Supplementary information: Fully analog end-to-end online training with real-time adaptability in integrated photonics

Zhimu Guo[1*], A. Aadhi[2], Ammar A. Ibrahim[1], Adam N. McCaughan[3], Alexander N. Tait[4], Nathan Youngblood[5], Sonia M. Buckley[3], Bhavin J. Shastri[1,4*]

[1*]Centre for Nanophotonics, Department of Physics, Engineering Physics, and Astronomy, Queen's University, Kingston, K7L 3N6, ON, Canada.
[2]Optics and Photonics Centre, Indian Institute of Technology Delhi, New Delhi, 110016, India.
[3]National Institute of Standards and Technology, Boulder, 80305, CO, USA.
[4]Smith Engineering, Department of Electrical and Computer Engineering, Queen's University, Kingston, K7L 3N6, ON, Canada.
[5]Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, 15261, PA, USA.

*Corresponding author(s). E-mail(s): 15zg11@queensu.ca; shastri@ieee.org;

1 Multiplexed gradient descent on integrated silicon photonic platform

The multiplexed gradient descent (MGD) algorithm was originally proposed by A. Dembo *et al.* [1] and J.C. Spall [2], and has been adapted to simulate the analog computing platforms in recent works [3, 4]. There are a variety of methods to calculate the gradient, and the most common method expresses the gradient as a collection of all partial derivatives of the loss function $L(t)$ with respect to each network parameter, also known as weights, w_i,

$$\nabla_W L(t) = \left(\frac{\partial L(t)}{\partial w_1}, \frac{\partial L(t)}{\partial w_2}, \frac{\partial L(t)}{\partial w_i}, ... \right), \tag{1}$$

where the loss function is calculated using the network output $y(t)$ and the training target $\hat{y}(t)$. The weight update that follows the next iteration subtracts the product of the calculated gradient value and the learning rate from the current weight values,

$$W(t+1) = W(t) - \eta \nabla_W L(t), \tag{2}$$

where $W(t+1)$ is the weight vector at the next training epoch, $W(t)$ is the weight vector at the current epoch, and η is the learning rate. The collection of partial derivatives in Eq 1 is commonly computed via large-scale matrix multiplications in digital training. Here, we implemented it directly on our photonic processor and achieved training performance competitive with digital training, despite its low bit precision. We also emphasise that 1-bit precision can be used to implement a large language model (LLM), thereby opening the possibility of implementing a photonic LLM.

Here, the weight perturbations that we introduce are a collection of weight-specific, time-varying functions, $\widetilde{W}(t)$,

$$\widetilde{W}(t) = \sum_i \widetilde{w}_i(t), \text{ where } \widetilde{w}_i(t) = \Delta w_i f_i(t), \tag{3}$$

Δw_i is perturbation amplitude of weight w_i, and $f_i(t)$ is a normalized, time-dependent signal for w_i in specific. Here i is the dummy index representing the i-th perturbation. The outputs measured with the presence of the perturbations also contain the effects of weight perturbations, which translates to the loss function

$$\widetilde{L}(t) = L_0 + \sum_i \Delta L_i(t), \tag{4}$$

L_0 is the baseline loss function measured without the weight perturbations, and $\Delta L_i(t)$ is the time-varying amplitude of the perturbation at weight w_i which can be identified by matching the weight-specific perturbation function $\widetilde{w}_i(t)$. The gradient can be approximated as

$$\nabla_W L(t) \approx \left(\frac{\Delta L_i}{\Delta w_i}, \frac{\Delta L_{i+1}}{\Delta w_{i+1}}, \frac{\Delta L_{i+2}}{\Delta w_{i+2}}, \ldots \right), \tag{5}$$

which connects the amplitude of the perturbation that we introduced to our system and the loss function.

The time-dependent signal, $f_i(t)$, in the weight perturbation $\widetilde{w}_i(t)$ can be any form of time-varying signal, and in our demonstration such signal is implemented as a sequence of randomly generated binary choices on a per-batch basis, as a result, the perturbation is constant during a batch, but it varies with different batches. Here we measure the gradient and update the weights after every batch, therefore, we maintain the weight perturbation to be constant for this short period $\widetilde{w}_i(t) \to \widetilde{w}_i(t_0)$, implying that the variation in the loss function also remains constant for the same duration, $\Delta L_i(t) \to \Delta L_i(t_0)$. As a result, the normalized instantaneous gradient for weight w_i becomes

$$\nabla_{w_i} L(t) \to \nabla_{w_i} L(t_0) \approx \frac{\Delta L_i(t_0) \cdot \widetilde{w}_i(t_0)}{\Delta w_i^2}. \tag{6}$$

We can update each weight value w_i with a product of the loss function variation and the weight perturbation, both using their instantaneous values at the time of measurement

$$w_i^{(t+1)} = w_i^{(t)} - \eta \frac{\Delta L_i(t_0) \cdot \widetilde{w}_i(t_0)}{\Delta w_i^2}. \tag{7}$$

The workflow implementing the above derivations is shown in Algorithm 1 below.

Algorithm 1 Perturbative training algorithm

```
Initialize parameters W
for n in epochs do
    for m in batches do
        Set perturbations W̃ ⇐ 0
        Measure baseline loss function L_0 ⇐ L(f(x, W), ŷ)
        Set perturbations W̃ ⇐ choice(−1, 1)
        Measure perturbed loss function L̃ ⇐ L(f(x, W̃), ŷ)
        Obtain change in loss function ΔL ⇐ L̃ − L_0
        Update gradient approximation G ⇐ Σ_i ΔL_i/Δw_i^2
        Update parameters W ⇐ W − ηG
        Reset gradient approximation G ⇐ 0
    end for
end for
```

In addition to concurrent training and inference functionality and fast adaptability, we also verify the stability of our system and measure the latency at various experimental steps within our setup. Unlike static, offline-trained systems, stability is challenging for adaptive systems because they typically sacrifice stability to quickly update parameters to meet the demands of a rapidly changing

environment. In our initial demonstration of the basic 2D classification example, we have already proven that our analog end-to-end adaptive system is able to maintain the same level of performance even after training has converged. However, we observed stability degradation during the online tracking demonstration when implementing the exact same learning steps, which is attributed to the constant learning rate and perturbation amplitude used throughout the training trials. Therefore, we implemented diminishing learning rate and perturbation amplitude to significantly improve stability [2]. The learning rate and the perturbation amplitude of the $(n+1)^{\text{th}}$ epoch are calculated based on the training accuracy of the current n^{th} epoch,

$$\begin{aligned} \eta_{n+1} &= \eta_0 \cdot (1 - acc_n) \cdot 25, \\ \Delta w_{n+1} &= \Delta w_0 \cdot (1 - acc_n), \end{aligned} \tag{8}$$

where η_0 is the initial learning rate, Δw_0 is the initial perturbation amplitude. As a result, we achieve both fast convergence and stable performance at every angle during the online tracking demonstration.

2 Experimental setup

Our MRR-based integrated silicon photonic platform is fabricated by Advanced Micro Foundry in Singapore, and the printed circuit boards (PCBs) that we use to package our silicon photonic chip are manufactured by PCBWay. The most daunting challenges are the fabrication variations at the device level in our silicon photonic processor, the custom PCB layout that provides all the connectivity required for our system-level demonstrations, the component selection for the construction of the transmission line, and the FGPA firmware that encodes/decodes the inputs and outputs and performs all training operations to complete an analog end-to-end training system.

In our experimental setup we use an RFSoC FPGA (Xilinx RFSoC 4×2) to modulate the inputs and measure the outputs from our integrated photonic weight bank, whose weights are controlled using 4 SMU channels (Keithley 2606B). RF amplifiers (Mini-Circuits ZX60-14012L-S+) are used to boost the output photocurrent, and bias-tees are employed to isolate the DC bias, which is measured with another SMU channel. The dataset used for both online tracking and stability validation comprises 20,000 samples, divided into batches of 100 samples each. Every sample has two parameters representing its coordinates in the 2D domain, and each parameter is modulated by a DAC on the RFSoC FPGA. Every training epoch iterates through all the batches in the dataset, and every batch starts by instantiating all input samples within a batch using the two DACs on the RFSoC FPGA. First we measure the output signal which gives us the baseline loss function values. After perturbing the weights, we measure again the loss function with variations, and the difference between the perturbed and unperturbed loss function corresponds to the gradient.

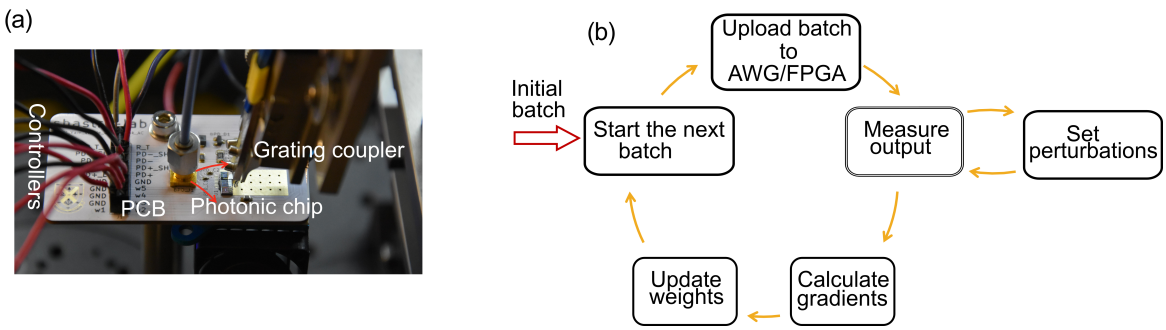

Fig. 1 **Photonic device and the training flowchart:** Packaged photonic chip and the training workflow showing the training procedure. The fully integrated photonic device is mounted on the PCB with low-speed and high-speed controllers along with the grating coupler.

3 Linear and non-linear classification results

We demonstrate linear and nonlinear classification problems using our analog end-to-end adaptive training system. For the linear classification, the learning rule is defined by an arbitrary line $(a \cdot$

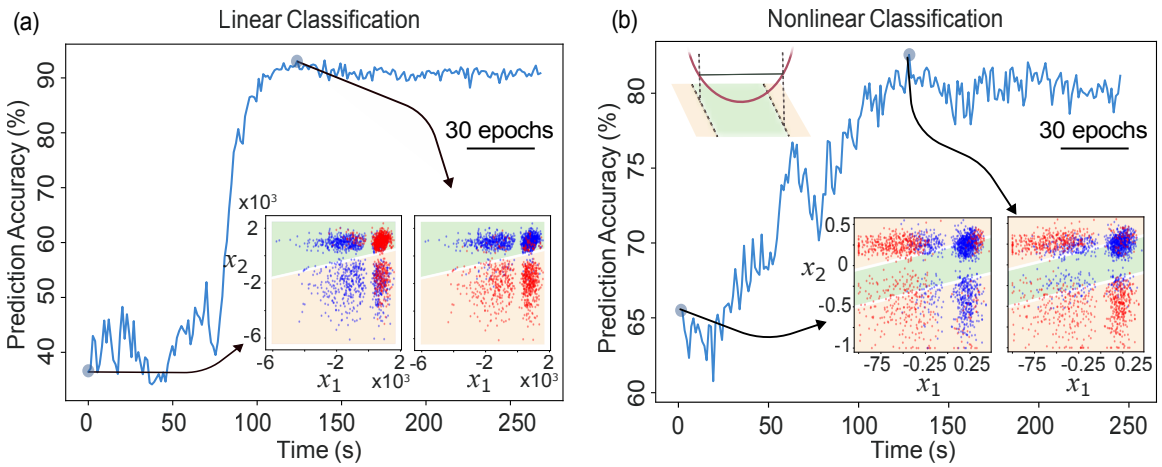

Fig. 2 Analog end-to-end training results: Experimentally measured training accuracy values in both linear and nonlinear classification problems, plus the system stability validation results. a) Linear classification of two sets of data trained over 200 epochs, and both the initial and the optimal accuracy results are shown in the inset scatter plots. The classification rule is defined by an arbitrary line ($0.0137 \cdot x_1 - 1 \cdot x_2 = 0$). The background colors indicate the correct classification results, and the color of the data points represents the results produced by the network. b) Nonlinear classification with a quadratic separation of two sets of data, also trained over 200 epochs. The nonlinear classification rule is defined by an arbitrary parameter ($(0 \cdot x_1 + 1 \cdot x_2)^2 - 0.05 = 0$), which can be visualized as the projection of a parabola onto a 2D plane as shown in the inset. The classification results are shown in the two scatter plot insets, where the background colors indicate the correct classification results, and the color of the data points represents the results produced by the network.

$x_1 - b \cdot x_2 = 1$), where a and b are real numbers. In the experiment, all the network parameters are initialized to random values, and the gradient is measured by comparing baseline and perturbed loss function values by without and with perturbations to the network parameters, respectively. These parameters are updated after each batch, and training continues until convergence is reached. As shown in Fig 2(a), our system achieves convergence in under 100 seconds. We train the system for 200 epochs, achieving a prediction accuracy exceeding 90% with no significant fluctuations. Furthermore, we have trained the system for a nonlinear classification task using the same approach. The nonlinear classification was implemented by introducing an arbitrary quadratic nonlinearity ($(a \cdot x_1 + b \cdot x_2)^2 - 0.05 = 1$) at the photodetector output. Unlike linear classification, the output of the quadratic function exists in the third dimension, where it intersects the original 2D plane to form a band that defines class separation, as illustrated in Fig. 2(b) insets. For the experiment, we follow the same procedure used in the linear classification case, with the addition of the quadratic nonlinearity programmed on the RFSoC FPGA. As shown in Fig. 2(b), our system reaches convergence within 100 training epochs, achieving a maximum accuracy of above 80%. After convergence, the prediction accuracy remains stable, with a standard deviation of 1.06%. In comparison, the simulation shows an accuracy of 100% for linear and nonlinear classification tasks, respectively. The performance can be improved by adjusting the learning rate and incorporating higher nonlinearity in the hardware.

4 Digital Twin model

In our digital twin model, the simulated photonic neural network is a $4 \times 30 \times 3$ classifier network implemented on a single 30×30 microring matrix multiplication core that is reused once per layer (Fig. 3). Each neuron has a weight bank of 30 add-drop MRRs. Every optical element is modelled as a 4×4 scattering transfer matrix (S-matrix), which relates the input and output complex electric fields [5, 6]. The S-matrix is converted to an A-matrix, which can be multiplied with the A-matrices of the subsequent cascaded elements to provide the combined response. The response of a weight bank at one wavelength is the product of the A-matrices of the MRRs and the interconnecting waveguides between them,

$$\mathbf{A} = \left(\prod_{i=1}^{N-1} \mathbf{A}_{\mathrm{WG}} \, \mathbf{A}_{\mathrm{MRR},i} \right) \mathbf{A}_{\mathrm{MRR},N}, \tag{9}$$

which is converted back to an S-matrix to read the thru t_{thru} and drop t_{drop} ports transmission of the entire weight bank [6]. The realized weight is the output of the balanced detector, which connects

and subtracts the thru and drop ports of the weight bank.

$$w = G\,R_p\,(t_{\text{drop}} - t_{\text{thru}})\,, \tag{10}$$

with responsivity R_p and the gain of the following amplification stage G.

For an add-drop ring of radius R at wavelength λ, with round-trip amplitude transmission α and thru coupling coeffecients $t_{1,2} = \sqrt{1-\kappa_{1,2}^2}$, the round-trip phase is $\theta = 4\pi^2 n_{\text{eff}} R/\lambda$. Writing $a = \alpha e^{j\theta}$ and $\Delta = 1 - a\,t_1 t_2$,

$$T_1 = \frac{t_1 - a\,t_2}{\Delta}, \qquad T_2 = \frac{t_2 - a\,t_1}{\Delta}, \qquad D = \frac{-\sqrt{\alpha}\,e^{j\theta/2}\,\kappa_1\kappa_2}{\Delta}, \tag{11}$$

and the MRR's A-matrix is

$$\mathbf{A}_{\text{MRR}} = \begin{bmatrix} 1/T_1 & 0 & 0 & -D/T_1 \\ 0 & (T_1T_2 - D^2)/T_2 & D/T_2 & 0 \\ 0 & -D/T_2 & 1/T_2 & 0 \\ D/T_1 & 0 & 0 & (T_1T_2 - D^2)/T_1 \end{bmatrix}. \tag{12}$$

The heater control n_{eff}, and therefore changes θ. The bus waveguide between two rings is modelled as a diagonal matrix

$$\mathbf{A}_{\text{WG}} = \text{diag}\big(e^{j\varphi}, e^{-j\varphi}, e^{j\varphi}, e^{-j\varphi}\big)\,, \qquad \varphi = (\beta - j\alpha_{\text{wg}})L_{\text{WG}}, \tag{13}$$

with $\beta = 2\pi n_{\text{eff}}/\lambda$ and field attenuation α_{wg}.

The cascaded $\mathbf{A}$ of (9) is converted to a scattering matrix following the generalized four-port analysis of Schwelb [7]. The transfer matrix $\mathbf{T}$ is formed and partitioned into 2×2 blocks,

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_A & \mathbf{T}_B \\ \mathbf{T}_C & \mathbf{T}_D \end{bmatrix} = \Pi_{23}\,\mathbf{A}\,\Pi_{23}, \qquad \Pi_{23} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{14}$$

The S-matrix is

$$\mathbf{S} = \begin{bmatrix} \mathbf{T}_C\mathbf{T}_A^{-1} & \mathbf{T}_D - \mathbf{T}_C\mathbf{T}_A^{-1}\mathbf{T}_B \\ \mathbf{T}_A^{-1} & -\mathbf{T}_A^{-1}\mathbf{T}_B \end{bmatrix}. \tag{15}$$

Numbering the ports 1 (in), 2 (drop), 3 (thru) and 4 (add), the block $\mathbf{T}_A^{-1}$ holds the transmissions from the input side to the thru and add ports, and the block $\mathbf{T}_C\mathbf{T}_A^{-1}$ includes the response returning to the input side, where the drop port is. The two transmissions read out of the weight bank are therefore $t_{\text{thru}} = |S_{31}|^2$ and $t_{\text{drop}} = |S_{21}|^2$ [5].

Ring weights and activation

The heater tunes the resonant wavelengths of the microrings by changing the effective refractive index. The simulation model uses a fourth-order fit to experimentally measured thermo-optic data [8],

$$\Delta n_{\text{eff}}(I) = \sum_{m=0}^{4} a_m I^m, \tag{16}$$

with $a_0 = 1.058 \times 10^{-6}$, $a_1 = -2.99 \times 10^{-2}$, $a_2 = 1.142 \times 10^3$, $a_3 = -2.48 \times 10^5$, and $a_4 = 2.00 \times 10^8$ for the current I over the 0 to 0.8 mA range. The rings are stepped by $\Delta R = 3.4$ nm in radius, which places 30 resolved channels between 1530 and 1549 nm at a spacing of 0.65 nm, each channel sitting on its own ring's cold resonance (Fig.4(a)).

The wavelength response of the weight bank is not uniform across the band. The peak transmission attenuates along the bus, which gives the rings towards the end of the weight bank a lower transmission. Adjacent rings interfere coherently through the shared buses, where two MRRs would form a compound cavity, producing irregularities in the lineshapes of the MRRs.

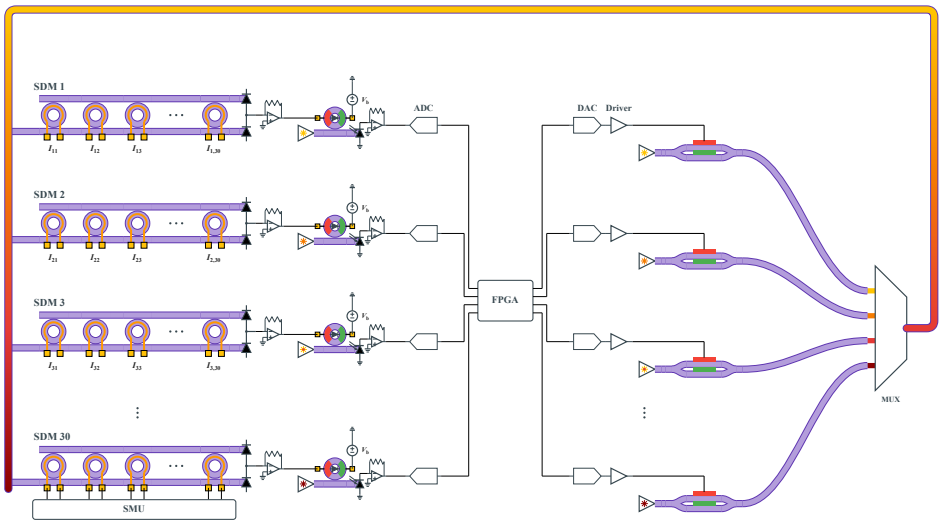

Fig. 3 The 30×30 matrix multiplication core. Each row is a 30 MRRs weight bank, followed by a balanced photodetector, a transimpedance amplifier, and an electro-optic activation MRR. The data is converted to the digital domain, processed by an FPGA, and then the weights are updated accordingly. The input to the optical system is converted to the optical domain through the Mach-Zehnder modulators on different wavelengths and then finally multiplexed and broadcast to all the weight banks.

Rings per bank	30	Responsivity (R_p)	0.8 A/W
Ring radius (R)	8 µm	Transimpedance gain (R_{TIA})	2.4 kΩ
Radius increment (ΔR)	3.4 nm	Activation operating point	1.62 V
Bus waveguide between rings (L_{WG})	70 µm	Heater current range	0–0.8 mA
Field cross-coupling ($\kappa_{1,2}$)	0.174	Heater settling time	5 µs
Round-trip transmission (α)	0.9988	Heater current DAC	16 bit (12 nA)
Waveguide loss (bus and ring)	2 dB/cm	Input DAC	14 bit
Effective index (n_0)	2.435	Readout ADC	11 ENOB
Channel span	1530–1549 nm	Detection bandwidth (B)	1 GHz
Channel spacing	0.65 nm	TIA noise	5 pA/$\sqrt{\mathrm{Hz}}$
Optical power per channel	0.9 mW	Laser RIN	−150 dB/Hz, 25 dB CMRR at detector
Heater resistance	500 Ω	Thermal crosstalk	3 pm/mW, 50 µm

Table 1 Simulation parameters for the photonic neural network.

The activation function is a PN-junction microring modulator driven by the output of the amplification stage of the weight banks output [8, 9] which sets the junction's voltage as $V_m = R_{\mathrm{TIA}} I_{\mathrm{BPD}} + V_b$, which shifts both the effective refractive index and the attenuation of the field in modulator ring through experimentally measured lookup tables [8]. The input is biased with a constant DC voltage at its steepest slope around 1.62V (Fig.4(b)) as an offset, so zero photocurrent gives zero output.

Performance comparison

For the performance comparison, the photonic core, a CPU (Intel Core Ultra 9 185H), and a GPU (NVIDIA A100) were used to solve the same Yin-Yang classification task as a benchmarking problem. The training algorithm and data were unified across the three platforms, with 500 samples and 200 validation samples. Each epoch involved two forward passes: one for the positive perturbation and one for the negative perturbation. The training time for the photonic core was based on the calculated time budget for each epoch in Table 2, while the training time for the CPU and GPU was based on the measured wall times.

According to our digital twin model, the photonic chip takes around 32.33 μs for a single epoch, because 30 μs of the time budget is taken up by the heaters' thermal settling time. The settling time for the heaters is 5 μs [10, 11], and the heaters are updated three times: the first two for the positive and negative perturbations and the third to set the actual weight. Since the photonic core is reused

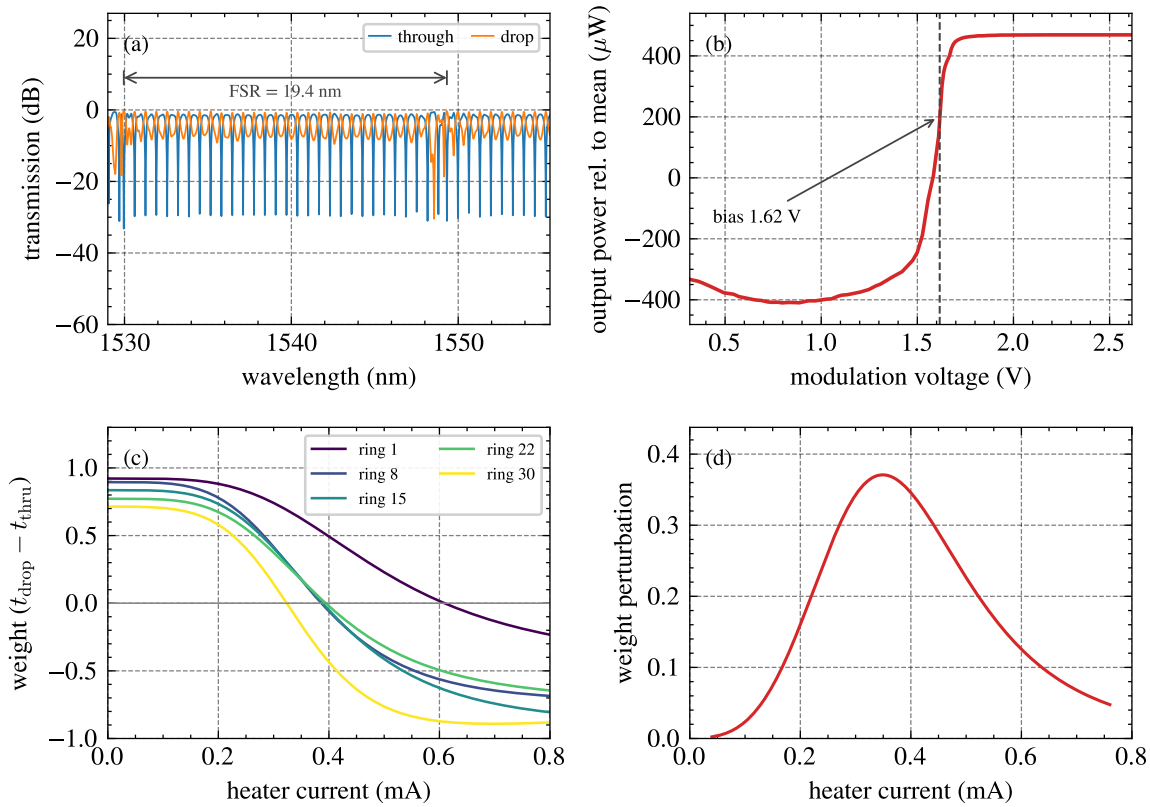

Fig. 4 (a) Optical response of one 30 MRR weight bank with all heaters off. The free spectral range spans 19.4 nm, and the peak drop transmission falls along the bus, from −0.35 dB at the first ring to about −1.45 dB at the last. (b) Electro-optic activation transfer, plotted relative to its rest point and biased at the steepest slope near 1.62 V. (c) the realized weight ($t_{drop} - t_{thru}$) of each MRR as a function of the current injected through the heaters. (d) the weight perturbation produced by a fixed $\pm 40\,\mu$A current perturbation on MRR 15.

for a second layer, this adds another 3 heater updates,

$$t_{\text{update}} = N_{\text{layers}} \left(N_{\text{set}}\, t_{\text{heat}} + N_{\text{fwd}} \frac{B}{B_{\text{det}}} + N_{\text{fwd}}\, t_{\text{fb}} + N_{\text{set}}\, t_{\text{dac}} \right) + t_{\text{fpga}} = 32.33\,\mu\text{s}. \tag{17}$$

Component	Events per update	Per event (μs)	Total (μs)
Heater thermal settling	6	5	30.00
Optical readout	2000	0.001	2.00
ADC and FPGA feedback	4	0.05	0.20
DAC electronic update	6	0.005	0.03
FPGA gradient and update logic	1	0.1	0.10
Total per update			32.33

Table 2 Estimated on-chip time per weight update. One update sets the heaters three times (W^+, W^-, updated W) and runs two forward passes over the 500 training samples at the 1 GHz detection bandwidth, and the 30×30 core is reused for both layers, so every event count is doubled.

The optimum weights for the photonic chip are achieved after 70,000 epochs, which ran for 2.3s. T he optimum validation accuracy was around 97%,and it yielded 95.8% on the test dataset. The photonic core also has the lowest test accuracy among the GPU and the CPU, mostly due to noisy readout.

Since the curve relating heater currents and realized weights is non-linear, as shown in Fig. 4 (c), a fixed current perturbation would produce different realized weight perturbations at different heater currents, as can be seen in Fig. 4 (d). Due to this nonlinear behavior, a fixed learning rate cannot suit

Platform	Time/update (μs)	Validation (%)	Updates ($\times 10^3$)	Time (s)	Test (%)
Photonic core	32.3	97	70	2.3	95.8
CPU, Intel Core Ultra 9 185H	72.7	97	77	5.6	97.2
GPU, NVIDIA A100	54.4	97	77	4.2	97.2

Table 3 The Yin-Yang problem solved on three different devices using MGD. Initalization and compilation times are excluded.

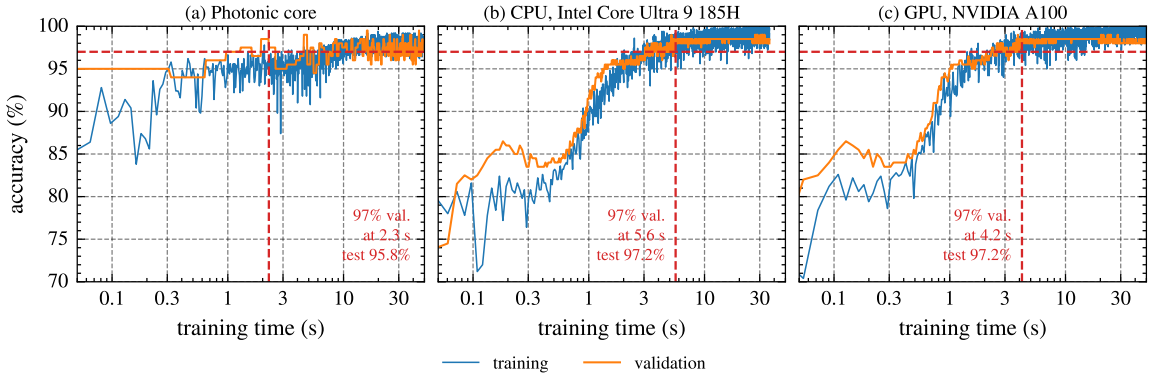

Fig. 5 The Yin-Yang problem solved on three different devices using MGD. The graph plots the training and validation accuracy vs wall time for the (a) photonic core, (b) CPU, and (c) GPU. The dashed line marks the point of the optimum validation accuracy.

all the MRRs simultaneously. Therefore, a fix is introduced by a full sweep before the training starts to map the currents to their corresponding weights for each MRR. The weight map is then used to vary the current perturbations such that the weight perturbation would be fixed for all the MRRs.

5 System latency

We also investigated the latency at different stages of our analog end-to-end adaptive training pipeline, as summarized in Table 1. These stages include the time to upload and download input/output data from the RF data converter, search for the output batch pattern from the received ADC signal, reconstruct symbols from the output pattern, set the weights, and measure the DC component of the output photocurrent.

Table 4 Processing latencies from the main stages in the training pipeline.

Pipeline stages	Average elapsed time[1] [ms]
RFDC[2] upload/download	1.06
Pattern search	1.72
Symbol reconstruction	12.9
Weight actuation[3]	3.61
DC current measurement[3]	2.72

[1]Average of 65,000 repeated measurements.

[2]RF data converter, includes both the DAC and the ADC.

[3]Includes 0.8 ms roundtrip LAN communication delay.

Here, we find that these stages make up a total execution time of roughly 22 milliseconds for every batch, which adds up to roughly 550 milliseconds per epoch, each with 25 batches. Considering additional overhead from code execution, these results line up well with our latency measurements for each epoch. This shows an execution time of approximately 1.5 seconds with a larger learning rate and perturbation amplitude and approximately 4 seconds when using smaller parameters. With our current implementation, we are able to showcase second-scale convergence time with fast adaptation and excellent stability.

6 Energy Budget

Here we break down the energy consumption of our proposed analog end-to-end adaptive training system and calculate the energy efficiency at both the experimentally demonstrated speed of 1 Gbaud and at the maximum achievable processing speed of 16 Gbaud. Our proposed system contains one laser driver per optical channel and one pair of balanced photodetectors. We also used one DAC per optical channel, but only one ADC and one transimpedance amplifier for the entire setup. We include the power budget for thermally tuning two microring resonators in our weight bank. The energy budget of all items are shown in Table 5, together with the calculated energy efficiency when operating at two different speeds. In addition, we also show an energy efficiency comparison in Table 6, between our proposed analog training system and other training platforms, including the edge TPUs [12], and hybrid systems that include photonics and GPUs [13], and photonics and digital electronics [14]. For the calculation, we used a theoretically possible large-scale hardware implementation that has a 108×108 MRR weight bank matrix.

Table 5 Energy of our proposed analog in-situ and on-the-fly training system which includes a microring weight bank with 2 rings and a pair of balanced photodetectors.

Components	Energy budget	Power consumption at 1 Gbaud	Power consumption at 16 Gbaud
Laser driver	15 mW	7.5 pJ/Ops	468 fJ/Ops
Detector	75 μW	37.5 fJ/Ops	2.3 fJ/Ops
DAC	6.1 mW	3.0 pJ/Ops	187 f J/Ops
ADC (1GHz)	2.5 mW	1.25 pJ/Ops	78 fJ/Ops
P_{weight}	1 mW	0.5 pJ/Ops	31 fJ/Ops
Transimpedance amplifier	2.8 mW	1.4 pJ/Ops	87 fJ/Ops
Total	27.4 mW	13.6 pJ/Ops	853 fJ/Ops
Energy efficiency		70 GOPS/W	1.2 TOPS/W

Table 6 Efficiency comparison of various training hardware platforms, including both digital electronics and analog photonics.

Platform	Energy efficiency
This work	1.2 TOPS/W 14 PetaOPS/W [1]
Edge TPUs [12]	2 TOPS/W
Photonics + GPU [13]	0.7 TOPS/W
Photonics + digital [14]	0.4 TOPS/W

[1] Projected performance with a 108×108 MRR weight bank matrix.

References

[1] Dembo, A., Kailath, T.: Model-free distributed learning. IEEE Transactions on Neural Networks **1**(1), 58–70 (1990) https://doi.org/10.1109/72.80205

[2] Spall, J.C.: Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Transactions on Automatic Control **37**(3), 332–341 (1992) https://doi.org/10.1109/9.119632

[3] McCaughan, A.N., Oripov, B.G., Ganesh, N., Nam, S.W., Dienstfrey, A., Buckley, S.M.: Multiplexed gradient descent: Fast online training of modern datasets on hardware neural networks without backpropagation. APL Machine Learning **1**(2), 026118 (2023) https://doi.org/10.1063/5.0157645 https://pubs.aip.org/aip/aml/article-pdf/doi/10.1063/5.0157645/18017061/026118_1_5.0157645.pdf

[4] Oripov, B.G., Dienstfrey, A., McCaughan, A.N., Buckley, S.M.: Scaling of hardware-compatible perturbative training algorithms. APL Machine Learning **3**(2), 026107

(2025) https://doi.org/10.1063/5.0258271 https://pubs.aip.org/aip/aml/article-pdf/doi/10.1063/5.0258271/20492805/026107_1_5.0258271.pdf

[5] Tait, A.N., Wu, A.X., De Lima, T.F., Zhou, E., Shastri, B.J., Nahmias, M.A., Prucnal, P.R.: Microring weight banks. IEEE Journal of Selected Topics in Quantum Electronics **22**(6), 312–325 (2016)

[6] Schwelb, O.: Transmission, group delay, and dispersion in single-ring optical resonators and add/drop filters-a tutorial overview. Journal of Lightwave Technology **22**(5), 1380 (2004)

[7] Schwelb, O.: Generalized analysis for a class of linear interferometric networks–part i: Analysis. IEEE Transactions on Microwave Theory and Techniques **46**(10), 1399–1408 (1998)

[8] Singh, J., Morison, H., Guo, Z., Marquez, B.A., Esmaeeli, O., Prucnal, P.R., Chrostowski, L., Shekhar, S., Shastri, B.J.: Neuromorphic photonic circuit modeling in verilog-a. APL Photonics **7**(4) (2022)

[9] Morison, H., Singh, J., Al Kayed, N., Aadhi, A., Moridsadat, M., Tamura, M., Tait, A.N., Shastri, B.J.: Nonlinear dynamics in neuromorphic photonic networks: Physical simulation in verilog-a. Physical Review Applied **21**(3), 034013 (2024)

[10] Jayatilleka, H., Murray, K., Guillén-Torres, M., Caverley, M., Hu, R., Jaeger, N.A.F., Chrostowski, L., Shekhar, S.: Wavelength tuning and stabilization of microring-based filters using silicon in-resonator photoconductive heaters. Opt. Express **23**(19), 25084–25097 (2015) https://doi.org/10.1364/OE.23.025084

[11] Jayatilleka, H., Shoman, H., Chrostowski, L., Shekhar, S.: Photoconductive heaters enable control of large-scale silicon photonic ring resonator circuits. Optica **6**(1), 84–91 (2019)

[12] Sun, Y., Kist, A.M.: Deep Learning on Edge TPUs (2022). https://arxiv.org/abs/2108.13732

[13] Feldmann, J., Youngblood, N., Karpov, M., Gehring, H., Li, X., Stappers, M., Le Gallo, M., Fu, X., Lukashchuk, A., Raja, A.S., Liu, J., Wright, C.D., Sebastian, A., Kippenberg, T.J., Pernice, W.H.P., Bhaskaran, H.: Parallel convolutional processing using an integrated photonic tensor core. Nature **589**(7840), 52–58 (2021) https://doi.org/10.1038/s41586-020-03070-1

[14] Zhou, T., Lin, X., Wu, J., Chen, Y., Xie, H., Li, Y., Fan, J., Wu, H., Fang, L., Dai, Q.: Large-scale neuromorphic optoelectronic computing with a reconfigurable diffractive processing unit. Nature Photonics **15**(5), 367–373 (2021) https://doi.org/10.1038/s41566-021-00796-w