Routing-Oblivious Network Tomography with Flow-Based Generative Model

直接测量网络流量矩阵（TM）的成本很高.
本文首次使用基于流量的生成模型来估计TM

Yan Qiao†, Xinyu Yuan †, Kui Wu‡
† School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China
‡ Department of Computer Science, University of Victoria, B.C., Canada
Emails: qiaoyan@hfut.edu.cn, yxy5315@gmail.com, wkui@uvic.ca

***Abstract*—Given the high cost associated with directly measuring the traffic matrix (TM), researchers have dedicated decades to devising methods for estimating the complete TM from low-cost link loads by solving a set of heavily ill-posed linear equations. Today's increasingly intricate networks present an even greater challenge: the routing matrix within these equations can no longer be deemed reliable. To address this challenge, we, for the first time, employ a flow-based generative model for TM estimation by establishing an invertible correlation between TM and link loads, oblivious of the routing matrix. We demonstrate that the lost information within the ill-posed equations can be independently segregated from the TM. Our model collaboratively learns the invertible correlations between TM and link loads as well as the distribution of the lost information. As a result, our model can unbiasedly reverse-transform the link loads to the true TM. Our model has undergone extensive experiments on two real-world datasets. Surprisingly, even without knowledge of the routing matrix, it significantly outperforms six representative baselines in deterministic and noisy routing scenarios regarding estimation accuracy and distribution similarity. Particularly, if the actual routing matrix is absent, our model can improve the performance of the best baseline by $41\% \sim 58\%$.**

***Index Terms*—Traffic matrix estimation, Network tomography, Flow-based generative model**

I. Introduction

In an era of exponentially increasing data traffic and the advent of technologies like cloud computing and edge computing, it has become imperative to enhance network observability for efficient management, optimization, and security. A key metric in network observability is the traffic matrix, delineating traffic volumes between all origin-destination (OD) pairs within the network. This matrix is an invaluable resource in understanding traffic status, aiding in various network management tasks from traffic engineering [1], anomaly detection [2], to capacity planning [3]. While the evolution of software-defined network (SDN) [4] architecture can quantify specific OD flows at programmable routers, direct network-wide OD flow measurement is typically impractical due to prohibitive costs. As the network expands, the network-wide OD flows surge exponentially. Frequently measuring these flows not only consumes vast amounts of expensive ternary content-addressable memory (TCAM) in routers, but also disrupts other network operations [5]. Thus, there is a pressing need for methodologies to estimate the traffic matrix without relying on direct network-wide measurements.

Currently, two *orthogonal* techniques have been devised for this purpose: matrix/tensor completion and network tomography (NT). The matrix/tensor completion-based methods measure partial OD flows and recover the missing ones by leveraging the spatio-temporal correlations between OD flows [6] [7]. Conversely, NT-based methods bypass flow measurements and estimate the complete traffic matrix from router port link load statistics [8] [9]. Each method has its cons and pros, and no conclusive evidence suggests that one outperforms the other. However, it's widely accepted that the performance of matrix/tensor completion-based methods depends heavily on the quantity and distributivity of the measurements. If the number of measured flows is inadequate or not sufficiently representative, the accuracy of matrix/tensor completion-based methods may be compromised.

The merits of NT-based methods are straightforward. Since link loads have significantly lower dimensions and can be collected efficiently using simple network management protocol (SNMP) tools at minimal cost, link load measurements can be executed frequently without occupying router TCAMs. This implies that if the complete traffic matrix can be accurately inferred from low-cost link loads, traffic matrix information can be monitored in real-time and measurement granularity can be refined.

Nevertheless, the NT-based methods have an inherent difficulty in solving a rank-deficient linear system. To briefly explain the reason while deferring the detailed problem formulation to Section III-A, we denote X as the flow vector formed from TM and Y as the vector of link loads. Then X and Y should satisfy a group of linear equations, i.e.,

$$\mathbf{A}X = Y, \tag{1}$$

where $\mathbf{A}$ is the routing matrix of flows in TM. However, the linear system (1) is generally rank-deficient, with innumerable solutions for X. In other words, *information loss* exists when transforming the high-dimensional X to the low-dimensional Y through routing matrix $\mathbf{A}$. To compensate for the lost information, traditional TM estimation methods impose additional assumptions on TM, which may not always hold in practice, to obtain a unique solution [8] [10]. Recently, with the rapid emergence of deep learning (DL) techniques [11], a few works leverage DL models to learn the traffic feature from historical data and then search for the optimal solution of (1) that can best satisfy the learnt feature [12] [13] [14]. *Almost all existing NT solutions, including the DL-based ones, assume that the routing matrix* $\mathbf{A}$ *is known or at least partially known.*

This assumption serves as a cornerstone in enabling NT-based traffic estimation. However, this foundation comes under scrutiny with the increasing adoption of adaptive routing policies in modern computer systems to balance network loads. Configured with an adaptive routing policy, routers often choose routing paths dynamically based on current network loads. This implies a lack of precise knowledge of the actual routing matrix $\mathbf{A}$, as represented in Eqn. (1). While some recent studies have explored solving NT-based problems using a probabilistic routing matrix [15], obtaining exact routing probabilities under an adaptive routing policy remains an unresolved challenge since the probabilities may be driven purely by traffic load. Therefore, a significant question emerges: ***Can we conduct NT-based traffic estimation if we completely dismiss this fundamental assumption?***

Novelty: This paper provides a positive answer to this fundamental question. Instead of forcing the estimation of TM to satisfy the equations (1) with routing matrix $\mathbf{A}$, we aim to learn a bijective correlation between TM and link loads when the routing matrix is absent. The "bijective" means that the TM is unique given an instance of link loads. Nevertheless, we still need to break the barrier of information loss between TM and link loads. Rather than exploring the traffic feature to compensate for the lost information, we propose to separate the lost information from TM and purposefully learn such lost information through a designed model. To achieve this goal, we first demonstrate that TM can be decomposed into two orthogonal components X_A and X_N under an unknown routing matrix $\mathbf{A}$. Then, we show that the two orthogonal components are one-to-one correspondence[1] to link loads Y and a latent vector Z, respectively. Due to the "orthogonality" of the two components, the variations of Y will not affect the distributions of Z, which provides us an opportunity to design a DL-based model to collaboratively learn the feature of Z while approaching the bijective function between X and Y.

Following the above novel idea, we develop a new TM estimation method, named FlowTM, based on flow-based generative models [16] [17], which were originally proposed to transform a simple distribution to a complex distribution. What sets the flow-based generative model apart from other DL models is its unique ability to maintain a bijective transformation, implying no information loss occurs during the transformation. Capitalizing on this capability, we design a flow-based learning network that can approach an invertible transformation from TM to link loads while capturing the lost information with a designed loss function. To the best of our knowledge, this is the first time that a flow-based model has been utilized to address the TM estimation problem. It is a compelling discovery that FlowTM can significantly elevate the current state-of-the-art TM estimation accuracy, even in the absence of the routing matrix.

In summary, this paper makes the following contributions to the field of NT-based TM estimation.

[1]One-to-one correspondence means given an instance of Y (or Z) X_A (or X_N) can be uniquely determined.

(1) We demonstrate that the lost information in the Traffic Matrix (TM) is separable. By first decomposing the TM into two orthogonal components, we identify one component with a one-to-one correspondence to the link loads, while the other corresponds to the lost information. Based on the separation, we provide theoretical proof that the invertible transformation from TM to link loads can be learned unbiasedly, even without knowledge about the routing matrix.

(2) We introduce a novel flow-based TM estimation model, FlowTM, that effectively estimates the TM from low-cost link loads. The model comprises an embedding network and a flow-based network. The embedding network scales the TM for processing without altering its original feature, while the flow-based network targets the invertible transformation from the embedding to the link loads, simultaneously learning the distribution of the lost information.

(3) We compared FlowTM with six representative baselines using two real-world datasets. Even without knowledge of the routing matrix, FlowTM significantly outperforms all baselines in deterministic and noisy routing scenarios regarding estimation accuracy and distribution similarity.

II. Related Work

TM estimation has been widely studied over two decades [18] and still attracts much attention in recent years due to its significant practical meaning. Existing methods can be roughly classified into two distinct categories.

The first category uses partial TM measurements to recover the missed flows in TM based on matrix/tensor completion [6] [7] [19] [20]. These methods suppose TM has a low rank due to the spatio-temporal correlations among flows and use matrix decomposition (such as SVD decomposition) or principal component analysis (PCA) to recover the missing volumes based on the known volumes in TM [21]. Zhang et al. [6] proposed a sparsity regularized SVD method to estimate the missing values by matrix factorization. Roughan et al. [22] improved [6] by proposing sparsity regularized matrix factorization (SRMF). Xie et al. [19] [20] [23] [24] proposed tensor completion approaches to recover the missing measurements based on Tucker decomposition. The method proposed in [7] used a GAN-based model to learn the spatio-temporal structure of TM and recovered the missed flows based on the learned structure. However, the cost of these methods may be high as the estimation accuracy depends on the number of known flows in TM. When the number of known flows is low, or the spatial-temporal structure is insignificant, the accuracy of these methods would suffer. NeuTomography [25] employs deep neural networks-based "NT" and operates without a routing matrix, but it significantly differs from our approach. NeuTomography learns the correlations between OD pairs and path performance metrics, utilizing end-to-end measurements of partial OD pairs. Consequently, although it bears the name NeuTomography, its methodology aligns more closely with matrix/tensor completion techniques.

The second category is network tomography (NT) based methods, which estimate TM from the link loads by solving the linear equations in Eqn (1) [10]. Since the linear system is generally rank deficient, traditional NT-based methods impose additional assumptions on the TM to obtain a unique solution. For example, Vardi [10] assumed that the traffic followed Poisson distribution, and Zhang et al. [8] imposed a gravity model on the TM. The accuracy of these methods heavily relies on the reliability of these assumptions.

Recently, a few works proposed to use DL models to automatically learn the traffic feature of TM from a group of training data and search for an optimal solution to Eqn. (1) that satisfies the learnt feature [12] [14]. Jiang et al. [26] first introduced a typical DL model with linear and sigmoid layers to learn the inverse mapping from link loads to OD flows. MNETME proposed in [27] improved the former methods by extending the inputs to incorporate the routing information and adjusting the outputted TM to be consistent with Eqn. (1). The method in [12] used a variational autoencoder (VAE) to learn the latent distribution that is "similar" to TM. It then generated an estimated TM with the learned distribution and adjusted the outputs by Eqn. (1). The method proposed in [28] estimated the OD flows that can satisfy both the linear equations and a particular distribution learned by generative adversarial networks (GAN). Compared with traditional TM estimation methods, DL-based methods show state-of-the-art performance on TM estimation without requiring any distribution or structural assumptions. Nevertheless, as mentioned above, almost all NT-based methods can only work well under predefined and static routing information, which is unrealistic in modern computer networks.

The flow-based generative model was first proposed to transform a simple distribution into a complex distribution [16] [17] [29]. As the transformation is invertible, the flow-based model can preserve all information on the distribution during the transformation. With this powerful ability, flow-based models have been widely applied in various applications. For example, Kumar et al. [30] designed a flow-based model for video prediction. Prenger et al. [31] used a flow-based model for speech synthesis. Xiao et al. [32] leveraged a flow-based model for image rescaling.

Different from all the above works, in this paper, we design a novel flow-based model (named FlowTM) for TM estimation. It approaches the invertible transformation from the complex TM to low-cost link loads even when the routing matrix is absent.

III. Problem and Background

A. Problem Formulation

We consider a backbone network $G(V, E)$ with $|V|$ end nodes and $|E|$ directed links. The traffic matrix between all end nodes can form a $|V| \times |V|$ matrix $\mathbf{X}$, whose entry x_{ij} denotes the traffic flow from the i-th node to the j-th node. To facilitate calculations, the traffic matrix $\mathbf{X}$ is often reshaped by a vector $X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^{n\times 1}$, where $n = |V| \times |V|$. Let $Y = \{y_1, y_2, \ldots, y_m\} \in \mathbb{R}^{m\times 1}$ denote the vector of link loads which can be obtained through SNMP tools. $\mathbf{A} \in \mathbb{R}^{m\times n}$ represents the routing matrix. With a deterministic routing policy, each entry a_{ij} of $\mathbf{A}$ has a binary value (1 or 0). If the j-th flow traverses the i-th directed link, $a_{ij} = 1$; otherwise, $a_{ij} = 0$. For the network with probabilistic routing policy, the a_{ij} value is within the $[0, 1]$ range, representing the probability that the j-th flow may traverse the i-th link. As mentioned earlier, an increasing number of networks apply adaptive routing policies, and thus, the true routing matrix is often unknown. Since in most cases the number of flows n is much greater than the number of link loads m, the linear system is highly rank-deficient. 高维向低维转换，信息丢失

The problem we aim to address in this paper is: given a group of historical measurements for TM X and link loads Y, learn an invertible transformation from X to Y when the routing matrix $\mathbf{A}$ is absent.

B. Flow-Based Generative Network

The principle of the flow-based generative model is built on the change of variable theorem [33]. Suppose z is a random variable with distribution $q(z)$ and $x = f(z)$ is an invertible function that connects variables z and x. Then, it is possible to compute the probability density function of x by

$$p(x) = q(z)|det(\frac{dz}{dx})| = q(f^{-1}(x))|det(\frac{df^{-1}}{dx})|, \quad (2)$$

where $det(\frac{df^{-1}}{dx})$ is the determinant of the Jacobian matrix of $f^{-1}(x)$. Eqn. (2) indicates if the mapping function $x = f(z)$ is bijective, the distributions of x and z can be transformed to each other.

It is impractical to use just one mapping function to transform a simple distribution into a complex one. Flow-based model applies normalizing flows to transform the distribution step by step using a sequence of functions $\{f_1, f_2, \ldots, f_K\}$:

$$\begin{aligned} q_i(z_i) =& q_{i-1}(z_{i-1})|det(\frac{f_i^{-1}}{dz_i})| \\ =& q_{i-1}(z_{i-1})\frac{1}{|det(\frac{df_i}{dz_{i-1}})|}. \end{aligned} \quad (3)$$

Taking logarithm on both sides, Eqn. (3) is equivalent to

$$\log q_i(z_i) = \log q_{i-1}(z_{i-1}) - \log|det(\frac{df_i}{dz_{i-1}})|. \quad (4)$$

Finally, the complex distribution $p(x)$ can be computed by

$$\log p(x) = \log q_0(z_0) - \sum_{i=1}^{K} \log|det(\frac{df_i}{dz_{i-1}})|. \quad (5)$$

To ensure Eqn. (5) is computable, two conditions must be satisfied: (1) all mapping functions should be invertible; (2) the determinants of the Jacobian matrices should be easy to compute.

One of the most classical models for normalizing flows is Real-valued Non-Volume Preserving (RealNVP) [17], which formulates the transformation functions by a sequence of affine coupling layers. The input of the affine coupling layer $z^l \in \mathbb{R}^D$

is split into two parts: $z^l_{1:d}$ and $z^l_{d+1:D}$, which undergo the additive affine transformations:

$$\begin{aligned} z^{l+1}_{1:d} &= z^l_{1:d}, \\ z^{l+1}_{d+1:D} &= z^l \odot exp(s(z^l_{1:d})) + t(z^l_{1:d}), \end{aligned} \tag{6}$$

where $s(\cdot)$ and $t(\cdot)$ can be arbitrary functions and $\odot$ denotes element-wise product. The inverse transformation of Eqn. (6) is straightforward:

$$\begin{aligned} z^l_{1:d} &= z^{l+1}_{1:d}, \\ z^l_{d+1:D} &= (z^{l+1}_{d+1:D} - t(z^l_{1:d})) \odot exp(-s(z^l_{1:d})). \end{aligned} \tag{7}$$

The Jacobian matrix is a lower triangular matrix:

$$\mathbf{J} = \begin{bmatrix} \mathbb{I}_d & \mathbb{O}_{d\times D-d} \\ \frac{\partial z^{l+1}_{d+1:D}}{\partial z^l_{1:d}} & diag(exp(s(z^l_{1:d}))) \end{bmatrix}. \tag{8}$$

Hence, the determinant of the Jacobian matrix can be easily computed by the production of the diagonal.

IV. Methods

We first demonstrate that the lost information in Eqn. (1) can be separated from TM. Then, we present the design of FlowTM that correlates TM with link loads by invertible neural networks while purposefully learning the lost information by a latent vector.

A. Orthogonal Decomposition of TM

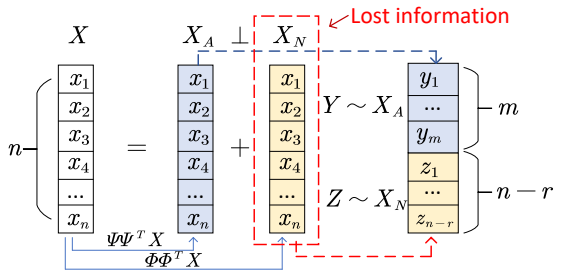


Fig. 1. The orthogonal decomposition of TM: X can be decomposed into X_A and X_N which are orthogonal to each other. X_A is one-to-one correspondence to Y while X_N is the lost information that can be one-to-one correspondence to a latent Z.

Before decomposing the TM X, we first decompose the routing matrix $\mathbf{A}$ through singular value decomposition (SVD).

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T, \tag{9}$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are orthogonal matrices, $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ is a rectangular diagonal matrix with positive singular values of $\mathbf{A}$ in a descending order. Let r denote the rank of matrix $\mathbf{A}$. According to the properties of SVD, the orthogonal matrix $\mathbf{V} = [\Psi, \Phi]$, where $\Psi = \{\psi_1, \psi_2, \ldots, \psi_r\} \in \mathbb{R}^{n\times r}$ is the group of the orthogonal basis for the row space of $\mathbf{A}$ while $\Phi = \{\phi_1, \phi_2, \ldots, \phi_{n-r}\} \in \mathbb{R}^{n\times(n-r)}$ is the orthogonal basis for the null space of $\mathbf{A}$. We represent the null space of $\mathbf{A}$ by $\mathcal{N} \triangleq Null(\mathbf{A})$, where $\forall N \in \mathcal{N}$, it has $\mathbf{A}N = 0$.

Theorem 1: *Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\forall X \in \mathbb{R}^{n\times 1}$, we can decompose X by $X = X_A + X_N$, where $X_A = \Psi\Psi^T X$ and $X_N = \Phi\Phi^T X$, X_A and X_N are orthogonal to each other.*

Proof: First, X_A and X_N are orthogonal because Ψ and Φ are orthogonal:

$$\begin{aligned} X_A^T \cdot X_N =& (\Psi\Psi^T \cdot X)^T \cdot \Phi\Phi^T \cdot X \\ =& X^T \cdot \Psi(\Psi^T\Phi)\Phi^T \cdot X = 0. \end{aligned} \tag{10}$$

Second,

$$\begin{aligned} X_A + X_N =& \Psi\Psi^T X + \Phi\Phi^T X \\ =& (\Psi\Psi^T + \Phi\Phi^T) \cdot X \\ =& \mathbf{V}\mathbf{V}^T \cdot X = X. \quad \square \end{aligned} \tag{11}$$

The orthogonal decomposition of X can be considered as mapping X onto two orthogonal space $span(\Psi\Psi^T)$ and $span(\Phi\Phi^T)$, respectively. X_A and X_N can be considered as the images of X on the two spaces, respectively. Next, we demonstrate that the information of X_A and X_N can be exactly encoded in link loads Y and a latent vector Z, respectively.

Theorem 2: *For matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ with rank r and $X \in \mathbb{R}^{n\times 1}$, if $Y = \mathbf{A}X$ and $Z = \Phi^T X$, where $Y \in \mathbb{R}^{m\times 1}$ and $Z \in \mathbb{R}^{(n-r)\times 1}$, respectively, then vector X_A is one-to-one correspondence to Y while X_N is one-to-one correspondence to Z.*

Proof: Since $X = X_A + X_N$, we left multiply both sides of the equation by $\mathbf{A}$, then

$$\mathbf{A}X = \mathbf{A}X_A + \mathbf{A}X_N \tag{12}$$

As $X_N = \Phi\Phi^T X$, and Φ is the group of basis for null space $\mathcal{N}$, we have $\mathbf{A}X_N = \mathbf{A}\Phi\Phi^T X = 0$. Therefore,

$$\mathbf{A}X_A = \mathbf{A}X = Y. \tag{13}$$

Decomposing $\mathbf{A}$ by SVD, Eqn. (13) can be written by

$$Y = \mathbf{A}X_A = \mathbf{U}\Sigma\mathbf{V}^T X_A \tag{14}$$

Recall that $\mathbf{V}^T = [\Psi, \Phi]$. As $\Psi^T\Psi = 1$, and $\Phi^T\Psi = 0$, Eqn. (14) can be rewritten by

$$\begin{bmatrix} \Sigma^{-1}_{\le r}\mathbf{U}^T_{\le r} \cdot Y \\ \mathbb{O}_{(n-r)\times 1} \end{bmatrix} = \begin{bmatrix} \Sigma^{-1}_{\le r}\bar{Y} \\ \mathbb{O}_{(n-r)\times 1} \end{bmatrix} = \mathbf{V}^T X_A, \tag{15}$$

where $\Sigma_{\le r} \in \mathbb{R}^{r\times r}$ is the diagonal matrix with the r positive singular values of $\mathbf{A}$, and $\Sigma^{-1}_{\le r}$ is its inverse matrix. $\mathbf{U}^T_{\le r} \in \mathbb{R}^{r\times m}$ is the first r rows of $\mathbf{U}^T$. $\mathbb{O}_{(n-r)\times 1}$ is an all-zero vector.

It is clear that as $\mathbf{V}^T$ has a full rank, X_A and $\bar{Y} = \mathbf{U}^T_{\le r}Y$ are bijective through Eqn. (15). Next, we prove Y and $\bar{Y}$ are also bijective.

Similar with $\mathbf{V}$, $\mathbf{U}^T = [\Psi', \Phi']$, where $\Psi' = \mathbf{U}^T_{\le r}$ is a group of orthogonal basis for the column space of $\mathbf{A}$, and Φ' is the basis for the null space $Null(\mathbf{A}^T)$. Because for all possible X, Y always lies in the column space of $\mathbf{A}$. Therefore, we have

$$\mathbf{U}^T Y = \begin{bmatrix} \Psi' Y \\ \mathbb{O}_{m-r} \end{bmatrix} = \begin{bmatrix} \bar{Y} \\ \mathbb{O}_{m-r} \end{bmatrix}. \tag{16}$$

As $\mathbf{U}^T$ has a full rank, Y and $\bar{Y}$ are bijective, which means Y is one-to-one correspondence to both $\bar{Y}$ and X_A.

Similarly, we left multiply both sides of $X = X_A + X_N$ by Φ^T, then

$$\Phi^T X = \Phi^T X_A + \Phi^T X_N = \Phi^T X_N = Z \tag{17}$$

We decompose Φ^T by $\Phi^T = \mathbf{U}'\Sigma'\mathbf{V}'^T$. As the rank of Φ^T is $n-r$, there are $n-r$ non-zero singular values in Σ'. As the null space of Φ^T is $span(\Psi)$, we have $\mathbf{V}' = [\Phi, \Psi]$. Therefore, Eqn. (17) can be rewritten by

$$\begin{bmatrix} \Sigma'^{-1}_{\leq n-r}\mathbf{U}'^T \cdot Z \\ \mathbb{O}_{r\times 1} \end{bmatrix} = \mathbf{V}'^T X_N, \tag{18}$$

where $\Sigma'_{\leq n-r}$ is the first $n-r$ columns of Σ' with $n-r$ positive singular values of $\mathcal{N}$, and $\Sigma'^{-1}_{\leq n-r}$ is the inverse. As $\mathbf{V}'$ and $\mathbf{U}'$ have full ranks, Z and X_N in Eqn. (18) are one-to-one correspondence.□

According to Theorem 1 and Theorem 2, the two orthogonal components of X can be exactly encoded into the link load vector Y and the latent vector Z, respectively. That means when transforming TM into the link loads, Z owns all lost information in Eqn. (1). Due to the "orthogonality" of the two components, the variations on Y will not affect the assignments of Z. Next, a flow-based TM estimation model (FlowTM) is designed based on the decomposition of X.

B. Overview

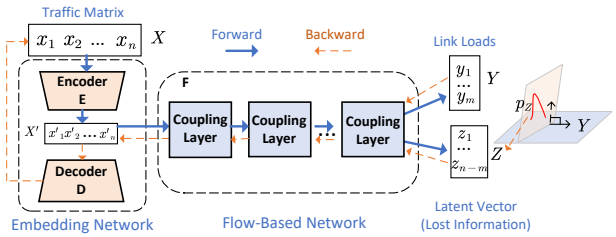


Fig. 2. Framework of FlowTM: FlowTM includes an embedding network and a flow-based network. With the designed loss function, FlowTM collaboratively approaches the bijective correlation between Y and X and learns the lost information Z.

The flow-based TM estimation model (FlowTM) framework is presented in Fig. 2. FlowTM comprises two networks: an embedding network composed of an encoder E and a decoder D, and a flow-based network F. The embedding network embeds the TM data onto an accessible learning scale while the flow-based network learns the invertible transformation from the embedded vector to the link loads. Note that, the dimension of the latent vector Z is set to $n-m$ instead of $n-r$ in FlowTM to ensure the input and output have the same dimensions. Actually, under a relatively complex routing (e.g. probabilistic or adaptively routing), the number of link loads m is usually equal to or slightly bigger than the rank of routing matrix r. The dimension difference (if any) between $n-m$ and $n-r$ has little impact on the learning results for Z.

In the forward process, TM X is firstly inputted into the encoder D. Then D outputs an embedding vector X', and feeds it to the flow-based network F. Afterwards, F transforms X' into the link load Y and a sample of Z through multiple coupling layers. In the backward process, the link load Y accompanied by a sample from the latent distribution is inputted into the right side of F. Then, F inversely transforms Y into the embedding X'. Finally, the embedding network recovers the TM X from X' through decoder D.

Next, we will explain three key designs that ensure the good performance of FlowTM: (1) The embedding network mitigates the skewness of TM while keeping its original features; (2) The flow-based network approaches an invertible transformation while learning the lost information; (3) Training with designed loss function ensures the lost information are learnable when the routing matrix is absent.

C. Embedding Network

Different from images, the TM data presents extremely skewed values, which will easily cause oversized parameters in the flow-based network. The most common way to mitigate the skewness of data is normalization or centralization (such as Min-Max or Z-score). However, these methods can not work well for TM data because many traffic volumes are polarizedly distributed: sometimes, most flows are close to zero, while a few elephant flows present giant volumes. In such a case, the small flows would be overwhelmed by the elephant flows by Min-Max normalization, and the normalized TM data would present a distorted distribution.

To scale TM data onto an accessible scale while keeping the true traffic pattern, we propose learning a TM representation using an encoder E instead of common normalization. The encoder network E contains multiple fully-connected layers and a Tanh activation layer, ensuring the values in the latent vector are within the range of $[-1, 1]$. To prevent information loss during the transformation, the embedding vector $X' \in \mathbb{R}^{n\times 1}$ has the same dimension as X. The embedding network also contains a decoder network D, which is responsible for supervising the encoder's training and restoring X' to the original input space. We pre-trained the embedding network by minimizing the loss function L_{emb} to ensure the embedding network is approximately invertible.

$$L_{emb} = ||X - D(E(X))||_2^2, \tag{19}$$

where $||\cdot||_2$ denotes the L_2 norm.

D. Flow-Based Network

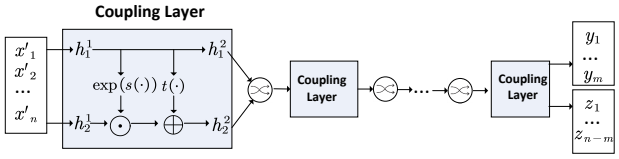


Fig. 3. Structure of flow-based network in FlowTM: The network comprises a sequence of coupling layers linked by shuffle operations. Each coupling layer splits the input into two parts and learns an invertible transformation using parameterized $s(\cdot)$ and $t(\cdot)$.

The flow-based network F contains a sequence of invertible coupling layers (as shown in Fig. 3), each applying a RealNVP transformation. In the first coupling layer, the embedding

vector X' is split into two parts: h_1^1 and h_2^1. The output h_1^2 and h_2^2 are computed by

$$h_1^2 = h_1^1, \text{ and } \quad h_2^2 = h_2^1 \odot \exp\left(s(h_1^1)\right) + t(h_1^1),$$

respectively. The two functions $s(\cdot)$ and $t(\cdot)$ are parameterized with two neural networks. The inverse mapping from output (h_1^2, h_2^2) to input (h_1^1, h_2^1) can be implemented by

$$h_1^1 = h_1^2, \text{ and } \quad h_2^1 = (h_2^2 - t(h_1^1)) \odot \exp\left(-s(h_1^1)\right),$$

respectively. There is a shuffle operation between two adjacent coupling layers. To avoid the violation of inversibility, we design the shuffle operation as follows: the output (h_1^l, h_2^l) is first reshaped to a matrix $\mathbf{H}^l$ whose dimension is $\sqrt{n} \times \sqrt{n}$. Then $\mathbf{H}^l$ is transposed and reshaped back to the dimension of $n \times 1$. The inverse shuffle operation is just the same as the forward shuffle operation. The next coupling layer takes the shuffled vector as input and splits it into two parts, just as the first coupling layer does. The final output of the last coupling layer is split into $Y = \{y_1, y_2, \ldots, y_m\}$ and $Z = \{z_1, z_2, \ldots, z_{n-m}\}$, respectively, where Y is corresponding to the link loads, and Z is a latent vector representing the lost information in TM. With the designed structure, network F is strictly invertible, i.e., $X' = F^{-1}(F(X')) = F^{-1}([Y, Z])$.

E. Model Training

The training of FlowTM faces two issues. Firstly, as the actual routing matrix is unknown, the latent vector Z lacks a specific training target. Secondly, storing the lost information for all possible TMs by Z is impractical. To address the two issues, we first demonstrate that the absent training target can be achieved indirectly through distribution constraints. Then we incorporate the lost information into the model parameters and transform the latent vector into a simple distribution.

Let $p_X(x)$ denote the prior distribution of X, $p_Y(y)$ and $p_Z(z)$ are the distributions of Y and Z, respectively.

Theorem 3: *Suppose $[Y, Z] = f(X)$ is a bijective function. Suppose $\forall X \sim p_X(x)$ the output $Y = f_Y(X) = \mathbf{A}X$. If the latent vector $Z = f_Z(X)$ is independent to Y, i.e., $p(y, z) = p_Y(y)p_Z(z)$, then Z is one to one correspondence to the lost information in X.*

Proof: As $[Y, Z] = f(X)$ is a bijective function, the output $[Y, Z]$ is one-to-one correspondence to X. When Y and Z are independent of each other, Y and Z will not encode duplicate information in X. According to Theorem 1, X can be orthogonally decomposed by $X = X_A + X_N$. If $\forall X \sim p_X(x)$ the output $Y = f_Y(X) = \mathbf{A}X$, Y are one-to-one correspondence to X_A. As X_A and X_N are orthogonal, they do not have overlapped information in X. That means Z is one-to-one correspondence to X_N, which is the lost information in X.□

Substituting $f(X)$ in Theorem 3 by FlowTM, the results indicate that if the following three conditions can be satisfied, the lost information in TM can be learned unbiasedly without knowing the routing matrix $\mathbf{A}$: (1) FlowTM is invertible, (2) the output Y is always equal to the corresponding link loads, and (3) Y and Z are independent to each other.

To enable FlowTM to learn the lost information for all possible TMs (i.e., $\forall X \sim p_X(x)$), we incorporate the information of Z into the model parameters and transform Z into a simple distribution (e.g. a normal distribution) through FlowTM. Doing so has double benefits: Firstly, the lost information can be easily synthesized by drawing samples from the distribution. Secondly, the lost information can be case-independent without requiring all possible TMs. Experimental results in Sec. V-E show that although the lost information is synthesized from 1000 random samples, the posterior distribution of the estimated TM is extremely sharp when inputting an instance of link loads. That means FlowTM can effectively transform the random samples into the lost information, which can uniquely determine the TM with link loads by high confidence.

Based on the above results, besides the embedding loss for E and D, and the negative log-likelihood loss (Eqn. (5)) for F, we design four additional training objectives for the whole FlowTM model, where the first three are corresponding to the three conditions in Theorem 3, and the last one is corresponding to the ultimate application of FlowTM.

1) Objective 1, Invertibility Loss: To enhance the invertibility of FlowTM, the invertibility loss penalizes the deviations between the input X and the inverse output $f^{-1}(f(X))$, i.e.,

$$\begin{aligned} L_{inv} =& ||f^{-1}(f(X)) - X||_2^2 \\ =& ||D(F^{-1}(F(E(X)))) - X||_2^2. \end{aligned} \tag{20}$$

2) Objective 2, Link-Load Loss: The link-load loss penalizes the deviations between the output $f_Y(X)$ and the measured link loads Y, i.e.,

$$\begin{aligned} L_{link} =& ||f_Y(X) - Y||_2^2 \\ =& ||F_Y(D(X)) - Y||_2^2. \end{aligned} \tag{21}$$

3) Objective 3, Independent Loss: The independent loss is used to force the distributions of the outputs $f_Z(X)$ and $f_Y(X)$ to be independent of each other, i.e. $p(y, z) = p_Y(y)p_Z(z)$. To implement the independent loss, the joint distribution of $p(y, z)$ is transformed into $p(f(x))$. Then, the loss function can be formulated by

$$\begin{aligned} L_{indep} =& \ell\left(p(f(x)), p_Y(y)p_Z(z)\right) \\ =& \ell\left(p(F(E(x))), p_Y(y)p_Z(z)\right), \end{aligned} \tag{22}$$

where $\ell(\cdot, \cdot)$ is a metric of the discrepancy of two distributions. $p_Y(y)$ is the link load distribution, while $p_Z(z)$ is a normal distribution. In our experiments, the discrepancy of the two distributions is measured by maximum mean discrepancy (MMD) [34], which compares two distributions by the samples based on a kernel function.

4) Objective 4, Estimation Loss: The estimation loss penalizes the deviations between the estimation of TM $f^{-1}(Y, Z)$ and the true TM X, where Y is a measurement of link loads and Z is a sample drawn from a normal distribution:

$$\begin{aligned} L_{est} =& ||f^{-1}(Y, Z) - X||_2^2 \\ =& ||D(F^{-1}([Y, Z])) - X||_2^2. \end{aligned} \tag{23}$$

5) Total Objective: FlowTM is trained based on the total loss function:

$$L_{FlowTM} = \lambda_1 L_{inv} + \lambda_2 L_{link} + \lambda_3 L_{indep} + \lambda_4 L_{est}, \quad (24)$$

where λ_1, λ_2, λ_3 and λ_4 are hyperparameters for balancing the weight of the four losses.

V. Experiments

A. Datasets

数据集

We use two real-world traffic datasets to validate the performance of our method: Abilene [35] and GÉANT [36]. Abilene contains 12 routers, 30 directed inner links and 24 outside links. The total size of the dataset is 48384×144, where 48384 is the number of TM instances and 144 is the number of flows within a TM. We use the instances during the first 15 weeks (total 30240 instances) as the training data and use the instances in the 16-th week (total 2016 instances) as the testing data. GÉANT network contains 23 routers and 120 directed links. The total size of the dataset is 10772×529. We use the first 10 weeks' collections (total 6720 instances) as the training data, and the 11-th week's collections (total 672 instances) as the testing data. All training instances were divided by the maximum value of link loads in the datasets.

B. Metrics

评价指标

Referring the mainstreaming metrics that were used in [7] [12] [27], the estimation accuracy of all methods is qualified by the normalized mean absolute error (NMAE), normalized root mean square error (NRMSE), temporal related mean absolute error (TRMAE), and spatial related mean absolute error (SRMAE). The smaller these metrics, the higher the estimation accuracy. We use an additional metric MMD [34] to assess the similarity between the estimation distribution and the true TM distribution. The smaller the MMD result, the more evident that the two distributions are the same.

C. Baselines

We compare our FlowTM with six representative TM estimation methods grouped into three categories.

The first category is the correlation-learning method. MNETME [27] learns the correlations between link loads and TM through a BPNN model that incorporates the routing matrix. It adjusts the outputted TM estimation to satisfy the linear equations. We modify MNETME to be a routing-matrix-free version (denoted by BPNN), whose model do not incorporate the routing matrix, and there are no further adjustments on the outputs.

The second category is the generative-based method. We first implemented a GAN-based model [7] [28] to learn the distribution of TM and then searches for a solution from Eqn. (1) that most satisfy the learnt distribution. We also implemented a VAE-based method [12] [37], which first learns a latent distribution of TM by VAE and then samples an estimation of TM from the latent distribution that most satisfies Eqn. (1). As current generative-based models can only learn the feature of TM, they must incorporate the linear equations (1) to estimate a TM based on link loads. Therefore, they do not have a routing-matrix-free version.

The last kind of baseline is the traditional method. Tomogravity [8] assumes the traffic in TM follows a gravity model and then uses the least square method to find a solution of Eqn. (1) that conforms to the gravity model. We modify Tomogravity to a routing-matrix-free version (denoted by Gravity), whose estimations only satisfy the gravity model.

D. Model Details

The detailed settings for the embedding and flow-based networks in FlowTM are listed in Table I. All experiments were conducted on a server with 16-core CPUs at 2.6GHz and 16GB GPU. The results were averaged over 10 runs.

TABLE I
HYPERPARAMETERS FOR FLOWTM

#FC Layers in Embedding	2 (Encoder), 1 (Decoder)
Non-Linearity in Embedding	Tanh (Encoder), Sigmoid (Decoder)
#Coupling Layers in Flow	3
#FC Layers in Flow	2 (both $s(\cdot)$ and $t(\cdot)$)
Non-Linearity in Flow	LeakyReLu (both $s(\cdot)$ and $t(\cdot)$)
Decaying Learning Rate	$10^{-3} \sim 10^{-5}$
Optimizer	Adam [38]
Dropout	0.0
#Batch	32
#Training Epoch	100
λ_1, λ_2, λ_3, λ_4	2, 2, 0.5, 10

E. Results

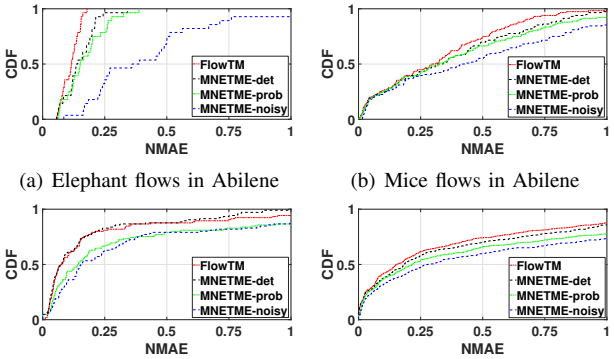
1) Performance under Different Routing Scenarios: We evaluate the seven methods under three routing scenarios: deterministic routing, probabilistic routing and noisy routing. Deterministic routing is the simplest routing scenario, where all OD flows abide by a fixed routing path throughout all measurement times. In the probabilistic routing scenario, flows from one source node may go through multiple routing paths to the destination node with a predefined probability. The last scenario is noisy routing, which is the most realistic scenario. In that scenario, routers adaptively select paths for OD flows, according to the current network loading. That means, the known routing matrix may not be the ground truth. We provide each method with a probabilistic routing matrix to simulate the noisy routing scenario where the probabilities are not the true values. As FlowTM, BPNN and Gravity (whose names are in bold in Table II) do not incorporate a routing matrix, they have the same results in probabilistic and noisy routing scenarios.

Table II lists the NMAE, NRMSE and MMD of all methods under three routing scenarios. From the table, FlowTM has significant superiority over other baselines under all scenarios. All methods perform the best in deterministic routing due to the simple correlations between TM and link loads. In the probabilistic scenario, the accuracies of the seven methods present varying degrees of decline as the correlations become more complex. In the noisy routing scenario, methods that incorporate Eqn. (1) encounter a significant degradation in terms of all metrics. That is because the noisy routing matrix misguides the adjusting of the outputs to bias solutions.

TABLE II
COMPARISON OF THE SEVEN METHODS IN THREE ROUTING SCENARIOS

Data		Abilene			GÉANT		
Metrics	Methods	Deterministic Routing	Probabilistic Routing	Noisy Routing	Deterministic Routing	Probabilistic Routing	Noisy Routing
NMAE	**FlowTM**	**0.1907**	**0.1928**	**0.1928**	**0.2666**	**0.2735**	**0.2735**
	MNETME	0.2313	0.2563	0.4689	0.2970	0.3597	0.5189
	BPNN	0.4131	0.4569	0.4569	0.4528	0.4669	0.4669
	GAN	0.3438	0.3544	0.6384	0.3953	0.3956	0.4893
	VAE	0.3953	0.4350	0.5086	0.4983	0.5771	0.6118
	Tomogravity	0.5214	0.6908	0.8736	0.6850	0.7972	0.9512
	Gravity	0.8193	1.3684	1.3684	0.8759	2.3351	2.3351
NRMSE	**FlowTM**	**0.1673**	**0.1715**	**0.1715**	**0.3589**	**0.3795**	**0.3795**
	MNETME	0.2073	0.2242	0.4969	0.3763	0.4322	0.5065
	BPNN	0.4094	0.4410	0.4410	0.5356	0.5384	0.5384
	GAN	0.2947	0.3130	0.6400	0.3885	0.3979	0.4584
	VAE	0.3726	0.3918	0.4630	0.5469	0.5340	0.5433
	Tomogravity	0.4047	0.6886	0.6901	0.4381	0.7591	0.7982
	Gravity	0.5771	1.1737	1.1737	0.6190	1.1676	1.1676
MMD	**FlowTM**	**0.0059**	**0.0068**	**0.0068**	**0.0463**	**0.0478**	**0.0478**
	MNETME	0.0092	0.0112	0.0454	0.0593	0.0677	0.0905
	BPNN	0.0206	0.0226	0.0226	0.0739	0.0746	0.0746
	GAN	0.0204	0.0182	0.0250	0.0751	0.0754	0.0793
	VAE	0.0813	0.0819	0.0916	0.1231	0.1194	0.1325
	Tomogravity	0.0289	0.0506	0.0644	0.0598	0.1148	0.1164
	Gravity	0.0927	0.1890	0.1890	0.0758	0.2510	0.2510

Baselines that do not incorporate a routing matrix perform generally worse than the other baselines in deterministic and probabilistic routing scenarios since they lack critical information compared with other methods. All methods perform inferiorly in GÉANT because TMs in this dataset have much higher dimensions and the training data is also insufficient. Overall, in Abilene (GÉANT), the estimation accuracy of FlowTM outperforms the best baseline by 18% (10%) in the deterministic scenario, 25% (24%) in the probabilistic scenario and 58% (41%) in the noisy scenario. Furthermore, the MMD of FlowTM is far below the baselines, which means the distribution of the estimated TM is much more similar to the true distribution of TM.

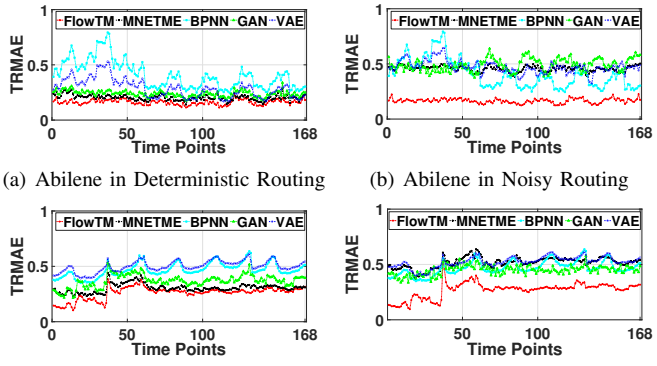


(a) Elephant flows in Abilene (b) Mice flows in Abilene

(c) Elephant flows in GÉANT (d) Mice flows in GÉANT

Fig. 4. Cumulative distribution of NMAE: Flows in TM are separated into two groups: elephant and mice. Flows with the top-20% volumes are labelled as elephant flows, whereas the remaining 80% flows are mice flows.

Fig. 4 plots the cumulative distributions of NMAE of FlowTM and MNETME under three routing scenarios, where MNETME-det, MNETME-prob and MNETME-noisy denote the method under deterministic routing, probabilistic routing and noisy routing, respectively. We only plot MNETME in the figure due to its suboptimal overall performance in Table II. We divide the flows in TM into two groups: elephant and mice, according to their flow volumes. Flows in the elephant group have the top 20% averaged traffic volumes, while the remaining 80% flows are classified into the mice group. From the figures, elephant flows in Abilene are much easier to estimate than that in GÉANT. We can also see a big gap between the curves of noisy routing and deterministic/probabilistic routing in figures of elephant flows. That means the noisy routing matrix has more impact on estimating the elephant flows than on the mice flows.



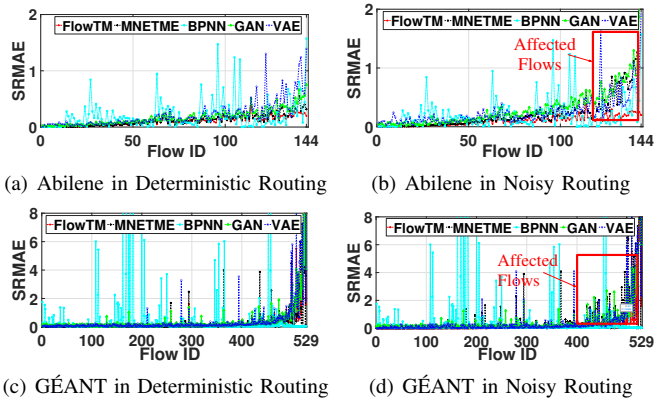
(a) Abilene in Deterministic Routing (b) Abilene in Noisy Routing

(c) GÉANT in Deterministic Routing (d) GÉANT in Noisy Routing

Fig. 5. Temporal estimation errors: the measurement results are aggregated into 168 records (i.e., once per hour).

2) Temporal and Spatial Accuracy: In this section, we evaluate the temporal and spatial accuracies of these methods. As Tomogravity and Gravity do not distinguish training and testing data, we only plot the testing results of the five DL-based methods to show their performance after training.

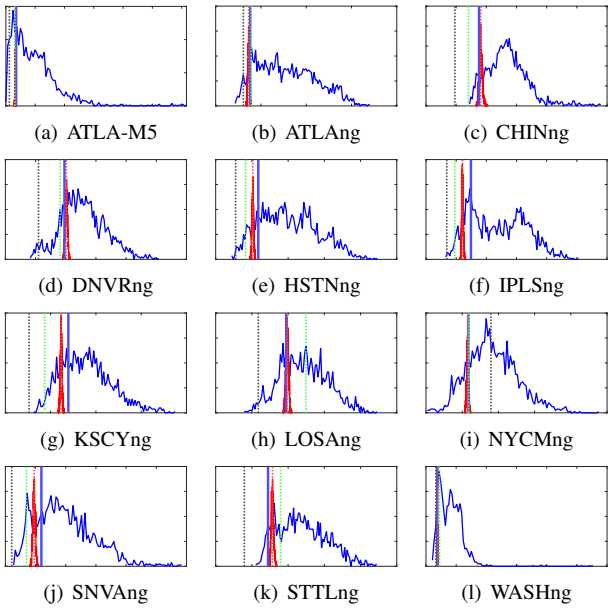
Fig. 5 plots the temporal estimation errors of the five DL-based methods under deterministic and noisy routings, respectively. In both two datasets, the 7-day's samples (2016 samples in Abilene and 627 samples in GÉANT) were aggregated into 168 records (i.e., each result represents the averaged NTRE per hour). In all figures, FlowTM has the least temporal estimation errors. Especially in the noisy routing scenario, there is a significant gap between FlowTM and baseline methods.



(a) Abilene in Deterministic Routing (b) Abilene in Noisy Routing

(c) GÉANT in Deterministic Routing (d) GÉANT in Noisy Routing

Fig. 6. Spatial estimation errors: the flows on the x-axis are sorted by their averaged volumes in ascending order.



(a) ATLA-M5 (b) ATLAng (c) CHINng

(d) DNVRng (e) HSTNng (f) IPLSng

(g) KSCYng (h) LOSAng (i) NYCMng

(j) SNVAng (k) STTLng (l) WASHng

Fig. 7. Prior and posterior distributions of flows going out from NYCMng to the labelled destination: the blue curve denotes the prior distribution of TM while the red curve denotes the posterior distribution of TM learned by FlowTM. The blue vertical line is the ground truth of TM. The red dashed line denotes the estimation of FlowTM. The green and black dashed lines are the estimations of MNETME under deterministic and noisy routings, respectively.

Fig. 6 presents the spatial estimation errors of the methods under deterministic and noisy routings, respectively, where the flows were sorted by their averaged volumes in ascending order. From these figures, all methods perform well in estimating small flows, especially in GÉANT dataset. From Fig. 6(b) and Fig. 6(d), for the routing-incorporated methods, the noisy routing causes estimation deterioration on the top 10% largest flows in Abilene and top 20% largest flows in GÉANT.

3) Effect of Learning the Lost Information: Different from former TM estimation methods, FlowTM not only considers the correlations between TM and link loads but also learns the lost information collaboratively. To evaluate the performance of FlowTM on learning the lost information, we input 1000 samples that were randomly drawn from normal distribution into FlowTM and see how it affects the posterior of TM.

Fig. 7 plots the prior and posterior distributions of 12 flows sourcing from node NYCMng (located in New York) at 8:00 am on 16 July 2004. Each figure in Fig. 7 represents the flow from NYCMng to the labelled node. From the figures, we can learn that (1) the posterior probability of TM outputted by FlowTM is quite sharp, although the inputs (1000 samples) of the lost information are randomly drawn from normal distributions. That indicates FlowTM has effectively learned the features of the lost information; (2) Compared with the estimation of MNETME, the TM estimated by FlowTM is much closer to the ground truth in almost all cases. This indicates, compared with MNETME, which compensates for the lost information by learning the TM feature, FlowTM, which purposefully captures the lost information while learning the bijective transformations between TM and link loads, can significantly improve the estimation accuracy even when the routing matrix is absent.

VI. Conclusion

We presented FlowTM, an innovative and low-cost traffic estimator based on a flow-based generative model. Unlike existing NT solutions—including those based on DL—which operate under the assumption of knowing a routing matrix, either deterministically or probabilistically, FlowTM confronts the challenge of correlating the Traffic Matrix (TM) and link loads using a bijective transformation, even in the absence of a routing matrix. Initially, we demonstrated that the lost information in TM can be orthogonally separated from the link loads. Subsequently, we utilized the invertible coupling layers of the flow-based model to capture this lost information, facilitating an invertible transformation from TM data to link loads. Through extensive experiments on two real-world traffic datasets, we demonstrated that even without using the routing matrix, FlowTM can improve the estimation accuracy of baselines that leverage the routing matrix information. Furthermore, FlowTM shows dominant superiority when the routing matrix information is also absent for baselines. We have provided public access to our code and data at https://github.com/duoduoqiao/FlowTM.

Acknowledgements

This research was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) (No. RGPIN-2018-03896), Anhui Provincial Natural Science Foundation (2008085MF203, 2108085MF202), and National Natural Science Foundation of China (62002097).

REFERENCES

[1] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 1, pp. 36–56, 2008.

[2] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, J. Cao, and D. Zhang, "Fast tensor factorization for accurate internet anomaly detection," *IEEE/ACM transactions on networking*, vol. 25, no. 6, pp. 3794–3807, 2017.

[3] P. Tune, M. Roughan, H. Haddadi, and O. Bonaventure, "Internet traffic matrices: A primer," *Recent Advances in Networking*, vol. 1, pp. 1–56, 2013.

[4] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[5] Y. Tian, W. Chen, and C.-T. Lea, "An sdn-based traffic matrix estimation framework," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1435–1445, 2018.

[6] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and internet traffic matrices," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009, pp. 267–278.

[7] K. Xie, Y. Ouyang, X. Wang, G. Xie, K. Li, W. Liang, J. Cao, and J. Wen, "Deep adversarial tensor completion for accurate network traffic measurement," *IEEE/ACM Transactions on Networking*, 2023.

[8] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale ip traffic matrices from link loads," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1, pp. 206–217, 2003.

[9] L. Ma, T. He, K. K. Leung, D. Towsley, and A. Swami, "Efficient identification of additive link metrics via network tomography," in *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE, 2013, pp. 581–590.

[10] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *Journal of the American statistical association*, vol. 91, no. 433, pp. 365–377, 1996.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[12] G. Kakkavas, M. Kalntis, V. Karyotis, and S. Papavassiliou, "Future network traffic matrix synthesis and estimation based on deep generative models," in *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2021, pp. 1–8.

[13] D. Aloraifan, I. Ahmad, and E. Alrashed, "Deep learning based network traffic matrix prediction," *International Journal of Intelligent Networks*, vol. 2, pp. 46–56, 2021.

[14] L. Nie, D. Jiang, L. Guo, and S. Yu, "Traffic matrix prediction and estimation based on deep learning in large-scale ip backbone networks," *Journal of Network and Computer Applications*, vol. 76, pp. 16–22, 2016.

[15] H. Ikeuchi, H. Saito, and K. Matsuda, "Network tomography based on adaptive measurements in probabilistic routing," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 2148–2157.

[16] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014.

[17] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[18] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 161–174, 2002.

[19] K. Xie, C. Peng, X. Wang, G. Xie, and J. Wen, "Accurate recovery of internet traffic data under dynamic measurements," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[20] K. Xie, R. Xie, X. Wang, G. Xie, D. Zhang, and J. Wen, "Nmmf-stream: A fast and accurate stream-processing scheme for network monitoring data recovery," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 2218–2227.

[21] E. Zhao and L. Tan, "A pca based optimization approach for ip traffic matrix estimation," *Journal of Network and Computer Applications*, vol. 57, pp. 12–20, 2015.

[22] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and internet traffic matrices (extended version)," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 662–676, 2011.

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

[24] Y. Ouyang, K. Xie, X. Wang, J. Wen, and G. Zhang, "Lightweight trilinear pooling based tensor completion for network traffic monitoring," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 2128–2137.

[25] L. Ma, Z. Zhang, and M. Srivatsa, "Neural network tomography," *arXiv preprint arXiv:2001.02942*, 2020.

[26] D. Jiang, X. Wang, L. Guo, H. Ni, and Z. Chen, "Accurate estimation of large-scale ip traffic matrix," *AEU-International Journal of Electronics and Communications*, vol. 65, no. 1, pp. 75–86, 2011.

[27] H. Zhou, L. Tan, Q. Zeng, and C. Wu, "Traffic matrix estimation: A neural network approach with extended input and expectation maximization iteration," *Journal of Network and Computer Applications*, vol. 60, pp. 220–232, 2016.

[28] S. Xu, M. Kodialam, T. Lakshman, and S. S. Panwar, "Learning based methods for traffic matrix estimation from link measurements," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 488–499, 2021.

[29] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *Advances in neural information processing systems*, vol. 31, 2018.

[30] M. Kumar, M. Babaeizadeh, D. Erhan, C. Finn, S. Levine, L. Dinh, and D. Kingma, "Videoflow: A flow-based generative model for video," *arXiv preprint arXiv:1903.01434*, vol. 2, no. 5, p. 3, 2019.

[31] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3617–3621.

[32] M. Xiao, S. Zheng, C. Liu, Z. Lin, and T.-Y. Liu, "Invertible rescaling network and its extensions," *International Journal of Computer Vision*, vol. 131, no. 1, pp. 134–159, 2023.

[33] S. Kamefuchi, L. o'Raifeartaigh, and A. Salam, "Change of variables and equivalence theorems in quantum field theories," *Nuclear Physics*, vol. 28, no. 1, pp. 529–549, 1961.

[34] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[35] "Abilene network topology data and traffic traces," https://www.cs.utexas.edu/ yzhang/research/AbileneTM/, online.

[36] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 83–86, 2006.

[37] G. Boquet, J. L. Vicario, A. Morell, and J. Serrano, "Missing data in traffic estimation: A variational autoencoder imputation method," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2882–2886.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.