Supplemental Document

OPTICA

Silicon photonic architecture for training deep neural networks with direct feedback alignment: supplement

Matthew J. Filipovich,[1,6,†] Zhimu Guo,[1,†] Mohammed Al-Qadasi,[2] Bicky A. Marquez,[1] Hugh D. Morison,[1] Volker J. Sorger,[3] Paul R. Prucnal,[4] Sudip Shekhar,[2] and Bhavin J. Shastri[1,5,7]

[1]*Department of Physics, Engineering Physics and Astronomy, Queen's University, Kingston, Ontario K7L 3N6, Canada*
[2]*Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, British Columbia V6T 1Z4, Canada*
[3]*Department of Electrical and Computer Engineering, George Washington University, Washington, DC 20052, USA*
[4]*Department of Electrical Engineering, Princeton University, Princeton, New Jersey 08542, USA*
[5]*Vector Institute, Toronto, Ontario M5G 1M1, Canada*
[6]*e-mail: matthew.filipovich@queensu.ca*
[7]*e-mail: shastri@ieee.org*
[†]*These authors contributed equally to this paper.*

Supplement DOI: https://doi.org/10.6084/m9.figshare.21518115

Parent Article DOI: https://doi.org/10.1364/OPTICA.475493

Silicon Photonic Architecture for Training Deep Neural Networks with Direct Feedback Alignment: Supplemental Document

In this supplemental document, we provide the experimental details, describe the direct feedback alignment training algorithm, and discuss the numerical simulation of our photonic neural network training architecture.

I. EXPERIMENTAL PHOTONIC HARDWARE OPERATION

To demonstrate matrix-vector multiplication operations, we performed two sets of measurements using two similar circuits which both contain identical 1×4 MRR arrays with coupled through and drop ports; the ports in the first circuit connect to an integrated balanced photodetector (BPD), and the ports in the other circuit connect to grating couplers for off-chip photodetection. The integrated BPD consists of two germanium doped PIN photodiodes (the configuration is detailed in Refs. [1, 2]), and off-chip photodetection was performed using a 5 GHz BPD from Thorlabs (part BDX1BA). The circuits were fabricated on an SOI wafer with a silicon thickness of 220 nm and a buried oxide thickness of 2 µm. The MRRs in both arrays have radii of 8.000 µm, 8.012 µm, 8.024 µm, and 8.036 µm, with a quality factor of ∼6000 and bandwidth of 200 pm each. The slight differences in ring radii within the MRR arrays were implemented to avoid resonance collision, and the spacing between the neighboring MRRs is 92 µm. The input optical channels were multiplexed into a single fiber using a 1×4 PLC fiber splitter in reverse and injected into the input waveguide bus of the MRR arrays using TE focusing grating couplers. The optical losses in the system include the round-trip loss that occurs when the light is coupled on and off the chip through a pair of grating couplers, which is measured to be 15 dB. The insertion loss of each MRR is estimated to be 3 dB, and the combiner introduces an additional 7.3 dB of insertion loss. To accommodate for this loss, we use an erbium-doped fiber amplifier (EDFA) with a gain of 20 dB to increase the amplitude of the input optical power. The transmission spectrum of the 1×4 MRR array is shown in Fig. 1. Approximately 0.8 mA of heating current is required to increase the optical power in the through port by 15 dB and decrease the optical power in the drop port by 10 dB, as shown in Fig. 2.

We initially calibrated the 1×4 MRR array to determine the mapping between the applied heating current and the effective weighting value w of each MRR, defined by $T_d - T_p$ as discussed in Sec. 2 in the manuscript. We injected light from four external cavity laser sources with different wavelengths (1546.558 nm, 1548.675 nm, 1549.595 nm,

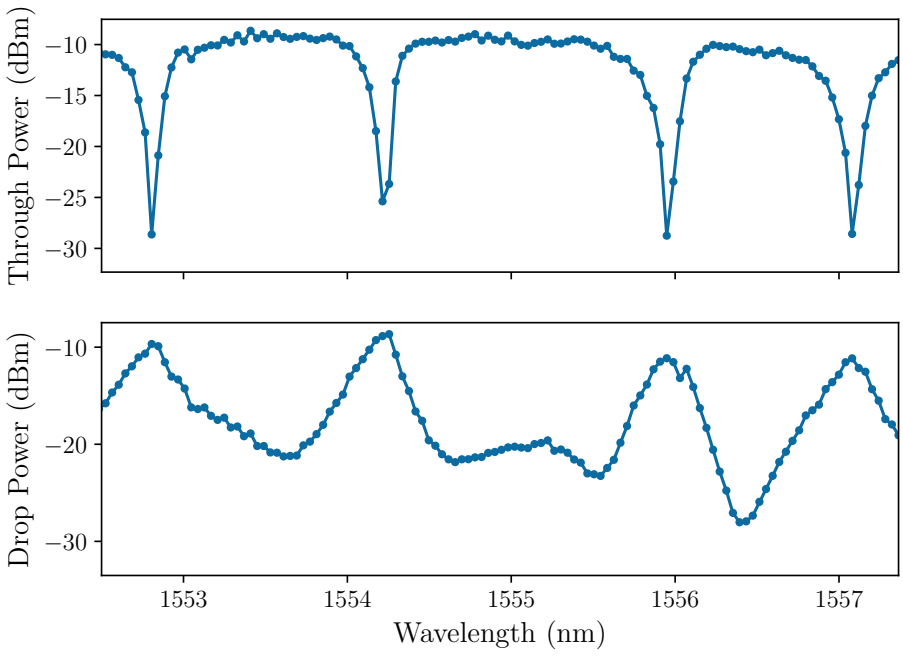

FIG. 1. Transmission in the through and drop ports of the 1×4 MRR array as a function of optical input frequency. An input optical power of 7 dBm was used.

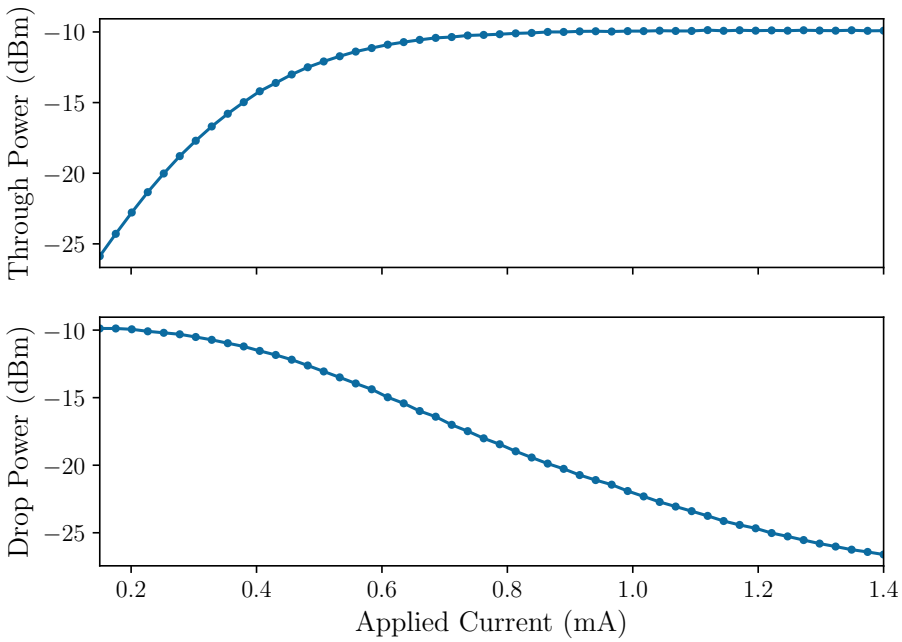

FIG. 2. Transmission in the through and drop ports of the 1×4 MRR array while sweeping the current applied to a single MRR. An input optical power of 7 dBm was used.

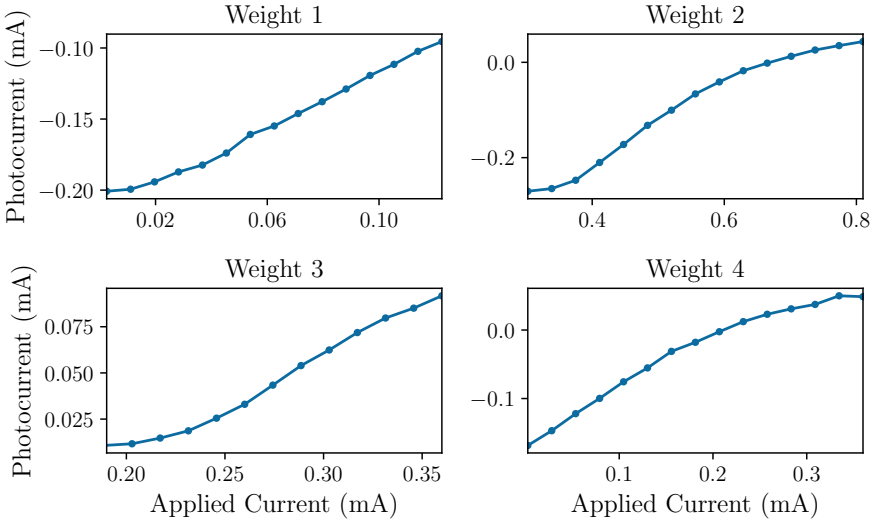

FIG. 3. Snapshots of the photocurrent output from the integrated BPD in the 1×4 MRR array while sweeping the current applied to all four MRRs simultaneously. All input optical powers were set to 7 dBm during the sweep.

and 1551.480 nm) into the weight bank and performed four individual sweeps of the applied heating current to each MRR. During the sweep of the applied heating current, we measured the output photocurrent from the BPD, which was proportional to $|E_0|^2(T_d - T_p)$ where E_0 is the amplitude of the input optical signal. The resulting modulation results are shown in Fig. 3. After collecting the mapping between the applied heating current and MRR weight value, we determined the reflection point of the MRR, which is the value of the applied heating current where equal optical power propagates into the through and drop ports. Finally, we defined the experimental range of applied heating current to be centered around the reflection point to allow the encoding of both positive and negative weight values.

Each MRR in the array required an applied voltage of 2 V and consumed 1.5 mW of power during steady-state operation. The additional average tuning power for each MRR was 0.2 mW. The power required by the photodetectors

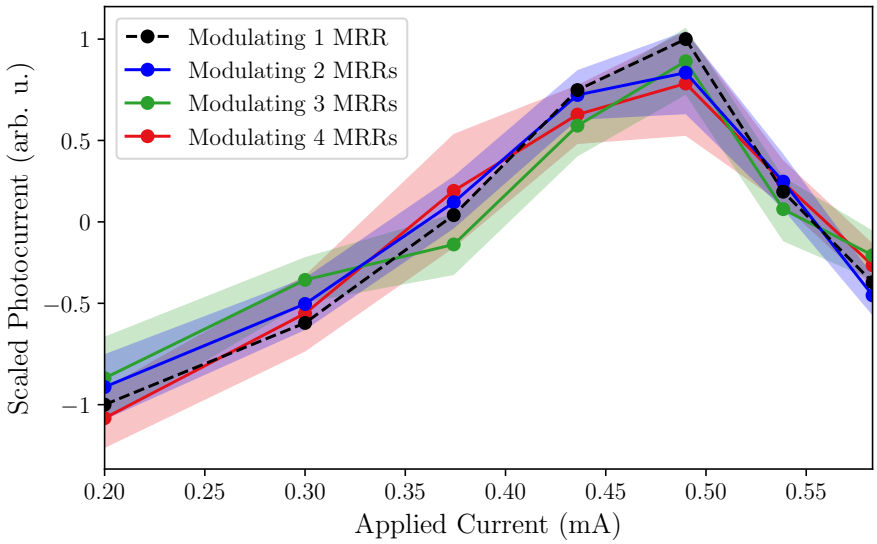

FIG. 4. Demonstration of simultaneous modulation of MRRs in the integrated BPD circuit. The black dashed line shows the measured photocurrent, scaled between −1 and 1, as a function of the applied current to the first MRR in the 1×4 array while keeping the other three MRR weights constant (i.e., only varying the applied current to the first MRR). The scaled photocurrent measurements are shown when sweeping across the applied current to the first MRR while simultaneously varying the applied current to the second MRR (blue), the second and third MRRs (green), and the second, third, and fourth MRRs (red). Each data point in the three lines is the mean of eight measurements which have been adjusted (by removing the expected difference in photocurrent when varying the other MRRs) to approximate the change in photocurrent generated solely by the first MRR (the standard deviation from the measurements is also shown). Therefore, the difference in the data points at each value of the applied current to the first MRR can be attributed to thermal crosstalk between adjacent MRRs, as well as optical and electronic noise in the system.

in the MRR array was negligible as they were unbiased during the experiments. Therefore, for a 1×4 MRR array, the total power consumption by the MRRs was ∼6.8 mW. The total power from all four laser sources was 40 mW, as each laser was modulated using an optical power centered around 10 dBm. Since weight updating of our device is currently implemented using thermal tuning and the MRRs can operate at a thermal tuning speed of 170 µs [3], the estimated energy consumption of our current device is ∼2.0 µJ per MAC operation.

The optical channels were modulated directly by Pure Photonic PPCL500 laser sources (using an embedded electronic modulation feature on the lasers). The optical power levels between 7 dBm and 11 dBm were mapped to the expected range between 0 and 1 for the input values. Using the known encoding mappings for the four laser sources and four MRRs, we then randomly varied both the input values of the lasers and the weight values of the MRRs over 5000 time steps. At each time step, we measured the photocurrent from the BPD, which was the output of the on-chip inner product operation. We determined the expected output of each operation using the known encoding mappings of the laser sources and MRRs. Thermal tuning of our MRRs can operate at kHz speeds [3]; however, the off-chip source meters and tunable lasers we used (four Keithley 2606B source meters, three custom source meters, and four Pure Photonic PPCL500 tunable laser) were not optimized for high frequency operation which bottlenecked the maximum operating speed. During our data collection using the on-chip BPD circuit, the process of updating all four weight and input values and then measuring the photocurrent took approximately one second. The adjacent MRRs are separated by 92 µm, and the experimental platform was thermally controlled to maintain a constant temperature of 23.4°C. Simultaneous modulation of up to four MRRs is demonstrated in Fig. 4.

II. DIRECT FEEDBACK ALIGNMENT ALGORITHM

In feedforward neural networks, the nodes are clustered in different layers where the signals can only pass from a preceding layer to a succeeding layer. The behaviour of all nodes in a feedforward neural network is identical: the input signals from the preceding layer are weighted and summed, and a non-linear activation function is applied to the sum, which is outputted to all the nodes in the succeeding layer. The forward propagation of the input signal using trained weights is known as inference. The inference of a neural network with l layers can be described for each

TABLE I. Machine learning notation used in this manuscript.

Symbol	Description
$\mathbf{W}^{(k)}$	Weight matrix between layers k and $k-1$
$\mathbf{b}^{(k)}$	Bias vector between layers k and $k-1$
$\mathbf{a}^{(k)}$	Sum of the weighted input signals in the layer k
$g(\cdot)$	Activation function of the nodes
$\mathbf{h}^{(k)}$	Activation of the nodes in the layer k
$\mathbf{x}$	Training input
$\mathbf{y}$	Target output
$\hat{\mathbf{y}}$	Output of neural network from training input $\mathbf{x}$
$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$	Loss function
$\boldsymbol{\theta}$	Network parameters including weights and biases
$\mathbf{e}$	Error from gradient of the loss function

layer k (defining $\mathbf{h}^{(0)} \equiv \mathbf{x}$) as

$$\mathbf{a}^{(k)} = \mathbf{W}^{(k)}\mathbf{h}^{(k-1)} + \mathbf{b}^{(k)}, \tag{1}$$

$$\mathbf{h}^{(k)} = g\left(\mathbf{a}^{(k)}\right), \tag{2}$$

where the variables used are defined in Table I.

Feedforward neural networks approximate a function f by defining a mapping $\hat{\mathbf{y}} = f(\mathbf{x}; \boldsymbol{\theta})$, and the goal of training is to find the values of the network parameters $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{b}\}$ that reduce the overall error between the actual and target outputs [4]. The error to be minimized can be expressed as a loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$, where $\mathbf{y}$ is the target output and the choice of loss function depends on the specific application of the neural network.

The DFA algorithm first determines the gradient of the loss function in the output layer l, which is the error propagated to the hidden layers:

$$\mathbf{e} = \nabla_{\mathbf{a}^{(l)}}\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \nabla_{\mathbf{h}^{(l)}}\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) \odot g'\left(\mathbf{a}^{(l)}\right), \tag{3}$$

where $\odot$ is the Hadamard product (element-wise multiplication operator) and $g'(\cdot)$ is the derivative of the activation function with respect to $\mathbf{a}^{(l)}$ [5]. Using the cross entropy loss function and the softmax activation at the output, the gradient of the loss function is $\hat{\mathbf{y}} - \mathbf{y}$. The DFA algorithm calculates the gradients for each hidden layer k as

$$\boldsymbol{\delta}^{(k)} = \mathbf{B}^{(k)}\mathbf{e} \odot g'\left(\mathbf{a}^{(k)}\right), \tag{4}$$

where $\mathbf{B}^{(k)}$ is a fixed random weight matrix with appropriate dimensions.

Using the gradients calculated in Eqs. (3) and (4) and defining $\boldsymbol{\delta}^{(l)} \equiv \mathbf{e}$, the update functions of the network parameters $\boldsymbol{\theta}$ for each layer k are

$$\nabla_{\mathbf{b}^{(k)}}\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \boldsymbol{\delta}^{(k)}, \tag{5}$$

$$\nabla_{\mathbf{W}^{(k)}}\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \boldsymbol{\delta}^{(k)}(\mathbf{h}^{(k-1)})^T. \tag{6}$$

The stochastic gradient descent optimization algorithm can be implemented to improve training performance. A minibatch of examples $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ of size m from the training set are used to compute the update functions found in Eqs. (5) and (6). The update value for each parameter is then averaged from the minibatch calculations and the network parameters are updated by following the estimated gradient downhill:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\alpha}{m}\sum_{i=1}^{m}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i), \tag{7}$$

where α is the learning rate.

III. PYTHON SIMULATION OF THE PHOTONIC DFA ARCHITECTURE

We simulated the training of neural networks with our 1×4 MRR arrays in Python using the PyTorch framework. The Python simulation adds correctly-scaled Gaussian noise $\mathcal{N}(\mu, \sigma^2)$, with mean μ and variance σ^2, to the matrix-

vector multiplication operations when calculating the gradient during the backward pass to emulate the experimental results:

$$\boldsymbol{\delta}^{(k)} = \left(\mathbf{B}^{(k)}\mathbf{e} + \mathcal{N}\left(\mu, \sigma^2\right)\right) \odot g'\left(\mathbf{a}^{(k)}\right). \tag{8}$$

Since our 1×4 MRR array performs 4 MAC operations at each time step, we can simulate the inner product operation between two vectors of size N by splitting the vectors into $N/4$ sub-vectors of size 4 (if N is not divisible by 4, then there will be an additional sub-vector with size N mod 4). We then perform the inner product between all sub-vectors, which takes $N/4$ time steps, and finally sum the $N/4$ outputs to produce the full inner product result.

To accurately inject Gaussian noise into the output of an inner product operation composed of vectors larger than 4, we scale the standard deviation and mean of the applied Gaussian noise according to the properties of normally distributed random variables: assuming each MAC operation has a normally distributed error with standard deviation σ and mean μ, the inner product operation between two vectors of length N will have a normally distributed error with standard deviation $\sqrt{N}\sigma$ and mean $N\mu$.

We trained a neural network of size $784 \times 800 \times 800 \times 10$ (two hidden layers), and thus the error vector $\mathbf{e}$ had a length of 10 (since the MNIST dataset has ten classes). The gradient $\boldsymbol{\delta}^{(k)}$ for each hidden layer k is calculated by performing matrix-vector multiplications between the $\mathbf{B}^{(k)}$ matrix (800×10) and error vector $\mathbf{e}$ (10×1). Therefore, the calculation of each gradient vector requires 800 inner product operations between the error vector $\mathbf{e}$ and each row in the $\mathbf{B}^{(k)}$ matrix. To avoid overfitting our model, the gradient vector $\boldsymbol{\delta}^{(k)}$ was regularized by $1/\sqrt{M}$ where M is the size of the hidden layer k.

To simulate inner product operations between vectors of size N=10, we injected Gaussian noise modeled using the standard deviation and mean from the experimental 1×4 MRR array measurements scaled between -4 and 4 (i.e., the expected output range of the multiplications). The error from the measurements using the integrated BPD circuit has a standard deviation of $\sigma_4 = 0.202 \times 4 = 0.808$ and mean of $\mu_4 = 0.003 \times 4 = 0.012$. Therefore, for $N = 10$, the Gaussian noise added to each element in the matrix-vector output of size 800×10 has a standard deviation $\sigma_{10} = \sqrt{10/4} \times 0.808 = 1.277$ and mean $\mu_{10} = 0.012 \times 10/4 = 0.030$. Similarly, the noise added in the simulation of the off-chip BPD circuit has a standard deviation $\sigma_{10} = 0.618$ and mean $\mu_{10} = 0.030$.

We trained the neural networks on the MNIST dataset, which is a collection of 70,000 grey-scale images of hand-written digits from zero to nine, each of size 28×28. During training, we split the MNIST dataset into a training set (60,000 examples), a validation set (5,000 examples), and a test set (5,000 examples). Each network was trained over 50 epochs, and the network performance was evaluated using the validation set after each epoch, as shown in Fig. 5(b) in the manuscript. The network parameters with the highest accuracy on the validation set were then evaluated on the test set to report the model's unbiased performance. The test accuracy as a function of the effective resolution in bits, defined by $\log_2(2/\sigma)$ where σ is the standard deviation of the error in each MAC operation, is shown in Fig. 5(c) in the manuscript.

[1] A. N. Tait, T. Ferreira de Lima, M. A. Nahmias, H. B. Miller, H.-T. Peng, B. J. Shastri, and P. R. Prucnal, Physical Review Applied **11**, 064043 (2019).

[2] M. S. Hai, M. N. Sakib, and O. Liboiron-Ladouceur, Optics Express **21**, 32680 (2013).

[3] W. Bogaerts, P. De Heyn, T. Van Vaerenbergh, K. De Vos, S. Kumar Selvaraja, T. Claes, P. Dumon, P. Bienstman, D. Van Thourhout, and R. Baets, Laser & Photonics Reviews **6**, 47–73 (2012).

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016).

[5] A. Nøkland, Advances in Neural Information Processing Systems **29**, 1037 (2016).