Machine learning in eigensubspace for network path identification and flow forecast

Irfan Lateef | Ali N. Akansu

Department of Electrical and Computer Engineering Department, New Jersey Institute of Technology, Newark, New Jersey, USA

Correspondence
Irfan Lateef, Department of Electrical and Computer Engineering Department, New Jersey Institute of Technology, University Heights, 818A Holmdel Road, Holmdel, NJ 07733, USA.
Email: ixl5825@njit.edu

Funding information: None

Abstract
This paper emphasizes the joint time-frequency interpretation of eigensubspace representation for network statistics as features for identification and tracking traffic flows based on the link level activity. Eigencoefficients (frequency domain feature) and eigenvector components (time domain feature) are jointly utilized to quantify their combined significance on the representation of each link data (each component of the link traffic vector) in the eigensubspace. The joint time-frequency method is employed to analyze the traffic data obtained from the Internet2 network. It is shown that the analysis with link-level resolution brings advantages for network traffic engineering applications. A machine learning method is investigated to identify network paths using eigenanalysis of link statistics as the feature set. The merit of the method is validated by experimental studies of the network scenarios considered in the paper. Eigenvectors and eigenflows in the subspace are jointly used as factors (features) for linear regression to forecast the network link traffic. It is demonstrated that the eigensubspace based auto-regressive order two, AR (2), predictor is superior to the time-domain based predictor to forecast the link level traffic of a network.

1 | INTRODUCTION

Recent developments in the analysis of network traffic data have mostly focused on classification and characterization of flows by using the flow level statistics [1–9]. Principal Component Analysis (PCA) was used to analyze the IP flow statistics to identify anomalous behaviour in the network [10]. They used PCA as a dimension reduction tool by reducing a very large number of flows into a few most significant eigenflows for analysis. Similarly, PCA was used for traffic classification with the IP flow statistics [3]. One of the major challenges with these techniques is the high number of flows in the network, which makes them impractical.

The merits of analyzing the traffic data with PCA to assess the state of the network for certain cases were presented in [10]. They argued that shifts in traffic behaviour can be analyzed better by looking at multiple links. They collected the time series data of the Origin-Destination (O-D) pair IP flows traversing multiple links in the network and transformed them onto an eigensubspace using PCA. It was demonstrated that the overall network behaviour could be studied by analyzing the first few eigenflows (eigencoefficients). This scheme suffers from two systemic problems. First, one needs to collect packet level statistics on a per flow basis for a large number of flows; there are \( n(n - 1)/2 \) flows for a network of \( n \) nodes. This is computationally very expensive in terms of data collection and processing. Second, these eigenflows do not reveal the significance and the status of a specific link within the O-D flow in the network. Therefore, if an O-D flow is identified as anomalous, it is not possible to find which link in the path is the root cause of the problem. We address these issues by demonstrating the advantages of the proposed method in the following sections.

The PCA for flow decomposition and prediction by adding extensions and improvement to the basic PCA was used in [11]. They have primarily employed robust PCA (RPCA) that allows the normal data to dwell in a low-dimensional space [12]. This is due to the strong correlation among normal observations, and special events as well as allowing noise to dwell in a sparse subspace. This technique was applied to highway traffic data. This paper reinforces our perspective of using the PCA components for prediction, but the use of RPCA is not applicable for our case since the focus here is for telecom...
network traffic data. The PCA based network traffic prediction problem was also studied in [13]. In that study, the K-means clustering algorithm to separate the flows into relevant groups was utilized. It also suffers from the same problems discussed above.

An overview of machine learning (ML) methods to analyze networks and their flows is given in [14]. They presented the state-of-the-art deep learning architectures and algorithms relevant to the network traffic control systems, and demonstrated a use case for intelligent traffic routing. They also discussed the applicability of deep learning for network flow prediction. Another survey on ML applications in all aspects of networking, including traffic classification, prediction, routing, congestion control, resource management, fault management and network security and their merits are given in [15].

Here, we utilize the link level statistics to identify large flows in the network by using PCA (eigendecomposition) in a novel way [1]. This technique reduces the data collection overhead and provides a finer tracking of the flows at the link level resolution. It allows improved solutions to network problems. We also use linear regression to predict eigenvectors and eigencoefficients that are successfully used as features to forecast the network traffic. Advantages of the proposed methods are explained and verified through performance simulations for the real data of Internet2 network.

2 | EIGENANALYSIS AND EIGENFLOWS

The eigendecomposition of $N \times N$ matrix $R$ is expressed as [1]

$$R \Theta = \Phi R \Phi^T = \Lambda,$$  \hspace{1cm} (1)

where $\Lambda$ is the diagonal matrix with its elements as the eigenvalues, and equivalently, $R_0$ is the covariance matrix of eigencoefficients. $\Phi$ is the eigenvector matrix of $R$ and populated by the eigenvectors as its columns, and $T$ indicates matrix transpose operation. It defines the resulting N-dimensional orthonormal eigensubspace for the given $R$ [16].

Here, eigensubspace representations of random vectors, that describe traffic variables like link bandwidth at measurement instances in megabits per second (Mbps), are used to analyze and understand the status of communication networks. Let us assume a network of $N$ links, and the snapshot of the link traffic vector at a periodic measurement time point is expressed by $r^T = [r_1, r_2, \ldots, r_N]$. The empirical correlation matrix $R$ of link traffic is calculated as for the predefined historical measurement data window of $W$ samples as [1]

$$R = \begin{bmatrix}
    \sigma_{r_1}^2 & \sigma_{r_1 r_2} & \cdots & \sigma_{r_1 r_N} \\
    \sigma_{r_2 r_1} & \sigma_{r_2}^2 & \cdots & \sigma_{r_2 r_N} \\
    \vdots & \vdots & \ddots & \vdots \\
    \sigma_{r_N r_1} & \sigma_{r_N r_2} & \cdots & \sigma_{r_N}^2
\end{bmatrix}, \hspace{1cm} (2)$$

where $r_{ij} = \cos(r_i r_j) / (\sigma_i \sigma_j)$ for $W'$, and $\sigma_{r_i}$ and $\sigma_{r_j}$ are the standard deviations of $r_i$ and $r_j$, respectively.

It is used in (1) to create the eigensubspace expressed in $\Phi$ matrix. The measurements are repeated periodically to update the eigensubspace due to the statistical variations (non-stationarity) of the network dynamics. The snapshot of $N$ link bandwidths populate the column vector $c^T$ where $n$ is the measurement time index in the regular clock with the assumption of the stationarity during the update period until $n+1$. Then, one can project this link traffic vector onto the currently defined eigensubspace as (forward transform) [1]

$$\Theta^T = \Phi^T c^T,$$  \hspace{1cm} (3)

where the most significant eigenvector, also known as the principal eigenvector, is commonly the first column of the eigenvector matrix $\Phi$. The link variables are expressed through the inverse transformation operator as

$$c^T = \Phi \Theta^T.$$  \hspace{1cm} (4)

Now, we can rewrite this expression more explicitly as follows:

$$\begin{bmatrix}
    c_1^T \\
    \vdots \\
    c_N^T
\end{bmatrix} = \begin{bmatrix}
    \Phi_{11} & \cdots & \Phi_{1N} \\
    \vdots & \ddots & \vdots \\
    \Phi_{N1} & \cdots & \Phi_{NN}
\end{bmatrix} \begin{bmatrix}
    \Theta_1 \\
    \vdots \\
    \Theta_N
\end{bmatrix}$$

$$= \begin{bmatrix}
    \Theta_1^T \Phi_{11} + \Theta_2^T \Phi_{12} & \cdots & + \Theta_N^T \Phi_{1N} \\
    \vdots & \ddots & \vdots \\
    \Theta_1^T \Phi_{N1} + \Theta_2^T \Phi_{N2} & \cdots & + \Theta_N^T \Phi_{NN}
\end{bmatrix}.$$  \hspace{1cm} (5)

Note that the components of the first eigenvector in (5) are $[\Phi_{11}, \Phi_{21} \ldots \Phi_{N1}]$ as the first column vector of $\Phi$.

We compute eigenflows $\{\Theta^T_k, k = 1, 2, \ldots, N\}$ that characterize the entire network at time $n$ by using (3). Figure 1(a) displays the variations of the most significant eigenflow, $\Theta^T_1$ at time $n$, for the network. The components of the principal eigenvector are displayed in Figure 1(b) for the case where 1 week of the Internet2 link data is used [16]. These components represent the contribution from the corresponding links into the most significant eigenflow.

As a corollary to (4), we express the $k^{th}$ link traffic for the measurement at time $n$ in the eigensubspace as

$$c^T_k = \sum_{l=1}^{N} \Theta^T_l \Phi_{kl} k = 1, 2, \ldots, N.$$  \hspace{1cm} (6)

In the next section, we define the joint time-frequency matrix to utilize the link or path specific (time/signal domain) features of the network along with the eigenflows.
3 | REPRESENTATION OF A LINK IN EIGENSUBSPACE

Let \( \{c_n^l\} \) represent link measurements at time instance \( n \) for the entire network, \( l = 1, 2, \ldots, N \). Let us define a more detailed eigen-subspace representation of the network status where the product of eigenflows and eigenvectors are interpreted for analysis as shown in the matrix

\[
\Psi = \Phi \Theta, \quad (7)
\]

where

\[
diag(\Theta) = I^n \Theta = diag(\Theta) \quad (8)
\]

and, \( I \) is \( N \times N \) identity matrix.

It is explicitly shown at time \( n \) as

\[
\Psi = \begin{bmatrix}
\theta_1^e \Phi_{11} & \ldots & \theta_N^e \Phi_{1N} \\
\vdots & \ddots & \vdots \\
\theta_1^e \Phi_{N1} & \ldots & \theta_N^e \Phi_{NN}
\end{bmatrix} \quad (9)
\]

The sum of the elements of the \( l^{th} \) row of \( \Psi \) is equivalent to the traffic measurement on the \( l^{th} \) link \( c_n^l \) as given in (6). \( \Psi \) is called the joint time-frequency matrix. Figure 2(b) displays the joint time-frequency matrix for the link measurements of the network that we analyze in the paper [1, 2]. We will use this matrix to identify large network data flows through the network topology as described in Section 5. In the next section, we give a new interpretation of eigenflows with the link-level focus for the network using the eigensubspace representation framework described above.

4 | LINK-LEVEL INTERPRETATION OF EIGENFLOWS

The vector \( \theta \) in (3) represents the eigenflows (eigencoefficients) in the network at time \( n \). For example, the first eigenflow is the inner
product of the first eigenvector and traffic measurement vector $e^T$ at time $n$ with link attributes as its components. A high value of an eigenvector component is related to high contribution of that link into the given eigenflow due to the least squares (L2) based optimality of the eigendecomposition [1].

Figure 2(a) shows the component values of all eigenvectors of the link data matrix where $N = 30$. In contrast, Figure 2(b) displays the corresponding joint time-frequency matrix $\Psi$ as defined in (9) below where the vertical axis is the component indices of eigenvectors and the horizontal axis is the eigenvector indices [1, 2]. In contrast to the traditional PCA based studies reported in the literature [17–19], our experiments show that a few most significant eigenvectors and eigenflows (dimension reduction) are not always able to capture all of the network characteristics. They are rather spread out to a larger subset as shown in Figure 2(a). We highlight that some components of eigenvectors with low eigenvalues that may not survive the dimension reduction step can be significant for the representation of certain link traffic in the eigensub-space. Therefore, we need to jointly look at the variations of all eigenvectors and all eigenflows in time in order to better track network dynamics and anomalies with the highest level of representation granularity.

Eigenflow (frequency domain) interpretation of network dynamics does not emphasize link or path specific (time/signal domain) features in a network. These features are equally critical to assess the state of the network. We propose the joint time-frequency interpretation [2] of eigenanalysis for network engineering as explained in the previous section [1]. The next section defines the process of identification of large flows using the techniques described in earlier sections.

5 | IDENTIFICATION OF LARGE NETWORK FLOWS

We demonstrate the use of the joint time-frequency matrix to identify large network flows. The identification process is comprised of the following steps.

- Step 1: Identify the column vector containing the most significant components. This is done by taking the L2-norm of the column vectors and sorting them. It is seen from Figure 2(b) that the 7th column vector of $\Psi$ has the most significant components for this case.
- Step 2: Find the components of the vector identified in Step 1 with significant contributions by setting a threshold $\alpha = 0.35$. The components of that vector are plotted as a bar graph in Figure 3(a).
- Step 3: Map the components on the network topology [16].
- Step 4: Identify the paths in the network created by the identified eigenvector components. This is done by verifying that more than one link corresponding to the components are connected together to form a path. It is seen in Figure 3(b) that the links identified in Step 3 traverse a path taken by a flow from DC to LA. We also observe that there is a second path taken by another flow from Seattle to Houston.
- Step 5: Go back to Step 1 and repeat the process for the next significant column vector of $\Psi$.

Note that Link6 is mapped in the figure even though it is below the threshold because its two adjacent links are significant contributors to a large flow. By inference, it has to pass through this link. While performing Step 4, there may be some links that are above the threshold but are not connected to any other link to form a path. Such links can be safely ignored from further analysis, mapping or plotting. For example, in Figure 3(b), Link5 and Link20 are ignored.

The link-level interpretation of eigenanalysis and its mapping on the topology demonstrates that large flows in the network can be detected. It helps us to better understand the network activity in a real-world scenario with a more efficient implementation than the currently used methods [3–5]. Note that the selection of the threshold to identify significant links requires some experimental study on the network of interest and may also be automated.

There are various applications of this research. Network traffic has been growing drastically in the last decade with the advent of smart phones and proliferation of video applications.
Due to this, it has become a huge challenge to characterize, forecast and engineer network traffic. The traffic largely comprises many small flows and a few large flows over a period of time with their specific network requirements. The identification of these large flows as addressed in the paper has a major impact in the overall network performance and security. If the flow is from a trusted source on the expected path, then the information is used for traffic engineering and optimization applications. The link level granularity of data is instrumental in long-term network planning applications. It is also used in congestion control, resource management, fault management and QOS management to name a few. Besides the network applications mentioned above, there are several other uses in the areas of application monitoring, security awareness and intrusion detection, policy validation and service assurance. For example, if a large flow is identified and the source or the path is anomalous, then it could be security violation or a Denial of Service (DoS) attack. It is extremely difficult to build these applications that infer the network behaviour by merely looking at the traffic on a single link or to the entire network. Thus, the joint time-frequency interpretation of network traffic data in the eigensubspace gives better insight and features in analyzing link level and network wide behaviour.

6 | NETWORK FORECAST IN EIGENSUBSPACE

Using variations of eigenvectors and eigencoefficients, we train a linear regression model to predict their future values which are tied to link parameters of the network. A subset of the link traffic data provided by a large service provider is used for this study. The network characteristics are summarized as follows:

- There are 50 nodes in the network \{n_1, \ldots, n_{50}\}
- These 50 nodes are fully connected to each other resulting in 2,450 links labelled as \{l_1, l_2, l_3, \ldots, l_{2,450}\}
- The link traffic measurement data (average bandwidth utilized) was collected for each link at 5-min time intervals for 1-h duration (12 samples)
- The data was collected for the same hour of the day for 92 days (3 months)
- Initially, we use a subset consisting of 24 links to run our experiments

In this analysis, we write link traffic measurements in the 3D data array \(L_{l,t,d}\) where
- \(l\) – link index \(l = 1, 2, \ldots, N\) where \(N = 24\),
- \(t\) – time index of 5-min intervals for the same 1-h period in each day \(t = 1, 2, \ldots, T\) where \(T = 12\),
- \(d\) – day index, \(d = 1, 2, \ldots, D\) where \(D = 92\).

This 3D array might also be represented as \(L^d_{l,t}\) with the same subscripts and superscript as defined. For convenience, we drop subscript and/or superscript and use the following notation:

\[ L^d_{l,t} \]

\(L^d\) – matrix of link traffic measurements for day \(d\), and for all 24 links, where each row vector of the matrix is 12-dimensional and populated by the corresponding measurements of the 5-min intervals for the given hour of the day. In this experiment, we only used the hour between 9:00 PM and 10:00 PM.

\(L^d_{l,t}\) – corresponds to a single row vector of \(L^d\) matrix as mentioned above for link \(l\).

The empirical correlation matrix of size 24 \(\times\) 24 for the link traffic on a given day \(d\) is computed as

\[
R^d_{L} = \begin{bmatrix}
    r_{11} & \cdots & r_{1N} \\
    \vdots & \ddots & \vdots \\
    r_{N1} & \cdots & r_{NN}
\end{bmatrix},
\]

(10)

where \(i\) and \(j\) are the link indices, and the pairwise correlations 
\(r_{ij} = E[L^d_{l,t}L^d_{l',t}']\) are calculated based on the 12 measurements of the 5-min intervals. The eigenmatrix \(\Phi^d\) of the network for day \(d\) is expressed as

\[
\Phi^d = \Lambda^d \Phi^d.
\]

(11)

Then, the daily eigencoefficients \(\Theta^d\) of size \(N \times T\) comprised of eigencoefficients vectors calculated for each 5-min interval of 9:00 to 10:00 PM as its columns is defined in the current subspace \(\Phi^d\) as

\[
\Theta^d = [\Phi^d]^T L^d.
\]

(12)

Now, we drop the superscript \(d\) for convenience and expand the equation where the first row of \([\Phi^d]^T\) is the principal eigenvector as follows:

\[
\Theta = \begin{bmatrix}
    \theta_{11} & \cdots & \theta_{1T} \\
    \vdots & \ddots & \vdots \\
    \theta_{N1} & \cdots & \theta_{NT}
\end{bmatrix} = \begin{bmatrix}
    \Phi_{11} & \cdots & \Phi_{N1} \\
    \vdots & \ddots & \vdots \\
    \Phi_{1N} & \cdots & \Phi_{NN}
\end{bmatrix}\begin{bmatrix}
    L_{11} & \cdots & L_{1T} \\
    \vdots & \ddots & \vdots \\
    L_{N1} & \cdots & L_{NT}
\end{bmatrix},
\]

(13)

The first column vector of the eigencoefficients matrix in (13) corresponds to the first 5-min time interval and written as

\[
\begin{bmatrix}
    \theta_{11} \\
    \vdots \\
    \theta_{N1}
\end{bmatrix} = \begin{bmatrix}
    \Phi_{11} L_{11} + \Phi_{21} L_{21} + \cdots + \Phi_{N1} L_{N1} \\
    \vdots \\
    \Phi_{1N} L_{11} + \Phi_{2N} L_{21} + \cdots + \Phi_{NN} L_{N1}
\end{bmatrix}.
\]

(14)

Similarly, its second column corresponds to the second 5-min time interval as

\[
\begin{bmatrix}
    \theta_{12} \\
    \vdots \\
    \theta_{N2}
\end{bmatrix} = \begin{bmatrix}
    \Phi_{11} L_{12} + \Phi_{21} L_{22} + \cdots + \Phi_{N1} L_{N2} \\
    \vdots \\
    \Phi_{1N} L_{12} + \Phi_{2N} L_{22} + \cdots + \Phi_{NN} L_{N2}
\end{bmatrix}.
\]

(15)
and so on. As a result, the elements of the first row vector of the $N \times T$ eigencoefficient matrix where $N = 24$ and $T = 12$ in this example. Similarly, the second row corresponds to the second most significant eigencoefficients for those 5-min time intervals, and so on. A subset of the rows of $\Theta$ matrix given in (13) are used for linear regression and prediction of link values as discussed later in this section.

From (4), the traffic of the link $l$, for day $d$, and the specific 5-min time interval $i$ is calculated as follows:

$$L_{d,l}^i = \sum_{k=1}^{V} \Theta_{kl}^d \Phi_{lk}, \quad (16)$$

where $V$ is the pre-defined number of most significant vectors (dimension reduction based on explained variance, $V \leq N$) running on the dimension index $k$ as used to approximate the link traffic values in the subspace. When we drop the day superscript $d$, for $i = 1$ and $V = N$ (no dimension reduction) (16) is expanded as

$$L_{11} = \begin{bmatrix} \theta_{11} \Phi_{11}^d + \theta_{21} \Phi_{12}^d & \cdots & \theta_{N1} \Phi_{1N}^d \\ \vdots & \ddots & \vdots \\ \theta_{1N} \Phi_{11}^d + \theta_{2N} \Phi_{12}^d & \cdots & \theta_{N1} \Phi_{1N}^d \end{bmatrix}. \quad (17)$$

In (17), only the first column of the $\Theta$ matrix has been used to compute the link traffic values for the first 5-min time slot. Similarly, the link values for the second 5-min time interval is computed by using the second column of the $\Theta$ matrix as follows:

$$L_{12} = \begin{bmatrix} \theta_{12} \Phi_{11}^d + \theta_{22} \Phi_{12}^d & \cdots & \theta_{N2} \Phi_{1N}^d \\ \vdots & \ddots & \vdots \\ \theta_{12} \Phi_{11}^d + \theta_{2N} \Phi_{12}^d & \cdots & \theta_{N2} \Phi_{1N}^d \end{bmatrix}. \quad (18)$$

Hence, the link traffic measurement matrix is defined for each 5-min time slot, and for all the links when $T = 12$ and $N = 24$ as

$$L_{d,l}^i = \begin{bmatrix} L_{11} & L_{12} & \cdots & L_{1T} \\ \vdots & \vdots & \ddots & \vdots \\ L_{N1} & L_{N2} & \cdots & L_{NT} \end{bmatrix}. \quad (19)$$

Now, we use the eigenvectors and eigencoefficients of the previous 2 days to predict the link traffic for the next day as [20, 21].

$$\hat{\Phi}^{d+1} = \rho_1 \Phi^d + \rho_2 \Phi^{d-1} + \varepsilon_{\Phi}, \quad (20)$$

$$\hat{\Theta}^{d+1} = \hat{\theta}_1 \Theta^d + \hat{\theta}_2 \Theta^{d-1} + \varepsilon_{\Theta}, \quad (21)$$

where $\rho_1$ and $\rho_2$ are the auto-regressive order two, AR(2), regression model parameters used to predict the eigencoefficient $\Phi$, $\varepsilon_{\Phi}$ is the prediction (white) noise corresponding to the AR(2) process of the eigencoefficient $\Phi$. $\hat{\theta}_1$ and $\hat{\theta}_2$ are the AR(2) regression model parameters to predict the eigencoefficient matrix $\Theta$, $\varepsilon_{\Theta}$ is the prediction (white) noise corresponding to the AR(2) process of the eigencoefficient matrix $\Theta$.

By using the predicted $\hat{\Phi}^{d+1}$ and $\hat{\Theta}^{d+1}$ in (4) we compute $\hat{T}_l^{d+1}$ as expressed in (22) below

$$\hat{T}_l^{d+1} = \sum_{k=1}^{V} \hat{\Theta}_{kl}^{d+1} \hat{\Phi}_{lk}^{d+1}, \quad (22)$$

where $V$ is the reduced dimension in the eigensubspace, $V \leq N$, to approximate the time series with the permissible prediction error (or explained variance).

## 7 PERFORMANCE COMPARISON OF TIME SERIES AND PCA BASED NETWORK FORECASTS

In this section, we develop an auto-regressive, order two, AR(2), model to predict the link traffic of a network based on its historical data [14]. We built AR(2) predictors in the time domain and also in the eigensubspace, and compared their performances as explained in the following two subsections.

### A. Linear Regression in Time Domain

We employed a vector auto-regressive model for linear regression in the time domain [22]. The 3D data set $L_{d,l}^i$ introduced in Section 6 is labelled as $L_d$. We regressed the link traffic in the time domain for day $d$ by using the data of the previous days $d - i$, $i = 1, 2, \ldots, M$ by using the vector auto-regressive (VAR) model expressed as [22, 23]

$$\hat{T}_l^d = C_i + \sum_{j=1}^{M} \psi_{i,j} L_{d-j}^i + \xi_i, \quad (23)$$

where

- $C_i$ is a column vector of constant offsets,
- $\psi_{i,j}$ is a vector of AR parameters of the model,
- $M$ is the order of the auto-regressive model for $i$, $M = 2$ for AR(2) model,
- $i$ is the time index of 5-min intervals for the same one hour period in each day $t = 1, 2, \ldots, T$ where $T = 12$,
- $\xi_i$ is a column vector of white noise.

Equation (23) is rewritten for AR(2) as follows:

$$\hat{T}_l^d = C_i + \psi_{i,1} L_{d-1}^i + \psi_{i,2} L_{d-2}^i + \xi_i, \quad (24)$$
This VAR model is written for all of the $T$ column vectors of the time intervals as

$$
\hat{L}_d^1 = C_1 + \psi_{1,1}L_d^{-1} + \psi_{1,2}L_d^{-2} + \xi_1 \\
\hat{L}_d^2 = C_2 + \psi_{2,1}L_d^{-1} + \psi_{2,2}L_d^{-2} + \xi_2 \\
\vdots \\
\hat{L}_d^{12} = C_{12} + \psi_{12,1}L_d^{-1} + \psi_{12,2}L_d^{-2} + \xi_{12}
$$

(25)

The column vectors $\hat{L}_d^1$ to $\hat{L}_d^{12}$ are combined to form the forecasted link traffic matrix for 5-min intervals for day $d$. Then, we estimate the parameters of AR(2) by using Equation (25).

Figure 4 displays the link measurements for the link $l = 24$ of day $d$ along with their forecasted values by the AR(2) model in the time domain. It is seen from Figure 4 that the downward trend of the link traffic is forecasted with about 10% prediction error. We will repeat the same experiment in the eigensubspace in the next section and show its merit.

B. Linear Regression with Joint Time-Frequency Features in Eigensubspace

In this experiment, the regression is performed by using the $\Theta$ and eigenmatrix $\Phi$, and the link traffic is predicted as per (22).

We used the same data set as the one used in the time-domain based experiment. The traffic measurements are mapped onto the currently defined eigensubspace. A small subset of eigenflows (dimension reduction) are used in the parameter calculations of AR(2) model. The eigensubspace based AR(2) predictor results are also displayed in Figure 4. It is observed from the figure that the eigensubspace based linear regression model forecasts the network traffic more accurately than the time domain based prediction [24].

8 | MACHINE LEARNING TO ID ENTIFY PATHS WITH NETWORK FEATURES IN EIGENSUBSPACE

The use of machine learning methods for network traffic classification and flow prediction was reported in [14]. We extend that work to investigate the use of network eigenfeatures in such learning methods. In Section 5, we used manual thresholding of the joint time-frequency matrix elements defined in the eigensubspace to identify large network flows. The manual thresholding as described in Section 5 suffers from several shortcomings:

a. The threshold needs to be set heuristically based on historical data. Therefore, the threshold detection will fail as soon as the trend diverges from the historical trend. Hence, it may produce inaccurate results in the long-run when the non-stationarity in data becomes significant.

b. The value of threshold is crucial for this method, and it becomes a major challenge due to the exponential growth in broadband traffic and increased data types. It may be an interesting research topic to employ ML method for threshold selection that is beyond the scope of this study.

c. Finally, the usage patterns are driven by many events and large geographies where the thresholding method may not work as the centre of gravity of source and destinations of the traffic change.

Hence, we came up with the ML method that learns from its past experience and quickly adapts to new trend in the traffic characteristics automatically without manual intervention.

In this section, we present the method to automate the identification process. Instead of thresholding, we employ a machine learning technique utilizing the same column vectors of the joint time-frequency matrix, see (9), to identify paths carrying large traffic flows. It is based on an artificial neural network (ANN) using a multilayer perceptron (MLP) model.

The MLP is a class of feed-forward (FF) artificial neural network [25]. A single neuron MLP model is shown in Figure 5. Its inputs $x^T = [x_1 \ x_2 \ \cdots \ x_n]$ are weighted by the coefficients $w^T = [w_1 \ w_2 \ \cdots \ w_n]$ and summed together in $\nu$. The output $y$ is obtained through the activation function $g(\nu)$ with
the threshold $T$. Each node in a neuron uses a nonlinear activation function except for the input nodes [25].

The MLP model is expressed as follows:

$$\nu = \sum_{i=0}^{n} x_i w_i = x^T w,$$  \hspace{1cm} (26)

where

$$x = [x_1 \ x_2 \ \cdots \ x_n]^T \in R^n$$

$$w = [w_1 \ w_2 \ \cdots \ w_n]^T \in R^n$$

$$y = g(\nu).$$ \hspace{1cm} (27)

There are several activation functions commonly used in machine learning algorithms. We mostly used the sigmoid function in this study given as

$$g(\nu) = \frac{1}{1 + e^{-\nu}}.$$ \hspace{1cm} (28)

We also experimented hyperbolic tangent and other activation functions as discussed later.

In this study, we use the neural network that is comprised of the input layer, one or more hidden layers and the output layer as displayed in Figure 6.

We use the column vectors of the joint time-frequency matrix of the network statistics defined in (9) as the input to the neural network. The network path numbers are labelled as the outputs of the neural network model. We have a training set of $p$ pairs of $\Psi_i$, $y_j$ where $\Psi_i$ is a column vector of the joint time-frequency matrix and $y_j$ is a network path index corresponding to that column vector where $i=1 \ldots p$. The network paths are identified by using the method described in 5. Herein, we intend to calculate the weights of the neurons that map a given joint time-frequency vector to the corresponding path index at the output.

The training process is an optimization problem expressed as

$$\min \frac{1}{2} \sum_{i=1}^{p} (y_i - \Psi_i^T w)^2.$$  \hspace{1cm} (29)

It can be rewritten as

$$\min f(w) = \frac{1}{2} \|y - \Psi^T w\|^2.$$ \hspace{1cm} (29)

In (29), we assumed that the activation function is the identity map to simplify the optimization model [26]. We use this supervised ANN algorithm in the eigensubspace where built-in dimension reduction is inherently achieved.

The optimization problem defined in (29) is solved to find the optimal sets of weights used by the neural network model to identify the paths. We used the feed forward with back propagation (FFBP) technique for learning as described in [27–29]. This technique is implemented as an unconstrained optimization that uses a gradient descent algorithm. The gradient descent algorithm used to find the optimal weights of the model is written as

$$w(k+1) = w(k) + \alpha(k) \psi \varepsilon(k),$$  \hspace{1cm} (30)

$$\varepsilon(k) = y_j - \Psi_i^T w(k)$$

where $k$ is the index of iteration step,

$\alpha(k)$ is the learning rate,

$\varepsilon(k)$ is the error between actual and predicted value.

It is noted that, we use the batch gradient descent (BGD) [30] in (30) due to its superior performance over scaled conjugate gradient (SCG) [31], Levenberg–Marquardt (LM) [32, 33] and Bayesian regularization (BR) algorithms [34, 35]. It yields an unbiased estimate of gradients and theoretically guaranteed to converge to the global minimum along with a straight trajectory if the loss function is convex.

Similarly, we used the tan sigmoid (TS) and Elliott Sigmoid (ES) activation functions [32]. We also used the minimization loss function of mean squared error (MSE) and cross-entropy error (CEE) in our experimental studies and identified the best ones for the task at hand [36, 37].

We experimented with various optimization algorithms and activation functions to train of the network. We varied the number of neurons in the hidden layers and the size of training data features in the eigensubspace, (9), to evaluate the convergence and performance of the algorithm.

Note that each $\Psi_i$ column vector of (9) corresponds to a network path identified with the label $y_j$. Using the network traffic data, we identified and labelled the large network flows traversing a certain path in the network during the training step. We had limited data for only 200 input vectors with 100 network paths from the actual network. This dataset was too small for these experiments. Therefore, we generated a dataset of 100 times
CONCLUSIONS

9 | COMPUTATIONAL COMPLEXITY

We analyzed the computational complexity of the proposed techniques and compared them with the prior work in [24]. The are two fundamental differences in the collection of time series data. First, the method in [10] collects O-D flows data which can be very large in a given network, versus in our proposed methods it is link level which scales only with the number of links in the network. For example, a network of \( n \) nodes, there are \( n(n-1)/2 \) possible O-D flows in a maximally connected network as compared to \( n-1 \) links in a minimally connected network. Second, the O-D flow data is collected by sampling the arriving packets; therefore, it depends on the link bandwidth whereas link level data is collected at regular intervals of 5 min regardless of the link bandwidth.

Assuming these flows are going over 100 Gbps links with a packet size of 1500 bytes, there are 8.3 M packets/s passing through each interface. If a sampling rate \( N = 1000 \) is used, then there are 8.3 K packet headers/second being collected per flow.

With a standard 20 bytes packet header size, the total data collected is approximately 600 Mbytes/h. For \( n = 10 \) nodes, there are 45 O-D flows, which results in 27 Gb/h data collection volume. Clearly, this is an insurmountable challenge for using this technique. It calls for further increase in sampling rate, aggregation and pre-processing at local nodes. This also adds to the infrastructure overhead and bandwidth load on the network. For the proposed method, the same network with \( n = 10 \), we have 9 links to monitor every 5 min, that is, 108 samples/h. Furthermore, this data is already being collected, and therefore, there is no overhead to the network. Therefore, in terms of computational complexity, our method is several orders of magnitude simpler than O-D flow analysis.

It is noted that the O-D flows use Deep Packet Inspection methods to infer the path of the flow using the origin and destination of the packets. They utilize an additional step of routing table lookup to identify the path in the network. The computation complexity analysis of this method is studied well. It is known to be computationally expensive and it cannot be deployed in very large networks [38, 39]. A performance comparison of the TomoGravity method used by O-D flows, the traditional PCA and the Deep architecture Long Short-Term Memory traffic matrix prediction method was reported in [40]. They demonstrated the superiority of the PCA and LSTM based methods over the O-D flow (TomoGravity) with respect to the temporal relative prediction error.

10 | CONCLUSIONS

O-D flow based eigensubspace methods are computationally costly. More importantly, they lack the link-level resolution required to efficiently track the dynamics of a heterogeneous network [16]. We developed a granular method to monitor the variations of network traffic to identify anomalies. The method utilizes the eigendecomposition of the empirical correlation matrix of link traffics in a given network. This joint time-frequency interpretation of eigensubspace representation of traffic data provides additional insights to better understand the overall network behaviour as well as the individual links. We demonstrate in the paper that the link-level focus on the statistical analysis leads to identify local anomalies as the building blocks of elephant flows in the network. We also show in the paper that the eigensubspace-based network forecast outperforms the methods that use the time domain based measurements and predictions.

We employed ANN with FFBP to identify network paths by using joint time-frequency (eigenvector/weighted) eigenvectors as features. Specifically, for our network model with 30 inputs (links/vectors) and 100 possible outputs (network paths), we have made the following observations:

1. A training set of approximately 25,000 inputs is required to accurately train an ANN model for network path identification.
2. The same number of neurons are required in the hidden layers as the number of neurons in the input layer.
3. The performance of the learning algorithm does not degrade with up to 20% of random noise in the input vector components.

4. The Scaled Conjugate Gradient (SCG) algorithm with Cross-entropy loss function (CEE) shows better performance and speed of convergence in our experiments for the given data set. The rate of convergence differs from one sample data set to another but always converges.

5. Elliott Sigmoid (ES) activation function works faster than tan sigmoid (TS) due to elimination of the exponential (e) factor in the function.

We conclude that an artificial neural network with feed forward and backward propagation model using scaled conjugate gradient optimization can be used as an effective machine learning technique to identify large network flow paths using the proposed joint time-frequency matrix as its input data features in the eigensubspace. This novel technique provides new insights to the operators for automating the network engineering operations.

**ORCID**

Irfan Lateef https://orcid.org/0000-0002-8201-841X

Ali N. Akansu https://orcid.org/0000-0001-5642-5276

**REFERENCES**

1. Akansu, A.N., Haddad, R.A.: Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets. CA Academic Press, San Diego (1992)

2. Haddad, R.A., Akansu, A.N., Benyassine, A.: Time-frequency localization in transforms, subbands, and wavelets: A critical review. Opt. Eng. 32(7), 1411–1430 (1993)

3. Yan, R., Liu, R.: Principal component analysis based network traffic classification. J. Comput. 9(5), 1234–1240 (2014)

4. Papagiannaki, K., et al.: A pragmatic definition of elephants in internet backbone traffic. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement, pp. 175–176. ACM, New York (2002)

5. Papagiannaki, K., et al.: On the feasibility of identifying elephants in internet backbone traffic. Sprint Labs, Sprint ATL, Tech Rep TR01-ATL-110918 (2001)

6. Jin, T., et al.: Traffic engineering of high-rate large-sized flows. In: Proc. 14th IEEE International Conference on High Performance Switching and Routing (HPSR), pp. 128–135. IEEE, Piscataway, NJ (2013)

7. Hayashi, R., et al.: Impact of traffic correlation on the effectiveness of multilayer traffic engineering. In: Asia-Pacific Conference on Communications, pp. 936–940. IEEE, Piscataway, NJ (2005)

8. Coates, A., et al.: Internet tomography. IEEE Signal Process. Mag. 19(3), 47–65 (2002)

9. Joshi, M., Hadi, T.: A review of network traffic analysis and prediction techniques. arXiv preprint, arXiv:1507.05722 (2015)

10. Lakhina, A.: Network-wide traffic analysis: Methods and applications. Ph.D. thesis, Boston University (2007)

11. Xing, X., et al.: Traffic flow decomposition and prediction based on robust principal component analysis. Int. Conf. Intell. Trans. Syst. 59, 2219–2224 (2015)

12. Candès, E.J., et al.: Robust principal component analysis. J. ACM 58(3), 1–37 (2011)

13. Filho, R.H., Maia, J.E.B.: Network traffic prediction using PCA and k-means. In: 2010 IEEE Network Operations and Management Symposium-NOMS 2010, pp. 938–941. IEEE, Piscataway, NJ (2010)

14. Fadullah, Z.M., et al.: State of the art deep learning: Evolving machine intelligence toward tomorrow’s intelligent network traffic control systems. IEEE Commun Surv. Tutor. 19(44), 2432–2455 (2017)

15. Boutaba, R., et al.: A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. J. Internet Serv. Appl. 9(1), 1–99 (2018)

16. Internet2: perfsonar user guide (2016). https://docs.perfsonar.net/index.html

17. Golub, G.H., der Vorst, H.A.V.: Eigenvalue computation in the 20th century. J. Comput. Appl. Math. III(1–2), 35–65 (2000)

18. MATLAB: Matlab vector autoregression (var) models (2019)

19. Liu, J., et al.: An effective method of monitoring the large-scale traffic pattern based on RMT and PCA. J. Probab. Stat. 2010, 375942 (2010)

20. Zivot, E., Wang, J.: Vector autoregressive models for multivariate time series. Modeling Financial Time Series with SPLUS, pp. 385–429. Springer, New York (2006)

21. Hawkins, D.: On the investigation of alternative regressions by principal component analysis. J. R. Stat. Soc. Ser. C Appl. Stat. 22(3), 275–286 (1973)

22. Internet2: Internet2 network infrastructure topology (2019). https://internet2.edu/network/state-and-regional-r-e-networks/

23. Luketepohl, H.: In: Stable Vector Autoregressive Processes, pp. 13–16. Springer, New York (2005)

24. Lateef, I., Akansu, A.N.: Link-level interpretation of eigenanalysis for network traffic flows. In: 2017 51st Annual Conference on Information Sciences and Systems (CISS), pp. 1–6 (2017), https://doi.org/10.1109/CISS.2017.7926117

25. Rosenblatt, F.F.: Principles of neurodynamics, perceptrons and the theory of brain mechanisms. Am. J. Psychol. 76(4), 705 (1963)

26. Chong, E.K.P., Zak, S.H.: In: An Introduction to Optimization, 3rd ed., pp. 247–265. John Wiley & Sons, New York (2008)

27. Chauvin, Y., Rumelhart, D.E.: Backpropagation: Theory, Architectures, and Applications. Lawrence Erlbaum Associates, Hillsdale, NJ (1995)

28. Svozil, D., Kvassnicka, J.P.V.: Introduction to multi-layer feed-forward neural networks. Chemometr. Intell. Lab. Syst. 59, 43–66 (1997)

29. Lee, J.H., Shin, M.J.R.J.: Machine learning: Overview of the recent progresses and implications for the process systems engineering field. Comput. Chem. Eng. 114(4), 111–121 (2018)

30. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. Neural Networks: Tricks of the Trade, vol. 7700. Springer, Berlin (2012)

31. Moller, M.E.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw. 6, 525–533 (1993)

32. Marquardt, D.: An algorithm for least squares estimation of nonlinear parameters. SIAM J. Appl. Math. 11(2), 431–441 (1963)

33. Hagan, M.T., Menhaj, M.: Training feedforward networks with the marquardt algorithm. IEEE Trans. Neural Netw. 5(6), 989–993 (1999)

34. MacKay, D.J.C.: Bayesian interpolation. Neural Comput. 4(No.3), 415–437 (1992)

35. Foresee, F.D., Hagan, M.T.: Gauss-newton approximation to bayesian learning. In: Proceedings of the International Joint Conference on Neural Networks, vol. 3, pp. 1930–1935. IEEE, Piscataway, NJ (1997)

36. Vogl, T.P., et al.: Accelerating the convergence of the backpropagation method. Biol. Cybern. 59, 257–263 (1988)

37. Bishop, C.M.: In: Pattern recognition and machine learning, pp. 235–236. Springer Verlag, New York (2006)

38. Niccolò, C., Luigi, C., Fulvio, R.: Optimizing deep packet inspection for a DPI traffic classifier. In: GLOBECOM 2009-2009 IEEE Global Telecommunications Conference, pp. 1–8. IEEE, Piscataway, NJ (2009)

39. Jianlong, Z., et al.: Towards traffic matrix prediction with LSTM recurrent neural networks. Electron. Lerr. 54(9), 566–568 (2018)

**How to cite this article**: Lateef I, Akansu AN. Machine learning in eigensubspace for network path identification and flow forecast. IET Commun. 2021;1–10. https://doi.org/10.1049/cmu2.12230