ADEPT: Automatic Differentiable DEsign of Photonic Tensor Cores

Jiaqi Gu, Hanqing Zhu, Chenghao Feng, Zixuan Jiang, Mingjie Liu, Shuhan Zhang,
Ray T. Chen, David Z. Pan
University of Texas at Austin

{ jqgu, hqzhu, fengchenghao1996, zixuan, jay_liu, shuhan.zhang }@utexas.edu, { chen, dpan }@ece.utexas.edu

ABSTRACT

Photonic tensor cores (PTCs) are essential building blocks for optical artificial intelligence (AI) accelerators based on programmable photonic integrated circuits. PTCs can achieve ultra-fast and efficient tensor operations for neural network (NN) acceleration. Current PTC designs are either manually constructed or based on matrix decomposition theory, which lacks the adaptability to meet various hardware constraints and device specifications. To our best knowledge, automatic PTC design methodology is still unexplored. It will be promising to move beyond the manual design paradigm and "nurture" photonic neurocomputing with AI and design automation. Therefore, in this work, for the first time, we propose a fully differentiable framework, dubbed ADEPT, that can efficiently search PTC designs adaptive to various circuit footprint constraints and foundry PDKs. Extensive experiments show superior flexibility and effectiveness of the proposed ADEPT framework to explore a large PTC design space. On various NN models and benchmarks, our searched PTC topology outperforms prior manually-designed structures with competitive matrix representability, **2×-30×** higher footprint compactness, and better noise robustness, demonstrating a new paradigm in photonic neural chip design. The code of ADEPT is available at link using the TorchONN library.

1 INTRODUCTION

With the advance in integrated photonics, the optical neural network (ONN) has become a promising candidate for ultra-efficient deep neural network (DNN) acceleration [2, 11, 13, 14]. As light propagates through the photonic integrated circuits (PICs), computation-intensive matrix multiplication can be achieved with sub-nanosecond latency and near-zero energy consumption [11]. Shen *et al.* [14] demonstrated a triangular photonic mesh with cascaded Mach-Zehnder interferometers (MZIs) to realize matrix multiplication using optics. They use singular value decomposition (SVD) to decompose the weight matrix W into $U\Sigma V$, parametrize the unitary matrices U and V with a series of planar rotators, and map them into a triangular photonic mesh. This matrix-decomposition-based photonic tensor core (PTC) design is universal but suffers from high area cost and unsatisfying noise robustness. To improve the area efficiency, a Fourier-transform (FFT) based PTC [5–7] was proposed that shrinks the circuit depth from linear to logarithmic using a butterfly circuit topology. This design removes large MZIs and constructs the PTC with smaller basic optical components instead.

However, previous PTCs are all hand-designed based on matrix decomposition theory, which leaves a large design space unexplored and lacks the adaptability to meet various device specifications and hardware constraints. Specifically, the MZI-based PTC [14] is universal at the cost of high area cost and low compute density. The FFT-based PTC [5–7] 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 [15], 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) [10, 16] 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 prevents 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.

DAC '22, July 10–14, 2022, San Francisco, CA, USA

ACM ISBN 978-1-4503-9142-9/22/07…$15.00
https://doi.org/10.1145/3489517.3530562

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, 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 handcrafted structures with higher flexibility, competitive expressiveness, **2×-30×** smaller footprint, and superior noise robustness.

2 BACKGROUND

2.1 Photonic Computing Basics

To perform neurocomputing 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.

2.2 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 [3, 14] 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 [6, 7] 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 [1] and AIM photonics [15], 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.

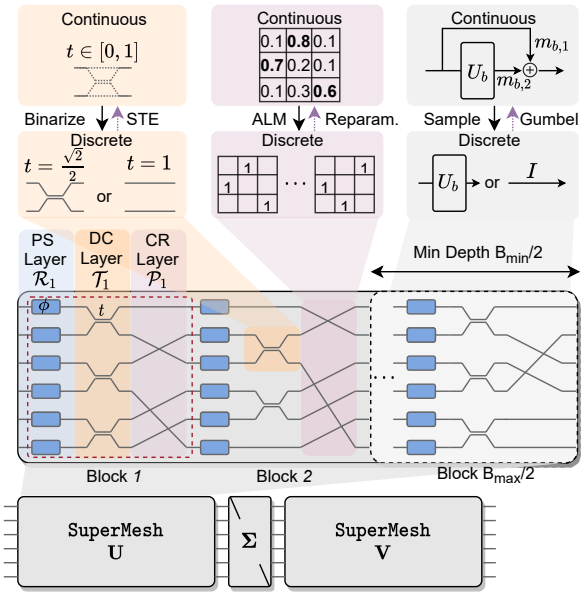

Figure 1: Overview of the probabilistic photonic SuperMesh.

2.3 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 [10] enables DNAS by using a softmax function to relax the categorical choice of candidate operations. FBNet [16] represents the search space by a stochastic SuperNet and then applies DNAS to discover low-latency DNN designs.

Recently, O-HAS [9] 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.

3 AUTOMATIC PHOTONIC TENSOR CORE DESIGN FRAMEWORK ADEPT

3.1 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,

$$
\begin{aligned}
&\min_{\alpha\in\mathcal{A}} \mathcal{L}(W^{*\alpha};\ \mathcal{D}^{val}), \quad \alpha = (B^U, B^V, \mathcal{P}, \mathcal{T}) \\
&\text{s.t. } W^* = \underset{W}{\operatorname{argmin}}\, \mathcal{L}(W^{\alpha};\ \mathcal{D}^{trn}),\ F_{min} \le \mathcal{F}(\alpha) \le F_{max}, \\
&W^{\alpha} \in \mathbb{C}^{M\times N} = \{W^{\alpha}_{pq}\}_{p=1,q=1}^{p=P,q=Q} = \{U^{\alpha}_{pq}\Sigma_{pq}V^{\alpha}_{pq}\}_{p=1,q=1}^{p=P,q=Q}, \\
&B^U, B^V \in [B_{min}/2, B_{max}/2], W_{pq} \in \mathbb{C}^{K\times K}, \\
&\mathcal{P} = (\cdots, \mathcal{P}_b, \cdots, \mathcal{P}_{B^U+B^V}), \mathcal{T} = (\cdots, \mathcal{T}_b, \cdots, \mathcal{T}_{B^U+B^V}).
\end{aligned} \tag{1}
$$

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}

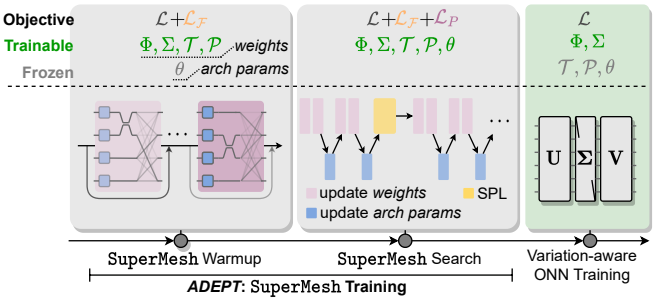

Figure 2: The proposed photonic SuperMesh training flow ADEPT, followed by variation-aware ONN training.

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.

3.2 Search Space Specification

Our search space focuses on the *tensor core circuit topology*, not layer configurations like conventional NAS work. As illustrated in Fig. 1, we define the following block-wise search space for the unitaries,

$$U^{\alpha}_{pq} = \prod_{b=1}^{B^U} \mathcal{P}_b \mathcal{T}_b \mathcal{R}(\Phi^b_{pq}), \quad V^{\alpha}_{pq} = \prod_{b=B^U+1}^{B^U+B^V} \mathcal{P}_b \mathcal{T}_b \mathcal{R}(\Phi^b_{pq}). \tag{2}$$

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.

The first structure in the block is a column of K phase shifters, which can be described by a diagonal matrix $\mathcal{R}(\Phi^b_{pq})$,

$$\mathcal{R}(\Phi^b_{pq}) = \mathtt{diag}(e^{-j\phi_1}, \cdots, e^{-j\phi_K}). \tag{3}$$

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. 1.

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,

$$\mathcal{P}_b \in \{0,1\}^{K\times K}, \sum_j \mathcal{P}_b^{i,j} = 1 \ \forall i \in [K], \sum_i \mathcal{P}_b^{i,j} = 1 \ \forall j \in [K]. \tag{4}$$

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 $O\big((K \cdot K!/2)^{B_{max}}\big)$.

3.3 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. 2.

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,

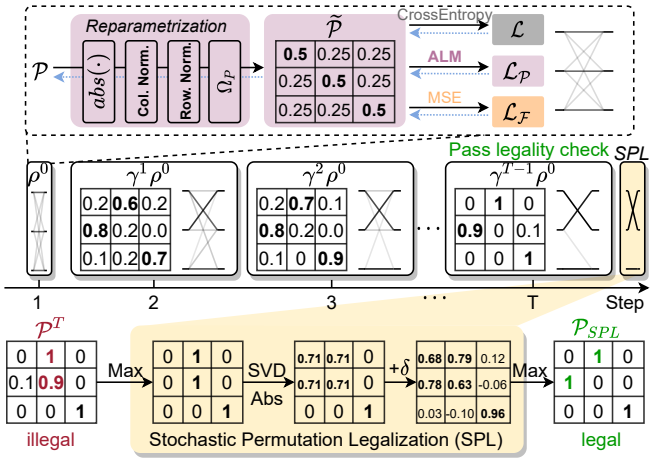

Figure 3: *Top*: permutation optimization procedure. *Bottom*: an example for stochastic permutation legalization (SPL).

(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*, and (2) B belong to the *architecture parameter* group. The entire ADEPT flow contains two stage, shown in Fig. 2. 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.

3.3.1 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

$$P_{\theta_b}(U_b = U_{b,i}) = e^{\theta_{b,i}} \Big/ \sum_i e^{\theta_{b,i}}. \tag{5}$$

The probability distribution of block-b is parametrized by the sampling coefficient θ_b. The forward propagation of the b-th block is,

$$x_{b+1} = \sum_{i=1}^{2} m_{b,i} U_{b,i} x_b, \quad U_{b,1} = I, \ U_{b,2} = \mathcal{P}_b \mathcal{T}_b \mathcal{R}_b, \tag{6}$$

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 [16] is employed as follows,

$$m_{b,i} = \mathtt{GumbelSoftmax}(\theta_{b,i}|\theta_b) = e^{(\theta_{b,i}+g_{b,i})/\tau} \Big/ \sum_i e^{(\theta_{b,i}+g_{b,i})/\tau}. \tag{7}$$

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., $m_{b,2} = 1, \forall b > B_{max}/2 - B_{min}/2$.

3.3.2 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) has a continuous format [4],

$$\mathcal{P}_b \geq 0;\ \|\mathcal{P}_b^{i,:}\|_1 = \|\mathcal{P}_b^{i,:}\|_2, \forall i;\ \|\mathcal{P}_b^{:,j}\|_1 = \|\mathcal{P}_b^{:,j}\|_2, \forall j, \tag{8}$$

where the row-wise and column-wise ℓ_1-norm equals to the ℓ_2-norm. Eq. (8) can be relaxed to its convex hull, i.e., Birkhoff polytope,

$$\mathcal{P}_b \geq 0,\ \mathbf{1}^T\mathcal{P}_b = \mathbf{1}^T,\ \mathcal{P}_b\mathbf{1} = \mathbf{1},\ \mathbf{1} = (1, \cdots, 1)^T. \tag{9}$$

As shown in Fig. 3, 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,

$$\begin{aligned}\mathcal{L}_P = &\sum_{b=1}^{B_{max}}\sum_{i=1}^{K}\lambda_{b,i}^r\Delta\widetilde{\mathcal{P}}_b^{i,:} + \sum_{b=1}^{B_{max}}\sum_{j=1}^{K}\lambda_{b,j}^c\Delta\widetilde{\mathcal{P}}_b^{:,j}\\ &+\frac{\rho}{2}\sum_{b=1}^{B_{max}}\sum_{i=1}^{K}\lambda_{b,i}^r(\Delta\widetilde{\mathcal{P}}_b^{i,:})^2+\frac{\rho}{2}\sum_{b=1}^{B_{max}}\sum_{j=1}^{K}\lambda_{b,j}^c(\Delta\widetilde{\mathcal{P}}_b^{:,j})^2,\end{aligned} \tag{10}$$

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. (9). We (1) first apply 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,

$$\begin{aligned}&\mathcal{P}_b' = \frac{|\mathcal{P}_b|}{\mathbf{1}^T|\mathcal{P}_b|}, \quad \mathcal{P}_b'' = \frac{\mathcal{P}_b'}{\mathcal{P}_b'\mathbf{1}}, \quad \widetilde{\mathcal{P}}_b = \Omega_P(\mathcal{P}_b'')\\ &\Omega_P(\mathcal{P}_b''^{i,j}) = \begin{cases}\texttt{Round}(\mathcal{P}_b''^{i,j}) \text{ if } \max(\mathcal{P}_b''^{i,:}) \geq 1-\epsilon,\\ \mathcal{P}_b''^{i,j} \text{ if } \max(\mathcal{P}_b''^{i,:}) < 1-\epsilon\end{cases},\end{aligned} \tag{11}$$

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,

$$\lambda_{b,i}^r \mathrel{+}= \rho\big(\Delta\widetilde{\mathcal{P}}_b^{i,:} + \frac{1}{2}(\Delta\widetilde{\mathcal{P}}_b^{i,:})^2\big), \quad \lambda_{b,j}^c \mathrel{+}= \rho\big(\Delta\widetilde{\mathcal{P}}_b^{:,j} + \frac{1}{2}(\Delta\widetilde{\mathcal{P}}_b^{:,j})^2\big). \tag{12}$$

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. 3. 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. (10), our ALM-based method does not guarantee convergence to a legal permutation. Instead, it may stuck at saddle points shown in Fig. 3. To force $\widetilde{\mathcal{P}}$ to a legal permutation after SuperMesh training, we propose the following stochastic permutation legalization (SPL),

$$PSQ^* = \texttt{SVD}\big(\texttt{Softmax}(\mathcal{P}/\tau)\big)\big|_{\tau\to 0^+},\ \mathcal{P}_{\texttt{SPL}} = \texttt{Softmax}\big((|PQ^*| + \delta)/\tau\big)\big|_{\tau\to 0^+}, \tag{13}$$

where $\delta \sim \mathcal{N}(0, \sigma^2)$. We given an example in Fig. 3. 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.

3.3.3 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 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,

$$\begin{aligned}&T(t_q) = T(Q(t)),\ Q(t) = (\texttt{sign}(t) + 1) \times \frac{2-\sqrt{2}}{4} + \frac{\sqrt{2}}{2},\\ &\frac{\partial\mathcal{L}}{\partial t} = \min\Big(1, \max\big(-1, \frac{\partial\mathcal{L}}{\partial t_q} \times \frac{2-\sqrt{2}}{4}\big)\Big).\end{aligned} \tag{14}$$

3.3.4 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.

3.4 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 penalty* term $\mathcal{L}_{\mathcal{F}}$,

$$\begin{aligned}&\mathcal{L}_{\mathcal{F}} = \begin{cases}\beta\Big(\mathbb{E}[\mathcal{F}_{\text{prox}}(\alpha)]/\widehat{F}_{max}\Big), & \mathbb{E}[\mathcal{F}(\alpha)] > \widehat{F}_{max},\\ -\beta\Big(\mathbb{E}[\mathcal{F}_{\text{prox}}(\alpha)]/\widehat{F}_{min}\Big), & \mathbb{E}[\mathcal{F}(\alpha)] < \widehat{F}_{min},\\ 0, & \text{otherwise},\end{cases}\\ &\mathbb{E}[\mathcal{F}(\alpha)] = \sum_{b=1}^{B_{max}} m_{b,2}\mathcal{F}_b, \quad \mathbb{E}[\mathcal{F}_{\text{prox}}(\alpha)] = \sum_{b=1}^{B_{max}} m_{b,2}\mathcal{F}_{b,\text{prox}},\\ &\mathcal{F}_b = \#\texttt{PS}(\mathcal{R}_b)\cdot\mathcal{F}_{\texttt{PS}} + \#\texttt{DC}(\mathcal{T}_b)\cdot\mathcal{F}_{\texttt{DC}} + \#\texttt{CR}(\mathcal{P}_b)\cdot\mathcal{F}_{\texttt{CR}},\\ &\mathcal{F}_{b,\text{prox}} = \#\texttt{PS}(\mathcal{R}_b)\cdot\mathcal{F}_{\texttt{PS}} + \#\texttt{DC}(\mathcal{T}_b)\cdot\mathcal{F}_{\texttt{DC}} + \beta_{\texttt{CR}}\|\widetilde{\mathcal{P}}_b - I\|_2^2\cdot\mathcal{F}_{\texttt{CR}},\\ &\#\texttt{PS}(\mathcal{R}_b) = K,\ \#\texttt{DC}(\mathcal{T}_b) = \sum_{i=1}^{(K-s_b)/2}\Big(\frac{2Q(t_i)}{\sqrt{2}-2} + \frac{2}{2-\sqrt{2}}\Big),\end{aligned} \tag{15}$$

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., $\#\texttt{PS}(\mathcal{R}_b) = K$.

Table 1: 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 [1], 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 [14]	FFT-ONN [6]	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

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 #CR ($\mathcal{P}_b$) itself is non-differentiable. When calculating the footprint penalty, we replace $\#\text{CR}(\mathcal{P}_b)\mathcal{F}_{\text{CR}}$ with a differentiable proxy term $\beta_{\text{CR}}\mathcal{F}_{\text{CR}}\|\widetilde{\mathcal{P}_b} - I\|_2^2$, where β_{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,

$$\begin{aligned} &\mathcal{F}_{b,min} = K\mathcal{F}_{\text{PS}} + \mathcal{F}_{\text{DC}},\ \mathcal{F}_{b,max} = \mathcal{F}_{b,min} + K\mathcal{F}_{\text{DC}}/2 + K(K-1)\mathcal{F}_{\text{CR}}/2, \\ &B_{max} = \lceil F_{max}/\mathcal{F}_{b,min} \rceil, \quad B_{min} = \lfloor F_{min}/\mathcal{F}_{b,max} \rfloor. \end{aligned} \tag{16}$$

4 EXPERIMENTAL RESULTS

4.1 Experiment Setup

Datasets. We search PTCs on MNIST and evaluate on MNIST, FashionMNIST, SVHN [12], 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 β_{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.

4.2 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 1, we compare our ADEPT-series to prior manual PTC designs, i.e., MZI-ONN [14] and FFT-based ONN [6, 7] 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 [7]. 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 [1] to AIM photonics [15], which provides much larger waveguide crossings. In Table 2, ADEPT finds feasible PTC topology that avoids using many crossings to honor 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 3. 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, we inject phase drifts into the circuit and perform variation-aware training on all PTC designs [8, 17]. Even with noise-aware training, the MZI-based ONN still suffers a severe 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.

Table 2: MNIST accuracy with 16×16 PTCs on AIM photonics PDKs [15], 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 [14]	FFT-ONN [6, 7]	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

Table 3: 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 [14]	FFT [6, 7]	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

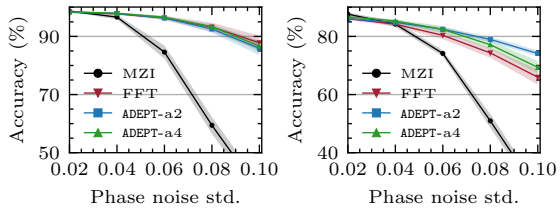

Figure 4: Robustness evaluation of 16×16 PTCs with various phase noise intensities. (a) 2-layer CNN on MNIST. (b) LeNet-5 on FMNIST. All models are trained with variation-aware training. The shadow marks ±3σ uncertainty over 20 runs.

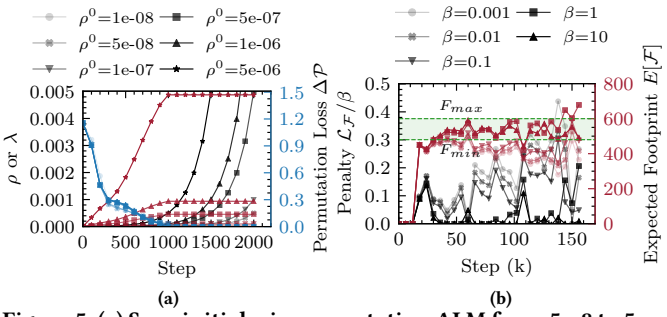

Figure 5: (a) Scan initial ρ in permutation ALM from 5e-8 to 5e-6. The red lines are averaged λ. Blue curves are permutation errors, i.e., average difference between ℓ_1-norm and ℓ_2-norm. (b) Scan β in footprint penalty from 0.001 to 10. Red lines are expected footprint $\mathbb{E}[\mathcal{F}(\alpha)]$ of ADEPT-a1. Black curves are footprint penalty. The green region marks the constraint.

4.3 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. 5(a). Our method is insensitive to the hyper-parameter settings and can stably converge with the proposed adaptive penalty scheduling.

Footprint Penalty. In Fig. 5(b), the expected PTC footprint is visualized with different penalty strengths. With β=~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.

5 CONCLUSION

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.

ACKNOWLEDGMENTS

The authors acknowledge the Multidisciplinary University Research Initiative (MURI) program through the Air Force Office of Scientific Research (AFOSR), contract No. FA 9550-17-1-0071, monitored by Dr. Gernot S. Pomrenke.

REFERENCES

[1] http://www.advmf.com/services/. Advanced Micro Foundry.
[2] Q. Cheng, J. Kwon, M. Glick, M. Bahadori, L. P. Carloni, and K. Bergman. 2020. Silicon Photonics Codesign for Deep Learning. *Proc. IEEE* (2020).
[3] William R. Clements, Peter C. Humphreys, Benjamin J. Metcalf, et al. 2018. Optimal Design for Universal Multiport Interferometers. *Optica* (2018).
[4] Rodrigo Santa Cruz, Basura Fernando, Anoop Cherian, and Stephen Gould. 2017. DeepPermNet: Visual Permutation Learning. In *Proc. CVPR.*
[5] Chenghao Feng, Jiaqi Gu, Hanqing Zhu, et al. 2021. Silicon photonic subspace neural chip for hardware-efficient deep learning. *arXiv:2111.06705* (2021).
[6] Jiaqi Gu, Zheng Zhao, Chenghao Feng, et al. 2020. Towards Area-Efficient Optical Neural Networks: an FFT-based architecture. In *Proc. ASPDAC.*
[7] Jiaqi Gu, Zheng Zhao, Chenghao Feng, et al. 2020. Towards Hardware-Efficient Optical Neural Networks: Beyond FFT Architecture via Joint Learnability. *IEEE TCAD* (2020).
[8] Jiaqi Gu, Zheng Zhao, Chenghao Feng, Hanqing Zhu, Ray T. Chen, and David Z. Pan. 2020. ROQ: A Noise-Aware Quantization Scheme Towards Robust Optical Neural Networks with Low-bit Controls. In *Proc. DATE.*
[9] Mengquan Li, Zhongzhi Yu, Yongan Zhang, Yonggan Fu, and Yingyan Lin. 2021. O-HAS: Optical Hardware Accelerator Search for Boosting Both Acceleration Performance and Development Speed. In *Proc. ICCAD.*
[10] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. DARTS: Differentiable Architecture Search. In *Proc. ICLR.*
[11] Mario Miscuglio and Volker J. Sorger. 2020. Photonic Tensor Cores for Machine Learning. *Applied Physics Review* (2020).
[12] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, et al. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning. In *Proc. NIPS.*
[13] Bhavin J. Shastri, Alexander N. Tait, T. Ferreira de Lima, Wolfram H. P. Pernice, Harish Bhaskaran, C. D. Wright, and Paul R. Prucnal. 2021. Photonics for Artificial Intelligence and Neuromorphic Computing. *Nature Photonics* (2021).
[14] Yichen Shen, Nicholas C. Harris, Scott Skirlo, et al. 2017. Deep Learning with Coherent Nanophotonic Circuits. *Nature Photonics* (2017).
[15] Erman Timurdogan, Zhan Su, Christopher V. Poulton, et al. 2018. AIM Process Design Kit (AIMPDKv2.0): Silicon Photonics Passive and Active Component Libraries on a 300mm Wafer. In *Optical Fiber Communication Conference.*
[16] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, et al. 2019. FBNet: Hardware-aware Efficient Convnet Design via Differentiable Neural Architecture Search. In *Proc. CVPR.*
[17] Zheng Zhao, Jiaqi Gu, Zhoufeng Ying, et al. 2019. Design Technology for Scalable and Robust Photonic Integrated Circuits. In *Proc. ICCAD.*