Tandem Neural Networks for the Inverse Programming of Linear Photonic Processors

José Roberto Rausell-Campo
Photonics Research Labs, iTEAM
Universitat Politecnica de València
Valencia, Spain
joraucam@upv.es

Daniel Pérez-López
iPronics Programmable Photonics S.L
Valencia, Spain
daniel.perez@ipronics.com

Bhavin Shastri
Dept. of Physics, Eng. Physics and Astronomy
Queen's University
Kingston, Canada
bhavin.shastri@queensu.ca

Daniele Melati
Centre de Nanosciences et de Nanotechnologies
Université Paris-Saclay, CNRS
Palaiseau, France
daniele.melati@universite-paris-saclay.fr

***Abstract*—Linear programmable photonic integrated processors have emerged as an alternative hardware platform for quantum, deep learning, and microwave photonic systems. Calibration and control of the photonic processor using deep learning techniques has proven to be challenging due to the one-to-many problem. This means that a given functionality of the processor can be achieved using different settings of the controllers. In this paper, we demonstrate how tandem neural networks can overcome this limitation in meshes of Mach-Zehnder interferometers by employing forward and inverse networks. This approach is independent of the mesh architecture and introduces a novel method for controlling photonic linear processors using deep neural networks. We provide an experimental demonstration using a 3x3 linear processor, achieving a control resolution higher than 7 bits.**

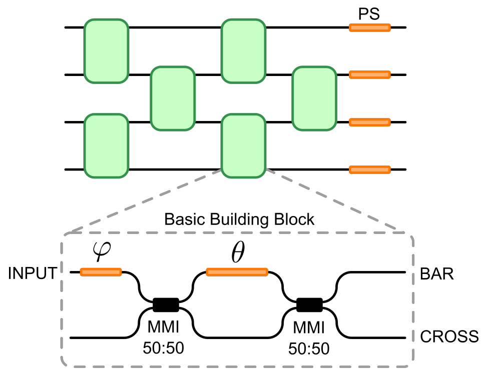
***Keywords*—integrated photonic circuits, deep learning, control algorithms, programmable photonics**

I. Introduction

There has been growing interest in the development of linear programmable photonic integrated processors that can serve as an alternative to current electronic systems [1]. Photonic processors are used as matrix multiplication accelerators in deep learning and wireless communications applications [2], performing linear operations "on the fly" in the optical domain. These photonic platforms can provide higher bandwidth, parallelism and lower latency compared to traditional processors. Moreover, linear photonic processors are among the main candidates for the future of quantum computing and quantum communication systems [3].

Different architectures for integrated photonic linear operators have been proposed. The most popular are those based on microring resonators (MRR) [4] and Mach-Zehnder Interferometers (MZI) [5]. When it comes to MZI-based meshes, they can be arranged in different architectures to implement any unitary linear transformation U(N). One of the most common used architecture is the rectangular or Clements one [6], which benefits from balanced insertion losses among the system. A general scheme is shown in Fig. 1. The basic building block consists of a tunable MZI and a phase shifter (PS) in one of the external arms. A final column of phase shifters enables the implementation of any U(N) transformation.

The work was supported in part by FPI-UPV Grant Program under Grant PAID-01-20-24 from the Universitat Politècnica de València, and in part by the H2020-ICT2019-2 Neoteric 871330 Project.



a)

Fig. 1. Linear programmable photonic processor using the Clements architecture. a) 4 x 4 matrix multiplier. Green boxes represent the basic building blocks of the system. Orange devices represent phase shifters. Basic building blocks consist of a tunable MZI with an external phase shifter.

To implement the desired matrix, electrical currents have to be applied to each of the phase shifters. Thus, precise calibration of each phase shifter is crucial. This entails having a precise mapping between the applied current and the implemented phase shift. In the case of thermo-optical phase shifters, the relationship between the applied current C and the phase shift θ is the following:

$$\theta = \theta_0 + \alpha C^2 \tag{1}$$

where θ_0 is the initial state when no current is applied, α the tunning coefficient that needs to be characterized and C the applied current. To calibrate the full system, a step by step procedure needs to be followed. In each step, one of the MZI structures is isolated from the rest of the system ensuring the input light exclusively goes through one of the input waveguides of the MZI, and the output light is guided to one of the outputs of the mesh. The applied current on the MZI is then swept and the optical power at the output is recorded. The dependence of the optical output power with the applied phase shift is:

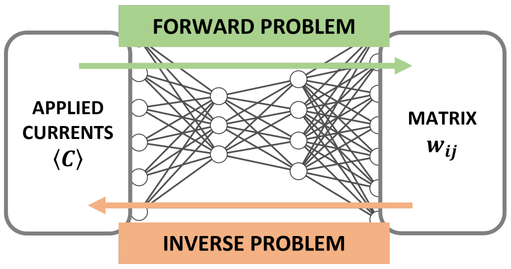
$$\begin{aligned} P_{cross} &= P_{in}\cos^2(\theta) \\ P_{bar} &= P_{in}\sin^2(\theta) \end{aligned} \tag{2}$$

where P_{in} represent the input optical power, and P_{cross} and P_{bar} represent the optical output power at the bar and cross port, respectively. The obtained curve is fitted using (1) and (2), and the free parameters θ_0 and α are estimated. The isolating procedure is architecture dependent. For the case of the rectangular mesh a calibration procedure has been proposed [7]. Once the calibration is finished, a decomposition process is necessary to relate the target matrix with the current that needs to be applied to each phase shifter in the mesh. This method usually assumes ideal components on each of the MZI, which is not the case in practice due to fabrication imperfections. These deviations from the ideal behaviour introduce discrepancies between the target matrix and the real matrix implemented on the chip. A calibration and decomposition algorithm to incorporate the effects of fabrication errors has been demonstrated in [8]. Nevertheless, other sources of errors such as thermal drift, thermal cross-talk and quantization errors cannot be characterized using this procedure.

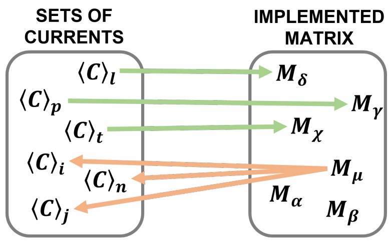
Recently, a data-driven model has been proposed to include the neglected effects in the previous approaches [9]. The authors demonstrate the use of a neural network based model for the calibration of a linear photonic integrated circuit without prior assumptions on the architecture of the chip. However, the proposed model only solves the so called forward or calibration problem. In the forward problem, a set of currents applied to the mesh is fed into the model and it predicts the expected implemented matrix on the chip, see Fig. 2 a).

It is still missing a way to solve the inverse or programming problem, which involves inputting the target matrix to be implemented into the model and obtaining the set of currents necessary for achieving it. The inverse problem is particularly relevant for applications in artificial intelligence or wireless communications but it is challenging to solve using deep learning models due to the one-to-many mapping. The one-to-many mapping appears because one target matrix can be obtained with many different sets of currents and this lack of uniqueness limits the learning capabilities of simple feed-forward neural networks. In the forward problem this mapping is one-to-one, as for one set of input currents there is only one possible output matrix. A schematic of these mappings is presented in Fig. 2 b). Solving the inverse problem would allow the control of photonic integrated circuits without use of the previously explained calibration and decomposition algorithms.

In this work, we propose the use of a tandem neural network (TNN) to solve the inverse problem. Tandem neural networks



a)



b)

Fig. 2. a) The forward problem takes as input the applied electrical currents and returns the expected matrix. The backward problem takes the target matrix as an input and predicts the set of electrical currents that are necessary to apply. b) Schematic of the one-to-many problem, one matrix can be implemented from different sets of currents.

consist of a forward model that is pre-trained to solve the calibration problem and one inverse network to solve the programming problem. TNN have been previously used for the inverse design of photonic devices [10], which faces the same one-to-many problem. We show how tandem neural networks can efficiently solve the one-to-many mapping using experimental data gathered from a 3x3 photonic processor with the Clements architecture. First, a fully-connected neural network is trained to solve the forward problem. Then, this trained neural network is connected with another fully-connected neural network, serving as the inverse model. The tandem network is then trained optimizing the loss between the target matrix inputted into the inverse network and the output matrix generated by the forward network.

II. Experimental Setup: Data Acquisition

For the acquisition of experimental data, we used a 3x3 linear mesh with the rectangular architecture. A photo of the fabricated device can be seen in Fig. 3. The photonic circuit is composed of three fundamental building blocks, each comprising a tunable MZI and a thermo-optical phase shifter. In this experimental demonstration, we disregarded the final column of phase shifters because direct photodetection was used. The chip was fabricated by Advanced Micro Foundry in a Silicon-On-Insulator platform.

Light is coupled into the chip through edge couplers and then divided into three paths using an integrated splitter tree. Prior to the rectangular mesh, an MZI column allows the modulation of the input vector that is subsequently multiplied by the photonic matrix. After multiplication, the outputs are photodetected on-chip and the resulting currents are recorded.

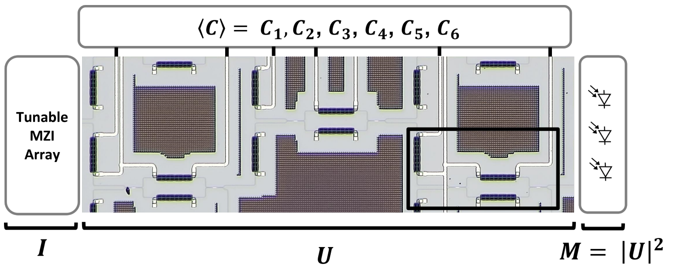


Fig. 3. Picture of the chip used to gather the data to train the tandem network. It is a 3x3 mesh consisting on 3 building blocks using the Clements architecture. The building block (black square) consists of one MZI and one external PS.

To generate the dataset for the tandem network, first we generate a set of 6 currents from a uniform random distribution between 0 and 0.94 mA, which corresponds to the half-period of the MZI. These currents are then applied to the phase shifters within the mesh using multi-channel current sources. To measure the implemented matrix on the mesh, we need to input the identity matrix (I on Fig. 3) so that after the multiplication and photodetection we can measure the implemented matrix $M = |U|^2 = IU$.

The identity matrix cannot however be directly encoded into the chip as we can only input 1x3 vectors. To circumvent this issue, for each of the samples on the dataset, we split the measuring process into three steps. In each step, one column of the identity matrix is encoded into the chip. At the output, the corresponding column of the photonic matrix is measured. After this process, the columns are merged and the full 3x3 matrix is obtained.

III. Tandem Neural Network: Training and Results

The proposed tandem neural network consists of two distinct fully-connected feedforward neural networks connected as depicted in Fig. 4. The forward model is first trained to predict the implemented matrix on the chip by minimizing the root mean squared error (RMSE) between the implemented matrix measured on the fabricated mesh, M_o, and the predicted matrix M_f. For our models, we use the squared of the currents instead of the currents, inspired by the physical behaviour of the thermo-optic phase shifters explained in (1).

The inverse model, has its outputs connected to the inputs of the forward model. The weights on the inverse model are optimized to minimize the RMSE between the matrix fed into the inverse model, denoted as M_i, and the predicted matrix by the forward model, M_f. During the training of the inverse model, the weights of the forward model remained fixed.

For both the forward and inverse model, we used a dataset of 7900 experimental samples gathered as explained in Section II. 70% of the dataset was used for training, 15% for validation and 15% for testing. Each of the samples consisted on a set of 6 currents and 9 weights. Both currents and weights were normalized between [-1,1]. The two models were trained using the L-BFGS optimizer and the Pytorch library. We

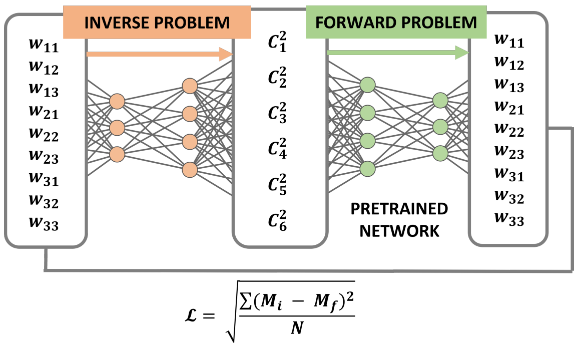


Fig. 4. Schematic of the tandem neural network used for the inverse programming of the photonic chip.

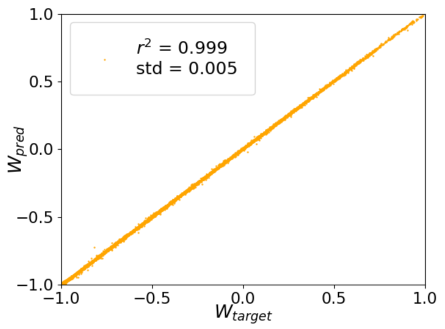


Fig. 5. Comparison between the weights predicted by the forward model and the experimental target weights.

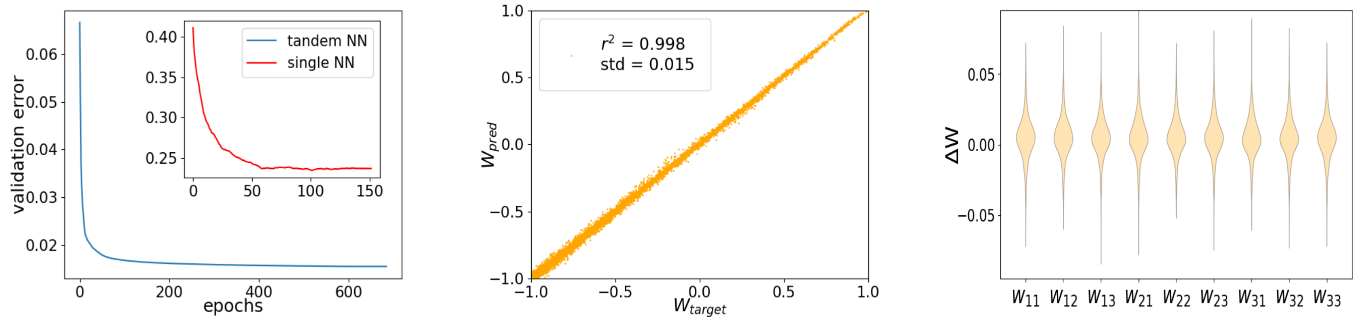
employed the hyperbolic tangent activation function between all the layers and the hyperparameters of the model were optimized using the Optuna framework and the validation set. The optimized hyperparameters are shown in Table I. Training is early stopped if the loss on the validation dataset does not improve more than 0.001 for more than 50 epochs, that means, one complete pass of the training dataset through the algorithm.

A. Forward Model

The forward model is trained during 464 epochs and achieved an RMSE of 0.0045. The results on the test dataset are shown in Fig. 5 where we compare the predicted weights with the experimental target weights. The standard deviation of the difference between the predicted and target weights is 0.005, which corresponds to a programming precision of 8-Bit [11].

B. Inverse Model

We trained the inverse model using the previously described TNN. The model is trained during 638 epochs and achieved an RMSE of 0.014. The importance of the tandem network to train the inverse model is appreciated in Fig. 6 a). In the single NN case we trained the same neural network used for the inverse model but without the pretrained network, trying to minimize the difference between the target and predicted currents. It can be seen that only using a single NN the learning



a) b) c)

Fig. 6. Tandem neural network results: a) difference between the validation error during the training of the inverse model when using a single feedforward network (red) and the proposed tandem network (blue), b) comparison between the predicted and target weights using the tandem network and c) distribution of the difference between the target weights and the predicted weights (ΔW) for each of the 9 weights of the 3x3 matrix.

TABLE I
MODEL HYPERPARAMETERS

	Forward Model	**Inverse Model**
# Layers	3	2
# Units per layer	190 - 172 - 101	388 - 95
Learning rate	1.056	1.019

capabilities of the inverse model are restricted. The training stops after less than 150 epochs with a validation error more than 15x higher than the obtained when using the TNN.

The weights predicted by the TNN are compared with the target weights that are fed into the inverse model. The results are shown in Fig. 6 b). The model achieves an r-squared of 0.998 and the difference between the target and predicted weights has a standard deviation of 0.015, corresponding to a programming precision $>$ 7-Bits. A detailed plot of the distribution of the errors for each of the 9 weights in the 3x3 matrix is presented in Fig. 6 c). All the weights present a precision higher than 7-Bits which demonstrate the ability of the proposed TNN to learn the structural pattern behind the training data. Once the inverse model is trained, it can be decoupled from the forward model and used to control the integrated photonic processor.

IV. CONCLUSION

In this work, we present the use of a tandem neural network to control linear integrated photonic processors. The TNN proves to be a powerful solution for addressing the one-to-many problem, where different sets of applied electrical currents can yield the same matrix output. The effectiveness of the models is demonstrated using experimental data taken from a 3x3 photonic mesh in a rectangular architecture. We compare the use of our model with a single network showing that the our proposal achieves results more than 15x better in terms of performance. Finally, the TNN can control the photonic mesh with a resolution higher than 7 bits avoiding the need of prior calibration and decomposition algorithms. To the best of our knowledge, this is the first time and inverse model is used for the programming of photonic processors. These results are architecture agnostic and provide a promising direction for employing deep learning models in the control of programmable integrated photonic devices for applications in artificial intelligence, wireless communications and quantum systems.

REFERENCES

[1] W, Bogaerts, D. Pérez, J. Capmany, D. Miller, J. Poon, et al., ”Programmable photonic circuits,” in Nature 586, 207–216, 2020.
[2] P. M. -C. Romero, J.R. Rausell-Campo, D. Pérez-Galacho, X. Li, T. Qing, et al., ”Integrated microwave photonics coherent processor for massive-MIMO systems in wireless communications,” in IEEE Journal of Selected Topics in Quantum Electronics, vol. 29, no. 6: Photonic Signal Processing, pp. 1-12, Nov.-Dec. 2023, Art no. 6101112.
[3] N. Peserico, B. J. Shastri and V. J. Sorger, ”Integrated photonic tensor processing unit for a matrix multiply: A Review,” in Journal of Lightwave Technology, vol. 41, no. 12, pp. 3704-3716, 15 June, 2023
[4] A. N. Tait, A. X. Wu, T. F. de Lima, E. Zhou, B. Shastri, et.al.,“Microring weight banks,” in IEEE Journal of Selected Topics in Quantum Electronics, vol. 22, no. 6, pp. 312–325, 2016.
[5] N. Harris, J. Carolan, D. Bunandar, M. Prabhu, M. Hochberg, Tom Baehr-Jones, et al., ”Linear programmable nanophotonic processors,” in Optica 5, 1623-1631, 2018.
[6] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walmsley, ”Optimal design for universal multiport interferometers,” in Optica 3, 1460-1465, 2016.
[7] C. Alexiev, J. C. C. Mak, W. D. Sacher, and J. K. S. Poon, ‘Calibrating rectangular interferometer meshes with external photodetectors,’ in OSA Continuum, vol. 4, no. 11, pp. 2892–2904, Nov. 2021.
[8] S. Bandyopadhyay, R. Hamerly, and D. Englund, ”Hardware error correction for programmable photonics,” in Optica 8, 1247-1255, 2021
[9] A. Cem, S. Yan, Y. Ding, D. Zibar and F. D. Ros, ”Data-driven modeling of Mach-Zehnder interferometer-based optical matrix multipliers,” in Journal of Lightwave Technology, to be published, doi: 10.1109/JLT.2023.3263235.
[10] D. Liu, Y. Tan, E. Khoram, and Z. Yu, “Training deep neural networks for the inverse design of nanophotonic structures,” in ACS Photonics, vol. 5, no. 4, pp. 1365–1369, 2018.
[11] W. Zhang, C. Huang, H. Peng, S. Bilodeau, A. Jha, et.al., ”Silicon microring synapses enable photonic deep learning beyond 9-bit precision,” in Optica 9, 579-584, 2022.