The Dissertation Committee for Jiaqi Gu
certifies that this is the approved version of the following dissertation:

Light-AI Interaction: Bridging Photonics and Artificial Intelligence via Cross-Layer Hardware/Software Co-Design

Committee:

David Z. Pan, Supervisor

Ray T. Chen, Co-Supervisor

Diana Marculescu

Atlas Wang

Song Han

Light-AI Interaction: Bridging Photonics and Artificial Intelligence via Cross-Layer Hardware/Software Co-Design

by

Jiaqi Gu

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2023

Acknowledgments

I would like to express my heartfelt gratitude to my Ph.D. advisor, Professor David Z. Pan, for his great support and guidance throughout my academic journey. As a mentor, he has not only been instrumental in shaping my research but has also honed my full-stack skills, ensuring that I am well-equipped for a successful career in academia. His invaluable insights and encouragement have been instrumental in every stage of my Ph.D. studies, and I am truly grateful for the opportunities he has given me to attend conferences, give talks, build connections, co-organize events, and be involved in proposal writing. Professor Pan's dedication to research, his warm advice, and his kind-heartedness will have positive impacts throughout my life in the future.

I am also deeply grateful to my co-advisor, Professor Ray T. Chen, for cultivating my passion for photonic computing and for providing great support throughout my graduate studies. The weekly meetings and interactions with him have been immensely beneficial, and my academic achievements would not be possible without the strong support of his research group. I would also like to thank Dr. Diana Marculescu, Dr. Song Han, and Dr. Atlas Wang, who played pivotal roles in my Ph.D. dissertation, offering valuable suggestions and feedback.

I would also like to extend my gratitude to my colleagues who directly contributed to the research projects that form part of this dissertation. Dr. Zheng Zhao has been an exceptional guide, offering valuable insights into hardware-software co-design of optical neural networks and collaborating with me on multiple projects. Dr. Zhoufeng Ying has provided critical suggestions with his expertise in optics and hardware design. I am also grateful to Cheng-hao Feng, my close collaborator, whose expertise in photonics and experiments has been invaluable to this dissertation. His generosity in discussions and help has been crucial.

I would like to thank all the UTDA alumni and members: Dr. Yibo Lin, Dr. Wuxi Li, Dr. Biying Xu, Dr. Shounak Dhar, Dr. Wei Ye, Dr. Zheng Zhao, Dr. Mohamed Baker Alawieh, Dr. Wei Shi, Dr. Arman Roohi, Dr. Keren Zhu, Dr. Mingjie Liu, Dr. Shuhan Zhang, Zixuan Jiang, Hanqing Zhu, Hao Chen, Rachel S. Rajarathnam, Ahmet F. Budak, Zhili Xiong, Xuyang Jin, Hyunsu Chae, Souradip Poddar, and Chen-Hao Hsu, who have provided me with invaluable help and suggestions throughout my academic journey. I am also deeply appreciative of the MURI research funding and UT graduate fellowships that funded my Ph.D. studies.

Finally, I would like to acknowledge the great love and support of my parents and other family members, who have been my pillars of strength. I am also deeply grateful to my friends, who have encouraged and supported me throughout my Ph.D. studies, making my life brighter.

Light-AI Interaction: Bridging Photonics and Artificial Intelligence via Cross-Layer Hardware/Software Co-Design

Publication No. _____

Jiaqi Gu, Ph.D.
The University of Texas at Austin, 2023

Supervisor: David Z. Pan
Co-Supervisor: Ray T. Chen

In the post-Moore era, conventional electronic digital computers have become a limiting factor in certain domains, most notably intelligent information processing. The proliferation of big data and artificial intelligence (AI) has motivated the investigation of next-generation AI computing hardware to support massively parallel and energy-hungry machine learning (ML) workloads. Photonic computing is a disruptive technology that can bring orders-of-magnitude performance and efficiency improvement to AI/ML with its ultra-fast speed, high parallelism, and low energy consumption. There has been growing interest in using nanophotonic processors for performing optical neural network (ONN) inference operations, which can make transformative impacts in future datacenters, automotive, smart sensing, and intelligent edge. However, the substantial potential in photonic computing also brings significant design challenges: (1) **area scalability issue** (i.e., large spatial sizes of

photonic devices); (2) **computing reliability concern** (i.e., sensitive to process variation and chip noises); (3) **adaptability difficulty** (i.e., hard to train the photonic chip for online adaptation); and (4) **high design complexity** (e.g., customized mixed-signal hardware). To build next-generation photonic AI computing platforms and bridge the gap between integrated photonics and AI, it is required to have a cross-layer co-design and design automation stack where the device, circuit, architecture, and algorithm are designed and optimized in synergy.

This dissertation attempts to close the light-AI virtuous cycle and proposes a holistic solution to enable scalable, robust, and adaptive photonic ML computing platforms with AI-assisted co-design and automation methodologies. The proposed co-design stack includes three aspects: specialized photonic AI hardware design, scalable ONN on-chip training algorithms, and AI-assisted intelligent automated photonic hardware design flow. The dissertation proposes new photonic neural networks with customized devices/circuits/architectures to solve scalability and robustness concerns with various cross-layer hardware/software co-optimization methods. It also introduces a series of customized algorithms for ONN on-chip training to enable self-learnable and adaptive photonic analog accelerators. Finally, the dissertation explores AI-assisted Maxwell equation-solving frameworks and automatic differentiable photonic integrated circuit (PIC) topology search frameworks for beyond-human design productivity, efficiency, and quality, closing the virtuous cycle of photonics for AI and AI for photonics.

The efficiency and performance of proposed hardware and algorithm designs are verified through simulation and compared with state-of-the-art designs on various machine learning tasks and experimentally demonstrated on photonic neural chip tapeout. Our specialized hardware and customized algorithms can synergistically improve the area efficiency, energy efficiency, noise tolerance, and adaptability of photonic AI computing platforms with beyond-human design productivity and quality.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Photonic Computing Background and Basics

Deep neural networks (DNNs) have received an explosion of interest for their superior performance in artificial intelligence (AI) tasks. As shown in Fig. 1.1, the computing capacity is in an arms race with the rapidly escalating model size and data volume. Certain applications, e.g., autonomous vehicles, data centers, and edge devices, have strict efficiency, latency, and bandwidth constraints, raising a surging need to develop more efficient computing solutions. However, as CMOS scaling is slowing down, it becomes increasingly challenging for conventional electrical processors to support such massively parallel and energy-hungry DNN workloads. Limited clock frequency, high latency, high heat density, and large energy consumption of CPUs, FPGAs, and GPUs motivate us to seek an alternative solution using optics.

In recent years, there has been a trend to evolve from electronics to

---

This Introduction section is based on the following publication.

1. Jiaqi Gu, Chenghao Feng, Hanqing Zhu, Ray T. Chen and David Z. Pan, "Light in AI: Toward Efficient Neurocomputing with Optical Neural Networks - A Tutorial," IEEE Transactions on Circuits and Systems–II: Express Briefs (TCAS-II), Apr. 2022.

I am the main contributor in charge of the literature review and manuscript writing.

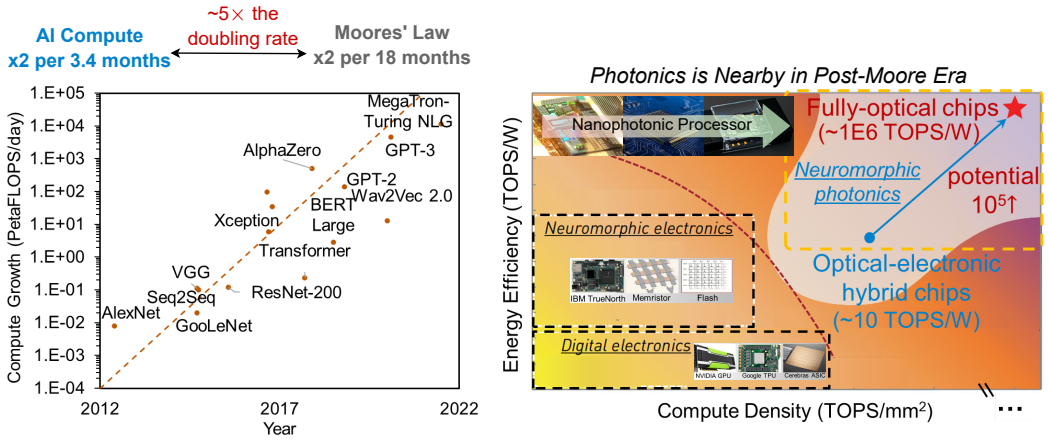


Figure 1.1: Exponential increase in computing demand for modern AI models (data from OpenAI and NVIDIA). Photonic computing shows great potential in efficiency and performance breakthrough compared to electronics.

photonics in next-generation computing systems for ultra-fast, and energy-efficient computing, especially parallel linear operations. Figure 1.2 explains how to use integrated photonic devices/circuits to realize important computing primitives in neural networks, including modulator-based scalar multiply, interferometer-based unitary matrix multiplication, and photonic integrated circuits for matrix-vector multiplication. A detailed introduction to the Mach-Zehnder Interferometer (MZI) and MZI-based ONNs can be found in Appendix .1.1.

Figure 1.3 compares electrical and photonic computing in speed, parallelism, and energy efficiency. For photonic matrix multiplication, the input vector can be encoded into optical signals using high-speed optical modulators, while the weight matrix can be programmed to the circuit transfer matrix by tuning the active devices. **Ultra-fast** MVM is implemented at the speed of

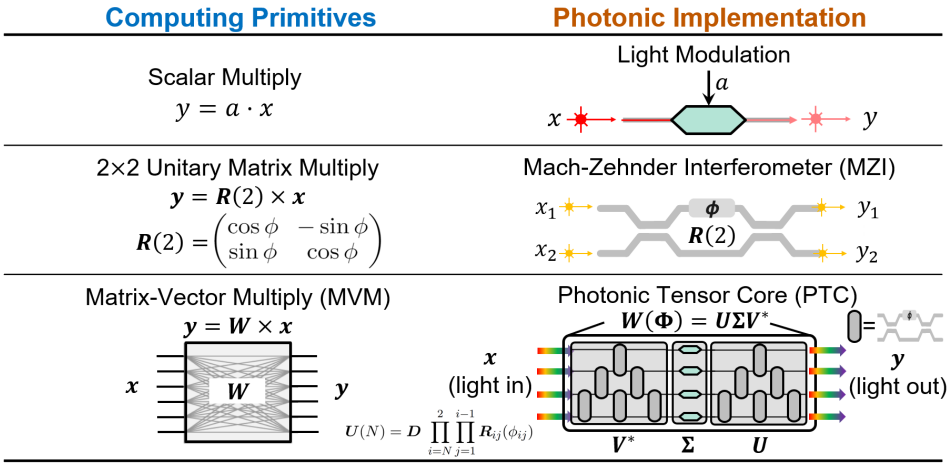


Figure 1.2: Examples of using photonic devices/structures to implement important computing primitives in neural networks.

light with sub-nanosecond latency by propagating light through an optical system. **Massive parallelism** can be supported by reusing the same hardware through wavelength/mode/polarization-division multiplexing. Photonic computing also shows superior **energy efficiency**. Unlike electrical circuits which usually have heat density issues due to the ohmic heat dissipation, passive photonic circuits if configured intrinsically consume near-zero static power, while only input/output consumes the most power. Hence the energy cost linearly scales with the matrix size, instead of quadratically like electrical matrix units.

1.2 Photonic AI Literature Review and Challenges

Photonic AI, as a newly emerging field, is booming both in academia and industry, shown in Fig. 1.4. Various photonic neural network designs based on different mechanisms with experimental demonstration have been

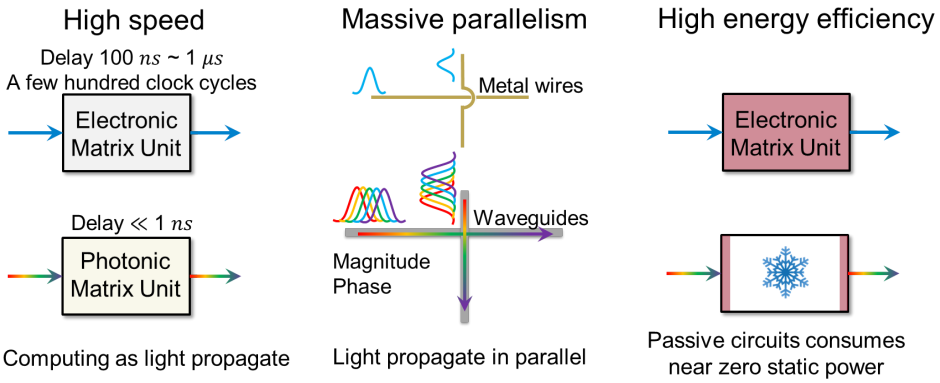


Figure 1.3: Compared to electrical computing, photonic computing is ultra-fast, massively parallel, and energy-efficient.

presented by academia. The integrated optical neural networks (ONNs) based on silicon photonics have attracted extensive research interest and represented a paradigm shift in efficient AI given their competitive integration density, ultra-high energy efficiency, and good CMOS-compatibility [171, 215, 170]. With potential petaOPS/mm^2 compute density and attojoule/MAC energy efficiency, fully-optical NNs can have orders-of-magnitude higher performance and efficiency than their electrical counterparts [171, 215, 170, 227, 50, 67].

Research efforts have been made on reservoir computing [11], spike processing [188], and Ising machines [164]. An important category of ONNs is based on coherent photonic integrated circuits. Based on matrix singular value decomposition (SVD), Shen *et al.* [171] designed and fabricated a fully optical NN that achieves a multi-layer perception (MLP) with MZI arrays. Later, a slimmed ONN [253] and a Fourier-transform-based ONN design [70] were proposed to reduce the area cost of photonic tensor cores. Diffractive optical

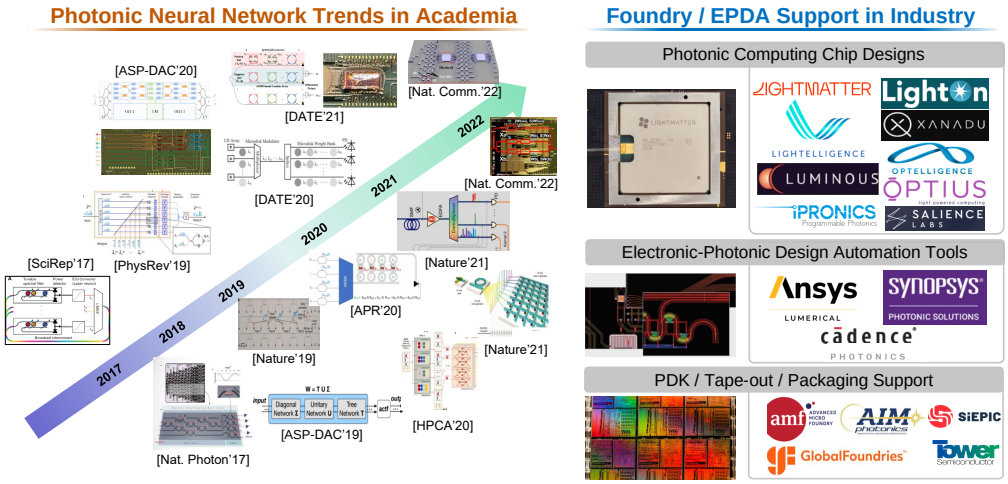


Figure 1.4: Photonic AI is booming both in academia and industry. Various photonic neural network designs are emerging with strong support from design companies, EDA vendors, and foundries.

components [260] and subwavelength devices [221, 212] were demonstrated to implement optical neuromorphic computing.

Another category of ONNs is based on multi-wavelength incoherent photonic integrated circuits, which do not require phase stability. ONN architectures based on micro-ring resonator weight banks [190, 131] were introduced to achieve direct matrix-vector multiplication with multiple wavelengths and compact photonic devices. Crossbar arrays with non-volatile photonic phase-change material (PCM) cells and frequency micro-combs were demonstrated to implement matrix multiplication with lower weight maintaining power [50]. Photonic convolution processors based on frequency-domain weight encoding and dispersion-based time shift were proposed to achieve high-speed image/video processing [227]. On the industry side, electronic-photonic

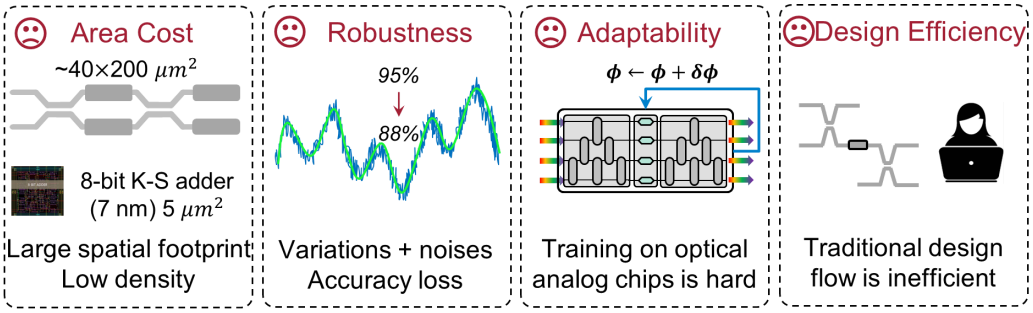


Figure 1.5: Designing emerging photonic AI computing platform encounters challenges in area scalability, noise robustness, adaptability, and design efficiency.

computing and interconnect systems have been prototyped with strong design/layout support from electronic-photonic design automation (EPDA) vendors and tape-out/packaging support from foundries.

Besides the above advantages, ONNs currently still encounter significant design challenges, shown in Fig. 1.5.

Large Area Cost: Due to the large spatial footprint of photonic devices, several orders-of-magnitude bulkier than nanometer transistors, photonic integrated circuits do not have high packing density and are generally not competitive in area efficiency. Hence, one may not be able to squeeze a large number of photonic components or a large matrix on the chip, which limits the hardware scalability and compute density.

Noise Robustness Issue: Due to the analog nature of photonic circuits, unlike CPUs/GPUs with high computing fidelity, photonic computing engines are sensitive to manufacturing errors, process variations, dynamic/static chip

noises, and environmental changes. The above non-ideal effects inevitably lead to undesired accuracy loss or even malfunction in real-world machine learning tasks.

Adaptability Challenge: Training the optical analog chips is considerably more challenging than training neural networks on conventional GPUs due various photonic hardware constraints. Therefore, it is hard to freely reprogram the circuits to adapt to changing working conditions or different workloads, which hinders many important edge applications, e.g., online learning and local adaptation.

Low Design Efficiency: Conventional photonic hardware design flows are based on hand-crafted designs based on standard device library and time-consuming numerical simulation, which limits the turn-around speed and fails to explore the huge design space.

Future photonic AI needs synergistic design technology and a holistic solution to push the limits of the practical deployment of photonic neural accelerators.

1.3 Overview of this Dissertation

This dissertation attempts to address the above challenges for photonic AI and close the light-AI virtuous cycle. Holistic solutions are developed to enable scalable, robust, and adaptive photonic ML computing platforms with AI-assisted co-design and automation methodologies. As shown in

Fig. 1.6, the proposed co-design stack consists of three synergistic aspects in the photonics-AI virtuous cycle: (1) hardware/software co-design for photonic neural network inference accelerators; (2) efficient on-chip training algorithms for self-learnable photonic neural engines; and (3) AI-assisted intelligent photonic integrated circuit design automation methodologies.

Chapter 2 of this dissertation proposes new ONN architectures and co-design methods to address the fundamental scalability and robustness issues in photonic AI accelerators. The proposed architectures include (1) FFT-ONN: a hardware-efficient butterfly-style ONN architecture with hardware-aware structured pruning toward higher compute density and lower noise sensitiv-

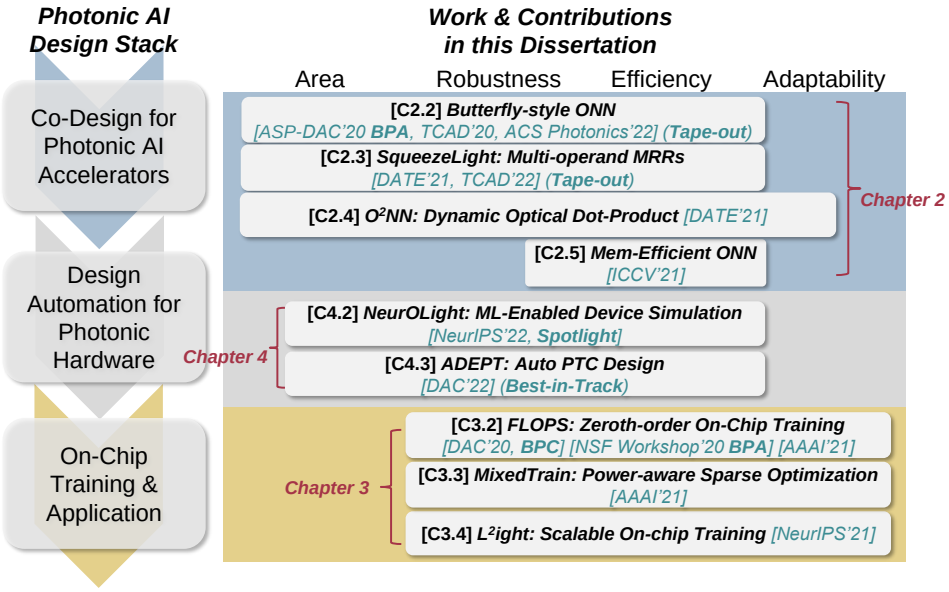


Figure 1.6: Summary of my PhD research in photonic AI. The hardware/software co-design stack tackles all critical challenges in optical AI with novel cross-layer device, circuit, architecture, and algorithm co-design.

ity; (2) SqueezeLight: an ultra-compact photonic neuron design based on customized multi-operand microrings and fine-grained device-level sparsity to break the compute density bottleneck of one operation/device; (3) O^{2}NN: a flexible photonic dot-product engine that supports dynamic tensor products with both operands being high-speed full-range optical signals; and (4) Memory-efficient ONN: a memory-efficient photonic accelerator architecture that leverages the weight matrix redundancy to significantly reduce memory access cost and thus system-level energy consumption via *in-situ* weight generation and mixed-precision computations. The proposed ONN architectures, together with the circuit-model co-optimization methods, enable scalable, robust, flexible, and energy-efficient photonic AI accelerators.

In Chapter 3, to enable self-learnable photonic AI engines to simultaneously address noise robustness and adaptability concerns, the following ONN on-chip training algorithms are introduced: (1) FLOPS: forward-only gradient estimator to enable ONN on-chip training with zeroth-order gradient descent; (2) MixedTrain: a mixed-training strategy with a power-aware sparse coordinate descent optimizer to further boost training scalability and energy efficiency; and (3) L^{2}ight: a subspace optimization framework with multi-level sparsity to enable million-parameter ONN online training and task transfer with significant hardware cost reduction.

Chapter 4 explores novel methods using AI and optimization for photonic circuit design automation to enhance design productivity, efficiency, and quality. Both device-level and circuit-level design automation techniques are

proposed: (1) NeurOLight, an AI-assisted Maxwell equation solving framework for photonic device simulation acceleration, and (2) ADEPT, an automatic differentiable photonic integrated circuit (PIC) topology search framework for compact, expressive, and noise-robust photonic tensor core designs. The proposed ML-assisted simulation and optimization-based photonic circuit design flow enable a fully-automated and efficient design automation flow, closing the virtuous cycle of photonics for AI and AI for photonics.

Chapter 2

Hardware/Software Co-Design of Photonic Neural Network Accelerator

2.1 Introduction

DNNs have demonstrated superior performance in a variety of intelligent tasks, for example, convolutional neural networks (CNNs) on image classification [111] and recurrent neural networks on language translation [139]. Multilayer perceptrons (MLPs) are among the most fundamental components in modern DNNs, which are typically used as regression layers, classifiers, embedding layers, attention layers, etc. However, it becomes challenging for traditional electrical digital von Neumann schemes to support escalating computation demands owing to the speed and energy inefficiency [171, 230, 56, 55, 144]. To resolve this issue, significant efforts have been made in the hardware design of neuromorphic computing frameworks to improve the computational speed of neural networks, such as electronic architectures [46, 208, 244] and photonic architectures [188, 12, 57, 262, 145]. Among extensive neuromorphic computing systems, optical neural networks distinguish themselves by ultrahigh bandwidth, ultralow latency, and near-zero energy consumption. Even though ONNs are currently not competitive in terms of area cost, they still offer a promising alternative approach to microelectronic implementations, given the

above advantages.

Recently, several works demonstrated that MLP inference can be efficiently performed at the speed of light with optical components, e.g., spike processing [188] and reservoir computing [11]. They claimed a photodetection rate over 100 GHz in photonic networks, with near-zero energy consumption if passive photonic components are used [204]. Based on matrix singular value decomposition and unitary matrix parameterization [161, 163], Shen *et al.* [171] designed and fabricated a fully optical neural network that achieves an MLP with MZI arrays. Once the weight matrices in the MLP are trained and decomposed, thermo-optic phase shifters on the arms of MZIs can be set up accordingly. Since the weight matrices are fixed after training, this fully optical neural network can be completely passive, thus minimizing the total energy consumption. Another work [253] proposed a slimmed ONN architecture (TΣU) based on the previous one [171], which substitutes one of the unitary blocks with a sparse tree network. Incoherent ONN architectures based on micro-ring resonator weight banks [190, 131] were introduced to achieve direct matrix-vector multiplication with multiple wavelengths and compact devices.

However, previous photonic accelerators were designed to be weight-stationary and targeted universal programmable linear operations. Their area efficiency, noise robustness, or flexibility are rather limited, which has become the bottleneck of their practical application.

In this chapter, we will present a series of domain-specific photonic computing engines with software-hardware co-design methodologies to trade

among matrix expressivity, hardware efficiency, robustness, and flexibility, which are beyond the universal linear units in previous work. In the rest of this chapter, Section 2.2 introduces a compact butterfly-style coherent ONN architecture `FFT-ONN` with improved area efficiency and noise robustness. Section 2.3 presents an incoherent ONN architecture `SqueezeLight` based on our customized multi-operand micro-ring resonators to achieve extreme compute density and power efficiency. Section 2.4 introduces a flexible photonic dot-product engine $\mathtt{O}^2\mathtt{NN}$ that enables both dot-product operands to be dynamically-encoded optical signals. Section 2.5 focuses on a memory-efficient photonic accelerator architecture that allows the ultra-fast computing engine to generate its operands *in-situ* in the analog domain.

2.2 FFT-ONN: Area-Efficient Butterfly-style Optical Neural Networks

In addition to hardware implementation, recent advances in neural architecture design and network compression techniques have shown a significant reduction in computational cost. For example, structured neural networks (SNNs)[121] were proposed to significantly reduce computational complexity and thus, become amenable to hardware. Besides, network pruning offers another powerful approach to slimming down neural networks by cutting off insignificant neuron connections. While non-structured pruning [83] produces random neuron sparsity, group sparsity regularization [174] and structured pruning [208] can lead to better network regularity and hardware efficiency. However, readily-available pruning techniques are rather challenging to be applied to the SVD-based architecture due to some issues, such as ac-

---

This `FFT-ONN` section is based on the following publications.

1. Jiaqi Gu, Zheng Zhao, Chenghao Feng, Mingjie Liu, Ray T. Chen, and David Z. Pan, "Towards Area-Efficient Optical Neural Networks: An FFT-based Architecture," IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC), Jan. 2020.
2. Jiaqi Gu, Zheng Zhao, Chenghao Feng, Zhoufeng Ying, Mingjie Liu, Ray T. Chen, and David Z. Pan, "Towards Hardware-Efficient Optical Neural Networks: Beyond FFT Architecture via Joint Learnability," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2020.
3. Chenghao Feng*, Jiaqi Gu*, Hanqing Zhu, Zhoufeng Ying, Zheng Zhao, David Z. Pan, and Ray T. Chen, "A Compact Butterfly-style Silicon Photonic-electronic Neural Chip for Hardware-efficient Deep Learning," ACS Photonics, Nov. 2022. (∗Equal Contribution, Chenghao Feng is in charge of chip testing and measurement. I am in charge of structure design, circuit layout, model training, and ML task evaluation.)

I am the main contributor in charge of problem formulation, algorithm development, and experimental validations.

curacy degradation and hardware irregularity. The gap between hardware-aware pruning and the SVD-based architecture gives another motivation for a pruning-friendly ONN architecture.

In this work, we propose a new ONN architecture that improves area efficiency over previous ONN architectures. It leverages the optical fast Fourier transform (OFFT) and its inverse (OIFFT) to implement structured neural networks, achieving lower optical component utilization. It also enables the application of structured pruning given its architectural regularity. The proposed architecture partitions the weight matrices into block-circulant matrices [63] and efficiently performs circulant matrix multiplication through OFFT/OIFFT. We also adopt a two-phase software training flow with structured pruning to further reduce photonic component utilization while maintaining comparable inference accuracy to previous ONN architectures. We extend this architecture to a hardware-efficient optical convolutional neural network design with joint learnability, and demonstrate its superior power efficiency and noise-robustness compared with Fourier transform-based design. The main contributions of this work are as follows:

- We propose a novel, area-efficient optical neural network architecture with OFFT/OIFFT, and exploit a two-phase software training flow with structured pruning to learn hardware-friendly sparse neural networks that directly eliminate part of OFFT/OIFFT modules for further area efficiency improvement.

- We experimentally show that pruning is challenging to be applied to previous ONN architectures due to accuracy loss and trainability issues.
- We experimentally demonstrate that our proposed architecture can lead to an area saving of 2.2∼3.7× compared with the previous SVD-based ONN architecture, with negligible inference accuracy loss.
- We extend the architecture to a novel design for microdisk-based frequency-domain optical convolutional neural networks with high parallelism.
- We propose a trainable frequency-domain transform structure and demonstrate it can be pruned with high sparsity and outperforms traditional Fourier transform with less component count, higher power efficiency, and better noise-robustness.
- We experimentally demonstrate the butterfly-style **photonic-electronic neural chip with silicon-proven evaluation** on various image recognition/disease detection tasks.

2.2.1 Preliminaries

In this section, we introduce the background knowledge for our proposed architecture. We discuss principles of cirulant matrix representation and its fast computation algorithms in Section 2.2.1.1 and illustrate structured pruning techniques with Group Lasso regularization in Section 2.2.1.2.

2.2.1.1 FFT-based Circulant Matrix Computation

Unlike the SVD-based ONNs which focus on classical MLPs, our proposed architecture is based on structured neural networks (SNNs) with circulant matrix representation. SNNs are a class of neural networks that are specially designed for computational complexity reduction, whose weight matrices are regularized using the composition of structured sub-matrices [121]. Among all structured matrices, circulant matrices are often preferred in recent SNN designs.

As an example, we show an $n \times n$ circulant matrix $\boldsymbol{W}$ as follows,

$$\begin{bmatrix} w_0 & w_{n-1} & \cdots & w_1 \\ w_1 & w_0 & \cdots & w_2 \\ \vdots & \vdots & \ddots & \vdots \\ w_{n-1} & w_{n-2} & \cdots & w_0 \end{bmatrix}.$$

The first column vector $\boldsymbol{w} = [w_0, w_1, \ldots, w_{n-1}]^T$ represents all independent parameters in $\boldsymbol{W}$, and other columns are just its circulation.

According to [63], circulant matrix-vector multiplication can be efficiently calculated through fast Fourier transform. Specifically, given an $n \times n$ circulant matrix $\boldsymbol{W}$ and a length-n vector $\boldsymbol{x}$, $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$ can be efficiently performed with $\mathcal{O}(n \log n)$ complexity as,

$$\boldsymbol{y} = \mathcal{F}^{-1}\big(\mathcal{F}(\boldsymbol{w}) \odot \mathcal{F}(\boldsymbol{x})\big), \tag{2.1}$$

where $\mathcal{F}(\cdot)$ represents n-point real-to-complex fast Fourier transform (FFT), $\mathcal{F}^{-1}(\cdot)$ represents its inverse (IFFT), and $\odot$ represents complex vector element-wise multiplication.

SNNs benefit from high computational efficiency while maintaining comparable model expressivity to classical NNs. Theoretical analysis [247] shows that SNNs can approximate arbitrary continuous functions with arbitrary accuracy given enough parameters, and are also capable of achieving the identical error bound to that of classical NNs. Therefore, based on SNNs with circulant matrix representation, the proposed architecture features low computational complexity and comparable model expressivity.

2.2.1.2 Structured Pruning with Group Lasso Penalty

The proposed ONN architecture enables the application of structured pruning to further save optical components while maintaining accuracy and structural regularity. Structured pruning trims the neuron connections in NNs to mitigate computational complexity. Unlike ℓ_1 or ℓ_2 norm regularization, which produces arbitrarily-appearing zero elements, structured pruning with Group Lasso regularization[208],[58] leads to zero entries in groups. This coarse-grained sparsity is more friendly to hardware implementation than non-structured sparsity. The formulation of Group Lasso regularization term is given as follows,

$$\boldsymbol{L}_{GL} = \sum_{g=0}^{G} \sqrt{1/p_g} \|\beta_g\|_2, \tag{2.2}$$

where G is the total number of parameter groups, β_g is the parameter vector in the g-th group, $\|\cdot\|_2$ represents ℓ_2 norm, p_g represents the vector length of β_g, which accounts for the varying group sizes. Intuitively, the ℓ_2 norm penalty $\|\beta_g\|_2$ encourages all elements in the g-th group to converge to 0, and the group-

wise summation operation is equivalent to group-level ℓ_1 norm regularization, which contributes to the coarse-grained sparsity. Leveraging the structured pruning together with Group Lasso regularization, our proposed architecture can save even more photonic components.

2.2.2 Proposed Photonic MLP with FFT-inspired Butterfly Transforms

In this section, we will discuss details about the proposed architecture and pruning method. In the first part, we illustrate the five stages of our proposed architecture. In the second part, we focus on the two-phase software training flow with structured pruning.

2.2.2.1 FFT-inspired Photonic MLP Architecture

Based on structured neural networks, our proposed architecture implements a structured version of MLPs with circulant matrix representation. A single layer in the proposed architecture performs linear transformation via block-circulant matrix multiplication $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$. Consider an n-input, m-output layer, the weight matrix $\boldsymbol{W} \in \mathbb{R}^{m\times n}$ is partitioned into $p \times q$ submatrices, each being a $k \times k$ circulant matrix. To perform tiled matrix multiplication, the input $\boldsymbol{x}$ is also partitioned into q segments $\boldsymbol{x} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{q-1})$. Thus $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$ can be performed in a tiled way,

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{p-1} \end{pmatrix} = \begin{pmatrix} \sum_{j=0}^{q-1} \boldsymbol{W}_{0j}\boldsymbol{x}_j \\ \sum_{j=0}^{q-1} \boldsymbol{W}_{1j}\boldsymbol{x}_j \\ \vdots \\ \sum_{j=0}^{q-1} \boldsymbol{W}_{p-1j}\boldsymbol{x}_j \end{pmatrix}. \tag{2.3}$$

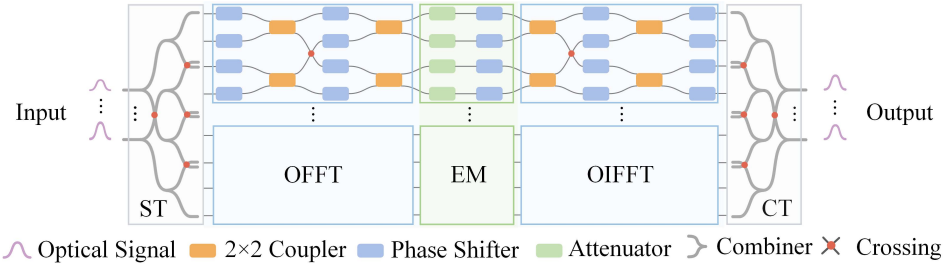


Figure 2.1: Schematic diagram of a single layer of the proposed architecture. All adjacent phase shifters on the same waveguide are already merged into one phase shifter.

The i_{th} segment $y_i = \sum_{j=0}^{q-1} \boldsymbol{W}_{ij}\boldsymbol{x}_j$ is the accumulation of q independent circulant matrix multiplications. Each $\boldsymbol{W}_{ij}\boldsymbol{x}_j$ can be efficiently calculated using the fast computation algorithm mentioned in Eq. (2.1). Based on the aforementioned equations, we realize block-circulant matrix multiplication $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$ in five stages: 1) Splitter tree (ST) stage to split input optical signals for reuse; 2) OFFT stage to calculate $\mathcal{F}(\boldsymbol{x})$; 3) element-wise multiplication (EM) stage to calculate $\mathcal{F}(\boldsymbol{w_{ij}}) \odot \mathcal{F}(\boldsymbol{x_j})$ as described in Eq. (2.1); 4) OIFFT stage to calculate $\mathcal{F}^{-1}(\cdot)$; 5) combiner tree (CT) stage to accumulate partial multiplications to form the final results. $\mathcal{F}(\boldsymbol{w}_{ij})$ can be precomputed and encoded into optical components, thus there is no extra stage to physically perform it. The schematic diagram of our proposed architecture is shown in Fig. 2.1. Details of the above five stages will be discussed in the rest of this section.

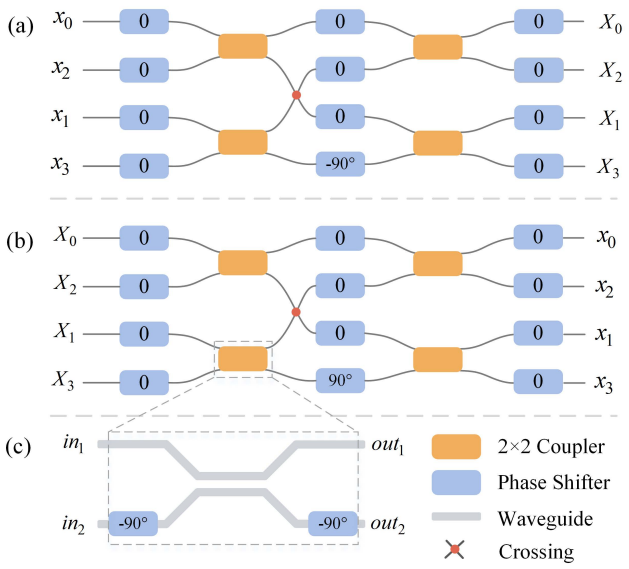


Figure 2.2: Schematics of (a) 4-point OFFT, (b) 4-point OIFFT, and (c) 2×2 coupler. Note that phase shifters shown above are not merged for structural completeness consideration.

OFFT/OIFFT Stages To better model the optical components used to implement the OFFT/OIFFT stages, we introduce a unitary FFT as,

$$\boldsymbol{X}_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \boldsymbol{x}_n e^{-i\frac{2\pi kn}{N}} \quad k = 0, 1, \cdots, N-1. \tag{2.4}$$

We denote this special operation as $\widehat{\mathcal{F}}(\cdot)$ and its inverse as $\widehat{\mathcal{F}}^{-1}(\cdot)$, to distinguish from the original FFT/IFFT operations. Equivalently, we re-write the circulant matrix multiplication with the above new operations,

$$\boldsymbol{y} = \widehat{\mathcal{F}}^{-1}\big(\mathcal{F}(\boldsymbol{w}) \odot \widehat{\mathcal{F}}(\boldsymbol{x})\big). \tag{2.5}$$

This unitary FFT operation can be realized with optical components. We first give a simple example of the optical implementation of a 2-point

unitary FFT. As shown in Eq. (2.7), the transformation matrix of a 2-point unitary FFT can be decomposed into three transform matrices. They can be directly mapped to a 3-dB directional coupler with two $-\pi/2$ phase shifters on its lower input/output ports. The transfer matrix of a 50/50 optical directional coupler is given by,

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & j\\ j & 1\end{pmatrix}. \tag{2.6}$$

The transfer function of a phase shifter is $out = in \cdot e^{j\phi}$. For brevity, we refer to this cascaded structure as a 2×2 coupler, which is shown in Fig. 2.2(c).

$$\begin{aligned}\begin{pmatrix}out_1\\ out_2\end{pmatrix} =& \frac{1}{\sqrt{2}}\begin{pmatrix}in_1 + in_2\\ in_1 - in_2\end{pmatrix}\\ =& \underbrace{\begin{pmatrix}1 & 0\\ 0 & -j\end{pmatrix}}_{\text{output phase shifter}} \underbrace{\frac{1}{\sqrt{2}}\begin{pmatrix}1 & j\\ j & 1\end{pmatrix}}_{\text{directional coupler}} \underbrace{\begin{pmatrix}1 & 0\\ 0 & -j\end{pmatrix}}_{\text{input phase shifter}} \begin{pmatrix}in_1\\ in_2\end{pmatrix}\end{aligned} \tag{2.7}$$

Based on 2×2 couplers and phase shifters, larger-sized OFFT/OIFFT can be constructed with a butterfly structure. The schematics of a simple 4-point OFFT and OIFFT are shown in Fig. 2.2(a) and Fig. 2.2(b). Extra 0-degree phase shifters are inserted for phase-tuning purposes.

This butterfly-structured OFFT may have scalability issues because the number of waveguide crossings (CR) will increase rapidly when the number of points gets larger. However, this unsatisfying scalability will not limit our proposed architecture for two reasons. First, only small values of k, e.g., 2, 4, 8, will be adopted to balance hardware efficiency and model expressivity. Second, input and output sequences can be reordered to avoid unnecessary waveguide crossings, as shown in Fig. 2.2.

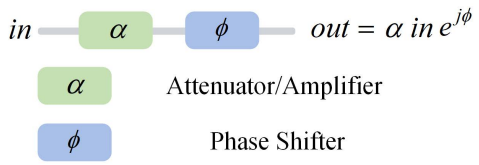


Figure 2.3: Complex number multiplication realized by cascaded attenuator/amplifier and phase shifter.

EM Stage In the EM stage, complex vector element-wise multiplications will be performed in the Fourier domain as $\alpha e^{\phi} \cdot I_{in} e^{\phi_{in}} = \alpha\ I_{in} e^{\phi_{in}+\phi}$, where I_{in} and ϕ_{in} are magnitude and phase of input Fourier light signals respectively. Leveraging the polarization of light, we use optical attenuators (AT) or amplification materials/optical on-chip amplifiers with a scaling factor α to realize modulus multiplication $\alpha \cdot I_{in}$ and phase shifters with ϕ phase shift for argument addition $e^{j(\phi+\phi_{in})}$, which is shown in Fig. 2.3.

ST/CT Stage We introduce tree-structured splitter/combiner networks to realize input signal splitting and output signal accumulation, respectively. To reuse input segments $\boldsymbol{x}_j$ in multiple blocks, optical splitters (SP) are used to split optical signals. Similarly, to accumulate partial multiplication results, i.e., $\boldsymbol{y}_i = \sum_{j=0}^{q-1} \boldsymbol{W}_{ij}\boldsymbol{x}_j$, we adopt optical combiners (CB) for signal addition. Given that optical splitters can be realized by using combiners in an inversed direction, we will focus on the combiner tree structure for brevity.

The transfer function of an N-to-1 optical combiner is,

$$out = \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} in_l. \tag{2.8}$$

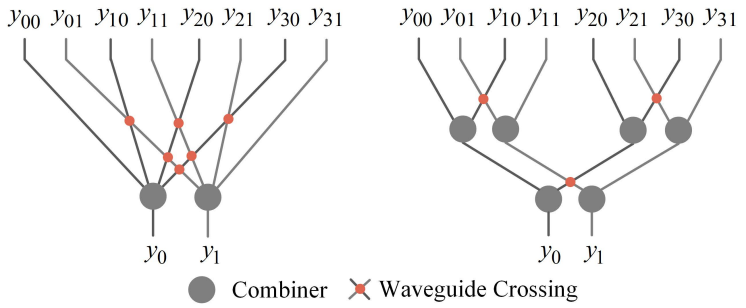


Figure 2.4: Comparison between direct combining (left) and combiner tree (right) with 4 length-2 vectors accumulated.

Accumulating q length-k vectors by simply using k q-to-1 combiners introduces a huge number of waveguide crossings which may cause intractable implementation difficulty. Also, combiners with more than two ports are still challenging to manufacture. In order to alleviate this problem, we adopt a tree-structured combiner network, shown in Fig. 2.4. This combiner tree consists of $k(q-1)$ combiners and reduces the number of waveguide crossings to $k(k-1)(q-1)/2$. Given that combiners will cause optical intensity loss by a factor of $1/\sqrt{N}$ as shown in Eq. (2.8), we assume there will be optical amplifiers added to the end to compensate for this loss.

In terms of cascading multiple layers, our proposed FFT-based MLP is fully optical, such that the output optical signals can be directly fed into the next layer without optical-electrical-optical (O-E-O) conversion. At the end of the last layer, photo-detection is used for signal readout, and the phase information of the outputs is removed, which can be fully modeled during our training process without causing any accuracy loss.

2.2.2.2 Two-Phase Training Flow with Structured Pruning

Structured pruning can be applied to our proposed architecture during training given its architectural regularity. We propose a two-phase software training flow with structured pruning to train a more compact ONN. We first pre-train the model with the Group Lasso regularization term to explore a good initialization. Then we progressively prune the weight blocks by forcing some groups to 0 based on an increasing threshold T such that the corresponding hardware modules can be completely eliminated. Meanwhile, we finetune the model to recover accuracy.

2.2.2.3 Theoretical Analysis on the Proposed Photonic MLP Architecture

In this section, we analyze the hardware utilization and compare it to previous architectures.

We derive a theoretical estimation of hardware utilization of the proposed architecture, the SVD-based architecture [171], and the slimmed $T\Sigma U$-based architecture [253]. By comparing the hardware component utilization, we show that theoretically, our proposed architecture costs fewer optical components than the SVD-based architecture and $T\Sigma U$-based architecture. The comparison results are summarized in Table 2.1 for a clear demonstration.

For simplicity, we convert all area-costly components, i.e., 2×2 couplers, MZIs, and attenuators, to 3-dB directional couplers (DCs) and phase shifters (PSs). Specifically, one 2×2 coupler can be taken as one DC and two PSs,

Table 2.1: Summary of hardware component cost on an $m \times n$ layer in SVD-based ONN and our proposed architecture (size-k circulant blocks). Most area-consuming components are considered. PS and DC represent a phase shifter and a directional coupler.

	#DC	#PS
SVD-ONN	$m(m-1)+n(n-1)+\max(m,n)$	$\frac{m(m-1)+n(n-1)}{2}$
$T\Sigma U$-ONN	$m(m-1)+2n+\max(m,n)$	$\frac{m(m-1)+2n}{2}$
Our ONN	$\frac{mn(\log_2 k+1)}{k}$	$\frac{mn(2\log_2 k+1)}{k}$

and one MZI can be taken as two DCs and one PS. Since an attenuator can be achieved by a single-input directional coupler with an appropriate transfer factor, we count one attenuator as one DC.

Given an n-input, m-output layer, the SVD-based implementation requires $m(m-1)/2 + n(n-1)/2$ MZIs and $\max(m,n)$ attenuators to realize the weight matrix. Therefore, with the aforementioned assumption, the total number of components it costs is given by,

$$\begin{aligned} \#DC_{\text{SVD}} &= m(m-1)+n(n-1)+\max(m,n) \\ \#PS_{\text{SVD}} &= m(m-1)/2+n(n-1)/2. \end{aligned} \tag{2.9}$$

For the slimmed $T\Sigma U$-based ONN architecture [253], one unitary matrix is replaced by a compact sparse tree network consisting of n MZIs. Therefore, the component utilization of $T\Sigma U$-based ONN is given by,

$$\begin{aligned} \#DC_{T\Sigma U} &= m(m-1)+2n+\max(m,n) \\ \#PS_{T\Sigma U} &= m(m-1)/2+n. \end{aligned} \tag{2.10}$$

For our architecture, each $k \times k$ circulant matrix costs k attenuators and corresponding components required by k-point OFFT/OIFFT. The following

formulation gives the number of components for a k-point OFFT/OIFFT.

$$\begin{aligned} \#DC_{\texttt{OFFT}}(k) &= 2 \times \#DC_{\texttt{OFFT}}(k/2) + k/2 = \frac{k}{2}\log_2 k \\ \#PS_{\texttt{OFFT}}(k) &= k(\log_2 k + 1) \end{aligned} \tag{2.11}$$

A phase shift is physically meaningful only when it is within $(-2\pi, 0]$ as phases can wrap around. Hence, multiple successive phase shifters on the same segment of a waveguide can be merged as one phase shifter, which can be seen when comparing Fig. 2.1 and Fig. 2.2. Then the total number of components used in our design to implement an $m \times n$ weight matrix with size-k circulant sub-matrices is given by,

$$\begin{aligned} \#DC_{\texttt{Ours}}(k) &= \frac{m}{k} \times \frac{n}{k} \times (2 \times \#DC_{\texttt{OFFT}}(k) + k) \\ &= \frac{mn}{k}(\log_2 k + 1) \\ \#PS_{\texttt{Ours}}(k) &= \frac{m}{k} \times \frac{n}{k} \times (2 \times \#PS_{\texttt{OFFT}}(k) - k) \\ &= \frac{mn}{k}(2\log_2 k + 1). \end{aligned} \tag{2.12}$$

In practical cases, k will be set to small values, such as 2, 4, and 8. Given arbitrary values of m and n, the proposed architecture costs theoretically fewer optical components than the SVD-based architecture.

We also give a qualitative comparison with incoherent microring resonator-based ONNs (MRR-ONN). There are two MRR-ONN variants. The first one is based on all-pass mircroring (MR) resonators [131]. The second one proposed later is based on the differential add-drop MR resonators [190]. We assume an $M \times N$ matrix multiplication in the following tasks. Since the physical dimensions of MRs are smaller than couplers and phase shifters in general, thus

a lower area cost can be expected for MRR-ONNs compared with ours. However, in terms of model expressivity, all-pass MRR-ONN is much less than the other two since it only supports positive weights. Add-drop MRR-ONN and our architecture can support a full-weight range without positive limitations. In terms of robustness, MRR-ONNs are less robust since the MR resonators are more sensitive to device variations and environmental changes than phase shifters. Especially for add-drop MRR-ONN, its differential structure amplifies the noise on the MR transmission factor by 2 times on its represented weight. Thus less robustness can be expected for MRR-ONNs. Furthermore, in terms of power consumption, our architecture can benefit from structured sparsity to obtain a much lower power, which will be shown in later Experimental Results sections. In contrast, for MRR-ONNs, even though a group of weights gets pruned to zero values, the corresponding MR resonators are not idle [131, 190], which means its power consumption can barely benefit from pruning techniques. Therefore, from the above qualitative analysis, though our architecture demonstrates a relatively larger footprint than MRR-ONNs, we outperform them in terms of model expressivity, robustness, and power.

2.2.3 Photonic CNN with Learnable Frequency-domain Transforms

To demonstrate the applicability of the proposed architecture, we extend this architecture to a compact frequency-domain microdisk (MD)-based optical convolutional neural network (CNN) with joint learnability, where the convolutional kernels and frequency-domain transforms are jointly optimized

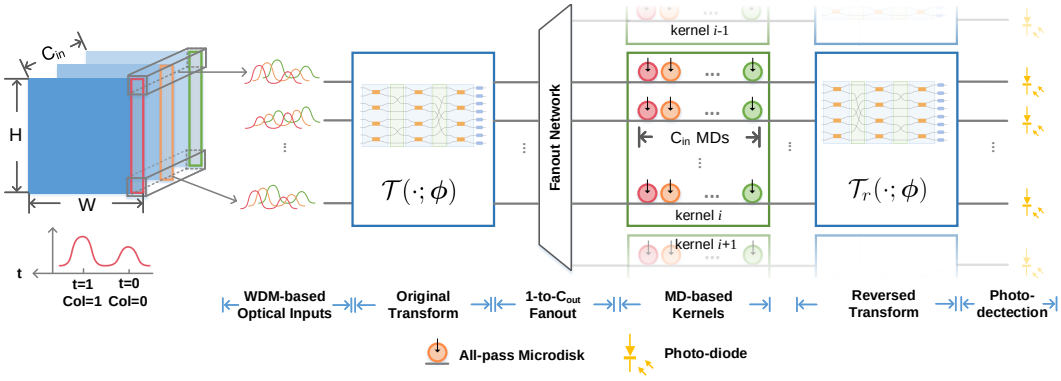


Figure 2.5: Architecture of an MD-based optical convolutional layer with trainable frequency-domain transforms. Columns of input features are fed into the architecture in different time steps. Multiple kernels are implemented with multiple photonic chiplets to achieve higher parallelism.

during hardware-aware training.

2.2.3.1 Microdisk-based Frequency-domain CNN Architecture

Given the two-dimensional (2-D) nature of photonic integrated chips (PICs), currently, we only demonstrate optical designs for MLPs. Previous solutions to accelerate convolutional neural networks (CNNs) are based on kernel sliding, convolution unrolling, and time multiplexing [6, 4]. At each time step, the input feature chunks and corresponding convolutional kernels are flattened as a one-dimensional vector and fed into the ONNs to perform vector dot-product. Another solution to solve this is to use *im2col* algorithm [131, 225], that transforms convolution to general matrix multiplication (GEMM). Convolutional kernels and input features are re-shaped as matrix-matrix multiplication, which can be directly mapped on ONNs. Such implementation is

inherently inefficient as overlapped convolutional patterns will create a huge amount of data redundancy in the unrolled feature maps. In this work, we proposed to achieve CNNs with a new ONN architecture equipped with learnable transformation structures. Figure 2.5 demonstrates our proposed optical MD-based CNN architecture featured by kernel sharing, learnable transformation, and augmented frequency-domain kernel techniques. Multi-channel input feature maps are encoded onto multiple wavelengths and input into the learnable frequency-domain transforms, then split into multiple branches through the fanout network for parallel multi-kernel processing. Frequency-domain convolution is performed in the MD-based kernel banks and the final results are transformed back to the real domain via the reversed transforms. Note that we do not include a detailed discussion on the pooling operations since they are not computationally-intensive parts in NNs. For example, optical comparators can be used to achieve max-pooling. Average pooling can be implemented by a fixed-weight convolution engine based on combiner-tree networks. Multiple layers can be cascaded through O-E-O conversion. The phase information loss during photo-detection can be fully modeled during training without harming the model expressivity, which is actually a competitive substitute for ReLU activation in the complex NN domain [201]. All of our experiments in later sections model this phase removal during training, which shows that this non-ideality induced by photo-detection does not cause any accuracy loss. We will introduce details of the principles of the designed photonic CNN in the following section.

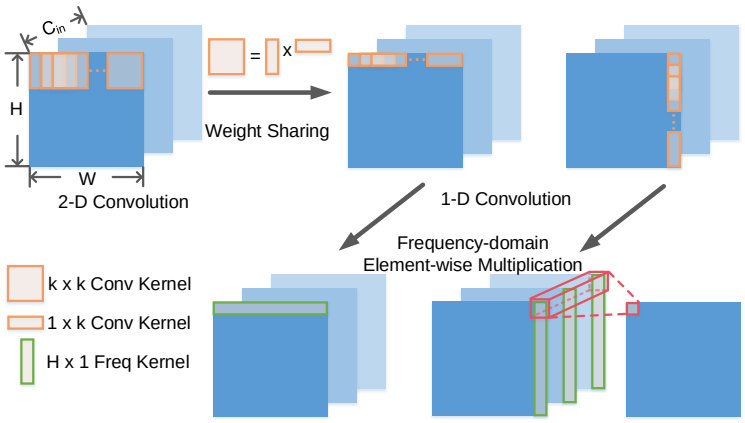


Figure 2.6: 2-D convolutional kernel decomposition using weight sharing and frequency-domain transformation.

2.2.3.2 Kernel Weight Sharing

Modern CNN architectures, e.g., inception architecture [185], adopt weight sharing to reduce the number of parameters in the convolutional layers. For example, a 5×5 2-D convolution involves 25 parameters. It can be replaced by two cascaded lightweight 1×5 and 5×1 convolutions, which only contain 10 unique variables. Such a strategy trains a low-rank convolutional kernel and can benefit its photonic implements as it can be directly applicable to 2-D PICs, which is visualized in Fig. 2.6.

2.2.3.3 Learnable Frequency-domain Convolution

Spatial domain convolution requires to slide the receptive field of convolutional kernels across the input features. This could induce hardware implementation difficulty and inefficiency as time multiplexing increases the latency and control complexity of photonic convolution. we solve this issue by

a parametrized frequency-domain convolution method. As mentioned before, we decompose the 2-D convolution as row-wise and column-wise 1-D convolutions through weight sharing. For brevity, we focus on the column-wise frequency-domain convolution in the following discussion. The same principle also applies to the row-wise convolution. The column-wise convolution can be formulated as

$$\boldsymbol{w} * \boldsymbol{x} = \mathcal{T}^{-1}(\mathcal{T}(\boldsymbol{w}; \boldsymbol{\phi}) \odot \mathcal{T}(\boldsymbol{x}; \boldsymbol{\phi}); \boldsymbol{\phi}), \tag{2.13}$$

where $\mathcal{T}(\cdot; \boldsymbol{\phi})$ is the learnable frequency-domain projection, and $\boldsymbol{\phi}$ represents the trainable parameters in it. This parametrized transformation enlarges the parameter space to compensate for the model expressiveness degradation induced by kernel weight sharing. Considering the learnable transform as a high-dimensional unitary rotation, it is not necessary to adopt an inverse transform pair to limit the exploration space. To enable the maximum learnability of our trainable transform structure, we relax the inverse transform to a reversed transform,

$$\boldsymbol{w} * \boldsymbol{x} = \mathcal{T}_r(\mathcal{T}(\boldsymbol{w}; \boldsymbol{\phi}) \odot \mathcal{T}(\boldsymbol{x}; \boldsymbol{\phi}); \boldsymbol{\phi}_r), \tag{2.14}$$

where $\mathcal{T}_r$ has a reversed butterfly structure but is not constrained to be the inverse of $\mathcal{T}$.

We now discuss how our proposed trainable transform structures can move beyond Fourier transform, thus enabling hardware-aware learnability. Fourier transform is a complex domain transformation that is mathematically

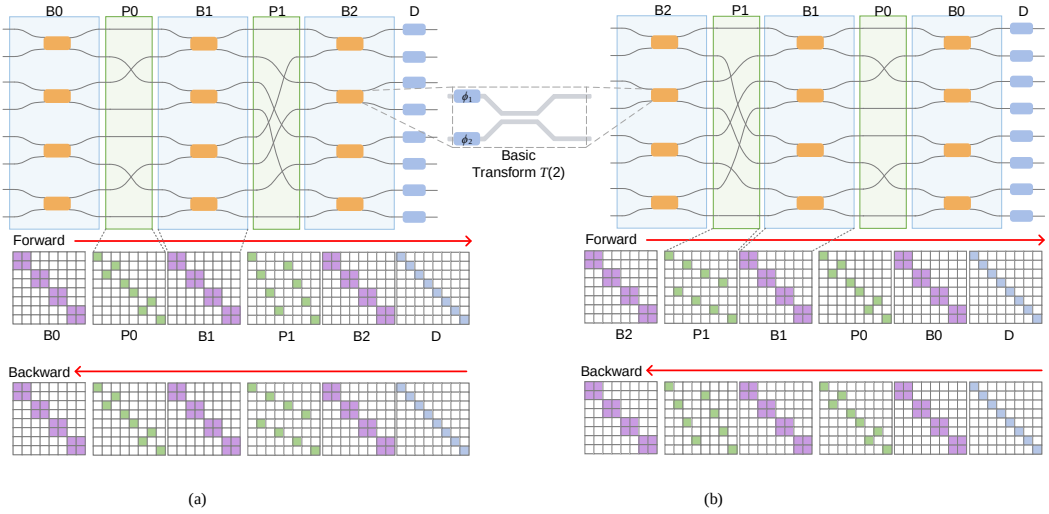


Figure 2.7: (a) The original learnable frequency-domain transformation structure. (b) The reversed learnable transformation structure.

designed for frequency component extraction. However, the Fourier transform is not necessary to be the best-performed transformation that can be used in CNNs. Other manually designed unitary transforms are also experimentally demonstrated to have a similar ability for signal integration and extraction [249]. Hence, we upgrade the fixed transformation structure to an adaptive structure where all phase shifters are trainable. As mentioned in the Section 2.2.2.3, phase shifters in the same segment of the waveguide can be merged into one phase shifter. Therefore, to avoid redundant trainable phase shifters, we re-design the learnable basic block, as shown in Fig. 2.7. For the original transformation, two phase shifters ϕ_1 and ϕ_2 are placed on the input port of the directional coupler. The transfer function of a learned basic block

can be formulated as,

$$\begin{aligned}\mathcal{T}(2) =& \frac{1}{\sqrt{2}}\begin{pmatrix}1 & j\\ j & 1\end{pmatrix}\begin{pmatrix}e^{j\phi_1} & 0\\ 0 & e^{j\phi_2}\end{pmatrix}\\ =& \frac{1}{\sqrt{2}}\begin{pmatrix}\cos\phi_1 + j\sin\phi_1 & -\sin\phi_1 + j\cos\phi_1\\ -\sin\phi_2 + j\cos\phi_2 & \cos\phi_2 + j\sin\phi_2\end{pmatrix}.\end{aligned} \tag{2.15}$$

In the reversed transformation structure, the basic block is the same as used in the original transforms since the inverse basic block requires a conjugate transposed transfer function which is not implementable with this basic block. Based on this basic block, we recursively build a trainable N-length transform with a butterfly structure, which can be described as $\log_2 N$ stages of projection, $\log_2 N - 1$ stages of permutation, and a final extra group of phase shifters. The original transformation, shown in Fig. 2.7(a), can be formulated as,

$$\mathcal{T}(N) = \mathcal{D}\ \mathcal{B}_{\log_2 N-1}(N) \prod_{i=0}^{\log_2 N-2} \mathcal{P}_i(N)\mathcal{B}_i(N), \tag{2.16}$$

where $\mathcal{B}_i(N)$ the i-th stage of butterfly projection, $\mathcal{P}_i(N)$ is the i-th stage signal permutation, and the diagonal matrix $\mathcal{D}$ represents the final extra column of phase shifters. The butterfly projection operator $\mathcal{B}(N)$ is a diagonal matrix with a series of $\mathcal{T}(2)$ as its diagonal sub-matrices,

$$\mathcal{B}(N) = \begin{pmatrix}\mathcal{T}_0(2) & 0 & \cdots & 0\\ 0 & \mathcal{T}_1(2) & \cdots & 0\\ \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & \cdots & \mathcal{T}_{N/2-1}(2)\end{pmatrix} \tag{2.17}$$

The index permutation operator $\mathcal{P}_i(N)$ can be expressed as a size-N identity matrix with reordered rows. As shown in $\mathcal{P}_0$ and $\mathcal{P}_1$ in Fig. 2.7, the green

entries represent 1, and other blank entries represent 0. Note that the permutation operators in the reversed structure are simply the reversed counterparts in the original structure, i.e., $\mathcal{P}_{i,ori}(N) = \mathcal{P}^T_{i,rev}(N)$. The reversed learnable transformation, shown in Fig. 2.7(b), is designed to have reversed butterfly structure which can be derived as follows,

$$\mathcal{T}_r(N) = \mathcal{D}\left(\prod_{i=0}^{\log_2 N-2} \mathcal{B}_{r,i}(N)\mathcal{P}_{r,i}(N)\right)\mathcal{B}_{r,\log_2 N-1}(N). \tag{2.18}$$

Note that the reversed transform is not guaranteed to be inverse to the original transform, which requires particular phase configurations discussed later.

Compared with its MZI-based counterparts, this trainable butterfly transformation structure has a constrained projection capability as only a limited set of unitary matrices can be implemented by it [105, 49]. As shown in unitary group parametrization, a full N-dimensional unitary space $\boldsymbol{U}(N)$ has $N(N-1)/2$ independent parameters, while the butterfly structure substitutes part of parametrized unitary matrices with fixed permutation operators. Hence, based on full two-dimensional unitary matrices $\boldsymbol{U}(2)$, the butterfly structure has $2N\log_2 N$ independent parameters. Our proposed learnable block $\mathcal{T}(2)$ is a reduced version of $\boldsymbol{U}(2)$, as it only covers half of the full 2-D planar rotation space. The pruned transform space $\mathcal{T}^*(2)$ can be expressed as the conjugate transpose of $\mathcal{T}(2)$, which is not implementable without waveguide crossings.

$$\mathcal{T}^*(2) = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & -j \\ -j & 0 \end{pmatrix}\begin{pmatrix} 1 & j \\ j & 1 \end{pmatrix}\begin{pmatrix} e^{j\phi_1} & 0 \\ 0 & e^{j\phi_2} \end{pmatrix} \tag{2.19}$$

Equivalently, our learnable transformation structure has $N \log_2 N$ free parameters.

2.2.3.4 Microdisk-based Augmented Kernels

To enable highly-parallel CNN architecture with reinforced model expressiveness, we propose MD-based augmented convolutional kernels with multi-level parallelism across input features, input channels, and output channels.

In our design, each 2-D convolutional layer consists of two cascaded 1-D frequency-domain convolutions along columns and rows. We will focus on the column-wise convolution, and the same architecture applies to its row-wise counterpart with an extra matrix transposition operation. We denote the input feature map as $\boldsymbol{I} \in \mathbb{R}^{C_{in} \times H \times W}$, which C_{in}, H, W represent the number of input channel, spatial height, and spatial width, respectively. At time step t, The corresponding column $\boldsymbol{I}_{:,t,:} \in \mathbb{R}^{C_{in} \times H \times 1}$ will be input into the photonic CNN. Different input channels are encoded by different wavelengths $\{\lambda_0, \lambda_1, \cdots, \lambda_{C_{in}-1}\}$. Through the wide-band learnable transformation structure, we obtain the frequency-domain features $\mathcal{T}(\boldsymbol{I}_{:,t,:}; \phi)$. This stage enables parallel transformation across the input channels. Then the optical signals carrying those features will be split into C_{out} planes for data reuse. Such a multi-dimensional ONN design can be supported by state-of-the-art integration technology with multiple photonic chiplets [138]. In the MD-based convolution stage, $C_{out} \times C_{in} \times H$ all-pass MDs are used to implement the frequency-domain kernels $\boldsymbol{W} \in \mathbb{R}^{C_{out} \times C_{in} \times H}$. Given that the working princi-

ple of MD is primarily optical signal magnitude modulation, our augmented kernels are trainable only in the magnitude space without phase modulation. Each convolutional core is designed to perform the convolution of one output channel. This MD-based convolution is different from the previous EM stage consisting of attenuators and phase shifters. First, all pass MDs can only perform configurable magnitude modulation of the input signals with fixed phase responses, which means the augmented kernels will not expand over the entire complex space. Here we give the transfer function of an MD,

$$\begin{aligned} I_{out} &= W \cdot I_{in} \\ \cos\theta &= \frac{a^2 + r^2 - W(1 + r^2 a^2)}{2(1 - W)ar} \\ \phi_{out} &= \pi + \theta + \arctan \frac{r\sin\theta - 2r^2 a \sin\theta\cos\theta + ra^2\sin\theta}{(a - r\sin\theta)(1 - ra\cos\theta)}, \end{aligned} \tag{2.20}$$

where I_{in} is the magnitude of the input light, I_{out}, ϕ_{out} are the magnitude and phase of the output optical signal, θ, a, r are the phase, self-coupling coefficient, and coupling loss factor of an MD, respectively. W is the transmitivity of the MD which corresponds to the trained augmented kernel weight. Typically, parameters a and r are very close to 1. Our proposed architecture enables another level of parallelism across output channels. Given that different convolutional kernels share the same input features, multiple MD convolution cores and reversed transform structures will share one original transform structure for hardware reuse and highly-parallel convolution.

A higher modeling capacity is enabled by our augmented kernel technique. Instead of training spatial kernels $\boldsymbol{w}$, we explicitly train the latent

weights $\boldsymbol{W}$ in the frequency domain without performing $\mathcal{T}(\boldsymbol{w};\boldsymbol{\phi})$ during training. The augmented latent weights $\boldsymbol{W}$ will not meet the conjugate symmetry constraint as its spatial-domain counterparts are not real-valued. Hence, this enables a potentially infinite solution space in the spatial kernel space with various kernel sizes and shapes.

We briefly discuss the scalability of this WDM-based highly-parallel architecture. WDM plays an important role in the high parallelism of our proposed frequency-domain photonic CNN. Currently, the widely acknowledged maximum number of wavelength in the single-mode dense-WDM (DWDM) is over 200 [191, 54, 233]. When mode-division multiplexing is further considered, higher parallelism can be supported given the current technology. This means in our architecture has enough parallelism to support most modern CNN architectures.

2.2.3.5 Discussion: Exploring Inverse Transform Pairs in Constrained Unitary Space

In manually designed frequency-domain convolution algorithms, domain transformation will be designed to be inverse, e.g., FFT and IFFT. This implies an inverse constraint between two mutually-reversed transform structures $\mathcal{T}$ and $\mathcal{T}_r$. To be able to realize trainable inverse transform pairs, we add unitary constraints to our learnable transform structures,

$$\mathcal{T}_r(\cdot,\boldsymbol{\phi}_r) = \mathcal{T}^{-1}(\cdot;\boldsymbol{\phi}). \tag{2.21}$$

Inverse constraints typically can be addressed by adding a regularization term in training,

$$\mathcal{L}_{inv} = \|\boldsymbol{U}_r\boldsymbol{U} - \boldsymbol{I}\|_2. \tag{2.22}$$

However, this requires explicit transfer matrices of $\mathcal{T}$ and $\mathcal{T}_r$ to compute this regularization term [34], which is memory-intensive and computationally expensive as indicated by Eq. (2.17), Eq. (2.18). We propose an efficient regularization method to exert the inverse constraint.

$$\mathcal{L}_{inv} = \|\mathcal{T}_r(\mathcal{T}(\boldsymbol{e})) - \boldsymbol{e}\|_2, \quad \boldsymbol{e} \in \mathbb{C}^N, \tag{2.23}$$

where $\boldsymbol{e}$ is the orthonormal bases of N-dimensional complex space. Notice that if $\mathcal{T}_r(\mathcal{T}(\boldsymbol{e})) = \boldsymbol{e}$, then for any $\boldsymbol{x} = \boldsymbol{\alpha}^T\boldsymbol{e}$ the following statement holds,

$$\mathcal{T}_r(\mathcal{T}(\boldsymbol{x})) = \mathcal{T}_r(\mathcal{T}(\boldsymbol{\alpha}^{\boldsymbol{T}}\boldsymbol{e})) = \boldsymbol{\alpha}^{\boldsymbol{T}}\mathcal{T}_r(\mathcal{T}(\boldsymbol{e})) = \boldsymbol{x}. \tag{2.24}$$

Thus transforms $\mathcal{T}$ and $\mathcal{T}_r$ are inverse transforms once the regularization loss reaches 0. This surrogate method reduce the computation complexity from $\mathcal{O}(N^2 \log_2 N)$ in Eq. (2.16) to $\mathcal{O}(N \log_2 N)$, where diagonal matrix multiplication with $\mathcal{B}(N)$ is simplified by 2×2 sub-matrix multiplication with $\mathcal{T}(2)$.

Using our proposed inverse pair regularization method, we show that our trainable transform $\mathcal{T}$ can efficiently learn Fourier transform by setting $\mathcal{T}_r$ as OIFFT. Figure 2.8 demonstrates that the trainable transform will quickly converge to the theoretical OFFT as the mean square error between trained phase settings and target phase shifter settings reduces to 0 when the loss converges.

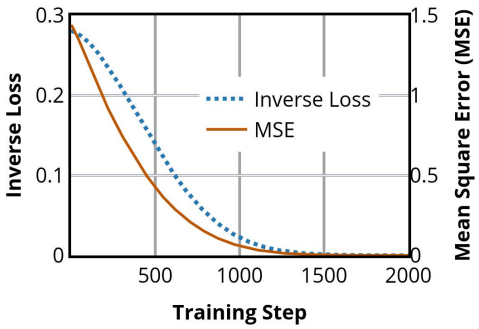


Figure 2.8: Training curve of inverse loss $\mathcal{L}_{inv}$ and mean square error between trained phase configurations and theoretical 4-point OFFT settings.

2.2.3.6 Discussion: Hardware-aware Pruning for Trainable Transforms

In this section, we demonstrate that our proposed trainable transform has excellent compatibility with hardware-aware pruning techniques. Compared to the fixed manual design of frequency-domain transforms, e.g., OFFT, we can further boost the hardware efficiency by eliminating a subset of phase shifter columns inside the trainable transforms. With this fine-grained structured pruning, we can improve the area, power, and noise robustness since phase shifters contribute to nearly 50% of the total area and the majority of the total power and noise. We adopt a phase-wrapping Group Lasso regularization similar to Eq. (2.2) together with an incremental pruning technique to slim the trainable transforms targeted at lower area cost and lower power consumption. The proposed phase-wrapping Group Lasso (PhaseGL) is for-

mulated as,

$$\begin{aligned} \boldsymbol{L}_{PhaseGL} =& \sum_{g=0}^{G} \sqrt{1/p_g} \|\phi_g - \phi_g^*\|_2, \\ \phi_{g,i}^* =& \begin{cases} 0, & \phi_{g,i} \in [0, \pi), 0 \leq i < p_g \\ 2\pi, & \phi_{g,i} \in [\pi, 2\pi), 0 \leq i < p_g, \end{cases} \end{aligned} \tag{2.25}$$

where ϕ_g is a column of phase shifters, and this regularization term encourages phases towards their corresponding prunable targets ϕ_g^*. G is the total columns of phase shifters, which is $(\log_2 N + 1)$ for a length-N transform. Once the group lasso of a column falls below a threshold $T_{\mathcal{T}}$, the entire column of phase shifters are pruned. The ratio of pruned columns to all phase shifter columns is called transform sparsity ($\mathcal{T}$ sparsity), defined as,

$$s_{\mathcal{T}} = \frac{|\{\phi_g | \sqrt{1/p_g} \|\phi_g - \phi_g^*\| < T_{\mathcal{T}}\}|}{G}.$$

Our proposed regularization and pruning strategy improves area cost as an entire column of phase shifters is pruned to save chip area in the actual layout. Furthermore, power consumption and noise robustness can also be improved as a majority of power consumption and noises are from trainable transform structures [252, 74, 253].

2.2.3.7 Discussion: Hardware Cost of the Proposed MD-based Photonic CNN

We give a summary of the hardware component usage of the proposed MD-based photonic CNN architecture in Table. 2.2. Our architecture shares the original transform among multiple kernels to save area. Our proposed pruning technique can regularly sparsify the transform structures for further

Table 2.2: Hardware cost summary on the proposed MD-based photonic CNN architecture. The input feature map is of size $H \times W \times C_{in}$, the number of output channels is C_{out}, and the sparsity of the learnable transforms is $s_{\mathcal{T}} \in [0, 1]$. For simplicity, we assume $H = W$, which is a widely used configuration for most CNNs. Given the ultra-compact footprint of an MD, e.g., 5×5 μm^2 [195], we count 100 MDs as one DC in the area estimation. The row-wise and column-wise convolutions are both counted in this table.

Structure	Hardware Cost
$\mathcal{T}$	$H \log_2 H$ DCs + $2s_{\mathcal{T}}H(1 + \log_2 H)$ PSs
Kernel	$2HC_{in}C_{out}$ MDs$\approx \frac{H}{50}C_{in}C_{out}$ DCs
$\mathcal{T}_r$	$H \log_2 HC_{out}$ DCs + $2s_{\mathcal{T}}H(1 + \log_2 H)C_{out}$ PSs
Total	$\approx H(\log_2 H + \frac{C_{in}}{50})C_{out}$ DCs + $2s_{\mathcal{T}}H(1 + \log_2 H)C_{out}$ PSs

area reduction. The MD-based convolution stage is very compact since the footprint of an MD is two-order-of-magnitude smaller than a DC. In contrast, the SVD-based ONN costs $H(C_{out}^2+C_{in}^2\times K^4)$ DCs and $H(C_{out}^2/2+C_{in}^2\times K^4/2)$ PSs to achieve the same latency with our architecture, i.e., H forwards to finish a convolutional layer, where K is the spatial kernel size. For example, if we set H=64, C_{in}=C_{out}=32, K=3, $s_{\mathcal{T}}$=0.5, our architecture uses >370× fewer DCs and >180× fewer PSs than the single-wavelength SVD-based ONN. If SVD-based ONNs also use WDM techniques for higher parallelism with the same number of wavelengths as ours, i.e., 32, we still outperform theirs by 11.6× fewer DCs and 5.6× fewer PSs. Hence, our frequency-domain CNN architecture outperforms previous MZI-ONNs with higher computational efficiency and better scalability by a large margin.

2.2.4 Experimental Results

We conduct numerical simulations for functionality validation and evaluate our proposed architecture on the hand-written digit recognition dataset (MNIST) [115] with various network configurations. Quantitative evaluation shows that our proposed architecture outperforms the SVD-based and TΣU-based ONN architectures in terms of area cost without any accuracy degradation. We further evaluate our proposed MD-based photonic CNN architecture and demonstrate its superior power reduction and robustness improvement on MNIST [115] and FashionMNIST [222] dataset.

2.2.4.1 Simulation Validation

To validate the functionality of our proposed architecture, we conduct optical simulations on a 4×4 circulant matrix-vector multiplication module using Lumerical INTERCONNECT tools. First, we encode a 4×4 identity weight matrix into our architecture and input 4 parallel optical signals to validate its functionality. For brevity, we plot several different representative cases in Fig. 2.9(a). It shows that our designed architecture can correctly realize identity projection. Further, we randomly generate a length-4 real-valued weight vector $\boldsymbol{w} = (0.2, -0.1, 0.24, -0.15)$ to represent a circulant matrix, and encode $\mathcal{F}(\boldsymbol{w}) = (0.19e^{0j}, 0.064e^{-2.246j}, 0.69e^{0j}, 0.064e^{2.246j})$ into attenuators and phase shifters in the EM stage. The simulation results in Fig. 2.9(b) show good fidelity ($< 1.2\%$ maximum relative error) to the ground truth results.

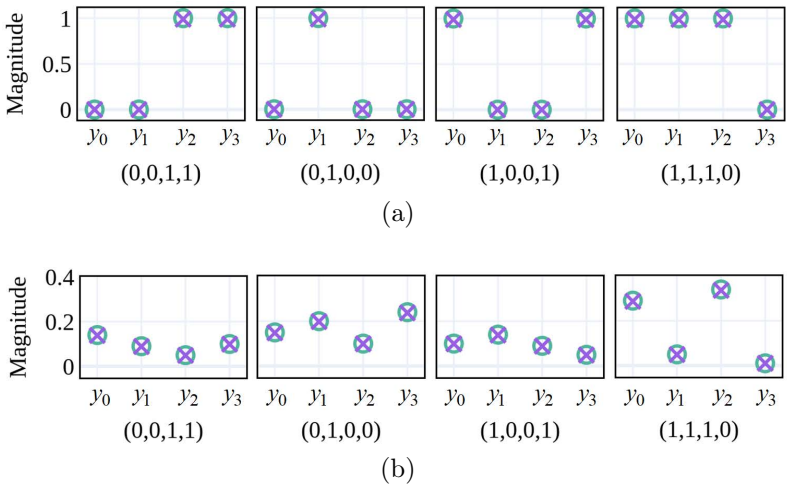


Figure 2.9: (a) Simulated output intensities (crosses) and ground truth (circles) of a 4×4 identity circulant matrix-vector multiplication. (b) Simulated output intensities (crosses) and ground truth (circles) of a 4×4 circulant matrix-vector multiplication, with $\boldsymbol{w}$=(0.2,-0.1,0.24,-0.15). E.g., (0,0,1,1) is the input signal.

Table 2.3: Optical component sizes used in the area estimation.

Optical Component	Length (μm)	Width (μm)
3-dB Directional Coupler [171]	54.4	40.3
Thermo-optic Phase Shifter [84]	60.16	0.50
2-to-1 Optical Combiner [172]	20.00	3.65
Waveguide Crossing[245]	5.9	5.9

2.2.4.2 Comparison Experiments on FFT-based ONNs

To evaluate our proposed ONN architecture, we conduct a comparison experiment on a machine learning dataset MNIST [115], and compare the hardware utilization, model expressivity among four architectures: 1) SVD-based architecture [171]; 2) TΣU-based architecture [253]; 3) Ours without pruning; 4) Ours with pruning.

We implement the proposed architecture with different configurations

Table 2.4: Comparison of inference accuracy and hardware utilization on MNIST dataset with different configurations. For example, configuration (28×28)-1024(8)-10(2) indicates a 2-layer neural network, where the first layer has 784 input channels, 1024 output channels with size-8 circulant matrices, and so on.

	Network Configurations	Block Sparsity	#Param	Accuracy	#DC	#PS	Area (cm^2)
Model 1	SVD[171]: (28×28)-400-10	0.00	318 K	98.49%	934 K	467 K	20.62
	TΣU[253]: (28×28)-400-10	0.00	318 K	98.49%	777 K	388 K	17.15
	Ours w/o Prune: (28×28)-1024(8)-10(2)	0.00	105 K	98.32%	412 K	718 K	9.33
	Ours w/ Prune: (28×28)-1024(8)-10(2)	0.40	63 K	98.26%	244 K	425 K	5.53
Model 2	SVD[171]: (14×14)-70-10	0.00	14 K	96.93%	48 K	24 K	1.07
	TΣU[253]: (14×14)-70-10	0.00	14 K	96.93%	44 K	22 K	0.97
	Ours w/o Prune: (14×14)-256(4)-10(2)	0.00	14 K	96.93%	40 K	67 K	0.90
	Ours w/ Prune: (14×14)-256(4)-10(2)	0.45	8 K	96.91%	22 K	36 K	0.49
Model 3	SVD[171]: (28×28)-400-128-10	0.00	366 K	98.58%	967 K	483 K	21.35
	TΣU[253]: (28×28)-400-128-10	0.00	366 K	98.58%	794 K	396 K	17.52
	Ours w/o Prune: (28×28)-1024(8)-128(4)-10(2)	0.00	134 K	98.53%	501 K	868 K	11.34
	Ours w/ Prune: (28×28)-1024(8)-128(4)-10(2)	0.39	81 K	98.43%	289 K	517 K	6.77
Model 4	SVD[171]: (14×14)-160-160-10	0.00	59 K	97.67%	141 K	70 K	3.10
	TΣU[253]: (14×14)-160-160-10	0.00	59 K	97.67%	91 K	45 K	2.00
	Ours w/o Prune: (14×14)-256(4)-256(8)-10(2)	0.00	22 K	97.67%	73 K	123 K	1.64
	Ours w/ Prune: (14×14)-256(4)-256(8)-10(2)	0.37	14 K	97.52%	47 K	79 K	1.05

in PyTorch and test the inference accuracy on a machine with an Intel Core i9-7900X CPU and an NVIDIA TitanXp GPU. We set λ to 0.3 for the Group Lasso regularization term, initialize all trainable weights with a Kaiming-Normal initializer [85], adopt the Adam optimizer [109] with initial learning rate=1×10^{-3} and a step-wise exponential-decay learning rate schedule with decay rate=0.9. We use the ideal rectified linear units (ReLU) activation function as nonlinearity. All NN models are trained for 40 epochs with a mini-batch size of 32 till fully converged. The structured sparsity for our proposed FFT-based MLP is defined as the percentage of pruned parameters in all parameters, i.e., $|\{w|\|\boldsymbol{w}_{ij}\|_2 < T\}|/|\boldsymbol{w}|$. We call it block sparsity.

For a fair comparison, all architectures are trained with the same hyper-parameters and have similar test accuracy in each experiment configuration. To estimate the component utilization and area cost, we adopt exactly the

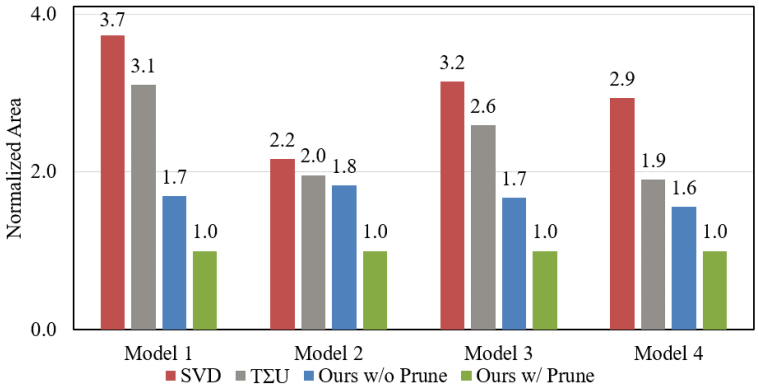


Figure 2.10: Normalized area comparison with different model configurations. *Model 1-4* refer to Table 2.4. *SVD* refers to [171] and $T\Sigma U$ refers to [253].

same type of photonic devices in all architectures, as listed in Table 2.3, and accumulate the area of each optical component for approximation. Placement or routing information is not considered in our estimation.

In Table 2.4, the first column indicates different neural network configurations. In the TΣU-based architecture, the total number of MZIs used to implement an $m \times n$ weight matrix is bounded by $n(n+1)/2$. Among various network configurations, our proposed architecture outperforms the SVD-based architecture and the TΣU-based architecture with lower optical component utilization and better area cost. We normalize all areas to our architecture with pruning applied and show the normalized area comparison in Fig. 2.10. Consistent with analytical formulations in Section 2.2.2.3, the experimental results show that, as the difference between input and output channels for each layer in the original MLPs gets larger, our proposed architecture can save a larger proportion of optical components.

Furthermore, ablation experiments on our structured pruning method validate the effectiveness of the proposed two-phase training flow. It can save an extra 30-50% optical components with negligible model expressivity loss.

2.2.4.3 Comparison Among Different Trainable Transform Settings

As mentioned in previous sections, we extend our ONN architecture to MD-based CNNs with trainable frequency-domain transforms. We will demonstrate several experimental evaluations on our proposed MD-based CNN architecture.

First, we discuss how different transform settings impact the CNN performance. Recall that each 2-D frequency-domain convolution involves total four trainable transforms, denoted as $\mathcal{T}_{row}, \mathcal{T}_{row,r}, \mathcal{T}_{col}, \mathcal{T}_{col,r}$. Therefore we evaluate the performance of four different transform settings on MNIST dataset: (1) four transforms are trained independently (`AllFree`); (2) Column-wise and row-wise convolutions share the same transform as $\mathcal{T}_{row} = \mathcal{T}_{col}, \mathcal{T}_{row,r} = \mathcal{T}_{col,r}$ (`Shared`); (3) Reversed transforms are constrained to be close to the inverse transform as $\mathcal{T}_{row,r} \approx \mathcal{T}_{row}^{-1}, \mathcal{T}_{col,r} \approx \mathcal{T}_{col}^{-1}$ (`Inverse`); (4) Transforms are shared between column-wise and row-wise convolutions and the inverse constraints are applied (`InvShared`). Table 2.5 shows the comparison results.

Based on the results, we observe that the inverse constraint and shared transform produce no benefits in terms of inference accuracy. Training the original and reversed transforms across row-wise and column-wise convolutions

Table 2.5: Accuracy comparison among four trainable transform settings. The model is 16×16-C16-BN-MaxPool5-F32-F10.

Settings	`AllFree`	`Shared`	`Inverse`	`InvShared`
Test Accuracy	96.88%	96.13%	96.41%	96.40%

Table 2.6: Transform sparsity ($\mathcal{T}$ sparsity) and power consumption comparison among optical FFT and our trainable transform with hardware-aware pruning on MNIST and FashionMNIST dataset. $\mathcal{T}$ sparsity represents how many columns of phase shifters are pruned in our trainable frequency-domain transforms. The power consumption assumes maximum parallelism across output channels, thus 1 original transform and C_{out} reversed transforms are counted for each layer. For the MNIST dataset, we adopt the ONN configuration as 16×16-C16-BN-ReLU-MaxPool5-F32-ReLU-F10, and for the FashionMNIST dataset we set the ONN configuration as 16×16-C24-BN-ReLU-MaxPool6-F64-ReLU-F10. The power consumption is estimated by the sum of phase shifts given that the phase shift is proportional to the thermal tuning power, i.e., $\phi \propto v^2$. Other power consumption sources, e.g., insertion loss, are not considered for simplicity.

Dataset	Transform	OFFT	Trainable (Pruned)
MNIST [115]	$\mathcal{T}$ Sparsity	0%	88.2%
	Normalized Power	100%	18.4%
FashionMNIST [222]	$\mathcal{T}$ Sparsity	0%	88.4%
	Normalized Power	100%	15.5%

independently offers the best results. Thus, we will use `AllFree` transform settings for our experiments.

2.2.4.4 Comparison with Hardware-Aware Transform Pruning

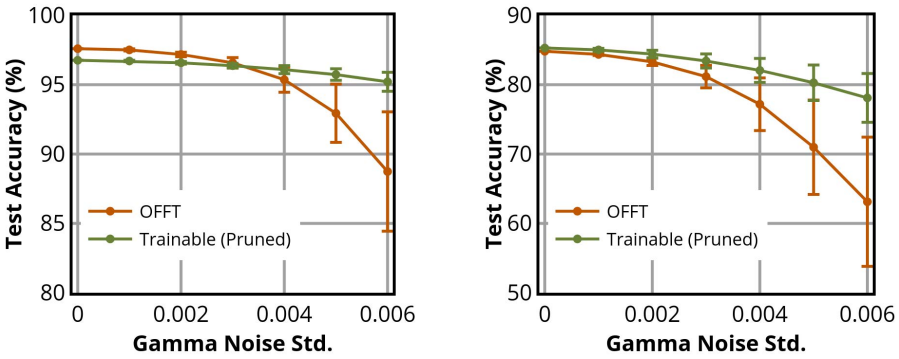
To jointly optimize classification accuracy and hardware cost in terms of area, power, and robustness, we perform hardware-aware pruning assisted by phase-wrapping Group Lasso regularization to our proposed trainable transforms. The weight for $L_{PhaseGL}$ is 0.05, and we set 10 epochs for the first pretraining phase and 40 epochs for incremental structured pruning.

Table 2.7: Comparison of block sparsity, frequency-domain transform ($\mathcal{T}$) sparsity, normalized power consumption, and estimated area (cm^2) among 1) SVD-based ONN, 2)$T\Sigma U$-based ONN, 3) optical FFT, 4) our trainable transform without pruning transforms, and 5) our trainable transform with hardware-aware pruning on MNIST dataset. SVD-based and $T\Sigma U$-based ONN configuration is $28\times28-400-10$, and ours is $28\times28-1024(8)-10(2)$. All ONNs have a similar inference accuracy with a 0.5% accuracy discrepancy among all architectures. Block sparsity is for pruned circulant blocks. $\mathcal{T}$ sparsity is for pruned trainable frequency-domain transforms. The power consumption is normalized to SVD-based ONN, which is estimated by the sum of all phase shifts given that the phase shift is proportional to the thermal tuning power, i.e., $\phi \propto v^2$.

Architecture	Block Sparsity	$\mathcal{T}$ Sparsity	Power	Area (cm^2)
SVD-based [171]	-	-	100%	20.62
$T\Sigma U$-based [253]	-	-	83.1%	17.15
Ours-OFFT [70]	0.40	0.00	98.9%	5.53
Ours-Trainable	0.71	0.00	79.9%	2.54
Ours-Trainable	0.66	0.96	9.9%	2.99

Power Consumption Evaluation We calculate the energy cost by summing all phase shifts as they are proportional to power consumption, and show the energy saved by our pruned transforms in Table 2.6. We also evaluate the power consumption by applying pruned trainable transform in our block-circulant matrix based MLP architecture in Table 2.7. Therefore, our energy-saving and area-efficient ONN architecture is more suitable for resource-constrained applications, e.g., edge computing and online learning tasks [72, 99].

Variation-Robustness Evaluation To evaluate the noise-robustness of the frequency-domain transform, we inject device-level variations into phase shifters to introduce phase programming errors and demonstrate the accuracy and its variance under different noise intensities σ on MNIST and FashionM-



(a) (b)

Figure 2.11: Robustness comparison among OFFT and pruned trainable transform on MNIST and FashionMNIST dataset. Error bar is drawn to show the $\pm 1\sigma$ accuracy variance from 20 runs. (a) For MNIST dataset, we adopt the ONN configuration as 16×16-C16-BN-ReLU-MaxPool5-F32-ReLU-F10. (b) For FashionMNIST dataset we set the ONN configuration as 16×16-C24-BN-ReLU-MaxPool6-F64-ReLU-F10.

NIST dataset. Specifically, we inject Gaussian noise $\Delta\gamma \sim \mathcal{N}(0, \sigma^2)$ into the γ coefficient of each phase shifter to perturb its phase response $\phi_n = (\gamma + \Delta\gamma)v^2$, where γ is calculated by the voltage that can produce π phase shift as $\gamma = \pi/v_\pi^2$ and we adopt 4.36V as the typical value of v_π [171, 74]. Figure 2.11 shows that $\sim 80\%$ structured sparsity can be achieved by our phase-wrapping pruning method, and our pruned trainable transform outperforms the OFFT structure with over 80% power reduction and much better robustness under various noise intensities.

We also evaluate the robustness on our circulant-matrix-based MLP architecture. Our FFT-based MLP and trainable transform-based architecture show superior robustness with over 97% accuracy on MNIST due to their structured sparsity and blocking design, while the SVD-based ONN drops

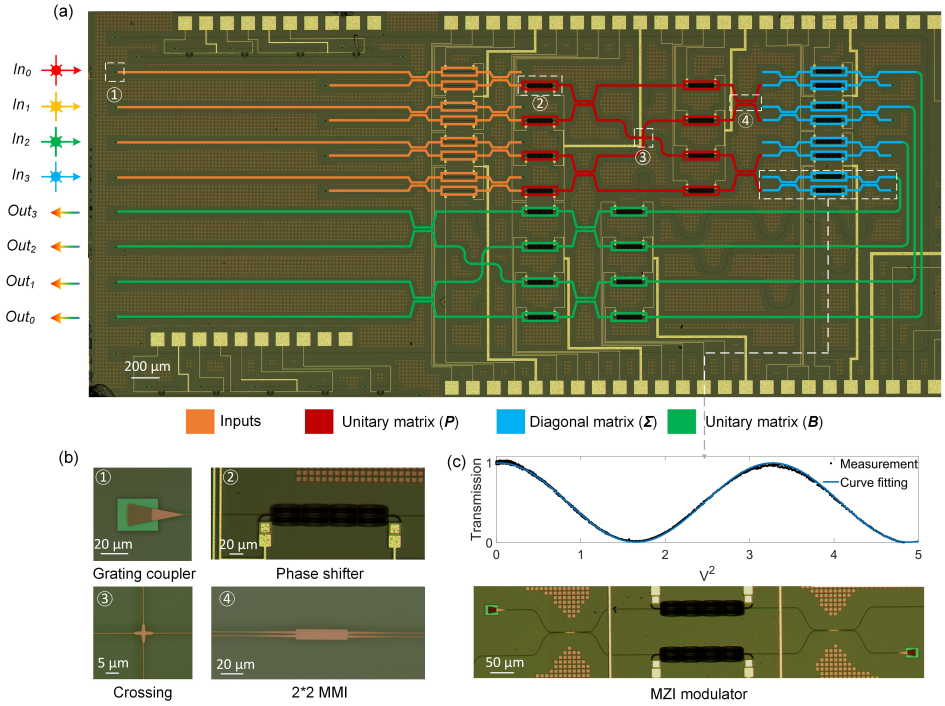


Figure 2.12: Schematic of the butterfly-style silicon photonic-electronic neural chip. The micrograph of the neural chip is shown in (a). The input optical beams with different wavelengths are shown in different colors. The necessary optical components are highlighted in (b). (c) shows the schematic and the normalized transmission curve of an MZI attenuator in the diagonal matrix unit (Σ unit). Only the attenuators in Σ are programmed in training.

below 90% due to severe error accumulation.

2.2.5 Experimental Demonstration with Butterfly-style Photonic Neural Chip Tape-out

We experimentally demonstrate the practicality of the FFT-ONN architecture using our butterfly-style photonic-electronic neural chip (BPNC), taped out at Advanced Micro Foundry, capable of implementing 4×4 matrix

multiplication. Since the butterfly photonic meshes B and P are fixed, and only the diagonal Σ units are trained, we refer to the implemented neural network as optical subspace neural network (OSNN) since a subspace of matrices can be expressed. The schematic of the fabricated BPNC is shown in Fig 2.12, with the close-ups of its components, such as phase shifters, 50-50 directional couplers, and crossings. The unitary matrix units B and P are marked in red/green. The active phase shifters in these regions support enough flexibility to realize different unitary transforms but note that they are not optimized as trainable parameters during ONN training. The diagonal matrix unit (Σ unit) is built using an array of MZI attenuators for magnitude and phase control.

The schematic of the testing setup is shown in Fig. 2.13. Continuous-wave (CW) light of different wavelengths is coupled in different input grating couplers separately. The input modulators and phase shifters of the BPNC are programmed by a 40-channel digital-to-analog converter (DAC). Off-chip photodetector arrays will receive the output signals, which will subsequently be read using oscilloscopes. A microcontroller (Raspberry Pi 4) is used to write electrical signals to the DAC and read the output signals from oscilloscopes. The measurement data are processed by computers to train and implement the DNN model.

We construct a 2-layer CNN (Fig. 2.15(a)) with our BPNC and benchmark its performance on a handwritten digits classification dataset MNIST [115]. We use MVM operations to implement CNNs with a widely-applied tensor un-

rolling method (im2col). Large-size tensor operations are partitioned into 4×4 blocks and mapped onto our BPNC. The partial product accumulation and non-linear activation functions, e.g., ReLU, are simulated on computers.

To boost the accuracy and make our BPNC tolerate various physical

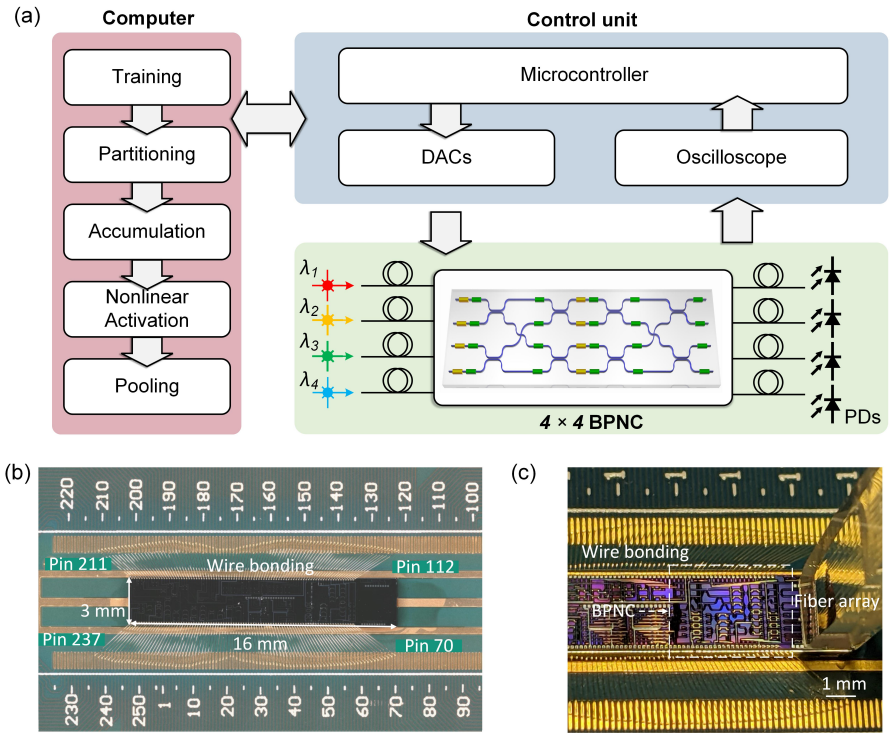


Figure 2.13: Experimental setup of OSNN. (a) Schematic of our OSNN test flow. The entire MVM is first partitioned into multiple 4×4 blocks, and each block is implemented optically on a butterfly-style photonic-electronic neural chip (BPNC). (b) shows the wire-bonded photonic chip and its starting/ending electrical pin numbers, while (c) is the photography of the chip testing setup. The parameters and the input signals are programmed by a multi-channel digital-to-analog converter (DAC), while the output signals are read by the oscilloscope. Both the oscilloscope and the DAC are controlled by a microcontroller. The MVM results are provided to the computer for data processing in order to train and deploy the DNN.

variations and noises, we adopt an AI-assisted variation-aware training flow, as shown in Fig. 2.14. The real chip responses are measured and used to train the differentiable PIC estimator (DPE) to model the behavior of the chip. Then, this accurate chip model is integrated into our variation-aware training flow to boost the deployment robustness.

The confusion matrix depicting the handwritten digit recognition prediction results is shown in Fig. 2.15(b). Figure 2.15(e) shows the output distribution of different handwriting digit inputs. Our experimental results show that when the voltage control resolution is set to 3-bit (8 attenuation levels

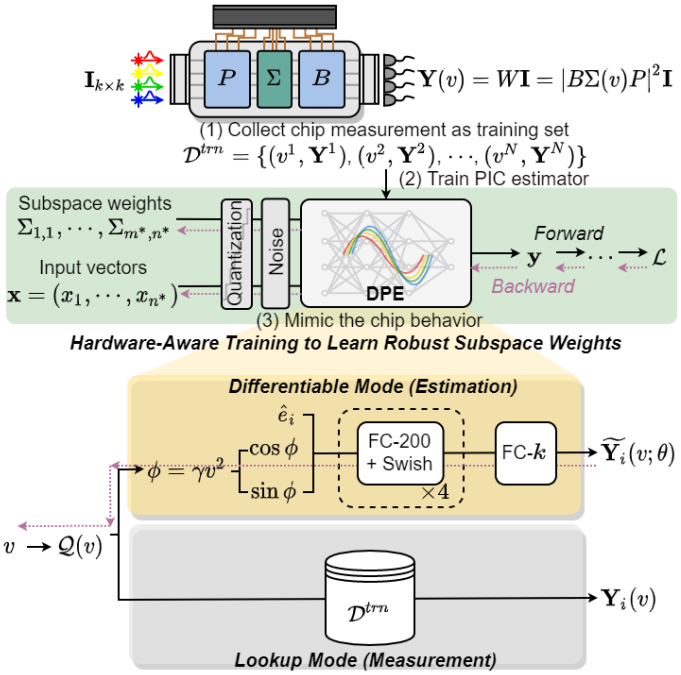


Figure 2.14: Proposed hardware-aware training framework where the differentiable PIC estimator learns the real chip's behavior and guides the OSNN weights toward a robust subspace.

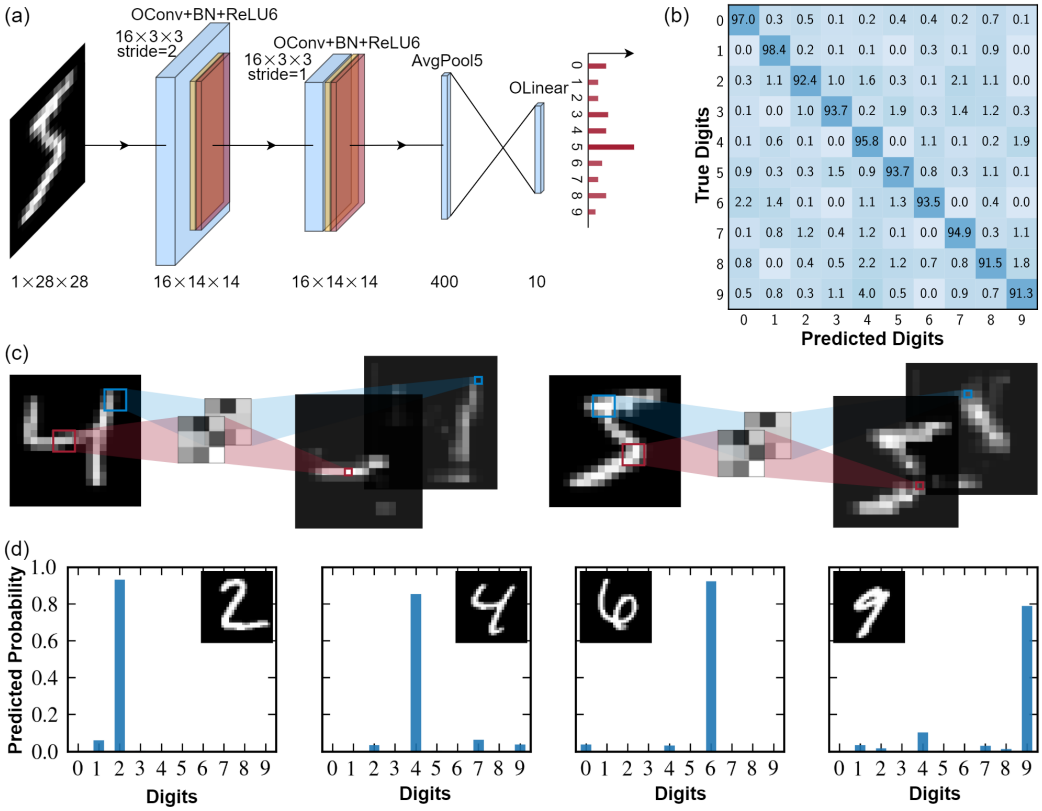


Figure 2.15: Experimental data of digit recognition with the OSNN. (a) Structure of the CNN, the convolution is realized by OSNN with the im2col approach. The first convolutional layer has one input channel and 16 output channels with a stride of 2. The subsequent convolutional layer has 16 input/output channels with a stride of 1, and the size of the convolutional kernel is 3×3. After adaptive average pooling, we have 5×5×16=400 hidden features, followed by a linear classifier with 10 outputs. (b) The confusion matrix of the trained OSNN on MNIST, showing a measured accuracy of 94.16%. (c) Experimental results of convolving two input images with convolution kernels of size 3×3 in our OSNN. (d) The predicted probability distribution of our OSNN on four selected test digits in the MNIST dataset.

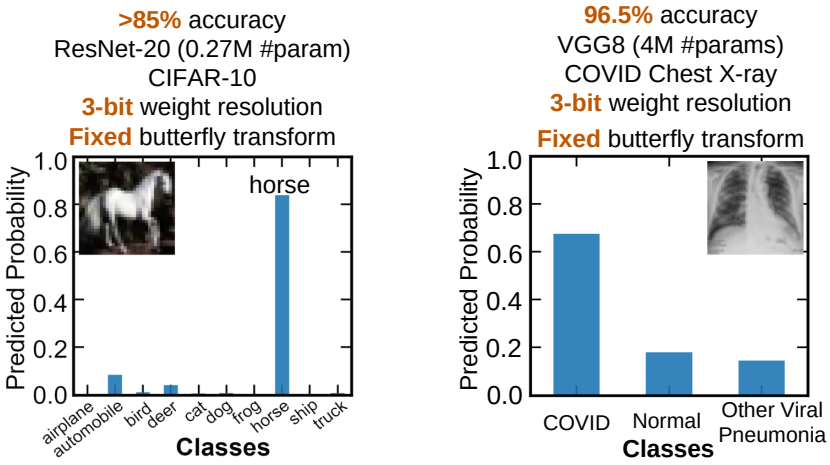


Figure 2.16: Evaluation on larger benchmarks: >85% accuracy with ResNet-20 on CIFAR-10 [112] and 96.5% accuracy with VGG-8 on Chest X-ray-based COVID detection [28].

for each MZI attenuator in the Σ unit), the inference accuracy of the CNN reaches 94.16% in our experimental demonstration, slightly below the simulated value of 94.59%. We further evaluate our chip on larger benchmarks to show the superior expressivity of our butterfly photonic tensor core design. On ResNet-20 CIFAR-10 image classification and VGG-8 Chest X-ray COVID detection tasks, our 3-bit photonic neural chip realize 85% and 96.5% accuracy, respectively, comparable to digital software models, which shows great application potential in different practical use domains. Figure 2.17(b) shows that after variation-aware training, our OSNN shows better noise tolerance compared to hardware-unaware training with higher accuracy under various noise intensities.

This 4×4 BPNC can implement photonic neural computing with 3×

fewer trainable optical components compared to MZI-based ONN architectures designed for general MVM [171], resulting in a ∼2× smaller footprint and ∼12× lower optical propagation delay. In Fig. 2.17(a), we can see the circuit footprint advantages of our butterfly structure will scale up to 4× with 32×32 block sizes. The optical delay can further be improved by 14× when the matrix size increases to 32×32.

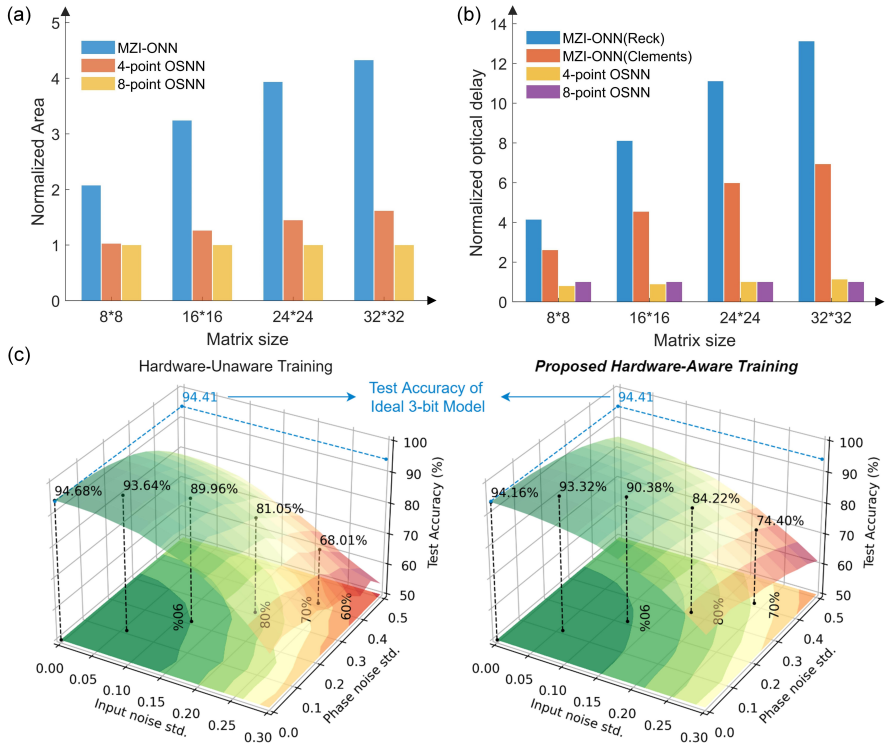


Figure 2.17: Experimental setup of OSNN. (a) Area and optical delay scaling with different matrix sizes. (b) Variation-aware training flow boosts the noise robustness of BPNC.

2.2.6 Summary

In this work, we propose a hardware-efficient butterfly-style optical neural network architecture. Our proposed ONN architecture is inspired by block-circulant matrix representation and efficiently realizes matrix-vector multiplication via photonic butterfly transform, theoretically saving 2.2∼3.7× area cost compared to the previous MZI-based ONN. We extend the proposed architecture to an optical microdisk-based frequency-domain CNN, and propose a trainable transform structure to enable a larger design space exploration. Our proposed training flow performs coarse-grained and fine-grained structured pruning to the butterfly circuit blocks and phase shifters in the butterfly transforms and further improves hardware efficiency with negligible accuracy degradation. We can realize over 80% power reduction in CNNs, over 90% power reduction in MLPs, and much better variation-robustness under device-level noises than prior work. We taped out a 4×4 butterfly-style photonic neural chip and experimentally demonstrated its usage on different ML tasks to show its great potential in various ML applications.

2.3 SqueezeLight: A Multi-Operand Ring-Based ONN with Cross-Layer Scalability

So far, we have discussed how to design specialized circuit structures beyond traditional universal programmable linear units to trade redundant matrix expressivity for higher hardware efficiency. However, prior state-of-the-art (SoTA) ONN designs still encounter scalability issues in terms of

high area cost. Though incoherent micro-ring resonator (MRR)-based ONN is considered one of the most compact ONNs given the small MRR device sizes [131, 190, 262, 145, 183], it reaches the current area lower bound, i.e., one optical device per multiply-accumulate (MAC) operation. It is technically challenging to further improve compactness by using traditional MRRs. Moreover, the high usage of wavelength limits the scalability of MRR-ONNs since practical weight matrix dimensions are far beyond the maximum wavelengths supported by modern dense WDM (DWDM) techniques, leading to unsatisfying throughput due to weight bank reuse [183]. MRR-ONNs also encounter robustness concerns under various noises and variations [183].

To break the current area lower bound of integrated ONNs, in this work, we propose a novel ONN architecture that squeezes matrix operations into arrays of ultra-compact customized multi-operand micro-ring resonators (MORRs), dubbed `SqueezeLight`, to enable scalable, efficient, and robust optical neurocomputing. We extend `SqueezeLight` to a separable optical CNN architecture with trainable MORR nonlinearity, showing augmented ex-

---

This `SqueezeLight` section is based on the following publications.

1. Jiaqi Gu, Chenghao Feng, Zheng Zhao, Zhoufeng Ying, Mingjie Liu, Ray T. Chen, and David Z. Pan, "SqueezeLight: Towards Scalable Optical Neural Networks with Multi-Operand Ring Resonators," IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE), Feb. 2021.
2. Jiaqi Gu, Chenghao Feng, Hanqing Zhu, Zheng Zhao, Zhoufeng Ying, Mingjie Liu, Ray T. Chen, and David Z. Pan, "SqueezeLight: A Multi-Operand Ring-Based Optical Neural Network with Cross-Layer Scalability," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), Jul. 2022.

I am the main contributor in charge of problem formulation, algorithm development, and experimental validations.

pressiveness and order-of-magnitude higher software training scalability than the original MORR-based CNN. The main contributions are as follows,

- **Scalability**: we explore the analog usage of multi-operand ring resonators to construct an ultra-compact ONN architecture with built-in nonlinearity, surpassing prior integrated ONNs by one order of magnitude in footprint.
- **Efficiency**: we employ fine-grained structured pruning in `SqueezeLight` for a quadratic efficiency boost.
- **Robustness**: we propose a sensitivity-aware learning technique to overcome thermal crosstalk and device variations to improve noise resilience.
- **Trainability**: We propose a novel separable optical CNN architecture with order-of-magnitude higher training scalability to support million-parameter ONNs.
- **Expressiveness**: We explore parametric MORR neurons with trainable nonlinearity to fortify the advantages of built-in nonlinearity of `SuqeezeLight`, leading to an average of $+2.1\%$ accuracy improvement on various vision recognition tasks.

2.3.1 Preliminaries

This section introduces the background knowledge of ONNs and our motivations.

2.3.1.1 Various Neural Network Designs

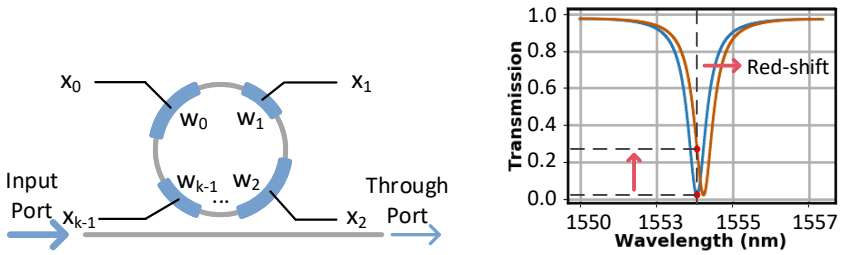
Convolutional neural networks (CNNs) learn discriminative representation via convolution-based linear operations. Kernelized NNs [137] have shown competitive performance by replacing convolutions with nonlinear projection kernels. Various linear and nonlinear convolution variants with better efficiency and robustness have been proposed, e.g., hyperbolic tangent convolution [132] and *AdderNet* [25]. In this work, we leverage the analog computing power of multi-operand ring resonators to construct compact optical neurons with built-in nonlinearity, achieving scalable optical neurocomputing with competitive model expressiveness.

2.3.1.2 Incoherent Optical Neural Network Architectures

Recently, ONN architectures have been rapidly evolving [171, 190, 253, 70, 74, 131, 262, 71, 72, 145]. Incoherent ONNs push the limits in circuit footprint by using MRR weight banks to implement matrices [131, 190]. However, the scalability of MRR-based ONNs is inevitably limited by the size of MRR weight banks and the high usage of wavelength. To break through the ONN scalability bound, in this work, we propose a more compact ONN architecture `SqueezeLight` with a lower device and wavelength usage.

2.3.1.3 Multi-Operand Ring Resonators

A multi-operand logic gate (MOLG) has been experimentally demonstrated to achieve multi-operand Boolean functions on a single MRR, achiev-



(a) (b)

Figure 2.18: (a) All-pass k-operand MORR. (b) Through port light intensity transmission of an all-pass MORR.

ing ultra-compact optical digital computing [231]. Figure 2.18(a) shows the structure of an all-pass multi-operand ring resonator (MORR). Unlike the traditional MORR with a single controller, an MORR has k active phase shifters independently controlled by k electrical signals x, each creating a phase shift $\phi_i(x_i)$. k phase shifts are accumulated $\phi = \sum_i \phi_i(x_i)$ and lead to a spectrum redshift $\Delta\lambda$, such that the transmitted light intensity on the through port changes accordingly. The transmission spectrum of an MORR is demonstrated in Fig. 2.18(b). A k-operand all-pass MORR has the following transfer function,

$$y = f(\phi) = \left| \frac{r - ae^{-j\phi}}{1 - rae^{-j\phi}} \right|^2 d, \ \phi = \sum_{i=0}^{k-1} \phi_i(x_i), \ \phi_i(x_i) \propto w_i x_i^2, \tag{2.26}$$

where x_i is the electrical input voltage, $\phi_i(\cdot)$ is the phase shift response curve of the actuator, ϕ is the accumulated round-trip phase shift of the MORR, r and a are self-coupling coefficient and single-pass amplitude transmission factor, and $d, y \in [0, 1]$ are the light intensity on the input port and through port, respectively. The weight w_i on the i-th input can be encoded into differ-

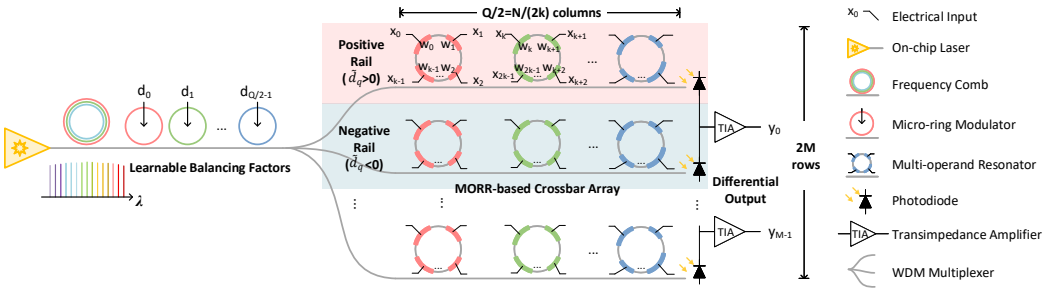


Figure 2.19: Proposed MORR-based ONN architecture `SqueezeLight` with learnable neuron balancing.

ent actuator arm lengths, different material properties, different input ranges, reconfigurable controller resistances, etc [231]. Instead of using MORR as a digital logic gate [231], we explore the *analog* usage of MORRs for optical neuromorphic computing.

2.3.2 Proposed MORR-based ONN Architecture

In this section, we present design details on the proposed `SqueezeLight` shown in Fig. 2.19 and introduce essential techniques for scalability and efficiency improvement. We summarize key notations for `SqueezeLight` in Table 2.8.

2.3.2.1 MORR-based Photonic Neuron

Different from the prior GEMM-based ONN design concept that only focuses on universal linear operations, we target unique nonlinear optical neurocomputing based on an ultra-compact MORR device. Recall that in Eq. (2.26), we can *squeeze length-k dot-product into the round-trip phase shift*

Table 2.8: Notations used in `SqueezeLight`.

$M \times N$	Matrix dimensions
$P \times Q$	Grid dimensions in blocking
k	Block size
k_{max}	Max #operands in an MORR
k'	#Non-zero weights per row after pruning
$w/\boldsymbol{W}$	Weights/weight matrix
x	Input signals
ϕ	Round-trip phase shift
$\hat{\phi}$	Round-trip phase shift after crosstalk
$\Delta\phi$	Phase noise
$f(\cdot)$	Nonlinear $y-\phi$ transmission
d	Learnable balancing factors
G	TIA gain
$\tilde{d}$	Balancing factor that absorbs G
$\gamma/\boldsymbol{\Gamma}$	Intra-MORR crosstalk coupling factor/matrix

of a single MORR. This dot-product result will be *probed* by the input light signal and activated by the MORR nonlinear transmission curve. The idle device will be initially calibrated to the on-resonance state, where the transmitted light intensity reaches the minimum. Then, k input voltage control signals will jointly create a phase shift to modulate the input light intensity. Hence, we model the MORR neuron as,

$$y = f(\sum_{i=0}^{k-1} \phi_i)d \propto f\Big(\sum_{i=0}^{k-1} w_i x_i^2\Big)d, \quad \text{s.t. } \mathrm{w_i} \geq 0 \tag{2.27}$$

where $f(\cdot)$ is the nonlinear $y-\phi$ transmission curve. Note that we are justified to assume all MORR nonlinear curves are identical because the shape of $f(\cdot)$ keeps almost the same within the practical wavelength range [195].

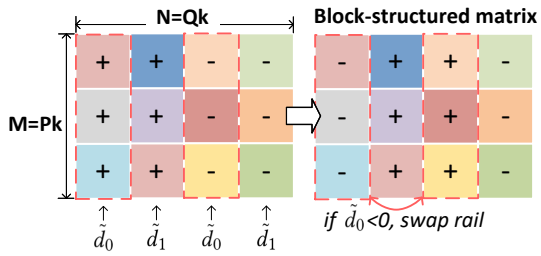


Figure 2.20: Block-structured matrices with learnable balancing factors.

2.3.2.2 `SqueezeLight` Architecture

Based on the above MORR neuron, we propose a novel ONN architecture `SqueezeLight` shown in Fig. 2.19. We assume to map an $M \times N$ weight matrix onto this MORR array. The matrix is partitioned into $P \times Q$ sub-matrices with size of $k \times k$, where $P = \lceil (M/k), Q = \lceil (N/k) \rceil \rceil$. `SqueezeLight` starts with an on-chip frequency comb to generate multiple wavelengths $(\lambda_0, \lambda_1, \cdots)$. Then, narrow-band MRRs are placed as wavelength-specific modulators $\boldsymbol{D} = (d_0, \cdots, d_{Q/2-1}) \in [0, 1]$ to achieve an adaptive dynamic MORR transmission range. Modulated probing light signals are evenly distributed into $2M$ rows. By placing a series of MORRs to form an array, we can implement a nonlinear ONN layer. Theoretically, we need total $2M$ rows and $\frac{Q}{2} = \frac{N}{2k}$ columns to implement an $M \times N$ weight matrix $\boldsymbol{W}$. The q-th MORR in one row will resonate at the wavelength λ_q and apply projection on a segment of length-k vector as $y_q = f(\sum_{i=0}^{k-1} w_{qi} x_{qi}^2) d_q$. At the end of the m-th row, a photo-detector will detect accumulated light intensity as $I_m = \sum_{q=0}^{Q/2-1} y_{mq}$.

Differential Detection for Full-range Weights. Typically, limited by the

physical implementation, the weights are restricted to be non-negative, which could limit the model representability. Hence we introduce a differential structure for full-range outputs and augment the expressiveness with learnable neuron balancing factors shown in Fig. 2.19. One row is halved into two adjacent rows as the positive rail I^+ and negative rail I^- respectively. The differential photo-current structure at the end enables full-range of outputs, equivalently forcing half weights, i.e., weights on rail I^-, to be non-positive values,

$$y_m = G(I_m^+ - I_m^-) = G\Big(\sum_{q=0}^{Q/2-1} y_{mq} - \sum_{q=Q/2-1}^{Q-1} y_{mq}\Big), \tag{2.28}$$

where G is the gain of the transimpedance amplifier (TIA), which can be used to extend the signal range. A direct benefit from this differential structure is that we can save 50% of wavelength usage by partitioning one width-Q row into two width-$\frac{Q}{2}$ rails.

Learnable Balancing Factors. With d=1, all MORRs are treated with the same importance as they have the same dynamic range $y_{mq} \in [0, 1], \forall q$, which loses the degree of freedom to assign different weights to different partial product results. To resolve this, we allow learnable MORR balancing factors $\tilde{\boldsymbol{D}} = \{\tilde{d}_q | \tilde{d}_q \in [-G_{max}, G_{max}], \tilde{d}_q = \tilde{d}_{q \ (mod) \ \frac{Q}{2}}, \ q \in [0, Q-1]\}$ and encode them in the MRRs at the beginning. Note that the maximum TIA gain G_{max} expands the implementable range to $\tilde{d} \in$[-G_{max}, G_{max}]. A column of MORRs share the same balancing factor as they share the same wavelength. Hence the MORR neuron is augmented as follows,

$$y_m = \sum_{q=0}^{Q-1} f\Big(\sum_{i=0}^{k-1} w_{mqi} x_{qi}^2\Big) \tilde{d}_q. \tag{2.29}$$

A natural question is how we can achieve full-range balancing factors as all-pass MRRs can only achieve non-negative transmission modulation. It turns out that by simply swapping two MORRs on the opposite rails, one can equivalently realize a negative factor $\tilde{d} < 0$ as shown in Fig. 2.20. This technique enables a learnable output range for different MORRs and thus boosts the expressiveness of `SqueezeLight`.

2.3.2.3 Peripheral Units

We briefly discuss peripheral units, with all system-level details being omitted since advanced system-level and architectural innovations in photonic NN accelerators [131, 262] are mostly applicable to `SqueezeLight`.

Normalization. Normalization operations, e.g., BatchNorm, can be implemented by the TIA gain and voltage signal offset with negligible latency overhead.

Nonlinear Activation. Since MORR-based neurons have built-in nonlinearity, extra electrical activations are not required.

Electrical Dataflow. The input signals/weights are loaded from high-bandwidth SRAM or ultra-fast photonic racetrack memory banks [182] and converted to analog signals through electrical digital-to-analog converters (DACs). The photo-currents are amplified by TIAs. Direct optical-electrical-optical (O-E-O) conversions will be used to cascade ONN layers without voltage-to-transmission encoding.

2.3.2.4 Area Reduction via Block-Squeezing

Thanks to the MORR device, we can squeeze a vector dot-product into one micro-ring. To achieve a quadratically more compact design, we further squeeze a matrix into one MORR via a block-squeezing method. Inspired by structured neural networks that restrict the weight matrix structure [40, 70] for better efficiency, we introduce this concept to `SqueezeLight` for higher compactness. An $M \times N$ block-structured matrix $\boldsymbol{W}$ contains $P \times Q$ square sub-matrices $\{w_{pq}\}_{p,q=0}^{P,Q}$, each being a $k \times k$ structured matrix. We use a circulant matrix as an example [40], where each column is essentially the circular shift of its length-k primary vector on the first column. Due to row-wise parameter sharing, the sub-matrix multiplication $\boldsymbol{w}_{pq} \cdot x_q$ can be efficiently *squeezed* into one k-operand MORR. Figure 2.21 visualizes the mapping from a 4×4 structured sub-matrix to an MORR. At time step t=0, we implement the first row. Then we shift the inputs around the ring to align with the corresponding weights on the second row and repeat this process. After k time steps with input rotation, we reuse the same MORR and finish an entire circulant matrix multiplication. In this way, we successfully achieve $O(k^2)$ times device usage reduction and save k times wavelength usage.

2.3.2.5 Sparsity Exploration via Fine-Grained Structured Pruning

For an $M \times N$ block-structured matrix, the total component usage adds up to $\frac{N}{2}$ MRRs and $(\frac{MN}{k^2})$ k-operand MORRs. Given fixed M and N, a larger k means fewer blocks and less MORR usage. However, implementing

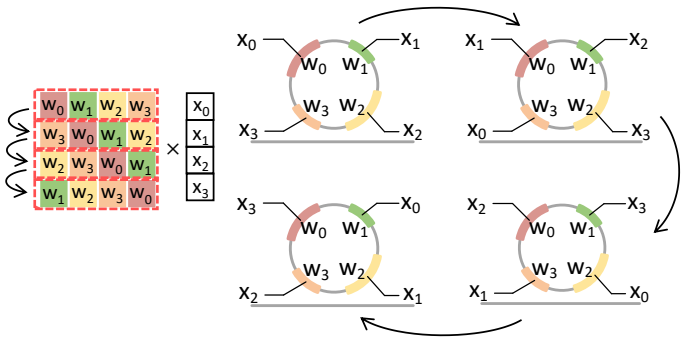


Figure 2.21: Squeezing a 4×4 block into one MORR using 4 cycles. The right part unfolds the *input rotation* mechanism temporally in 4 cycles on a single MORR.

MORRs with too many actuators can be challenging in practice. If one has a tight device usage budget, it is preferable to use a large sub-matrix that could exceed the MORR capacity, i.e., $k > k_{max}$. To overcome this, we prune each sub-matrix with a fine-grained structured sparsity. In Fig. 2.22, less important entries in the primary vector are forced to zero, leaving k' non-zero weights. The same sparsity pattern will be automatically imposed on other columns according to the pre-defined matrix structure. Our block-squeezing technique allows mapping the pruned sparse block with $k' \leq k_{max}$ into one MORR to maintain the highest compactness. We adopt a two-stage pruning procedure with learning rate rewinding to train `SqueezeLight` with structured sparsity, described in Alg. 1. We first pre-train and prune the weights with a target sparsity. Then we re-train the model from scratch with a rewound learning rate to achieve better accuracy than traditional post-training fine-tuning.

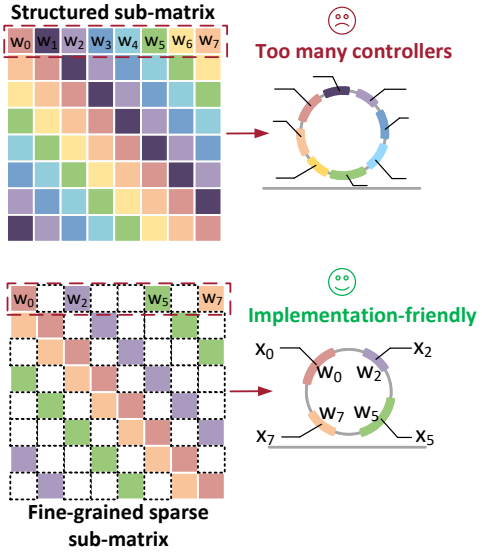


Figure 2.22: Fine-grained pruning enables squeezing a 8×8 structured block into a 4-op MORR.

2.3.2.6 Robustness Boost via Sensitivity-Aware Optimization

For analog computing, noise robustness is considered a practical concern [74, 252, 171, 261, 183, 142]. For MORRs, we consider random phase variations and intra-MORR crosstalk as the main non-ideal effects. The random variations can be estimated as a Gaussian noise on the phase shift $\Delta\phi \in \mathcal{N}(0, \sigma^2)$. We formulate the dynamic intra-MORR crosstalk among k actuators as $\hat{\mathbf{\Phi}} = \mathbf{\Gamma} \cdot \mathbf{\Phi}$ governed by a coupling matrix $\mathbf{\Gamma}$,

$$\begin{pmatrix} \hat{\phi}_0 \\ \hat{\phi}_1 \\ \cdots \\ \hat{\phi}_{k-1} \end{pmatrix} = \begin{pmatrix} \gamma_{0,0} & \gamma_{0,1} & \cdots & \gamma_{0,k-1} \\ \gamma_{1,0} & \gamma_{1,1} & \cdots & \gamma_{1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{k-1,0} & \gamma_{k-1,1} & \cdots & \gamma_{k-1,k-1} \end{pmatrix} \begin{pmatrix} \phi_0 \\ \phi_1 \\ \cdots \\ \phi_{k-1} \end{pmatrix}, \tag{2.30}$$

Note that the crosstalk effect $|\hat{\phi}_0 - \phi_0|$ is dynamically determined by the weight w and input x, but the coupling matrix $\mathbf{\Gamma}$ is constant after manufacturing. The

self-coupling factor $\gamma_{i,i} = 1$ and all mutual coupling factors $\gamma_{i,j}$ are basically determined by the spacing among phase shifters [140]. Hence we assume that they share the same value γ. We found that intra-MORR crosstalk is equivalent to a constant scaling factor on ϕ as follows,

$$\hat{y}_m = \sum_{q=0}^{Q-1} f\Big((1 + (k' - 1)\gamma)\phi_{mq} + \Delta\phi\Big)\tilde{d}_q. \tag{2.31}$$

This equation implies that pruning can reduce the crosstalk noise since only the left k' actuators have crosstalk after pruning. To better understand the sensitivity of MORR neurons to crosstalk, we show the transmission curve in Figure 2.23. We observe that the transmission curve $f(\cdot)$ has different sensitivity (gradient) at different wavelengths. Crosstalk effects induce an extra redshift in the spectrum, forcing all $\phi < \phi_s$ to have higher sensitivity and $\phi \geq \phi_s$ to have less sensitivity. Based on this observation, we introduce a sensitivity-aware optimization method to improve the robustness of`SqueezeLight`, shown in Alg. 1. We adopt the following the objective to train an L-layer `SqueezeLight`,

$$\mathcal{L} = \mathcal{L}_0(x; \boldsymbol{W}, \tilde{\boldsymbol{D}}, \boldsymbol{\Gamma}, \Delta\phi) + \alpha \sum_{l,m,q=0}^{L-1,M-1,Q-1} \nabla_\phi f(\hat{\phi}_{lmq} + \Delta\phi), \tag{2.32}$$

where $\mathcal{L}_0(x; \boldsymbol{W}, \tilde{\boldsymbol{D}}, \boldsymbol{\Gamma}, \Delta\phi)$ is the task-specific loss with noise injection, and the second term, denoted as $\mathcal{L}_S(\boldsymbol{\Gamma}, \Delta\phi)$, is a sensitivity-aware penalty term weighted by α. This method jointly considers variations and crosstalk with a gradient-based sensitivity penalty, enabling close-to-ideal test accuracy.

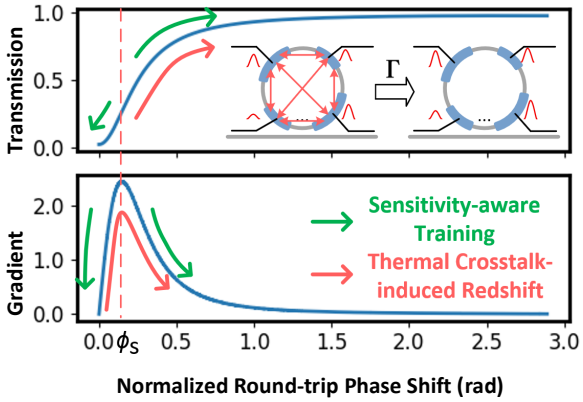


Figure 2.23: Transmission curve f and its gradient $\nabla_\phi f$ with thermal crosstalk and sensitivity-aware training.

2.3.3 Hardware Feasibility and Efficiency

We theoretically analyze the hardware feasibility and efficiency, and qualitatively compare essential features with previous ONNs.

2.3.3.1 MORR Physical Feasibility

Our MORR leverages the analog property of a successfully demonstrated digital MOLG [231]. We discuss how to encode weights and apply inputs to the analog MORR device. We can use high-speed DACs and high-speed E-O controllers to switch the input signals. Weight reprogramming is much less frequent than input signal switching. There are multiple possible approaches to implementing weights as modulation coefficients. If the weights are pre-defined and fixed, we can simply use controller length to encode the weights with zero energy cost in weight encoding. If the weights need a dynamic update, we can implement the weights as power scaling factors on the

Algorithm 1 Training algorithm of `SqueezeLight` with fine-grained structured pruning and sensitivity-aware optimization.

Input: Initial weights $\boldsymbol{W}^0 \in \mathbb{R}^{P\times Q\times k}$ and $\tilde{\boldsymbol{D}}^0 \in \mathcal{R}^{Q/2}$, pruning percentage $T = 1-\frac{k'}{k}$, pretraining step t_{pre}, initial step size η^0, decay factor β, penalty weight α, variation $\Delta\phi$, and crosstalk coupling matrix $\boldsymbol{\Gamma}$;
Output: Converged $\boldsymbol{w}^t$, $\boldsymbol{d}^t$, and a pruning mask $\mathcal{M} \in \mathbb{Z}^{P\times Q\times k}$;

for $t \leftarrow 1, \cdots, t_{pre}$ **do** ▷ Stage 1: Pretraining
  $\mathcal{L} \leftarrow \mathcal{L}_0^t(x; \boldsymbol{W}^{t-1}, \tilde{\boldsymbol{D}}^{t-1})$
  $(\boldsymbol{W}^t, \tilde{\boldsymbol{D}}^t) \leftarrow (\boldsymbol{W}^{t-1}, \tilde{\boldsymbol{D}}^{t-1}) - \eta^{t-1}(\nabla_{\boldsymbol{W}}\mathcal{L}, \nabla_{\tilde{\boldsymbol{D}}}\mathcal{L})$
  $\eta^t \leftarrow \eta^{t-1}\beta$ ▷ Learning rate decay
end for
$\eta^t \leftarrow \eta^0, \mathcal{M} \leftarrow 1$ ▷ Learning rate rewinding and initialize mask
for all $\boldsymbol{W}^t_{pqi} \in \boldsymbol{W}^t$ **do**
  if $\boldsymbol{W}^t_{pqi} <$ `percentile`$(\boldsymbol{W}^t_{pq}, T)$ **then**
    $\mathcal{M}_{pqi} \leftarrow 0$ ▷ Compute pruning mask
  end if
end for
while not converged **do** ▷ Stage 2: Fine-grained pruning
  $\mathcal{L} \leftarrow \mathcal{L}_0^t(x; \mathcal{M} \odot \boldsymbol{W}^{t-1}, \tilde{\boldsymbol{D}}^{t-1}, \boldsymbol{\Gamma}, \Delta\phi) + \alpha\mathcal{L}_S(\boldsymbol{\Gamma}, \Delta\phi)$ ▷ Sensitivity-aware regularization
  $(\boldsymbol{W}^t, \tilde{\boldsymbol{D}}^t) \leftarrow (\boldsymbol{W}^{t-1}, \tilde{\boldsymbol{D}}^{t-1}) - \eta^{t-1}(\nabla_{\boldsymbol{W}}\mathcal{L}, \nabla_{\tilde{\boldsymbol{D}}}\mathcal{L})$
  $\eta^t \leftarrow \eta^{t-1}\beta$ ▷ Learning rate decay
end while

input signals, e.g., program the electrical attenuation units to modulate the input signals. Low-speed electrical attenuators are enough to handle low-frequency weight reprogramming in most NN workloads.

Note that one may have concerns about the limited finesse of the low-Q MORR we show. This is a proof-of-concept example and not necessarily the most suitable ring design for `SqueezeLight`. In later simulation results, we show that our MORR array works well with high-Q MORRs. Simply shrinking the range of round-trip phase shift ϕ, either by scaling down the power of phase

Table 2.9: Symbolic hardware cost and qualitative feature comparison. The matrix is $M \times N$ with size-k blocks. B is the DWDM capacity. For a fair comparison, the device counts are converted to #MRRs based on real device sizes [171, 84, 70]. The area ratio β_a and power ratio β_p between one MZI (240×40 μm^2 [171], $\sim 48mW$ [84]) and one MRR (20×20 μm^2,~ 10 mW [179]) are β_a=24 and β_p=4.8.

	MZI-ONN [171]	Slim-ONN [253]	FFT-ONN [70]	MRR-ONN-1 [131]	MRR-ONN-2 [190]	SqueezeLight
#MRRs	$\beta_a MN$	$\sim \frac{\beta_a}{2} MN$	$\sim \frac{\beta_a}{4} MN$	$M \min(N, B)$	$M \min(N, B)$	$\frac{2M}{k} \min(\frac{N}{2k}, B)$
#Wavelength	1	1	1	$\min(N, B)$	$\min(N, B)$	$\min(\frac{N}{2k}, B)$
Latency	1	1	1	$\lceil \frac{N}{B} \rceil$	$\lceil \frac{N}{B} \rceil$	$k\lceil \frac{N}{2kB} \rceil$
Power	$\beta_p MN$	$\sim \frac{\beta_p}{2} MN$	$\sim \beta_p MN$	$M \min(N, B)$	$M \min(N, B)$	$\frac{2M}{k} \min(\frac{N}{2k}, B)$
Nonlinearity	Electrical	Electrical	Electrical	Electrical	Electrical	Built-in
Output range	Non-negative	Non-negative	Non-negative	Non-negative	Full range	Full range
Control cost	High	Medium-High	High	High	High	Medium

tuning signal x or reducing the tuning coefficient w, can create the same $y - \phi$ nonlinear curve as the low-Q MORR. Hence, we do not require a flat MORR spectrum. Instead, MORRs with high quality values and finesse are actually preferred to enable a larger WDM capacity for higher throughput and less spectrum crosstalk.

2.3.3.2 Symbolic Analysis on Area, Latency, and Power

In Table 2.9, our architecture outperforms three coherent ONNs by a large margin [171, 253, 70]. We focus on the comparison with the most compact designs MRR-ONN-1 [131] and MRR-ONN-2 [190] in terms of area cost $\mathcal{A}$, latency τ, and power $\mathcal{P}$. We assume the current DWDM capacity supports maximum B different wavelengths [191, 233].

First, the size and power of an MRR and a k-operand MORR can be assumed the same since they have the same phase tuning range, i.e., half of the resonance curve. Therefore, we focus on the number of resonators

Table 2.10: Comprehensive performance comparison between SqueezeLight and MRR-ONN. †To keep the same area cost, SqueezeLight uses 16 32×16 MORR arrays, and MRR-ONN uses 16 64×16 MRR weight banks in the accelerator. We use *DNN-Chip Predictor* [251] to search for an optimal hierarchical tiling strategy for SqueezeLight and MRR-ONN, respectively, and use their optimal tiling strategies for energy simulation.

Design	Area ↓ (mm^2)	Power ↓ (W)	Latency ↓ (ps)	Operate Freq ↑ (GHz)	Comp. Density ↑ (TOPS/mm^2)	Energy Eff. ↑ (TOPS/W)	† Sys. Energy ↓ (μJ)
SqueezeLight	5.12	6.972 (-48%)	141.3 (-21.4%)	7.0 (+25%)	11.3 (4.9×)	8.32 (9.8×)	0.2440 (-63.5%)
MRR-ONN	5.02	13.402	179.7	5.6	2.3	0.85	0.6676

in the discussion. We denote the computation efficiency as $\mathcal{E} = (\mathcal{A}\mathcal{P}\tau)^{-1}$. SqueezeLight achieves the following improvement over two MRR-ONNs when the matrix dimension is smaller than the DWDM capacity, i.e., $N < B$,

$$\frac{\mathcal{A}_{ours}}{\mathcal{A}_{prev}} \approx \frac{\mathcal{P}_{ours}}{\mathcal{P}_{prev}} \approx \frac{1}{k^2},\ \frac{\tau_{ours}}{\tau_{prev}} = \frac{k\lceil N/B \rceil}{\lceil N/(2kB) \rceil} = k,\ \frac{\mathcal{E}_{ours}}{\mathcal{E}_{prev}} \approx k^3. \tag{2.33}$$

Once the matrix width is larger than the maximum number of wavelengths available as $\frac{N}{2k} < B < N$, we can achieve,

$$\frac{\mathcal{A}_{ours}}{\mathcal{A}_{prev}} \approx \frac{\mathcal{P}_{ours}}{\mathcal{P}_{prev}} < \frac{2}{k},\ \frac{\tau_{ours}}{\tau_{prev}} = \frac{k}{\lceil \frac{N}{B} \rceil},\ \frac{\mathcal{E}_{ours}}{\mathcal{E}_{prev}} \approx \frac{Bk^3}{N} > \frac{k^2}{2}. \tag{2.34}$$

If the weight matrix is even larger, i.e., $B < \frac{N}{2k}$, we have

$$\frac{\mathcal{A}_{ours}}{\mathcal{A}_{prev}} \approx \frac{\mathcal{P}_{ours}}{\mathcal{P}_{prev}} \approx \frac{2}{k},\ \frac{\tau_{ours}}{\tau_{prev}} \approx \frac{1}{2},\ \frac{\mathcal{E}_{ours}}{\mathcal{E}_{prev}} \approx \frac{k^2}{2}, \text{if } B < \frac{N}{2k}. \tag{2.35}$$

It can be observed that our ONN gains more hardware efficiency advantage as B scales up, thus our scalability grows together with the development of the DWDM technology.

2.3.3.3 Qualitative Feature Comparison

In Table 2.9 we compare several key features of 6 ONN designs. Previous ONNs mainly focus on general matrix multiplication and offload the nonlinear activation to the electrical domain. In contrast, our proposed neuron leverages the built-in nonlinearity in MORRs to eliminate the overhead from electrical activation, enabling higher speed and efficiency. In terms of model expressivity, MRR-ONN-1 [131] has a limited solution space with only positive weights, while our designs support full-range weights with augmented representability via learnable balancing factors. `SqueezeLight` also benefits from lower control complexity and higher efficiency due to direct signal encoding $v_x = x$, while previous MRR-ONNs require additional nonlinear mapping to encode inputs/weights into voltage signals $v_x = \sqrt{\phi^{-1}(f^{-1}(x))}$.

2.3.3.4 Quantitative System Performance Evaluation

We give a more rigorous performance analysis on `SqueezeLight` and compare it with MRR-ONNs.

Compute Density and Delay. We assume to implement a 256×256 block-structured (k=8) weight matrix. We assume the ring spacing is 60 μm. The 4-op MORR radius is 20 μm, and the MRR radius is 5 μm. The WDM capacity is 16. If the MORR array contains 32×16 4-op MORRs, it takes roughly $32 \times 16 \times 100^2 \mu m^2$. Given the same footprint budget and same WDM capacity, we can construct a 64×16 MRR weight bank.

Taking into account the delay by modulators (10 ps), photodetec-

tors (10 ps), ADCs (100 ps), and the optical path ($100\mu m \times 16 \times n_g/c = 21.3\ ps$). The total delay of our MORR array is 141.3 ps, which corresponds to an operating frequency of 7 GHz. Every cycle, our MORR array can finish 8192 FLOPs. The compute density of `SqueezeLight` is $\frac{16\times256\times2\ OPs}{141.3ps\times(32\times16\times100\times100\mu m^2)} = 11.3\ TOPS/mm^2$. It takes `SqueezeLight` 16 cycles (2.26 ns) to implement the 256×256 matrix.

The latency for the 64×16 MRR weight bank is $10 + 10 + 100 + (70\mu m \times 2 \times 16 \times n_g/c) = 179.7\ ps$, which corresponds to an operating frequency of 5.6 GHz. Therefore, the compute density for MRR-ONN is $\frac{64\times16\times2\ OPs}{179.7ps\times(64\times16\times70\times70\mu m^2)} = 2.3\ TOPS/mm^2$. It takes the MRR-ONN 64 cycles (11.5 ns) to implement this 256×256 weight matrix.

Power. We consider power consumption including laser, 8-bit DAC, 8-bit ADC, ring locking, and ring programming. We use an 8-bit 10 GSPS ADC [1], which consumes 39 mW per channel. Each high-speed microring modulator approximately achieves 18 fJ/bit [179], which corresponds to the power P_{ring} of 0.126 mW under 7 GHz. The static locking power P_{lock} of each ring is around $P_{lock} \approx 0.5P_{\pi} = 9.75\ mW$ [179, 189]. For high-speed input x modulation, each DAC power is $P_{DAC} = 3.92\ mW$ [155, 93]. Since weight configuration is much less frequent than input signals, typically, the weight DAC dynamic power can be ignored. Based on the detection sensitivity and circuit insertion loss, the laser power [146] is $P_{laser} = \frac{h\nu}{\eta\times\mathtt{IL}}2^{2N_b+1} \times \mathtt{freq.} = 131.62\ mW$, where $h\nu$ is the photon energy at 1550 nm, η is the laser efficiency (0.2), `IL` is the insertion loss (0.25 dB/ring), and N_b is the resolution (8-bit). The power consumption

for a 32×16 MORR array is

$$\begin{aligned}&((32\times 16+16)P_{lock}+(32\times 16)P_{ring})+P_{laser}\\&\quad+256P_{DAC}+16P_{ADC}\\&\approx 5212.5+131.62+1003.52+624.00\ mW\\&\approx 6.972\ W.\end{aligned}\tag{2.36}$$

The energy efficiency of the MORR array is $\frac{16\times 256\times 2\ OPs}{141.3\ ps\times 6.972\ W}=8.32\ TOPS/W$.

For the 64×16 MRR weight bank, P_{ring} is 0.101 mW under 5.6 GHz [179]. Each MRR needs extra weight configuration power [189] $P_w=P_\pi/(2\times\texttt{finesse})\approx 0.4875\ mW$, where the finesse is around 20 [179]. The total power is

$$\begin{aligned}&(16P_{ring}+(16\times 64)(P_{lock}+P_w))+P_{laser}+16P_{DAC}+64P_{ADC}\\&\approx 10640.8+214.99+50.18+2496\ mW\approx 13.402\ W.\end{aligned}\tag{2.37}$$

The energy efficiency of the MRR weight bank is $\frac{64\times 16\times 2\ OPs}{179.7\ ps\times 13.402\ W}=0.850\,TOPS/W$.

System Energy Cost. We use a DNN-Chip Predictor [251] to simulate a 256×256 fully connected layer with a four-level memory hierarchy, including DRAM, SRAM-based global buffer (GB), network-on-chip (NoC) which describes the spatial data tiling and the parallelism of the system, and register files (RF). For `SqueezeLight`, we use 16 32×16 MORR array in the accelerator. For MRR-ONN, we use 16 64×16 MRR weight banks in the accelerator. We searched for optimal tiling strategies for them and applied them to those two accelerators. The basic memory energy model is based on Eyeriss [23]. MRR-ONN consumes 0.5135 μJ on data movement. It consumes 0.1541 μJ in computation. The total energy consumption of MRR-ONN is 0.6676 μJ.

In contrast, our sparse block-squeezing technique helps save 94% of the weight loading cost, such that `SqueezeLight` only consumes 0.2237 μJ on data movement. Plus the 0.0158 μJ in computation, the total energy consumption `SqueezeLight` is 0.2440 μJ, achieving 63.5% overall energy reduction. We summarize the above analysis in Table 2.10.

2.3.4 Extension to MORR-based Separable CNN with Augmented Trainability

To enable a real scalable ONN design, the three most important metrics are *representability*, *hardware efficiency*, and *software trainability*. Based on the nonlinear MORR neuron, we have demonstrated an ONN architecture with high *hardware efficiency* and *representability* in Fig. 2.19. However, the unsatisfying *trainability* of the MORR-based ONN fundamentally restricts the scalability of `SqueezeLight`. Specifically, for convolution (CONV) layers, partial convolution results for each MORR need to be stored and activated by the built-in nonlinearity. Such a mechanism turns out to consume considerable GPU memory and training time. This software trainability issue motivates us to design a more suitable architecture based on MORR arrays that can fully unleash the scalability advantages of `SqueezeLight` with augmented trainability.

2.3.4.1 MORR-based Separable CNN with Layer-Squeezing

An important trade-off in MORR-based ONN design is between representability and trainability. Hence, we propose an MORR-based separable

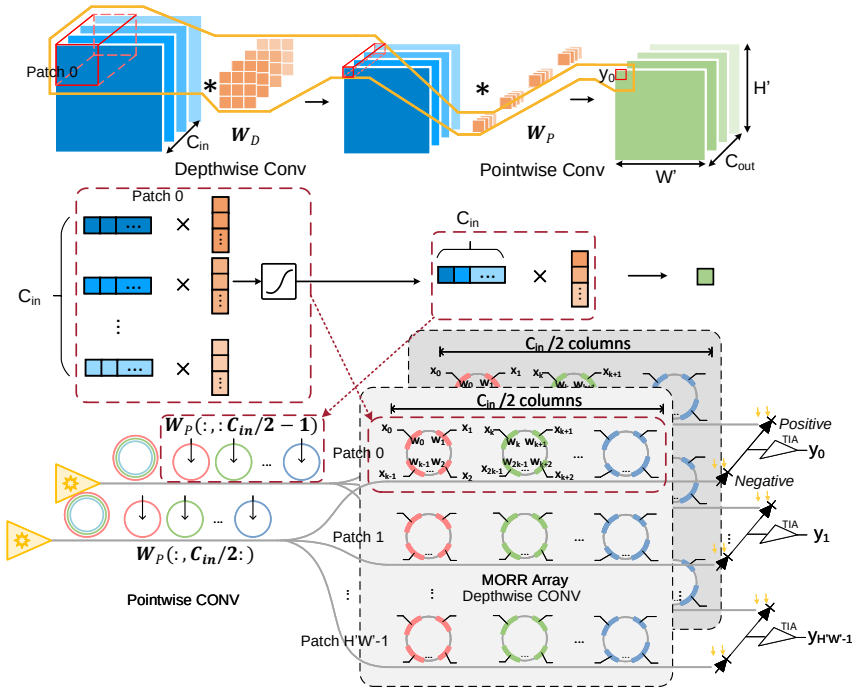


Figure 2.24: Architecture of separable `SqueezeLight`. Squeeze depthwise and pointwise convolutional layers into one MORR array.

CNN architecture that coincides with an advanced neural network design concept, i.e., *depth-wise separable convolution* (`DSCONV`).

`DSConv` contains a depth-wise convolution (`DWConv`) with per channel convolution and a point-wise convolution with 1×1 kernels (`PWConv`), which can be taken as a low-rank decomposition of an original CONV. Such advanced convolution is widely used in efficient NN architectures, e.g., MobileNet-family, to trim unnecessary computations without degrading the representability. The most exciting observation is the perfect match between `DSConv` and our MORR array, shown in Fig. 2.24. In other words, we *squeeze `DWConv` and*

PWConv layers into one MORR array. A feature patch with size of $C_{in} \times K \times K$ will convolve with the DWConv kernel $\boldsymbol{W}_D \in \mathbb{R}^{C_{in} \times 1 \times K \times K}$, corresponding to C_{in} length-K^2 dot-products. Hence, we can assign a row of C_{in} MORRs to implement it. Note that each k-operand MORR corresponds to one $K \times K$ CONV filter. Then, PWConv will perform pointwise linear projection on all channels with a kernel $\boldsymbol{W}_P \in \mathbb{R}^{C_{out} \times C_{in} \times 1 \times 1}$ and generate the final feature map. The pointwise linear projection can be directly mapped to the MRR-based balancing factors. To achieve balanced output, we need to split the MORR array into a positive and a negative array, each implementing half of DSConv. The negative array equivalently achieves the negative half of $\boldsymbol{W}_P$. The reason why we do not adopt negative/positive rails on the same array is that we want to maximize the parameter space of $\boldsymbol{W}_P$ without weight sharing. Theoretically, there will be $H'W'$ rows to map all feature patches. For different output channels, we can either duplicate the array for C_{out} times or reuse the array and sequentially reprogram the MRR-based $\boldsymbol{W}_P$.

Compared with the original MORR CONV engine, this augmented DSConv engine has the following advantages.

Excellent Trainability. When mapping one CONV layer to the original MORR array using *im2col*, the largest intermediate partial product feature map contains $H'W'BPQk \approx H'W'BC_{out}C_{in}/k$ elements. In contrast, the largest feature map in the DSConv module only contains $H'W'BC_{in}$ elements. The training memory footprint is approximately improved by C_{out}/k times, which significantly boosts the software trainability of our SqueezeLight.

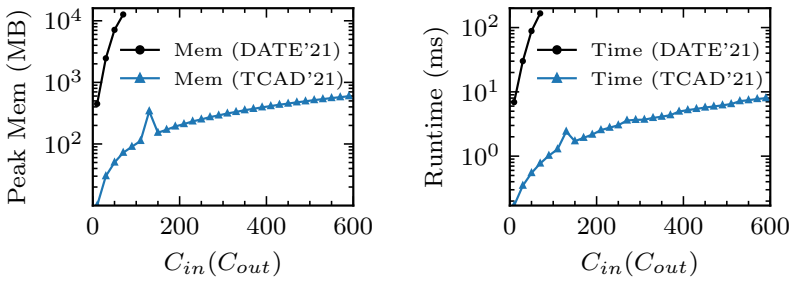


Figure 2.25: Peak GPU memory consumption (a) and average GPU runtime (b) evaluation on an MORR-based CONV3x3 layer (DATE'21) and a `DSConv`3x3 layer (TCAD'21) with different input/output channels.

Figure 2.25 shows *2-order-of-magnitude higher memory efficiency and runtime reduction* of the augmented `SqueezeLight` compared with the original MORR-based CONV engine [66]. Hence, by filling the trainability gap, all three aforementioned key metrics for scalable ONNs are met.

Compressed Model Size. This benefit naturally comes from the low-rank parameter space of `DSConv`. The weight size is reduced from $C_{out}C_{in}K^2$ to $C_{in}K^2 + C_{out}C_{in}$, with a compression ratio of $\sim K^2$.

Patch-Level Parallelism. The original MORR array essentially performs sequential matrix-vector multiplication, which processes one feature patch at one time. In contrast, the augmented MORR array maps multiple image patches to different rows in parallel, which share the same group of MRRs. Another advantage of this patch-level parallelism is the massive reuse of MRRs for $\boldsymbol{W}_P$. With the extensive MRR reuse, the advantages of ultra-compact MORR neurons will not be diluted by the usage of MRRs.

2.3.4.2 Parametric MORR Neuron via Trainable Nonlinearity

Thanks to the augmented software trainability of the MORR-based separable CONV engine, we are able to efficiently explore the representability of `SqueezeLight` deeper. Moving beyond the fixed nonlinear transmission curve of an MORR, we further explore more expressivity in our MORR-based neuron via trainable nonlinearity. Inspired by previous work on learning activations for NNs, we try to adapt the shape and the sharpness of the nonlinear curve by tuning the phase bias b and the input scaling factor s,

$$f_{b,s}(\phi) = f(s\phi + b) = f(s\sum_{i=0}^{k-1} w_i x_i^2 + b). \tag{2.38}$$

Figure 2.26 shows how b and s change the nonlinearity applied to the dot-product results. A dedicated biasing current can be applied to the actuators on MORRs to tune the curve's center wavelength. By scaling the heating power range of x with the factor s, the sharpness of the nonlinearity can also be efficiently tuned. Such tunable nonlinearity introduces extra non-convexity, leading to stronger representability than conventional activation functions, e.g., ReLU. In the later section, we will show the performance benefits of our trainable MORR neuron.

2.3.4.3 Nonlinearity-aware Initialization

A proper initialization is critical to the convergence of nonlinear non-convex optimization problems, especially for DNN training. Though various normalization methods, e.g., BatchNorm, can relax the sensitivity of DNN

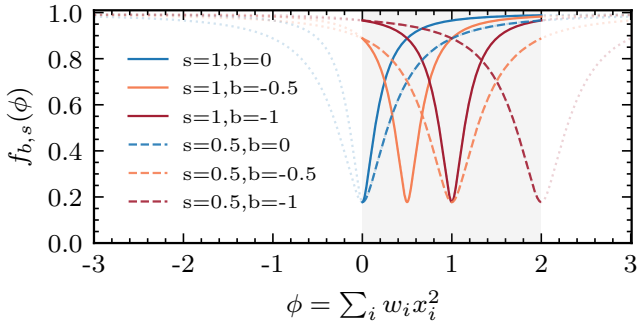


Figure 2.26: Trainable nonlinearity curve of parametric MORR neurons with different bias b and scale s. Curves highlighted in the shadow region are the activation functions applied to the dot-product ϕ.

learning to parameter initialization, the built-in nonlinearity of MORRs still requires appropriate weight distribution to avoid dot-product values falling into saturation ranges. The second reason for a specialized initialization method is the potential activation explosion due to normalized MORR output. Each MORR has a normalized output range of $[0, 1]$, such that the final activation magnitude is nearly proportional to the number of MORRs cascaded on one row.

Therefore, we show a nonlinearity-aware initialization algorithm to maintain nearly constant variance after layer cascading. We first assume the input x is normalized with zero center, i.e., $\mathbb{E}[x] = 0, \mathbb{D}[x] = \sigma_x^2$, and the statistics of non-negative weights are denoted as $\mathbb{E}[w]$ and $\mathbb{D}[w]$. Based on $\phi = \sum_{i=0}^{k-1} w_i x_i^2$, thus we can derive the variance of the accumulated round-trip phase shift of a k-segment MORR since x and w are independent random

variables,

$$\begin{aligned}\mathbb{D}[\phi] =& k(\mathbb{E}^2[w]\mathbb{D}[x^2] + \mathbb{E}^2[x^2]\mathbb{D}[w] + \mathbb{D}[w]\mathbb{D}[x^2]) \\ =& k\sigma_x^4(2\mathbb{E}^2[w] + 3\mathbb{D}[w]).\end{aligned} \tag{2.39}$$

In our algorithm, the weights will be sampled from a non-negative uniform distribution, i.e., $w \sim \mathcal{U}(0, L)$. Thus Eq. (2.39) can be rewritten as,

$$\mathbb{D}[\phi] = k\sigma_x^4\left(2\left(\frac{L}{2}\right)^2 + \frac{3L^2}{12}\right) = \frac{3k\sigma_x^4 L^2}{4}. \tag{2.40}$$

To solve L analytically, we need to know $\mathbb{D}[\phi]$. Considering the inter-MORR crosstalk due to spectrum leakage, a typical spectral distance between two adjacent wavelengths is at least 4 FWHM, where the full width half maximum (FWHM) represents the peak width when the energy is reduced to 50%. We heuristically and conservatively set a constraint to the maximum tuning range ($\pm 2\sigma_\phi$) of the round-trip phase shift, i.e., $4\sqrt{\mathbb{D}[\phi]} \approx 3$ FWHM. Now we give the uniform distribution for the weights w,

$$\texttt{morr\_uniform}(w) \sim \mathcal{U}\left(0, \sigma_x^2 \text{ FWHM}\sqrt{\frac{3}{4k}}\right). \tag{2.41}$$

So far, we properly initialize weights w considering the MORR transmission curve $f(\cdot)$. The next step is to initialize the learnable balancing factors $\tilde{\boldsymbol{D}}$ to keep the variance of the final activation the same as that of inputs x, i.e., $\mathbb{D}[y] = \mathbb{D}[x]$. The target distribution of balancing factors is zero-centered normal distribution, i.e., $\tilde{d} \sim \mathcal{N}(0, \sigma_{\tilde{d}}^2)$. Given the differential detection result $y = \sum_{q=0}^{Q-1} f(\phi)\tilde{d}_q$, we can rewrite it as $y = \sum_{q=0}^{Q/2} \Delta f(\phi)\tilde{d}_q$, where $\Delta f(\phi)$ is the equivalent differential MORR transmission between positive and negative

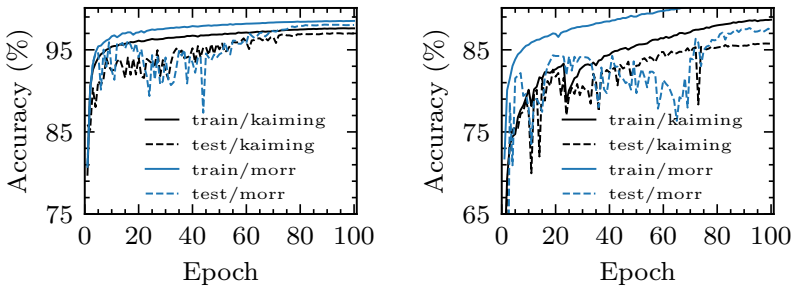


Figure 2.27: Compare the training and test accuracy curves on MNIST (a) and FashionMNIST (b). We compare our proposed `morr_uniform` with the kaiming initializer.

rails. We have $\mathbb{E}[\Delta f(\phi)] = 0$. Then we can derive the variance of the final activation,

$$\begin{aligned}\mathbb{D}[y] =& \frac{Q}{2}(\mathbb{E}^2[\Delta f(\phi)]\mathbb{D}[\tilde{d}] + \mathbb{E}^2[\tilde{d}]\mathbb{D}[\Delta f(\phi)] + \mathbb{D}[\Delta f(\phi)]\mathbb{D}[\tilde{d}]) \\ =& \frac{Q}{2}\mathbb{D}[\Delta f(\phi)]\sigma_{\tilde{d}}^2 = Q\mathbb{D}[f(\phi)]\sigma_{\tilde{d}}^2 = \sigma_x^2.\end{aligned} \tag{2.42}$$

The variance of the balancing factor is given by

$$\mathbb{D}[\tilde{d}] = \frac{\sigma_x^2}{Q\ \mathbb{D}[f(\phi)]} \approx \frac{\sigma_x^2}{Q \cdot g_f^2 \mathbb{D}[\phi]} = \frac{16\sigma_x^2}{9Q \cdot g_f^2 \cdot \mathrm{FWHM}^2}, \tag{2.43}$$

where g_f is a linear approximation to the gradient of the nonlinear transmission, i.e, $g_f = \frac{f(\phi_c+2FWHM)-f(\phi_c)}{2FWHM}$, where ϕ_c is the on-resonance phase shift.

Now, we show an ablation study to validate the effectiveness of the proposed nonlinearity-aware initialization method. From the training curves shown in Figure 2.27, we observe considerably faster convergence and higher test accuracy by using our proposed MORR-aware initialization method. In the following experiments, we will use the proposed initialization by default.

2.3.5 Experimental Results

We conduct optical simulation to validate the functionality and evaluate `SqueezeLight` on MNIST [115], FashionMNIST (FMNIST) [222], SVHN [149], CIFAR-10 [112], and CIFAR-100 dataset. All models are implemented with a PyTorch-centric ONN library TorchONN [75]. All ONNs are trained for 100 epochs using the Adam optimizer. Quantization-aware training [258] is applied to perform 8-bit weight/input/activation quantization.

2.3.5.1 Functionality Validation via Optical Simulation

One MORR Neuron. The MORR-based neuron is simulated using the commercial Lumerical INTERCONNECT tool for functional validation. Figure 2.28 plots the theoretical and simulated outputs of a 4-operand MORR under 1- to 4-bit precision. The design specification of the MORR is as follows. Radius $R = 20\mu m$, transmission coefficient $r = 0.8985$, attenuation factor $a = 0.8578$, effective index $n_{eff} = 2.35$. The central resonance wavelength is 1554.252 nm. We assume the 4-op MORR is programmed with 1- to 4-bit weights w, and we apply 1- to 4-bit voltage signals x to its controllers. We use Lumerical INTERCONNECT to simulate the intensity transmission of this MORR under given input/weights. The detector sensitivity is set to 1 A/W. Only the insertion loss of MORR is considered, while the loss in the waveguide is ignored. The derived neuron model has a high fidelity with <1% relative error compared with simulation results.

MORR Array. We further simulate a 2×4 MORR array with 4 MRRs to

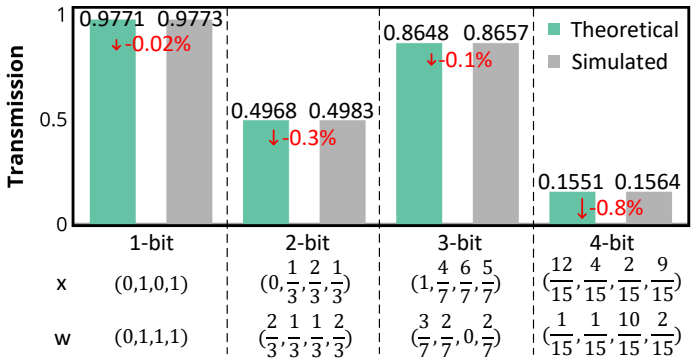


Figure 2.28: Compare theoretical and simulated results of a 4-op MORR.

implement balancing factors, together with 4 4-op MORRs on the positive rail and another 4 MORRs on the negative rail, as shown in Fig. 2.29. In the end, we add the differential photo-detection. All electrical voltage controls are of 4-bit precision. We use 1550, 1554, 1558, and 1562 nm as WDM sources. For the above 4 resonance wavelengths, we design the rings with a radius of 10.08 μm, 10.10 μm, 10.13 μm, and 10.16 μm, respectively. The transmission coefficient is $r = 0.98$, and the attenuation coefficient is $a = 0.97$. This MORR has much higher Q values and larger FSR than the 20 μm MORR used in the single MORR neuron simulation, which can enable higher WDM capacity with less spectrum crosstalk issue. In 4 test cases, the simulation results slightly deviate from the theoretical values due to wavelength misalignment, spectrum crosstalk, MORR insertion loss, etc., which validate the functionality of `SqueezeLight`.

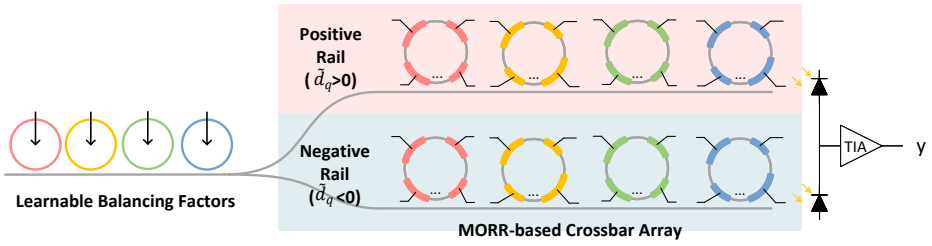


Figure 2.29: 2×4 MORR array used in simulation.

Table 2.11: Length-16 4-bit nonlinear vector-product simulated on a 2×4 4-op MORR array with 4 MRRs.

Test case	Simulated $\hat{y}$	Theoretical y	Error $\lvert\hat{y}-y\rvert$
1	0.3709	0.3708	0.0001
2	-0.1070	-0.0811	0.0259
3	-0.5916	-0.6170	0.0254
4	0.8505	0.8717	0.0212

2.3.5.2 Compare `SqueezeLight` with Prior MRR-ONNs

In Table 2.12, we compare the test accuracy among three ONNs: 1) MRR-ONN-1 with all-pass MRRs [131], 2) MRR-ONN-2 with add-drop MRRs [190], and 3) our proposed `SqueezeLight` without pruning (Ours). In all dataset and ONN settings, `SqueezeLight` achieves comparable test accuracy with 20-30× fewer ring resonators, 8× lower wavelength usage, and ∼80% fewer parameters.

2.3.5.3 Quantization

We also evaluate our architecture with low-bit quantization in Fig. 2.30. Even binarized `SqueezeLight` can achieve >95% accuracy on MNIST with

Table 2.12: Accuracy and hardware cost comparison. *small* model is C32K5S2-BN-C32K5S2-BN-F10, where *C32K5S2* is 5×5 convolution with 32 kernels and stride 2, *BN* is BatchNorm, and *F10* is a linear layer. *large* model is C64K5S2-BN-C64K5S2-BN-F10. We use $k = 8$ in convolutional layers and $k = 4$ in the final classifier. *#Device*, *#λ*, and *#Param* are the number of used resonators, wavelengths, and parameters, respectively. Normalized ratios are shown in the parenthesis. All models are trained with 8-bit weight/input/activation quantization.

Dataset	Model	MRR-ONN-1 [131]				MRR-ONN-2 [190]				Ours			
		Test Acc.	#Device	#λ	#Param	Test Acc.	#Device	#λ	#Param	Test Acc.	#Device	#λ	#Param
MNIST	small	97.81	39.90 K (23.86)	1152(8)	38 K	98.55	39.90 K (23.86)	1152(8)	38 K	98.01	1.67 K (1.00)	144(1)	8 K
MNIST	large	97.89	130.97 K (31.64)	2304(8)	127 K	98.84	130.97 K (31.64)	2304(8)	127 K	98.36	4.14 K (1.00)	288(1)	22 K
FMNIST	small	86.97	39.90 K (23.86)	1152(8)	38 K	89.52	39.90 K (23.86)	1152(8)	38 K	86.65	1.67 K (1.00)	144(1)	8 K
FMNIST	large	87.75	130.97 K (31.64)	2304(8)	127 K	90.30	130.97 K (31.64)	2304(8)	127 K	87.21	4.14 K (1.00)	288(1)	22 K
CIFAR-10	large	48.79	143.37 K (28.50)	3136(8)	139 K	61.69	143.37 K (28.50)	3136(8)	139 K	58.29	5.03 K (1.00)	392(1)	26 K

the *large* model, and >98% accuracy can be maintained with 2∼8 bit precision. Note that prior work has demonstrated MRR weight banks with higher than 7-bit weight precision [93]. Our `SqueezeLight` can work well with low-bit weight precision, which further justifies the practicality of our design.

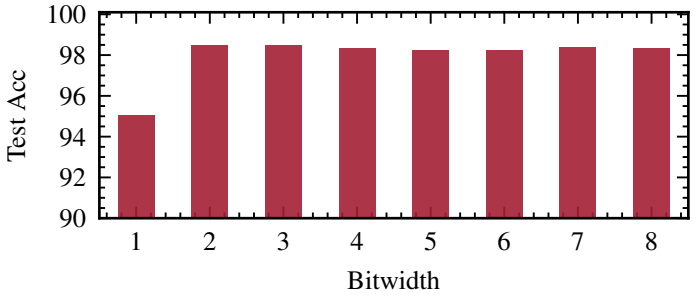


Figure 2.30: 1- to 8-bit quantization of `SqueezeLight` on MNIST.

2.3.5.4 Fine-Grained Structured Pruning

In Table 2.13, the pruned `SqueezeLight` only requires 4-operand MORRs to implement sparse sub-matrices with k'=4, which reduces the man-

Table 2.13: Fine-grained structured pruning evaluation. *#8op* represents the number of 8-operand MORRs. *Ours-P* represents all convolutional layers are pruned from k=8 to k'=4.

Dataset	Model	Ours				Ours-P			
		Acc.	#8op	#4op	#Param	Acc.	#8op	#4op	#Param
MNIST	small	98.01	416	864	8 K	98.02	0	1280	6 K
MNIST	large	98.36	1632	1728	22 K	98.58	0	3360	16 K
FMNIST	small	86.65	416	864	8 K	86.50	0	1280	6 K
FMNIST	large	87.21	1632	1728	22 K	87.36	0	3360	16 K
CIFAR-10	large	58.29	1680	2352	26 K	60.52	0	4032	19 K

ufacturing and control complexity with no accuracy loss. Moreover, the saved 30% parameters lead to less weight storage cost. This enables us to achieve better scalability by squeezing larger blocks into one MORR with negligible accuracy loss.

2.3.5.5 Variation Robustness Evaluation

In Fig. 2.31, we evaluate the variation robustness on 1) MRR-ONN-1, 2) MRR-ONN-2, 3) unpruned `SqueezeLight` (Ours), 4) pruned `SqueezeLight` (Ours-P), and 5) ours with pruning and robustness-aware training (Ours-PR). In the presence of the additional intra-MORR crosstalk, our ONN shows lower accuracy than other MRR-ONNs if no pruning or noise-aware training is performed. When we apply fine-grained structured pruning, the crosstalk sources are cut down from $k = 8$ to $k' = 4$, achieving improved noise tolerance. With sensitivity-aware training based on Eq. (2.32), `SqueezeLight` can stably maintain above 97% accuracy, which is reasonably close to the ideal accuracy, while other ONNs suffer from a sharply-degrading trend as the noise intensity increases. Therefore, our proposed lightweight robustness-aware training

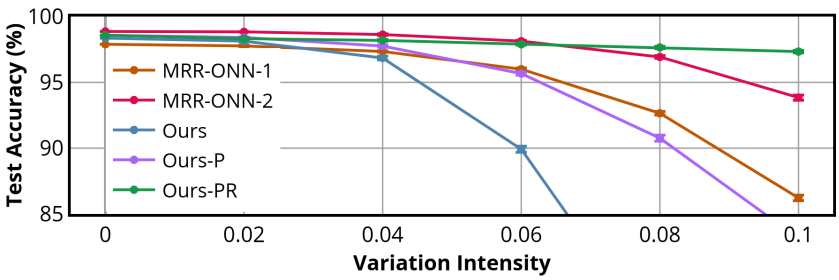


Figure 2.31: Robustness evaluation of the *large* model on MNIST. The error bar shows $\pm 1\sigma$ over 20 runs, e.g., 0.04 means γ=0.04 and std. $\Delta\phi$=0.04. *Ours-PR* means our pruned model with sensitivity-aware training (α=0.02).

guarantees `SqueezeLight` to have reliable inference performance even under practical non-ideal variations.

2.3.5.6 Extended MORR-based Separable CNN

In Table 2.14, we thoroughly evaluate the scalability and effectiveness of the extended separable CNN architecture on various learning tasks and models. On large models, the training of the original MORR CNN [66] fails due to prohibitive GPU memory and runtime cost. Thanks to the superior software trainability of our MORR-based separable convolution, we can scale the extended `SqueezeLight` to *million-parameter* ONN models, e.g., VGG-8, on various vision recognition datasets. Meanwhile, our separable MORR-based architecture saves $\sim 9\times$ parameters compared with the original Conv-based ONN model, leading to significant storage cost reduction.

We further compare `SuqeezeLight` with and without trainable MORR nonlinearity. Figure. 2.32 visualizes the learned channel-wise MORR nonlin-

Table 2.14: Compare the accuracy of separable SqueezeLight with fixed and learnable MORR nonlinearity on various tasks and models. We further prune convolutional kernels from k=9 to k'=4 to make them implementable with 4-operand MORRs. The suffix *-L* and *-P* represent using trainable MORR nonlinearity and structured pruning, respectively. The settings for CNN-2 are C64-C64-Pool5-F10. The settings for CNN-3 are C64-C64-C64-Pool5-F10. All convolutional layers (except for the first layer) in the model are implemented by the proposed MORR-based separable convolution.

Model	CNN-2	CNN-3	VGG-8		
Dataset	MNIST	FMNIST	SVHN	CIFAR-10	CIFAR-100
Ours	98.07	87.66	93.09	83.61	56.92
Ours-L	**98.67**	89.07	93.75	84.78	58.61
Ours-LP	98.37	**90.65**	**93.82**	**86.31**	**60.83**

earity curves in two DSConv layers of VGG-8. We observe that the SqueezeLight explores various monotonic or non-monotonic activation functions with augmented representability than a fixed zero-bias MORR nonlinearity curve. Our trainable MORR neurons boost the representability to effectively compensate for the performance loss from parameter compression, leading to an average of ∼1.1% test accuracy improvement on 5 learning tasks.

Note that the 3×3 depthwise convolution maps 9 weights to 1 MORR, which exceeds the typical capacity of 4 operands per MORR. We apply structured pruning to leave 4 non-zero weights in each depthwise convolutional kernel and demonstrate an average 2.13% accuracy improvement in Table 2.14.

2.3.6 Summary

In this work, we propose a novel ONN architecture SqueezeLight to break the compactness record of previous designs with higher scalability and ef-

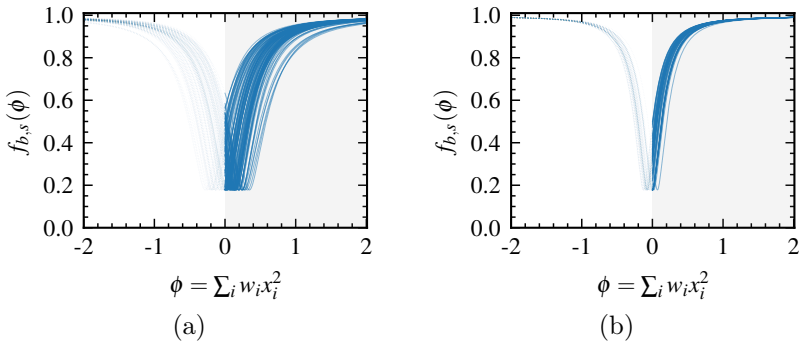


Figure 2.32: Learned MORR nonlinearity for the 1st (a) and 3rd (b) `DSConv` layers in VGG-8 on CIFAR-10. Each curve represents the nonlinearity curve of one input channel.

ficiency. An MORR-based optical neuron with built-in nonlinearity is proposed to squeeze vector dot-product into a single device. A block-squeezing technique with fine-grained structured pruning is proposed to further squeeze a matrix into an MORR to enable a quadratically more compact ONN design. We introduce sensitivity-aware training to enable close-to-ideal neurocomputing with high noise robustness. We give a theoretical analysis and thorough comparison to show the scalability and efficiency advantage of `SqueezeLight`. We extend `SqueezeLight` to an MORR-based separable CNN architecture with layer-wise squeezing and learnable nonlinearity, showing order-of-magnitude higher software training scalability and expressiveness improvement. Experiments show that `SqueezeLight` breaks the area lower bound of previous MRR-based ONNs with 20-30× better scalability and competitive expressiveness.

2.4 O^{2}NN: Optical Neural Networks with Differential Detection-Enabled Optical Operands

Previous coherent and incoherent ONN architectures all encode the weight matrices to the device configurations or circuit states as a stationary linear transform function applied to the dynamic input vectors. However, those weight-stationary photonic tensor cores are unable to support linear dot-product between two dynamically-encoded, full-range optical tensors, which potentially limits the application range of ONNs to accelerate modern advanced DNNs, e.g., essential operations in attention-based models [203] and advanced NNs with dynamically-generated weights [25]. Moreover, fully-optical operands can potentially benefit ONN on-chip training and online learning applications with frequent and high-speed weight updating [72, 99]. In terms of robustness, previous MZI-based ONN architectures encounter nontrivial accuracy degradation under low-bit signal quantization and practical device variation [252, 74], lacking compatibility with modern neural compression techniques.

In this work, we propose a new ONN architecture O^{2}NN to enable high-performance and versatile photonic neuromorphic computing. We present a

---

This O^{2}NN section is based on the following publication.

1. Jiaqi Gu, Zheng Zhao, Chenghao Feng, Zhoufeng Ying, Ray T. Chen, and David Z. Pan, "O2NN: Optical Neural Networks with Differential Detection-Enabled Optical Operands," IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE), Feb. 2021.

I am the main contributor in charge of problem formulation, algorithm development, and experimental validations.

WDM-based differential dot-product unit with augmented and balanced optical weights as the core engine. The main contributions and key features are as follows,

- Flexibility: we propose a novel ONN architecture based on WDM and differential detection to enable dynamic neural computing between two dynamically-encoded, full-range optical operands.
- Expressivity: we introduce extended optical weights and augmented quantization to improve the model expressivity.
- Robustness: we give a comprehensive analysis of the variation-robustness of our photonic core and provide an effective solution to improve the computational fidelity with knowledge-distillation-based noise-aware training.

2.4.1 Preliminaries

In this section, we introduce background knowledge about ONNs and our motivations.

2.4.1.1 DNNs with Stationary or Dynamic Linear Operations

Modern neural networks extensively adopt fully-connected layers and convolutional layers to achieve linear projection and feature extraction. Those linear operators can ultimately be implemented by general matrix multiplication (GEMM). For example, a 2-dimensional $K \times K$ convolution can be

described as $\boldsymbol{y} = \boldsymbol{W} * \boldsymbol{x},\ \boldsymbol{W} \in \mathbb{R}^{C_{out} \times C_{in} \times K \times K}, \boldsymbol{x} \in \mathbb{R}^{C_{in} \times H \times W}$, where C_{in}, C_{out}, k, H, W are input channel, output channel, kernel size, input height and width. To efficiently implement this algorithm, an *im2col* algorithm is widely adopted to unroll each convolution patch as a $(C_{in} \times K \times K)$-length vector. Therefore, the convolution is transformed to a GEMM $\boldsymbol{y}^{C_{out} \times (H'W')} = \boldsymbol{W}^{C_{out} \times N} \cdot \boldsymbol{x}^{N \times (H'W')}$, where H',W' are spatial height and width of $\boldsymbol{y}$, and N represents the unrolled vector length $(C_{in} \times K \times K)$. This *im2col* algorithm lays the foundation for modern high-performance CNN accelerator designs. Besides GEMM with static weights, advanced DNN architectures, e.g., attention-based natural language processing models [203] and dynamic CNNs with real-time-generated weights [25], require dynamic tensor-product-based operations to achieve better representability. Such essential and computationally-expensive modules require high-performance accelerators to support both operands to be dynamic signals.

2.4.2 Proposed O^{2}NN Architecture

In this section, we introduce the architecture and features of the proposed O^{2}NN, including expressivity, efficiency, and robustness.

2.4.2.1 Dot-Product Engine with Both Optical Operands

Our proposed architecture is designed with a WDM-based differential structure to support flexible *fully-optical vector dot-product* computations. It allows both operands to be dynamically-encoded optical signals, which is inher-

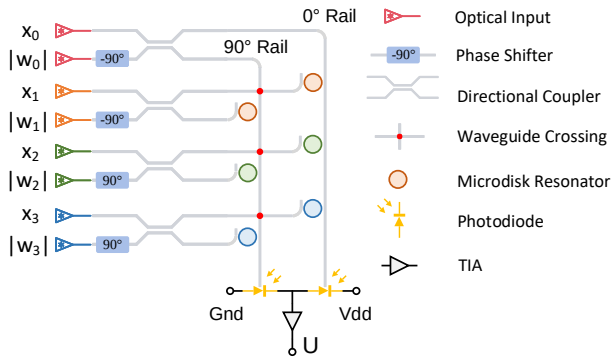


Figure 2.33: Schematic of proposed WDM-based differential dot-product architecture with optical-weight extension.

ently different from previous electro-optic neural architectures that are limited to stationary electrical weights [171, 131, 253, 70, 145]. Figure 2.33 demonstrates the structure of the engine to achieve dot-product between two optical vectors. In this architecture, the optical input vectors are denoted as $\boldsymbol{x} \in \mathbb{R}_+^N$ and $\boldsymbol{w} \in \mathbb{R}_+^N$, which are encoded into the light magnitude with a non-negative range of $[0, 1]$. Each pair of elements x_i and w_i is encoded in a unique wavelength λ_i. Interestingly, by putting a -$\pi/2$ degree phase shifter (PS) on the lower input port of a 2×2 optical directional coupler (DC), we can achieve an orthogonal addition/subtraction pair in the complex domain,

$$\begin{pmatrix} z_i^0 \\ z_i^1 \end{pmatrix} = \underbrace{\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & j \\ j & 1 \end{pmatrix}}_{\text{directional coupler}} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & e^{-j\pi/2} \end{pmatrix}}_{\text{phase shifter}} \begin{pmatrix} x_i \\ w_i \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} x_i + w_i \\ j(x_i - w_i) \end{pmatrix}, \tag{2.44}$$

where z_i^0, z_i^1 represent the upper and lower output port of the directional coupler, respectively. Different z_i with different wavelengths λ_i will be redirected by the resonated MR onto their corresponding rails, i.e., z_i^0 onto 0° rail and z_i^1 onto 90° rail. According to the WDM technique, different

optical wavelengths can propagate on the same waveguide without mutual interference, which enables highly parallel signal processing. At the end of the rail, photodiodes (PDs) are used to accumulate the energy of the WDM optical signals, proportional to the square of magnitude, and generate photocurrent I^0 and I^1,

$$\begin{pmatrix} I^0 \\ I^1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \|\boldsymbol{x} + \boldsymbol{w}\|_2^2 \\ \|j(\boldsymbol{x} - \boldsymbol{w})\|_2^2 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \sum_{i=0}^{N-1} (x_i + w_i)^2 \\ \sum_{i=0}^{N-1} (x_i - w_i)^2 \end{pmatrix}. \tag{2.45}$$

To calculate the optical dot product, we adopt a differential structure to transfer the two rails of photocurrent to an electrical voltage signal U which carries the dot product result,

$$U = G(I_0 - I_1) = 2G \sum_{i=0}^{N-1} x_i w_i \propto \sum_{i=0}^{N-1} x_i w_i, \tag{2.46}$$

where G is the gain of the on-chip transimpedance amplifiers (TIA). The superiority of the proposed architecture is that both operands are high-speed optical signals that allow dynamic encoding. Also, all components in this computing core are of fixed configuration, which can be fully passive with near-zero energy consumption, no external control overhead, and no potential thermal crosstalk, especially when both operands are dynamically generated from other optical circuits.

2.4.2.2 Expressivity Boost with Optical-Weight Extension

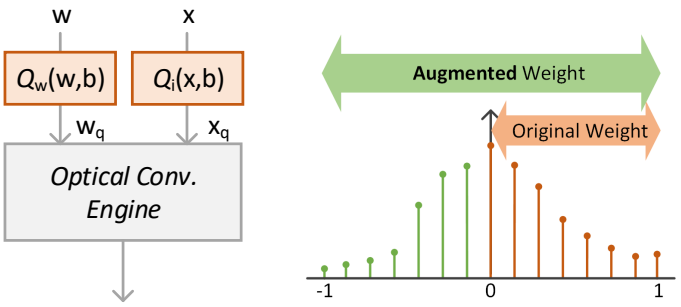
As analyzed in the previous section, both operands are constrained to be non-negative values as they are encoded into the light magnitude. However,

if one operand is weight, then it will cause trainability issues since non-negative weights inevitably limit the model expressivity due to abnormal activation distribution and pruned solution space. To solve this weight range limitation problem, we apply a static weight extension technique to augment the proposed architecture with better model expressivity and minimum hardware cost. By simply changing half of the passive phase shifters from $-90°$ to $90°$ and encoding $|w| \in [0, 1]$ into the light magnitude, shown in Fig. 2.33, we can statically allow half of the weights to be negative. The advantage is that the sign bit is offloaded to the extra π phase shift in the passive phase shifter without changing the input optical signal range. With static weight extension, our engine is able to generate a balanced output distribution with negligible hardware cost, which is the key feature that guarantees our superior model expressivity.

2.4.2.3 Performance Boost with Augmented Optical Quantization

For efficient optical neuromorphic computing, low-bitwidth inputs and weights are highly preferable. [171, 131, 74]. In this section, we introduce how augmented optical quantization empowers our proposed architecture with superior compatibility with low-bit quantization shown in Fig. 2.34(a). Given a b-bit quantized signal within $[0, 1]$, all possible quantized values can be expressed as $\{\frac{k}{2^b-1}\}_{k=0}^{2^b-1}$ using a uniform quantizer,

$$\mathcal{Q}(x, b) = \frac{1}{2^b - 1}\texttt{Round}\left(\frac{x}{1/(2^b - 1)}\right) \tag{2.47}$$



(a) (b)

Figure 2.34: (a) Distribution of weights with 3-bit augmented quantization. (b) Augmented optical quantization flow.

With the extra π phase shift on the negative optical path mentioned in Section 2.4.2.2, the engine can equivalently express negative weights $\{-\frac{k}{2^b-1}\}_{k=0}^{2^b-1}$, thus the number of implementable quantized weights $\boldsymbol{w}_q$ is almost doubled for free with a zero-centered symmetric distribution shown in Fig. 2.34(b). Even with binarized weights $|w| \in \{0, 1\}$, augmented optical quantization will boost our architecture to a ternary ONN $w \in \{-1, 0, 1\}$ with a higher model expressivity and representability but still maintain high performance from binarized laser modulation and potential ADC/DAC elimination. Moreover, our proposed engine can naturally implement scaled quantized weights $w \in \{-\mathbb{E}[|w|], 0, \mathbb{E}[|w|]\}$ [25, 117, 259], where $\mathbb{E}[|w|]$ calculates the layer-wise average of absolute weights, to achieve better trainability by setting the laser input intensity corresponding to those scaled values at no hardware cost. A quantization-aware training procedure [258] is adopted to train our proposed ONN. We denote the b-bit quantized weights and input as $\mathcal{Q}_w(w, b)$ and $\mathcal{Q}_i(x, b)$ respectively.

2.4.2.4 Robustness Analysis and Solution

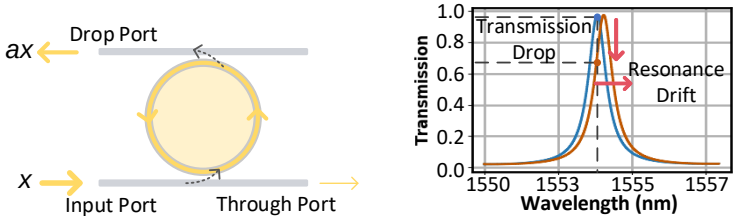
In this section, we analyze the variation-robustness of the proposed $\mathtt{O}^2\mathtt{NN}$ and present a solution to maximize its fidelity.

Dynamic Variation Analysis Considering there is stochastic dynamic drift in the analog optical signals, we have $\hat{x}_i = (x_i + \delta x_i)e^{j\delta\phi_i^d}$ and $\hat{w}_i = (w_i + \delta w_i)e^{j\delta\phi_i^d}$, where $\delta\phi_i^d$ is the dynamic phase drift. For a given input signal speed $\mathcal{B}$, the signal-to-noise ratio (SNR) is,

$$SNR = \frac{\bar{P}(\boldsymbol{x})}{\bar{P}(\delta\boldsymbol{x})} = \frac{\mathbb{E}[\boldsymbol{x}^2]}{\sigma^2} \approx \frac{C}{\mathcal{B}}, \quad \delta\boldsymbol{x} \sim \mathcal{N}(0, \sigma_x^2), \tag{2.48}$$

where the SNR is empirically to be inversely proportional to the input signal rate, e.g., 40 Gb/s signal rate corresponds to an SNR of 10 [178], thus the constant C is approximately set to 40. We extract the relative phase drift between two operands to an equivalent dynamic phase perturbation on the phase shifter, i.e., $\hat{x}_i = (x_i + \delta x_i)$ and $\hat{w}_i = (w_i + \delta w_i)$, and $\phi_i = \pm\pi/2 + \delta\phi_i^d$, where $\delta\phi^d \sim \mathcal{N}(0, \sigma_\phi^2)$ is the dynamic input phase drift.

Static Variation Analysis Considering the phase shifter produces an extra phase drift $\delta\phi^s \sim \mathcal{N}(0, \sigma_\phi^2)$. Though this drift is deterministic, it is expensive to evaluate each device drift individually for a large accelerator, hence we assume the static phase error is also a Gaussian random variable. Hence we have $\phi_i = \pm\pi/2 + \delta\phi_i^d + \delta\phi_i^s \sim \mathcal{N}(\pm\pi/2, 2\sigma_\phi^2)$, then the output of the directional



(a) (b)

Figure 2.35: (a) Add-drop MR resonator structure with non-ideal transmission factor. (b) Drop port transmission decay caused by resonation wavelength shift.

coupler can be derived as,

$$\begin{aligned}\begin{pmatrix}\hat{z}_i^{\,0}\\ \hat{z}_i^{\,1}\end{pmatrix} =& \frac{1}{\sqrt{2}}\begin{pmatrix}1 & j\\ j & 1\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & e^{j\phi_i}\end{pmatrix}\begin{pmatrix}\hat{x}_i\\ \hat{w}_i\end{pmatrix}\\ =& \frac{1}{\sqrt{2}}\begin{pmatrix}\hat{x}_i - \hat{w}_i\sin\phi_i + j\hat{w}_i\cos\phi_i\\ \hat{w}_i\cos\phi_i + j(\hat{x}_i + \hat{w}_i\sin\phi_i)\end{pmatrix}.\end{aligned} \tag{2.49}$$

Then we further consider the non-ideal transmission factor of the MR resonator that only transmits $\alpha \in [0,1]$ of the light energy to the rail due to resonance spectrum drift and insertion loss, shown in Fig. 2.35(a) and 2.35(b). α is estimated by a unilateral normally distributed variable $\alpha \sim \max(0, 1-|\mathcal{N}(0,\sigma_\alpha^2)|)$. Thus, the photocurrent can be given by,

$$\begin{pmatrix}\hat{I}_0\\ \hat{I}_1\end{pmatrix} = \frac{1}{2}\begin{pmatrix}\sum_{i=0}^{N-1}\alpha_i^0\left(\hat{x}_i^2 - 2\hat{x}_i\hat{w}_i\sin\phi_i + \hat{w}_i^2\right)\\ \sum_{i=0}^{N-1}\alpha_i^1\left(\hat{x}_i^2 + 2\hat{x}_i\hat{w}_i\sin\phi_i + \hat{w}_i^2\right)\end{pmatrix}. \tag{2.50}$$

Therefore, the differential output of the engine becomes,

$$\hat{U} \propto \sum_{i=0}^{N-1}\left(\frac{\alpha_i^0 - \alpha_i^1}{4}(\hat{x}_i^2 + \hat{w}_i^2) - \frac{\alpha_i^0 + \alpha_i^1}{2}\hat{x}_i\hat{w}_i\sin\phi_i\right). \tag{2.51}$$

Our engine is highly robust to static device noises since the design point $\phi = \pm\frac{\pi}{2}$ and $\alpha = 1$ are the local optima of *sin* and MR resonance curve with the minimum sensitivity.

Address Static and Dynamic Noises via Variation-Aware Knowledge Distillation We handle the above static and dynamic variations by training ONNs with the non-ideality modeling, shown in Eq. (2.51). We apply a knowledge distillation training strategy to improve the noise tolerance of our architecture. First, we pre-train an ideal ONN model without noise injection. as the teacher model $f_t(\cdot; \boldsymbol{W})$. Then we inject both static and dynamic variations to a noisy student model $f_s(\cdot; \boldsymbol{W}, \sigma_x, \sigma_\phi, \sigma_\alpha)$. We train the student model with a combined objective of hard target and soft target,

$$\begin{aligned} &\mathcal{L} = \beta T^2 \mathcal{D}_{KL}(q, p) + (1-\beta) H(y, \texttt{softmax}(f_s)), \\ &p = \frac{\exp(f_s/T)}{\sum \exp(f_s/T)}, \quad q = \frac{\exp(f_t/T)}{\sum \exp(f_t/T)}, \end{aligned} \tag{2.52}$$

where $\mathcal{D}_{KL}$ is the KL divergence, T is a temperature to control the smoothness, $H(y, \texttt{softmax}(f_s))$ is the cross-entropy loss, and β is a weighting factor to balance the soft and hard targets. Though this method introduces marginal training time overhead, it can effectively improve the ONN robustness to both static and dynamic errors. A noise source cooling strategy that gradually reduces the noise intensity is leveraged in low-bit (e.g., <3 bit) quantized training for better convergence.

2.4.2.5 Discussion: Hardware Cost and Features

In this section, we analyze and compare the hardware cost and features of our proposed ONN with previous ONN designs.

Table 2.15: Comparison among ONNs. Area cost is normalized to O^{2}NN on a size-N matrix-vector multiplication based on real device sizes [195, 171, 190, 131], i.e., one MZI ≈240×40 μm^2, one DC≈60×40 μm^2, one PS≈60×40 μm^2, and one MRR≈20×20 μm^2. Note that our area is not a simple accumulation of device sizes but is estimated with real layout information as a reference. Power is normalized to ours with the same statistics from the PDK [195], i.e., one PS≈20 mW and one MRR≈4 mW. The block size is set to k=4 for FFT-ONN [70].

	MZI-ONN [171]	Slim-ONN [253]	FFT-ONN [70]	MRR-ONN [190]	O^{2}NN
Norm. Area Cost	∼1.71×	∼0.86×	∼0.86×	∼0.1×	1×
Norm. Power	∼2×	∼1×	∼1.25×	∼0.2×	1×
GEMM	Yes	No	No	Yes	Yes
Optical Operands	Only One	Only One	Only One	Only One	Both
Robustness	Medium	Medium	Medium	Low	High
Control Complexity	Medium-High	Medium	Medium-Low	High	Medium
CNN Support	Yes	No	No	Yes	Yes
Quantization Compatibility	Low	Low	Medium	Medium-High	High
Output Range	Positive	Positive	Positive	Pos&Neg	Pos&Neg

Optical Input Encoding Cost The optical inputs are driven by coherent sources with phase shifters to control their phases. The weight encoding cost can be amortized by broadcasting to multiple processing units [80]. Moreover, since the weights are relatively stationary in ONNs, they can be directly modulated by phase change materials [145] or efficient laser modulation, which has near-zero area cost and power overhead. Since our architecture supports both operands to be optical signals, our architecture is the first integrated ONN that can achieve multiplication beyond static synaptic weights. Dynamic optical signals can be directly fed into our engine to support fully-optical attention-like operations [203] and NNs with dynamically-generated weights [25], where no extra energy is required due to its fully-passive design.

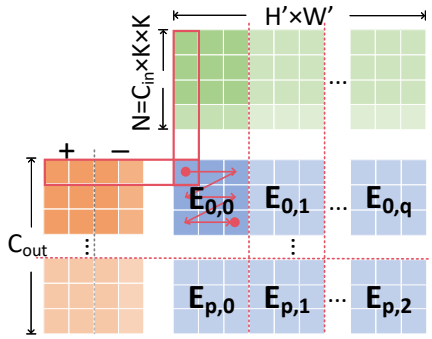


Figure 2.36: Tiling-based engine assignment for parallel GEMM.

Area Cost, Latency, and Energy Consumption Now we give a theoretical analysis of the hardware cost. Figure 2.36 shows how we assign multiple engines to a GEMM task with the *im2col* algorithm. Without losing generality, we only consider the most area-consuming directional couplers (DCs) and phase shifters (PSs) and assume they share the same size and aspect ratio of $w_{dc}/h_{dc} = 2$. We partition the GEMM task into $P \times Q$ sub-tasks to balance hardware cost and parallelism. For a matrix multiplication $\boldsymbol{A}_{M\times N} \cdot \boldsymbol{B}_{N\times L}$, the proposed architecture costs PQN PSs and PQN DCs. This partitioned engine assignment has an estimated latency of $\tau_{ours} = \frac{ML(2w_{dc}+Nh_{dc})}{PQc} = \frac{ML(N+4)h_{dc}}{PQc}$, where c is the speed of light. The previous MZI-based ONN architecture costs $M(M-1) + N(N-1) + 2\max(M,N)$ DCs and the same number of PSs to implement an $M \times N$ matrix-vector multiplication with latency $\tau_{mzi} = \frac{4(M+N)Lw_{dc}}{c} = \frac{8(M+N)Lh_{dc}}{c}$ [171]. We compare their latency-area product (LAP),

$$\begin{aligned} A_{mzi} \cdot \tau_{mzi} =& 8(M+N)(M^2+N^2)Lh_{dc}/c \\ A_{ours} \cdot \tau_{ours} =& MNL(N+4)h_{dc}/c. \end{aligned} \tag{2.53}$$

For fully-connected layers, we assume $M = N$, then we have $\frac{32N}{N+4}$ times smaller LAP than MZI-ONN. For a typical convolutional layer, we assume $N = K^2M$. Then the LAP improvement is around $8(K^2 + 1)$ times. If the MZI-based ONN adopts $P \times Q$ MZI sub-arrays, it costs around $\frac{8LNh_{dc}}{Qc}$ latency and PQN^2 components, which is still $8P$ times less efficient than our architecture.

Our architecture is also more energy-efficient than prior ONNs. For the photonics part, the only optical device tuning power is phase control and modulation. As mentioned before, the power of the weight modulation can be amortized by weight sharing and even reduced by direct laser modulation. In attention-like operations and layers with dynamically-generated weights, since both operands are directly from the previous layer and already in the optical domain, our architecture potentially consumes near-zero energy. Hence we have comparable or better energy efficiency than previous coherent ONNs in different application scenarios. For the electrical part, since our engine supports binarized inputs, our architecture is compatible with a DAC/ADC-less design, enabling potentially-ultra-low power as ADCs/DACs take most power [131, 74].

Differences from Prior Work We compare with previous ONNs in Table 2.15. Though larger than MRR-ONN, compared with other coherent ONNs [171, 253, 70], our architecture has a smaller area cost. No previous ONN can directly perform linear inner-product between two optical signals. Our proposed architecture is the first integrated ONN that supports both

operands to be optical signals, making it possible to realize direct layer cascading and *optical-optical product* that is necessary in attention-based neural architectures and NNs with dynamically-generated weights [25]. Compared with noise-sensitive MRR-ONN and unscalable, error-prone MZI-ONN [171, 252, 74] , our architecture achieves a relatively-low hardware cost, good model expressivity, and much better variation-tolerance. Furthermore, our architecture can well-support a wide spectrum of modern DNN architectures across CNN, MLP, and AdderNet, etc. MZI-ONN has low compatibility with network compression given its complicated principle [70, 74], and MRR-ONN only scales its weight range without increasing the valid quantization levels. In contrast, our ONN can seamlessly support extremely-low-bit quantization with better expressivity.

2.4.3 Experimental Results

We conduct experiments on the MNIST and FashionMNIST (FMNIST) dataset. We use a CNN setting C16-C16-P5-F32-F10, where C16 is a 3×3 convolutional (Conv) layer with 16 kernels, P5 means average pooling with output size 5×5, and F32 is a fully-connected (FC) layer with 32 neurons. We implement ONNs with PyTorch and train all models for 50 epochs with the Adam optimizer. and a mini-batch size of 32. We use Lumerical INTERCONNECT to do optical simulation with real devices from the AIM PDK [195], which should already model comprehensive and practical non-ideal factors. In knowledge distillation, we set T=6 and β=0.9. We gradually cool down the

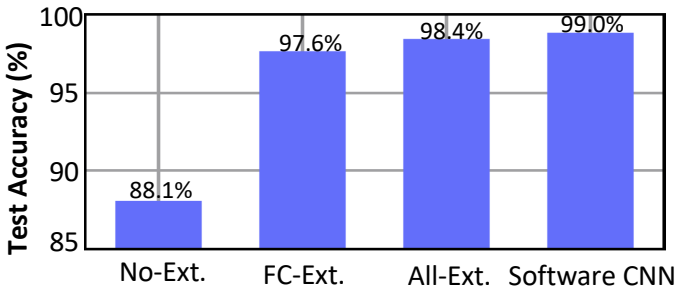


Figure 2.37: Evaluation of 8-bit optical-weight extension on MNIST. *Ext.* is short for extension.

noise intensity by 20% in lower than 3-bit cases.

2.4.3.1 Comparison Experiments

We first validate the effectiveness of optical-weight extension and augmented optical quantization, then evaluate the robustness via optical simulation and comparison experiments.

Optical-Weight Extension We compare four configurations of an 8-bit quantized optical CNN: 1) no optical-weight extension, 2) only extend FC layers, 3) apply weight extension to both FC and Conv layers, and 4) ideal software CNN. Figure. 2.37 shows that weight extension for fully-connected layers is essential for model expressivity. With balanced convolutions and fully-connected layers, the ONN model can recover its full modeling capacity with the highest inference accuracy. Therefore, the proposed ONN architecture can be used to accelerate modern CNN models with negligible accuracy degradation (∼0.5%) compared with the original software CNNs.

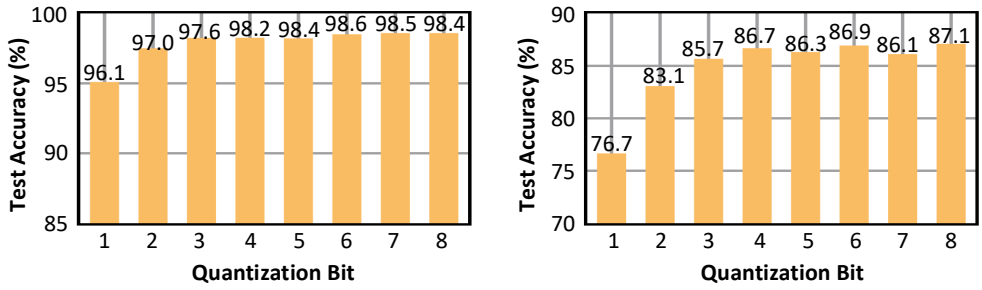


Figure 2.38: O^2NN quantization on (a) MNIST and (b) FMNIST.

Augmented Optical Quantization In analog DNN accelerators, the maximum precision is 8-bit or even 4-bit considering control complexity [171, 131, 74] In Fig. 2.38, our augmented optical quantization enlarges the solution space and achieves high accuracy even under low-bit quantization on both dataset. Even for binarized optical inputs, we can still maintain >96% accuracy on MNIST and 76% on FashionMNIST, which enables the coexistence of hardware-efficient optical computing and improved inference accuracy. In contrast, the state-of-the-art quantized MZI-ONN still suffers from > 10% accuracy drop on MNIST under extremely-low-bit quantization [74].

Variation-Robustness Evaluation We use Lumerical INTERCONNECT tools with the AMF process design kit (PDK) [2] to validate the fidelity of our architecture under static device variations. The simulation results in Fig. 2.39 show that phase shifter drift and MRR non-ideality lead to 10-15% dot-product error. Then, we further consider dynamic variations in our accuracy evaluation with our PyTorch-based ONN simulator on different setups, 1)

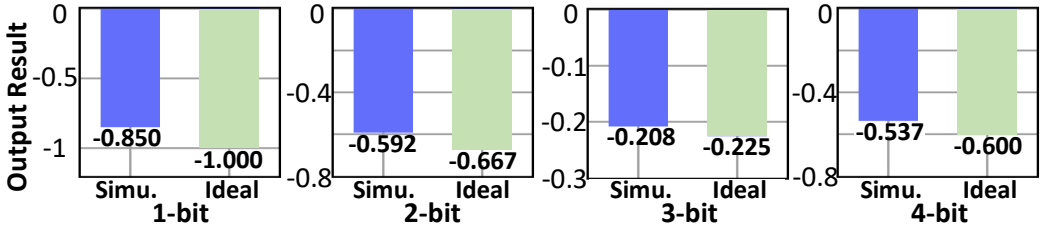


Figure 2.39: Optical simulation results with 1- to 4-bit precision. 1-bit: $\boldsymbol{x}$=(1,0,1,1), $\boldsymbol{w}$=(1,0,-1,-1). 2-bit: $\boldsymbol{x}$=$(\frac{2}{3}, \frac{2}{3}, \frac{1}{3}, \frac{2}{3})$, $\boldsymbol{w}$=$(\frac{1}{3}, 0, -\frac{2}{3}, -1)$. 3-bit: $\boldsymbol{x}$=$(0, \frac{1}{7}, \frac{1}{7}, \frac{6}{7})$, $\boldsymbol{w}$=$(\frac{1}{7}, \frac{6}{7}, -\frac{5}{7}, -\frac{2}{7})$. 4-bit: $\boldsymbol{x}$=$(\frac{1}{15}, \frac{1}{5}, \frac{11}{15}, \frac{7}{15})$, $\boldsymbol{w}$=$(\frac{8}{15}, \frac{2}{15}, -\frac{11}{15}, -\frac{4}{15})$.

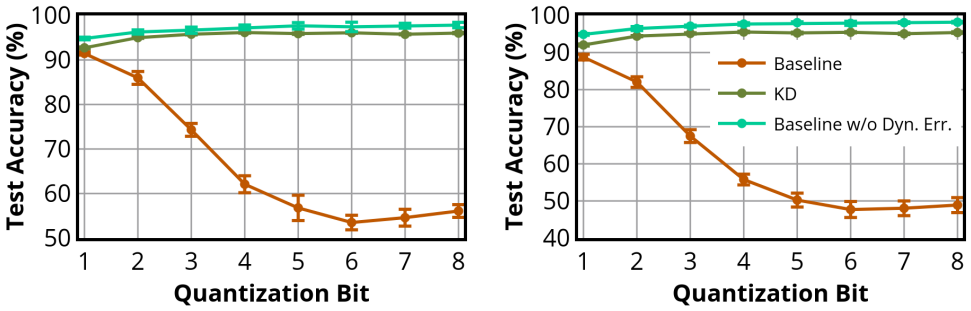


Figure 2.40: Robustness evaluation on MNIST. Error bars show the $\pm 1\sigma$ variance. (a) σ_ϕ=0.04, σ_α=0.04, SNR=39.81 (16 dB) (b) σ_ϕ=0.05, σ_α=0.05, SNR=31.62 (15 dB).

noise-unaware training (`Baseline`), 2) noise-unaware training w/o dynamic variations (`Baseline w/o Dyn. Err.`), and 3) variation-aware knowledge distillation (`KD`). Figure 2.40 shows that our `O`^{2}`NN` is extremely robust to large static device error [74, 252], consistent with the analysis in Section 2.4.2.4, but sensitive to dynamic variations. Our knowledge-distillation-based training method can help recover the majority of the accuracy with ∼3% degradation under both static and dynamic noises. We also observe higher robustness to

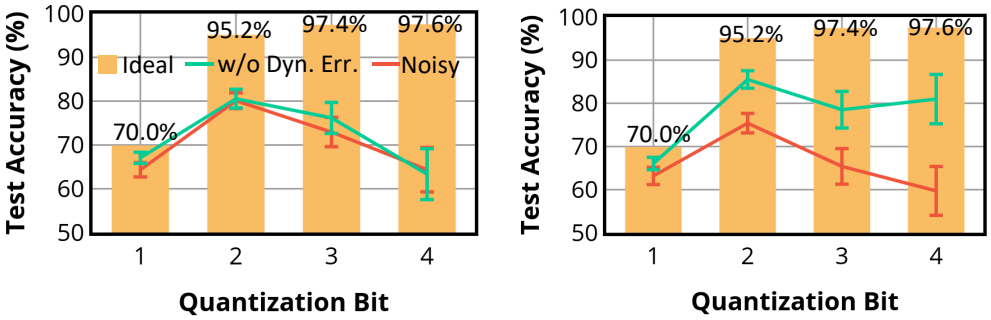


Figure 2.41: Robustness of MRR-ONN [190] on MNIST. (a) σ_α=0.04, SNR=39.81 (16 dB). (b) σ_α=0.05, SNR=31.62 (15 dB).

dynamic noise with lower bitwidth, which is beneficial for binary or ternary ONNs and thus further shows our superior compatibility with low-bit quantization. As a comparison, we show the robustness of MRR-ONN in Fig. 2.41. We observe that it has a high sensitivity to both static and dynamic errors and suffers from a larger accuracy degradation than our O^{2}NN under low-bit quantization. MZI-ONN typically suffers from even larger accuracy loss due to severe phase error accumulation effects [171, 74], thus we do not show its accuracy here for brevity.

2.4.4 Summary

In this work, we propose a new optical neural network architecture O^{2}NN to enable efficient and noise-robust photonic neuromorphic computing, which is the first one that supports tensor products with both operands to be dynamically-encoded light signals. A novel WDM-based differential dot-product engine is presented with extended optical weights and augmented

quantization techniques, demonstrating enhanced model expressivity and performance under low-bit quantization. We give an analysis of static and dynamic variations and present a knowledge-distillation-based training method to enable variation-tolerant optical neurocomputing under practical noises. A thorough comparison with prior work shows our advantages in hardware cost, efficiency, and features. Experimental results demonstrate that our O^{2}NN can support flexible, robust, and efficient optical neural computing, with both operands being optical signals even when low-bit optical quantization and practical variations exist.

2.5 Towards Memory-Efficient Photonic Neural Accelerators via Multi-Level *in-situ* Generation

So far, we have discussed customized photonic computing engine designs with various hardware-algorithm co-optimization approaches to boost their performance, efficiency, and flexibility. Now, we extend the scope to the upper architecture/system-level optimization. The system-level performance of photonic ML accelerators is still bottlenecked by peripheral electrical circuitry. Memory access and data movement are critical bottlenecks since it fails to match the computing capability of emerging tensor cores. Especially for

---

This memory-efficient ONN section is based on the following publication.

1. Jiaqi Gu, Hanqing Zhu, Chenghao Feng, Mingjie Liu, Zixuan Jiang, Ray T. Chen, and David Z. Pan, "Towards Memory-Efficient Neural Networks via Multi-Level in situ Generation," International Conference on Computer Vision (ICCV), Oct. 2021.

I am the main contributor in charge of problem formulation, algorithm development, and experimental validations.

emerging accelerators, e.g., ReRAM-based and photonics-based engines, the enormous latency, power, and bandwidth gap between memory and computing engines severely prohibits fully utilizing their advanced computing power.

Previous efforts towards memory-efficient accelerator designs focus on weight quantization [258, 160, 82], pruning with sparsity exploration [119, 82, 83, 214, 243, 37], structured weight matrices [40, 125, 70, 209], slim architectures [101, 27, 13], better hardware scheduling [136, 246], low-rank approximation [38, 187, 119, 250, 229, 228], etc. However, limited research has been done to investigate the intrinsic redundancy in CNN kernels thoroughly. It is in high demand to provide a unique memory optimization strategy that fully exploits the potential of advanced ultra-fast AI acceleration platforms.

Therefore, this study proposes a unified framework that generalizes prior low-rank solutions for memory-efficient NN designs via a multi-level *in situ* weight generation technique with mixed-precision quantization. We are the first to jointly explore multi-level redundancy in channel, kernel, and bitwidth based on a strong intuition of the intrinsic correlations within convolutions. A photonic *in-situ* weight generator is presented to show how our method can help unleash the full power of emerging neuromorphic computing systems. The main contributions of this work are as follows,

- We explore the multi-level intrinsic correlation in CNNs and propose a unified framework that generalizes prior low-rank-based convolution designs for higher memory efficiency.

- We fully leverage the ultra-fast execution speed of emerging accelerators and propose a hardware-aware multi-level *in-situ* generation to trade expensive memory access for much cheaper computations.
- We integrate a precision-preserving mixed-precision strategy to leverage the bit-level redundancy in multi-level bases for a larger design space exploration.
- Experiments and a photonic accelerator case study show that our proposed multi-level *in-situ* generation and mixed-precision techniques can save ∼97% weight load latency and significantly reduce memory cost by 10-20× with competitive accuracy compared to prior methods, even on compact networks and complex tasks.

2.5.1 Preliminary

In this section, we give a brief introduction to the background knowledge and our motivation.

2.5.1.1 Memory Bottleneck in NN Accelerator Designs

Previous works have proposed extensive NN accelerator architectures to enable efficient DNN inference. Recent emerging non-Von Neumann accelerators mainly focus on the innovation of the core matrix multiplication engine. However, the computation speed and efficiency of the cores are no longer the bottlenecks of the overall system. To prove this claim, Figure 2.42(d) shows that multiple cascaded small convolutional layers have fewer floating-point

operations (FLOPs) than a single wide convolutional layer but have higher execution time due to lower parallelism and more memory transactions. Hence, the expensive memory transaction and interconnect delay turn out to a pain point.

Most accelerators still rely on on-chip SRAMs and off-chip DRAMs to store/access weights, bringing serious challenges regarding the significant data movement cost. First, the mismatch between memory and computing cores in terms of latency and bandwidth heavily limits the potential performance of modern accelerators, especially for ultra-fast optical accelerators. Typical DRAM and SRAM have an access time of tens of nanoseconds, and the fastest SRAM runs at only 5 GHz. However, for example, the computation is executed at the speed of light (picosecond-level delay) in optical NNs with massive parallelism and potentially over 100 GHz photo-detection rate [171, 9].

Moreover, data movement becomes the power bottleneck. Figure 2.42(a) shows the power breakdown on a recent photonic neural chip `Mars` [159, 196]. The SRAM access dominates the total power consumption. The same issue also exists in state-of-the-art (SOTA) electrical digital accelerators like famous `Eyeriss` [23, 24] shown in Figure 2.42(b).

Limited prior works have explicitly optimized memory cost for emerging accelerators by leveraging their ultra-fast computing speed. Hence, a specialized memory-efficient NN design methodology to minimize data movement cost is exciting and essential to explore.

2.5.1.2 Efficiency and Accuracy Trade-off

Extensive work has been done to explore the NN design space for higher efficiency with less accuracy degradation. Efficient neural architectures are designed with lightweight structures, e.g., depthwise separable convolution [27], blueprint convolution [79], channel shuffling [101], etc. Besides, network compression techniques are often utilized to explore the sparsity and redundancy

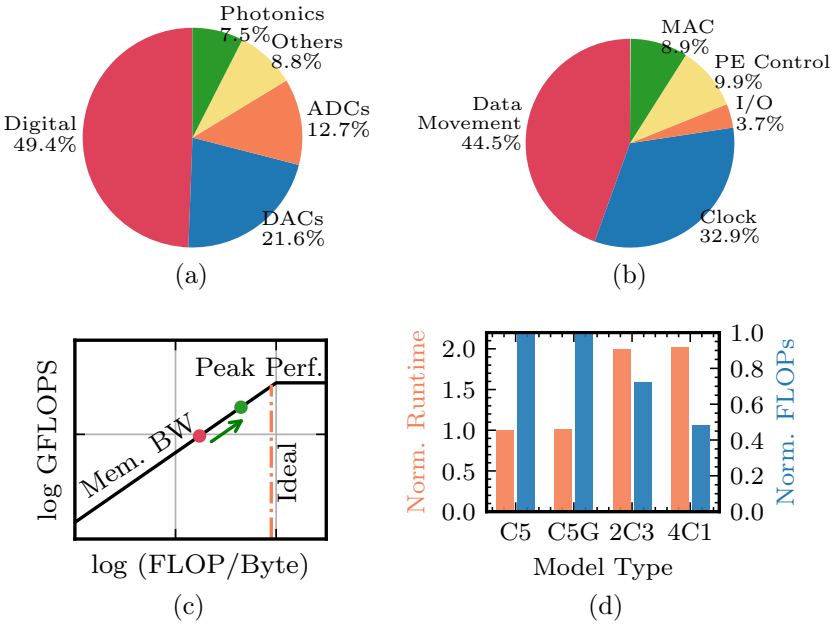


Figure 2.42: Power breakdown of a silicon photonic accelerator `Mars` [159, 196] (a) and an electrical accelerator `Eyeriss` [23] (b). The data movement (red) takes the most power for both. (c) Roofline model of emerging accelerators. Memory-bounded designs (red point) need to be improved to a better design (green point) (d) Normalized runtime and number of floating-point operations (FLOPs) among different convolution (Conv) types. C5 is 5×5 Conv, C5G is 5×5 Conv with low-rank decomposition, 2C3 is two cascaded 3×3 Conv, and 4C1 is four cascaded 1×3 Conv.

of DNNs and trim the model size by pruning and quantization [83, 82]. Furthermore, low-rank decomposition [119, 250] is a widely adopted technique to reduce the number of parameters by approximating a weight matrix by two smaller matrices. Also, structured neural networks [70, 71, 125] have been proposed to reduce memory cost with block-circulant matrix representation and Fourier-transform-based algorithm.

The above generic methods are applicable for emerging ultra-fast neuromorphic engines but do not fully leverage their powerful computing capability. It will be interesting and promising to explore the intrinsic correlation in DNN weights and enable *in-situ* weight generation by the computing core itself to minimize data movement from memory.

2.5.2 Proposed Memory-Efficient Architecture Design

Motivated by prior work [119, 250, 27, 79], we focus on widely deployed convolutional neural networks (CNNs) to thoroughly explore their intrinsic multi-level redundancy for better efficiency. We consider a 2-dimensional (2-D) convolutional kernel $\boldsymbol{W} \in \mathbb{R}^{C_o \times C_i \times k \times k}$ with C_o kernels, C_i input channels, and kernel sizes k. Interestingly we observe intrinsic multi-level correlation within the kernel that we can leverage for memory compression. This memory compression directly translates to latency/power improvement since convolutions have frequent weight access, whose memory cost is even higher than feature maps [22].

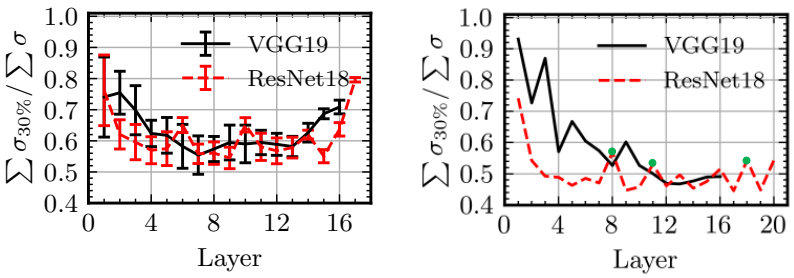


Figure 2.43: Convolutional kernel correlations in ImageNet-pretrained models are shown by the proportion of the sum of the top 30% singular values ($\sum \sigma_{30\%}$). (a) Intra-kernel correlations averaged on different kernels. Error bars show the $\pm\sigma$ variance. We skip 1×1 Conv. (b) Cross-kernel correlations, where green dots are 1×1 Conv.

2.5.2.1 Multi-Level Weight Generation

Intra-Kernel Correlation We first explore the low-rank property among different channels of a kernel. The i-th kernel $\boldsymbol{W}_i \in \mathbb{R}^{C_i \times k^2}$ can be treated as a matrix with C_i row vectors with length k^2. From its singular values $\boldsymbol{\Sigma} = \texttt{SVD}(\boldsymbol{W}_i) = \texttt{diag}(\sigma_0, \sigma_1, \cdots)$, we observe relatively strong correlations between those column vectors since the first several major components $\sigma_{30\%}$ concentrates the majority of the total values. Figure 2.43(a) shows the intra-kernel low-rank property of modern CNNs. Different layers tend to have different intra-kernel correlations, and shallower layers show higher correlations. This provides us an opportunity to generate the i-th kernel $\boldsymbol{W}_i \in \mathbb{R}^{C_i \times k^2}$ using a low-dimensional *channel basis* $\boldsymbol{W}_i^b \in \mathbb{R}^{B_i \times k^2}$ with a cardinality of $B_i < \min(C_i, k^2)$ and a corresponding coefficient matrix $\boldsymbol{U}_i \in \mathbb{R}^{C_i \times B_i}$. Figure 2.44 visualizes the procedure for convolutions with a general matrix multiplication (GEMM) interpretation using the *im2col* algorithm [16]. This intra-

kernel generation is formally expressed as.

$$\boldsymbol{W}_i = \boldsymbol{U}_i \boldsymbol{W}_i^b, \quad \forall i \in [C_o] \tag{2.54}$$

Therefore, we reduce the parameter of the i-th kernel from $|\boldsymbol{W}_i| = C_i k^2$ to $|\boldsymbol{W}_i^b| + |\boldsymbol{U}_i| = B_i k^2 + C_i B_i$. Note that for 1×1 convolution, we skip this intra-kernel generation and directly use all C_i channels given the constraint $B_i < \min(C_i, 1^2)$.

Cross-Kernel Correlation Furthermore, we explore the second-level correlation across C_o kernels. We view the entire convolutional kernel $\boldsymbol{W} \in \mathbb{R}^{C_o \times (C_i k^2)}$ as a matrix with C_o row vectors with length of $C_i k^2$. Figure 2.43(b) quantifies the correlation among different kernels. Though it is slightly weaker than the intra-kernel correlation, it still brings another opportunity to further decompose the weight along another dimension. Instead of generating C_o kernels independently, we only generate a subset of kernels as our *kernel basis* $\boldsymbol{W}_c = \{\boldsymbol{W}_i \in \mathbb{R}^{C_i k^2}, \forall i \in [B_c], B_c < \min(C_o, C_i k^2)\}$ using Eq. (2.54). This generated kernel basis $\boldsymbol{W}_c$ is used to span the entire kernel together with another coefficient matrix $\boldsymbol{V} \in \mathbb{R}^{C_o \times B_c}$ as follows,

$$\boldsymbol{W} = \boldsymbol{V}\boldsymbol{W}_c = \boldsymbol{V}\{\boldsymbol{U}_i \boldsymbol{W}_i^b\}_{i \in [B_c]}, \tag{2.55}$$

If $B_c \geq \min(C_o, C_i k^2)$, we only consider intra-kernel correlation by setting $B_c = C_o$ without performing Equation (2.55). After the proposed two-level generation, the parameter compression ratio is,

$$r = \frac{|\boldsymbol{V}| + \sum_{i \in [B_c]} (|\boldsymbol{U}_i| + |\boldsymbol{W}_i^b|)}{|\boldsymbol{W}|} = \frac{\big(C_o + B_i k^2 + C_i B_i\big) B_c}{C_o C_i k^2}. \tag{2.56}$$

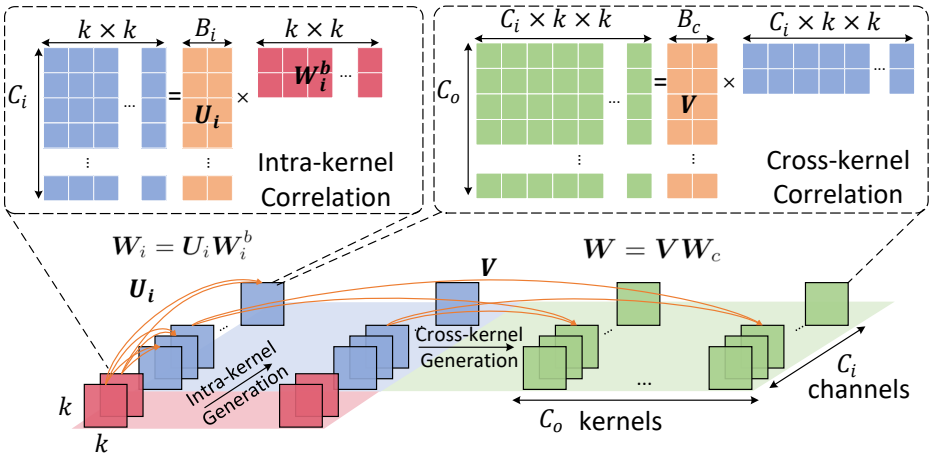


Figure 2.44: Intra-kernel and cross-kernel generation.

The extra computation for *in situ* kernel generation $\mathcal{O}(2B_cC_iB_ik^2+2C_oB_cC_ik^2)$ is marginal compared with the convolution itself $\mathcal{O}(2C_oC_ik^2HW)$, where H and W are output feature map sizes. Thus the runtime overhead is negligible, consistent with what we showed before in Figure 2.42(d). In this way, we successfully save expensive memory transactions with marginal computation overhead, which fully leverages the emerging accelerators' ultra-fast computing capability to mitigate the critical memory bound.

2.5.2.2 Augmented Mixed-Precision Generation

Besides the weight correlation that explores parameter-level reduction, we further explore the bit-level redundancy with mixed-precision bases. Modern NN accelerator designs, especially emerging analog engines, prefer to use low-bit weights to reduce memory access latency and simplify the control circuitry complexity [258, 160, 171, 74, 254]. In this section, we utilize the pre-

cision preserving feature of analog engines and propose an augmented mixed-precision generation strategy to recover high-precision weights with low-bitwidth basis and coefficients.

We assume the bitwidths for $\boldsymbol{W}_i^b$, $\boldsymbol{U}_i$, and $\boldsymbol{V}$ are q_b, q_u, and q_v, respectively. The first-level intra-kernel generation is capable of generating $\boldsymbol{W}_c \in \mathbb{R}^{B_c \times (C_i k^2)}$ with at most $(2^{q_b}-1)(2^{q_u}-1)B_i + 1$ possible distinct values, which corresponds to a bitwidth upper bound $\sup(q_c) = (q_b + q_u + \log_2 B_i)$. Unlike digital cores, this precision can be maintained by the direct cascade of two analog tensor units without resolution loss caused by the analog-to-digital conversion. Then, the cross-kernel generator will output $\boldsymbol{W}$ with an equivalent bitwidth $\sup(q) = (q_v + \sup(q_c) + \log_2 B_o)$ that can also be preserved in the matrix multiplication unit. The advantages are clear that our method enables the weight generator to be completely in the analog domain to recover a high-precision, i.e., $q > q_b, q_u, q_v$, weight matrix using low-precision basis and coefficient matrices. The memory compression ratio r_m is thus calculated as,

$$\begin{aligned} r_m &= \frac{\sum_{i\in[B_c]} \left(q_b|\boldsymbol{W}_{\boldsymbol{i}}^b| + q_u|\boldsymbol{U}_i|\right) + q_v|\boldsymbol{V}|}{q_w|\boldsymbol{W}|} \\ &= \frac{B_c B_i k^2 q_b + B_c C_i B_i q_u + C_o B_c q_v}{C_o C_i k^2 q_w}. \end{aligned} \tag{2.57}$$

Hence, given a target q_w, we can explore fine-grained mixed-precision settings of q_b, q_u, and q_v to further cut down the memory cost in the bit-level, which is an orthogonal technique to the above parameter-level counterparts.

Algorithm 2 Training with *in situ* generation

Input: A pretrained teacher $\widehat{\mathcal{M}}$ with weights $\widehat{\boldsymbol{W}}$, a student model $\mathcal{M}$ with $\boldsymbol{W}_i^b$, $\boldsymbol{U}_i$, and $\boldsymbol{V}$, mixed-precision bitwidths q_b, q_u, and q_v, training dataset $\mathcal{D}^{trn}$, total iterations T, initial step size η^0;

Output: Converged student model;

Step 1: ℓ_2 Initialization from the teacher model

$\boldsymbol{W}_i^b, \boldsymbol{U}_i, \boldsymbol{V} \leftarrow \text{argmin } \|\widehat{\boldsymbol{W}} - \boldsymbol{V}\{\boldsymbol{U}_i \boldsymbol{W}_i^b\}_{i\in[B_c]}\|_2^2$

Step 2: Quantization-aware knowledge distillation

for $t \leftarrow 0 \cdots T-1$ **do**

  Randomly sample a mini-batch $\mathcal{I}^t$ from $\mathcal{D}^{trn}$

  $\boldsymbol{U}_i^{t+1} \leftarrow \boldsymbol{U}_i^t - \eta^t \nabla_{\boldsymbol{U}_i}(\mathcal{L}_{KD} + \lambda \mathcal{L}_{ort}),\ \forall i \in [B_c]$

  $\boldsymbol{W}_i^{b,t+1} \leftarrow \boldsymbol{W}_i^{b,t} - \eta^t \nabla_{\boldsymbol{W}_i^b}(\mathcal{L}_{KD} + \lambda \mathcal{L}_{ort}),\ \forall i \in [B_c]$

  $\boldsymbol{V}^{t+1} \leftarrow \boldsymbol{V}^t - \eta^t \nabla_{\boldsymbol{V}_i}(\mathcal{L}_{KD} + \lambda \mathcal{L}_{ort})$

  $\eta^{t+1} = \texttt{Update}(\eta^t)$ ▷ Step size decay

end for

2.5.2.3 Training with *in-situ* Weight Generation

Our main target is to reduce memory cost with acceptable accuracy loss. Now we introduce how to optimize the designed CNN with *in situ* generators such that the desired accuracy can be achieved. We adopt a two-stage quantization-aware knowledge distillation to train our proposed NN, described in Alg. 2. Firstly, we obtain a pre-trained full-precision model without *in situ* generation as our teacher model $\widehat{\mathcal{M}}$ whose weight matrix is denoted as $\widehat{\boldsymbol{W}}$. Our low-rank mixed-precision model is the corresponding student model $\mathcal{M}$ whose weight matrix $\boldsymbol{W}$ is generated by quantized $\boldsymbol{W}_i^b$, $\boldsymbol{U}_i$, and $\boldsymbol{V}$. A differentiable quantizer [258] is used in our quantization-aware training. For simplicity, we omit the quantization notation for quantized $\boldsymbol{W}_i^b$, $\boldsymbol{U}_i$, and $\boldsymbol{V}$ if mixed-precision quantization is used. Then we let the student mimic the teacher using a two-

stage training algorithm. First, we solve the following problem to project the teacher model onto the student parameter space by minimizing their ℓ_2 distance,

$$\min \ \|\widehat{\mathcal{M}}(\widehat{\boldsymbol{W}}) - \mathcal{M}(\boldsymbol{W})\|_2^2 \approx \|\widehat{\boldsymbol{W}} - \boldsymbol{V}\{\boldsymbol{U}_i \boldsymbol{W}_i^b\}_{i\in[B_c]}\|_2^2. \tag{2.58}$$

Given the smoothness of $\mathcal{M}$ and $\widehat{\mathcal{M}}$, the above ℓ_2 distance can be approximated by the first-order term of its Taylor expansion. This ℓ_2 distance-based subspace projection is an effective and efficient initialization method for the student model. Then we try to find local optima in the low-rank space starting from this projected solution point. Therefore, in the second stage, we train the student model with knowledge distillation [91] as,

$$\begin{aligned}
\min \ & \mathcal{L}_{KD} = \beta T^2 \mathcal{D}_{KL}(q_T, p_T) + (1-\beta) H(q, p_{T=1}), \\
\text{s.t.} \ & p_T = \frac{\exp(\frac{\mathcal{M}(\boldsymbol{W})}{T})}{\sum \exp(\frac{\mathcal{M}(\boldsymbol{W})}{T})}, q_T = \frac{\exp(\frac{\widehat{\mathcal{M}}(\widehat{\boldsymbol{W}})}{T})}{\sum \exp(\frac{\widehat{\mathcal{M}}(\widehat{\boldsymbol{W}})}{T})}, \\
& \boldsymbol{W} = \boldsymbol{V}\{\boldsymbol{U}_i \boldsymbol{W}_i^b\}_{i\in[B_c]}, \\
& 0 < B_i < \min(C_i, k^2), B_i \in \mathbb{Z} \\
& 0 < B_c < \min(C_o, C_i k^2), B_i \in \mathbb{Z},
\end{aligned} \tag{2.59}$$

where $\mathcal{M}(\boldsymbol{W})$ is the output logits, $\mathcal{D}_{KL}$ is the Kullback–Leibler divergence between two probability distributions, $H(\cdot,\cdot)$ is the cross entropy, q is the ground truth distribution, T and β are hyper-parameters controlling the smoothness. This training method [91] can distill the representability of the high-rank full-precision model to our low-rank quantized student.

However, we notice that once the basis and coefficient matrices have a deficient row-rank or column-rank, the spanning subspace of the generated

matrix will become too small to approximate the original full-rank matrix. Therefore, to maximize the rank of the spanned weight matrix, we set a row orthonormality constraint to the basis $\boldsymbol{W}_i^b$ and a column orthogonality constraint to the coefficient matrices. This constraint can be relaxed using penalty methods as a multi-level orthogonal regularization term $\mathcal{L}_{ort}$ as follows,

$$\begin{aligned} &\sum_{i=1}^{B_c}\Big(\|\boldsymbol{W}_i^b(\boldsymbol{W}_i^b)^T-\boldsymbol{I}\|_2^2+\|\tilde{\boldsymbol{U}}_i^T\tilde{\boldsymbol{U}}-\boldsymbol{I}\|_2^2\Big)+\|\tilde{\boldsymbol{V}}^T\tilde{\boldsymbol{V}}-\boldsymbol{I}\|_2^2, \\ &\tilde{\boldsymbol{U}}_i=\left(\frac{u_0}{\|u_0\|_2^2}\cdots\frac{u_0}{\|u_{B_i-1}\|_2^2}\right),\tilde{\boldsymbol{V}}=\left(\frac{v_0}{\|v_0\|_2^2}\cdots\frac{v_0}{\|v_{B_c-1}\|_2^2}\right). \end{aligned} \tag{2.60}$$

Equation (2.60) is a generalization to a previous single-level penalty [79, 229] and exerts a soft constraint to multi-level correlations such that the spanning space will not collapse to a low-dimensional subspace. Therefore, the overall loss function is $\mathcal{L}=\mathcal{L}_{KD}+\lambda\mathcal{L}_{ort}$.

2.5.2.4 Case Study: Silicon Photonics Implementation

We showcase a photonic implementation of the proposed *in situ* weight generator in Figure 2.45. We focus on a SOTA design based on micro-ring resonators [190]. Other accelerators can also benefit from our method as long as the multi-level correlation and precision-preserving properties hold.

After loading the lightweight basis and coefficient matrices from the local electrical buffer, two cascaded ultra-fast optical weight banks will achieve the first-level and second-level generation to obtain the final weights $\boldsymbol{W}$. Without intermediate storage, the analog weights are directly broadcast to all photonic tensor units via ultra-low-power optical interconnects [9] to perform the

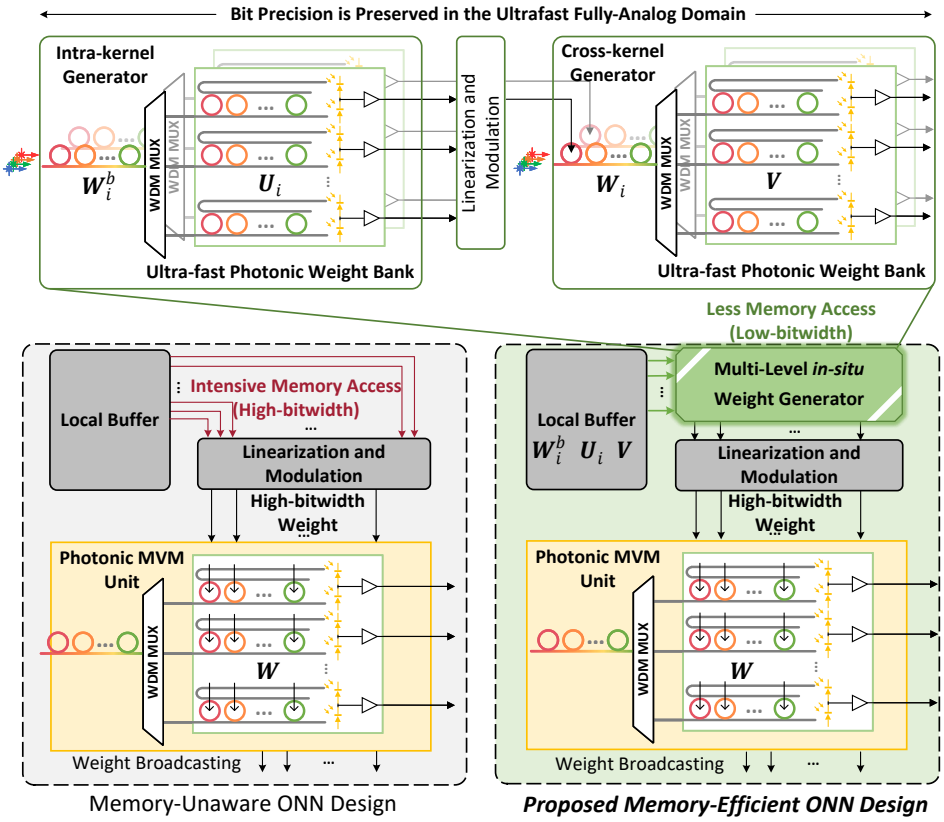
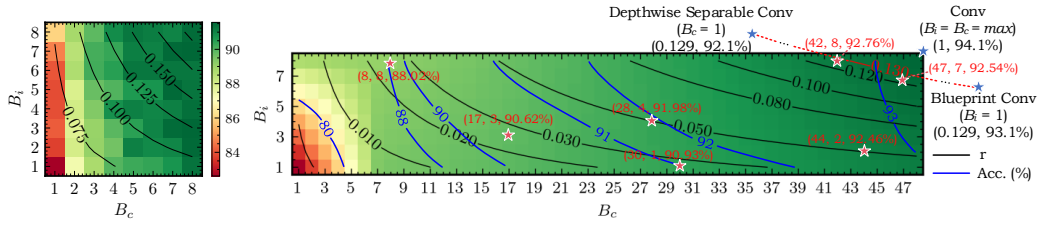


Figure 2.45: Photonic implementation of *in-situ* weight generator and peripheral structures.



(a) (b)

Figure 2.46: (a) Accuracy (color) and compression ratio (contour) of the customized 3-layer CNN on FashionMNIST [222] with various B_i and B_c (92.14% Acc. for the original Conv). (b) Accuracy (blue contour) and compression ratio r (black contour) for ResNet-18 on CIFAR-10. Red stars are representative settings of our method. Blue stars show previous designs.

primary operation, e.g., convolution. Compared with the memory-agnostic design, which requires massive and frequent weight loading, our proposed design can effectively cut down memory footprint and access latency. Consider a 16-bit (q_w=16) kernel $\boldsymbol{W} \in \mathbb{R}^{128\times128\times3\times3}$ and a setting $(B_i, B_c, q_b, q_u, q_v)$=(2,40,4,4,4) implemented by micro-rings of diameter R=20 μm, the extra latency introduced by *in situ* generator is as follows,

$$\begin{aligned}\tau_{gen} &= (\tau_{DAC} + \tau_{mod} + \tau_{prop1} + \tau_{oe}) + (\tau_{mod} + \tau_{prop2} + \tau_{oe}) \\ &\approx \tau_{DAC} + 2 \times (\tau_{mod} + \tau_{oe}) + \frac{4B_iR}{c} + \frac{4B_cR}{c} \\ &\approx 400 \text{ ps} + 2 \times (50 \text{ ps} + 10 \text{ ps}) + 25.2 \text{ ps} = 545.2 \text{ ps} \\ &\ll \frac{2(1-r_m)|\boldsymbol{W}|}{BW_{SRAM}} \approx \frac{(1-0.0272) \times 288 \text{ KB}}{34 \text{ GB/s}} = 7.9 \text{ μs},\end{aligned} \tag{2.61}$$

where τ_{DAC} is the latency for 10 Gb/s digital-to-analog converter, τ_{mod} is the device modulation delay, τ_{prop} is the photonic weight bank propagation delay, τ_{oe} is the optical-to-electrical conversion delay for layer cascade, c is the light speed, and BW_{SRAM} is the SRAM bandwidth [106]. The generator saves 7.9 μs latency (>97% of total weight load latency) with merely 545.2 ps weight generation latency overhead. Given ∼50% of total latency is consumed by kernel loading [22], our weight generation leads to at least 2× overall speedup. More speedup can be expected if activation quantization is further applied. In terms of power, our method can achieve significant energy reduction since we save $(1-r_m) \approx 97\%$ weight loading and replace all high-resolution DACs with $(1-r) \approx 89\%$ fewer low-bit DACs [166] (power is exponential to bitwidth), which account for most power as shown in Figure 2.42(a).

We further perform quantitative evaluation on a neuromorphic simulator MNSIM-2.0. On ResNet-18/ImageNet, compared with 8-bit `BSConv`, our method reduces the overall latency from 56.46 ms to 41.11 ms (**27.2%**↓), reduces the overall energy from 25.77 mJ to 3.69 mJ (**85.7%**↓), and improves energy-delay-product by **9.6**×.

2.5.3 Experimental Results

In this section, we first conduct ablation experiments on the proposed techniques and compare our method with prior efficient designs in memory cost and accuracy.

2.5.3.1 Dataset

Our ablation and comparison experiments are based on FashionMNIST [222], CIFAR-10 [112], and CIFAR-100. We also test on more tasks including SVHN [150], TinyImagetNet-200 [35], StanfordDogs-120 [107] and StandfordCars-196 [110] for fine-grained classification.

2.5.3.2 Neural Network Architectures

We first use a customized 3-layer CNN as a toy example to do multi-level correlation exploration on FashionMNIST, whose settings are (C32K5S2-C32K5S1-C32K5S1-AvgPool3-FC10), where C32K5S2 is a 5×5 convolution with 32 kernels and stride 2, AvgPool3 is an average pooling layer with output size 3×3, and FC10 means the output linear layer. BatchNorm and ReLU ac-

tivation are used between convolutional layers. Then, the rest ablation experiments and comparison experiments are based on ResNet-18 [1] [86], DenseNet-121 [2] [95], and MobileNetV2 [168], which are adapted to CIFAR-10/100.

2.5.3.3 Training Settings

We train all models for 200 epochs using RAdam [129] optimizer with an initial learning rate of 0.002, an exponential decay rate of 0.98 per epoch, and a weight decay of 5e-4. On CIFAR-10/100, images are augmented by random horizontal flips and random crops with 4 paddings. On TinyImageNet, StanfordDogs-120, and StanfordCars-196, additional color jitter is added. Mini-batch sizes are 64, 128, 64, and 64 for our 3-layer CNN, ResNet-18, DenseNet-121, and MobileNetV2, respectively.

2.5.3.4 Ablation: Multi-Level Correlation Exploration

To explore the impact of the multi-level basis cardinality B_i and B_c on the parameter count and accuracy, we first perform a grid search on Fashion-MNIST with our customized 3-layer CNN, shown in Figure 2.46(a). In terms of parameter compression ratio r, B_c shows a stronger impact than B_i since $r \propto B_c$ while B_i only partially contributes to r. For test accuracy, generally larger B_i and B_c lead to higher accuracy. However, the accuracy is much more sensitive to B_c than B_i, where we find a great opportunity to minimize

[1] https://github.com/kuangliu/pytorch-cifar
[2] https://github.com/gpleiss/efficient_densenet_pytorch

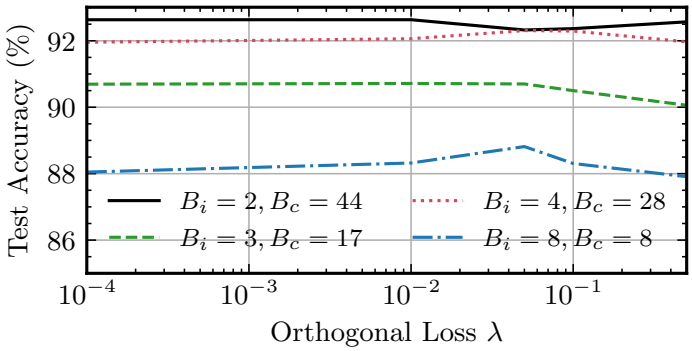


Figure 2.47: Exploration of different orthogonal regularization weights with ResNet-18 on CIFAR-10 [112].

memory cost with a small accuracy drop. Therefore, we conclude a heuristic design guidance that a small B_i and medium B_c leads to sweet points. We further validate it on CIFAR-10 with ResNet-18, whose contours are shown in Figure 2.46(b). In the design space exploration, we also plot full-rank Conv, depthwise separable Conv [86], and blueprint Conv [79] as our special cases. The blueprint Conv can be generalized by our method once B_i=1 and B_c=max. To some extent, separable Conv can also be generalized by setting B_i=max and B_c=1 while using different $\boldsymbol{V}$ for different input channels. Note that sharing $\boldsymbol{V}$ across channels is the key-point for our efficiency superiority. With the concluded design guidance, we indeed can quickly find design points that outperform the above prior works in memory efficiency with comparable accuracy, e.g., (B_i=2, B_c=44). Note that we assume a global (B_i, B_c) setting for all layers, while layer-specific cardinalities can be an interesting future topic to push toward the Pareto front.

Table 2.16: Accuracy evaluation on orthogonal regularization (*Ortho*), initialization (ℓ_2 and *SVD*), and knowledge distillation (*KD*). ResNet-18 is evaluated on CIFAR-10.

	Param Ratio r=0.025				Param Ratio r=0.05			
	B_i	B_c	B_i	B_c	B_i	B_c	B_i	B_c
	3	17	8	8	2	44	4	28
Baseline	90.62%		88.02%		92.46%		91.98%	
Ortho Reg	90.82%		88.52%		92.88%		92.32%	
SVD Init	91.32%		88.10%		93.05%		92.80%	
ℓ_2 Init	91.32%		88.85%		93.18%		92.75%	
ℓ_2+Ortho	91.40%		88.65%		93.17%		92.93%	
ℓ_2+Ortho+KD	**91.52%**		**88.96%**		**93.29%**		**93.19%**	

2.5.3.5 Ablation: Multi-Level Orthogonality Regularization

Several representative (B_c,B_i) pairs are evaluated on ResNet-18 CIFAR-10 with various regularization weights λ. Figure 2.47 reveals that the model performance can be consistently improved by 0.5%-1% with proper λ values ($0.01 \sim 0.05$). This shows that the proposed multi-level orthogonal penalty term can encourage the spanned kernel to be as high-rank as possible with augmented representability.

2.5.3.6 Ablation: Initialization and Distillation

We further evaluate different combinations of the proposed ℓ_2 initialization and knowledge distillation with representative (B_i, B_c) pairs in Table. 2.16. In our ℓ_2 initialization, we optimize Equation (2.58) using RAdam [129] for 3k iterations with lr=2e-2. We first compare with a traditional truncated singular value decomposition (SVD) based method [38, 229]. Both methods benefit accuracy while our ℓ_2 initialization demonstrates better results. With

orthogonality penalty and knowledge distillation (β=0.9, T=3), our method achieves the highest accuracy. In conclusion, a good initialization and knowledge from the teacher are critical to the accuracy of the student model.

2.5.3.7 Ablation: Mixed-Precision Bases Exploration

We perform a fine-grained investigation on the mixed-precision bitwidth (q_b, q_u, q_v) to justify the trade-off between accuracy and memory efficiency. For simplicity, we assume the same bitwidth combination for all layers. Figure 2.48 plots the accuracy-memory curve with equal q_b, q_u, and q_v. Above 3-bit, we can maintain over 93% accuracy (∼1% drop). Equal bit-precision for basis and

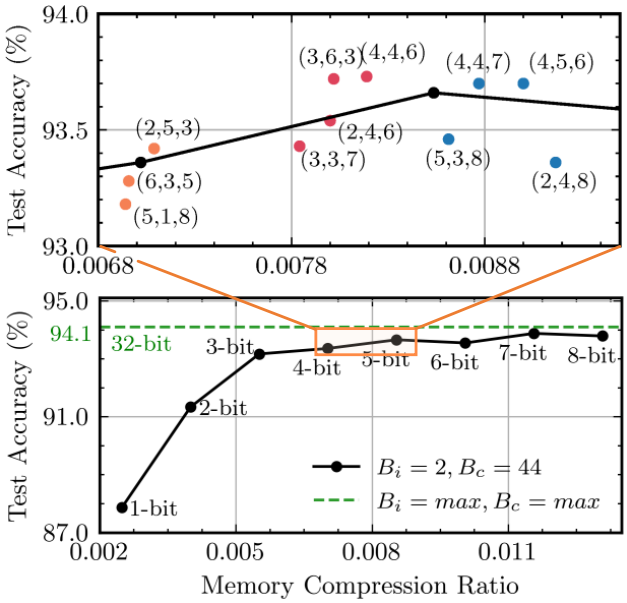


Figure 2.48: Accuracy and memory compression ratio contour of ResNet-18 on CIFAR-10 with mixed-precision quantization (q_b, q_u, q_v). Black dots show q_b=q_u=q_v.

Table 2.17: Comparison among efficient convolutions in terms of parameter/memory compression ratio (smaller is better) and accuracy. The cardinality d in PENNI is 2. CirCNN uses a block size k=4. (Ours-B_i-B_c-q_b-q_u-q_v) is the network setup.

	CIFAR-10			CIFAR-100		
	Param Ratio	Mem Ratio	Acc	Param Ratio	Mem Ratio	Acc
ResNet-18 (Conv) [86]	1.0000	1.0000	94.10%	1.0000	1.0000	73.53%
ResNet-18 (DSConv) [27]	0.1287	0.1287	92.10%	0.1323	0.1323	68.65%
ResNet-18 (PENNI d=2) [119]	0.2352	0.2352	92.77%	0.2383	0.2383	70.14%
ResNet-18 (BSConv) [79]	0.1291	0.1291	93.10%	0.1327	0.1327	71.11%
ResNet-18 (CirCNN k=4) [40]	0.2510	0.2510	92.16%	0.2541	0.2541	67.93%
ResNet-18 (Ours-2-44-32-32-32)	0.0497	0.0497	93.29%	0.0536	0.0536	70.85%
ResNet-18 (Ours-2-44-8-8-8)	0.0497	0.0131	**93.79%**	0.0536	0.0140	71.05%
ResNet-18 (Ours-2-44-3-6-3)	0.0497	**0.0080**	93.72%	0.0536	**0.0090**	**71.47%**
DenseNet-121 (Conv) [95]	1.0000	1.0000	94.69%	1.0000	1.0000	76.51%
DenseNet-121 (DSConv) [27]	0.7362	0.7362	93.81%	0.7396	0.7396	74.35%
DenseNet-121 (PENNI d=2) [119]	0.7608	0.7608	94.32%	0.7640	0.7640	75.26%
DenseNet-121 (BSConv) [79]	0.7291	0.7291	94.24%	0.7326	0.7326	**75.79%**
DenseNet-121 (CirCNN k=4) [40]	0.2601	0.2601	92.86%	0.2698	0.2698	72.45%
DenseNet-121 (Ours-1-25-32-32-32)	0.1986	0.1986	**94.89%**	0.2091	0.2091	75.09%
DenseNet-121 (Ours-1-25-8-8-8)	0.1986	0.0587	94.78%	0.2091	0.0612	75.59%
DenseNet-121 (Ours-1-25-4-6-6)	0.1986	**0.0395**	94.68%	0.2091	**0.0422**	75.05%

coefficients may not be the best combination. Thanks to our mixed-precision bit-level generation mechanism, we allow larger freedom to further explore different q_b, q_u, and q_v settings around a region of interest where the accuracy starts to drop. One key observation is that mixed-precision settings indeed can lead to higher accuracy with lower memory cost than equal settings. We also observe that relatively-balanced settings, e.g., (2,5,3), (4,5,6), generally outperform extremely-imbalanced ones, e.g., (5,1,8), (2,4,8). Hence, we claim that relatively-balanced mixed-precision bases are preferred to achieve better memory efficiency and less accuracy loss.

2.5.3.8 Comparison with Prior Work

Our method can serve as a memory-efficient drop-in substitution for normal convolutions. To show the superiority of our method over prior arts, we compare the memory compression ratio and inference accuracy with the baseline convolution (`Conv`) and four representative prior works, depthwise separable Conv (`DSConv`) [86], single-level low-rank decomposition (`PENNI`) [119], blueprint Conv (`BSConv`) [79], and block-circulant Conv (`CirCNN`) [40] on ResNet-18 and DenseNet-121 in Table. 2.17. For fair comparisons, all methods only apply to convolutional layers and use the same training settings as mentioned. To clarify, the selection of (B_i,B_c,q_b,q_u,q_v) is not from exhaustive enumeration but simply based on the target compression ratio and the heuristic design guidance we concluded. We only evaluate the unpruned `PENNI` version since pruning is an orthogonal technique to our method. We use a low-rank factor d=2 for `PENNI` [119] and a circulant block size k=4 for `CirCNN` [40] for a comparable memory cost and accuracy.

Compared with the baseline convolution, our 32-bit version achieves 5×-20× memory reduction. Compared with our special cases `DSConv` and `BSConv`, our method with a small B_i and a medium B_c shows 2×-4× memory reduction and comparable accuracy. Our multi-level generation outperforms the single-level low-rank decomposition method `PENNI` with 3.8×-4.7× lower memory cost and better accuracy. We outperform `CirCNN` in both metrics. With mixed-precision generation, we boost the memory efficiency by 25×-125× and 16×-19× over the baseline `Conv` and the best prior work `BSConv`

respectively, with competitive accuracy. Though on DensetNet-121 CIFAR-100, we have ∼0.7% accuracy drop, we have much lower memory cost. A larger B_c and higher bitwidths can be selected to recover the accuracy as a trade-off.

2.5.3.9 Boost Compact Models on Harder Tasks

To fully justify our superiority, we need to answer another three important questions: 1) how does it perform on architectures that are already compact; 2) is it compatible with activation quantization that is more memory bottlenecked; and 3) does the compressed low-rank kernel still have enough representability to capture critical features in high-resolution images. Similar to Figure 2.43, we also observe strong intra-kernel correlation for depth-wise Conv (`DWConv`) and cross-kernel correlation for point-wise Conv (`PWConv`). Hence we further apply our *in-situ* generation scheme to each individual `DWConv` and `PWConv` in the inverted residual block of MobileNet-V2 for further weight compression. Besides, we perform quantization to activation for each layer to save the most critical activation memory cost. Table 2.18 shows that we can further save >10× weight storage and reduce the largest activation memory cost by 4× even on compact architectures. On fine-grained image recognition tasks where the input images have high resolutions and low categorical variances, the compressed models still demonstrate strong model representability that can capture subtle but critical traits with negligible accuracy drop. Table 2.19 evaluates our methods further on searched compact networks on detection tasks, which are known to be energy/memory-demanding, our method

Table 2.18: *In-situ* generation with activation/weight quantization on MobileNetV2 [168]. The setup follows (`Ours`-B_i-B_c-q_b-q_u-q_v). *A8* means 8-bit activation. † means teacher models are initialized with ImageNet-pretrained models. The setup for TinyImageNet is (6-60-5-5-5).

	CIFAR-10		CIFAR-100	
	Mem Ratio	Acc	Mem Ratio	Acc
Original [168]	1.0000	93.06%	1.0000	73.90%
`Ours`-5-40-4-4-4	0.0783	94.03%	0.0867	73.11%
`Ours`-5-40-4-4-4 (A8)	0.0783	94.02%	0.0867	72.90%
	SVHN		TinyImageNet-200†	
Original [168]	1.0000	96.37%	1.0000	67.13%
`Ours`-5-40-4-4-4	0.0783	96.61%	0.1251	65.59%
`Ours`-5-40-4-4-4 (A8)	0.0783	96.63%	0.1251	65.44%
	StanfordDogs-120†		StanfordCars-196†	
Original [168]	1.0000	72.25%	1.0000	89.32%
`Ours`-5-40-4-4-4	0.0885	71.06%	0.0948	89.54%
`Ours`-5-40-4-4-4 (A8)	0.0885	71.42%	0.0948	89.47%

can lead to 5-12× compression with marginal performance loss.

Table 2.19: Evaluate compact models beyond simple tasks and classification.

	StanfordDogs-120		ImageNet-50		PASCAL VOC	
	Mem Ratio	Acc	Mem Ratio	Acc	Mem Ratio	mAP
MobilenetV2 (SSD-lite)	1.0000	72.25%	1.0000	87.56%	1.0000	0.683
`Ours` (SSD-lite)	0.0885	71.06%	0.0821	87.52%	0.1392	0.655
MobilenetV3-S (SSD-lite)	1.0000	65.41%	1.0000	85.04%	1.0000	0.544
`Ours` (SSD-lite)	0.2082	66.64%	0.2060	85.44%	0.2238	0.513
EfficientNet-B0	1.0000	75.43%	1.0000	89.56%	-	-
`Ours`	0.1257	75.00%	0.1132	88.52%	-	-

2.5.4 Summary

In this work, we focus on architecture-level optimization and propose a general and unified framework for memory-efficient photonic accelerator architecture designs via multi-level *in-situ* weight generation. We jointly leverage the intrinsic correlation and bit-level redundancy within convolutional ker-

nels and allow the ultra-fast accelerator to generate the weights *in situ* in the analog domain by itself to boost the performance. A photonic case study is given to show our latency/power advantages. Experiments show that our method achieves 10×-20× memory efficiency boost compared with prior methods. Our system-level solution provides a unified view of prior single-level low-rank methods and enables a new design paradigm to break through the ultimate memory bottleneck for emerging DNN accelerators by their tremendous computing power.

Chapter 3

In-situ Training for Self-Learnable Photonic Neural Engines

3.1 Introduction

Besides the hardware scalability and efficiency, the **adaptability or trainability** of photonic analog computing platforms is another major challenge in their practical application [252, 74, 261]. Especially for edge deployment scenarios, it is required that photonic AI engines be self-learnable and adaptable to non-ideal and changing environments or workloads. As a recent trend in most analog AI platforms, on-chip training becomes a promising solution to simultaneously solve the task performance degradation concern and adaptability issues.

The first reason to pursue on-chip training is its benefits on task performance and noise robustness. The mainstream approach is to offload the training process to electronic digital computers, obtain pre-trained weight matrices using classical back-propagation (BP) algorithm, and then map the trained model onto photonic hardware through matrix decomposition and unitary parametrization [171, 253]. Nevertheless, due to the analog computing nature of ONNs, the photonic DNN model inevitably suffers from

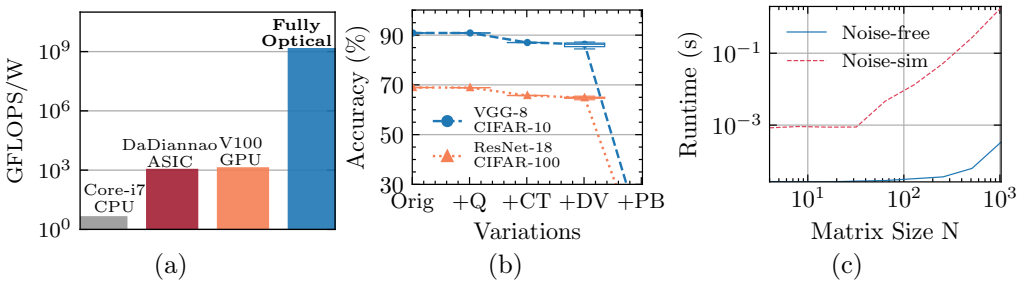


Figure 3.1: Comprehensive motivations. (a) Computational efficiency superiority of ONNs [171]. (b) Noise sensitivity of ONNs (Q: 8-bit quantization, CT: crosstalk, DV: device variation, PB: phase bias). (c) Runtime of noise-free matrix multiplication vs. w/ noise simulation (Q+CT+DV).

performance degradation or even complete malfunction [252, 261] with the existence of manufacturing errors, non-ideal device controls, and undesired circuit noises, shown in Figure 3.1(b). Thus the ONN model trained by pure software methods could suffer severe performance degradation and poor variation-robustness. Noise-aware training is one possible method to model the non-ideal effects during software training to mitigate the noise-induced performance drop. Though non-ideal effects can be simulated and considered during software training [252, 74], the variation simulation is physically inaccurate (especially with unknown process variations) and prohibitively expensive, shown in Figure 3.1(c), which might not pay off due to the inevitable simulation-reality gap. Training ONNs on the photonic chip *in situ* and deploying on the same hardware can make the model fully aware of the physical non-ideality, thus leading to the same task performance as the training results show. Also, it can benefit from the high speed and efficiency of the ultra-fast photonic computing engine.

The second reason to adopt on-chip training is its benefits on hardware adaptability. Once the working environments/condition drifts for the photonic computing systems, it require fast online adaptation to regain accuracy. Even with stable environments, once the workload needs to be switched or data distribution drifts, we still need on-device model fine-tuning or training from scratch. Once the on-chip training is realized, we can enable important edge learning applications, e.g., local online learning, transfer learning, lifelong learning, on-device adaptation, etc. This on-device self-learnability will maximize the adaptability of the computing engine to changing working environments and workloads and minimize communication costs and data privacy issues when communicating with remote cloud machines.

However, it is challenging to realize photonic on-chip training. As a unique hardware-restricted optimization problem, ONN *in-situ* learning encounters fundamental challenges causing scalability issues in prior methods:

- **Lack of full-observability for *in-situ* light field.** Tracking the physical optical field on every waveguide in $\boldsymbol{U}$ and $\boldsymbol{V}^*$ is not scalable or practical when ONNs scale up. Per device light field monitoring and calibration [64, 99] involves intractable hardware complexity. In practice, only $\boldsymbol{\Sigma}$ can be precisely monitored. Therefore, we assume no intermediate circuit states can be observed.

- **Limited input/output observability**. In photonic tensor cores, for efficiency consideration, only the final output signals after $\boldsymbol{U\Sigma V}^*$ can be

coherently detected. Intermediate signals of a single unitary projection can not be easily read out without extra hardware support.

- **Inaccessible gradients for most control variables.** Due to the above two limitations, it is challenging to use backpropagation or adjoint variable method to obtain first-order derivatives w.r.t. the MZI rotation phases in $\boldsymbol{U}$ and $\boldsymbol{V}^*$ [171, 72, 65], casting fundamental *in-situ* optimization difficulty as ONN scales up.

- **Randomness in noisy circuit evaluation**. The photonic analog circuits are noisy, which means the optimization methods must be robust and stable enough to handle the randomness to avoid divergence.

- **High training efficiency requirement**. The training algorithm must be simple and efficient enough to be able to execute on the photonic chip. Complicated gradient calculation and parameter update rules are not practical for implementation.

Customized learning algorithms are required to enable scalable ONN on-chip training.

In the rest of this chapter, Section 3.2 introduces a forward-only zeroth-order optimizer to enable on-chip training of 1,000 parameters with a built-in crosstalk handling mechanism. Section 3.3 presents a mixed-training strategy with a power-aware sparse coordinator descent optimizer to further boost the training scalability by 10× with over 90% training energy cost reduction.

Section 3.4 introduces a subspace optimization algorithm and a multi-level sparse training method to enable *in-situ* first-order partial gradient calculation and thus enable on-chip training of million-parameter ONNs with 1,000× scalability breakthrough and 30× training cost reduction.

3.2 FLOPS: Efficient On-Chip Learning for ONNs Through Stochastic Zeroth-Order Optimization

Previously, ONN on-chip training relied on a brute-force phase tuning algorithm that tunes each device one by one [171, 256]. Each MZI phase is individually perturbed by a small value and then updated based on the function evaluation results. This greedy parameter search algorithm is intractable when tuning a large number of phases and may not be robust enough to handle non-ideal variations. To mitigate the inefficiency issue of the above brute-force algorithm, an *in situ* adjoint variable method (AVM) [99] is applied to perform ONN on-chip training. This inverse design method directly computes the gradient w.r.t. MZI phases by propagating through the chip back and forth several times. However, it strictly assumes the photonic system is fully-observable and requires light field intensity measurement on each device, which

---

This FLOPS section is based on the following publication.

1. Jiaqi Gu, Zheng Zhao, Chenghao Feng, Wuxi Li, Ray T. Chen, and David Z. Pan, "FLOPS: Efficient On-Chip Learning for Optical Neural Networks Through Stochastic Zeroth-Order Optimization," ACM/IEEE Design Automation Conference (DAC), Jul. 2020.

I am the main contributor in charge of problem formulation, algorithm development, and experimental validations.

is technically challenging to scale to larger systems. Evolutionary algorithms, e.g., genetic algorithm (GA) and particle swarm optimization (PSO), are applied to train ONNs by emulating the biological evolution process [241] with a large population.

The above on-chip training protocols are only demonstrated to handle small photonic systems with <100 MZIs, while in this work, we propose a novel framework `FLOPS` that extends the learning scale to ∼1000 MZIs with higher training efficiency, higher inference accuracy, and better robustness to non-ideal thermal variations. Compared with the previous state-of-the-art methods [171, 256, 241], our on-chip learning framework `FLOPS` has the following advantages.

- **Efficiency**: our learning method leverages stochastic zeroth-order optimization with a parallel minibatch-based gradient estimator and achieves 3∼4× fewer ONN forward than previous methods.
- **Accuracy**: our proposed optimization algorithm is extended with a lightweight second-stage learning procedure `SparseTune` to perform sparse phase tuning, achieving further accuracy boost while the efficiency superiority still maintains.
- **Robustness**: our method is demonstrated to improve the test accuracy of ONNs under thermal cross-talk and produces better variation-robustness than previous methods.

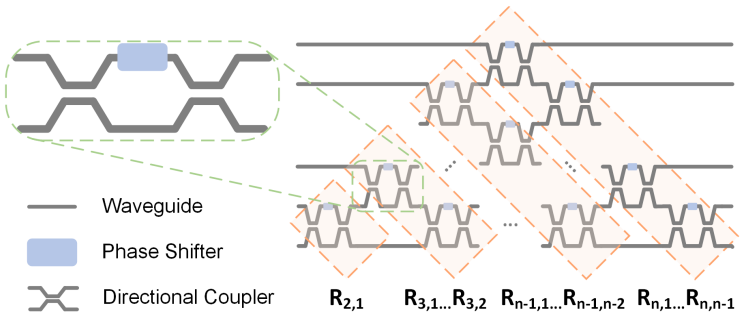


Figure 3.2: Schematic of an MZI triangular array and a closeup view of the MZI structure.

3.2.1 Preliminaries

In this section, we will introduce the architecture of integrated ONNs, current ONN training methods, and background knowledge about stochastic zeroth-order optimization with gradient estimation.

3.2.1.1 ONN Architecture and Training Methods

The integrated optical neural network (ONN) is a hardware platform that implements artificial neural networks with silicon photonics. As a case study, we focus on an ONN architecture based on singular value decomposition (SVD) [171]. It decomposes an $m \times n$ weight matrix using SVD, i.e., $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$. The diagonal matrix $\boldsymbol{\Sigma}$ can be simply implemented by on-chip attenuators, e.g., single-port Mach-Zehnder interferometers (MZIs), to perform signal scaling. The unitary matrices $\boldsymbol{U}$ and $\boldsymbol{V}^*$ can be realized by a cascaded MZI triangular array [161], shown in Fig. 3.2. The unitary group

parametrization is given by,

$$\boldsymbol{U}(n) = \boldsymbol{D} \prod_{i=n}^{2} \prod_{j=1}^{i-1} \boldsymbol{R}_{ij}(\phi_{ij}), \tag{3.1}$$

where $\boldsymbol{D}$ is a diagonal matrix with ± 1 on its diagonal entries, and the 2-dimensional planar rotator $\boldsymbol{R}_{ij}(\phi_{ij})$ is an n-dimensional identity matrix where entries on (i,i), (i,j), (j,i), (j,i) are $\cos\phi_{ij}$, $\sin\phi_{ij}$, -$\sin\phi_{ij}$, $\cos\phi_{ij}$, respectively. Each rotator $\boldsymbol{R}_{ij}$ can be implemented by a 2×2 MZI that produces unitary interference of input light signals with a rotation angle ϕ as follows [171],

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \tag{3.2}$$

To train ONNs, the traditional procedure trains the weight matrix $\boldsymbol{W}$ using gradient back-propagation and then maps it to photonic circuits through SVD and unitary group parametrization [161], which is inefficient and hardware-agnostic. Later, several ONN on-chip learning protocols are proposed to perform *in-situ* circuit optimization. To solve the problem, a straightforward approach is to compute the gradient w.r.t each MZI configuration given by,

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{R}_{ij}} &= \big(\boldsymbol{D}\boldsymbol{R}_{n1}\boldsymbol{R}_{n2}\boldsymbol{R}_{n3}\big)^T \nabla_y \mathcal{L}\ x^T \big(\cdots \boldsymbol{R}_{32}\boldsymbol{R}_{21}\boldsymbol{\Sigma}\boldsymbol{V}^*\big)^T \\ \frac{\partial \mathcal{L}}{\partial \phi_{ij}} &= \mathrm{Tr}\bigg(\Big(\frac{\partial \mathcal{L}}{\partial \boldsymbol{R}_{ij}} \odot \frac{\partial \boldsymbol{R}_{ij}}{\partial \phi_{ij}}\Big)(e_i + e_j)(e_i + e_j)^T\bigg), \end{aligned} \tag{3.3}$$

where e_i is the i-th orthonormal basis. On edge computing platforms, this analytical Jacobian is computationally-prohibitive, especially since $\nabla_y \mathcal{L}\ x^T$ is intractable in practical deployment. Later, a brute-force phase tuning method is

proposed [171, 257] using finite-difference-based gradient estimation. Adjoint variable method (AVM) [99] is proposed to model the circuit state as a partial-differential-equation-controlled linear system, and directly measures the exact gradient via *in situ* light intensity measurement. Evolutionary algorithms, e.g., particle swarm optimization and genetic algorithm, are demonstrated to train MZIs on chip [242]. However, a high-dimensional parameter space could lead to optimization inefficiency and unsatisfying optimality as they typically require a large population and inevitably faces pre-mature issues.

3.2.1.2 Optimization with Zeroth-Order Gradient Estimation

Zeroth-order optimization (ZOO) has received much attention recently [59, 130] as it enables effective optimization when the explicit gradient of the objective function is infeasible, e.g., reinforcement learning and black-box adversarial attack on DNNs. ZOO gains much momentum as it is capable of handling higher-dimensional problems than traditional methods, e.g., Bayesian optimization, and can be integrated with extensive state-of-the-art first-order optimization algorithms with the true gradient being replaced by the approximated zeroth-order gradient [59, 130]. ZOO uses a Gaussian function approximation to estimate the local value of the original function $f(\theta)$ as follows,

$$f_\sigma(\theta) = \mathbb{E}_{\Delta\theta}\Big[f(\theta+\Delta\theta)\Big] = \frac{1}{\sigma\sqrt{(2\pi)^d}}\int f(\theta+\Delta\theta)e^{\frac{-||\Delta\theta||_2^2}{2\sigma^2}}d\Delta\theta, \tag{3.4}$$

where $\Delta\theta$ is a perturbation sampled from Gaussian distribution $\mathcal{N}(0,\sigma^2)$. Nesterov *et al.* [148] proves a bound of the difference between the true gradient

and its Gaussian approximation if the original function $f(\theta)$ is β-smooth,

$$||\nabla f_\sigma(\theta) - \nabla f(\theta)|| \leq \frac{\sigma}{2}\beta(d+3)^{3/2}, \quad \forall \theta \in \mathbb{R}^d. \tag{3.5}$$

Based on this, we can approximate the true gradient $\nabla f(\theta)$ by estimating this surrogate gradient $\nabla f_\sigma(\theta)$ with the above error bound. To estimate $\nabla f_\sigma(\theta)$, a symmetric difference quotient method is typically used,

$$\hat{\nabla} f(\theta) = \frac{1}{2\sigma^2}(f(\theta + \Delta\theta) - f(\theta - \Delta\theta))\Delta\theta. \tag{3.6}$$

Such a sampling-based first-order oracle is an unbiased estimator of $f_\sigma(\theta)$. Passing this zeroth-order gradient estimation to a first-order optimization algorithm, e.g., gradient descent, we can perform optimization only with function queries.

3.2.2 On-Chip ONN Training based on Zeroth-order Gradient Estimation

3.2.2.1 Phase Domain Characterization

To better characterize the optimization problem of ONN on-chip training, we illustrate details of our optimization domain. Back-propagation based methods optimize in the *weight matrix domain*, while on-chip learning methods optimize in the latent *phase domain*. Bridged by SVD and unitary parametrization, the above two domains can switch to each other equivalently. However, co-optimization between those two domains is theoretically difficult. First, two domains are fully coupled and the transformation is highly-nonlinear and not element-wise. Any change in a phase represents a high-dimensional rotation

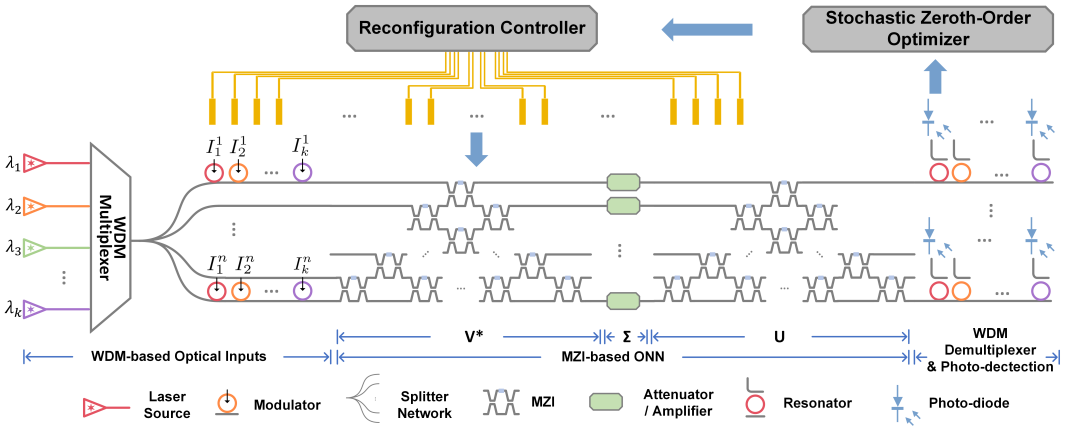


Figure 3.3: Framework of ONN on-chip training with stochastic zeroth-order optimization. Parallel signals with k different wavelengths are shown.

of the weight matrix, thus leading to perturbation of all entries in the weight matrix, and vice versa. Besides, the phase domain is not an unbounded space as the weight domain but a high-dimensional hypercube with a valid phase shift range of $[-\pi, \pi)$. Though this validity constraint can be guaranteed by projection onto a feasible set, it will cause optimization performance penalty. Since phases will intrinsically wrap around to the valid range, the solution space can also be viewed as a periodically expanded space, such that the validity constraint can be relaxed, shown in Fig. 3.5. This feature is leveraged in our optimization algorithm in later sections. The above analysis on phase domain characteristics casts both theoretical and practical difficulty on possible co-optimization between the weight matrix and the phase domain, which provides a strong motivation for us to design an efficient on-chip learning method directly in the phase space.

3.2.2.2 On-Chip Learning with Zeroth-Order Gradient Estimation

As shown in Fig. 3.3, an ONN consists of cascaded MZIs configured with external controls. We denote all programmable MZI phases as $\mathbf{\Phi}$. During ONN on-chip training, the final objective function $\mathcal{L}$ is optimized based on the following gradient descent formulation,

$$\mathbf{\Phi} \leftarrow \mathbf{\Phi} - \alpha \, \hat{\nabla}_{\mathbf{\Phi}} \mathcal{L}, \tag{3.7}$$

where $\hat{\nabla}_{\mathbf{\Phi}} \mathcal{L}$ is the zeroth-order estimation whose expectation approximates the true gradient $\nabla_{\mathbf{\Phi}} \mathcal{L}$ with an error bound shown in Eq.(3.5). Similar to Eq.(3.6), the stochastic zeroth-order gradient can be evaluated on a mini-batch as,

$$\hat{\nabla}_{\mathbf{\Phi}} \mathcal{L} = \frac{1}{2\sigma^2 |\mathcal{S}|} \sum_{i=0}^{|\mathcal{S}|} \big(\mathcal{L}(x_i; \mathbf{\Phi} + \mathbf{\Delta\Phi}_i) - \mathcal{L}(x_i; \mathbf{\Phi} - \mathbf{\Delta\Phi}_i) \big) \mathbf{\Delta\Phi}_i, \tag{3.8}$$

where x_i is one example in a mini-batch $\mathcal{S}$; $\mathbf{\Delta\Phi}_i$ is a random high-dimensional perturbation sampled from a multivariate Gaussian distribution $\mathcal{N}(0, \sigma^2)$. However, the optimization performance of stochastic gradient descent based methods highly depends on the accuracy of gradient estimation [130]. High variance in gradient estimation could generally lead to a slow convergence rate and degraded solution optimality. In this section, we will focus on the adopted techniques and theoretical analysis to improve ONN on-chip training efficiency and accuracy by reducing the computational and sampling complexity as well as minimizing the gradient estimation error.

On-Chip Learning Efficiency Improvement High efficiency is one of the major targets when performing ONN on-chip training. To efficiently leverage

the ultra-fast ONN hardware platform to estimate the zeroth-order gradient, we propose to use a parallel mini-batch-based asymmetric gradient estimator as follows,

$$\begin{aligned} \mathcal{L}_{\mathcal{S}}(x;\mathbf{\Phi}) &= \frac{1}{|\mathcal{S}|}\sum_{i=0}^{|\mathcal{S}|-1}\mathcal{L}(x_i;\mathbf{\Phi}), \\ \hat{\nabla}_{\mathbf{\Phi}}\mathcal{L} &= \frac{1}{\sigma^2}\big(\mathcal{L}_{\mathcal{S}}(x;\mathbf{\Phi}+\mathbf{\Delta\Phi}) - \mathcal{L}_{\mathcal{S}}(x;\mathbf{\Phi})\big)\mathbf{\Delta\Phi}, \end{aligned} \tag{3.9}$$

where $\mathcal{L}_{\mathcal{S}}(\cdot)$ averages the loss over a mini-batch. Two reasons account for the superior efficiency of the proposed estimator. First, a sample-efficient asymmetric estimator replaces its symmetric counterpart Eq. (3.8) [148] to achieve fewer function queries. Second, this estimation is performed in parallel with a fixed perturbation $\mathbf{\Delta\Phi}$ used for all examples within a mini-batch $\mathcal{S}$. Thus, this method is at least $|\mathcal{S}|$ times more efficient than the single-example-based method (Eq. (3.8)). Specifically, it eliminates expensive averaging operations over $|\mathcal{S}|$ length-d gradient samples, while only a cheap averaging operation on scalar loss functions is required. Moreover, our method shares the same $\mathbf{\Delta\Phi}$ in a mini-batch, leading to $|\mathcal{S}|$ times fewer Gaussian variable samples than its single-example-based counterpart.

From the perspective of hardware implementation, this parallel gradient estimation can be achieved by a readily available wavelength-division multiplexing (WDM) technique that enables fully parallel optical signal processing [177, 54]. As shown in Fig. 3.3, a mini-batch of input data, e.g., 16 or 32, can be encoded into parallel optical signals with $k = |\mathcal{S}|$ different wavelengths and then input into the ONN chip through the same waveguides.

Different output wavelengths can be filtered and separated by corresponding WDM de-multiplexer, and finally detected by photodiode arrays in the end.

Gradient estimation based on Eq. (3.9) costs only two function queries, thus can lead to convergence issues due to a large variance, especially with a larger dimension d. In the following section, we will focus on variance analysis and reduction techniques.

On-Chip Learning Accuracy Improvement To reduce the gradient estimation variance, we adopt its sample average with $Q+1$ function queries as follows,

$$\hat{\nabla}_{\mathbf{\Phi}}\mathcal{L} = \frac{1}{Q\sigma^2}\sum_{q=0}^{Q-1}\big(\mathcal{L}_{\mathbb{S}}(x;\mathbf{\Phi}+\mathbf{\Delta\Phi}_q)-\mathcal{L}_{\mathbb{S}}(x;\mathbf{\Phi})\big)\mathbf{\Delta\Phi}_q, \tag{3.10}$$

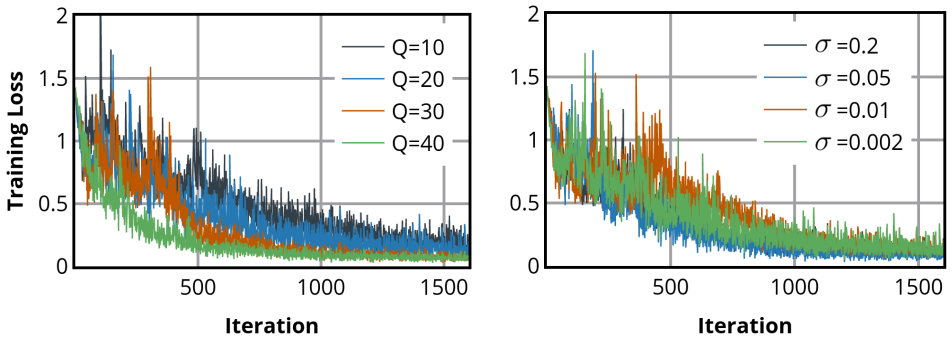
where the sampling factor Q is the number of independent perturbations used to calculate the sample average of gradient estimation.

We show the variance bound of this parallel mini-batch-based asymmetric zeroth-order gradient estimator. First, a standard assumption of stochastic gradient descent is given on the upper bound of the variance of the stochastic gradient on a mini-batch [148]. If the original function $\mathcal{L}$ is β-smooth, we can assume

$$\mathbb{E}_{\mathbb{S}}[||\nabla_{\mathbf{\Phi}}\mathcal{L}_{\mathbb{S}}-\nabla_{\mathbf{\Phi}}\mathcal{L}||^2]\leq\sigma_s^2. \tag{3.11}$$

Based on the above assumption, the variance upper bound of stochastic zeroth-order gradient estimator [148] is derived as,

$$\mathbb{E}_{\mathbb{S},\mathbf{\Delta\Phi}}[||\hat{\nabla}_{\mathbf{\Phi}}\mathcal{L}-\nabla_{\mathbf{\Phi}}\mathcal{L}_{\sigma}||^2]\leq\mathcal{O}\Big(\frac{\sigma^2\beta^2d^3}{Q}+\frac{\sigma_s^2d}{|\mathcal{S}|Q}+\frac{||\nabla_{\mathbf{\Phi}}\mathcal{L}||^2d}{Q}\Big). \tag{3.12}$$



(a) (b)

Figure 3.4: (a) Training curve with different sampling factor Q; (b) training curve with different sampling variances σ. A 3-layer ONN with configuration of 8-16-16-4 is used, where 16 represents 16 neurons at that hidden layer.

The above theoretical conclusion implies that increasing the sampling factor Q can effectively minimize the gradient estimation variance as illustrated in Fig. 3.4(a). Besides, a smaller sampling variance σ theoretically reduces the first term of the variance upper bound, but it is not sensitive within a wide range as observed in Fig. 3.4(b). Hence, during ONN on-chip training, the sampling factor, and mini-batch size are major tunable hyperparameters that can achieve a trade-off between gradient estimation error and function query complexity under certain ONN forward budget.

This zeroth-order gradient estimation based method quickly explores in the phase space till a roughly converged solution, as shown in Fig. 3.5, but it may still have some accuracy degradation due to stochastic gradient sampling error (Eq.(3.12)). If more function queries are allowed to recover the accuracy loss, we propose to extend this algorithm with `SparseTune`, a light-weight fine-tuning procedure based on random coordinate-wise phase tuning. The

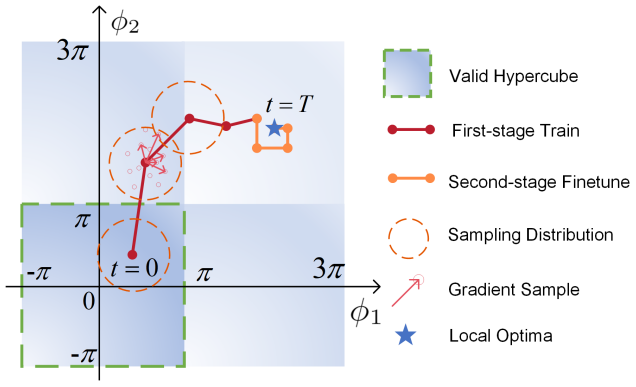


Figure 3.5: Optimization trajectory with the proposed on-chip training algorithm in the relaxed, periodic phase space.

complete two-stage on-chip learning algorithm is described in Alg. 3. At this second-stage `SparseTune`, a randomly selected subset of phases $\{\phi_i\}_{i=1}^{M}$ with cardinality M are sequentially tuned for each individual coordinate, shown in the second part of optimization trajectory in Fig. 3.5. This sparse tuning costs more function queries per iteration than the first stage as $M > Q$, but it is overall more efficient than brute-force training as $M \ll d$ and $T_f \ll T$. It is effective as it leverages the sparsity assumption in the parameter space [5] for variance reduction, thus can boost the optimization performance with less expensive parameter sweeping.

3.2.3 Robust ONN Learning with *in situ* Thermal Variation

As a high-performance analog neuromorphic platform, ONN inevitably encounters robustness issues that could lead to possible accuracy degradation, where thermal cross-talk is among one of the most critical concerns [171, 140]. In this section, we will justify the robustness advantages of our on-chip learning

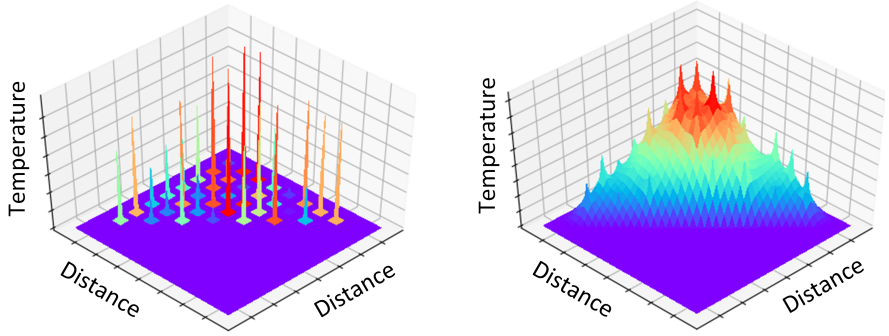
Algorithm 3 FLOPS+: ONN On-Chip Training With Zeroth-Order Optimization

Input: ONN forward function $\mathcal{L}(\cdot)$, initial MZI phases $\mathbf{\Phi}^0$, training dataset X, initial learning rate α^0, total iterations T, starting iteration for `SparseTune` T_f, cardinality of finetuned phases M, and initial tuning step size $\delta\phi^0$;
Output: Converged phases $\mathbf{\Phi}^{T-1}$;

for $t \leftarrow 0 \cdots T_f - 1$ **do** ▷ First stage training
  $\mathcal{L}_{\mathbb{S}}(x^t; \mathbf{\Phi}^t), \quad x^t \sim X$ ▷ ONN forward on a mini-batch
  $\{\mathbf{\Delta\Phi}_0^t, \cdots, \mathbf{\Delta\Phi}_{Q-1}^t\} \sim \mathcal{N}(0, (\sigma^t)^2 \mathbb{G}^t)$ ▷ Sample $\mathbf{\Delta\Phi}$
  $\hat{\nabla}_{\mathbf{\Phi}^t}\mathcal{L} = \frac{1}{Q(\sigma^t)^2} \sum_{q=0}^{Q-1} \big(\mathcal{L}_{\mathbb{S}}(x^t; \mathbf{\Phi}^t + \mathbf{\Delta\Phi}_q^t) - \mathcal{L}_{\mathbb{S}}(x^t; \mathbf{\Phi}^t)\big)\mathbf{\Delta\Phi}_q^t$
  $\hat{\mathbf{\Phi}}^t \leftarrow \mathbf{\Phi}^t - \alpha^t \hat{\nabla}_{\mathbf{\Phi}^t}\mathcal{L}$ ▷ Phase updating
  $\alpha^{t+1} = \texttt{Update}(\alpha^t)$ ▷ Learning rate decay
end for
for $t \leftarrow T_f \cdots T - 1$ **do** ▷ Second stage sparse tuning
  Randomly sample a mini-batch x^t from X
  Randomly select a set of phases $\{\phi_i\}_{i=1}^M \subseteq \mathbf{\Phi}^t$
  for each phase $\phi_i \in \{\phi_i\}_{i=1}^M$ **do**
    if $\mathcal{L}(x^t; \phi_i^t + \delta\phi^t) < \mathcal{L}(x^t; \phi_i^t)$ **then**
      $\phi_i^{t+1} \leftarrow \phi_i^t + \delta\phi^t$
    else
      $\phi_i^{t+1} \leftarrow \phi_i^t - \delta\phi^t$
    end if
  end for
  $\delta\phi^{t+1} = \texttt{Update}(\delta\phi^t)$
end for

method over traditional software training with thermal cross-talk.

Thermo-optic phase shifters are widely used to configure the MZI arrays on the ONN chip. Cross-talk exists among nearby devices, e.g., between two phase shifters or between phase shifters and waveguides, by influencing their relative refractive index n, which is difficult to accurately model in an efficient way for several reasons. First, due to heat propagation, the temperature at any point on the chip is fully correlated with others, which can be ideally modeled

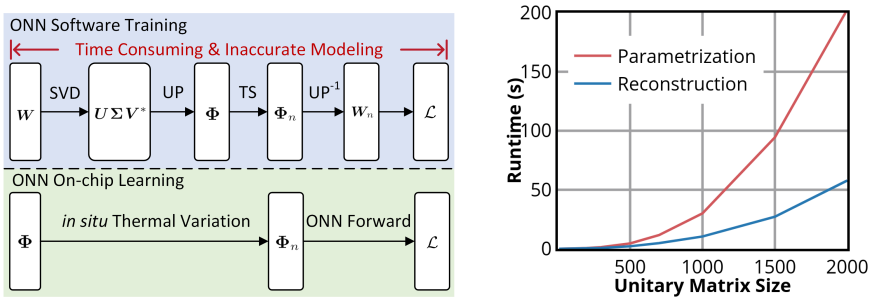


(a) (b)

Figure 3.6: Thermal variation simulation for a 9×9 MZI triangular array based on Poisson's equation. (a) Initial heat source distribution; (b) steady normalized temperature distribution.

with a Poisson's equation. Solving the steady temperature distribution of the whole chip will be time-consuming during training. Second, The heat source is not a single point but has a heat distribution along the physical dimensions of the device, which means different segments of the phase shifter will have different values of refractive index n under different temperatures. Hence the phase shift induced is given by the integral along the device dimensions. Third, the thermal impact from the phase shifters to neighboring waveguides is more complicated, as waveguides have different shapes, e.g., lines, curves, and circles. An accurate cross-talk model requires prohibitive computation that is rather challenging to consider as the chip scales up. In the presence of thermal cross-talk $\mathcal{T}(\cdot)$, ONN on-chip training can be formulated as optimizing in the projected phase space,

$$\mathbf{\Phi}^* = \underset{\mathbf{\Phi}}{\operatorname{argmin}} \mathcal{L}(x; \mathcal{T}(\mathbf{\Phi})), \tag{3.13}$$



(a) (b)

Figure 3.7: (a) Comparison between software training and on-chip learning. UP, TS, and UP^{-1} represent unitary parametrization, thermal simulation, and inverse unitary reconstruction, respectively. (b) Runtime cost of unitary parametrization and inverse reconstruction.

To resolve this robustness issue when optimizing Eq. (3.13), thermal variation can be modeled during ONN training. Back-propagation-based software training may encounter severe efficiency and effectiveness issues, as shown in Fig. 3.7(a). Modeling thermal variations during software training is time-consuming and inaccurate. It requires computationally-intensive SVD, unitary parametrization UP, and its inverse reconstruction UP^{-1} to switch between the weight matrix domain $\boldsymbol{W}$ and the corresponding phase domain $\boldsymbol{\Phi}$. To accurately obtain the projected phases $\boldsymbol{\Phi}_n = \mathcal{T}(\boldsymbol{\Phi})$, thermal variation with cross-talk simulation needs to be injected in each training iteration. This whole procedure suffers from inefficiency and poor scalability given the high computational cost and runtime cost of unitary parametrization and accurate thermal simulation. Our proposed on-chip learning method can inherently avoid those expensive domain transfer and inaccurate variation modeling via phase domain optimization with *in situ* thermal variation. Thus it can im-

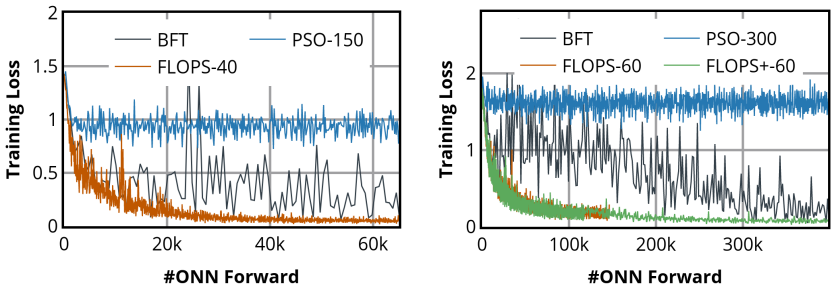
prove the ONN robustness with much higher learning efficiency than software training.

3.2.4 Experimental Results

To evaluate the effectiveness and efficiency of our proposed ONN on-chip learning algorithm, we compare inference accuracy and the number of ONN forward propagation with 1) brute-force phase tuning (BFT) [171, 256] algorithm, 2) particle swarm optimization (PSO) based on-chip training [241] algorithm, 3) our proposed algorithm with stochastic zeroth-order gradient estimation (FLOPS), and 4) our proposed algorithm with a second-stage sparse tuning (FLOPS+). Experiments are conducted on a Vowel Recognition dataset [39] to perform vowel phoneme classification. We implement all methods in PyTorch with an NVIDIA GTX 1080 GPU and an Intel Core i7-3770 CPU. We use two 3-layer ONN configurations in our experiments: 1) 8-16-16-4 and 2) 10-24-24-6, where 10 and 24 represent the input length and the number of neurons, respectively. We adopt a learning rate α=2 with an exponential decaying rate of 0.985, a sampling standard deviation σ=0.002, and a mini-batch size $|\mathcal{S}|$=32, which is technically realizable by modern WDM techniques.

3.2.4.1 ONN Training Method Comparison

In the comparison experiments, the brute-force on-chip phase tuning method (BFT)[171, 256] sequentially perturbs $|\mathbf{\Phi}| = d$ phases with a decaying perturbation $\delta\phi$, compare the perturbed loss function with the original one,



(a) (b)

Figure 3.8: (a), (b) are training curve comparisons among different methods with ONN configurations of 8-16-16-4, and 10-24-24-6, respectively. *BFT* is trained for 50 epochs, and other methods are trained for 200 epochs.

and update each phase according to,

$$\phi_i \leftarrow \begin{cases} \phi_i + \delta\phi, & \text{if } \mathcal{L}(x, \phi_i + \delta\phi) < \mathcal{L}(x, \phi_i) \\ \phi_i - \delta\phi, & \text{if } \mathcal{L}(x, \phi_i + \delta\phi) \geq \mathcal{L}(x, \phi_i) \end{cases} \tag{3.14}$$

The particle swarm optimization (`PSO`) initializes a population of P phase solutions $\{\mathbf{\Phi}_p\}_{p=1}^{P}$, and iteratively updates the population towards a randomly mixed direction between their globally best-performing solution and individually best solution after P function evaluations. Note that as mentioned in Section 3.2, the adjoint variable method (AVM) [99] is not compared because it requires expensive *in situ* light intensity measurement in the device-level such that it is technically intractable to realize on larger systems.

Based on the training curve comparison in Fig. 3.8, we notice a slow and unstable convergence for `BFT` method. We cut off the plot after certain function queries for clear demonstration, while `BFT` actually takes an extremely long time to converge. For `PSO`-based ONN on-chip training [241], we adopt an

experimentally optimal set of hyper-parameters for a fair comparison, where initial velocity is within [-2,2], inertia weight w=0.5, individual cognitive constant c_1=0.5, and social constant c_2=1. PSO stagnates at a poor saddle point in an early stage, which is hard to overcome since only zeroth-order oracle information is used [5]. Our proposed algorithm (FLOPS) quickly explores the phase space along the estimated zeroth-order gradient directions towards a relatively low training loss with cheap function query complexity. Extended with sparse tuning procedure, shown in Fig. 3.8(b), FLOPS+ takes longer to converge but effectively boosts the ONN learning performance, such that the accuracy gap is minimized compared to the best result.

Besides inference accuracy, we give theoretical analysis and practical measurements for the learning efficiency of the above four methods. BFT sequentially sweeps over all phases $\mathbf{\Phi} \in \mathbb{R}^d$, leading a query complexity of $\mathcal{O}(T\lambda d)$, where λ is the average number of function query at each tuning step. Assuming either case in Eq.(3.14) happens with the same probability, we estimate λ as $3/2$. PSO method performs practically well on small-scale ONN training (<100 MZIs) [241], but a population-related complexity $\mathcal{O}(TP)$ makes it query-inefficient when optimizing a larger number of phases. FLOPS is sample-efficient with a query complexity of $\mathcal{O}(TQ)$, where Q is practically much lower than either P or λd. FLOPS+ offers a controllable approach to trade off between efficiency and performance. It costs more ONN forward as $\mathcal{O}((T - T_f)Q + T_f\lambda M)$, while better solution optimality can be obtained, and the training efficiency advantages still hold. To validate the potential and

scalability of FLOPS, we estimate the runtime of on-chip training methods and BP-based software training with a 3-layer example ONN configuration of 200-500-500-10. As analyzed in Section. 3.2.3, BP-based software training suffers from computational inefficiency due to expensive domain transfer between $\boldsymbol{W}$ and $\boldsymbol{\Phi}$. The runtime breakdown for each software training iteration is estimated as,

$$\begin{aligned} t_{sw} &\approx t_{fp} + t_{svd} + t_{up} + t_{rec} + t_{bp} \\ &\approx 70ms + 200ms + 20s + 10s + 20ms \approx 30s \end{aligned} \tag{3.15}$$

where forward t_{fp}, backward t_{bp}, and SVD t_{svd} take around 300 ms. Unitary parametrization t_{up} and its inverse reconstruction t_{rec} takes the majority of the runtime. Figure. 3.7(b) shows that t_{up} and t_{rec} grow rapidly as the matrix size scales up. This runtime cost could be unaffordable once the thermal simulation is added. For the same ONN configuration, the runtime estimation of on-chip training is [195],

$$\begin{aligned} t_{oc} &\approx t_{prog} + t_{opt} + t_{pd} + t_{ad} + t_{iter} \\ &\approx 10\mu s + 1000ps + 20ps + 1ns + 1ms \approx 1ms, \end{aligned} \tag{3.16}$$

where t_{prog} is the thermal constant time for programming MZIs, t_{opt} is the propagation latency of optics through the 3-layer MZI arrays, t_{pd} is the photo-detection time with WDM de-multiplexing, t_{ad} accounts for the analog-to-digital conversion time, and t_{iter} adds the computation overhead for gradient calculation and phase updates. Given the sampling factor $Q \ll \frac{30s}{1ms}$, our proposed ONN on-chip learning method potentially benefits from order-of-magnitude learning efficiency improvement over the traditional software-based training.

Table 3.1: On-chip training methods comparison in terms of inference accuracy and number of ONN forward on a Vowel Recognition dataset. *PSO-150* represents a population of 150; *FLOPS-40* sets Q to 40. *FLOPS+-40*, *FLOPS+-60* are extended with `SparseTune` with M=200 and 400 respectively. Normalized number of ONN forward is also shown for efficiency comparison.

ONN Setup	Methods	Test Accuracy	#ONN Forward
8-16-16-4($\|\mathbf{\Phi}\| = 448$)	BFT [171, 256]	99.08%	268.8 K (4.1)
	PSO-150 [241]	56.89%	256.0 K (3.9)
	FLOPS-40	99.08%	65.6 K (1.0)
10-24-24-6($\|\mathbf{\Phi}\| = 960$)	BFT [171, 256]	99.38%	864.0 K (3.9)
	PSO-300 [241]	43.83%	720.0 K (3.3)
	FLOPS-60	95.06%	219.6 K (1.0)
	FLOPS+-60	98.17%	405.1 K (1.8)

3.2.4.2 On-chip Training under *in situ* Thermal Variation

To demonstrate the robustness of our proposed learning method in the presence of device thermal variation, we evaluate the impact of thermal cross-talk among MZIs on inference accuracy. Given that the phase shift is proportional to the temperature $\mathbf{\Delta\Phi} \propto \Delta T$ [84, 195], all phase shifts will increase since thermal cross-talk slows down the heat dissipation. We show the inference accuracy under thermal cross-talk for different methods in Fig. 3.9.

The pure software learning method achieves high accuracy under the variation-free case ($\sim$98%), but degrades by $\sim$5% when the thermal variation is considered. The brute-force phase tuning (`BFT`) method demonstrates higher inference accuracy than pure software training but still suffers from inefficiency due to a considerable amount of function queries. Particle swarm optimization (`PSO`) generally shows unsatisfying robustness against thermal variation, leading to less than 50% accuracy for both ONN setups. Our proposed method `FLOPS` naturally considers the thermal non-ideality during on-

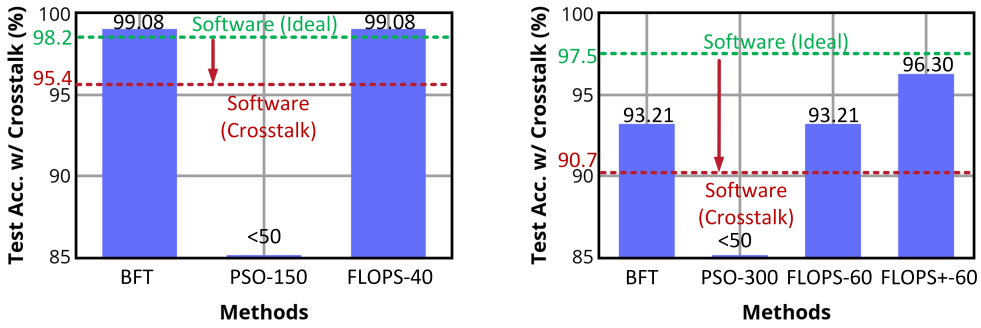


Figure 3.9: (a), (b) are accuracy comparisons under thermal cross-talk with ONN configurations of 8-16-16-4 and 10-24-24-6, respectively. The gap between *Software (Ideal)* and *Software (Crosstalk)* shows the accuracy drop caused by thermal cross-talk.

chip learning and demonstrates better robustness and better function query efficiency than previous works. After fast exploration of `FLOPS`, an extra `SparseTune` is considered to trade off between efficiency and performance. At the cost of more function queries, our two-stage zeroth-order optimization method `FLOPS+` achieves the best accuracy and robustness under thermal cross-talk while still more efficient than previous methods.

3.2.5 Summary

In this work, we propose a solution to enable efficient and robust ONN on-chip learning based on stochastic optimization with zeroth-order gradient estimation. A parallel mini-batch-based asymmetric gradient estimator `FLOPS` is adopted to leverage the ultra-fast parallel photonic chips to improve training efficiency as well as learning performance. Extended with a lightweight

sparse phase tuning SparseTune, a two-stage FLOPS+ is introduced to further boost the accuracy under thermal variation while still maintaining better efficiency than previous works. We give a theoretical analysis of the variance bound of FLOPS, function query complexity, and runtime comparison with other methods. Experimental results on a Vowel Recognition dataset with two ONN setups are demonstrated. Compared with the brute-force method and PSO-based method, our proposed framework FLOPS provides a 3∼4× more efficient on-chip learning protocol with better inference accuracy and robustness to thermal variation.

3.3 MixedTrain: Power-Aware Sparse Zeroth-Order Optimization for ONN On-Chip Learning

Training methodologies for integrated ONNs still lack a scalable and efficient solution so far. Previous works have the following disadvantages: 1) nontrivial Gaussian sampling cost, 2) divergence issues due to high variance, 3) high energy consumption, and 4) hardware-unfriendly weight update step size. In this work, we propose a novel mixed-training framework that enables scalable on-chip optimization with more stable convergence, higher training

---

This MixedTrain chapter is based on the following publication.

1. Jiaqi Gu, Chenghao Feng, Zheng Zhao, Zhoufeng Ying, Ray T. Chen, and David Z. Pan, "Efficient On-Chip Learning for Optical Neural Networks Through Power-Aware Sparse Zeroth-Order Optimization," Association for the Advancement of Artificial Intelligence (AAAI), Feb. 2021.

I am the main contributor in charge of problem formulation, algorithm development, and experimental validations.

efficiency, and much lower power consumption under a non-ideal environment. Compared with previous state-of-the-art (SOTA) methods, our mixed-training framework has the following advantages.

- Efficiency: our mixed-training strategy achieves 3∼7× fewer ONN forward and much lower computation complexity than SOTA ONN on-chip learning methods.
- Robustness: our method adopts a novel optical device mapping with mixed-active/passive regions to protect ONNs from device variations and thermal crosstalk, leading to better noise tolerance than previous solutions.
- Stability: our stochastic zeroth-order sparse coordinate descent optimizer (`SZO-SCD`) outperforms SOTA zeroth-order optimizers with more stable convergence and better performance in on-chip accuracy recovery.
- Scalability: our proposed optimizer leverages two-level sparsity in on-chip training, extending the ONN learning scale to >2,500 MZIs.
- Power: we propose a lightweight power-aware dynamic pruning technique, achieving >90% lower power consumption with near-zero accuracy loss or computation overhead.

3.3.1 Preliminaries

In this section, we will introduce the architecture of integrated ONNs, prior work in ONN on-chip training, and background knowledge about stochas-

tic zeroth-order optimization.

3.3.1.1 Stochastic Zeroth-Order Optimization

To solve optimization problems when analytical Jacobian is infeasible to compute, zeroth-order optimization (ZOO) plays a significant role, e.g., black-box adversarial attacks, policy-gradient-based reinforcement learning, and circuit parameter optimization [20, 21, 59, 130, 61, 248, 200, 205]. Various ZOO methods have been proposed with a mathematically-proven convergence rate, including stochastic gradient descent with Nesterov's acceleration [148], zeroth-order coordinate-wise Adam `ZADAM` and Newton's method `ZNewton` [20], zeroth-order adaptive momentum method `ZOO-AdaMM` [20], stochastic three-points [10], stochastic momentum three-points [61], etc. Most zeroth-order optimizers have a convergence rate dependent on dimensionality, which intrinsically makes them less efficient and less scalable than higher-order optimizers. In this work, we explore two-level sparsity in stochastic zeroth-order optimization to enable scalable and efficient ONN on-chip training.

3.3.2 Problem Formulation and Analysis

Before discussing our proposed on-chip learning framework, we give a formulation to the resource-limited ONN learning problem. In practical ONN applications, the ultimate target is to leverage photonic neural chips to complete machine learning tasks with high accuracy, low latency, and low energy consumption, under environmental changes and device-level variations.

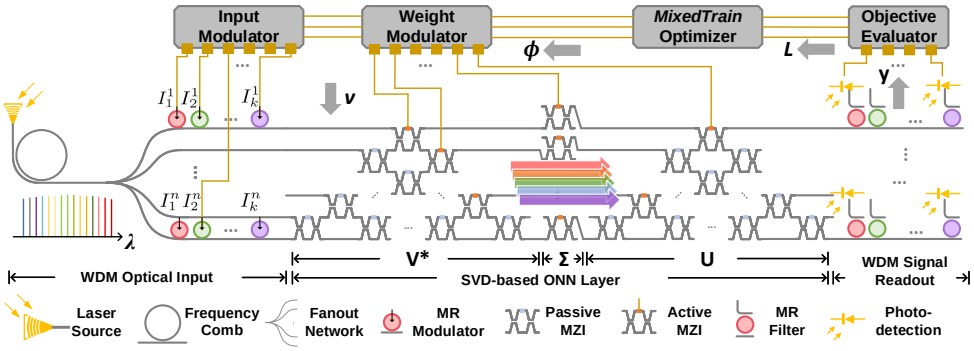


Figure 3.10: Schematic of ONN on-chip learning framework with stochastic zeroth-order mixed-training.

The optimization variables are optical device configurations, i.e., phase shift for all MZIs $\boldsymbol{\Phi}$, including those in the unitary matrices $\boldsymbol{U}$ and $\boldsymbol{V}^*$ and the diagonal matrix $\boldsymbol{\Sigma}$. The objective is the task-specific loss function. We are the first to formulate the ONN on-chip learning as a resource-limited accuracy recovery problem in the unitary space,

$$
\begin{aligned}
& \boldsymbol{\Phi}^* = \underset{\boldsymbol{\Phi}\sim\mathcal{R}}{\operatorname{argmin}} \mathcal{L}_{\mathcal{D}^{trn}}(\boldsymbol{W}(\boldsymbol{\Phi})), && (3.17)\\
\text{s.t.}\quad & \boldsymbol{W}(\boldsymbol{\Phi}) = \boldsymbol{U}(\boldsymbol{\Phi}^U)\boldsymbol{\Sigma}(\boldsymbol{\Phi}^{\boldsymbol{S}})\boldsymbol{V}^*(\boldsymbol{\Phi}^V),\\
& \boldsymbol{U}(\boldsymbol{\Phi}^U) = \boldsymbol{D}^U \prod_{i=N}^{2}\prod_{j=1}^{i-1} \boldsymbol{R}_{ij}(\phi_{ij}^U),\\
& \boldsymbol{V}^*(\boldsymbol{\Phi}^V) = \boldsymbol{D}^V \prod_{i=M}^{2}\prod_{j=1}^{i-1} \boldsymbol{R}_{ij}(\phi_{ij}^V),\\
& \|\boldsymbol{\Sigma}(\boldsymbol{\Phi}^S)\|_\infty < m,\\
& \boldsymbol{\Phi} \in [0, 2\pi),\\
& \texttt{Power}(\boldsymbol{\Phi}) \leq \tilde{P}, \int_t \texttt{Power}(\boldsymbol{\Phi}^t)\,\mathrm{d}t \leq \tilde{E},\\
& \mathcal{C}(\nabla_{\boldsymbol{\Phi}}\mathcal{L}) \gg \tilde{C} \gg \mathcal{C}(\mathcal{L}), \mathcal{C} \leq \tilde{C},
\end{aligned}
$$

where $\mathcal{D}^{trn}$ is the training set. In each layer, the weight matrix $\boldsymbol{W} \in \mathbb{R}^{M \times N}$ is constructed by $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{V}^*$, where $\boldsymbol{U}$ and $\boldsymbol{V}^*$ are constrained in the Stiefel manifold, and the ℓ_∞-norm of the diagonal matrix $\boldsymbol{\Sigma}$ is bounded by a empirically largest signal scaling range m. The optimization parameters $\boldsymbol{\Phi}$ are constrained in a hypercube within 0 degree and 2π degree. The photonic device programming power has to honor a maximum power budget $\tilde{P}$ during ONN inference. Also, the total energy used to program ONN devices during on-chip training is bounded by an energy budget $\tilde{E}$. The last constraint is the computation budget for the optimizer, which can not afford to calculate the Jacobian $\nabla_{\boldsymbol{\Phi}}(\mathcal{L})$ shown in Eq. (3.3), but the objective evaluation is ultra-fast with optics.

To solve the optimization problem on this SVD-based architecture, we directly optimize the decomposed matrices $\boldsymbol{U}$ and $\boldsymbol{V}^*$ within the Stiefel manifold. Previous work proposed Riemannian optimization [97], unitary regularization [253], and unitary projection [74] to satisfy the unitary constraints. In this work, we optimize the phases $\boldsymbol{\Phi}$ in the Reck-style unitary parametrization space to achieve minimum computation complexity. For the diagonal matrix, we optimize it as $\texttt{diag}(\boldsymbol{\Sigma}) = m(\cos\phi_0^S, \cdots, \cos\phi_{\min(M,N)-1}^S)$, such that the optimization variables can be unified with $\boldsymbol{\Phi}^U$ and $\boldsymbol{\Phi}^V$. To facilitate power optimization, we do not use the periodic phase space relaxation [72]. Instead, we wrap the phase within the valid hypercube by $\phi = (\phi \mod 2\pi)$ at each iteration to avoid unnecessary power once the updated phase exceeds 2π. To meet the computation budget, we will introduce a lightweight tech-

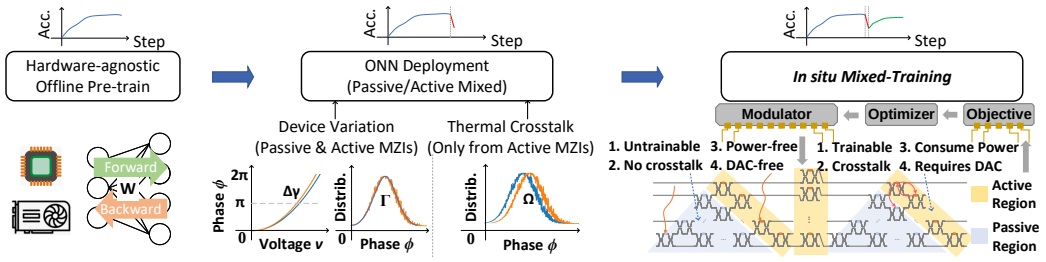


Figure 3.11: Mixed-training flow in the practical ONN deployment.

nique to handle power and energy constraints. The computation budget of the resource-constrained platform can be satisfied by manual searching in the optimizer design space, where lightweight zeroth-order optimization methods will be promising candidates.

3.3.3 Proposed Power-Aware Mixed-Training Framework

In the practical ONN deployment, apart from the basic constraints listed in Eq. (3.17), non-ideal environment and device noises are necessary to be considered in the learning framework. Therefore, we propose a mixed-training strategy to efficiently solve this noisy learning problem with all the aforementioned constraints in Fig. 3.11.

3.3.3.1 Scalable Mixed-Training Strategy

To enable efficient ONN on-chip learning on practical networks and datasets, we propose a mixed-training strategy to reduce the optimization dimensionality and minimize tunable devices for better convergence and lower power consumption. Specifically, we assume a model is pre-trained and pre-

pared for edge ONN deployment. Then, our target is to implement the pre-trained model on practical ONN engines while recovering the accuracy given non-ideal environment and device variations. The naive solution is to deploy a fully active ONN where all optical devices are thermo-optically tunable with maximum learnability [72]. However, This leads to high control complexity, power consumption, and non-ideal thermal crosstalk. In this work, we propose a mixed-training strategy that integrates passive and active ONNs to balance efficiency, robustness, and learnability, shown in Fig. 3.11. Now we introduce three stages in the entire ONN on-chip mixed training flow.

Hardware-Unaware Pre-training For the MZI-based ONN architecture, Hardware-unaware training based on back-propagation is firstly performed with an ideal computational model on digital electrical platforms, e.g., GPUs and CPUs, to obtain target device configurations.

ONN Deployment with Mixed Active/Passive Regions We proposed to deploy the ideally-trained model on the photonic circuits using a mixed passive/active design, where most parameters in two unitary matrices $\boldsymbol{U}$ and $\boldsymbol{V}^*$ are fixed by using passive optical devices. Only the diagonal matrix $\boldsymbol{\Sigma}$ and a small fraction ($\alpha \ll 1$) of phases in two unitary matrices are implemented by active devices to enable its adaptability and learnability. We denote fixed phases in the passive region as $\mathcal{P}$ and tunable phases in the active region as $\mathcal{A}$.

In practical applications, device variations, e.g., phase shifter γ coef-

ficient drift, and thermal crosstalk among MZIs will be present, leading to output perturbation and thus accuracy loss, shown in Fig. 3.11. The phase shifter variations come from environmental temperature changes or manufacturing errors. Under variations, the power-to-phase-shift factor γ of both active and passive phase shifters will drift from the ideal value as $\gamma^v = \gamma + \Delta\gamma$, where we assume the noise is sampled from a truncated Gaussian distribution $\Delta\gamma \in \mathcal{N}(0, \sigma_\gamma^2)$. We assume the rotation angle is proportional to the device-related coefficient as $\phi \propto \gamma$, and the noisy phase is denoted as $\phi^v = \phi\gamma^v/\gamma$. For N rotation angles, this variation is described as a diagonal perturbation matrix $\mathbf{\Phi^v} = \mathbf{\Gamma\Phi}$. In terms of thermal crosstalk, the correlated heat distribution among thermo-optic devices leads to an increase in the steady temperature. In the heat steady state, the mutual correlation of phases within N noisy rotation angles $\mathbf{\Phi}^v$ can be described by a coupling matrix as $\mathbf{\Phi}^c = \mathbf{\Omega\Phi^v}$,

$$
\begin{pmatrix} \phi_0^c \\ \phi_1^c \\ \vdots \\ \phi_{N-1}^c \end{pmatrix} = \begin{pmatrix} \omega_{0,0} & \omega_{0,1} & \cdots & \omega_{0,N-1} \\ \omega_{1,0} & \omega_{1,1} & \cdots & \omega_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{N-1,0} & \omega_{N-1,1} & \cdots & \omega_{N-1,N-1} \end{pmatrix} \begin{pmatrix} \phi_0^v \\ \phi_1^v \\ \vdots \\ \phi_{N-1}^v \end{pmatrix}
$$

$$
\begin{aligned}
\text{s.t.} \quad & \omega_{i,j} = 1, \quad \forall\, i = j \\
& \omega_{i,j} = 0, \quad \forall\, i \neq j \text{ and } \phi_j \in \mathcal{P} \\
& 0 \leq \omega_{i,j} < 1, \quad \forall\, i \neq j \text{ and } \phi_j \in \mathcal{A}.
\end{aligned} \tag{3.18}
$$

The diagonal factor $\omega_{i,j}, i = j$ is the self-coupling coefficient, which is typically set to 1. $\omega_{i,j}, i \neq j$ is the mutual coupling coefficient [140]. As a physical fact, only active devices are thermal aggressors that perturb adjacent devices, while

passive devices do not impact their neighbors since they have zero heat dissipation. Hence mutual coupling factors $\omega_{i,j}, i \neq j$ are set to 0 if ϕ_j represents a passive MZI. We can unify the γ noise with the crosstalk as $\mathbf{\Phi}^{\boldsymbol{c}} = \mathbf{\Omega\Gamma\Phi}$. Therefore, the objective is re-written as

$$\mathbf{\Phi}^* = \underset{\mathbf{\Phi}\sim\mathcal{R}}{\operatorname{argmin}} \mathcal{L}_{\mathcal{D}^{trn}}(\boldsymbol{W}(\mathbf{\Omega\Gamma\Phi})). \tag{3.19}$$

Mixed-Training with Stochastic Zeroth-Order Sparse Coordinate Descent (SZO-SCD) In this stage, we introduce stochastic zeroth-order sparse coordinate descent (`SZO-SCD`) to tune the active devices for *in situ* accuracy recovery. Since the pre-trained model is roughly converged, the ZO-gradient-based method [72] will suffer from divergence issues due to its gradient estimation variance. In contrast, our `SZO-SCD` optimizer is more suitable for near-convergence fine-tuning in the phase space. In iteration t, only a fraction ($s \ll 1$) of active devices $\mathbf{\Phi}_s = \{\phi_0, \cdots, \phi_{s|\mathcal{A}|-1}\} \subseteq \mathcal{A}$ are selected for coordinate descent as follows,

$$\phi_i^{t+1} \leftarrow \underset{\phi_i}{\operatorname{argmin}}\{\mathcal{L}_{\mathcal{J}^t}(\phi_i^t + \delta\phi), \mathcal{L}_{\mathcal{J}^t}(\phi_i^t - \delta\phi)\}. \tag{3.20}$$

The mini-batch evaluation of $\mathcal{L}_{\mathcal{J}}(\cdot)$ can be processed in parallel by using WDM [191, 54] shown in Fig. 3.10.

The advantages of the mixed-training strategy with `SZO-SCD` lie in several aspects. First, since the method reduces the tunable parameters of an $N \times N$ weight matrix from N^2 to $N + s\alpha N^2$ per iteration, the optimization efficiency will be considerably improved. Second, the passive ONN part

consumes nearly zero energy, leading to approximately (1-α) power saving. Third, the optimization dimensionality is reduced from the full $\mathcal{O}(N^2)$ space to a sparse subspace, which accelerates the convergence of our zeroth-order learning algorithm with a slimmed computation demand. Fourth, this method has $\mathcal{O}(1)$ computation complexity and $\mathcal{O}(1)$ memory complexity per iteration, which is nearly the cheapest optimizer in the design space.

3.3.3.2 Power-Aware Dynamic Pruning

On resource-limited edge applications, low power consumption will be a preferable feature to enhance endurance. We assume the power of an active phase shifter is proportional to the rotation angle $P \propto \phi$, then we can use the phase $\phi \in [0, 2\pi)$ as a fast device tuning power estimator. A straightforward approach to handle this power constraint is to use Lagrangian relaxation to add the power constraint in the objective as follows,

$$\begin{aligned} \boldsymbol{\Phi}^* &= \underset{\boldsymbol{\Phi} \sim \mathcal{R}}{\operatorname{argmin}} \mathcal{L}_{\mathcal{D}^{trn}}(\boldsymbol{W}(\boldsymbol{\Omega}\boldsymbol{\Gamma}\boldsymbol{\Phi})) + \lambda P(\boldsymbol{\Phi}) \\ P(\boldsymbol{\Phi}) &= \sum_{\phi \in \boldsymbol{\Phi}} (\boldsymbol{\phi} \mod 2\pi), \end{aligned} \tag{3.21}$$

and solve it using the alternating direction multiplier method (ADMM). However, the dual update for power optimization will cause convergence issues, which will be shown in our later experiment. To implicitly consider power constraints in the optimization, we propose a power-aware dynamic pruning technique to further boost power efficiency with stable convergence. The detailed power-aware optimization algorithm is described in Alg. 4. In lines 8-12, the optimizer will prune backward steps with probability p if the objective in-

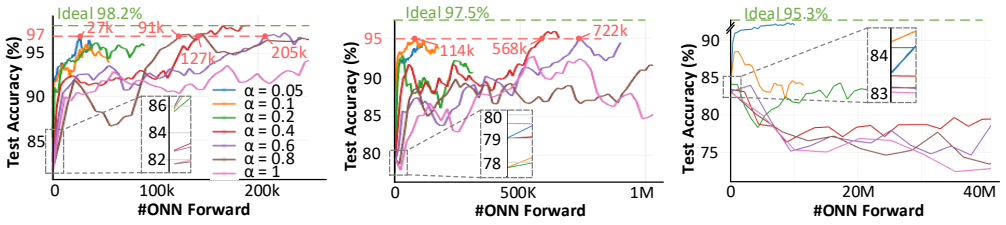
Algorithm 4 `SZO-SCD` with Power-aware Dynamic Pruning

Input: ONN forward function $\mathcal{L}(\cdot)$, phases $\mathbf{\Phi}^0$ after ONN deployment, training dataset $\mathcal{D}^{trn}$, total iterations T, Active set $\mathcal{A}$, sparsity of fine-tuned phases s, initial tuning step size $\delta\phi^0 > 0$, and power awareness $p \in [0, 1]$, power estimator `power`$(\cdot)$;

Output: Converged phases $\mathbf{\Phi}^{T-1}$;

for $t \leftarrow 0 \cdots T-1$ **do**
  Randomly sample a mini-batch $\mathcal{I}^t$ from $\mathcal{D}^{trn}$
  Randomly select $\mathbf{\Phi}_s^t = \{\phi_0^t, \cdots, \phi_{s|\mathcal{A}|-1}^t\} \subseteq \mathcal{A}$ without replacement
  for $\phi_i^t \leftarrow \phi_0^t, \cdots, \phi_{s|\mathcal{A}|-1}^t$ **do**
    if $\mathcal{L}_{\mathcal{I}^t}(\phi_i^t + \delta\phi^t) < \mathcal{L}_{\mathcal{I}^t}(\phi_i^t)$ **then**
      $\phi_i^{t+1} \leftarrow \phi_i^t + \delta\phi^t$
    else
      if `power`$(\phi_i^t - \delta\phi^t) >$ `power`(ϕ_i^t) **then**
        $b \sim \mathcal{B}(p)$ ▷ Sample from Bernoulli distribution with probability p to take 1
        $\phi_i^{t+1} \leftarrow \phi_i^t - b \cdot \delta\phi^t$
      else
        $\phi_i^{t+1} \leftarrow \phi_i^t - \delta\phi^t$
      end if
    end if
  end for
  $\delta\phi^{t+1} =$ `Update`$(\delta\phi^t)$ ▷ Step size decay
end for

creases in the positive direction and the power consumption increases in the negative direction. The intuition behind this efficient power-aware pruning is that our `SZO-SCD` only queries the zeroth-order oracle, such that the step-back is not guaranteed to be a descent direction. This uncertainty enables us to embed a power-constraint handling mechanism to dynamically prune step-backs that do not have a descent guarantee but lead to a certain power increase. The probabilistic power-awareness factor p also provides a parametric approach to



(a) (b) (c)

Figure 3.12: Test accuracy with different mixed-training sparsity α. (a) MLP (8-16-16-4) on Vowel Recognition, (b) MLP (10-24-24-6) on Vowel Recognition, and (c) MLP (8×8-24-24-10) on MNIST. Close-up views show the accuracy after deployment.

balance power and solution quality.

3.3.4 Experimental Results

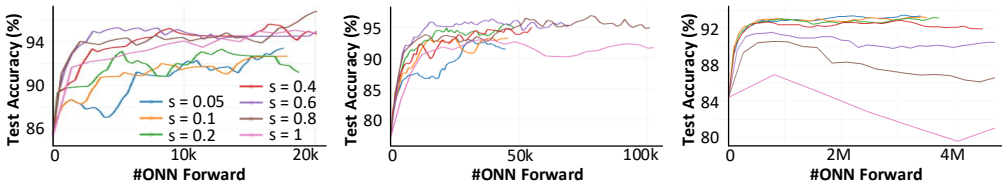
Experiments are conducted on a vowel recognition dataset [39], MNIST [115], FashionMNIST [222], and CIFAR-10 [112] for image classification. As a proof-of-concept demonstration, those datasets are standard and practical for ONNs. We implement all methods in PyTorch with an NVIDIA Quadro RTX 6000 GPU and an Intel Core i7-9700 CPU. We adopt a step size $\delta\phi$=0.02 with an epoch-wise exponential decaying rate of 0.985 and a mini-batch size of 32. The upper bound m set for $\mathbf{\Sigma}$ is 3. Following a common setting, the std. of phase variation $\sigma(\gamma)$ is set to 2e-3 and the mutual-coupling factor ω is set to 2e-3 only for adjacent MZIs. Rectified linear units (ReLU) [85] with an upper limit of 4 are used as the nonlinearity.

3.3.4.1 Evaluation on Mixed-Training Strategy

Figure 3.12 demonstrate the inference accuracy curve after on-chip deployment using our mixed-training strategy. Active phases are randomly selected from all phases. With $\Delta\gamma \in \mathcal{N}(0, 0.002)$ phase shifter variations and ω=2e-3 thermal crosstalk between adjacent devices, the initial accuracy varies among different mixed-training sparsity values. A larger mixed-training sparsity can provide protection to the PIC from thermal crosstalk since more passive devices generate fewer thermal noises. The convergence of those curves shows that only a small fraction (5%-15%) of devices is necessary to perform on-chip learning for accuracy recovery, while simply tuning every parameter has the lowest efficiency and effectiveness among all settings. When α is set to 5%-15%, we observe the fastest convergence speed, leading to 3.7×-7.6× higher training efficiency, i.e., fewer function queries, than ours with a large α. This mixed-training strategy makes it possible to recover the accuracy of larger-scale ONNs with much fewer function queries and lower power.

3.3.4.2 Evaluation on the Sparsity of `SZO-SCD`

Figure 3.13 demonstrates how different sparsity s influences the on-chip learning performance under a given mixed-training sparsity $\alpha = 0.15$. Among all sparsity values, 60% is the best tuning sparsity that leads to the fastest convergence speed on small datasets. In contrast, on MNIST, since the dataset variance is much larger than Vowel Recognition, a highly-sparse learning strategy (s <0.1) is more suitable to balance the variance and generalization in the



(a) (b) (c)

Figure 3.13: Evaluation with different sparsity s in SZO−SCD. α is set to 0.15 for all models. (a) 8-16-16-4 on Vowel Recognition dataset. (b) 10-24-24-6 on Vowel Recognition dataset. (c) (8×8)-24-24-10 on MNIST dataset.

Table 3.2: Comparison with SOTA ZO optimizers in terms of optimizer cost per iteration, ONN query complexity per iteration, and memory complexity. lr is the step size. We evaluate on MNIST with a 3-layer optical MLP (64-24-24-10). T is the total iteration. d is the total number of variables (d=2,350). The sampling factor Q is set to 60 as used in FLOPS [72].

Optimizer	α	s	lr	Computation	#ONN forward	Memory	Best Acc.
ZADAM [20]	1	1	1e-3	$\mathcal{O}(d)$	$2Td$ ($4700T$)	$\mathcal{O}(d)$	diverge
ZADAM [20]	0.15	0.1	1e-3	$\boldsymbol{\mathcal{O}(\alpha sd)}$	$2T\alpha sd$ ($70.5T$)	$\mathcal{O}(\alpha d)$	88.1%
ZNewton [20]	1	1	1e-3	$\mathcal{O}(d)$	$3Td$ ($7050T$)	$\boldsymbol{\mathcal{O}(1)}$	diverge
ZNewton [20]	0.15	0.1	1e-3	$\boldsymbol{\mathcal{O}(\alpha sd)}$	$3T\alpha sd$ ($105.75T$)	$\boldsymbol{\mathcal{O}(1)}$	diverge
STP [10]	1	1	2e-2	$\mathcal{O}(d)$	$2Td$ ($4700T$)	$\boldsymbol{\mathcal{O}(1)}$	diverge
STP [10]	0.15	0.1	2e-2	$\boldsymbol{\mathcal{O}(\alpha sd)}$	$2T\alpha sd$ ($70.5T$)	$\boldsymbol{\mathcal{O}(1)}$	90.2%
FLOPS [72]	1	1	1e-1	$\mathcal{O}(Qd)$	TQ ($60T$)	$\mathcal{O}(d)$	diverge
FLOPS [72]	0.15	0.1	1e-1	$\mathcal{O}(Qd)$	TQ ($60T$)	$\mathcal{O}(\alpha sd)$	83.5%
SZO−SCD (Proposed)	0.15	0.1	2e-2	$\boldsymbol{\mathcal{O}(\alpha sd)}$	$\boldsymbol{1.5T\alpha sd}$ **($52.88T$)**	$\boldsymbol{\mathcal{O}(1)}$	**93.5%**

stochastic optimization. In other words, overly-greedy optimization caused by large s values is harmful to stochastic learning. Note that a higher sparsity, e.g., s <0.02, will lead to accuracy loss since the variance is too large for the optimizer to converge, which is not shown in the figure for brevity.

3.3.4.3 Compare with Other Zeroth-Order Optimizers

To validate the efficiency of our proposed SZO−SCD, we compare a variety of state-of-the-art ZO optimizers on different sparsity in Table. 3.2.

Proper α and s are adopted to obtain a good trade-off between accuracy and efficiency. Learning rates reported are empirically most suitable values with equal parameter searching efforts for all methods. The comparison results provide several important insights. First, in high-dimensional ONN parameter space, the gradient-based methods, e.g., `FLOPS`, generally show poor performance and unstable convergence due to gradient estimation variance. Even for `ZADAM` and `ZNewton` that adaptively adjust the step size, they suffer from divergence in the phase-domain optimization unless a descent in the objective can be partially guaranteed like our proposed coordinate descent method. Second, the two-level sparsity is indeed necessary to achieve stable convergence and good model generalization. `FLOPS` with two-level sparsity coincides with [152] and shows better convergence than the dense counterpart. Third, gradient-based methods require an arbitrarily tiny step size (<1e-3) for gradient estimation and weight updating, which is not practical given limited device control resolution. In contrast, our method only needs a medium step size, corresponding to 8-bit control precision, a more hardware-friendly configuration in analog neuromorphic computing. Fourth, interestingly, the stochastic coordinate-wise three-points method (`STP`) leads to worse inference accuracy than our method due to its overly-greedy updating mechanism that potentially harms generalization as follows,

$$\phi_i^t \leftarrow \underset{\phi_i}{\operatorname{argmin}}\{\mathcal{L}(\phi_i^{t-1}), \mathcal{L}(\phi_i^{t-1}+\delta\phi), \mathcal{L}(\phi_i^{t-1}-\delta\phi)\}. \tag{3.22}$$

Table. 3.3 further shows the average performance on different datasets with CNNs. Overall, our proposed mixed-training strategy with `SZO-SCD` shows

Table 3.3: Average accuracy(std.) among different optimizers over 3 runs. The CNN setting is 16×16-c8s2-c6s2-10 for MNIST, 32×32-c8s2-c8s2-10 for FMNIST, and 32×32-c8s2-c8s2-c8s2-10 for CIFAR-10. *c8s2* is 8 kernels with size 3×3 and stride 2. α and s are set to 0.05 and 0.1 for all optimizers.

Optimizer	MNIST	FMMIST	CIFAR-10
`ZAdam`	88.51%(0.10)	68.16%(0.13)	diverge
`ZNewton`	diverge	67.60%(0.23)	diverge
`STP`	93.74%(0.30)	75.43%(3.86)	diverge
`FLOPS`	diverge	67.27%(0.19)	diverge
`SZO-SCD`	94.88%(0.26)	82.63%(0.04)	51.35%(0.86)

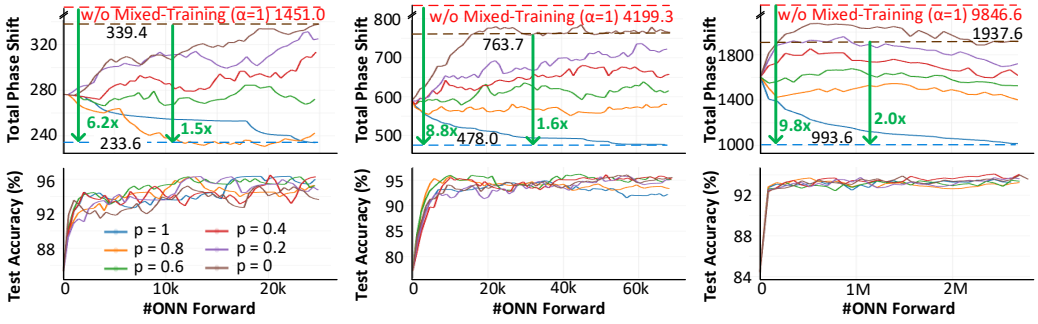


Figure 3.14: Estimated power and inference accuracy with different power awareness p. The mixed-training sparsity α is selected as 0.15. The sparsity s for `SZO-SCD` is set to 0.6 for (a) and (b), and is set to 0.1 for (c). Models/datasets are the same as Fig. 3.12

the best convergence and accuracy with the smallest computation and memory cost.

3.3.4.4 Evaluation on the Power-Aware Dynamic Pruning

We evaluate the effectiveness of our proposed power-aware dynamic pruning technique in Fig. 3.14. Different power awareness values lead to slightly different inference accuracy after convergence. However, fully-power-

Table 3.4: Comparison among ADMM-based method and our dynamic power-aware pruning. Power is estimated by the total phase shifts of active MZIs. λ is the weight of the power penalty. The 3-layer optical MLP is 64-24-24-10, and the dataset is downsampled MNIST. We use α=0.15, s=0.1.

Method	Hyperparam.	Power (rad)	Test Accuracy
ADMM	$\lambda = 0.05 \sim 0.3$	2077∼2367	92.8%∼79.0%
ADMM	$\lambda > 0.3$	-	diverge
Proposed	$p = 0 \sim 1$	1938∼994	93.9%∼93.1%

aware (p=1) pruning can cut down 30%-50% power compared with the power-unaware version (p=0). Compared with the naive ONN deployment without mixed-training, our power-aware mixed-training can save a total ∼90% power. This lightweight pruning method not only reduces the power in inference $\mathcal{P}$, shown in the final power value at the end of the curve, it also saves the training energy $\int_t \mathcal{P}\, \mathrm{d}t$ indicated by the area under the power curve. We also compare with ADMM to show the superiority of our dynamic pruning technique in Table 3.4. The Lagrangian-relaxation-based formulation and ADMM-based optimization algorithm are not suitable for power-aware ONN on-chip learning. A small λ in the dual update step has a trivial influence on the total power, while a large λ leads to unstable convergence. In contrast, our proposed method can provide stable power constraint handling with a parametric mechanism to achieve a trade-off between accuracy and power.

3.3.4.5 Evaluation on CNNs and Different Datasets

We further evaluate the effectiveness of our proposed mixed-training strategy with sparse tuning on convolutional neural networks (CNNs). We use

Table 3.5: Power reduction on CNNs (same as Table. 3.3). *DAcc.* and *RAcc.* mean deployed and recovered accuracy. *PR-Ours* and *PR-FLOPS* are power reduction compared to ours(p=0) and FLOPS. All datasets use α=0.05, s=0.1, and p=1.

Dataset	DAcc.	RAcc.	PR-Ours.	PR-FLOPS
MNIST	87.4%	95.5%	98.8%	97.6%
FMNIST	65.7%	82.6%	95.6%	98.1%
CIFAR-10	36.0%	52.5%	96.7%	96.7%

im2col algorithm to implement convolution with general matrix multiplication (GEMM). Table 3.5 shows the accuracy recovery results and power improvement on three different datasets compared with w/o mixed-training or power handling. On three practical datasets, our proposed methods demonstrate stable accuracy recovery for optical CNN architectures under device variations while reducing the total inference power by >95%.

3.3.5 Summary

In this work, we propose an efficient ONN on-chip learning framework MixedTrain to perform *in situ* accuracy recovery. We are the first to formulate the ONN on-chip learning problem with device non-ideality and power constraints. A sparse mixed-training strategy SZO−SCD is proposed to explore two-level sparsity in ONN deployment and optimization, leading to better training efficiency and robustness. A lightweight dynamic power-aware pruning is proposed to implicitly optimize power with near-zero computational cost or accuracy loss. Compared with SOTA methods, our framework MixedTrain boosts the efficiency by 3.7×-7.6× with better crosstalk-

robustness, 2× better scalability, and over 10× better power efficiency.

3.4 L²ight: Enabling Scalable ONN On-Chip Learning via Efficient *in-situ* Subspace Optimization

Prior work [242, 99, 72, 65] only demonstrated small prototypes, and their scalability and efficiency are rather limited. To push the limits of ONN on-chip training, we propose an efficient three-stage learning framework L²ight to achieve 10,000× improvement in the training scalability. The framework consists of variation-agnostic identity calibration, alternate projection-based parallel mapping, and multi-level sparse subspace learning. The main contributions of this work are four-fold,

- **Scalability**. *For the first time*, an ONN learning protocol can scale up to million-level parameters under practical circuit non-ideality, over 4-order-of-magnitude more scalable than prior arts.
- **Efficiency**. We explore multi-level sparsity in *in-situ* gradient evaluation to trim down unnecessary on-chip training energy and runtime cost.
- **Learnability**. By trading redundant representability, our restricted sub-

This L²ight section is based on the following publication.

1. Jiaqi Gu, Hanqing Zhu, Chenghao Feng, Zixuan Jiang, Ray T. Chen, and David Z. Pan, "L2ight: Enabling On-Chip Learning for Optical Neural Networks via Efficient in-situ Subspace Optimization," Conference on Neural Information Processing Systems (NeurIPS), Dec. 2021.

I am the main contributor in charge of problem formulation, algorithm development, and experimental validations.

space optimization can provide ONNs with enough adaptability for on-device self-learning and task transfer.

- **Robustness**. Various practical device noises and process variations are considered *in situ* to facilitate noise-resilient photonic AI engines.

- To our best knowledge, this is the first framework that supports on-chip training on million-parameter ONNs, over **10,000**× more scalable and **30**× more efficient than prior arts. We open-source a PyTorch-centric [154] ONN library TorchONN and release our on-chip training framework at link.

3.4.1 Preliminaries

Optical Neural Network Training Methods. Beyond offline training [74], ONN on-chip training methods are proposed to offload the process back onto photonics [99, 72, 65], shown in Table 3.6. Brute-force device tuning (`BFT`) [171, 257] and evolutionary algorithms [242] are applied to search MZI settings. An adjoint variable method (`AVM`) [99] is proposed to directly evaluate gradients using *in-situ* light field monitoring. Stochastic zeroth-order optimization (ZOO) [72, 65] is later applied to improve the training efficiency. However, prior methods are hard to scale to larger ONNs either due to algorithmic inefficiency or unrealistic hardware complexity.

Efficient NN Training Methods. Extensive work has been devoted to accelerating DNN training, among which an important branch is sparse back-propagation. Previous methods mainly focus on approximating matrix multi-

Table 3.6: Scalability comparison with prior ONN on-chip training protocols in terms of #Params they can handle, used algorithm, resolution requirement (*Req.*), and circuit observability requirement. *Coh. I/O* is short for coherent input/output [141, 238]. ZO, FO mean zeroth- and first-order methods.

	BFT [171]	PSO [242]	AVM [99]	FLOPS [72]	MixedTrn [10]	L²ight
#Params	∼100	∼100	∼100	∼1000	∼2500	**∼10 M**
Algorithm	ZO	ZO	FO	ZO	ZO	ZO+FO
Resolution Req.	Medium	High	Medium	High	Med	Medium
Observability Req.	Coh. I/O	Coh. I/O	Coh. I/O + Per device monitor	Coh. I/O	Coh. I/O	Coh. I/O

plication by sparsifying the pre-activation gradients [181], forward and feedback matrices [3, 157], and input feature maps [153]. Quantization to the pre-activation gradients is adopted in [216] to induce sparsity by trading off quantization steps and performance. Other methods also exist, e.g., distributed and low-precision training [7, 8, 104]. However, they are not readily applicable to analog photonic engines, thus not in the scope of our discussion.

Subspace Neural Networks. Subspace neural networks are special DNN models with restricted parameter space but demonstrate comparable representability to classical NNs. Sparse NNs [82, 214], low-rank NNs [38, 114, 187], structured NNs [40, 121, 209], Fourier-domain NNs [70, 144, 143], and general frequency-domain NNs [71] were introduced to trim down the redundancy in DNNs by restricting the NN structure, matrix rank, numerical resolution, etc. In this work, we deeply explore the trade-off between ONN learnability, trainability, and efficiency in the restricted unitary subspace.

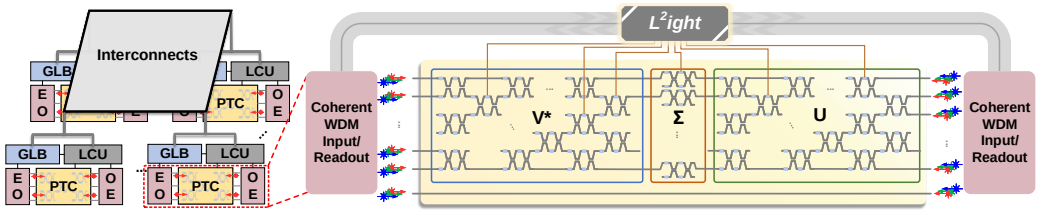


Figure 3.16: ONN architecture. PTC: photonic tensor core, GLB: global buffer, LCU: local control unit, EO: electrical-to-optical conversion.

3.4.2 Synergistic ONN On-Chip Learning Framework L²ight

In this section, we give a formal description of the ONN on-chip training problem and detailed demonstration of our proposed three-stage learning flow L²ight, shown in Figure. 3.15.

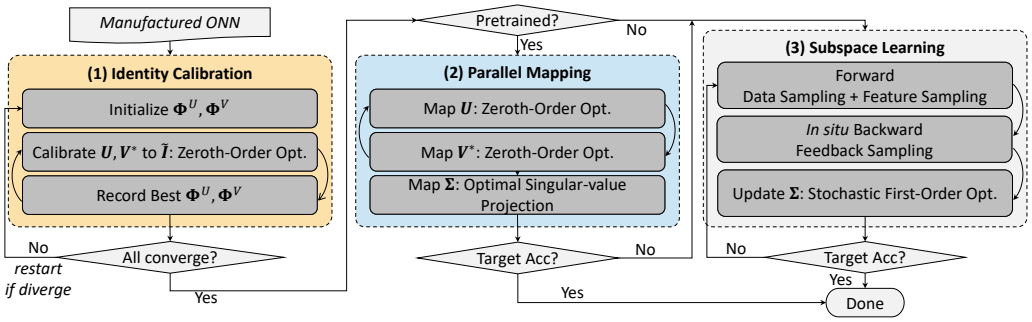


Figure 3.15: Proposed three-stage ONN on-chip learning flow L²ight.

3.4.3 Understanding the ONN On-Chip Learning Problem

The ONN that supports on-chip learning is shown in Figure 3.16, constructed by local storage, control units, interconnects, and photonic tensor cores with coherent I/O [141, 238] and wavelength-division multiplexing (WDM) [233, 191] for parallel processing. The target is to optimize MZI phases $\mathbf{\Phi}$ directly on chip under variations. Formally the *hardware-restricted* learning

problem is,

$$
\begin{aligned}
&\boldsymbol{\Phi}^* = \underset{\boldsymbol{\Phi}}{\operatorname{argmin}}\ \mathcal{L}\big(\boldsymbol{W}(\boldsymbol{\Omega}\boldsymbol{\Gamma}\mathcal{Q}(\boldsymbol{\Phi}) + \boldsymbol{\Phi}_b); \mathcal{D}_{trn}\big),\\
\text{s.t. }\ &\boldsymbol{W}(\boldsymbol{\Phi}) = \big\{\boldsymbol{W}_{pq}(\boldsymbol{\Phi}_{pq})\big\}_{p=0,q=0}^{p=P-1,q=Q-1}, \quad \boldsymbol{W}_{pq}(\boldsymbol{\Phi}_{pq}) = \boldsymbol{U}_{pq}(\boldsymbol{\Phi}_{pq}^U)\boldsymbol{\Sigma}_{pq}(\boldsymbol{\Phi}_{pq}^S)\boldsymbol{V}_{pq}^*(\boldsymbol{\Phi}_{pq}^V),\\
&\boldsymbol{U}_{pq}(\boldsymbol{\Phi}_{pq}^U) = \boldsymbol{D}_{pq}^U \prod_{i=k}^{2}\prod_{j=1}^{i-1} \boldsymbol{R}_{pqij}(\phi_{pqij}^U), \quad \boldsymbol{V}_{pq}^*(\boldsymbol{\Phi}_{pq}^V) = \boldsymbol{D}_{pq}^V \prod_{i=k}^{2}\prod_{j=1}^{i-1} \boldsymbol{R}_{pqij}(\phi_{pqij}^V),\\
&\boldsymbol{\Sigma}_{pq}(\boldsymbol{\Phi}_{pq}^S) = \max(|\boldsymbol{\Sigma}_{pq}|)\texttt{diag}(\cdots, \cos\phi_{pq,i}^S, \cdots), \quad \boldsymbol{\Phi}_b \sim \mathcal{U}(0, 2\pi),\ \boldsymbol{\Gamma} \sim \mathcal{N}(\gamma, \sigma_\gamma^2).
\end{aligned}
\tag{3.23}
$$

The linear projection in an ONN adopts blocking matrix multiplication, where the $M \times N$ weight matrix is partitioned into $P \times Q$ blocks of size $k \times k$. During the optimization of $\boldsymbol{\Phi}$, we jointly consider control resolution limits $\mathcal{Q}(\cdot)$ [74, 171], device process variations $\boldsymbol{\Gamma}$ [74, 72, 65], thermal crosstalk among adjacent devices $\boldsymbol{\Omega}$ [72, 261], and unknown phase bias due to manufacturing error $\boldsymbol{\Phi}_b$ for *in-situ* robustness-aware training. A detailed non-ideality analysis is in Appendix .2.1. For practicality, robustness, and convergence consideration, we select k=9, which is explained in Appendix .2.6.

3.4.4 Identity Calibration (IC): Variation-Agnostic Circuit State Preparation

After manufacturing, unknown process variations in waveguides make the initial state of PTCs unpredictable [195, 261]. A primary task is to prepare $\boldsymbol{U}$ and $\boldsymbol{V}^*$ to be identity matrices. However, the calibration problem, i.e., $\min_{\boldsymbol{\Phi}^U, \boldsymbol{\Phi}^V}\ \sum_{p,q}\big(\|\boldsymbol{U}_{pq}(\boldsymbol{\Phi}_{pq}^U) - \boldsymbol{I}\|_2^2 + \|\boldsymbol{V}_{pq}^*(\boldsymbol{\Phi}_{pq}^V) - \boldsymbol{I}\|_2^2\big)$, is not solvable given the observability and controllability constraints on $\boldsymbol{U}$ and $\boldsymbol{V}^*$. The closest auxiliary problem that we can solve is the one with absolute operations on unitaries, i.e., $\min_{\boldsymbol{\Phi}^U, \boldsymbol{\Phi}^V}\ \sum_{p,q}\big(\||\boldsymbol{U}_{pq}(\boldsymbol{\Phi}_{pq}^U)| - \boldsymbol{I}\|_2^2 + \||\boldsymbol{V}_{pq}^*(\boldsymbol{\Phi}_{pq}^V)| - \boldsymbol{I}\|_2^2\big)$. We

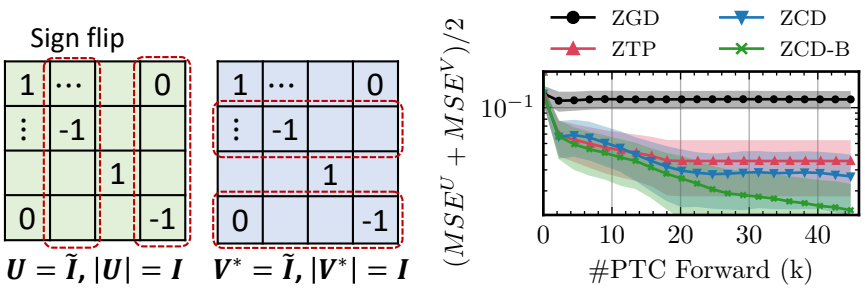


Figure 3.17: (a) Identity calibration with sign flip. (b) Different ZO optimizers on identity calibration. (ZGD: ZO gradient descent with momentum, ZCD: ZO coordinate descent, ZTP: ZO three-point. B is the best solution recording.)

denote those two mean square errors as MSE^U and MSE^V. We rewrite it as a surrogate minimization of $\mathcal{L}_{IC}$ that can lead to the same solution,

$$\min_{\boldsymbol{\Phi}} \sum_{p,q} \|\boldsymbol{U}_{pq}(\boldsymbol{\Phi}^U_{pq})\boldsymbol{\Sigma}_{pq}\boldsymbol{V}^*_{pq}(\boldsymbol{\Phi}^V_{pq})\boldsymbol{\Sigma}^{-1}_{pq} - I\|. \tag{3.24}$$

The optimal solution for this auxiliary problem is $\boldsymbol{U} = \boldsymbol{V}^* = \tilde{\boldsymbol{I}}$, where $\tilde{\boldsymbol{I}}$ is not guaranteed to be an identity matrix but a more general *sign-flipping matrix* with *arbitrary and unobservable sign flips* on the same columns in $\boldsymbol{U}$ and rows in $\boldsymbol{V}^*$, shown in Figure 3.17(a). We adopt zeroth-order optimization (ZOO) on $\boldsymbol{\Phi}^U$ and $\boldsymbol{\Phi}^V$ to calibrate $\boldsymbol{U}$ and $\boldsymbol{V}^*$ to approach $\tilde{\boldsymbol{I}}$, shown in Figure 3.17(b). We show the converged solution of Eq. (3.24) with unobservable sign flips and suboptimality only has marginal impacts on the following training procedure in later sections.

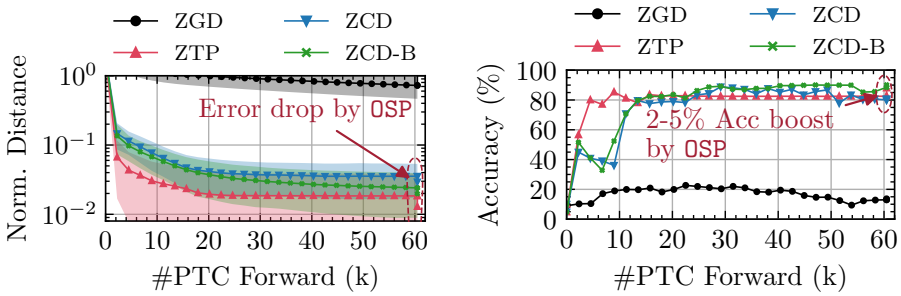


Figure 3.18: ZTP and ZCD-B perform the best in parallel mapping. The optimal singular-value projection leads to a significant error drop and accuracy jump.)

3.4.5 Parallel Mapping (PM): Alternate Projection-based Model Deployment

The target is to map the pre-trained weights $\boldsymbol{W}$ onto photonic MZI meshes $\widetilde{\boldsymbol{W}}(\boldsymbol{\Phi})$ with high fidelity.

We formulate the parallel mapping as a batched $k \times k$-block-wise regression problem,

$$\min_{\boldsymbol{\Phi}} \sum_{p,q} \|\widetilde{\boldsymbol{W}}_{pq}(\boldsymbol{\Phi}_{pq}) - \boldsymbol{W}_{pq}\|_2^2. \tag{3.25}$$

As analyzed before, $\frac{\partial \boldsymbol{W}}{\partial \boldsymbol{\Phi}^U}$ and $\frac{\partial \boldsymbol{W}}{\partial \boldsymbol{\Phi}^V}$ are too expensive to compute *in situ*. We propose a parallel mapping flow with alternate zeroth-order optimization on $\boldsymbol{\Phi}^U$ and $\boldsymbol{\Phi}^V$. After convergence, we will perform analytical optimal singular-value projection (OSP) to minimize the regression error given fixed singular vectors.

We show why OSP gives the optimal solution under sign flips and how to perform it on the PTC.

Claim 1. *Optimal singular-value projection (OSP): the optimal singular value problem, i.e.,* $\boldsymbol{\Sigma}_{opt} = \text{argmin}_{\boldsymbol{\Sigma}} \ \|\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* - \boldsymbol{W}\|$*, can be analytically solved on-chip with arbitrary and unknown sign flip.*

Proof.

$$\boldsymbol{\Sigma}_{opt} = \texttt{diag}\big(\boldsymbol{U}^{-1}\boldsymbol{W}(\boldsymbol{V}^*)^{-1}\big) = \texttt{diag}\big(\boldsymbol{U}^*\boldsymbol{W}\boldsymbol{V}\big) = \texttt{diag}\big((\tilde{\boldsymbol{I}}^*\boldsymbol{V}^*\boldsymbol{W}^*\boldsymbol{U}\tilde{\boldsymbol{I}})^*\big). \tag{3.26}$$

OSP can be directly achieved using the limited operation set, i.e., $\{\boldsymbol{U}, \boldsymbol{U}^*, \boldsymbol{V}, \boldsymbol{V}^*\}$, supported by the reciprocal PTC itself. Specifically, we configure $\boldsymbol{V}^* = \tilde{\boldsymbol{I}}$ and $\boldsymbol{\Sigma} = \boldsymbol{I}$, and shine in a coherent WDM light beam that carries $\boldsymbol{W}$ from right ports. Since the coherent photonic circuit is reciprocal [141], we can read $\tilde{\boldsymbol{I}}\boldsymbol{U}^*\boldsymbol{W}$ on the left ports. Then we configure $\boldsymbol{U} = \tilde{\boldsymbol{I}}$ and $\boldsymbol{\Sigma} = \boldsymbol{I}$, and shine in its adjoint field from left, i.e., $\boldsymbol{W}^*\boldsymbol{U}\tilde{\boldsymbol{I}}^*$. We can directly read out the projected optimal diagonal on the right because the sign flips in the unitary matrices naturally cancel out on the diagonal. □

Figure 3.18 compares different ZO optimizers on this task. Coordinate-wise optimizers (ZCD [124] and ZTP [44]) outperform the gradient-based ZGD [60] with higher accuracy and convergence speed. This procedure is highly parallel and efficient since the mapping involves *no stochasticity* and only happens *locally* within each PTC. We can also observe that OSP effectively reduces the normalized matrix distance ($\|\boldsymbol{W} - \widetilde{\boldsymbol{W}}\|_2^2 / \|\boldsymbol{W}\|_2^2$) and boosts the accuracy by 2-5% almost for free.

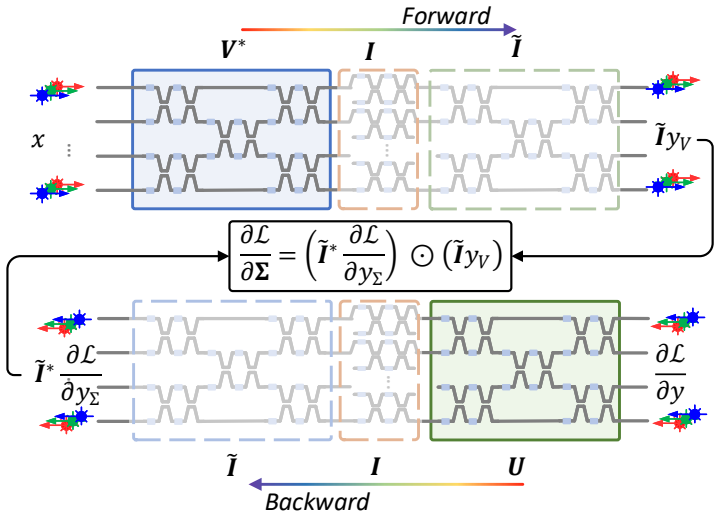


Figure 3.19: The proposed three-step *in-situ* subspace gradient calculation method.

3.4.6 Subspace Learning: Hardware-Aware Multi-Level Sparse Training

Besides mapping from an offline-trained model, L^{2}ight also supports *in-situ* self-learning fully on chip. We name this feature as *subspace learning*. To make L^{2}ight hardware-aware, we trade expensive full-space trainability for efficient subspace gradient evaluation, i.e., $\frac{\partial \mathcal{L}}{\partial \Sigma}$ which coincides with the general frequency-domain ONNs [70, 71] and subspace NN design concept [180]. Since this learning stage involves stochasticity, it turns out to be the efficiency bottleneck, especially the backward pass. Hence, we explore multi-level sparsity for efficient *in-situ* gradient approximation.

3.4.6.1 *In-situ* Subspace Gradient Acquisition via Reciprocity in Optics

The conventional way to compute first-order gradients w.r.t. $\boldsymbol{\Sigma}$ is $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}} = \texttt{diag}\big(\boldsymbol{U}^* \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} \boldsymbol{V}\big)$. However, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{y}} \boldsymbol{x}^T$ requires arbitrary matrix multiplication, which is not implementable by weight-stationary PTCs. Hence, we remap it as,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{y}_\Sigma} \odot \boldsymbol{y}_V = \big(\tilde{\boldsymbol{I}} \boldsymbol{U}^* \frac{\partial \mathcal{L}}{\partial \boldsymbol{y}}\big) \odot (\tilde{\boldsymbol{I}} \boldsymbol{V}^* \boldsymbol{x}). \tag{3.27}$$

By shining in coherent WDM beams carrying the inputs and upstream gradients forward and backward through the reciprocal PTCs, respectively, as shown in Figure 3.19, the weight gradients can be efficiently obtained with lightweight element-wise multiplication $\odot$, which can be offloaded to electrical units. $\tilde{\boldsymbol{I}}$ naturally cancels out by the Hadamard product with no impacts on gradient fidelity.

3.4.6.2 Multi-Level Sparse Subspace Learning

Inspired by sparse backpropagation methods [181, 153, 210, 151, 157], we propose multi-level sparse subspace learning to cut down both energy cost and total time steps in on-chip gradient evaluation.

Balanced Feedback Sampling. To improve the efficiency of the error feedback process, i.e., $\boldsymbol{W}^T \frac{\partial \mathcal{L}}{\partial y}$, as shown in Figure 3.20, we sample the feedback matrix $\boldsymbol{W}^T \in \mathbb{R}^{N \times M}$ with a structured sparse mask $\mathcal{P}_W = c_W(\mathcal{S}_W \otimes \mathbf{1})$ generated by the Kronecker product between a boolean mask $\mathcal{S}_W \in \{0,1\}^{Q \times P}$ with sparsity α_W and an all-ones matrix $\mathbf{1}$, where the scaling factor c_W is

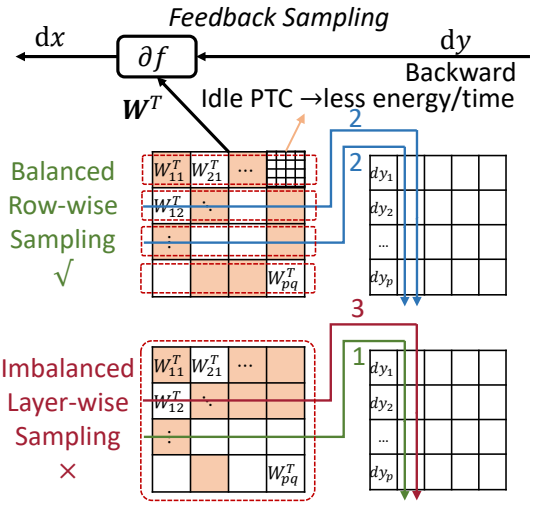


Figure 3.20: Balanced v.s. imbalanced feedback matrix sampling.

set to $\frac{1}{\alpha_W} = \frac{PQ}{\mathtt{Tr}(\mathcal{S}_W^T \mathcal{S}_W)}$ for unbiased estimation, proven in Appendix .2.4. The efficiency benefits come from two aspects: (1) the structurally masked PTCs are entirely idle, directly saving energy, and (2) the product accumulation depth/step is reduced by a factor of α_W, effectively trimming time steps.

However, two major downsides exist on traditional `uniform` and layer-wise `topk` sampling [157].

First, on a backward path, multiple feedback sampling operators will be cascaded, such that importance-unaware `uniform` sampling can lead to an exponentially large variance [153]. Second, `topk` sampling is overly greedy and tends to break the load balance as the feedback latency can be bottlenecked by the longest partial product accumulation path, shown in Figure 3.20. To tackle this, we propose a balanced top-K sampling (`btopk`) to draw $\mathcal{S}_W$ from a guided distribution that locally prefers blocks with large Frobenius

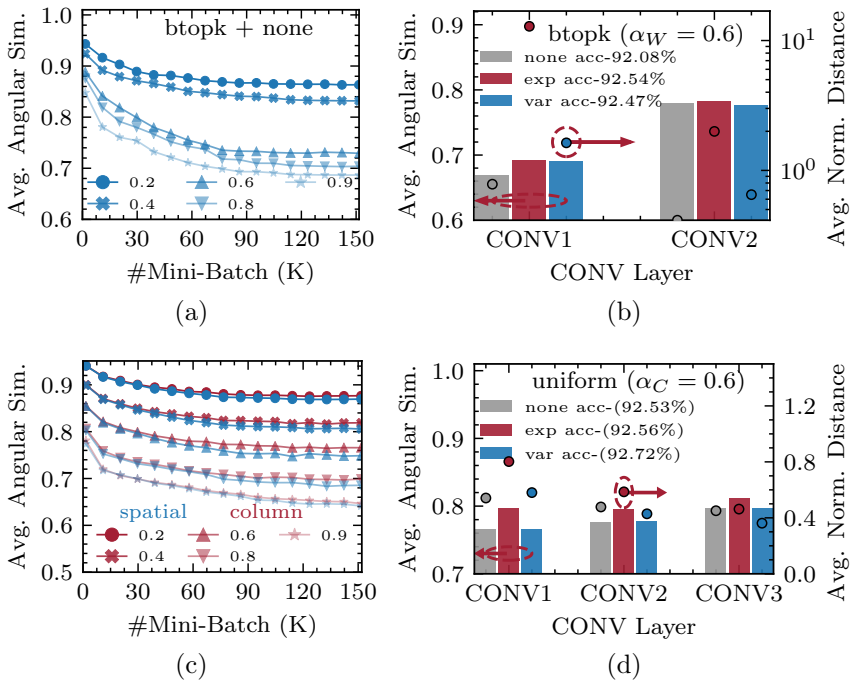


Figure 3.21: Average gradient angular similarity with different feedback sparsity (a) and three normalization methods (b). *none, exp,* and *var* represents no, expectation-maintained, and variance-maintained normalization. Average gradient angular similarity with spatial and column sampling (c) and three normalization methods (d).

norm, which can be efficiently evaluated by $\|\boldsymbol{W}_{pq}\|_{\mathcal{F}}^2 = \texttt{Tr}(|\boldsymbol{\Sigma}_{pq}|^2)$. It strikes a *balance between gradient variance and bias* by fine-grained row-wise top-K sampling and *eliminates load-imbalance* by guaranteeing the same sparsity for different rows of $\boldsymbol{W}^T$, i.e., $\sum_p \mathcal{S}_W(1,:) = \sum_p \mathcal{S}_W(2,:) = \cdots = \sum_p \mathcal{S}_W(Q,:)$. Figure 3.21(a), 3.21(b) shows the gradient approximation fidelity in terms of average angular similarity [15] and normalized matrix distance. Our `btopk`-sampled weight gradients align well with the true gradients. With the unbiased (`exp`) normalization factor α_W, `btopk` shows the best gradient angular simi-

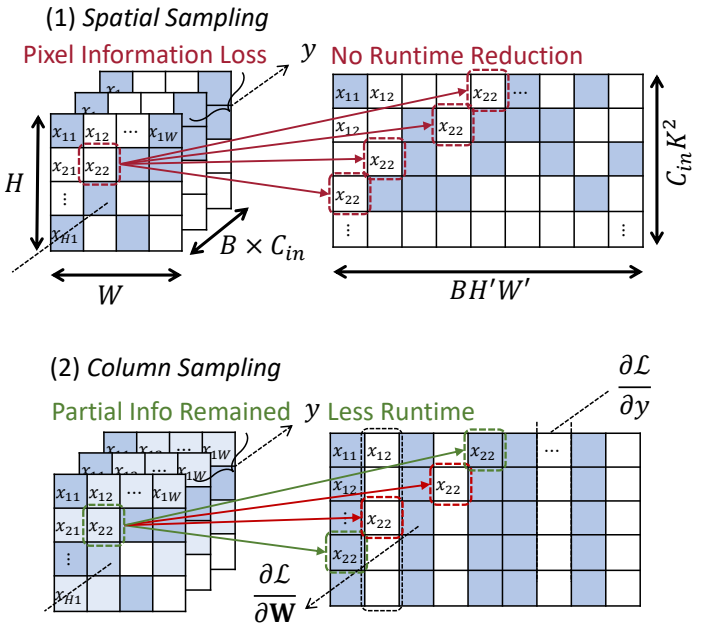


Figure 3.22: Spatial and column sampling for CONV.

larity and inference accuracy compared with others.

Information-Preserving Column Sampling. Input feature sparsification can also effectively cut down the gradient evaluation cost [157, 153], especially for costly CONV layers. However, with traditional *spatial sampling* (SS) [157, 153], the input feature map x barely maintains its sparsity regularity after being transformed to flattened patches X via *im2col* if the kernel size is larger than 1, shown in Figure 3.22. Hence, we propose a novel *column sampling* (CS) as a better solution. We sample X using a mask $\mathcal{S}_C\{0,1\}^{H'W'}$ with a uniform sparsity α_C, which is shared across batches with negligible overhead. This leads to both information preservation and efficiency improvement. First, in Figure 3.22, a pixel appears in multiple columns, such that

partial information can be maintained after column sampling. Second, this highly-structured column dropping directly translates to less PTC forward energy and fewer partial gradient accumulation steps. In contrast, with a spatial mask $\mathcal{S}_S$ and spatial sparsity α_S, the masked pixel will be completely dropped with poor regularity after *im2col*, at the cost of large variance due to information loss and almost no runtime improvement on this dense linear projection engines. Note that for CONV1×1, CS turns out to be equivalent to SS, which can simultaneously save memory and runtime. Figures 3.21(c), 3.21(d) show that our proposed CS can obtain better gradient approximation fidelity than prior SS. Different normalization has small effects on model accuracy since feature sampling only happens locally within each layer, without any variance cascade effect. Note that simultaneous scaling by α_W and α_C tends to generate overly-confident gradient approximation, which empirically leads to harmful gradient variance. Hence, we will adopt α_C=1 in all experiments.

Data Sampling. After parallel mapping, the ONN is initialized fairly close to the target pre-trained model. It is reasonable and intuitive to calibrate it with a representative calibration set instead of the entire training set. Inspired by the mini-batch dropping (`SMD`) technique [210], we integrate this `SMD` technique into our framework to further explore data-level sparsity. Within one training epoch, we randomly skip each iteration with probability α_D, directly translating to training time reduction.

3.4.7 Complexity Analysis of Three Stages in L^{2}ight

We assume the total step in IC, PM, and SL is T_1, T_2, and T_3, respectively. The ONN has L layers, each including an $N \times N$ weight matrix partitioned into multiple $k \times k$ blocks.

Identity Calibration and Parallel Mapping. Each block optimizes $k(k-1)$ phases using ZOO. All LN^2/k^2 blocks are optimized in parallel. The total step is $2k(k-1)T_1$ for IC and $2LN^2(k-1)T_2/k + 3$ for PM. The total PTC call is around $2LN^2T_1$ or $2LN^2T_2$ for IC and PM, respectively.

Subspace Learning. We assume the feature map size is $H \times W$ with a batch size of B. The detailed complexity analysis is given in Appendix .2.7. The total step is approximately T_3LNBHW/k.

According to our training cost profiler, IC and PM in total is 3-order-of-magnitude cheaper than the SL stage, since the batched parallel regression is deterministic and data-independent.

3.4.8 Experimental Results

3.4.8.1 Experiment Setup

Datasets. We evaluate L^{2}ight on Vowel [39], MNIST [115], FashionMNIST [222], CIFAR-10, CIFAR-100 [112], and TinyImagenet [35]. On CIFAR-10/100 and TinyImagenet, we adopt random crop, flip, and color jittering for augmentation.

Models. All models are implemented with our open-source PyTorch-centric

ONN library torchonn. We evaluate on a customized MLP (8-16-16-4) [65] on Vowel, CNN-S (CONV8K3S2-CONV6K3S2-FC10) [65] on MNIST, a CNN-L ({CONV64K3}×3-Pool5-FC10) on FashionMNIST, and VGG-8 [36] / ResNet-18 [1] [86] on CIFAR-10/100. CNN-L/FashionMNIST is used for ablation studies. VGG-8/ResNet-18 on CIFAR-10/100 are used for accuracy and efficiency comparison. Training details can be found in Appendix .2.5.

Efficiency Evaluation. We assume fully parallel 9×9-blocking matrix multiplication in photonic tensor cores and sequential partial product accumulation in electronics. All experiments and performance measurements are based on software simulation with various noise modeling. Our simulator counts the total number of PTC calls as the normalized energy indicator and the longest accumulation path as the normalized latency/runtime indicator. Details of profiling can be found in Appendix .2.7.

3.4.8.2 Main Results

Scalability Comparison with Prior ONN Learning Protocols. Figure 3.23 compares `L`^{2}`ight` with two SOTA ONN on-chip training protocols, `FLOPS` [72] and `MixedTrn` [65]. For ZO methods, i.e., FLOPS and MixedTrn, we count the energy and latency of forward PTC query in Appendix .2.7. Prior protocols can only handle toy models (∼1,000 params) given their algorithmic inefficiency and instability, while our `L`^{2}`ight` shows >10,000× higher scalability to handle large ONNs (∼10 M) on challenging tasks with comparable

[1] `https://github.com/kuangliu/pytorch-cifar`

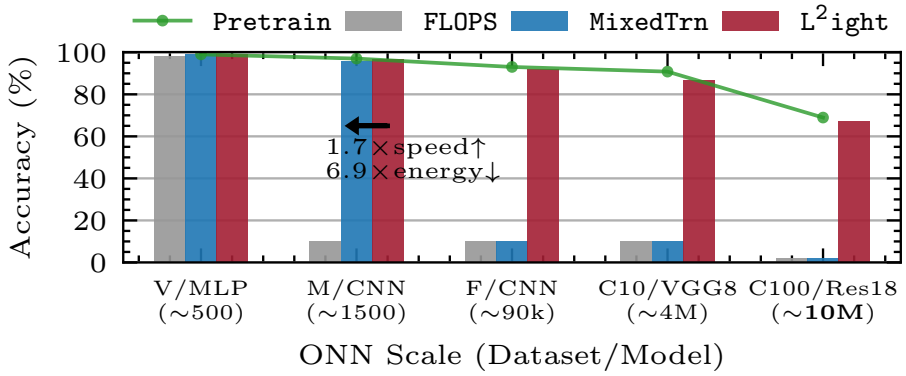


Figure 3.23: Compare scalability with prior protocols [72, 65].

accuracy to full-space pre-trained models. Though `MixedTrn` achieves comparable accuracy to `L`^{2}`ight` on small benchmarks, we are still 1.7× faster and 6.9× more energy efficient.

The superiority of `L`^{2}`ight` provides three important insights: (1) *decoupling ZOO from stochasticity* and *partitioning a large-scale regression problem into a batch of sub-tasks* can greatly mitigate the curse of dimensionality both in convergence and efficiency. (2) *mapping before learning* can fully leverage the pre-trained model to reduce the learning cost. Prior methods have to learn from a severely corrupted solution under variations, while `L`^{2}`ight` recovers most accuracy via mapping, leaving a very light workload for subspace learning. (3) Restricted subspace learning provides *adequate degree of freedom* for training from scratch and task transfer. Also, its *compatibility with first-order* methods significantly boosts the trainability and breaks the scalability barrier for ONN training. We now validate the above insights through extensive experiments.

Training Efficiency Comparison with Prior Sparse Training Methods. In Figure 3.24, we show accuracy and efficiency comparison of 1) baseline L^{2}ight-SL (BS), 2) L^{2}ight-SL with spatial sampling (RAD), 3) L^{2}ight-SL with weight and spatial sampling (SWAT-U), and 4) L^{2}ight-SL with all three introduced sampling methods (feedback, column, and data sampling), and 5) our proposed full flow with IC, PM, and sparse SL (L^{2}ight). To clarify, L^{2}ight-SL performs subspace learning on-chip from scratch without using pre-trained weights, while L^{2}ight includes the full flow, i.e., pre-training, mapping, and on-chip training. When we perform subspace learning from scratch, our proposed *multi-level sampling* strategies outperform previous RAD and SWAT-U by $\sim$**3**$\times$ in hardware cost with comparable accuracy. Though RAD can save the forward peak memory, it leaves the most expensive backward pass unoptimized, which does not fully exploit the sparsity in ONN training. SWAT-U tries to save forward cost by simultaneously sparsifying the forward and feedback weights with shared masks/patterns. However, in our experiment, the forward sparsification considerably degrades the performance, which dilates the efficiency benefits from it. Parallel mapping can fully leverage the pre-trained weights and help our full three-stage flow L^{2}ight achieve the best accuracy with *much faster convergence*, leading to **over 30**$\times$ higher energy efficiency and fewer time steps.

Note that the energy efficiency and latency improvement is *not just on the photonic part but a systematic performance boost.* Our three-level sampling methods directly skip the pruned block, which means the corresponding cost of

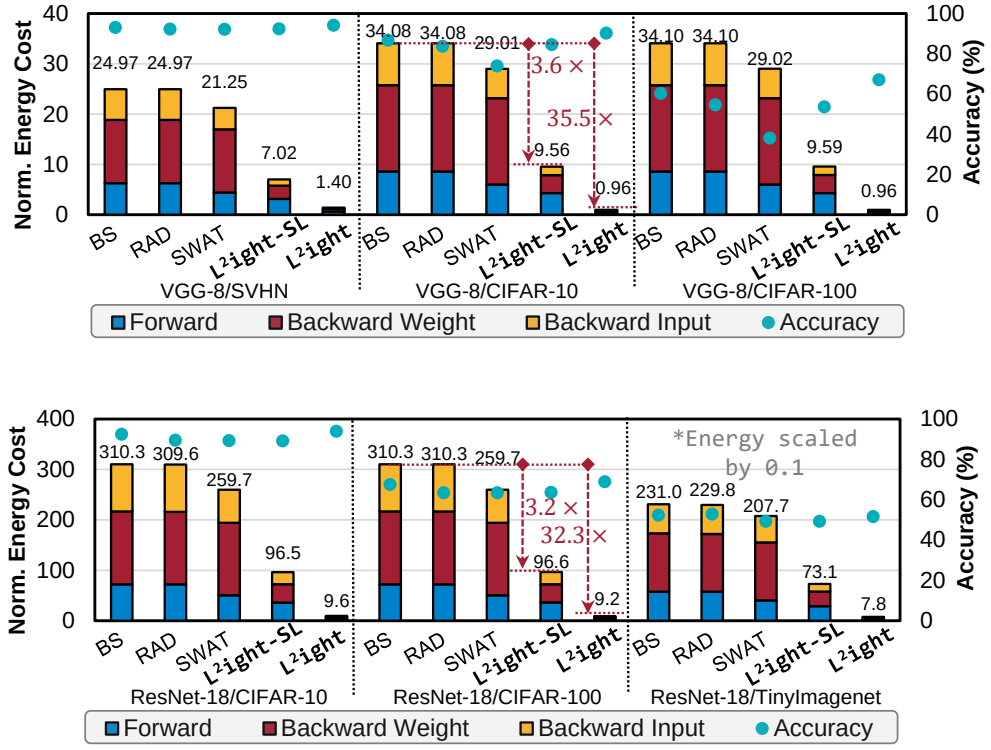


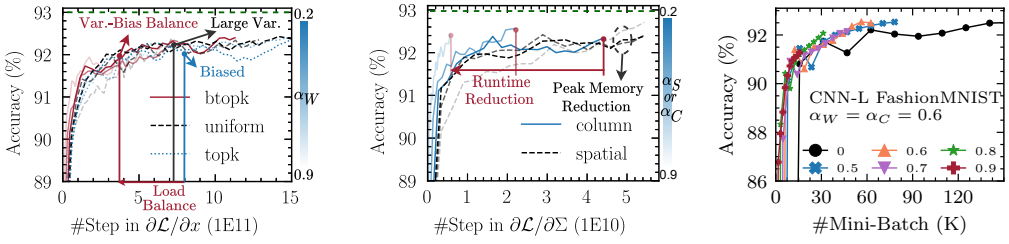
Figure 3.24: Accuracy and hardware efficiency comparison on VGG-8 (*Top*) and ResNet-18 (*Bottom*).

memory transaction, computation, control, and communication are removed together. Therefore, the sampling sparsity can be directly translated to the energy/latency improvement ratio regardless of whether the electrical part dominates the total cost.

3.4.9 Ablation Studies and Discussion

3.4.9.1 Multi-Level Sparsity in Efficient Training

Feedback Sparsity. To investigate the impact of feedback sampling strategies, we visualize the gradient approximation fidelity and accuracy curves in Figure 3.25(a). `uniform` sampling shows varying performance under different sparsity values due to large gradient variances. `topk` shows worse performance after sufficient steps due to its biased gradient estimation from overly greedy block selection. In contrast, our proposed *load-balancing* `btopk` strikes a balance between variance and bias via block-wise sampling and also leads to less runtime as it forces load balance among massively parallel PTCs. In Table 3.7, feedback sampling saves 50-60% time steps on the most costly error feedback $\nabla_x \mathcal{L}$, leading to 1.5-1.8× overall time step reduction with minimum accuracy drop.



(a) (b) (c)

Figure 3.25: Accuracy v.s. weight gradient computation steps with three feedback sampling strategies (a) and different feature sampling techniques (b). Accuracy (93.02%) from a full-space trained model (green). CNN-L/FashionMNIST is used for (a) and (b). Compare different data sampling sparsity (c).

Feature Sparsity. Figure 3.25(b) compares the accuracy and weight gradient computation time steps on two feature sampling techniques. Though *spatial*

sampling (ss) can save peak storage by dropping a subset of activations during the forward pass, it shows no gradient computation step reduction. Our hardware-friendly *column sampling* (cs) directly leads to energy and runtime reduction due to its structured sparsity. In Table 3.7, when column sampling is further added, we observe ∼50% PTC energy saving on weight gradient computation $\nabla_\Sigma \mathcal{L}$ at the cost of ∼1% accuracy drop.

Data Sparsity. In the data level, we also demonstrate how SMD with sparsity α_D impacts the training efficiency in Figure 3.25(c). With the best selected α_W and α_C, data sparsity directly reduces training time by skipping iterations [210]. The data sampling selects a uniform subset of the training set to represent the original data distribution, leading to less data processing with comparable generalization in the extracted features. Another explanation is that the variance increased by partial replacement serves as a regularization mechanism to improve the model performance [210]. For relatively easy tasks,

Table 3.7: Compare sampling strategies on CIFAR-10 in terms of accuracy, activation size reduction, energy, and time step. Forward, weight gradient, and error feedback are denoted as $\mathcal{L}$, $\nabla_\Sigma \mathcal{L}$, and $\nabla_x \mathcal{L}$. L^{2}ight-SL is learning *from scratch*, and L^{2}ight (IC→PM→SL) is the full flow with pre-trained weights and non-ideal $\tilde{\boldsymbol{I}}$.

	$\text{Acc}_{\pm\sigma}$ (%)	Act↓(%)	Norm. PTC Energy				Norm. #Step			
			$\mathcal{L}$	$\nabla_\Sigma \mathcal{L}$	$\nabla_x \mathcal{L}$	Total (Ratio)	$\mathcal{L}$	$\nabla_\Sigma \mathcal{L}$	$\nabla_x \mathcal{L}$	Total (Ratio)
L^{2}ight-SL (Baseline) VGG-8	$86.66_{\pm 0.13}$	-	8.58	17.16	8.34	34.08 (1.00)	32.64	5.49	92.02	130.14 (1.00)
+ Feedback Sampling (α_W=0.6)	$86.41_{\pm 0.25}$	-	8.58	17.16	3.38	29.13 (1.17)	32.64	5.49	35.76	73.89 (1.76)
+ Column Sampling (α_C=0.6)	$85.58_{\pm 0.01}$	-	8.58	7.16	3.38	19.12 (1.78)	32.64	4.67	35.76	73.07 (1.78)
+ Data Sampling (α_D=0.5)	$84.45_{\pm 0.45}$	-	4.29	3.58	1.69	9.56 (3.56)	16.32	2.34	17.89	36.54 (3.56)
+ RAD [153] (α_S=0.85)	$83.68_{\pm 0.58}$	11.78	8.58	17.16	8.34	34.08 (1.00)	32.64	5.49	92.02	130.14 (1.00)
+ SWAT-U [157] (α_W=0.3, α_S=0.6)	$73.91_{\pm 0.27}$	8.31	6.01	17.16	5.84	29.01 (1.17)	25.98	5.49	82.19	113.66 (1.15)
L^{2}ight (IC→PM→SL)	$\mathbf{90.20}_{\pm 0.05}$	-	**0.43**	**0.36**	**0.17**	**0.96 (35.64)**	**1.63**	**0.23**	**1.79**	**3.65 (35.64)**
L^{2}ight-SL (Baseline) ResNet-18	$92.37_{\pm 0.08}$	-	72.24	144.49	93.60	310.33 (1.00)	463.40	27.23	1,478.84	1,969.48 (1.00)
+ Feedback Sampling (α_W=0.5)	$91.35_{\pm 0.03}$	-	72.24	144.49	48.13	264.86 (1.17)	463.40	27.23	747.22	1,237.85 (1.59)
+ Column Sampling (α_C=0.5)	$90.02_{\pm 0.16}$	4.47	72.24	72.49	48.13	192.86 (1.61)	463.40	15.68	747.21	1,226.30 (1.61)
+ Data Sampling (α_D=0.5)	$89.07_{\pm 0.04}$	4.47	36.13	36.26	24.07	96.46 (3.22)	231.76	7.84	373.71	613.31 (3.21)
+ RAD [153] (α_S=0.9)	$89.44_{\pm 0.17}$	46.60	72.26	143.72	93.60	309.58 (1.00)	463.53	26.03	1,478.84	1,969.00 (1.00)
+ SWAT-U [157] (α_W=0.3, α_S=0.5)	$89.21_{\pm 0.16}$	25.89	50.57	143.64	65.52	259.73 (1.19)	358.40	26.56	1,417.96	1,802.00 (1.09)
L^{2}ight (IC→PM→SL)	$\mathbf{93.91}_{\pm 0.02}$	4.47	**3.61**	**3.62**	**2.41**	**9.64 (32.20)**	**23.16**	**0.78**	**37.34**	**61.29 (32.13)**

aggressive sparsity (α_D=0.8) is a sweet point, while for larger datasets shown in Table 3.7, a medium sparsity (0.5) can be a good setting to balance both the training cost and accuracy. With all three sampling methods integrated, our L^{2}ight-SL shows competitive accuracy and ∼3× higher efficiency than RAD and SWAT-U. More advanced dataset sampling methods are left for future exploration.

3.4.9.2 Learnability of Restricted Subspace ONNs

Impacts of Calibration/Mapping Quality. Table 3.7 shows that with IC and PM, the full L^{2}ight flow achieves the highest accuracy with 32-35× efficiency boost over baselines. We further evaluate the impact of different mapping accuracy and calibration quality on subspace learning in Figure 3.26. First, *parallel mapping or pre-training is not a must*. Our subspace learning supports first-order optimization on-chip from random initialization. Second, *the optimality on subspace bases influences the final accuracy* as it determines the upper bound of accuracy that can be recovered by subspace learning. With roughly optimized space bases, i.e., $\boldsymbol{U}, \boldsymbol{V}^*$, subspace learning can efficiently train basis coefficients, i.e., $\boldsymbol{\Sigma}$, achieving 5-6% higher accuracy and 9.9× less energy and steps compared with random unitaries (train from scratch). Third,

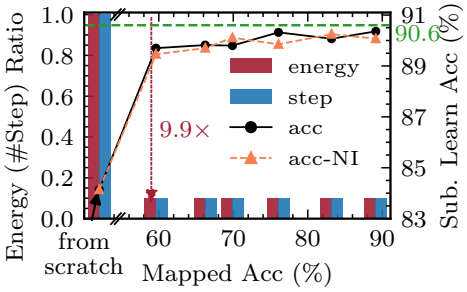


Figure 3.26: Impact of mapping accuracy (VGG-8 CIFAR-10 with α_W=α_C=0.6, α_D=0.5). *acc-NI* is the curve with non-ideal $\tilde{\boldsymbol{I}}$.

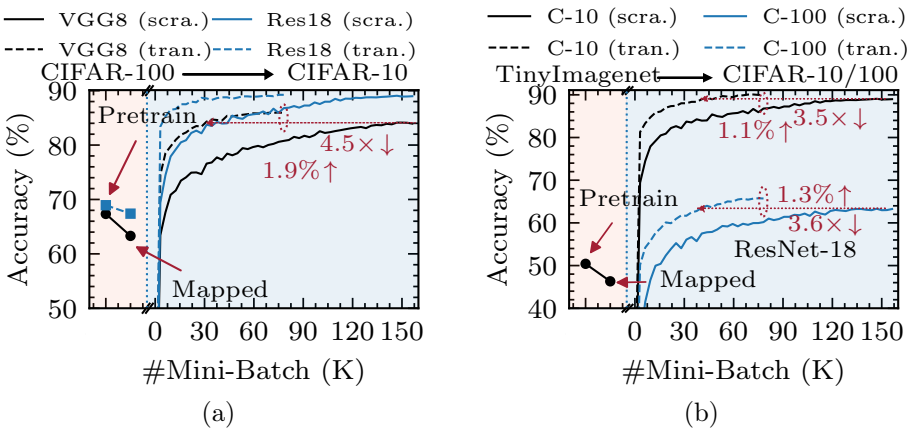


Figure 3.27: (a) Transfer VGG8/Res18 from CIFAR-100 to CIFAR-10. (b) Transfer Res18 from TinyImagenet to CIFAR-10 and 100.

subspace optimization shows low sensitivity on mapping quality and is able to compensate for the suboptimality in singular vectors within a reasonable range. Even with 60% mapped accuracy, singular value optimization has enough capability to recover the accuracy to ∼90%. Fourth, our subspace learning is *robust to gradient noises* caused by non-ideal $\tilde{\boldsymbol{I}}$ ($MSE^U \approx MSE^V \approx 0.013$), which shows that L^{2}ight can tolerate reasonable suboptimality in the calibration and mapping stages.

***In-situ* Transferability in the Restricted Subspace.** Another important question to answer is the transferability of subspace learning. After mapping, we fix the inherited unitaries and adapt to different tasks by only training the singular values. Figure 3.27 shows that the inherited bases span a good design space with enough transferability. The *in-situ* subspace transfer learning shows 1-2% higher final accuracy. Also, it uses 3∼5× fewer steps to obtain the same

accuracy as training from scratch. Hence, our proposed L^{2}ight finds a highly trainable design point while the learnability is still mostly maintained.

3.4.10 Summary

In this work, we propose the *first* scalable and efficient on-chip learning framework L^{2}ight for emerging optical neural networks. Our proposed three-stage flow synergistically enables on-chip self-learning via automatic circuit state calibration, parallel model mapping, and efficient subspace learning. To further improve learning efficiency, we explore multi-level sparsity, including balanced feedback sampling, information-preserving column feature sampling, and runtime-reduced data sampling. Extensive ablation studies and comparison experiments show 4-order-of-magnitude scalability improvement over prior on-chip training protocols and 30× efficiency boost compared with previous sparse training methods. We open-source a PyTorch-centric ONN library torchonn, based on which we release our on-chip learning framework L^{2}ight at link. In the future, we will go beyond current software simulation and experimentally validate the effectiveness of L^{2}ight on real photonic neural chips.

Chapter 4

AI-Assisted Intelligent Photonic Integrated Circuit Design Automation

4.1 Introduction

With recent advances in integrated photonics technology, optical deep learning represents a new paradigm in next-generation efficient artificial intelligence (AI) [171, 170, 26]. An increasing number of co-design efforts have been made to enable synergistic *light-AI interaction*. We see extensive developments for photonic AI with rapidly evolving optical neural network (ONN) hardware accelerator designs [171, 70, 227, 50, 131, 173, 52] and various circuit-algorithm co-optimization methodologies [70, 74, 73, 184, 75, 80]. However, conventional design flow is based on manual design, which requires many trials-and-errors and extensive domain expertise, and lacks automation to explore the design space, inevitably leading to unsatisfying design optimality.

An intelligent, fully-automated photonic IC design flow with customized photonic structure is promising to achieve breakthroughs in design quality and productivity. However, the field of AI for photonics and photonic design automation is still under-explored.

To close the virtuous cycle of photonics for AI and AI for photonics, in

the rest of this chapter, we introduce an AI-assisted ultra-fast Maxwell equation solving framework NeurOLight for photonic device simulation acceleration in Section 4.2. Section 4.3 presents the first automatic differentiable photonic circuit topology search framework ADEPT that achieves beyond-human design quality in integrated photonic tensor cores.

4.2 NeurOLight: A Physics-Agnostic Neural Operator Enabling Parametric Photonic Device Simulation

When we move toward an advanced photonic circuit design automation flow, a natural question is *whether AI can assist in the lower-level device simulation* to speed up the forward performance evaluation process. AI-assisted photonic device simulation is a critical step to closing the synergistic loop of *light-AI interaction*. Besides using standard devices that already have a compact transfer matrix [171, 70, 190], modern optical AI shows a trend to exploit customized photonic structures for scalable optical computing [73, 184, 260, 212]. Unfortunately, because customized devices do not have analytical transfer functions, understanding their behavior heavily relies on numerical simulators [100] to solve Maxwell partial differential equations (PDEs)

This NeurOLight section is based on the following publication.

1. Jiaqi Gu, Zhengqi Gao, Chenghao Feng, Hanqing Zhu, Ray T. Chen, Duane S. Boning, and David Z. Pan, "NeurOLight: A Physics-Agnostic Neural Operator Enabling Parametric Photonic Device Simulation," Conference on Neural Information Processing Systems (NeurIPS), Dec. 2022.

I am the main contributor in charge of problem formulation, algorithm development, and experimental validations.

to obtain the optical field distribution. Even solving a single 2-dimensional (2-D) simulation instance on a 5×20 μm^2 region can cost nearly 4 s on 8 CPUs, as shown in Figure 4.1(a), which significantly hinders scalable circuit-level simulation and optimization. Hence, our target is to propose a surrogate model that learns the *light propagation principle* and efficiently approximates the field solutions to new simulation instances while running *orders-of-magnitude faster* than the numerical solver, as shown in Figure 4.1(c).

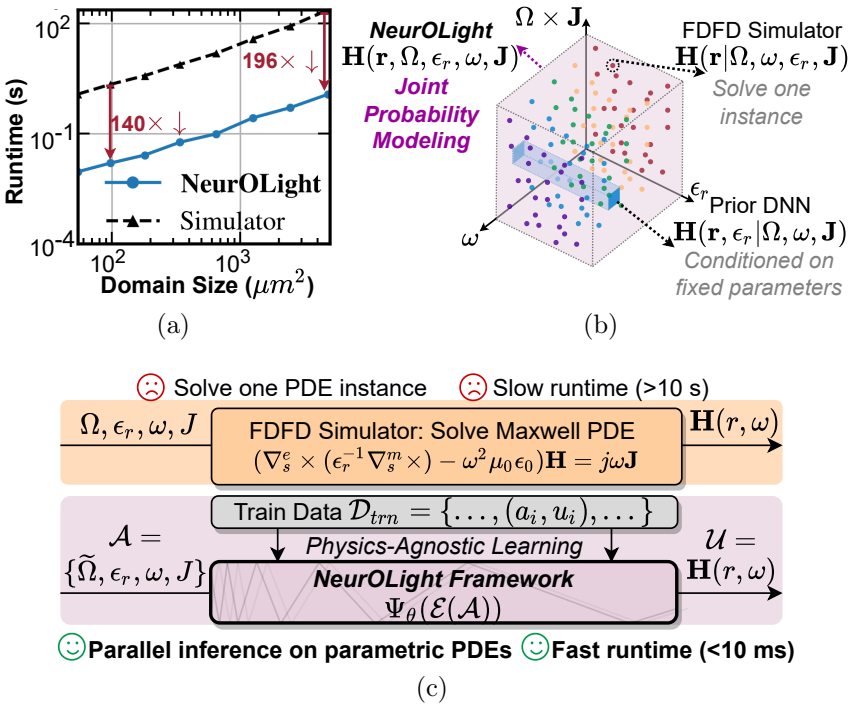


Figure 4.1: (a) Compare FDFD simulation and our `NeurOLight` framework. (b) Different methods cover different solution spaces. (c) `NeurOLight` (1 Quadro RTX 6000 GPU) runs 140×-200× faster than the FDFD simulator (8-core i7-9700 CPUs) across different domain sizes (50 nm grid step).

Most prior work still uses conventional NNs to predict several key properties based on a few design variables [186, 199], which is an ad-hoc function approximator without learning the light propagation property. Several works attempt to leverage physics-informed NNs (PINNs) [193, 19, 126] with physics-augmented residual loss to predict electromagnetic field solutions. However, previous methods have three major limitations. First, as illustrated in Figure 4.1(b), they only model the field distribution conditioned on pre-defined solving domains, input sources, and frequency. In other words, their models only learn the solution of a certain PDE instance with fixed parameters. Second, they all belong to the category of PINNs [158] that require an explicit description of the PDE as well as strict initial/boundary conditions. Constructing complicated Maxwell equation-based residual loss is no easier than re-implementing a numerical solver [135, 122, 123]. Third, their CNN-based models show inadequate modeling capacity with too small receptive fields to learn important light propagation, scattering, and interference effects.

To learn *a family of parametric Maxwell PDEs* that models the *joint probability* of different design variables as shown in Figure 4.1(b), we propose a physics-agnostic light field prediction framework `NeurOLight` that consists of a joint PDE encoder and an efficient cross-shaped neural operator backbone. The main contributions of the work are as follows:

- *For the first time*, an AI-based photonic device simulation framework is proposed to *learn a family of parametric* Maxwell PDEs for ultra-fast optical field prediction.

- We propose a novel joint PDE encoder for compact PDE representation and an efficient cross-shaped Fourier neural operator backbone for end-to-end optical field prediction, over **2-order-of-magnitude faster** than numerical simulators.

- We propose a superposition-based mixup technique that dynamically boosts the data efficiency and generalization during the training of `NeurOLight`.

- On different photonic device simulation benchmarks, our `NeurOLight` achieves the best prediction fidelity, generalizability, and transferability, outperforming UNet and state-of-the-art (SoTA) Fourier neural operators by an average of 53.8% lower prediction error with 44.2% fewer parameters.

- To our best knowledge, this is *the first* AI-based framework that can learn the terahertz light propagation inside photonic devices that generalizes to different domains, permittivities, input sources, and frequencies. We open-source our `NeurOLight` framework at link.

4.2.1 Preliminaries

Optical field simulation with machine learning. Finite difference frequency domain (FDFD) simulation is a widely adopted method to analyze silicon-photonic devices. Numerical simulators are used to solve frequency-domain Maxwell PDEs to obtain electromagnetic field distributions of an optical component with terahertz incident light sources. To accelerate this time-consuming process, NNs have been utilized as surrogate models for fast

optical simulation approximation. A multi-layer perceptron was used to map the design variables to a scalar performance metric of a power splitter [186]. NNs were also utilized to predict intermediate values to accelerate the convergence of the numerical solver [199]. MaxwellNet [126] was proposed to train a UNet with physics-informed loss to predict the scattered field based on the material permittivity of the free-space lens. WaveYNet [19] also adopted a 2-D UNet as the model trained with both data-driven supervision loss and Maxwell-equation-based physics-augmented loss to predict the optical field of silicon meta-lens. However, prior NN-based methods require explicit physics knowledge of the Maxwell PDEs and only learn a small field solution space conditioned on fixed parameters.

Learning PDEs via neural operators. Recently, neural operators have been proposed as new NN models that learn a family of parametric PDEs in the infinite-dimensional function space in a mesh-free and purely data-driven fashion. The Fourier neural operator (FNO) [123] approximates the nonlinear mapping from PDE observations to solutions through Fourier-domain kernel integral operations, achieving record-breaking performance and efficiency on a wide range of challenging applications. Several variants have been proposed to improve the performance and efficiency of the original FNO models, e.g., Factorized FNO [198], U-FNO [213], and multiwavelet-based neural operator [78].

4.2.2 Proposed Optical Simulation Framework `NeurOLight`

4.2.2.1 Understanding Optical Simulation for Photonic Devices

Waveguides can confine the incident laser beam and allow the optical signals to propagate and interfere with each other. Various optical components, e.g., couplers, shifters, and multi-mode interference (MMI) devices [31], can create phase shifts, magnitude modulation, and interference, especially useful for optical communication and neuromorphic computing. Analyzing how light wave propagates through those components are critical to device optimization and photonic integrated circuit design. Given a linear isotropic optical component, we will shine a time-harmonic continuous-wave light beam on its input ports and analyze the steady-state electromagnetic field distributions $\mathbf{E} = \hat{\mathbf{x}}\mathbf{E}_x + \hat{\mathbf{y}}\mathbf{E}_y + \hat{\mathbf{z}}\mathbf{E}_z$ and $\mathbf{H} = \hat{\mathbf{x}}\mathbf{H}_x + \hat{\mathbf{y}}\mathbf{H}_y + \hat{\mathbf{z}}\mathbf{H}_z$ in it, each of which includes horizontal (x), vertical (y), and longitudinal (z) components. We can solve the steady-state optical field $\mathbf{E}(\boldsymbol{r})$ and $\mathbf{H}(\boldsymbol{r})$ from the frequency-domain *curl-of-curl* Maxwell PDE under absorptive boundary conditions [100] (details in Appendix .3.1),

$$\begin{aligned} \big((\mu_0^{-1}\nabla\times\nabla\times) - \omega^2\epsilon_0\boldsymbol{\epsilon}_r(\boldsymbol{r})\big)\mathbf{E}(\boldsymbol{r}) &= j\omega\mathbf{J}_e(\boldsymbol{r}), \\ \big(\nabla\times(\boldsymbol{\epsilon}_r^{-1}(\boldsymbol{r})\nabla\times) - \omega^2\mu_0\epsilon_0\big)\mathbf{H}(\boldsymbol{r}) &= j\omega\mathbf{J}_m(\boldsymbol{r}) \end{aligned} \tag{4.1}$$

where $\nabla\times$ is the curl operator of a vector function, μ_0 is the vacuum magnetic permeability, ϵ_0 and $\boldsymbol{\epsilon}_r$ are the vacuum and relative electric permittivity, and $\mathbf{J}_m$ and $\mathbf{J}_e$ are the magnetic and electric current sources. FDFD simulation discretizes the continuous domain into an $M \times N$ mesh grid and solves the above linear equation $\mathbf{A}\mathbf{X} = \boldsymbol{b}$ to obtain the fields. Detailed forms of $\mathbf{A}$ and

$\boldsymbol{b}$ can be found in [100]. Solving for these optical fields exactly with a sparse matrix $\mathbf{A} \in \mathbb{C}^{MN \times MN}$ is prohibitively expensive and not scalable to large photonic structures. A fast surrogate model that predicts optical fields with high fidelity is of tremendous interest.

4.2.2.2 The proposed `NeurOLight` Framework

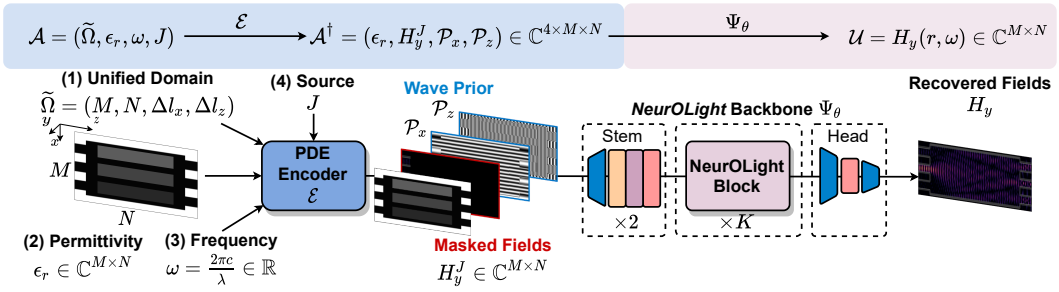


Figure 4.2: `NeurOLight` framework for optical field simulation. Real part is plotted for complex fields.

As shown in Figure 4.2, our `NeurOLight` framework models the optical field simulation problem as an infinite-dimensional-space mapping from Maxwell PDE observations $\mathcal{A} \in \mathbb{C}^{\Omega \times d_a}$ to the optical field solution $\mathcal{U} \in \mathbb{C}^{\Omega \times d_u}$. Here, Ω is the continuous 2-D physical solving domain, $\Omega = (l_x, l_z)$, typically in units of micrometers (μm), where the photonic device-of-interest can be tightly located. $\mathcal{A}$ and $\mathcal{U}$ take values with d_a and d_u dimensions, respectively. To learn the ground truth nonlinear mapping $\Psi^* : \mathcal{A} \rightarrow \mathcal{U}$, we construct `NeurOLight` with a PDE encoder $\mathcal{E}$ that produces compact PDE representations, followed by an efficient neural operator-based approximator Ψ_θ that

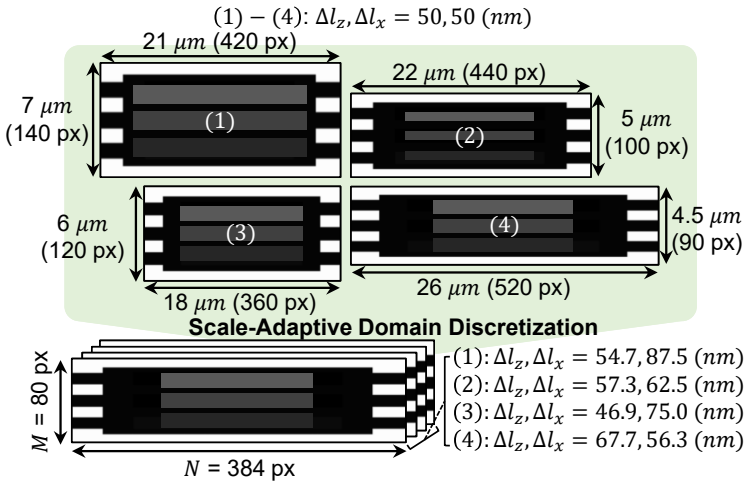


Figure 4.3: Scale-adaptive domain discretization enables generalization to different solving domain dimensions and efficient batched processing.

minimizes the empirical error on discrete PDE observable samples $a \sim \mathcal{A}$,

$$\theta^* = \min_{\theta} \mathbb{E}_{a \sim \mathcal{A}} \big[\mathcal{L} \big(\Psi_\theta(\mathcal{E}(a)), \Psi^*(a) \big) \big]. \tag{4.2}$$

4.2.2.3 Scale-Adaptive Domain Discretization: $\Omega \to \widetilde{\Omega}$

To generalize to PDEs in *different physical domains* and support batched parallel inference, we adopt an $M \times N$ discrete unified domain $\widetilde{\Omega} = (M, N, \Delta l_x, \Delta l_z)$ with an *adaptive mesh granularity*, i.e., with grid steps $\Delta l_x = l_x/M$ and $\Delta l_z = l_z/N$. As shown in Figure. 4.3, multiple photonic devices with different physical dimensions are normalized to the same $\widetilde{\Omega}$. Their original physical dimensions can be elegantly encoded into the re-calculated mesh granularities. This unified discrete domain gives `NeurOLight` the flexibility to handle parallel inference on different physical domain dimensions, unlike prior work [126, 19] that requires time-consuming model retraining once the physical

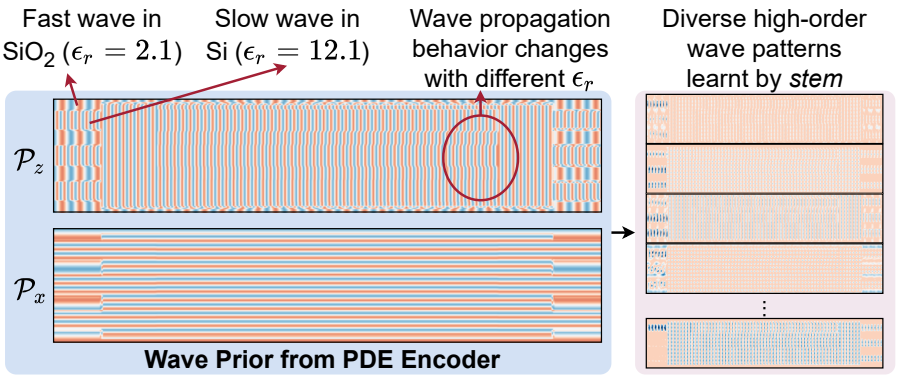


Figure 4.4: Wave prior as joint PDE representations.

domain changes.

4.2.2.4 Joint PDE Representation: $\mathcal{A} \to \mathcal{A}^\dagger$

After we define a unified solving domain, we need to construct effective PDE representations that describe the raw observations $\mathcal{A} = (\widetilde{\Omega}, \boldsymbol{\epsilon}_r, \omega, \mathbf{J})$. The relative permittivity distribution can be simply represented by $\boldsymbol{\epsilon}_r \in \mathbb{C}^{M\times N}$. However, how to compactly encode other parameters, i.e., $(\widetilde{\Omega}, \omega, \mathbf{J})$, remains a non-trivial challenge. Let us first consider what makes a good representation. First, it needs to be compatible with the model input, i.e., it can be fused with the 2-D image representation in a *compact* way. Second, it is preferred to reveal the physical essence of the parameters and inject useful *prior knowledge* that helps model generalization. Based on the above considerations, we propose a PDE encoder $\mathcal{E} : \mathcal{A} \to \mathcal{A}^\dagger$ that converts the raw observations to a joint PDE representation $\mathcal{A}^\dagger = (\boldsymbol{\epsilon}_r, \mathbf{H}_y^J, \mathcal{P}_x, \mathcal{P}_z)$.

Encoding $(\widetilde{\Omega}, \boldsymbol{\epsilon}_r, \omega)$ via wave prior. The intuition behind the wave prior

design is that the vacuum angular frequency $\omega = \frac{2\pi c}{\lambda}$ and electric permittivity $\boldsymbol{\epsilon}_r$ together decide the physical light wavelength inside the material, i.e., $\lambda' = \lambda/\sqrt{\boldsymbol{\epsilon}_r}$. The mesh granularity determines how many pixels can depict a wave period along both directions, i.e., $(\lambda'/\Delta l_x, \lambda'/\Delta l_z)$. Therefore, as shown in Figure 4.4, we construct artificial wave patterns, named *wave prior*, as $\mathcal{P}_z = e^{j\frac{2\pi\sqrt{\boldsymbol{\epsilon}_r}}{\lambda}\mathbf{1}z^T\Delta l_z}$ and $\mathcal{P}_x = e^{j\frac{2\pi\sqrt{\boldsymbol{\epsilon}_r}}{\lambda}x\mathbf{1}^T\Delta l_x}$, where $x = (0, 1, \cdots, M-1)$ and $z = (0, 1, \cdots, N-1)$. The wave prior jointly translates the $(\widetilde{\Omega}, \boldsymbol{\epsilon}_r, \omega)$ pair to a unified representation with strong prior knowledge, significantly reducing the learning difficulty on overly-abstract raw observations. We note that the `NeurOLight` stem learns complex combinations of the wave prior and generates diverse high-order wave patterns for later feature transformation.

Masked image modeling-inspired light source (J) encoding. In the optical simulation, *light source* **J** will be injected by shining light on the input waveguides of the photonic devices as stimuli to the system. **J** will have a vacuum angular frequency ω and a polarization mode. For example, in the transverse magnetic (TM) mode, we have $\mathbf{H}_x = \mathbf{H}_z = 0$. Thus we focus on the prediction of $\mathbf{H}_y$. However, as shown in Figure 4.5, **J** is a combination of multiple length-w 1-D vectors, where w is the input port width, placed at the input waveguides, which is hard to be encoded in

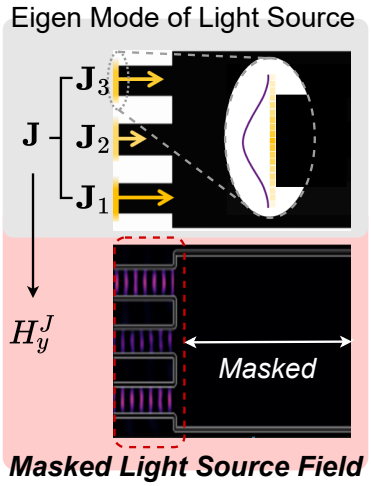


Figure 4.5: Masked light source modeling.

the image prediction flow. Therefore, we borrow the idea of *masked image modeling* [18] to light source encoding. In the source representation $\mathbf{H}^J$, we only maintain the fields in the input waveguides before entering the key region of the device, which is easy to obtain and irrelevant to the structure it enters into, and mask out all the fields after. In this way, the field prediction task translates to a *masked field restoration* task conditioned on the input light source as a *hint*.

4.2.2.5 Efficient `NeurOLight` Model Architecture: Ψ_θ

Convolutional stem. The proposed `NeurOLight` architecture starts with a convolutional stem $\mathcal{S} : a^\dagger(\boldsymbol{r}) \to v_0(\boldsymbol{r}), \forall \boldsymbol{r} \in \Omega$ that encodes each complex-valued observation sample $a^\dagger(\boldsymbol{r}) \in \mathbb{C}^{4\times M\times N}$ to a real-valued representation $v_0(\boldsymbol{r}) \in \mathbb{R}^{C\times M\times N}$. Lightweight blueprint convolutions [79] are used to perform local wave pattern transformation with a low hardware cost.

Cross-shaped `NeurOLight` block. In the projected C-dimensional space, we place K cascaded `NeurOLight` blocks to gradually restore the complex light field in the frequency domain as $v_0(\boldsymbol{r}) \to v_1(\boldsymbol{r}) \to \cdots \to v_K(\boldsymbol{r})$. Each `NeurOLight` block is formulated as

$$\begin{aligned} v_{k+1}(\boldsymbol{r}) &:= \texttt{FFN}\big((\mathcal{K}v_k)(\boldsymbol{r})\big) + v_k, \ \forall \boldsymbol{r} \in \Omega; \\ (\mathcal{K}v_k)(\boldsymbol{r}) &= \int_\Omega \kappa(\boldsymbol{r}_1, \boldsymbol{r}_2) v_k(\boldsymbol{r}_2) \mathrm{d}v_k(\boldsymbol{r}_2), \forall \boldsymbol{r}_1 \in \Omega, \end{aligned} \tag{4.3}$$

where $\mathcal{K}$ is a learnable kernel integral transform, and $\texttt{FFN}(\cdot)$ is a feedforward network. When the kernel satisfies $\kappa(\boldsymbol{r}_1, \boldsymbol{r}_2) = \kappa(\boldsymbol{r}_1 - \boldsymbol{r}_2)$, the above integral kernel operator is equivalent to a spatial-domain 2-D convolution, which can be

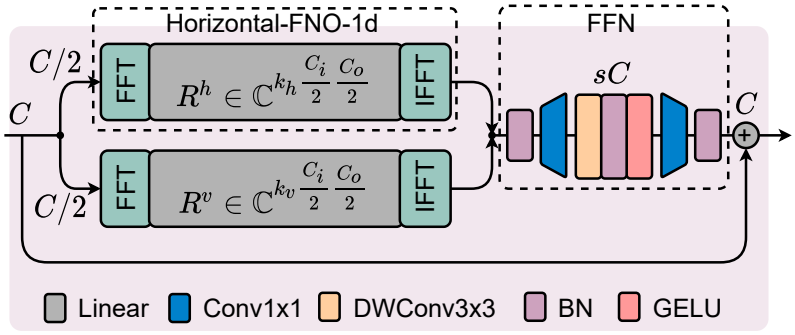


Figure 4.6: `NeurOLight` backbone model design.

efficiently computed by using Fourier transform $\mathcal{F}(\cdot)$ [123]. A clear downside of the original 2-D FNO is the huge parameter cost, i.e., $\mathcal{F}(\kappa)(\boldsymbol{r}) \in \mathbb{C}^{k_v \times k_h \times C \times C}$, and the resultant severe overfitting issues. To improve the model *efficiency and generalization* simultaneously, we introduce a *cross-shaped Fourier neural operator*, shown in Figure 4.6. The input feature is first bi-partitioned along the channel dimension into two chunks $v_k(\boldsymbol{r}) = [v_k^h(\boldsymbol{r}); v_k^v(\boldsymbol{r})]$, representing horizontal and vertical patterns, and 1-D FNO is applied to both directions,

$$\begin{aligned}
(\mathcal{K}^h v_k^h)(\boldsymbol{r}) &= \mathcal{F}_z^{-1}\big(\mathcal{F}_z(\kappa^h) \cdot \mathcal{F}_z(v_k^h)\big)(\boldsymbol{r}) = \mathcal{F}_z^{-1}\big(R^h(z) \cdot \mathcal{F}_z(v_k^h(\boldsymbol{r}))\big), \forall z \in \Omega_z, \forall r \in \Omega, \\
(\mathcal{K}^v v_k^v)(\boldsymbol{r}) &= \mathcal{F}_x^{-1}\big(\mathcal{F}_x(\kappa^v) \cdot \mathcal{F}_x(v_k^v)\big)(\boldsymbol{r}) = \mathcal{F}_x^{-1}\big(R^v(x) \cdot \mathcal{F}_x(v_k^v(\boldsymbol{r}))\big), \forall x \in \Omega_x, \forall r \in \Omega, \\
(\mathcal{K} v_k)(\boldsymbol{r}) &= [(\mathcal{K}^h v_k^h)(\boldsymbol{r}); (\mathcal{K}^v v_k^v)(\boldsymbol{r})].
\end{aligned} \tag{4.4}$$

We parametrize the Fourier kernels as lightweight complex-valued tensors $R^h(z) \in \mathbb{C}^{k_z \times \frac{C}{2} \times \frac{C}{2}}$ and $R^v(x) \in \mathbb{C}^{k_x \times \frac{C}{2} \times \frac{C}{2}}$. These orthogonal 1-D kernel operations intrinsically perform spatial and channel-wise feature aggregation in an interleaved way that are able to provide global receptive fields to achieve long-distance modeling. Compared with $k_v k_h C^2$ parameters in the 2-D FNO, our cross-shaped `NeurOLight` block only has $\frac{(k_v + k_h + 8s)C^2}{4}$ parameters.

To increase nonlinearity and enhance local information interaction, we append an `FFN` block after the cross-shaped FNO. Inspired by the MixFFN designs in SoTA vision transformers [224], our `FFN` expands the channels by s times, performs local information aggregation via 3×3 depth-wise convolution (DWConv), activates using the GELU function, and projects it back to C channels.

Projection head. In the end, two point-wise convolutional layers are used to project $v_K(\boldsymbol{r})$ to the light field space $u(\boldsymbol{r}) = \mathcal{Q}(v_K(\boldsymbol{r}))$. A dropout layer is inserted to mitigate overfitting issues.

Loss function. Even with normalized light source power, optical fields tend to have distinct statistics. To balance the optimization effort among different fields, we adopt the normalized mean absolute error (N-MAE) as the objective $\mathcal{L}\big(\Psi_\theta(\mathcal{E}(a)), \Psi^*(a)\big) = (\|\Psi_\theta(\mathcal{E}(a)) - \Psi^*(a)\|_1)/\|\Psi^*(a)\|_1$.

4.2.2.6 Superposition-based Mixup for Better Data Efficiency and Generalization

The PDE observations $\mathcal{A}$ can cover a huge design space. Hence, the data efficiency and generalization of pure data-driven models raise a concern. Simply drawing large numbers of random training examples with all possible light sources has an intractable data acquisition cost. Standard augmentation techniques are effective in improving data efficiency and generalization on tasks with *natural images*; however, their direct application is *not compatible with PDE simulation*. Since $\Psi^*(a_i)$ is a highly nonlinear function of a_i and closely

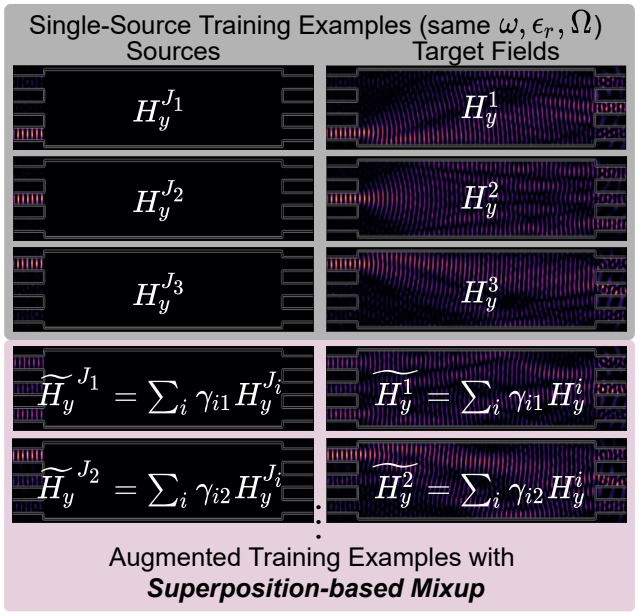


Figure 4.7: Data augmentation with superposition-based mixup. Only the real part is plotted for each field.

related to the boundary conditions, simultaneously augmenting $\boldsymbol{\epsilon}_r$, Ω, ω, and $\mathbf{H}$, e.g., via cropping, distortion, or resizing, leads to *invalid* field solutions.

Interestingly, we notice that the photonic system satisfies the *superposition* property w.r.t. the light source,

$$\widetilde{\mathbf{H}} = \Psi^*(\widetilde{\mathbf{H}}^J) = \Psi^*(\sum_{i=1}^{|\mathbf{J}|} \gamma_i \mathbf{H}^{J_i}) = \sum_{i=1}^{|\mathbf{J}|} \gamma_i \Psi^*(\mathbf{H}^{J_i}). \tag{4.5}$$

Based on this, we only involve *single-input simulation data* in the training set and dynamically mix multiple ($|\mathbf{J}|$) input sources via *Superposition-based Mixup*, as shown in Figure 4.7,

$$\begin{pmatrix} \widetilde{\mathbf{H}}^{J_1} & \dots & \widetilde{\mathbf{H}}^{J_{|\mathbf{J}|}} \\ \widetilde{\mathbf{H}}^1 & \dots & \widetilde{\mathbf{H}}^{|\mathbf{J}|} \end{pmatrix}^T = \boldsymbol{\Gamma} \begin{pmatrix} \mathbf{H}^{J_1} & \dots & \mathbf{H}^{J_{|\mathbf{J}|}} \\ \mathbf{H}^1 & \dots & \mathbf{H}^{|\mathbf{J}|} \end{pmatrix}^T, \tag{4.6}$$
$$\boldsymbol{\Gamma} \in \mathbb{C}^{|\mathbf{J}| \times |\mathbf{J}|}, \|\boldsymbol{\Gamma}_j\|_2 = 1, \phi(\gamma_{j1}) = 0, \forall j \in [|\mathbf{J}|].$$

At each iteration, we randomly generate the mixup coefficient matrix $\mathbf{\Gamma}$ and make it have a unit row-wise ℓ_2-norm for light field power normalization. The global phase of the complex-valued optical field is normalized by forcing the first input port always to have a phase that equals 0. In this way, `NeurOLight` learns how multiple incident light sources interfere with one another. Once `NeurOLight` can generalize to arbitrary light sources, multi-source simulation only needs *an efficient one-shot inference with superposed source fields* instead of explicitly accumulating $|\mathbf{J}|$ single-source inference results.

4.2.3 Experimental Results

4.2.3.1 Experiment Setup

Datasets. We focus on widely applied multi-mode interference (MMI) photonic devices. We select MMIs with rectangular tunable control pads (*Tunable MMI*). The permittivity of the tuning region can be programmed by external signals, so this family of devices exemplifies photonic structures with reconfigurable transmissions [116]. We also evaluate MMIs with rectangular etched cavities (*Etched MMI*) [186], which exemplifies another popular category of passive sub-wavelength photonic devices with fixed yet highly discrete permittivity distributions. We use an open-source CPU-based 2-D FDFD simulator angler [100] to simulate optical fields for randomly generated MMIs as our dataset. Details on dataset generation are in Appendix .3.2.

Table 4.1: Comparison of parameter count, train error, and test error on two benchmarks among four different models.

Benchmarks	Model	#Params (M) ↓	Train Err ↓	Test Err ↓
Tunable MMI	UNet [126, 19]	3.47	0.776	0.801
	FNO-2d [123]	3.29	0.231	0.244
	F-FNO [198]	3.16	0.272	0.292
	NeurOLight	**1.58**	**0.145**	**0.122**
Etched MMI	UNet [126, 19]	3.47	0.779	0.792
	FNO-2d [123]	3.29	0.601	0.648
	F-FNO [198]	3.16	0.411	0.525
	NeurOLight	**2.11**	**0.376**	**0.387**
Average Improvement		**-44.2%**	**-49.1%**	**-53.8%**

4.2.3.2 Main Results

In Table 4.1, we compare four models: (1) UNet [126, 19], (2) a 5-layer FNO-2d [123], (3) a 12-layer factorized FNO (F-FNO) [198], and (4) our `NeurOLight`. Detailed training settings and model architectures can be found in Appendix .3.3 and Appendix .3.4, respectively. On these benchmarks, `NeurOLight` outperforms UNet and prior SoTA FNO variants by **53.8%** lower test error with **44.2%** fewer parameters on average.

Results on Tunable MMI. On tunable MMI, `NeurOLight` achieves the best prediction error with only half the parameter cost. Figure 4.8 visualizes the field prediction for one test MMI. UNet is significantly limited by its small receptive field and lack of long-distance modeling capability, thus failing to predict the full field even with the hint of wave prior. As representative neural operators, FNO-2d and factorized FNO (F-FNO) manifest the superior advantages of the Fourier-domain kernel integral operations, showing consid-

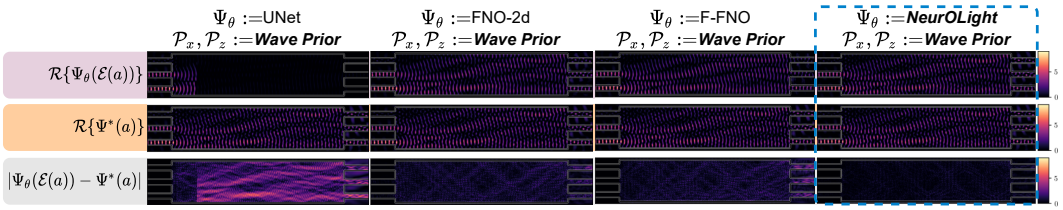


Figure 4.8: Visualization on one test tunable MMI. ($\Delta l_x = 83.1\ nm, \Delta l_z = 70.8\ nm, \lambda = 1.54\ \mu m$).

erably lower prediction errors than their CNN counterparts. However, given the parameter budget (∼3 M), the 5-layer FNO-2d only has 10 modes in the x-direction and 32 modes in the z-direction, which may not be enough to extract high-frequency waves. The 12-layer F-FNO adopts factorized 1-D Fourier kernel to save parameters; however, its modeling capability is limited by the lack of local feature extractors. Our `NeurOLight` blocks benefit from the global view of the cross-shaped 1-D kernel and local feature aggregation from convolutional `FFN` blocks. In the training curves in Figure 4.10(a), `NeurOLight` achieves the fastest convergence and best generalization among all models.

Results on Etched MMI. Compared with tunable MMIs, predicting the field in etched MMIs, even with 2× more training examples, is a much harder task given the complicated scattering at the cavity-silicon interface and the considerably larger and highly discrete design space, shown in Figure 4.9. Hence, we increase the model capacity of `NeurOLight` by using 16 layers. Among all prediction models, `NeurOLight` achieves the best results with 42% lower error while still saving 36% parameters on average.

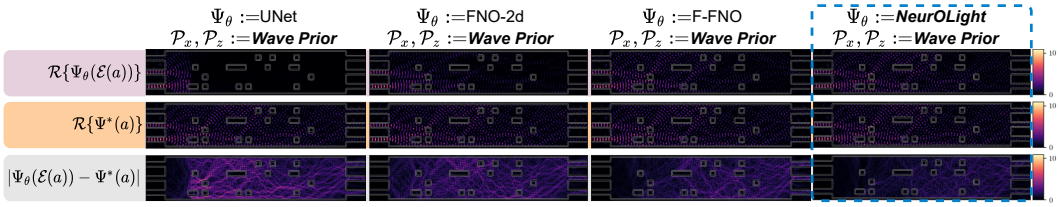
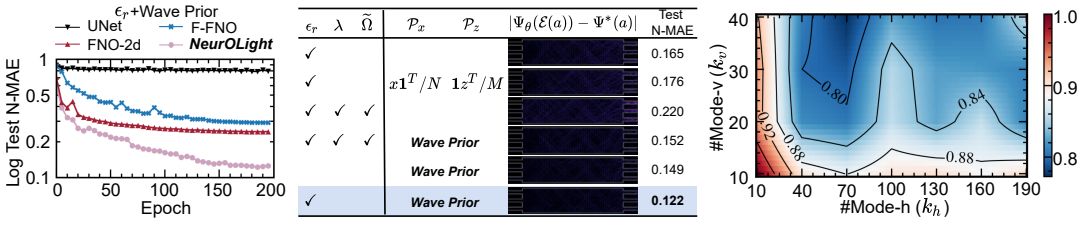


Figure 4.9: Visualization on one test etched MMI. ($\Delta l_x = 91.3\ nm, \Delta l_z = 89.1\ nm, \lambda = 1.55\ \mu m$).

4.2.3.3 Ablation Studies

PDE Encoding. In Figure 4.10(b), we extensively compare different combinations of PDE encoding methods. The first two methods only model the distribution over $\boldsymbol{\epsilon}_r$ conditioned on fixed wavelength and domain sizes like prior work [126, 19], which fail to generalize to examples in larger design space. With raw PDE parameters $(\boldsymbol{\epsilon}_r, \lambda, \widetilde{\Omega})$, the model finds it difficult to learn a generalizable representation, thus showing large errors on the test dataset. The last three combinations validate that permittivity and our wave prior are compact and effective encodings in our joint PDE representation method, while extra raw wavelength and domain information are redundant and even harmful.



(a) (b) (c)

Figure 4.10: (a) Test N-MAE curves of four models. (b) Our PDE encoder achieves the lowest error. (c) Normalized test error contour of a 8-layer `NeurOLight` with different # of Fourier modes.

Table 4.2: Ablation on proposed techniques. Each entry changes one technique independently. Runtime is averaged over multiple runs on 1 NVIDIA Quadro RTX 6000 GPU.

Variants	#Params (M)↓	Train Err ↓	Test Err ↓	Runtime (ms) ↓
NeurOLight	**1.58**	**0.145**	**0.122**	**12.1**
ConvStem → Lifting	1.58	0.156	0.134	11.9
Extra Parallel Conv Path	1.64	0.149	0.129	14.5
FFN → BN-GELU	1.37	0.469	0.446	6.3
Remove DWConv in `FFN`	1.57	0.164	0.144	10.6
Extra GELU After FNO	1.58	0.164	0.148	12.4
Remove DropPath	1.58	0.131	0.136	12.1

Fourier Modes. As shown in Figure 4.10(c), we perform a fine-grained exploration in the Fourier mode space to find the most suitable configuration. Unlike the flow prediction tasks evaluated in FNO [123] that only require a few modes, using inadequate Fourier modes fails to learn the terahertz high-frequency optical fields in the photonic device simulation task. However, using all Fourier series is not necessary and makes the model prone to overfitting issues. (k_h, k_v)=(70, 40) is the best setting that balances expressiveness and efficiency in our `NeurOLight`.

Cross-Shaped `NeurOLight` Block. In Table 4.2, we *independently* change one technique in the full `NeurOLight` model to verify each individual contribution. Our proposed essential techniques *synergistically* boost the modeling capacity and generalization. Compared to the linear lifting in FNO-2d which only performs point-wise projection, our lightweight convolution stem can extract complex high-order wave patterns with negligible runtime over-

head. Similar to U-FNO [213], we append an additional parallel convolution path alongside the cross-shaped FNO block; however, the extra 20% runtime penalty does not pay off. The proposed convolutional `FFN` significantly improves the nonlinearity and local feature extraction ability of `NeurOLight`. Changing it to a simple BatchNorm-GELU causes significant degradation. Different from the MLP-based FFN in F-FNO [198], our extra depthwise CONV is critical to local feature extraction and can reduce the test error by 16%. Note that an extra GELU after the FNO block will distort the feature in the low-dimensional space and have a negative impact on the performance [168]. Besides the dropout in the head, stochastic network depth [96] in the residual `NeurOLight` block is also effective in mitigating overfitting.

4.2.3.4 Discussion

Superposition-based Mixup. As shown in Table 4.3, without augmentation, `NeurOLight` only sees single-source training examples, thus failing to generalize when multiple sources are fused as a unified input source for *fast one-shot* prediction, named multi-source inference mode. A simple work-around would be to perform single-source prediction on $|\mathbf{J}|$ ports and superpose the resultant $|\mathbf{J}|$ fields, named single-source inference mode. When training on a large enough training set, this method indeed works. However, it quickly deteriorates as the training set reduces with a $|\mathbf{J}|$ times higher runtime cost for a $|\mathbf{J}|$-port device. With our dynamic superposition-based mixup, `NeurOLight` works well both in single-source and multi-source inference modes with supe-

Table 4.3: Test N-MAE of an 8-layer `NeurOLight` with different number of training examples. Multi-source inference mode has similar performance as the single-source method but shows 3× faster runtime on 3×3 MMIs.

Train Augmentation	Inference Mode	#Train Examples (K)					Runtime (ms)
		1.4	4.1	6.9	9.7	12.4	
None	Single-Source	0.346	0.257	0.202	0.198	0.194	23.8
	Multi-Source	0.892	0.882	0.880	0.865	0.873	8.3
Superposition Mixup	Single-Source	0.229	0.205	0.204	0.200	0.199	23.8
	Multi-Source	**0.230**	**0.208**	**0.206**	**0.202**	**0.202**	**8.3**

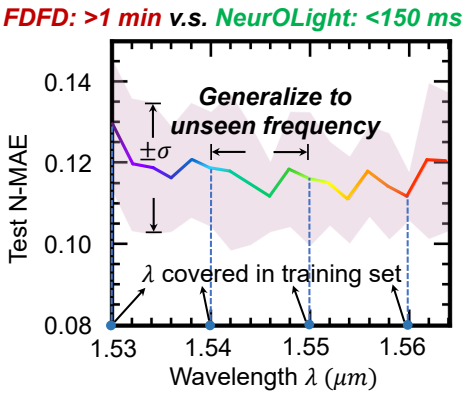


Figure 4.11: `NeurOLight` can generalize to unseen devices and wavelengths.

rior generalizability even with only 10% training data.

Spectrum Analysis. Spectroscopy is an important approach to understanding the broadband response of a photonic device. A traditional FDFD simulator has to sweep the spectrum with multiple simulations. In contrast, `NeurOLight` models the joint probability over wavelengths, and thus only needs to perform parallel inference with different $\omega = \frac{2\pi}{\lambda}$ values *at one shot*. Figure 4.11 demonstrates that, though `NeurOLight` is only trained with five selected wavelengths, it can generalize to unseen devices with unseen wave-

lengths. Sweeping in the standard C-band (1550 nm-1565 nm) with a 2 nm granularity, NeurOLight can finish within 150 ms, achieving 450× speedup over the FDFD simulator.

Device Adaptation. We evaluate the transferability of NeurOLight via device adaptation. In Figure 4.12, we transfer NeurOLight trained on 3-port MMIs to larger MMIs with 4 and 5 ports. Directly predicting new devices shows unsatisfying test error out of distribution (OOD). We adapt the model with 20-epoch fast linear probing and 30-epoch finetuning [113] on 3.7 K 4-port MMI examples and 4.6 K 5-port MMI examples. The model quickly transfers to new photonic devices with good prediction fidelity.

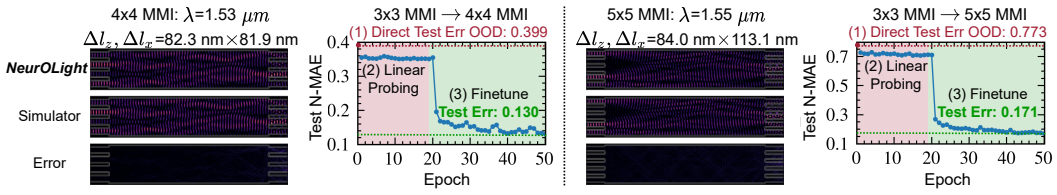


Figure 4.12: Device adaptation from 3-port to 4-/ 5-port MMI via linear probing and finetuning.

4.2.4 Summary

In this work, *for the first time*, a physics-agnostic neural operator, named NeurOLight, is proposed for ultra-fast parametric photonic device simulation. We propose a joint PDE encoder with wave prior and masked source modeling for compact PDE representation. Our lightweight cross-shaped NeurOLight backbone design achieves a superior balance between

modeling capability and parameter efficiency. In addition, our novel superposition-based mixup technique significantly boosts the data efficiency and model generalizability. Experiments show that NeurOLight outperforms prior DNN models with 53.8% better prediction fidelity and 44.2% less parameter cost, serving as an over 100× faster surrogate model to the numerical solvers in photonic device simulation. Currently, our model focuses on device-level simulation. As a future direction, we look forward to exploring the circuit-level simulation and utilizing our model to streamline the optimization loop for efficient AI-assisted optical circuit design automation.

4.3 ADEPT: Automatic Differentiable Design of Photonic Tensor Cores

After we have a fast device simulation oracle, we need to further enable the upper-level circuit design/optimization automation. Previous photonic tensor cores (PTCs) are all hand-designed based on matrix decomposition theory [171], which leaves a large design space unexplored and lacks the adaptability to meet various device specifications and hardware constraints. Specifically, the MZI-based PTC [171] is universal at the cost of high area

---

This ADEPT section is based on the following publication.

1. Jiaqi Gu, Hanqing Zhu, Chenghao Feng, Zixuan Jiang, Mingjie Liu, Shuhan Zhang, Ray T. Chen, and David Z. Pan, "ADEPT: Automatic Differentiable DEsign of Photonic Tensor Cores," ACM/IEEE Design Automation Conference (DAC), Jul. 2022.

I am the main contributor in charge of problem formulation, algorithm development, and experimental validations.

cost and low compute density. The FFT-based PTC [70, 71, 52] significantly reduces the usage of couplers and phase shifters. However, its area efficiency may not scale well with different PTC sizes and foundry process design kits (PDKs). As the PTC size scales up, the butterfly mesh in the FFT-based PTC introduces quadratically many waveguide crossings. If the foundry does not provide compact crossings, e.g., AIM photonics [195], those routing-related crossings will take up most of the circuit area. Besides, the butterfly mesh only has a logarithmic depth. Thus it restricts the matrix representability, which may lead to inadequate ONN learnability as PTC scales up.

Based on the above analysis, we observe strong demand for an automatic, efficient, and flexible PTC design methodology. Inspired by the success of neural architecture search (NAS) [128, 220] in the machine learning community, an interesting question to be answered is *whether we can jump out of the conventional manual design paradigm and use AI to "nurture" photonic neurocomputing with higher flexibility.* However, PTC design search encounters the following unique and difficult challenges. First, unlike NAS, where the NN architecture can be re-designed in software for application/platform adaptation at a relatively low cost, the photonic circuits need to be carefully designed before chip manufacturing and cannot be easily changed given the *high cost of chip tape-out.* Second, the PTC design can only be searched on a proxy NN model and learning task, but it has to be expressive and general enough to be *adapted to various AI workloads* after chip manufacturing. Third, the PTC circuit topology has an extremely large and highly discrete

search space, which casts *significant optimization difficulties* that prevent the direct application of off-the-shelf NAS algorithms to this unique problem.

To handle those challenges, in this work, we propose the *first* automatic differentiable search framework for photonic tensor core topology design, which we refer to as `ADEPT`. Our target is, given certain footprint constraints, we can efficiently search for a photonic circuit topology with good matrix *representability*, compact *footprint*, and high noise *robustness*. `ADEPT` enables differentiable PTC topology exploration via the following approaches: (1) we construct a probabilistic photonic `SuperMesh` to enable differentiable optimization in a huge and highly discrete PTC search space; (2) we adopt reparametrization and augmented Lagrangian method to learn waveguide connections; (3) binarization-aware training is employed to learn the location to place optical couplers; (4) `ADEPT` integrates the device specification from foundry PDKs into the `SuperMesh` training flow and optimizes PTC designs under various footprint constraints in a fully differentiable approach.

Our main contributions are as follows,

- In this work, *for the first time*, we automate the photonic tensor core design process and propose a differentiable framework to efficiently explore the PTC design space.
- To enable PTC topology search in a differentiable way, we introduce probabilistic photonic `SuperMesh` to search the PTC depth, an augmented Lagrangian method to learn waveguide connections, and binarization-aware

training to learn the coupler placement.

- The proposed ADEPT flow can adaptively generate various PTC designs based on different foundry PDKs and circuit footprint constraints. Experiments on various NN models and datasets show that the searched PTC topology outperforms prior hand-crafted structures with higher flexibility, competitive expressiveness, **2×-30×** smaller footprint, and superior noise robustness.

4.3.1 Preliminaries

4.3.1.1 Photonic Computing Premitivies

To perform computing in optics, we construct photonic integrated circuits (PICs) by cascaded optical devices.

Phase shifter (PS). Phase shifters can manipulate the effective refractive index of waveguides to produce a controlled phase shift ϕ on the propagating light signal x, $y = e^{-j\phi}x$. Phase shifters are typically active devices that are reprogrammable after PIC manufacturing.

Directional coupler (DC). 2-by-2 directional couplers (DC) can produce interference between two coherent light signals, whose transfer matrix is $T_{2\times2}$, where $T_{11} = T_{22} = t$ and $T_{12} = T_{21} = \sqrt{1-t^2}j$, and $t \in [0, 1]$ is the transmission coefficient. Couplers are typically passive devices that are fixed after chip fabrication.

Waveguide Crossing (CR). Given the 2-dimensional topology of the PIC, signal routing requires waveguide crossings. Unlike electrical wires, silicon

waveguides allow independent light propagation through crossed waveguides. Crossings of n waveguides can be described as an $n \times n$ permutation matrix. In photonic tensor cores, crossings can enhance signal flow and are typically not programmable after PIC fabrication.

Mach-Zehnder interferometer (MZI). MZI is a hand-designed structure consisting of two cascaded couplers and two phase shifters. MZI can perform arbitrary 2-D unitary projection, which is widely used to construct PTCs at the cost of a large circuit footprint.

4.3.1.2 Programmable PTCs

PTCs are essential building blocks in photonic accelerators constructed with passive and active optical devices. Current PTC topologies are hand-designed and barely involve any automation. Various [171, 30] MZI meshes were proposed to realize arbitrary $N \times N$ unitary matrices using $N(N-1)/2$ cascaded MZIs. Based on this, a weight matrix can be decomposed using SVD and mapped onto MZI meshes. Besides this universal photonic mesh design, a Fourier transform (FFT)-based PTC design [70, 71] was introduced to realize restricted linear operations with a butterfly-style mesh topology. This design utilizes basic optical components, i.e., `PS`, `DC`, and waveguide crossings `CR` without large MZIs to reduce footprint.

PTC designs need to consider device specification in foundry PDKs to honor circuit footprint constraints. Different foundries, e.g., AMF [2] and AIM photonics [195], provide devices of considerably different sizes, which makes it

challenging to manually search for good PTC designs that fit the area budget. This motivates us to provide an automatic solution for PDK-adaptive PTC design.

4.3.1.3 Differentiable Neural Architecture Search

Differentiable neural architecture search (DNAS) is widely adopted to automate the manual process of DNN architecture design with high efficiency. DNAS relaxes the discrete search space into continuous representation, such that the architecture can be optimized with gradient-based methods. DARTS [128] enables DNAS by using a softmax function to relax the categorical choice of candidate operations. FBNet [220] represents the search space by a stochastic SuperNet and then applies DNAS to discover low-latency DNN designs.

Recently, O-HAS [118] proposed an optical accelerator search framework that can automatically generate the optimal accelerator architecture. Different from our PTC circuit topology design, O-HAS focuses on searching for a mapping strategy to implement DNN models with manually-designed PTCs.

To the best of our knowledge, automated PTC design flow remains unexplored. It will be promising to develop a flexible and efficient framework to automatically search PTC topologies with high expressiveness, compact footprint, and good noise robustness, adaptive to various PDKs and footprint constraints.

4.3.2 Automatic Photonic Tensor Core Design Framework **ADEPT**

4.3.2.1 Problem Formulation

Our target is to use basic optical components, including `DC`, `PS`, and `CR`, to design a photonic mesh with a controlled footprint that can construct ONNs with high expressiveness, formulated as follows,

$$\begin{aligned}
&\min_{\alpha\in\mathcal{A}} \mathcal{L}\left(W^{*\alpha};\ \mathcal{D}^{val}\right), \quad \alpha=(B^U,B^V,\mathcal{P},\mathcal{T})\\
\text{s.t. }\ &W^*=\operatorname*{argmin}_{W} \mathcal{L}(W^{\alpha};\ \mathcal{D}^{trn}),\quad F_{min}\le\mathcal{F}(\alpha)\le F_{max},\\
&W^{\alpha}\in\mathbb{C}^{M\times N}=\left\{W^{\alpha}_{pq}\right\}_{p=1,q=1}^{p=P,q=Q}=\left\{U^{\alpha}_{pq}\Sigma_{pq}V^{\alpha}_{pq}\right\}_{p=1,q=1}^{p=P,q=Q},\\
&B^U,B^V\in[B_{min}/2,B_{max}/2],W_{pq}\in\mathbb{C}^{K\times K},\\
&\mathcal{P}=(\cdots,\mathcal{P}_b,\cdots,\mathcal{P}_{B^U+B^V}),\mathcal{T}=(\cdots,\mathcal{T}_b,\cdots,\mathcal{T}_{B^U+B^V}).
\end{aligned} \tag{4.7}$$

The weight matrix W in an ONN layer is partitioned into $K\times K$ sub-matrices. Each sub-matrix is constructed by two unitaries U^{α}_{pq} and V^{α}_{pq} and a diagonal matrix Σ_{pq}. The layout topology α of two unitaries is the primary search target, shared among all blocks.

4.3.2.2 Search Space Specification

Our search space focuses on the *tensor core circuit topology*, not layer configurations like conventional NAS work. As illustrated in Fig. 4.13, we define the following block-wise search space for the unitaries,

$$U^{\alpha}_{pq}=\prod_{b=1}^{B^U}\mathcal{P}_b\mathcal{T}_b\mathcal{R}(\Phi^b_{pq}),\quad V^{\alpha}_{pq}=\prod_{b=B^U+1}^{B^U+B^V}\mathcal{P}_b\mathcal{T}_b\mathcal{R}(\Phi^b_{pq}). \tag{4.8}$$

Unitaries U and V consist of B^U and B^V blocks, respectively. For simplification, we focus on U and refer to $\{B^U,B^V\}$ as B thereafter.

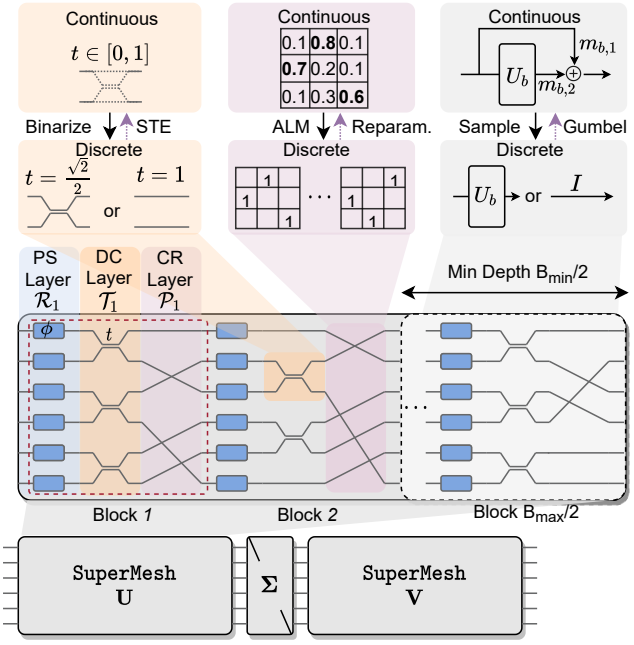


Figure 4.13: Overview of the probabilistic photonic `SuperMesh`.

The first structure in the block is a column of K phase shifters, which can be described by a diagonal matrix $\mathcal{R}(\Phi_{pq}^b)$,

$$\mathcal{R}(\Phi_{pq}^b) = \texttt{diag}(e^{-j\phi_1}, \cdots, e^{-j\phi_K}). \tag{4.9}$$

The second structure in the block is a column of 2-by-2 directional couplers T's placed from the s_b-th waveguide, which can be described by a block diagonal matrix $\mathcal{T}_b$. We will only include 50:50 DCs in our design, i.e., $t = \sqrt{2}/2$. This coupler column enables information interaction between adjacent waveguides. Besides, cascading `DC` layers in an *interleaved* way naturally allows more light signals to interfere with each other. Thus we have $s_b = 1$ if b is even and $s_b = 0$ if b is odd, as shown in Fig. 4.13.

The last layer in the block is designed for pure waveguide routing. This layer consists of a network of waveguide crossings, whose transfer matrix belongs to the permutation matrix family,

$$\mathcal{P}_b \in \{0,1\}^{K\times K}, \sum_j \mathcal{P}_b^{i,j} = 1 \ \forall i \in [K], \sum_i \mathcal{P}_b^{i,j} = 1 \ \forall j \in [K]. \tag{4.10}$$

The search space for $\mathcal{P}$ is extremely large since B_{max} size-K permutations contain total $(K!)^{B_{max}}$ possible combinations.

In summary, one unitary photonic mesh contains B blocks, each including a `PS` layer, a `DC` layer, and a `CR` layer. The topology α includes the number of blocks B^U and B^V, the waveguide connections $\mathcal{P}$ in the permutation layer, and the locations to put directional couplers described by $\mathcal{T}$. The total search space is $\mathcal{O}\big((K \cdot K!/2)^{B_{max}}\big)$.

4.3.2.3 Fully Differentiable `SuperMesh` Training

To solve the highly discrete PTC topology design problem in such an enormous search space, we propose a differentiable `SuperMesh` training flow `ADEPT` in Fig. 4.14.

The total optimization variables in the `SuperMesh` training contain (1) diagonal matrix Σ, (2) phases Φ in the `PS` layer, (3) directional couplers $\mathcal{T}$ in the `DC` layer, (4) permutation matrices $\mathcal{P}$ of the `CR` layer, (5) the number of blocks B. Jointly optimizing all those continuous and discrete variables is highly ill-conditioned, leading to prohibitive optimization difficulty. We separate them into two sets: (1) Σ, Φ, $\mathcal{T}$, and $\mathcal{P}$ belong to the `SuperMesh` *weights*,

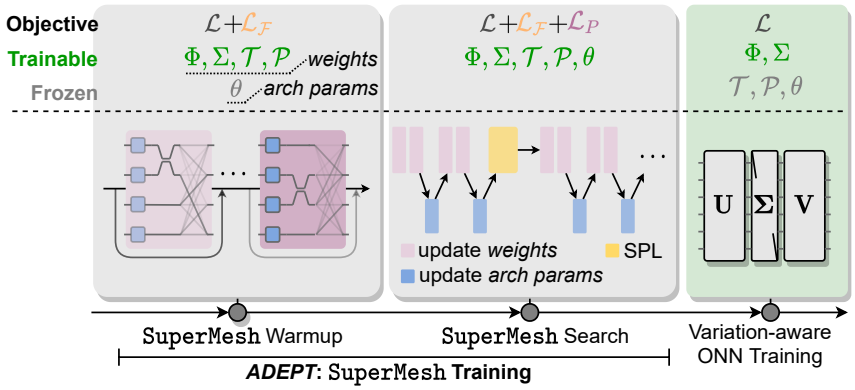


Figure 4.14: The proposed photonic SuperMesh training flow ADEPT, followed by variation-aware ONN training.

and (2) B belong to the *architecture parameter* group. The entire ADEPT flow contains two stage, shown in Fig. 4.14. The first SuperMesh *Warmup* stage only optimizes *weights* for initial exploration. The second SuperMesh *Search* stage optimizes two parameter groups alternately. We periodically enter the SuperMesh *weight training* phase to optimize Σ, Φ, $\mathcal{T}$, and $\mathcal{P}$ and switch to the *architecture parameter training* phase to search B. After ADEPT SuperMesh training, we apply variation-aware training to target ONN models with the searched PTC topology. Now we introduce how to optimize those variables one by one.

Optimize SuperMesh Depth B. The depth of SuperMesh can be relaxed by constructing a stochastic super block. During the inference, the b-th block U_b is either sampled and executed ($U_{b,1}$) or skipped as an identity projection

$(U_{b,2})$ with the probability of

$$P_{\theta_b}(U_b = U_{b,i}) = e^{\theta_{b,i}} \Big/ \sum_i e^{\theta_{b,i}}. \tag{4.11}$$

The probability distribution of block-b is parametrized by the sampling coefficient θ_b. The forward propagation of the b-th block is,

$$x_{b+1} = \sum_{i=1}^{2} m_{b,i} U_{b,i} x_b, \quad U_{b,1} = I,\ U_{b,2} = \mathcal{P}_b \mathcal{T}_b \mathcal{R}_b, \tag{4.12}$$

where the variable $m_{b,i}$ determines the probability to select the b-th block. Therefore, instead of searching B in the discrete space, the problem can be relaxed to the optimization of the probability P_θ. Gumbel-Softmax (GS) trick [220] is employed as follows,

$$m_{b,i} = \texttt{GumbelSoftmax}(\theta_{b,i}|\theta_b) = e^{(\theta_{b,i}+g_{b,i})/\tau} \Big/ \sum_i e^{(\theta_{b,i}+g_{b,i})/\tau}. \tag{4.13}$$

`Softmax` achieves continuous relaxation, and the Gumbel noise $g_{b,i}$ introduces stochasticity for better exploration controlled by the temperature τ. Note that the depth B has a range of $[B_{min}/2, B_{max}/2]$. Hence, the `SuperMesh` U consists of $(B_{max}/2)$ super blocks to upper-bound the search space. Meanwhile, the last $(B_{min}/2)$ blocks are always sampled with 100% certainty to lower-bound the search space, i.e., $m_{b,2} = 1, \forall b > B_{max}/2 - B_{min}/2$.

Optimize Permutation Matrices $\mathcal{P}$. Permutations are hard to search directly due to the factorial and highly discrete search space. The discrete constraint in Eq. (4.10) has a continuous format [32],

$$\mathcal{P}_b \geq 0;\ \|\mathcal{P}_b^{i,:}\|_1 = \|\mathcal{P}_b^{i,:}\|_2, \forall i;\ \|\mathcal{P}_b^{:,j}\|_1 = \|\mathcal{P}_b^{:,j}\|_2, \forall j, \tag{4.14}$$

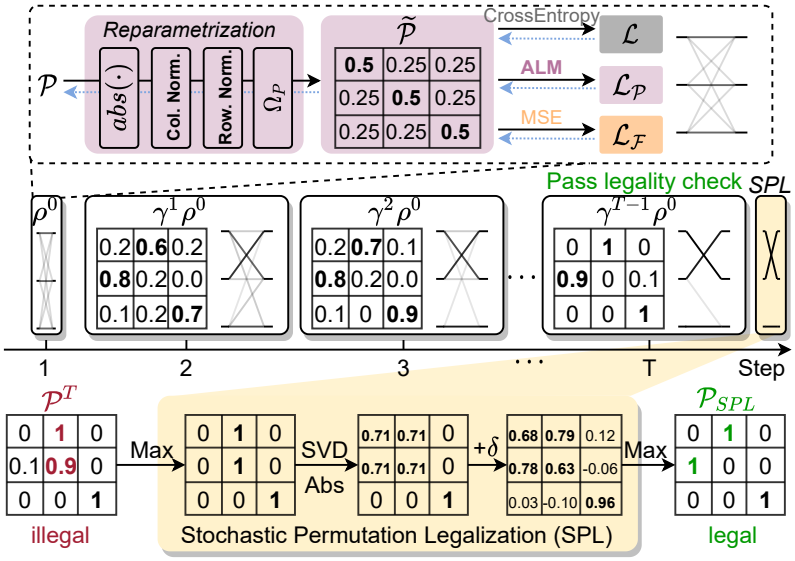


Figure 4.15: *Top*: permutation optimization procedure. *Bottom*: an example for stochastic permutation legalization (SPL).

where the row-wise and column-wise ℓ_1-norm equals to the ℓ_2-norm. Eq. (4.14) can be relaxed to its convex hull, i.e., Birkhoff polytope,

$$\mathcal{P}_b \geq 0,\ \mathbf{1}^T\mathcal{P}_b = \mathbf{1}^T,\ \mathcal{P}_b\mathbf{1} = \mathbf{1},\ \mathbf{1} = (1, \cdots, 1)^T. \tag{4.15}$$

As shown in Fig. 4.15, we use 1) *reparametrization* to approximately bound $\mathcal{P}$ in the Birkhoff polytope and 2) *augmented Lagrangian method* (ALM) to push $\mathcal{P}$ to a real permutation. Hence we enable differentiable permutation optimization during the *SuperMesh weight* training phase. We add an extra ALM term $\mathcal{L}_P$ in the objective,

$$\begin{aligned}\mathcal{L}_P = &\sum_{b=1}^{B_{max}}\sum_{i=1}^{K}\lambda_{b,i}^r\Delta\widetilde{\mathcal{P}}_b^{i,:} + \sum_{b=1}^{B_{max}}\sum_{j=1}^{K}\lambda_{b,j}^c\Delta\widetilde{\mathcal{P}}_b^{:,j}\\ &+\frac{\rho}{2}\sum_{b=1}^{B_{max}}\sum_{i=1}^{K}\lambda_{b,i}^r(\Delta\widetilde{\mathcal{P}}_b^{i,:})^2+\frac{\rho}{2}\sum_{b=1}^{B_{max}}\sum_{j=1}^{K}\lambda_{b,j}^c(\Delta\widetilde{\mathcal{P}}_b^{:,j})^2,\end{aligned} \tag{4.16}$$

where $\lambda^r, \lambda^c \in \mathbb{R}^{B_{max} \times K}$ are the row-wise and column-wise Lagrangian multipliers, ρ is the scalar quadratic penalty coefficient, and Δ denotes the difference between the ℓ_1 norm and ℓ_2 norm of the vector, e.g., $\Delta\widetilde{\mathcal{P}}_b^{i,:} = \|\widetilde{\mathcal{P}}_b^{i,:}\|_1 - \|\widetilde{\mathcal{P}}_b^{i,:}\|_2$. This is different from the standard ALM formulation as the quadratic term is also controlled by λ. In this way, the optimization is dominated by the task-specific loss at the beginning and gradually honors the constraint.

We *reparametrize* $\mathcal{P}_b$ as $\widetilde{\mathcal{P}}_b$ to simplify the constraints in Eq. (4.15). We (1) first apply an absolute operation to the relaxed matrix to guarantee non-negativity, (2) then apply column-/row-wise normalization, and (3) finally apply row-wise soft projection to force binarization,

$$\begin{aligned} &\mathcal{P}'_b = \frac{|\mathcal{P}_b|}{\mathbf{1}^T|\mathcal{P}_b|}, \quad \mathcal{P}''_b = \frac{\mathcal{P}'_b}{\mathcal{P}'_b\mathbf{1}}, \quad \widetilde{\mathcal{P}}_b = \Omega_P(\mathcal{P}''_b) \\ &\Omega_P(\mathcal{P}_b^{''i,j}) = \begin{cases} \texttt{Round}(\mathcal{P}_b^{''i,j}) & \text{if } \max(\mathcal{P}_b^{''i,:}) \geq 1-\epsilon, \\ \mathcal{P}_b^{''i,j} & \text{if } \max(\mathcal{P}_b^{''i,:}) < 1-\epsilon \end{cases}, \end{aligned} \tag{4.17}$$

where ϵ is the projection threshold, typically set to 0.05. The soft projection stops gradients when $\widetilde{\mathcal{P}}$ is very close to a real permutation, which is designed to avoid gradient instability issues caused by an overly large linear penalty term as λ quickly increases.

At each iteration in the `SuperMesh` *weight* training phase, we first update the relaxed permutation matrices using gradient-based methods, then we update the Lagrangian multipliers as follows,

$$\lambda^r_{b,i} \mathrel{+}= \rho\big(\Delta\widetilde{\mathcal{P}}_b^{i,:} + \frac{1}{2}(\Delta\widetilde{\mathcal{P}}_b^{i,:})^2\big), \quad \lambda^c_{b,j} \mathrel{+}= \rho\big(\Delta\widetilde{\mathcal{P}}_b^{:,j} + \frac{1}{2}(\Delta\widetilde{\mathcal{P}}_b^{:,j})^2\big). \tag{4.18}$$

Stabilize Optimization via Initialization and Normalization. The relaxed $\widetilde{\mathcal{P}}$ cannot guarantee orthogonality during optimization. Thus cascading

multiple such matrices ruins the orthogonality of U and V and causes training difficulty due to statistical instability. To mitigate it, we initialize $\mathcal{P}$ with a smoothed identity, i.e., $\mathcal{P}^0 = I(\frac{1}{2} - \frac{1}{2K-2}) + \frac{1}{2K-2}$, shown in Fig. 4.15. Note that initializing it with random permutations does not work since no gradients will flow back to zero entries. In addition, we propose a second technique to solve this via row-wise and column-wise ℓ_2 normalization on the constructed U and V, respectively. By doing this, the normalized U and V can approximate the properties of true unitaries. This normalization takes no effects when U and V converge to real unitaries but helps to stabilize the matrix statistics.

Scheduling Coefficient ρ. ρ determines the speed to increase λ. A large ρ quickly traps $\widetilde{\mathcal{P}}$ to a nearby suboptimal permutation. An overly small ρ has too weak constraints on the permutation. Thus we increase ρ as $\rho \leftarrow \rho\gamma^t, \forall t = 0, \cdots, T$, such that $\rho^T \approx 1e4 \cdot \rho^0$.

Stochastic Permutation Legalization (`SPL`). Due to high non-convexity in the problem Eq. (4.16), our ALM-based method does not guarantee convergence to a legal permutation. Instead, it may stuck at saddle points shown in Fig. 4.15. To force $\widetilde{\mathcal{P}}$ to a legal permutation after `SuperMesh` training, we propose the following stochastic permutation legalization (`SPL`),

$$\begin{aligned} PSQ^* &= \texttt{SVD}\big(\texttt{Softmax}(\mathcal{P}/\tau)\big)\big|_{\tau\to 0^+}, \\ \mathcal{P}_{\texttt{SPL}} &= \texttt{Softmax}\big((|PQ^*| + \delta)/\tau\big)\big|_{\tau\to 0^+}, \end{aligned} \tag{4.19}$$

where $\delta \sim \mathcal{N}(0, \sigma^2)$. We give an example in Fig. 4.15. The first `Softmax` binarizes the matrix. Then, the SVD-based projection pushes the solution away from saddle points. After that, random perturbations are added to

break the ties between different rows. The final `Softmax` pushes it into a legal permutation in a stochastic manner. We repeat the second equation by multiple times until we find a legal solution without introducing too many extra crossings.

Optimize Directional Couplers $\mathcal{T}$ The transmission coefficient t of each directional coupler in the `DC` layer is a binary optimization variable $t \in \{\frac{\sqrt{2}}{2}, 1\}$. t=1 represents a direct waveguide connection without placing a `DC`. We treat t as a `SuperMesh` *weight* and perform quantization-aware training to learn the `DC` layers. The `DC` binarization and its gradient are given as follows,

$$\begin{aligned} T(t_q) = T(\mathcal{Q}(t)),\ \mathcal{Q}(t) = (\texttt{sign}(t)+1) \times \frac{2-\sqrt{2}}{4} + \frac{\sqrt{2}}{2}, \\ \frac{\partial \mathcal{L}}{\partial t} = \min\Big(1, \max\big(-1, \frac{\partial \mathcal{L}}{\partial t_q} \times \frac{2-\sqrt{2}}{4}\big)\Big). \end{aligned} \tag{4.20}$$

Optimize Diagonal Matrix Σ and Phases Φ We treat the diagonal matrix Σ and phase shifter configurations Φ as the `SuperMesh` *weights*. During `SuperMesh` training, we simply apply the standard backpropagation to train them.

4.3.2.4 PDK-Adaptive Footprint-Constrained `SuperMesh` Optimization

An important hardware constraint we need to honor is the target photonic circuit footprint, given the component sizes from a foundry PDK. We solve the inequality footprint constraint by adding a *probabilistic footprint*

Table 4.4: Evaluate searched PTCs with different sizes and footprint targets on MNIST with a 2-layer CNN. The total block number is #Blk=$B^U + B^V$. #PS is omitted since we have #PS =K·#Blk. All footprint constraints follow $F_{min} = 0.8F_{max}$. ADEPT-a1 to ADEPT-a5 cover 5 different footprint targets with the device specification from AMF foundry PDKs. In the AMF PDKs [2], the footprint of PS, DC, and CR is 6800 μm^2, 1500 μm^2, and 64 μm^2, respectively. All footprint is reported in the unit of 1/1000 μm^2.

PTC Size	Metrics	MZI-ONN [171]	FFT-ONN [70]	ADEPT-a1	ADEPT-a2	ADEPT-a3	ADEPT-a4	ADEPT-a5
8×8	#CR/#DC/#Blk	0/112/32	16/24/6	24/17/5	17/19/6	26/27/8	27/36/11	33/41/13
	[F_{min}, F_{max}]	-	-	[240, 300]	[336, 420]	[432, 540]	[528, 660]	[624, 780]
	Footprint $\mathcal{F}$	1909	363	299	356	478	654	771
	Accuracy (%)	98.63	98.43	98.26	98.49	98.56	98.48	98.69
16×16	#CR/#DC/#Blk	0/480/64	88/64/8	45/28/4	68/43/6	127/59/8	174/71/10	131/85/12
	[F_{min}, F_{max}]	-	-	[480, 600]	[672, 840]	[864, 1080]	[1056, 1320]	[1248, 1560]
	Footprint $\mathcal{F}$	7683	972	480	722	967	1206	1441
	Accuracy (%)	98.65	98.25	98.16	98.40	98.24	98.56	98.57
32×32	#CR/#DC/#Blk	0/1984/128	416/160/10	223/60/4	333/87/6	628/178/8	691/150/10	717/179/12
	[F_{min}, F_{max}]	-	-	[960, 1200]	[1344, 1680]	[1728, 2160]	[2112, 2640]	[2496, 3120]
	Footprint $\mathcal{F}$	30829	2443	975	1457	1959	2445	2926
	Accuracy (%)	98.68	97.97	98.10	98.18	98.36	98.49	98.39

penalty term $\mathcal{L}_{\mathcal{F}}$,

$$
\begin{aligned}
\mathcal{L}_{\mathcal{F}} &= \begin{cases} \beta\Big(\mathbb{E}[\mathcal{F}_{\text{prox}}(\alpha)]/\widehat{F}_{max}\Big), & \mathbb{E}[\mathcal{F}(\alpha)] > \widehat{F}_{max}, \\ -\beta\Big(\mathbb{E}[\mathcal{F}_{\text{prox}}(\alpha)]/\widehat{F}_{min}\Big), & \mathbb{E}[\mathcal{F}(\alpha)] < \widehat{F}_{min}, \\ 0, & \text{otherwise}, \end{cases} \\
\mathbb{E}[\mathcal{F}(\alpha)] &= \sum_{b=1}^{B_{max}} m_{b,2}\mathcal{F}_b, \quad \mathbb{E}[\mathcal{F}_{\text{prox}}(\alpha)] = \sum_{b=1}^{B_{max}} m_{b,2}\mathcal{F}_{b,\text{prox}}, \\
\mathcal{F}_b &= \#\texttt{PS}(\mathcal{R}_b)\cdot\mathcal{F}_{\texttt{PS}} + \#\texttt{DC}(\mathcal{T}_b)\cdot\mathcal{F}_{\texttt{DC}} + \#\texttt{CR}(\mathcal{P}_b)\cdot\mathcal{F}_{\texttt{CR}}, \\
\mathcal{F}_{b,\text{prox}} &= \#\texttt{PS}(\mathcal{R}_b)\cdot\mathcal{F}_{\texttt{PS}} + \#\texttt{DC}(\mathcal{T}_b)\cdot\mathcal{F}_{\texttt{DC}} + \beta_{\texttt{CR}}\|\widetilde{\mathcal{P}}_b - I\|_2^2\cdot\mathcal{F}_{\texttt{CR}}, \\
\#\texttt{PS}(\mathcal{R}_b) &= K,\ \#\texttt{DC}(\mathcal{T}_b) = \sum_{i=1}^{(K-s_b)/2}\Big(\frac{2\mathcal{Q}(t_i)}{\sqrt{2}-2} + \frac{2}{2-\sqrt{2}}\Big),
\end{aligned}
\tag{4.21}
$$

where β is the penalty weight, and $\widehat{F}_{max}$ and $\widehat{F}_{min}$ is set to $0.95F_{max}$ and $1.05F_{min}$ to leave a 5% constraint margin. This penalty term allows SuperMesh to control its *expected footprint*. Now we give a detailed breakdown of our probabilistic footprint penalty.

Footprint of `PS`. As an active device, PS is not fixed after manufacturing. Instead, the phase shifts Φ are important weights to guarantee enough PTC reprogrammability and ONN expressiveness. Hence, we always assume a *full column* of PS, i.e., $\#\texttt{PS}(\mathcal{R}_b) = K$.

Footprint of `DC`. DC is typically fixed and not tunable. Hence the position to place a DC need to be determined during the PTC design stage. The footprint of a DC layer is a simple summation of all couplers parameterized by their binarized coefficient t_q, which is fully differentiable by using straight-through estimators.

Footprint of `CR`. The number of waveguide crossings, i.e., #CR, can be obtained by sorting rows of the permutation $\mathcal{P}_b$ to an identity I and finding the *minimum number of adjacent swaps*. However, this crossing counting procedure $\#\texttt{CR}\ (\mathcal{P}_b)$ itself is non-differentiable. When calculating the footprint penalty, we replace $\#\texttt{CR}(\mathcal{P}_b)\mathcal{F}_{\texttt{CR}}$ with a differentiable proxy term $\beta_{\texttt{CR}}\mathcal{F}_{\texttt{CR}}\|\widetilde{\mathcal{P}_b} - I\|_2^2$, where $\beta_{\texttt{CR}}$ is used to balance the penalty on DC and CR.

Analytical Bound of the `SuperMesh` Block Number. Given the device footprint specification, we can actually calculate the maximum/minimum footprint of each block. Based on the target footprint, we can find an analytical bound of the block number for our SuperMesh, i.e., B_{max} and B_{min}, without manual definition,

$$\begin{aligned} &\mathcal{F}_{b,min} = K\mathcal{F}_{\texttt{PS}} + \mathcal{F}_{\texttt{DC}}, \quad \mathcal{F}_{b,max} = \mathcal{F}_{b,min} + K\mathcal{F}_{\texttt{DC}}/2 + K(K-1)\mathcal{F}_{\texttt{CR}}/2, \\ &B_{max} = \lceil F_{max}/\mathcal{F}_{b,min} \rceil, \quad B_{min} = \lfloor F_{min}/\mathcal{F}_{b,max} \rfloor. \end{aligned} \tag{4.22}$$

4.3.3 Experimental Results

4.3.3.1 Experiment Setup

Datasets. We search PTCs on MNIST and evaluate on MNIST, FashionMNIST, SVHN [149], and CIFAR-10 datasets.

NN Models. We perform `SuperMesh` training on MNIST with a 2-layer CNN (C32K5-BN-ReLU-C32K5-BN-ReLU-Pool5-FC10), where C32K5 is a 32-channel convolution with a kernel size of 5×5. In variation-aware training, we use LeNet-5 and VGG-8.

Training Settings. We train `SuperMesh` for 90 epochs using Adam optimizer with an initial learning rate (lr) of 0.001 and a cosine lr scheduler. We set the weight decay rate to 1e-4 for Φ and Σ, and 5e-4 for θ. We exponentially decrease the Gumbel-softmax temperature τ from 5 to 0.5. We set 10 epochs in the `SuperMesh` *Warmup* stage. In the `SuperMesh` *Search* stage, we train weights and arch. params with a ratio of 3:1. In the permutation ALM, we set the initial ρ^0=(1e-7)×$K/8$. We set β and $\beta_{\texttt{CR}}$ to 10 and 100 in the footprint penalty. At the 50-th epoch, we force $\mathcal{P}$ to a legal permutation by stochastic permutation legalization (`SPL`). Then we continue the alternate `SuperMesh` training in the rest 40 epochs. During re-training, we sample a `SubMesh` from the learned distribution P_θ that satisfies the footprint constraints. Then we perform variation-aware training with Gaussian phase noises $\Delta\phi \sim \mathcal{N}(0, 0.02^2)$ to increase robustness.

Table 4.5: MNIST accuracy with 16×16 PTCs on AIM photonics PDKs [195], where $\mathcal{F}_{\texttt{PS}}$=2500 μm^2, $\mathcal{F}_{\texttt{DC}}$=4000 μm^2, and $\mathcal{F}_{\texttt{CR}}$=4900 μm^2.

PTC Size	Metrics	MZI-ONN [171]	FFT-ONN [70]	ADEPT-a0	ADEPT-a1	ADEPT-a2	ADEPT-a3	ADEPT-a4	ADEPT-a5
16×16	#CR/#DC/#Blk	0/480/64	88/64/8	15/35/5	1/58/8	26/58/8	17/92/13	25/99/14	89/111/16
	$[\mathcal{F}_{min}, \mathcal{F}_{max}]$	-	-	[384, 480]	[480, 600]	[672, 840]	[864, 1080]	[1056, 1320]	[1248, 1560]
	Footprint $\mathcal{F}$	4480	1007	414	557	679	971	1079	1520
	Accuracy (%)	98.77	98.10	98.15	98.30	98.32	98.55	98.64	98.72

4.3.3.2 Main Results

We search PTC topologies with the proposed ADEPT flow on three different PTC sizes (8×8, 16×16, 32×32) with various footprint constraints. We denote our searched PTC designs as ADEPT-a1 to ADEPT-a5. In Table 4.4, we compare our ADEPT-series to prior manual PTC designs, i.e., MZI-ONN [171] and FFT-based ONN [70, 71] on AMF foundry PDKs. For a fair comparison, the butterfly mesh in the FFT-based PTC is not limited to Fourier-transform but a general trainable transform [71]. On three PTC sizes, the searched ADEPT-series shows superior adaptability to various footprint constraints. Compared to the largest MZI-based PTC, our ADEPT-series shows competitive learnability with **2×-30×** footprint reduction. ADEPT-series outperforms the FFT-based PTC with higher expressivity, especially on large PTC sizes, and saves up to **2.5×** area. ADEPT shows superior adaptability to balance footprint and expressiveness.

Adapt PTCs to Different Foundry PDKs. To adapt ADEPT to different device specifications, we change the foundry PDK from AMF [2] to AIM photonics [195], which provides much larger waveguide crossings. In Table 4.5, ADEPT finds feasible PTC topology that avoids using many crossings to honor

Table 4.6: Adapt searched 16×16 PTCs to LeNet-5/VGG-8 and different datasets on AMF PDKs. Test accuracy (%) is given in the table. The PTC is searched on MNIST and a 2-layer CNN.

Model	Datasets	MZI [171]	FFT [70, 71]	ADEPT-a2	ADEPT-a4
Footprint		7683	972	722	1206
LeNet-5	FMNIST	87.33	85.87	85.89	87.07
	SVHN	69.91	65.04	65.26	69.20
	CIFAR-10	51.40	42.75	51.26	52.42
VGG-8	FMNIST	89.59	88.62	89.23	89.16
	SVHN	77.87	75.22	75.86	77.20
	CIFAR-10	68.90	63.57	66.30	68.50

the strict footprint constraints. The smallest ADEPT-a0 achieves comparable accuracy to the FFT-based PTC with **2.4**× smaller footprint. Compared to MZI-based PTC, our ADEPT-a5 is **2.9**× more compact with similar expressiveness.

Transfer to Different ONNs and Datasets. To further validate the expressiveness of ADEPT-series searched on a proxy NN model and dataset, we apply searched PTC structures to other NN architectures and more challenging datasets in Table 4.6. On three datasets with LeNet-5 and VGG-8, our searched 16×16 ADEPT-a2 and ADEPT-a4 significantly outperform FFT-based design with much higher accuracy and 26% footprint reduction. Compared to the MZI-based PTC, ADEPT-a4 can save over **84%** footprint with competitive performance.

Noise Robustness of Searched PTCs. In Fig. 4.16, we inject phase drifts into the circuit and perform variation-aware training on all PTC designs [252, 74]. Even with noise-aware training, the MZI-based ONN still suffers a severe

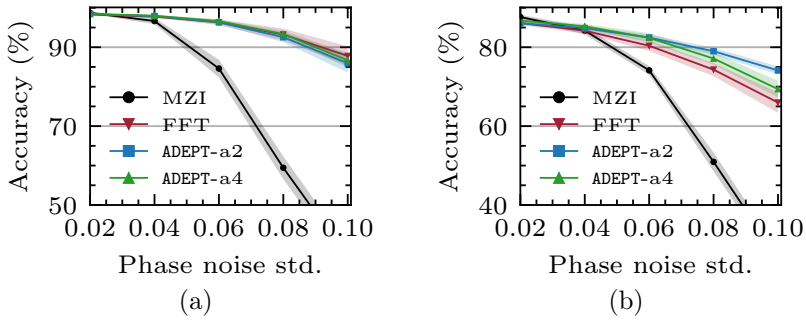


Figure 4.16: Robustness evaluation of 16×16 PTCs with various phase noise intensities. (a) 2-layer CNN on MNIST. (b) LeNet-5 on FMNIST. All models are trained with variation-aware training. The shadow marks $\pm 3\sigma$ uncertainty over 20 runs.

accuracy drop due to overly large PTC depth. In contrast, our searched PTCs show similar or even better noise robustness than the logarithmic-depth FFT-based design.

4.3.3.3 Ablation Studies

Permutation ALM. To better understand the permutation learning process, we scan different initial values of the ALM penalty coefficient ρ^0 and plot the optimization curves in Fig. 4.17(a). Our method is insensitive to the hyper-parameter settings and can stably converge with the proposed adaptive penalty scheduling.

Footprint Penalty. In Fig. 4.17(b), the expected PTC footprint is visualized with different penalty strengths. With β=$\sim$10, the expected footprint of `SuperMesh` can be well-bounded. If β is too small, most sampled PTC structures from P_θ will violate the constraint.

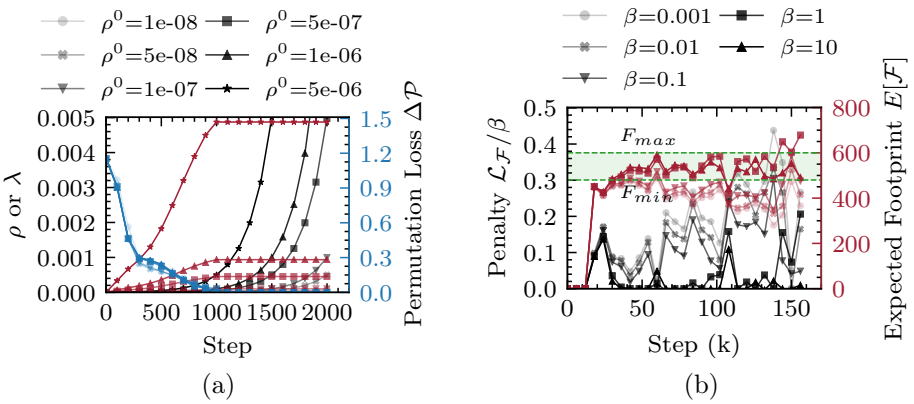


Figure 4.17: (a) Scan initial ρ in permutation ALM from 5e-8 to 5e-6. Red lines are averaged λ. Blue curves are permutation errors, i.e., the average difference between ℓ_1- and ℓ_2-norm. (b) Scan β in footprint penalty from 0.001 to 10. Red lines are expected footprint $\mathbb{E}[\mathcal{F}(\alpha)]$ of ADEPT-a1. Black curves are footprint penalty. The green region marks the constraint.

4.3.4 Summary

In this work, *for the first time*, we propose an automatic differentiable framework ADEPT for efficient photonic tensor core design. Our ADEPT constructs a probabilistic photonic SuperMesh, employs an augmented Lagrangian method to learn waveguide connections, and adopts binarization-aware training to search coupler locations. With a probabilistic footprint penalty method, ADEPT integrates circuit area constraints into SuperMesh training procedure to adapt the PTC to various device specifications and footprint constraints. Extensive experiments show the superior flexibility of ADEPT for automated PTC topology search adaptive to foundry PDKs. The searched PTC design outperforms prior manual designs with competitive expressiveness, **2×-30×** smaller footprint, and superior robustness. ADEPT

opens a new paradigm in photonic neurocomputing by "nurturing" photonic circuit design via AI and automation.

Chapter 5

Conclusion and Future Work

In this dissertation, we build a holistic framework for photonic AI computing platforms, while carrying out cross-layer device, circuit, architecture, and algorithm co-design and design automation, closing the virtuous cycle of photonics for AI and AI for photonics. In the photonics for ML acceleration aspects, specialized photonic neural network accelerators with customized devices/circuits/architectures, i.e., `FFT-ONN`, `SqueezeLight`, `O`^{2}`NN`, and `memory-efficient ONNs`, are proposed to reduce the area cost, improve the compute density, flexibility, and system-level energy efficiency. Customized on-chip training algorithms `FLOPS`, `MixedTrain`, and `L`^{2}`ight`, are proposed to enable gradient calculation in situ for efficient ONN on-chip learning to simultaneously solve noise robustness and adaptability issues.

On the AI for photonics side, we introduce an ML-assisted ultra-fast Maxwell equation-solving framework `NeurOLight` to accelerate photonic device simulation, enabling future ONN exploration with customized photonic devices. At the circuit level, we present the first automatic differentiable design flow `ADEPT` to search photonic circuit topology toward beyond-human compactness and noise robustness.

To enable the practical application of photonic computing systems in real-world applications, more technical challenges need to be addressed in the future. From the system's perspective, electronics-photonics integration is the most prominent challenge for optical neural accelerators. Most of the system-level complication still comes from the I/O and control. The overall system performance bottleneck is mainly on the data transaction from memory and analog-to-digital converters (ADCs). In terms of power consumption, nearly 50% of power is from electrical memory, and ADCs/DACs take another 20-30%, while the photonic circuit only consumes less than 10% total power [159]. The speed and power of memory and ADCs determine the overall benefits one could gain from optical neurocomputing.

The following are some possible research directions.

Scaling to Larger Models and More Advanced Tasks. Promising directions include (1) *trading universality for higher scalability* [70, 52, 260] by restricting the matrix parameter space with specialized circuit and device designs to balance the hardware cost and programmability; (2) designing customized photonic devices to achieve higher compute density and power efficiency, e.g., multi-operand MRRs [66], multi-operand MZIs [53], meta-lens [212], diffractive devices; (3) utilizing wavelength/time/mode-division multiplexing to reuse the hardware for more parallel operations, which is one of the major advantages of photonic integrated circuits compared to electronic computing units; (4) supporting large-scale NN models, e.g., large-scale attention-based Transformer models, and applying photonic computing to more advanced edge/cloud AI

applications, e.g., vision, audio, language, and control tasks.

Efficient Nonlinearity. Though there exists optical nonlinearity, e.g., saturable absorbers, and electrical-optical nonlinear units [218], the current activation function is still offloaded to electrical parts. It is of practical usage to design programmable optical nonlinearity with less energy loss and E-O conversion latency, which can potentially fuse tensor computation with nonlinear activations for higher efficiency. Post-CMOS electronics can be potentially used to achieve nonlinear activation functions. Built-in nonlinear transmission in photonic modulators, e.g., Lorentzian curve for MRRs and sinusoidal curves for MZIs can also be explored as potential nonlinearity.

CMOS+Photonics+X: Heterogeneous Integration with New Device and New Material. Device-level innovation is essential in the ONN design stack, for example, phase shifters with high tuning efficiency, phase change materials with short programming latency and high lifetime, on-chip lasers with high energy efficiency, and high-sensitivity photodetector to support high readout resolution, transparent monitors to detect intermediate circuit states for on-chip training and calibration, etc. Besides photonic devices, electronic-photonic heterogeneous hardware design is also promising. We can leverage the high speed advantages of photonics for both computing and interconnects. We can use more efficient electrical memory solutions, e.g., RRAM and magnetic RAM (MRAM), as a substitution for traditional SRAM/DRAM.

System Innovation: Compute + Sensing/Memory/Interconnect. System designs open a new dimension to optimize the performance of photonic

accelerators, including optimization on devices, tensor core type and size, on-chip interconnect topology, ADCs/DACs, etc [131, 184]. Combining computing with sensing, memory, and interconnect can resolve the data movement bottleneck and achieve system-level performance breakthroughs.

Co-Design and Electronic-Photonic Design Automation. EPDA is another critical direction to explore, given the design complexity of heterogeneous electronic-photonic hardware platforms. Fast and scalable device/circuit simulation, as well as automated photonic circuit layout generation, are crucial for the efficient development of photonic computing systems. AI-assisted intelligent co-design and EPDA stack can enable unprecedented design quality, productivity, and efficiency, e.g., automated architecture search, automatic circuit topology design, layout generation, AI-assisted simulation and performance evaluation [76, 118, 69].

Reliability and Security. The computing fidelity and reliability, e.g., resolution, noise tolerance, hardware security, and adversarial robustness, are all critical to the practical deployment of photonic computing systems in real-world applications. Less sensitive device designs, fault-tolerant architecture innovations, secure communication protocols, and customized model training recipes are promising directions to boost the reliability and security of photonic computing platforms.

Appendices

.1 Appendices for Introduction

.1.1 ONN Principles

.1.1.1 Mach-Zehnder Interferometers (MZIs)

A basic coherent optical component used in this work is an MZI. One of the most general MZI structures is shown in Figure 1, consisting of two 50-by-50 optical directional couplers and four phase shifters θ_T, θ_L, ω_P, and ω_W. An MZI can achieve arbitrary 2×2 unitary matrices $SU(2)$. The physical

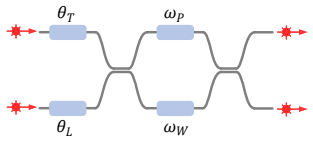


Figure 1: 2-by-2 MZI with top (T), left (L), upper (P), and lower (W) phase shifters.

transfer matrix $R(\theta_g, \Delta\theta, \Delta\omega)$ of an MZI shown in Fig. 1 is,

$$
\begin{aligned}
SU(2) = R(\theta_g, \Delta\theta, \Delta\omega) &= \begin{pmatrix} t & kj \\ kj & t \end{pmatrix} \begin{pmatrix} e^{j\omega_P} & 0 \\ 0 & e^{j\omega_W} \end{pmatrix} \begin{pmatrix} t & kj \\ kj & t \end{pmatrix} \begin{pmatrix} e^{j\theta_T} & 0 \\ 0 & e^{j\theta_L} \end{pmatrix} \\
&= e^{j\theta_g} \begin{pmatrix} \sin\frac{\Delta\omega}{2} & \cos\frac{\Delta\omega}{2} \\ \cos\frac{\Delta\omega}{2} & -\sin\frac{\Delta\omega}{2} \end{pmatrix} \begin{pmatrix} e^{j\frac{\Delta\theta}{2}} & 0 \\ 0 & e^{-j\frac{\Delta\theta}{2}} \end{pmatrix}, \\
\theta_g &= \bar{\theta} + \bar{\omega} + \frac{\pi}{2},\ \bar{\theta} = \frac{\theta_T + \theta_L}{2},\ \bar{\omega} = \frac{\omega_P + \omega_W}{2}, \\
\Delta\theta &= \theta_T - \theta_L,\ \Delta\omega = \omega_P - \omega_W,\ t = k = \frac{\sqrt{2}}{2}.
\end{aligned}
\tag{1}
$$

where the global phase θ_g is determined by the common mode $\bar{\theta}$ and $\bar{\omega}$, and the light splitting is determined by the differential mode $\Delta\theta$ and $\Delta\omega$. To achieve the 2-D planar rotator $R(2)$ in the real space parametrized by ϕ, we

let $\theta_T = \pi/2$, $\theta_L = 3\pi/2$, $\bar{\omega} = \pi$. To convert the simplified transfer matrix $M(\Delta\omega)$ to the planar rotator, we set $\Delta\omega = \pi - 2\phi$ as follows,

$$\begin{aligned} R(2) =& e^{j\frac{3\pi}{2}} \begin{pmatrix} \sin\frac{\Delta\omega}{2} & \cos\frac{\Delta\omega}{2} \\ \cos\frac{\Delta\omega}{2} & -\sin\frac{\Delta\omega}{2} \end{pmatrix} \begin{pmatrix} j & 0 \\ 0 & -j \end{pmatrix} \\ =& \begin{pmatrix} \sin\left(\frac{\pi-2\phi}{2}\right) & -\cos\left(\frac{\pi-2\phi}{2}\right) \\ \cos\left(\frac{\pi-2\phi}{2}\right) & \sin\left(\frac{\pi-2\phi}{2}\right) \end{pmatrix} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}. \end{aligned} \tag{2}$$

.1.1.2 MZI-based Photonic Tensor Core Architecture

By cascading $N(N-1)/2$ MZIs into a triangular mesh (Recks-style) or rectangular mesh (Clements-style), we can construct arbitrary $N \times N$ unitary $U(N)$.

As a simple example, we show the principle of Recks-style MZI array for a simple demonstration. A similar decomposition can be derived for the Clements style. It decomposes an $M \times N$ weight matrix using SVD, i.e., $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$. The diagonal matrix $\boldsymbol{\Sigma}$ can be simply implemented by on-chip attenuators, e.g., single-port MZIs, to perform signal scaling. The unitary matrices $\boldsymbol{U}$ and $\boldsymbol{V}^*$ can be realized by a cascaded MZI triangular array [161]. The unitary group parametrization is given by,

$$\boldsymbol{U}(N) = \boldsymbol{D} \prod_{i=N}^{2} \prod_{j=1}^{i-1} \boldsymbol{R}_{ij}(\phi_{ij}), \tag{3}$$

where $\boldsymbol{D}$ is a diagonal matrix with ± 1 on its diagonal entries, and the 2-dimensional planar rotator $\boldsymbol{R}_{ij}(\phi_{ij})$ is an n-dimensional identity matrix where entries on (i,i), (i,j), (j,i), (j,i) are $\cos\phi_{ij}$, -$\sin\phi_{ij}$, $\sin\phi_{ij}$, $\cos\phi_{ij}$, respectively. Each rotator $\boldsymbol{R}_{ij}$ can be implemented by a 2×2 MZI that produces unitary interference of input light signals with a rotation angle ϕ as we show before.

.2 Appendices for L^{2}ight

.2.1 Optical Circuit Non-ideality

Rotation Quantization. Given the control resolution limits, we can only achieve discretized MZI rotation phase configurations. We assume the phases ϕ is uniformly quantized into b-bit within [0,2π],

$$\mathcal{Q}(\phi) = \texttt{Round}\Big(\frac{\phi \bmod 2\pi}{2\pi/(2^b-1)}\Big)\frac{2\pi}{2^b-1}. \tag{4}$$

We assume 8-bit quantization for phases of $\boldsymbol{U}$ and $\boldsymbol{V}^*$. For $\boldsymbol{\Sigma}$ matrices, we assume larger bitwidths can be affordable and practical.

Phase Shifter Variation. Due to manufacturing error and thermal noises, the phase shift ϕ caused by a phase shifter is proportional to the device-related parameter, $\phi \propto \gamma$. Assume the real coefficient drifts from the theoretical value γ by $\Delta\gamma$, the real phase shift will become $\tilde{\phi} = \frac{\gamma+\Delta\gamma}{\gamma}\phi$. We assume $\Delta\gamma \sim \mathcal{N}(0, 0.002^2)$. We denote this multiplicative error for all phase shifters as a diagonal $\boldsymbol{\Gamma}$ matrix, such that the non-ideal phase shifts become $\boldsymbol{\Phi}^v = \boldsymbol{\Gamma}\boldsymbol{\Phi}$.

MZI Crosstalk. Due to signal crosstalk, adjacent MZIs will have mutual coupling effects, such that the part of the phase shift ϕ for the i-th MZI will partially contribute to its neighboring MZI ϕ_j with a factor of $\omega_{i,j}$. This crosstalk effect can be simply modeled as coupling matrix $\boldsymbol{\Omega}$,

$$
\begin{pmatrix} \phi_0^c \\ \phi_1^c \\ \vdots \\ \phi_{N-1}^c \end{pmatrix} = \begin{pmatrix} \omega_{0,0} & \omega_{0,1} & \cdots & \omega_{0,N-1} \\ \omega_{1,0} & \omega_{1,1} & \cdots & \omega_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{N-1,0} & \omega_{N-1,1} & \cdots & \omega_{N-1,N-1} \end{pmatrix} \begin{pmatrix} \phi_0^v \\ \phi_1^v \\ \vdots \\ \phi_{N-1}^v \end{pmatrix}
$$
$$
\begin{aligned} \text{s.t. } & \omega_{i,j} = 1, \quad \forall\, i = j \\ & \omega_{i,j} = 0, \quad \forall\, i \neq j \text{ and } \phi_j \in \mathcal{P} \\ & 0 \leq \omega_{i,j} < 1, \quad \forall\, i \neq j \text{ and } \phi_j \in \mathcal{A}. \end{aligned} \tag{5}
$$

The diagonal factor $\omega_{i,j}, i = j$ is the self-coupling coefficient. $\omega_{i,j}, i \neq j$ is the mutual coupling coefficient [140, 72, 65]. We assume the self-coupling coefficient to be 1, and the mutual coupling coefficient is 0.005 for adjacent MZIs.

.2.2 Intractable Gradients for MZI Rotations

To optimize the MZI meshes, a straightforward idea is to use first-order methods to optimize all rotations phases $\mathbf{\Phi}^U$, $\mathbf{\Phi}^V$, and $\mathbf{\Phi}^\Sigma$. The analytical gradients for phases in unitary matrices are shown as,

$$
\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{R}_{ij}} &= \big(\boldsymbol{D}\boldsymbol{R}_{n1}\boldsymbol{R}_{n2}\boldsymbol{R}_{n3}\big)^T \nabla_y \mathcal{L}\ x^T \big(\cdots \boldsymbol{R}_{32}\boldsymbol{R}_{21}\boldsymbol{\Sigma}\boldsymbol{V}^*\big)^T \\ \frac{\partial \mathcal{L}}{\partial \phi_{ij}} &= \mathrm{Tr}\bigg(\Big(\frac{\partial \mathcal{L}}{\partial \boldsymbol{R}_{ij}} \odot \frac{\partial \boldsymbol{R}_{ij}}{\partial \phi_{ij}}\Big)(e_i + e_j)(e_i + e_j)^T\bigg). \end{aligned} \tag{6}
$$

Therefore, it is prohibitively expensive to derive the analytical phase gradients, which is one of the key motivations for our subspace optimization method.

.2.3 Detailed Description of the Proposed Parallel Mapping Algorithm

We give a detailed description of our parallel mapping algorithm. Zeroth-order coordinate descent (ZCD) is used as an example. In line 4, we first derive and implement the optimal theoretical singular values and initialize $\boldsymbol{\Phi}^U$ and $\boldsymbol{\Phi}^V$ using the decomposed values. In lines 8-13, we use ZCD to alternately optimize phases in $\boldsymbol{U}$ and $\boldsymbol{V}^*$ under all non-ideal effects till convergence. The step size is strictly bounded by the smallest phase control resolution. Exponential decay is used to quickly reduce the learning rate to avoid divergence. Note that cosine-annealing will not work since the ZO descent will rapidly converge, given its greedy search nature. Then at the end, due to the suboptimality in ZCD, we will perform OSP to find the current optimal singular values that minimize the mapping error given the trained $\boldsymbol{U}^T$ and $\boldsymbol{V}^{*,T}$.

Algorithm 5 Parallel Mapping with ZCD and OSP

Input: Mapping loss $\mathcal{L}^M$, mapping target $\boldsymbol{W}$, total iterations T, inner ZCD iterations S, step size decay factor β, ZCD step size upper bound $\delta\phi_u = \frac{2\pi}{2^{\min(b_l,b)}-1}$, ZCD step size lower bound $\delta\phi_l = \frac{2\pi}{2^{\min(b_m,b)}-1}$ $\delta\phi = \delta\phi_u$

for Weight block $\boldsymbol{W}_{pq} \sim \boldsymbol{W}$ **do**

Step 1: SVD and Parametrization via Eq. (3.23)

$\boldsymbol{U}_{pq}(\boldsymbol{\Phi}^U_{pq}), \boldsymbol{\Sigma}_{pq}(\boldsymbol{\Phi}^S_{pq}), \boldsymbol{V}^*_{pq}(\boldsymbol{\Phi}^V_{pq}) = \texttt{UP}\big(\texttt{SVD}(\boldsymbol{W}_{pq})\big)$

Step 2: ZCD on $\boldsymbol{U}_{pq}, \boldsymbol{V}^*_{pq}$

  for $t \leftarrow 0 \cdots T-1$ **do**

    for $s \leftarrow 0 \cdots S-1$ **do** Randomly sample a phase $\phi \sim \{\boldsymbol{\Phi}^U_{pq}, \boldsymbol{\Phi}^V_{pq}\}$

      if $\mathcal{L}^M_{pq}(\phi^{tS+s} + \delta\phi) < \mathcal{L}^M_{pq}(\phi^{tS+s})$ **then** $\phi^{tS+s+1} \leftarrow \phi^{tS+s} + \delta\phi$

      else $\phi^{tS+s+1} \leftarrow \phi^{tS+s} - \delta\phi$ $\quad \delta\phi \leftarrow \max(\delta\phi/\beta, \delta\phi_l)$

Step 3: Optimal Projection on $\boldsymbol{\Sigma}_{pq}$ $\quad \Sigma_{pq} \leftarrow \texttt{diag}(\tilde{\boldsymbol{I}}^* \boldsymbol{U}^*_{pq} \boldsymbol{W}_{pq} \boldsymbol{V}_{pq} \tilde{\boldsymbol{I}})$

Return: Converged phases $\boldsymbol{\Phi}^M$

.2.4 Prove of Unbiased Gradient Approximation with Feedback and Feature Sampling

Claim 2. *Considering the l-th layer with input $x \in \mathbb{R}^N$ and pre-activation $y \in \mathbb{R}^M$, we denote the blocking weight matrix as $\boldsymbol{W} = \{\boldsymbol{W}_{pq}\}_{p,q=1,1}^{P=\frac{M}{k},Q=\frac{N}{k}}$ and nonlinear activation as σ. During backward, we randomly sample the feedback matrix $\boldsymbol{W}^T \in \mathbb{R}^{N\times M}$ with a structured sparse mask $\mathcal{P}_{\boldsymbol{W}} = c_W(\mathcal{S}_W \otimes \mathbf{1})$. A similar sampling matrix $\mathcal{P}_x$ is applied to input features. The estimated gradients are unbiased, i.e., $\mathbb{E}[(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}})_{\mathcal{S}}] = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}}$.*

Proof. Given $\mathbb{E}[\mathcal{P}] = \mathbf{1}$, we have

$$\begin{aligned} \mathbb{E}[(\boldsymbol{W}_l^T)_{\mathcal{S}_{\boldsymbol{W}_l}}] =& \mathbb{E}[\boldsymbol{W}_l^T \odot \mathcal{P}_{\boldsymbol{W}_l}] = \boldsymbol{W}_l^T \\ \mathbb{E}[(\boldsymbol{x}_l^T)_{\mathcal{S}_{\boldsymbol{x}_l}}] =& \mathbb{E}[\boldsymbol{x}_l^T \odot \mathcal{P}_{\boldsymbol{x}_l}] = \boldsymbol{x}_l^T. \end{aligned} \tag{7}$$

Then we can derive

$$\begin{aligned} \mathbb{E}[(\frac{\partial \mathcal{L}}{\partial y_l})_{\mathcal{S}_{\boldsymbol{W}_l}}] =& \mathbb{E}\Big[\sigma_l' \prod_{i=l+1}^{L-1} ((\boldsymbol{W}_i^T)_{\mathcal{S}_{\boldsymbol{W}_l}} \odot \sigma_i')(\boldsymbol{W}_L^T)_{\mathcal{S}_{\boldsymbol{W}_l}} \frac{\partial \mathcal{L}}{\partial y_L}\Big] = \frac{\partial \mathcal{L}}{\partial y_l} \\ \mathbb{E}\big[(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_l})_{\mathcal{S}}\big] =& \mathbb{E}\Big[\boldsymbol{U}^* (\frac{\partial \mathcal{L}}{\partial y_l})_{\mathcal{S}_{\boldsymbol{W}_l}} (x_l^T)_{\mathcal{S}_{\boldsymbol{x}_l}} \boldsymbol{V}\Big] = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_l}. \end{aligned} \tag{8}$$

□

.2.5 Training Details

We implement ONN simulation, all models, and training logic in PyTorch 1.8.1. All experiments are conducted on a machine with an Intel Core i7-9700 CPU and an NVIDIA Quadro RTX 6000 GPU. For identity calibration, we set the epoch to 400 with an initial learning rate of 0.1, a decay rate of 0.99, and a phase resolution of 8 bit. For parallel mapping, we set the epoch

to 300 with an initial learning rate of 0.1, a decay rate of 0.99, and a phase resolution of 8 bit. For subspace learning, we adopt AdamW as the optimizer with a learning rate of 0.002 and a weight decay rate of 0.01 for subspace learning from scratch. Epochs are set to 100 for MNIST, FashionMNIST training, 200 for CIFAR-10/100, and TinyImageNet. For subspace learning after mapping, we reduce the epoch to 20 and the learning rate to 0.0002. We use cosine-annealing as the learning rate scheduler. When compared with prior on-chip learning protocols, we adopt the recommended settings for `FLOPS` and `MixedTrn` in [74, 65]. For `FLOPS`, the total epochs are set to 50, the initial learning rate is 2, and the gradient samples are set to 5. For `MixedTrn`, we train for 20 epochs, the mixed-training sparsity is set to 0.4, the parameter sparsity is set to 0.1, and the initial learning rate is set to 0.02. When compared with prior sampling methods, we apply uniform spatial sampling with expectation-maintained normalization for `RAD` [153]. For `SWAT-U` [157], we apply uniform spatial feature sampling without normalization and uniform weight matrix sampling with expectation-maintained normalization. Since we only perform efficient training, we turn off any sampling in inference.

.2.6 MZI Array Scaling

A single MZI array has a limited size due to its high area cost, e.g., up to 32 or 64. However, this is not an issue for our framework. Multi-core systems with small subarrays are trends for analog computing, which is the design concept of our accelerator in Figure 3.16. Multiple PTCs are

Table 1: Relative matrix error with different MZI array sizes.

Blk size	8	9	12	16	24	32
Rel. Err.	0.025	0.032	0.043	0.061	0.094	0.126
std.	2e-4	3e-4	3e-4	5e-4	9e-4	1e-3

interconnected to support a large tensor computation in parallel. Therefore, our system's performance will not be limited by the scale of a single PTC. Actually, partitioning a large tensor operation into small chunks is widely adopted and recently considered as a better solution than large array sizes due to noise robustness consideration.

We adopt 9×9 blocks based on the following considerations.

Hardware practicality. The largest commercial demonstration of optical neural chips is 32×32 so far. 9×9 is a practical, robust, and efficient setting according to recent experimental demonstrations.

Robustness. Larger MZI arrays will cause severe phase error accumulation effects. Cascaded phase error will cause non-trivial fidelity and robustness issues as the block size increases. 9×9 is generally a robust design configuration when cascaded noises are still tolerable. Here we show a table of noise-induced errors (relative matrix distance) with various block sizes on a 256×256 weight matrix. Std. is calculated based on 20 runs. Phase shifter gamma noise std=0.002, crosstalk factor=0.005, quantization bitwdith=8-bit. We observe large array sizes are noise-sensitive in general.

ZOO Convergence. IC and PM are zeroth-order optimization techniques. Each block indicates an optimization instance. Larger block size will have

Table 2: IC optimality with different array sizes.

Blk size	8	9	12	16	24	32
$(MSE^U + MSE^V)/2$	0.0135	0.013	0.03	0.039	0.04	0.045

Table 3: Subspace learning accuracy with different block sizes.

Blk size	8	9	12	16	24	32
Accuracy	84.26	84.45	83.36	81.27	80.68	78.40

negative impacts on the optimization convergence and solution optimality, which is the intrinsic limitation of most zeroth-order optimizers. In the IC procedure, for relatively large block sizes, our ZO optimizers, unfortunately, will have solution quality degradation due to the curse of dimensionality and efficiency degradation due to low parallelism. Here we show how solution quality in identity calibration changes with various block sizes. 9×9 block is a good selection with high solution quality.

Parameter Space. Subspace learning only optimizes the singular values while U and V are fixed. For an $N \times N$ weight matrix with $k \times k$ blocks, only N^2/k singular values are trainable. Increasing the block size k will decrease the parameter space. According to the experience from the field of structured/subspace neural networks, e.g., block-circulant neural nets, the block size is typically set to a number around 8. Here we add new results on `L`^{2}`ight`-SL (α_W=α_C=0.6, α_D=0.5) CIFAR-10 VGG8 with various block sizes. According to our experiments below, 16×16 blocks already show inadequate trainability due to overly small parameter space, leading to a clear accuracy drop. In conclusion, we recommend using multiple interconnected 9×9 PTCs

for parallel computing, since this choice of 9×9 block balances both systematic performance, hardware complexity, robustness, and on-chip trainability.

.2.7 Hardware Cost Evaluation

.2.7.1 PTC Energy Estimation

For simplicity, we count the number of PTC calls as the indicator of the total energy estimation of the PTC cluster. For example, we focus on a 2-D convolutional layer with kernel shape of $C_{out} \times C_{in} \times K \times K$, input feature size $B \times C_{in} \times H \times W$ output feature size of $B \times C_{out} \times H' \times W'$. We partition the unfolded weight matrix into $P \times Q$ blocks with size of $k \times k$ and assign each to a PTC. We have $P = \lceil \frac{C_{out}}{k} \rceil$ and $Q = \lceil \frac{C_{in} \times K^2}{k} \rceil$. Each PTC can utilize k wavelengths to achieve parallel processing. Now we give detailed computation of energy breakdown per optimization iteration.

$$\begin{aligned} \textbf{Forward Energy} &= C_{out} C_{in} K^2 B H' W' \\ \textbf{Backward Weight Energy} &= 2\mathrm{Tr}(\mathcal{S}_C^T \mathcal{S}_C) BPQ \\ \textbf{Backward Input Energy} &= \mathrm{Tr}(\mathcal{S}_W^T \mathcal{S}_W) BHW. \end{aligned} \tag{9}$$

Note that in backward weight energy, we double the PTC call since the *in-situ* subspace gradient acquisition requires 2 PTC calls.

.2.7.2 Total Time Step Estimation

We assign k electrical adders for each PTC to implement sequential cross-PTC reduction and parallel local accumulation. Each PTC call counts as one step, each partial product/gradient accumulation stage counts as one step, and the Hadamard multiplication in gradient computation also counts

as one step. Given this assumption, we derive the time step as,

$$\begin{aligned}
\textbf{Forward Step} &= (Q-1)_+ BH'W' + \lceil \frac{BH'W'}{k} \rceil \\
\textbf{Backward Weight Step} &= 4\mathrm{Tr}(\mathcal{S}_C^T \mathcal{S}_C) B \\
\textbf{Backward Input Step} &= \begin{cases} \lceil \frac{C_{in}}{P} \rceil \lceil \log_2 2k \rceil \lceil \frac{1}{2} \max\limits_q \left((\sum \mathcal{S}_W(q,:) - 1)_+ \right) \rceil BHW, & K > 1,\ \text{stride} < K \\ \max\limits_q \left((\sum \mathcal{S}_W(q,:) - 1)_+ \right) BH'W', & K = 1 \end{cases}
\end{aligned} \tag{10}$$

.2.7.3 WDM Dispersion Discussion

Theoretically, coherent photonic circuits will have slightly different phase responses to different working wavelengths. However, we claim that this frequency-specific phase shift has minimum impacts on our learning procedure.

Negligible Dispersion. Our PTC core is intentionally designed to have a small-scale, i.e., 9×9. Hence we require 9 wavelengths in our framework. This avoids too many wavelengths being used. Therefore, the spectrum range will be relatively small. Conservatively we assume 8 nm between the furthest two wavelengths. Based on the phase response equation, $\Delta\phi(\lambda) = 2\pi n_{eff}(\lambda)L/\lambda$, this leads to a maximum 1-2% phase difference for the furthest two wavelengths. On a small MZI array, this phase difference will only cause negligible transfer function drift. We simulate this effect when the weight block size is set to 9×9 and inject 1-2% dispersion-induced MZI phase response drift; the transfer matrix has 0.5% relative error and 0.5% mean square error. Compared with the gradient approximation error caused by our three-level sparse sampling, phase variation, and thermal crosstalk, shown in Fig. 8, this slight

drift caused by WDM dispersion is negligible.

High Non-ideality Tolerance. Our experiments show that first-order subspace learning is very robust to all these gradient approximation errors. With all the above non-ideality, the approximated gradient directions are still well-aligned with the true gradients. The on-chip learning procedure works as expected even when WDM dispersion effects are considered. This effect can be considered in-situ when using WDM on MZI array training, therefore, the model can tolerate this non-ideal effect without inference accuracy degradation.

Dispersion-free Devices. In the literature, there are WDM dispersion-free MZI devices being proposed [43]. Within the 45nm range, the coefficient of phase shifters can be maintained. Thus, the phase response to 9 different wavelengths can be compensated to almost the same response. This further shows that WDM dispersion is not a major concern for our assumed ONN architecture and proposed training flow.

.3 Appendices for `NeurOLight`

.3.1 Optical Field Simulation

Analyzing how light field propagates through those components is critical to device optimization and photonic integrated circuit design. Given a linear isotropic optical component, we will shine time-harmonic continuous-wave light on its input ports and analyze the steady-state electromagnetic field distributions $\mathbf{E} = \hat{\mathbf{x}}\mathbf{E}_x + \hat{\mathbf{y}}\mathbf{E}_y + \hat{\mathbf{z}}\mathbf{E}_z$ and $\mathbf{H} = \hat{\mathbf{x}}\mathbf{H}_x + \hat{\mathbf{y}}\mathbf{H}_y + \hat{\mathbf{z}}\mathbf{H}_z$ in it,

each of which includes horizontal (x), vertical (y), and longitudinal (z) components. The light field follows the Maxwell PDE under certain absorptive boundary conditions [100],

$$\nabla \times \mathbf{E}(\boldsymbol{r},t) = \mu_0 \frac{\partial \mathbf{H}(\boldsymbol{r},t)}{\partial t} + \mathbf{J}_e(\boldsymbol{r},t),\ \nabla \times \mathbf{H}(\boldsymbol{r},t) = -\epsilon_0 \boldsymbol{\epsilon}_r(\boldsymbol{r}) \frac{\partial \mathbf{E}(\boldsymbol{r},t)}{\partial t} + \mathbf{J}_e(\boldsymbol{r},t), \tag{11}$$

where $\nabla\times$ is the curl operator of a vector function, μ_0 is the vacuum magnetic permeability, ϵ_0 and $\boldsymbol{\epsilon}_r$ are the vacuum and relative electric permittivity, $\mathbf{J}_m$ and $\mathbf{J}_e$ are the magnetic and electric current sources. Since the input light is time-harmonic at a vacuum angular frequency ω, the time-domain PDE can be transformed to the frequency domain for the steady state as follows,

$$\nabla \times \mathbf{E}(\boldsymbol{r}) = j\omega\mu_0 \mathbf{H}(\boldsymbol{r}) + \mathbf{J}_m(\boldsymbol{r}),\ \nabla \times \mathbf{H}(\boldsymbol{r}) = -j\omega\epsilon_0 \boldsymbol{\epsilon}_r(\boldsymbol{r}) \mathbf{E}(\boldsymbol{r}) + \mathbf{J}_e(\boldsymbol{r}). \tag{12}$$

A simple variable substitution gives us the *curl-of-curl* Maxwell PDE,

$$\big((\mu_0^{-1}\nabla \times \nabla\times) - \omega^2 \epsilon_0 \boldsymbol{\epsilon}_r(\boldsymbol{r})\big)\mathbf{E}(\boldsymbol{r}) = j\omega \mathbf{J}_e(\boldsymbol{r}),\ \big(\nabla \times (\boldsymbol{\epsilon}_r^{-1}(\boldsymbol{r})\nabla\times) - \omega^2 \mu_0 \epsilon_0\big)\mathbf{H}(\boldsymbol{r}) = j\omega \mathbf{J}_m(\boldsymbol{r}). \tag{13}$$

To restrict a unique solution without boundary reflection, complicated boundary conditions will be inserted [100]. An artificial material, i.e., coordinate-stretched perfectly matched layer (SC-PML), will be padded around the solving domain. Such PML materials have large imaginary parts in the permittivities to introduce strong energy absorption and change the derivative operator to $\nabla = (\frac{1}{s_x(x)}\frac{\partial}{\partial x}, \frac{1}{s_y(y)}\frac{\partial}{\partial y}, \frac{1}{s_z(z)}\frac{\partial}{\partial z})$, where s is a location-determined complex value. Solving the above PDEs will give the steady-state frequency-domain complex magnitude of the optical fields.

.3.2 Dataset Generation

We generate our customized MMI device simulation dataset using an open-source FDFD simulator angler [100]. The tunable MMI dataset has 5.5 K *single-source* training data, 614 validation data, and 1.5 K multi-source test data. The etched MMI dataset has 12.4 K *single-source* training data, 1.4 K validation data, and 1.5 K *multi-source* test data. We summarize how we generate random devices in Table 4. We randomly sample the physical dimension of the MMI, input/output waveguide width, the width of the perfectly matched layer (PML), device border width away from PML, controlling pad sizes, input light source frequencies, etched cavity sizes and ratio (determines the number of cavities in the MMIs), and permittivities in the controlling region.

Table 4: Summary of device design variable's sampling range, distribution, and unit.

Variables	Value/Distribution		Unit
	$\lvert\mathbf{J}\rvert \times \lvert\mathbf{J}\rvert$ Tunable MMI	$\lvert\mathbf{J}\rvert \times \lvert\mathbf{J}\rvert$ Etched MMI	
Length	$\mathcal{U}(20, 30)$	$\mathcal{U}(20, 30)$	μm
Width	$\mathcal{U}(5.5, 7)$	$\mathcal{U}(5.5, 7)$	μm
Port Length	3	3	μm
Port Width	$\mathcal{U}(0.8, 1.1)$	$\mathcal{U}(0.8, 1.1)$	μm
Border Width	0.25	0.25	μm
PML Width	1.5	1.5	μm
Pad Length	$\mathcal{U}(0.7, 0.9)\times$Length	$\mathcal{U}(0.7, 0.9)\times$Length	μm
Pad Width	$\mathcal{U}(0.4, 0.65)\times$Width/$\lvert\mathbf{J}\rvert$	$\mathcal{U}(0.4, 0.65)\times$Width/$\lvert\mathbf{J}\rvert$	μm
Wavelengths λ	$\mathcal{U}(1.53, 1.565)$	$\mathcal{U}(1.53, 1.565)$	μm
Cavity Ratio	-	$\mathcal{U}(0.05, 0.1)$	-
Cavity Size	-	0.027 Length $\times$ 0.114 Width	μm^2
Relative Permittivity $\boldsymbol{\epsilon}_r$	$\mathcal{U}(11.9, 12.3)$	$\{2.07, 12.11\}$	-

.3.3 Training Settings

We implement all models and training logic in PyTorch 1.10.2. All experiments are conducted on a machine with Intel Core i7-9700 CPUs and an NVIDIA Quadro RTX 6000 GPU. For training from scratch, we set the number of epochs to 200 with an initial learning rate of 0.002, cosine learning rate decay, and a mini-batch size of 12. For the tunable MMI dataset, we split all 7,680 examples into 72% training data, 8% validation data, and 20% test data. For the etched MMI dataset, we split all 15,360 examples into 81% training data, 9% validation data, and 10% test data. For device adaptation, we first perform linear probing for 20 epochs with an initial learning rate of 0.002 and cosine learning rate decay; then we perform finetuning for 30 epochs with an initial learning rate of 0.0002 and a cosine learning rate decay. We apply stochastic network depth with a linear scaling strategy and a maximum drop rate of 0.1.

.3.4 Model Architectures

UNet. We construct a 4-level convolutional UNet with a base channel number of 34. The total parameter count is 3.47 M.

FNO-2d. For Fourier neural operator (FNO), we use 5 2-D FNO layers with a channel number of 32. The Fourier modes are set to (#$\texttt{Mode}_z$=32, #$\texttt{Mode}_x$=10). The final projection head is CONV1×1(256)-GELU-CONV1×1(2). The total parameter count is 3.29 M.

F-FNO. For factorized Fourier neural operator (F-FNO), we use 12 F-

FNO layers with a channel number of 48. The Fourier modes are set to (#Mode$_z$=70, #Mode$_x$=40). The final projection head is CONV1×1(256)-GELU-CONV1×1(2). The total parameter count is 3.16 M.

NeurOLight. For our proposed NeurOLight, we use 12 F-FNO layers for tunable MMIs and 16 layers for etched MMIs with a base channel number C=64. The convolution stem is BSConv3×3(32)-BN-ReLU-BSConv3×3(64)-BN-ReLU, where BSConv is blueprint convolution [79]. The Fourier modes are set to (#Mode$_z$=70, #Mode$_x$=40). The channel expansion ratio in the FFN is set to s=2. The final projection head is CONV1×1(256)-GELU-CONV1×1(2). The total parameter count is 1.58 M.

Bibliography

[1] Adc (analog-to-digital converters) – alphacore., https://www.alphacoreinc.com/adc-analog-to-digital-converters/.

[2] Advanced micro foundry, http://www.advmf.com/services/.

[3] Menachem Adelman and Mark Silberstein. Faster neural network training with approximate tensor operations. *arXiv preprint arXiv:1805.08079*, 2018.

[4] Hengameh Bagherian, Scott A. Skirlo, Yichen Shen, et al. On-chip optical convolutional neural networks. *ArXiv*, abs/1808.03303, 2018.

[5] Krishnakumar Balasubramanian and Saeed Ghadimi. Zeroth-order nonconvex stochastic optimization: Handling constraints. *Arxiv*, 2019.

[6] V. Bangari, B. A. Marquez, H. Miller, A. N. Tait, M. A. Nahmias, T. F. de Lima, H. Peng, P. R. Prucnal, and B. J. Shastri. Digital electronics and analog photonics for convolutional neural networks (DEAP-CNNs). *IEEE JSTQE*, 2020.

[7] Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. In *Proc. NeurIPS*, 2018.

[8] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *Proc. ICML*, volume 80, pages 560–569, 2018.

[9] Liane Bernstein, Alexander Sludds, Ryan Hamerly, Vivienne Sze, Joel Emer, and Dirk Englund. Freely scalable and reconfigurable optical hardware for deep learning. *ArXiv*, abs/2006.13926, 2020.

[10] Adel Bibi, El Houcine Bergou, Ozan Sener, Bernard Ghanem, and Peter Richtarik. A stochastic derivative-free optimization method with importance sampling: Theory and learning to control. In *Proc. AAAI*, 2020.

[11] D. Brunner, M. C. Soriano, C. R. Mirasso, et al. Parallel photonic information processing at gigabyte per second data rates using transient states. *Nature Communications*, 2013.

[12] J. Bueno, S. Maktoobi, L. Froehly, et al. Reinforcement learning in a large-scale photonic recurrent neural network. *Optica*, 2018.

[13] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for All: Train One Network and Specialize it for Efficient Deployment. In *Proc. ICLR*, 2020.

[14] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End object detection with transformers. In *Proc. ECCV*, 2020.

[15] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[16] K. Chellapilla, Sidd Puri, and P. Simard. High performance convolutional neural networks for document processing. In *Proc. ICFHR*, 2006.

[17] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. CrossViT: Cross-attention multi-scale vision transformer for image classification. In *Proc. ICCV*, 2021.

[18] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *Proc. NeurIPS*, 2020.

[19] Mingkun Chen, Robert Lupoiu, Chenkai Mao, Der-Han Huang, Jiaqi Jiang, Philippe Lalanne, and Jonathan Fan. Physics-augmented deep learning for high-speed electromagnetic simulation and optimization. *Nature*, 2021.

[20] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: Zeroth Order Optimization Based Black-Box Attacks to Deep Neural Networks without Training Substitute Models. In *Proc. AISec*, page 15–26, 2017.

[21] Xiangyi Chen, Sijia Liu, Kaidi Xu, Xingguo Li, Xue Lin, Mingyi Hong, and David Cox. Zo-AdaMM: Zeroth-order adaptive momentum method for black-box optimization. In *Proc. NeurIPS*, pages 7204–7215, 2019.

[22] Y. Chen, J. Emer, and V. Sze. Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks. In *Proc. ISCA*, pages 367–379, 2016.

[23] Y. Chen, T. Krishna, J. Emer, and V. Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. In *Proc. ISSCC*, pages 262–263, 2016.

[24] Y. Chen, T. Krishna, J. S. Emer, and V. Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal Solid-State Circuits*, 52(1):127–138, 2017.

[25] Yinpeng Chen, Xiyang Dai, Mengchen Liu, et al. Dynamic convolution: Attention over convolution kernels. In *Proc. CVPR*, 2020.

[26] Q. Cheng, J. Kwon, M. Glick, M. Bahadori, L. P. Carloni, and K. Bergman. Silicon Photonics Codesign for Deep Learning. *Proceedings of the IEEE*, 2020.

[27] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proc. CVPR*, pages 1800–1807, 2017.

[28] M.E.H. Chowdhury, A. Khandakar T. Rahman, R. Mazhar, M.A. Kadir, Z.B. Mahbub, K.R. Islam, M.S. Khan, A. Iqbal, N. Al-Emadi, M.B.I.

Reaz, and M. T. Islam. Can AI help in screening Viral and COVID-19 pneumonia? *IEEE ACCESS*, 8:132665 – 132676, 2020.

[29] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the Design of Spatial Attention in Vision Transformers. In *Proc. NeurIPS*, 2021.

[30] William R. Clements, Peter C. Humphreys, Benjamin J. Metcalf, et al. Optimal Design for Universal Multiport Interferometers. *Optica*, 2018.

[31] Kieran Cooney and Frank H. Peters. Multimode Interference Couplers with Tunable Power Splitting Ratios. *Optics Express*, 2016.

[32] Rodrigo Santa Cruz, Basura Fernando, Anoop Cherian, and Stephen Gould. DeepPermNet: Visual Permutation Learning. In *Proc. CVPR*, 2017.

[33] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR Workshop*, 2020.

[34] Tri Dao, Albert Gu, Matthew Eichhorn, Atri Rudra, and Christopher Ré. Learning fast algorithms for linear transforms using butterfly factorizations. In *Proc. ICML*, 2019.

[35] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, pages 248–255, 2009.

[36] Lei Deng, Peng Jiao, Jing Pei, Zhenzhi Wu, and Guoqi Li. GXNOR-Net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework. *Neural Networks*, 2018.

[37] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 2020.

[38] Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation . In *Proc. NIPS*, 2014.

[39] D.H. Deterding. Speaker normalisation for automatic speech recognition. PhD thesis, University of Cambridge, 1989.

[40] Caiwen Ding, Siyu Liao, Yanzhi Wang, Zhe Li, Ning Liu, et al. CirCNN: Accelerating and Compressing Deep Neural Networks Using Block-Circulant Weight Matrices. In *Proc. MICRO*, pages 395–408, 2017.

[41] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. CSWin Transformer:

A General Vision Transformer Backbone with Cross-Shaped Windows. *arXiv preprint arXiv:2107.00652*, 2021.

[42] A. Dosovitskiy, L. Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, M. Dehghani, Matthias Minderer, G. Heigold, S. Gelly, Jakob Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proc. ICLR*, 2021.

[43] Nicolas Dupuis, Benjamin G. Lee, et al. Design and Fabrication of Low-Insertion-Loss and Low Crosstalk Broadband 2x2 Mach-Zehnder Silicon Photonic Switches. *JLT*, 2015.

[44] El Houcine Bergou and Eduard Gorbunov and Peter Richtárik. Stochastic Three Points Method for Unconstrained Smooth Minimization. *SIAM Journal on Optimization*, 2020.

[45] J.L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. In *Machine Learning*, 1991.

[46] Steven K. Esser, Paul A. Merolla, John V. Arthur, et al. Convolutional networks for fast, energy-efficient neuromorphic computing. *PNAS*, 2016.

[47] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale Vision Transformers. In *Proc. ICCV*, 2021.

[48] Michael Y. S. Fang, Sasikanth Manipatruni, Casimir Wierzynski, et al. Design of optical neural networks with component imprecisions. *Optics Express*, 2019.

[49] Michael Y.-S. Fang, Sasikanth Manipatruni, Casimir Wierzynski, Amir Khosrowshahi, and Michael R. DeWeese. Design of optical neural networks with component imprecisions. *Opt. Express*, 2019.

[50] Johannes Feldmann, Nathan Youngblood, Maxim Karpov, Helge Gehring, Xuan Li, Maik Stappers, Manuel Le Gallo, Xin Fu, Anton Lukashchuk, Arslan Raja, Junqiu Liu, David Wright, Abu Sebastian, Tobias Kippenberg, Wolfram Pernice, and Harish Bhaskaran. Parallel convolutional processing using an integrated photonic tensor core. *Nature*, 2021.

[51] Chenghao Feng, Jiaqi Gu, Hanqing Zhu, Zhoufeng Ying, Zheng Zhao, David Z Pan, and Ray T Chen. Silicon photonic subspace neural chip for hardware-efficient deep learning. *arXiv preprint arXiv:2111.06705*, 2021.

[52] Chenghao Feng, Jiaqi Gu, Hanqing Zhu, Zhoufeng Ying, Zheng Zhao, David Z Pan, and Ray T Chen. A compact butterfly-style silicon photonic-electronic neural chip for hardware-efficient deep learning. *ACS Photonics*, Nov 2022.

[53] Chenghao Feng, Rongxing Tang, Jiaqi Gu, Hanqing Zhu, David Z. Pan, and Ray T. Chen. Optically-Interconnected, Hardware-Efficient, Electronic-

Photonic Neural Network using Compact Multi-Operand Photonic Devices. In *SPIE Photonics West*, January 2023.

[54] Chenghao Feng, Zhoufeng Ying, Zheng Zhao, et al. Wavelength-division-multiplexing-based electronic-photonic network for high-speed computing. In *Proc. SPIE, Smart Photonic and Optoelectronic Integrated Circuits XXII*, 2020.

[55] Chenghao Feng, Zhoufeng Ying, Zheng Zhao, Jiaqi Gu, et al. Integrated WDM-based Optical Comparator for High-speed Computing. In *Proc. CLEO*, 2020.

[56] Chenghao Feng, Zhoufeng Ying, Zheng Zhao, Jiaqi Gu, et al. Wavelength-division-multiplexing (WDM)-based integrated electronic–photonic switching network (EPSN) for high-speed data processing and transportation. *Nanophotonics*, 2020.

[57] Chenghao Feng, Zheng Zhao, Zhoufeng Ying, Jiaqi Gu, David Z. Pan, and Ray T. Chen. Compact design of On-chip Elman Optical Recurrent Neural Network. In *Proc. CLEO*, 2020.

[58] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

[59] Saeed. Ghadimi and Guanghui. Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal*

on Optimization, 2013.

[60] Saeed. Ghadimi and Guanghui. Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 2013.

[61] Eduard Gorbunov, Adel Bibi, Ozan Sener, El Houcine Bergou, and Peter Richtárik. A stochastic derivative free optimization method with momentum. In *Proc. ICLR*, 2020.

[62] Ben Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference. In *Proc. ICCV*, 2021.

[63] Ove Grandstrand. *Innovation and Intellectual Property Rights*. Oxford University Press, 2004.

[64] Stefano Grillanda, Marco Carminati, Francesco Morichetti, et al. Non-invasive monitoring and control in silicon photonics using CMOS integrated electronics. *Optica*, 2014.

[65] Jiaqi Gu, Chenghao Feng, Zheng Zhao, Zhoufeng Ying, Ray T Chen, and David Z Pan. Efficient on-chip learning for optical neural networks through power-aware sparse zeroth-order optimization. In *Proc. AAAI*, 2021.

[66] Jiaqi Gu, Chenghao Feng, Zheng Zhao, Zhoufeng Ying, Mingjie Liu, Ray T. Chen, and David Z. Pan. SqueezeLight: Towards Scalable

Optical Neural Networks with Multi-Operand Ring Resonators. In *Proc. DATE*, February 2021.

[67] Jiaqi Gu, Chenghao Feng, Hanqing Zhu, Ray T. Chen, and David Z. Pan. Light in AI: Toward Efficient Neurocomputing with Optical Neural Networks - A Tutorial. *IEEE Transactions on Circuits and Systems–II: Express Briefs (TCAS-II)*, April 2022.

[68] Jiaqi Gu, Chenghao Feng, Hanqing Zhu, Zheng Zhao, Zhoufeng Ying, Mingjie Liu, Ray T. Chen, and David Z. Pan. queezeLight: A Multi-Operand Ring-Based Optical Neural Network with Cross-Layer Scalability. *IEEE TCAD*, July 2022.

[69] Jiaqi Gu, Zhengqi Gao, Chenghao Feng, Hanqing Zhu, Ray T. Chen, Duane S Boning, and David Z. Pan. NeurOLight: A Physics-Agnostic Neural Operator Enabling Parametric Photonic Device Simulation. In *Proc. NeurIPS*, 2022.

[70] Jiaqi Gu, Zheng Zhao, Chenghao Feng, et al. Towards area-efficient optical neural networks: an FFT-based architecture. In *Proc. ASPDAC*, 2020.

[71] Jiaqi Gu, Zheng Zhao, Chenghao Feng, et al. Towards Hardware-Efficient Optical Neural Networks: Beyond FFT Architecture via Joint Learnability. *IEEE TCAD*, 2020.

[72] Jiaqi Gu, Zheng Zhao, Chenghao Feng, Wuxi Li, Ray T. Chen, and David Z. Pan. FLOPS: Efficient On-Chip Learning for Optical Neural Networks Through Stochastic Zeroth-Order Optimization. In *Proc. DAC*, 2020.

[73] Jiaqi Gu, Zheng Zhao, Chenghao Feng, Zhoufeng Ying, Ray T. Chen, and David Z. Pan. O2NN: Optical Neural Networks with Differential Detection-Enabled Optical Operands. In *Proc. DATE*, February 2021.

[74] Jiaqi Gu, Zheng Zhao, Chenghao Feng, Hanqing Zhu, Ray T. Chen, and David Z. Pan. ROQ: A noise-aware quantization scheme towards robust optical neural networks with low-bit controls. In *Proc. DATE*, 2020.

[75] Jiaqi Gu, Hanqing Zhu, Chenghao Feng, Zixuan Jiang, Ray T. Chen, and David Z. Pan. L2ight: Enabling On-Chip Learning for Optical Neural Networks via Efficient in-situ Subspace Optimization. In *Proc. NeurIPS*, 2021.

[76] Jiaqi Gu, Hanqing Zhu, Chenghao Feng, Zixuan Jiang, et al. Adept: Automatic differentiable design of photonic tensor cores. In *Proc. DAC*, 2022.

[77] Jiaqi Gu, Hanqing Zhu, Chenghao Feng, Mingjie Liu, Zixuan Jiang, Ray T. Chen, and David Z. Pan. Towards Memory-Efficient Neural Networks via Multi-Level in situ Generation. In *Proc. ICCV*, October 2021.

[78] Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. In *Proc. NeurIPS*, 2021.

[79] Daniel Haase and Manuel Amthor. Rethinking depthwise separable convolutions: How intra-kernel correlations lead to improved mobilenets. In *Proc. CVPR*, 2020.

[80] Ryan Hamerly, Liane Bernstein, Alexander Sludds, et al. Large-scale optical neural networks based on photoelectric multiplication. *Phys. Rev. X*, 2019.

[81] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. EIE: Efficient Inference Engine on Compressed Deep Neural Network. In *Proc. ISCA*, 2016.

[82] Song Han, Huizi Mao, and William Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *Proc. ICLR*, 2016.

[83] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Proc. NIPS*, 2015.

[84] Nicholas C. Harris et al. Efficient, compact and low loss thermo-optic phase shifter in silicon. *Opt. Express*, 2014.

[85] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[86] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, pages 770–778, 2016.

[87] Zhezhi He, Jie Lin, Rickard Ewetz, et al. Noise injection adaption: End-to-end reram crossbar non-ideal effect adaption for neural network mapping. In *Proc. DAC*, 2019.

[88] D. Hillerkuss, A. Marculescu, J. Li, M. Teschke, G. Sigurdsson, K. Worms, S. B. Ezra, N. Narkiss, W. Freude, and J. Leuthold. Novel optical fast fourier transform scheme enabling real-time ofdm processing at 392 gbit/s and beyond. In *Proc. IEEE OFC*, 2010.

[89] D. Hillerkuss, M. Winter, M. Teschke, A. Marculescu, J. Li, G. Sigurdsson, K. Worms, S. Ben Ezra, N. Narkiss, W. Freude, and J. Leuthold. Simple all-optical fft scheme enabling tbit/s real-time signal processing. *Opt. Express*, Apr 2010.

[90] Geoffrey Hinton. Neural networks for machine learning. *Coursera Video Lecture*, 2012.

[91] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[92] M. Hu, J. P. Strachan, Z. Li, et al. Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication. In *Proc. DAC*, 2016.

[93] C. Huang et al. A silicon photonic–electronic neural network for fibre nonlinearity compensation. *Nat. Electron.*, 2021.

[94] Chaoran Huang, Simon Bilodeau, Thomas Ferreira de Lima, et al. Demonstration of scalable microring weight bank control for large-scale photonic integrated circuits. *APL Photonics*, 5(4):040803, 2020.

[95] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. In *Proc. CVPR*, pages 2261–2269, 2017.

[96] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep Networks with Stochastic Depth. In *Proc. ECCV*, 206.

[97] Lei Huang, Xianglong Liu, Bo Lang, Adams Wei Yu, Yongliang Wang, and Bo Li. Orthogonal Weight Normalization: Solution to Optimization over Multiple Dependent Stiefel Manifolds in Deep Neural Networks. In *Proc. AAAI*, 2018.

[98] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, et al. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.*, 2017.

[99] Tyler W. Hughes, Momchil Minkov, Yu Shi, and Shanhui Fan. Training of photonic neural networks through in situ backpropagation and gradient measurement. *Optica*, 2018.

[100] Tyler W. Hughes, Momchil Minkov, Ian A. D. Williamson, and Shanhui Fan. Adjoint method and inverse design for nonlinear nanophotonic devices. *ACS Photonics*, 2018.

[101] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and$<$ 0.5MB model size. In *Proc. ICLR*, 2017.

[102] Y. Ji, Y. Zhang, S. Li, et al. NEUTRAMS: Neural network transformation and co-design under neuromorphic hardware constraints. In *Proc. MICRO*, 2016.

[103] Yu Ji, Youhui Zhang, Wenguang Chen, et al. Bridge the gap between neural networks and neuromorphic hardware with a neural network compiler. In *Proc. ASPLOS*, 2018.

[104] Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, et al. Highly Scalable Deep Learning Training System with Mixed-Precision: Training ImageNet in Four Minutes. *arXiv preprint arXiv:1807.11205*, 2018.

[105] Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljačiundefined. Tunable efficient unitary neural networks (EUNN) and their application to RNNs. In *Proc. ICML*, 2017.

[106] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proc. ISCA*, 2017.

[107] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *Proc. CVPR*, 2011.

[108] W. Kim, Robert L. Bruce, T. Masuda, G. Fraczak, Nanbo Gong, Praneet Adusumilli, Stefano Ambrogio, Hsinyu Tsai, J. Bruley, J.-P. Han, M. Longstreet, F. Carta, K. Suu, and Matthew J. BrightSky. Confined pcm-based analog synaptic devices offering low resistance-drift and 1000 programmable states for deep learning. *2019 Symposium on VLSI Technology*, pages T66–T67, 2019.

[109] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

[110] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D Object Representations for Fine-grained Categorization. In *International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, 2013.

[111] A Krizhevsky, I Sutskever, and GE Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, 2012.

[112] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[113] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *Proc. ICLR*, 2022.

[114] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lem. Speeding-up convolutional neural networks using fine-tuned cp-decomposition . In *Proc. ICLR*, 2015.

[115] Y. LeCun. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`, 1998.

[116] Juerg Leuthold and Charles H. Joyner. Multimode Interference Couplers with Tunable Power Splitting Ratios. *J. Lightwave Technol.*, 2001.

[117] Jingxi Li, Deniz Mengu, Yi Luo, et al. Class-specific differential detection in diffractive optical neural networks improves inference accuracy. *Advanced Photonics*, pages 1 – 13, 2019.

[118] Mengquan Li, Zhongzhi Yu, Yongan Zhang, Yonggan Fu, and Yingyan Lin. O-has: Optical hardware accelerator search for boosting both acceleration performance and development speed. In *Proc. ICCAD*, 2021.

[119] Shiyu Li, Edward Hanson, Hai Li, and Yiran Chen. PENNI: Pruned Kernel Sharing for Efficient CNN Inference . In *Proc. ICML*, 2020.

[120] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. LocalViT: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021.

[121] Zhe Li, Shuo Wang, Caiwen Ding, et al. Efficient recurrent neural networks using structured matrices in fpgas. In *ICLR Workshop*, 2018.

[122] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint, arXiv:2003.03485*, 2020.

[123] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *Proc. ICLR*, 2021.

[124] Xiangru Lian, Huan Zhang, Cho-Jui Hsieh, Yijun Huang, and Ji Liu. A Comprehensive Linear Speedup Analysis for Asynchronous Stochastic Parallel Optimization from Zeroth-Order to First-Order. In *Proc. NeurIPS*, 2016.

[125] S. Liao, Z. Li, X. Lin, Q. Qiu, Y. Wang, and B. Yuan. Energy-efficient, high-performance, highly-compressed deep neural network design using

block-circulant matrices. In *Proc. ICCAD*, pages 458–465, 2017.

[126] Joowon Lim and Demetri Psaltis. Maxwellnet: Physics-driven deep neural network training based on maxwell's equations. *Appl. Phys. Lett.*, 2022.

[127] D. Liu, Z. Zhao, Z. Wang, Z. Ying, R. T. Chen, and D. Z. Pan. Operon: Optical-electrical power-efficient route synthesis for on-chip signals. In *Proc. DAC*, 2018.

[128] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. In *Proc. ICLR*, 2018.

[129] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *Proc. ICLR*, April 2020.

[130] Sijia Liu, Bhavya Kailkhura, Pin-Yu Chen, Paishun Ting, Shiyu Chang, and Lisa Amini. Zeroth-order stochastic variance reduction for nonconvex optimization. In *Proc. NeurIPS*, 2018.

[131] W. Liu, W. Liu, Y. Ye, Q. Lou, Y. Xie, and L. Jiang. Holylight: A nanophotonic accelerator for deep learning in data centers. In *Proc. DATE*, 2019.

[132] Weiyang Liu, Zhen Liu, Zhiding Yu, et al. Decoupled Networks. In *Proc. CVPR*, 2018.

[133] Yingjie Liu, Zhiyu Li, Shuai Wang, Nan Zhang, Yong Yao, Jiangbing Du, Zuyuan He, Qinghai Song, and Ke Xu. Ultra-compact and polarization-insensitive mmi coupler based on inverse design. In *Proc. IEEE OFC*, 2019.

[134] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *Proc. ICCV*, 2021.

[135] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint, arXiv:1910.03193*, 2019.

[136] Sangkug Lym, Armand Behroozi, Wei Wen, Ge Li, Yongkee Kwon, and Mattan Erez. Mini-Batch Serialization: CNN Training with Inter-Layer Data Reuse. In *Proc. MLSys*, 2017.

[137] Julien Mairal, Piotr Koniusz, Zaid Harchaou, and Cordelia Schmid. Convolutional Kernel Networks. In *Proc. NeurIPS*, 2014.

[138] R. Meade, S. Ardalan, M. Davenport, J. Fini, C. Sun, M. Wade, A. Wright-Gladstein, and C. Zhang. TeraPHY: A high-density electronic-photonic chiplet for optical i/o from a multi-chip module. In *Proc. IEEE OFC*, 2019.

[139] Tomas Mikolov, Martin Karafiát, Lukás Burget, et al. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

[140] Maziyar Milanizadeh, Douglas Aguiar, Andrea Melloni, and Francesco Morichetti. Canceling thermal cross-talk effects in photonic integrated circuits. *J. Light. Technol.*, 2019.

[141] David A.B. Miller. Analyzing and generating multimode optical fields using self-configuring networks. *Optica*, 2020.

[142] Asif Mirza, Febin Sunny, Peter Walsh, Karim Hassan, Sudeep Pasricha, and Mahdi Nikdast. Silicon Photonic Microring Resonators: A Comprehensive Design-Space Exploration and Optimization under Fabrication-Process Variations. *IEEE TCAD*, pages 1–1, 2021.

[143] Mario Miscuglio, Zibo Hu, Shurui Li, Jonathan K. George, Roberto Capanna, Hamed Dalir, Philippe M. Bardet, Puneet Gupta, and Volker J. Sorger. Massively parallel amplitude-only fourier neural network. *Optica*, 7(12):1812–1819, Dec 2020.

[144] Mario Miscuglio, Zibo Hu, Shurui Li, Jiaqi Gu, et al. Million-channel parallelism Fourier-optic convolutional filter and neural network processor. In *Proc. CLEO*, 2020.

[145] Mario Miscuglio and Volker J. Sorger. Photonic tensor cores for machine learning. *Applied Physics Review*, 2020.

[146] M. A. Nahmias, T. F. de Lima, A. N. Tait, H. Peng, B. J. Shastri, and P. R. Prucnal. Photonic multiply-accumulate operations for neural networks. *JSTQE*, 2020.

[147] Hani Nejadriahi and Volker J. Sorger. On-chip integrated all-optical fast fourier transform: Design and analysis. In *Frontiers in Optics*, 2017.

[148] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 2017.

[149] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, et al. Reading Digits in Natural Images with Unsupervised Feature Learning. In *Proc. NIPS*, 2011.

[150] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *Proc. NIPS*, 2011.

[151] Arild Nøkland. Direct Feedback Alignment Provides Learning in Deep Neural Networks. In *Proc. NIPS*, 2016.

[152] Mayumi Ohta, Nathaniel Berger, Artem Sokolov, and Stefan Riezler. Sparse perturbations for improved convergence in stochastic zeroth-order optimization. *arXiv preprint arXiv:2006.01759*, 2020.

[153] Deniz Oktay, Nick McGreivy, Joshua Aduol, Alex Beatson, and Ryan P Adams. Randomized automatic differentiation. In *Proc. ICLR*, 2021.

[154] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[155] B. S. G. Pillai et al. End-to-end energy modeling and analysis of long-haul coherent transmission systems. *J. Light. Technol.*, 2014.

[156] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do Vision Transformers See Like Convolutional Neural Networks? *arXiv preprint arXiv:2108.08810*, 2021.

[157] Md Aamir Raihan and Tor M. Aamodt. Sparse weight activation training. In *Proc. NeurIPS*, 2020.

[158] M. Raissia, P. Perdikarisb, and G. E. Karniadakisa. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comp. Phys.*, 2019.

[159] Carl Ramey et al. Silicon photonics for artificial intelligence acceleration. In *Proc. HotChips*, 2020.

[160] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *Proc. ECCV*, pages 525–542, 2016.

[161] M Reck, A Zeilinger, HJ Bernstein, et al. Experimental realization of any discrete unitary operator. *Physical review letters*, 1994.

[162] Angad S. Rekhi, Brian Zimmer, Nikola Nedovic, et al. Analog/mixed-signal hardware error modeling for deep learning inference. In *Proc. DAC*, 2019.

[163] Antonio Ribeiro, Alfonso Ruocco, Laurent Vanacker, et al. Demonstration of a 4×4-port universal linear circuit. *Optica*, 2016.

[164] C. Roques-Carmes, Y. Shen, and C. Zanoci. Heuristic recurrent algorithms for photonic ising machines. *Nat. Commun.*, 2020.

[165] David Rosenbluth, Konstantin Kravtsov, Mable P. Fok, et al. A high performance photonic pulse processing device. *Opt. Express*, 17(25), Dec 2009.

[166] M. Saberi, R. Lotfi, K. Mafinezhad, and W. A. Serdijn. Analysis of Power Consumption and Linearity in Capacitive Digital-to-Analog Converters Used in Successive Approximation ADCs. *IEEE TCAS I*, 58(8):1736–1748, 2011.

[167] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Proc. ICASSP*, 2013.

[168] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proc. CVPR*, 2018.

[169] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar. ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars. In *Proc. ISCA*, pages 14–26, 2016.

[170] Bhavin J. Shastri, Alexander N. Tait, T. Ferreira de Lima, Wolfram H. P. Pernice, Harish Bhaskaran, C. D. Wright, and Paul R. Prucnal. Photonics for Artificial Intelligence and Neuromorphic Computing. *Nature Photonics*, 2021.

[171] Yichen Shen, Nicholas C. Harris, Scott Skirlo, et al. Deep learning with coherent nanophotonic circuits. *Nature Photonics*, 2017.

[172] Z. Sheng, Z. Wang, C. Qiu, L. Li, A. Pang, A. Wu, X. Wang, S. Zou, and F. Gan. A compact and low-loss mmi coupler fabricated with cmos technology. *IEEE Photonics Journal*, 2012.

[173] Kyle Shiflett, Dylan Wright, Avinash Karanth, and Ahmed Louri. PIXEL: Photonic Neural Network Accelerator. In *Proc. HPCA*, pages 474–487, 2020.

[174] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani.

A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 2013.

[175] Vikas Sindhwani, Tara Sainath, and Sanjiv Kumar. Structured transforms for small-footprint deep learning. In *Proc. NIPS*, 2015.

[176] L. Song, X. Qian, H. Li, et al. Pipelayer: A pipelined reram-based accelerator for deep learning. In *Proc. HPCA*, 2017.

[177] Chen Sun, Mark T. Wade, Yunsup Lee, Jason S, et al. Single-chip microprocessor that communicates directly using light. *Nature*, 2015.

[178] J. Sun, R. Kumar, M. Sakib, et al. A 128 Gb/s PAM4 Silicon Microring Modulator With Integrated Thermo-Optic Resonance Tuning. *Journal of Lightwave Technology*, 2019.

[179] Jie Sun, Ranjeet Kumar, Meer Sakib, Jeffrey B. Driscoll, Hasitha Jayatilleka, and Haisheng Rong. A 128 gb/s pam4 silicon microring modulator with integrated thermo-optic resonance tuning. *Journal of Lightwave Technology*, 37(1):110–115, 2019.

[180] Mengying Sun, Inci M. Baytas, Liang Zhan, Zhangyang Wang, and Jiayu Zhou. Subspace network: Deep multi-task censored regression for modeling neurodegenerative diseases. In *Proc. KDD*, page 2259–2268, 2018.

[181] Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. meProp: Sparsified Back Propagation for Accelerated Deep Learning with Reduced Overfitting. In *Proc. ICML*, 2017.

[182] Zhenyu Sun, Wenqing Wu, and Hai (Helen) Li. Cross-Layer Racetrack Memory Design for Ultra High Density and Low Power Consumption. In *Proc. DAC*, 2013.

[183] Febin Sunny, Asif Mirza, Mahdi Nikdast, and Sudeep Pasricha. CrossLight: A Cross-Layer Optimized Silicon Photonic Neural Network Accelerator. In *Proc. DAC*, pages 1069–1074, 2021.

[184] Febin Sunny, Asif Mirza, Mahdi Nikdast, and Sudeep Pasricha. Crosslight: A cross-layer optimized silicon photonic neural network accelerator. In *Proc. DAC*, 2021.

[185] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proc. CVPR*, 2016.

[186] Mohammad H. Tahersima, Keisuke Kojima, Toshiaki Koike-Akino, Devesh Jha, BingnanWang, and Chungwei Lin. Deep neural network inverse design of integrated photonic power splitters. *Sci. Rep.*, 2019.

[187] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, and Weinan E. Convolutional neural networks with low-rank regularization . In *Proc. ICLR*, 2016.

[188] A. N. Tait, M. A. Nahmias, B. J. Shastri, et al. Broadcast and weight: An integrated network for scalable photonic spike processing. *J. Light. Technol.*, 2014.

[189] Alexander N. Tait. Quantifying power use in silicon photonic neural networks. *arxiv preprint, arXiv:2108.04819*, 2021.

[190] Alexander N. Tait, Thomas Ferreira de Lima, Ellen Zhou, et al. Neuromorphic photonic networks using silicon photonic weight banks. *Sci. Rep.*, 2017.

[191] D. T. H. Tan, A. Grieco, and Y. Fainman. Towards 100 channel dense wavelength division multiplexing with 100ghz spacing on silicon. *Opt. Express*, 2014.

[192] Yehui Tang, Kai Han, Chang Xu, An Xiao, Yiping Deng, Chao Xu, and Yunhe Wang. Augmented Shortcuts for Vision Transformers. In *Proc. NeurIPS*, 2021.

[193] Yingheng Tang, Jichao Fan, Xinwei Li, Jianzhu Ma, Minghao Qi, Cunxi Yu, and Weilu Gao. Physics-guided and physics-explainable recurrent neural network for time dynamics in optical resonances. *Nat. Compu. Sci.*, 2022.

[194] E. Timurdogan et al. An ultralow power athermal silicon modulator. *Nat. Commun.*, 2014.

[195] Erman Timurdogan, Zhan Su, Christopher V. Poulton, et al. AIM Process Design Kit (AIMPDKv2.0): Silicon Photonics Passive and Active Component Libraries on a 300mm Wafer. In *Optical Fiber Communication Conference*, 2018.

[196] A. R. Totović, G. Dabos, N. Passalis, A. Tefas, and N. Pleros. Femtojoule per MAC Neuromorphic Photonics: An Energy and Technology Roadmap. *IEEE Journal of Selected Topics in Quantum Electronics*, 26(5):1–15, 2020.

[197] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers: distillation through attention. In *Proc. ICML*, pages 10347–10357, 2021.

[198] Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *NeurIPS Workshop*, 2021.

[199] Rahul Trivedi, Logan Su, Jesse Lu, Martin F. Schubert, and JelenaVuckovic. Data-driven acceleration of photonic simulations. *Sci. Rep.*, 2019.

[200] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. AutoZOOM: Autoencoder-Based Zeroth Order Optimization Method for Attacking Black-Box Neural Networks. In *Proc. AAAI*, 2019.

[201] Wouter Uijens. Activating frequencies: Exploring non-linearities in the fourier domain. M.S. dissertation, Sch. of Elect. Eng., Math. and Comp. Sci., Delft Univ. of Technology Netherlands., 2018.

[202] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *Proc. CVPR*, 2021.

[203] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. In *Proc. NIPS*, 2017.

[204] Laurent Vivien, Andreas Polzer, Delphine Marris-Morini, et al. Zero-bias 40gbit/s germanium waveguide photodetector on silicon. *Opt. Express*, 2012.

[205] Hong Wang, Hong Qian, and Yang Yu. Noisy Derivative-Free Optimization With Value Suppression. In *Proc. AAAI*, 2018.

[206] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions. In *Proc. ICCV*, 2021.

[207] Wenxiao Wang, Lu Yao, Long Chen, Deng Cai, Xiaofei He, and Wei Liu. CrossFormer: A Versatile Vision Transformer Based on Cross-scale Attention. *arXiv preprint arXiv:2108.00154*, 2021.

[208] Y. Wang, W. Wen, B. Liu, et al. Group scissor: Scaling neuromorphic computing design to large neural networks. In *Proc. DAC*, 2017.

[209] Yitu Wang, Fan Chen, Linghao Song, et al. REBOC: Accelerating Block-Circulant Neural Networks in ReRAM. In *Proc. DATE*, 2020.

[210] Yue Wang, , Ziyu Jiang, Xiaohan Chen, Pengfei Xu, Yang Zhao, Yingyan Lin, and Zhangyang Wang. E2-Train: Training State-of-the-art CNNs with Over 80% Less Energy. In *Proc. NeurIPS*, 2019.

[211] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, , and Huaxia Xia. End-to-end video instance segmentation with transformers. In *Proc. CVPR*, 2021.

[212] Zi Wang, Lorry Chang, Feifan Wang, Tiantian Li, and Tingyi Gu. Integrated photonic metasystem for image classifications at telecommunication wavelength. *Nature Communications*, 2022.

[213] Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M. Benson. U-fno: an enhanced fourier neural operator based-deep learning model for multiphase flow. *arXiv preprint, arXiv:2109.03697*, 2021.

[214] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Proc. NIPS*, 2016.

[215] Gordon Wetzstein, Aydogan Ozcan, Sylvain Gigan, Shanhui Fan, Dirk Englund, Marin Soljačić, Cornelia Denz, , David A. B. Miller, and

Demetri Psaltis. Inference in artificial intelligence with deep optics and photonics. *Nature*, 2020.

[216] Simon Wiedemann, Temesgen Mehari, Kevin Kepp, and Wojciech Samek. Dithered backprop: A sparse and quantized backpropagation algorithm for more efficient deep neural network training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[217] Ian A. D. Williamson, Tyler W. Hughes, Momchil Minkov, Ben Bartlett, Sunil Pai, and Shanhui Fan. Reprogrammable electro-optic nonlinear activation functions for optical neural networks. *JSTQE*, 26(1):1–12, 2020.

[218] Ian A. D. Williamson, Tyler W. Hughes, Momchil Minkov, et al. Reprogrammable electro-optic nonlinear activation functions for optical neural networks. *JSTQE*, 2019.

[219] Scott Wisdom, Thomas Powers, John R. Hershey, et al. Full-capacity unitary recurrent neural networks. In *Proc. NIPS*, 2016.

[220] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, et al. FBNet: Hardware-aware Efficient Convnet Design via Differentiable Neural Architecture Search. In *Proc. CVPR*, 2019.

[221] Changming Wu, Heshan Yu, Seokhyeong Lee, Ruoming Peng, Ichiro Takeuchi, and Mo Li. Programmable phase-change metasurfaces on waveguides for multimode photonic convolutional neural network. *Nature Communications*, 2021.

[222] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR*, abs/1708.07747, 2017.

[223] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early Convolutions Help Transformers See Better. In *Proc. NeurIPS*, 2021.

[224] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. In *Proc. NeurIPS*, 2021.

[225] Shaofu Xu, Jing Wang, Rui Wang, Jiangping Chen, and Weiwen Zou. High-accuracy optical convolution unit architecture for convolutional neural networks by cascaded acousto-optical modulator arrays. *Opt. Express*, 2019.

[226] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. In *Proc. ICCV*, 2021.

[227] Xingyuan Xu, Mengxi Tan, Bill Corcoran, Jiayang Wu, Andreas Boes, Thach G. Nguyen, Sai T. Chu, Brent E. Little, Damien G. Hicks, Roberto

Morandotti, Arnan Mitchell, and David J. Moss. 11 TOPS photonic convolutional accelerator for optical neural networks. *Nature*, 2021.

[228] Yuhui Xu, Yuxi Li, Shuai Zhang, Wei Wen, Botao Wang, Yingyong Qi, Yiran Chen, Weiyao Lin, and Hongkai Xiong. TRP: Trained Rank Pruning for Efficient Deep Neural Networks. In *Proc. IJCAI*, pages 977–983, 2020.

[229] Huanrui Yang, Minxue Tang, Wei Wen, Feng Yan, Daniel Hu, Ang Li, Hai Li, and Yiran Chen. Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification. In *Proc. CVPR Workshops*, 2020.

[230] Zhoufeng Ying, Chenghao Feng, Zheng Zhao, Shounak Dhar, et al. Electronic-photonic arithmetic logic unit for high-speed computing. *Nature Communications*, 2020.

[231] Zhoufeng Ying, Chenghao Feng, Zheng Zhao, et al. Integrated multi-operand electro-optic logic gates for optical computing. *Appl. Phys. Lett.*, 2019.

[232] Zhoufeng Ying, Chenghao Feng, Jiaqi Gu Zheng Zhao, et al. Sequential logic and pipelining in chip-based electronic-photonic digital computing. *IEEE Photonics Journal*, 2020.

[233] J. Yu and X. Zhou. Ultra-high-capacity dwdm transmission system for 100g and beyond. *IEEE Communications Magazine*, 2010.

[234] Qihang Yu, Yingda Xia, Yutong Bai, Yongyi Lu, Alan Yuille, and Wei Shen. Glance-and-Gaze Vision Transformer. *arXiv preprint arXiv:2106.02277*, 2021.

[235] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token ViT: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

[236] Yuhui Yuan, Rao Fu, Lang Huang, Weihong Lin, Chao Zhang, Xilin Chen, and Jingdong Wang. HRFormer: High-Resolution Transformer for Dense Prediction. In *Proc. NeurIPS*, 2021.

[237] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proc. ICCV*, 2019.

[238] H. Zhang, M. Gu, X. D. Jiang, J. Thompson, H. Cai, S. Paesani, R. Santagati, A. Laing, Y. Zhang, M. H. Yung, Y. Z. Shi, F. K. Muhammad, G. Q. Lo, X. S. Luo, B. Dong, D. L. Kwong, L. C. Kwek, and A. Q. Liu. An optical neural chip for implementing complex-valued neural network. *Nature Communications*, 2021.

[239] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, , and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[240] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *Proc. ICCV*, 2021.

[241] Tian Zhang et al. Efficient training and design of photonic neural network through neuroevolution. *arXiv*, 2019.

[242] Tian Zhang, Jia Wang, Yihang Dan, Yuxiang Lanqiu, Jian Dai, Xu Han, Xiaojuan Sun, and Kun Xu. Efficient training and design of photonic neural network through neuroevolution. *Optics Express*, 2019.

[243] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proc. ECCV*, 2018.

[244] Y. Zhang, X. Wang, and E. G. Friedman. Memristor-based circuit design for multilayer neural networks. *IEEE TCAS I*, 2018.

[245] Yang Zhang, Amir Hosseini, Xiaochuan Xu, David Kwong, and Ray T. Chen. Ultralow-loss silicon waveguide crossing using bloch modes in index-engineered cascaded multimode-interference couplers. *Opt. Lett.*, 2013.

[246] Zhekai Zhang, Hanrui Wang, Song Han, and William J. Dally. SpArch: Efficient Architecture for Sparse Matrix Multiplication. In *Proc. HPCA*, 2020.

[247] Liang Zhao, Siyu Liao, Yanzhi Wang, Zhe Li, Jian Tang, and Bo Yuan. Theoretical properties for neural networks with weight matrices of low displacement rank. In *Proc. ICML*, 2017.

[248] Pu Zhao, Pin-Yu Chen, Siyue Wang, and Xue Lin. Towards query-efficient black-box adversary with zeroth-order natural gradient descent. In *Proc. AAAI*, 2020.

[249] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. Building efficient deep neural networks with unitary group convolutions. In *Proc. CVPR*, 2019.

[250] Yang Zhao, Xiaohan Chen, Yue Wang, Chaojian Li, Haoran You, Yonggan Fu, Yuan Xie, Zhangyang Wang, and Yingyan Lin. SmartExchange: Trading Higher-cost Memory Storage/Access for Lower-cost Computation . In *Proc. ISCA*, 2020.

[251] Yang Zhao, Chaojian Li, Yue Wang, Pengfei Xu, Yongan Zhang, and Yingyan Lin. Dnn-chip predictor: An analytical performance predictor for dnn accelerators with various dataflows and hardware architectures. In *Proc. ICASSP*, pages 1593–1597, 05 2020.

[252] Zheng Zhao, Jiaqi Gu, Zhoufeng Ying, et al. Design technology for scalable and robust photonic integrated circuits. In *Proc. ICCAD*, 2019.

[253] Zheng Zhao, Derong Liu, Meng Li, et al. Hardware-software co-design of slimmed optical neural networks. In *Proc. ASPDAC*, 2019.

[254] Qilin Zheng, Zongwei Wang, Zishun Feng, Bonan Yan, Yimao Cai, Ru Huang, Yiran Chen, Chia-Lin Yang, and Hai Helen Li. Lattice: An ADC/DAC-less ReRAM-based Processing-In-Memory Architecture for Accelerating Deep Convolution Neural Networks. In *Proc. DAC*, pages 1–6, 2020.

[255] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proc. AAAI*, 2020.

[256] Hailong Zhou, Yuhe Zhao, Xu Wang, Dingshan Gao, Jianji Dong, and Xinliang Zhang. Self-learning photonic signal processor with an optical neural network chip. *arXiv*, 2019.

[257] Hailong Zhou, Yuhe Zhao, Gaoxiang Xu, Xu Wang, Zhipeng Tan, Jianji Dong, and Xinliang Zhang. Chip-Scale Optical Matrix Computation for PageRank Algorithm. *JSTQE*, 2020.

[258] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[259] Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. Trained ternary quantization. In *Proc. ICLR*, 2017.

[260] H. H. Zhu, J. Zou, H. Zhang, Y. Z. Shi, S. B. Luo, et al. Space-efficient optical computing with an integrated chip diffractive neural network. *Nature Communications*, 2022.

[261] Ying Zhu, Grace Li Zhang, Bing Li, et al. Countering Variations and Thermal Effects for Accurate Optical Neural Networks. In *Proc. ICCAD*, 2020.

[262] Farzaneh Zokaee, Qian Lou, Nathan Youngblood, et al. LightBulb: A Photonic-Nonvolatile-Memory-based Accelerator for Binarized Convolutional Neural Networks. In *Proc. DATE*, 2020.

Index

Vita

Jiaqi Gu received his B.Eng. from Fudan University, Shanghai, China, in 2018. He started his Ph.D. program at the University of Texas at Austin in 2018, with research advisor David Z. Pan and co-advisor Ray T. Chen. He has interned at Meta reality labs, Austin in 2021 summer, and Nvidia, Austin in 2022 summer.

Jiaqi Gu's research interests include emerging post-Moore hardware design for efficient computing, hardware/software co-design, photonic computing, and AI/ML algorithms. He has received the Best Paper Award at the ACM/IEEE Asian and South Pacific Design Automation Conference (ASP-DAC) in 2020, the Best Paper Finalist at the ACM/IEEE Design Automation Conference (DAC) in 2020, the Best Poster Award at the NSF Workshop for Machine Learning Hardware Breakthroughs Towards Green AI and Ubiquitous On-Device Intelligence in 2020, the Best Paper Award at the IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems (TCAD) in 2021, the ACM Student Research Competition Grand Finals First Place in 2021, and Winner of the Robert S. Hilbert Memorial Optical Design Competition in 2022.

Permanent address: jqgu@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.