Martín, I., et al. Machine learning-based routing and wavelength assignment in software-defined optical networks**.** *In, IEEE Transactions on Network and Service Management, 16(3), Sept. 2019, Pp. 871-883*

DOI: https://doi.org/10.1109/TNSM.2019.2927867

Machine Learning-Based Routing and Wavelength Assignment in Software-Defined Optical Networks

Ignacio Martín, Sebastian Troia, José Alberto Hernández, Alberto Rodríguez, Francesco Musumeci, Guido Maier, Rodolfo Alvizu and Óscar González de Dios

***Abstract*—Recently, Machine Learning (ML) has attracted the attention of both researchers and practitioners to address several issues in the optical networking field. This trend has been mainly driven by the huge amount of available data (i.e., signal quality indicators, network alarms, etc.) and to the large number of optimization parameters which feature current optical networks (such as, modulation format, lightpath routes, transport wavelength, etc.). In this paper, we leverage the techniques from the ML discipline to efficiently accomplish the Routing and Wavelength Assignment (RWA) for an input traffic matrix in an optical WDM network. Numerical results show that near-optimal RWA can be obtained with our approach, while reducing computational time up to 93% in comparison to a traditional optimization approach based on Integer Linear Programming. Moreover, to further demonstrate the effectiveness of our approach, we deployed the ML classifier into an ONOS-based Software Defined Optical Network laboratory testbed, where we evaluate the performance of the overall RWA process in terms of computational time.**

***Index Terms*—Optical WDM Networks; Routing and Wavelength Assignment; Machine Learning; Deep Neural Networks; ONOS; Software Defined Networking; Network Automation.**

I. Introduction

The interest in *Artificial Intelligence* (AI) and, more specifically, in the area of *Machine Learning* (ML) has been increasing rapidly in the networking community in recent years. This is mainly due to the fact that nowadays enormous amounts of data can be retrieved from telecommunication networks, e.g., provided by network telemetry, quality indicators of physical signals, traffic traces and logs, user profiling, etc. Leveraging the methodologies of ML, several complex networking tasks can be performed with high accuracy and with limited or even without any human intervention. Examples of applications of ML to the networking area can be found in the following surveys: [1]–[3].

Such a disruptive paradigm is expected to be applicable to any kind of telecommunication network, regardless of its topology, implementation or underlying technology, and may also benefit from network automation as enabled by the *Software Defined Networking* (SDN) principle [4]. After ten years of research and development, Software-Defined Networking (SDN) is finally becoming a reality and offers the opportunity to re-think and build highly programmable networks.

Ignacio Martín and José Alberto Hernández are with the Telematic Engineering Department, Universidad Carlos III de Madrid, Madrid, Spain.

Sebastian Troia, Alberto Rodriguez, Francesco Musumeci, Guido Maier and Rodolfo Alvizu are with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milan, Italy.

Óscar González de Dios is with Telefonica, Madrid, Spain.

Thanks to SDN, global-view networked datasets comprising forwarding, performance and configuration states can be gathered and further exploited with ML/AI algorithms, offering a new range of possibilities to continuously improve how network services are provided and network resources allocated. Indeed, some authors are foreseeing how AI in combination with SDN will revolutionize telecommunication networks and mark paradigm shifts to *Cognitive Network Management* [5] or even *Knowledge Defined Networking* [6]. An extensive tutorial on algorithms, frameworks and applications of Machine Learning and Artificial Intelligence to networking, including open challenges and specific examples of *Intelligent Networking* may be found in [7].

SDN technology lends itself perfectly to the implementation of ML algorithms, or more specifically to intelligent algorithms to speed up control actions on the network. The main strength of this new paradigm is that different network optimization algorithms can be implemented, each of which targets a different cost function. Moreover, thanks to the centrality of the control plane, it is possible to simultaneously train different ML algorithms in off-line mode, i.e. without implementing the output in the network, and only after generalizing and testing the model, this can be deployed. This procedure can be repeated any time the model needs to be retrained, following the evolution of the network behavior itself. In short, SDN enables intelligent control and configuration actions in a very short time and represents a fundamental feature of future telecommunication networks. Indeed, 5G and the rise of new services (i.e. IoT, connected vehicles, Augmented and Virtual Reality AR/VR, etc) is expected to make traffic matrices much more dynamic than today, thus requiring frequent network reconfiguration to better adapt the network resources to the actual traffic needs.

In this article, we investigate how to use ML to target the *Routing and Wavelength Assignment* (RWA) problem in the context of optical WDM networks. Performing RWA corresponds to assigning physical resources, consisting of dedicated wavelength(s) along a physical route between two end-points, to each of the demands in a given traffic matrix. RWA is typically solved in two major use cases: 1) the design of an optical network aiming at the dimensioning of necessary amount of resources to be deployed under some traffic forecast assumptions; 2) the reconfiguration of the optical network, where the assignment of the existing network resources is re-optimized triggered by some dynamic traffic changes, pursuing some optimization objective which typically aims at avoiding traffic congestion, resource underutilization, improved energy-efficiency, etc.

In small or medium-sized network topologies, RWA is formulated and solved using Integer Linear Programming (ILP) that provides optimal (e.g., cost-minimized) solutions at the expense of complex, time-consuming and intensive computations since the problem is known to be NP-hard. In large network topologies, suboptimal heuristic algorithms have been proposed in the literature to speed up the RWA procedure. For an extensive overview on RWA the reader is referred to [8].

In this paper we transform the RWA problem into an ML-based classification problem, where the RWA solution is provided by a classifier in response to a given input traffic matrix. To this end, several ML-based classifiers are trained based on previous data (i.e. optimal RWA for different input traffic matrices). Once trained, such classifiers are able to provide the network configuration for newly-incoming traffic matrices in an online fashion, offering an RWA configuration within a few milliseconds, thus allowing to perform dynamic network adaptation and reconfiguration in response to frequently changing traffic patterns. To further show the applicability of our a ML-based RWA classifier, we report a proof-of-concept experimental integration of ML and SDN in a realistic SDN network environment emulated within the *mininet* [9] framework.

II. Previous Work

Probably, traffic classification was the first networking application where ML techniques showed real applicability and good performance results, either using classical supervised learning algorithms, fuzzy clustering, or alternative approaches like the *Bag of Flows* (BoF) approximation [10]–[13]. However, the rise of novel algorithms (like Deep Learning and Deep Reinforcement Learning) in conjunction with the availability of techniques to analyze Big Data has risen a new wave of ML applications in computer networks [14], often focused on automated network management with reduced human intervention [6], [15]. In particular, such novel techniques have shown applicability in network routing and virtualization [16]–[21].

However, for ML algorithms to be successful, these need to be fed with large quality training datasets, which is sometimes difficult to obtain. Some studies have pointed in this direction reporting the lack of data in networking scenario [22], the need for creating standardized datasets [6], the challenges arising from the collection of high-quality networking data [23] or showing the opportunity for upcoming technologies, such as SDN, to introduce network monitoring systems aware of the ML needs [24].

Concerning optical WDM networks, the use of AI and ML tools has the potential to provide solutions at the physical layer; examples include to estimate optical signal quality of transmission, characterize and operate transmitters, mitigate nonlinearities at receivers, detect link failures or recommend wavelength assignments with low power excursions [25], [26].

In the specific case of the RWA problem, exact Integer Linear Programming (ILP) formulations for its optimal solution exist since long ago [27]. In spite of the cost-minimized solutions provided by ILPs, they suffer from high computational complexity, often requiring minutes or even hours to solve medium-size network topologies. A large number of *Heuristic* algorithms have also been proposed in the literature, offering faster-than-ILP solutions but rather sub-optimal.

Complex mathematical optimization models have also been used in other applications, such as routing in multicast SDN networks [28] and virtual network embedding [29]. Authors in [28] have introduced an ILP and an approximation algorithm to design NFV-enabled multicast trees based on SDN. An important outcome of this work regards the comparison of the execution of run time between the two algorithms. When the fan-out of the multicast trees increases, the run time of the approximation algorithm becomes much lower than that of ILP. Authors in [29] have proposed an ILP formulation to solve the online virtual-network embedding problem. The paper compares the ILP with virtual-network embedding heuristics published so far by other authors. Simulation experiments showed how far the state of the art heuristics are from an ILP-based optimization method. The difference between the virtual-network request acceptance-ratio of the heuristics and the proposed ILP is at least 30%.

In this work, we introduce the idea of using supervised classification algorithms to address the RWA problem. Essentially, a classifier which is trained with already labeled RWA configurations (solved either with an ILP or a heuristic algorithm) is capable of reporting optimal or near-optimal RWA configurations much faster than ILPs and Heuristics. To the best of our knowledge, this is the first study that shows how to map the RWA problem into a supervised classification problem that can be tackled with well-known ML algorithms like Logistic Regression or Deep Neural Networks. We show that, once trained, the ML model can be queried to solve RWA within a few microseconds (Logistic Regression) or milliseconds (Deep Neural Network). Other AI-based studies have addressed the RWA problem using Genetic Algorithms [30] or Reinforcement Learning [31], but these are also computationally costly and slow to converge. We have further implemented this ML-based RWA methodology in an ONOS-based lab test to show real applicability in SDN scenarios.

The remainder of this work is organized as follows: Section III reviews how RWA can be modeled as a supervised classification problem to be trained in an offline fashion, and further shows its applicability in an ONOS-based SDN scenario. Section IV provides a summary of the experiments carried out and the results obtained, both from a ML perspective and networking viewpoint. Finally, Section V concludes this paper with a summary of its main contributions.

III. Methodology

A. A review on RWA algorithms

Routing and Wavelength Assignment (RWA) in optical networks is typically modeled as a multi-commodity flow problem. In such a problem, a directed graph is used to represent the optical network physical topology, consisting of a set of Reconfigurable Add/Drop Multiplexers or ROADMs (i.e., the nodes in the graph), interconnected by optical fiber

links (i.e., the links in the graph). For a given set of traffic demands between node pairs in the physical topology, solving the RWA consists of deciding, for each traffic demand, both the route and wavelength to be used through the network, subject to a set of different main constraints, namely: 1) *flow conservation*, i.e., at each node the amount of incoming flows is equal to the amount of outgoing flows, excluding the traffic inserted/terminated at that node; 2) *capacity constraint*, i.e., in each link, the number of wavelengths used must not exceed a given threshold W; 3) *wavelength usage*, that is, in each link a given wavelength can be assigned to only one flow (i.e., one traffic demand).

Additionally, other constraints can be considered depending on other technological aspects, such as: 4) *wavelength continuity*, which must be ensured whenever no optical-electronic-optical (OEO) conversion with wavelength switching is allowed in transit nodes; 5) *maximum optical reach*, which may be enforced to limit the optical path length, due to physical-layer impairments affecting the optical signal quality level.

Typical objective functions adopted when solving the RWA pursue the minimization of the number of wavelengths used, the number of optical fibers to be deployed (e.g., under a scenario where multiple fibers per link are allowed), the number of transponders (i.e., receiving/transmitting devices), the energy consumption, etc. Exact mathematical models to optimally solve the RWA problem are typically based on Integer Linear Programming (ILP), however this approach is known to be NP-hard and particularly slow in large-topology scenarios with multiple wavelengths W and under high-load conditions. To alleviate such computational complexity, a number of heuristic algorithms have been proposed in the literature to speed up finding near-optimal RWA network configurations in a more scalable way. Such heuristics are mainly based on the decomposition of the RWA problem into two steps: first routing, then wavelength assignment, or vice-versa. In turn, several approaches can be adopted to solve each of these two sub-problems. On one hand, to perform the routing step, pre-computed fixed routes can be assigned to each demand, based on shortest-path. However, such an approach may lead to a high number of required wavelengths or even unfeasible solutions in case the number of wavelengths is constrained. Therefore, other approaches consider fixed alternate routes to be selected for each of the given demands, or even adaptive routing, where demands are sorted and routed one after the other. On the other hand, to perform wavelength assignment, several methods exist, taking into account either the most-used or the least-used wavelength, or even considering simpler approaches such as random assignment, first-fit, etc. For a more comprehensive overview of such approaches, the reader is referred to [8].

In this paper, we consider a version of the RWA problem in which no wavelength conversion (i.e., O-E-O) is allowed at transit nodes, and where lightpath route-length is not constrained, namely, we consider the *flow conservation*, *capacity*, *wavelength usage*, and *wavelength continuity* constraints mentioned above. As for the optimization objective, we consider the minimization of the total number of transponders.

B. Turning RWA into a supervised classification problem

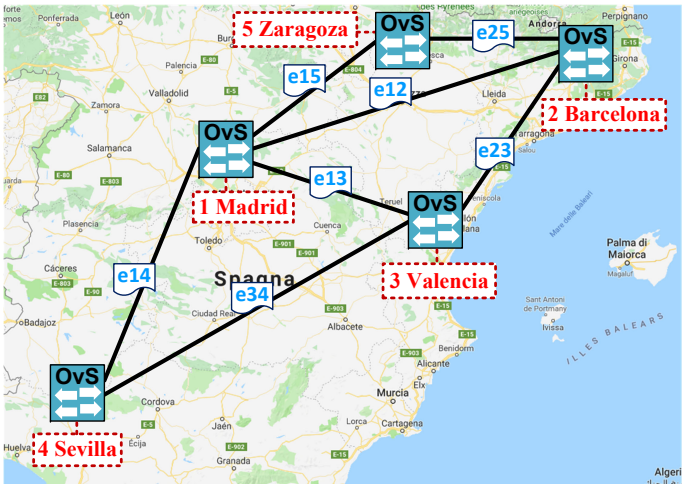


Fig. 1. Schematics of the proposed 5-node network for RWA modeling. The network models both offline training phase and online prediction phase

Consider the 5-node network topology of Fig. 1 and let us assume that each link is equipped with a number of optical transponders allowing the use of W wavelengths per link each one operating at C Gbps (in what follows, W lambdas @ C Gbps).

Let us also consider a 5×4 Traffic Matrix (TM) whose elements d_{ij} (in Gbps) denote the traffic demand from source i to destination node j ($i \neq j$). The RWA algorithm produces a list mapping each traffic demand d_{ij} (input) to a sequence of links and wavelength assignments (output). In the network of Fig. 1, such a routing and wavelength configuration may for instance be:

$$\begin{aligned} d_{12} &\rightarrow (e_{12}, \lambda_1) \\ d_{13} &\rightarrow (e_{13}, \lambda_1) \\ &\vdots \\ d_{24} &\rightarrow (e_{23}, e_{34}, \lambda_2) \\ &\vdots \\ d_{54} &\rightarrow (e_{52}, e_{23}, e_{34}, \lambda_3) \end{aligned}$$

which states that demand d_{12} from source node 1 to destination node 2 uses direct link e_{12} and the first wavelength λ_1. Similarly, demand d_{54} goes through the route defined by links $e_{52} - e_{23} - e_{34}$ and uses the third wavelength λ_3. This routing and wavelength configuration (RWC) applies only to that particular demand matrix. In other words, the RWA receives as input a serialized traffic matrix with all source-destination traffic demand requests and outputs its optimal RWC list:

- Input: $TM_1 = \{d_{12}, d_{13}, \ldots, d_{54}\}$
- Output: $RWC_1 = \{(e_{12}, \lambda_1), \ldots, (e_{52}, e_{23}, e_{34}, \lambda_3)\}$

The output label $y_1 = RWC_1$ is obtained by solving the RWA ILP for that particular Traffic Matrix TM_1. If the ILP is run again for a second traffic matrix TM_2, then a new optimal RWC, namely $y_2 = RWC_2$ should be obtained, but perhaps the previous RWC_1 is also valid to satisfy all traffic demands

while meeting the RWA constraints. In general, if $10,000$ different traffic matrices are fed to the ILP, up to $10,000$ different optimal $RWCs$ could be potentially produced by the ILP. However, in practical scenarios, many RWCs are applicable to several input traffic matrices, thus reducing the space of RWC under consideration.

Consequently, the RWA problem can be transformed into a multi-class classification problem where the input variables are the elements of a serialized traffic matrix and the output labels are the full network routing and wavelength configuration (RWC) set as obtained from solving the ILP. This dataset can then be used to train a classical supervised ML algorithm. Essentially, when fed with sufficient data examples (both traffic matrices and the associated optimal $RWCs$), the ML algorithm should be able to generalize and produce an optimal RWC upon a new unseen traffic matrix, thus performing RWA accurately without the need for solving the ILP. Furthermore, the ML model should learn from the input data provided, i.e. if instead of providing RWC solutions from an ILP, the ML model is fed with data manually configured from an operator or from any other algorithm (i.e. heuristics), the ML algorithm should also be able to reproduce that way of solving RWA, thus mimicking the algorithm from which it learnt.

Concerning the number of RWC classes, having too many of them may prevent the ML algorithm from correctly learning the data patterns due to the so-called *curse of dimensionality*. Hence, aiming at reducing the number of RWCs in use, we performed a forward evaluation of RWC to reduce the number of classes by re-assigning TMs with more frequent RWC as long as they provide a feasible solution and meet some minimum requirements in terms of average network load and hop count.

C. Dataset generation with Netgen and Net2Plan

To generate RWA datasets, we have developed the Netgen tool, built upon Net2Plan[1] planner tool [32]. Net2Plan is an open-source Java tool designed for planning, optimization and evaluation of communication networks. For that, the tool provides a both a Graphical User Interface (GUI) and a Command Line Interface (CLI) that allows the design of abstract representations of networks and provides a diverse set of optimization algorithms for different networking scenarios. In particular, both ILP and several heuristic-based algorithms for RWA are available in Net2Plan.

Netgen[2] gives users the ability to leverage Net2plan's features to be used in a highly scalable and efficient setting that enables synthetic dataset generation. Netgen is composed by three modules, aiming to cover three different phases in the dataset generation process:

- The first module in Netgen is the *Traffic Matrix generator*. When invoked, the program reads the topology and a canonical traffic matrix which is then modified (adding random noise) to generate a new TM for RWA solving by Net2Plan. Random noise is introduced through statistical distributions centered at the canonical value of the matrix and with configurable variability. Currently, Gaussian and Uniform noise distributions are supported.
- Next, the *Net2Plan Wrapper* invokes Net2Plan to solve the desired algorithm for the input traffic matrices in parallel. Indeed, Netgen acts as a scheduler and manager of a pool of Net2Plan instances aiming at a better execution performance through parallelization.
- The final component of Netgen is the *Result Parser*, which reads the output of Net2Plan, parses it and outputs a Comma-Separated Values (CSV) file suitable for ML processing.

D. ML models: LR and DNN

Once the datasets have been generated, two of the most popular ML algorithms have been trained and tested, namely: *Logistic Regression* (LR) and *feed-forward Deep Neural Network* (DNN).

Logistic Regression [33] is a simple linear classifier that adjusts a linear regression to the data together with a *softening* sigmoid function that approximates class probability. Logistic Regression classifiers are very fast to train and interpret but prone to underfitting due to their linear nature. In contrast, Deep Neural Networks [34] have shown great performance in a large number of scenarios thanks to the fact that they stack linear layers of neurons paired with non-linear activations, thus producing non-linear classifiers.

In particular, our DNN configuration comprises six fully connected layers, with *dropout* in the first and fourth layers and ℓ_2 *regularization* in the third and fifth layers. Activation is performed using the well-known rectified linear unit (*ReLU*) and hyperbolic tangent (*tanh*) functions. The optimization process for the training phase has been performed using Tensorflow's Stochastic Gradient Descent (SGD)[3] aiming at minimizing *softmax cross-entropy* as the classification loss function. Specifically, the network performs $16,000$ training steps (each step uses 400 training samples as batch size) with a learning rate of 0.02. Such a DNN architecture has been obtained after multiple trial and error experiments until satisfactory generalization results were obtained by checking the classification error in the train, test and validation sets, along with stable accuracy and loss results. Such a manual inspection process aimed at producing a DNN architecture as simple as possible. Regularization and dropout were added to make the model robust against noise and overfitting.

Thanks to softmax, for a given input TM, both ML algorithms output an array of values between 0 and 1 that can be interpreted as the probability for an RWC to satisfy such TM. Then, the RWC class with largest likelihood, i.e. the most suitable RWC for that TM according the ML algorithm, is checked for feasibility to verify whether it does indeed satisfy all source-destination demands. If it does not (i.e. such first-ranked RWC is *unfeasible*), then the second largest probability RWC class is selected as potential candidate and checked for feasibility. This is repeated for the top-10 RWC. Remark that a given RWC is feasible for a network setting

[1] See http://www.net2plan.com, last access April 2019

[2] Available at: https://github.com/ignmarti/Netgen, last access: April 2019

[3] See TensorFlow's main website https://www.tensorflow.org/, last access April 2019

(i.e. TM and network topology) in an optical WDM scenario if all the next three conditions are met:

- All traffic demands are allocated meeting the wavelength continuity constraint
- All link occupations (link loads) are below 100% occupation
- No link is used more than once per wavelength

E. ML performance assessment

In supervised classification learning, the ML model learns a function g that best maps input X and output y as $y = g(X)$ from a set of training examples with labeled data $\{X_i, y_i\}_{i=1}^{N_{tr}}$. In our case, y_i refers to the RWC for a given input traffic matrix TM_i (i.e. X_i) and N_{tr} is the number of data samples used for training. ML algorithms are in charge of constructing such function $g(X)$ from the training set that separates all RWC classes with minimum error.

Once a model is obtained, the following stage consists on testing its ability to predict the result of unobserved data samples, i.e. evaluate the model's generalization capabilities. Classification performance is evaluated making use of the well-known *Precision*, *Recall* and *F-score* metrics [35].

Precision measures the fraction of events when the ML algorithm correctly assigned the appropriate RWC over the total cases assigned to that particular RWC. Recall quantifies the number of samples from each RWC class that were identified correctly. In a multi-class classification problem with up to M RWC classes, the following confusion matrix C with true classes as rows and modeled classes as columns characterizes the effectiveness of the ML algorithm:

$$C = \begin{pmatrix} tp_{11} & fn_{12} & \dots & fn_{1M} \\ fp_{21} & tp_{22} & \dots & fn_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ fp_{M1} & fp_{M2} & \dots & tp_{MM} \end{pmatrix} \quad (1)$$

where (TP, FP, TN, FN) refer to True/False Positive/Negative respectively. Precision and Recall for the j-th RWC is computed from the matrix elements as:

$$\text{Prec}_j = \frac{c_{jj}}{\sum_k c_{jk}} \qquad \text{Recall}_j = \frac{c_{jj}}{\sum_k c_{kj}} \quad (2)$$

The average Precision and Recall values can be computed over all M classes as:

$$\overline{\text{Prec}} = \frac{\text{Prec}_j}{M} \qquad \overline{\text{Recall}} = \frac{\text{Recall}_j}{M} \quad (3)$$

Finally, the F-score metric trades off Precision and Recall by computing their geometrical mean:

$$\text{F-score} = 2 \times \frac{\overline{\text{Prec}} \cdot \overline{\text{Recall}}}{\overline{\text{Prec}} + \overline{\text{Recall}}} \quad (4)$$

F. Time complexity of ML algorithms

The time complexity of ML for both training and testing datasets varies significantly across different algorithms. For example, Decision Trees employing C4.5 takes complexity of $O(mn^2)$ where m is the number of features and n is the number of data samples. Logistic Regression and Support Vector Machines can be solved in general with $O(mn)$ and $O(n^3)$ operations respectively. Others like kNN (Nearest Neighbors) requires $O(nlog(n))$ for training while prediction requires $O(klog(n))$ for k the number of neighbors. Finally, stochastic gradient descent (SGD) used in DNNs requires $O(en\overline{p})$, where n is the number of training samples, e is the number of iterations (epochs) and $\overline{p}$ is the average number of non-zero attributes per sample. The reader is referred to [36], [37] for a good summary on the time complexity of most popular ML algorithms.

In general, finding an optimal set of weights for a classical NN is NP-complete, as demonstrated in [38]. However, it is possible to learn good NN parameters using the so-called back-propagation algorithm, in particular the Stochastic Gradient descent (SGD) iterative method, which is faster and easier to program than most optimization methods. Indeed, the computational complexity of back-propagation through SGD is polynomial time.

Essentially, back-propagation adjusts the weights of the neurons in a top-down approach comparing the output with the expected value and computing derivatives of the error with respect to each parameter [39]. In particular, the Stochastic Gradient descent (SGD) method iteratively adjusts the weights in the direction reducing the error size, making the training process relatively simple and yielding a time complexity of $O(N\sum_{l=1}^{L} s_l s_{l-1})$ for a minibatch of N data points and layer size s_l (L total layers), i.e. approximately $O(N^3)$ [40].

In conclusion, neural networks and machine learning in general are not intended to solve NP-complete problems like the RWA ILP. However, ML algorithms are effective techniques for finding patterns from data which applies in our use-case. In other words, the next experiments show that a training set with some thousand traffic matrices and their optimal $RWCs$ contain enough patterns that can be effectively learnt with a DNN. This DNN model offers ILP-like RWA solutions within milliseconds, hence mimicking the behavior of the ILP from which it learnt the patterns but in a much shorter time.

G. Deployment of Machine Learning classification models in an emulated SDN network scenario

In addition the proposal of an ML-based method to solve the RWA problem, we have further implemented the model described in Section III-D to work in an SDN context. Implementing a network optimization algorithm inside an SDN network means implementing a software application that interacts with the control plane in order to deploy the RWA configurations in the network automatically. In Fig. 2 we show a typical SDN architecture composed by the data, control and application plane. In particular, the application that implements the algorithms described in Section III-D is called Machine Learning Routing Computation (MLRC) component.

In this light, the 5-node network topology of Fig. 1 has been implemented in an ONOS-based [41] SDN architecture, as shown in Fig. 2. In this scenario, the MLRC application is capable of reading data, training and applying both ML models (i.e. LR and DNN) to determine the optimal network config-

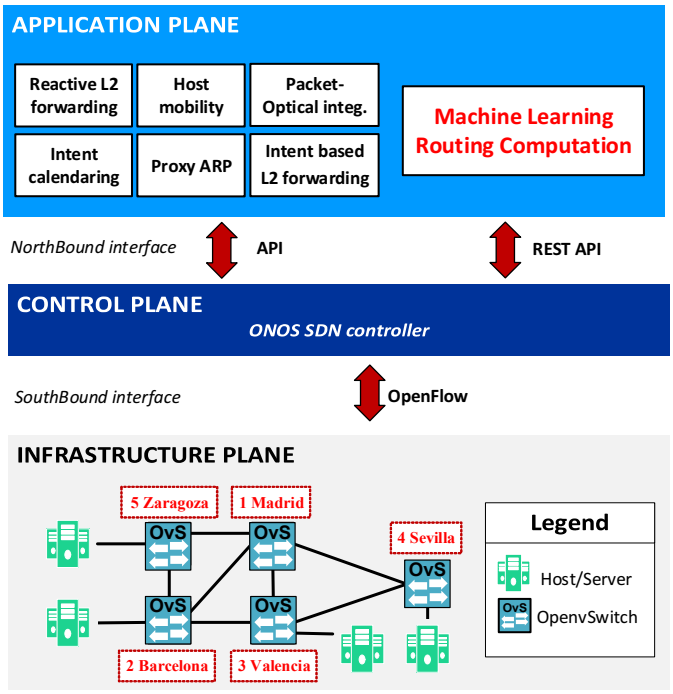


Fig. 2. Machine Learning-aware SDN network architecture composed by three layers. The *infrastructure plane* represents the dataplane of the network composed by 5 nodes and 4 servers; the *control plane* represents the framework in which the routing configurations are performed. It has been realized with ONOS controller and uses the Openflow protocol to communicate directly with the nodes (OvS); finally, the *application plane* represents the intelligence of the network. The whole set of network configuration decisions are taken in this layer exploiting specific API and REST API libraries to interact with the control plane.

uration upon distinct traffic matrices. Using REST APIs[4], it captures traffic matrices with variable granularity (e.g., every 5 seconds) and trains the ML models continuously in order to keep it updated. The purpose of MLRC is to acquire and classify traffic matrices by means of a supervised learning algorithm trained with a set of optimal routing solutions. Once the MLRC module is trained, it may be queried to provide real-time decisions upon the detection of changes in the network traffic matrix.

As shown in Fig. 3, the MLRC module comprises five sub-modules, namely: Traffic Matrix Acquisition, Data Storage, Model training, Classification model and Routing computation.

Traffic Matrix Acquisition. This is the first phase of the module, that is, the acquisition of traffic matrices from the network. Every 5 seconds, the amount of bytes on each traffic flow is extracted from the switches.

Data Storage. The data storage receives the traffic matrices in order to make them available to the other sub-modules. It has the goal of storing not only the traffic matrices but also the trained classification models.

Model training. In this sub-module the machine learning algorithm is trained. This represents the heart of the whole proposed application. It implements the classification models

[4]See: https://wiki.onosproject.org/display/ONOS/ (Appendix B of the Developer Guide), last access: April 2019

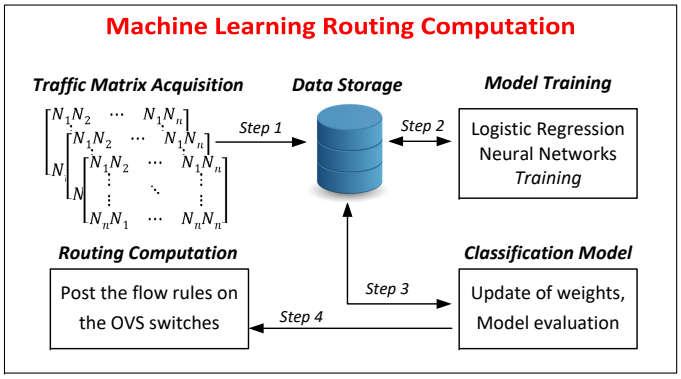


Fig. 3. Machine learning Routing Computation module workflow. The *Traffic Matrix Acquisition* is the first stage of the module. In order to train the MLRC, we need to collect a lot of data, in terms of traffic matrices (Step 1). They are stored in a *Data Storage*, useful for the other sub-modules. Then the model start to be trained in the *Model Training* sub-module (Step 2). Once the training is over, the *Classification Model* will update the classifier with the trained model (Step 3). In the end, the *Routing Computation* module will be called in order to instantiate the flow rules inside the OvS switches (Step 4).

explained in the previous section. The goal of this sub-module is to learn how to classify traffic matrices that share the same routing configuration.

Classification model. This sub-module hosts the last updated classifier that is actually used to classify the input traffic matrix. It restores the model from the data storage and provides the optimal routing scheme that is passed to the *Routing computation* module. The subdivision between *Model training* and *Classification model* was made to have a scalable model, as it is possible to add other classification algorithms, and always updated, as the training is done continuously.

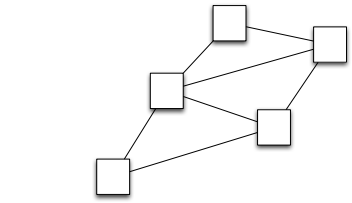
Routing computation. Finally, the routing scheme obtained by the classifier is appropriately translated into flow rules for network switches. After that it overwrites the old flows with those just obtained, avoiding memory over-flow of the switches.

IV. Experiments

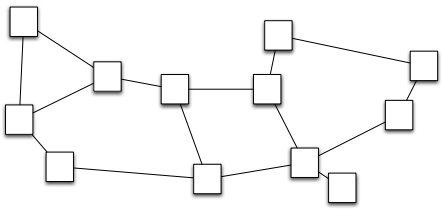
A. Scenario setup

For the experiments, we will consider the two network settings in Fig. 4. The former *5-node* topology is the same one as previously depicted in Figs. 1 and 2; the latter comprises the well-known US Abilene network topology with 12 nodes and 15 bidirectional links.

All nodes in both topologies are assumed to have the same type of transponders with the same wavelength configurations each time. In the 5-node topology, the population of each node is associated with population of the corresponding city in Spain (namely Madrid, Barcelona, Valencia, Zaragoza and Seville) and the distance (in kilometers) is computed from real GPS coordinates. The canonical traffic matrix has been obtained using the so-called population-distance (aka gravity) model where the traffic demands between each two cities is proportional to their population size and distance. The population-distance network traffic model was proposed in [42].



(a) 5-NODE



(b) ABILENE

Fig. 4. Network topologies for Experiments.

Starting from a base or canonical traffic matrix obtained with such population-distance model, Netgen generates 10,000 new traffic matrices by adding white Gaussian noise with zero mean and squared coefficient of variation equals to 0.15 (i.e. the standard deviation of each source-destination pair is obtained as 0.15 times the mean traffic demand (in Gb/s). This way, the dataset has been augmented up to $10,000$ different Traffic Matrices (TM), which are then labeled with the help of Netgen and Net2Plan. Concisely, the 10,000 TMs were solved by the ILP providing some thousand optimal RWCs; these were further downsized to a few tens or hundreds RWCs allowing a tolerance of 0.3 in average hop count. Following this process, four different datasets have been generated allowing different optical configurations in the network (i.e. lambdas and capacity), namely: *5@400, 5@100, 8@40 and 10@40* datasets. Remark that 5@400 refers to 5 lambdas per fiber link each one operating at 400 Gb/s.

In the Abilene topology, instead of using the population-distance model along with Gaussian noise, a number of 48,096 real traffic matrices collected every five minutes during six months are publicly available at SNDlib[5]. For this topology, different optical configurations have also been fed along to Netgen for data generation: 20@400, 40@100 and 20@100 Gb/s.

B. ML performance results for the 5-node network topology

The datasets for the 5-node network have been used to train the three ML models described in Section III-D, namely LR with Lasso and Ridge regularization and DNN. The first two models are examples of simple linear models with typical poor classification performance (due to underfitting) but offering interpretable results, while the third one represents a more advanced and accurate non-linear model.

Lasso and Ridge regularization schemes (i.e. ℓ_1 LR and ℓ_2 LR respectively) have been selected as they are the most commonly used and stablished regularization schemes. For further reference, the work in [43] provides a comparison between both approaches. The regularization constant for logistic regression algorithm is determined through 10-fold cross-validation, where training data is split into 10 chunks and each is used once as testing set with different candidate parameters. A *hold-out validation* set is kept aside for final validation.

Concerning the DNN model, no cross-validation is performed and instead regularization and dropout hyper-parameters are set by heuristics, which are validated through the hold-out set reported as validation along with stable accuracy and loss results (see next sections).

For the elaboration of the probability ranking, the ML models are set up to produce class probability estimates and consider the top 10 most likely RWC solutions. If none of these provide a feasible solution (that is, satisfy demands and requirements for the traffic matrix), that particular data sample is considered unfeasible.

Table. I shows the experimental results for all algorithms and optical configuration pairs together with the ILP solution provided by Net2Plan. The table provides evaluation results both in terms of ML and networking metrics. Regarding ML metrics, which summarize how well each ML model performs at classifying, F-score values for the training, test and validation sets are shown. Concerning networking-related metrics, the table shows average and bottleneck Link Loads (ave_LL and max_LL), average and maximum number of hops per demand (ave_hops and max_hops), number of wavelength-link resources used (wl_used) and percentage of feasible solutions obtained by the ML algorithm in both training and validation sets.

In the first dataset, the three ML models show good performance in terms of networking metrics, while the DNN outperforms LR in terms of ML metrics. In this scenario, the ML models have to choose between 15 optimal RWC configurations. Essentially, the DNN is very accurate at selecting the right RWC (F-score 0.787) while LR fails more often (F-score 0.606 and 0.604). However, since the ML algorithms produce a top-10 list, it often happens that there is one RWC in the top-10 which is suitable, feasible and does not have a large impact regarding link load, average hop count or wavelengths used. Thus, an RWC classification error may still be a good solution in terms of networking metrics, slightly suboptimal only from the right RWC. Finally, it is worth remarking that this first dataset comprises a low-load scenario (13% load).

In the second dataset, the average link load is 35% with some links fully utilized (max load is one), and the RWA ILP generates more RWC classes, up to 69 different RWC for the ML models to choose from. We observe that LR models have a clear performance decrease since they are unable to select the right RWC (F-score 0.542) while the DNN achieves high F-score values up to 0.952. Regarding networking metrics, again the DNN reaches nearly 100% feasibility and similar networking metrics than the ILP.

Finally, in the third and fourth datasets, we observe the same behavior as before: the DNN is both good at selecting the right RWC (F-score above 0.8 with 215 and 197 RWC classes re-

[5]See http://sndlib.zib.de/home.action for further details, last access April 2019.

TABLE I
5-NODE TOPOLOGY: ML RESULTS

DATASET #1	5@400 ($M = 15$ RWC classes, 70 wavelenght-link resources)									
	ML Metrics - F-score			Network Metrics						
Algo.	Train	Test	Val	avg_LL	max_LL	ave_hops	max_hops	wl_used	Feas train	Feas val
ILP	-	-	-	0.130	0.51	2.05	4	41	100%	-
$\ell_1 LR$	0.613	0.604	0.606	0.129	0.51	2.05	4	41	99.9%	99.9%
$\ell_2 LR$	0.616	0.600	0.604	0.129	0.51	2.05	4	41	99.9%	99.9%
DNN	0.807	0.804	0.787	0.130	0.51	2.05	4	44	100%	100%
DATASET #2	5@100 ($M = 69$ RWC classes, 70 wavelength-link resources)									
	ML Metrics - F-score			Network Metrics						
Algo.	Train	Test	Val	avg_LL	max_LL	ave_hops	max_hops	wl_used	Feas train	Feas val
ILP	-	-	-	0.36	1	1.77	3	49	100%	-
$\ell_1 LR$	0.549	0.540	0.542	0.32	1	1.69	3	50	99.05 %	99.30%
$\ell_2 LR$	0.545	0.511	0.542	0.37	1	1.78	3	48	62.22%	61.30%
DNN	0.954	0.949	0.952	0.36	1	1.77	3	49	99.98%	99.88%
DATASET #3	10@40 ($M = 215$ RWC classes, 140 wavelength-link resources)									
	ML Metrics - F-score			Network Metrics						
Algo.	Train	Test	Val	avg_LL	max_LL	ave_hops	max_hops	wl_used	Feas train	Feas val
ILP	-	-	-	0.50	1	1.89	4	98	100%	-
$\ell_1 LR$	0.399	0.327	0.361	0.50	1	1.87	3	95	85.37%	83.80%
$\ell_2 LR$	0.395	0.318	0.295	0.49	1	1.86	4	94	80.10%	81.00%
DNN	0.815	0.801	0.808	0.49	1	1.86	3	97	99.45%	97.16%
DATASET #4	8@40 ($M = 197$ RWC classes, 112 wavelength-link resources)									
	ML Metrics - F-score			Network Metrics						
Algo.	Train	Test	Val	avg_LL	max_LL	ave_hops	max_hops	wl_used	Feas train	Feas val
ILP	-	-	-	0.55	1	1.65	3	84	100%	-
ℓ_1 LR	0.400	0.330	0.350	0.45	1	1.87	3	81	81.45%	82.00%
ℓ_2 LR	0.390	0.329	0.319	0.51	1	1.78	3	68	85.32%	85.80%
DNN	0.851	0.837	0.854	0.52	1	1.63	3	82	99.63%	96.88%

spectively) and show networking performance metrics similar to those provided by the ILP. Linear models are again far from the DNN in terms of ML performance and feasibility.

In summary, the Deep Neural Network model is very good at replicating the results obtained by the ILP solution, resulting in a very high feasibility score and showing comparable link load and hop count metrics to the ILP. Concisely, neural networks achieve almost complete feasibility, being in all datasets above 95% of the data samples. Similarly, logistic regression algorithms may be suitable in some scenarios (low load and few RWC to choose from), but far worse than the DNN model. In the next section, only DNN models are considered, this time trained with both ILP and First-Fit heuristic-based RWCs.

C. ML performance results for the Abilene network

Table. II shows the results obtained for the DNN in the Abilene network topology assuming different capacity and wavelength configurations. In this case, we consider training a DNN with the output result of both optimal ILP and first-fit heuristic RWA of Net2Plan. We observe that in all ILP cases, the number of RWC classes is about $1,500$ for $48,096$ training points, and the results are very good both in terms of F-score and network related metrics (link load, hop count and wavelength-link resources), i.e. very similar to ILP solutions. Concerning feasibility, values around 95% are obtained in all DNN (ILP) cases.

Concerning the First Fit Heuristics (FF-Heur) results, we observe below-100% feasibility cases in the second and third datasets which suggests the existence of some blocked source-destination demands. Concerning the DNN trained with such FF-Heur RWCs, we observe poor results in the second and third datasets both concerning F-score values and feasibility. Essentially, the number of classes is too large (5 and 11 thousand respectively), thus making the DNN model not able to learn and generalize well (F-score values below 0.5). However, in the first dataset, where the Heuristic algorithm produces good results and the number of RWC labels is small enough (only 133 classes), the DNN trained with the Heuristic dataset provides outstanding results.

In conclusion, this experiment shows that DNN models can learn the patterns of both datasets, those generated after solving the RWA using an ILP or a First-Fit Heuristic. However, for the DNN to outperform, it is necessary that sufficient data is provided and the number of possible RWC classes is kept small, otherwise the ML model cannot accurately capture the patterns within the data.

D. DNN accuracy and loss per training step

This section aims at evaluating the accuracy and cross-entropy loss function of the DNN models for the 5-node and Abilene network topologies, on attempts to show that the DNN models have been well designed and trained.

Fig. 5 depicts the evolution of accuracy and cross-entropy loss for training (blue) and test sets (red) for the two network scenarios (5-node and Abilene). Each of the points in the x axis represents $2,000$ steps, where each step comprises a pass over a batch of training data (400 points batch size). The figure shows how the test curve flattens in the range between 10,000 and 20,000 steps while the training accuracy curve continues growing, suggesting that using more steps may cause the DNN overfit the data. Recall that while the accuracy is roughly near

TABLE II
NUMERICAL RESULTS OBTAINED FOR THE ABILENE NETWORK

DATASET #1	20@400, 600 wavelength-link resources										
	ML Metrics - F-score			Network Metrics							
Algo.	Train	Test	Val	avg_LL	max_LL	ave_hops	max_hops	wl_used	Feas train	Feas val	RWCs
ILP	-	-	-	0.025	0.421	2.640	6	349	100	-	-
$DNN(ILP)$	0.886	0.883	0.884	0.025	0.416	2.643	6	349	97.77	98.08	1,470
$FFHeur$	-	-	-	0.029	0.422	3.007	8	397	100	-	-
$DNN(FFHeur)$	0.987	0.984	0.979	0.028	1	3.007	8	397	99.92	99.86	133
DATASET #2	40@100, 1200 wavelength-link resources										
	ML Metrics - F-score			Network Metrics							
Algo.	Train	Test	Val	avg_LL	max_LL	ave_hops	max_hops	wl_used	Feas train	Feas val	RWCs
ILP	-	-	-	0.0594	1	3.064	7	413	100	-	-
$DNN(ILP)$	0.748	0706	0.712	0.057	1	3.056	7	413	94.67	94.65	1,775
$FFHeur$	-	-	-	0.047	1	2.503	5	334	97.25	-	-
$DNN(FFHeur)$	0.456	0.423	0.422	0.047	1	2.503	5	334	85.07	84.38	5,868
DATASET #3	20@100, 600 wavelength-link resources										
	ML Metrics - F-score			Network Metrics							
Algo.	Train	Test	Val	avg_LL	max_LL	ave_hops	max_hops	wl_used	Feas train	Feas val	RWCs
ILP	-	-	-	0.111	1	3.000	6	406	100	-	-
$DNN(ILP)$	0.771	0.731	0.725	0.110	1	3.003	6	405	95.55	94.87	1,462
$FFHeur$	-	-	-	0.093	1	2.503	5	332	84.33	-	-
$DNN(FFHeur)$	0.425	0.389	0.389	0.090	1	2.503	5	332	71.33	71.41	11,128

0.7, the feasibility of this model (in terms of network metrics) is over 90%, hence providing excellent performance results from a networking point of view.

E. Fast RWA reconfiguration time of ML models

Due to the definition of the classification problem and its ranking approach solution, whenever the ML classifier selects one feasible RWC from the top-10, that network setting can be directly implemented in the network. In case that none of the top-10 RWCs are feasible, then the ILP/heuristic is invoked for solving that particular TM. The previous experiments have demonstrated that the DNN model accurately selects feasible solutions in the first attempt for a large amount of cases and, therefore, reduces noticeably the RWA solution time with respect to the ILP or Heuristic.

In this light, we measure the total time employed by such a hybrid RWA approach taking into account both the cases where the ML produces a fast-and-feasible solutions combined with the cases where none of the top-10 RWC are valid and a slow invocation to the ILP or Heuristic needs to be performed. In mathematical terms, we use the following equation to measure the improvement with respect to the ILP by using a hybrid approach:

$$\text{Impr} = \frac{t_{\text{ILP}} - (N_{\text{feasible}} * t_{\text{ML}} + N_{\text{unfeasible}} * t_{\text{ILP}})}{t_{\text{ILP}}} \quad (5)$$

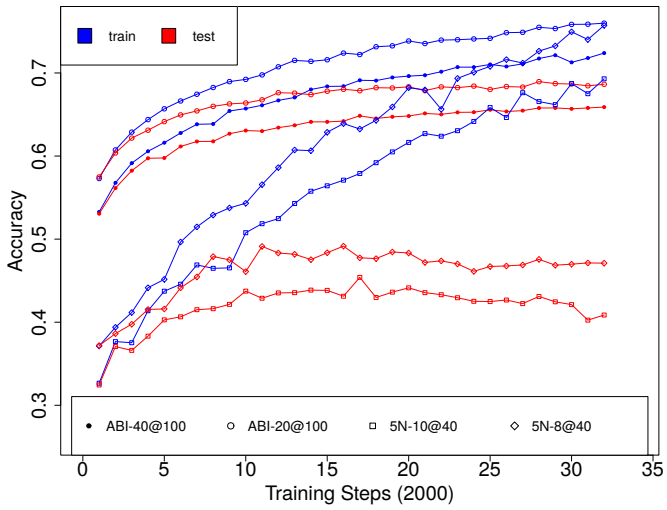
where t_{ILP} and t_{ML} correspond to the configuration time of the ILP and the ML prediction respectively and N_{feasible} (i.e. Feas val in Tables I and II) and $N_{\text{unfeasible}}$ determine the percentage of feasible/unfeasible cases.

Table III illustrates the completion times for training and prediction of the ML models and datasets, along with the time improvement values with respect to running the ILP in Net2Plan. Besides, the time proportion imputed to feasible (ML) and unfeasible (ILP) solutions is displayed as well. These values have been obtained by running the experiments on an Intel Xeon E5-2630 server with 24 cores and 190 GB of RAM memory. In particular, the ML algorithms execute TensorFlow for DNNs and Python's Scikit Learn for Logistic Regression while the ILP resolution of Net2Plan uses the CPLEX optimization toolbox [44].

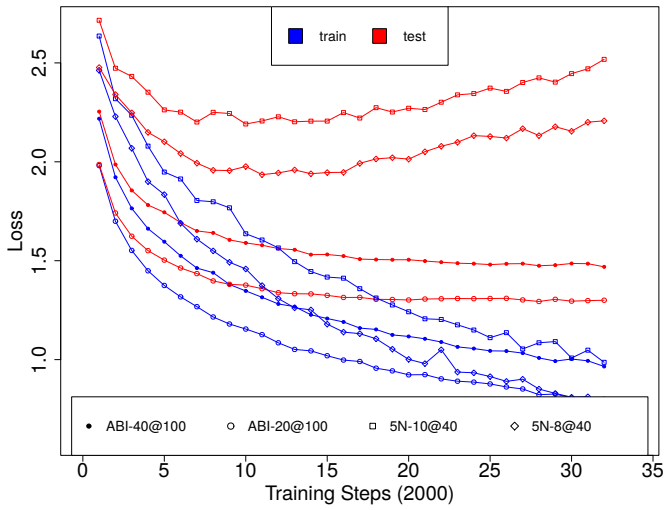
As shown in the table, linear models like LR are fast to train (seconds) and query for prediction (microseconds), while the DNN models require a few minutes to train and milliseconds for prediction (in line with Section III-F). Still, both approaches are well below the hundreds of milliseconds

TABLE III
TRAINING AND PREDICTION TIMES WITH AMOUNT OF TIME IMPROVEMENT IN A HYBRID ML-ILP/HEUR SETTING

5-Node ℓ_1 LR				
Dataset	Train.	t_{ML}	t_{ILP}	Impr.(ILP)
5@400	2.0 s	0.6 μs	478 ms	99.89%
5@100	8.4 s	1.6 μs	393 ms	99.29%
10@40	21.0 s	2.6 μs	436 ms	83.79%
8@40	19.6 s	2.3 μs	470 ms	81.99%
5-Node ℓ_2 LR				
Dataset	Train.	t_{ML}	t_{ILP}	Impr.(ILP)
5@400	2.2 s	0.7 μs	478 ms	99.89%
5@100	2.8 s	8.1 μs	393 ms	61.22%
10@40	2.8 s	3.7 μs	436 ms	80.99%
8@40	3.7 s	2.2 μs	470 ms	85.79%
5-Node ILP DNN				
Dataset	Train.	t_{ML}	t_{ILP}	Impr.(ILP)
5@400	15.1 $mins$	1.1 ms	478 ms	99.81%
5@100	15.5 $mins$	1.6 ms	393 ms	99.63%
10@40	16.5 $mins$	3.3 ms	436 ms	96.74%
8@40	16.3 $mins$	1.8 ms	470 ms	96.65%
Abilene ILP DNN				
Dataset	Train.	t_{ML}	t_{ILP}	Impr.(ILP)
20@400	17.2 $mins$	8.8 ms	861.8 ms	97.08%
40@100	16.5 $mins$	10 ms	1382.1 ms	93.96%
20@100	18.2 $mins$	3.5 ms	1899.2 ms	94.69%
Abilene Heuristic DNN				
Dataset	Train.	t_{ML}	t_{heur}	Impr.(heur)
20@400	12.3 $mins$	2.6 ms	256 ms	98.90%
40@100	21.5 $mins$	5.9 ms	237 ms	82.28%
20@100	33.6 $mins$	14 ms	262 ms	67.60%



(a) Accuracy



(b) Cross-entropy loss

Fig. 5. Accuracy and Cross-entropy loss for the 5-node and Abilene network topologies.

required by the ILP to optimally solve a given traffic matrix. In addition, linear ML models may be faster than DNNs but they often report unfeasible solutions requiring to run the ILP with subsequent penalties ; in this sense, the DNN is more accurate at providing feasible solutions so it is not often penalized with the computational time of running the ILP. In addition, the DNN model also shows 1 - 2 orders of magnitude faster times than the First-Fit Heuristic in the Abilene topology (see final rows of the Table).

Finally, it is worth remarking that the DNN training times can be further reduced with the use of specialised hardware (i.e. GPUs) or even training the DNN with an already close-to-optimal setup where the DNN architecture and weights are initialised from an existing past optimal configuration. In the experiments, both ILP, Heur and DNN have been initialised with random numbers for a fair comparison on the numbers.

F. Experiments with SDN's MLRC module on the 5-node topology

In order to test the ML models in an SDN context, we have developed a test-bed in which we test the proposed ML-based classification algorithms. First of all, as shown in Section III-G, we proceed with the acquisition of traffic matrices in order to collect a large enough historical dataset to start training the model. After that, when the training step is complete, we run the test and generalize the model to provide the RWA configurations for the 5-node network under consideration. The advantage of this approach is that we can keep training our model with new data and only when it passes the testing phase, it can be deployed and replace the previous one.

We exploited a virtualized lab created with Mininet [9]. Mininet is a network emulator that runs a collection of end-hosts, switches and links on a single Linux kernel. In Fig. 2 we show the 5-node network topology composed by 5 Open vSwitches[6] and 4 end hosts/servers.

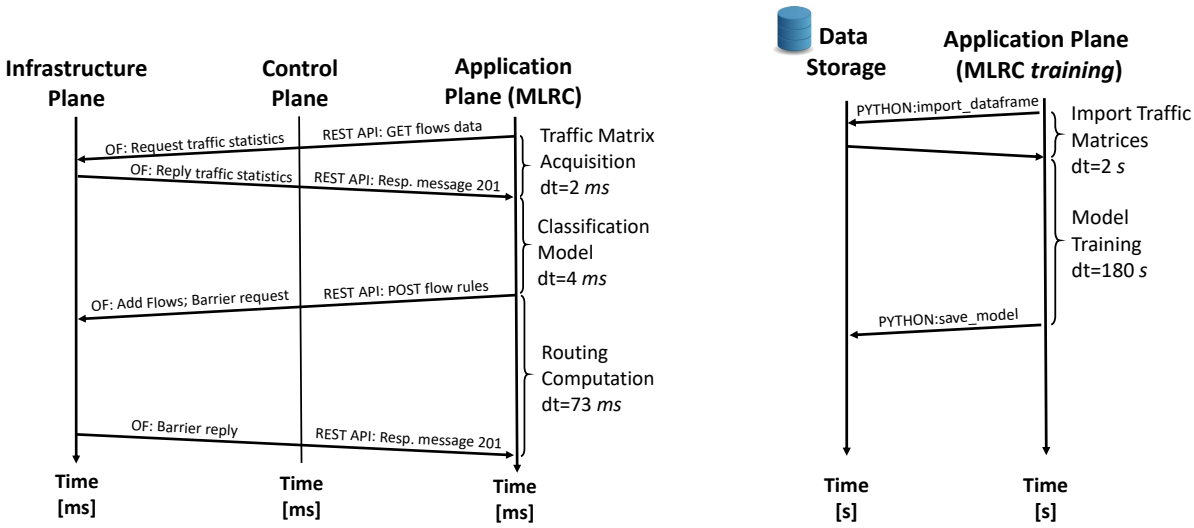
The control plane is deployed with the ONOS SDN controller [41] (version 1.12). Via the Northbound interface, it provides REST APIs to the application plane. ONOS is a carrier-grade SDN controller that consists of applications that manage several network functions, such as: host mobility, Packet-Optical integration, proxy ARP, etc. Moreover, the Southbound interface is used by the ONOS controller to implement communication with the infrastructure plane. We use the OpenFlow [45] protocol that gives access to the forwarding plane of the Open vSwitches. Furthermore, ONOS provides a graphical user interface (GUI) from which the network topology and installed routing paths can be observed.

Packet-based traffic is generated by the four hosts using the Distributed Internet Traffic Generator (D-ITG) tool [46]. D-ITG is a platform capable of producing packet-based traffic, emulating various stochastic processes for both IDT (Inter Departure Time) and PS (Packet Size) random variables (e.g., with Exponential, Uniform, Cauchy, Normal, Pareto, etc.). D-ITG supports both IPv4 and IPv6 traffic generation and it is capable to generate traffic at network, transport, and application layer.

To demonstrate the performance of dynamic reconfiguration of the routing plan given by the machine learning module, we run several experiments on the described testbed. Assuming that the ML model is continuously trained with the traffic matrices extracted from the network, it collected $2,000$ traffic matrices and provided the configuration for each one of them. A summary of the stages described in Section III-G and Fig. 2 (i.e. data acquisition, storage, classification model and routing computation) along with the ONOS and OpenFlows messages is depicted in Fig. 6. The MLRC app is a python module that uses Python's Pandas data frames to work with machine learning functions.

ML-based routing improves the ONOS Native Path Computation Module (ONPC), because in addition to proposing a routing scheme based on traffic history, it minimizes net-

[6]Open vSwitch is an open-source implementation of a distributed virtual multilayer switch. See *https://www.openvswitch.org/*, last access April 2019



(a) Interaction of MLRC with control and infrastructure plane

(b) Interaction of MLRC with data storage during the training phase

Fig. 6. MLRC in action: (a) Communication between the application and the control planes is done through the REST API exposed by ONOS, while the control plane and infrastructure planes use built-in OpenFlow protocol (b) Interaction between the MLRC application and the data storage during the training phase.

TABLE IV
MLRC AND ONPC MODULE TIMES PERFORMANCE

Machine Learning Routing Computation (MLRC) module performance	
Traffic Matrix Acquisition	∼2 ms
Classification Model	∼4 ms
Routing Computation	∼73 ms
Full Matrix Reconfiguration	∼80 ms
ONOS Native Path Computation (ONPC) module performance	
Full Matrix Reconfiguration	∼180 ms

work congestion in a dynamic way. Furthermore, the machine learning module takes about $80\ ms$ to perform the whole loop, i.e. to acquire the traffic matrix, obtain the configuration, and install the flow rules (see Table IV). Comparing the last result with the ONPC module performance, the MLRC module clearly enables real-time network reconfiguration.

V. SUMMARY AND DISCUSSION

In summary, this work has presented a machine learning-aware methodology to enhance optical WDM networks in three phases: *data generation*, *modeling* and *SDN implementation*.

For the data generation phase, we have presented NetGen, a scalable tool for the creation of networking labeled datasets. This tool wraps the normal functioning of the Net2Plan tool to scale and speed up its behavior.

Concerning Machine Learning, we have transformed the well-known *Routing and Wavelength Assignment* problem into a Supervised Learning problem that can be addressed using classical ML algorithms. In particular, we have trained logistic regression and Deep Neural Networks with a ground-truth dataset of thousands traffic matrices and their associated RWA solutions provided by the RWA ILP or First-Fit heuristic (the labels in our classification problem).

In general, DNNs provide useful non-linear learning structures applicable for learning RWA structures associated with traffic matrices in the context of an optical WDM network. For these DNNs to effectively learn to apply RWA configurations, they need to be fed with sufficient data examples and a reduced number of RWA classes to avoid the so-called curse of dimensionality.

Since the network scenario we are referring to is that of a long-haul WAN network with stable traffic patterns that may often suffer from sudden changes, the DNN is capable of learning the patterns of the training data offering outstanding classification results, providing almost perfect feasibility (that is, network applicability) for network configuration at a time cut of more than 93% with respect to classic ILP approaches. That ensures a vey high effectiveness of our approach in sparing total computational time comparing to ILP, without degrading the RWA performance. These results are supported with experiments that have considered a small five-node network with synthetic population-distance based traffic matrices, but also a large topology (Abilene) with real data measurements collected every five minutes for six months.

Finally, we have demonstrated how SDN and ML can come together through the *Machine Learning Routing Computation Module* (MLRC) to drive the provisioning of paths into an SDN Network. Using this module, we are capable of delivering the *Routing and Wavelength Configurations* obtained via ML into an SDN network rapidly. As a result, we have advanced in the combination of Machine Learning and SDN paradigms to facilitate traffic-aware and responsive networks, capable of reacting to changes very fast and implementing such configuration changes easily through SDN. Indeed, our results show how the required network configuration update time is considerably reduced with our approach.

VI. Acknowledgments

The authors would like to acknowledge the support of the project TEXEO (TEC2016-80339-R), funded by Spanish MINECO and the EU-H2020 Metrohaul project (grant no. 761727).

References

[1] F. Musumeci, C. Rottondi, A. Nag, I. Macaluso, D. Zibar, M. Ruffini, and M. Tornatore, "A survey on application of machine learning techniques in optical networks," *arXiv preprint arXiv:1803.07976*, pp. 1–21, 2018.

[2] J. Mata, I. de Miguel, R. J. Duran, N. Merayo, S. K. Singh, A. Jukan, and M. Chamania, "Artificial intelligence (ai) methods in optical networks: A comprehensive survey," *Optical Switching and Networking*, vol. 28, no. 1, pp. 43–57, 2018.

[3] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shariar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 16, pp. 1–99, 2018.

[4] M. Trevisan, I. Drago, M. Mellia, H. H. Song, and M. Baldi, "Awesome: Big data for automatic web service management in sdn," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 13–26, March 2018.

[5] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, and O. M. Caicedo, "Machine learning for cognitive network management," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 158–165, 2018.

[6] A. Mestres, A. Rodriguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett *et al.*, "Knowledge-defined networking," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 2–10, 2017.

[7] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrows intelligent network traffic control systems," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017.

[8] H. Zang, J. P. Jue, and B. Mukherjee, "A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks," *Optical Networks Magazine*, vol. 1, pp. 47–60, 2000.

[9] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 2012, pp. 253–264.

[10] P. Wang, S.-C. Lin, and M. Luo, "A framework for qos-aware traffic classification using semi-supervised machine learning in sdns," in *Services Computing (SCC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 760–765.

[11] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[12] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, "Network traffic classification using correlation information," *IEEE Transactions on Parallel and Distributed systems*, vol. 24, no. 1, pp. 104–117, 2013.

[13] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 4, pp. 1257–1270, 2015.

[14] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, 2017.

[15] J. Barron, M. Crotty, E. Elahi, R. Riggio, D. R. Lopez, and M. P. de Leon, "Towards self-adaptive network management for a recursive network architecture," in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 2016, pp. 1143–1148.

[16] S. Haeri, "Applications of reinforcement learning to routing and virtualization in computer networks," Ph.D. dissertation, University of Malaysia, 2016.

[17] T.-K. Hui and C.-K. Tham, "Adaptive provisioning of differentiated services networks based on reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 33, no. 4, pp. 492–501, 2003.

[18] B. Mao, Z. M. Fadlullah, F. Tang, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Transactions on Computers*, vol. 66, no. 11, pp. 1946–1960, 2017.

[19] F. Tang, B. Mao, Z. M. Fadlullah, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "On removing routing protocol from future wireless networks: A real-time deep learning approach for intelligent traffic control," *IEEE Wireless Communications*, 2017.

[20] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. Harold, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," in *IEEE Conf. on Computer Communications*. IEEE, 2018, pp. 1871–1879.

[21] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," 2018.

[22] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," *IEEE Network*, vol. 32, no. 2, pp. 92–99, 2018.

[23] K. Zheng, Z. Yang, K. Zhang, P. Chatzimisios, K. Yang, and W. Xiang, "Big data-driven optimization for mobile networks toward 5g," *IEEE network*, vol. 30, no. 1, pp. 44–51, 2016.

[24] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," *IEEE Network*, vol. 30, no. 3, pp. 52–58, 2016.

[25] C. Rottondi, L. Barletta, A. Giusti, and M. Tornatore, "Machine-learning method for quality of transmission prediction of unestablished lightpaths," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 2, pp. A286–A297, 2018.

[26] S. Shahkarami, F. Musumeci, F. Cugini, and M. Tornatore, "Machine-learning-based soft-failure detection and identification in optical networks," in *2018 Optical Fiber Communications Conference and Exposition (OFC)*. IEEE, 2018, pp. 1–3.

[27] H. Zang, J. P. Jue, B. Mukherjee *et al.*, "A review of routing and wavelength assignment approaches for wavelength-routed optical wdm networks," *Optical networks magazine*, vol. 1, no. 1, pp. 47–60, 2000.

[28] S. Q. Zhang, Q. Zhang, H. Bannazadeh, and A. Leon-Garcia, "Routing algorithms for network function virtualization enabled multicast topology on sdn," *IEEE Transactions on Network and Service Management*, vol. 12, no. 4, pp. 580–594, Dec 2015.

[29] M. Melo, S. Sargento, U. Killat, A. Timm-Giel, and J. Carapinha, "Optimal virtual network embedding: Node-link formulation," *IEEE Transactions on Network and Service Management*, vol. 10, no. 4, pp. 356–368, December 2013.

[30] F. Martinelli, N. Andriolli, P. Castoldi, and I. Cerutti, "Genetic approach for optimizing the placement of all-optical regenerators in wson," *IEEE/OSA Journal of Optical Communications and Networking s*, vol. 6, no. 11, pp. 1028–1037, 2014.

[31] Y. Pointurier and F. Heidari, "Reinforcement learning based routing in all-optical networks with physical impairments," in *Int. Conf. on Broadband Communications, Networks and Systems (BROADNETS '07)*, 2007.

[32] P. Pavon-Marino and J.-L. Izquierdo-Zaragoza, "Net2plan: an open source network planning tool for bridging the gap between academia and industry," *IEEE Network*, vol. 29, no. 5, pp. 90–96, 2015.

[33] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 215–242, 1958.

[34] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[35] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, pp. 427–437, 2009.

[36] A. F. A. C. N. Burlutskiy, M. Petridis and N. Ali, "An investigation on online versus batch learning in predicting user behaviour," in *Research and Development in Intelligent Systems XXXIII*. Springer, 2016.

[37] P. Orponen, "Computational complexity of neural networks: A survey," *Nordic J. of Computing*, vol. 1, no. 1, pp. 94–110, Mar. 1994. [Online]. Available: http://dl.acm.org/citation.cfm?id=640186.640192

[38] A. Blum and R. L. Rivest, "Training a 3-node neural network is np-complete," in *Proceedings of the First Annual Workshop on Computational Learning Theory*, ser. COLT '88. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988, pp. 9–18. [Online]. Available: http://dl.acm.org/citation.cfm?id=93025.93033

[39] L. Cao, "A computational viewpoint for deep learning," 2014, available at http://llcao.net/cu-deeplearning15/.

[40] S. Goel and A. R. Klivans, "Learning depth-three neural networks in polynomial time," *CoRR*, vol. abs/1709.06010, 2017. [Online]. Available: http://arxiv.org/abs/1709.06010

[41] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. M. Parulkar, "Onos: towards an open, distributed sdn os," in *HotSDN*, 2014.

[42] M. D. Vaughn and R. Wagner, "Metropolitan network traffic demand study," in *Lasers and Electro-Optics Society 2000 Annual Meeting. LEOS 2000. 13th Annual Meeting. IEEE*, vol. 1. IEEE, 2000, pp. 102–103.

[43] L. Melkumova and S. Y. Shatskikh, "Comparing ridge and lasso estimators for data analysis," *Procedia Engineering*, vol. 201, pp. 746–755, 2017.

[44] I. I. CPLEX, "V12. 1: Users manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[45] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[46] A. Botta, A. Dainotti, and A. Pescapé, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.