Completing and Predicting Internet Traffic Matrices Using Adversarial Autoencoders and Hidden Markov Models

Alessio Sacco, *Member, IEEE,* Flavio Esposito, *Member, IEEE,* and Guido Marchetto, *Senior Member, IEEE*

***Abstract*—Internet traffic matrices are used nowadays for a variety of network management operations, from planning to repairing. Despite years of research on the topic, obtaining a global view of traffic is still challenging and error-prone. Due to flaws in the measurement systems and possible failure in data collection tools, missing values are unavoidable. It is thus helpful for many network operators to recover the missing data from the partial direct measurements. While some existing matrix completion methods allowed this reconstruction, they do not fully consider network traffic behavior and hidden traffic characteristics, showing the inability to adapt to multiple scenarios. Others instead make assumptions about the matrix structure that may be invalid or impractical, curtailing the applicability. In this paper, we propose *Hide & Seek*, a novel matrix completion and prediction algorithm based on a combination of generative autoencoders and Hidden Markov Models. After an extensive experimental evaluation based on both real-world datasets and on a testbed, we demonstrated how our algorithm can accurately reconstruct missing values while also predicting their short-term evolution.**

***Index Terms*—traffic matrix, machine learning, inference**

I. Introduction

A Traffic Matrix (TM), representing the volume of network flows between all possible origin-destination (OD) pairs in the network over a given time interval, is a critical input for many distributed system management tasks, including capacity planning, anomaly detection, and even business intelligence. For example, they can help with provider selection in routing [1]–[3], network resources provisioning for highly-demanding applications [4], network debugging [5], or even with network anomaly detection and network security [6], [7].

Despite their importance, obtaining a complete TM at any given time is a challenge. TM incompleteness may be caused by measurement impracticality [1], (voluntary) data amputations [8], or both. For example, mirroring an interface with a fine-grain resolution can significantly impact the performance of network elements, and the distributed nature of the network infrastructure often hinders visibility. Despite several years of research on this topic, operators still rely mostly on low-resolution measurements such as SNMP messages, and even excellent measurement systems suffer from errors and missing data [8], [9]. For this reason, TM is often required to be complete or reconstructed before it can be used in any application, or as an input of new Machine Learning (ML) model utilized in network management operations, *e.g.*, [10].

The TM estimation problem finds a foundation in statistical signal processing [11], where the community identified sufficient conditions and algorithms to estimate the missing elements of a partially-observed matrix. The majority of existing techniques to estimate a TM's element given a limited set of measurements rely on network inference and network tomography methods, *e.g.*, [12], [13]. In addition to those methods, many other network inference problems are formulated as an Under-Determined Linear Inverse (UDLI) problem, but those solutions only work when the number of measurements is sufficient to uniquely and accurately determine the solution [14].

Other authors have studied how, in some cases, a TM can be efficiently completed when at most some specific portions are missing [15]–[17]. Such general approaches are based on the assumption that traffic matrix elements show strong statistical regularities and there are predictable relationships between elements. In such a way, the missing elements can often be cast in terms of linear functions of observable elements, leveraging the presence of a low effective rank that enables splitting the TM into smaller submatrices. Once these conditions are verified, a statistical inference method is applied, often based on signal processing, to infer the missing elements of the TM. While these solutions are sound, they only work on matrices that have low effective rank, see, *e.g.*, [2]. Moreover, although the literature has already addressed the problem from both spatial and temporal perspectives, giving birth to Machine Learning (ML)-based methods as in [18]–[20], a more general approach that can account simultaneously for both space and time is still missing, *i.e.*, a method that can both solve the matrix completion problem and predict future values of such matrix elements.

To this aim, we propose *Hide & Seek*, a novel solution that can complete the TM starting from *hidden information* while also predicting the future values of these missing entries. Our solution is based on an augmented Hidden Markov Model (HMM), where the traditionally employed Viterbi algorithm [21] is replaced with a more performant algorithm based on Adversarial AutoEncoder (AAE) [22]. In particular, the AAE-based encoding method is applied to complete the matrix, while the more general HMM keeps track of the evolution of the traffic data over time to predict the next value.

This work has been partially supported by Comcast and by NSF awards 1647084, 1836906, and 2201536.

Alessio Sacco and Guido Marchetto are with DAUIN, Politecnico di Torino, 10129 Turin, Italy (e-mail: alessio_sacco@polito.it, guido.marchetto@polito.it).

Flavio Esposito is with the Department of Computer Science, Saint Louis University, St. Louis, MO 63103 USA (e-mail: flavio.esposito@slu.edu).

The key to our matrix completion/prediction approach is the ability to observe a sufficiently useful subset of the matrix entries, which we denote as *hidden information*. We find this piece of information both using the ability of AAE and with the application of some eXplainable AI (XAI) techniques to determine the more important features. *The main advantage of our method is that we do not rely on any statistical assumptions about the rank of the traffic matrix*. Conversely, in our model, we convert the estimation problem into a process aiming to learn the hidden relationship between the partial traffic data, viewed as hidden information, and the missing traffic value.

In particular, extending [23], our contribution in this paper is two-folds. (i) We first propose a traffic matrix completion and prediction algorithm built atop the traditional HMM. Given the limited set of requirements and its learning-based nature, the algorithm is applicable in a variety of contexts, within traffic inference or outside network management. (ii) Then, we study how to improve the reliability of a network analyzer tool in terms of what information any solution using network data for decisions should tolerate the absence. In particular, by using XAI, we attempt to answer the question *what is the piece of information more crucial in matrix completion and what are the data redundant, whose absence is tolerated?*

We extensively evaluate our solution on (i) real-world Internet traces, namely collections from the Abilene, GEANT, and Mawi networks, and (ii) in an emulated SDN-based testbed scenario over Mininet. We demonstrate how effective our AAE-based model is in finding the hidden relationship between the missing value and the observed traffic entries that are adjacent in the TM. Besides, the results also confirm that HMM can properly predict the evolution of these unknown entries.

The rest of the paper is structured as follows. We discuss in Section II the existing literature about TM completion problem. Section III describes our model used and presents the specific problems addressed by *Hide & Seek* (*H&S* in short). We then present in Section IV the methods used in the solution, highlighting how we combined AAE with HMM in our algorithm. We present results in Section V, and finally, we conclude our paper in Section VI.

II. Related Work

Given the variety of fields where it finds applicability and the importance in networking, the problem of traffic matrix estimation has been well-studied and addressed from different angles [24]. Common methods for matrix completion are based on the incorporation of side information from different sources, such as total incoming bytes and number of customers [12]. For example, in [13] the proposed solution takes advantage of using multiple readily available data sources. In particular, the combination of flow measurement and link load measurement is used to effectively identify and remove dirty data, and this approach can reduce errors in traffic matrix estimation. Similarly, side information can be used for an active version of matrix completion, where queries can be made to the true underlying matrix, as it has been proposed in [2]. By unifying a matrix-completion approach and a querying strategy into a single algorithm, their solution is able to identify and alleviate insufficient information by judiciously querying a small number of additional entries.

Another widely common approach is the low-rank matrix completion, spanning a wide range of techniques, from norm minimization [11], to singular value thresholding [25], to alternating minimization [16], to mention a few. These approaches share the assumption that the whole matrix has low rank, posing an optimization to fit the entire matrix with a single rank-r model.

At the same time, some studies have acknowledged some spatial and temporal properties in the TM. Based on the spatial traffic feature, in [26] TM is modeled as multi-Gaussian models and then used to estimate the missing data. Using the traffic spatial affinity feature, the TM is partitioned into many clusters by spectral clustering, and then the subparts of TM with similar behavior are identified to finish the matrix completion process. To recover the missing entries in traffic data, also spatio-temporal tensor completion methods have been studied in the literature [8], [9], [27]. For example, [27] introduces a tensor (a multidimensional array) to model a time series of pure spatial traffic matrices. To extract this latent structure of traffic using tensor factorization, the model takes into account the lower-dimensional latent structure of network traffic and hidden traffic characteristics.

Different from these solutions, we propose a learning-based approach that combines statistical with ML features and takes advantage of the concept of spatial affinity inside a smaller submatrix, after having opportunely studied the decisions taken with XAI. Recently, the application of ML-based algorithms to restore missing information has appeared as a viable approach. For example, in [28] a neural network solves a regression problem to impute missing data that are sent from IoT gateways to the cloud. In [29], a novel solution has been presented to estimate traffic measurements and select the most rewarding flows. Considering an SDN scenario and the specific problem of partitioning Ternary Content Addressable Memory (TCAM) entries of switches, this solution exploits Compressed Sensing (CS) inference methods. A Multi-Armed Bandit (MAB) based algorithm, then, can adaptively measure the most rewarding flows. More recent deep learning-based algorithms, e.g., RNN, LSTM, ConvLSTM, have been presented in [18]–[20], respectively, to solve the traffic matrix estimation process. While the design of these solutions is sound, our H&S does not require any additional information, e.g., link load, and is independent of specific matrix assumptions, e.g., the assumption of a low-rank matrix.

III. Problem Definition

A traffic matrix, *i.e.*, a matrix reporting the traffic volumes between origin and destination in a network, has a potential utility for network capacity planning and management operations [30], [31]. In large operational IP networks, however, traffic matrices are often hard to measure directly, and it is thus required to complete the matrix in the unknown cells.

In this paper, we analyze the problem of Traffic Matrix (TM) estimation from two different perspectives: completing

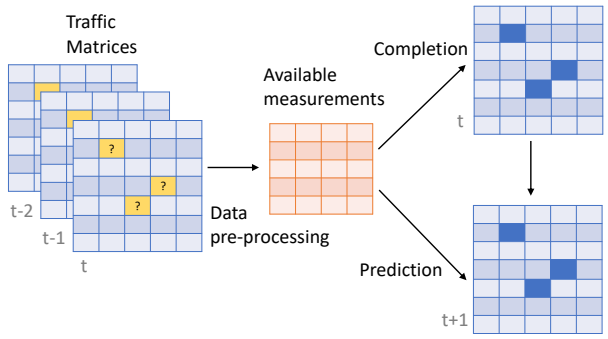


Fig. 1: System overview: we first identify the missing traffic matrix elements, then we extract the information required by our prediction model and finally, we restore the traffic matrix values and predict future traffic demands.

the matrix and predicting future values. Although they differ in model resolution, both problems, TM completion and prediction, start with a partially-observed traffic matrix. Since these two tasks share the same system model, we first describe the details of our model and then we formally define the two problems addressed by *Hide & Seek*.

A. System Model

In our model, we consider a network with n nodes, and we denote with Ω the non-empty set of all sources and destinations in a network, where $|\Omega| = n$. Hence, the resulting Traffic Matrix (TM) at time t is an $n \times n$ square matrix, whose element represents the number of bytes sent from node i to node j during the considered measurement interval.

We then consider the evolution of the sampling time, turning the TM into a 3-dimensional array, $Q \in \mathbb{R}^{n \times n \times m}$, where n is the cardinality of the nodes, and there are m time intervals. Its entry $Q(i,j,t)$ denotes the traffic bytes at time t, *i.e.*, in the measurement interval $[t-1, t)$, of the origin-destination pair $i-j$ where $i = 1, \dots n$, $j = 1, 2 \dots n$, $t = 1, 2 \dots m$. Therefore, the entries $Q(i,j,:)$ represent the variation in the number of traffic bytes along with the time for the Origin and Destination (OD) pair (i,j). We model the TM entries as continuous values, as we consider this assumption appropriate for most traffic volumes [15].

In addition, we use a binary matrix $Z \in \mathbb{R}^{n \times n \times m}$ to indicate whether entries of the TM Q are missing, defined as follows:

$$Z(i,j,t) = \begin{cases} 0 & \text{if } Q(i,j,t) \text{ is missing,} \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

Consequently, the observed measurement matrix $R \in \mathbb{R}^{n \times n \times m}$, which denotes the set of information that is available, is obtained as:

$$R = Z \cdot Q, \tag{2}$$

where $\cdot$ represents the scalar product of two matrices, *i.e.*, $R(i,j,t) = Z(i,j,t) \cdot Q(i,j,t)$.

B. Solving the Matrix Completion and Inference Problems

Based on the foundations of the aforementioned traffic model, we define two different yet related problems, namely *completion* and *prediction*, where the latter is also referred to as inference. The *completion* problem for traffic matrices is defined as follows. Let Q_t be the actual TM at time t, R_t the observed TM, and D the set with all the entry points of R_t whose measurement is missing. In *matrix completion*, a mapping function F_1 must be found to constitute Q_t given as input R_t and the set D. Conversely to other similar studies [12], [27], in this paper, we do not limit this mapping function to be linear, but we instead consider a model based on neural networks, as described in Section IV-C.

Then, we define the TM *prediction* or *inference* problem as follows. Given R_t as the observed TM at time t and D as the set with all the entry points of R_t whose measurement is missing, in *matrix inference*, a mapping function F_2 must be found to reconstruct the actual TM at time $t+1$, Q_{t+1}, given as input R_t and D.

In Fig. 1 we visualize the main steps of our solution along with the defined notation. After pre-processing the information from the collected TM, we solve the two problems. Although the matrix prediction can potentially regard the entire TM, in this paper we limit our attention to matrix entries whose historical information is only partially available. Although a large amount of literature has already addressed the problem of predicting Internet traffic with excellent results [3], [32]–[34], these solutions make predictions without considering missing data. Therefore, we present a solution that is orthogonal to these traffic prediction methods and that can be used in conjunction with them to optimize network planning and management.

From traffic theory, we know that some spatial and temporal properties in the traffic matrices exist [8], [9], [27], and therefore our model should guess missing values by leveraging these similarities. More specifically, *spatial properties* of the TMs refer to the statistical properties between the TM entries at a fixed t, *i.e.*, a snapshot $Q(:,:,t)$, while *temporal properties* refer to the statistical properties when varying with t, either with fixed spatial indices, *i.e.*, the process $Q(i,j,:)$, or some summary such as the total traffic $S(t) = \sum_{i,j} Q(i,j,t)$ at each time point t. We empirically evaluate how the spatial correlation of TM cells (not necessarily geographical correlation) can be exploited to reconstruct missing information (Section V-G). In *Hide & Seek*, we rely on these assumptions to estimate the missing entries of TMs, and we employ a general Hidden Markov Model to understand traffic models and traffic characteristics. Such a model is then opportunely empowered with adversarial autoencoders to improve the resolution of both completion and inference problems, as detailed in the next section.

IV. Predictive Model Design

This section describes the model used to estimate the missing values within a traffic matrix. Our estimator consists of an HMM model that dictates the evolution over time of traffic values and an autoencoder used to improve the performance of the HMM. *We show that performance of traditional HMM can improve by making use of adversarial autoencoder as an alternative method for the decoding problem.* We start by

describing the Hidden Markov Model (HMM) framework and its parameters, with particular focus on the decoding problem. Then, we overview how the autoencoder is applied in the general HMM model; we conclude detailing how we employ the obtained model throughout the estimation process.

A. Hidden Markov Model Framework

Given their ability to capture important traffic statistical characteristics with only a relatively small number of states, Hidden Markov Models have received much attention for traffic models [35], [36]. These studies have examined the effectiveness of HMM in modeling the packet flow generated by an individual application or the aggregate traffic on a single channel. Alongside, [37], [38] suggest that an HMM model may be effective in capturing the dynamic behavior of losses and delays on end-to-end communication channels. In the wake of this analysis, we model the traffic data exchanged by a pair of nodes by means of the HMM.

Hidden Markov Models are widely used time invariant state-space models defined as follows:

$$p(X,Y) = \pi(x_0)\prod_{i=0}^{T} p(y_i \,|\, x_i)\prod_{i=0}^{T-1} p(x_{i+1} \,|\, x_i), \qquad (3)$$

where x_i is the *hidden* variable and y_i is the *observed* variable, $p(x_{i+1} \mid x_i)$ is the transition probability describing the dynamic behavior of the system, and $p(y_i \mid x_i)$ represents the emission probability describing how the system generates the observation based on the hidden variable. To start the process, the model needs an initial state distribution, *i.e.*, $\pi(x_0)$.

The main assumption in HMM is that the state evolves as a Markov process, in which the probability distribution of the current state depends only on the state of the previous epoch. In other words, $p(x_i \,|\, x_{i-1}, ..., x_1) = p(x_i \,|\, x_{i-1})$. It has been shown that this first-order Markov process is sufficient for modeling the temporal properties of networks [38], [39].

The state evolution over time is commonly described using a transition probability matrix (PM), containing all the transition probabilities $p(x_{i+1} \,|\, x_i), \forall x_i \in \chi$. It should be noted that, despite the similarity, the PM matrix is completely different from the traffic matrix (TM) that we consider in our model, as the TM gives the number of bytes transmitted between a source and a destination, while the PM contains the transition probabilities between the states of the Hidden Markov Model. A typical way to represent HMM is as $\lambda = (A, B, \pi)$, where A denotes the transition probability matrix, B refers to the emission probability matrix, and π is the vector of initial states probabilities.

PM, $\pi(x_0)$, and $p(y_i \,|\, x_i = j)$ are generally unknown, so we need to estimate them either using some parametric or data-driven approaches. In fact, HMMs are characterized by three basic problems: *training*, *likelihood*, and *decoding*. The first problem, *training*, common to other ML algorithms, is formally defined as: Given the observation sequence in time Y, finding the model $\lambda = (A, B, \pi)$ that maximizes the probability of Y. The training problem is crucial for any HMM applications, to find model parameters adapting to the training observation sequence. The standard solution for this problem is the *Baum-Welch* algorithm (forward-backward algorithm), which is an instance of a general family of expectation-maximization (EM) algorithms [40]. In such an algorithm, there are two main steps: the E-step, which computes the probabilities of being at state s at time t; and the M-step, which fixes the model parameters maximizing the likelihood of posteriors found in the E-step.

The second problem, the *likelihood*, can be described as follows: Given the observation sequence over time Y and the HMM model λ, determine the likelihood $P(Y|\lambda)$, *i.e.*, the probability that the observed sequence was produced by the model λ. This problem can be solved via the recursive forward algorithm, responsible for computing the joint probability of observing the sequence up to time t and the Markov process being in state s_t. These probabilities are then used to obtain the likelihood values $P(Y|\lambda)$.

Similarly, the *decoding* phase attempts to find the most likely hidden state sequence X given the observation sequence over time Y and the HMM model λ. This problem is usually solved by means of the *Viterbi* [21] algorithm for hidden state estimation, which uses a dynamic programming approach in order to maximize the likelihood of the whole generating state sequence. In a first step, it gets the most likely state s_t at time t through a γ_t parameter. In a second step, the γ parameter can be calculated using the forward-backward method. Namely, the problem of finding the most likely state sequence can be summarized as follows: given a sequence of observed values $(\widetilde{y}_0, \widetilde{y}_1, ..., \widetilde{y}_n)$, we would like to infer the corresponding hidden variable $\widetilde{x}_t$, *i.e.*,

$$\widetilde{x}_t \sim p(x_t \,|\, \widetilde{y}_t, ..., \widetilde{y}_0). \qquad (4)$$

In H&S we design to replace the traditional Viterbi algorithm with an AAE, described in the following. Given its ability to encode and decode information between different spaces, we observe how this learner can be helpful and effective in empowering HMM (as demonstrated by results in Section V). Combining a recent data-driven algorithm (AAE) with the statistical approach imposed by HMM leads to an optimized *decoding* procedure.

B. Learning with Adversarial Autoencoder

Adversarial AutoEncoder (AAE) has been firstly presented in [22] as a model that can turn an autoencoder into a generative model. Following the more general approach of generative adversarial networks (GAN), AAE can perform variational inference by matching the aggregated posterior of the hidden code vector of the autoencoder with an arbitrary prior distribution.

In recent years GAN has been at the basis of a variety of alterations, giving rise to a large number of GAN-based models, such as CycleGAN [41], BiGAN [42], Super-Resolution GAN [43], to cite a few. These variants are generally applied to bits of an image, but any model consists of at least two neural networks: a generator and a discriminator. The former network receives as input a vector of randomly generated noise and produces as output an "imitated" image that looks similar, if not identical, to the authentic image. The latter network

attempts to determine whether a given image is "authentic" or "fake". Similarly, in AAE, an autoencoder is trained with two objectives—a reconstruction error criterion (typical of autoencoders) and an adversarial training criterion (typical of GAN). AAE differs from the traditional autoencoder, *i.e.*, the Variational autoencoder (VAE), in that it has a discriminator and a latent space as well as a different training procedure.

GAN-based models are often used to capture rich distributions such as audio, images, or video, and to generate a synthetic version. Following the recent and mostly unexplored trend of applying these models in different domains [44], [45], we consider this class of problems for Markovian environments. Specifically, as suggested in [45], we use the autoencoder to map some of the available entries of the TM (observed state of HMM) to the missing values (hidden state of HMM).

After a necessary training phase, the encoder of AAE learns to convert input data to an intermediate representation (latent space), while the decoder learns a deep generative model that maps this representation to a posterior data distribution (final output). The adversarial network (generator + discriminator) guides this matching. The generator is also an encoder that draws samples aiming to create an aggregated posterior distribution that can fool the discriminator network into believing that the latent space comes from the true prior distribution. The autoencoder (encoder + decoder), meanwhile, attempts to minimize the reconstruction error so that once the training procedure is done, the autoencoder is able to map the imposed input to the desired data distribution. The input data of our scenario represents the traffic data with partial information, R, while the output data is the complete traffic matrix with reconstructed values, Q.

C. H&S Procedure

Referring to the previous notations of HMM (Section IV-A), we can now define variables' meaning in our system and map them to the HMM notations. For each missing value at time t we define the current evidence y_t as a square matrix with dimension $k \times k$. Such submatrix represents the elements surrounding the missing value, located at (i, j), which we are interested in estimating. The hidden state value x_t is instead the missing entry of TM at position (i, j). While modeling a single hidden value is a standard procedure in HMM-based algorithms [21], there is a tradeoff in choosing the best matrix size k. Such submatrix dimension k affects the cardinality of the observed states y_t and is a crucial parameter that must be specified when designing the model. On the one hand, a smaller size implies a simpler model but may yield an inadequate representation of the space of possible behaviors, accounting for insufficient spatial similarities. On the other hand, a large k leads to a more complex model with more parameters but may, in turn, lead to overfitting. In our validation, we used a 7×7 submatrix, which was the result of a cross-validation study and was highlighted by the saliency maps that showed the importance of neighboring cells (Section V-G).

Pre-processing. As a best practice, before using the traffic quantities in our model, or any ML model, they must be prepared. Data preparation includes the use of normalization or other standardization techniques to re-scale input and output variables before training the ML model. Differences in the scales of the input variables may increase the difficulty of the problem being modeled [46]. For this reason, we apply a standard normalization approach to scale the input values in a range $[1 - 10]$, making the model more general and transferable to different scenarios. Although input is traditionally normalized over the interval $[0 - 1]$, we experienced how this smaller interval can not capture the traffic diversity, leading to a higher reconstruction error. Since our problem belongs to the family of regression, a small range can simplify the training but may lead to considerable differences when de-normalizing the output values. This effect was particularly manifested when data distributions were less uniform and the spectrum of values was broad. Along with the normalization of input, we perform an edge padding operation. As explained in previous sections, to estimate the missing values, our solution is based on the adjacent normal data to obtain the close submatrix. However, specific locations, *i.e.*, the edges of the TM, could have an insufficient number of available adjacent data. To solve this problem, the most direct and effective method is to arrange some additional data on the edges. In particular, we apply the notion of *circularity* so that the left edge and the right edge of the matrix appear "adjacent", as well as for the top and down rows. Thus, we use data of the matrix itself as padding data. Lastly, to inform the model of missing values, we apply the concept of masking. Masking consists of marking the locations of the input space to be ignored with an identifiable value, for example -1. The AAE model, then, always expects the same number of inputs ($k \times k$) but can distinguish between measured and missing values. This procedure can be easily generalized when there exist multiple missing values to estimate, so we mark all those values that the neural network must neglect for the learning process.

HMM parameters. Although we set the number of hidden states to 1, we must define how to model the evolution of this value, reflected by the transition probability matrix (PM). This design is made more difficult since we are dealing with continuous values for the bytes of traffic, and we are also interested in predicting future values when an entry is missing. For this reason, we decide to use an approach where the computation of the future value x_{t+1} is equivalent to estimating the difference with respect to the last hidden state x_t. Since we have normalized values, this difference resides within the interval $[-9.9, 9.9]$. Besides, since the PM must be limited, we only consider 100 possible values inside this interval so that the prediction is accurate, but the problem is treatable. A higher number would increase the dimensions of the PM, making the model intractable. In other words, we compute the probability over a discrete set of possible evolutions rather than predicting the future traffic directly. We referred to this set as E, and the value at time t is e_t. Hence, the PM reports the probabilities that the next hidden state is obtained by adding e_t to the current state, where considered traffic values have been opportunely normalized.

Matrix completion. In view of the foregoing, the *traffic matrix completion* task in H&S is equivalent to the decoding problem

of HMM, where the objective is to find the value of a missing entry (*hidden state*), given as input the adjacent submatrix (*observed matrix*). Using the above notation, the problem is as follows: Given the observation y_t at time t, the traffic matrix is completed by finding the hidden x_t with the highest probability. By using AAE to learn this mapping function, H&S is also able to deal with non-linear relationships, which allows generalization of our model over a broader range of traffic conditions.

Future traffic prediction. Along with the matrix completion problem at time t, it may be necessary to predict the future state at time $t+1$. This case is equivalent to the HMM task of computing posterior distribution over the future states given the current TM and the past evolution. The predicted next hidden state is derived as the one with the highest probability, according to:

$$\begin{aligned} e_t &= \arg\max_{e_t} p(e_t \mid \widetilde{y}_t, ..., \widetilde{y}_0), \\ \widetilde{x}_{t+1} &= \widetilde{x}_t + e_t. \end{aligned} \tag{5}$$

Since in our HMM model we consider the evolution of the hidden states as a difference from the previous step, we must first determine this difference, e_t, and then add this value to the traffic value at time t, *i.e.*, x_t. We can summarize the complete *traffic matrix prediction* procedure as follows: Given the sequence of observations at time t, we decode them into the hidden state using AAE and predict the next hidden variable using maximization of posterior probabilities. Finally, in this paper, we are interested in one-step ahead prediction, since predicting for more steps ahead degrades the accuracy, especially for HMMs, as suggested by other studies [47], [48].

V. Evaluation Results

In this section we quantify the benefits brought by our *Hide & Seek* algorithm in the resolution of the traffic matrix completion and prediction problem. Thus, separately for these tasks, we present the results of experiments performed to assess the effectiveness of the proposed approach.

A. Experimental Settings

Implementation. We implemented our AAE agent as an application importing the `Keras` library [49], while the HMM model is built upon the `hmmlearn` library [50]. We will release our source code with an open-source license upon manuscript acceptance.

Internet traffic traces. In our trace-driven evaluation we used three publicly available datasets. The first was the GEANT [51] dataset, consisting of 11460 traffic matrices built using full routing information of 23 routers, sampled Netflow data, and routing information of the European GEANT network, with a sampling interval of 15 minutes and duration of one week. The second set of traces was imported from the Abilene traffic matrix dataset [52], a backbone network consisting of 11 nodes of major cities in the USA. In this case, we used one week of traffic collected with a granularity of 10 minutes for a total of 48386 TMs. The third was captured within the WIDE backbone network that connects Japanese universities and research institutes to the Internet, whose collection was made publicly available by the MAWI group [53]. This archive is an ongoing collection of Internet traffic traces, but we considered ten consecutive traces dated 2020, spanning over two hours and thirty minutes from samplepoint-F. The result is a collection of 9010 24×24 matrices obtained by aggregating by IP address prefix, with a granularity of 1 second, and by filtering smaller flows to keep the matrices' size at a reasonable level. While GEANT and Abilene constitute two of the most used datasets in this field, our MAWI dataset is a more recent collection that, being sparse and highly varying, exposes different patterns of traffic to learn during the estimation process.

Benchmark algorithms. To demonstrate the effectiveness of our proposed H&S for matrix completion, we compare its performance with the following algorithms, opportunely adapted to our context. First, *CCAE*, an algorithm that transforms the recovery problem to images inpainting [54], a computer vision technique used to reconstruct missing segments in images. In [55], the inpainting method reconstructs the missing values using cascaded convolutional autoencoders, where matrices are regarded as "generalized" images. In the paper, it has been shown that inpainting enhances the robustness even in extreme conditions, *i.e.*, several missing values. We adapt it to our network traffic matrix context.

The second benchmark algorithm is the *Spatio-Temporal Tensor Completion* method, or *STTC* [27]. This method models network traffic as a tensor pattern, projecting tensors into a lower-dimensional latent space via tensor factorization, while preserving the multi-way nature of the network traffic data. The method, then, exploits the multidimensional structure correlation properties of tensors to estimate the missing entries. Tensor-based interpolation methods have been shown to capture more global information than matrix-based methods due to the intrinsic multidimensional characteristics of the tensor model [56], [57].

Third, we compare H&S with *LMaFit*, the Low-rank Matrix Fitting (LMaFit) algorithm [16]. This predictor solves a low-rank factorization model for matrix completion by applying a successive nonlinear over-relaxation. It is one of the most commonly used methods since it can be applied in a wide range of matrix completion or low-rank approximation problems. Forth, we test H&S against the classical k nearest neighbors or *kNN*, where we assume that the missing values of the TM are predicted by local interpolation of the targets associated to the nearest k neighbors [58]. Based on the similarity between rows and columns, we use a weighted average with $k = 7$. Fifth, we compare against *ConvLTSM*, a recent solution that integrates a Convolutional Neural Network (CNN) model and a Long Short-Term Memory (LSTM) network for spatiotemporal modeling and estimating the future network traffic [19]. This solution exploits a backward network to process the input and correct the (usually inaccurate) previously predicted data before feeding it into the predictive model.

Alongside, we consider the efficacy in traffic values forecasting and, to this end, we compare our estimator against two other well-known regressor algorithms. *(i) ARIMA*, a time-series approach that can model the evolution of data over time.

TABLE I: Time required for the different solutions to be trained, complete, and predict the matrix over the three datasets.

	GEANT			**Abilene**			**MAWI**		
	Training [s]	*Compl. [ms]*	*Pred. [ms]*	*Training [s]*	*Compl. [ms]*	*Pred. [ms]*	*Training [s]*	*Compl. [ms]*	*Pred.[ms]*
H&S	4857.32	1.3671e-01	2.7334e02	6669.86	7.5961e-02	1.1691e02	1673.19	4.4790e-01	1.5564e02
CCAE	9513.83	2.8152e-01	—	8545.44	1.1139	—	12150.23	7.4157	—
kNN	0.0030	1.2067e-01	—	0.0034	2.7166e-01	—	0.0045	2.7014e-01	—
STTC	—	2.4552e02	4.2130e02	—	3.1745e02	6.7511e02	—	2.0548e02	4.5784e02
LMaFit	—	2.9710e-01	—	—	5.7675e-01	—	—	7.3232	—
ARIMA	3925.16	—	6.6097e01	6228.44	—	6.2012e01	2297.80	—	4.3232e01
ConvLSTM	1522.94	—	8.9529e-01	2259.75	—	2.5503e-01	2272.37	—	1.8944
RFR	3032.49	—	5.2943e-01	7617.70	—	4.5876e-01	864.77	—	2.7903e-01

ARIMA is typically used to represent stationary time series in almost all domains where a variable is measured at equidistant times, such as in financial market data. *(ii) RFR*, Random Forest Regression (RFR) is an additive model that predicts by combining decisions from a sequence of base models, typically a simple decision tree. This broad technique of using multiple models to obtain better predictive performance is also known as model ensembling and provides robustness to RFR.

B. Training and Prediction Time

We report in Table I the time required for all the methods implemented to train the different traces and to complete and predict the single TM. For LMaFit and STTC, we do not report the training time since they are two statistical approaches where the former only needs to set rank k, while the latter the ρ parameter. We set both of them via cross-validation. In kNN, the training is not to learn parameters but rather to create the appropriate data structure to be used for the search, *i.e.*, completion process. CCAE and LMaFit are used only in completion, and the prediction time is not reported because not measurable. Similarly, the completion time is absent for ARIMA, ConvLSTM, and RFR since these methods are used to predict future values. First, we can observe that our AAE-based method is faster than the other AE-based method, CCAE. This time is also compatible with the other methods and depends on the size of the training set and of the matrix. Therefore, we can see how training on Abilene is generally longer than others, given its major dimensionality.

Second, in the completion and prediction process, the time of our solution is line with benchmarks and in the order of milliseconds. This outcome makes H&S available to be used in real-time systems where TM is updated frequently, and the estimation cannot last long enough. We analyze this behavior in detail in a prototype later in Section V-F.

C. Matrix Completion Performance on Random Loss Patterns

In this subsection, we study the performance of our traffic matrix completion algorithm when varying the amount of known information. To this end, we hide data points independently at random to evaluate the completion performance. The missing values range from 1 to 80 percent of the total entries.

Starting with the Abilene network dataset, we compare our solution against a similar approach also based on autoencoders, as in the CCAE solution, and the traditional version of HMM based on the Viterbi decoding algorithm. We quantify the Mean Absolute Error (MAE), because this metric can deliver the order of magnitude of error, along with 90% of confidence intervals, and report the results in Fig. 2a. It can be observed that our AAE-based method for completing the traffic matrix outperforms the benchmark CCAE. The adversarial training criterion of our AAE-based method is particularly effective in strengthening the traditional reconstruction process and filling the missing traffic matrix cells. In particular, our approach can handle a significant percentage of missing entries, conversely to CCAE. We have experienced how the performance of CCAE largely depends on the position of the missing entries in the matrix, and that its masking model poorly scales when the majority of the elements is unknown. Moreover, we can observe how traditional HMM hardly manages the missing entries, and a few missing cells hinder the learning process. The traditional statistical algorithm of Viterbi barely tolerates the absence of input data, rapidly raising the MAE error for all three datasets (Fig. 2a, Fig. 3a, Fig. 4a). On the contrary, a data-driven approach as AAE can better learn the correlations among input data and efficiently reconstruct the missing data. Given this observation, in the following, we ignore the traditional HMM procedure and focus on other state-of-the-art algorithms.

To validate this result, we then consider all other benchmark algorithms for matrix completion, reporting the results in Fig. 2b. The MAE error of H&S is the lowest among all the percentages of missing entries. The error achieved is also marginal considering the traffic volumes present in the matrices. This outcome is particularly important because it suggests that this technique can be used in real-world deployments to take network decisions even when the available information is incomplete.

Besides these numerical values completion tasks, we also analyze the accuracy when these values are then divided into classes. In this case, we are interested in a discretized classification that entails five value classes. More specifically, using a bin discretizer, the continuous values are binned into intervals by means of a quantile strategy, *i.e.*, the bins have a similar population. Such a classification is important as it simplifies the problem while providing information that is still important in network measurements [59], [60]. Fig. 2c displays the accuracy score in such a multi-class classification problem for a subset of approaches. While we note how our H&S solution outperforms the benchmark, it can also be observed how the advantages are more notable compared to the previous graph. Nonetheless, this high accuracy is due to the limited

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3270166

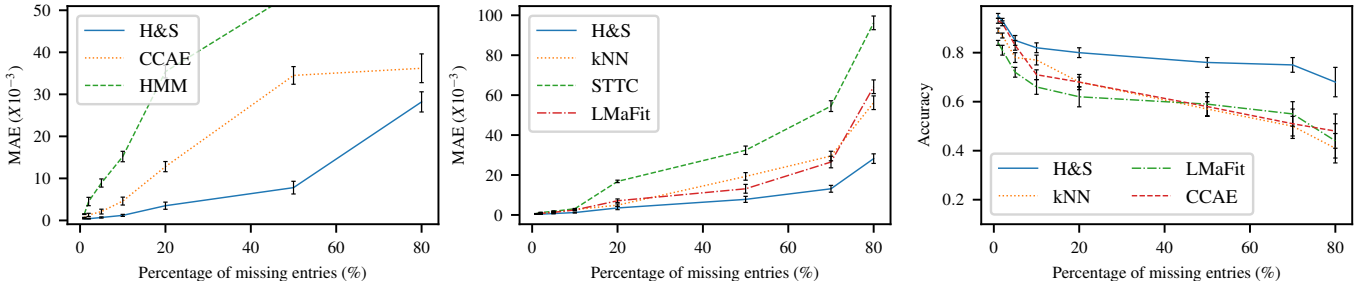


Fig. 2: **Abilene network.** (a) MAE error for autoencoders methods and (b) other benchmark solutions in completing the traffic matrix. (c) Accuracy for indicating the class of missing entries in the matrix.

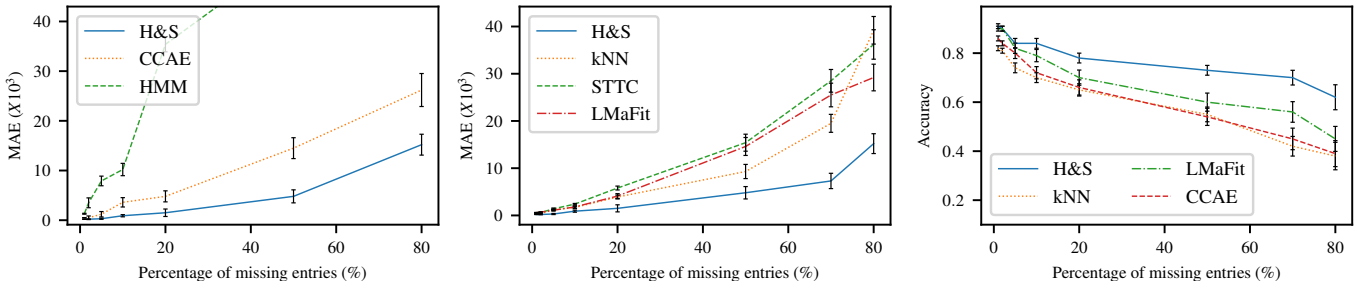


Fig. 3: **GEANT network.** (a) MAE error for autoencoders methods and (b) other benchmark solutions in completing the traffic matrix. (c) Error in terms of accuracy (the higher the better) in completing the matrix by indicating the class.

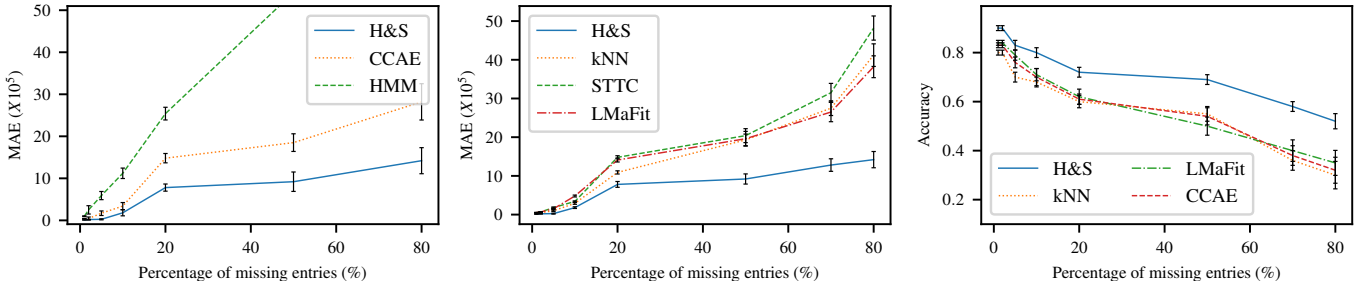


Fig. 4: **MAWI network.** (a) MAE error for autoencoders methods and (b) other benchmark solutions in completing the traffic matrix. (c) Error in terms of accuracy (the higher the better) in completing the matrix by indicating the class.

error shown before, which causes the predicted value to fall into the same class as the original traffic measurement.

To generalize these findings, we perform the same set of experiments over the GEANT traffic data and compare the ability of the two different autoencoders to complete the traffic matrix (Fig. 3a). Although the error obtained for matrix completion is higher compared to the Abilene use case, due to the different order of magnitude of values themselves, our adversarial autoencoder method is still more effective than CCAE. Besides, H&S shows the same ability to handle a considerable amount of missing data.

Similar considerations are valid when comparing our model against other related benchmarks, as seen in Fig. 3b. Other solutions, such as kNN and LMaFit, are able to provide a limited error in either one or the other scenario, but not consistently. In particular, when the percentage of missing entries reaches 50%, the difference between ours and others becomes more apparent.

Moreover, when analyzing the accuracy for predicting the class of missing values (Fig. 3c), it is possible to observe how H&S achieves the highest accuracy score. These results confirm our hypothesis that a GAN -based model can learn even when there are no guarantees of specific properties between cells in the traffic matrix.

We then estimate the ability of H&S to complete the TM over the MAWI traffic dataset. In MAWI traffic, we have more sparse and highly varying values both in space and over time. This makes the cell estimation process more difficult because as the number of missing cells increases, so does the likelihood that information important in the reconstruction process will be lost. Consider for example Fig. 4a, reporting the MAE of the two autoencoder-based models. The presence

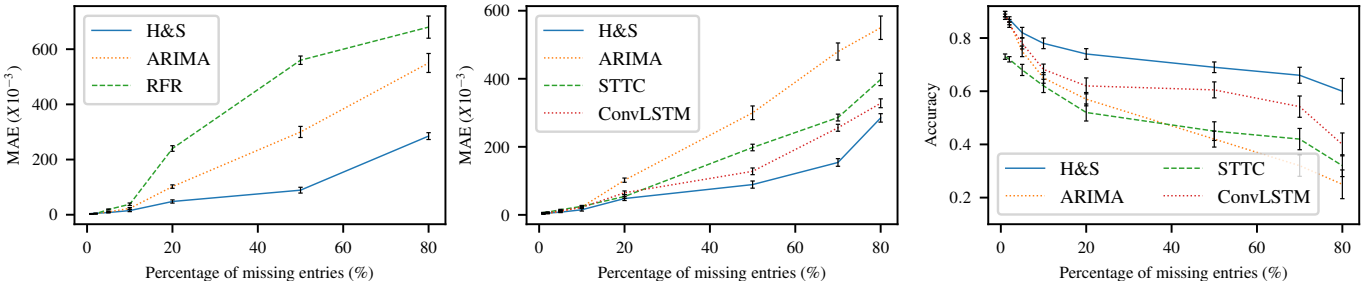


Fig. 5: **Abilene network.** (a) MAE error for regressor methods and (b) other benchmark solutions achieved in predicting the future missing values, *i.e.*, for time $t+1$. (c) Accuracy score for prediction task to specify the class of missing entries of traffic matrix.

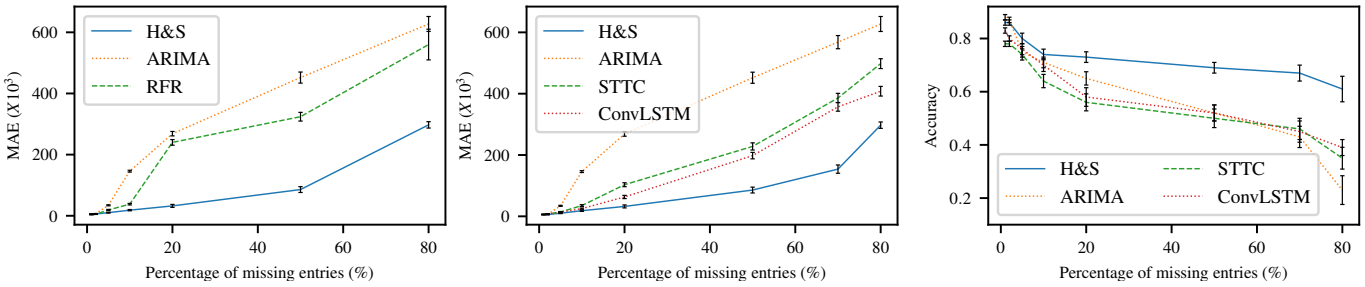


Fig. 6: **GEANT network.** (a) MAE error for regressor methods and (b) other benchmark solutions for future values prediction. (c) Accuracy in predicting future value class.

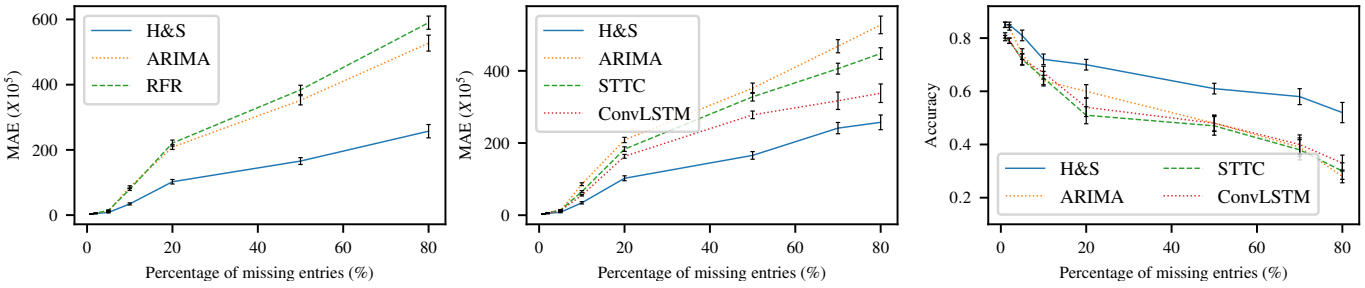


Fig. 7: **MAWI network.** (a) MAE error for regressor methods and (b) other benchmark solutions for future values prediction. (c) Accuracy in classifying future traffic.

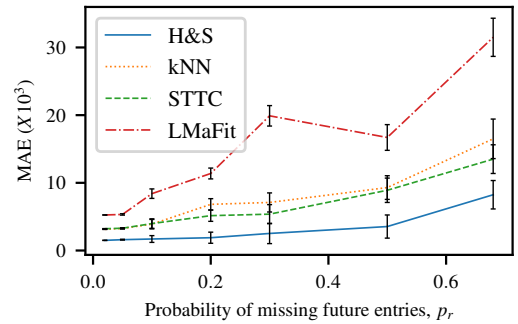
of 20% of missing entries leads to an abrupt increase in the error. However, despite these difficulties, H&S can achieve acceptable errors, lower than alternatives, as confirmed in Fig. 4b. If we consider the accuracy in Fig. 4c, it is clear that this value is lower than for the Abilene and GEANT traces due to the intrinsic absence of predictability in MAWI traffic.

D. Matrix Prediction Performance

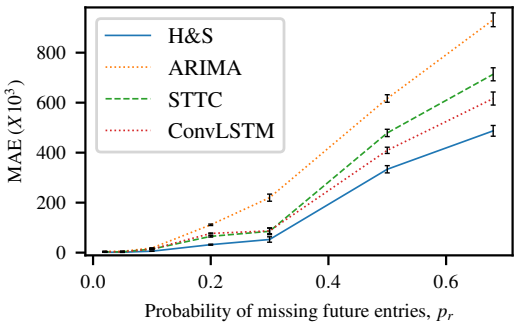
Similar to previous experimental settings, we now consider the capacity of *Hide & Seek* in predicting the future values of missing entries. As described in Section IV, not only is our solution able to complete the entries missing at timestamp t, but it can also predict their future evolution in subsequent timestamps. However, in light of the fact that the one-step ahead prediction is one of the most common scenarios (see Section IV), in the following, we limit our attention to this condition.

Starting with the Abilene network traffic, in Fig. 5a we show the MAE for H&S against the two forecasting algorithms, one belonging to the time-series class, *i.e.*, ARIMA, and one to the ML regression class, *i.e.*, RFR. While these two alternatives can well predict when the percentage of known entries is particularly high, they are clearly ineffective when the number of missing entries increases. These results motivate the need to define novel approaches in predicting traffic values given the limited visibility of the network or the impossibility of collecting all the metrics.

We hence consider more solutions to predicting future values. Among the previous benchmark algorithms, we consider the approaches that are suited for this task, and results are compared to ARIMA for a clear validation of the benefits. In this task of predicting values, for spatio-temporal matrix representations that consider multiple timestamps in only one matrix as STTC, we set as unknown all the values for future timestamps, *i.e.*, all columns whose index is greater than $t+1$



(a) Completion



(b) Prediction

Fig. 8: MAE error over the GEANT dataset and structural loss pattern for (a) the traffic matrix completion problem and (b) the prediction problem.

when the prediction occurs at time t. Fig. 5b reports the MAE error for our H&S, and the three alternatives. First, it can be noted that prediction leads to higher errors compared to the previous completion task, given the limited visibility of the dataset for future forecasting. Second, our HMM model can well mitigate the effect of data loss and, consequently, minimize the error for the whole shown percentage range of missing entries.

Moreover, we consider the accuracy when predicting the class of future values, reporting the score in Fig. 5c. Predicting the next class rather than the next real value limits the amount of information, but is still a key parameter, as it indicates whether the value is increasing or decreasing and approximately to what amount. It can be noted how the benefits brought by H&S are even more notable in this scenario w.r.t. the completion task. This outcome stems from the ability of HMM to appropriately model the evolution of traffic by quantifying the increment or decrement of the traffic itself, and from its interior design based on the concept of expectation and probabilities.

Furthermore, we examine the performance of the prediction task over the GEANT dataset. Starting from the comparison with the two regressors, *i.e.*, ARIMA and RFR, we can observe in Fig. 6a how similar conclusions to the Abilene network hold. Besides, even for a high percentage of data loss, around 50%, our method can provide a very limited MAE, which then increases only when the percentage hits 80%.

Consider then the MAE for the other traffic matrix prediction solutions shown in Fig. 6b. Although for 80% the error of H&S starts rising rapidly, it must be noted that this error is constantly modest compared to the benchmark algorithms. We can still observe how ARIMA can well predict the next class, but, given its forecasting nature, it is unable to handle missing values. H&S, conversely, can consistently provide high accuracy. Similarly, by looking at the accuracy in predicting the next class (Fig. 6c), we can confirm the efficacy of H&S in not only regression problems, but also in the broader task of future value evolution.

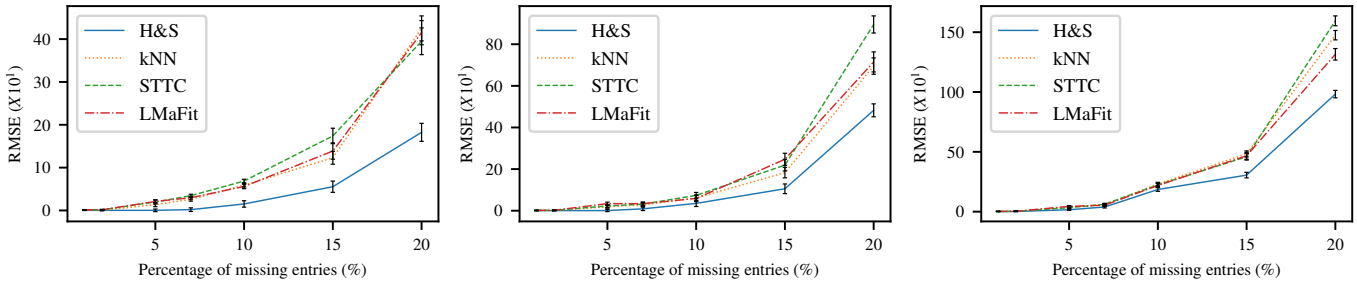
In addition, we perform the prediction process over the MAWI dataset and report results in Fig. 7. Starting from the comparison with other regressors (Fig. 7a), passing from the comparison with other matrix values predictors (Fig. 7b), and concluding with the accuracy evaluation (Fig.7c), we have conclusions similar to previous traces. Among the regressors, ARIMA is more precise than RFR in this case because of the variability of the traffic. Besides, the presence of missing entries causes performance degradation from around 20%. However, H&S can learn traffic patterns and mitigate the effects of such data loss. The accuracy obtained for benchmarks (Fig.7c), decreases quickly for significant missing portions of TM. We can thus conclude that our approach is stable in providing excellent results among different conditions, e.g., in the percentage of missing entries and dataset.

E. Structural Loss Patterns

In this subsection, we carry out simulation experiments on structural loss patterns. In practice, not all data loss is random, and network traffic shows high structure loss due to software or hardware reasons. We simulate one particular case of structural loss pattern referred to as Synchronous Spatio-Temporal Loss (SSTL). This simulates a loss event in a set of ODs that undergoes synchronous loss because of systematic failure. To simulate this scenario, we randomly chose a certain proportion of OD whose statistics are lost over time with probability p_r. In particular, in our SSTL pattern, we re-create scatter holes by assigning a loss percentage probability of 50% to cells adjacent to a missing one, and cells are marked as lost until 25% of the TM is reached. Then, for these cells, the loss probability in subsequent time instants is chosen with probability p_r from 0.02 to 0.68.

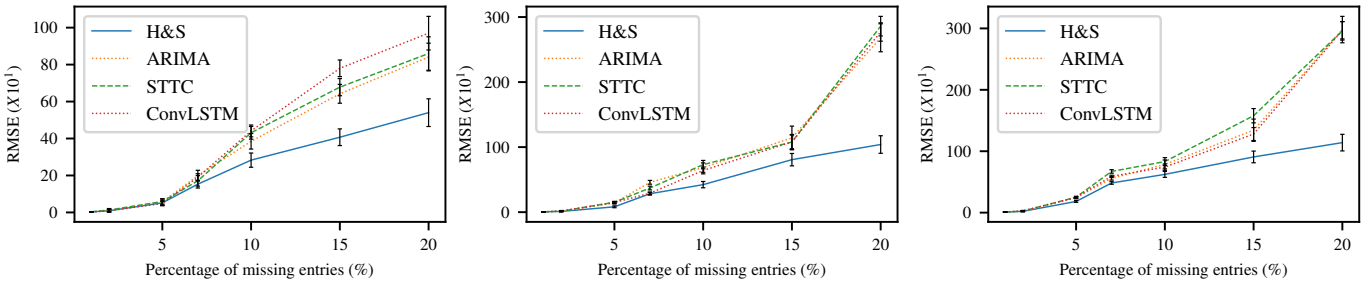
In Fig. 8 we quantify the error when the SSTL pattern is applied over GEANT dataset. In particular, Fig. 8a measures the error when completing the TM for some benchmarks. We can first observe that the performance of LMaFit is generally poor because the SSTL pattern does not meet the mathematical conditions of its matrix completion approach based on a low-rank structure. Compared to the random pattern, in this case, the error of other methods (and H&S) is more stable for p_r up to 0.5 and then increases for higher p_r. For the prediction task, instead, we can observe a higher error (Fig. 8b). When p_r exceeds 0.4, the error rises significantly. The fact that only some, but always the same, positions are missing makes the learning process more difficult. The problem is exacerbated for traditional algorithms such as ARIMA, which highly rely on past data. In these extreme cases, however, we can observe that H&S results the solution with the smallest error. In conclusion, even for structural losses, *Hide & Seek* shows ability to complete and predict a TM.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3270166



(a) net utilization = 10 % (b) net utilization = 30 % (c) net utilization = 50 %

Fig. 9: **Mininet prototype completion.** RMSE for predicting the missing entry when the network is utilized at (a) 10 %, (b) 30 %, (c) 50 %.



(a) net utilization = 10 % (b) net utilization = 30 % (c) net utilization = 50 %

Fig. 10: **Mininet prototype prediction.** RMSE for predicting the missing entry when the network is utilized at (a) 10 %, (b) 30 %, (c) 50 %.

F. Real-time traffic prediction

In addition to trace-based evaluation, we now investigate how effective our solution is when running in real time. To this end, we deployed it over an SDN emulator, *i.e.*, Mininet [61], where switches interact with a centralized controller, implemented in our prototype with Ryu [62]. While other solutions are available for implementing the control plane logic, e.g., Floodlight, ONOS, and OpenDayLight, we chose the Ryu language, given the easiness of prototyping and of collecting switch/flow information with simple function calls. In addition, Ryu is developed in Python, facilitating the integration with H&S that was developed in Python as well, which would have been more though with the other Java-based frameworks. We set up a network with 10 hosts, which leads to a 10×10 matrix. Since we are now interested in assessing performance in real-time rather than over realistic traffic (analyzed with trace-based experiments), we instruct the hosts to send traffic randomly to another host present in the network for a period of time chosen uniformly between 1 and 20 seconds. At the end of this period, the host randomly selects a new host again and draws the transmission time in the same way. We repeat the process until the SDN controller has collected 5000 matrices, sampled every 5-seconds. First, we pre-trained the model over the entire dataset; then, we ran the real-time completion and prediction of TM over our controller when the hosts communicate in the same way but with different destinations and transmission times (achieved by changing the seed).

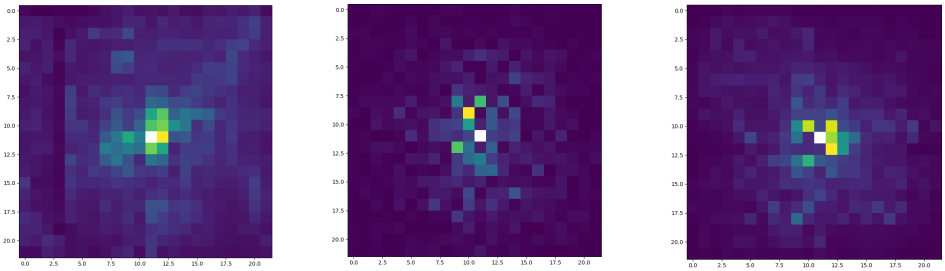
We start analyzing the training, completion, and prediction time in Table II. Given the minor number of samples and minor TM size, the training time is generally lower than in trace-driven experiments (Table I). These results induce us to design an offline training process for all ML methods. However, we can still observe how the completion and prediction time of H&S is in the order of milliseconds and in line with alternatives. This confirms our hypothesis that *Hide & Seek* can find applicability in a variety of network management applications, *e.g.*, re-routing, network planning [3], [10]. For example, in SDN architecture, where routing can be adaptive, it can be convenient to *predict* future traffic value and steer the traffic opportunely. Whereas, in flow consolidation, it can be convenient to *complete* the TM and have a global view of the current network status and take the more appropriate response.

TABLE II: Time required for the different solutions to complete and predict the matrix.

	Training [s]	Completion [ms]	Prediction [ms]
H&S	731.56	1.54e-01	2.75e01
CCAE	877.20	4.87e-01	—
kNN	0.0028	1.69e-01	—
STTC	—	2.86e02	3.44e02
LMaFit	—	5.34e-01	—
ARIMA	832.91	—	1.32
ConvLSTM	604.18	—	6.21e-01

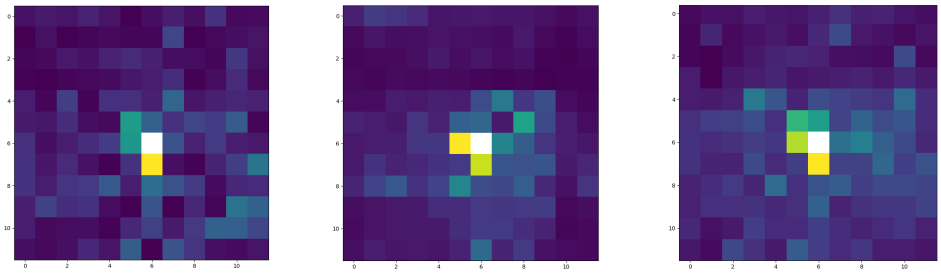
We then show the Root Mean Square Error (RMSE) of TM completion, to enrich the global solution evaluation, at varying the volume of traffic in the network in Fig. 9. We found confirmation to previous results: H&S has the ability to complete the missing entries of TM for different loads when running in real-time over synthetically generated traffic.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3270166



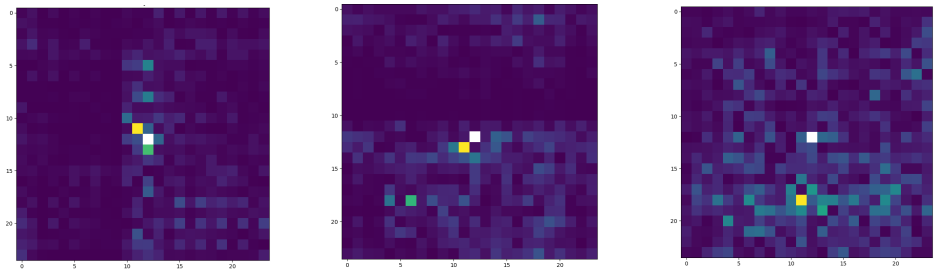
(a) (11, 7) (b) (12, 18) (c) (1, 3)

Fig. 11: Saliency maps for GEANT traces for different missing positions in the traffic matrix. Missing location is marked in white and centered in the final image.



(a) (6, 8) (b) (2, 8) (c) (3, 4)

Fig. 12: Saliency maps for Abilene traces for different missing positions in the traffic matrix. Missing location is marked in white and centered in the final image.



(a) (11, 2) (b) (1, 8) (c) (6, 7)

Fig. 13: Saliency maps for MAWI traces for different missing positions in the traffic matrix. Missing location is marked in white and centered in the final image.

In Fig. 10 we show the RMSE in predicting future entry of TM. We can observe a similar behavior as in traces-based experiments, where the error slightly increases from the completion task. However, H&S still provides the lowest error among alternatives. In light of these results, we can finally conclude that our *Hide & Seek* algorithm is an effective approach to both traffic matrix completion and future traffic prediction.

G. Design Rationale: submatrix dimension

To obtain insights into how the input is used, with a particular focus on the spatial correlation of matrix cells, we use Saliency maps. Saliency maps, introduced in [63], are a way to visualize classification models' spatial support for a given class in an image. The idea behind saliency is to rank the influence of single pixels of an image over the score function (of a class for classifiers or value variation in our regression case). The saliency values for each pixel are computed by differentiating the score function of choice with respect to the input image. The result is a map that has the same size as the original image, where each cell constitutes the degree to which the corresponding pixel of the image is influential in defining the score value. Continuing with the analogy between an image and traffic matrix, we apply the same model opportunely adapted, where the matrix cell is equivalent to the image's pixel.

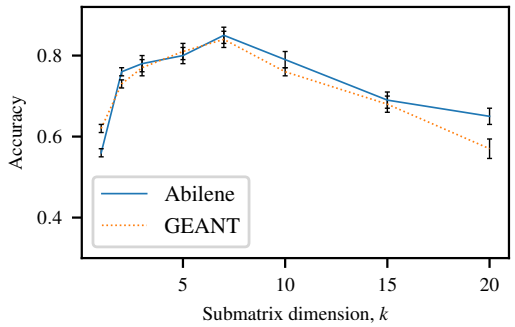
In Fig. 11 we show the saliency maps obtained by our method when completing the GEANT matrices, giving as input to our model the entire matrix. We consider four different missing locations, and we center the final results so as to have a clear view of the importance of the cell. The central element is marked in white for clarity, and colors in the blue-yellow range indicate importance, with importance increasing toward yellow. In other words, yellow means maximum importance, and blue means no importance. It can be observed that the most influential cells surround the missing element, with the green and yellow cells in close proximity. The same behavior can be observed in the Abilene dataset (Fig. 12) and the MAWI dataset (Fig. 13). While for some locations (12c, 13c) the cells dominating the estimation process are numerous, and

for others (11b, 12b, 13a, 13b) this number is limited, we can still observe how yellow cells are in proximity of the white cell. This result confirms our hypothesis that *although adjacent matrix cells are not necessarily topologically close and statistically dependent, our ML model can use neighboring cells to complete the matrix.* Moreover, this also induced us to wonder if all matrix cells must be used in the completion/prediction process or if it is convenient to discard part of the information for a lighter process. Since we limit the input to a $k \times k$ matrix as explained in Section IV-C, in what follows we provide more evidence on the rationale behind the dimension of the submatrix chosen in our algorithm.

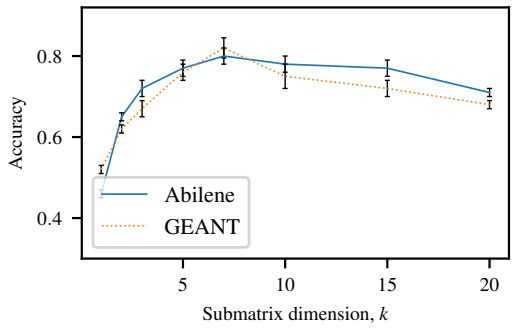
We study the performance of H&S when changing the dimension k of the submatrix used as observation (Section IV). This matrix is used as evidence to reconstruct missing information, and its dimension k is a crucial parameter obtained after an ablation analysis—an analysis of the performance of an AI system by removing certain components, to understand the contribution of the component to the overall system.

To this end, we compute the accuracy of our model when modifying the parameter k over the Abilene and GEANT datasets, and report the results for a percentage of missing entries to 5% in Fig. 14. First, for the matrix completion task (Fig. 14a) and then for traffic prediction (Fig.14b), we observe how a submatrix 7×7 allows to attain the highest accuracy score. This parameter allows us to consider enough knowledge and depart from two corner cases: limited ability to learn and to generalize the training (for small k), and a slow training phase (for large k). Clearly, the traffic submatrix dimension parameter should be adapted to different datasets and traffic matrices sizes, but our observed results motivate our choice of having set $k = 7$ as the default size for our evaluation.

We believe that this result is important not only for estimating traffic volumes, but also for developing future efficient network measurement techniques that can focus only on the most important cells. By knowing in advance the area of attention in the input, network telemetry systems can be designed to privilege the information that plays a critical role in the decision process, to the detriment of other unnecessary metrics. Therefore, this would also play a crucial role in the feasibility of such telemetry systems.



(a) Completion



(b) Prediction

Fig. 14: Accuracy analysis of increasing submatrix dimensions for (a) the traffic matrix completion problem and (b) the prediction problem. Both graphs suggest the default value of $k = 7$.

VI. Conclusion

In this paper, we presented *Hide & Seek*, a method to efficiently achieve traffic matrix completion and inference. *Hide & Seek* is based on a novel HMM-based approach in which the traditional encoding algorithm is replaced by an Adversarial AutoEncoder (AAE). We used our algorithm to estimate the missing entries in the traffic matrix and to predict their values in the short horizon. Our evaluation, performed over three publicly available real datasets, *i.e.*, obtained from Abilene, GEANT, and MAWI networks, validate the performance of our approach, highlighting the efficacy of AAE in computing the missing values starting from a limited set of information. Results also showed that our solution clearly outperforms the state-of-the-art, both in completing and predicting matrix values. We also tested the generality of *Hide & Seek* in an emulated prototype, showing how our implementation is practicable and efficient. Lastly, studying the decision process of our ML-based model, we observed how the spatial correlation hypothesis finds an empirical foundation.

References

[1] V. Bharti, P. Kankar, L. Setia, G. Gürsun, A. Lakhina, and M. Crovella, "Inferring invisible traffic," in *Proceedings of the 6th International Conference*, 2010, pp. 1–12.

[2] N. Ruchansky, M. Crovella, and E. Terzi, "Matrix completion with queries," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1025–1034.

[3] A. Sacco, F. Esposito, and G. Marchetto, "Rope: An architecture for adaptive data-driven routing prediction at the edge," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 986–999, 2020.

[4] A. Sacco, F. Esposito, P. Okorie, and G. Marchetto, "LiveMicro: An Edge Computing System for Collaborative Telepathology," in *Proceedings of the 2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[5] M. Mardani and G. B. Giannakis, "Estimating traffic and anomaly maps via network tomography," *IEEE/ACM transactions on networking*, vol. 24, no. 3, pp. 1533–1547, 2015.

[6] A. Soule, K. Salamatian, and N. Taft, "Combining filtering and statistical methods for anomaly detection," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement (IMC '05)*, 2005, pp. 331–344.

[7] 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.

[8] 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.

[9] 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.

[10] A. Sacco, M. Flocco, F. Esposito, and G. Marchetto, "Supporting sustainable virtual network mutations with mystique," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2714–2727, 2021.

[11] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

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

[13] Q. Zhao, Z. Ge, J. Wang, and J. Xu, "Robust traffic matrix estimation with imperfect information: Making use of multiple data sources," in *Proceedings of the joint international conference on Measurement and modeling of computer systems (SIGMETRICS '06)*, 2006, pp. 133–144.

[14] N. Benameur and J. Roberts, "Traffic matrix inference in ip networks," *Networks and Spatial Economics*, vol. 4, no. 1, pp. 103–114, 2004.

[15] P. Tune and M. Roughan, "Spatiotemporal traffic matrix synthesis," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication (SIGCOMM '15)*, 2015, pp. 579–592.

[16] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," *Mathematical Programming Computation*, vol. 4, no. 4, pp. 333–361, 2012.

[17] J. Zhao, H. Qu, J. Zhao, and D. Jiang, "Spatiotemporal traffic matrix prediction: A deep learning approach with wavelet multiscale analysis," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 12, p. e3640, 2019.

[18] Z. Liu, Z. Wang, X. Yin, X. Shi, Y. Guo, and Y. Tian, "Traffic matrix prediction based on deep learning for dynamic traffic engineering," in *IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2019, pp. 1–7.

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

[20] F. Xiao, L. Chen, H. Zhu, R. Hong, and R. Wang, "Anomaly-tolerant network traffic estimation via noise-immune temporal matrix completion model," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1192–1204, 2019.

[21] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[22] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[23] A. Sacco, F. Esposito, and G. Marchetto, "Hide & Seek: Traffic Matrix Completion and Inference Using Hidden Information," in *IEEE 20th Consumer Communications & Networking Conference (CCNC)*. IEEE, 2023, pp. 529–534.

[24] G. Gürsun and M. Crovella, "On traffic matrix completion in the internet," in *Proceedings of the Internet Measurement Conference (IMC '12)*, 2012, pp. 399–412.

[25] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[26] H. Zhou, D. Zhang, and K. Xie, "Accurate traffic matrix completion based on multi-gaussian models," *Computer Communications*, vol. 102, pp. 165–176, 2017.

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

[28] C. M. França, R. S. Couto, and P. B. Velloso, "Data imputation on iot gateways using machine learning," in *Proceedings of the Mediterranean Communication and Computer Networking Conference (MedComNet '21)*, 2021.

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

[30] A. Sacco, M. Flocco, F. Esposito, and G. Marchetto, "Owl: Congestion control with partially invisible networks via reinforcement learning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[31] S. Salman, C. Streiffer, H. Chen, T. Benson, and A. Kadav, "Deepconf: Automating data center network topologies management with machine learning," in *Proceedings of the Workshop on Network Meets AI & ML (NetAI '18)*. ACM, 2018, pp. 8–14.

[32] L. Nie, D. Jiang, S. Yu, and H. Song, "Network traffic prediction based on deep belief network in wireless mesh backbone networks," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2017, pp. 1–5.

[33] R. Vinayakumar, K. Soman, and P. Poornachandran, "Applying deep learning approaches for network traffic prediction," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2017, pp. 2353–2358.

[34] A. Sacco, F. Esposito, G. Marchetto, and P. Montuschi, "A self-learning strategy for task offloading in uav networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4301–4311, 2022.

[35] K. Salamatian and S. Vaton, "Hidden markov modeling for network communication channels," *ACM SIGMETRICS Performance Evaluation Review*, vol. 29, no. 1, pp. 92–101, 2001.

[36] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the 2016 ACM SIGCOMM Conference on Data communication*, 2016, pp. 272–285.

[37] P. S. Rossi, G. Romano, F. Palmieri, and G. Iannello, "A hidden markov model for internet channels," in *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology (IEEE Cat. No. 03EX795)*. IEEE, 2003, pp. 50–53.

[38] J. Liu, I. Matta, and M. Crovella, "End-to-end inference of loss nature in a hybrid wired/wireless environment," in *WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003, pp. 1–9.

[39] S. Tao and R. Guérin, "On-line estimation of internet path performance: an application perspective," in *IEEE INFOCOM 2004-IEEE Conference on Computer Communications*, vol. 3. IEEE, 2004, pp. 1774–1785.

[40] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[41] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[42] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *arXiv preprint arXiv:1605.09782*, 2016.

[43] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[44] K.-Y. Chen, C.-P. Tsai, D.-R. Liu, H.-Y. Lee, and L.-s. Lee, "Completely unsupervised phoneme recognition by a generative adversarial network harmonized with iteratively refined hidden markov models," in *INTERSPEECH 2019 - Annual Conference of the International Speech Communication Association*, 2019, pp. 1856–1860.

[45] A. Sacco, F. Esposito, and G. Marchetto, "Restoring application traffic of latency-sensitive networked systems using adversarial autoencoders," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2521–2535, 2022.

[46] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[47] M. Marcellino, J. H. Stock, and M. W. Watson, "A comparison of direct and iterated multistep ar methods for forecasting macroeconomic time series," *Journal of econometrics*, vol. 135, no. 1-2, pp. 499–526, 2006.

[48] A. Bayati, K. K. Nguyen, and M. Cheriet, "Multiple-step-ahead traffic prediction in high-speed networks," *IEEE Communications Letters*, vol. 22, no. 12, pp. 2447–2450, 2018.

[49] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[50] HMM Learn library, 2021, https://github.com/hmmlearn/hmmlearn/.

[51] 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.

[52] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan, "Network anomography," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement (IMC '05)*, 2005, pp. 317–330.

[53] K. Cho, K. Mitsuya, and A. Kato, "Traffic data repository at the WIDE project," in *USENIX Annual Technical Conference (USENIX ATC '00)*. USENIX Association, 2000.

[54] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, 2000, pp. 417–424.
[55] X. Wang, Y. Chen, W. Ruan, Q. Gao, G. Ying, and L. Dong, "Intelligent detection and recovery of missing electric load data based on cascaded convolutional autoencoders," *Scientific Programming*, vol. 2020, 2020.
[56] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 208–220, 2012.
[57] H. Tan, G. Feng, J. Feng, W. Wang, Y.-J. Zhang, and F. Li, "A tensor-based method for missing traffic data completion," *Transportation Research Part C: Emerging Technologies*, vol. 28, pp. 15–27, 2013.
[58] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.
[59] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, "Internet traffic classification by aggregating correlated naive bayes predictions," *IEEE transactions on information forensics and security*, vol. 8, no. 1, pp. 5–15, 2012.
[60] Y. L. Gwon and H. Kung, "Inferring origin flow patterns in wi-fi with deep learning," in *11th International Conference on Autonomic Computing (ICAC 14)*. USENIX Association, 2014, pp. 73–83.
[61] R. L. S. De Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using mininet for emulation and prototyping software-defined networks," in *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*. IEEE, 2014, pp. 1–6.
[62] Ryu controller. Accessed: 2023-2-7. [Online]. Available: https://ryu-sdn.org/
[63] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

Alessio Sacco received the received the M.Sc. degree (summa cum laude) and the Ph.D. degree (summa cum laude) in computer engineering from the Politecnico di Torino, Torino, Italy, in 2018 and 2022, respectively, His research interests include architecture and protocols for network management; implementation and design of cloud computing applications; algorithms and protocols for service-based architecture, such as Software Defined Networks (SDN), used in conjunction with Machine Learning algorithms.

Flavio Esposito is an Associate Professor with the Department of Computer Science at Saint Louis University (SLU). He received an M.Sc. degree in Telecommunication Engineering from the University of Florence, Italy, and a Ph.D. in computer science from Boston University in 2013. Flavio's main research interests include network management, network virtualization, and distributed systems. Flavio is the recipient of several awards, including several National Science Foundation awards and the Comcast Innovation Award in 2021.

Guido Marchetto received the Ph.D. degree in computer engineering from the Politecnico di Torino, in 2008, where he is currently an Associate Professor with the Department of Control and Computer Engineering. His research topics cover distributed systems and formal verification of systems and protocols. His interests also include network protocols and network architectures.