Data preprocessing for machine-learning-based adaptive data center transmission

Citation for the original published paper (version of record):

Keykhosravi, K., Hamednia, A., Rastegarfar, H. et al (2022). Data preprocessing for machine-learning-based adaptive data center transmission. ICT Express, 8(1): 37-43. http://dx.doi.org/10.1016/j.icte.2022.02.002

(article starts on next page)

KICS
The Korean Institute of
Communications and
Information Sciences

Available online at www.sciencedirect.com
ScienceDirect

ICT Express 8 (2022) 37–43

www.elsevier.com/locate/icte

Data preprocessing for machine-learning-based adaptive data center transmission

Kamran Keykhosravi[a],*, Ahad Hamednia[a], Houman Rastegarfar[b], Erik Agrell[a]

[a] *Department of Electrical Engineering/Chalmers University of Technology, Gothenburg 412 96, Sweden*
[b] *MathWorks Inc., 1 Apple Hill Dr, Natick, MA 01760, USA*

Received 25 September 2021; received in revised form 21 December 2021; accepted 5 February 2022
Available online 14 February 2022

Abstract

To enable optical interconnect fluidity in next-generation data centers, we propose adaptive transmission based on machine learning in a wavelength-routing network. We consider programmable transmitters that can apply N possible code rates to connections based on predicted bit error rate (BER) values. To classify the BER, we employ a preprocessing algorithm to feed the traffic data to a neural network classifier. We demonstrate the significance of our proposed preprocessing algorithm and the classifier performance for different values of N and switch port count.

Keywords: Data center network; Neural network; Adaptive transmission

1. Introduction

Programmable optical switching, based on software-defined networking (SDN), is being examined to resolve the scalability challenges of data centers and high-performance computing systems. By exploiting SDN in a data center network, optimizations can be performed to allow for data exchange with code-rate adaptive, high-order modulation schemes [1]. Code-rate adaptation is a promising technique for fine-tuning the physical-layer performance in communication networks [2] and adjusting the redundancy due to forward error correction [3]. This requires the possibility of monitoring or estimating the performance at the transceiver level and using the information collected from the physical layer to perform cross-layer scheduling [4]. Multiplicity of nodes, the complexities of models, and the scheduling constraints in data centers pose significant challenges to estimating the signal quality for code-rate adaptation.

With the recent advances in computing power and machine learning, it is now possible to support a wide range of services in communication networks, including performance monitoring, quality of transmission (QoT) estimation, failure detection, and resource allocation [5]. For instance, Rottondi et al. [6] propose a machine-learning framework to optimize the performance in long-haul optical networks by estimating the QoT of unestablished lightpaths. By taking into account important features such as traffic volume, modulation format, and route, their machine-learning algorithm determines whether the bit error rate (BER) of a candidate connection can meet the required system threshold. In [7] the probability distribution of the generalized signal-to-noise ratio (GSNR) is estimated by different regression methods in order to improve the deployment of unestablished optical networks.

While QoT predictions can also be employed to optimize the performance of data center networks, machine learning in data centers today is being considered only for the problems of flow classification and traffic scheduling. For example, [8] employ machine learning to detect mice and elephants and route them on data center links, considering an ideal physical layer. However, in a short-reach optical data center network, the accumulation of impairments along multiple switching elements can render a signal irretrievable at the receiver side. It has been shown that ignoring the physical-layer effects in such a data center can lead to expensive and impractical networking solutions [4]. As a result, it is imperative to come up with cross-layer scheduling algorithms that address the requirements in both the network and physical layers.

* Corresponding author.
E-mail address: kamrank@chalmers.se (K. Keykhosravi).
Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

https://doi.org/10.1016/j.icte.2022.02.002
2405-9595/

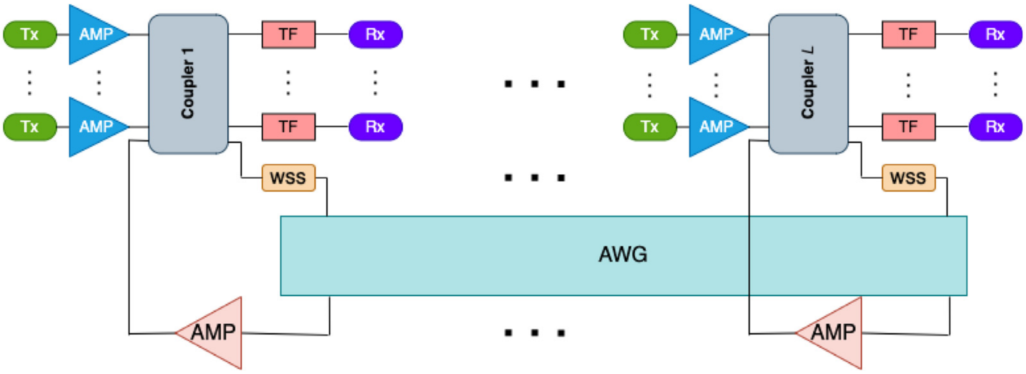

Fig. 1. Optical switching architecture based on AWG and star couplers, AWG: arrayed waveguide grating, WSS: wavelength-selective switch, TF: tunable filter.

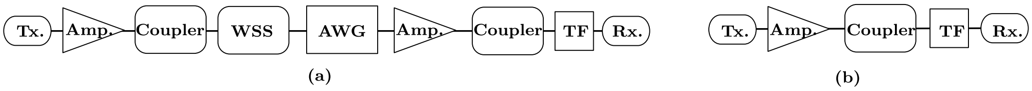

Fig. 2. Signal path for (a) a global, and (b) a local connection.

In this paper, we propose, for the first time, a cross-layer machine-learning framework to enable adaptive transmission in data centers. Instead of relying on purely theoretical models to estimate the QoT of lightpaths, our proposed algorithm can be trained with realistic data captured from the physical layer (for example, based on in-network BER measurements as in [9]). The machine-learning-based approach enables cross-layer scheduling without the need for complex or inaccurate physical-layer models. Furthermore, as opposed to model-based approaches, the proposed machine learning method is less affected by the uncertainties of the model parameters (such as amplifiers' noise figure, insertion losses, etc.), since it can learn these uncertainties. We address the problem of adaptive transmission in data centers by (1) employing pulse amplitude modulation (PAM) to generate the data required for training, validating, and testing a neural network (NN), (2) incorporating a novel preprocessing algorithm to refine the raw traffic data before feeding it to the NN, and (3) performing BER classification in a wavelength-routing scenario with a rich set of physical-layer impairments. In particular, we demonstrate the effectiveness of our preprocessing step by showing that the accuracy of the NN with preprocessing is orders of magnitude better than with a benchmark, in which the raw data is fed directly to the NN. Our cross-layer resource allocation framework can be generalized to other modulation formats and traffic patterns.

Notation: Vectors and scalars are denoted by lowercase letters (e.g., a) and matrices by capital ones (e.g., D). The ith element of the vector a is specified as a_i; also, the entry of the matrix D in the ith row and jth column is specified as $d_{i,j}$.

2. Switch architecture and impairments

To demonstrate machine-learning-based adaptive coding in a data center environment, we study an optical switch fabric that is operated in a realistic setting. These impairments include amplifier noise, linear interference, and filtering effects. Due to the short reach of data center networks, we disregard dispersion and nonlinear effects in our analysis.

We consider the switching architecture presented in Fig. 1, which is based on an arrayed waveguide grating (AWG) interconnecting L star couplers [4,10]. Each transmitter can modulate its signal to one of L different wavelengths. Each coupler broadcasts the signal at each of its entry ports to all of its outputs. The AWG guides the signal at its ith input port to its jth output if and only if the signal has the wavelength

$$w_{i,j} = \mathrm{mod}(i + j - 2, L) + 1. \tag{1}$$

Assuming an $L \times L$ AWG and L $K \times K$ star couplers, a wavelength-routing switch can support up to $M = L(K - 1)$ nodes. Consider a connection request from a node in coupler $i \in \{1, \dots, L\}$ to a node in coupler $j \in \{1, \dots, L\}$. The connection is local if $i = j$ and global if $i \neq j$. With local connections, the signal from the transmitter is broadcast to all nodes connected to the coupler. The tunable filter (TF) at the receiver is then tuned to the transmitted signal's wavelength and blocks other signals. With global connections, after being broadcast by the coupler, the signal passes though a wavelength-selective switch (WSS) to the AWG. Then according to its wavelength the signal is routed to its destination coupler and then to the receiver.

The signal paths of a global and a local connection are depicted in Fig. 2(a) and Fig. 2(b), respectively. The chain of components might include optical amplifiers, AWG, star

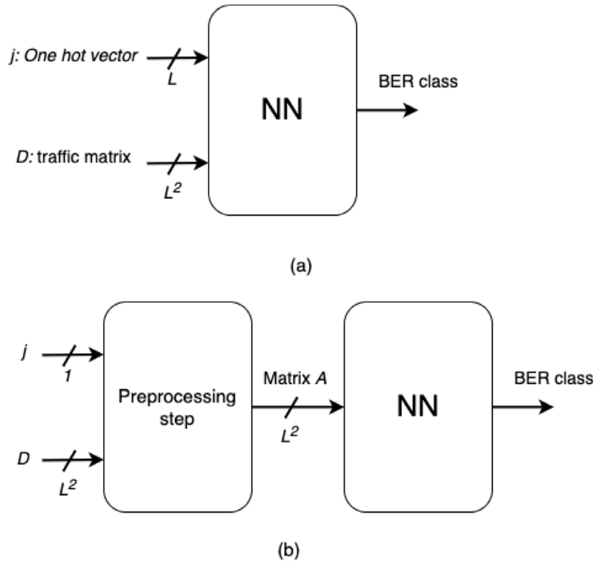

Fig. 3. Input and output for (a) fully ML-based approach (b) hybrid approach.

couplers, WSSs, and TFs. In this letter, we focus on global connections as they are more susceptible to impairments due to traversing a longer chain of optical components.

3. BER calculation methods

Without loss of generality, we focus on a global connection transmitted from node 1 of coupler 1 ($i = 1$) and destined to coupler $j \in \{2, \ldots, L\}$. Due to various hardware impairments, this connection encounters errors. There are three different ways to calculate or classify the connection's BER: *(i)* analytical physical models, *(ii)* fully ML-based approaches *(iii)* hybrid approaches. In the rest of this section we study these three methods and we explain how they are used in the paper.

3.1. Analytical physical model

In this approach, the physical properties of different elements of the switch are taken to consideration to model all the impairments and based on that the BER is calculated. Such a model is described in [4, Appendix] for M-PAM modulation, where the BER [4, Eq. (4)] is calculated by modeling various noise mechanisms, which can be classified into the following three classes:

- The *constant noises* added at the transmitter, receiver, and amplifiers (see Fig. 2(a)), which are independent of the scheduled traffic. These noises include thermal noise, shot noise, laser intensity noise, signal–spontaneous beat noise, and spontaneous–spontaneous beat noise, which can be calculated based on [4, Eqs. (6)–(10)], respectively.
- The *in-band interference* (linear crosstalk) induced by the co-existing signals in the AWG. These noises include signal-in-band crosstalk beat noise, in-band crosstalk–crosstalk beat noise, in-band crosstalk–spontaneous beat noise, whose variances can be calculated in [4, Eqs. (15)–(17)], respectively.
- The *out-of-band (OOB) interference* induced by other signals that appear in destination coupler j. It includes the OOB–OOB beat noise, whose variance can be found in [4, Eq. (18)].

The noises in Class 1 depend on the physical properties of the components in the switch and can be assumed constant for every global connection. The noises in Class 2 and 3 originate from imperfections in AWGs and TFs, respectively, and depend on the traffic traversing the switch. The AWG crosstalk (interferences in Class 2) can be calculated by keeping track of the number of connections from coupler l to coupler $k \neq l$ for all $l, k \in \{1, \ldots, L\}$ that satisfy

$$l + k \equiv i + j \pmod{L}. \tag{2}$$

This is due to the fact that if (2) holds, then $w_{i,j} = w_{l,k}$ according to (1). Therefore, the two signals propagate through the AWG with the same wavelength and interfere with each other due to AWG imperfections. To determine OOB interference in Class 3, one needs to know the type (global/local) of all connections destined to coupler j. Apart from this, the wavelength of these connections also impacts the OOB interference.

The model-based approaches in general have two shortcomings, (i) it may be unavailable (or hard to derive) for the considered switching architecture or the modulation (ii) it may not be accurate due to system parameters' uncertainties. Therefore, we need also data-driven ML-based approaches. In this paper, due to lack of experimental data, we use the analytical model explained in this section to calculated the BER for 4-PAM signal modulation. These data are used to train and test the two ML-based approaches, which are described in the following sections.

3.2. Fully ML-based approach

In this approach, we use the NN to classify the BER of the connection of interest (from coupler 1 to j) for a given set of traffic connection requests. As illustrated in Fig. 3 (a), the input to the NN is the destination coupler j as a one-hot vector[1] and an $L \times L$ traffic matrix D whose elements $d_{l,k}$ denote the number of connections from coupler l to coupler k. In order to manage the complexity of our machine-learning algorithm, we neglect wavelength occupancy information to calculate OOB interference. Neglecting this information prevents the algorithm from considering the frequency gap between the two connections in calculating the interference between them. Our results in Section 4 indicate that the NN performs well even without this information. For training and testing the NN, we calculate the BER for each example according to the analytical model described in Section 3.1. In this paper this approach is used as a benchmark.

[1] The one-hot vector is the standard format for inputs whose values do not indicate their importance.

$$D = \begin{bmatrix} 2 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 \\ 0 & 0 & 2 & 1 \\ 1 & 1 & 0 & 2 \end{bmatrix} \quad \tilde{A} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 1 & 0 & 0 & 3 \\ 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 1 \end{bmatrix} \quad A = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 2 & 0 & 0 & 3 \\ 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & 1 \end{bmatrix}$$

Fig. 4. Generating matrix A from D ($K = L = 4$, $i = 1$, $j = 3$).

3.3. Hybrid approach

In this section, we propose a hybrid approach, which benefits from the known properties of the switch structure, and specifically the AWG, and also uses an NN with a reduced interface to classify the BER. Here, instead of directly feeding j and D to the NN, we apply a preprocessing algorithm before the NN (see Fig. 3 (b)). In our proposal, the NN is fed an $L \times L$ matrix A, which is constructed by applying a permutation to D such that the in-band and OOB interferences can be calculated from A regardless of j.

To generate A, we first construct an auxiliary $L \times L$ matrix $\tilde{A}$ by applying a $(j-1)$-step circular rotation to the columns of D. Specifically, for all $m, n \in \{1, \ldots, L\}$ we let

$$\tilde{a}_{m,n} = d_{m,(n+j-2 \bmod L)+1}. \tag{3}$$

Equivalently, (3) can be written as

$$d_{l,k} = \tilde{a}_{l,(k-j \bmod L)+1}. \tag{4}$$

for all $l, k \in \{1, \ldots, L\}$. Specifically if l and k satisfy (2) with $i = 1$, we have that $k - j = 1 - l$ and hence

$$d_{l,k} = \tilde{a}_{l,(1-l \bmod L)+1}. \tag{5}$$

Therefore, the in-band interference can be calculated from $\tilde{A}$ regardless of j. Furthermore, as was mentioned earlier in this section, the OOB interference is a function of the number of local and global connections destined to coupler j. The former is equal to $d_{j,j} = \tilde{a}_{j,1}$ by (4). To make this independent of j, we switch the two elements $\tilde{a}_{j,1}$ and $\tilde{a}_{2,1}$. Specifically we build the matrix A as

$$a_{m,n} = \begin{cases} \tilde{a}_{2,1} & \text{if } (m,n) = (j,1) \\ \tilde{a}_{j,1} & \text{if } (m,n) = (2,1) \\ \tilde{a}_{m,n} & \text{otherwise.} \end{cases} \tag{6}$$

Also, the number of global connections destined to coupler j is $\sum_{l \neq j} d_{l,j} = \sum_{l \neq 1} \tilde{a}_{l,1} = \sum_{l \neq 2} a_{l,1}$. Therefore, the in-band and OOB interference can be calculated from A regardless of j (one can see that for all $j \neq 1$, (5) still holds if $\tilde{a}$ is replaced by a).

Example 1. Fig. 4 illustrates the process of generating A from a traffic matrix D for $K = L = 4$. In the example, the connection of interest is sent from coupler $i = 1$ to coupler $j = 3$. The matrix $\tilde{A}$ is generated by applying a 2-step circular rotation to the columns of D, and A is constructed by swapping elements $\tilde{a}_{2,1}$ and $\tilde{a}_{3,1}$ in $\tilde{A}$.

Table 1
BER thresholds in Example 2, computed to give uniform BER distributions.

N	BER ($K = L = 32$)	BER ($K = L = 16$)
2	$\{0, 10^{-5}, 1\}$	$\{0, 10^{-7}, 1\}$
3	$\{0, 10^{-6}, 10^{-4}, 1\}$	$\{0, 10^{-9}, 7 \cdot 10^{-7}, 1\}$
4	$\{0, 10^{-7}, 10^{-5}, 10^{-4}, 1\}$	$\{0, 10^{-12}, 10^{-8}, 10^{-6}, 1\}$

4. System evaluation

In this section, we detail our machine-learning simulations including data generation, NN design, training, validating, and testing the NN. In this section, we focus on the NN structure for the hybrid approach. The NN for the fully ML-based approach has the same structure but with an extended input layer (it has $L^2 + L$ inputs instead of L^2).

4.1. Data generation

To generate the data, we simulate the switch performance using a nonuniform Bernoulli traffic model and offline scheduling (see [4]). In a scheduling step, each node in each coupler generates a connection with probability $\rho \in \{0.1, 0.2, \ldots, 1\}$, except Node 1 of Coupler 1, which requests a global connection with probability 1. We focus on this connection and calculate its BER using the model in [4] for M-PAM.[2] All parameters are selected based on [4, Table I] (we do not consider any parameter variations). The calculated BER data is partitioned into N intervals of approximately the same size. The thresholds specifying the intervals are denoted by $\text{BER}_{\text{thd},0}, \ldots, \text{BER}_{\text{thd},N}$. For each step, based on the traffic requests, the matrix A is constructed using the description in Section 3. These data values along with the calculated BER classes are then used to train and test the NN.

We simulate 100 000 instances (10 000 per ρ) for each combination of N, K, and L. Then, we divide the data set into three distinct subsets for training (60%), validation (20%), and test (20%) (see [11, Sec. II-A] for definitions of these subsets).[3] Each generated connection is local with probability 0.75 and the destination is selected uniformly over all local/global nodes (with the exception that the source does not transmit to itself).

Example 2: Table 1 shows an example of the BER thresholds for different values of N, K, and L. The thresholds are calculated based on our data, such that the number of instances are almost the same within each BER class. Throughout the rest of the paper, we consider these values when classifying BERs.

[2] Note that our solution in this paper is not restricted by the modulation format. The same solution can be applied to coherent systems with other modulation formats such as quadrature amplitude modulation (QAM) by training the NN via experimental data [9].

[3] The validation data set is used to tune the hyperparameters of the NN. The test data set is never used in training and thus enables an unbiased evaluation of the final NN.

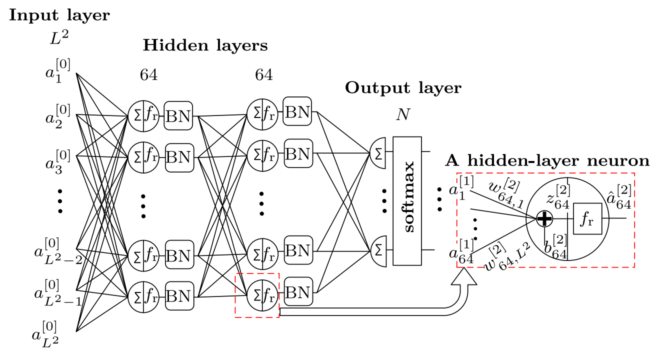

Fig. 5. NN architecture for the hybrid approach and the structure of a hidden-layer neuron. The fully ML-based approach follows the same NN architecture, but with an extended input interface (the number of inputs is $L^2 + L$ instead of L^2).

4.2. Neural network setup

To classify the BER in the data center switch, we consider an NN with two fully connected hidden layers (Fig. 5). There are 64 neurons in each hidden layer. The hyperparameters (e.g., the number of hidden layers and the number of neurons) were manually set to avoid over- or under-fitting. The NN is trained using the training data set, which comprises pairs of (A_i, y_i), where A_i is the ith input matrix and y_i is its corresponding BER class label. The input layer is fed with elements $a_1^{[0]}, \ldots, a_{L^2}^{[0]}$ of the $L \times L$ matrix A. Then linear combinations of the input layer are calculated within each neuron of the first layer, i.e.,

$$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]} \tag{7}$$

where the superscripts denote the layer number, $z^{[1]}$ is a vector of length 64, $W^{[1]}$ is the $64 \times L^2$ weight matrix, and $b^{[1]}$ is a bias vector of length 64. Next, the output of the kth neuron can be calculated as $\hat{a}_k^{[1]} = f_\text{r}(z_k^{[1]})$, where $f_\text{r}(\cdot)$ is the *relu* activation function, i.e., $f_\text{r}(z_k^{[1]}) = \max\{0, z_k^{[1]}\}$. To improve the performance and stability of the NN, we apply batch normalization (BN) to $\hat{a}^{[1]}$ to obtain $a^{[1]}$, i.e., $a_k^{[1]} = \text{BN}(\hat{a}_k^{[1]})$ for $k \in \{1, \ldots, 64\}$. The BN function applies an affine transform to its input with parameters that are learned during the training phase of the NN (see [12, Algorithm 1] for more information). In the second hidden layer, we obtain $a^{[2]}$ from $a^{[1]}$ in the same fashion as was explained for the first hidden layer. Finally, the output layer takes $a^{[2]}$ as an input and calculates $z^{[2]}$ the same way as in (7). Then, to achieve classification, we use *softmax* as the activation function. The function maps the input vector $z^{[3]}$ to the output vector $a^{[3]}$ such that $a_m^{[3]} = e^{z_m^{[3]}} / \sum_{k=1}^{N} e^{z_k^{[3]}}$. It appears that $a^{[3]}$ is a probability vector with length N. The label of the input vector is predicted by choosing the one that corresponds to the highest probability, i.e., $y = \arg\max_m a_m^{[3]}$.

We employ mini-batch gradient descent (see [13, Sec. 2.2]) with a batch size of 128 to train this network. To optimize the NN parameters ($W^{[j]}$ and $b^{[j]}$ for $j = 1, 2, 3$ and also the BN parameters), the algorithm runs over the training data multiple times using the outcome of the previous iteration as the initial state of the current iteration. Each iteration is called an epoch. We consider 30 epochs. We use the optimizer *Adam* [14] in order to minimize the loss function, which is set to categorical cross-entropy [13, Sec. 2.1.1].

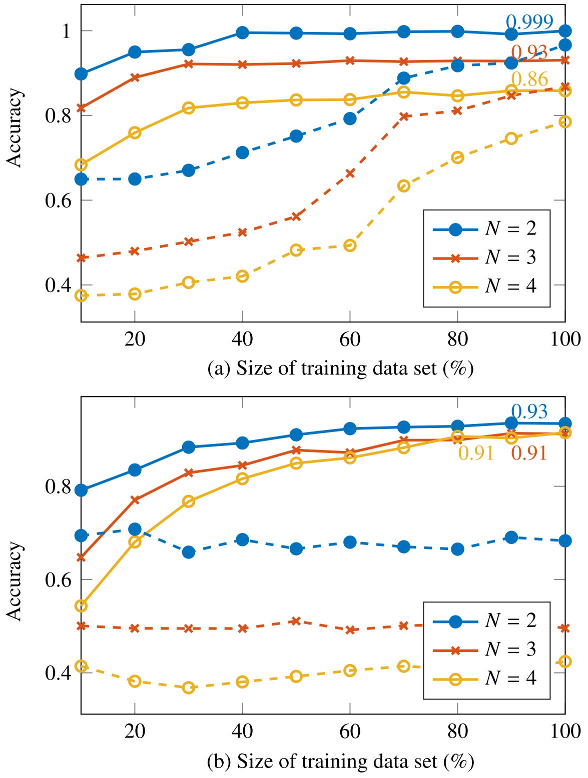

Fig. 6. Accuracy of test data for different percentages of training data: (a) $K = L = 16$, and (b) $K = L = 32$. The solid and dashed lines represent results with and without preprocessing, respectively.

4.3. Numerical results

We assess the performance of our physical-layer NN in terms of accuracy, i.e., the fraction of the examples classified correctly in each data set. All results are averaged over ρ. Fig. 6 depicts the accuracy of the NN on the test data for different values of N, where the size of the training data set varies between 10% to 100% of its original size (60% of the total data), and the sizes of validation and test data sets remain unchanged. We consider two switch sizes $K = L = 16$ in Fig. 6(a) and $K = L = 32$ in Fig. 6(b). Apart from the proposed *hybrid approach* (with preprocessing) described in Section 3.3, we present the accuracy of the *fully ML-based approach* (without preprocessing) described in Section 3.2 as a benchmark. As we show in this section, our proposal achieves a much higher accuracy than the benchmark because of the preprocessing performed on D and j to achieve A.

It can be seen that with the proposed scheme, a much higher accuracy can be achieved and a lower amount of data is required for training the NN. Also, the gap between the benchmark and our algorithm increases with the switch size. This is

Table 2
Confusion probabilities in percent between $N = 4$ BER classes, with and without preprocessing. The switch size is $K = L = 32$. Rows and columns indicate true and predicted BER classs, resp.

with preprocessing				without preprocessing			
21.1	1.9	0	0	13.3	7.6	4.2	0.8
1.9	18.7	1	0	8.5	6.2	5.2	1.6
0.0	2.1	26	0.5	6.4	6.4	9.8	6
0	0.0	1.2	22.6	1.4	2	6.6	13.9

due to the fact that, for the benchmark, the NN is unable to process the value of j properly to calculate the BER. However, with the proposed preprocessing step, given the matrix A, the dependence of BER to j is removed. The effectiveness of preprocessing becomes more pronounced when its negligible computational complexity is taken into account (it involves only circular rotation of the traffic matrix). Specifically, the limiting behavior of the computational complexity is the same for both the proposed method and the benchmark, that is $\mathcal{O}(L^2)$. Therefore, our comparison with the benchmark is fair. We note that, in general, any constant permutation does not affect the performance of a fully-connected NN. However, the permutation used in the preprocessing step is a function of j and therefore is different for different traffic connection requests. Finally, it can be seen that with increasing N the accuracy decreases. This is due to the fact that with larger values of N, the number of classes is larger and class borders become closer together, therefore, there is a higher chance that the predicted BER ends up in a wrong class.

Table 2 represents two confusion matrices for the case with $N = 4$ and $K = L = 32$, where we compare the results with and without the preprocessing step. An element of the confusion matrix in the ith row and the jth column represents the percentage of the examples in the ith class that were predicted to be in the jth one by the NN. It provides a more comprehensive metric for the NN performance than the accuracy as it illustrates the distribution of the classification errors. It can be seen that with the preprocessing step, in the case of an error, almost always the NN selects a class adjacent to the true one. However, without the preprocessing step, such characteristic is no longer present.

5. Concluding remarks

We proposed, for the first time, a BER classifier whose output can be used to adjust the coding rate in an optical data center network. We developed a preprocessing algorithm that converts data traffic values for optimal classifier operation. Our preprocessing algorithm consists of permutations on the traffic matrix, which depend on the coupler destination j. Based on our results, the NN cannot effectively learn this input-dependent permutation function, and therefore the proposed preprocessing step can have a significant impact on the classifier performance. This work illustrates how incorporating a stage of theoretical calculations can improve conventional machine learning-based designs that are directly fed raw data.

In this work, we used simulations based on a physical-layer model to acquire the data needed to train, validate, and test the NN. However, we did not use the model to design the NN structure. Our proposed algorithm can also be employed for different modulation formats or different traffic patterns. This is due to the fact that by altering the properties of the transmitted signal, the set of connections that interfere with each other do not change, but only the noises scale. This scaling can be learned by the NN via experimental data. Future work should examine the performance of the proposed algorithm using experimental data.

CRediT authorship contribution statement

Kamran Keykhosravi: Writing – original draft, Developing the method, Software, Visualization, Paper revision. **Ahad Hamednia:** Writing – original draft, Developing the method, Software, Paper revision. **Houman Rastegarfar:** Writing – original draft, Analysis, Paper revision. **Erik Agrell:** Supervision, Editing, Funding acquisition, Paper revision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the Knut and Alice Wallenberg Foundation, Sweden under Grant No. 2018.0090.

References

[1] Y. Ou, et al., Optical network virtualisation using multitechnology monitoring and SDN-enabled optical transceiver, J. Lightwave Technol. 36 (10) (2018) 1890–1898.

[2] D.A. Mello, et al., Optical networking with variable-code-rate transceivers, J. Lightwave Technol. 32 (2) (2014) 257–266.

[3] M.N. Sakib, O. Liboiron-Ladouceur, A study of error correction codes for PAM signals in data center applications, IEEE Photon. Technol. Lett. 25 (23) (2013) 2274–2277.

[4] H. Rastegarfar, et al., PAM performance analysis in multicast-enabled wavelength-routing data centers, J. Lightwave Technol. 35 (13) (Jul. 2017) 2569–2579.

[5] R. Gu, Z. Yang, Y. Ji, Machine learning for intelligent optical networks: A comprehensive survey, J. Netw. Comput. Appl. 157 (2020) 102576.

[6] C. Rottondi, et al., Machine-learning method for quality of transmission prediction of unestablished lightpaths, J. Opt. Commun. Netw. 10 (2) (2018) A286–A297.

[7] M. Ibrahimi, et al., Machine learning regression for QoT estimation of unestablished lightpaths, J. Opt. Commun. Netw. 13 (4) (2021) B92–B101.

[8] L. Wang, et al., Scheduling with machine-learning-based flow detection for packet-switched optical data center networks, J. Opt. Commun. Netw. 10 (4) (2018) 365–375.

[9] D. Zhuo, et al., RAIL: A case for redundant arrays of inexpensive links in data center networks, in: Symposium On Networked Systems Design And Implementation, NSDI, 2017, pp. 561–576.

[10] K. Keykhosravi, et al., Overcoming the switching bottlenecks in wavelength-routing, multicast-enabled architectures, J. Lightwave Technol. 37 (16) (2019) 4052–4061.

[11] D. Rafique, L. Velasco, Machine learning for network automation: Overview, architecture, and applications, J. Opt. Commun. Netw. 10 (10) (2018) D126–D143.

[12] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015, arXiv preprint arXiv:1502.03167.

[13] P. Murugan, Implementation of deep convolutional neural network in multi-class categorical image classification, 2018, arXiv preprint arXiv:1801.01397.

[14] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint:1412.6980.