Future Network Traffic Matrix Synthesis and Estimation Based on Deep Generative Models

Grigorios Kakkavas*, Michail Kalntis*, Vasileios Karyotis*† and Symeon Papavassiliou*
*School of Electrical & Computer Engineering, National Technical University of Athens
Iroon Polytechniou 9, Zografou, Athens, 15780, Greece
e-mails: gkakkavas@netmode.ntua.gr, el15123@central.ntua.gr, vassilis@netmode.ntua.gr, papavass@mail.ntua.gr
†Department of Informatics, Ionian University
Tsirigoti Square 7, Corfu, 49132, Greece
e-mail: karyotis@ionio.gr

***Abstract*—Traffic matrices (TMs) contain information that is essential for network management, traffic engineering, and anomaly detection. However, constructing a TM through direct traffic measurements has a high administrative and computational cost. A more feasible approach is to estimate the TM from the easily obtainable link load measurements. In this paper, we address the issue of traffic matrix estimation (TME) from link loads using a deep generative model – namely, a variational autoencoder (VAE) – to solve the respective ill-posed inverse problem. In particular, we train the VAE with historical data (previously observed TMs) and we leverage the trained decoder to transform TME into a minimization problem in the latent space, which in turn can be solved by employing a gradient-based optimizer. Furthermore, the trained decoder can be used for traffic matrix synthesis, i.e., for generating synthetic TM examples that have "similar" properties to the samples of the training set. Finally, we explore the incremental optimization of the sequence of objectives constructed from the sequence of decoders that we obtain at different stages of the VAE training. The performance of the proposed methods is evaluated using a publicly available dataset of actual traffic matrices recorded in a real backbone network.**

***Index Terms*—Network monitoring, network tomography, traffic matrix estimation, deep learning, variational autoencoder.**

I. Introduction

The traffic matrix (TM) quantifies the demand between all possible pairs of origin and destination entities (usually nodes or set of nodes) in a network. Its entries represent the volume of traffic flowing between the respective origins and destinations. The information available within a TM is essential for many network planning functions. Knowing the volume of expected demand and the locality of traffic flows is a fundamental prerequisite for capacity planning, network provisioning and slice dimensioning. The latter are all of high significance towards meeting the goal of serving the demand with satisfactory quality in terms of delay, packet loss rate and/or other Quality of Service (QoS) / Quality of Experience (QoE) parameters.

Particularly in the context of 5G, which is envisioned to be a multi-service network capable of accommodating a wide range of verticals and the diverse set of performance and service requirements that they entail over the same physical infrastructure, understanding the underlying traffic requirements, patterns, and trends is vital for Network Slicing [1]. Each network slice represents an end-to-end independent logical network, operating on the single shared physical infrastructure and comprising dedicated and/or shared resources (e.g., processing power, storage, bandwidth). Analyzing and predicting the traffic related to the targeted functionality will enable a smart, data-driven decision making process regarding the selection of an appropriate slice, the allocation of sufficient resources and the flexible adaptation in cases of network failures.

Many traffic engineering tasks such as routing and load balancing, as well as fault diagnosis and management, leverage the information of the distribution of traffic within the network and the ability to predict the latter's evolution in the presence of changes in topology, traffic patterns or even functional roles of specific elements. For example, in the context of the emerging network virtualization paradigm, the pursuit of the optimal placement (with regard to minimizing the waste of resources and fairly balancing the network load without performance degradation) of the virtualized network functions (VNFs) further highlights the importance of TMs [2].

The most straightforward approach for constructing a TM is obtaining direct measurements at the network ingress and egress points, through collecting packet traces, performing flow-level aggregation and/or employing packet/flow sampling [3]. However, such an approach requires the deployment of additional modules at nodes that consume storage and processing resources. Consequently, direct measurement gives rise to high administrative costs and computational overhead, possibly leading to performance degradation, since it can affect the forwarding performance of the involved nodes and does not scale well to larger networks. Another, more practical alternative is Traffic Matrix Estimation (TME) by means of path-level Network Tomography (NT) [4], which consists of inferring a TM from link-level measurements that are readily available to network administrators as SNMP link loads.

Recent advances and progress in deep neural networks and stochastic optimization methods have enabled the creation of deep generative models [5]. Considering any kind of observed

This work was supported by the European Commission in the framework of the H2020-ICT-19-2019 project 5G-HEART (Grant Agreement No. 857034).

data as a finite set of samples from an underlying distribution, these models constitute a powerful way of approximating this data distribution using unsupervised learning, and they are able to generate new "similar" data points. In this paper, we leverage a deep generative model to propose a data-driven method for solving the *ill-posed statistical inverse problem* of estimating the unobserved traffic matrix from the corresponding measured link loads (see Section III-A). The key contributions of our work are summarized as follows:

- The limited availability of public TM datasets poses an obstacle to researchers and/or professionals who wish to test and experimentally validate their proposed solutions. In order to synthesize traffic matrices that conform to the observed constraints of a particular network, we train a Variational Autoencoder (VAE) [6], [7] with the available historical data and we use the trained decoder neural network to generate new data points.
- Leveraging the trained decoder neural network of the VAE, we formulate the TME NT problem as an optimization problem in the latent space, by assuming that the solution is "similar" to the previously observed TMs and, therefore, can be generated by said decoder. Since the generative model is differentiable, the minimization of the objective function can be done using an iterative optimizer.
- We transform the aforementioned two-stage TME method (first VAE training and, then, optimization in the latent space) into a type of "concurrent" optimization over the parameters of the generative model and the latent vectors. To that end, motivated by [8], we incrementally optimize the sequence of objective functions constructed from the respective sequence of generative models that is obtained during different stages of the training process (i.e., as the parameters of the generative network change during training).
- We implement the proposed methods with all alternative options discussed in this paper and we publish[1] the source code under a permissive free software license, along with detailed documentation.
- We extensively evaluate the performance of the proposed methods, using a publicly available dataset of real traffic matrices recorded in the Abilene [9] backbone network over a period of 24 weeks.

The remainder of this paper is organized as follows. In Section II, we briefly review related work. Section III introduces the TME problem in the context of NT, and sets up the stage for our work regarding optimization methods and variational autoencoders. The main part and key contributions of this paper are listed in Section IV, which describes in detail the proposed method for TM synthesis and estimation, along with the "concurrent" incremental optimization process in the latent space. In Section V, we present the implementation details and the employed experimental setup, and we report the respective performance evaluation results. Finally, Section VI concludes the paper.

[1]https://github.com/MikeKalnt/VAE-TME

II. Related Work

Traffic matrix estimation has been extensively researched in the last two decades and remains still a rather active topic [10]–[14]. In this section, we focus on approaches that attempt to overcome the ill-posed (i.e., under-determined) nature of the TME from link load measurements NT problem by incorporating artificial neural networks. These approaches, which are most relevant to our work, do not rely on additional assumptions about origin-destination (OD) flows and statistical modeling techniques. Instead, as a general rule, they try to exploit large volumes of measurement data, in order to learn in a supervised fashion the spatio-temporal properties of the inverse system and obtain the direct mapping of link counts to the traffic matrix. After the formulation of the estimation inverse problem as a regression problem and the construction of a suitable cost/objective function, the optimal parameters of the employed neural network architecture can be obtained by an iterative optimization algorithm. Then, the traffic matrix can be estimated from the observed link loads using the trained network.

Jiang et al. [15] were among the first to introduce the use of neural networks for estimating large-scale traffic matrices. The method they proposed is based on a back-propagation neural network (BPNN), combined with the iterative proportional fitting procedure (IPFP). In [16], the authors combine a non-linear autoregressive exogenous model (NARX) that is capable of capturing the time-varying patterns of traffic, with the genetic algorithm (GA) for obtaining the optimized weights and biases. In [17], end-to-end traffic is decomposed into a low-frequency component that captures the change trend and a high-frequency component that reflects the fluctuation. The former is described with an auto-regressive (AR) model, wheras the latter is modeled by a BPNN. A deep belief network (DBN) architecture is used in [18], in order to capture the dynamic features of traffic and learn the properties of the ill-posed inverse inference system. The DBN is trained with pairs of historical data (link counts and respective traffic matrices) and can, then, be used to estimate the traffic matrix that corresponds to the provided input of link counts. This approach is further examined in the context of data center networks in [19].

More recent works [20], [21] have attempted to extend the input of the employed neural networks by explicitly incorporating the routing information and the network topological structure. More specifically, in [20], the authors use a BPNN whose input is formed as the product of the Moore-Penrose inverse of the network's routing matrix and the link load vector. The expectation maximization (EM) algorithm is applied to the output of the BPNN in order to improve the estimation accuracy. In [21], TME is examined in the light of graph embedding. The network is modeled as a time-varying graph that is transformed into a vector of estimated OD flows, by feeding link load adjacency matrices to a convolutional

neural network (CNN). A feedforward backpropagation neural network trained with the Levernberg-Marquardt algorithm (LMA) is employed in [22]. Finally, deep learning models have achieved promising results in TM prediction, an instance of multivariate time series prediction, where the goal is to predict future network-wide traffic assuming that historical TM data are available. For example, Zhao et al. [23] decompose the original TM series into multilevel subseries using the discrete wavelet decomposition. The spatial dependencies among flows are extracted by a CNN, while the temporal evolution features are captured by a long short-term memory neural network (LSTM) with a self-attention mechanism.

In our work, we leverage a deep generative model (namely, a variational autoencoder) to solve the TME inverse problem, an approach that has been applied successfully in the field of computational imaging [24]. Instead of trying to learn the mapping from link counts to the traffic matrix, we use the trained decoder network of the VAE to transform TME into a minimization problem in the latent space that can be solved using a gradient-based optimizer (see Section III-B).

III. Problem Formulation and Deep Learning Background

A. Network Tomography

Network tomography refers to the inference of unmeasured network attributes based on measurements realized at a subset of accessible network elements. The traffic matrix estimation problem from the NT perspective is based on a system of linear equations describing the relationship between the traffic matrix, the underlying routing and the observed link counts, and regards statistical inference methods for estimating the unobserved traffic matrices from the available link load measurements. In particular, assuming a network with n nodes and m links, the traffic matrix is an $n \times n$ matrix whose element at row i and column j represents the traffic between origin node i and destination node j. For convenience, the traffic matrix is often organized into a p-dimensional vector $\boldsymbol{x}$, where p is the number of OD flows. By denoting the m-dimensional vector of link counts as $\boldsymbol{y}$ and assuming that routing is fixed during the measurement period, we can formulate the linear model $\boldsymbol{y} = \boldsymbol{R} \cdot \boldsymbol{x}$, where $\boldsymbol{R}$ is the $m \times p$ routing matrix whose element r_{ij} is equal to 1 if OD flow j traverses link i or 0 otherwise. In this context, TME is posed as the inverse problem of recovering the $\boldsymbol{x}$ that corresponds to a specific measured $\boldsymbol{y}$, given a known $\boldsymbol{R}$. The difficulty lies in the fact that the aforementioned system of linear equations is heavily under-determined (i.e., matrix $\boldsymbol{R}$ is not full rank and there are many solutions that fit the observations), since the number of OD flows is almost always much larger than the number of links. This ill-posed nature of the TME NT problem is usually addressed by using statistical models and regularization to impose additional structural assumptions.

B. Optimization of Objective Functions

Gradient descent is a first-order iterative optimization algorithm for minimizing a differentiable objective function by updating the parameters in the opposite direction of its gradient with respect to these parameters. The size of the steps taken to reach a (local) minimum is determined by the learning rate. Depending on the amount of data used for the computation of the gradient of the objective function, the following variants of gradient descent are defined [25]:

- *Batch gradient descent*: The gradient of the objective function $\ell(\theta)$ with respect to the parameters θ is computed for the entire training dataset:

  $$\theta = \theta - \eta \cdot \nabla_\theta \ell(\theta), \text{ where } \eta \text{ is the learning rate.}$$

  Since the entire training set is considered every time the parameters are updated, batch gradient descent can be very slow and it is not suitable for online learning.
- *Stochastic gradient descent (SGD)*: A parameter update is performed for each training data point:

  $$\theta = \theta - \eta \cdot \nabla_\theta \ell(\theta; x_i; y_i),$$

  where x_i is the i-th training sample and y_i the respective label. SGD is usually much faster and can be used to learn online, i.e., with new samples on-the-fly. However, the frequent updates of high variance make the objective function to fluctuate a lot, which might hinder convergence to the exact minimum.
- *Mini-batch gradient descent*: Combining the previous two approaches, in mini-batch gradient descent, an update is performed for every mini-batch of n training samples:

  $$\theta = \theta - \eta \cdot \nabla_\theta \ell(\theta; x_{i:i+n}; y_{i:i+n}).$$

  In that way, the trade-off between the accuracy/efficiency of the parameter update and the respective time required is better balanced.

Kingma et al. [26] proposed Adam, an adaptive learning rate optimizer that computes per-parameter adaptive learning rates. This is achieved by maintaining an exponentially decaying average of past gradients and past squared gradients, using bias-corrected estimates of their first and second moments. It has been demonstrated to be fast in terms of convergence and highly robust for modeling complex structures.

C. Variational Autoencoders

Variational autoencoders [6], [7] are a deep generative model with latent variables. They can be defined as a regularized version of autoencoders that avoids overfitting and makes the generative process possible by ensuring that the latent space is continuous (i.e., points of the latent space that are close are not decoded to completely different contents) and complete (i.e., any point sampled from the latent space corresponds to a "meaningful" decoded content). Contrary to vanilla autoencoders, VAEs encode inputs as distributions over the latent space instead of single points. In practice, these encoding distributions are regularized to be close to a standard Normal. Since the posterior distribution of the encoded variable given the decoded one is intractable for a continuous latent space, the variational inference technique is used to

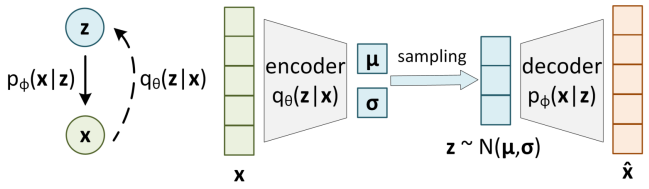

Fig. 1. The variational autoencoder model architecture.

find a deterministic approximation. The training process of the model can be summarized as follows. First, the input is encoded as a distribution over the latent space and a latent point is generated by sampling from this distribution. Then, the sampled point is decoded and the reconstruction error is computed and, finally, back-propagated through the network. The loss function that is minimized during training consists of a reconstruction term that aims to optimize the performance of the encoding-decoding scheme and a regularization term that ensures the desired properties of the latent space. This regularization term is expressed as the Kulback-Leibler (KL) divergence between the returned encoding distribution and a standard Normal prior.

More formally, a VAE consists of an encoder, a decoder, and a loss function. The enconder $q_\theta(z \mid \boldsymbol{x})$ is a neural network with parameters (i.e., weights and biases) θ that compresses its input $\boldsymbol{x}$ into a lower-dimensional latent representation space. In particular, the encoder outputs the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ of $q_\theta(z \mid \boldsymbol{x})$, which is regularized to be close to a standard Normal distribution. The hidden representation z can be obtained by sampling from this distribution. The decoder $p_\phi(\boldsymbol{x} \mid z)$ is another neural network with parameters ϕ that takes as input the hidden representation z and reconstructs (i.e., "decodes") the original input $\hat{\boldsymbol{x}}$. In order to measure how well the decoder reconstructs the input $\boldsymbol{x}$ given its hidden representation z, we use the reconstruction log-likelihood $\log p_\phi(\boldsymbol{x} \mid z)$. The loss function for a data point $\boldsymbol{x}$ is

$$\ell(\theta, \phi; \boldsymbol{x}) = D_{KL}(q_\theta(z|\boldsymbol{x}) \| p(z)) - \mathbb{E}_{q_\theta(z|\boldsymbol{x})}\left[\log p_\phi(\boldsymbol{x}|z)\right],$$

where the expectation in the second term (i.e., the reconstruction error) is taken with respect to the encoder's distribution over the representations, and the first term denotes the Kullback-Leibler divergence between the encoder's distribution $q_\theta(z \mid \boldsymbol{x})$ and the prior distribution $p(z)$, which as previously mentioned is specified as a standard Normal $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. The variational autoencoder is trained to minimize loss with respect to the parameters of the encoder θ and the parameters of the decoder ϕ using stochastic gradient descent. Fig. 1 illustrates the described architecture.

IV. The Proposed VAE Enabled Methods

A. Traffic Matrix Synthesis

Traffic matrices are necessary as input to a variety of network design, planning and traffic engineering tasks. However, the limited number of publicly available TM datasets, combined with the high administrative and computational cost of constructing TMs by direct measurements at a given network, constitute a serious barrier to the researches and professionals who wish to test and evaluate the performance of their proposed network applications, algorithms and protocols via simulations or experimental trials. One way to overcome this obstacle is to artificially create realistic traffic matrices, i.e., to synthetically generate OD traffic levels and organize them in a matrix that is feasible and well-matched to a given topology, potentially obeying the spatio-temporal patterns previously observed in real traffic matrices of this particular network. In this work, we propose the use of a deep generative model – particularly, a variational autoencoder – for traffic matrix synthesis. Given a number of prior TMs used for the training of the VAE, new synthetic TMs that are "similar" to the employed examples can be generated by the trained decoder. These prior TMs can be historical data that have been obtained at some point in the past, or can be explicitly constructed for this purpose, mitigating the overall induced cost by reducing the number of the measured TMs; only a subset is directly measured, whereas the rest can be synthetically generated. Apart from its intrinsic value, traffic matrix synthesis is a necessary enabling step for the estimation method that will be presented in the following section.

A deep generative model learns the underlying data distribution of the training set, which explains how data are generated, and allows us to sample from it in order to produce new data points with some variations. As described in Section III-C, a VAE maps its input to a z-dimensional Normal distribution (a task performed by the encoder) and reconstructs sampled hidden representations of this latent space back into its original dimension (a task performed by the decoder). More precisely, the encoder outputs a mean and a standard deviation z-dimensional vector corresponding to an approximate Normal encoding distribution that must be close to the posterior distribution of the latent variable given the decoded one. This is achieved by modifying the loss function used in training to include, apart from the typically used reconstruction error, a KL divergence term as well, providing a measure of the extent to which the approximate encoding distribution is different from the prior distribution of the latent variable (assumed to be standard Normal). In this particular case, the KL divergence of a diagonal multivariate Normal distribution with mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ and a standard Normal with zero mean and unit variance takes the form:

$$D_{KL}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}) \| \mathcal{N}(\mathbf{0}, \boldsymbol{I})) = \frac{1}{2} \sum_{i=1}^{z} \left(\sigma_i^2 + \mu_i^2 - 1 - \ln\left(\sigma_i^2\right)\right).$$

Mapping each input to such a multivariate Normal distribution instead of a fixed point ensures that the latent space is continuous and that hidden representations (i.e., latent vectors) are regularized to be centered around the origin. As a consequence, new synthetic data points can be generated from latent vectors sampled from a z-dimensional standard Normal distribution and fed to the trained decoder neural network.

B. Traffic Matrix Estimation

In this section, we address the problem of estimating the traffic matrix $\boldsymbol{x}$ that corresponds to a specific measured vector of link counts $\boldsymbol{y}$, given a known routing matrix $\boldsymbol{R}$, as a constrained minimization problem with generative models. The idea is to search in the latent space of a suitable generative model for a generated TM that best explains the measured link loads. Given a set of previously observed TMs, we confine the TM estimate to have properties similar to these past TMs. In particular, we leverage a variational autoencoder that is trained with a dataset of expected targets, as described in Section IV-A. The trained decoder neural network of the VAE learns the underlying distribution of the historical data and the observed spatio-temporal patterns of traffic. As a consequence, it is capable of synthesizing artificial examples $\boldsymbol{x}$ from random low-dimensional latent vectors $\boldsymbol{z}$ that fit the examined network topology and resemble the samples of the training set.

Assuming that the solution we are searching for belongs to the range of the trained decoder (i.e., it can be synthesized by said decoder), we can use the learned distribution to transform the TME inverse problem in the following minimization problem in the low-dimensional latent space:

$$\arg\min_{\boldsymbol{z}} \ \left\|\boldsymbol{y} - \boldsymbol{R} \cdot d(\boldsymbol{z})\right\|_2^2, \tag{1}$$

where $d(\cdot)$ represents the decoder neural network. In this way, we constrain the solution to be within the range of the employed generative model (the estimated $\boldsymbol{x}$ is by definition generated by $\boldsymbol{z}$) and we ensure agreement to the link measurements by minimizing the distance to the observations $\boldsymbol{y}$.

In order to encourage the exploration of regions that are preferred by the decoder, we can add a regularization term to the employed objective function. Taking into consideration that the VAE imposes a Gaussian prior distribution on $\boldsymbol{z}$ (see Section III-C), the minimization problem can be restated as follows:

$$\arg\min_{\boldsymbol{z}} \ \left[\left\|\boldsymbol{y} - \boldsymbol{R} \cdot d(\boldsymbol{z})\right\|_2^2 + c \cdot \left\|\boldsymbol{z}\right\|_2^2\right], \tag{2}$$

where parameter c provides a means to balance between the importance of the prior and the measurement error. Since decoder d is differentiable, the gradient of the objective function can be computed using the chain rule and the optimization problem can be solved by starting from a random initial point $\boldsymbol{z}_0$ and employing a gradient based optimizer (e.g., SGD or Adam). The optimization iterations required can be reduced by choosing a "good" initial latent vector $\boldsymbol{z}_0$. To that end, we examine a number K of random latent vectors and choose the one that has the smallest distance to the measured link counts:

$$\boldsymbol{z_0} = \boldsymbol{z_k} \text{ s.t. } \left\|\boldsymbol{y} - \boldsymbol{R} \cdot d(\boldsymbol{z}_k)\right\|_2^2 \leq \left\|\boldsymbol{y} - \boldsymbol{R} \cdot d(\boldsymbol{z}_i)\right\|_2^2, \text{ for } i \in 1, \cdots K.$$

After finding the optimal $\boldsymbol{z}^*$ that minimizes (1) or (2), we can obtain the estimated TM through the mapping $\hat{\boldsymbol{x}} = d(\boldsymbol{z}^*)$.

Compared to conventional NT TME methods that rely on additional assumptions regarding the OD flows, in the proposed deep learning enabled approach, all prior knowledge needed for the reconstruction of the TM from the observed link measurements is indirectly learned from the training dataset.

C. Incremental Optimization

The estimation method described in the previous section constitutes a two-step procedure. First, we train a VAE with historical data and, then, we leverage the trained decoder to formulate the TME problem as a minimization problem in the latent space, which we solve using a gradient based optimizer. However, the objective functions in (1) or (2) might be non-convex with many local minima, in which case gradient descent might not work well (i.e., might not converge to the true global minimum). To address this potential issue, instead of constructing the objective with the fixed trained decoder and then applying gradient descent, we incrementally optimize the sequence of objective functions constructed with the sequence of decoder networks that are obtained during different stages of the VAE training by adopting the respective parameter values. This kind of "concurrent" optimization over the latent vectors and the parameters of the decoder network has been experimentally shown to find the global minimum even in cases where direct gradient descent on the objective formed with the final learned network fails [8].

More precisely, given a specific architecture for the VAE, we obtain the sequence of decoder networks $d_0, d_1, \cdots, d_T$ (d_T represents the final trained network) from the sequence of parameters $\phi_0, \phi_1, \cdots, \phi_T$, produced at different stages of training (i.e., every predetermined number of training epochs). As soon as we have decoder d_i, we construct the objective $\left\|\boldsymbol{y} - \boldsymbol{R} \cdot d_i(\boldsymbol{z})\right\|_2^2$ or $\left\|\boldsymbol{y} - \boldsymbol{R} \cdot d_i(\boldsymbol{z})\right\|_2^2 + c \cdot \left\|\boldsymbol{z}\right\|_2^2$ and we find optimum $\boldsymbol{z}_i^*$ by applying, for example, Adam. This "current" optimum will be used for the initialization of Adam when optimizing the next objective. That is, after another predetermined number of VAE training epochs are completed and we obtain the decoder d_{i+1}, we will use $\boldsymbol{z}_i^*$ as the starting point for the Adam optimizer on the new objective $\left\|\boldsymbol{y} - \boldsymbol{R} \cdot d_{i+1}(\boldsymbol{z})\right\|_2^2$ or $\left\|\boldsymbol{y} - \boldsymbol{R} \cdot d_{i+1}(\boldsymbol{z})\right\|_2^2 + c \cdot \left\|\boldsymbol{z}\right\|_2^2$ and we will get the updated optimum $\boldsymbol{z}_{i+1}^*$. At the end, the TM estimate will be $\hat{\boldsymbol{x}} = d_T(\boldsymbol{z}_T^*)$.

V. Performance Evaluation

A. Experimental Setup and Implementation

In order to evaluate the performance of the proposed VAE-enabled methods, we have conducted numerical experiments with publicly available data [9], collected from the Abilene network (Fig. 2) for a period of 24 weeks. The network consists of 12 nodes, resulting in 144 traffic pairs (i.e., a 12×12 TM), which are captured in 5-minute intervals for consecutive weeks from 2004-03-01 to 2004-09-10. The provided dataset includes also the routing matrix of the network and the OSPF weight of every link. We used the first 13 weeks (i.e., the first 26 208 TMs) as the training set and the 14th week (i.e., the next 2016 TMs) as the testing set. We employed the Adam optimizer with a learning rate of 0.001, and ReLU was chosen as the activation function for the neural networks. The number of dimensions of the latent space was set to 10.

Fig. 2. The Abilene network.

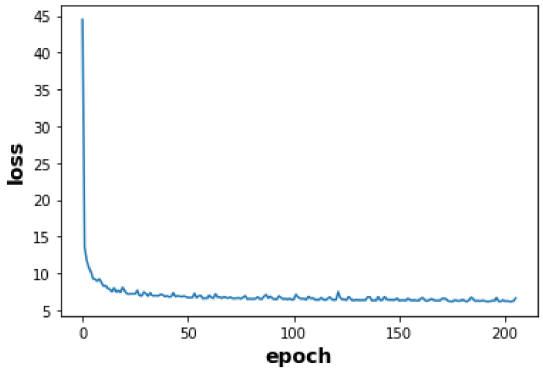

Fig. 3. Evolution of loss along training epochs.

Tables I and II present the structure of the VAE's encoder and decoder, respectively. The implementation was done using the Keras [27] deep learning API, running on top of the TensorFlow [28] machine learning platform.

TABLE I
STRUCTURE OF VAE ENCODER

Layer type	Kernel	Stride	Padding	Output Shape
Input	-	-	-	(12, 12, 1)
Dropout	-	-	-	(12, 12, 1)
Conv2D	(3, 3)	(2, 2)	SAME	(6, 6, 32)
Conv2D	(3, 3)	(2, 2)	SAME	(3, 3, 64)
Conv2D	(3, 3)	(1, 1)	SAME	(3, 3, 128)
Flatten	-	-	-	(1152)
Dense	-	-	-	(64)
Dense	-	-	-	(10)
Dense	-	-	-	(10)
Sampling	-	-	-	(10)

TABLE II
STRUCTURE OF VAE DECODER

Layer type	Kernel	Stride	Padding	Output Shape
Input	-	-	-	(10)
Dense	-	-	-	(64)
Dense	-	-	-	(576)
Reshape	-	-	-	(3, 3, 64)
Conv2DTranspose	(3, 3)	(1, 1)	SAME	(3, 3, 128)
Conv2DTranspose	(3, 3)	(2, 2)	SAME	(6, 6, 64)
Conv2DTranspose	(3, 3)	(2, 2)	SAME	(12, 12, 32)
Conv2DTranspose	(3, 3)	(1, 1)	SAME	(12, 12, 1)

B. Results and Discussion

As mentioned in Section IV-B and validated during our experiments, by choosing a "good" initial latent vector we can reduce the number of the required optimization iterations. In particular, after selecting the "best" initial latent vector in terms of distance to the measured link counts among 3000 random candidates for minimization (1) and 500 for minimization (2), we managed to reduce the optimization iterations from 10 000 to 5000 for both cases. We note that all the subsequently reported results are obtained following this approach. Fig. 3 illustrates the VAE's total loss (i.e., reconstruction error and KL divergence) over the training epochs.

The performance of the proposed TME methods is evaluated using the following metrics: the root mean square error (RMSE), the normalized mean absolute error (NMAE), the spatial relative error (SRE) that expresses the relative estimation error of each individual OD flow over its entire lifetime, and the temporal relative error (TRE) that summarizes the relative estimation error of all OD flows (i.e., the entire TM) at a given time point. The aforementioned errors are calculated as follows:

$$\mathrm{RMSE}(t) = \frac{\left\|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\right\|_2}{\sqrt{N}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{\boldsymbol{x}}_t(i) - \boldsymbol{x}_t(i)\right)^2},$$

$$\mathrm{NMAE}(t) = \frac{\left\|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\right\|_1}{\left\|\boldsymbol{x}_t\right\|_1} = \frac{\sum_{i=1}^{N}\left|\hat{\boldsymbol{x}}_t(i) - \boldsymbol{x}_t(i)\right|}{\sum_{i=1}^{N}\left|\boldsymbol{x}_t(i)\right|},$$

$$\mathrm{SRE}(i) = \frac{\left\|\hat{\boldsymbol{x}}_{1:T}(i) - \boldsymbol{x}_{1:T}(i)\right\|_2}{\left\|\boldsymbol{x}_{1:T}(i)\right\|_2} = \frac{\sqrt{\sum_{t=1}^{T}\left(\hat{\boldsymbol{x}}_t(i) - \boldsymbol{x}_t(i)\right)^2}}{\sqrt{\sum_{t=1}^{T}\left(\boldsymbol{x}_t(i)\right)^2}},$$

$$\text{and } \mathrm{TRE}(t) = \frac{\left\|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\right\|_2}{\left\|\boldsymbol{x}_t\right\|_2} = \frac{\sqrt{\sum_{i=1}^{N}\left(\hat{\boldsymbol{x}}_t(i) - \boldsymbol{x}_t(i)\right)^2}}{\sqrt{\sum_{i=1}^{N}\left(\boldsymbol{x}_t(i)\right)^2}},$$

where $i = 1, 2, \cdots, N$ indicates each individual OD flow and $t = 1, 2, \cdots, T$ denotes each measurement time point. Table III summarizes the results for the three examined variations of the proposed TME method. As can be seen, minimization (2) with $c = 0.1$ does indeed improve all the employed metrics (particularly SRE) by incorporating the regularization term.

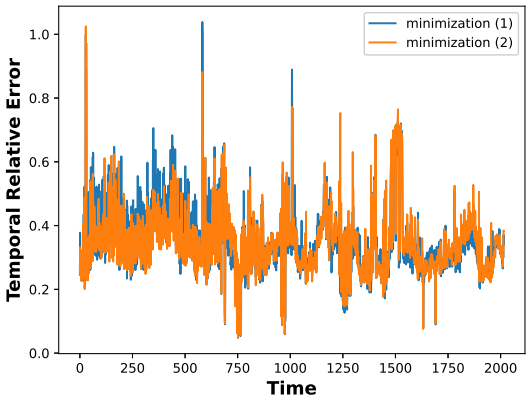

Fig. 4. Temporal relative errors.

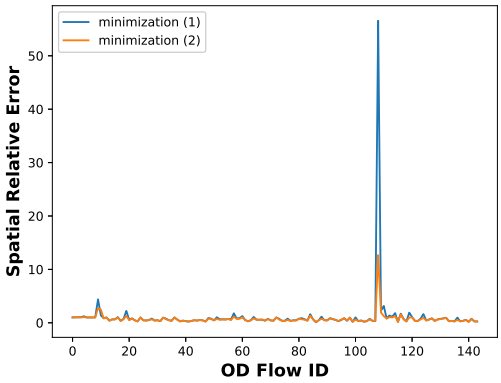

Fig. 5. Spatial relative errors.

Regarding the concurrent incremental optimization, we have used the objective (2), the number of training epochs after which we obtain each decoder was set to 20, and the number of optimization iterations for every objective of the sequence was set to 3000. For this case only, the results reported in Table III regard the first 500 TMs of the testing set.

TABLE III
ESTIMATION ERRORS

VAE Training and Minimization (1)				
Error	**Mean**	**Median**	**Std**	**Max**
RMSE (Mbps)	13.9045	12.6126	5.2089	59.6762
NMAE	0.3544	0.3415	0.0824	0.8176
TRE	0.3557	0.3320	0.1098	1.0387
SRE	1.1233	0.6323	4.6677	56.5583
VAE Training and Minimization (2)				
Error	**Mean**	**Median**	**Std**	**Max**
RMSE (Mbps)	13.6893	12.6701	5.0450	60.6747
NMAE	0.3466	0.3350	0.0748	0.9496
TRE	0.3488	0.3381	0.0989	1.0249
SRE	0.7330	0.5769	1.0698	12.6865
Concurrent Training and Minimization (2)				
Error	**Mean**	**Median**	**Std**	**Max**
RMSE (Mbps)	13.3673	12.4220	5.5080	43.2572
NMAE	0.4262	0.3972	0.1473	1.3309
TRE	0.4033	0.3791	0.1505	1.3824
SRE	1.7464	0.8027	6.9723	81.7783

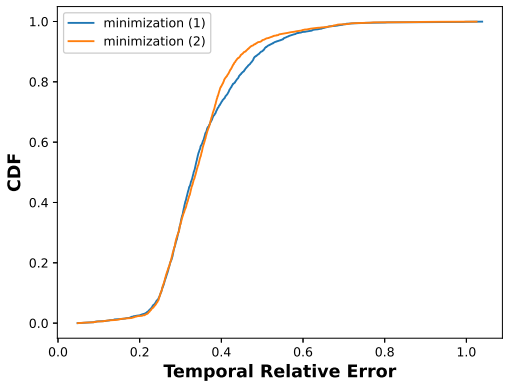

Fig. 6. Cumulative distribution function (CDF) of TREs.

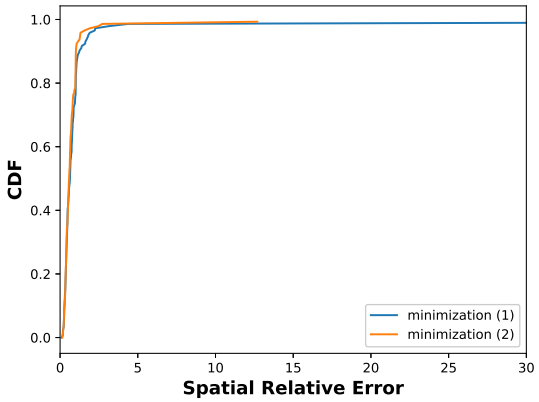

Fig. 7. Cumulative distribution function (CDF) of SREs.

Fig. 4 illustrates the temporal relative errors for the 2016 TMs of the testing set (each TM corresponds to a time point) for the two examined minimization objectives, and Fig. 5 depicts the spatial relative errors of every OD Flow (144 in total). The respective cumulative distribution functions (CDFs) are presented in Fig. 6 and Fig. 7. As can be observed, minimization (2) consistently leads to slightly better estimates for all OD flows and testing TMs.

Finally, we would like to point out that the goal of this experimental evaluation was to verify the feasibility of the proposed method and the suitability of using a VAE deep generative model for solving the inverse TME problem. As

such, in this work, we did not focus on finding the best (hyper)parameters that would potentially lead to the optimal estimates for the conducted experiments.

VI. Conclusion

In this paper, we have addressed traffic matrix estimation from link-level measurements, an ill-posed inverse problem belonging to the broad class of compressed sensing. However, instead of relying on sparsity as usual [4], we proposed the use of a deep generative model as a prior for the TM reconstruction from compressive measurements. In particular, we have trained a variational autoencoder over a set of historical data (i.e., previously measured TMs) and we have used the trained decoder for traffic matrix synthesis and for transforming the TME NT problem into a minimization problem in the low-dimensional latent space. Furthermore, we have explored the alternative approach of incrementally optimizing the sequence of objective functions corresponding to the sequence of the decoder's parameters, as the latter are produced at different stages of the VAE's training process. Experimental results validated the achieved performance of the proposed methods.

A trained generative model can reliably generate data points that are similar to its training set. Therefore, the use of a fixed pretrained decoder for TME is expected to work well only if the unknown TM belongs to the range of the employed decoder, making the proper selection of suitable training sets critical for the success of the proposed method. Moreover, training a generative model can be challenging in terms of computational resources and time. Investigating the use of *untrained generative models* (which, somewhat counter-intuitively, have recently been shown to achieve promising results [29], [30]) for overcoming the aforementioned issues is left for future work. Finally, another direction worth exploring is the use of *conditional generative models* [31], which allow the generation of multiple solutions from the same measurements (i.e., sampling from the recovered posterior distribution).

References

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, May 2017.

[2] U. Fiore, P. Zanetti, F. Palmieri, and F. Perla, "Traffic matrix estimation with software-defined NFV: Challenges and opportunities," *J. Comput. Sci.*, vol. 22, pp. 162–170, Sep. 2017.

[3] P. Tune and M. Roughan, "Internet traffic matrices: A primer," in *Recent Advances in Networking, Volume 1*, H. Haddadi and O. Bonaventure, Eds. ACM SIGCOMM, 2013.

[4] 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, Jan. 2020.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[6] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2014. [Online]. Available: https://arxiv.org/abs/1312.6114v10

[7] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of the 31st International Conference on Machine Learning*, vol. 32, no. 2. Bejing, China: PMLR, 2014, pp. 1278–1286.

[8] G. Song, Z. Fan, and J. Lafferty, "Surfing: Iterative optimization over incrementally trained deep networks," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 15 034–15 043.

[9] Y. Zhang, "Abilene network topology data and traffic traces." [Online]. Available: https://www.cs.utexas.edu/~yzhang/research/AbileneTM/

[10] L. Nie, Y. Li, and X. Kong, "Spatio-temporal network traffic estimation and anomaly detection based on convolutional neural network in vehicular ad-hoc networks," *IEEE Access*, vol. 6, pp. 40 168–40 176, 2018.

[11] D. Li *et al.*, "Estimating SDN traffic matrix based on online adaptive information gain maximization method," *Peer-to-Peer Netw. Appl.*, vol. 12, no. 2, pp. 465–480, Mar. 2018.

[12] F. Xiao, L. Chen, H. Zhu, R. Hong, and R. Wang, "Anomaly-tolerant network traffic estimation via noise-immune temporal matrix completion model," *IEEE J. Select. Areas Commun.*, vol. 37, no. 6, pp. 1192–1204, Jun. 2019.

[13] M. Abareshi, M. Zaferanieh, and M. R. Safi, "Origin-destination matrix estimation problem in a Markov chain approach," *Netw Spat Econ*, vol. 19, no. 4, pp. 1069–1096, Feb. 2019.

[14] D. Jiang, W. Wang, L. Shi, and H. Song, "A compressive sensing-based approach to end-to-end network traffic reconstruction," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 507–519, Jan. 2020.

[15] 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, Jan. 2011.

[16] A. Omidvar and H. Shahhoseini, "Intelligent IP traffic matrix estimation by neural network and genetic algorithm," in *2011 IEEE 7th International Symposium on Intelligent Signal Processing*. IEEE, Sep. 2011.

[17] D. Jiang, Z. Zhao, Z. Xu, C. Yao, and H. Xu, "How to reconstruct end-to-end traffic based on time-frequency analysis and artificial neural network," *AEU - International Journal of Electronics and Communications*, vol. 68, no. 10, pp. 915–925, Oct. 2014.

[18] L. Nie, D. Jiang, L. Guo, and S. Yu, "Traffic matrix prediction and estimation based on deep learning in large-scale IP backbone networks," *J. Netw. Comput. Appl.*, vol. 76, pp. 16–22, Dec. 2016.

[19] L. Nie, D. Jiang, L. Guo, S. Yu, and H. Song, "Traffic matrix prediction and estimation based on deep learning for data center networks," in *2016 IEEE Globecom Workshops (GC Wkshps)*. IEEE, Dec. 2016.

[20] H. Zhou, L. Tan, Q. Zeng, and C. Wu, "Traffic matrix estimation: A neural network approach with extended input and expectation maximization iteration," *J. Netw. Comput. Appl.*, vol. 60, pp. 220–232, Jan. 2016.

[21] 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," *Trans Emerging Tel Tech*, vol. 30, no. 6, Mar. 2019.

[22] 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, Dec. 2019.

[23] J. Zhao, H. Qu, J. Zhao, and D. Jiang, "Spatiotemporal traffic matrix prediction: A deep learning approach with wavelet multiscale analysis," *Trans Emerging Tel Tech*, vol. 30, no. 12, Jun. 2019.

[24] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett, "Deep learning techniques for inverse problems in imaging," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 39–56, May 2020.

[25] S. Ruder, "An overview of gradient descent optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1609.04747v2

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*.

[27] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io

[28] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: http://tensorflow.org/

[29] V. Lempitsky, A. Vedaldi, and D. Ulyanov, "Deep image prior," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2018.

[30] R. Heckel and P. Hand, "Deep decoder: Concise image representations from untrained non-convolutional networks," in *International Conference on Learning Representations*, 2019.

[31] J. Adler and O. Öktem, "Deep Bayesian inversion," 2018. [Online]. Available: https://arxiv.org/abs/1811.05910v1