





Control-free and efficient integrated photonic neural networks via hardware-aware training and pruning: supplement

TENGJI XU,^{1,†}  WEIPENG ZHANG,^{2,†}  JIAWEI ZHANG,^{2,†} ZEYU LUO,¹ QIARONG XIAO,¹ BENSHAN WANG,¹ MINGCHENG LUO,¹ XINGYUAN XU,³ BHAVIN J. SHASTRI,⁴  PAUL R. PRUCNAL,² AND CHAORAN HUANG^{1,*} 

¹Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong SAR, China

²Department of Electrical and Computer Engineering, Princeton University, Princeton, New Jersey 08544, USA

³State Key Lab. of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, Beijing 100876, China

⁴Department of Physics, Engineering Physics and Astronomy, Queen's University, Kingston, Ontario K7L 3N6, Canada

[†]These authors contributed equally to this work.

*crhuang@ee.cuhk.edu.hk

This supplement published with Optica Publishing Group on 23 July 2024 by The Authors under the terms of the [Creative Commons Attribution 4.0 License](#) in the format provided by the authors and unedited. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

Supplement DOI: <https://doi.org/10.6084/m9.figshare.26206952>

Parent Article DOI: <https://doi.org/10.1364/OPTICA.523225>

Control-free and efficient integrated photonic neural networks via hardware-aware training and pruning

1. SINGLE MICRO-RING RESONATOR(MRR) SIGNAL-TO-NOISE RATIO(SNR) ANALYSIS

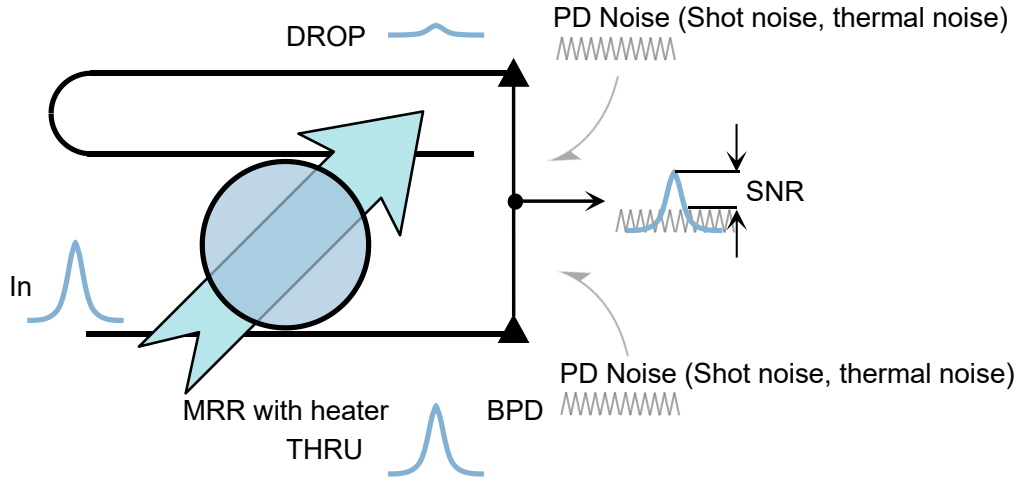


Fig. S1. Illustration of MRR weight bank operation with noise. MRR: Micro-ring resonator. THRU: Through. BPD: Balanced photodetector. SNR: Signal-to-noise ratio.

The signal-to-noise ratio (SNR) is defined as the ratio between signal power and noise power. Noise includes two kinds, shot noise and thermal noise. Shot noise originates from electrons being generated at random times. Thermal noise originates from electrons' random movement in a resistor in the absence of an applied voltage at a finite temperature. The signal, which is the photocurrent generated in PD can be calculated through:

$$I_p = RP \quad (S1)$$

where I_p is the photocurrent, R is the responsibility of PD, P is the optical power received by PD. The noise power can be calculated through:

$$\sigma_s^2 = 2q(I_p + I_d)\Delta f \quad (S2)$$

$$\sigma_T^2 = \frac{4kT}{R_L} F_n \Delta f \quad (S3)$$

where σ_s^2 is the shot noise power, σ_T^2 is the thermal noise power, q represents the elementary charge, I_d corresponds to the dark current, k stands for the Boltzmann constant, T is the temperature, R_L is the load resistance, F_n is the noise factor where thermal noise is enhanced by various resistors used in pre and main amplifier, Δf is the effective noise bandwidth.

Now, for the MRR weight bank, we define the power at port "in" as P_{in} , the power at port "DROP" as P_{drop} , the power at port "THRU" as P_{thru} , they can be tuned by actuating current on the heater over MRR.

$$P_{thru} = TP_{in} \quad (S4)$$

$$P_{drop} = (1 - T)P_{in} \quad (S5)$$

$$w = 2T - 1 \quad (S6)$$

where T is the transmission of MRR Through port, w is the weight of MRR.

Above all, the SNR of the system shown in Fig.S1 can be calculated through:

$$SNR = \frac{|(RP_{thru})^2 - (RP_{drop})^2|}{\sigma_{s,thru}^2 + \sigma_{T,thru}^2 + \sigma_{s,drop}^2 + \sigma_{T,drop}^2} = \frac{|w|P_{in}^2}{2qRP_{in}\Delta f + 4qI_d\Delta f + \frac{8kT}{R_L}F_n\Delta f} \quad (S7)$$

From Eq.S7, SNR is linearly proportional to the weight absolute value. Thus SNR attains the peak when w=-1 or w=1, and reaches the bottom when w=0.

2. PRUNING PROCESS

The optimization process consists of three steps (Shown in Fig. S2). Firstly, the neural network (NN) is trained without the regularization term. The resulting weights follow a Gaussian distribution centered around 0. Secondly, tuning power estimation is performed on each layer to identify the layer that requires the largest tuning power during inference. The value of this layer is selected as W in the regularization term. The NN is then trained again. During this optimization step, the inference accuracy will decrease due to regularization effect [1]. However, if the decrease is acceptable, the ‘MRR pruning optimization’ can be performed until the inference accuracy is close to the lowest allowable level. Finally, weights that are close to 1 but not exactly 1 are assigned a value of 1. During training, a clamping function is applied to confine the NN weights’ values. Due to the variance in extinction ratio for each MRR within an MRR weight bank, the minimum attainable weight value is found to be approximately -0.9 empirically.

The loss continuously decreases to 0 during pruning because weights are gradually relocated to 1, as shown in Fig. S4. The corresponding training parameters are shown in Table. S2.

We also do a simulation to validate whether the first step training NN without the regularization term will affect the final training results. The simulation results are shown in Fig. S3, the accuracy increases to 98% in the first training epoch and remains the same in the remaining training epochs. In the meantime, loss gradually, meaning weights are gradually pruned to 1. This indicates the first step training NN without the regularization term can be removed without affecting the final result. The corresponding training parameters are shown in Table. S1.

Table S1. Detailed training parameters for the test without training NN without the regularization term.

Neural network	Selected layer	Regularization coefficient	Learning rate	Optimizer	Environment
LeNet-5	Linear, channel 400	0.1	0.02	SGD	Pytorch 2.1.2

3. NN STRUCTURE

Specific NN structures (Two-layer convolutional NN (CNN), LeNet-5 [2], ResNet-18 [3]) are shown in Fig. S5.

4. TASK AND DATASET

The datasets we applied are the Modified National Institute of Standards and Technology (MNIST) dataset and the Canadian Institute for Advanced Research, 10 classes (CIFAR-10) dataset [4, 5]. The MNIST dataset is a collection of 60,000 training images and 10,000 test images, all grayscale and normalized to fit within a 28×28 pixel bounding box. These images contain handwritten digits from 0 to 9, each accompanied by corresponding labels. CIFAR-10 consists of 10 different categories, with each category containing 6,000 color images of size 32×32 pixels. The dataset is designed for benchmarking and research in the fields of computer vision and machine learning.

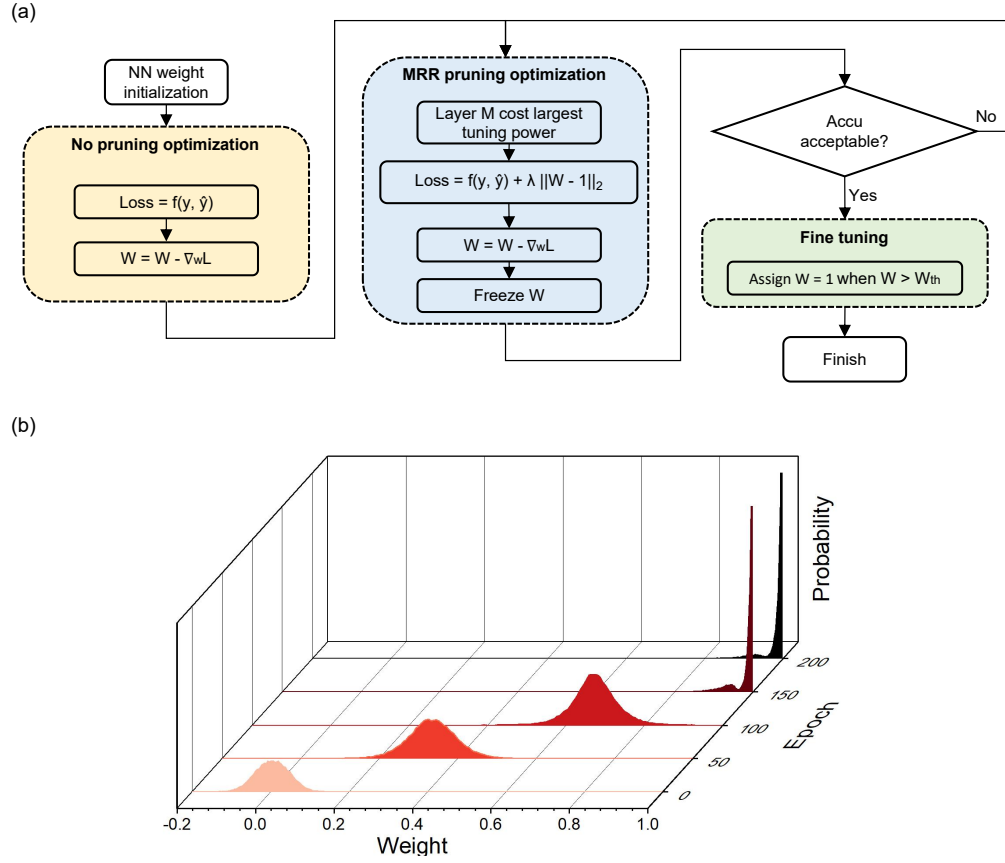


Fig. S2. (a) Optimization block diagram. (b) Weight distribution changes during optimization.

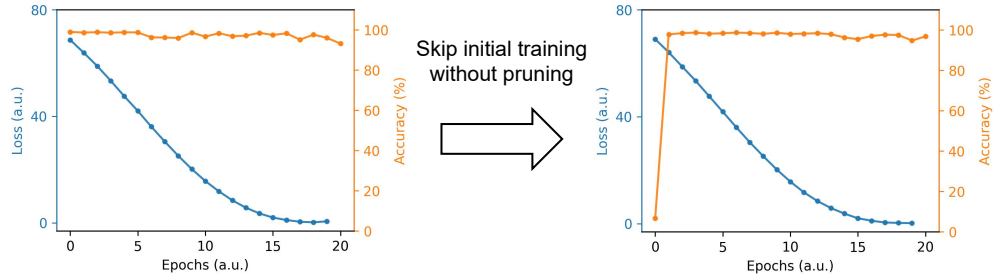


Fig. S3. Comparison between with and without initial training without pruning.

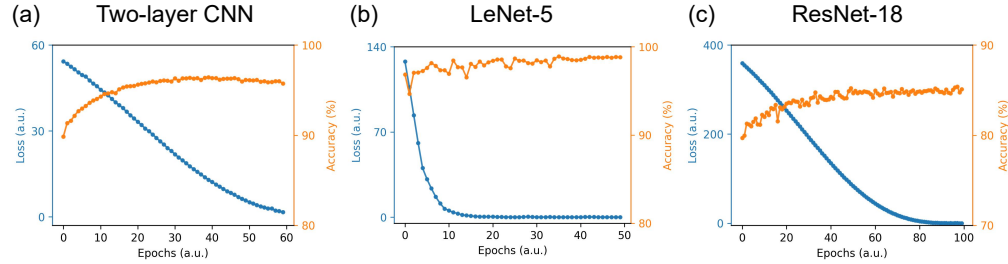


Fig. S4. Convergence test for three different size neural networks.

Table S2. Detailed training parameters for convergence test.

Neural network	Selected layer	Regularization coefficient	Learning rate	Optimizer	Environment
Two-layer CNN	Linear, channel 676	0.2	0.001	SGD	Pytorch 2.1.2
LeNet-5	Linear, channel 400	0.2	0.02	SGD	Pytorch 2.1.2
ResNet-18	BasicBlock2, channel 512	0.1	0.01	SGD	Pytorch 2.1.2

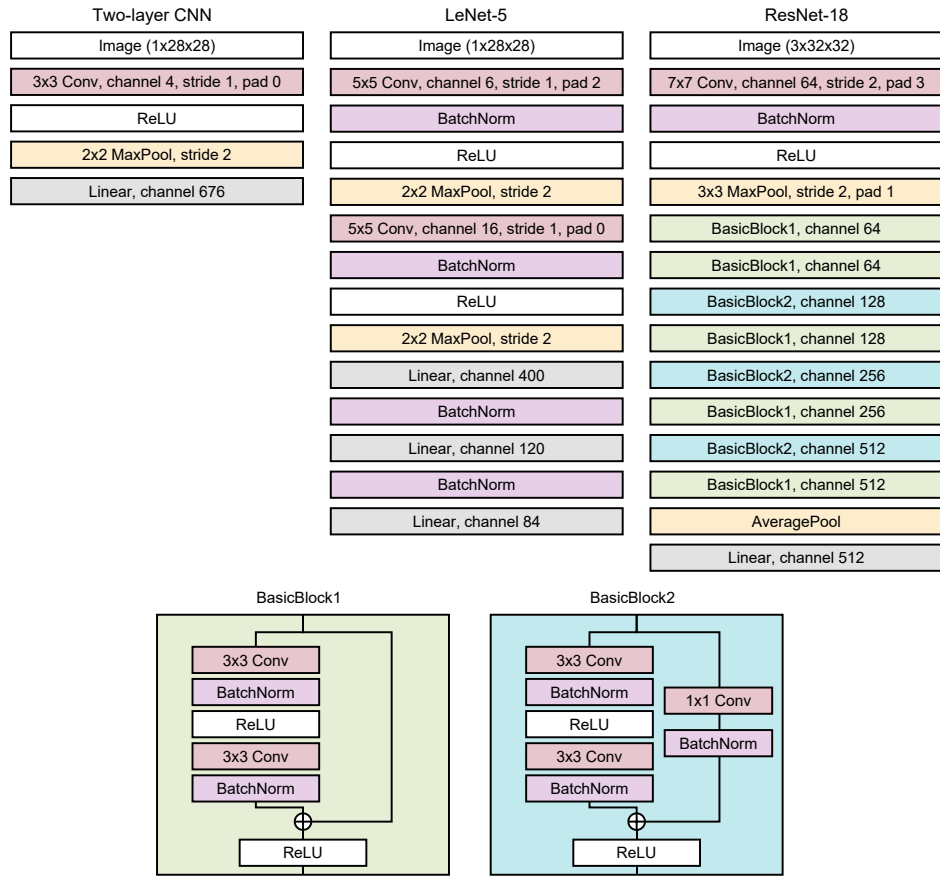


Fig. S5. Specific applied NN structures.

5. EXPERIMENT DETAILS

The weight bank is fabricated on a silicon-on-insulator (SOI) wafer with a silicon thickness of 220nm and a buried oxide thickness of $2\mu\text{m}$. It consists of four MRRs (radii around $r = 8\mu\text{m}$) coupled with two bus waveguides in an add/drop configuration. A slight difference ($\Delta r = 12\text{nm}$) is introduced in the ring radii to avoid resonance collision. This results in resonance wavelengths roughly being spaced by 1.6nm. The gap between the ring and bus waveguide is 200 nm. MRR Q factor is about 6000. Circular metal heaters are built on top of each MRR for thermal weight tuning. Metal vias and traces are deposited to connect heater contacts of the MRR weight bank to electrical metal pads. The heater resistance is around 2000 Ohms. These MRRs' waveguides are N-doped for efficient thermal tuning. The tuning efficiency is around 6.3 mW/nm. On-chip BPD is used for electrical-to-optical (E/O) conversion.

The experimental setup in this work has a lot in common with the setup in [6]. The coupling loss is tested to about 7 dB per single chip between the fiber array and the chip. Connected via the ribbon cable, a customized PCB (bottom) offers tuning currents for MRR weight banks and biasing voltages for BPDs. This bottom PCB is connected to a FPGA board and can be programmed via a high-speed SPI protocol. The signal path starts from the Mach-Zehnder Modulator (MZM) and ends at the scope. As shown in Fig. S6, the resonant wavelengths of four MRRs are situated at 1551.7, 1553.0, 1554.7, and 1555.8nm. During the experiment, laser wavelengths are set to off-resonance region, 1552.0, 1553.3, 1555.0, and 1556.1 nm. The corresponding tuning curves are shown in Fig. S7. Fig. S7(a) shows weight change with actuate current and weight change range doesn't cover whole -1 and 1 due to extinction ratio variance. The tuning efficiency is 6.3 mW/nm. It is experimentally measured through 3 steps. (1) Sequentially modulate sinusoidal signal (or other amplitude-modulated signals) on each laser through the corresponding MZM (Reference vector $A(t)$). (2) Recording the output electrical signal as record vector $S(t)$. (3) The weight w is then obtained by mathematically calculating the product of the pseudo-inverse matrix of $A(t)$ and $S(t)$ [7]. Fig. S7(b) is calculated based on the assumption that the resistance of heater is 2000 ohms. Optical coupling is done through a glued fiber array, which has a polished angle of 41 degrees to allow working with on-chip grating couplers of an 8-degree incident angle [6].

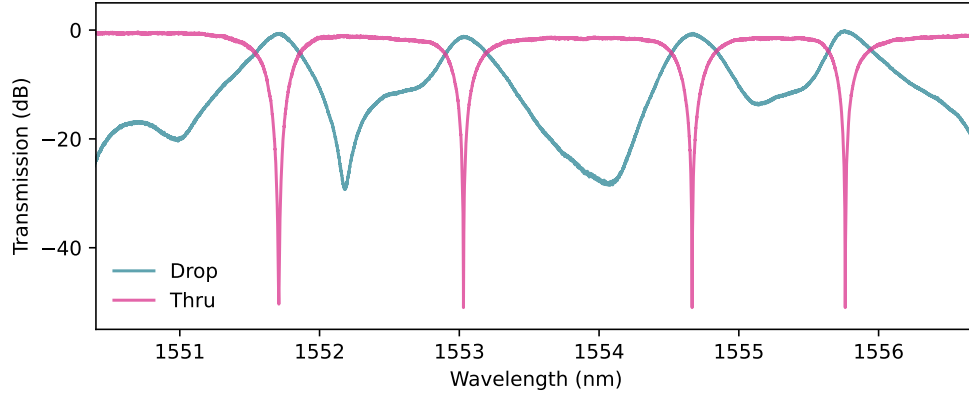


Fig. S6. MRR weight bank spectrum.

A. Implementing weights on 4×1 MRR weight bank

We first validate the effectiveness of our approach in reducing weight error on a two-layer CNN for MNIST digits classification during experiments. We select the second fully connected layer weights to implement on 4×1 MRR weight bank. The size of the fully connected layer is 676×10 . The first step is flattening the weight matrix into 6760×1 vector. Then sequentially implement every 4 weights on 4×1 MRR weight bank using the tuning curve as the look-up table, shown in Fig. S7(a). Without external thermoelectric controller (TEC), we go through all the weights, both with and without pruning. Each rotation spans approximately 12 seconds, and it requires a total of 1690 rotations to implement 6760 weights. After actuating every 4 weights on MRR weight bank, we apply pseudo-inverse method to acquire true weights and calculate weight error. After getting the weight error for all 6760 weights both without and with pruning, we can further

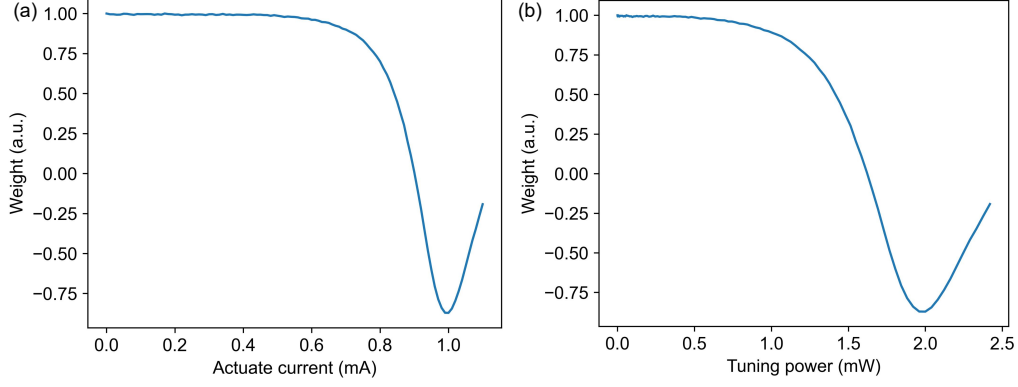


Fig. S7. MRR tuning curve. (a) Weight changes with actuating current. (b) Weight changes with tuning power.

analyze the system weight control precision and stability.

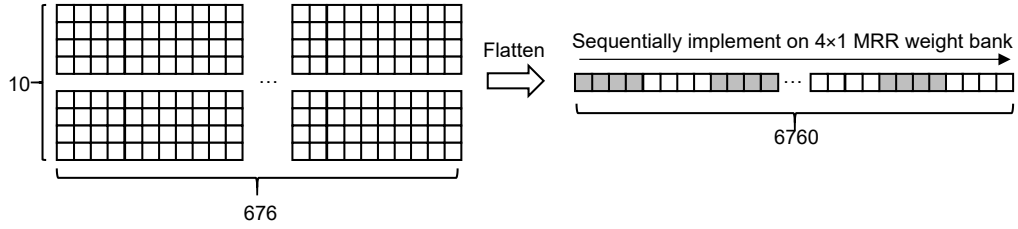


Fig. S8. Details about weights implementation.

B. Analyze on MRR drift

MRR drift value $\Delta\lambda$ can be calculated by the mean value of the weight error and the slope of the tuning curve.

$$\Delta\lambda = \frac{1}{K} |G'(w)| |\overline{\Delta w}| \quad (S8)$$

where K is the tuning efficiency of MRR, roughly 6.3mW/nm , $G(w)$ is the mathematical function of the tuning curve, shown in Fig.S7(b), $\overline{\Delta w}$ denotes the mean value of weight error.

Thus we can estimate the rough thermal drift is around 12pm through weight error of weights without pruning. Since the weights without pruning cluster around 0, we use $|G'(w=0)|$ as the constant value during calculation.

C. Analyze on system precision

Weight precision is defined as:

$$\text{Precision}(\text{bit}) = \log_2 \left(\frac{\text{maximal weight}(1) - \text{minimal weight}(-1)}{\text{std}(\Delta w)} \right) \quad (S9)$$

where $\text{std}(\cdot)$ is the standard deviation, Δw is the weight error.

$$\text{std}(\Delta w) = \sqrt{\frac{1}{6760} \sum_i^{6760} (\Delta w_i - \overline{\Delta w})^2} \quad (S10)$$

We don't directly use the definition of precision in [7], this definition assumes that the mean weight error is 0 when the control precision of the system is high, which is not the case in our experiments. As we don't use TEC in our experiments. Thus we modify Equ.S9 and Equ.S10 as:

$$\text{Precision}(\text{bit}) = \log_2 \left(\frac{\text{maximal weight}(1) - \text{minimal weight}(-1)}{f(\Delta w)} \right) \quad (S11)$$

$$f(\Delta w) = \frac{1}{6760} \sum_i^{6760} |\Delta w| \quad (\text{S12})$$

Thus we can estimate that the system precision is 3.1 bits when implementing weights without pruning. While it increases to 6.3 bits when implementing weights with pruning. Our approach attains 3-bit control precision improvement. This conclusion can be verified in main text Fig.2(c). The maximum precision improvement can reach 4 bits.

6. MATRIX FACTORING METHODS

The implementation of whole NN involves two steps, the first step is transforming convolution operation into matrix multiplication [8], the second step is factoring matrix multiplication into small sub-block matrix multiplication. For a large matrix multiplication $AB = C$, matrix A size is $T \times (N \times L)$, matrix B size is $(N \times L) \times K$. Here N is the number of MRR within a MRR weight bank. The mathematical transformation is shown in Fig. S9. Matrix A can be splitted into L small matrix $A_i (i = 1, 2, 3, \dots, L)$ (The size is $T \times N$), Matrix B can be splitted into $L \times K$ small matrix $B_{ij} (i = 1, 2, 3, \dots, L, j = 1, 2, 3, \dots, K)$ (The size is $N \times 1$). In this circumstance, the multiplication result matrix C can be splitted into K small matrix $C_j (j = 1, 2, 3, \dots, K)$ (The size is $T \times 1$). For each $C_j (j = 1, 2, 3, \dots, K)$, its mathematical calculation can be conducted through Eq. (S13). And the product of $A_i B_{ij}$ can be conducted by modulating A_i into N wavelengths wavelength division multiplexing (WDM) light and actuating B_{ij} values on MRR weight bank.

$$C_j = \sum_{i=1}^L A_i B_{ij} \quad (\text{S13})$$

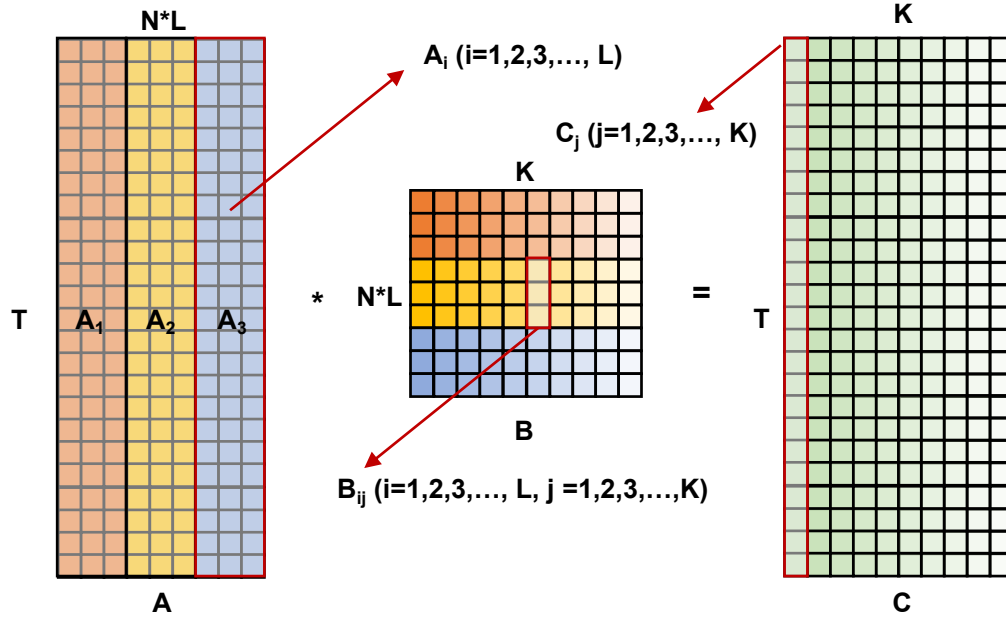


Fig. S9. Matrix factoring.

7. METHOD FOR ERROR SAMPLING

To evaluate the NN’s robustness under experiment weight error, the detailed implementation steps are

1. Iterate through each weight w in the neural network.
2. Randomly sample a value Δw from the statistical distribution of weight errors obtained from experiments.
3. Add the error to the weight w and reassign it as $w \leftarrow \Delta w$.
4. Conduct inference on the computer and observe the inference accuracy.

We compare the inference accuracy obtained from experiments on the actual device and from error sampling simulations, as shown in the Table.S3. It can be observed that the accuracies of the two are very close, indicating that randomly introducing weight error has been experimentally confirmed to validate the robustness of the neural network.

Table S3. Inference accuracy contrast between experiment inference results and random introduce noise inference.

	w/o Pruning	w/ Pruning
Experiment inference	67.0%	95.0%
Random introduce noise inference	66.7%	94.1%

8. ACCURACY AND POWER CONSUMPTION CHANGE WITH EXPERIMENT ERROR

During the optimization, we define the sparsity to reflect how much weight are pushed to 1 Eq. (S14). We store the NN parameters during training. Then we perform the inference on MRR weight bank (Two-layer CNN, experiment) or add weight errors based on experiment results (LeNet-5 and ResNet-18, simulation). The results are shown in Fig. S10, with the increase of sparsity, the inference accuracy increases and the average MRR tuning power decreases. The shaded areas in Fig. S10(b) and (c) represent simulation 95% confidence intervals.

$$Sparsity = \frac{N(w = 1)}{Total} \quad (S14)$$

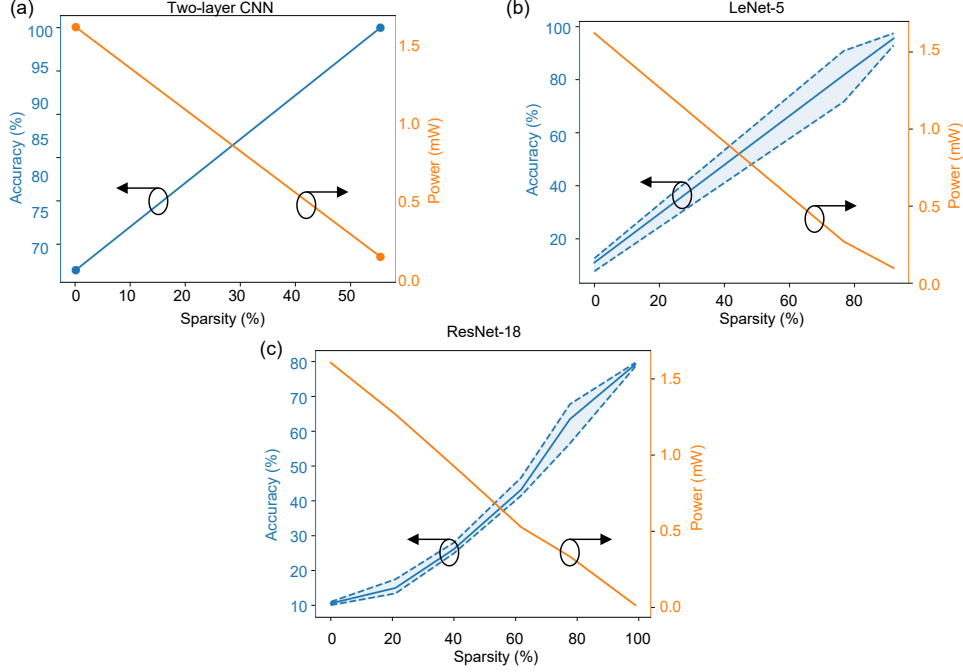


Fig. S10. Two-layer CNN, LeNet-5 and ResNet-18 inference accuracy and power consumption change. Left axis shows the inference accuracy after adding experiment weight error. Right axis shows the estimated average MRR tuning power. The color dots indicate the experiment results.

9. MATRIX MULTIPLICATION RESULT RELATIVE ERROR

The matrix multiplication relative error d_r is defined as:

$$d_r = \frac{|z_{\text{experiment}} - z_{\text{theory}}|}{|z_{\text{experiment}} + z_{\text{theory}}|/2} \quad (\text{S15})$$

Where $z_{\text{experiment}}$ is the experiment mathematical multiplication value, z_{theory} is the initial training result mathematical multiplication value. This parameter represents the relative percentage of the error caused by the experiment to the matrix multiplication result. The result is shown in Fig. S11. By setting most weights to 1, the signal intensity is preserved instead of being subtracted by the balanced photodetector. This results in remarkable improvement on signal to noise ratio so as decrease on relative error. The average relative error without pruning is 4.933 while it decreases to 0.099 after applying MRR pruning method. This value indirectly indicates the noise power is much higher than signal power (Roughly five fold), but it is much smaller after MRR pruning (Roughly one-tenth).

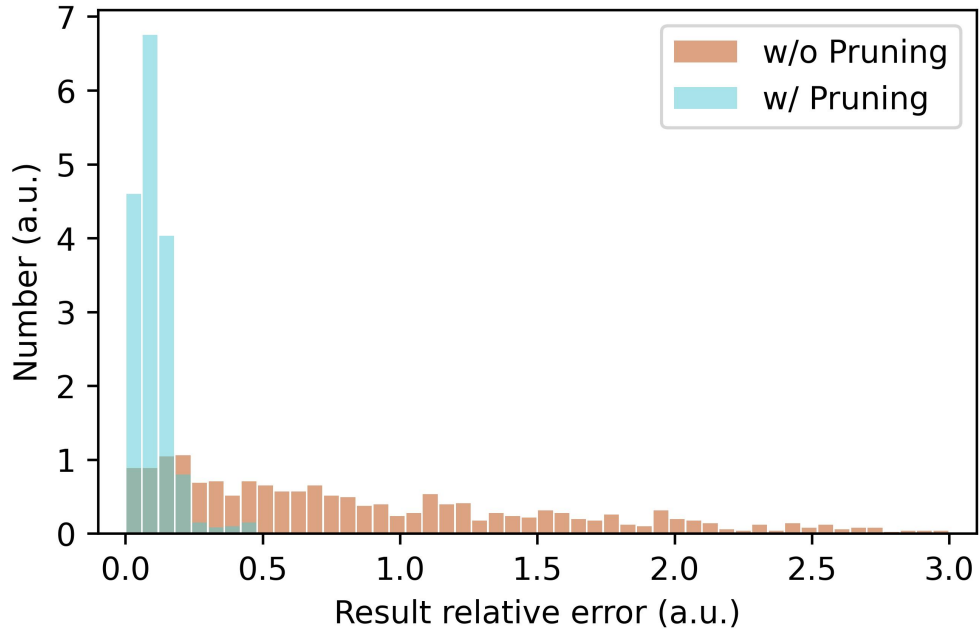


Fig. S11. Experiment result relative error contrast.

10. MRR DRIFT WEIGHT ERROR ESTIMATION METHOD

We estimate the weight error by taking a standard Lorentz function:

$$w = \frac{-A}{1 + \left(\frac{\lambda - \lambda_0}{\gamma}\right)^2} + A \quad (\text{S16})$$

$$w = \frac{A}{1 + \left(\frac{\lambda - \lambda_0}{\gamma}\right)^2} \quad (\text{S17})$$

$$w = \frac{-2A}{1 + \left(\frac{\lambda - \lambda_0}{\gamma}\right)^2} + A \quad (\text{S18})$$

Where w is the weight value, λ is the specific wavelength, λ_0 is the center resonant wavelength, A and γ are the coefficients depicting a Lorentz curve. Equ.S16, Equ.S17 and Equ.S18 correspond to single-end detection, crossbar and balanced MRR weight bank. We fit the equation with our experiment measured spectrum Fig.S6. The fitting coefficients A is 1, γ is 0.15.

The weight error Δw is linearly proportional to the derivative $\frac{dw}{d\lambda}$ and resonance drift value $\Delta\lambda$.

$$\Delta w = \frac{dw}{d\lambda} * \Delta\lambda \quad (\text{S19})$$

Therefore, we add the Gaussian noise Δw_{noise} as weight error to the NN to evaluate the robustness of the NN. Here $\Delta w_{noise} \sim N(0, |\frac{dw}{d\lambda} * \Delta\lambda|^2)$.

During the robustness simulation, we gradually increase the resonance drift value $\Delta\lambda$, the simulation results show that all different MRR weight robustnesses are improved after pruning.

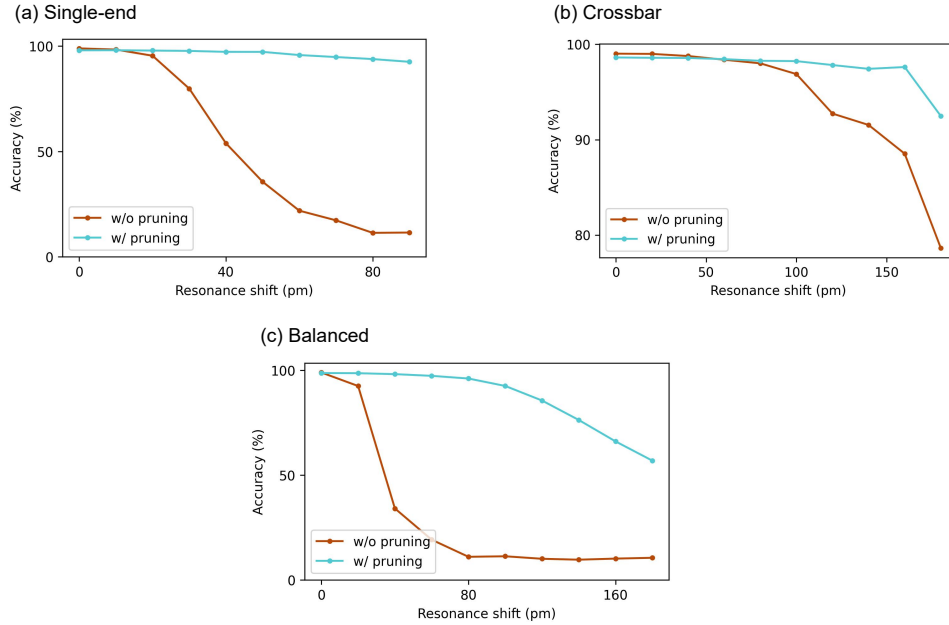


Fig. S12. Neural network robustness analysis simulation.

11. PHASE CHANGE MATERIAL (PCM) TEMPERATURE FLUCTUATION WEIGHT ERROR ESTIMATION METHOD

We estimate the weight error by taking a standard Sigmoid function:

$$w = \frac{\exp(\frac{-T}{\alpha})}{1 + \exp(\frac{-T}{\alpha})} \quad (\text{S20})$$

Where w is the weight value, T is the specific temperature, α is the coefficient depicting a Sigmoid curve. The fitting coefficient α is 4.

We replace the variable λ of Equ.S19 into temperature T , and perform a similar robustness test, the simulation result shows that PCM weight robustness is improved after pruning.

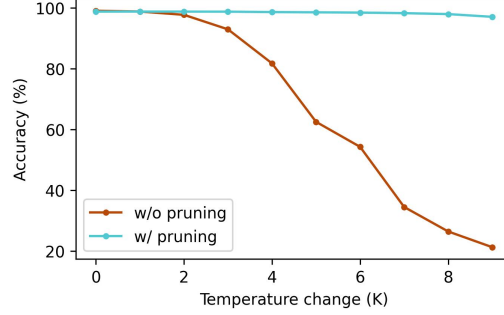


Fig. S13. Neural network robustness analysis simulation.

12. PRUNING EFFICIENCY IN DIFFERENT TASKS

Our method is applicable to various AI tasks. For more challenging tasks, fewer weights can be pruned to "1". As shown in Fig. S14, in the MNIST dataset, 92% of weights can be pruned. In contrast, for the more challenging Fashion-MNIST dataset, this percentage is 80%. The results are obtained to ensure the accuracy after pruning is close to that before pruning. This reduction in pruned weights can be compensated by using a larger neural network.

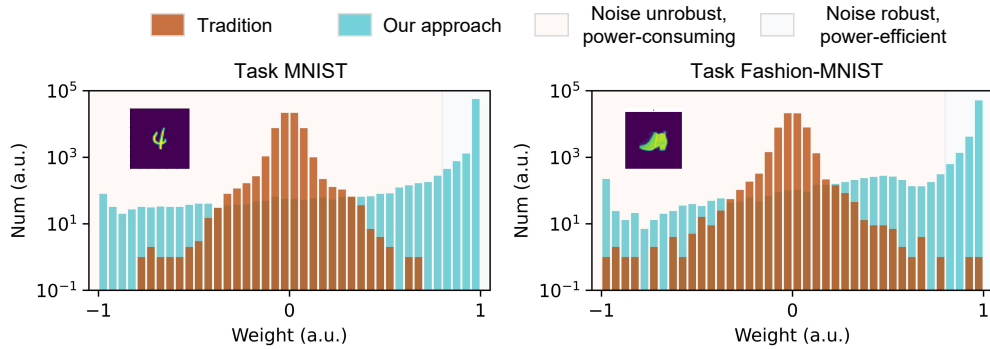


Fig. S14. Contrast test on different classification tasks.

REFERENCES

1. J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," arXiv:1803.03635 [cs.LG] (2018).
2. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.* **1**, 541–551 (1989).
3. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, (2016), pp. 770–778.
4. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE* **86**, 2278–2324 (1998).
5. A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," (2009). <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>.
6. W. Zhang, J. C. Lederman, T. Ferreira de Lima, J. Zhang, S. Bilodeau, L. Hudson, A. Tait, B. J. Shastri, and P. R. Prucnal, "A system-on-chip microwave photonic processor solves dynamic rf interference in real time with picosecond latency," arXiv:2306.14727 [physics.optics] (2023).
7. W. Zhang, C. Huang, H.-T. Peng, S. Bilodeau, A. Jha, E. Blow, T. Ferreira de Lima, B. J. Shastri, and P. Prucnal, "Silicon microring synapses enable photonic deep learning beyond 9-bit precision," *Optica* **9**, 579–584 (2022).
8. S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," arXiv:1410.0759 [cs.NE] (2014).