On the use of machine learning and network tomography for network slices monitoring

Anouar Rkhami, Inria, Univ. Rennes, IRISA
Yassine Hadjadj-Aoul, Univ. Rennes, Inria, IRISA
Gerardo Rubino, Inria, Univ. Rennes, IRISA
Abdelkader Outtagarts, Nokia-Bell Labs, France

Abstract—Network Slicing (NS) is a key technology that enables network operators to accommodate different types of services with varying needs on a single physical infrastructure. Despite the advantages it brings, NS raises some technical challenges, mainly ensuring the Service Level Agreements (SLA) for each slice. Hence, monitoring the state of these slices will be a priority for ISPs. However, due to the high measurements overhead, it is generally forbidden to directly measure the performance of all of these slices. To overcome this limitation, network tomography is a promising solution, consisting of a set of methods of inferring unmeasured network metrics using end-to-end measurements between monitors. In this work, we focus on inferring the additive metrics of slices such as delays or logarithms of loss rates. We model the inference task as a regression problem that we solve using neural networks. In our approach, we train the model on an artificial dataset. This not only avoids the costly process of collecting a large set of labeled data but has also a nice covering property useful for the procedure's accuracy. Moreover, to handle a change on the topology or the slices we monitor, we propose a solution based on transfer learning in order to find a trade-off between the quality of the solution and the cost to get it. Simulation results with both, emulated and simulated traffic show the efficiency of our method compared to existing ones in terms of both accuracy and computation time.

Index Terms—5G, Deep Learning, Transfer Learning, Network slicing, Overlay Network monitoring,Network Tomography,

I. Introduction

Network Slicing (NS) [1] is a key technology that allows network operators to place dynamically different types of services with different types of requirements on top of a single physical infrastructure. Accordingly, NS opens up new business opportunities for network operators by allowing them to automatically lease customized network slices to third parties.

To achieve such an objective, NS leverages the last advents of Network Function Virtualization (NFV) [15] and Software-Defined Networking (SDN) [12]. NFV enables flexible deployment of services, while SDN offers a finer control of resources, including service isolation.

Despite the advantages it brings, NS raises some technical challenges. In fact, once the slices are leased, operators have to ensure that the Service Level Agreements (SLA) for each slice is guaranteed. Hence, monitoring the state of these slices will be a priority for both, operators and slices' tenants. Nonetheless, due to the high measurements overhead associated with the monitoring traffic, and sometimes the lack of cooperation of internal elements of the network, it is generally forbidden to directly measure the performance of all these slices. To overcome these limitations, Network Tomography is considered as a promising solution [13].

Network Tomography (NT) is defined as a set of methods that aim to infer unmeasured network metrics using an end-to-end measurement between monitors. Hence, it reduces the complexity of the monitoring and the network diagnostic process. Another motivation for using NT approaches for the slices monitoring is their perfect fit with the SDN and NFV paradigms. Indeed, NT methods require a holistic view of the state of the network, which is guaranteed through SDN technologies.

Most existing works in Network Tomography had put the accent on inferring all links metrics [17] or a subset of links in a network [6] from end-to-end measurements. In this work and as [28], we focus on inferring metrics for a set of paths of interest (i.e., slices). In fact, to infer the slices metrics, we don't necessarily need to monitor all the links involved in the slices, moreover, we may need to monitor links that don't belong to these slices. Hence, inferring, directly, the slices' metrics, is more efficient and uses less monitoring resources than the approaches that focus on links metrics inferring.

In this work, we focus on identifying additive metrics. Therefore, inferring the slices' metrics is equivalent to solving a system of linear equations, where the unknown variables are the slice metrics and the known constants are the end-to-end path measures. However, in practice, this system of linear equations is non-invertible [10], [4]. In other words, in most cases slices metrics cannot be uniquely identified.

To deal with these drawbacks, we formulate the task of solving the linear system of equations into a regression problem, which we solve using neural networks. By doing so, our approach can also be used in the case where the system of equations can be uniquely solved. Indeed, we can determine the values in an efficient way. We avoid the matrix inversion process that takes too much time in large networks.

Unfortunately, formulating the problem as a regression problem and the use of neural networks to infer slices metrics, raise two main challenges. First, the training process requires a large and diverse amount of real labeled

data to avoid over-fitting. Hence, a real large labeled dataset of metrics must be collected in a small window of time, this will introduce an important overhead. The second issue is that with a learning based approach, if a change occurs in the network topology or the slices we monitor, the training of the neural network architecture must start from scratch to take into consideration the new changes. Repetitive training will then also introduce an important overhead. As response to the first concern, we train our neural network using simulated data. While for the second one, we unveil the potential of transfer learning techniques. [30]

Overall the main contributions of our paper can be summarized as follows:

- We propose a one-step solution to infer the slices' metrics using end-to-end measurements. We formulate it as a regression problem, which we solve using a neural architecture.
- Since neural network based methods require a huge and diverse amount of data, and to avoid generation an overhead in the collection process, we propose a self-training approach to solve the regression problem mentioned above.
- We avoid frequent retraining from scratch when a change occurs in the network topology, or in the monitored slices by using transfer learning techniques. We retrain only a part of the neural network architecture instead of updating all the weights. Hence we reduce the complexity of the approach.
- We conduct extensive simulations and we compare our approach against benchmark solutions. The results show the superiority of our approach in terms of error approximation of slices' metrics as well as the computational complexity.

The remainder of this paper is organized as follows. Sec. II presents the main related work on network tomography. In Sec. III, we provide an overview of the context of our work and the notations we have used. Then, in Sec. IV, we delve into the details of our proposed method. Later, in Sec. V, we describe both the simulation setup and the results analysis. Finally, in Sec VI, we conclude and give an overview of our future work.

II. Related Work

Network tomography (NT) was introduced by Vardi [26] in 1996. Since then, NT has been extensively studied [2], [18], [19], [7].

In NT, we distinguish between two main categories of approaches: algebraic and statistical procedures. In the algebraic approaches, the aim is to find exact solutions, link metrics are modeled as unknown constants metrics and they are inferred using linear algebra techniques. Regarding the statistical approaches, the objective is to approximate the link metrics. In this category, link metrics are modeled as a random variable with unknown probability distribution, then the link metrics are inferred using statistical techniques. Our work can be classified in the statistical category even if it is based on machine learning techniques instead of statistical approaches (e.g., Expectation Maximization (EM)).

Network slices monitoring can be defined as a particular case of overlay network monitoring [3]. Several works focused on the use of network tomography techniques to monitor overlay networks. For example, in [3], authors proposed an algebraic solution to select the necessary paths in an overlay network to fully describe an additive metric on the rest of the available paths. Moreover, M. Demirci et al. focused on the placement of overlay networks for diagnosability of the underlay network [5]. Compared to these works, our work has a different scope, we aim to identify the state of the slices and not use the slices to diagnose the underlay network.

When the target is the additive metrics, the approaches suggested in the literature to infer link metrics can be used to infer the metrics of the slices. In fact, to infer the slice metrics from end-to-end measurements, we can infer the metric related to each link in the slice or all links in the underlay networks and then sum them up.

In [9], A. Gopalan and al proposed an algorithm to determine the minimal number of monitors as well as their placement in a given network to collect end-to-end measurements, that will be used to infer the links' metrics. These approaches are feasible, but, they fail to find the optimal number of monitors to be used to infer the slices' metrics. Hence, in [28], the authors demonstrated that to infer the metrics of a slice, it is not necessary to know the metric of each link on the path. Based on that, they proposed an algorithm to find the minimum number of monitors and their placements to directly identify the slices measurements. What distinguishes our work from this work is that here we do not focus on monitor placement, but on inferring slices' metrics.

Regarding the statistical methods, the focus was on inferring links' metrics. In [16], the authors presented a solution based on multicast probing and the expectation-maximization algorithm (EM) to estimate loss rate on links. Moreover, in [14], authors proposed Flexicast, a method based on a mix of unicast and multicast probing to infer links' delays in a tree topology. They formulate the problem as a likelihood maximization problem, which was solved using the EM algorithm. This kind of approach is characterized by its slow convergence time. In [21] there is a statistical algorithm called ESA merging the EM technique with an evolutionary approach, that outperforms previous mentioned proposals.

More recently, machine learning based solutions have been introduced for network monitoring. Sirnivasan and al used supervised learning, in particular a mix of classification and regression to locate the failed links [25]. Compared to our work, we also use supervised learning but we train a model using an automatically generated dataset. In [22], M.Rahali and al, used machine learning

algorithms and more particularly neural networks. They trained them using a simulated training dataset. However, their focus was on the inference of underlying metrics using information collected in the overlay networks. If a change occurs in the underlay topology or the overlay networks, their approach has to repeat the training from the beginning, which will introduce an important overhead in the monitoring process.

III. Context and problem formulation

A. Context

As mentioned earlier, in this work we aim to monitor a set of network slices placed on a physical network using NT and machine learning techniques.

In general, the construction of an NT solution is done in three main steps: the selection of nodes for traffic data collection (i.e., monitors), the deployment of a probing strategy, and the inference of metrics.

In this work, we focus on the inference part. It is a key part because it is the one that quantifies the quality of the first and the second steps. Hence, we consider that the placement of monitors and the list of probed paths as known. Then, we collect the monitoring data and feed it to our model to estimate the slices' metrics.

B. Model and problem formulation

The notations used in this subsection are summarized in Table I.

Notation	Description
$\mathcal{G} = (\mathbb{V}, \mathbb{L})$	The physical topology
$\mathbb{V}, \mathbb{L}$	Set of physical nodes and links
$\mathbb{P}$	Set of measurement paths
$\mathbb{S}$	Set of slices to monitor
Y_s	Unknown slices metrics vector l_s
Y_p	measurements paths metrics vector
X	Unknown links' metrics vector

TABLE I
Main notations

We assume that the topology of the physical network is known and we model it as an undirected graph $\mathcal{G} = (\mathbb{V}, \mathbb{L})$ where $\mathbb{V}$ and $\mathbb{L}$ represent the sets of physical nodes and of the physical links, respectively. Let $\mathbb{P}$ denote the set of the probed paths and $\mathbb{S}$ the set of slices we want to monitor and which are deployed on top of the physical network $\mathcal{G}$. The cardinals of the sets $\mathbb{V}$, $\mathbb{L}$, $\mathbb{P}$ and $\mathbb{S}$ are respectively, $m = |\mathbb{V}|$, $n = |\mathbb{L}|$, $p = |\mathbb{P}|$ and $s = |\mathbb{S}|$. Let X denote the vector of the unknown links metrics, hence with size n. Similarly we denote by Y_s the vector of the unknown slices metrics, size s. Finally, we define Y_p as the vector of the measured or observed probed paths metrics, whose size is p.

To model the relation between the physical links and the slices being monitored, as well as the relationship between the physical links and the measured paths, we define two matrices A_s and A_p. These are Boolean matrices and their shapes are respectively equal to (s, n) and (p, n). For A_s the entry $A_s(i, j)$ is equal to 1 if the jth physical link belongs to the ith slice, and to 0 otherwise. Likewise, the $A_p(k, l)$ equals 1 if the lth link is on the kth path, 0 otherwise. Finally, we assume that the links' metrics have a known upper bound B.

Using the notation stated before we derive two main equations :

$$A_s X = Y_s, \tag{1}$$

$$A_p X = Y_p. \tag{2}$$

The objective here is to infer the Y_s vector knowing A_s, A_p and Y_p. To solve this problem, one may start by determining X solving Equation (2) and then, a simple multiplication with A_s will give Y_s. However, in practice, the linear system depicted in Equation (2) is undetermined (that is, $p < m$). Moreover, even in determined cases we have another issue here. Solving Equation (2) involves inverting matrix A_p. This takes $O(n^3)$ time, which can be too much for large graphs.

IV. Proposed Method

In this section, we detail our approach and explain how we model the inferred slices' metrics as a regression problem.

A. Inferring slices' metrics using neural networks

To solve the problems discussed in previous section, we model the metrics' inference as a multi-output regression problem, where the features are represented by Y_p and the targets by Y_s. Thus, solving the equation is equivalent to learning the f function defined by

$$Y_s = f(Y_p). \tag{3}$$

To learn f, we use neural networks (NNs). NNs can approximate any function as long as the hidden layers contain sufficient numbers of hidden neurons.

By doing so, our solution can approximate the slices metrics in the case of an undermined system. Moreover, the direct learning of the relation between Y_s and Y_p makes the task independent of the size of the physical network, hence it can scale easily.

The neural architecture we used is a Multi Layer Perceptron (MLP) [23]. It consists of one input layer with p neurons, one output layer with s neurons and h hidden layers. The number of neurons in each hidden layer is a hyperparameter that we fine-tune to reach an optimal performance.

To train this neural architecture, a set of pairs (Y_s, Y_p) must be collected. Since we know the topology of the physical network, we can simulate the distribution of traffic in the network in many configurations and observe the pairs (Y_s, Y_p).

B. Self training

Neural networks are efficient and autonomous once trained, however, as the case for any data-driven approach, it requires a huge and diverse pool of sample and therefore a large amount of labeled data. Moreover, to avoid the over-fitting problem [29], this process must be done under diverse network conditions Unfortunately, the latter process, will generate an overhead and may disturb the network functioning. To solve such a challenge, we propose to train our neural network using a simulated data. We propose to generate a large amount of random pairs $(Y_p^{'}, Y_s^{'})$.

The dataset is generated as follows: we generate a large amount of random samples of links metrics denoted as $X^{'}$. We associate with each link metric a random value between 0 and an upper bound B to generate one sample of $X^{'}$. Then, given A_s and A_p and based on Equations (1) and (2), we generate one pair $(Y_p^{'}, Y_s^{'})$.

After collecting the simulated dataset, we feed it to the neural network and train the latter using the backpropagation algorithm in order to minimize the mean square error (MSE) between the predicted and the target values [24].

Once the training of the neural network finished, we use it to estimate the slice metrics from path measurements.

C. Transfer learning

In real scenarios, the number of slices to monitor may vary with time. In addition, the path measurements can also change, either due to a failure on the physical network, or to an update of the monitors in charge of launching the probes. When these changes occur, and in order to take them into consideration, we have to update the neural network architecture and start the training process from scratch. However, changing the neural network architecture and restart training from scratch each time a new slice is added to the set of slices to monitor may be inefficient and may introduce an important overhead in the monitoring process. To deal with this challenge, we unveil the potential of transfer learning.

Transfer learning is a machine learning technique that allows to transfer what a model learned in one task to another one. Thus, when a change occurs, instead of building a new neural network (NN) and start the training process from the beginning, we take the last version of our model and we modify only the input or the output layer. If the update concerns the slices we monitor, it is the output layer that will be changed. While it will be the input layer which will be changed if the update occurs in the measurements' paths. Moreover, once we make this slight update, we train efficiently the new version of the model. We only update the weights of the new layer and freeze the other ones. This way, the model transfers what it learned before and then starts to learn how to adapt to the new changes.

V. Performance Evaluation

In this section, we evaluate the performance of our monitoring approach. To evaluate its effectiveness, we compare it to two main solutions. In what follows, we present a description of the simulation setup, then we define the approaches against which we compare our solution. Finally, we present and discuss the results.

A. Simulation Setup

In order to test the performance of our method, we tested it using two different topologies. The first one was taken from [20] and the second one from [11]. They are depicted, respectively, in Fig. 1 and Fig. 2. The First topology contains 9 nodes and 22 links, while the second one contains 54 nodes and 68 links.

For each network, two nodes are selected to start probes and exchange the monitoring traffic. Besides, we considered a predefined list of measurement paths, as well as the slices to monitor. The first topology is provided with a dataset of multiple samples of delay measurements preformed on its different links. To get this dataset, the traffic was simulated with the Omnet++4 network emulator [27]. Regarding the second topology, we generate a random dataset of link delays.

Based on these datasets, we computed the end-to-end delay on each path and on each monitored slice according to the equations (1) and (2). For the first topology we monitor 3 slices while for the second topology we monitor 20 slices. The main parameters of our simulation setup are summarized in Table II.

In order to assess the quality of the estimates,we define, in the following, the error formula:

$$\text{Error} = 100 \times |Y_s^{\text{estimated}} - Y_s^{\text{real}}|. \tag{4}$$

In this equation, $Y_s^{\text{estimated}}$ represents the estimated delay on the slices, while Y_s^{real} is the exact value we computed.

Topology	m	n	p	s	Test set
Topology 1	9	22	$[16, 20]$	3	Emulation
Topology 2	54	68	$[20, 60]$	20	Simulation

TABLE II
Topologies specifications

B. Baseline methods

In order to benchmark the performance of our proposed method, we compare it against two well-known state of the art solutions. They both aim to estimate the links' metrics and then use it to estimate the slices metrics.

They can be described as follows:

- Singular Value Decomposition-based method: This method relies on the singular value decomposition method (SVD) [8] to compute the pseudo-inverse of the matrix A_p, denoted A_p^+. Then, the slices metrics are obtained by the following equation:

$$Y_s^{\text{estimated}} = A_s \, A_p^+ \, Y_p \tag{5}$$

- Link level metric prediction: In this method, we apply the learning method we propose but to estimate the link metrics. Therefore, in this case, we define a neural network whose input is the path metrics and the output is the estimate of the link metrics. Once the training is complete, we replace the estimation of the link metrics in Equation (1) to get $Y_s^{\text{estimated}}$.

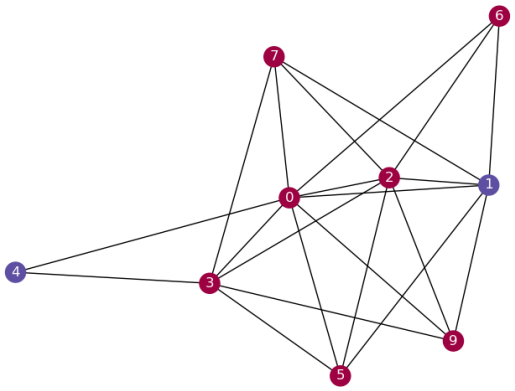

Fig. 1. Topology 1.

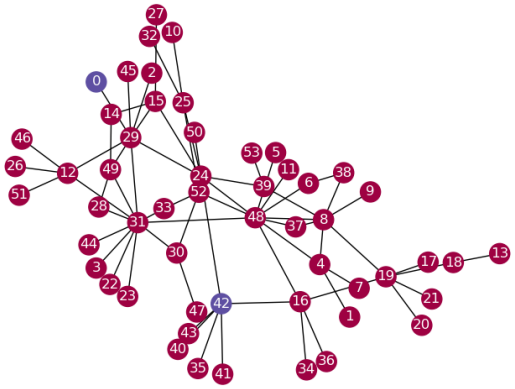

Fig. 2. Topology 2.

C. Simulation results

To compare the performance of our method with the above described approaches, we consider a neural network with only one hidden layer. In each case, we generate a simulated training set with 5×10^5 samples. The hidden layer contains 50 hidden units. These parameters were fixed after several tests when we saw that after increasing the number of hidden units we only increase the computation time while the performance is quasi the same. Moreover the training lasts for 5 epochs.

1) Estimation error vs number of paths used: With each topology, and based on the monitor placement, we fix a list of measurements' paths. For the first topology we fixed 20 paths, while we considered 60 paths for the second one.

To test the relation between the number of measurements paths used and the accuracy of the estimation, we made multiple tests. In each test, we select k paths from the predefined list and we compare the performance of our approach to the SVD and link-based estimations as a function of the number of paths used.

For example, for the first topology, first we use only 15 paths among the 20, then 16, and so on, in order to analyze the effect of the number of paths on the estimation error.

Figures 3 and 4 illustrate the dispersion of the percentage error of the three methods with both topologies.

Increasing the number of measurements paths reduces the percentage error of the solutions. Our method has always a better estimation accuracy. With the first topology and while using 20 paths, the median of the absolute error is 3.45% with our method, while it was 3.71% for the link based method, and 10.76% for the SVD based method. With the second topology, when we use for example 20 paths, the median error is 4.34% for our approach, while it is 7.13% for the link based method and 12.07% for SVD.

Overall, we find that learning-based solutions have taken the lead from the SVD method. In addition, direct estimation of slice metrics is more precise and efficient. In fact, by dividing the task into link estimation and then using that estimation to obtain slice metrics, the error of the first step is propagated, and therefore the performance is lower than the direct method. With our approach, the neural network focuses directly on learning the solution that minimizes the error.

Finally, when using the first topology, we observe that the difference between our method and the one that learns to estimate link metrics is narrow. However, in terms of temporal complexity our solution is more efficient, since we only predict for 3 slices, while the other approach focuses on 22 links.

2) Transfer learning evaluation : In this section we study the performance of the transfer learning methodology we used. For space constraint, and without loss of generality, in these tests, we highlight results related to the second topology. We start with a case where we monitor only 10 slices. Once the training is done, we suppose that new slices are added to be monitored. we added 5, 10 and 15 new slices to the existing 10 slices.

For each case, we used the model trained to monitor the first 10 slices. Then we freeze all its layers except the last one, we change it with a new layer with 15 slices for example for the first case. Then, we compare the performance of this models with the ones that are retrained from scratch and do not use any weights from the original model, which was in charge of monitoring the first 10 slices.

In Figures 5 and 6 illustrate the comparison in term of the median error of estimation and the time to finish one epoch of training after a change occurs. We observe that pre-trained models are less accurate than the whole

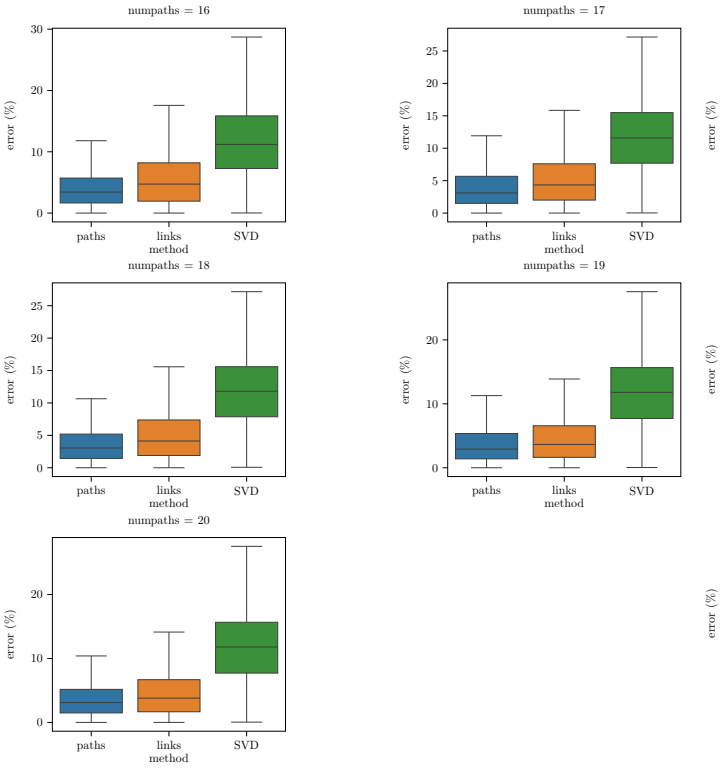

Fig. 3. Topology 1 error vs # of measurements paths

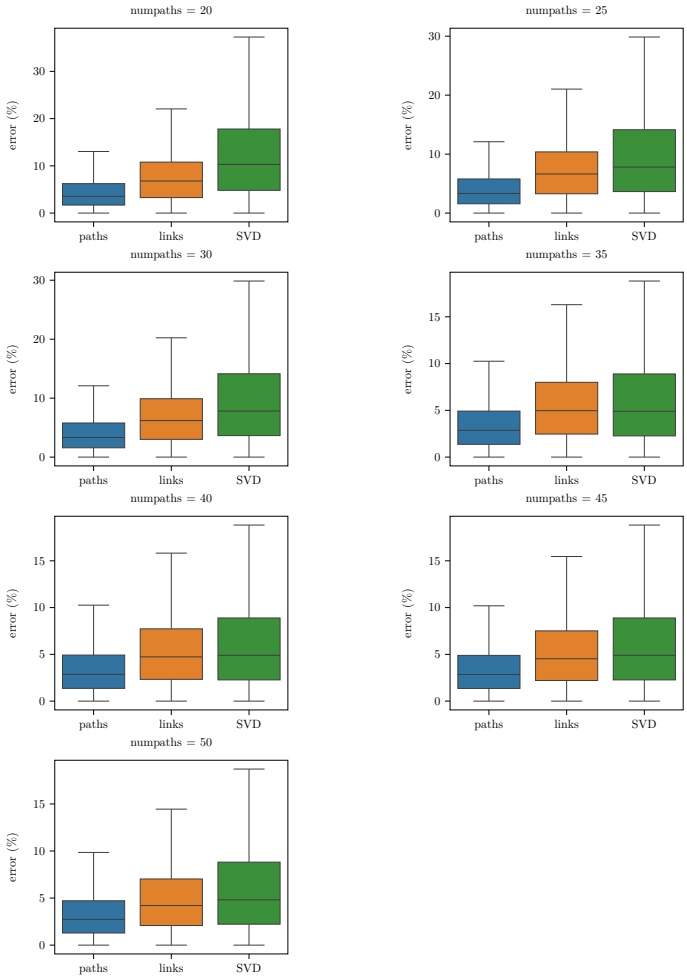

Fig. 4. Topology 2: Error VS # of measurements paths

trained ones. However this difference is tight, besides, when it comes to the training time to get these model, we observe that pre-trained models are three times faster than the other models.

To summarize, the use of transfer learning allows us to find a trade-off between the quality and the complexity of the monitoring solution. It transfers the knowledge acquired during past experiences to get a fast adaptation to the new changes.

VI. Conclusions and Future Work

In this work, we unveiled the potential of machine learning methods and network tomography techniques for network slices monitoring. We focused on additive metrics inference but the approach can be used for the non-additive metrics as well. Unlike existing work, we focused on estimating the slices metrics and not the metrics of the physical links. Hence, we modeled the inference problem as a regression problem and we solved it by training neural networks. To deal with the challenges associated with the training of neural networks, we used a simulated data in the training step. Finally, we proposed a transfer learning technique in order to handle changes in the physical topology as well as the slices subject of monitoring. We evaluated the performance of our approach using an emulated and simulated network traffic. The results show that our approach outperforms the methods with which we compared in terms of both, accuracy and computational complexity.

For future work, we plan to extend this work, by incorporating the first and second stages of network tomography monitoring. Namely, an intelligent selection of measurements' paths as well as an intelligent monitor placement for slices' monitoring. Furthermore, we plan to use neural architectures suitable for graph structured data.

References

[1] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck. Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. IEEE Communications Surveys Tutorials, 20(3):2429–2453, 2018.

[2] A. Chen, J. Cao, and T. Bu. Network tomography: Identifiability and fourier domain estimation. IEEE Transactions on Signal Processing, 58(12):6029–6039, 2010.

[3] Y. Chen, D. Bindel, and R. H. Katz. Tomography-based overlay network monitoring. In Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement, IMC '03, page 216–231, New York, NY, USA, 2003. Association for Computing Machinery.

[4] Y. Chen, D. Bindel, H. Song, and R. Katz. An algebraic approach to practical and scalable overlay network monitoring. volume 34, pages 55–66, 10 2004.

[5] M. Demirci, F. Gillani, M. Ammar, and E. Al-Shaer. Overlay network placement for diagnosability. In 2013 IEEE Global

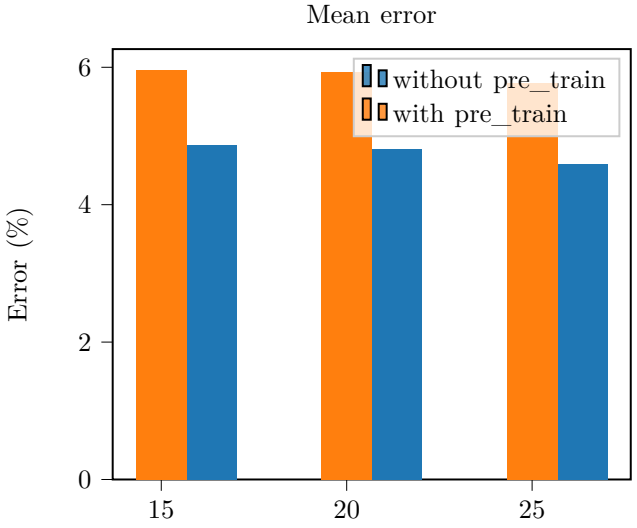

Fig. 5. Topology 2. Error: Pre training vs Retraining from scratch

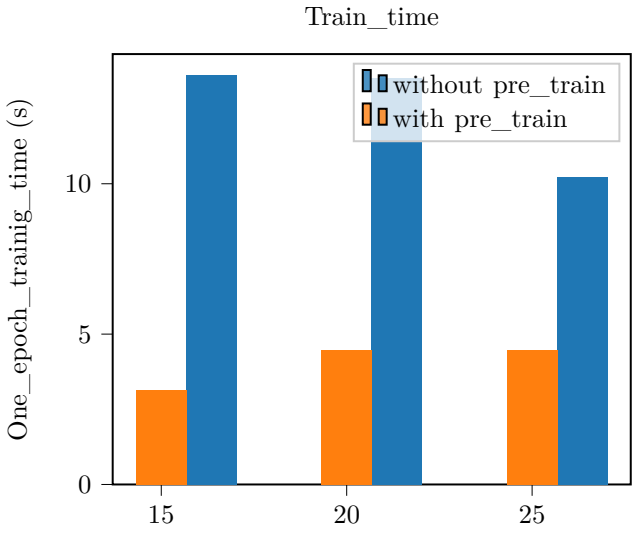

Fig. 6. Topology 2. One Epoch time: Pretrain vs retraining from scratch

Communications Conference (GLOBECOM), pages 2236–2242, 2013.

[6] W. Dong, Y. Gao, W. Wu, J. Bu, C. Chen, and X. Li. Optimal monitor assignment for preferential link tomography in communication networks. IEEE/ACM Transactions on Networking, 25(1):210–223, 2017.

[7] Y. Gao, W. Dong, W. Wu, C. Chen, X. Li, and J. Bu. Scalpel: Scalable preferential link tomography based on graph trimming. IEEE/ACM Transactions on Networking, 24(3):1392–1403, 2016.

[8] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. In Linear Algebra, pages 134–151. Springer, 1971.

[9] A. Gopalan and S. Ramasubramanian. On identifying additive link metrics using linearly independent cycles and paths. IEEE/ACM Transactions on Networking, 20(3):906–916, 2012.

[10] O. Gurewitz and M. Sidi. Estimating one-way delays from cyclic-path delay measurements. In Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213), volume 2, pages 1038–1044 vol.2, 2001.

[11] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. Selected Areas in Communications, IEEE Journal on, 29(9):1765 –1775, october 2011.

[12] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1):14–76, 2015.

[13] E. Lawrence, G. Michailidis, V. Nair, and B. Xi. Network tomography: A review and recent developments. 07 2006.

[14] E. Lawrence, G. Michailidis, and V. N. Nair. Network delay tomography using flexicast experiments. B, pages 785–813, 2006.

[15] Y. Li and M. Chen. Software-defined network function virtualization: A survey. IEEE Access, 3:2542–2553, 2015.

[16] C. Liu, T. He, A. Swami, D. Towsley, T. Salonidis, A. I. Bejan, and P. Yu. Multicast vs. unicast for loss tomography on tree topologies. In MILCOM 2015 - 2015 IEEE Military Communications Conference, pages 312–317, 2015.

[17] L. Ma, T. He, K. K. Leung, A. Swami, and D. Towsley. Inferring link metrics from end-to-end path measurements: Identifiability and monitor placement. IEEE/ACM Transactions on Networking, 22(4):1351–1368, 2014.

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

[19] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. Inferring link weights using end-to-end measurements. In Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment, IMW '02, page 231–236, New York, NY, USA, 2002. Association for Computing Machinery.

[20] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett, and et al. Knowledge-defined networking. ACM SIGCOMM Computer Communication Review, 47(3):2–10, Sep 2017.

[21] M. Rahali, G. Rubino, and J.-M. Sanner. Unicast inference of additive metrics in general network topologies. In Proc. of the 27th IEEE Int. Symp. on the Modeling, Analysis, and Simulation of Comp. and Telecomm. Sys. (MASCOTS'19), 2019.

[22] M. Rahali, G. Rubino, and J.-M. Sanner. TOM: a self-trained tomography solution for overlay networks monitoring. In 2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC'20), pages 1–6, 2020.

[23] H. Ramchoun, M. A. J. Idrissi, Y. Ghanou, and M. Ettaouil. Multilayer perceptron: Architecture optimization and training. IJIMAI, 4(1):26–30, 2016.

[24] C. Sammut and G. I. Webb, editors. Mean Squared Error, page 653. Springer US, Boston, MA, 2010.

[25] S. M. Srinivasan, T. Truong-Huu, and M. Gurusamy. Machine learning-based link fault identification and localization in complex networks. IEEE Internet of Things Journal, 6(4):6556–6566, 2019.

[26] Y. Vardi. Network tomography: Estimating source-destination traffic intensities from link data. Journal of the American statistical association, 91(433):365–377, 1996.

[27] A. Varga. Discrete event simulation system. In Proc. of the European Simulation Multiconference (ESM'2001), pages 1–7, 2001.

[28] R. Yang, C. Feng, L. Wang, W. Wu, K. Wu, J. Wang, and Y. Xu. On the optimal monitor placement for inferring additive metrics of interested paths. In IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, pages 2141–2149, 2018.

[29] X. Ying. An overview of overfitting and its solutions. Journal of Physics: Conference Series, 1168:022022, 02 2019.

[30] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning, 2020.