Hardware-software Co-design of Slimmed Optical Neural Networks

Zheng Zhao
UT Austin
zhengzhao@utexas.edu

Derong Liu
UT Austin
deronliu@utexas.edu

Meng Li
UT Austin
mengli@utexas.edu

Zhoufeng Ying
UT Austin
zfying@utexas.edu

Lu Zhang
CUHK
lzhang@cse.cuhk.edu.hk

Biying Xu
UT Austin
biying@utexas.edu

Bei Yu
CUHK
byu@cse.cuhk.edu.hk

Ray T. Chen
UT Austin
chen@ece.utexas.edu

David Z. Pan
UT Austin
dpan@ece.utexas.edu

ABSTRACT

Optical neural network (ONN) is a neuromorphic computing hardware based on optical components. Since its first on-chip experimental demonstration, it has attracted more and more research interests due to the advantages of ultra-high speed inference with low power consumption. In this work, we design a novel slimmed architecture for realizing optical neural network considering both its software and hardware implementations. Different from the originally proposed ONN architecture based on singular value decomposition which results in two implementation-expensive unitary matrices, we show a more area-efficient architecture which uses a sparse tree network block, a single unitary block and a diagonal block for each neural network layer. In the experiments, we demonstrate that by leveraging the training engine, we are able to find a comparable accuracy to that of the previous architecture, which brings about the flexibility of using the slimmed implementation. The area cost in terms of the Mach-Zehnder interferometers, the core optical components of ONN, is 15%-38% less for various sizes of optical neural networks.

1 INTRODUCTION

Computing using light has recently been reignited as a promising alternative to digital electronics, for its ultra-high speed and power efficient on-chip optical interconnects and computing as Moore's law winds down [1, 2]. Besides the intensive study on realizing optical logics [3–6] and developing optical on-chip interconnect for on- and inter-chip communications [7, 8], optical neural network (ONN) distinguishes itself by directly exploiting optical principles to perform neuromorphic operations, diverging from the long-established digital paradigms.

The recent research of ONN expands across optical reservoir computing [9, 10], photonic spike processing [11, 12] and the design and realization of integrated photonic platforms for multilayer perceptron (MLP) inference [13]. MLP is the basic neural network architecture which lays the foundation for the more advanced convolutional and recurrent neural networks. In MLP, the core and performance-critical computation is the matrix multiplication, a computationally expensive operation for electronics. While for optics, this computation has been successfully demonstrated with an interconnected array of optical components [14–17], by which matrix multiplication can be performed with near-zero energy. Furthermore, the computation is executed at the speed of light and the detection rate can go over 100 GHz with the state-of-the-art electro-optical components [13].

Based on these works, Shen *et al.* [13] present and fabricate a generic MLP platform consisting of an interconnected array of Mach-Zehnder interferometers (MZIs) with thermal phase shifters on the arms of the waveguides to realize the matrix multiplications of the hidden layers of the network. Once the network is trained by software, the phase of each phase shifter can be computed and set up. The whole optical implementation thus becomes completely passive such that the power consumption is minimal. Furthermore, because optical signals can transport in the same channel independently with wavelength-division multiplexing (WDM), it offers the possibilities of scaling the process bandwidth by tens of times. In other words, the same ONN can process tens of inputs in parallel.

Our work slims down the previous MLP architecture proposed in [13] by a software-hardware co-design methodology. Compared with the previous ONN architecture where the optical hardware implementation is derived from a software-trained neural network, we adopt a different methodology where the optical hardware implementation and software training implementation are co-designed to reduce the hardware implementation cost. As will be shown, the co-designed architecture eliminates one of the area-expensive unitary blocks of the previous architecture by leveraging a much smaller sparse tree network. The loss of the testing accuracy is negligible compared to the previous architecture. The main contributions of this paper are summarized as follows:

- We propose an area-efficient architecture for optical neural network where the number of optical components are effectively reduced.
- We co-study the hardware and software implementation of the proposed architecture, incorporating the optical structures and constraints as well as their software embodiment.
- We demonstrate experimentally that the slimmed architecture can lead to an area saving of 15%-38% and better robustness compared with the previous basic architecture with minimal accuracy loss.

ASPDAC '19, January 21–24, 2019, Tokyo, Japan

ACM ISBN 978-1-4503-6007-4/19/01…$15.00
https://doi.org/10.1145/3287624.3287720

The remainder of this paper is organized as follows. Section 2 introduces the principles of ONN building blocks, followed by the basic ONN architecture. Section 3 presents our slimmed architecture and co-design details. Section 4 reports the experimental results on MNIST dataset [18] for MLP neural network, followed by the conclusion in Section 5.

2 BACKGROUND

In this section, we introduce the principles of the fundamental optical components of optical neural network (ONN). Then the standard ONN architecture is described.

2.1 Mach-Zehnder Interferometer

Mach-Zehnder interferometer (MZI) is the fundamental building block of the optical neural network. Figure 1(a) shows the schematic of a 2×2 MZI. The working principle of MZI is based on interference of light. Initially, two lights source from the inputs, entering the first coupling region and split into the two arms of the interferometer by the input coupler and re-combined by the output coupler. With the phase difference (ϕ) induced by the coupling region in the middle, MZI can provide constructive or destructive interference of the two input lights. Specifically, suppose the input modal amplitudes are y_1 and y_2 and output modal amplitudes are y_1', y_2'; the transfer relation of a 2×2 MZI can be written as

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}. \tag{1}$$

Thus a response dependent on the phase difference (realized by a phase shifter) is produced on the amplitude of the output light. The phase difference ϕ is determined by $\phi = 2\pi/\lambda \cdot \Delta N_{eff} \cdot L$, where λ is the light wavelength, N_{eff} is the effective refractive index for the propagation mode of the waveguide and L is the sensitive arm length. A heater or electrical signal can be employed to control the phase shift by adjusting the effective refractive index N_{eff}, so that the transfer behavior of MZI can be adjusted.

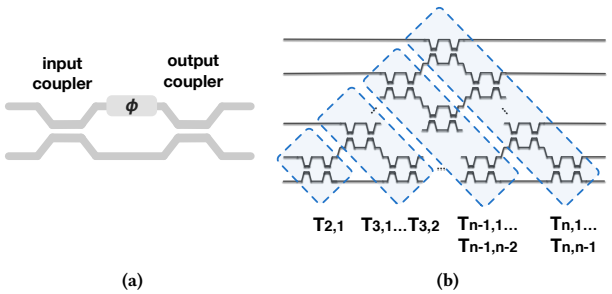

Figure 1: (a) MZI schematic; (b) MZI array for unitary implementation.

2.2 ONN Architecture

The fundamental optical neural network [13] realizes the basic MLP neural network. In MLP, each hidden layer is a fully connected layer. The optical implementation of each layer is shown in Figure 2. The fully connected layer has two parts: the linear part which realizes a weight matrix $\boldsymbol{W}$ and the non-linear part which realizes an activation function.

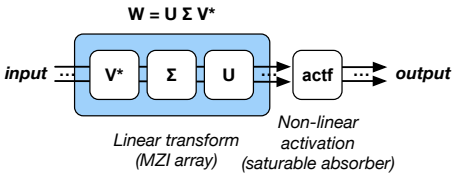

Figure 2: Basic ONN architecture layer.

To implement an $m \times n$ real weight matrix $\boldsymbol{W}$ for the linear part, it is first decomposed into three matrices using singular value decomposition $\boldsymbol{W} \overset{\text{svd}}{=} \boldsymbol{U}\Sigma\boldsymbol{V}^*$, where $\boldsymbol{U}$ is an $m \times m$ unitary matrix; Σ is an $m \times n$ non-square diagonal matrix whose diagonal values are non-negative real numbers called singular values; $\boldsymbol{V}^*$ is an $n \times n$ unitary matrix. $(\cdot)^*$ denotes the conjugate transpose. A square matrix $\boldsymbol{A}$ is unitary if the product of it and its conjugate transpose is an identity matrix, i.e., $\boldsymbol{A}\boldsymbol{A}^* = \boldsymbol{I}$. For real matrix $\boldsymbol{W}$, both $\boldsymbol{U}$ and $\boldsymbol{V}^*$ are real and the conjugate transpose $(\cdot)^*$ is equivalent to the transpose operation. Note that $\boldsymbol{V}^*$ is denoted with the transpose symbol as a convention; the physical implementation is directly based on the unitary matrix $\boldsymbol{V}^*$, rather than based on $\boldsymbol{V}$.

Each of the unitary transformations $\boldsymbol{U}$ and $\boldsymbol{V}^*$ is implemented with a triangular planar array with 2×2 MZIs shown in Figure 1(b). As can be verified, the transfer matrix of a 2×2 MZI is a 2-dimensional unitary matrix denoted as $\boldsymbol{SU}(2)$. It is proved in [14] that, by joining multiple $\boldsymbol{SU}(2)$ blocks into the triangular array, we can realize any arbitrary $n \times n$ unitary matrix. In the planar triangular array, each MZI is connected with the neighboring MZIs and thus, each $\boldsymbol{SU}(2)$ will be extended to a special n-dimensional matrix $\boldsymbol{T}_{ij}$. $\boldsymbol{T}_{ij}$ is an n-dimensional identity matrix except for the four elements at index (i,i), (i,j), (j,i) and (j,j), which are replaced by $\boldsymbol{SU}(2)$ elements: $\cos\phi$, $\sin\phi$, -$\sin\phi$, and $\cos\phi$, respectively. It can be seen that $\boldsymbol{T}_{ij}$ is also unitary. The array naturally implements a matrix multiplication of the individual $\boldsymbol{T}_{ij}$ transfer matrices. The resultant $n \times n$ unitary matrix of the triangular MZI array is calculated as [14, 16, 17]:

$$\boldsymbol{U}(n) = \boldsymbol{D} \cdot \prod_{i=n}^{2} \prod_{j=1}^{i-1} \boldsymbol{T}_{ij}. \tag{2}$$

$\boldsymbol{D}$ is a square diagonal matrix. For real unitary matrices $\boldsymbol{U}$ and $\boldsymbol{V}^*$, the elements of $\boldsymbol{D}$ are either 1 or -1. The values can be computed simultaneously with the parameters ϕ's of $\boldsymbol{T}_{ij}$ to satisfy the equation. Given a unitary matrix, the computation of the parameters is derived from the following inductive relation:

$$\boldsymbol{T}^*_{n,n-1} \cdot \boldsymbol{T}^*_{n,n-2} \cdots \boldsymbol{T}^*_{n,1} \cdot \boldsymbol{U}(n) = \begin{pmatrix} \boldsymbol{U}(n-1), & \boldsymbol{0} \\ \boldsymbol{0}, & \pm 1 \end{pmatrix}.$$

The inverse of the consecutive terms $\boldsymbol{T}^*_{n,n-1} \cdot \boldsymbol{T}^*_{n,n-2} \cdots \boldsymbol{T}^*_{n,1}$ corresponds to a squared area as shown in Figure 1(b). Each of $\boldsymbol{T}^*_{n,j}$ for $j = n-1, \cdots, 1$, aims at transforming $\boldsymbol{U}(n)$ such that one element in the last row and in the last column of $\boldsymbol{U}(n)$ become zero. Based on this, the corresponding ϕ is calculated. We can successively apply $\boldsymbol{T}^*_{ij}$'s so that the n-dimensional unitary matrix is reduced by

one dimension. The whole process is performed upon the $(n-1)$-dimensional matrix, etc. Finally, all the off-diagonal elements of the given unitary matrix become 0 and all the phases for the triangular array are decided. As a last step, the triangular array is combined with the diagonal $\boldsymbol{D}$. The 1 and -1 element of $\boldsymbol{D}$ are implemented by a direct waveguide and a single π phase shifter, respectively.

As for the real non-negative diagonal matrix Σ which simply performs a scaling operation, it can be implemented using optical attenuators or optical amplification materials. The area overhead of each attenuator is estimated to be the size of an MZI: indeed, if the scaling factor is smaller than 1, we can readily use an MZI with one of its outputs to realize it. It should be noted that the nonlinear activation part is not easy to be fully integrated in optical domain. The authors of [13] offered the saturable absorber as a potential candidate to simulate the nonlinearity function in their experiment. Based on Equation (2), an $n \times n$ unitary matrix needs $n(n-1)/2$ MZIs and is the most costly block of the architecture. Given an $m \times n$ layer weight matrix $\boldsymbol{W}$, the number of the MZIs for the three parts can be calculated as $m(m-1)/2$, n and $n(n-1)/2$, respectively.

3 SLIMMED ONN ARCHITECTURE

In this section, we discuss the details of the proposed architecture from both hardware and software perspectives. The general architecture of a single fully connected layer is shown in Figure 3. The linear weight matrix ($\boldsymbol{W}$) of each layer is implemented by three parts, a tree network ($\boldsymbol{T}$), a unitary network ($\boldsymbol{U}$) and a diagonal network (Σ). Based on Figure 3, the transfer matrix of the linear part is then

$$\boldsymbol{W} = \boldsymbol{T}\boldsymbol{U}\Sigma.$$

The tree network allows sparse connections of the inputs and outputs of the network. We will discuss its physical implementation as well as the setup and constraints for the training. The next two blocks of the linear part, the unitary network and the diagonal network, share the same physical implementation as their counterparts of the basic neural network [13]. The difference from the basic architecture is, in the training, their parameters are directly encoded as training parameters and are decided by the training engine, so that the neural network accuracy is optimized. Since the area bottleneck comes from the unitary block, by skillfully eliminating one of the unitary matrix, we are able to achieve area reduction.

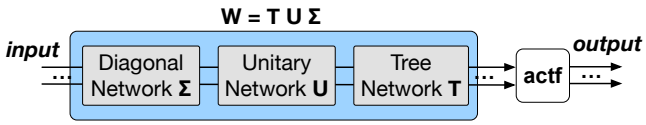

Figure 3: Proposed slimmed layer implementation.

3.1 Tree Network Construction

The goal of the tree network is two folds: to connect the inputs and outputs which may have different dimensions, and to implement the connection efficiently with MZIs. Assume that we have n input and m output ports. The following discussion is based on the case of $n > m$. If $n \leq m$, we can straightforwardly connect the n inputs to the first n of the m outputs without losing the input information. The connection can have different scaling factors which are decided by the training process.

For the case of $n > m$, our goal is to construct a network that is optically implementable and area-efficient. We propose the tree network detailed as follows. We first divide the total n inputs into m groups, each group corresponding to a *subtree* connected to one output port. For the first $(m-1)$ groups, there are $\lfloor n/m \rfloor$ inputs in each individual group; while the residual inputs are contained in the last group for the m-th output. For each subtree, we connect the first two inputs with a single-output MZI (2×1 MZI); the output of their connection is subsequently joined with the third input, etc., until all the inputs are connected.

Figure 4 provides an example of a 7-input 3-output tree network. It is shown that for the first 2 groups, each consists of $\lfloor 7/3 \rfloor$ inputs; their outputs are connected to the first and the second outputs respectively. The residual 3 inputs are formed as a 2-level subtree; the output of the subtree is then connected to the third output. In this way, the first $(m-1)$ subtrees share the same tree-topology and the regularity is enhanced for the ease of fabrication.

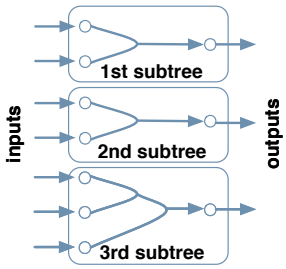

Figure 4: Tree network example.

The transfer matrix of the example tree network is provided as follows, which has 3 rows and 7 columns:

$$\begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_{2,1} & \alpha_{2,2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} \end{pmatrix}$$

where α is employed to denote the transfer ratio, i.e., the modal amplitude ratios, of the input and output of the subtrees. Specifically, for each $N \times 1$ subtree, the output y is a linear combination of the input x's: $y = \sum_{i=1}^{N} \alpha_i \cdot x_i$, each x_i being scaled by the modal amplitude ratio α_i. In the matrix, each box corresponds to one subtree. Since the inputs of the subtrees are independent, for each row which corresponds to an output, the non-zero α elements are also column-wise independent. In other words, for each column, there is at most one non-zero element. The resultant matrix is a sparse matrix with at most n non-zero elements and the non-zero items are distributed along the diagonal line of the matrix.

The tree network is realized by an MZI array. As briefly mentioned, the basic connection components are 2×1 MZIs. By using the upper branch of the 2×2 MZI, we obtain a 2×1 MZI, the transfer function of which is $y_1 = \cos\phi \cdot x_1 + \sin\phi \cdot x_2$. Hence, the modal amplitude ratio $\alpha_1 = \cos\phi$ and $\alpha_2 = \sin\phi$.

Note that *the necessary and sufficient condition of the modal amplitude ratios* (α_1, α_2) *of a feasible* 2×1 *MZI is* $\alpha_1^2 + \alpha_2^2 = 1$, *for*

$-1 \le \alpha_1, \alpha_2 \le 1$. In the physics perspective, the constraint can be directly derived from the energy conservation constraint. Generally, for a subtree with N inputs, the energy conservation requires that $\sum_{i=1}^{N} \alpha_i^2 = 1$ with $-1 \le \alpha_i \le 1$. We can further prove that

Claim 1. Given an N-input subtree with arbitrary modal amplitude ratios satisfying the constraint

$$\alpha_1, \alpha_2, \cdots, \alpha_N, s.t., \sum_{i=1}^{N} \alpha_i^2 = 1, -1 \le \alpha_i \le 1, i = 1, \cdots, N$$

we can implement it by cascading $(N-1)$ (2×1) MZIs, with $N-1$ phases $\phi_1, \phi_2, \cdots, \phi_{N-1}$.

In the other words, as long as the the energy conservation constraint is satisfied, it is always possible to use the proposed tree structure to realize the given amplitude ratios. The claim can be proved by mathematical induction and is omitted due to the page limit. Hence, the area of this construction is upper bounded by the dimension n. The best modal amplitude ratios α_i's are determined by training. The energy conservation constraint is also encoded with these parameters and guaranteed. Once the amplitude ratios are decided, the phase of each subtree component can be calculated iteratively, from the deepest branch which is closest to the output to the shallowest branch which is farthest to the output.

3.2 Unitary and Diagonal Block Construction

Following the tree network, the second and the third blocks of the proposed architecture are the unitary block and the diagonal block. The optical implementations follow the same as the basic architecture. However, in order to save one unitary matrix, SVD is not reflected in our physical architecture. On that account, the software setup for the training is also changed. Basically, we will apply the gradient descent optimization engine to train all the parameters of the unitary block and the diagonal block as well.

For the unitary block $\boldsymbol{U}$, which is constrained by $\boldsymbol{U}\boldsymbol{U}^* = \boldsymbol{I}$, we add the regularization term

$$reg = \|\boldsymbol{U}\boldsymbol{U}^* - \boldsymbol{I}\|_F \tag{3}$$

to the training objective that originally embodies the accuracy objective, where $\|\cdot\|_F$ is the Frobenius norm. This regularization will impel the optimization engine to find a matrix that is close to a unitary matrix. As shown in the experiments, the trained result $\boldsymbol{U}_t$ can be sufficiently close to the real unitary. However, as only a true unitary can be implemented by the MZI array, we further find a closest unitary matrix by leveraging the software SVD-decomposition. Specifically, the trained matrix is decomposed as

$$\boldsymbol{U}_t \overset{\text{svd}}{=} \boldsymbol{P}\boldsymbol{S}\boldsymbol{Q}^*.$$

Since $\boldsymbol{U}_t$ is close to a unitary matrix, meaning that the column vectors being close to orthogonal, the decomposed singular value diagonal matrix $\boldsymbol{S}$ will be close to an identity matrix $\boldsymbol{I}$. Therefore, $\boldsymbol{U}_t$ can be approximated by

$$\boldsymbol{U}_a = \boldsymbol{P}\boldsymbol{Q}^*. \tag{4}$$

The matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ are true unitary and so is their product $\boldsymbol{U}_a$. As for the effectiveness of this approximation, we have the following claim.

Claim 2. To minimize the regularization term reg is equivalent to minimize the Frobenius norm of the difference

$$\epsilon := \|\boldsymbol{U}_t - \boldsymbol{U}_a\|_F.$$

Proof. Rewriting the Frobenius norm with trace Tr,

$$\begin{aligned}\epsilon^2 &= \|\boldsymbol{U}_t - \boldsymbol{U}_a\|_F^2 = Tr[(\boldsymbol{U}_t - \boldsymbol{U}_a)(\boldsymbol{U}_t - \boldsymbol{U}_a)^*] \\ &= Tr(\boldsymbol{U}_t\boldsymbol{U}_t^* - \boldsymbol{U}_t\boldsymbol{U}_a^* - \boldsymbol{U}_a\boldsymbol{U}_t^* + \boldsymbol{U}_a\boldsymbol{U}_a^*) \\ &= Tr(\boldsymbol{U}_t\boldsymbol{U}_t^* + \boldsymbol{U}_a\boldsymbol{U}_a^*) - Tr(\boldsymbol{U}_t\boldsymbol{U}_a^* + \boldsymbol{U}_a\boldsymbol{U}_t^*) \\ &= Tr(\boldsymbol{U}_t\boldsymbol{U}_t^* + \boldsymbol{I}) - 2 \cdot Tr(\boldsymbol{U}_t\boldsymbol{U}_a^*).\end{aligned}$$

Once trained, the matrix $\boldsymbol{U}_t$ is fixed, so $\boldsymbol{U}_t\boldsymbol{U}_t^*$ remains constant. Thus, the minimization of ϵ is equivalent to the maximization of $Tr(\boldsymbol{U}_t\boldsymbol{U}_a^*)$. Substitute $\boldsymbol{U}_t = \boldsymbol{P}\boldsymbol{S}\boldsymbol{Q}^*$ and we obtain

$$Tr(\boldsymbol{U}_t\boldsymbol{U}_a^*) = Tr(\boldsymbol{P}\boldsymbol{S}\boldsymbol{Q}^*\boldsymbol{U}_a^*) = Tr(\boldsymbol{S}\boldsymbol{P}\boldsymbol{Q}^*\boldsymbol{U}_a^*). \tag{5}$$

Let $\boldsymbol{R} := \boldsymbol{P}\boldsymbol{Q}^*\boldsymbol{U}_a^*$ and denote the elements of $\boldsymbol{R}$ to be r_{ij}'s, the singular values contained in $\boldsymbol{S}$ to be s_i's. Since $\boldsymbol{R}$ is unitary, each element $-1 \le r_{ij} \le 1$ must satisfy $\sum_i |r_{ij}|^2 = 1$ for $j = 1, \cdots, N$. Since the square matrix $\boldsymbol{U}_t$ is real, the singular values s_i's are real and positive. Equation (5) can be rewritten as: $Tr(\boldsymbol{S}\boldsymbol{R}) = \sum_{i=1}^{N} s_i r_{ii}$. Thus, the maximum is obtained when $r_{ii} = 1$ for $i = 1, \cdots, N$, or equivalently, $\boldsymbol{R}$ being an identity matrix. By definition of $\boldsymbol{R} := \boldsymbol{P}\boldsymbol{Q}^*\boldsymbol{U}_a^*$, the best unitary approximation matrix $\boldsymbol{U}_a = \boldsymbol{P}\boldsymbol{Q}^*$. □

Thus, based on the true unitary matrix $\boldsymbol{U}_a$, we parameterize the phases of the MZI array for constructing the unitary block. The last block before the activation function is a diagonal block. The dimension of the diagonal block is the same as the unitary block. The diagonal elements are also encoded in training to optimize the objective.

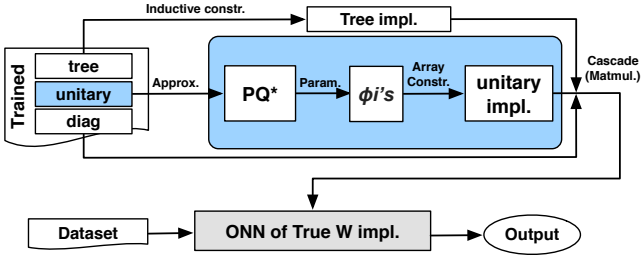

Figure 5: Implementation setup.

The setup of our proposed architecture is summarized in Figure 5. The three network blocks, tree, unitary and diagonal blocks, are determined in the training. Given their trained transfer matrices: the tree block is constructed by the inductively calucating the phase of each subtree components; the trained unitary matrix is first approximated by a true unitary $\boldsymbol{U}_a = \boldsymbol{P}\boldsymbol{Q}^*$ and the phases ϕ_i's are computed by parameterizing $\boldsymbol{U}_a$ which form the MZI array in implementation; the diagonal matrix is directly implementable with MZIs and amplification materials/optical on-chip amplifiers. The three implementations are cascaded together to form the physical realization of the weight matrix for a fully connected layer; the transfer matrix is the multiplication of the three individual matrices. Finally, the calculated transfer matrix is applied to each layer for experimental evaluation. Given an n-input m-output weight matrix,

the area in terms of the number of the MZIs for the three parts is upper bounded by n, $n(n-1)/2$ and n. Compared to the basic architecture [13], the saving comes from the elimination of a unitary matrix, which requires $\mathcal{O}(m^2)$ number of MZIs.

4 EXPERIMENTAL RESULTS

We implemented the proposed architecture in Tensorflow and tested it on a machine with an Intel Core i9-7900X CPU and an NVIDIA TitanXp GPU. The experiments were conducted on the machine learning dataset MNIST [18]. We also implemented the neural network model of different sizes, detailed in Table 1. We assume both the proposed architecture and the previous architecture use the ideal *Relu* activation. Both architectures are trained for 300 epochs, where the original architecture can converge and our architecture can achieve a satisfactory solution balancing the accuracy and the unitary approximation effectiveness.

In Table 1, the first column shows the neural network setup, with different numbers of layers and dimensions considered. The output dimensions for the input and each fully connected layer are listed. For example, (14×14)-100-10 means the input layer has a dimension of (14×14); then follows a fully connected layer whose output dimension is 100, meaning the matrix size of the transfer function is $(14 \times 14) \times 100$; and finally, a fully connected layer whose output dimension is 10, which is the number of classes of the dataset, the matrix size being 100×10. Note that the original dimension for the MNIST image dataset is (28×28). We use max-pooling to downsize the images in order to produce a small size network for a better evaluation covering different neural network expressivity. Generally, a higher-dimensional layer has a higher expressivity, in terms of how many functions or distributions it is able to learn. Nonetheless, a smaller neural network is still preferred due to its lower computational or implementation cost.

In columns 2-4, the test accuracy of the basic architecture [13] and our architecture are listed. For our architecture, both the accuracy directly achieved by the trained unitary $\boldsymbol{U}_t$ and the true implementable unitary $\boldsymbol{U}_a$ are presented. Comparing the test accuracy of the previous architecture and our architecture (using $\boldsymbol{U}_a$), the greatest degradation of all the setup is a negligible 0.0088. The average as computed in the last row is 0.0058. The accuracy difference between using the trained unitary $\boldsymbol{U}_t$ and the true unitary $\boldsymbol{U}_a$ is also small, with the maximum value as 0.0002, which proves the effectiveness of the approximation.

In columns 5-7, the number of MZIs for both architectures and their ratios are computed. As discussed in Section 3, the main saving of our architecture comes from the elimination of an MZI-intensive array for each fully connected layer. For different neural network setup, the saving varies from 15% to 38%. The average saving is 28.7%. Columns 8 and 9 display the effectiveness of approximating the unitary. Column 8 shows the closeness of the trained unitary $\boldsymbol{U}_t$ to the true unitary, by calculating $\|\boldsymbol{U}_t\boldsymbol{U}_t^* - I\|_F$; Column 9 shows the closeness of the unitary approximation $\boldsymbol{U}_a$ to the trained $\boldsymbol{U}_t$, by calculating $\|\boldsymbol{U}_a - \boldsymbol{U}_t\|_F$. These values intend to provide the reference for selecting the number of training epochs for the corresponding network setup. Finally, the time for training our architecture for each epoch is listed in seconds in the last column. Due to the special software implementation and the extra regularization, the training time is generally longer than that of the plain architecture.

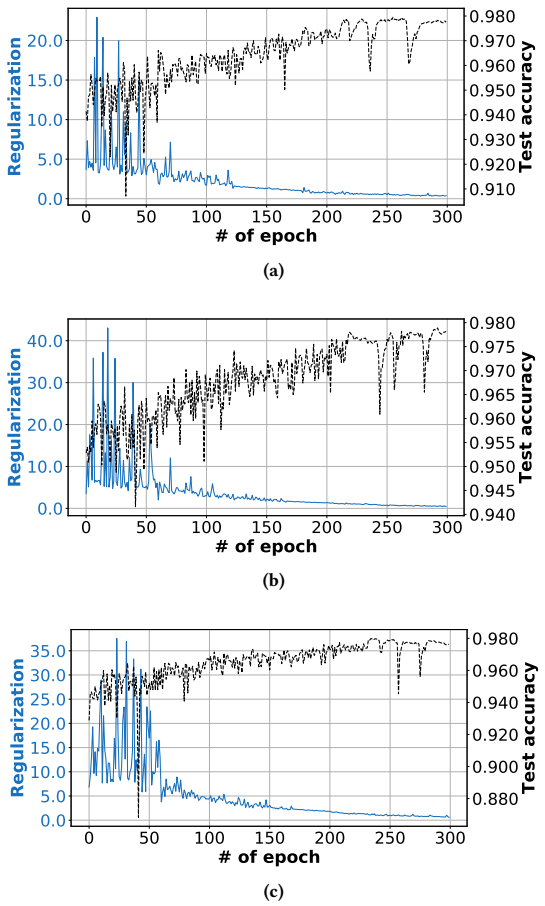

Figure 6: Training curve for ONN setup (a) (28×28)-400-10 (b) (28×28)-600-300-10 (c) (28×28)-600-600-300-10.

Figure 6 plots the three training curves for three ONN settings: (28×28)-400-10, (28×28)-600-300-10 and (28×28)-600-600-300-10. The blue curves (left y-axis) correspond to the regularization defined by Equation (3) and the black dashed curves (right y-axis) correspond to the test accuracy. We can see that at the early stage, the accuracy and regularization curves both oscillate greatly until at the later stage a balance is reached with high test accuracy and relatively low regularization, which suffices to meet the required approximation effectiveness.

The last set of experiments studies the robustness advantage of a slimmed architecture. Similar to other common neuromorphic computing hardwares, the ONN hardwares, especially the phases of the MZIs, are also exposed to the problem of noise, such as manufacturing imperfectness and temperature crosstalk [13]. This set of experiments intends to demonstrate that by decreasing the number of optical components, the neural network robustness can also be improved. As depicted in the box plots of Figure 7, there are three random noise amplitude settings imposed upon the phases of MZI: 0.020, 0.025 and 0.050. Each conforms with a truncated norm distribution. For each noise setting, we generate 20 noisy

Table 1: Experimental Results for Different ONN Setups.

ONN setup	testing accuracy			# of MZIs			approx. results		epoch time
	prev. [13]	ours (U_t)	ours (U_a)	ours	prev. [13]	#ratio	U_t ~unitary	$U_a \sim U_t$	
(14×14)-100-10	0.9744	0.9668	0.9668	24652	29165	84.53%	0.0879	0.0439	7.51
(14×14)-150-10	0.9772	0.9684	0.9684	30977	41665	74.35%	0.1083	0.0542	10.05
(28×28)-400-10	0.9834	0.9772	0.9772	389104	466991	83.32%	0.3800	0.1900	22.33
(14×14)-150-150-10	0.9753	0.9690	0.9690	42302	64165	65.93%	0.1562	0.0781	10.02
(28×28)-400-400-10	0.9827	0.9776	0.9774	469304	626991	74.85%	0.4426	0.2214	24.32
(28×28)-600-300-10	0.9824	0.9782	0.9781	534854	756991	70.66%	0.4524	0.2262	70.53
(14×14)-150-150-150-10	0.9787	0.9756	0.9756	53627	86665	61.88%	0.1864	0.0932	11.49
(28×28)-400-400-200-10	0.9822	0.9777	0.9776	489604	666991	73.40%	0.5575	0.2787	41.45
(28×28)-600-600-300-10	0.9820	0.9763	0.9764	715154	1116991	64.03%	0.5917	0.2959	70.87
average	0.9799	0.9741	0.9741	305509	428513	71.30%	0.3292	0.1646	29.84

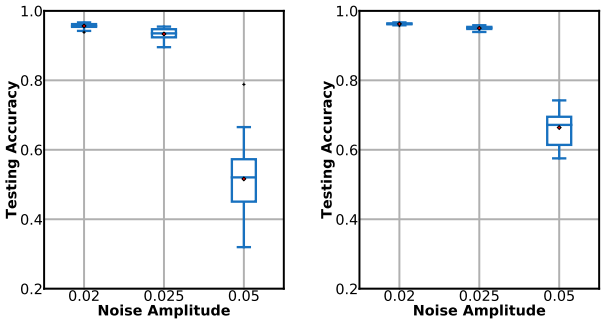

Figure 7: Noise robustness of (a) the previous architecture (b) the proposed architecture with the (14×14)-150-150-10 setup.

samples for both the previous architecture (Figure 7(a)) and the slimmed architecture (Figure 7(b)). Taking (14×14)-150-150-10 as an example, it can be seen that the accuracy distribution of the slimmed architecture not only has higher average and geometric means but also a smaller variation range between the best and worst among all the samples.

5 CONCLUSION

In this work, we study the hardware-software co-design of a slimmed architecture for optical neural networks. The proposed methodology directly considers the structures and constraints of the optical hardware implementation during the software training process. The slimmed architecture contains a sparse tree network block, a single unitary block and a diagonal block for each neural network layer. The new design reduces the number of the MZI-intensive unitary blocks in the previous architecture, leading to a smaller optical hardware implementation. The reduction of the cascaded optical components also brings about better robustness against MZI-related noise.

ACKNOWLEDGEMENT

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] D. A. Miller, "Attojoule optoelectronics for low-energy information processing and communications," *Journal of Lightwave Technology*, 2017.

[2] C. Sun, M. T. Wade, Y. Lee, J. S. Orcutt, L. Alloatti, M. S. Georgas, A. S. Waterman, J. M. Shainline, R. R. Avizienis, S. Lin *et al.*, "Single-chip microprocessor that communicates directly using light," *Nature*, 2015.

[3] C. Condrat, P. Kalla, and S. Blair, "Logic Synthesis for Integrated Optics," in *Proc. GLSVLSI*, 2011.

[4] R. Wille, O. Keszocze, C. Hopfmuller, and R. Drechsler, "Reverse BDD-based Synthesis for Splitter-free Optical Circuits," in *Proc. ASPDAC*, 2015.

[5] Z. Zhao, Z. Wang, Z. Ying, S. Dhar, R. T. Chen, and D. Z. Pan, "Logic synthesis for energy-efficient photonic integrated circuits," in *Proc. ASPDAC*, 2018.

[6] Z. Ying, Z. Wang, Z. Zhao, S. Dhar, D. Z. Pan, R. Soref, and R. T. Chen, "Silicon microdisk-based full adders for optical computing," *Optics letters*, 2018.

[7] D. Ding, B. Yu, and D. Z. Pan, "GLOW: A global router for low-power thermal-reliable interconnect synthesis using photonic wavelength multiplexing," in *Proc. ASPDAC*, 2012.

[8] 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.

[9] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nature communications*, 2013.

[10] J. Bueno, S. Maktoobi, L. Froehly, I. Fischer, M. Jacquot, L. Larger, and D. Brunner, "Reinforcement learning in a large-scale photonic recurrent neural network," *Optica*, 2018.

[11] D. Rosenbluth, K. Kravtsov, M. P. Fok, and P. R. Prucnal, "A high performance photonic pulse processing device," *Optics Express*, 2009.

[12] A. N. Tait, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, "Broadcast and weight: an integrated network for scalable photonic spike processing," *Journal of Lightwave Technology*, 2014.

[13] Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund *et al.*, "Deep learning with coherent nanophotonic circuits," *Nature Photonics*, 2017.

[14] M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, "Experimental realization of any discrete unitary operator," *Physical review letters*, 1994.

[15] A. Ribeiro, A. Ruocco, L. Vanacker, and W. Bogaerts, "Demonstration of a 4 × 4-port universal linear circuit," *Optica*, 2016.

[16] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walmsley, "Optimal design for universal multiport interferometers," *Optica*, 2016.

[17] D. A. Miller, "Perfect optics with imperfect components," *Optica*, 2015.

[18] Y. LeCun, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/, 1998.