Performance evaluation of data-driven techniques for the softwarized and agnostic management of an N×N photonic switch

Performance Evaluation of Data-Driven Techniques for Softwarized and Agnostic Management of $N\times N$ Photonic Switch

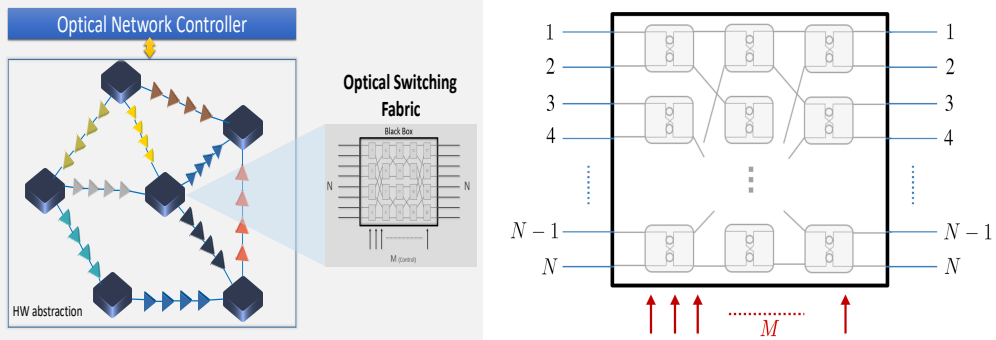
Ihtesham Khan,[1,*] Lorenzo Tunesi,[1] Muhammad Umar Masood,[1] Enrico Ghillino,[2] Paolo Bardella,[1] Andrea Carena ,[1] and Vittorio Curri[1]

[1] *Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129, Torino, Italy*
[2] *Synopsys, Inc., 400 Executive Blvd Ste 101, Ossining, NY 10562, United States*
[*] *ihtesham.khan@polito.it*

Abstract: The emerging Software Defined Networking (SDN) paradigm paves the way for flexible and automatized management at each layer. The SDN-enabled optical network requires each network element's software abstraction to enable complete control by the centralized network controller. Nowadays, silicon photonics due to its low energy consumption, low latency, and small footprint is a promising technology for implementing photonic switching topologies, enabling transparent lightpath routing in re-configurable add-drop multiplexers. To this aim, a model for the complete management of photonic switching systems' control states is fundamental for network control. Typically, photonics-based switches are structured by exploiting the modern technology of Photonic Integrated Circuit (PIC) that enables complex elementary cell structures to be driven individually. Thus PIC switches' control states are combinations of a large set of elementary controls, and their definition is a challenging task. In this scenario, we propose the use of several data-driven techniques based on Machine Learning (ML) to model the control states of a PIC $N\times N$ photonic switch in a completely blind manner. The proposed ML-based techniques are trained and tested in a completely topological and technological agnostic way, and we envision their application in a real-time control plane. The proposed techniques' scalability and accuracy are validated by considering three different switching topologies: the Honey-Comb Rearrangeable Optical Switch (HCROS), Spanke-Beneš, and the Beneš network. Excellent results in terms of predicting the control states are achieved for all of the considered topologies.

1. Introduction

Recently, the remarkable increase in the global internet traffic [1], compelled by the introduction of evolving technologies of connectivity such as 5G, internet of things and cloud services, has marked up high demands for flexible and dynamic network management at each layer. The latest optical SDN concept can provide the required degree of flexibility by implementing the complete virtualization of each Network Element (NE) and function within the network control system. Modern technologies like a coherent optical transmission for wavelength-division multiplexed optical transport and re-configurable optical switches for transparent wavelength routing provide a path to extend the SDN paradigm down to the physical layer [2–4]. To aim this, optical NE and transmission functions must be abstracted for Quality-of-Transmission (QoT) impairments and for controlling to empower the entire management by the optical control plane within the network controller [5, 6] as illustrated in Fig. 1a. The present effort is the evolution towards the fast-expanding trend in optical networks that goes towards the disaggregation and application of the SDN paradigm down to the WDM transport layer. This work mainly focuses on providing the abstraction of control states of re-configurable optical switches based on PICs with a complete structure-agnostic methodology based on several ML techniques.



(a) (b)

Fig. 1. (a) Abstraction of the optical switch in a SDN-controlled optical network. (b) Generic $N\times N$ optical switch fabric.

In recent times, the *smart* optical NEs extensively utilize PICs to execute complex functions at the photonic level. Specifically, in the core optical networks and data centers, large-scale photonic switching systems have a significant role, with the key benefits of their wide-band capabilities together with low latency and low power consumption. These photonic switches are mainly 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) [7] and beam-steering technique [8]. These technologies 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.

PIC-based systems mainly depend on elementary units such as Mach-Zehnder Interferometers (MZI) [9] or optical Micro Ring Resonators (MRR) [10]. The standard $N\times N$ optical switch architectures are assembled by cascading multiple stages of elementary units following a distinct switching topology, where N input signals can be routed to any of the N output ports by varying M control signals, as depicted in Fig. 1b. To scale up the range of $N\times N$ switching fabrics, the underlying requirement is to efficiently define the control states of the internal switching elements to obtain the requested signals permutation at the output of the integrated circuit.

The research on control/routing states of the photonic switching system has been scarcely stated at present. Nothing Like the electronic switches routing algorithms [11], where the performance of all paths are equal, the optical switches usually offer path-dependent performances [12]. Variations in performance can be fundamentally due to the topology and physical behavior of the elementary units. On the other hand, they can be derived from fabrication and design defects, which may alter the different switching states of the elementary units and their cascading effect on the entire component. Deterministic routing algorithms can effectively determine the control state of the internal switches for any requested output permutation. The effectiveness of these algorithms initiates in their topology dependence, which allows for a faster and more effective assessment of the multistage networks. However, creating the necessity to implement different algorithmic solutions depending on the network structure, with the cost of generality loss. In contrast, general-purpose routing and path-finding algorithms do not provide scalable solutions, as the computational complexity increases rapidly [13–15]. 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 complete routing space unfeasible, as well as the evaluation of the weighted penalties for all configurations.

In opposite with conventional topology-dependent methods, we propose a data-driven method

based on ML techniques to predict the control/routing states of the photonic integrated switch. Numerous ML-based methods have already been evaluated in managing PICs, such as [16] where an algorithm-driven by the artificial neural network is proposed to calibrate 2×2 dual-ring assisted-MZI switches. In [17], the author experimentally demonstrated a complete 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 de-scrambling and tunable optical filtering. In [18], the author proposed to use the Deep Reinforcement Learning (DRL) technique to drive reconfiguration of silicon photonic Flexible Low-latency Interconnect Optical Network Switch (Flex-LIONS) according to the traffic characteristics in High-Performance Computing (HPC) systems. Furthermore, in [19] a novel reinforcement ML-based framework is introduced. It is called DeepConf, for automatically learning and implementing a range of data center networking techniques. The developed DeepConf simplifies configuring and training deep learning agents by using intermediate representation to learn different tasks.

In this work, we introduce a unique topology-agnostic *blind* approach exploiting several ML techniques for predicting the control states of the $N\times N$ photonic switch with arbitrary and potentially unspecified internal configuration. The models are trained using 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 device software simulator.

Initial and partial results for this approach are presented in [20]. In this paper, besides describing in detail the methodology, we extend the framework by suggesting several ML techniques and presenting a detailed performance assessment of these data-driven approaches to model the control states of the photonic switching system. The optimization of these ML models in terms of prediction, accuracy, and complexity is also performed. Furthermore, ML models' error distribution in the predicted control states is also verified and analyzed. The error analysis aims at assessing the quantitative effectiveness of the trained ML agent in predicting the proper internal switching routing, given the PIC topology. Future evolution of the proposed framework will target the inclusion of transmission penalties in the set ML agent prediction, to enable a full component virtualization within a SDN optical transport network scenario.

The remainder of the paper is organized as follows. In Sec. 2, we describe the specific architecture of the Beneš, Spanke-Beneš and HCROS switches used for the demonstration of the proposed ML techniques. In Sec. 3, we describe the simulation environment used to generate datasets, presenting its structure and various statistics. In Sec. 4, we illustrate the architecture of the proposed ML models used to predict the control state of the PIC-based switching system. Then, in Sec. 5, 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 signals of internal switching elements. In this work, we do not aim to develop specific ML models; instead, our focus is to show the general effectiveness of ML in this scenario. So, we exploit an extensively tested open-source projects, namely the TensorFlow© and *scikit-learn*© libraries [21,22]. Results of our proposed approach are shown in detail in Sec. 6. We demonstrate that the trained ML models enable the correct estimation of the internal switching elements control states for different $N\times N$ architectures. We also show that a heuristically enhanced ML can further improve the predictions' accuracy in the present scenario. Finally, conclusions are presented in Sec. 7.

2. Multistage switching networks

The routing operation can be carried out through a variety of switching devices, with diverse structures and implementations depending on the transmission requirements and constraints. Instead of designing ad-hoc switches for the desired number of inputs N, a standard class of implemented structures are based around the multistage network paradigm. In these components,

the routing is achieved through a cascade of multiple stages, made from smaller switching elements, as to reduce the overall complexity of the circuit, as well as the footprint and number of switching sections. The configurations of this switching network are defined by the control signals applied to each of the M Optical Switching Elements (OSEs), which determine the output configuration of the device.

2.1. 2×2 crossbar switches

In optical transmission these elements are typically implemented as 2×2 elementary switching devices, represented as black-box modelled crossbar switches, as shown in Fig. 2. The 2×2 crossbar switch is defined as a two-state device, piloted by a control signal M, which toggle between the two configurations. The bar state, defined for $M = 0$, represent the straightforward propagation of the two input signals ($\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} \rightarrow \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix}$), while in the cross state, for $M = 1$, the output signals order is reversed ($\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} \rightarrow \begin{pmatrix} \lambda_2 \\ \lambda_1 \end{pmatrix}$). As previously stated this fundamental block can be physically implemented through a variety of approaches, with the most prominent two being the Microring Resonator(MRR) and the Mach Zehnder Interferometer (MZI). These implementations offer different performances based on physical design and can be tailored for both colorless or chromatic-dependent applications. The binary control signal present in the black-box model is typically provided through an electrical signal in the OSE, with a dependency on the device implementation. Nonetheless on a virtual abstraction of the component, for the routing path evaluation, the binary model is suitable while maintaining a general device-independent scope.

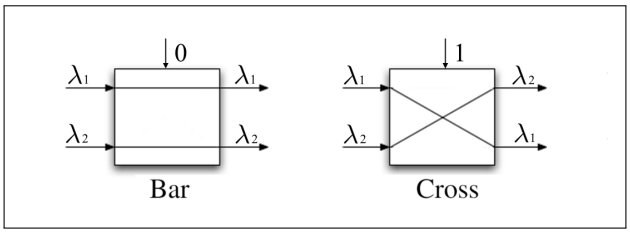


Fig. 2. Illustration of Bar and Cross states of a 2×2 elementary switching element (CrossBar switch).

2.2. Multistage Rearrangeable Non-Blocking networks

To design a N×N switch the crossbar elements must be placed in a suitable topology, which determines the properties of the switch, both in terms of routing capabilities and number of elements required. The focus of this approach is directed toward a sub-class of switching networks which are defined as *Rearrangeable Non-Blocking* networks. Switching networks can be defined as non-blocking if all the possible permutations of the input signals can be routed to the output ports: any input-output (I/O) request targeted at an unoccupied port can be established without creating conflicts inside the network, taking also into account the already established I/O links.

In rearrangeable networks this property partially upholds, as the routing of all permutations is achievable, although potentially requiring a reconfiguration of the previously established I/O connections. In this class of multistage networks the reconfiguration of the switches is required, as the topology doesn't guarantee path-availability if traffic is already present in the device. This is a clear disadvantage with respect to strict-sense non-blocking structures, as it requires the implementation of a more complex control unit to evaluate the routing and the conflicts inside the network. The trade-off is acceptable in most applications, as this wider-sense property allows most of the topologies to implement an N×N switch with fewer elements with respect to the strict-sense devices, with a clear reduction of transmission losses, power consumption and

footprint.

2.3. Problem complexity

Considering a generic device in this class, the two main parameters determining the model complexity are the number of inputs N, or the network size, and the number of needed OSE M, which correspond to the number of variables to establish a target routing path. The solution space of all the output permutations ($N!$), as well as the states configurations (2^M) grows as a Non-Polynomial (NP) function as the network size increases. This is due to the dependency of the number of OSE on the number of inputs. Typically for these networks, the relationship follows $M = O(N \cdot \log(N))$ with a small degree of variance between the different available topologies. This phenomenon leads to scalability issues concerning traditional topology-independent path-finding algorithms, as the NP complexity increase cannot be directly approached. Topology-specific routing algorithms exist for each class, although this solution bears two main disadvantages: it requires a rigid control unit which cannot pilot a device with a different topology, as well as the unavailability of researching an optimal solution.
Multi-switching networks, especially in the optical domain, are prone to path-dependant degradation of performance, with a wide range of QoT between the equivalent paths available for the same output configuration. The deterministic topology-dependant routing algorithm needs to evaluate all the equivalent paths, with a complexity dependence on N as $N_{eq} = O(2^M \div N!)$, leading to severe scalability issues. As such ML-based methods can overcome the limitation and can be trained on performance-aware data-sets. Under this scenario the NP size of the solution space becomes an asset instead of a disadvantage, allowing the generation of large data-set for training the specific ML agent.

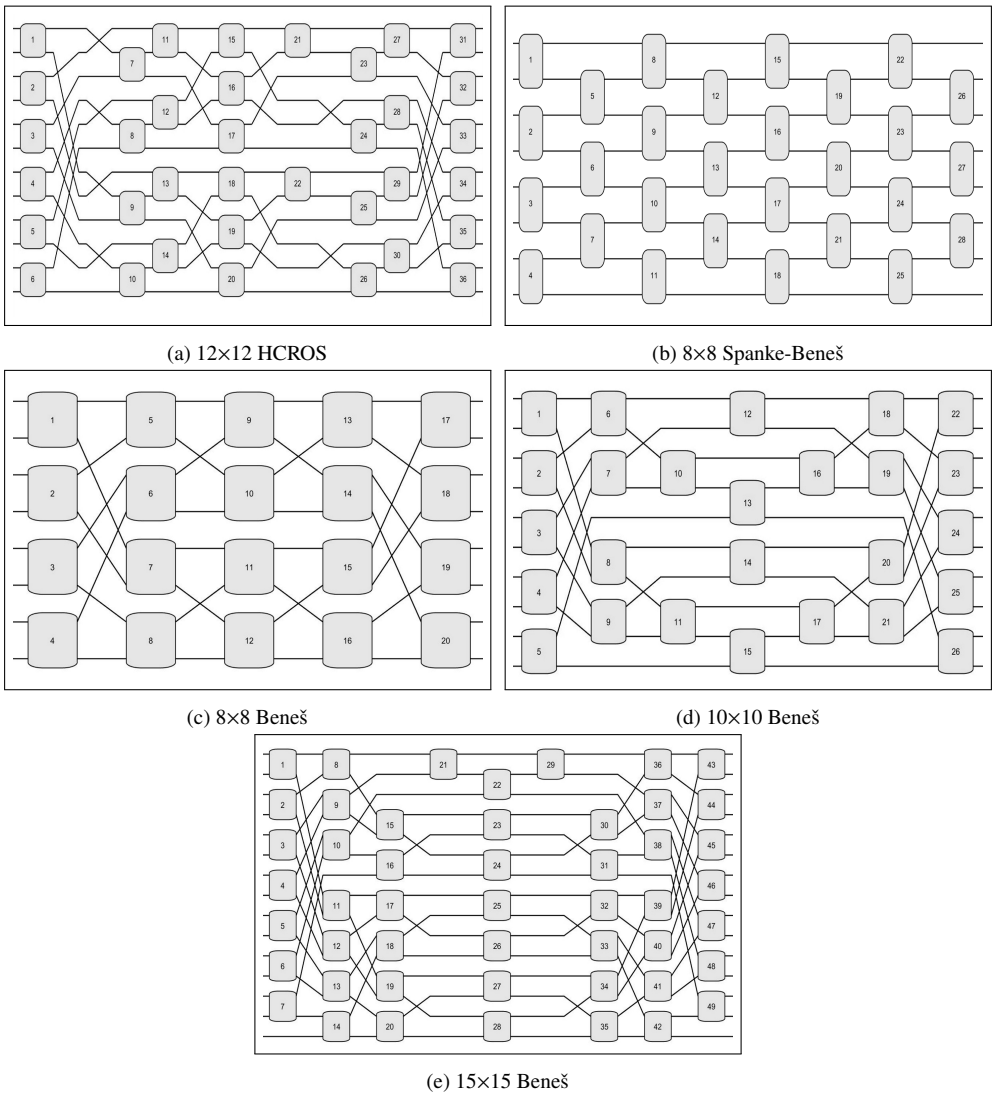
2.4. Topologies under analysis

In order to test the performance of the proposed method, both the scalability as well as the robustness with respect to the topology variation must be tested. To this end, three main topological structures were tested, and are depicted in Fig. 3.

The first network under analysis is the Beneš switch. This device follows a recursive structure based on the Clos network paradigm, with number of OSE $M = N\log_2(N) - \frac{N}{2}$. The Beneš network is a common approach to multistage switching networks, as it's characterized by a low number of 2×2 switching elements, implying reduced footprint and power consumption with respect to larger topologies. Three instances of Beneš structures have been tested, with network size $N = 8,\ 10,\ 15$ and $M = 20,\ 26,\ 36$.

An alternative to the recursive Beneš structure is the Spanke-Beneš network: this topology is distinguished by its planarity, as no crossing interconnection is used between the switching stages. The planarity comes as a cost in terms of number of OSE, which are equal to $M = \frac{N\cdot(N-1)}{2}$, increasing the already severe effect of the NP complexity growth. This topology is still considered for applications where the crossing technology cannot be relied upon to guarantee the needed QoT, as such the size $N = 8 \quad M = 28$ was chosen, as to allow similar complexity to the Beneš networks under analysis.

The third considered topology is an optimized with equal footprint with respect to the Beneš counterpart. The device is not based on the planarity constraint of the Spanke-Beneš nor the recursive generation of the traditional Beneš networks. This structure, referred as the HoneyComb Rearrangeable Optical Switch (HCROS) [23], shows an asymmetric topology with respect to the traditional implementations, acting as an extremely useful benchmark to test the proposed ML-method robustness related to irregular and uncommon structures. The device proposed was extended to a 12×12 structure as to offer a valid comparison to the other switching devices under



(a) 12×12 HCROS

(b) 8×8 Spanke-Beneš

(c) 8×8 Beneš

(d) 10×10 Beneš

(e) 15×15 Beneš

Fig. 3. Multistage switching networks topologies under analysis

analysis.

All three considered structures belongs to the same class with respect to non-blocking properties, as they require rearrangement of the control states to route new input-output links, when traffic already occupies part of the circuit. Similarly each topology has more OSE configuration states than signal output permutations, as such each device can obtain the same output configuration through a different number of equivalent paths.

3. Simulation and Dataset Generation

The dataset has a vibrant role in the training and verification of data-driven models. In the current scenario, the required dataset is retrieved by implementing the abstraction of the considered topologies. Each 2×2 switch factor is driven by a control bit, with 0 characterizing the BAR state and 1 the CROSS state: each configuration of the network can be described by a bit-array of length

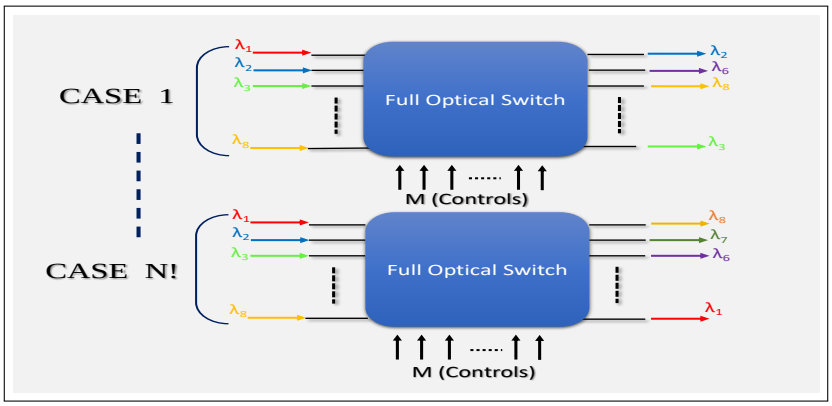


Fig. 4. Graphical representation of all possible $N!$ output states for a $N \times N$ fabric.

equal to the number of switches M. The considered architectures are implemented as a cascade of permutation matrices, representing each switch and crossing stage. In multistage structures like the Beneš, Spanke-Beneš and HCROS, the network's output for a given control vector can be calculated by applying the permutation matrices sequentially for each stage [11], without considering a more complex and computationally expensive graph structure. In non-blocking switching topologies, the M control signals' 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.

The size of a realistic dataset for training should be much smaller than the full-sized look-up table. If such a table could be evaluated in a realistic environment, 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 specific switch configurations or preferential paths within the network. The total train and test dataset for the considered Beneš, Spanke-Beneš and HCROS architectures are a subset of the total 2^M control combinations, as reported in Table 1. After having sufficient ML module training, the testing is performed for each of the randomly selected permutations of the output channels; the corresponding combination of switches returned by the ML framework is determined by calculating the product of the permutation matrices corresponding to the specific switches states. Finally, the ML agent's accuracy is determined by comparing the combination obtained with the original permutation requested as input.

4. Analysis of Machine Learning Models

The standard ML framework allows the translation of effective system attributes that cannot be easily or directly measured. Generally, ML models develop cognition capability by exploiting a series of intelligent algorithms that can understand the training data's intrinsic information. The information inherited by the intelligent algorithms is then abstracted into the decision models that manage the testing phase. These well-trained cognitive models provide real-time operational

Table 1. Dataset Statistics

Network type	**Beneš**	**Beneš**	**HCROS**	**Beneš**	**Spanke-Beneš**	**Spanke-Beneš**
Size ($N \times N$)	**8x8**	**10x10**	**12x12**	**15x15**	**8x8**	**10x10**
Permutations ($N!$)	40,320	3,628,800	479,001,600	$1{,}307\times10^9$	40,320	3,628,800
Switches (M)	20	26	36	49	28	45
Combinations (2^M)	1,048,576	67,108,864	68×10^9	562×10^{12}	268,435,456	35×10^{12}
Dataset	100,000	300,000	300,000	1,000,000	300,000	1,000,000

improvements by enabling the system to draw *smart* conclusions and react autonomously.

In the current work, five distinct ML models have been proposed to model the control states of a PIC-based $N\times N$ photonic switching system. The proposed ML framework consists of three basic units; pre-processing, training, and testing. The pre-processing section standardizes the data set before utilizing it in the training section. The training section exploits the standardized train set for the training of the proposed models. Following training, the testing unit uses a test set of the data to initiate the testing phase. The proposed ML models are developed by using high-level python Application Program Interface (API) of two open-source ML libraries, *TensorFlow*© [21] and *scikit-learn*© [22] . Both of these libraries provides a vast range of APIs for data-driven models and different functions to pre-process and clean the dataset from the noise before applying it as an input to the ML model.

4.1. Decision Tree Regression

The Decision Tree Regressor (DTR) model is developed to model the control states of a PIC-based $N\times N$ photonic switching system. Normally, DTR provides direct relationships between the input and response variables [24] by constructing a tree based on various decisions made by exploring several dimensions of the provided features and ultimately provides the desired response variable. The proposed DTR has two key tuning parameters; *min_samples_leaf* and *max_depth*. The optimum values of these two main parameters are obtained by tuning them in order to achieve the best trade-off between precision and computational time in the proposed simulation environment.

4.2. Random Forest Regression

The considered Random Forest Regressor (RFR) uses *ensemble* learning which is based on the *bagging* tree technique [25]. In this technique, various subsets of data from the training set are chosen with replacement. The extracted subset of training samples is managed to train the individual decision tree that runs independently. All the individual decision trees (without giving importance to any particular tree) are averaged to give the final output. In contrast to the classical *Bagging* mechanism, the proposed RFR has a step extension as it chooses a random subset of train sample and a random selection of features rather than using all the features to train several decision trees. The final prediction of the developed RFR is made by simply averaging the predictions of each decision tree. Similar to DTR the RTR has also two key parameters *min_samples_leaf* and *max_depth*. The tuning of these two parameters is performed in such a way as to obtain the best trade-off between accuracy and complexity.

4.3. Boosted Tree Regression

The proposed Boosted Tree Regressor (BTR) also uses *ensemble* learning, but in contrast to RTR, it is based on the *Gradient-Boosting* technique. The developed BTR works by combining various regression trees' models, especially decision trees, using *Gradient-Boosting* technique [26]. The mathematical representation of this class of models can be written as in Equation 1, where the final regressor f is the sum of

$$f(x) = r_0 + r_1 + r_2 + ... + r_i \tag{1}$$

simple base regressor r_i. Like the other tree regressors key parameters, we also tune these parameters for BTR to obtain the optimum values between precision and complexity.

4.4. Linear Regression

Linear Regression (LR) is a kind of ML model which utilizes a statistical method to learn the linear relationship between the input feature (x) and the output response variable (y). Generally, the mathematical description of LR is as follows:

$$y = B_0 + B_1 x \tag{2}$$

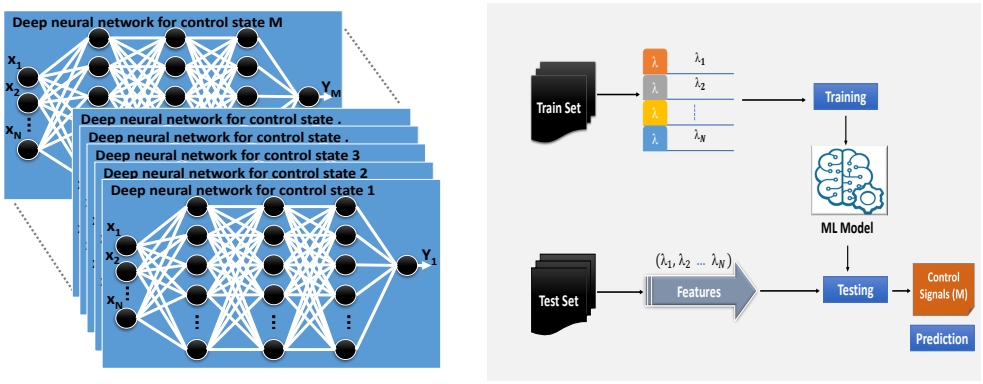
where y is the output variable, B_0 is the intercept, B_1 is the co-efficient of each variable, and x is the input features set. The model estimates the values of intercept (B_0) and the co-efficient (B_1). LR has a different kind of optimization strategy. In our work to model the control states of a PIC-based $N\times N$ photonic switching system, we applied the ordinary least square method that takes more than one input feature and requires no weighting function.

Table 2. Machine learning Models Detail

Machine learning Model	API	Parameter	Value
Decision Tree Regressor	*scikit-learn*©	*Min samples leaf*	4
		Max depth	100
Random Forest Regressor	*scikit-learn*©	Method	$'Bagging'$
		Min samples leaf	4
		Max depth	100
Boosted Tree Regressor	*TensorFlow*©	Method	*'Gradient Boosting'*
		Min samples leaf	4
		Max depth	100
		Learning rate	0.01
		L_1 regularization	0.001
Linear Regressor	*TensorFlow*©	$Equation$	Linear
		Training steps	1000
		Method	*Ordinary Least Squares*
Deep Neural Network	*TensorFlow*©	*Hidden layers*	3
		Keras optimizer	$'ADAGRAD'$
		Activation function	$'ReLU'$
		L_1 regularization	0.001
		Learning rate	0.01
		Training steps	1000

4.5. Deep Neural Network

The Deep Neural Network (DNN) is one of the most frequently used ML models inspired by the human nervous system to process information. Generally, DNN is not a single artificial neuron with multiple layers but multiple artificial neurons with multiple layers. Typically, DNN consist of the input layer, hidden layers, and output layer, where each layer has a set of neurons [27]. To model the control states of a PIC-based $N\times N$ photonic switching system, the considered DNN is configured by several parametric values that have been optimized (such as the *training steps*), loaded with the *Adaptive Gradient Algorithm (ADAGRAD)*, *learning rate* and L_1 regularization [28]. 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 [29]. 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 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. 5a: in practice,



(a) (b)

Fig. 5. (a) Parallel architecture of DNN with hidden layers. (b) Description of the ML agent.

we have an independent DNN for the prediction of each of the control states.

5. Machine Learning Framework

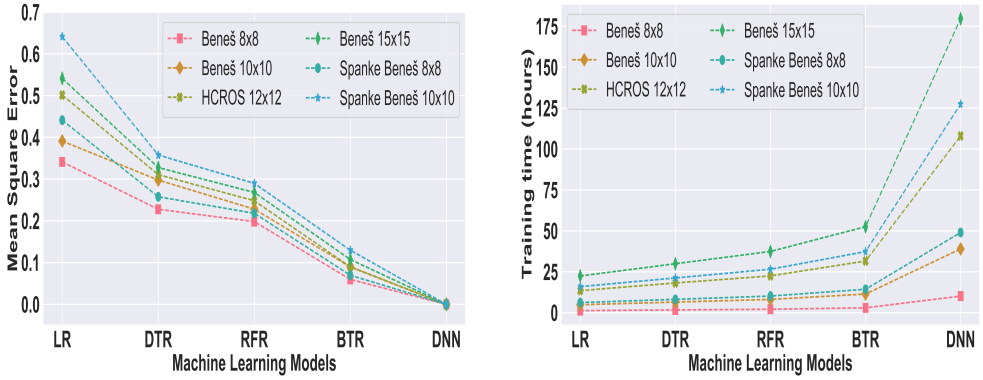
The proposed ML-based methods operate in a complete black-box set up, requiring a sufficiently large amount of training data to develop cognitive models without considering the photonic circuits' internal structural design. We evaluate five ML techniques and compare the prediction performance in the proposed framework of the investigation. Like all other supervised ML-based learning methods, 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 features comprise the numerous permutations of the input signals (λ_1, λ_2, λ_3....λ_n) at the output ports of the switch, and it exploits its M control signals as labels shown in Fig. 5b. Initially, the training of the ML models is performed. After that, we tested the trained models on the independent subset of the dataset; the standard rule of 70% and 30% has been preferred to set the subset ratios. In order to avoid over-fitting the models, for each particular M we set the *tree size* (for all tree regression models) and *training steps* (for LR and DNN) as the stopping factor and the *Mean Square Error* (MSE) as the loss function, given by:

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n}\left(\sum_{m=1}^{M}\left|\text{Ctrl Signal}^{p}_{i,\text{m}} - \text{Ctrl Signal}^{c}_{i,\text{m}}\right|\right)^{2} \tag{3}$$

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, $\text{Control Signal}^{p}_{i,\text{m}}$ and $\text{Control Signal}^{c}_{i,\text{m}}$ are the predicted and correct control bits of the m-th switching element of the considered configuration. The general tuning parameters and the API used to build the proposed models are reported in Tab.2.

6. Results and Discussion

This section describes the performance evaluation of numerous ML models developed using higher-level python APIs of the *scikit-learn*© and *TensorFlow*© libraries. The numerical assessment of the proposed data-driven methods is illustrated in Fig. 6a against the different $N\times N$ photonics switching configuration (i.e., Beneš, Spanke-Beneš, and HCROS). Fig. 6a describes the MSE achieved against each of the proposed ML models for different considered



(a) (b)

Fig. 6. (a) Mean Square Error of ML models (b) Training Time of ML models.

$N \times N$ architectures. The MSE in the current simulation environment follows the following order LR>DTR>RFR>BTR>DNN for almost all the considered $N \times N$ configurations. The LR and DTR shows the worst performance in terms of MSE as they cannot uncover the underlying relationship and irregularities. On the other hand, the RFR benefits from averaging numerous decision trees instead of a single decision tree trained on randomly selected subsets of the training sample. Moreover, the BTR exploits the boasting technique, merging various regression trees' models and pick out the new tree that best degrades the loss function instead of randomly choosing. Therefore, the overall performance of the BTR is better than the RFR. Finally, the DNN performed remarkably well because of its cognitional capability supplied by internally aligned artificial neurons compared to the RFR and BTR.

Besides this analysis, the training time for a single control state M is also shown in Fig. 6b for all the proposed ML models. The training timing shows the reverse order as we observed during MSE analysis: LR<DTR<RFR<BTR<DNN. The proposed DNN takes a longer time duration during training than the other suggested models due to its internal hidden layers, containing numerous neuron units. The RFR and BTR take a slighter longer time than the LR and DTR because of their dependency on the bagging and boasting techniques. The proposed models are simulated on a workstation having specifications, 32 GB of 2133 MHz RAM and an Intel® Core™ i7 6700 3.4 GHz CPU. Furthermore, the produce simulation is performed without considering quantum computing.

Typically, in all the scenarios where data-driven models are exploited, the main objective of using these learning methods is their high accuracy in contrast to the training time. As the ML-based models only need initial training that takes a long time, but the testing can be done in real-time once the models are adequately trained. To this aim, we selected DNN as the preferable ML model to proceed with further investigation. In the rest of the paper, all results are obtained using the proposed parallel DNN approach. To further verify our selection, we observed the complete trend of the loss function (i.e., MSE) with respect to the training steps for the proposed DNN, shown in Fig. 7a for a single considered case of Beneš 8x8. Similar behavior is observed for all the other considered switching architectures.

The first assessment we performed is the prediction accuracy dependency on the dimension of the training dataset and the size of hidden layers shown in Fig. 7b and Fig. 7c. In Fig. 7b, the effect of increasing training dataset size is described. The trend reveals that the prediction capability of the proposed DNN improves with an increase in the training dataset size. Likewise, in Fig. 7c, the effect of increasing the number of neurons per hidden layer is shown: the prediction

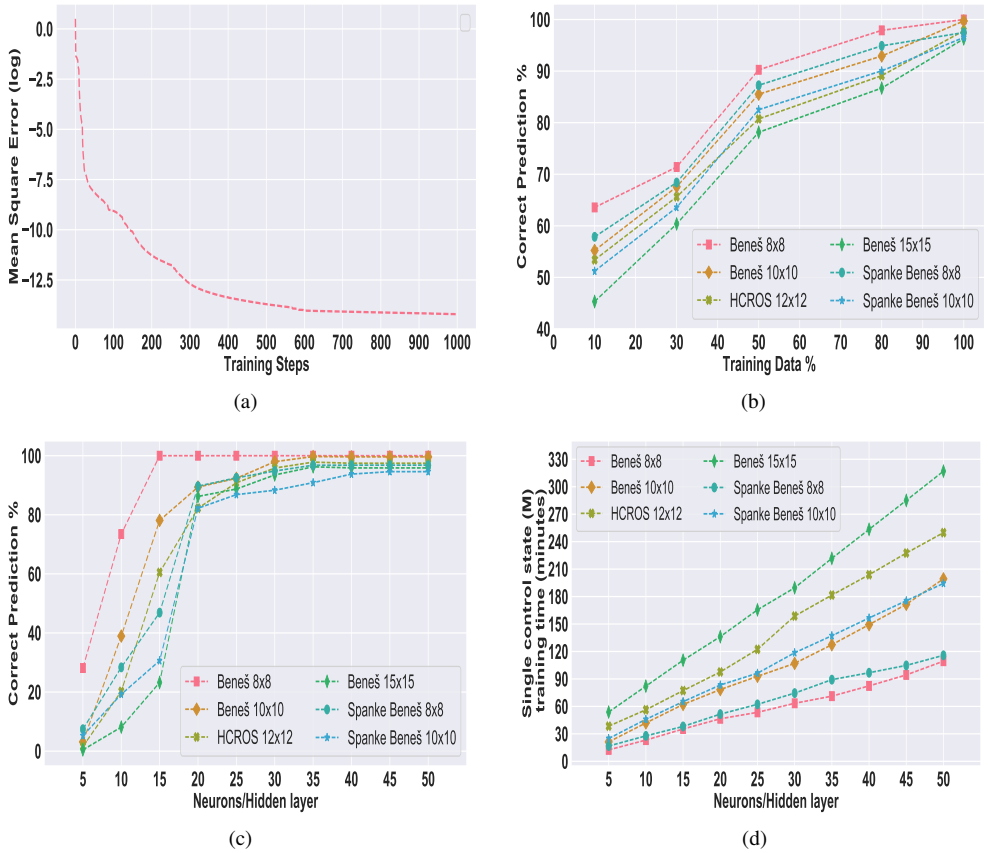


Fig. 7. (a) DNN loss function vs. the training steps for Beneš 8x8 architecture. (b) Percentage of correct predictions vs. normalized training dataset size. The normalization is performed with respect to the total generated dataset dimension for the considered N×N fabric, see data in Tab. 1. (c) Percentage of correct predictions vs. hidden layer size for the considered switching configurations (d) Single switch training time vs. hidden layer size.

ability of the DNN improves when increasing the hidden layer size until the diminishing or constant trend is encountered. The lowest possible number of mandatory neurons per layer depends on the structure under examination: the values selected for the following analysis are listed in Table 3. Finally, the correct prediction percentage for the optimized DNN is summarized for the Beneš (8x8, 10x10 and 15x15), Spanke-Beneš (8x8, 10x10) network along with the 12x12 HCROS in Table 3. In the Beneš network, we notice an excellent accuracy level (>96%). However, with reduced prediction efficiency when expanding N: correct predictions reach 100%, 99.72%, and 96.25% for N equal to 8, 10, and 15, respectively. To further validate the results, similar results were obtained based on Spanke-Beneš and HCROS: also, in both of these considered architectures, we observe a high level of accuracy (97.47%, 96.51%, and 97.83% for Spanke-Beneš (8x8, 10x10) and 12x12 HCROS, respectively).

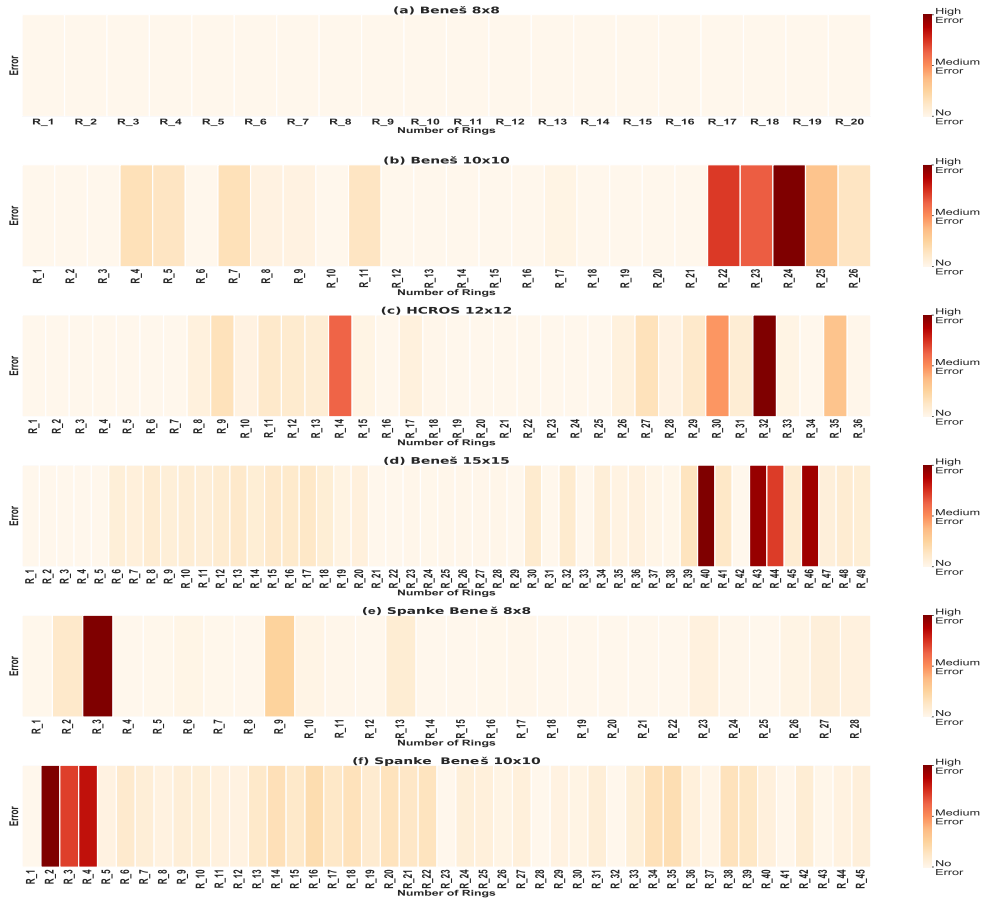


Fig. 8. Heatmap showing normalized error in prediction of control signals using DNN.

Algorithm 1 Heuristic to correct single-ring errors

Require: *Topology Graph* $\mathcal{G}$, *Control Signals* M, *Number of Inputs/outputs* N, *Test set* $\mathcal{TS}$, *ML Predicted set* $\mathcal{PS}$
Ensure: *Control Signal Correction*

error-index = find control signals where $\mathcal{TS} \neq \mathcal{PS}$
for all $\mathcal{D} \in$ error-index **do**
  $Predicted_{\text{Control States (ON/OFF)}}\ (P_{Ctrl}) = \mathcal{PS}(\mathcal{D})$
  $Actual_{\text{Control States (ON/OFF)}}\ (A_{Ctrl}) = \mathcal{TS}(\mathcal{D})$
  $Actual_{\text{Output Signals}}\ (A_{otp}) = \mathcal{TS}(\mathcal{D})$
  $Predicted_{\text{Output Signals}}\ (P_{otp})$ = Test-Control-States(P_{Ctrl},$\mathcal{G}$,N)
  $Check_{\text{Output Signals}}\ (C_{otp}) = P_{otp}$
  for $flip-bit \in M$ **do**
    if find $A_{otp} \neq C_{otp}$ **then**
      $Transit_{\text{Control States (ON/OFF)}}\ (T_{Ctrl})$ = 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

Following assessing the accuracy, we studied the distribution of errors in the predicted states, as demonstrated in Fig. 8. The extent of errors in each switching element's control state prediction when considering the test set is encoded in the color heatmap. A non-uniform distribution is observed in which errors are clustered on a small number of switch elements of the total fabric. Based on this observation, we analyzed the number of wrong switch elements where the prediction fails. Furthermore, the results in Table 3 show that only a single error in one of the switches controls is responsible for the incorrect routing in most of the wrong prediction in all the considered architectures instances. Observing this phenomenon, we formulate a simple heuristic that can further improve the DNN prediction performance (see Algo. 1). The heuristic we suggest requires several device properties such as topological graph ($\mathcal{G}$), M control signals, and N number of inputs/outputs signals. Additionally, the Test set ($\mathcal{TS}$) and ML Predicted set ($\mathcal{PS}$) are also loaded as an input. The proposed heuristic corrects the single switch errors by switching the state of one element at a time while comparing the output sequence against the target output permutation of wavelengths. This heuristic makes only M iteration, and this number is reasonably small, so it can be considered feasible for real-time operations. Besides this, it is also topologically and technologically agnostic as for all the considered architectures, DNN assisted by heuristic enhances the accuracy up to 100%.

Table 3. Summary of ML prediction results

Network type **Size ($N \times N$)**	**Beneš** **8x8**	**Beneš** **10x10**	**HCROS** **12x12**	**Beneš** **15x15**	**Spanke-Beneš** **8x8**	**Spanke-Beneš** **10x10**
Neurons per hidden layer	15	35	35	35	35	45
Accuracy (no heuristic)	100%	99.72%	97.83%	96.25%	97.47%	96.51%
Single switch control error	0%	0.28%	2.17%	3.75%	2.53%	3.49%
Multiple switch control error	0%	0%	0%	0%	0%	0%
Accuracy (with heuristic)	100%	100%	100%	100%	100%	100%

7. Conclusion

We propose and analyze several data-driven ML techniques to control and manage photonic switching systems in the framework of SDN-enabled optical networks in this work. The proposed scheme demonstrates that the ML-based softwarized system is both topological and technological agnostic and can be operated in real-time. The proposed ML approaches can effectively determine the control states for a generic $N\times N$ photonic switch without any required knowledge on the structure topology. The presented ML framework is trained and tested assuming the $N\times N$ photonic switch as black-box: the ML-based models only need a sufficient amount of training instances for the development of model cognition, without taking into consideration the device internal architecture. The techniques we propose are 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 the DNN shows an excellent accuracy than other proposed models. Finally, a simple heuristic approach can increase the prediction accuracy of DNN to 100% with a small increase of the computational cost for the considered switching architecture.

Disclosures

The authors declare no conflicts of interest.

Data Availability Statement

Data underlying the results presented in this paper are not publicly available at this time but may be obtained from the authors upon reasonable request.

References

1. Cisco, "Cisco Visual Networking Index: Forecast and Trends, 2017–2022," Tech. rep., Cisco (2017).
2. C.-S. Li and W. Liao, "Software defined networks," IEEE Commun. Mag. **51**, 113–113 (2013).
3. 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. **49**, 164–172 (2011).
4. A. Ferrari, M. Filer, K. Balasubramanian, Y. Yin, E. Le Rouzic, J. Kundrát, G. Grammel, G. Galimberti, and V. Curri, "Gnpy: an open source application for physical layer aware open optical networks," JOCN **12**, C31–C40 (2020).
5. V. Curri, "Software-defined WDM optical transport in disaggregated open optical networks," in *2020 22nd International Conference on Transparent Optical Networks (ICTON),* (2020), pp. 1–4.
6. L. Velasco, A. Sgambelluri, R. Casellas, L. Gifre, J.-L. Izquierdo-Zaragoza, F. Fresi, F. Paolucci, R. Martínez, and E. Riccardi, "Building autonomic optical whitebox-based networks," J. Light. Technol. **36**, 3097–3104 (2018).
7. J. Kim, C. J. Nuzman, B. Kumar, D. F. Lieuwen, J. S. Kraus, A. Weiss, C. P. Lichtenwalner, A. R. Papazian, R. E. Frahm, N. R. Basavanhally, D. A. Ramsey, V. A. Aksyuk, F. Pardo, M. E. Simon, V. Lifton, H. B. Chan, M. Haueis, A. Gasparyan, H. R. Shea, S. Arney, C. A. Bolle, P. R. Kolodner, R. Ryf, D. T. Neilson, and J. V. Gates, "1100 x 1100 port MEMS-based optical crossconnect with 4-dB maximum loss," IEEE PTL **15**, 1537–1539 (2003).
8. A. N. Dames, "Beam steering optical switch," (2008). US Patent 7,389,016.
9. K. Suzuki, R. Konoike, J. Hasegawa, S. Suda, H. Matsuura, K. Ikeda, S. Namiki, and H. Kawashima, "Low-insertion-loss and power-efficient 32× 32 silicon photonics switch with extremely high-δ silica PLC connector," JLT **37**, 116–122 (2019).
10. Q. Cheng, L. Y. Dai, N. C. Abrams, Y.-H. Hung, P. E. Morrissey, M. Glick, P. O'Brien, and K. Bergman, "Ultralow-crosstalk, strictly non-blocking microring-based optical switch," Photonics Res. **7**, 155–161 (2019).
11. D. Opferman and N. Tsao-Wu, "On a class of rearrangeable switching networks part I: Control algorithm," The Bell Syst. Tech. J. **50**, 1579–1600 (1971).
12. Y. Huang, Q. Cheng, Y.-H. Hung, H. Guan, X. Meng, A. Novack, M. Streshinsky, M. Hochberg, and K. Bergman, "Multi-stage 8 × 8 silicon photonic switch based on dual-microring switching elements," JLT **38**, 194–201 (2020).
13. 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 *CLEO,* (OSA, 2015), pp. JTh2A–60.
14. Y. Qian, H. Mehrvar, H. Ma, X. Yang, K. Zhu, H. Fu, D. Geng, D. Goodwill, P. Dumais, and E. Bernier, "Crosstalk optimization in low extinction-ratio switch fabrics," in *OFC,* (OSA, 2014), pp. Th1I–4.
15. Q. Cheng, Y. Huang, H. Yang, M. Bahadori, N. Abrams, X. Meng, M. Glick, Y. Liu, M. Hochberg, and K. Bergman, "Silicon photonic switch topologies and routing strategies for disaggregated data centers," JSTQE **26**, 1–10 (2020).
16. W. Gao, L. Lu, L. Zhou, and J. Chen, "Automatic calibration of silicon ring-based optical switch powered by machine learning," Opt. Express **28**, 10438–10455 (2020).
17. H. Zhou, Y. Zhao, X. Wang, D. Gao, J. Dong, and X. Zhang, "Self-learning photonic signal processor with an optical neural network chip," arXiv preprint arXiv:1902.07318 (2019).
18. 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 *2020 IPC,* (2020), pp. 1–2.
19. S. Salman, C. Streiffer, H. Chen, T. Benson, and A. Kadav, "Deepconf: Automating data center network topologies management with machine learning," in *Proceedings of the 2018 Workshop on Network Meets AI & ML,* (Association for Computing Machinery, New York, NY, USA, 2018), NetAI'18, p. 8–14.
20. I. Khan, L. Tunesi, M. Chalony, E. Ghillino, M. U. Masood, J. Patel, P. Bardella, A. Carena, and V. Curri, "Machine-learning-aided abstraction of photonic integrated circuits in software-defined optical transport," in *Next-Generation Optical Communication: Components, Sub-Systems, and Systems X,* vol. 11713 G. Li and K. Nakajima, eds., International Society for Optics and Photonics (SPIE, 2021), pp. 146 – 151.
21. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} ({OSDI} 16),* (2016), pp. 265–283.
22. G. Hackeling, *Mastering Machine Learning with scikit-learn* (Packt Publishing Ltd, 2017).
23. 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 **8**, 844 (2019).
24. L. Rokach and O. Z. Maimon, *Data mining with decision trees: theory and applications*, vol. 69 (WS, 2008).
25. L. Breiman, "Random forests," Mach. learning **45**, 5–32 (2001).
26. J. Elith, J. R. Leathwick, and T. Hastie, "A working guide to boosted regression trees," JAE **77**, 802–813 (2008).
27. I. Khan, M. Bilal, and V. Curri, "Assessment of cross-train machine learning techniques for qot-estimation in agnostic optical networks," OSA Continuum **3**, 2690–2706 (2020).
28. J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," JMLR **12**, 2121–2159 (2011).
29. C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," arXiv preprint arXiv:1811.03378 (2018).