Diffusion Models Meet Network Management: Improving Traffic Matrix Analysis with Diffusion-based Approach

Xinyu Yuan†, Yan Qiao†, Zhenchun Wei†, Zeyu Zhang‡, Minyue Li†, Pei Zhao†, and Rongyao Hu†
†School of Computer Science and Information Engineering, Hefei University of Technology, China
‡Faculty of Information Technology, Macau University of Science and Technology, China

***Abstract*—Due to network operation and maintenance relying heavily on network traffic monitoring, traffic matrix analysis has been one of the most crucial issues for network management related tasks. However, it is challenging to reliably obtain the precise measurement in computer networks because of the high measurement cost, and the unavoidable transmission loss. Although some methods proposed in recent years allowed estimating network traffic from partial flow-level or link-level measurements, they often perform poorly for traffic matrix estimation nowadays. Despite strong assumptions like low-rank structure and the prior distribution, existing techniques are usually task-specific and tend to be significantly worse as modern network communication is extremely complicated and dynamic. To address the dilemma, this paper proposed a diffusion-based traffic matrix analysis framework named Diffusion-TM, which leverages problem-agnostic diffusion to notably elevate the estimation performance in both traffic distribution and accuracy. The novel framework not only takes advantage of the powerful generative ability of diffusion models to produce realistic network traffic, but also leverages the denoising process to unbiasedly estimate all end-to-end traffic in a plug-and-play manner under theoretical guarantee. Moreover, taking into account that compiling an intact traffic dataset is usually infeasible, we also propose a two-stage training scheme to make our framework be insensitive to missing values in the dataset. With extensive experiments with real-world datasets, we illustrate the effectiveness of Diffusion-TM on several tasks. Moreover, the results also demonstrate that our method can obtain promising results even with 5% known values left in the datasets.**

***Index Terms*—diffusion models, deep learning, network traffic matrix, network tomography, network management.**

I. Introduction

A traffic matrix (TM) is applied to track the traffic volumes between all possible pairs of network nodes, which are usually mentioned as origin to destination (OD) flows [1]. It is a critical input for many network management tasks, including capacity planning, anomaly detection, and traffic engineering [2]. For example, the traffic measurement can help with facing collisions, congestion in network [3], security hazards [4], and inefficient utilization of network resources [5].

There are two types of methods to obtain the crucial network measurement: TM Completion and Network Tomography [6]. The first one is a direct TM measurement through flow-level monitoring tools, such as Cisco's NetFlow/TMS [7], and OpenTM in the emerging software-defined network (SDN) [8]. Unfortunately, with the continuous expansion of network scale, the complete measurement of these OD flows requires extremely high administrative costs as well as computational overhead [9]. Moreover, not all devices in legacy networks can support SDN modules [10]. Thus the collected flow data is usually partial, and obtaining a complete TM is still an open challenge. Based on low-rank assumptions of real-world TMs, recent studies generally use matrix or tensor completion algorithms to recover the traffic data from sparse known entries [11, 12]. Although these methods only require very few sampling entities to obtain traffic data satisfying the low-rank characteristics, their rationality and usefulness are still limited. Despite extremely low efficiency as large amounts of data need to be processed, they rely heavily on sparse assumptions while foregoing the usage of some crucial information from the whole system. To be more specific, firstly, these solutions estimate the conditional mean of the observed samples and can only work when the application data follow the Gaussian distributions [13], but can not handle a more complicated traffic data distribution. Secondly, they did not take the low-cost link load data and its corresponding routing information that plays a generally significant role in network management into consideration, leading to these useful and easily accessible resources not being utilized.

The second way is to estimate the flow-level traffic from the link-level measurements by means of Network Tomography (NT). This method infers fine-grained OD flows by solving a group of linear equations that involve both coarse-grained link loads and flow routing matrix. However, the key problem for NT-based traffic matrix estimation (TME) is that such linear equations are usually highly rank deficient, which means there is no unique solution of OD flows corresponding to the measured link loads. To tackle this problem, numerous types of research explore the pattern of OD flows and transform the ill-posed inverse problem into a constrained inverse problem, for decades years, so that the variables of OD flows may have a unique solution. In the early years, the solutions to the NT problem are provided based on the assumptions about the prior information of traffic data, such as each OD flow follows a Poisson distribution [14] or Gaussian distribution [15]. Some also suggested the independence of source and destination which is equivalent to a gravity model [16]. However, these methods typically have low accuracy as the unrealistic assumptions are not valid. Then with the development of deep learning, various neural networks have been built to learn

the inverse mapping from link loads to OD flows [17, 18]. Regardless of whether it has modeled the distribution of estimations or not, the simple deep-learning method is able to reconstruct the dynamic properties of network traffic via link counts and routing information without additional hypotheses. However, the solution must ensure that the routing matrix used for training and inference is consistent. The condition serves as a cornerstone in enabling NT-based traffic estimation since it is almost impossible to keep the routing information static nowadays, for example, routers configured with an adaptive routing policy often choose routing paths dynamically based on current network loads. And similar to current TM completion based methods, all these methods discussed above failed to leverage additional information. Their goal is just to solve the tomography equation, and output the complete TM by inputting only the link load data. It means that even knowing more than half of the OD pairs will not have any impact on the results, whereas they would greatly reduce the solution space.

To sum up the current works, their problems can be divided into three categories: (1) Unable to capture the traffic data distribution; (2) Unable to harness known traffic and routing information simultaneously, thus lack of flexibility; (3) Unsatisfactory prerequisites such as low-rank feature, prior distribution, and immutable routing. Fortunately, the development of deep generative models provides a positive answer to these fundamental questions. One is allowed to train a variational autoencoder (VAE) or generative adversarial network (GAN) to estimate TMs with the learned distribution and adjust the outputs by tomography equations or other constraints [19]. Although promising, they require a large amount of complete measurements which can be hardly obtained, for training the underlying network. And despite the efficiency of iterative adjustments, either VAE or GAN has its own defects that prevent them from accurately approximating the complicated distribution. As the dimensions of TMs keep growing nowadays, VAEs that rely on a surrogate loss tend to produce unrealistic samples, while GANs are known for potentially unstable training and mode collapse problem due to their adversarial nature [20].

Recently, diffusion models (DMs) [21] have emerged as a new paradigm for generative models, theoretically underpinned by non-equilibrium thermodynamics [22] and score-matching network [23]. They are gaining significant popularity and nearly replacing GANs and VAEs in a wide range of synthesis domains owing to their superior sampling quality and stable training dynamics [24]. Nevertheless, this comes at the expense of poor scalability and increased sampling times due to the long Markov chain sequences required. Therefore, applying DMs directly to deal with the TME problem in a cyclical manner like the previous generative-model-based method is intractable [25], as the computing cost can be extremely huge for traffic matrix in large-scale IP backbone networks.

However, we noticed that samples obtained from the DMs depend on the initial state of the simple distribution and each transition. It makes diffusion models strong candidates for producing TMs that satisfy the conditions imposed by the set of measurements via a "plug-and-play" approach that combines the diffusion model and the measurement process [26–28]. Thus to bridge research gaps for solving both NT and TMC problems, this paper focuses on a DM-based framework, which can model complex behaviors of real-world activities and generate high-quality traffic matrix given partial link loads and (or) OD pairs. Formally, we propose a versatile solution for various TM-related tasks in network management, which we call Diffusion-TM. By refining OD flows at each state during the reverse diffusion sampling, our solution only requires an off-the-shelf diffusion model to yield realistic and data-consistent results, without any extra training nor needing any modifications to model structures. Also note that most deep generative models are sensitive to massive data missing, while any large set of TM measurements is bound to have a significant number of missing values in the real-world networks, the challenge here is how to leverage known information to keep the distribution of training set as accurately as possible. To answer the practical requirements, we further designed an efficient two-stage strategy to alleviate the effect of missing values in diffusion model training and allowed them to be performed even when as much as over 95% of the data is missing. We demonstrate the effectiveness of our method on three different tasks: TM estimation, completion, and synthesis.

The key contributions of this paper can be summarized as follows:

- We not only propose a **diffusion**-based approach for IP-network **t**raffic **m**atrix analysis called Diffusion-TM, but also theoretically prove its efficiency on recovering traffic matrices while capturing the traffic data distribution via a novel approximation. To the best of our knowledge, this is one of the first works that leverage DMs to analyze traffic matrix.
- To ensure that the result of Diffusion-TM conforms to the desired distribution across the whole range of missing values scenarios, we provide a two-stage training scheme with additional pre-processing work and missing data aware objective. The results suggest that the framework can be applied when large amounts of missing data exist in training datasets.
- We conduct extensive experiments with real-world traffic trace data to evaluate the effectiveness of our approach in a wide range of TM-related problems including network tomography, traffic recovery, and synthetic data generation.

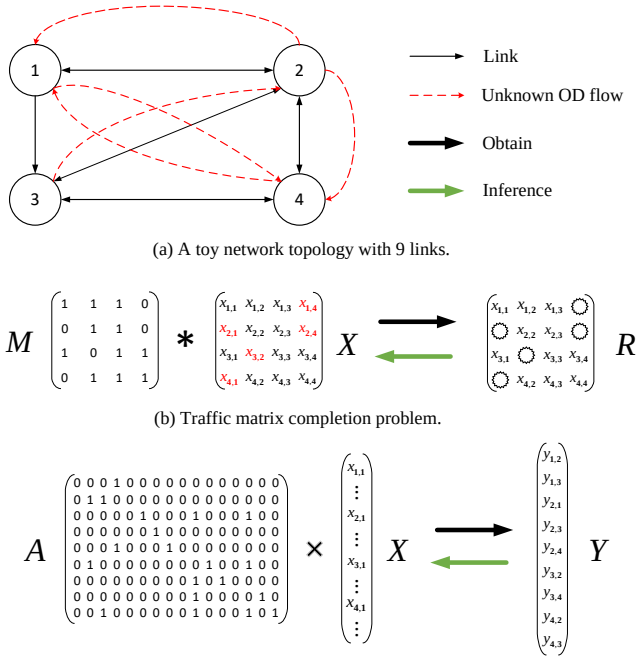
The rest of the article is organized as follows. We discuss in Section II the existing literature about the TM measurement problem. In Section III and Section IV, we introduce relevant background and basic concepts, respectively. In Section V, we transform the original problem into an approximation problem and explain why it takes effect. We then formally introduce our DM-based approach and the model structure in Section VI. We evaluate the performance of the proposed Diffusion-TM through extensive experiments in Section VII. Finally, We conclude the article in Section VIII.

II. Related Work

Existing TM estimation methods can be classified into two categories. In the first category, traffic matrix completion (TMC) was proposed to recover the missing entries from a low-rank matrix with the development of sparse techniques. Some works [29–31] show that TM data may have spatio-temporal correlations, thus the matrix potentially has the low rank characteristic. Zhang et al. [29] proposed a sparsity regularized SVD method to estimate the missing values, then they improved the algorithm by proposing sparsity regularized matrix factorization (SRMF) in [31]. However, the performance of two-dimensional matrix-based data recovery methods is relatively low due to the matrix's limitations in information extraction. For a better data recovery, Xie et al. [11, 12, 32] start to model the network traffic data as a higher dimensional array called tensor and propose algorithms based on tensor completion for more accurate missing data recovery. Despite its effectiveness, the accuracy of current tensor completion based solutions is still low as they can only capture linear and simple correlations and is not applicable to complex distribution of the network measurements. Furthermore, these methods have not taken other useful information (link loads and topology) of the backbone network into consideration.

The second category uses network tomography (NT) to estimate TM from the link loads by solving the linear equations [33]. Since the linear system is generally rank deficient, NT-based methods must impose additional assumptions on the TM to obtain a unique solution. The accuracies of these methods heavily rely on the underlying assumptions. For example, Vardi et al. [33] assumed that the traffic followed the Poisson distribution, and Zhang et al. [34] imposed a gravity model on the TM estimation. Different from these solutions, the application of deep learning algorithms has appeared as a viable approach recently. [35] used graph embedding to integrate the network topology with the model input. [36] utilized convolutional neural network (CNN) and long short-term memory (LSTM) network to exploit the spatio-temporal correlations within TM. In order to improve the accuracy of future traffic estimation, the method proposed in [37] combined the forward and backward Convolutional LSTM (ConvLSTM) network to correct the input TM data. And [17, 18] introduced a back-propagation neural network (BPNN) to estimate TM. Although their performance does not rely on additional assumptions or the spatial-temporal structure of TM, these techniques still struggle with distribution alignment.

Over the past few years, deep generative models have granted people the ability to model traffic data distribution in a variety of ways. Xie et al. [13] designed a Deep Adversarial Tensor Completion (DATC) scheme based on GAN. Besides, a matrix completion and prediction algorithm based on a combination of generative autoencoders and Hidden Markov Models was proposed in [38]. The methods in [19, 39] used a VAE and GAN to learn the latent distribution that is "similar" to the training set of TM, respectively. Because such Plug-and-Play methods allow network to generalize to any partial or/and link-level measurement during inference, they are emerging alternative paradigms for generative modeling traffic matrix.



(a) A toy network topology with 9 links.

(b) Traffic matrix completion problem.

(c) Network tomography problem.

Fig. 1. **Illustration of studied problems in this paper.** We seek an estimated TM $\boldsymbol{X}$ that satisfies the conditions imposed by the set of measurements $\boldsymbol{R}$ or $\boldsymbol{Y}$. However, the considered problem is highly underdetermined .

The approach leverages the prior (i.e. Gaussian) space of a pre-trained generative model to solve inverse problems in a zero-shot way. More concretely, they first train a generator (or decoder) network, then optimize TME objective function through gradients of data which can be easily computed by the chain rule. Therefore, estimations can be updated iteratively by using simple stochastic gradient descent. However, either VAE or GAN has its inherent model defects: VAE tends to produce unrealistic and blurry samples, meanwhile, the training of GAN is often unstable [25].

This paper studies an entirely new approach, namely Diffusion-TM, to handle the network measurement problems using diffusion models. DMs have a spectacular ability to capture both diversity and fidelity [40]. Moreover, our method is not trained for any specific target, and instead, we take full advantage of the prior unconditional model, so each traffic matrix is optimized independently. To our best knowledge, only algorithm in [25] tends to use DMs for traffic data estimation. But different from [25] repeating the sampling process several times, our proposed method conducts estimation through the reverse diffusion process itself and needs to run along the Markov chain just once. Additionally, we theoretically prove that such sampling strategy is the key to significantly improving the performance of traffic reconstruction. To overcome the problem of missing data, we also designed a novel two-stage training scheme so that our Diffusion-TM can directly use the incomplete network traffic data to train its off-the-shelf denoiser.

III. System Model and Problem Formulation

We start by introducing the basic notations and definitions in this section. Then, we will present the problem formulation

in our article.

A. System Model

In our system model, we consider the network graph as $G = (V, E)$ where V and E are network nodes and links, respectively. The TM is defined as a $|V| \times |V|$ matrix where each entry represents an OD flow between a pair of nodes in the network. To facilitate calculations, we reshape TM to a vector $\{X_{1:N}\}$, where $N = |V| \times |V|$ is the total number of OD flows in TM. We denote a sequence of TMs from time point 1 to T as $\boldsymbol{X} = \{X_{1:N,1:T}\}$.

B. Problem Formulation

As shown in Fig. 1, the problem refers to the inference of unmeasured network attributes based on measurements realized at a subset of accessible network elements. Let $\boldsymbol{Y} = \{Y_{1:M,1:T}\}$ denote the sequence of link loads, and $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ denote the routing matrix, where each entry a_{ij} of $\boldsymbol{A}$ has a binary value (0 or 1). For deterministic routing policy, if the j-th flow traverses the i-th link, then $a_{ij} = 1$; otherwise, $a_{ij} = 0$. For probabilistic routing policy (such as ECMP), the value of a_{ij} is within the range of $[0, 1]$, representing the probability that the j-th flow may transverse the i-th link. The relationship between TM $\boldsymbol{X}$ and link load $\boldsymbol{Y}$ can be formulated as the linear equations:

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{Y}. \tag{1}$$

In 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.

As inferring $\boldsymbol{X}$ from the compressed measurements $\boldsymbol{Y}$ is a severely underdetermined task, a more general approach is the direct flow-level measurements although only for partial traffic volumes. Let us denote an observation mask as $\boldsymbol{M} = \{m_{1:N,1:T}\} \in \{0, 1\}^{N \times T}$ where $m_{n,t} = 1$ if $x_{n,t}$ is observed, and $m_{n,t} = 0$ if $x_{n,t}$ is unobserved. Consequently, the known-measurement matrix $\boldsymbol{R}$ which denotes the set of information that is available, is defined as

$$\boldsymbol{M} \odot \boldsymbol{X} = \boldsymbol{R}, \tag{2}$$

where $\odot$ represents elementwise multiplication.

Problem: Now the estimation problem for TMs can be defined as follows: given the measurement $\{\boldsymbol{Y}, \boldsymbol{R}\}$ obeying Eqn. 1 and/or Eqn. 2, we aim to accurately recover the unknown traffic data, with $\{\boldsymbol{A}, \boldsymbol{M}\}$ known. And we have a set of linear constraints on the TM

$$\boldsymbol{\mathcal{A}}(\boldsymbol{X}) + \boldsymbol{z} = \boldsymbol{\mathcal{Y}} \tag{3}$$

where $\boldsymbol{\mathcal{A}}(\cdot)$ is a linear operator, the matrix $\boldsymbol{\mathcal{Y}}$ contains the available measurements, and $\boldsymbol{z}$ is the measurement noise as the Simple Network Management Protocol (SNMP) used for collecting link measurements is often noisy [41] and flow-level collection usually involves sampling at quite high rates. Specifically, we consider white Gaussian noise $\boldsymbol{z} \sim \mathcal{N}\left(0, \sigma_z^2\right)$ in this work. In later parts of the paper, we may also denote the linear operation as $\boldsymbol{\mathcal{Y}} = \boldsymbol{H}\boldsymbol{X} + \boldsymbol{z}$, where $\boldsymbol{\mathcal{A}}(\boldsymbol{X})$ is replaced by a matrix operation $\boldsymbol{H}\boldsymbol{X}$.

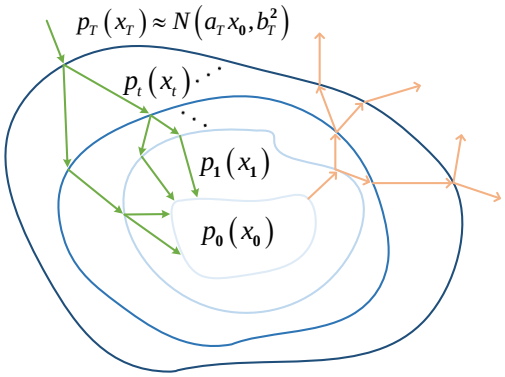


Fig. 2. **Geometrical visualization of diffusion models.** The central area represents the original data manifold which has been proved to be encircled by manifolds of noisy data $p_t(\boldsymbol{x_t})$ [42]. The encoding (forward) process depicted by orange arrows gradually converts original data distribution $p_0(\boldsymbol{x_0})$, into a simple isotropic Gaussian $\mathcal{N}(0, \mathrm{I})$. While the decoding (reverse) process depicted by green arrows can be considered as transitions from $p_t(\boldsymbol{x_t})$ to $p_{t-1}(\boldsymbol{x_{t-1}})$ through a Markov Chain.

Target: The aim of this work is to sample points from data distribution conditioned on partially observed traffic or/and link measurements. We formulate the traffic estimation problem as a penalized least-squares problem, i.e.

$$\min_{\boldsymbol{x}} \|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{A}}(\boldsymbol{x})\|_2^2 - 2 \cdot \sigma_z^2 \log p_0(\boldsymbol{x}), \tag{4}$$

where we model the estimation $\boldsymbol{x}$ as being drawn from prior distribution with density $p_0(\boldsymbol{x})$. The objective can also be written as the following form with so-called posterior probability density:

$$\max_{\boldsymbol{x}} \log p(\boldsymbol{x}|\boldsymbol{y}), \text{ s.t. } \boldsymbol{y} : \boldsymbol{\mathcal{A}}(\boldsymbol{x}) + \boldsymbol{z} = \boldsymbol{\mathcal{Y}} \tag{5}$$

which treats this as a maximum likelihood problem. Heuristically, we choose the solution that best fits prior distribution while satisfying the constraints.

IV. Preliminaries of Diffusion Models

Diffusion model is a class of deep generative models that parameterize the encoding (forward) and decoding (reverse) processes via a diffusion process. As shown in Fig. 2, the key idea behind them is to add small amounts of noise gradually to a data point for transitions from data manifold to noisy manifolds, then train the underlying neural network to transport from the pure Gaussian to the clean area through inverting the diffusion process. In this section, we will introduce score-based diffusion models and its special case, denoising diffusion probabilistic models (DDPMs) in sequence.

A. Score-based Diffusion Models

Suppose $p_0(\boldsymbol{x_0})$ be the d-dimensional data distribution. We consider the general class of score-based diffusion models in a continuous form that can be described with the following Ornstein-Uhlenbeck stochastic differential equation (SDE) [43]

$$d\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, t)\, dt + g(t)\, d\boldsymbol{w} \tag{6}$$

where $\boldsymbol{w}$ denotes the standard Brownian motion, $\boldsymbol{f} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, $g : \mathbb{R} \to \mathbb{R}$ is the linear drift function and scalar

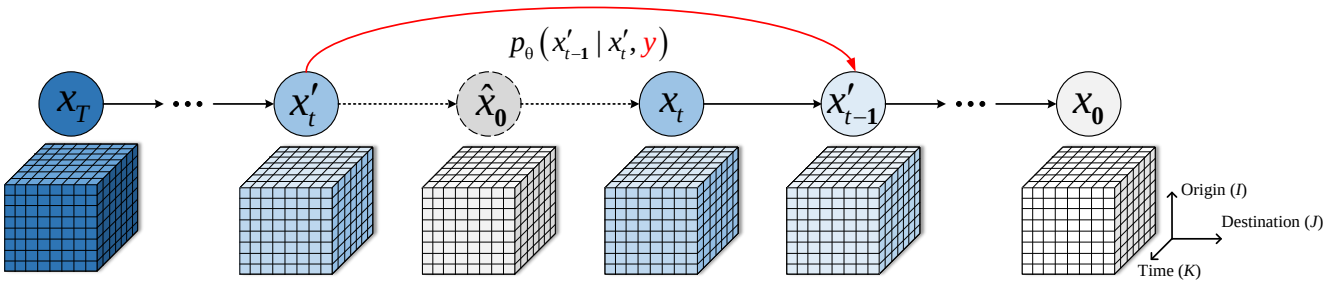


Fig. 3. **Illustration of our diffusion-based approach for solving TM estimation problems.** The reverse inference process (from right to left) iteratively denoises the target traffic matrix $\boldsymbol{x_0}$ conditioned on the measurement $\boldsymbol{y}$. Concretely, following the prediction of the estimated $\hat{\boldsymbol{x}}_\mathbf{0}$ by an unconditional diffusion model, the measurement $\boldsymbol{y}$ is incorporated by solving a proximal subproblem depicted by red arrows in the VP-SDE.

diffusion coefficient, respectively. The SDE results in a series of marginal distributions $\{p_t(\boldsymbol{x_t})\}$ where $t \in [0, T]$, so that $\boldsymbol{x_T} \sim \mathcal{N}\left(0, \sigma_T^2\right)$ with some constants $\sigma_T > 0$.

It is known that the density evolution process can be reversed by another SDE with the Anderson's theorem [44]

$$d\boldsymbol{x} = \left[\boldsymbol{f}(\boldsymbol{x}, t) - g^2(t)\nabla_{\boldsymbol{x_t}} \log p_t(\boldsymbol{x_t})\right] dt + g(t)\, d\bar{\boldsymbol{w}} \quad (7)$$

where $\bar{\boldsymbol{w}}$ is a Brownian motion running backward in time from T to 0. Then diffusion models approximate this reverse process by learning the score function $\nabla_{\boldsymbol{x_t}} \log p_t(\boldsymbol{x_t})$ through a score-matching loss:

$$\mathbb{E}_{t, \boldsymbol{x_0}, \boldsymbol{x_t}|\boldsymbol{x_0}}\left[\lambda_t \| s_\theta(\boldsymbol{x_t}, t) - \nabla_{\boldsymbol{x_t}} \log p_t(\boldsymbol{x_t})\|_2^2\right] \quad (8)$$

where λ_t is a positive weight coefficient, and t is uniformly sampled over $[0, T]$. With a learned score function $s_\theta(\boldsymbol{x_t}, t)$, one can then solve the reverse SDE through

$$d\boldsymbol{x} = \left[\boldsymbol{f}(\boldsymbol{x}, t) - g^2(t)\, s_\theta(\boldsymbol{x_t}, t)\right] dt + g(t)\, d\bar{\boldsymbol{w}} \quad (9)$$

by first sampling from the prior Gaussian $p_T(\boldsymbol{x_T})$.

B. Denoising Diffusion Probabilistic Models

In this paper, we focus on Denoising Diffusion Probabilistic Models [21] which is equivalent to the variance preserving form of the SDE (VP-SDE). To be more specific, we have the forward and reverse SDEs as the continuous version of the diffusion process in DDPM with the choice of $\boldsymbol{f}(\boldsymbol{x}, t) = -\beta(t)\boldsymbol{x}/2$ and $g(t) = \sqrt{\beta(t)}$. The corresponding forward SDE can be formulated as the following:

$$d\boldsymbol{x} = -\frac{\beta(t)}{2}\boldsymbol{x}dt + \sqrt{\beta(t)}d\boldsymbol{w}. \quad (10)$$

In particular, the forward process of DDPMs gradually corrupts original data $x_0 \in \mathbb{R}^d$ via a fixed Markov chain $x_0, \ldots, x_T$ with each variable in $\mathbb{R}^d$ as follows:

$$\begin{cases} \boldsymbol{x_t}|\boldsymbol{x_{t-1}} \sim \mathcal{N}\left(\boldsymbol{x_t}; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)\mathrm{I}\right), \\ \boldsymbol{x_t}|\boldsymbol{x_0} \sim \mathcal{N}\left(\boldsymbol{x_t}; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathrm{I}\right), \\ \boldsymbol{x_t} = \sqrt{\bar{\alpha}_t}\boldsymbol{x_0} + \sqrt{1-\bar{\alpha}_t}\epsilon \end{cases} \quad (11)$$

with $\epsilon \sim \mathcal{N}(0, \mathrm{I})$, $\alpha_t := 1 - \beta_t$ where $\beta_t \in (0, 1)$ is a variance at diffusion step t, scaling factor $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$.

Starting from the Gaussian noise $\boldsymbol{x_T}$, we can run the reverse process parametrized by the model $p_\theta(\boldsymbol{x_{t-1}}|x_t) := \mathcal{N}(\boldsymbol{x_{t-1}}; \mu(\boldsymbol{x_t}, t, \theta), \boldsymbol{\Sigma}(\boldsymbol{x_t}, t, \theta))$ to get $\boldsymbol{x_0}$. The diffusion model is trained to maximize the marginal likelihood of the data $\mathbb{E}_{\boldsymbol{x_0}}[\log p_\theta(\boldsymbol{x_0})]$, and we can write the variational lower bound (VLB) with KL divergence as follows:

$$\begin{aligned} \boldsymbol{\mathcal{L}}_{vlb} := & \underbrace{-\log p_\theta(\boldsymbol{x_0}|\boldsymbol{x_1})}_{\boldsymbol{\mathcal{L}}_0} + \underbrace{\mathcal{D}_{KL}(p(\boldsymbol{x_T}|\boldsymbol{x_0}) \;||\; p(\boldsymbol{x_T}))}_{\boldsymbol{\mathcal{L}}_T} \\ & + \sum_{t=2}^{T} \underbrace{\mathcal{D}_{KL}(p(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}) \;||\; p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}))}_{\boldsymbol{\mathcal{L}}_{t-1}}. \end{aligned} \quad (12)$$

The main objective is a sum of independent terms $\boldsymbol{\mathcal{L}}_{t-1}$. There are many different ways to parameterize the posterior mean $\mu(\boldsymbol{x_t}, t, \theta)$, and the most obvious option is to predict $\mu(\boldsymbol{x_t}, t, \theta)$ directly:

$$\boldsymbol{\mathcal{L}}_o := \mathbb{E}_{t, \boldsymbol{x_0}}\left[\frac{1}{2\boldsymbol{\Sigma}^2(\boldsymbol{x_t}, t, \theta)}\|\hat{\mu}(\boldsymbol{x_t}, \boldsymbol{x_0}) - \mu(\boldsymbol{x_t}, t, \theta)\|^2\right], \quad (13)$$

where $\boldsymbol{\Sigma}(\boldsymbol{x_t}, t, \theta)$ is often set to pre-defined time dependent constants, and $\hat{\mu}(\boldsymbol{x_t}, \boldsymbol{x_0})$ is the mean of the posterior $p(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}, \boldsymbol{x_0})$ which are defined as follows:

$$\hat{\mu}(\boldsymbol{x_t}, \boldsymbol{x_0}) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x_0} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x_t}. \quad (14)$$

Alternatively, the network could also predict $\boldsymbol{x_0}$ or $\boldsymbol{\epsilon}$ using 11 and 14. In this work, we train our model to reconstruct input $\boldsymbol{x_0}$ itself combined with a reweighted loss function:

$$\boldsymbol{\mathcal{L}}_{simple} = \mathbb{E}_{t, \epsilon, \boldsymbol{x_0}}\left[\|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_\mathbf{0}(\boldsymbol{x_t}, t, \theta)\|^2\right]. \quad (15)$$

The connection between the score function and the prediction in DDPMs can be formulated approximately as: $s_\theta(\boldsymbol{x_t}, t) \approx \left(\boldsymbol{x_t} - \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{x}}_\mathbf{0}(\boldsymbol{x_t}, t, \theta)\right) / (1-\bar{\alpha}_t)$ [21, 45]. Data generation through denoising depends on the score function and can be seen as noise conditional score-based generation. Then the reverse process $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})$ in DDPMs can be written as follows:

$$\begin{aligned} \boldsymbol{x_{t-1}} = & \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{\boldsymbol{x}}_\mathbf{0}(\boldsymbol{x_t}, t, \theta) \\ & + \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\boldsymbol{z_t}. \end{aligned} \quad (16)$$

Algorithm 1 Diffusion-TM Sampling for Network Tomography

Require: scale coefficients $\{\rho_t\}_{t=1}^T$, link loads $\boldsymbol{Y}$, and routing matrix $\boldsymbol{A}$
Ensure: estimated TM $\hat{\boldsymbol{x}}_{\mathbf{0}}$
1: $\boldsymbol{x_T} \sim \mathcal{N}(0, \mathrm{I})$;
2: **for** all t from T to 1 **do**
3: $z \sim \mathcal{N}(0, \mathrm{I})$;
4: $\hat{s}_\theta \leftarrow$ **Score** $(\hat{\boldsymbol{x}}_{\mathbf{0}}(\boldsymbol{x_t}, t, \theta), t)$;
5: $\boldsymbol{x'_{t-1}} = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x_t} - (1-\alpha_t) s_\theta(\boldsymbol{x_t}, t)) + \sigma_t z$;
6: $\hat{\boldsymbol{x}}_{\mathbf{0}} \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}(\boldsymbol{x_t} + (1-\bar{\alpha}_t)\hat{s}_\theta)$;
7: $\boldsymbol{x_{t-1}} = \boldsymbol{x'_{t-1}} + \rho_t \nabla_{\boldsymbol{x_t}} \|\boldsymbol{Y} - \boldsymbol{A}\hat{\boldsymbol{x}}_{\mathbf{0}}\|_2^2$;
8: **end for**
9: $\hat{\boldsymbol{x}}_{\mathbf{0}} \leftarrow$ **EM_Optimization** $(\hat{\boldsymbol{x}}_{\mathbf{0}}, \boldsymbol{Y})$;
10: **return** $\hat{\boldsymbol{x}}_{\mathbf{0}}$;

Algorithm 2 Diffusion-TM Sampling for Traffic Matrix Completion

Require: scale coefficients $\{\rho_t\}_{t=1}^T$, observed TM $\boldsymbol{X}^o$, and observation matrix $\boldsymbol{M}$
Ensure: estimated TM $\hat{\boldsymbol{x}}_{\mathbf{0}}$
1: $\boldsymbol{x_T} \sim \mathcal{N}(0, \mathrm{I})$;
2: **for** all t from T to 1 **do**
3: $z \sim \mathcal{N}(0, \mathrm{I})$;
4: $\hat{s}_\theta \leftarrow$ **Score** $(\hat{\boldsymbol{x}}_{\mathbf{0}}(\boldsymbol{x_t}, t, \theta), t)$;
5: $\boldsymbol{x'_{t-1}} = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x_t} - (1-\alpha_t) s_\theta(\boldsymbol{x_t}, t)) + \sigma_t z$;
6: $\hat{\boldsymbol{x}}_{\mathbf{0}} \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}(\boldsymbol{x_t} + (1-\bar{\alpha}_t)\hat{s}_\theta)$;
7: $\boldsymbol{x_{t-1}} = \boldsymbol{x'_{t-1}} + \rho_t \nabla_{\boldsymbol{x_t}} \|\boldsymbol{M} \odot \boldsymbol{X}^o - \boldsymbol{M} \odot \hat{\boldsymbol{x}}_{\mathbf{0}}\|_2^2$;
8: $\hat{\boldsymbol{x}}_{\mathbf{0}} \leftarrow$ **Replace** $(\hat{\boldsymbol{x}}_{\mathbf{0}}, \boldsymbol{X}, \boldsymbol{M}, t)$;
9: **end for**
10: **return** $\hat{\boldsymbol{x}}_{\mathbf{0}}$;

In order to sample with diffusion models more quickly, [46] proposed Denoising Diffusion Implicit Models (DDIMs) that can be rewritten as:

$$\begin{aligned}\boldsymbol{x_{t-1}} =& \sqrt{1 - \frac{\sigma_{\psi_t}^2}{1-\bar{\alpha}_t}}\left(\boldsymbol{x_t} - \sqrt{\bar{\alpha}_t}\hat{\boldsymbol{x}}_{\mathbf{0}}(\boldsymbol{x_t}, t, \theta)\right) \\ &+ \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{x}}_{\mathbf{0}}(\boldsymbol{x_t}, t, \theta) + \sigma_{\psi_t}\boldsymbol{z_t}\end{aligned} \tag{17}$$

where σ_{ψ_t} controls the stochastic degree of the diffusion process. Compared to DDPM, DDIM extended from Markovian to non-Markovian is able to generate higher-quality samples using a much fewer number of sampling steps.

V. Problem Variation and Solution

This section explores improving the analysis of traffic matrices with diffusion models which has demonstrated very appealing performance in general distribution modeling. We propose thinking an approach for refining the reverse process of an unconditional diffusion model for TM-related tasks. Given the learned TM distribution $p(\boldsymbol{x})$ of DMs, the main challenge of the problem is how to conduct the mapping from $\boldsymbol{y}$ to $\boldsymbol{x}$ without explicit information on the conditional probability $p(\boldsymbol{x}|\boldsymbol{y})$. Below we first decompose our target step by step, then we re-formulate and solve the problem by leveraging the Tweedie's method [47]. And finally, we theoretically show that one can find a solution to both constraint and original data consistency, so the result becomes more accurate and stable.

A. Main Idea

Shown as Eqn. 5, both traffic matrix completion and traffic tomography problem could be divided into constrained generation tasks, the goal is to produce TMs from the posterior distribution $p(\boldsymbol{x_0}|\boldsymbol{y})$ given the condition $\boldsymbol{y}$ which could be observed entities or/and link loads. Therefore, we consider rewriting Eqn. 7 as follows for conditional transition

$$d\boldsymbol{x} = \left[\boldsymbol{f}(\boldsymbol{x}, t) - g^2(t)\nabla_{\boldsymbol{x_t}} \log p_t(\boldsymbol{x_t}|\boldsymbol{y})\right] dt + g(t)\, d\bar{\boldsymbol{w}}. \tag{18}$$

Leveraging the diffusion model as the prior, the question here is how to compute the conditional score $\nabla_{\boldsymbol{x_t}} \log p_t(\boldsymbol{x_t}|\boldsymbol{y})$.

We start with the problem-specific score which can be decomposed via Bayes' rule as below:

$$\nabla_{\boldsymbol{x_t}} \log p_t(\boldsymbol{x_t}|\boldsymbol{y}) = \nabla_{\boldsymbol{x_t}} \log p_t(\boldsymbol{x_t}) + \nabla_{\boldsymbol{x_t}} \log p_t(\boldsymbol{y}|\boldsymbol{x_t}), \tag{19}$$

where the first term can be approximated by a pre-trained score function $s_\theta(\boldsymbol{x_t}, t)$, and the second is a guidance term which is intractable to compute because there is no explicit dependence between $\boldsymbol{x_t}$ and $\boldsymbol{y}$.

Thus, we have to resort to approximate $\nabla_{\boldsymbol{x_t}} \log p_t(\boldsymbol{y}|\boldsymbol{x_t})$ to circumvent using the likelihood term directly. Note that the forward diffusion is able to be represented by Eqn. 11, we can consider an independent graphical model: $\boldsymbol{x_0} \rightarrow \boldsymbol{y},\ \boldsymbol{x_0} \rightarrow \boldsymbol{x_t}$. Then, by factorizing $p(\boldsymbol{y}|\boldsymbol{x_t})$ as follows:

$$\begin{aligned}p(\boldsymbol{y}|\boldsymbol{x_t}) &= \int p(\boldsymbol{y}|\boldsymbol{x_0}, \boldsymbol{x_t})\, p(\boldsymbol{x_0}|\boldsymbol{x_t})\, d\boldsymbol{x_0} \\ &= \int p(\boldsymbol{y}|\boldsymbol{x_0})\, p(\boldsymbol{x_0}|\boldsymbol{x_t})\, d\boldsymbol{x_0},\end{aligned} \tag{20}$$

we can now transform the problem into approximating another intractable $p(\boldsymbol{x_0}|\boldsymbol{x_t})$ as the likelihood of $p(\boldsymbol{y}|\boldsymbol{x_0})$ is tractable in general.

B. Approximation Problem

Here, our solution to above issue is to use the specialized representation of the posterior mean to obtain reasonable approximations to the true $p(\boldsymbol{x_0}|\boldsymbol{x_t})$ through a generalization of *Tweedie's Formula*.

Lemma 1 (**Tweedie's Formula**): *Given $\boldsymbol{\eta} \sim g(\cdot)$, suppose $p(\boldsymbol{x}|\boldsymbol{\eta})$ belong to the exponential family distribution*

$$p(\boldsymbol{x}|\boldsymbol{\eta}) = p_0(\boldsymbol{x}) \exp\left(\boldsymbol{\eta}^T F(\boldsymbol{x}) - \psi(\boldsymbol{\eta})\right), \tag{21}$$

the unique posterior mean $\hat{\boldsymbol{\eta}}$ of $p(\boldsymbol{\eta}|\boldsymbol{x})$ will satisfy

$$(\nabla_{\boldsymbol{x}} F(\boldsymbol{x}))^T \hat{\boldsymbol{\eta}} = \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p_0(\boldsymbol{x}), \tag{22}$$

where $F(\boldsymbol{x})$ is the function of $\boldsymbol{x}$, $\boldsymbol{\eta}$ the natural or canonical parameter of the family, $\psi(\boldsymbol{\eta})$ the cumulant generating function (which makes the density $p(\boldsymbol{x}|\boldsymbol{\eta})$ integrate to 1), and $p_0(\boldsymbol{x})$ the density when $\boldsymbol{\eta} = 0$.

Conclusion: *If we suppose that $\boldsymbol{\eta}$ has been sampled from a prior distribution $g(\boldsymbol{\eta})$, and $\boldsymbol{x}|\boldsymbol{\eta} \sim \mathcal{N}(\boldsymbol{\eta}, \sigma^2)$ has been observed, we can write:*

$$\mathbb{E}(\boldsymbol{\eta}|\boldsymbol{x}) = \boldsymbol{x} + \sigma^2 \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}). \tag{23}$$

that comes from rewriting Tweedie's formula where

$$F(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\sigma^2},\ \ \psi(\boldsymbol{\eta}) = \frac{\boldsymbol{\eta}}{2\sigma^2},\ \text{and}\ \ p_0(\boldsymbol{x}) \sim \mathcal{N}(0, \sigma^2). \tag{24}$$

The proof of Lemma 1 can be found in Appendix. And the conclusion tells us that one can achieve the denoised result

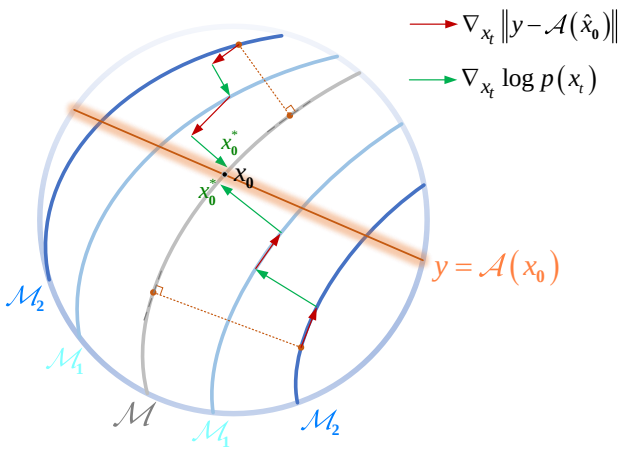


Fig. 4. **Guiding generation process toward target solutions.** Each curve represents a manifold $\mathcal{M}_i$ of (noisy) TM data. The proposed correction step (red arrow) alleviates reverse diffusion step (green arrow) leaving the solution space of inverse problems.

by computing the posterior expectation, then associated with diffusion models, we have a classic result of the formula.

Proposition 1: *Suppose* $s_\theta(\boldsymbol{x_t}, \boldsymbol{t})$ *minimizes the score matching loss* $\mathbb{E}_{t,\boldsymbol{x_0},\boldsymbol{x_t}|\boldsymbol{x_0}}\left[\lambda_t\|s_\theta(\boldsymbol{x_t},t)-\nabla_{\boldsymbol{x_t}}\log p_t(\boldsymbol{x_t})\|_2^2\right]$. *For the case of reverse diffusion sampling,* $p(\boldsymbol{x_0}|\boldsymbol{x_t})$ *has the unique posterior mean*

$$\hat{\boldsymbol{x}}_\mathbf{0} = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\boldsymbol{x_t} + (1-\bar{\alpha}_t)\, s_\theta(\boldsymbol{x_t}, \boldsymbol{t})\right) \tag{25}$$

The proof of Proposition 1 can be found in Appendix. We are now able to approximate $p(\boldsymbol{x_0}|\boldsymbol{x_t})$ with the mapping function $\mathcal{D}_t : \boldsymbol{x_t} \to \boldsymbol{x_0} := \hat{\boldsymbol{x}}_\mathbf{0}$. Hence, the likelihood function $p(\boldsymbol{y}|\boldsymbol{x_t})$ can be replaced with $p(\boldsymbol{y}|\hat{\boldsymbol{x}}_\mathbf{0})$. Formally, we have the following approximation

$$\nabla_{\boldsymbol{x_t}} p(\boldsymbol{y}|\boldsymbol{x_t}) \simeq \nabla_{\boldsymbol{x_t}} p(\boldsymbol{y}|\hat{\boldsymbol{x}}_0)\,, \text{ where } \hat{\boldsymbol{x}}_\mathbf{0} := \mathbb{E}[\boldsymbol{x_0}|\boldsymbol{x_t}] \tag{26}$$

As this paper considers cases with Gaussian noise, the likelihood function $p(\boldsymbol{y}|\boldsymbol{x_0})$ should satisfy $\mathcal{N}(\mathcal{A}(\boldsymbol{x_0}), \sigma_z^2)$. Then using Eqn. 26, we get

$$\nabla_{\boldsymbol{x_t}} \log p(\boldsymbol{y}|\boldsymbol{x_t}) \simeq -\frac{1}{\sigma_z^2}\nabla_{\boldsymbol{x_t}}\|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{x}}_\mathbf{0})\|_2^2\,. \tag{27}$$

Now putting Eqn. 19 and Eqn. 27 together, the discrete reverse diffusion under the additional guidance can be represented by

$$\begin{aligned} \boldsymbol{x'_{t-1}} &= \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x_t} - (1-\alpha_t) s_\theta(\boldsymbol{x_t}, t)\right) + \sigma_t z, \\ \boldsymbol{x_{t-1}} &= \boldsymbol{x'_{t-1}} + \rho_t \nabla_{\boldsymbol{x_t}}\|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{x}}_\mathbf{0})\|_2^2 \end{aligned} \tag{28}$$

where $\rho_t \triangleq (1-\alpha_t)/\left(\sqrt{\alpha_t}\sigma_z^2\right)$ is set as the strength of the guidance.

C. Theoretical Guarantee

Theorem 1: *The correction imposed by the gradient-based guidance at each step will not leave the data manifold* $\mathcal{M} \subset \mathbb{R}^n$ *which is the set of all traffic data points* $\boldsymbol{x_0}$.

The proof of Theorem 1 can be found in Appendix which is mainly concluded from [42]. It indicates that reasonable estimations of all flows can be obtained by combining the score function and our gradient term, given the measurement model. Concretely, the illustration of our sampling method is demonstrated in Fig. 3, and we also present our scheme visually in Fig. 4. The term can be considered as forcing the diffusion model to search for the optimal solution along the data manifold at every noisy state space.

VI. Diffusion-TM: Diffusion-based Approach for General Traffic Matrix Analysis

Based on the discussion in Section V, we summarize the detailed sampling algorithm of our method called Diffusion-TM, and list them for traffic tomography and TM completion in Alg. 1 and Alg. 2. Additionally, we further improve the algorithm by adding the expectation maximization iteration [48] and additional replace-based guidance [43] in Alg. 1 and Alg. 2 respectively, as there is still room for imposing some optimizations. The detailed descriptions will be shown later in this section.

Since these algorithms only require a pre-trained unconditional diffusion model, our proposed approach is a plug-and-play framework which does not depend on specific applications. This makes the model amenable to three different TM-related tasks at the same time: synthetic TM generation, TM recovery and tomography. Therefore, the choice of model architecture and training strategy for the key DM is an important issue.

First, note that there is no shortage of studies proving the network monitoring data usually have hidden spatio-temporal redundancies [31, 49] which propels us to designing models that take TM series as input for learning. And we choose a Transformer [50] that enhances the models' ability to capture global correlation and patterns of TM sequences. As aforementioned, obtaining the complete training set of the OD traffic is usually difficult or even impossible, let alone building a set of continental sequences. To alleviate the effect of missing values in the training set, we designed a pre-processing workflow based on an autoencoder network. The module provides Diffusion-TM a coarse-grained estimation of missing OD flows to make the underlying network as insensitive to these missing values as possible. And for both two networks, only the estimation loss on observed TM is used in back propagation to update their weights.

A. Expectation Maximization (EM) Algorithm for Tomography

The Expectation Maximization (EM) algorithm can be regarded as a solution to an optimization problem with latent variables. It adopts an iterative procedure to calculate the Maximum Likelihood (ML) estimation [48, 51, 52]. Specifically, we follow the canonical form of the EM iteration for solving the NT problem proposed by [48] as follows:

$$x_j \leftarrow \frac{x_j}{\sum_{i=1}^{M} a_{ij}} \sum\nolimits_{i=1}^{M} \frac{a_{ij} * y_i}{\sum_{k=1}^{N} a_{ik} * x_k} \tag{29}$$

where $a_{i,j}$ represents the value located in the i-th row and j-th column of the routing matrix. And x_j is the j-th element of OD flows X, y_i is the i-th element of link loads Y. The EM algorithm can approximate the solution of Eqn. 1 as the iteration running on [18]. Thus as shown in Alg. 1 line 9, we select the iterative procedure to further optimize the estimation generated by our diffusion model.

B. Additional Replace-based Guidance for Completion

From the above part, the EM algorithm is designed to help constrain the possible values of estimation through tomography equations. But *how can additional optimization be applied to completion tasks?*

We first define the known and unknown OD-pairs of $\boldsymbol{x_t}$ as $\Omega(\boldsymbol{x_t})$ and $\bar{\Omega}(\boldsymbol{x_t})$ respectively. For the traffic matrix completion task, our goal is to sample from $p\left(\bar{\Omega}(\boldsymbol{x_0})\,|\Omega(\boldsymbol{x_0})=\boldsymbol{y}\right)$. A notable property of such task is that we can run the reverse process only to known dimensions since the element-wise forward noise is applied to the dimensions independently. Now back to Eqn, 18, we again focus on the likelihood $p_t(\boldsymbol{x_t}|\boldsymbol{y})$ which is then equal to $p_t\left(\bar{\Omega}(\boldsymbol{x_t})\,|\Omega(\boldsymbol{x_t})=\boldsymbol{y}\right)$. Formally, we have

$$\begin{aligned}
&p_t\left(\bar{\Omega}(\boldsymbol{x_t})\,|\Omega(\boldsymbol{x_0})=\boldsymbol{y}\right) := p_t\left(\bar{\Omega}(\boldsymbol{x_t})\,|Y\right)\\
&=\int p_t\left(\bar{\Omega}(\boldsymbol{x_t})\,|\Omega(\boldsymbol{x_t}),Y\right)p_t\left(\Omega(\boldsymbol{x_t})\,|Y\right)d\Omega(\boldsymbol{x_t})\\
&=\mathbb{E}_{\Omega(\boldsymbol{x_t})|Y}\left[p_t\left(\bar{\Omega}(\boldsymbol{x_t})\,|\Omega(\boldsymbol{x_t}),Y\right)\right]\\
&=\mathbb{E}_{\Omega(\boldsymbol{x_t})|Y}\left[p_t\left(\bar{\Omega}(\boldsymbol{x_t})\,|\Omega(\boldsymbol{x_t})\right)\right]\\
&\approx p_t\left(\bar{\Omega}(\boldsymbol{x_t})\,|\hat{\Omega}(\boldsymbol{x_t})\right)=p_t\left(\left[\bar{\Omega}(\boldsymbol{x_t});\Omega(\boldsymbol{x_t})\right]\right),
\end{aligned}\tag{30}$$

where $\hat{\Omega}(\boldsymbol{x_t})$ denotes samples from $p_t\left(\Omega(\boldsymbol{x_t})\,|\Omega(\boldsymbol{x_0})=\boldsymbol{y}\right)$, and $\left[\bar{\Omega}(\boldsymbol{x_t});\Omega(\boldsymbol{x_t})\right]$ represents the concatenation of two sets of dimensions.

That means there is space for conducting additional constraints while still leveraging the unconditional score function, and [43] proposed this general method for imputation from the jointly trained diffusion model.

To flesh this out, the samples for $\Omega(\boldsymbol{x_t})$ are replaced by exact samples from the forward process $q(\Omega(\boldsymbol{x_t})\,|\Omega(\boldsymbol{x_0}))$ in Eqn. 11, at each iteration, while the sampling procedure for updating $\bar{\Omega}(\boldsymbol{x_t})$ is still sampling from $p_\theta(\bar{\Omega}(\boldsymbol{x_t})\,|\bar{\Omega}(\boldsymbol{x_{t+1}}))$. The samples $\Omega(\boldsymbol{x_t})$ then have the correct marginal distribution, and $\bar{\Omega}(\boldsymbol{x_t})$ will conform with $\Omega(\boldsymbol{x_t})$ through the denoising process. Using this strategy, we can generate an intact sample that follows the correct conditional distribution in addition to the correct marginal. We refer to the approach as the replacement method for extra guidance during the reverse process, and run it at the end of each sampling step in Alg. 2.

C. Transformer-based Underlying Network

We use a Transformer with Sigmoid Non-linear processing the final output to estimate $\hat{\boldsymbol{x}}_{\mathbf{0}}(\boldsymbol{x_t},t,\theta)$. As shown in Fig. 5 Model 2, the underlying network is divided into two modules, i.e., a transformer encoder and a transformer decoder. Both the encoder and decoder network consist of multiple transformer blocks. Each transformer block in encoder (decoder) contains a full attention (a full attention, a cross attention to combine encoding information) and a feed forward layer. For the diffusion embedding, we follow previous works [21, 53] with transformer sinusoidal positional embedding to encode the diffusion step. After getting the diffusion step embedding, we sum them up and add them to each block. Specifically, the diffusion step t is injected into the network using the Adaptive Layer Normalization operator, which can be written as $a_t\mathrm{Laynorm}(w)+b_t$ where w is the intermediate activations, a_t and b_t are obtained from a linear projection of the diffusion embedding.

D. Training Procedure under Traffic-deficient Setting

Our training goal is to obtain a converging diffusion model for stably generating high-quality TM samples, which in this paper is tantamount to easing the effect of missing values in the measured data. According to our solution overview above, the distribution of traffic data can be hard to fit for deep generative models if only very few of the OD flows were observed, while the TM is generally incomplete with lots of entries in the matrix unobserved under the measurements. Thus to minimize the influence, a pre-processing module for missing data in the training set is needed. Since traffic matrices with unobserved volumes are often close to complete data after dimensionality reduction [54], we choose an autoencoder (AE) to obtain coarse-grained estimations of unknown flows before formal training of DMs. As illustrated in Fig. 5 Model 1, the module contains one encoder which is composed of two fully connected layers using ReLU as Non-linear function, and one decoder which is responsible for mapping the outputs of representation space onto the original space through Sigmoid regularization. Between the two of them, a bi-directional Recurrent Neural Network (Bi-RNN), which can be implemented with either LSTM or GRU units, is established to extract per-point features. The input TM sequences after downscaling are fed to the Bi-RNN, and then their temporal information about the observed time points will be stored and passed through our decoder to produce the first-stage output.

Specifically, given a TM series represented by a sequence of points $\boldsymbol{s}=\{s_1,s_2,\ldots,s_K\}$, the output of the AE can be computed by

$$\begin{aligned}
h_i^1&=\mathrm{ReLU}\left(W_1 s_i+B_1\right),\\
h_i^2&=\mathrm{ReLU}\left(W_2 h_i^1+B_2\right),\\
\left[h_i^3,c_i\right]&=\textbf{Bi-RNN}\left(h_{i-1}^2,\left[h_{i-1}^3,c_{i-1}\right]\right),\\
f_i&=\mathrm{Sigmoid}\left(W_3 h_i^3+B_3\right)
\end{aligned}\tag{31}$$

where h and c are the hidden states and the optional cell states, W and B are the weights and biases of a fully connected layer. Respectively, $\mathrm{ReLU}(\cdot)$ is the activation function of ReLU, and $\mathrm{Sigmoid}(\cdot)$ is the activation function of Sigmoid. The loss function in the pre-processing work can be written as follows:

$$\mathcal{L}_p=\|\boldsymbol{X}^o\odot\boldsymbol{M}-D\left(\boldsymbol{X}^o\right)\odot\boldsymbol{M}\|_2^2\tag{32}$$

where $\boldsymbol{X}^o$ and $\boldsymbol{M}$ is a partially observed traffic data and sampling indication matrix, respectively. During the pre-training process, the loss is back propagated to deep network D to update its parameters. The empirical loss guarantees that only the estimation loss on observed samples is used in back propagation, while the training errors for missing data are discarded. After pre-processing module training, we leveraged it to update the measured traffic set by replacing all the missing OD pairs with values reconstructed through our autoencoder. Then, the "complete" dataset would be used to boost the distribution learning of Diffusion-TM.

In the diffusion probabilistic model training procedure, we first sample the diffusion step t from a uniform distribution, and then compute its corresponding $\boldsymbol{x_t}$ through Eqn. 11 with a

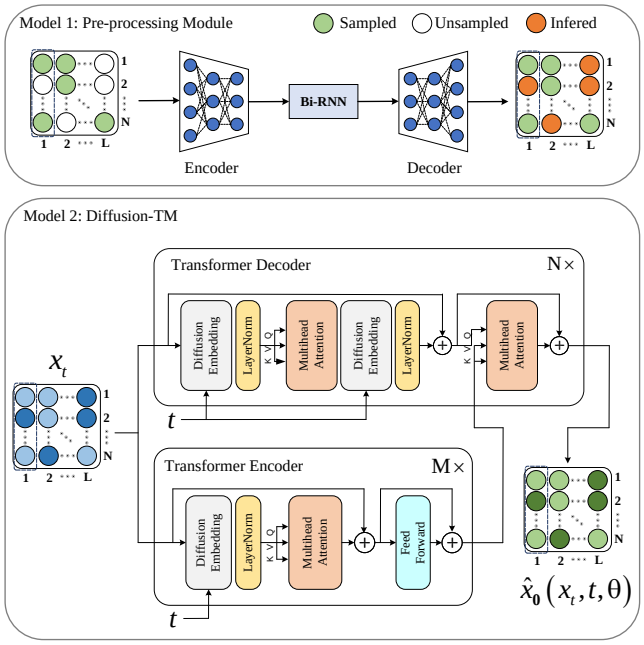


Fig. 5. **(Top) Architecture of Pre-processing Module.** The training of diffusion models starts with a AutoEncoder-based pre-processing module, which generates coarse-grained estimations of missing values in training set. **(Bottom) Overall model structure of Diffusion-TM.** The underlying Transformer is fed a TM sequence x_t in a diffusion step t, as well as t itself, then the diffusion model predicts the clean sample $\hat{x}_0$.

random Gaussian noise ϵ. But it should be noted that a direct reconstruction of $\boldsymbol{x_0}$ also takes missing values into account although they have been recovered by the pre-processing work to some extent. Thus again, we further adjust the training objective like Eqn. 32 as

$$\mathcal{L}_{VLB} = \|\boldsymbol{x_0} \odot \boldsymbol{M} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}(\boldsymbol{x_t}, t, \theta) \odot \boldsymbol{M}\|_2^2 . \quad (33)$$

The detailed training and sampling algorithm is shown in Alg. 3 and Alg. 4, respectively. Through applying DDIM algorithm to accelerate the sampling, we can use a different number of iterations $S \leq T$ during inference, making it possible to explicitly trade off between inference computation and output quality.

Algorithm 3 Training Algorithm of Diffusion-TM

Require: training epoch of pre-processing model (D) $\boldsymbol{I}'_{max}$, training epoch of Diffusion-TM $\boldsymbol{I}''_{max}$, training set with missing values $\{\boldsymbol{X}_k^o\}_{k=1}^K$, and the corresponding observation matrices $\{\boldsymbol{M}_k\}_{k=1}^K$
Ensure: trained denoising network θ

for all i from 1 to $\boldsymbol{I}'_{max}$ **do**
  $\boldsymbol{X}^o, \boldsymbol{M} \leftarrow$ **Get_Batch** $\left(\{\boldsymbol{X}_k^o\}_{k=1}^K, \{\boldsymbol{M}_k\}_{k=1}^K\right)$;
  Take gradient descent step on $\nabla_D \|\boldsymbol{X}^o \odot \boldsymbol{M} - D(\boldsymbol{X}^o) \odot \boldsymbol{M}\|_2^2$;
end for
$\{\boldsymbol{X}'_k\}_{k=1}^K \leftarrow D\left(\{\boldsymbol{X}_k^o\}_{k=1}^K\right)$;
$\{\boldsymbol{X}_k\}_{k=1}^K \leftarrow$ **Update** $\left(\{\boldsymbol{X}'_k\}_{k=1}^K, \{\boldsymbol{X}_k^o\}_{k=1}^K, \{\boldsymbol{M}_k\}_{k=1}^K\right)$;
for all i from 1 to $\boldsymbol{I}''_{max}$ **do**
  $\boldsymbol{X}, \boldsymbol{M} \leftarrow$ **Get_Batch** $\left(\{\boldsymbol{X}_k\}_{k=1}^K, \{\boldsymbol{M}_k\}_{k=1}^K\right)$;
  $t \leftarrow \text{Uniform}(\{1, \ldots, T\})$;
  $\epsilon \leftarrow \mathcal{K}(0, \text{I})$;
  $\boldsymbol{x_t} \leftarrow \sqrt{\bar{\alpha}_t}\boldsymbol{X} + \sqrt{1-\bar{\alpha}_t}\epsilon$
  Take gradient descent step on $\nabla_\theta \|\boldsymbol{X} \odot \boldsymbol{M} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}(\boldsymbol{x_t}, t, \theta) \odot \boldsymbol{M}\|_2^2$;
end for
return θ;

Algorithm 4 Fast Sampling Algorithm of Diffusion-TM

Require: trained denoising network θ, fast inference time stride $\triangle_t$
Ensure: synthetic traffic matrix $\boldsymbol{x_0}$

$\boldsymbol{x_T} \sim \mathcal{N}(0, \text{I})$;
while $t > 0$ **do**
  $\boldsymbol{z} \sim \mathcal{N}(0, \text{I})$ if $t > \triangle_t$, else $\boldsymbol{z} = 0$ and $\triangle_t = t$;
  $\boldsymbol{x_{t-\triangle_t}} = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-\triangle_t})}{1-\bar{\alpha}_t}\boldsymbol{x_t} + \frac{\sqrt{\bar{\alpha}_{t-\triangle_t}}\beta_t}{1-\bar{\alpha}_t}\hat{\boldsymbol{x}}_{\boldsymbol{0}}(\boldsymbol{x_t}, t, \theta) + \frac{1-\bar{\alpha}_{t-\triangle_t}}{1-\bar{\alpha}_t}\beta_t\boldsymbol{z}$;
  $t \leftarrow t - \triangle_t$;
end while
return $\boldsymbol{x_0}$;

VII. Experiments

In this section, we conduct experiments to evaluate the performances of the proposed Diffusion-TM. Three tasks are considered in the experiments: synthetic traffic data generation, network tomography, and traffic matrix completion. Our source code is available at https://github.com/Y-debug-sys/DTM.

A. Experimental Setup

1) Datasets: To evaluate the performance of our proposed method, we use two real-world traffic datasets.

- **Abilene** [55]. Abilene is derived from the U. S. Internet2 Network. 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. We use the first 3000 samples as training data and use 672 samples in the next week for testing.
- **GÉANT** [56]. GÉANT is derived from the pan-European research backbone network. 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 2003. We also train models with the first 3000 time slots, and then 672 samples are used to report the results of inference.

2) Baselines: We compare Diffusion-TM with two types of methods including 4 network tomography algorithms and 5 traffic matrix completion algorithms.

- **Network Tomography algorithms**. We implement 4 algorithms with deep learning technology, among which, two methods VAE-TME [19] and WGAN-TME [39] are based on generative models, the other two algorithms BPTME [17], MNETME [18] used a neural network to learn the inverse mapping directly.
- **TM completion algorithms**. The first four algorithms

(NTC [32], NTM [57], NTF [58], CoSTCo [59]) are based on neural tensor factorization. NTF and CoSTCo adopt multi-layered perceptron and convolution neural networks as interaction functions, respectively. NTC and NTM design interaction functions based on outer-product. The last one is DATC [13], a recent work that exploited autoencoder and GANs to complete the traffic data.

Almost all learning-based NT algorithms assume that the training data has zero missing data. Thus, we infill these missing values before running the baseline. Following the work of [31], we construct the interpolation matrix X_{base} by computing row and column means of the observed traffic samples. Let $X(i,j)$ denote the value of the i-th OD flow pair at the j-th time point. Then formally, the pre-processing result is given by

$$X_{base}(i,j) = \bar{X} + X_{flow}(i) + X_{time}(j), \tag{34}$$

where $\bar{X}$ is the mean value of traces X over all observed elements, and

$$\begin{aligned} X_{flow}(i) &= \frac{1}{m}\sum_{j=1}^{m}\left(X(i,j) - \bar{X}\right), \\ X_{time}(j) &= \frac{1}{n}\sum_{i=1}^{n}\left(X(i,j) - \bar{X}\right) \end{aligned} \tag{35}$$

For these baselines, we reuse their released source codes in their official repositories[1] and rely on their designed training and model selection procedures in the original paper. Regarding algorithms that do not provide any code, we follow their supplementary description of the implementation to the best of our ability.

3) Implementation Details: We apply a grid search to find default hyper-parameters in the underlying transformer that perform well across datasets. The range considered for each hyper-parameter is the batch size tuned in [32, 64, 128], the number of attention heads in [4, 8, 16], the number of basic dimension searched in [64, 96, 128], the diffusion steps in [50, 100, 300, 500, 1000] and the guidance strength in [1e-0, 1e-1, 5e-2, 1e-2, 1e-3]. The activation function of the output layer we consider is *Sigmoid*. A single Nvidia 3090 GPU is used for model training. In all of our experiments, we use $cosine$ noise scheduling [60] and optimize our network using Adam with $(\beta_1, \beta_2) = (0.9, 0.96)$. And a linearly decay learning rate starts at 0.0008 after 500 iterations of warmup. For a fair comparison, all the deep learning based algorithms use $\mathrm{L1Norm}$ loss function. We set the sliding window size $w = 12$ for Abilene and GÉANT in all experiments.

Finally, we replicate the experimental setup in [61], in which they clipped outliers larger than the 99% percentile to the 99% percentile, then normalize all training samples were normalized by dividing the maximum values.

B. Evaluation Metrics

We implement two methods to evaluate the learnt **distribution**: 2-dimensional visualization via t-SNE analysis [62] and Maximum Mean Discrepancy (MMD) [63], a kernel-based method for computing the difference of the statistics of the two sets of samples. Given samples $X := \{x_1, \ldots, x_n\}$ and $Y := \{y_1, \ldots, y_m\}$ drawn independently and identically distributed (i.i.d.) from two distributions

$$\begin{aligned} &\mathrm{MMD}_k^2(X,Y) \\ &:= \frac{1}{n(n-1)}\sum_{i,j=1}^{n} k(x_i, x_j) - \frac{2}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m} k(x_i, y_j) \\ &\quad + \frac{1}{m(m-1)}\sum_{i,j=1}^{m} k(y_i, y_j) \end{aligned} \tag{36}$$

Our experiments use the universal Gaussian kernel, defined as $k(x_i, x_j) = \exp\left(\frac{1}{2\sigma^2}|x_i - x_j|^2\right)$, where σ is the bandwidth parameter. For characteristic kernel functions [64], it can be proven that $\mathrm{MMD}_k(p,q)$ if and only if $p = q$, leading to consistent results.

In addition, we qualify the estimation **accuracy** using three mainstreaming metrics: Normalized Mean Absolute Error (NMAE), Normalized Root Mean Square Error (NRMSE), and Temporal Related Mean Absolute Error (TRE). Specifically, we calculate

$$\begin{aligned} \mathrm{NMAE} &= \frac{\sum_{i,j:M(i,j)=0}\left|X(i,j) - \hat{X}(i,j)\right|}{\sum_{i,j:M(i,j)=0}|X(i,j)|}, \\ \mathrm{NRMSE} &= \frac{\sqrt{\sum_{i,j:M(i,j)=0}\left(X(i,j) - \hat{X}(i,j)\right)^2}}{\sqrt{\sum_{i,j:M(i,j)=0}\left(X(i,j)\right)^2}}, \\ \mathrm{TRE}(j) &= \frac{\sum_i\left|X(i,j) - \hat{X}(i,j)\right|}{\sum_i X(i,j)} \end{aligned} \tag{37}$$

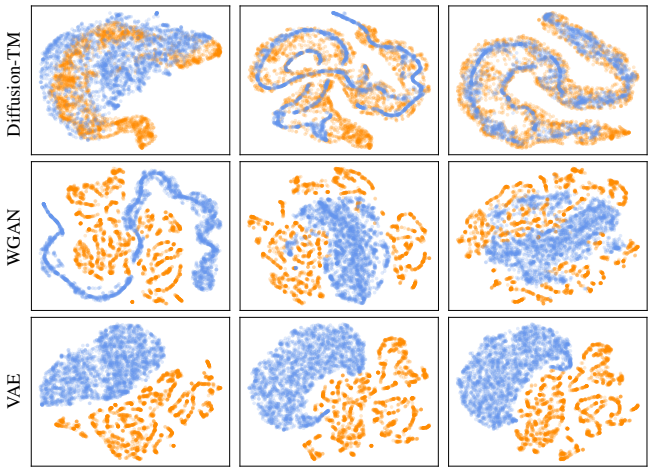
where $\hat{X}$ is the estimated traffic matrix. For traffic matrix completion, we first drop some data from existing measurements and then only measure errors on the pseudo-missing values.

C. Learnt Distribution Visualization

We first report the quality of synthetic traffic data produced by the generative models with respect to the distributions over the original and generated data. We flatten the temporal dimension and use t-SNE plots to compress them into 2-dimensional space for visualization. Fig. 6 and Fig. 7 present results with different sampling rates (i.e., the proportion of training data from the known entries of different datasets), where a greater overlap of blue (fake) and orange (real) dots shows a better distributional-similarity between the generated TMs and original TMs. There are several key observations: (i) we observe that Diffusion-TM consistently matches the realistic distribution better than other benchmarks. Despite the powerful generative ability of diffusion models, all generative models except ours synthesize inferior data (covering a much smaller area across the original data) on real-world datasets, which indicates that previous GAN (or VAE) methods may not be able to model high-dimensional and complex network traffic distribution well. (ii) One can also note that the performance of Diffusion-TM does not degrade significantly with the percentage of missing values, even in the extreme

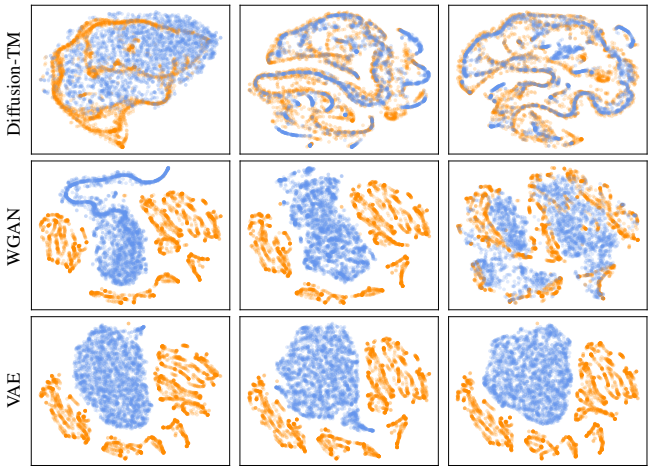
[1]The codes of experiments with the tensor factorization methods are available at https://github.com/MerrillLi/LightNestle. And VAE-TME is available at https://github.com/MikeKalnt/VAE-TME.

case of only 2% observed data. The significant improvement of Diffusion-TM demonstrates our two-stage training algorithm can greatly ease the effect of missing values, making the learned distribution much more accurate. Overall, our proposed diffusion framework shows the best data diversity and robust quality, with the closest match to the original traffic data. Thus, the Diffusion-TM provides a promising approach to generating large-scale network measurement data in practical applications.



(a) 2% (b) 50% (c) 100%

Fig. 6. t-SNE plots for Diffusion-TM (1st row), WGAN (2nd row), VAE (3rd row) in Abilene dataset with different sampling rates (2%, 50%, 100%) of training traffic traces.



(a) 2% (b) 50% (c) 100%

Fig. 7. t-SNE plots for Diffusion-TM (1st row), WGAN (2nd row), VAE (3rd row) in GÉANT dataset with different sampling rates (2%, 50%, 100%) of training traffic traces.

D. Traffic Matrix Completion Performance Comparison

In Table[2]. I, we present the experimental results of our Diffusion-TM and five neural network tensor completion algorithms (DATC, NTC, NTM, NTF, CoSTCo) on *training* dataset, when varying the amount of known information range from 2 to 50 percent of the OD-flow entries. Except for NRMSE under 2% sampling rate in GÉANT, we can see that the proposed Diffusion-TM yields the best performance on all the table items. Specifically, Diffusion-TM can reduce the average NMAE of the best baseline by 15% in Abilene and 23% in GÉANT. Even when the sampling rate is less than 5% which is a very low ratio, Diffusion-TM is still effective thanks to the combination of pre-processing module and designed reconstruction loss on observed flow loads, which forces the diffusion model to optimize the distribution for unobserved flows during the training stage. Although four neural network (NN) based tensor completion algorithms exhibit some ability to handle a considerable amount of missing data, the overall imputation accuracy is still limited compared with our approach. Among them, the effect of NTC is generally better than other methods, because the algorithm designed for monitoring performs well to extract features in the network traffic while NTF, CoSTCo, and NTM are more suitable for recommender systems. On the contrary, the autoencoder-based method DATC performs marginally worse than ours when the sampling rate is low. However, the accuracy gap between Diffusion-TM and DATC widens for a small missing ratio (e.g., $90\% \sim 50\%$), since the larger number of observed entities allows DMs to fit the distribution of training traffic better, so Diffusion-TM can achieve a completion error of 0.16 which significantly outperforms the best DATC with different datasets when the sampling ratio is 50%.

In addition, our Diffusion-TM achieves a much more excellent MMD score. In contrast, the peer NN-based algorithms perform poorly for all loss probabilities, as their performance largely depends on the position of the missing entries in the tensor (matrix) while unable to capture the data distribution. DATC may perform better than these low-rank methods owing to its adversarial nature in this case, but nevertheless, Diffusion-TM still shows the best performance over the widest range of missing ratios. The typically very large performance gap also suggests that our gradient-oriented completion algorithm can solve the recovery problem while preserving the learned distribution of powerful diffusion models to the greatest extent.

Complex and dynamic network behavior can not guarantee that training data would not undergo distribution shift in the real world. Retraining on new traffic data may be a solution, but it cannot meet the needs of real-time filling. Thus we then perform additional infilling experiments on *testing* dataset to validate the performance of the online TMC with an already trained model. Since NTC, NTF, NTM, and CoSTCo use the whole tensor data as input to complete it together, and have no concept of training set (or inference set), they can not run online to complete continuously arrived traffic matrix sequence (sliding window). We do not compare all baselines and eliminate them from the experiment. Moreover, in actual online executions, it is normal to observe some easily obtained link-load data during inference. As aforementioned, our algorithm can be easily extended to solve multi-objective problems. Thus we will also experience and show how the Diffusion-TM performs by combining the additional useful constraints. As shown in Table. II, we report results for newly collected data imputation on the test set, where Diffusion-TM (p%)

[2]In the table, results (except for MMD) of all baselines are from [13].

TABLE I
COMPLETION PERFORMANCE ON TRAINING SET OF ABILENE AND GÉANT.

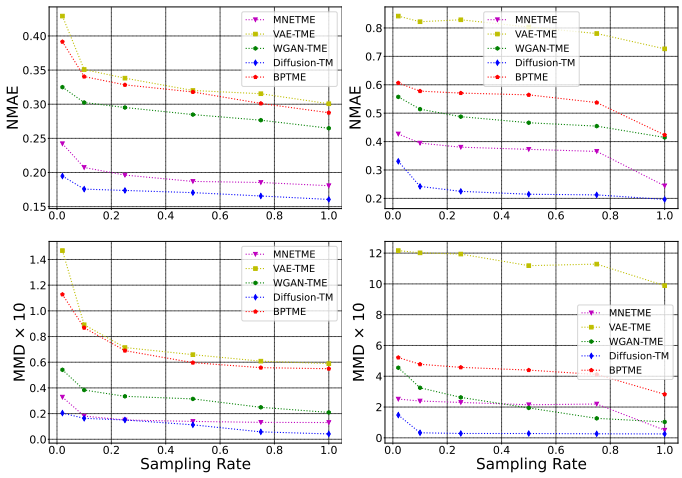
Dataset		Abilene						GÉANT					
Metric	Model	2%	4%	6%	8%	10%	50%	2%	4%	6%	8%	10%	50%
NMAE	Diffusion-TM	**0.2712**	**0.2446**	**0.2329**	**0.2261**	**0.2111**	**0.1623**	**0.4703**	**0.3152**	**0.2514**	**0.2360**	**0.2185**	**0.1627**
	DATC	0.2954	0.2748	0.2631	0.2596	0.2541	0.2381	0.4939	0.4332	0.3651	0.3215	0.3037	0.2487
	NTC	0.3702	0.3362	0.3243	0.3115	0.3073	0.2497	0.7042	0.6796	0.6239	0.5781	0.4751	0.2871
	NTM	1.2214	0.8818	0.7633	0.6686	0.6178	0.3758	1.4547	1.3646	1.3266	1.2003	1.1987	0.5150
	NTF	0.7529	0.4147	0.3637	0.3486	0.3353	0.2639	0.9061	0.7783	0.7189	0.5958	0.5125	0.3208
	CoSTCo	0.3857	0.3739	0.3381	0.3224	0.3145	0.2971	0.8588	0.7826	0.7291	0.7241	0.7042	0.6851
NRMSE	Diffusion-TM	**0.3115**	**0.2901**	**0.2777**	**0.2611**	**0.2467**	**0.2034**	0.5501	**0.3383**	**0.2795**	**0.2457**	**0.2250**	**0.1643**
	DATC	0.3453	0.3312	0.3226	0.3202	0.3178	0.3076	**0.3818**	0.3415	0.2979	0.2547	0.2368	0.1742
	NTC	0.4388	0.4072	0.3951	0.3858	0.3766	0.3436	0.6412	0.6266	0.5888	0.5450	0.4310	0.2319
	NTM	0.9720	0.7573	0.7234	0.6613	0.6197	0.4473	0.9230	0.9057	0.8877	0.7854	0.7629	0.3976
	NTF	0.6856	0.4201	0.3882	0.3848	0.3723	0.3667	0.7110	0.6621	0.5838	0.4959	0.4128	0.2728
	CoSTCo	0.4081	0.3929	0.3736	0.3666	0.3600	0.3579	0.7215	0.6579	0.6252	0.6037	0.5927	0.5827
MMD	Diffusion-TM	**0.0183**	**0.0156**	**0.0149**	**0.0088**	**0.0059**	**0.0019**	**0.1168**	**0.0335**	**0.0075**	**0.0059**	**0.0058**	**0.0007**
	DATC	0.0446	0.0359	0.0275	0.0250	0.0122	0.0049	0.1978	0.0993	0.0460	0.0139	0.0104	0.0047
	NTC	0.1565	0.0426	0.0297	0.0184	0.0120	0.0055	0.5598	0.2010	0.1458	0.1265	0.1259	0.0074
	NTM	0.5883	0.5249	0.4887	0.3806	0.3256	0.0983	0.8679	0.6743	0.5619	0.4594	0.3951	0.1271
	NTF	0.2959	0.1899	0.1571	0.1089	0.0512	0.0151	0.7584	0.5704	0.4016	0.2554	0.1854	0.0212
	CoSTCo	0.1607	0.0513	0.0329	0.0246	0.0155	0.0098	0.6089	0.3713	0.3368	0.2289	0.1677	0.0184

TABLE II
COMPLETION PERFORMANCE ON TESTING SET OF ABILENE AND GÉANT.

Dataset		Abilene						GÉANT					
Metric	Model	2%	4%	6%	8%	10%	50%	2%	4%	6%	8%	10%	50%
NMAE	Diffusion-TM	0.3041	0.2854	0.2804	0.2701	0.2395	0.2162	0.4834	0.4142	0.3714	0.3470	0.2714	0.2368
	Diffusion-TM (10%)	0.2835	0.2604	0.2577	0.2542	0.2313	0.2066	0.4734	0.3269	0.3181	0.3025	0.2630	0.2350
	Diffusion-TM (50%)	0.2366	0.2221	0.2193	0.2170	0.1962	0.1760	0.4038	0.3230	0.3064	0.2971	0.2576	0.2186
	Diffusion-TM (100%)	**0.1981**	**0.1871**	**0.1853**	**0.1802**	**0.1715**	**0.1577**	**0.3492**	**0.2831**	**0.2820**	**0.2660**	**0.2155**	**0.1829**
	DATC	0.4052	0.3559	0.3304	0.3218	0.2980	0.2746	0.5562	0.4456	0.4030	0.3918	0.3815	0.2889
NRMSE	Diffusion-TM	0.3463	0.3192	0.3123	0.3048	0.2792	0.2501	0.5401	0.4686	0.4239	0.3893	0.2923	0.2416
	Diffusion-TM (10%)	0.3167	0.3017	0.3055	0.2941	0.2708	0.2422	0.5418	0.3476	0.3393	0.3158	0.2631	0.2368
	Diffusion-TM (50%)	0.2708	0.2610	0.2602	0.2597	0.2269	0.2022	0.4275	0.3263	0.3182	0.3056	0.2411	0.2083
	Diffusion-TM (100%)	**0.2011**	**0.2003**	**0.1952**	**0.1941**	**0.1862**	**0.1660**	**0.3452**	**0.2732**	**0.2663**	**0.2528**	**0.2027**	**0.1782**
	DATC	0.3972	0.3882	0.3614	0.3536	0.3314	0.2958	0.5050	0.4686	0.4574	0.4486	0.4507	0.3258
MMD	Diffusion-TM	0.0684	0.0618	0.0569	0.0519	0.0350	0.0081	0.3348	0.2218	0.2047	0.1541	0.0409	0.0087
	Diffusion-TM (10%)	0.0706	0.0610	0.0576	0.0528	0.0276	0.0067	0.2448	0.1306	0.1190	0.1014	0.0320	0.0081
	Diffusion-TM (50%)	0.0489	0.0431	0.0414	0.0406	0.0157	0.0035	0.2200	0.1346	0.1277	0.0840	0.0314	0.0066
	Diffusion-TM (100%)	**0.0198**	**0.0183**	**0.0157**	**0.0131**	**0.0127**	**0.0027**	**0.1442**	**0.0808**	**0.0792**	**0.0537**	**0.0216**	**0.0047**
	DATC	0.1249	0.1087	0.0773	0.0739	0.0539	0.0350	0.5570	0.2852	0.2598	0.2254	0.1881	0.0576

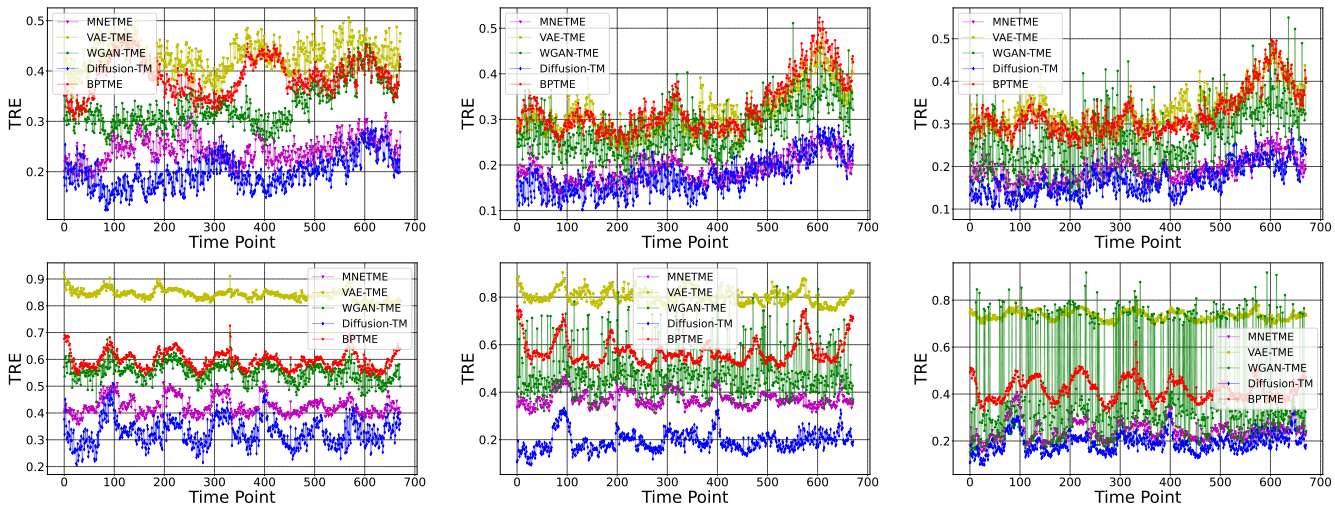
signifies Diffusion-TM with $p\%$ link loads measured. Due to the distribution drift and probable overfitting, both DATC and Diffusion-TM show inferior accuracy. The MMD score further supports the problem of distribution alignment. Nevertheless, Diffusion-TM can still improve the accuracy of DATC by $16\% \sim 25\%$ ($15\% \sim 29\%$) in Abilene (GÉANT). Moreover, if any link measurements of the network are provided, then Diffusion-TM's performance improves dramatically. Finally, by combining 100 percent link loads and arbitrary known TM elements, Diffusion-TM gets the best of both networks and even outperforms itself on the training set in some cases.

These results demonstrate the superiority of our model in recovering missing data. To summarize, our Diffusion-TM has three key advantages: (i) it does not require a complete training dataset which is generally expensive; (ii) it can precisely produce unobserved flows that closely fit the prior distribution; and (iii) it can also impose extra constraints in a plug-and-play way, preventing useful information waste.



(a) Abilene (b) GÉANT

Fig. 8. Network tomography performance for complete TM estimation under different sampling rates.



(a) Sampling Rate $= 2\%$ (b) Sampling Rate $= 50\%$ (c) Sampling Rate $= 100\%$

Fig. 9. Temporal relative errors (TREs) on three sampling rates in Abilene (top) and GÉANT (bottom) dataset.

E. Network Tomography Performance Comparison

In this section, we consider the performance of Diffusion-TM with respect to the network tomography problem of inferring a TM from link-load measurements. Fig. 8 presents the NMAE and MMD of all NT methods under different sampling rates. Note that we assume here we can measure all of the link loads on the networks. From the figures, the errors of all methods grow with the increasing of the unknown rate, indicating the missing values in the training set do affect the reliability of the estimations. However, we can see that Diffusion-TM consistently better approaches the ground truth and tracks the distribution compared with other methods. VAE-TME performs the worst as VAE cannot capture the statistics of the real-world traffic data, especially in a larger network such as GÉANT. It is uniformly shown that BPTME is better than VAE-TME, meanwhile, MNETME using the routing matrix's Moore–Penrose inverse and EM algorithm with a BPTME architecture provides a definite improvement that is closest to our Diffusion-TM. But note also that both methods belong to supervised learning that requires a fixed routing matrix for learning the inverse mapping directly, which means that the topology and routing of the network must be unchanged during training and testing, otherwise, they need to be retrained. So regardless of implicit distribution learning without theoretical support, the flexibility of these methods is greatly restricted. The last algorithm is WGAN-TME, which is similar to VAE-TME but uses the generator network of GAN to search for the optimal solution. It is apparent that the competition between the generator and discriminator results in the generator learning to produce a wide variety of more plausible outputs compared to VAEs. However, apart from the problem of training instability, the ability of GAN to learn the high-dimensional traffic distribution is still worse than that of Diffusion-TM, leading to significantly greater errors across all loss models. One can also observe that Diffusion-TM exhibits very stable estimation performance with $10\% \sim 100\%$ observed TM elements in the training set. In other words, if as few as 10% of the TM elements are used to train our model, then Diffusion-TM achieves a performance similar to that requiring a sampling ratio of 90% or more. The phenomenon again demonstrates the robustness of our proposed diffusion-based framework.

Fig. 9 plots the TREs of all baselines under three sampling rates. In both two datasets, the curves of all methods are gradually showing periodicity as the measurement time goes beyond. Overall, our method improves the TRE of the best baseline by 10% in Abilene dataset, and 27% in GÉANT, averaging over all unknown ratios. Interestingly, the frequency and amplitude of WGAN-TME's curve suffer from a rapid change when the sampling rate is high, especially in high-dimensional GÉANT. That is because, on the one hand, it does not take into account temporal characteristics (but it may also lead to more unstable training). On the other hand, the synthetic data of WGANs fail to capture the diversity of the complex traffic data, which means any traffic estimated on that basis would also fail to diversify with increasing of the observed elements. That indicates our method can achieve high accuracy for all flows without concerns on specific monitoring for small traffic measurements.

F. Computational Times

In this section, we measure the computation times of Diffusion-TM and its competitors on both traffic matrix completion and network tomography tasks. For a fair comparison of these tasks, we collected 3000 samples (10% entities observed) as a training set, and then tested the inference times on them. As shown in Table. III, we list all detailed computation times. Here the total diffusion step is set as 300, thus there is still space to reduce the running time of Diffusion-TM. Regarding NT solutions, it can be seen that Diffusion-TM outperforms other generative-model-based TME algorithms (WGAN-TME, VAE-TME) in terms of sampling time because they leverage extra thousands of iterations to adjust each

TABLE III
AVERAGE TRAINING AND INFERENCE TIME

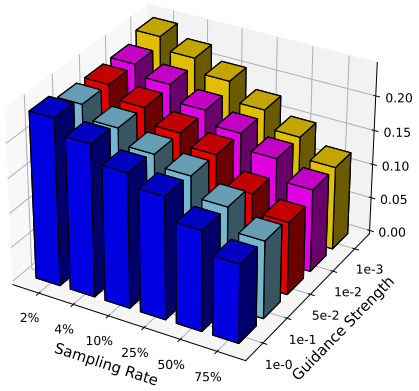
Solving Network Tomography Problem				
Methods	Abilene		GÉANT	
	Training Time (s)	Inference Time (s)	Training Time (s)	Inference Time (s)
MNETME	84.22	5.50	105.01	5.61
BPTME	72.98	0.03	93.47	0.04
WGAN-TME	1331.65	120.54	1614.18	123.72
VAE-TME	109.70	156.25	130.48	162.30
Diffusion-TM	542.08	11.45	550.64	12.29
Solving Traffic Matrix Completion Problem				
Methods	Abilene		GÉANT	
	Training Time (s)	Inference Time (s)	Training Time (s)	Inference Time (s)
NTC	\	339.03	\	6290.32
NTM	\	101.86	\	1683.34
NTF	\	8903.63	\	17045.31
CoSTCo	\	41.90	\	202.56
DATC	1602.92	1.33	2914.15	1.35
Diffusion-TM	542.08	10.26	550.64	10.73

estimation to be consistent with both the learned distribution and the NT constraints. The full-supervised TME methods (MNETME, BPTME) have the fastest computation times, although they also require intact label and routing information during training. For TMC times, tensor factorization methods usually take the longest recovering time as expected, at the cost of no off-line training requirements. DATC achieves the shortest inference time among its competitors, but similar to WGAN-TME, the algorithm suffers from longer training time than Diffusion-TM due to its adversarial nature. Overall, our Diffusion-TM is capable of providing more accurate estimates within a reasonable time, indicating our approach can be applied in practical scenarios.

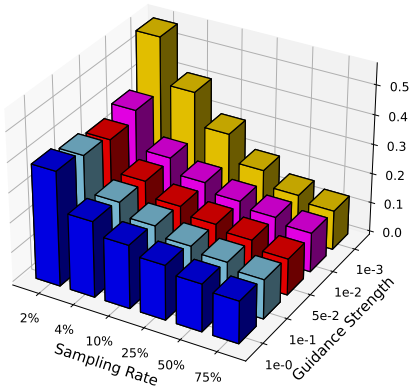
G. Ablation and Sensitivity Analysis

The section mainly focuses on evaluating the impact of the crucial choices. In what follows, we will first test the following diffusion-related hyperparameters: a) trade-off scale parameter; b) diffusion and sampling steps. Then we conduct an ablation study including different algorithm components. Where not otherwise stated, a fixed scenario with 25% link loads observed in the target set will be implemented thorough our analysis.

1) Impact of scaling coefficients: Fig. 10 draws the recovery performance of our Diffusion-TM with different fixed scaling coefficient ρ, which is an important hyper-parameter that directly affects the consistency of sampling results. From the figure, we can find that the parameter is not very sensitive in Abilene. However, the choice of ρ tends to have a significant influence on TME accuracy in GÉANT, especially when a smaller portion of the data is collected. Also note that there are a number of cases where the optimal ρ is around 0.05. Therefore, we use this value in our experiments.



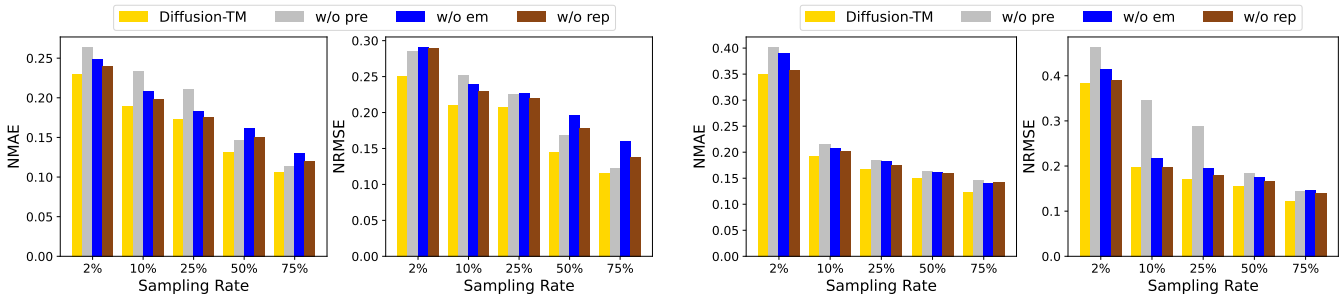
(a) Abilene



(b) GÉANT

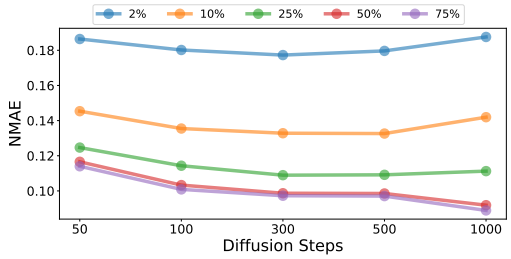
Fig. 10. The performance (NMAE) of Diffusion-TM with different guidance strength ρ under different known rates.

2) Ablational Study: We start by analyzing the effect of each part in our approach. Here, three variants of the Diffusion-TM were investigated: (i) **w/o pre**. We replace the pre-processing module using Eqn. 34 to train the diffusion model; (ii) **w/o em**. We remove the EM algorithm after each sampling through solving NT equations; (iii) **w/o rep**. We cancel additional correction steps to further clarify the efficacy of replace-based guidance. Their ablation results are shown in Fig. 11. First, we see our model works across different known ratios and generates better quality for real-world network TM estimation in general, indicating the effectiveness of the combination of its components. It can also be seen our missing-data-aware strategy does boost the robust performance of Diffusion-TM. When there is a large number of network nodes (i.e. GÉANT), the dimension dependency is more prosperous, and the pre-processing module performs much better, especially under a sampling rate $< 50\%$. Moreover, we notice a clear positive impact of the expectation maximization, even with only 25% of link measurements known. Diffusion-TM outperforms the counterpart without replace-based correction, which verifies the additional guidance is beneficial for recovering real-world traces.

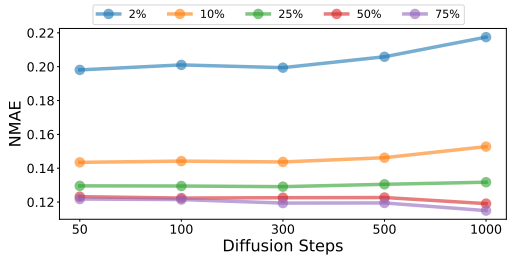


(a) Abilene (b) GÉANT

Fig. 11. Results of Diffusion-TM and its three variants.



(a) Abilene



(b) GÉANT

Fig. 12. Diffusion-TM for different diffusion steps.

3) Impact of diffusion steps: Here we evaluate the impact of a different number of sampling steps in the diffusion models. To do that, we first train a network with forward steps randomly selected in $\{1, \cdots, 1000\}$. Then when it comes to the inference period, we vary the diffusion steps in [50, 100, 300, 500, 1000] using DDIM shown in Eqn. 16. Fig. 12 reports the quantitative results versus number of diffusion steps. Overall, The recovery performance of different diffusion steps is close and as the known ratio increases, the diffusion model with more steps brings a better result. Also in this figure, one can observe that increasing the diffusion steps does not improve or even exacerbate the results when the sampling rate is low, due to potential overfitting. Therefore, to adapt to fast estimation in practice, in all our experiments we considered the most economical but effective setup of step number = 300.

VIII. Conclusion

We presented in this paper Diffusion-TM, a novel diffusion framework to traffic matrix (TM) analysis in computer networks. Diffusion-TM bridges the gap between denoising diffusion models and traditional TM-related problems. By refining the generative process with available measurements and sampling from the space of plausible TMs, our diffusion-based approach achieves outstanding performance over state-of-the-art models on various tasks while avoiding expensive problem-specific training. We prove the feasibility of our method theoretically, then optimize the sampling procedure using the EM algorithm and replace-based guidance. Additionally, we proposed a two-stage training scheme to adapt Diffusion-TM to practical scenarios with a large number of missing values in the training set. Finally, we conducted extensive experiments on two real-world traffic datasets. Our results demonstrate the superiority of our Diffusion-TM on producing qualified TMs in different scenarios with incomplete training instances, offering an attractive alternative to the mainstream TM analysis methods.

Unlike existing TM analysis solutions, Diffusion-TM is versatile in that it can be flexibly implemented for multiple tasks (TM synthesis, tomography, and completion) at the same time by only collecting the traffic data of a subset of OD-flows within a short period for offline training. We thus believe that the method provides profound insights into a broad range of traffic matrix related applications. However, it is also known that the key limitation of our method is the high computational cost of the iterative denoising process. So exploring faster solvers that provide a trade-off between performance and computational overhead leaves considerable work to do in the future.

Appendix

Proof for Lemma 1: We first compute the derivative of the marginal distribution $p(\boldsymbol{x})$ with respect to $\boldsymbol{x}$ which could be expressed as

$$\begin{aligned}
\nabla_{\boldsymbol{x}} p(\boldsymbol{x}) &= \nabla_{\boldsymbol{x}} \int p(\boldsymbol{x}|\boldsymbol{\eta})\, g(\boldsymbol{\eta})\, d\boldsymbol{\eta} \\
&= \nabla_{\boldsymbol{x}} \int p_0(\boldsymbol{x}) \exp\left(\boldsymbol{\eta}^T F(\boldsymbol{x}) - \psi(\boldsymbol{\eta})\right) g(\boldsymbol{\eta})\, d\boldsymbol{\eta} \\
&= \left(\nabla_{\boldsymbol{x}} F(\boldsymbol{x})\right)^T \int \boldsymbol{\eta} p_0(\boldsymbol{x}) \exp\left(\boldsymbol{\eta}^T F(\boldsymbol{x}) - \psi(\boldsymbol{\eta})\right) g(\boldsymbol{\eta})\, d\boldsymbol{\eta} \\
&\quad + \nabla_{\boldsymbol{x}} p_0(\boldsymbol{x}) \int \exp\left(\boldsymbol{\eta}^T F(\boldsymbol{x}) - \psi(\boldsymbol{\eta})\right) g(\boldsymbol{\eta})\, d\boldsymbol{\eta} \\
&= \left(\nabla_x F(\boldsymbol{x})\right)^T \int \boldsymbol{\eta} p(\boldsymbol{x}, \boldsymbol{\eta}) d\boldsymbol{\eta} + \frac{\nabla_{\boldsymbol{x}} p_0(\boldsymbol{x})}{p_0(\boldsymbol{x})} \int p(\boldsymbol{x}|\boldsymbol{\eta})\, g(\boldsymbol{\eta})\, d\boldsymbol{\eta} \\
&= \left(\nabla_x F(\boldsymbol{x})\right)^T \int \boldsymbol{\eta} p(\boldsymbol{x}, \boldsymbol{\eta}) d\boldsymbol{\eta} + \frac{\nabla_{\boldsymbol{x}} p_0(\boldsymbol{x})}{p_0(\boldsymbol{x})} p(\boldsymbol{x}).
\end{aligned} \tag{38}$$

As a consequence,

$$\left(\nabla_{\boldsymbol{x}} F\left(\boldsymbol{x}\right)\right)^{T} \int \boldsymbol{\eta} p\left(\boldsymbol{\eta}|\boldsymbol{x}\right) d\boldsymbol{\eta} = \frac{\nabla_{\boldsymbol{x}} p\left(\boldsymbol{x}\right)}{p\left(\boldsymbol{x}\right)} - \frac{\nabla_{\boldsymbol{x}} p_0\left(\boldsymbol{x}\right)}{p_0\left(\boldsymbol{x}\right)}. \quad (39)$$

Then, we have

$$\left(\nabla_{\boldsymbol{x}} F\left(\boldsymbol{x}\right)\right)^{T} \hat{\boldsymbol{\eta}} = \nabla_{\boldsymbol{x}} \log p\left(\boldsymbol{x}\right) - \nabla_{\boldsymbol{x}} \log p_0\left(\boldsymbol{x}\right), \quad (40)$$

which concludes the proof. □

Proof for Proposition 1: If we consider a diffusion model in which the forward step can be modeled as Eqn. 11, the corresponding formula is then given by $\boldsymbol{x}|\boldsymbol{\eta} \sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\boldsymbol{\eta}, \left(1-\bar{\alpha}_t\right)\mathbf{I}\right)$. Therefore, using Eqn. 23, we have

$$\mathbb{E}\left(\boldsymbol{x_0}|\boldsymbol{x_t}\right) = \frac{1}{\sqrt{\bar{\alpha}_t}}\boldsymbol{x_t} + \frac{1-\bar{\alpha}_t}{\sqrt{\bar{\alpha}_t}}\nabla_{\boldsymbol{x_t}} \log p_t\left(\boldsymbol{x_t}\right). \quad (41)$$

This concludes the proof. □

Lemma 2: *Let vector function $\mathcal{Q}$ is the orthogonal projection onto a subspace $\mathcal{M} \subset \mathbb{R}^n$. Then $\boldsymbol{J}_{\mathcal{Q}}$, the Jacobian matrix of $\mathcal{Q}$, is symmetric, i.e., $\boldsymbol{J}_{\mathcal{Q}} = \boldsymbol{J}_{\mathcal{Q}}^T$.*

Proof. We start by defining $T_s\left(\mathcal{M}, \mathcal{Q}\left(\boldsymbol{x}\right)\right)$ as the tangent space at a point $\mathcal{Q}\left(\boldsymbol{x}\right)$ on $\mathcal{M}$. Let $\boldsymbol{u_1} \in T_s\left(\mathcal{M}, \mathcal{Q}\left(\boldsymbol{x}\right)\right)$, $\boldsymbol{u_2} \perp T_s\left(\mathcal{M}, \mathcal{Q}\left(\boldsymbol{x}\right)\right)$, and $\boldsymbol{u} = \boldsymbol{u_1} + \boldsymbol{u_2}$. Then given a constant k,

$$\mathcal{Q}\left(\boldsymbol{x} + k\boldsymbol{u}\right) = \mathcal{Q}\left(\boldsymbol{x}\right) + k\boldsymbol{u_1}, \quad (42)$$

which comes from the only tangent vector $\boldsymbol{u_1}$ influence the orthogonal projection onto $\mathcal{M}$. And by differentiating the equation with respect to k, we can write $\boldsymbol{J}_{\mathcal{Q}}\boldsymbol{u} = \boldsymbol{u_1}$. Now considering another vector $\boldsymbol{v} = \boldsymbol{v_1} + \boldsymbol{v_2}$ with the same settings: $\boldsymbol{v_1} \in T_s\left(\mathcal{M}, \mathcal{Q}\left(\boldsymbol{x}\right)\right)$, $\boldsymbol{v_2} \perp T_s\left(\mathcal{M}, \mathcal{Q}\left(\boldsymbol{x}\right)\right)$. We have

$$\begin{aligned} \boldsymbol{u}^T \boldsymbol{J}_{\mathcal{Q}} \boldsymbol{v} &= \left(\boldsymbol{u_1} + \boldsymbol{u_2}\right)^T \boldsymbol{v_1} = \boldsymbol{u_1}^T \boldsymbol{v_1} \\ &= \left(\boldsymbol{J}_{\mathcal{Q}}\boldsymbol{u}\right)^T \boldsymbol{v_1} = \boldsymbol{u}^T \boldsymbol{J}_{\mathcal{Q}}^T \boldsymbol{v}. \end{aligned} \quad (43)$$

Thus, $\boldsymbol{J}_{\mathcal{Q}} = \boldsymbol{J}_{\mathcal{Q}}^T$ which concludes the proof. □

Proof for Theorem 1: First by combining with the condition that $\mathcal{M}$ has linear structure, we have $\mathcal{D}_t\left(\boldsymbol{x_t}\right) \in \mathcal{M}$ as $\mathcal{D}_t\left(\boldsymbol{x_t}\right) = \mathbb{E}\left(\boldsymbol{x_0}|\boldsymbol{x_t}\right) = \int \boldsymbol{x_0} p\left(\boldsymbol{x_0}|\boldsymbol{x_t}\right) d\boldsymbol{x_0}$ is the weighted average of points on the traffic data manifold.

Note that $p\left(\boldsymbol{x_0}|\boldsymbol{x_t}\right)$ is not only a Gaussian, but also a radial function $r\left(\boldsymbol{x_0}\right) = \hat{r}\left(\|\boldsymbol{x_0} - c\|\right)$ with center $c = \mathbb{E}\left(\boldsymbol{x_0}|\boldsymbol{x_t}\right)$. Thus intuitively, $\mathcal{D}_t\left(\boldsymbol{x_t}\right)$ should be the nearest point on $\mathcal{M}$ to $\boldsymbol{x_t}$ on the noisy data manifold $\mathcal{M}_t$. Then since the distance is usually the closest, we can say that $\mathcal{D}_t$ is locally an orthogonal projection onto $\mathcal{M}$, and $\boldsymbol{J}_{\mathcal{D}_t}$ is the orthogonal projection onto $T_s\left(\mathcal{M}, \mathcal{D}_t\left(\boldsymbol{x_t}\right)\right)$. Considering the constrained gradient term, we have

$$\begin{aligned} &\rho_t \nabla_{\boldsymbol{x_t}} \left\|\boldsymbol{y} - \mathcal{A}\left(\hat{\boldsymbol{x}}_{\boldsymbol{0}}\right)\right\|_2^2 = -2\rho_t \boldsymbol{J}_{\boldsymbol{HD_t}}^T \left(y - \boldsymbol{H}\hat{\boldsymbol{x}}_0\right) \\ &= -2\rho_t \boldsymbol{J}_{\boldsymbol{D_t}}^T \boldsymbol{M}^T \left(\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{x}}_0\right) \end{aligned} \quad (44)$$

Therefore, $\rho_t \nabla_{\boldsymbol{x_t}} \left\|\boldsymbol{y} - \mathcal{A}\left(\hat{\boldsymbol{x}}_{\boldsymbol{0}}\right)\right\|_2^2 = \boldsymbol{J}_{\boldsymbol{D_t}} s \in T_s\left(\mathcal{M}, \mathcal{D}_t\left(\boldsymbol{x_t}\right)\right)$ where $s = -2\rho_t \boldsymbol{M^T}\left(\boldsymbol{y} - \boldsymbol{M}\hat{\boldsymbol{x}}_0\right)$, which comes from the result of Lemma 2. As the gradient is a vector on $T_s\left(\mathcal{M}, \mathcal{D}_t\left(\boldsymbol{x_t}\right)\right)$, we finally conclude that constraint term would guide the diffusion model to lie on the data manifold $\mathcal{M}$, which may lead to more accurate inference. □

REFERENCES

[1] D. Jiang, Z. Xu, H. Xu, Y. Han, Z. Chen, and Z. Yuan, "An approximation method of origin–destination flow traffic from link load counts," *Computers & Electrical Engineering*, vol. 37, no. 6, pp. 1106–1121, 2011.

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

[3] S. S. Hussain, M. A. Sultan, S. Qazi, and M. Ameer, "Intelligent traffic matrix estimation using levenberg-marquardt artificial neural network of large scale ip network," in *2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)*. IEEE, 2019, pp. 1–5.

[4] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *ACM SIGCOMM computer communication review*, vol. 35, no. 4, pp. 217–228, 2005.

[5] A. Sacco, F. Esposito, P. Okorie, and G. Marchetto, "{LiveMicro}: An edge computing system for collaborative telepathology," in *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[6] 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.

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

[8] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "Opentm: Traffic matrix estimator for openflow networks," in *Proceedings of the 11th International Conference on Passive and Active Measurement*, ser. PAM'10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 201–210.

[9] R. A. Memon, S. Qazi, and A. A. Farooqui, "Network tomography using genetic algorithms," in *TENCON 2012 IEEE Region 10 Conference*. IEEE, 2012, pp. 1–6.

[10] 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.

[11] X. Li, K. Xie, X. Wang, G. Xie, K. Li, J. Cao, D. Zhang, and J. Wen, "Tripartite graph aided tensor completion for sparse network measurement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 1, pp. 48–62, 2022.

[12] H. Zhou, D. Zhang, K. Xie, and Y. Chen, "Spatio-temporal tensor completion for imputing missing internet traffic data," in *2015 ieee 34th international performance computing and communications conference (ipccc)*. IEEE, 2015, pp. 1–7.

[13] 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.

[14] C. Tebaldi and M. West, "Bayesian inference on network traffic using link count data," *Journal of the American Statistical Association*, vol. 93, no. 442, pp. 557–573, 1998.

[15] J. Cao, D. Davis, S. Vander Wiel, and B. Yu, "Time-varying network tomography: Router link data," *Journal of the American statistical association*, vol. 95, no. 452, pp. 1063–1075, 2000.

[16] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "An

information-theoretic approach to traffic matrix estimation," *Computer Communication Review*, vol. 33, 07 2003.

[17] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1434841110000531

[18] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804515002854

[19] 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.

[20] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[21] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[22] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[23] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.

[24] A. Das, Y. Yang, T. Hospedales, T. Xiang, and Y.-Z. Song, "Chirodiff: Modelling chirographic data with diffusion models," *arXiv preprint arXiv:2304.03785*, 2023.

[25] X. Yuan, Y. Qiao, P. Zhao, R. Hu, and B. Zhang, "Traffic matrix estimation based on denoising diffusion probabilistic model," in *2023 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2023, pp. 316–322.

[26] H. Chung, J. Kim, M. T. Mccann, M. L. Klasky, and J. C. Ye, "Diffusion posterior sampling for general noisy inverse problems," *arXiv preprint arXiv:2209.14687*, 2022.

[27] J. Song, A. Vahdat, M. Mardani, and J. Kautz, "Pseudoinverse-guided diffusion models for inverse problems," in *International Conference on Learning Representations*, 2022.

[28] W. Yinhuai, Y. Jiwen, and Z. Jian, "Zero-shot image restoration using denoising diffusion null-space model," *arXiv preprint arXiv:2212.00490*, vol. 3, 2022.

[29] 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.

[30] D. Jiang, W. Wang, L. Shi, and H. Song, "A compressive sensing-based approach to end-to-end network traffic reconstruction," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 507–519, 2018.

[31] 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.

[32] K. Xie, H. Lu, X. Wang, G. Xie, Y. Ding, D. Xie, J. Wen, and D. Zhang, "Neural tensor completion for accurate network monitoring," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1688–1697.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] A. Sacco, F. Esposito, and G. Marchetto, "Completing and predicting internet traffic matrices using adversarial autoencoders and hidden markov models," *IEEE Transactions on Network and Service Management*, 2023.

[39] 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.

[40] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International conference on machine learning*. Pmlr, 2021, pp. 8821–8831.

[41] M. Roughan, "A case study of the accuracy of snmp measurements," *Journal of Electrical and Computer Engineering*, vol. 2010, pp. 1–7, 2010.

[42] H. Chung, B. Sim, D. Ryu, and J. C. Ye, "Improving diffusion models for inverse problems using manifold constraints," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 683–25 696, 2022.

[43] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[44] B. D. Anderson, "Reverse-time diffusion equation models," *Stochastic Processes and their Applications*, vol. 12, no. 3, pp. 313–326, 1982.

[45] Y. Zhu, K. Zhang, J. Liang, J. Cao, B. Wen, R. Timofte, and L. Van Gool, "Denoising diffusion models for

plug-and-play image restoration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 1219–1229.

[46] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[47] B. Efron, "Tweedie's formula and selection bias," *Journal of the American Statistical Association*, vol. 106, no. 496, pp. 1602–1614, 2011.

[48] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *Journal of the American statistical association*, pp. 365–377, 1996.

[49] B. Eriksson, P. Barford, R. Bowden, N. Duffield, J. Sommers, and M. Roughan, "Basisdetect: A model-based network event detection framework," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 451–464.

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[51] G. J. McLachlan and T. Krishnan, *The EM algorithm and extensions*. John Wiley & Sons, 2007.

[52] G. J. McLachlan, S. X. Lee, and S. I. Rathnayake, "Finite mixture models," *Annual review of statistics and its application*, vol. 6, pp. 355–378, 2019.

[53] S. Gu, D. Chen, J. Bao, F. Wen, B. Zhang, D. Chen, L. Yuan, and B. Guo, "Vector quantized diffusion model for text-to-image synthesis," 2022.

[54] 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.

[55] Y. Zhang, "Abilene network topology data and traffic traces," 2004.

[56] 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.

[57] H. Chen and J. Li, "Neural tensor model for learning multi-aspect factors in recommender systems," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 2449–2455, main track. [Online]. Available: https://doi.org/10.24963/ijcai.2020/339

[58] X. Wu, B. Shi, Y. Dong, C. Huang, and N. Chawla, "Neural tensor factorization for temporal interaction learning," *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:59528341

[59] H. Liu, Y. Li, M. Tsang, and Y. Liu, "Costco: A neural tensor completion model for sparse tensors," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:198117731

[60] A. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," *ArXiv*, vol. abs/2102.09672, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:231979499

[61] Y. Li, W. Liang, K. Xie, D. Zhang, S. Xie, and K. Li, "Lightnestle: Quick and accurate neural sequential tensor completion via meta learning," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

[62] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[63] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola, "A kernel method for the two-sample-problem," *Advances in neural information processing systems*, vol. 19, 2006.

[64] B. K. Sriperumbudur, K. Fukumizu, and G. R. Lanckriet, "Universality, characteristic kernels and rkhs embedding of measures." *Journal of Machine Learning Research*, vol. 12, no. 7, 2011.