Machine Learning Driven Model for Software Management of Photonics Switching Systems

Machine Learning Driven Model for Software Management of Photonics Switching Systems

Ihtesham Khan
DET, Politecnico di Torino, Italy
ihtesham.khan@polito.it

Lorenzo Tunesi
DET, Politecnico di Torino, Italy
tunesi.lorenzo@gmail.com

Muhammad Umar Masood
DET, Politecnico di Torino, Italy
muhammad.masood@polito.it

Enrico Ghillino
Synopsys, Inc., 400,United States.
enrico.ghillino@synopsys.com

Paolo Bardella
DET, Politecnico di Torino, Italy
paolo.bardella@polito.it

Andrea Carena
DET, Politecnico di Torino, Italy
andrea.carena@polito.it

Vittorio Curri
DET, Politecnico di Torino, Italy
curri@polito.it

***Abstract*—Modern elastic optical networking requires additional flexibility at each layer compared to the traditional approach. The application of the Software-defined Networking (SDN) paradigm can provide the required degrees of freedom. The implementation of optical SDN down to the physical layer requires the complete abstraction of network elements to support full control by the centralized controller. In this work, we propose a topological and technological agnostic model based on Machine Learning (ML) to abstract the behavior of optical switches for the computation of Quality-of-transmission (QoT) penalties and the definition of control states. Training and testing datasets are obtained synthetically by software simulation of the photonic switching structure. Results show the capability of the proposed method to predict QoT impairments with high accuracy, and we envision its application in a real-time control plane.**

***Index Terms*—Machine Learning, Optical Switches, Photonic Integrated Circuits, Software-defined Networking, Quality-of-transmission.**

I. Introduction

The rapid increase in internet traffic due to bandwidth-intensive applications and the latest developing concepts of the Internet of Things (IoT) require higher degrees of flexibility at each network layer. The implementation of SDN has the potential to provide them effectively. Adopting the SDN paradigm enables the complete virtualization of network elements and functions inside the network operating system. Moreover, technologies like coherent optical techniques for Wavelength-Division Multiplexed (WDM) optical transport and re-configurable optical switches for transparent wavelength routing pave a path to extend SDN applications down to the physical layer [1]. To achieve SDN implementation down to the physical layer, network key elements and transmission functionalities must be abstracted for QoT penalties and control states. This abstraction empowers the optical network controller to fully manage network elements, and transmission functionalities [2].

Network components are increasingly utilizing Photonic Integrated Circuits (PICs) to execute different complex operations. Specifically, in the latest *smart* optical networks and data centers, large-scale photonic switches and wavelength selective switches play a prominent role due to their wide-band capabilities, minimal latency, and low power consumption. These distinctive properties increase the possibilities of using PICs-based network elements, especially photonic switches, and hence they generate a demand for a generic softwarized model for control states and QoT degradation to enable full control by a centralized controller, as shown in **Fig. 1**.

At present, the investigation related to the softwarized control of the photonic switching system has been sparingly registered. In contrast to electronic switches [3], where the performance of all routes are identical, the optical switches generally have a path-dependent performance [4]. The variations in the performance are mainly due to the photonic circuit topology, or they can depend on mask-level fabrication and design flaws. Usually, the deterministic routing algorithms presented in the literature can efficiently determine the control state of the internal switches for any given output permutation. The effectiveness of these algorithms comes from their topology dependent nature, which enables a faster and efficient assessment of the multiple-stage networks. On the contrary, generic routing algorithms do not offer scalable solutions, as the computational complexity rises quickly, mainly due to the exponential growth of the control states N_{st}, which eventually depends on the number of switches M as $N_{st} = 2^M$ in the given topology [5]–[7].

In contrast with conventional topology-dependent schemes, we propose a generic model based on ML to obtain the softwarized control of any $N \times N$ photonic switching system. The proposed ML technique has already been well assessed for managing PICs [8], in which a neural network is proposed to calibrate 2×2 dual-ring switches. In [9], [10], ML techniques are used to deliver an augmented knowledge of the physical parameters of integrated circuits. In [11], a deep reinforcement learning technique is proposed to reconfigure silicon photonic switch according to the traffic profile in high-performance computing systems.

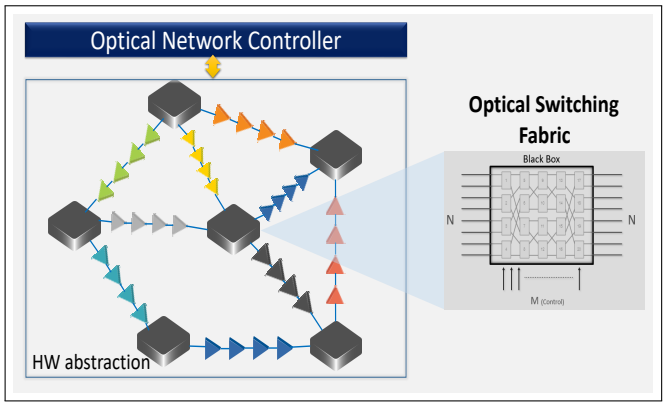


Fig. 1: Abstraction of the optical switch in a SDN-controlled optical network.

In this work, we extend our previous demonstration of the definition of the control states of a PIC $N\times N$ photonic switching system with a completely topology-agnostic *blind* solution exploiting an ML inverse design approach [12]. To complete the switch model, for a full description of the impact on the physical layer, we pair up an extra ML network with a direct design method to predict the QoT degradation due to the switching element. The two ML networks will work synergically and provide a generic softwarized and QoT aware control and management system for any $N\times N$ optical switch. The proposed abstracted model can be easily extended to measure the impact of $N\times N$ optical switch on the network layer metrics. The presented data-driven scheme is trained by a dataset obtained by considering any $N\times N$ photonic switch under test as a black-box. The training dataset can either be acquired experimentally or *synthetically* by using a software simulator for components.

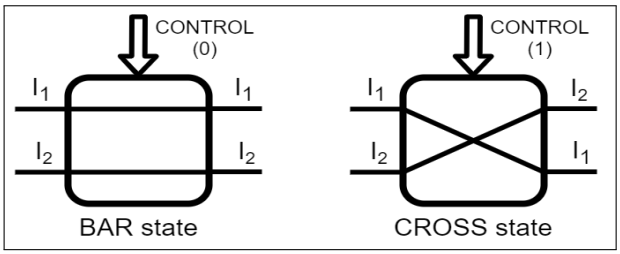
II. Elementary Switching Module and Topologies

The target devices for the proposed strategy are multistage crossover switching architecture based on the 2×2 crossbar switch. These devices allow input-output routing through a cascade of multiple stages of elementary switching elements, overall reducing the number of such devices and the overall footprint with respect to general N×N crossbar switching systems.

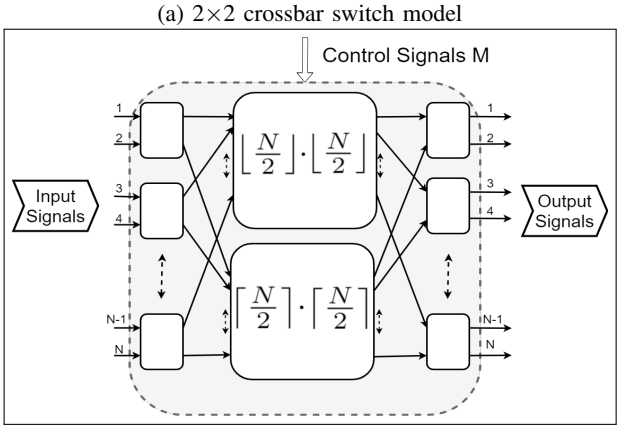
A. 2×2 *crossbar switch*

A typical base element used for generating multistage topologies for optical application is the 2×2 crossbar switch. The device state is piloted by a control signal $V_{control}$, with two possible routing states: in the BAR state, the input signals are routed straightforwardly to the output ports, while in the CROSS state, the signal order at the output is inverted, as shown in **Fig. 2a**.

These devices can be implemented through different PICs depending on the required transmission parameters and format. The two most prominent solutions are the Micro Ring Resonator (MRR) filter and the Mach-Zehnder Interferometer (MZI). In this paper, simulations of the system performance



(a) 2×2 crossbar switch model



(b) Generic Beneš network

Fig. 2: Model for the components and topology

are based on a second-order MRR filter, designed to operate in the center of the C-band and developed in the Optsim© environment. In contrast, the routing evaluation and logical channel transmission are based on the elementary black-box model of the device, which acts as an ideal 2×2 switching element.

B. *Multistage switching architectures*

Having defined the fundamental element for the switching operation, the generic $N\times N$ switch can be constructed according to the chosen topological structure, which dictates the transmission behavior and performance of the overall device. The main class of structures under analysis consists of rearrangeable non-blocking networks. These devices can route the N input signal into any ordered permutation at the output, following a bijective relationship, as each signal is routed to a different unoccupied port. In a strict-sense non-blocking network, this property upholds even when traffic already occupies parts of the switch. At the same time, in rearrangeable topologies, the previously established connections may need to be routed through different input-output paths.

This constrained class of devices is widely implemented in optical networks due to the smaller footprint and overall smaller number of fundamental elements, which allows a reduction of both costs and power consumption. One of the most common architectures belonging to this subset of topologies is the Beneš switch, which can be constructed under a recursive definition as shown in **Fig. 2b**. The Beneš network has been chosen for the analysis due to its reduced number of elements with respect to other rearrangeable non-blocking structures, such as the Spanke-Beneš or Multi-

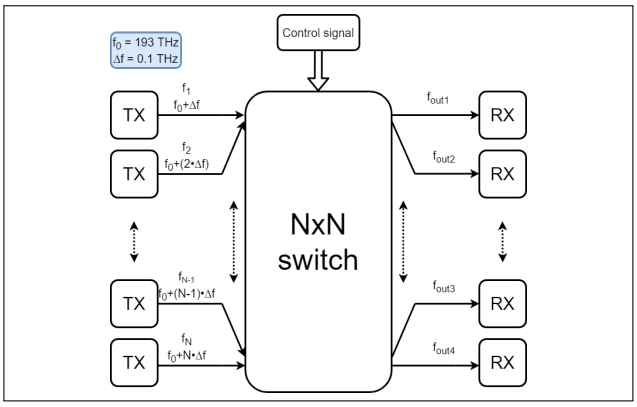


Fig. 3: Generic $N \times N$ transmission model

Butterfly while maintaining the conflict-avoidance property.

C. Beneš switching networks

The Beneš multistage switch is characterized by a number of switching elements $M = N \cdot \log_2(N) - \frac{N}{2}$ for $N = 2^x \quad x \in \mathbb{N}$, but can be generalized for any number of input signals $N \in \mathbb{N}$: the number of routable output permutations, due to the non-blocking property, is equal to $N_{out} = N!$, while the number of switches configurations are equal to $N_{states} = 2^M$. The exponential nature of the number of configuration limits severely the scalability of any brute-force approach, such as complete look-up tables, to pilot and control the routing requests in the device.

Topology-dependent path-finding algorithms can be employed to solve a given configuration request, as illustrated in [13], although this solution cannot be generalized to arbitrary switching structures due to the scalability issues of topology-agnostic algorithms. Moreover, these devices typically allow alternative routing for a given output request, which coupled with the path-dependency of the performance leads to optimality issues for the solutions obtained through deterministic methods

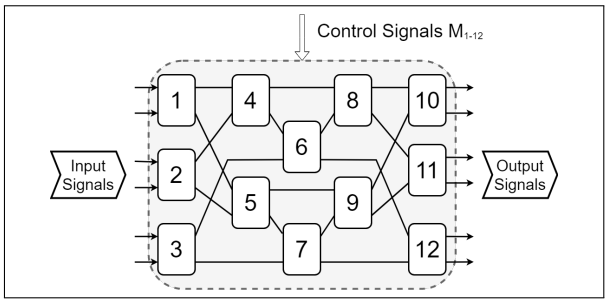
III. Simulation Model And Dataset Acquisition

The Beneš devices used for the data generation and training of the ML engine have been modeled following two main strategies. Concerning the routing evaluation, the black-box model for the fundamental 2×2 element is sufficient for the path evaluation, as the input-output link can be represented as an edge in a suitable graph structure. For the evaluation of the QoT, the logical model is insufficient; as such, the device has been simulated, considering an MRR-based implementation of the crossbar elements.

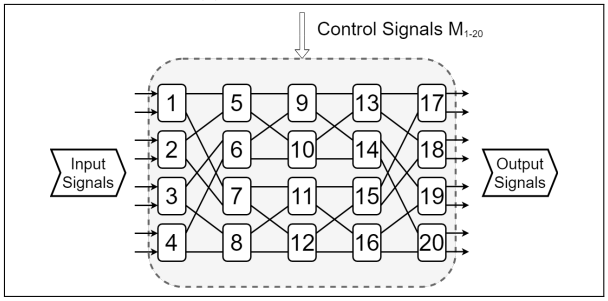
A. Routing model

For the evaluation of the routing states inside the device, the Beneš network has been implemented through a matrix representation composed by cascading the permutation vectors of each switching stage.

The output permutation can be obtained as a function of a given control state, providing the BAR and CROSS



(a) 6×6 Beneš switch



(b) 8×8 Beneš switch

Fig. 4: Circuit representation for the two device under test

TABLE I: Dataset Statistics

Network type **Size** ($N \times N$)	**Beneš 6x6**	**Beneš 8x8**
Permutations ($N!$)	720	40,320
Switches (M)	12	20
Combinations (2^M)	4096	1048576
Dataset	1000	1000

configuration for each of the M elements. The data-sets obtained through this approach are composed by a binary control vector $V \in \mathbb{R}^{1,M}$, with $V_i = 0$ representing the BAR state and $V_i = 1$ the CROSS alternative configuration, while the output configuration is represented by a permutation vector of size N. This logical model is also fundamental in the verification step: the predicted control states are evaluated directly on the abstracted device, as to verify the correctness of the ML solution, taking into account the alternative routing states for the required configuration.

B. Transmission model

In order to gather data concerning the QoT, the device must be simulated with an higher degree of realism, taking into account both physical design of the components and transmission format.

The crossbar has been modeled as a second order MRR switch, and following the recursive definition, the simulation-ready model was created in the Optsim© environment [14]. The ingress and egress stages of the device have been connected to a transceiver and receiver module, respectively, as shown in **Fig. 3**, allowing QoT simulations under a realistic modulation format and transmission characteristics. The system has been tested under a PM-64-QAM modulation format,

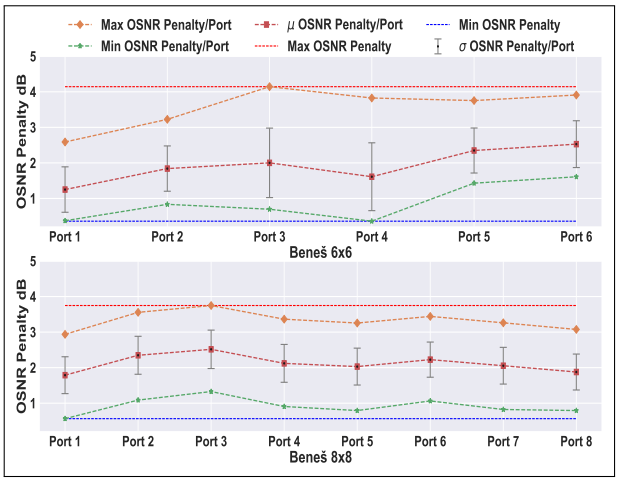


Fig. 5: Statistical Analysis of OSNR Penalty

with central frequencies $f = (193.1 + 0.1 \times x)$ THz for $x \in [1, N]$ and symbol rate $R_S = 50 \times 10^9$.

The training and validation sets have been generated by evaluating the OSNR penalty for random unique state configurations, considering a target Bit-Error Rate (BER) of $\text{BER}_{\text{th}} = 5 \times 10^{-3}$. OSNR penalties have been obtained by simulating the propagation trough the component under test of the considered signal [15]. The generated datasets contain the penalty at each port of the device for a random set of 1000 realization of control states (see **Table I**): this simulation has been performed on two different Beneš structures, namely the 6×6 and 8×8 configurations as depicted in **Fig. 4**.

The detailed statistical analysis of the acquired OSNR penalty of both the considered architectures is reported in **Fig. 5**. Observing the OSNR penalty statistics in **Fig. 5**, we can see that the worst-case scenario reaches about 4 dB, as reported with a red dotted line. We can conclude that an OSNR penalty equal to this maximum penalty should be considered for all states to prevent any switching configuration from being out-of-service without any prior knowledge. This value is the same for both the considered topologies. From **Fig. 5**, we can also observe that the average OSNR penalty that can satisfy most of the cases is much lower, 1.93 dB for Beneš 6x6 and 2.12 dB for Beneš 8x8, respectively.

IV. Inverse and Direct Machine Learning Modeling

The proposed photonic switch abstraction considers two ML networks. The first network, the ML Routing Agent, is intended to define the control state of the switch through an inverse design approach. In contrast, the second network, the ML QoT Agent, takes the first network's output and predicts, through a direct design approach, the estimation of QoT Penalty. This allows the network controller to evaluate the optimal solution of any $N \times N$ photonic switch considered as a black-box shown in **Fig. 6**

A Deep Neural Network (DNN) [16] is considered as a cognition engine for both the proposed ML networks since it is a powerful tool that has shown significant results in

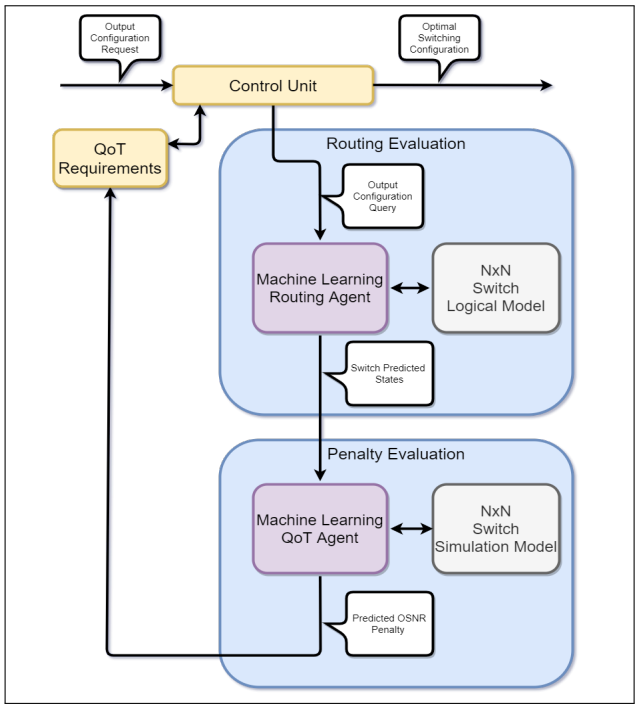


Fig. 6: Control Unit block model

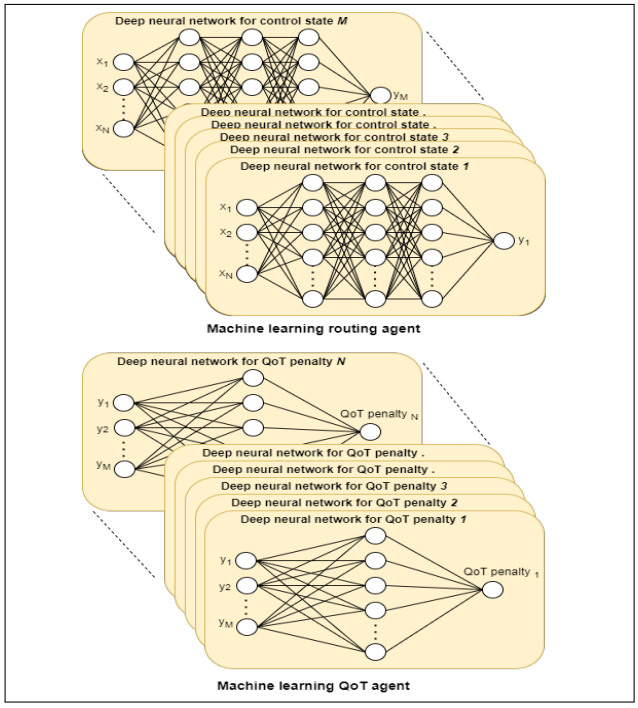


Fig. 7: Parallel architecture of a deep neural network

numerous frameworks. The proposed DNN is developed by using a higher-level Application Programming Interface (API) of the TensorFlow© platform [17]. To improve the prediction efficiency, we propose for both networks to use a parallel architecture for the DNN as shown in **Fig. 7**. The parallel DNN engines of both networks are trained and tested on a separate subset of the dataset: the conventional rule of 70% and 30% has been chosen to partition the available dataset. In

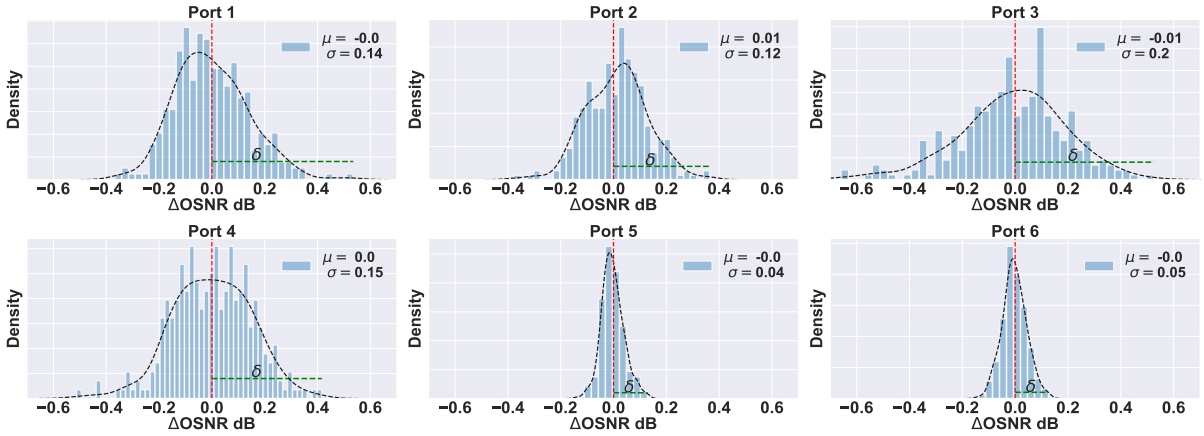


Fig. 8: Probability density functions of ΔOSNR for each port of the 6x6 Beneš switch.

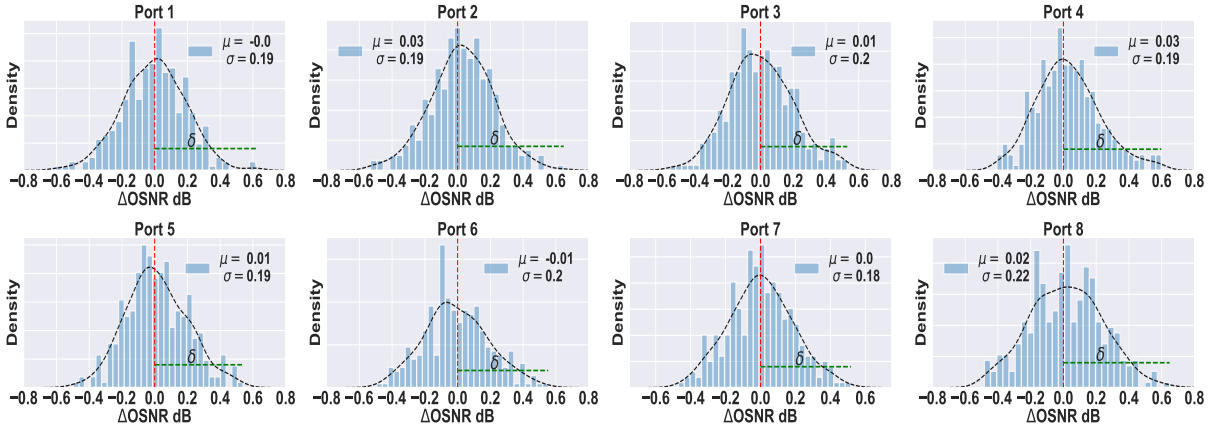


Fig. 9: Probability density functions of ΔOSNR for each port of the 8x8 Beneš switch.

order to avoid over-fitting of the models, the *training steps* are set as the stopping factor and the *Mean Square Error* (MSE) as the loss function as defined in Eq.1 and Eq.2, respectively for the first and second network.

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

$$\text{QoT MSE} = \frac{1}{n}\sum_{i=0}^{n}\left(\frac{1}{N}\sum_{k=1}^{N}\left(\text{OSNR Penalty}_{i,\text{k}}^{p} - \text{OSNR Penalty}_{i,\text{k}}^{a}\right)^2\right) \tag{2}$$

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 $\text{Control State}_{i,\text{m}}^{a}$ are the predicted and actual control states of the m-th switching element of the considered configuration. Similarly, N is the total number of input/output ports of the specific $N\times N$ switching system and $\text{OSNR Penalty}_{i,\text{k}}^{p} - \text{OSNR Penalty}_{i,\text{k}}^{a}$ are the predicted and actual OSNR penalty of the k-th output port of the considered topology.

Furthermore, the DNN engines of both networks are configured by several common 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}. 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 [18].

The ML Routing Agent considers various permutations of the input signals (λ_1, λ_2, λ_3....λ_n) at the output ports of the switch as features while it exploits its M control states as labels. The inverse model has been configured on considerable 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 over-fitting and increases the computational time. Following this trade-off assessment, we opted upon a DNN with *three hidden-layers* with 10 and 15 cognitive neurons for each hidden layer optimized for Beneš 6x6, and Beneš 8x8, respectively.

The ML QoT Agent considers the output of the first ML network (ML Routing Agent), i.e., the M controls states. At the same time, the utilized response variable is the OSNR penalty of the specific output port of the $N\times N$ switching system. The direct model has also been configured on considerable numbers of *hidden-layers* and neurons to achieve the best trade-off between precision and computational time. Following this trade-off assessment, we opted upon a DNN

with *one hidden-layers* with 11 and 18 cognitive neurons for Beneš 6x6, and Beneš 8x8, respectively.

V. Results And Discussion

We analyzed the performance of the proposed ML modules using a two-steps approach: first, we predicted the switch control states, and then we exploited this result to obtain the QoT impairments in terms of $\mathrm{OSNR\ Penalty}_{i,\mathrm{k}}$ for each port k of the considered Beneš network.

The ML Routing Agent gives an excellent level of accuracy in terms of predicting the control states. The agent gives 100% accuracy for both of the considered Beneš topologies N equal to 6 and 8. The scalability and detailed analysis of ML Routing Agent is reported in detail in [12].

For the ML QoT Agent, the predicted control states are given as an input to get the QoT penalty. The metric used to evaluate the accuracy of the QoT module is defined as:

$$\Delta OSNR_{i,\mathrm{k}} = \mathrm{OSNR\ Penalty}^{a}_{i,\mathrm{k}} - \mathrm{OSNR\ Penalty}^{p}_{i,\mathrm{k}} \tag{3}$$

where all parameters have same meaning as described for Eq.2.

The module's scalability and reliability are cross-verified by analyzing it on two different Beneš sizes, namely the 6×6 and the 8×8 configurations. The distribution of ΔOSNRs at the ports of the 6×6 Beneš are shown in **Fig. 8**, along with their mean (μ) and standard deviation (σ) statistics. Similarly, the distribution of all the eight ΔOSNRs of the 8×8 Beneš are reported in **Fig. 9**.

In **Fig. 8** and **Fig. 9**, all the distribution of ΔOSNRs in both the cases are split by the dotted red line ($\Delta\mathrm{OSNR} = 0$) into two slices. The slice where $\Delta\mathrm{OSNRs} \leq 0$ is not critical as the $\mathrm{OSNR\ Penalty}^{a}_{i,\mathrm{k}} \leq \mathrm{OSNR\ Penalty}^{p}_{i,\mathrm{k}}$ so, in this case we only waste some capacity but the system will never turns into out-of-service. In contrast the section where $\Delta\mathrm{OSNRs} > 0$ is the critical one as $\mathrm{OSNR\ Penalty}^{a}_{i,\mathrm{k}} > \mathrm{OSNR\ Penalty}^{p}_{i,\mathrm{k}}$. In this case, it is necessary to deploy some margin on top of the ML prediction to keep the system working all the time. The maximum required margins (δ_k) for this case where $\Delta\mathrm{OSNRs} > 0$ are shown as a green dotted line for each port k of Beneš 6x6 and Beneš 8x8, respectively.

Examining the required margin, we can observe the high level of accuracy achieved by the ML QoT Agent. In the 6×6 Beneš, the worst-case prediction performance is observed on port 1; the δ_1 is less than 0.6 dB. For the larger 8×8 Beneš, the worst-case prediction is observed on port 8; the δ_8 is less than 0.65 dB. With the availability of such accurate prediction, we can envision that in practical applications the OSNR penalty margin on top of the ML prediction can be reduced to 0.6 dB and 0.65 dB for Beneš 6x6 and Beneš 8x8, respectively.

VI. Conclusions

In this work, the concept of a softwarized and autonomous configuration of PIC-based optical switches is introduced for optical SDN. We analyzed a data-driven ML technique to give the abstraction of any $N\times N$ photonics switch for QoT penalty evaluation and control states definition. The proposed scheme demonstrates two separate DNN based ML agents that are both topological and technological agnostic and can be engaged in real-time. The implemented ML approach first efficiently determines the control states for a generic $N\times N$ photonic switch using inverse ML design and then exploits these control states in direct ML design to predict QoT impairments without considering the device's internal architecture.

The technique we propose is scalable to larger input sizes N since a high level of accuracy can be reached with limited-size datasets. Also, the given abstracted model can be expanded to assess the performance of $N\times N$ optical switch on the network layer metrics. Furthermore, the model achieved promising results: for control states, prediction accuracy is 100% while for QoT penalty, we got a prediction error always less than 0.65 dB.

References

[1] V. Curri, "Software-defined WDM optical transport in disaggregated open optical networks," in *2020 ICTON,* (2020), pp. 1–4.

[2] L. Velasco et. al, "Building autonomic optical whitebox-based networks," J. Lightwave Technol. **36**, 3097–3104 (2018).

[3] D. Opferman et. al, "On a class of rearrangeable switching networks part I: Control algorithm," The Bell System Technical Journal **50**, 1579–1600 (1971).

[4] Y. Huang et. al, "Multi-stage 8 × 8 silicon photonic switch based on dual-microring switching elements," J. Lightwave Technol. **38**, 194–201 (2020).

[5] M. Ding et. al, "Routing algorithm to optimize loss and IPDR for rearrangeably non-blocking integrated optical switches," in *CLEO,* (OSA, 2015), pp. JTh2A–60.

[6] Y. Qian et. al, "Crosstalk optimization in low extinction-ratio switch fabrics," in *OSA,* (OSA, 2014), pp. Th1I–4.

[7] Q. Cheng et. al, "Silicon photonic switch topologies and routing strategies for disaggregated data centers," IEEE JSTQE **26**, 1–10 (2020).

[8] W. Gao et. al, "Automatic calibration of silicon ring-based optical switch powered by machine learning," Opt. Express **28**, 10438–10455 (2020).

[9] I. Khan et.al, "Effectiveness of machine learning in assessing QoT impairments of photonics integrated circuits to reduce system margin," in *IEEE Photonics Conference (IPC),* (IEEE, 2020), pp. 1–2.

[10] I. Khan et.al, "Machine learning assisted abstraction of photonic integrated circuits in fully disaggregated transparent optical networks," in *2020 22nd ICTON,* (2020), pp. 1–4.

[11] R. Proietti et. al, "Self-driving reconfiguration of data center networks by deep reinforcement learning and silicon photonic flex-lion switches," in *2020 IPC,* (2020), pp. 1–2.

[12] I. Khan et.al, "Machine-learning-aided abstraction of photonic integrated circuits in software-defined optical transport," (SPIE, 2021), pp. 146 – 151.

[13] A. Chakrabarty et. al, "Matrix-based nonblocking routing algorithm for Beneš networks," in *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns,* (2009), pp. 551–556.

[14] L. Tunesi et. al, "Automatic design of NxN integrated Benes optical switch," in *Silicon Photonics XVI,* , vol. 11691 G. T. Reed and A. P. Knights, eds., International Society for Optics and Photonics (SPIE, 2021), pp. 164 – 174.

[15] E. Ghillino et.al, "The Synopsys software environment to design and simulate photonic integrated circuits: A case study for 400G transmission," in *2018 20th International Conference on Transparent Optical Networks (ICTON),* (2018), pp. 1–4.

[16] C. M. Bishop, *Pattern recognition and machine learning* (springer, 2006).

[17] https://www.tensorflow.org/.

[18] C. Nwankpa et.al, "Activation functions: Comparison of trends in practice and research for deep learning," arXiv preprint arXiv:1811.03378 (2018).