Online training and pruning of photonic neural networks

Weipeng Zhang[1,*], Tengji Xu[2], Jiawei Zhang[1], Bhavin J. Shastri[3], Chaoran Huang[2] and Paul Prucnal[1]

1. Department of Electrical and Computer Engineering, Princeton University, 41 Olden St, 08544, Princeton, USA
2. Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin N. T., Hong Kong SAR, China
3. Department of Physics, Engineering Physics & Astronomy, Queen's University, 64 Bader Lane, K7L 3N6, Kingston, Canada

weipengz@princeton.edu

2023 IEEE Photonics Conference (IPC) | 979-8-3503-4722-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IPC57732.2023.10360757

***Abstract*—Photonic neural networks have unique weight-actuating mechanisms and manufacturing variations, resulting in a suboptimal performance by conventional offline training. By incorporating a power-pruning regularization term in the loss function, we demonstrate an online training method that can overcome manufacturing errors and minimize power consumption.**

***Keywords*—Photonic neural network, neural network training**

I. Introduction

Photonic neural networks (PNNs) [1] offer superior speed and energy efficiency for computing tasks but are challenging to train accurately. Like human brains, no two PNNs are identical due to manufacturing variations. This leads to errors that accumulate within layers, even if designed to be the same. Although chip design optimization has improved error tolerance and robustness [2], it could be more effective. Online training, which iterates trainable parameters while monitoring actual PNN output, offers a more straightforward way to compensate for errors. Gradient-based [3] and gradient-free [4] algorithms have been used for online training, with the latter showing better error immunity. Furthermore, previous training models have overlooked a physical peculiarity of PNNs that the power consumption can differ with executed weights. The on-chip weighting actuators, including microring (MRR) and Mach-Zehnder interferometer (MZI), require different amounts of current (or voltages) for setting different weights. Therefore, optimizing the trade-off between applied current and network performance can help result in reduced power consumption, similar to" pruning" in digital electrical neural networks [5].

Here, we propose a training approach for PNNs that effectively addresses manufacturing errors and optimizes power consumption. Specifically, we demonstrate a gradient-free online training method based on particle swarm optimization (PSO) [6]. Moreover, we implement pruning for PNNs by incorporating an additional regularization term into the loss function to account for power consumption. We evaluate the proposed method through experiments on a 2x2 PNN and a simulation on a larger PNN with three layers and 804 random MRRs. Results indicate a one-third reduction in power consumption while maintaining high prediction accuracy.

II. Results

As a gradient-free training algorithm, PSO treats a potential solution of trainable parameters as a particle. In each iteration, a population of particles is tested on a PNN. The outcome is evaluated using a loss function, and PSO adjusts each particle based on its reward and that of its neighbors. The particles are randomly placed within the search space, bounded by the applicable current or voltages, until they converge. The velocity of each particle is calculated using empirically selected weighting factors, cognitive and social factors, and random factors. The cognitive factor reflects the particle's best position, while the social factor represents the best position found by the entire swarm. Random factors add exploration to the search.

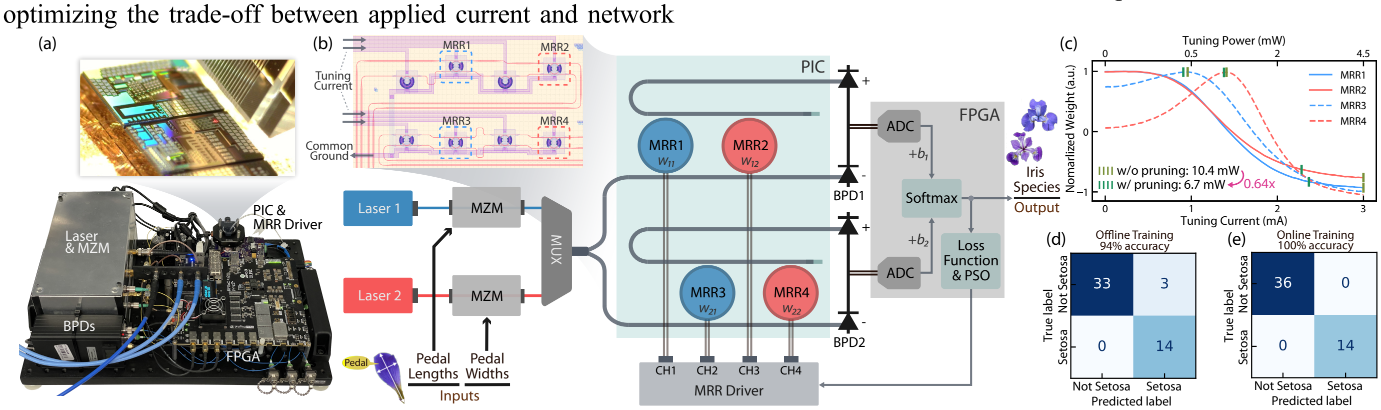

Fig. 1. (a) Photo and (b) schematic of the experimental setup. PIC, photonic integrated circuit. BPD, balanced photodetector. MUX, wavelength multiplexer. ADC, analog-to-digital converter. (c) Weight tuning curves of MRRs. MRR1 and MRR3 have the same diameter of 22.29 µm, and that of MRR2 and MRR4 are 22.32 µm. Vertical lines represent tuning current obtained by online training. Smaller currents are used by the pruning method, reducing power from 10.4 mW to 6.7 mW. (d) and (e) Confusion matrix for the 50 test samples resulting from offline and online training, respectively.

This research is supported by the National Science Foundation (NSF) (ECCS-2128616 and ECCS-1642962), the Office of Naval Research (ONR) (N00014-18-1-2297 and N00014-20-1-2664), and the Defense Advanced Research Projects Agency (HR00111990049).

The loss function comprises cross-entropy (CE) and a power-related regularization term, as $f_{\text{loss}} = \text{CE} + \lambda \Sigma_i I_i^2$. The CE is determined by the probability of each output item generated by a softmax function, which translates as the prediction accuracy of the PNNs. As we focus on MRR-based PNNs in this paper, the additional regularization term is a coefficient λ times the sum of the square of the applied current (in mA). This term is proportional to the total tuning power since the power consumed by the ith MRR is given by $I_i^2 \times R$, where R is the resistance of the heater. By minimizing the output of this loss function, the training algorithm considers both prediction and power performance, finding the optimal trade-off affected by the empirically determined λ.

We experimentally test the proposed training method on a classification task of two iris flower types using a modified iris dataset with only petal width and length features. The task is fitted into a 2x2 PNN with six trainable parameters: the weights of four MRRs, w_{11}, w_{12}, w_{21}, w_{22}, and two biases, b_1 and b_2. We randomly select 100 samples for training and 50 samples for testing. We modulate values of the petal widths and lengths onto laser 1 and 2, respectively, and combine the two laser lights before splitting them equally into two MRR weight banks. An FPGA analyzes the weighted addition output by built-in ADCs and then updates the tuning current by programming the MRR driver. The PSO algorithm is carried out via Python-coded software. The actual weights of all four MRRs against tuning currents (Fig. 1c) illustrate the misalignment of resonance frequencies between MRRs of the same diameters due to manufacturing variations. We train the same PNN under three conditions: offline, online with a regular loss function, and online with a power-pruning loss function. All three training results converge at 100% accuracy for the training samples. However, offline-trained PNN shows errors on the test samples due to incorrectly executed weights. In contrast, online-trained PNNs maintain error-free prediction, and the pruning results show a one-third reduction in power consumption.

Furthermore, we extend our evaluation to simulate a larger PNN that classifies images of handwritten digits ranging from 0 to 9. The PNN architecture comprises three layers with 1510 trainable parameters that include 1480 MRR currents and 30 biases. To simulate weight tuning curves for each MRR, we vary their resonance frequencies in a Gaussian distribution while maintaining the same transmission width, as illustrated in Fig. 2b. We compare the same three training methods as in the previous experiment and observe a lower prediction accuracy of 76% for the offline training. In contrast, online-trained models produce accuracies of higher than 93%. Notably, smaller tuning currents are used when applying the pruning loss function, resulting in a significant reduction in power consumption from 5W to 3W, while maintaining a high accuracy of 90%. These observations emphasize the importance of online training and power pruning as PNNs scale up and highlight the potential of our proposed method for future studies involving an increasing number of photonic neurons.

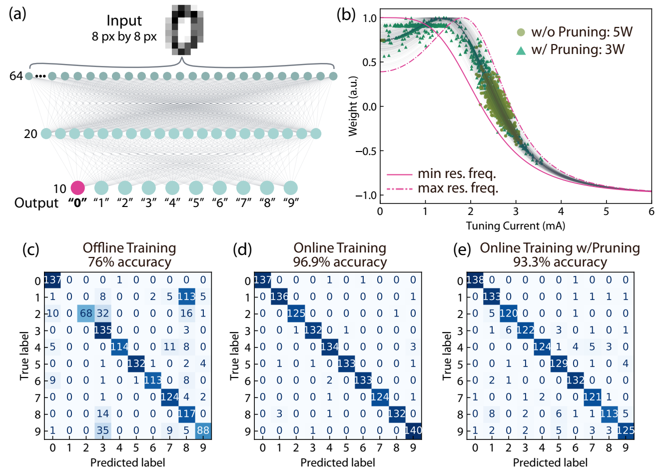

Fig. 2. Simulation on MNIST dataset. (a) PNN architecture has one hidden layer with 20 nodes fully connected to the input and output layers. (b) Weight tuning curves and trained current of all the 1480 MRRs. With pruning, the actuated tuning currents, shown as deep green triangles, are smaller than the non-pruning case, shown as light green dots. (c)-(e) Confusion matrice resulted from three training methods.

III. Conclusion

In conclusion, our proposed online training mechanism for PNNs enables the self-correction of manufacturing errors and minimizes inference power budgets. This approach can be extended to a more efficient learning process, where offline learning provides a starting point for PNNs, and individual PNNs further optimize their parameters through online learning. This methodology allows for transferable knowledge, such as weights obtained through offline learning, and non-transferable knowledge resulting from self-adaptations. We anticipate further investigation of exploiting the co-packaged FPGAs for real-time learning rates, resulting in greater versatility and adaptiveness. Our online training demonstration will serve as a methodology foundation for future PNN studies, enhancing their ability to address real-world tasks.

References

[1] G. Mourgias-Alexandris, M. Moralis-Pegios, A. Tsakyridis, S. Simos, G. Dabos, A. Totovic, N. Passalis, et al. "Noise-resilient and high-speed deep learning with coherent silicon photonics," Nat. Commun., vol. 13, pp. 1-7, 2022.

[2] R. Hamerly, S. Bandyopadhyay and D. Englund, "Asymptotically fault-tolerant programmable photonics," Nat. Commun., vol. 13, pp. 1-7, 2022.

[3] T. W. Hughes, M. Minkov, Y. Shi and S. Fan, "Training of photonic neural networks through in situ backpropagation and gradient measurement," Optica, vol. 5, pp. 864-871, 2018.

[4] M. J. Filipovich, Z. Guo, M. Al-Qadasi, B. A. Marquez, H. D. Morison, V. J. Sorger, et al. "Silicon photonic architecture for training deep neural networks with direct feedback alignment," Optica, vol. 9, pp. 1323-1332, 2022.

[5] H. Zhang, J. Thompson, M. Gu, X. Jiang, H. Cai, P. Y. Liu, et al, "Efficient On-Chip Training of Optical Neural Networks Using Genetic Algorithm," ACS Photon., vol. 8, pp. 1662-1672, 2021.

[6] J. Kennedy, R. Eberhart, "Particle swarm optimization," Proc. of international conference on neural networks, vol. 14, pp. 1942-1948, 1995.