AutoTomo: Learning-Based Traffic Estimator Incorporating Network Tomography

Yan Qiao†, *Member, IEEE*, Kui Wu‡, *Senior Member, IEEE*, Xinyu Yuan† *Member, IEEE*
†School of Computer Science and Information Engineering,
Hefei University of Technology, Hefei, Anhui 230009 China
‡Department of Computer Science, University of Victoria,
Victoria, BC V8P5C2, Canada

***Abstract*—Estimating the Traffic Matrix (TM) is a critical yet resource-intensive process in network management. With the advent of deep learning models, we now have the potential to learn the inverse mapping from link loads to origin-destination (OD) flows more efficiently and accurately. However, a significant hurdle is that all current learning-based techniques necessitate a training dataset covering a comprehensive TM for a specific duration. This requirement is often unfeasible in practical scenarios. This paper addresses this complex learning challenge, specifically when dealing with incomplete and biased TM data.**

Our initial approach involves parameterizing the unidentified flows, thereby transforming this problem of target-deficient learning into an empirical optimization problem that integrates tomography constraints. Following this, we introduce AutoTomo, a learning-based architecture designed to optimize both the inverse mapping and the unexplored flows during the model's training phase. We also propose an innovative observation selection algorithm, which aids network operators in gathering the most insightful measurements with limited device resources. We evaluate AutoTomo with two public traffic datasets Abilene and GÉANT. The results reveal that AutoTomo outperforms four state-of-the-art learning-based TM estimation techniques. With complete training data, AutoTomo enhances the accuracy of the most efficient method by 15%, while it shows an improvement between 30% to 53% with incomplete training data. Furthermore, AutoTomo exhibits rapid testing speed, making it a viable tool for real-time TM estimation.

***Index Terms*—Traffic Estimation, Deep Learning, Learning with Missing Data, Network Tomography.**

I. Introduction

The traffic matrix (TM) describes traffic volumes between all origin-destination (OD) pairs in a network during a specific period. It can provide critical knowledge of traffic status for solving many network management problems, such as traffic engineering, anomaly detection, and capacity planning [1]. Although monitoring tools, such as Cisco's NetFlow/TMS [2] and Juniper's JFlow [3], and the emerging software-defined network (SDN) architecture [4] offer an opportunity to count the volumes of particular OD flows, obtaining an accurate and complete TM is still an open challenge. First, the measurement cost of OD flows is prohibitive for SDN devices [5]. Even worse, a considerable number of network devices in legacy networks do not support SDN modules [6]. Second, tools like NetFlow are usually CPU intensive, so the collected flow data is usually partial. Third, network administrators often want to know the traffic volumes between intermediate core routers, which may only forward rather than generate packets. In this case, the concept of *flow* between these routers is virtual in the sense that their IP addresses would not be tagged in the packets [7]. SDN modules and NetFlow tools are incapable of counting the volumes of these virtual flows.

To answer the open challenge, innumerable studies have been devoted to TM estimation to obtain a timely and accurate TM with the lowest measurement cost [8] [9]. Existing TM estimation methods can be roughly grouped into three categories. The **first** is network-tomography (NT) based methods [10] [11]. This kind of methods utilizes the low-cost link loads to infer the OD flow volumes by solving a group of linear equations [12]. As the linear equations are generally rank-deficient (i.e., there is no unique solution for the equations), these methods typically have low accuracies with imposing additional, sometimes unrealistic assumptions on TM. The **second** category is matrix-completion (MC) based methods [9] [13]. These methods estimate the missing volumes in TM using the known volumes based on the spatial-temporal structure of TM. However, the measurement cost of MC-based method is quite high as it requires a large among of flow measurements, and each OD flow should be measured at least once. Otherwise, these methods suffer from low accuracy due to the biased spatial-temporal structure of TM. The **third** kind is deep learning (DL) based methods [14] [15] [16]. They first establish a deep neural network [17] to learn the inverse mapping from link loads to OD flows based on a collection of historical data, and then estimate the complete TM through the trained model by inputting only the low-cost link load data.

This paper focuses on DL-based methods due to their two intrinsic advantages. First, as their performance does not rely on additional assumptions or the spatial-temporal structure of TM, the DL-based methods have much stronger generality. Second, after model training, they can output all OD flows in linear time with only the link loads as the input, making them suitable for real-time TM estimation tasks. Nevertheless, *all existing DL-based methods need accurate and complete historical TM data to train the DL models sufficiently*. This prerequisite is hardly true for most computer networks due to the three reasons discussed at the beginning of the paper. That means *many OD flows, in practice, have never been measured at all*. In such a scenario, both MC-based methods and learning-based methods suffer because the spatial-temporal feature and the learning target are highly deficient in the

collected data.

We aim to tackle the above missing data problem in DL-based methods where *a considerable number of OD flows are absent throughout all measurement times.* Naturally, a naïve solution is to fill each of the unobserved flows in the training data with estimated values before we train the model. This naïve method, however, would perform poorly because the model will inevitably approach a biased training target if the estimated values are not close to the true values. Another possible solution is that we first train the model with a biased target, then adjust the outputs with additional constraints (like NT-based methods). However, the two-stage optimization may lead to sub-optimal results. In addition, this method would incur extra computing overhead, compromising its advantage of real-time TM estimation.

This paper proposes an entirely new approach to handle the target-deficient learning problem. The main idea is that we parameterize the missing flows in the training data and optimize them together with the DL parameters. By doing this, the unobserved flows and the DL parameters will best be compatible. The superiority of this idea is guaranteed theoretically by a theorem proposed in this paper. Considering the hardness of optimizing both two groups of parameters simultaneously, we transform the original problem into an easy-to-learn problem under theoretical proofs, and develop a DL model with a designed loss function to implement the optimization. Finally, the experimental results demonstrated that when both the parameters for capturing the unobserved flows and the DL parameters achieve their optimum, the trained model can map the input link loads onto the complete OD flows with surprisingly high accuracy.

As far as we know, this is the first time the DL models have been applied to traffic estimation with many absent OD flows in the training data. In summary, this paper makes the following contributions to the field of TM estimation.

(1) We apply three theorems to transform the target-deficient learning problem into a typical learning problem. Under the first theorem, the original target-deficient problem is first transformed into a regulated optimization problem. The second theorem indicates the hardness of the regulated problem, motivating us to further transform the problem into a tailored problem. Finally, the last theorem ensures the solution to the tailored problem is equivalent to the solution to the original problem.

(2) We design a DL-based traffic estimator model named AutoTomo, which can accurately learn the inverse mapping from link loads to OD flows with incomplete training data. Corresponding to the objectives of the final tailored problem, AutoTomo utilizes a designed loss function to guarantee consistency with both training instances and tomography constraints. With gradient descent, both neural parameters and unobserved flows can be constantly optimized until they are compatible with each other.

(3) We develop an observation selection algorithm to aid network operators to design an optimal flow-monitor rule to improve the quality of the observations with limited resources. The quality of the observations is quantified by how much can they reduce the uncertainty of the unknown flows. Considering the hardness of computing the uncertainties of flows directly, we propose two theorems to reveal the key factors that affect the uncertainty of flows. Finally, we design a heuristic algorithm based on the two theorems to efficiently select the near-optimal observations.

(4) We compared our methods with four state-of-the-art baselines using two real-world datasets[1]. The results showed that, in the scenario of complete TM estimation our methods can improve the accuracy of the best-performed baseline by $14\% \sim 53\%$, and in the scenario of unobserved flows completion, the accuracy can be improved by $19\% \sim 46\%$. With the proposed observation selection algorithm, the accuracy of our method can further be improved by $9\% \sim 59\%$. Moreover, AutoTomo has a fast testing time, which enables it for real-time estimation tasks.

In the remaining sections of this paper, we first review the related works on traffic estimation in Sec. II. Then the problem that we aim to address in this paper is formulated mathematically in Sec. III. To make the problem solvable, we transform the original problem into an empirical learning problem in Sec. IV, and propose our DL-based traffic estimation method and the observation selection method in Sec. V and Sec. VI, respectively. Finally, we evaluate the performance of our proposals in Sec. VII.

II. Related Work

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

In the first category, network tomography (NT) was proposed to estimate TM from the link loads by solving the linear equations [10]. Since the linear system is generally rank deficient, NT-based methods must 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. [19] imposed a gravity model on the TM. The accuracies of these methods heavily rely on the underlying assumptions.

The second category uses matrix completion (MC). 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 [20]. Zhang et al. [13] proposed sparsity regularized SVD method to estimate the missing values by matrix factorization. Roughan et al. [21] improved [13] by proposing sparsity regularized matrix factorization (SRMF). Fan et al. [22] proposed factor group-sparse regularizers (FGSR) for low-rank matrix completion. Xie et al. [9] [23] [24] proposed tensor completion approaches to recover the missing measurements based on Tucker decomposition. However, the accuracy of these methods depends on the

[1]The codes of the experiments with the proposed methods and baseline methods are available at https://github.com/Y-debug-sys/AutoTomo-TME.

number of known volumes in TM as well as the strong spatial-temporal correlation among the volumes. When there are a large number of missing volumes or the spatial-temporal structure is insignificant, the accuracy of MC-based methods would suffer. Moreover, these methods perform poorly when TM misses whole rows (i.e., absence of flows in all measurement times).

The third category is deep-learning (DL) based methods. These methods learn the correlations among OD flows and link loads with a deep neural network and estimate TM with the trained network. Jiang et al. [25] first introduced a back-propagation neural network (BPNN) to estimate TM. Nie et al. [16] developed a deep-belief-network (DBN) based model to estimate TM from link loads. The method (named MNETME) proposed in [26] improved the former methods by extending the inputs to incorporate the routing information and adjusting the outputted TM to be consistent with the tomography equations. [27] used graph embedding to integrate the network topology with the model input. [15] utilized convolutional neural network (CNN) and long short-term memory (LSTM) network to exploit the spatio-temporal correlations within TM. The method in [14] used a variational autoencoder (VAE) to learn the latent distribution that is "similar" to the training set of TM. It then generated an estimated TM with the learned distribution and adjusted the outputs by tomography equations. The method proposed in [28] combined the forward and backward Convolutional LSTM (ConvLSTM) networks to correct the input TM data to improve the accuracy of future traffic estimation. The method in [29] estimated the OD flows that can satisfy both the tomography equations and a particular distribution learned by generative adversarial networks (GAN). Although DL-based methods present promising results on TM estimation without requiring any distribution or structural assumptions, all the above methods require historical traffic data that covers all OD pairs to well train their models.

To summarize, no existing TM estimation methods can handle a realistic scenario where a considerable number of OD flows are absent throughout all measurement times. In this paper, we aim to transform the target-deficient learning problem into a typical learning problem under the guarantees of high learning accuracy.

III. Problem Formulation

We consider a backbone network $G(V, E)$ with $|V|$ core routers and $|E|$ directed links. Let $X \in \mathbb{R}^{n\times 1}$ denote the vector of all OD flows. Let $Y \in \mathbb{R}^{m\times 1}$ denote the vector of all link loads which can be obtained through traditional SNMP protocols. We use $X(t)$ and $Y(t)$, where $t = \{1, 2, \ldots, T\}$, to represent the instances of OD flows and link loads, respectively, measured at the t-th time point. $\mathbf{X} = \{X(1), X(2), \ldots, X(T)\}$ is the traffic matrix (TM) that we aim to estimate in this paper, and $\mathbf{Y} = \{Y(1), Y(2), \ldots, Y(T)\}$ denotes the link load matrix. $\mathbf{A} \in \mathbb{R}^{m\times n}$ represents the routing matrix. With a determined 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 (such as equal cost multipath, ECMP), the value of a_{ij} is within the range of $[0, 1]$, representing the probability that the j-th flow may traverse the i-th link. The traffic matrix $\mathbf{X}$ and the link load matrix $\mathbf{Y}$ satisfy the linear equations:

$$\mathbf{AX} = \mathbf{Y}. \tag{1}$$

For most networks, the number of flows n is much greater than the number of link loads m, leading to a highly rank-deficient system. That means Eqn. (1) does not have a unique solution in most cases.

With the emerging deep learning techniques [17], the inverse mapping from lower-dimensional link loads to higher-dimensional OD flows has been demonstrated to be learnable through a collection of historical samples of link loads and OD flows [26]. However, collecting a complete training data that includes all OD flows is almost impossible for many networks. Hence, we divide all the OD flows into two groups: flows $X^o = \{x_1^o, x_2^o, \ldots, x_{|o|}^o\}^T$ whose samples can been collected, and flows $X^u = \{x_1^u, x_2^u, \ldots, x_{|u|}^u\}^T$ whose samples are not available, where $|o|$ and $|u|$ are the numbers of observed and unobserved flows, respectively. Our goal is to approach the inverse mapping from the vector of link loads Y to the vector of OD flows X, where $X = [X^o; X^u]$, when only measurements of Y and X^o are available. Next, we first give some definitions and notations, then formulate the problem mathematically.

Definition 1 (Hypothesis)*: A hypothesis f is defined as a model (often with a group of parameters) that approximates a particular function and performs mappings of inputs to outputs.*

Definition 2 (Loss function of a hypothesis)*: Given an input vector I and a target output O, we define a loss function of hypothesis f as follows:*

$$\ell(f, (I, O)) = ||f(I) - O||_2^2, \tag{2}$$

where $||\cdot||_2$ is L2-norm.

Definition 3 (Expected risk of a hypothesis)*: The expected risk of hypothesis f (denoted by $R_{\mathcal{D}}(f)$) is defined as the expected loss of f for all vectors that reside in the joint space $\mathcal{D}$ of the inputs and outputs.*

In our problem, suppose the vector of link loads Y is within a distribution space $\mathcal{Y}$. The distribution spaces for X^o and X^u are denoted by $\mathcal{X}^o$ and $\mathcal{X}^u$, respectively. Let $\mathcal{D}_{yxx} \triangleq \mathcal{Y} \times \mathcal{X}^o \times \mathcal{X}^u$ denote the joint space for the 3-tuple vector (Y, X^o, X^u) and $\mathcal{D}_{yx}$ denote the joint space for (Y, X^o). Let $Z = (Y, X^o, X^u)$ and $S = (Y, X^o)$ denote the vectors from $\mathcal{D}_{yxx}$ and $\mathcal{D}_{yx}$, respectively. The hypothesis $f \in \mathcal{F}$ is the model that maps the link loads $Y \in \mathcal{Y}$ onto the OD flows $X \in \mathcal{X}^o \times \mathcal{X}^u$. $\mathcal{F}$ is the hypothesis space. The loss function of f is $\ell(f, Z) = ||f(Y) - [X^o; X^u]||_2^2$, and the expected risk of f can be written by $R_{\mathcal{D}_{yxx}}(f) = \mathbb{E}_{Z\in\mathcal{D}_{yxx}}[\ell(f, Z)]$. Hence, the fundamental problem that we aim to address in this paper can be formulated as *Problem 1*.

Problem 1 (Fundamental Problem)*: Given a training dataset with samples $\mathbf{S} = \{(Y(1), X^o(1)), (Y(2), X^o(2)), \ldots, (Y(T), X^o(T))\}$, which were drawn from $\mathcal{D}_{yx}$, search for an optimal hypothesis $f_1^* \in \mathcal{F}$ that can minimize the expected risk $R_{\mathcal{D}_{yxx}}(f)$.*

TABLE I
DEFINED SYMBOLS

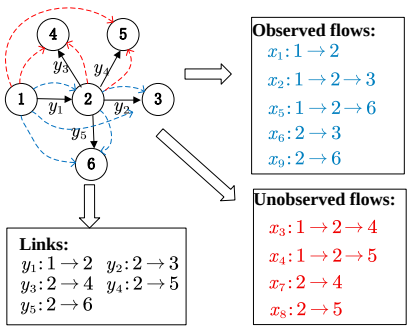
Symbols	Meanings
X	The vector of OD flows
n	The number OD flows, i.e., $n = \|X\|$
X^o	The vector of observed flows
X^u	The vector of unobserved flows
$\mathbf{X}$	The traffic matrix with multiple measurements
Y	The vector of link loads
m	The number of link loads, i.e., $m = \|Y\|$
$\mathbf{Y}$	The link loads matrix with multiple measurements
$\mathbf{A}$	The routing matrix
$\mathcal{Y}$	The distribution space of Y
$\mathcal{X}^o$	The distribution space of X^o
$\mathcal{X}^u$	The distribution space of X^u
$\mathcal{D}_{yxx}$	The joint distribution of $\mathcal{Y} \times \mathcal{X}^o \times \mathcal{X}^u$
$\mathcal{D}_{yx}$	The joint distribution of $\mathcal{Y} \times \mathcal{X}^o$
f	A hypothesis that maps link loads to OD flows
$\mathcal{F}$	The space of hypotheses, i.e., $f \in \mathcal{F}$
S	$S = (Y, X^o)$ is a vector from $\mathcal{D}_{yx}$
Z	$Z = (Y, X^o, X^u)$ is a vector from $\mathcal{D}_{yxx}$
$\mathbf{S}$	The set of training data with incomplete OD flows
$\ell(f, Z)$	The loss function of hypothesis f
$R_{\mathcal{D}_{yxx}}(f)$	The expected risk of hypothesis f
$\mathcal{G}^u(S, \mathbf{A})$	Value space for X^u under tomography constrains
$h(f_a, X^u_a)$	The MSE of f_a when assigning the unobserved flows with X^u_a
$R_{\mathcal{D}_{yx}}(f, X^u)$	The expected risk of hypothesis f with parameterized X^u
$R_{\mathcal{S}_{yx}}(f, \mathbf{X}^u)$	The empirical risk of hypothesis f with parameterized $\mathbf{X}^u$
$\mathcal{N}(\epsilon, \mathcal{F}, T)$	The covering number of hypotheses $\mathcal{F}$
$\mathcal{N}_{p3}(\epsilon, \mathcal{F}, T)$	The covering number of hypotheses for *Problem 3*
D	The deep neural network in AutoTomo
$TVR(x_i)$	The tightest value range of flow x_i
FV	The free variables defined in *Definition 8*
PV	The pivot variables defined in *Definition 8*
$TNR(x_i)$	The tightest natural range of x_i defined in *Definition 9*
$\mathbf{B}^o$	The one-hot matrix for X^o
r_B	The rank of the extended matrix $[\mathbf{A}; \mathbf{B}^o]$

Obviously, solving the fundamental problem directly is intractable due to the limited information of the training samples. On one hand, as there is no information on unobserved flows, the learning problem lacks a specific learning target; On the other, as the true distribution of $\mathcal{D}_{yxx}$ is unknown, it is impractical to compute $R_{\mathcal{D}_{yxx}}(f)$ directly. In the following section, we will transform the unsolvable problem into a solvable problem step by step under theoretical guarantees. All important symbols in this paper are listed in Tab. I.

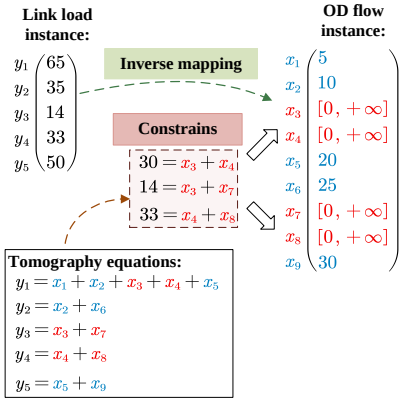
IV. PROBLEM VARIATIONS

A. Main Idea

The main challenge of *Problem 1* is how to learn the mapping from the distribution of Y to the distribution of $[X^o; X^u]$ when there is no information on X^u. To tackle this problem, we leverage the tomography equations (i.e., Eqn. (1)) to compensate for the missing information. We first give a toy example shown in Fig. 1 to explain how tomography equations can help with the target-deficient learning problem, then we re-formulate the problem under the constraints of tomography equations.



(a) A toy network with 5 link loads and 9 OD flows.



(b) Constrained learning target with tomography equations.

Fig. 1. Illustration of tomography constraints on incomplete training instance.

Example: Suppose the network has 6 nodes and 5 directed links as shown in Fig. 1(a). There are totally 9 OD flows including 5 observed flows $X^o = \{x_1, x_2, x_5, x_6, x_9\}$ (in blue color) and 4 unobserved flows $X^u = \{x_3, x_4, x_7, x_8\}$ (in red color). The goal is to find an optimal hypothesis f_1^* that can accurately approach the inverse mapping from the link loads of the 5 links Y to the volumes of the 9 OD flows $[X^o; X^u]$. One sample of link loads and OD flows are given in Fig. 1(b). Without any information on the unobserved flows X^u, f will be optimized to approach the mapping from the link loads sample to OD flows sample with infinite values of unobserved flows. By leveraging the tomography equations, the values of unobserved flows can be constrained into a limited value space. Hence, the hypothesis f will be optimized to approach the mapping from link loads to a constrained flow space.

B. Regulated Problem

From the above example, tomography equations can help constrain the possible values of X^u in the training set. However, the learning target is still deficient since X^u does not

have a distinct assignment yet. Rather than assigning X^u with an arbitrary solution to the tomography constraints, we propose to search for an optimal assignment of X^u that are most compatible with the hypothesis f. By doing this, the optimized hypothesis will, to the most extent, bound the bias from the true hypothesis (which will be demonstrated later in *Theorem 1*). Hence, we parameterize the variables in vector X^u and define a value space $\mathcal{G}^u(S, \mathbf{A})$ for X^u as follows: given $S = (Y, X^o)$ and routing matrix $\mathbf{A}$, $\forall X^u : X^u \in \mathcal{G}^u(S, \mathbf{A})$, it has

$$\mathbf{A}[X^o; X^u] = Y. \tag{3}$$

Then leveraging the constraints of network tomography, *Problem 1* can be transformed into a regulated problem.

Problem 2 (Regulated Problem)*: Given a training dataset with samples* $\mathbf{S} = \{(Y(1), X^o(1)), (Y(2), X^o(2)), \ldots, (Y(T), X^o(T))\}$ *that were drawn from* $\mathcal{D}_{yx}$*, search for an optimal algorithm* $f_2^* \in \mathcal{F}$ *and an optimal assignment* $X^{u*} \in \mathcal{G}^u(S, \mathbf{A})$ *that can minimize the expected risk* $R_{\mathcal{D}_{yx}}(f, X^u)$*, where*

$$\begin{aligned} &R_{\mathcal{D}_{yx}}(f, X^u) \\ =&\mathbb{E}_{S \in \mathcal{D}_{yx}}[\ell(f, (S, X^u))]]. \end{aligned} \tag{4}$$

We denote (f_2^*, X^{u*}) as an optimal solution to *Problem 2* that

$$(f_2^*, X^{u*}) = \arg\min_{f \in \mathcal{F}, X^u \in \mathcal{G}^u(S, \mathbf{A})} R_{\mathcal{D}_{yx}}(f, X^u) \tag{5}$$

Different from *Problem 1*, *Problem 2* has a specific learning target: approaching the mapping from Y to $[X^o; X^{u*}]$, where X^{u*} is the optimal assignment for X^u. Next, we demonstrate the superiority of X^{u*} over other assignments for X^u.

Let $X_t^u \in \mathcal{G}^u(S, \mathbf{A})$ denote the true value of X^u. We use f_t^* to denote the *true hypothesis* that can minimize $R_{\mathcal{D}_{yx}}(f, X_t^u)$. The *true hypothesis* f_t^* can also be considered as the optimal hypothesis with complete and unbiased training data. We define $h(f_a, X_a^u)$ as the supremum of mean squared error (MSE) of hypothesis f_a with assignment X_a^u, which can be written by

$$h(f_a, X_a^u) = \sup\{\mathbb{E}_{S \in \mathcal{D}_{yx}}[||f_a(Y) - f_t^*(Y)||_2^2]\}. \tag{6}$$

Theorem 1: *Suppose* (f_2^*, X^{u*}) *is the optimal solution to Problem 2.* $\forall X_a^u : X_a^u \in \mathcal{G}^u(S, \mathbf{A}), X_a^u \neq X^{u*}$*, suppose* f_a^* *is the optimal hypothesis that minimizes* $R_{\mathcal{D}_{yx}}(f_a, X_a^u)$*. It always holds that* $h(f_2^*, X^{u*}) \leq h(f_a^*, X_a^u)$.

The proof of *Theorem 1* can be found in Appendix. *Theorem 1* means when both hypothesis and the unobserved flows achieve their optimum, the bias bound of the solution can be minimized. Although *Problem 2* has a specific learning target, the distribution of $\mathcal{D}_{yx}$ is still unknown. In the following section, we will further transform *Problem 2* into an empirical problem that utilizes the empirical distribution to approach the true distribution.

C. Empirical Problem

Definition 4 (Empirical risk)*: The empirical risk (denoted by* $R_{\mathcal{S}_{yx}}(f, \mathbf{X}^u)$*) on the training set* $\mathbf{S}$ *is defined as the expected loss of hypothesis* f *for all samples in* $\mathbf{S}$*, which can be formulated by*

$$R_{\mathcal{S}_{yx}}(f, \mathbf{X}^u) = \frac{1}{T} \sum_{t=1}^{T} \ell(f, (S_t, X^u(t))), \tag{7}$$

where $S_t = (Y(t), X^o(t))$*,* $t = \{1, 2, \ldots, T\}$ *and* $\mathbf{X}^u = \{X^u(1), X^u(2), \ldots, X^u(T)\}$.

Problem 3 (Empirical Problem)*: Given a training dataset with samples* $\mathbf{S} = \{(Y(1), X^o(1)), (Y(2), X^o(2)), \ldots, (Y(T), X^o(T))\}$ *that were drawn from* $\mathcal{D}_{yx}$*, search for an optimal hypothesis* f_3^* *and an optimal assignment* $\mathbf{X}^{u*} = \{X^{u*}(1), X^{u*}(2), \ldots, X^{u*}(T)\}$*, where* $X^{u*}(t) \in \mathcal{G}^u(S_t, \mathbf{A})$*, that minimize* $R_{\mathcal{S}_{yx}}(f, \mathbf{X}^u)$.

We denote $(f_3^*, \mathbf{X}^{u*})$ as an optimal solution to *Problem 3* that

$$(f_3^*, \mathbf{X}^{u*}) = \arg\min_{f \in \mathcal{F}, \mathbf{X}^u \in \mathcal{G}^u(\mathbf{S}, \mathbf{A})} R_{\mathcal{S}_{yx}}(f, \mathbf{X}^u) \tag{8}$$

To verify how well the solution of *Problem 3* can approach the solution of *Problem 2*, we first give three definitions and then import a theorem to analyze the impact factors of this approach.

Definition 5 (ϵ-Cover) [30]*: For* $\epsilon > 0$*, and two vector sets* $\mathbf{U}$ *and* $\mathbf{W}$*, where* $\mathbf{U} \subseteq \mathbf{W}$*, we say that* $\mathbf{U}$ *is an* ϵ*-Cover of* $\mathbf{W}$ *if for each* $w \in \mathbf{W}$ *there is a* $u \in \mathbf{U}$ *such that* $d(w, v) \leq \epsilon$*, where* $d(w, v)$ *is any distance metric.*

Definition 6 (Covering number) [30]*: The covering number for* $\mathbf{W}$ *(denoted by* $\mathcal{N}(\epsilon, \mathbf{W}, d)$*) is defined to be the minimum cardinality of an* ϵ*-cover for* $\mathbf{W}$*. When* $\mathbf{W}$ *is within a bounded area and* $\epsilon > 0$*,* $\mathcal{N}(\epsilon, \mathbf{W}, d)$ *is a finite integer.*

Definition 7 (Covering number for hypotheses) [30]*: Suppose* $\mathcal{F}$ *is a set of hypotheses from input space* $\mathcal{Y}$*. The covering number of* $\mathcal{F}$ *(denoted by* $\mathcal{N}(\epsilon, \mathcal{F}, T)$*) is defined as the maximum covering number over all* $Y \in \mathcal{Y}^T$*, i.e.,*

$$\mathcal{N}(\epsilon, \mathcal{F}, T) = \max\{\mathcal{N}(\epsilon, \mathcal{F}_{|Y}, d) : Y \in \mathcal{Y}^T\}, \tag{9}$$

Where $\mathcal{F}_{|Y} = \{f(Y(1)), f(Y(2)), \ldots, f(Y(T)) : f \in \mathcal{F}\}$.

The physical meaning of covering number $\mathcal{N}(\epsilon, \mathcal{F}, T)$ is a measurement of the richness of the hypotheses in $\mathcal{F}$, which quantifies how fast the outputs can "fill up" the target space. In our problem, as all flow volumes are bounded by the maximum link load, the covering number $\mathcal{N}(\epsilon, \mathcal{F}, T)$ is a finite integer. Next, we introduce a theorem (adapted from [30]) to show the key factors that may affect the approximate performance of *Problem 3*.

Theorem 2: *Suppose* $\mathcal{F}$ *is the hypothesis space and* $\ell(f, Z)$ *is a loss function bounded in* $[0, B]$*, where* $f \in \mathcal{F}$ *and* $Z \in \mathcal{D}_{yxx}$*. Given the training instances* $\mathbf{Z} = \{(Y(1), X(1)), (Y(2), X(2)), \ldots, (Y(T), X(T))\}$*, and an* $\epsilon > 0$*, we have*

$$P(R_{\mathcal{D}_{yxx}}(f) - R_{\mathcal{Z}_{yxx}}(f) < \epsilon) \geq 1 - \delta(\mathcal{F}, \epsilon, T, B), \tag{10}$$

where $\delta(\mathcal{F}, \epsilon, T, B)$ *has the complexity of* $O(\mathcal{N}(\epsilon, \mathcal{F}, T) \cdot exp(-\frac{T\epsilon^2}{B^2}))$.

The proof of *Theorem 2* can be concluded from [30]. According to *Theorem 2*, the approximate accuracy of a regular mapping problem depends on the following three factors: (1) The covering number of the hypothesis space, (2) the number

of training samples, and (3) the bound on the loss function. In other words, given a group of training samples and a bounded loss function [2], the approximate accuracy is determined by the covering number of the hypothesis space. Therefore, under the guarantee of minimized *empirical risk*, it is necessary to construct the model with minimized covering number to reduce the potential learning bias from the true distribution. For example, many DL models were designed to use as few parameters as possible or normalized parameters to prevent overfitting [31].

Now we consider the covering number for the hypothesis space in *Problem 3* (denoted by $\mathcal{N}_{p3}(\epsilon, \mathcal{F}, T)$). As the variables of $\mathbf{X}^u$ in the target space are parameterized in *Problem 3*, the covering number can be expressed by

$$\mathcal{N}_{p3}(\epsilon, \mathcal{F}, T) = \max\{\mathcal{N}(\epsilon, \mathcal{F}_{|Y} \times \mathcal{G}^u(S, \mathbf{A}), d) : Y \in \mathcal{Y}^T\}. \tag{11}$$

As *Problem 3* has a much expanded target space (i.e., $\mathcal{F}_{|Y} \times \mathcal{G}^u(S, \mathbf{A})$) compared with the regular mapping problem, it has an exponentially increased covering number. That means *Problem 3* will face both much higher searching complexity and much higher risk of larger empirical errors. To overcome this problem, in the next section we will propose a tailored problem and demonstrate the optimal solution to the tailored problem is equivalent to the optimal solution to *Problem 3*.

D. Tailored Problem

Problem 4 (Tailored Problem)*: Given a training dataset with samples* $\mathbf{S} = \{(Y(1), X^o(1)), (Y(2), X^o(2)), \ldots, (Y(T), X^o(T))\}$ *that were drawn from* $\mathcal{D}_{yx}$*, search for an optimal hypothesis* f_4^* *that can minimize* $R^o_{\mathcal{S}_{yx}}$ *and* $R^u_{\mathcal{S}_{yx}}$*, where*

$$R^o_{\mathcal{S}_{yx}}(f) = \frac{1}{T}\sum_{t=1}^{T} ||\hat{X}^o(t) - X^o(t)||_2^2, \tag{12}$$

$$R^u_{\mathcal{S}_{yx}}(f) = \frac{1}{T}\sum_{t=1}^{T} ||\mathbf{A} \cdot [X^o(t); \hat{X}^u(t)] - Y(t)||_2^2, \tag{13}$$

$$[\hat{X}^o(t); \hat{X}^u(t)] = f(Y(t)), \quad t = \{1, 2, \ldots, T\}.$$

Problem 4 divides the objective into two parts and only searches for the optimal solution in the hypothesis space instead of the joint space of f and X^u. Next, we demonstrate the optimal hypothesis for *Problem 4* is equivalent to the optimal hypothesis for *Problem 3*.

Theorem 3: *(1) If* f_4^* *can minimize both two objectives simultaneously in Problem 4,* f_4^* *is an optimal hypothesis for Problem 3; (2) Otherwise, if* f_4^* *can make* $R^o_{\mathcal{S}_{yx}}(f_4^*) < \epsilon_o$ *and* $R^u_{\mathcal{S}_{yx}}(f_4^*) < \epsilon_u$*, then* $R_{\mathcal{S}_{yx}}(f_4^*, \mathbf{X}^{u*}) < \epsilon_o + \frac{||\mathbf{A}_u||_2^2 \cdot \epsilon_u}{\lambda^2_{min}}$*, where* $\mathbf{A}_u$ *is the sub-matrix of* $\mathbf{A}$ *that only contains the columns of unobserved flows,* λ_{min} *is the minimum non-zero eigenvalue of matrix* $\mathbf{A}_u^T\mathbf{A}_u$*, and* $\mathbf{A}_u^T$ *is the transpose matrix of* $\mathbf{A}_u$*.*

The proof of *Theorem 3* can be found in Appendix. According to the first result of *Theorem 3*, if we can find an optimal hypothesis for *Problem 4* that minimizes both the mapping errors on observed flows X^o and the reconstruction errors on the link loads Y, it can minimize the mapping errors on the optimal vector $[X^o; X^{u*}]$ in *Problem 3*. The second result indicates that if we cannot find an optimal solution to both of the two objectives, we can instead to reduce the two *empirical risks* in *Problem 4* as much as possible. By doing this, the *empirical risk* in *Problem 3* can be reduced to the utmost. That means the optimization of X^u can be incorporated into the optimization of the hypothesis f under the two objectives in *Problem 4*. Transforming *Problem 3* to *Problem 4* can not only significantly reduce the potential empirical risk, but also facilitate solving the problem with standard deep neural networks. Hence, we develop a deep network architecture in next section to approach the optimal hypothesis for *Problem 4*, and optimize the network parameters to reduce the two risks in *Problem 4* as much as possible.

V. AutoTomo: Learning-Based Traffic Estimator

A. Overview

In this section, we develop a deep network architecture named AutoTomo to approach the optimal hypothesis in *Problem 4*. The architecture of AutoTomo is shown in Fig. 2, which contains two modules: mapping and reconstructing. In the mapping module, a group of training data Y is drawn from the link loads space $\mathcal{Y}$. Then the link loads data is mapped onto the estimated OD flows through a deep neural network. Afterwards, the estimated observed flows $\hat{X}^o$ are compared with the training data X^o drawn from observations space to compute the mapping error. In the reconstructing module, the estimated unobserved flows $\hat{X}^u$ together with the training data drawn from observation space are inputted into the tomography equations. Then the output is compared with the data from the link loads space to compute the reconstructing error. The mapping error and reconstructing error are considered as two losses, which will be backpropagated to the deep neural network to optimize the parameters in the neural units. Note that, all instances have been regularized to the range of [0,1] before inputting them into the model (refer to Sec. VII-A for more details).

AutoTomo applies a concise one-in-one-out paradigm that both training and testing only involve the link loads and OD flows at the same measurement time. We abandoned the temporal module (such as LSTM and GRU) because such a sequence-in-one-out paradigm did not present significant improvement on TM estimation in our experiments but caused more difficulties on model training (such as the swinging training process and the longer training time).

B. Deep Neural Network in Mapping Module

The inverse mapping from link loads to OD flows is much more complicated than a linear transformation. We apply both linear and non-linear units in the deep neural network to approximate the complicated mapping from the lower-dimensional vector of link loads to the higher-dimensional vector of OD flows.

Specifically, the deep neural network (denoted by D) contains two non-linear units, one flatten layer and one Sigmoid

[2]When all inputs and outputs were scaled to a particular range, the loss function can be bounded.

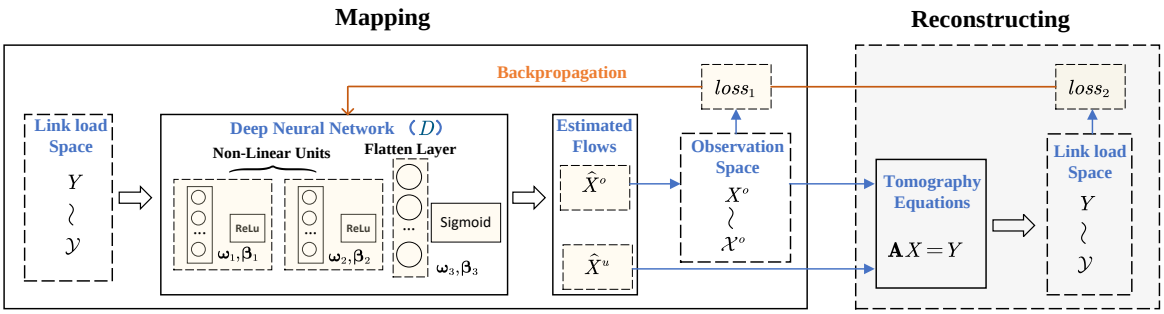


Fig. 2. The architecture of AutoTomo: it uses mapping and reconstructing modules to learn the inverse mapping from incomplete training data.

layer as shown in Fig. 2. Each non-linear unit is composed of one linear layer and one ReLu layer as the activation function. It is responsible for learning the complicated correlation between link loads and OD flows. The flatten layer after the non-linear units is responsible for mapping the outputs of the non-linear units onto the n-dimensional space (recall that n is the number of OD flows). Finally, the last Sigmoid layer regulates the ranges of the outputs onto [0,1].

Given an instance of input Y, the output of the neural network can be computed by

$$h_1 = r(\omega_1 Y + \beta_1), \tag{14}$$
$$h_2 = r(\omega_2 h_1 + \beta_2), \tag{15}$$
$$\hat{X} = \sigma(\omega_3 h_2 + \beta_3), \tag{16}$$

where h_1 and h_2 are the hidden states of the two non-linear units. ω_i and β_i, $i = \{1,2,3\}$ are the weight and bias parameters in the non-linear units and the flatten layer, respectively. $r(\cdot)$ is the activation function of ReLu, and $\sigma(\cdot)$ is the activation function of Sigmoid.

C. Loss Function in AutoTomo

The loss function in AutoTomo is composed of two losses (shown in Eqn. (17) and (18), respectively), which assess the current accuracy of the deep model based on the two objectives in *Problem 4*.

$$loss_1 = ||\hat{X}^o - X^o||_2^2, \tag{17}$$
$$loss_2 = ||\mathbf{A} \cdot [X^o; \hat{X}^u] - Y||_2^2, \tag{18}$$

where $\hat{X}^o$ and $\hat{X}^u$ are the estimated observed and unobserved flows by inputting Y, respectively. $loss_1$ in Eqn. (17) can be considered as the mapping errors on observed flows and $loss_2$ in Eqn. (18) can be considered as reconstructing errors on link loads. To reduce both of the two errors, we integrate them into a total loss function as follows:

$$loss = loss_1 + \lambda loss_2, \tag{19}$$

where λ is a hyperparameter that balances the weight between the two losses. The settings of λ will be evaluated in Sec. VII-E. During the training process, the total losses are backpropagated to deep network D to update the parameters ω_i and β_i, $i = \{1,2,3\}$ by gradient descent.

D. Detailed Training Algorithm

Given the training samples $\mathbf{Y} = \{Y(1), Y(2), \ldots, Y(T)\}$ and $\mathbf{X}^o = \{X^o(1), X^o(2), \ldots, X^o(T)\}$, the parameters in D will be updated with I_{max} training epochs. The detailed training algorithm is shown in Alg. 1

Algorithm 1: Training Algorithm of AutoTomo

Input: $\mathbf{A}$, I_{max}, λ, $\mathbf{Y} = \{Y(1), Y(2), \ldots, Y(T)\}$, $\mathbf{X}^o = \{X^o(1), X^o(2), \ldots, X^o(T)\}$
Output: Trained D

1 $D \leftarrow Initial()$
2 $i \leftarrow 0$
3 **while** $i \leq I_{max}$ **do**
4 **for** $t \in \{1, 2, \ldots, T\}$ **do**
5 $[\hat{X}^o(t); \hat{X}^u(t)] \leftarrow Mapping(D, Y(t))$
6 $loss_1 \leftarrow MSENorm(\hat{X}^o(t), X^o(t))$
7 $\hat{Y}(t) \leftarrow \mathbf{A} \cdot [X^o; \hat{X}^u]$
8 $loss_2 \leftarrow MSENorm(\hat{Y}(t), Y(t))$
9 $loss \leftarrow loss_1 + \lambda \cdot loss_2$
10 $D \leftarrow GradientDescent(loss, D)$
11 $i \leftarrow i + 1$
12 **return** D

Alg. 1 inputs the routing matrix $\mathbf{A}$, the maximum number of training epochs I_{max}, the hyperparameter λ, the training samples $\mathbf{Y}$ and $\mathbf{X}^o$. It outputs the trained network D. In Alg. 1, it first initializes the parameters in D with random values (line 1). Then it optimizes the parameters in D with I_{max} epochs (line 3∼11). In each epoch, Alg. 1 first computes the output of D for each sample $Y(t)$ (line 5). The output is divided into two parts: the estimation of observed flows $\hat{X}^o(t)$ and the estimation of unobserved flows $\hat{X}^u(t)$. Then $loss_1$ and $loss_2$ are computed using Eqn. (17) and (18), respectively (line 6∼8). Finally, Alg. 1 computes the total loss (line 9) and updates the parameters in D through gradient descent (line 10).

E. TM Estimation

After model training, the trained network D in AutoTomo can be used to estimate all OD flows by inputting only the link loads. Specifically, the deep neural network D is trained with a collection of link loads $\{Y(1), Y(2), \ldots, Y(T)\}$ and observed flows $\{X^o(1), X^o(2), \ldots, X^o(T)\}$ with Alg. 1. After training, we input D with an sample of link loads $Y(t')$. The output $\hat{X}(t') = D(Y(t'))$ is the estimation of all OD flows at the

time point t'. AutoTomo can either be used to estimate the complete unknown TM (i.e., when $t' > T$) or estimate the missing volumes in the current TM (i.e., when $1 \leq t' \leq T$).

VI. Observation Selection for AutoTomo

In many networks, network operators have the responsibility to design the flow-monitor rules to observe specific flows for a period of time. Actually, carefully designing the flow rules to observe the most informative flows can significantly improve the performance of downstream tasks [32] [33]. Most previous works prefer to observe the elephant flows that may own the top volumes over other flows [34] [35]. However, in practice, it is difficult to know the flow volumes before we collect their measurements. Moreover, in our scenario, where the observed flows are used for model training, the volumes of flows are not the key factor that determines the importance of the flows (which will be demonstrated later). For AutoTomo, the best observations should reveal the maximum information on the unknown flows in order to minimize the learning bias. To select the most informative observations for AutoTomo, we first develop a metric to quantify the information of each flow and then propose an observation selection algorithm to select the most informative flows as the observations.

To measure the flow information, we first analyze the uncertainty of each flow under the tomography constraints, and then we consider the information of an observation as how much it can reduce the uncertainty of the remaining unknown flows. Actually, given $S = (Y, X^o)$, the uncertainty of unknown flows X^u can be considered as the size of its value space $\mathcal{G}^u(S, \mathbf{A})$, which depends on the value range of each flow in X^u. Motivated by [36], we denote the *value range* of flow x_i under the tomography constraints as $VR(x_i)$. $TVR(x_i)$ denotes the *tightest value range* (TVR) of x_i. For example in Fig. 1(a), if we know that flow x_3 traverses link y_1, and the total loads on link y_1 is 65, then $[0, 65]$ is one assignment for $VR(x_3)$ but not necessary the assignment for $TVR(x_3)$. Note that there may be multiple assignments for $VR(x_i)$, but there is only one assignment for $TVR(x_i)$. *Only $TVR(x_i)$ are nontrivial for quantifying the uncertainty of flow x_i.* Next, we first give two definitions and then provide a theorem to show how to compute the TVRs of all flows.

Definition 8 (Free variables (FVs) & Pivot variables (PVs)): FVs and PVs are unknown variables in the tomography equations, where all FVs (denoted by X^p) can be represented by the PVs (denoted by X^f).

For example, suppose all flows have not been observed in tomography equations in Fig. 1, then given the values of link loads $Y = \{y_1, \ldots, y_5\}$ the equations can be transformed into

$$\begin{aligned} x_1 &= c + x_6 + x_7 + x_8 + x_9, \\ x_2 &= y_2 - x_6, \\ x_3 &= y_3 - x_7, \\ x_4 &= y_4 - x_8, \\ x_5 &= y_5 - x_9, \end{aligned} \tag{20}$$

where $c = y_1 - y_2 - y_3 - y_4 - y_5$. Hence, $X^f = \{x_6, x_7, x_8, x_9\}$ are FVs and the remaining variables $X^p = \{x_1, x_2, x_3, x_4, x_5\}$ are PVs. Note that the combinations of FVs and PVs are not unique. For example, x_2, x_3, x_4 and x_5 in Eqn. (20) can also be selected as FVs.

Definition 9 (Tightest Natural Range (TNR)): The TNR of x_i (denoted by $TNR(x_i)$) is the value range $[0, \mu_{x_i}]$, where μ_{x_i} is minimum value of link load that include x_i.

For example in Fig. 1(b), since x_3 traverses both link y_1 (with loads 65) and y_3 (with loads 14), $TNR(x_3) = [0, 14]$.

Theorem 4: *If the tomography equations contain n unknown variables and the rank of the system is r, the minimum number of FVs is $n - r$. The TVRs of the unknown variables can be computed from the TNRs of the optimal FVs $X^{f*} = \{x_1^{f*}, x_2^{f*}, \ldots, x_{n-r}^{f*}\}$ that X^{f*} has the least total length of TNRs over other combinations of FVs, i.e.,*

$$X^{f*} = \underset{X^f = \{x_1^f, \ldots, x_{n-r}^f\} \in \{All\ FVs\}}{\arg\min} \sum_{i=1}^{n-r} |TNR(x_i^f)|. \tag{21}$$

The proof of *Theorem 4* can be concluded from [36]. *Theorem 4* indicates that TVRs of all flows can be obtained by searching the optimal combination X^{f*} from all combinations of FVs in the linear system. After obtaining the optimal FVs, the TVRs of all unknown flows can be computed as follows: if the flow is a FV, its TVR is its TNR itself; otherwise, if the flow is a PV, the TVR can be computed by plugging the TNRs of FVs into the linear equations. Once we can compute the TVRs of all flows, the uncertainty of these flows can be measured by the lengths of their TVRs. Hence, the goal of observation selection for AutoTomo is converted to *finding a group of optimal observations X^{o*} that can minimize the total length of TVRs of the unknown flows X^u*, which can be formulated by

$$\begin{aligned} X^{o*} = \arg\min_{X^o} & \sum_{x_i \in X^u} |TVR(x_i)|, \\ s.t. \quad & \mathbf{A} \cdot [X^o; X^u] = Y. \end{aligned} \tag{22}$$

However, in practice there are two barriers to find the optimal X^{o*} in Eqn. (22). Firstly, before computing the TVRs of flows in X^u, it needs to substitute the variables of X^o in the system for their true values, which is impossible to accomplish before measurement. Secondly, finding an optimal group of X^{f*} is time intensive [36] since we need to search for all combinations of FVs. Fortunately, our ultimate goal is to minimize the total length of TVRs rather than computing them, which provides us opportunities to bypass the two barriers and turn to explore the key factors that can reduce the TVRs.

Theorem 5: *Suppose X is the set of unknown variables in linear system $\mathbf{A}X = Y$. $\forall x_i : x_i \in X$, the length of $TVR(x_i)$ is proportional to both the number of optimal FVs involved in the calculation of $TVR(x_i)$ and the length of TNR of each involved FV.*

The proof of *Theorem 5* can be found in Appendix. According to *Theorem 5*, the observations selection can be considered from two aspects: (1) reduce the number of optimal FVs in the remaining linear system and (2) reduce the lengths of TNBs of the optimal FVs. Neither of the above two aspects needs to know the exact values of the observations before measurement or search for the optimal FVs.

We first consider how to reduce the number of optimal FVs. Let $B_i = \{0, \ldots, 1, \ldots, 0\}$ denote the one-hot code for flow x_i, where the i-th element is one whereas the rest $n-1$ elements are all zeros. Let $\mathbf{B}^o \in \mathbb{R}^{|o| \times n}$ denote the one-hot matrix for observed flows X^o, and r_B denote the rank of the extended matrix $[\mathbf{A}; \mathbf{B}^o]$. Then the tomography equations can be expanded to

$$[\mathbf{A}; \mathbf{B}^o] \cdot X = [Y; \hat{X}^o]. \tag{23}$$

Recall that $X = [X^o; X^u]$. $\hat{X}^o$ is an arbitrary assignment for X^o. Note that the value of assignment $\hat{X}^o$ does not affect the value of rank r_B. According to *Theorem 4*, the minimum number of optimal FVs for Eqn. (23) is $n - r_B$. Therefore, to minimize the number of optimal FVs, we can select the observations that can maximize the rank r_B.

Next, we consider how to reduce the lengths of TNRs of the optimal FVs. From this aspect, we simply select the variables with the maximum TNRs as observations. So that the FVs in the remaining variables are more likely to have smaller TNRs.

Algorithm 2: Observation Selection for AutoTomo

Input: $\bar{Y}$, $\mathbf{A}$, k
Output: X^{o*}

1 $X^{o*} \leftarrow Null$
2 $X \leftarrow \{x_1, x_2, \ldots, x_n\}$
3 **for** $x_i \in X$ **do**
4 $\quad [0, \mu_{x_i}] \leftarrow computeTNR(\bar{Y}, \mathbf{A}, x_i)$
5 **while** $|X^{o*}| < k$ *or* $X \neq Null$ **do**
6 $\quad x^* \leftarrow \arg\max\{\mu_{x_i} | x_i \in X\}$
7 $\quad B^* \leftarrow oneHot(x^*)$
8 $\quad r \leftarrow rank(\mathbf{A})$
9 $\quad r' \leftarrow rank([\mathbf{A}; B^*])$
10 $\quad$ **if** $r' > r$ **then**
11 $\quad\quad X^{o*} \leftarrow add(X^{o*}, x^*)$
12 $\quad\quad \mathbf{A} \leftarrow [\mathbf{A}; B^*]$
13 $\quad\quad X \leftarrow remove(X, x^*)$
14 **if** $|X^{o*}| < k$ **then**
15 $\quad \tilde{X}^{o*} \leftarrow maxTNR(X, k - |X^{o*}|)$
16 $\quad X^{o*} \leftarrow add(X^{o*}, \tilde{X}^{o*})$
17 **return** X^{o*}

Comprehensively considering the above two aspects, we design a heuristic algorithm to select the optimal observations for AutoTomo. The details of the observation selection algorithm is shown in Alg. 2. The algorithm takes the averaged link loads $\bar{Y}$, routing matrix $\mathbf{A}$ and the number of observations k as input, and outputs the selected observations X^{o*}. At the beginning, the set of observations X^{o*} is initialized with null (line 1), and the candidate flow set X is initialized with all flows (line 2). For each flow x_i the algorithm computes $TNR(x_i)$, whose upper bound is μ_{x_i}, according to the averaged link loads that include the flow (line 4). Afterwards, the algorithm starts to select the k observations through k-rounds while loop (line 5$\sim$13). In each loop, it first picks the flow x^* with the maximum upper bound of TNR (line 6), and generates the one-hot code B^* for the flow (line 7) to check whether it can increase the rank of matrix $\mathbf{A}$ (line 8$\sim$10). If the rank can be raised by observing the flow[3], x^* will be added to the observation set X^{o*} (line 11), and the matrix $\mathbf{A}$ as well as the candidate flows in X will be updated accordingly (line 12$\sim$13). If the number of flows in X^{o*} is smaller than k after the while loop, we directly select $k - |X^{o*}|$ flows in X with the maximum TNRs (line 15), and add them into observations (line 16).

The most time-consuming operation in Alg. 2 is computing the rank of matrix $[\mathbf{A}; B^*]$, which has the time complexity of $O(m^2(n_A+1))$, where m and n_A are the numbers of columns and rows of $\mathbf{A}$, respectively. With k rounds of while loops, the number of rows of matrix $\mathbf{A}$ expands from n to $n+k$. Then the highest complexity of Alg. 2 is $O(knm^2)$. In our experiments, it only costs around 5 seconds at most to select all observations in both two datasets.

VII. Experiments

In this section, we compare the performance of our methods with four state-of-the-art methods on both complete TM estimation and unobserved flow completion. We use AutoTomo and AutoTomo-OS to denote our method and our method with observation selection, respectively. All experimental codes are available at https://github.com/Y-debug-sys/AutoTomo-TME.

A. Datasets

We use two real-world traffic datasets to validate the performance of our methods.

Abilene [37]: The Abilene dataset contains 12 routers, 30 directed inner links and 24 outside links. The dataset collected the volumes of all OD flows in the network every 5 minutes from March to September 2004. The total size of the dataset is 48384×144, where 48384 is the total number of samples and 144 is the number of OD flows. We used the samples during the first 15 weeks (totally $15 \times 7 \times 288 = 30240$ samples) as the training data and use the samples in the 16-th week (totally $7 \times 288 = 2016$ samples) as the testing data to evaluate the performance of the methods.

GÉANT [38]: The GÉANT network contains 23 routers and 120 directed links. All flows in this dataset were collected in 15-minute intervals from January to April 2004. The total size of the dataset is 10772×529, where 10772 is the total number of samples and 529 is the number of OD flows. We used the first 10 weeks' collections (totally $10 \times 7 \times 96 = 6720$ samples) as the training data, and the 11th week's collections (totally $7 \times 96 = 672$ samples) as the testing data.

For the above two datasets, we randomly selected q percent of the OD flows as the unobserved flows, assuming the rest of the OD flows were observations. In our experiments, q was considered as *unknown rate* of the training data. We varied q from 0% to 70% to evaluate the performance of the methods under different unknown rates. All measurement data for unobserved flows were uniformly assigned with the mean values of the observed flows within the same measurement

[3]It is easy to prove that observing one flow can at most increase the rank by one.

time. To regularize the training data onto range [0,1], all training samples were normalized by dividing the maximum values of link loads in the datasets.

B. Metrics

Referring the mainstreaming metrics that were widely used for traffic estimation [14] [26], the estimation accuracy of all methods is qualified by the normalized mean absolute error for all flows ($NMAE_f$), the normalized mean absolute error for unobserved flows ($NMAE_u$), temporal related mean absolute error (TRE), and spatial related mean absolute error (SRE).

Let $\mathbf{X} = \{X(T_{train}+1), X(T_{train}+2), \ldots, X(T_{train}+T_{test})\}$ denote the true values of OD flows during the testing period, where T_{train} and T_{test} are the numbers of training and testing samples, respectively. We use $\hat{\mathbf{X}} = \{\hat{X}(T_{train}+1), \hat{X}(T_{train}+2), \ldots, \hat{X}(T_{train}+T_{test})\}$ to denote the set of corresponding estimated values of flows $\mathbf{X}$. The $NMAE_f$ for estimating all flows can be calculated by

$$NMAE_f = \frac{\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} ||X(t) - \hat{X}(t)||_1}{||\mathbf{X}||_1}, \quad (24)$$

where $||\cdot||_1$ denotes L1-norm.

Let $\mathbf{X}^u = \{X^u(1), X^u(2), \ldots, X^u(T_{train})\}$ denote the true values of unobserved flows in the training dataset. Let $\hat{\mathbf{X}}^u = \{\hat{X}^u(1), \hat{X}^u(2), \ldots, \hat{X}^u(T_{train})\}$ denote the corresponding estimated values of $\mathbf{X}^u$. The $NMAE_u$ for estimating unobserved flows can be calculated by

$$NMAE_u = \frac{\frac{1}{T_{train}} \sum_{t=1}^{T_{train}} ||X^u(t) - \hat{X}^u(t)||_1}{||\mathbf{X}^u||_1}. \quad (25)$$

Let $X(t) = \{x_1(t), x_2(t), \ldots, x_n(t)\}$ denote the true values of all OD flows measured at the t-th time point. We use $\hat{X}(t) = \{\hat{x}_1(t), \hat{x}_2(t), \ldots, \hat{x}_n(t)\}$ to denote the set of corresponding estimated flows. Then TRE and SRE can be calculated by

$$TRE(t) = \frac{\frac{1}{n} \sum_{i=1}^{n} |x_i(t) - \hat{x}_i(t)|}{\frac{1}{n} \sum_{i=1}^{n} x_i(t)}, \quad (26)$$

and

$$SRE(i) = \frac{\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} |x_i(t) - \hat{x}_i(t)|}{\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} x_i(t)}, \quad (27)$$

respectively.

C. Baselines

For complete TM estimation, we compare AutoTomo and AutoTomo-OS with three recent learning-based methods that were designed for estimating complete TM from link loads. The first is the VAE-based method [14], which first learns a latent distribution of Abilene flows and then estimates the volumes of OD flows that satisfy both the tomography equations and the latent distribution. The second baseline is DBN based method [16], which learns the inverse mapping by a DBN network, and considers the outputs of the trained network as the estimation of the OD flows. The third baseline is BPNN based method (named MNETME) [26], which trains a BPNN network with extended inputs, and estimates the volumes of OD flows by both the outputs of the trained model and tomography equations.

For unobserved flow completion, besides the three learning-based methods, we additionally compare our methods with a state-of-the-art matrix completion method (named FGSR) [22], which approximates the unknown values in the matrix through matrix decomposition.

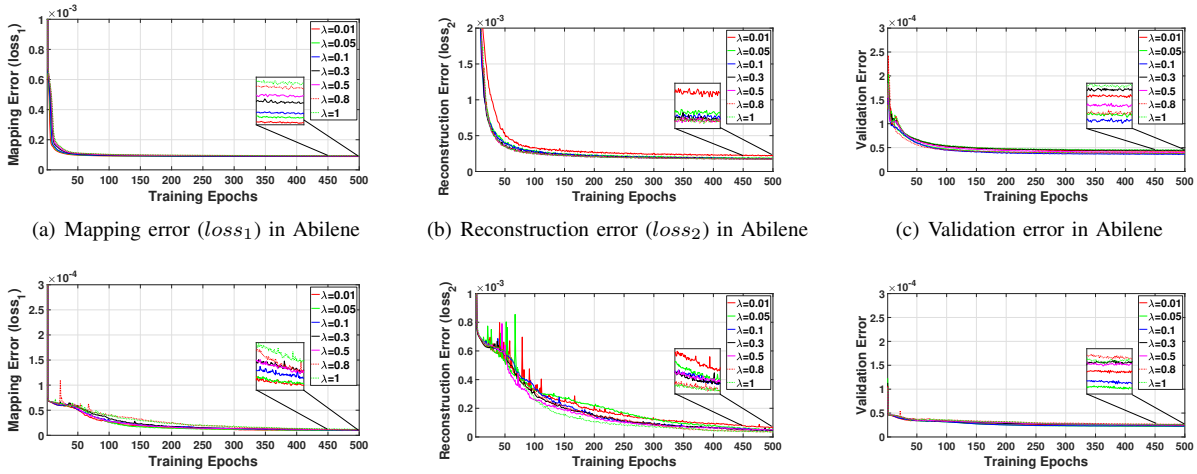
D. Model Details

For AutoTomo and AutoTomo-OS, the linear layers in the first and second non-linear unit contain 80 and 100 hidden nodes, respectively. The flatten layer contains 144 nodes. For the baseline methods, the model were built according to the settings mentioned in their respective works. Specifically, VAE has one decoder network and one encoder network, each of which has 4 convolutional layers and 2 dense layers. The number of dimensions of the latent space in VAE was set to 10. Each output was adjusted by 3000 iterations of gradient descent to approach the tomography equations. The network structure of DBN-based method contains 3 hidden layers, each of which has 100 hidden units. MNETME has one hidden layer with 12 units. The outputs of MNETME were adjusted by 200 rounds of EM iterations to approach the tomography equations. The initial learning rate of all networks is 10^{-3} according to their best performance. As the model of VAE is only applicable to Abilene dataset, we only present the results of VAE with Abilene dataset.

All gradient-descent optimizations were carried by the optimizer of Adam [39]. The maximum number of training epochs was set to 500. We adopted an early stopping strategy that the training process would be terminated if the training loss did not decrease in 50 consecutive epochs. All experiments were carried on a private server with 16-core CPUs at 2.6GHz, 64GB memory and 24GB GPU. The recorded results were averaged over 10 runs.

E. Results

1) Model Training under Different λ: Fig. 3 present the mapping error ($loss_1$), reconstruction error ($loss_2$) and the validation error (the error between an output and a real TM sample) of AutoTomo during the training process under different λ (from 0.01 to 1). The unknown rate q is set to 0.3. We can see from Fig. 3, with the number of training epochs going up, all three errors dropdown. That indicates AutoTomo can be effectively trained under the designed loss function. The training process of GÉANT is not as stable as that of Abilene due to the high dimensionality of TM samples in GÉANT dataset. When λ increases from 0.01 to 1, the mapping error ($loss_1$) increases whereas the reconstruction error decreases. That indicates turning up λ will let AutoTomo give more priority to reconstructing the link loads and less priority to mapping the observations. For example in Fig. 3(a) and Fig. 3(d), AutoTomo with $\lambda = 0.01$ achieves the least mapping error whereas in Fig. 3(b) and Fig. 3(e), $\lambda = 1$ reaches the least reconstruction errors. From Fig. 3(c) and Fig. 3(f), neither a small mapping error nor a small reconstruction error necessarily leads to a small validation error. In these



(a) Mapping error ($loss_1$) in Abilene (b) Reconstruction error ($loss_2$) in Abilene (c) Validation error in Abilene

(d) Mapping error ($loss_1$) in GÉANT (e) Reconstruction error ($loss_2$) in GÉANT (f) Validation error in GÉANT

Fig. 3. Three errors during AutoTomo training under different values of λ.

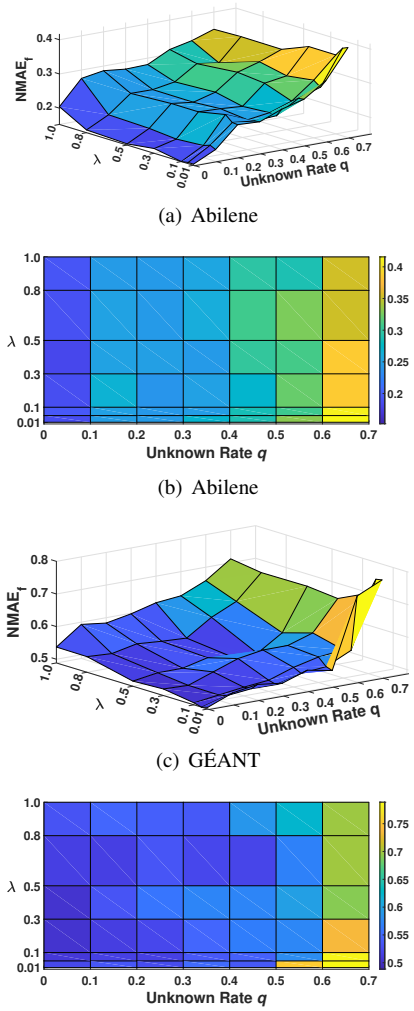
two figures, AutoTomo has the lowest validation error when $\lambda = 0.1$ in Abilene and $\lambda = 0.05$ in GÉANT.

Fig. 4 further reveals the performance of AutoTomo under different λ and different q. The performance of AutoTomo is qualified by the metric $NMAE_f$. It can be seen in Fig. 4 that both the values and the diversities of $NMAE_f$ increase with the growth of unknown rate q. Specifically, when q is in a small value, differences of λ seem not influence the performance of AutoTomo much. Whereas when q increases to a relatively large value, AutoTomo presents various performance under different λ. Nevertheless, we can see a phenomenon that when the unknown rate q is low, AutoTomo with a small λ can perform well. When the unknown rate is relatively large, AutoTomo with a larger λ is more likely to have a better performance. That means if the training set loses a large number of flow data, it is necessary to turn up the weight of the reconstruction error ($loss_2$) in the total loss function. In the following experiments, we uniformly set the value of λ to 1, and leave a more intelligent selection of λ in our future work. We can find in the following experiments that, AutoTomo can significantly outperform the baseline methods even when λ is fixed to 1.

2) Complete TM Estimation: Fig. 5 presents the $NMAE_f$ of all methods under different unknown rates. From the figures, AutoTomo and AutoTomo-OS significantly outperform other methods, especially when the unknown rate is in a high degree. The errors of our methods slightly grow with the increasing of the unknown rate while other methods have steep rises. MNETME performs better than DBN as it adjusts the outputs with tomography equations. VAE performs the worst, indicating the missing values in the training set seriously affect the reliability of the learned distribution on the latent space. So that its outputs suffer from low accuracy due to the biased distribution. With Abilene dataset, AutoTomo can improve the accuracy of the best-performed baseline by $15\% \sim 53\%$. With GÉANT dataset, all methods have inferior performances due to the high dimensionality and sample deficiency of GÉANT. That means all methods suffer from inadequate training. Nevertheless, AutoTomo can still improve the accuracy of the best baseline by $14\% \sim 37\%$. As AutoTomo-OS were trained based on selected observations, the accuracy of AutoTomo-OS can further improve AutoTomo by 18% in Abilene and 9% in GÉANT on average. The superior performance of our methods is owning to incorporating the optimizations of model parameters with the optimizations of unobserved flows, based on the designed loss function. When the model parameters as well as unobserved flows achieve compatible with each other after the optimizations, AutoTomo can always maintain a high accuracy even when 70% of flows cannot be observed in the training set. It is notable that, even when all flows can be observed (i.e., $q = 0$), AutoTomo can still improve the estimation accuracy by $14\% \sim 15\%$. That indicates the architecture of deep network (D) in AutoTomo is superior than that in the baseline methods on learning the inverse mapping.

Fig. 6 plots the TREs of all methods under three unknown rates. In both two datasets, the 7-days' samples (2016 samples in Abilene and 627 samples in GÉANT) were aggregated into 168 records (i.e., each result in Fig. 6 represents the averaged TRE per hour). From the figures, as the unknown rate increases, the gap between our methods and the other methods becomes even wider. Overall, our methods can at least improves the TRE of other methods by 43% in Abilene dataset, and 31% in GÉANT, averaging over all unknown rates. There is an interesting phenomena in Fig. 6 that: as the measurement time goes beyond, the curves of the three baselines are gradually showing periodicity. As we have known that, the real-world traffic usually varies periodically. That means these methods can only perform relatively well when the volumes of OD flows fall in a particular value range. In the contrast, AutoTomo has relatively stable performance throughout all measurement times.

Fig. 7 presents the SREs of these methods, where the flows were sorted by their averaged true volumes in an ascending order. With increasing of the unknown rate, the gaps between our methods and other methods gradually become obvious. From the figures, all methods have a good performance on



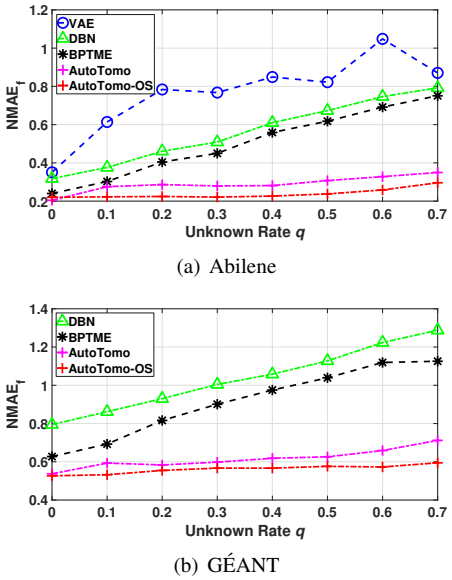
(a) Abilene

(b) Abilene

(c) GÉANT

(d) GÉANT

Fig. 4. The performance of AutoTomo with different λ under different unknown rates.



(a) Abilene

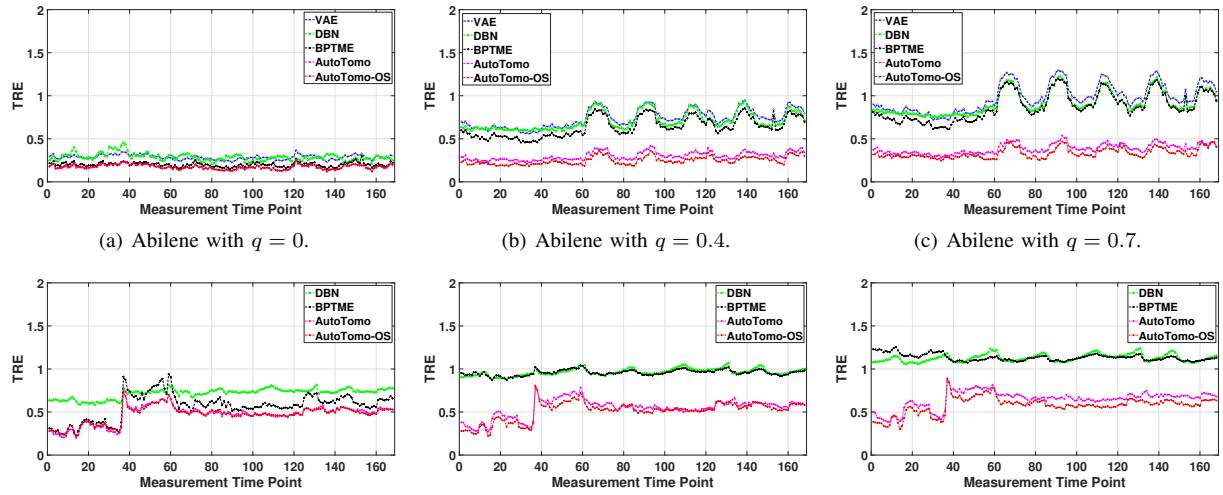
(b) GÉANT

Fig. 5. Errors for complete TM estimation under different unknown rates.

estimating small flows, especially in GÉANT dataset. That means flows with large volumes are generally difficult to estimate. From Fig. 7(d)∼(f), large flows in GÉANT have extremely high errors. That means the large averaged errors in Fig. 5(b) are mainly from the top 10% largest flows of GÉANT, which also indicates the deficient training may mainly affect the estimation performance on large flows rather than on small flows.

3) Unobserved Flows Completion: Fig. 8 plots the averaged estimation errors for unobserved flows under varied unknown rates. We added the curve of FGSR as a baseline to compare our methods with the-state-of-the-art MC-based method. From the figure, AutoTomo and AutoTomo-OS significantly outperform other methods among all unknown rates. Specifically, AutoTomo can reduce the estimation error of the best baseline by $28\% \sim 46\%$ in Abilene and $19\% \sim 30\%$ in GÉANT. The superior performance of our methods on estimating the unobserved flows is owning to incorporating the reconstruction loss on link loads, which forces the mapping module optimizing the outputs for unobserved flows in each training epoch. FGSR has the sub-optimal estimation accuracy because it utilizes the volumes of observed flows to continuously adjust the volumes of the unobserved flows through matrix factorization, until the TM can achieve the lowest rank. On the contrary, VAE, DBN and MNETME do not consider the optimization of the unobserved flows at all during the training process. AutoTomo-OS can further improve AutoTomo by 46% in Abilene and 59% in GÉANT. The significant improvement of AutoTomo-OS demonstrates our observation selection algorithm can select high quality observations, making the remaining unknown flows much easier to estimate.

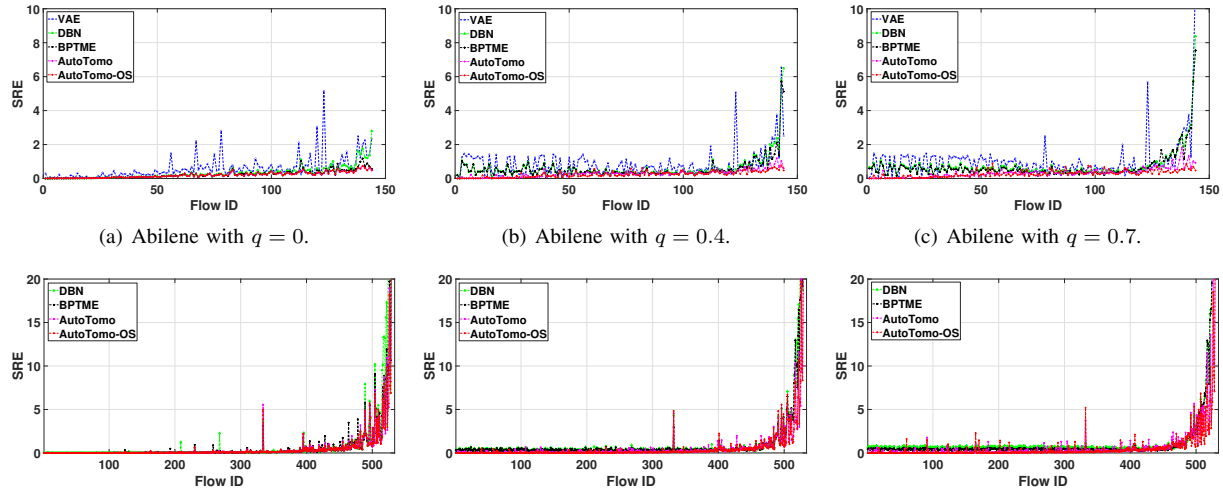
To further verify the performance of learning-based methods on learning the inverse mapping, we compare the learning processes of different models throughout all training epochs. The learning performance were measured by the averaged mean squared error (MSE) (i.e., $||\hat{X}' - X'||_2^2$) between the outputted flow volumes ($\hat{X}'$) and the true flow volumes (X'). We define the MSE on observed and unobserved flows as *observed flow errors* and *unobserved flow errors*, respectively. As the training objective of VAE is to learn the latent distribution instead of the inverse mapping, we only plot the curves of DBN and BPTME as baselines. Since AutoTomo-OS has different training data with the other methods, we only plot the curves of AutoTomo to ensure the fairness of comparison. Fig. 9 plots the *observed flow errors* and *unobserved flows errors* of different methods, respectively, during 500 training epochs, under the unknown rate of 0.5. From Fig. 9(a) and Fig. 9(c), the *observed flow errors* for all methods decline slowly with training epochs going on. For example, the error of MNETME is 2.1585×10^{-5}/ 3.2274×10^{-5} in the 200-th training epoch, and then reduces to 1.9381×10^{-5}/3.0382×10^{-5} in the 500-th epoch. For AutoTomo, the error is reduced from 1.9533×10^{-5}/3.0911×10^{-5} in the 200-th epoch to 1.8126×10^{-5}/3.0007×10^{-5} in the 500-th epoch. That means all methods can effectively learn the mappings to observed



(a) Abilene with $q = 0$. (b) Abilene with $q = 0.4$. (c) Abilene with $q = 0.7$.

(d) GÉANT with $q = 0$. (e) GÉANT with $q = 0.4$. (f) GÉANT with $q = 0.7$.

Fig. 6. Temporal estimation errors on three unknown rates.



(a) Abilene with $q = 0$. (b) Abilene with $q = 0.4$. (c) Abilene with $q = 0.7$.

(d) GÉANT with $q = 0$. (e) GÉANT with $q = 0.4$. (f) GÉANT with $q = 0.7$.

Fig. 7. Spatial estimation errors on three unknown rates.

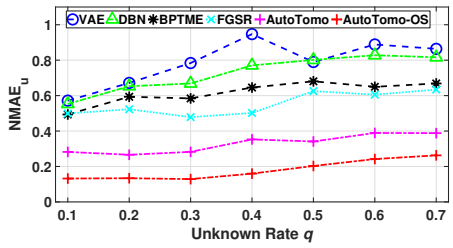
flows during model training.

In Fig. 9(b) and Fig. 9(d), we can see a significant gap between AutoTomo and the other two methods, indicating AutoTomo has a clear superiority on estimating the unobserved flows. Moreover, the *unobserved flow errors* of DBN and MNETME always keep stable or even slightly rise during the training epochs whereas that of AutoTomo maintains a downward trend. That demonstrates AutoTomo can constantly learn the mappings to unobserved flows during the training epochs while the baseline methods cannot. This is another evidence that optimizing the unobserved flows during the model training can significantly help the model learn the inverse mapping even with incomplete training data.

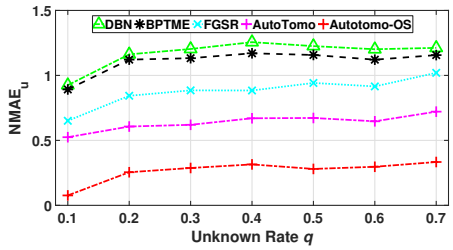
4) Training and Testing Time: Tab. II lists the training and testing time for each method. The recorded training time is the averaged time for training model with all training samples by one epoch. The recorded testing time is the averaged time for testing the model with one testing sample. The training time of these methods are correlated with the complexities of their learning models. As AutoTomo and AutoTomo-OS use the same model, their training and testing times are the same. From Tab. II, VAE has the longest training time due to its complicate network structure (8 convolutional layers and 4 dense layers). AutoTomo has comparable number of parameters with DBN, but has longer training time in Abilene.

TABLE II
AVERAGED TRAINING AND TESTING TIME

Methods	Abilene		GÉANT	
	Training Time (s)	Testing Time (ms)	Training Time (s)	Testing Time (ms)
VAE	8.64	83.09×10^3	-	-
DBN	3.52	0.024	1.28	0.037
MNETME	2.20	4.327	0.35	5.105
AutoTomo	6.36	0.032	0.93	0.036

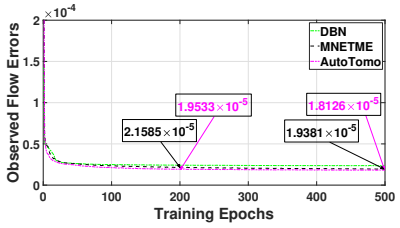


(a) Abilene

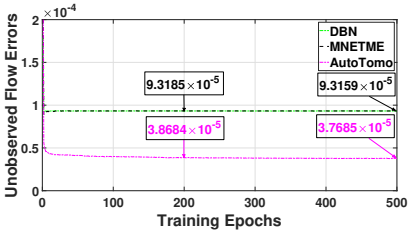


(b) GÉANT

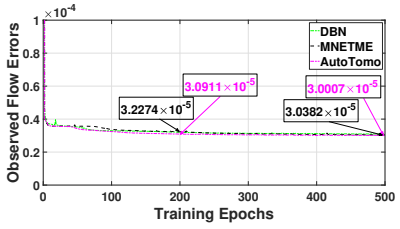
Fig. 8. Errors for unobserved flows completion under different unknown rates.



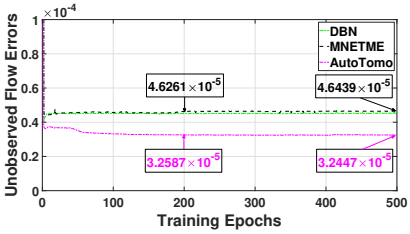
(a) Observed flow errors in Abilene.



(b) Unobserved flow errors in Abilene.



(c) Observed flow errors in GÉANT.



(d) Unobserved flow errors in GÉANT.

Fig. 9. Training effects on estimating observed and unobserved flows.

That is because the loss function of AutoTomo is a little more complicate than DBN. MNETME has the least training time because its model has the least number of parameters. All methods have lower training times in GÉANT due to the reduced number of training samples in GÉANT dataset. As DBN model is more sensitive to data dimension, the training time of DBN does not have much reduction since the samples in GÉANT have much higher dimensions than in Abilene.

VAE has extremely high testing time because it uses extra thousands of iterations to adjust each testing instance to be consistent with both the learned distribution and the tomography equations. As MNETME adjusts the outputs with hundreds of EM iterations to approach tomography equations, it has the second longest testing time. Although DBN has the shortest testing time, it suffers from low accuracy as well. As AutoTomo incorporates tomography constrains into the training process, it can efficiently output accurate estimations of OD flows without further computations. That indicates our method is more appropriate to real-time estimation tasks.

VIII. Conclusion

We presented AutoTomo, an innovative and cost-effective traffic estimator based on deep neural networks. Our approach addressed the challenges associated with collecting comprehensive training instances of complete Origin-Destination (OD) flows. AutoTomo leverages incomplete training data, where numerous OD flows lack any measurements, by learning the inverse mapping from link loads to OD flows. By simultaneously optimizing the model parameters and incorporating unobserved flows during the training process, AutoTomo achieves accurate and efficient estimation of both observed and unobserved flows using low-cost link load data. To further enhance AutoTomo's performance, we introduced an observation selection algorithm that assists network operators in identifying and measuring the most informative flows for model training. Through rigorous experimentation on two real-world traffic datasets, we compared AutoTomo with four recent classical methods. The results demonstrated the clear superiority of our method in scenarios with both complete and incomplete training instances. AutoTomo outperformed the alternatives, confirming its exceptional capability in accurately estimating traffic flows while operating under resource constraints.

Appendix

Proof for Theorem 1:

We first compute the MSE of f_2^* under the optimal X^{u*} as

follows.

$$\begin{aligned}
&\mathbb{E}_{S\in\mathcal{D}_{yx}}[||f_2^*(Y)-f_t^*(Y)||_2^2] \\
=&\int_S P(S)||f_2^*(Y)-f_t^*(Y)||_2^2 dS \\
=&\int_S P(S)||f_2^*(Y)-[X^o;X^{u*}]+[X^o;X^{u*}] \\
&-[X^o;X_t^u]+[X^o;X_t^u]-f_t^*(Y)||_2^2 dS \\
\leq&\int_S P(S)||f_2^*(Y)-[X^o;X^{u*}]||_2^2 dS+||X^{u*}-X_t^u|| \\
&+\int_S P(S)||f_t^*(Y)-[X^o;X_t^u]||_2^2 dS.
\end{aligned} \tag{28}$$

Let g_{max} denote the maximum distance between any two vectors in space $\mathcal{G}^u(S,\mathbf{A})$, i.e.,

$$g_{max}=\max_{X_i^u,X_j^u\in\mathcal{G}^u(S,\mathbf{A})}||X_i^u-X_j^u||_2^2. \tag{29}$$

Then Eqn. (28) can be written by

$$\begin{aligned}
&\mathbb{E}_{S\in\mathcal{D}_{yx}}[||f_2^*(Y)-f_t^*(Y)||_2^2] \\
\leq&\int_S P(S)||f_2^*(Y)-[X^o;X^{u*}]||_2^2 dS+g_{max} \\
&+\int_S P(S)||f_t^*(Y)-[X^o;X_t^u]||_2^2 dS \\
=&R_{\mathcal{D}_{yx}}(f_2^*,X^{u*})+R_{\mathcal{D}_{yx}}(f_t^*,X_t^u)+g_{max} \\
=&h(f_2^*,X^{u*}).
\end{aligned} \tag{30}$$

$\forall X_a^u: X_a^u\in\mathcal{G}^u(S,\mathbf{A}), X_a^u\neq X^{u*}$, the MSE of f_a^* satisfies the similar inequality

$$\begin{aligned}
&\mathbb{E}_{S\in\mathcal{D}_{yx}}[||f_a^*(Y)-f_t^*(Y)||_2^2] \\
\leq&R_{\mathcal{D}_{yx}}(f_a^*,X_a^u)+R_{\mathcal{D}_{yx}}(f_t^*,X_t^u)+g_{max} \\
=&h(f_a^*,X_a^u).
\end{aligned} \tag{31}$$

As (f_2^*,X^{u*}) is an optimal solution that minimizes $R_{\mathcal{D}_{yx}}(f,X^u)$, it always holds that $\forall X_a^u: X_a^u\in\mathcal{G}^u(S,\mathbf{A}), X_a^u\neq X^{u*}$, $R_{\mathcal{D}_{yx}}(f_2^*,X^{u*})\leq R_{\mathcal{D}_{yx}}(f_a^*,X_a^u)$. Hence, we have $h(f_2^*,X^{u*})\leq h(f_a^*,X_a^u)$.□

Proof for Theorem 3:

(1) Eqn. (13) can be written by

$$\begin{aligned}
&R_{\mathcal{S}_{yx}}^u(f) \\
=&\frac{1}{T}\sum_{t=1}^T||\mathbf{A}\cdot[X^o(t);\hat{X}^u(t)]-Y(t)||_2^2 \\
=&\frac{1}{T}\sum_{t=1}^T||\mathbf{A}_u(\hat{X}^u(t)-\tilde{X}^u(t))||_2^2 \\
=&\frac{1}{T}\sum_{t=1}^T||\mathbf{A}_u\cdot u(t)||_2^2,
\end{aligned} \tag{32}$$

where $\tilde{X}^u(t)$ is an arbitrarily vector in space $\mathcal{G}^u(S_t,\mathbf{A})$, and $u(t)=\hat{X}^u(t)-\tilde{X}^u(t)$ denotes the difference between $\hat{X}^u(t)$ and $\tilde{X}^u(t)$. It is easy to get that $R_{\mathcal{S}_{yx}}^u$ is concave with only one minimum over all values of $u(t)$. That means if $R_{\mathcal{S}_{yx}}^u(f)$ can achieve its minima, $\forall t: t\in\{1,2,\ldots,T\}$, $u(t)$ will be most close to zero. Let $[\hat{X}_4^{o*}(t);\hat{X}_4^{u*}(t)]=f_4^*(Y(t))$, $t=\{1,2,\ldots,T\}$. If f_4^* can minimize Eqn. (32), $\tilde{\mathbf{X}}^{u*}$ will be the vector in $\mathcal{G}^u(\mathbf{S},\mathbf{A})$ that is most close to the output $\hat{\mathbf{X}}_4^{u*}$. As f_4^* can minimize Eqn. (12) at the same time, $R_{\mathcal{S}_{yx}}(f_4^*,\tilde{\mathbf{X}}^{u*})=R_{\mathcal{S}_{yx}}^o(f_4^*)+||\hat{\mathbf{X}}_4^{u*}-\tilde{\mathbf{X}}^{u*}||_2^2$ can achieve the minimum value of $R_{\mathcal{S}_{yx}}(f,\mathbf{X}^u)$, i.e., f_4^* is an optimal hypothesis for *Problem 3*.

(2) Let $J(v)=\mathbf{A}_u^T\mathbf{A}_u\cdot v$, $\{\lambda_1,\lambda_2,\ldots,\lambda_{|u|}\}$ denote the eigenvalues of symmetric matrix $\mathbf{A}_u^T\mathbf{A}_u$, and $\{\xi_1,\xi_2,\ldots,\xi_{|u|}\}$ denote the corresponding orthogonal eigenvectors. According to the properties of symmetric matrix, for any vector $v=\sum_{i=1}^{|u|}v_i\xi_i$ we have

$$||v||_2^2\leq\frac{1}{\lambda_{min}^2}||\mathbf{A}_u^T\mathbf{A}_u\cdot v||_2^2\leq\frac{||\mathbf{A}_u||_2^2}{\lambda_{min}^2}||\mathbf{A}_u\cdot v||_2^2. \tag{33}$$

By plugging Eqn. (32) into Eqn. (33),

$$\frac{1}{T}\sum_{t=1}^T||u(t)||_2^2\leq\frac{||\mathbf{A}_u||_2^2}{\lambda_{min}^2}\cdot\frac{1}{T}\sum_{t=1}^T||\mathbf{A}_u\cdot u(t)||_2^2 \tag{34}$$

$$<\frac{||\mathbf{A}_u||_2^2\cdot\epsilon_u}{\lambda_{min}^2}. \tag{35}$$

We plug f_4^* and $\tilde{\mathbf{X}}^{u*}$ into the *empirical risk* of *Problem 3*:

$$\begin{aligned}
&R_{\mathcal{S}_{yx}}(f_4^*,\tilde{\mathbf{X}}^{u*}) \\
=&R_{\mathcal{S}_{yx}}^o(f_4^*)+\frac{1}{T}\sum_{t=1}^T||\hat{X}_4^{u*}(t)-\tilde{X}^{u*}(t)]||_2^2 \\
<&\epsilon_o+\frac{||\mathbf{A}_u||_2^2\cdot\epsilon_u}{\lambda_{min}^2}.\square
\end{aligned} \tag{36}$$

Proof for Theorem 5:

Suppose the optimal FV in the linear system is $X^{f*}\subseteq X$. For any $x_i\in X$, if $x_i\in X^{f*}$, then $TVR(x_i)=TNR(x_i)$. As the FV involved in the calculation of $TVR(x_i)$ is itself alone, the results hold.

If $x_i\notin X^{f*}$, then x_i is a PV. Suppose $x_i=\sum_{j=i_1}^{i_n}a_jx_j^{f*}+c_i$, where $\{x_{i_1}^{f*},\ldots,x_{i_n}^{f*}\}$ are involved FVs that can linearly represent x_i, $\{a_{i_1},\ldots,a_{i_n}\}$ and c_i are coefficients and constant, respectively. Then $TVR(x_i)$ is calculated by plugging $\{TNR(x_{i_1}^{f*}),\ldots,TNR(x_{i_n}^{f*})\}$ into the linear representation. Specifically, for each $x_j^{f*},j=\{i_1,\ldots,i_n\}$, if the coefficient $a_j>0$, we plug the lower bound (upper bound) of $TNR(x_j^{f*})$; otherwise we plug the upper bound (lower bound) of $TNR(x_j^{f*})$ into the linear representation to compute the lower bound (upper bound) of $TVR(x_i)$. Therefore, the length of $TVR(x_i)$ is proportional to the number of optimal FVs involved as well as the lengths of the TNRs of the involved FVs.□

REFERENCES

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

[2] B. Claise, "Cisco systems netflow services export version 9," Tech. Rep., 2004.

[3] Í. Cunha, F. Silveira, R. Oliveira, R. Teixeira, and C. Diot, "Uncovering artifacts of flow measurement tools," in *Passive and Active Network Measurement: 10th International Conference, PAM 2009, Seoul, Korea, April 1-3, 2009. Proceedings 10*. Springer, 2009, pp. 187–196.

[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] P.-W. Tsai, C.-W. Tsai, C.-W. Hsu, and C.-S. Yang, "Network monitoring in software-defined networking: A review," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3958–3969, 2018.

[7] J. Galán-Jiménez, M. Polverini, and A. Cianfrani, "A scalable and error-tolerant solution for traffic matrix assessment in hybrid ip/sdn networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 251–264, 2019.

[8] Y. Chen, S. Bhojanapalli, S. Sanghavi, and R. Ward, "Coherent matrix completion," in *International Conference on Machine Learning*. PMLR, 2014, pp. 674–682.

[9] 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.

[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] G. Kakkavas, D. Gkatzioura, V. Karyotis, and S. Papavassiliou, "A review of advanced algebraic approaches enabling network tomography for future network infrastructures," *Future Internet*, vol. 12, no. 2, p. 20, 2020.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[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] 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.

[20] 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.

[21] 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.

[22] J. Fan, L. Ding, Y. Chen, and M. Udell, "Factor group-sparse regularization for efficient low-rank matrix recovery," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[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] 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.

[26] 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.

[27] M. Emami, R. Akbari, R. Javidan, and A. Zamani, "A new approach for traffic matrix estimation in high load computer networks based on graph embedding and convolutional neural network," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 6, p. e3604, 2019.

[28] P. Le Nguyen, Y. Ji *et al.*, "Deep convolutional lstm network-based traffic matrix prediction with partial information," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 261–269.

[29] 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.

[30] M. Anthony, P. L. Bartlett, P. L. Bartlett *et al.*, *Neural network learning: Theoretical foundations*. cambridge university press Cambridge, 1999, vol. 9.

[31] M. M. Bejani and M. Ghatee, "A systematic review on overfitting control in shallow and deep neural networks," *Artificial Intelligence Review*, pp. 1–48, 2021.

[32] S. Bera, S. Misra, and A. Jamalipour, "Flowstat: Adaptive flow-rule placement for per-flow statistics in sdn," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 530–539, 2019.

[33] X. Wang, Q. Deng, J. Ren, M. Malboubi, S. Wang, S. Xu, and C.-N. Chuah, "The joint optimization of online traffic matrix measurement and traffic engineering for software-defined networks," *IEEE/ACM transactions on networking*, vol. 28, no. 1, pp. 234–247, 2019.

[34] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "Opentm: traffic matrix estimator for openflow networks," in *Passive and Active Measurement: 11th International Conference, PAM 2010, Zurich, Switzerland, April 7-9, 2010. Proceedings 11*. Springer, 2010, pp. 201–210.

[35] M. Malboubi, L. Wang, C.-N. Chuah, and P. Sharma, "Intelligent sdn based traffic (de) aggregation and measurement paradigm (istamp)," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 934–942.

[36] C. Feng, L. Wang, K. Wu, and J. Wang, "Bound-based network tomography with additive metrics," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 316–324.

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

[38] 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.

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