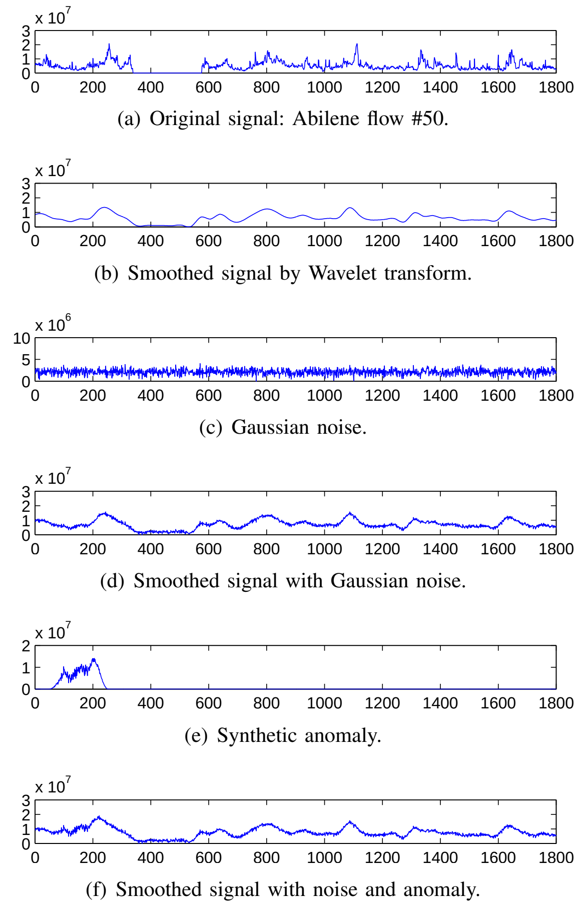
V. Experimental Setup

We use the proposed method to solve the problem of unsupervised detection of traffic anomalies in computer networks, for which PCA and other extensions have been used so far. In particular, we consider *volume anomaly* in network traffic.

1) Datasets: We evaluate the performance of the proposed scheme with sate-of-the-art schemes for real network datasets and synthetic datasets.

A) Real Dataset Abilene: We consider the real Abilene network dataset which is a multivariate timeseries of OD flow traffic collected from the Abilene Internet2 backbone network. Abilene network has 11 point-of-presence (PoP) and spans across the universities in United States. The traffic is measured in #bytes from every PoP (i.e. router). The sampled flow data of Abilene was collected for a period of three weeks. Sampling was done randomly to capture 1% of all packets entering every router. Using the Juniper's Traffic Sampling [62], the sampled packets were aggregated for every minute by comparing 5-tuple information in IP-flow level. The 5-tuple was defined as source and destination IP addresses, two port numbers, and used protocol type. The #bytes in each sampled IP-flow were recorded for generating the data set. Finally, for avoiding synchronization issue, that could have arisen in the data collection procedure, the data was aggregated in 5 minute bin. Therefore, we have 2016 (= 7 days x 24 hours x 60 min / 5 min) samples in every week dataset. The traffic exchanged between OD pairs actually enters the network at a given ingress PoP and exits at another PoP. Those ingress and egress PoPs are identified by inspecting the router configuration files, and the BGP and ISIS routing tables [63], respectively. In brief, the collected dataset has 121 OD flows and 2016 samples. However, the *flow-level* data is not the input to our algorithm, but used for validation purpose. Our technique operates on *link-level* data. Therefore, we convert the dataset from flow-level to link-level. Figure 3(a) shows the directional links in the Abilene network. Let the link-flow matrix is denoted by $\mathbf{A}$ of size (# links)$\times$(# flows), and the sampled OD flow dataset by $\mathbf{C}$ of size (# flows)$\times$(# samples). Then, the matrix of traffic counts on links is computed as $\mathbf{X} = \mathbf{AC}$. The Abilene network has 41 directed links, and, thus, the input dataset is $\mathbf{X} \in \mathbb{R}^{41\times 2016}$.

B) Synthetic Dataset Abilene: We also have generated synthetic data using the Abilene network dataset and topology information. In this approach, we inject synthetic anomaly into the real traffic after performing some pre-requisite operations on the real datasets. With synthetic dataset, we are able to test our methods for various anomalous time-stamps and anomaly patterns to mimic different types of volume anomaly such as DDoS attack, alpha event and outrage. We use volume anomaly spreading on 5% of total samples. We generate the anomalous traffic by using a multiplicative factor β which is multiplied with original signal for the selected anomalous time duration. Generally the values of β remains in between 0 and 2.5 to mimic a variety of different volume anomaly [61]. In this work, we have considered $\beta = 2$. Further, the initial rise and the fall of traffic at the ending of volume anomaly is simulated by an exponential ramp shape. We randomly selected the flow #50 for injecting synthetic anomaly. All the remaining flows carry normal traffic. Thereafter, we follow the same approach to convert the flow-level traffic to link-level traffic as described for real dataset of Abilene network.



(a) Original signal: Abilene flow #50.

(b) Smoothed signal by Wavelet transform.

(c) Gaussian noise.

(d) Smoothed signal with Gaussian noise.

(e) Synthetic anomaly.

(f) Smoothed signal with noise and anomaly.

Fig. 4: Synthetic dataset: Anomaly injection procedure in Abilene (50th flow). Note that the vertical axes have different scales.

In this juncture, we explain the pre-requisite operations which we perform on the real dataset before injecting synthetic anomaly into it. An example of the overall procedure is explained in Figure 4 in which we consider the flow #50 for injecting anomaly. Figure 4(a) shows the original OD flow #50, and the Figure 4(b) shows the extracted long-term statistical *trend* of the signal by *smoothing* the original OD flow. Similar to previous works (e.g. [61], [60]), we perform signal smoothing by Wavelet approximation of the signal. We add the Gaussian noise with SNR= 20dB as shown in Figure 4(c) onto the de-noised smoothed signal. The resultant signal is shown in Figure 4(d). Finally, we inject anomalous signal, shown in Figure 4(e), to generate the anomaly-injected synthetic signal as shown in Figure 4(f). We follow the same process except the anomaly injection step for the rest of the flows of Abilene network datasets. Thus, we get three synthetic datasets corresponding to three real datasets of Abilene networks.

C) Real Dataset RSS: We use another real dataset collected using wireless sensor network (WSN) randomly deployed inside and outside a laboratory at the University of Michigan [64]. In total 14 sensor nodes were deployed, and they communicated with each other asynchronously by broadcasting RF signal. Every receiver measured the received signal strength (RSS) which has been recorded. During this experiment, the university students walk into and out of the lab which is treated as anomaly. The timestamps of occurring such anomaly were recorded using an web camera by which the gound truth was determined in the dataset. The RSS measurements were collected over a 30-minute period, and each sample was acquired in every 0.5 sec. The dataset has been synchronised using interpolations and the temperature drifts has been removed. In total, there are 3127 measurements. As the experiment uses mesh topology in communication, we have $14 \times 13 = 182$ dimensional data samples. However, we set a minimum threshold value for capturing the RSS, and, thus, we have less number of edges than the fully connected network graph. Finally, we take the mean of measured RSS values by each node at every timestamp for generating the desired dataset $\mathbf{X} \in \mathbb{R}^{14\times 3127}$ where the dimension of each sample is 14.

D) Real Dataset SWaT: We use one more real dataset systematically generated from the Secure Water Treatment (SWaT) Testbed at iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design [65]. The data was collected from the testbed for 11 days of continuous operation – 7 days with normal operation and 4 days with attack scenarios. The dataset consists of all the values obtained from all the 51 sensors and actuators available in SWaT. During the data collection process, the data samples were labeled according to normal and abnormal behaviours. In this work, we have only consider one type of attack scenario out of 36 types of attacks performed in 4 days, and we don't have any graph structure relationship among the sensors and actuators. Therefore, the collected dataset was $\mathbf{X} \in \mathbb{R}^{51\times 10000}$. After doing some pre-processing, we get the desired mean removed SWaT dataset $\mathbf{X} \in \mathbb{R}^{37\times 10000}$ where the dimension of each sample is 37.

2) Data Preprocessing: In this paper, we pre-process all the three real datasets to zero mean and unit standard deviation along the features. We perform the same after introducing synthetic anomaly and Gaussian noise for synthetic anomaly datasets.

3) Anomaly Detection Metric: For evaluating the anomaly detection performance of the proposed model as well as the the benchmark models, similar to previous works (e.g. [8], [27]), we rely on the separation of link traffic into normal and abnormal components. This is done based on the concept of subspace-based detection [8]. The basic idea of subspace-based detection is to determine the normal and abnormal subspaces followed by the projection of link traffic onto the subspaces. After the projection of data onto the subspaces, we compute *anomaly detection score*. We compute the orthogonal projection matrices $\mathcal{O}_n$ and $\mathcal{O}_a$ corresponding to normal and abnormal subspaces decided by the reduced dimension k. Thereafter, we measure the *anomaly detection score* for each data instance i as follows: $\zeta_i^{det} = \|\mathcal{O}_n \mathbf{z}_i\|_2^2 - \|\mathcal{O}_a \mathbf{z}_i\|_2^2$, where $\mathbf{z}_i = \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|_2}$ is the i^{th} column of normalized data matrix $\mathbf{Z} \in \mathbb{R}^{p\times n}$ computed from $\mathbf{X}$. In general, the anomaly score greater than a threshold indicates the occurrence of network anomaly at that instance. In order to show the overall performance of each detection method, different thresholds are chosen to generate the receiver operating characteristic (ROC) curve and the area under the curve (AUC) score. The ROC curve plots the true positive rate against the false positive rate at various discrimination thresholds.

4) Parameter Selection for SLCA: In Table III, we mention the range of parameter values used for searching the optimal set-up in each model. We followed the grid search methodology for finding the optimal grid point of the data grid as shown in the Table III. For example, we use the data grid $\{k,\theta_c,\theta_h,\delta,\gamma\}$ for SLCA. We search optimal set up for each model-dataset pair. For each model-dataset pair and their corresponding data grid, we compute average AUC score at every grid point by following 10-fold cross validation procedure. Finally, we select the grid point which produces maximum AUC score as optimal set-up for that model-dataset pair.

VI. Validation

In this section, we evaluate the performance of the proposed model and compare with the other benchmark methods for the Abilene dataset as well as three synthetic datasets.

A. Using Real Dataset: Abilene

We consider the real datasets collected from Abilene network, which is the standard dataset for the volume anomaly detection problem [8]. We evaluate the anomaly detection performance for all the benchmark models and the proposed models as mentioned in Table I. Specifically, we compare with the following standard and state-of-art subspace analysis methods: PCA, Locality Preserving Projections(LPP) [15], Robust PCA (RPCA) [4], Robust PCA on Graph (RPCAG) [2], Fast Robust PCA on Graph (FRPCAG) [3], and Graph Laplacian PCA (GLPCA) [1]. Note that RPCAG and FRPCAG are state-of-the-art improvements of PCA using spectral graph regularization. Table IV shows a comparative analysis on the average value of AUC scores and their corresponding standard deviation. We

TABLE III: Range of parameter values for each of the models considered in this work, and the process of tuning those parameters. Note that for LCA and SLCA, we have also experimented using the default feature graph computed the same as FRPCAG [3], and in such case the graph parameters (θ_c, θ_h, Δ_c, Δ_h) are not needed.

Model	Parameters		Parameter Range
	Graph	Optimization	
PCA		k	$k \in \{2, 3, \ldots, min(n, p)\}$
LPP [15]	knn		$knn = 10$
RPCA [4]		δ	$\delta \in \{\frac{2^{-4}}{\sqrt{max(n,p)}}, \frac{2^{-3}}{\sqrt{max(n,p)}}, \ldots, \frac{2^{10}}{\sqrt{max(n,p)}}\}$; $\gamma_1, \gamma_2 \in \{2^{-4}, 2^{-2}, \ldots, 2^{10}\}$; $\sigma_1, \sigma_2 = 1$; $knn = 10$
RPCAG [2]	knn,σ_1	δ, γ_1	
FRPCAG [3]	knn,σ_1,σ_2	γ_1,γ_2	
GLPCA [1]	knn	k, γ_1	$\gamma_1 \Rightarrow \beta$ using transformation, $\beta \in \{0.1, 0.2, \ldots, 0.9\}$, $k \in \{2, 3, \ldots, min(n, p)\}$, $knn = 10$
LCA (this work)	θ_c, θ_h, Δ_c, Δ_h	γ, k	$\Delta_c = \Delta_h = 1$; $\theta_h \in \{1 : 1 : 5\}$; $\theta_c \in \{0.1 : 0.1 : 1\}$; $k \in \{2, 3, \ldots, min(n, p)\}$; $\gamma \in \{0.002 : 0.002 : 1\}$; $\delta_1 \in (1e - (10 : 1 : 18))$; $\delta_j \in \{0.01 : 0.01 : 1\}$ for all j $\neq 1$
SLCA (this work)	θ_c, θ_h, Δ_c, Δ_h	δ, γ, k	

TABLE IV: Comparison of average AUC score, Standard Deviation (S.D.) in 10-fold AUC, reduced dimension k (i.e., size of the normal subspace), and optimal model parameters for the different models using 10-fold cross validation procedure on Abilene dataset of three weeks. Note that, for each models, k is chosen to achieve the optimal AUC.

(a) Using Abilene dataset of 1^{st} week.

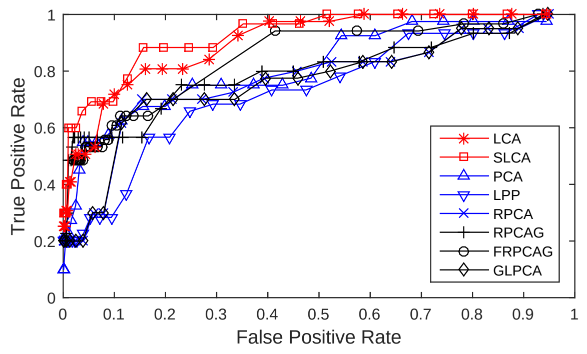
Model	Avg. AUC	S.D. AUC	k	Optimal parameter values
PCA	76.19	0.169	4	$k = 4$
LPP	65.23	0.246	5	$k = 5$
RPCA	72.35	0.221	10	$\delta = 0.45$
RPCAG	73.51	0.245	6	$\delta = 0.0445, \gamma = 2^7$
FRPCAG	76.54	0.145	20	$\gamma_1 = 0.0625, \gamma_2 = 0.5$
GLPCA	71.68	0.218	15	$k = 15, \beta = 0.5$
LCA	**85.04**	0.107	27	$k = 27, \theta_c = 0.2, \theta_h = 1, \gamma = 0.04$
SLCA	**87.10**	0.103	18	$k = 18, \theta_c = 0.3, \theta_h = 3, \gamma = 0.02, \delta_j = 0.01, \delta_1 = 1e-17$

(b) Using Abilene dataset of 2^{nd} week.

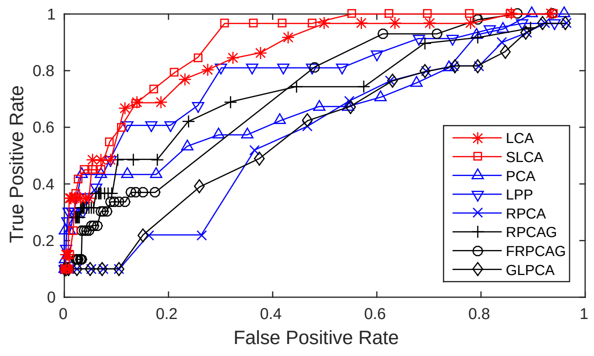
Model	Avg. AUC	S.D. AUC	k	Optimal parameter values
PCA	64.54	0.285	3	$k = 3$
LPP	72.71	0.195	6	$k = 6$
RPCA	54.26	0.270	9	$\delta = 0.7$
RPCAG	63.10	0.218	35	$\delta = 0.0891, \gamma = 2^5$
FRPCAG	65.49	0.184	8	$\gamma_1 = 4, \gamma_2 = 0.5$
GLPCA	54.94	0.277	15	$k = 15, \beta = 0.9$
LCA	**80.06**	0.158	22	$k = 22, \theta_c = 0.2, \theta_h = 2, \gamma = 0.014$
SLCA	**82.64**	0.113	24	$k = 24, \theta_c = 0.2, \theta_h = 2, \gamma = 0.004, \delta_j = 0.01, \delta_1 = 1e-17$

(c) Using Abilene dataset of 3^{rd} week.

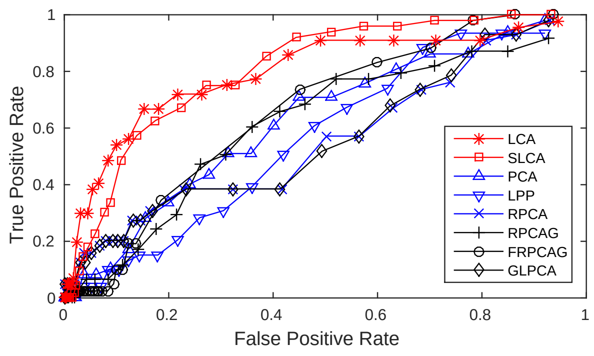
Model	Avg. AUC	S.D. AUC	k	Optimal parameter values
PCA	55.08	0.172	5	$k = 5$
LPP	49.55	0.136	40	$k = 40$
RPCA	49.18	0.224	13	$\delta = 0.2$
RPCAG	54.68	0.176	8	$\delta = 0.0445, \gamma = 2^7$
FRPCAG	59.69	0.107	19	$\gamma_1 = 0.125, \gamma_2 = 1$
GLPCA	49.20	0.225	20	$k = 24, \beta = 0.5$
LCA	**73.84**	0.158	21	$k = 21, \theta_c = 0.2, \theta_h = 2, \gamma = 0.012$
SLCA	**74.34**	0.124	15	$k = 15, \theta_c = 0.3, \theta_h = 2, \gamma = 0.014, \delta_j = 0.02, \delta_1 = 1e-17$



(a) Using the dataset of 1^{st} week.



(b) Using the dataset of 2^{nd} week.



(c) Using the dataset of 3^{rd} week.

Fig. 5: Comparison of ROC curves under the different models using 10-fold cross validation on Abilene datasets of three weeks

TABLE V: AUC score comparison using the default feature graph and our dual-input feature graph i.e. source-graph

Dataset	Graph Type	LCA	SLCA
1st week	Standard Feature Graph [3]	83.98	86.35
	Source-Graph	**85.04**	**87.10**
2nd week	Standard Feature Graph [3]	79.77	82.50
	Source-Graph	**80.06**	**82.64**
3rd week	Standard Feature Graph [3]	68.38	70.58
	Source-Graph	**73.84**	**74.34**

capture the average AUC score using 10-fold cross validation process. We run the experiment for three sets of data captured in Abilene network for three weeks. The results for the three weeks datasets are shown in Tables IV(a), IV(b) and IV(c), respectively. We also show the reduced dimension size corresponding to each model under each dataset. The optimal parameter values corresponding to the presented AUC score are shown in the last column of each table. We observe that both the versions (LCA and SLCA) of the proposed model outperforms all the existing benchmark models for the real datasets of Abilene network. It is pertinent to mention that the performance of SLCA is better than LCA, as the sparse components are smoother than their non-sparse counterparts. Figures 5(a), 5(b), and 5(c) show the comparison on ROC curve for all the three real datasets of Abilene network, respectively. The figures once again demonstrate better accuracy on true positive rate with respect to a certain false positive rate.

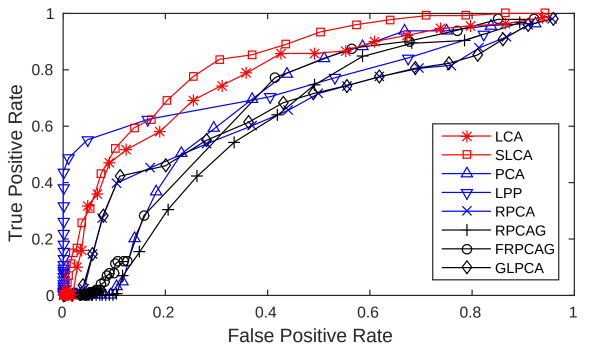
To understand the effect of the proposed source-graph, we compare the AUC score computed using the proposed models under the standard feature graph [3] and our proposed source-graph. The result is shown in Table V. Once again the result validates that the traffic measurements are smoother with respect to the source graph compared to the default feature graph. Importantly, when comparing Table IV and V, we observe that LCA / SLCA can outperform previous work when using the standard feature graph, demonstrating that our proposed LCA and SLCA are superior in subspace analysis. The use of source graph leads to further improvements.

TABLE VI: Comparison of average AUC scores under different models for the synthetic Abilene datasets using 10-fold cross validation experiment.

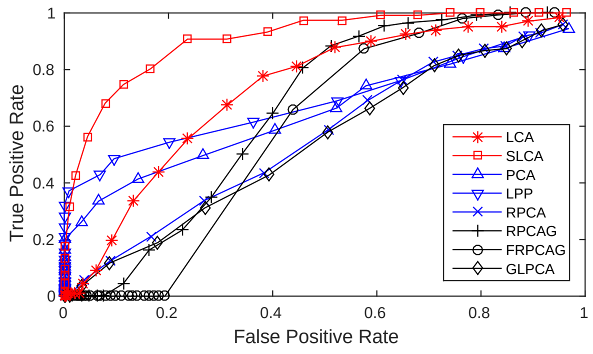
Model	1st week	2nd week	3rd week
PCA	61.27	61.60	66.62
LPP	58.89	58.94	57.50
RPCA	61.43	51.87	50.50
RPCAG	52.52	58.29	60.06
FRPCAG	60.33	53.55	60.31
GLPCA	61.99	50.27	50.23
LCA	**71.80**	**67.69**	**70.16**
SLCA	**77.48**	**87.16**	**73.57**

B. Using Synthetic Dataset: Abilene

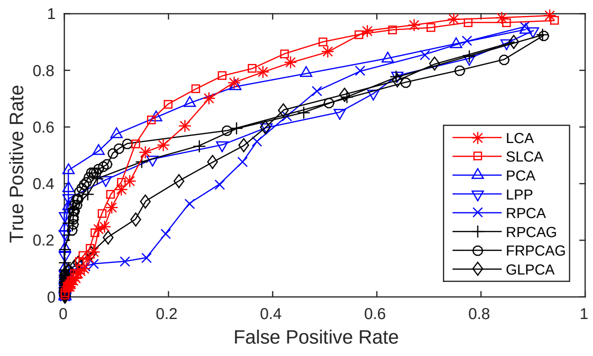
We also perform comparison using the synthetic datasets. As we have created the synthetic datasets from the real Abilene network datasets, we also use the Abilene network topology in this experiment. In the experiments, we varied the detection threshold to generate ROC curve for all the models. The results are plotted in Figure 6. We observe that the proposed models outperforms the benchmark methods for all the three synthetic



(a) Using the dataset of 1st week.



(b) Using the dataset of 2nd week.



(c) Using the dataset of 3rd week.

Fig. 6: Comparison of ROC curves under the different models using 10-fold cross validation on synthetic datasets of three weeks

datasets. To understand the improvement, we computed AUC score using the same parameters for individual methods as shown in Table III. The summarized result of AUC score is presented in Table VI, which suggests our proposed LCA and sparse LCA outperform other methods.

C. Using Real Dataset: RSS,SWaT

We performed the above experiments using two more real datasets - RSS and SWaT. In the experiments, we varied the detection threshold to generate ROC curve for all the models. The received results are plotted in Figure 7. It is important to mention that, we used the present network graph as prior information in the experiments using RSS dataset. Therefore, we were able to construct the source graph corresponding to

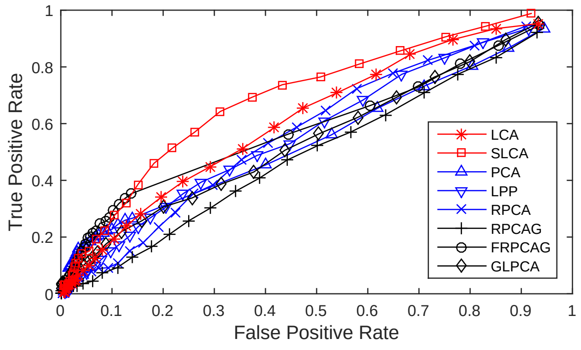
TABLE VII: Comparison of average AUC score, Standard Deviation (S.D.) in 10-fold AUC, reduced dimension k (i.e., size of the normal subspace), and optimal model parameters for the different models using 10-fold cross validation procedure on RSS and SWaT datasets. Note that, for each models, k is chosen to achieve the optimal AUC.

(a) Using RSS Dataset.

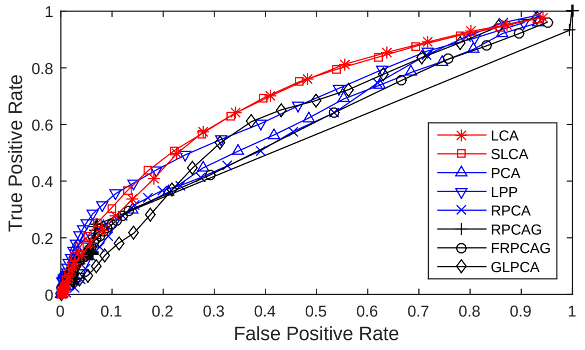
Model	Avg. AUC	S.D. AUC	k	Optimal parameter values
PCA	49.47	0.052	1	$k = 1$
LPP	49.12	0.075	12	$k = 12$
RPCA	49.07	0.048	14	$\delta = 0.305$
RPCAG	43.43	0.039	4	$\delta = 0.0358, \gamma = 8$
FRPCAG	53.05	0.062	14	$\gamma_1 = 1, \gamma_2 = 0.5$
GLPCA	48.83	0.032	35	$k = 3, \beta = 0.6$
LCA	**55.76**	0.054	6	$k = 6, \theta_c = 0.4, \theta_h = 2, \gamma = 0.004$
SLCA	**61.98**	0.043	10	$k = 10, \theta_c = 0.4, \theta_h = 1, \gamma = 0.04, \delta_j = 0.03, \delta_1 = 1e-17$

(b) Using SWaT Dataset.

Model	Avg. AUC	S.D. AUC	k	Optimal parameter values
PCA	54.76	0.019	3	$k = 3$
LPP	57.01	0.059	8	$k = 8$
RPCA	54.31	0.028	14	$\delta = 0.105$
RPCAG	55.63	0.019	26	$\delta = 0.04, \gamma = 0.5$
FRPCAG	54.66	0.013	7	$\gamma_1 = 1, \gamma_2 = 2$
GLPCA	56.02	0.023	3	$k = 3, \beta = 0.5$
LCA	**63.39**	0.017	5	$k = 5, \gamma = 0.032$
SLCA	**62.97**	0.026	8	$k = 8, \gamma = 0.24, \delta_j = 0.012, \delta_1 = 1e-17$



(a) Using the RSS Dataset.



(b) Using the SWaT Dataset.

Fig. 7: Comparison of ROC curves under the different models using 10-fold cross validation on RSS and SWaT datasets

RSS dataset. However, such prior information is not present for the SWaT dataset, and, thus, we used standard feature graph [3] for constructing the graph Laplacian and applying it in the proposed model. We observe that the proposed models outperforms the benchmark methods for both the datasets. We also computed the optimal AUC score under individual methods. The optimal results are shown in Table VII which once again supports our claim the LCA and SLCA outperform other methods irrespective of prior information graph.

VII. Conclusion

Different from previous works that either uses spectral graph regularization or uses first k eigenvectors of sample graph Laplacian, our work uses the first k smooth eigenvectors of the normalized dual-input feature graph Laplacian to impose graph smoothness on the low-rank data matrix. Moreover, we propose to use a regression-based optimization framework to compute these eigenvectors (Laplacian components, LCs). The framework allows us to add the lasso penalty and achieve LCs with sparse loadings. Furthermore, we exploit the inclusion of prior information in computing the LCs within the framework. Experiment results suggest that the proposed method is superior in identifying the essential low-dimensional subspace from real/synthetic Internet traffic, compared to other state-of-the-art. The regression framework is flexible in incorporating other regularization of LCs. Future work investigates the improvements of these variants and application of the framework for other data with hidden graph structure: brain signals [66], transportation network data.

References

[1] B. Jiang, C. Ding, B. Luo, and J. Tang, "Graph-Laplacian PCA: Closed-form solution and robustness," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3490–34 968.

[2] N. Shahid, V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Robust principal component analysis on graphs," in *Proceedings of International Conference on Computer Vision*, Santiago, Chile, 2015, pp. 2812–2820.

[3] N. Shahid, N. Perraudin, V. Kalofolias, G. Puy, and P. Vandergheynst, "Fast robust PCA on graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 740–756, 2016.

[4] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM*, vol. 58, no. 3, pp. 11:1–11:37, May 2011.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Survey*, vol. 41, no. 3, pp. 15:1–15:58, 2009.

[6] H. Huang, H. Al-Azzawi, and H. Brani, "Network traffic anomaly detection," New Mexico State University, Las Cruces, USA, Tech. Rep., 2014.

[7] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, pp. 83–98, May 2013.

[8] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *Proceedings of ACM SIGCOMM*, 2004, pp. 219–230.

[9] K. Nyalkalkar, S. Sinha, M. Bailey, and F. Jahanian, "A comparative study of two network-based anomaly detection methods," in *Proceedings of IEEE INFOCOM*, 2011, pp. 176–180.

[10] Y.-J. Lee, Y.-R. Yeh, and Y.-C. F. Wang, "Anomaly detection via online oversampling principal component analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1460–1470, July 2013.

[11] F. Palmieri, U. Fiore, and A. Castiglione, "A distributed approach to network anomaly detection based on independent component analysis," *Concurrency and Computation: Practice & Experience*, vol. 25, no. 5, pp. 1113–1129, April 2014.

[12] Y. Jiang, C. Zeng, J. Xu, and T. Li, "Real time contextual collective anomaly detection over multiple data streams," in *Proceedings of ODD*, NY, USA, August 2014.

[13] T. Huang, H. Sethu, and N. Kandasamy, "A fast algorithm for detecting anomalous changes in network traffic," in *Proceedings of International Conference on Network and Service Management*, 2015.

[14] S. Gajjar, M. Kulahci, and A. Palazoglu, "Use of sparse principal component analysis (SPCA) for fault detection," in 11^{th} *IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems*, NTNU, Trondheim, Norway, June 2016, pp. 693–698.

[15] X. He and P. Niyogi, "Locality preserving projections," in *Proceedings of NIPS*, 2003, pp. 153–160.

[16] C. Li and H. Li, "Network-constrained regularization and variable selection for analysis of genomic data," *Bioinformatics*, vol. 24, no. 9, pp. 1175–1182, 2008.

[17] D. Cai and J. Han, "Spectral regression: a regression framework for efficient regularized subspace learning," Ph.D. dissertation, University of Illinois at Urbana-Champaign Champaign, IL, USA, 2009, ISBN: 978-1-109-22296-8.

[18] S. Yang, L. Yuan, Y.-C. Lai, X. Shen, P. Wonka, and J. Ye, "Feature grouping and selection over an undirected graph," in *Proceedings of KDD*, Beijing, China, 2012.

[19] Z. Zhang and K. Zhao, "Low-rank matrix approximation with manifold regularization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1717–1729, 2013.

[20] T. Jin, J. Yu, J. You, K. Zeng, C. Li, and Z. Yu, "Low-rank matrix factorization with multiple hypergraph regularizers," *Pattern Recognition*, vol. 48, no. 3, pp. 1011–1022, March 2015.

[21] L. Tao, H. H. S. Ip, Y. Wang, and X. Shu, "Low rank approximation with sparse integration of multiple manifolds for data representation," *Applied Intelligence*, vol. 42, pp. 430–446, 2015.

[22] N. Shahid, N. Perraudin, G. Puy, and P. Vandergheynst, "Compressive PCA for Low-Rank Matrices on Graphs," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 4, pp. 695–710, 2017.

[23] B. Jiang, C. Ding, and B. Luo, "Robust data representation using locally linear embedding guided PCA," *Neurocomputing*, vol. 275, pp. 523–532, 2018.

[24] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[25] A. Sandryhaila and J. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[26] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs: Frequency analysis." *IEEE Trans. Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

[27] H. E. Egilmez and A. Ortega, "Spectral anomaly detection using graph-based filtering for wireless sensor networks," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 1085–1089.

[28] L. Rui, H. Nejati, and N.-M. Cheung, "Dimensionality reduction of brain imaging data using graph signal processing," in *Proc. of IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 1329–1333.

[29] R. Liu, H. Nejati, S. H. Safavi, and N.-M. Cheung, "Simultaneous low-rank component and graph estimation for high-dimensional graph signals: Application to brain imaging," in *Proc. of ICASSP*, 2017, pp. 4134–4138.

[30] M. Khatua, S. H. Safavi, and N. Cheung, "Detection of internet traffic anomalies using sparse laplacian component analysis," in *Proc. of the IEEE GLOBECOM*, Singapore, 12 2017.

[31] G. Thatte, U. Mitra, and J. Heidemann, "Parametric Methods for Anomaly Detection in Aggregate Traffic," *IEEE/ACM Transactions on Networking*, vol. 19, no. 2, pp. 512–525, 2011.

[32] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection for discrete sequences: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 823–839, May 2012.

[33] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[34] L. Akoglu, H. Tong, and D. Koutra, "Graph-based anomaly detection and description: a survey," *Data Mining and Knowledge Discovery*, vol. 29, pp. 626–688, 2015.

[35] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova, "Anomaly detection in dynamic networks: a survey," *WIREs Computational Statistics*, vol. 7, pp. 223–247, 2015.

[36] M. Mardani, G. Mateos, and G. B. Giannakis, "Dynamic anomalography: Tracking network anomalies via sparsity and low rank," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 1, pp. 50–66, February 2013.

[37] M. Mardani and G. B. Giannakis, "Estimating traffic and anomaly maps via network tomography," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1533–1547, June 2016.

[38] M. Xie, J. Hu, S. Guo, and A. Y. Zomaya, "Distributed Segment-Based Anomaly Detection With Kullback–Leibler Divergence in Wireless Sensor Networks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 101–110, January 2017.

[39] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian, "Anomaly extraction in backbone networks using association rules," *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1788–1799, December 2012.

[40] R. Jiang, H. Fei, and J. Huan, "A family of joint sparse PCA algorithms for anomaly localization in network data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2421–2433, November 2013.

[41] A. Lung-Yut-Fong, C. Levy-Leduc, and O. Cappe, "Distributed detection/localization of change-points in high-dimensional network traffic data," *Statistics and Computing*, vol. 22, pp. 485–496, 2009.

[42] S. Yang and W. Zhou, "Anomaly detection on collective moving patterns: Manifold learning based analysis of traffic streams," in *IEEE* 3^{rd} *International Conference on Social Computing*, 2011, pp. 704–707.

[43] S. T. Rowels and L. K. Saul, "Nonlinear dimensionality reduction by local linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[44] F. Silveira and C. Diot, "URCA: pulling out anomalies by their root causes," in *Proceedings of IEEE INFOCOM*, 2010, pp. 1–9.

[45] J. P. Cunningham and Z. Ghahramani, "Linear dimensionality reduction: Survey, insights, and generalizations," *Journal of Machine Learning Research*, vol. 16, pp. 2859–2900, 2015.

[46] Z. Xia, X. Zhou, W. Chen, and C. Chang, "A graph-based elastic net for variable selection and module identification for genomic data analysis," in *Proceedings of IEEE International Conference on Bioinformatics and Biomedicine*, 2010, pp. 357–362.

[47] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Matrix completion on graphs," *arXiv:1408.1717*, 2014.

[48] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proceedings of NIPS*, vol. 14, 2001, pp. 585–591.

[49] ——, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[50] S. Yan, D. Xu, B. Zhang, and S. Lin, "Graph embedding and extensions: a general framework for dimensionality reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40–51, 2007.

[51] J.-Y. Kao, D. Tian, H. Mansour, A. Ortega, and A. Vetro, "Disc-glasso: Discriminative graph learning with sparsity regularization," in *Proc. ICASSP*, 2017.

[52] H. P. Maretic, D. Thanou, and P. Frossard, "Graph learning under sparsity priors," in *Proc. ICASSP*, 2017.

[53] S. Sardellitti, S. Barbarossa, and P. D. Lorenzo, "On the graph fourier transform for directed graphs," *IEEE Journal of Selected Topics in Signal Processing*, 2017.

[54] R. Shafipour, A. Khodabakhsh, G. Mateos, and E. Nikolova, "A digraph fourier transform with spread frequency components," *arXiv preprint arXiv:1705.10821v1*, 2017.

[55] L. Le Magoarou, R. Gribonval, and N. Tremblay, "Approximate fast graph fourier transforms via multi-layer sparse approximations," *IEEE Transactions on Signal and Information Processing over Networks*, 2017.

[56] B. Girault, A. Ortega, and S. Narayanan, "Irregularity-Aware Graph Fourier Transforms," Feb. 2018, preprint. [Online]. Available: https://hal.inria.fr/hal-01708695

[57] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of Computational and Graphical Statistics*, vol. 15, no. 2, pp. 265–286, 2006.

[58] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[59] F. R. K. Chung, *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*, 2nd ed. American Mathematical Society, 1997, vol. 92.

[60] Y. Li, X. Luo, Y. Qian, and X. Zhao, "Network-wide traffic anomaly detection and localization based on robust multivariate probabilistic calibration model," *Mathematical Problems in Engineering*, pp. 1–26, 2015, article ID: 923792.

[61] H. Kasai, W. Kellerer, and M. Kleinsteuber, "Network volume anomaly detection and identification in large-scale networks based on online time-

structured traffic tensor tracking," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 636–650, September 2016.

[62] Juniper traffic sampling. [Online]. Available: www.juniper.net/techpubs/software/junos/junos60/swconfig60-policy/html/sampling-overview.html

[63] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving traffic demands for operational IP networks: Methodology and experience," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 265–280, June 2001.

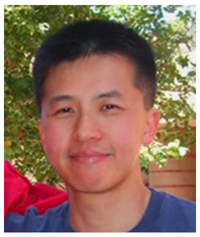
[64] A. O. Hero, N. Patwari, and K. Sricharan, "CRAWDAD dataset umich/rss (v. 2011-08-10)," Downloaded from https://crawdad.org/umich/rss/20110810, August 2011, [Onlice Accessed on 04 December, 2017].

[65] J. Goh, S. Adepu, K. Junejo, and A. Mathur. (2016) Secure Water Treatment (SWaT) Dataset. [Online]. Available: https://itrust.sutd.edu.sg/dataset/

[66] Y. Guo, H. Nejati, and N.-M. Cheung, "Deep neural networks on graph signals for brain imaging analysis," in *Proc. of IEEE International Conference on Image Processing (ICIP)*, 2017.

Manas Khatua is an Assistant Professor in the Department of Computer Science & Engineering at Indian Institute of Technology Jodhpur, India. Prior to that he was a Postdoctoral Research Fellow at Singapore University of Technology and Design (SUTD), Singapore, from 2015 to 2016. He completed his Ph.D. from Indian Institute of Technology Kharagpur, India, in 2015. He was associated with Tata Consultancy Services (India) for two and half years, and worked as a Lecturer of Bankura Unnayani Institute of Engineering, India, for more than two years. He passed M.Tech. in Information Technology from Bengal Engineering and Science University, India, and B.Tech. in Computer Science and Engineering from University of Kalyani, India. His research interests include Performance Evaluation, Wireless LANs, Sensor Networks, Network Security, Mobile Cloud Computing, and Internet of Things. He is a member of the IEEE.

Seyed Hamid Safavi was born in Ardabil, Iran. He received his PhD in electrical engineering from Shahid Beheshti University (SBU), Tehran, Iran in 2017. He also received his B.Sc. and M.Sc. degrees both in electrical engineering from K. N. Toosi University of Technology (KNTU), Tehran, Iran in 2010 and 2012, respectively. He was also a visiting fellow at Singapore University of Technology and Design (SUTD) from December 2015 to September 2016. He is currently a postdoctoral researcher with the Digital Signal Processing Laboratory (DiSPLaY), SBU. He received the best PhD thesis award from IEEE SBU Student Branch. DiSPLaY team leading by him achieved third prize at the ROCC grand challenge. He also received the IEEE travel grant to participate at the 1st IEEE ComSoc Summer School in 2015. His research interests include Signal and Image Processing, Machine Learning, Sparse Representation, Compressive Sensing, Linear Inverse Problems, and Convex Optimization. He is a student member of the IEEE.



Ngai-Man Cheung is an Assistant Professor with the Singapore University of Technology and Design (SUTD), Singapore. He received the Ph.D. degree in electrical engineering from the University of Southern California, in 2008. From 2009 to 2011, he was a Post-Doctoral Researcher with the Image, Video and Multimedia Systems Group, Stanford University. He has worked at the Texas Instruments Research Center Japan, Nokia Research Center, IBM T. J. Watson Research Center, the Hong Kong University of Science and Technology, and Mitsubishi Electric Research Laboratories. His research interests are signal and image processing, and computer vision. He is a senior member of the IEEE.