Automatic Management of N×N Photonic Switch Powered by Machine Learning in Software-defined Optical Transport

Received 14 April 2021; revised 28 May 2021; accepted 29 May 2021. Date of publication 7 June 2021; date of current version 16 June 2021.

Digital Object Identifier 10.1109/OJCOMS.2021.3085678

Automatic Management of $N \times N$ Photonic Switch Powered by Machine Learning in Software-Defined Optical Transport

IHTESHAM KHAN [1] (Graduate Student Member, IEEE), LORENZO TUNESI [1], MUHAMMAD UMAR MASOOD [1] (Graduate Student Member, IEEE), ENRICO GHILLINO [2], PAOLO BARDELLA [1] (Member, IEEE), ANDREA CARENA [1] (Senior Member, IEEE), AND VITTORIO CURRI [1] (Senior Member, IEEE)

[1] Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, 10129 Torino, Italy

[2] Synopsys, Inc., Ossining, NY 10562, USA

CORRESPONDING AUTHOR: I. KHAN (e-mail: ihtesham.khan@polito.it)

This work was supported by the Synopsys within the activities of a Research MSA with Politecnico di Torino.

ABSTRACT Optical networking is fast evolving towards the applications of the Software-defined Networking (SDN) paradigm down to the (Wavelength-division Multiplexing) WDM transport layer for cost-effective and flexible infrastructure management. Optical SDN requires each network element's software abstraction to enable full control by the centralized network controller. Nowadays, modern network elements, especially photonic switching systems, are developed by exploiting the fast-emerging technology of Photonic Integrated Circuit (PIC) that consists of complex fabrics of elementary units that can be driven individually using a large set of elementary controls. In this work, we focus on modeling the elementary control states of the topological structures behind PIC $N \times N$ switches under a fully blind approach based on Machine Learning (ML) techniques. The ML agent's training and testing datasets are obtained synthetically by software simulation of the photonic switch structure. The proposed technique's scalability and accuracy are validated by considering different dimensions N and applying it to two different switching topologies: the Honey-Comb Rearrangeable Optical Switch and the Beneš network. Excellent results in terms of prediction of the control states are achieved for both of the considered topologies.

INDEX TERMS Machine learning, optical switches, photonic integrated circuits, silicon photonics, microring resonators.

I. INTRODUCTION

THE EVER-INCREASING demand for global Internet traffic and evolving concepts of connectivity demand for flexible and dynamic networking at every layer. To obtain the required degree of flexibility, network elements and functions must be virtualized within the network operating system, implementing the SDN paradigm. With the introduction of coherent optical technologies for wavelength-division multiplexed optical transport and re-configurable optical switches for transparent wavelength routing, in optical networking, the SDN paradigm extends down to the physical layer [1], [2]. To pursue such an objective, optical network elements and transmission functions must be abstracted for quality-of-transmission (QoT) impairments and for controlling to enable full management by the optical control plane within the optical network controller [3], [4] as pictorially described in Fig. 1. This work focuses on the abstraction of control states of optical switches based on PICs with a structure-agnostic approach based on ML techniques.

Nowadays, *smart* optical network elements are progressively exploiting PICs to perform complex functions at the photonic level. In particular, in fiber-optics communications and data centers, large-scale photonic switches together with wavelength selective switches have a ubiquitous role, with primary merits due to their wide-band capabilities together

with low latency and low power consumption. Typically photonic switches are based on the principle that electrical control signals can maneuver the flow of light: using this mechanism, optical signals can be routed to different paths. Before the development of PIC solutions, different switching technologies have been proposed, such as three-dimensional Micro-Electro-Mechanical Systems (MEMS) [5] and beam-steering technique [6]. They both give stable optical switching and a reasonable degree of scalability, but the obligation for precise calibration and installation of discrete components makes them much more costly and bulkier. This increases the trend of using PICs-based components, especially photonic switches, which demands a generic softwareized control model for photonic switches' control states to enable full control by the single centralized controller.

PIC-based solutions mostly rely on elementary cells such as Mach-Zehnder Interferometers (MZI) [7] or optical Micro Ring Resonators (MRR) [8]. The generic $N \times N$ optical switch fabric is built by interconnecting multiple stages of elementary cells following a defined switching topology, where N input signals at different wavelengths can be routed to any of the N output ports by varying M control states. In scaling up the size of $N \times N$ switch fabrics, the fundamental requirement is to efficiently define the control states of the internal switches and obtain the requested signals' permutation at the output of the integrated circuit.

Currently, the research on control/routing states of the photonic switches has been sparsely reported. Unlike the electronic switches routing algorithms [9], where the performance of all paths are equal, the optical switches generally have path-dependent performance [10]. Variations in performance can be intrinsically due to the topology, or they can originate by fabrication and design defects, which may affect the elementary cells' different switching states and their cascading effect on the whole device. Deterministic routing algorithms can efficiently calculate the control state of the internal switches for any requested output permutation. The efficiency of these algorithms originates in their topology dependence, which allows for a faster and more effective evaluation of the multistage networks. In contrast, general-purpose routing algorithms do not provide scalable solutions, as the computational complexity increases rapidly [11], [12], [13]. This is caused by the exponential growth of the control states N_{st} in the network, which depends on the number of switches M as $N_{st} = 2^M$. This makes the generation and evaluation of the entire routing space unfeasible and evaluating the weighted penalties for all configurations.

In contrast with traditional topology-dependent strategies, we propose a generic data-driven model based on ML to predict the control states of any $N \times N$ photonic integrated switching system. ML strategies have already been tested in managing PICs. An algorithm powered by the artificial neural network is proposed in [14] to calibrate 2×2 dual-ring assisted-MZI switches. In [15], the author reported and experimentally demonstrated a full self-learning and reconfigurable photonic signal processor based on an optical neural network chip. The proposed chip performs various functions by self-learning, such as multi-channel optical switching, optical multiple-input-multiple-output descrambling, and tunable optical filtering. In [16], it is proposed to use the Deep Reinforcement Learning (DRL) technique to reconfigure silicon photonic Flexible Low-latency Interconnect Optical Network Switch (Flex-LIONS) according to the traffic characteristics in High-Performance Computing (HPC) systems. Furthermore, a novel reinforcement ML-based framework called DeepConf is introduced in [17], for automatically learning and implementing a range of data center networking techniques. Such a framework simplifies configuring and training of deep learning agents by using intermediate representation to learn different tasks.

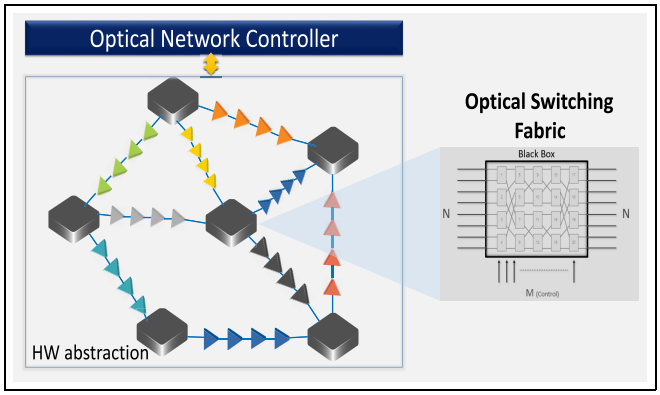


FIGURE 1. Abstraction of the optical switch in a SDN-controlled optical network.

In this work, we present a novel topology-agnostic *blind* approach exploiting an ML agent to predict the control states of the $N \times N$ photonic switch with an arbitrary and potentially unknown internal structure. The agent is trained by a dataset obtained by the component under test used as a black-box. The training dataset can be either obtained experimentally or *synthetically* by relying on a component software simulator.

Preliminary and partial results for this approach are presented in [18]. In this paper, besides describing in detail the methodology, we extend the framework towards the optimization of the ML models in terms of prediction, accuracy, and complexity, as well as verifying and analyzing the error distribution in the predicted control states. The error analysis aims at assessing the quantitative effects of the trained ML agent in predicting the proper internal switching routing, given the PIC topology. Furthermore, the future evolution of the proposed work will target the inclusion of transmission penalties in ML agent prediction to evaluate the impact on the QoT of channels processed by the switching system.

The remainder of the paper is organized as follows. In Section II, we describe the specific architecture of the Beneš and HCROS switches used for the demonstration of the proposed ML agent. In Section III, we describe the simulation environment used to generate datasets, presenting

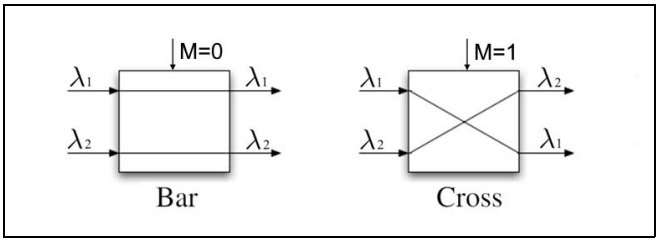


FIGURE 2. Illustration of Bar and Cross states of a 2 × 2 elementary switching element (CrossBar switch).

its structure and various statistics. Then, in Section IV, we describe the structure of the proposed ML agent, showing how it is trained on the datasets of different controls and output signals permutations in order to predict the control states of internal switching elements. In this work, we do not aim to develop a specific ML model; instead, our focus is to show the general effectiveness of ML in this scenario. So, we exploit an extensively tested opensource project, namely the TensorFlow© library [19]. Results of our approach are shown in detail in Section V. We demonstrate that the trained ML agent enables the correct estimation of the internal switching elements control states for different $N \times N$ sizes and topologies. We also show that a heuristically enhanced ML agent can further improve the predictions' accuracy in the present scenario. Finally, conclusions are presented in Section VI.

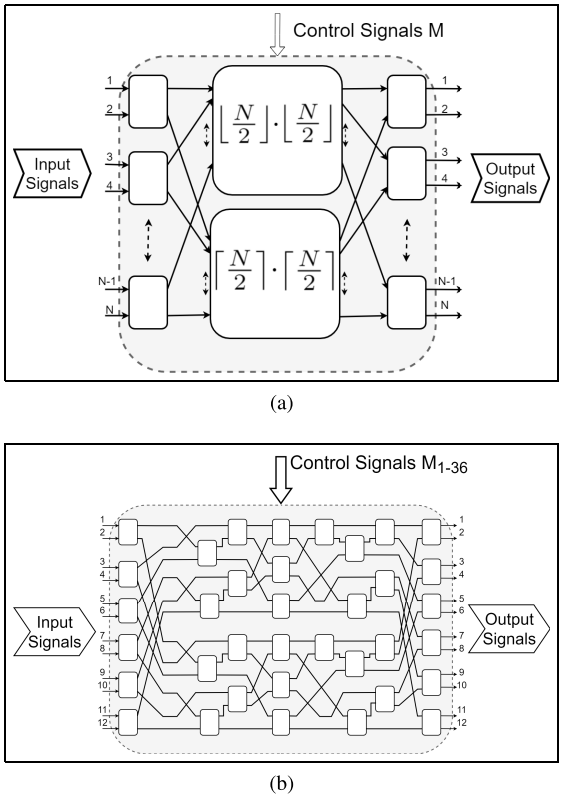
II. SWITCHING TOPOLOGIES

The switching networks analyzed to validate the proposed ML-based routing approach belong to a class of multistage crossover switches akin to the Banyan and Clos networks: these topologies are composed by several elementary 2×2 crossbar switches, arranged in multiple stages with variable interconnections, to route a generic number N of inputs to a required output configuration. From a topological point of view, the most important property for the optical application is to route each possible requested output permutation without internal conflicts.

Based on this property, the networks can be divided into two main classes: non-blocking and blocking, representing the ability to route all possible permutations of N inputs to the N output ports. For the scope of this analysis, only non-blocking networks have been analyzed, as they provide a more useful application and more complex topological characteristics with respect to the blocking networks.

A. THE 2 × 2 CROSSBAR SWITCH

A 2×2 CrossBar switch is the basic building block used for the generation of these networks. Such a device has two main states: the BAR state, where the inputs are directly routed to the output ports ($\begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix} \rightarrow \begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix}$), and the CROSS state, where the signals target ports are inverted ($\begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix} \rightarrow \begin{pmatrix}\lambda_2\\ \lambda_1\end{pmatrix}$), as shown in Fig. 2. These ideal switches can be implemented through different structures at the circuit level:



(a)

(b)

FIGURE 3. Topology of switching architectures: (a) Beneš network, (b) HCROS 12 × 12.

add-drop MRR and MZI can be used to design these devices for both colorless and wavelength-dependent systems.

B. THE N × N BENEŠ NETWORK

A Beneš network is a sub-type of Clos networks with 2×2 basic elements. By definition, the number of inputs is limited to $N = 2^k, k \in \mathbb{N}$ although it is possible to generalize its size to any number of inputs through a structure called Arbitrary Sized Beneš (AS-Beneš) [20]: the properties of the network are unchanged with respect to the constrained definition, so the generalization will be called Beneš as the strict-sense one (Fig. 3a). In the strict-sense definition of non-blocking network, new links can always be established without changing the previously set paths; the Beneš networks is a rearrangeable non-blocking structure, meaning that new input-output links can always be established (all output permutations are possible), but the existing links may need to be routed through different paths.

In terms of complexity, the Beneš networks can be analyzed through two main parameters:

- the number of unique output permutations, equal to $N!$.
- the number of different configurations of the network that grows exponentially with the number of switches as 2^M, with $M = N \log_2 N - \frac{N}{2}$ for strict-sense Beneš, while for AS-Beneš the formula must be evaluated recursively [20].

Due to the size mismatch between the number of unique output and network configurations, this class of topologies allows alternative routings for the same output permutations: the number of equivalent routings differs depending on the requested output permutation, with an average of $2^M/N!$ alternatives for each output. It is worth observing that deterministic algorithms exist for the Beneš network, which allows calculating a control configuration driving the requested output permutation [9], [21]. In this work, however, we consider the whole switch as a generic black box, regardless of the specific internal implementation, to validate the ML-based approach. To demonstrate the scalability of the proposed method with respect to the network complexity, three different instances of Beneš topologies were studied, with increasing size $N = 8,\ 10$ and 15. The corresponding number of 2×2 switching elements in each configuration is $M = 20,\ 26$ and 49, which represent the control vector size, or the labels of the ML agent.

C. THE HCROS CONFIGURATION

The HCROS is an alternative with respect to the Beneš network [22]. It maintains similar properties as non-blocking rearrangeability and number of switches, although showing a different topology without the horizontal symmetry and recursive structure, as shown in Fig. 3b. The 6×6 structure has been used to create a 12×12 switch, to achieve a dimension more suitable for the proposed analysis. The device is composed of $M = 36$ basic switching elements, comparable with the three Beneš switches studied. It offers a necessary benchmark for evaluating the ML-agent performance for less regular and recursive structures and, in particular, to demonstrate the capability to handle any switch structure.

III. SIMULATION & DATASET GENERATION

Datasets are needed for training and verification: they are obtained through an abstracted implementation of the previously discussed topologies. Each 2×2 switch element is driven by a control bit, with 0 representing the BAR state and 1 the CROSS state: each configuration of the network can be then described by a bit-array of length equal to the number of switches M. The considered structures are implemented as a cascade of permutation matrices, representing each switch and crossing stage. In multistage structures like the Beneš switch and HCROS, the network's output for a given control vector can be calculated applying the permutation matrices for each stage sequentially [9], without having the explore a more complex and computationally expensive graph structure.

In non-blocking network switch topologies, the M control states' variation typically generates 2^M total combination, whereas $N!$ is the number of distinct permutations of the N input signals, as shown in Fig. 4. For practical purposes, the dataset's size for training should be much smaller than the full-sized look-up table: if such a table could be evaluated, no other algorithm would be needed to route the inputs. Here, the training dataset is assembled from unique random control vectors to avoid any possible bias toward certain switch configurations or preferential paths within the network. The total train and test dataset for the considered Beneš networks and HCROS is a subset of the total 2^M control combinations, as reported in Table 1.

TABLE 1. Dataset statistics.

Network type Size ($N \times N$)	Beneš 8x8	Beneš 10x10	HCROS 12x12	Beneš 15x15
Permutations ($N!$)	40,320	3,628,800	479,001,600	$1{,}307\times10^9$
Switches (M)	20	26	36	49
Combinations (2^M)	1,048,576	67,108,864	68×10^9	562×10^{12}
Dataset	100,000	300,000	300,000	1,000,000

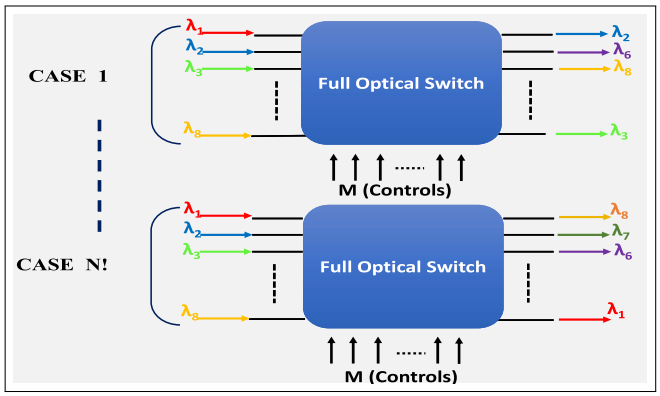


FIGURE 4. Graphical representation of all possible $N!$ output states for a $N \times N$ fabric.

After having trained the ML agent, the testing procedure is the following. For each of the randomly selected permutations of the test dataset's output channels, a corresponding sequence of control states is returned by the ML agent. This sequence is used to calculate the actual outputs permutation by using the simulator based on the permutation matrices' product. Comparing the requested output permutation with the one obtained using the predicted control states sequence, we can determine the ML agent's accuracy.

IV. MACHINE LEARNING FRAMEWORK

This section describes the proposed ML agent's structure and workflow trained on the generated dataset. We also describe the orchestration of the trained module specifying features, labels, and the additional configuration parameters of the ML engine.

The proposed ML-based technique considers the $N \times N$ photonic switch as a black-box, requiring a sufficiently large amount of training data to develop a cognitive model without considering the internal architecture. We select a Deep Neural Network (DNN) [23] as an ML algorithm since it is a powerful tool that has shown significant results in numerous frameworks like the one under investigation. Like all other supervised ML-based learning methods, in order to perform the training and prediction processes, the proposed model requires the definition of the features and labels representing the system inputs and outputs, respectively. The manipulated

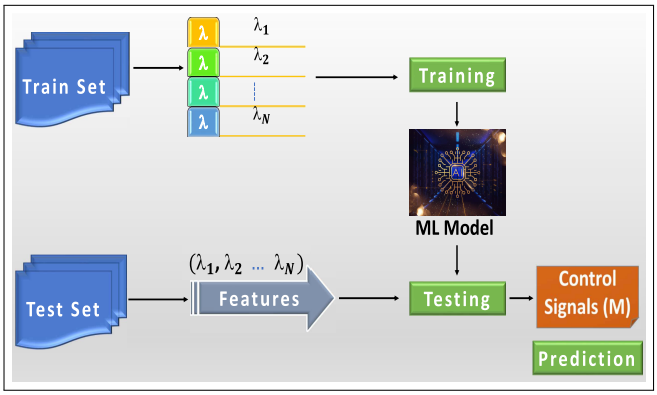


FIGURE 5. Description of the Machine Learning agent.

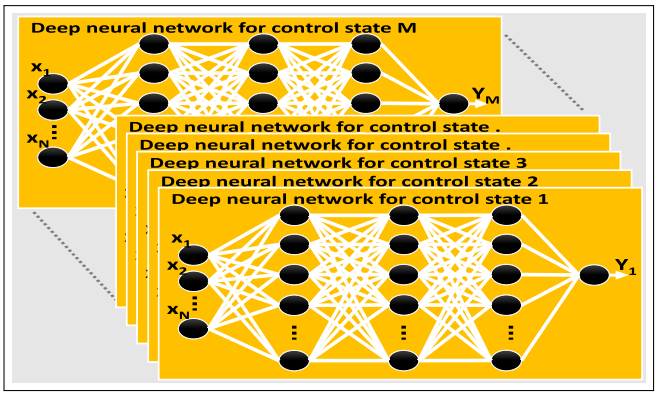


FIGURE 6. Parallel architecture of a deep neural network with three hidden layers.

features include the various permutations of the input signals (λ_1, λ_2, $\lambda_3 \cdots \lambda_n$) at the output ports of the switch, and it exploits its M control states shown in Fig. 5 as labels. The proposed DNN is developed by using higher-level APIs of the TensorFlow© platform [19], which provides various learning algorithms as well as appropriate functions to refine the dataset before using it as the model input.

The considered DNN is configured by several parametric values that have been optimized (such as the *training steps*, set to 1000), loaded with the *Adaptive Gradient Algorithm (ADAGRAD)* Keras optimizer, with *learning rate* set to 10^{-2} and L_1 regularization set to 10^{-3} [24]. Moreover, several non-linear activation functions such as *Relu, tanh, sigmoid* have been tested during the model building. After testing, *Relu* has been selected to implement DNN as it outperforms the others in terms of prediction and computational load [25].

Another important DNN parameter is the number of *hidden-layers*. The model has been tuned on several numbers of *hidden-layers* and neurons to achieve the best trade-off between precision and computational time. Although an increase in the number of layers and neurons improves the accuracy of the DNN up to a certain extent, a further increase in these values introduces diminishing returns that cause over-fitting while simultaneously increasing the computational time. After this trade-off analysis, we decided upon

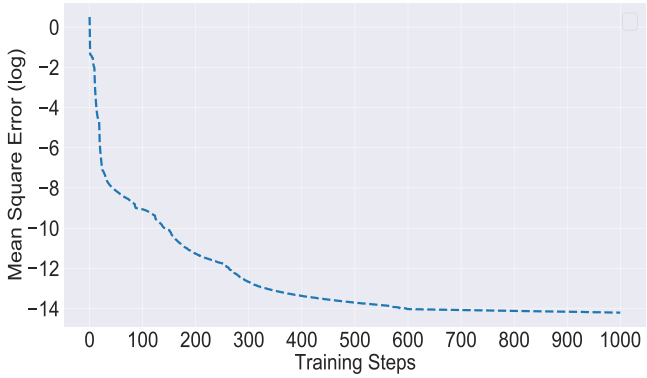


FIGURE 7. DNN loss function vs. the training steps for Beneš 8 × 8 architecture.

a DNN with *three hidden-layers* with several cognitive neurons for each hidden layer optimized for each dimension N. To improve prediction accuracy, we propose to use a parallel architecture for the DNN as shown in Fig. 6: in practice, we have an independent DNN for the prediction of each of the control states. The reason for using the parallel architecture of DNN is to give better cognition to the DNN engine and consequently achieve high efficiency in terms of prediction. Firstly, we performed the ML module training; after that, we tested the trained model on a separate subset of the dataset: the conventional rule of 70% and 30% has been chosen to partition the available dataset. The train set is 70% while the test set is 30% of the total considered dataset in Table 1. In order to avoid over-fitting the model, we set the *training steps* as the stopping factor and the *Mean Square Error* (MSE) as the loss function, given by:

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n}\left(\sum_{m=1}^{M}\left|\text{Ctrl State}_{i,\text{m}}^{p} - \text{Ctrl State}_{i,\text{m}}^{c}\right|\right)^{2} \quad (1)$$

where n is the number of test realizations, M is the total number of switching elements in the specific $N \times N$ switching system, while for each tested case i, $\text{Control State}_{i,\text{m}}^{p}$ and $\text{Control State}_{i,\text{m}}^{c}$ are the predicted and correct control states of the m-th switching element of the considered configuration. The MSE loss function, with respect to the training steps, is shown in Fig. 7 for the single considered case of 8×8 Beneš. Similar behavior is observed for all the other considered switching architectures. Once the desired accuracy of the model predictions has been reached; the trained ML agent can be used to predict control states of the switch.

V. VALIDATION RESULTS

In this section, we describe the results achieved for each considered Beneš size and HCROS regarding the predictions of control states. The validation has been carried out considering a dataset completely independent with respect to the one used in the training step. The first analysis we conducted is the prediction accuracy dependence on the training dataset's

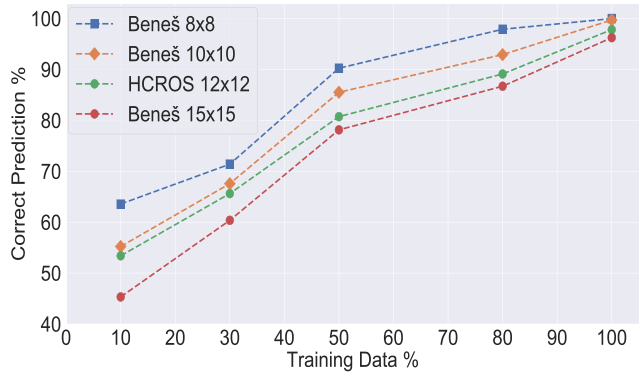


FIGURE 8. Percentage of correct predictions vs. normalized training dataset size. The normalization is performed with respect to the total generated dataset dimension for the considered $N \times N$ fabric, see data in Table 1.

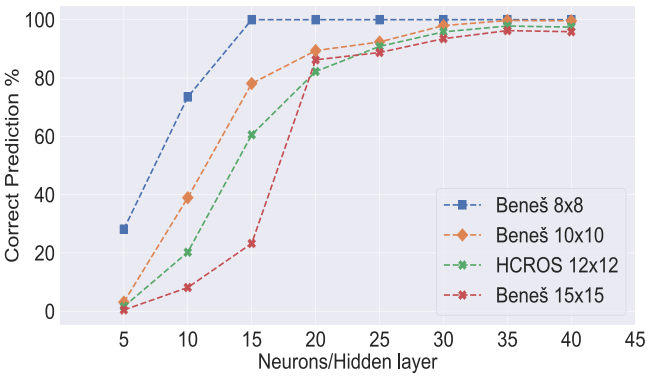


FIGURE 9. Percentage of correct predictions vs. hidden layer size for the considered switching configurations.

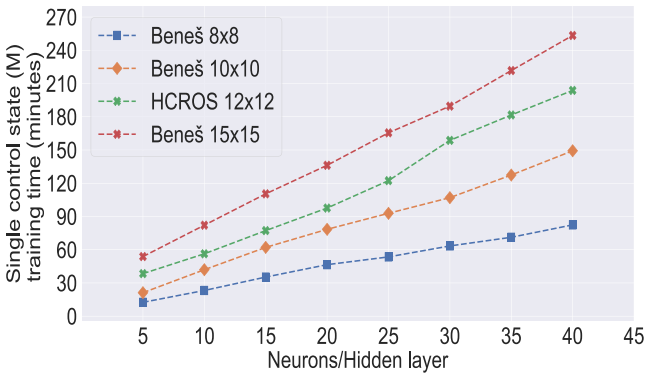


FIGURE 10. Single switch training time vs. hidden layer size.

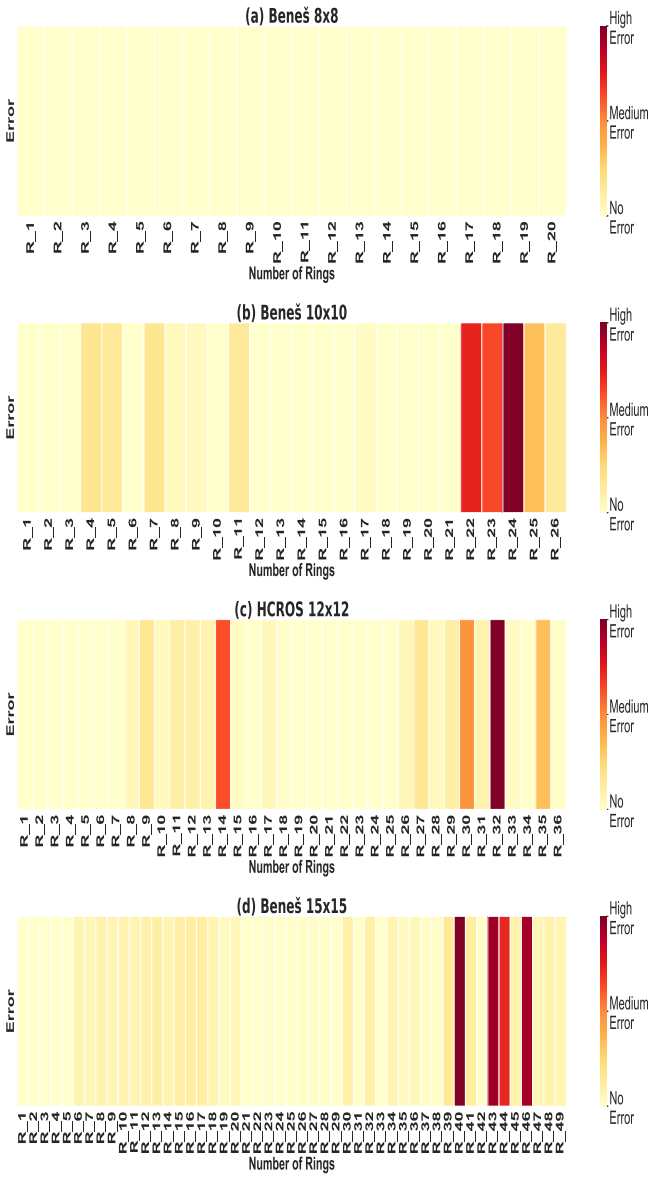


FIGURE 11. Heatmap showing normalized error in prediction of control states using DNN.

dimension and the size of hidden layers, as shown in Fig. 8 and Fig. 9.

In Fig. 8, the effect of increasing training dataset size is depicted. The trend shows that the prediction ability of the ML agent improves by increasing the training dataset size. It can be observed that a reduction in the dataset dimension induces a loss of performance. To achieve a good level of accurate predictions, we must use the whole dataset, whose size is indeed a small fraction of the total number of possible combinations of the switching states; see Table 1.

Similarly, in Fig. 9, the effect of increasing the number of neurons per hidden layer is shown: the prediction ability of the ML model improves when increasing the hidden layer size until the diminishing return trend is encountered, as explained previously. The minimum number of required neurons per layer depends on the configuration under analysis: the values selected for the following analysis are listed in Table 2.

The effect of increasing the hidden layer size on the training time is shown in Fig. 10, for an Intel Core i7 6700 3.4 GHz CPU workstation equipped with 32 GB of

TABLE 2. Summary of ML prediction results.

Network type Size ($N \times N$)	Beneš 8x8	Beneš 10x10	HCROS 12x12	Beneš 15x15
Neurons per hidden layer	15	35	35	35
Accuracy (no heuristic)	100%	99.72%	97.83%	96.25%
Single switch control error	0%	0.28%	2.17%	3.75%
Multiple switch control error	0%	0%	0%	0%
Accuracy (with heuristic)	100%	100%	100%	100%

Algorithm 1 Heuristic to Correct Single-Ring Errors

Require: *Topology Graph* $\mathcal{G}$*, Control States M, Number of Inputs/outputs N, Test set* $\mathcal{TS}$*, ML Predicted set* $\mathcal{PS}$
Ensure: *Control State Correction*
error-index = find control state where $\mathcal{TS} \neq \mathcal{PS}$
for all $\mathcal{D} \in$ error-index **do**
  $Predicted_{Control\ States\ (ON/OFF)}$ $(P_{Ctrl}) = \mathcal{PS}(\mathcal{D})$
  $Actual_{Control\ States\ (ON/OFF)}$ $(A_{Ctrl}) = \mathcal{TS}(\mathcal{D})$
  $Actual_{Output\ Signals}$ $(A_{otp}) = \mathcal{TS}(\mathcal{D})$
  $Predicted_{Output\ Signals}$ (P_{otp}) = Test-Control-States(P_{Ctrl},$\mathcal{G}$,N)
  $Check_{Output\ Signals}$ $(C_{otp}) = P_{otp}$
  for flip bit = 1 **to** M *Step* = 1 **do**
    if find $A_{otp} \neq C_{otp}$ **then**
      $Transit_{Control\ States\ (ON/OFF)}$ (T_{Ctrl}) = Flip-One-Bit(P_{Ctrl})
      C_{Ctrl} = Test-Control-States(T_{Ctrl},$\mathcal{G}$,N)
      Clear T_{Ctrl}
    else
      error-index corrected
      Break
    end if
  end for
end for

2133 MHz RAM. Results reveal that to train each of the M parallel DNN, i.e., a single control state, the time increases linearly with the number of neurons per hidden layer. Consider that even if training may be computationally demanding, with required times on the order of hours, this will not affect the ML agent's operation. Each prediction of control states can be obtained in real-time because of the negligible computational effort required once the agent is trained and it is compatible with the envisioned application in control-planes of open optical networks.

Finally, the rate of correct prediction for the optimized ML agents are summarized for the three considered Beneš sizes (8×8, 10×10 and 15×15) along with the 12×12 HCROS in Table 2. In the case of the Beneš network, we observe an excellent level of accuracy (>96%), although with reduced effectiveness of prediction when increasing N: correct predictions reach 100%, 99.72% and 96.25% for N equal to 8, 10 and 15, respectively. In order to further validate the proposed approach, similar results were obtained based on HCROS: also, here, we observe a high level of accuracy (>97%).

After evaluating the accuracy, we analyzed the distribution of errors in the predicted states, as shown in Fig. 11. The amount of errors in the control state prediction of each switching element when considering the validation dataset is encoded in color. We observe a non-uniform distribution: errors are concentrated on a small number of switch elements of the overall fabric. Based on this observation, we analyzed the number of wrong switch elements where the prediction fails. Results in Table 2 show that only a single error in one of the switches controls is responsible for the incorrect routing in wrong prediction instances. This is common for all the analyzed sizes of Beneš network as well as for the HCROS. Observing this phenomenon, we derived a simple heuristic that can further improve the prediction performance of our ML agent (see Algorithm 1).

The heuristic we propose requires several device properties such as topological graph ($\mathcal{G}$), M control states, and N number of inputs/outputs signals. Additionally, Test set ($\mathcal{TS}$) and ML Predicted set ($\mathcal{PS}$) are also loaded as inputs. The simple idea is to correct single switch errors by switching the control state of one element at a time while comparing the output sequence against the target output signals permutation. This approach requires only a maximum of M attempts, and this number is reasonably small, so it can be considered practical for real-time implementations. Moreover, it is topologically agnostic like the whole ML framework. For all four considered cases, the ML assisted by heuristic improves the accuracy to 100%.

VI. CONCLUSION

Photonic switching systems hold great importance for core optical transport and disaggregated data center networks since they provide high bandwidth, better reconfigurability, and high computing performance. In addition to these features, they present a low-cost, small footprint and high energy efficiency. To improve photonic switching systems' scalability, efficient routing strategies always stand out as a key challenge.

In this work, we analyzed a data-driven ML technique to control and manage photonic switching systems in an open optical networking context. The proposed scheme demonstrates a DNN based softwarized system that is both topological and technological agnostic and can be employed in real-time. The adopted ML approach can effectively determine the control states for a generic $N \times N$ photonic switch without requiring any topology knowledge. The presented ML approach is trained and tested assuming the $N \times N$ photonic switch as black-box: the ML only needs sufficient training instances without considering the device's internal architecture. The technique we propose is also scalable to larger input sizes N since a high level of accuracy can be reached with limited size datasets. Moreover, we have shown that a simple heuristic approach can increase the prediction accuracy to 100% with a marginal increase of the control state computational cost for the considered switching topologies.

REFERENCES

[1] C.-S. Li and W. Liao, "Software defined networks," *IEEE Commun. Mag.*, vol. 51, no. 2, p. 113, Feb. 2013.
[2] M. Jinno, T. Ohara, Y. Sone, A. Hirano, O. Ishida, and M. Tomizawa, "Elastic and adaptive optical networks: Possible adoption scenarios and future standardization aspects," *IEEE Commun. Mag.*, vol. 49, no. 10, pp. 164–172, Oct. 2011.

[3] V. Curri, "Software-defined WDM optical transport in disaggregated open optical networks," in *Proc. 22nd Int. Conf. Transparent Opt. Netw. (ICTON)*, Bari, Italy, 2020, pp. 1–4.

[4] L. Velasco *et al.*, "Building autonomic optical whitebox-based networks," *J. Lightw. Technol.*, vol. 36, no. 15, pp. 3097–3104, Aug. 1, 2018. [Online]. Available: http://jlt.osa.org/abstract.cfm?URI=jlt-36-15-3097

[5] J. Kim *et al.*, "1100 x 1100 port MEMS-based optical crossconnect with 4-dB maximum loss," *IEEE Photon. Technol. Lett.*, vol. 15, no. 11, pp. 1537–1539, Nov. 2003.

[6] A. N. Dames, "Beam steering optical switch," U.S. Patent 7 389 016, Jun. 17, 2008.

[7] K. Suzuki *et al.*, "Low-insertion-loss and power-efficient 32× 32 silicon photonics switch with extremely high-δ silica PLC connector," *J. Lightw. Technol.*, vol. 37, no. 1, pp. 116–122, Jan. 1, 2019.

[8] Q. Cheng *et al.*, "Ultralow-crosstalk, strictly non-blocking microring-based optical switch," *Photon. Res.*, vol. 7, no. 2, pp. 155–161, 2019.

[9] D. C. Opferman and N. T. Tsao-Wu, "On a class of rearrangeable switching networks part I: Control algorithm," *Bell Syst. Techn. J.*, vol. 50, no. 5, pp. 1579–1600, May/Jun. 1971.

[10] Y. Huang *et al.*, "Multi-stage 8 × 8 silicon photonic switch based on dual-microring switching elements," *J. Lightw. Technol.*, vol. 38, no. 2, pp. 194–201, Jan. 15, 2020. [Online]. Available: http://jlt.osa.org/abstract.cfm?URI=jlt-38-2-194

[11] M. Ding, Q. Cheng, A. Wonfor, R. V. Penty, and I. H. White, "Routing algorithm to optimize loss and IPDR for rearrangeably non-blocking integrated optical switches," in *Proc. Conf. Lasers Electro-Opt. (CLEO)*, 2015, pp. 1–2.

[12] Y. Qian *et al.*, "Crosstalk optimization in low extinction-ratio switch fabrics," in *Proc. Opt. Fiber Commun. Conf.*, 2014, pp. 1–3.

[13] Q. Cheng *et al.*, "Silicon photonic switch topologies and routing strategies for disaggregated data centers," *IEEE J. Sel. Topics Quantum Electron.*, vol. 26, no. 2, pp. 1–10, Mar./Apr. 2020.

[14] W. Gao, L. Lu, L. Zhou, and J. Chen, "Automatic calibration of silicon ring-based optical switch powered by machine learning," *Opt. Exp.*, vol. 28, no. 7, pp. 10438–10455, Mar. 2020. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-28-7-10438

[15] H. Zhou, Y. Zhao, X. Wang, D. Gao, J. Dong, and X. Zhang, "Self-learning photonic signal processor with an optical neural network chip," 2019. [Online]. Available: arXiv:1902.07318.

[16] R. Proietti, X. Chen, Y. Shang, and S. J. B. Yoo, "Self-driving reconfiguration of data center networks by deep reinforcement learning and silicon photonic flex-lion switches," in *Proc. IEEE Photon. Conf. (IPC)*, Vancouver, BC, Canada, 2020, pp. 1–2.

[17] S. Salman, C. Streiffer, H. Chen, T. Benson, and A. Kadav, "DeepConf: Automating data center network topologies management with machine learning," in *Proc. Workshop Netw. Meets AI ML*, 2018, pp. 8–14. [Online]. Available: https://doi.org/10.1145/3229543.3229554

[18] I. Khan *et al.*, "Machine-learning-aided abstraction of photonic integrated circuits in software-defined optical transport," in *Proc. Next Gener. Opt. Commun. Compon. Sub-Syst. Syst. X*, vol. 11713, 2021, pp. 146–151. [Online]. Available: https://doi.org/10.1117/12.2578770

[19] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

[20] C. Chang and R. Melhem, "Arbitrary size Benes networks," *Parallel Process. Lett.*, vol. 7, no. 3, pp. 279–284, 1997.

[21] A. Chakrabarty, M. Collier, and S. Mukhopadhyay, "Matrix-based nonblocking routing algorithm for Beneš networks," in *Proc. Comput. World Future Comput. Serv. Comput. Cogn. Adapt. Content Patterns*, Athens, Greece, 2009, pp. 551–556.

[22] M. R. Yahya, N. Wu, G. Yan, T. Ahmed, J. Zhang, and Y. Zhang, "Honeycomb ROS: A 6 × 6 non-blocking optical switch with optimized reconfiguration for ONoCs," *Electronics*, vol. 8, no. 8, p. 844, Jul. 2019.

[23] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[24] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Jan. 2011.

[25] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018. [Online]. Available: arXiv:1811.03378.