This article has been accepted for publication in IEEE Journal of Selected Topics in Quantum Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTQE.2023.3242992

Pruning Coherent Integrated Photonic Neural Networks

Sanmitra Banerjee, *Member, IEEE*, Mahdi Nikdast, *Senior Member, IEEE*, Sudeep Pasricha, *Senior Member, IEEE*, and Krishnendu Chakrabarty, *Fellow, IEEE*

***Abstract*—Coherent integrated photonic neural networks (IPNNs) are increasingly being explored for rapidly growing artificial intelligence applications. However, the principal roadblocks in the scalability of IPNNs are their large area footprint and high tuning power consumption during both training and inferencing. In deep neural networks (DNNs), software techniques to prune redundant weights are often utilized to reduce resource (e.g., memory, computation, and power) overheads. However, due to the complex nature in which the software weights are mapped to the building blocks of IPNNs, prior efforts to apply existing pruning approaches to IPNNs have been ineffective. We present CHAMP and LTPrune, two novel hardware-aware pruning techniques for IPNNs. Using a case study of three IPNNs with different footprints, we show that both these methods can prune more than 99% of the phase angles (which are similar to the weight parameters in DNNs). We also analyze the performance of the pruned IPNNs under phase uncertainties and present a comparative analysis of the two methods to enable advanced hardware-software-assisted design-optimization techniques for IPNNs. To expedite pruning, we also propose HybridPrune, where CHAMP and LTPrune are used in conjunction to obtain similar network sparsity as standalone-LTPrune but with up to 78.3% fewer retraining epochs.**

I. Introduction

Deep neural networks (DNNs) have seen remarkable advances and are being widely deployed for speech and action recognition, image classification, and natural-language processing. The primary computational primitive while querying such advanced DNNs is the time- and energy-intensive matrix multiplication operation. Leveraging the inherently parallel nature and high-speed of optical-domain computation, coherent integrated photonic neural networks (IPNNs), which operate with a single wavelength, can reduce the computational complexity of matrix multiplication from $O(N^2)$ to $O(1)$ [1]. Using singular value decomposition (SVD), several IPNNs have been recently proposed [2].

The weight matrix corresponding to a linear multiplier in the fully-connected layer of a multi-layer perceptron can be factorized into two unitary matrices and one diagonal matrix using singular value decomposition. Several approaches for representing the unitary matrices using an array of Mach-Zehnder interferometers (MZIs) have been proposed in prior work [3] [4]. Similarly, a diagonal matrix can be realized using a single layer of MZIs. Essentially, MZIs are the building blocks of IPNNs, and each MZI consists of two phase shifters (with phase angles ϕ and θ in Fig. 1(d)) and two 50:50 beam-splitters. The phase angles in an array of MZIs can be tuned to realize different unitary transformations; during backpropagation in IPNN training, the phase angles are iteratively updated using an optimizer (e.g., stochastic gradient descent) to minimize the loss function.

While IPNNs can perform high-speed matrix multiplication, they suffer from high static power consumption in the thermo-optic PhS in the MZIs. The power consumption in such PhS is directly proportional to the tuned phase angle [5]. In fact, even in power-efficient PhS, up to 25 mW tuning power can be dissipated for a phase shift of π [6]. Additionally, PhS with high phase angles are more susceptible to inevitable uncertainties phase uncertainties due to fabrication process variations and thermal crosstalk. In prior work [7], we have shown that such uncertainties can result in up to a 70% reduction in the inferencing accuracy. In addition to the high tuning power consumption, the large area footprint of MZIs is a crucial roadblock in the scalability of IPNNs. High-performance MZIs proposed in recent work (e.g., [8]) have a footprint of 300 μm. Such footprint is exceedingly high compared to electronic circuits, where sub-5 nm transistors are now commonly used. Clearly, an IPNN will have a higher area footprint compared to an electronic DNN that is trained on the same workload (and, therefore, has similar architectural complexity).

A potential solution to both the aforementioned problems can be to prune redundant PhS from the MZI arrays in the IPNN. Such PhS can either be removed from the array or power-gated to reduce the tuning power consumption. Additionally, PhS account for a significant portion (up to 90%) of the area footprint in an MZI [8]). Pruning PhS can thereby considerably reduce the area footprint of IPNNs, particularly for edge devices where the IPNN application workload is known and remains fixed. Moreover, a sparse IPNN (where most of the PhS have a zero phase angle) can be defined by fewer parameters (e.g., tuned phase angles). Therefore, incremental retraining of such pruned IPNNs has a lower memory overhead. This is particularly crucial for edge applications (e.g., automotive ICs), where the phase angles may need to be updated in-field when the application workload changes. Prior attempts at pruning IPNNs have primarily focused on a software-based approach. In such methods, a DNN is first trained in software and pruned using magnitude-based techniques (e.g., [9]). The DNN weights in the sparse network are then mapped to tuned phase angles in the IPNN. However, as we show in Section II.D, due to the complex

S. Banerjee is with NVIDIA Corporation, Santa Clara, CA. He worked on this paper as a Ph.D. student at Duke University. His work at NVIDIA is unrelated to the contents of this paper.

M. Nikdast and S. Pasricha are with the Department of Electrical and Computer Engineering at Colorado State University, Fort Collins, CO.

K. Chakrabarty is with the Department of Electrical and Computer Engineering at Duke University, Durham, NC.

This work was supported in part by the National Science Foundation (NSF) under grant numbers CCF-1813370, CCF-2006788, and CNS-2246226.

This article has been accepted for publication in IEEE Journal of Selected Topics in Quantum Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTQE.2023.3242992

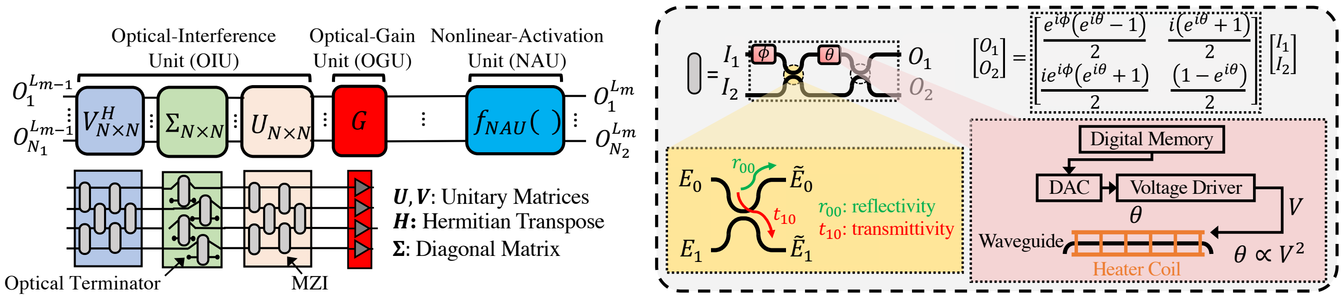


Fig. 1: Schematic of a linear layer in MLP-based coherent IPNNs. Right: detailed view of an MZI.

bidirectional many-to-one mapping between the DNN weights and the IPNN phase angles, a sparse weight matrix may not always lead to sparse phase angles. Moreover, simply pruning the phase angles post-training leads to a significant accuracy loss due to the unretrainability of the IPNN [10].

To enable efficient pruning in IPNNs, we propose two novel hardware-aware pruning techniques. The first method, CHAMP, uses *photonic-training*, where we perform backpropagation on the phase angles rather than the edge weights of the fully connected layers; consequently, we have increased control over the phase angles and obtain higher sparsity with lower accuracy loss. The second method, LTPrune, builds upon CHAMP and leverages the lottery ticket hypothesis to identify sparse subnetworks in IPNNs. Using two example IPNNs of different footprints, we show that CHAMP and LTPrune can both prune more than 99% of the phase angles (89% in the smaller IPNN) with an accuracy loss of less than 5%. We also propose HybridPrune for high-speed in-field IPNN pruning – this method can achieve a similar sparsity as LTPrune with up to 78.3% lower run-time necessary to retrain the pruned network.

The main contributions of this paper are as follows:

- Highlighting the challenges associated with conventional hardware-unaware software DNN pruning when applied to IPNNs;
- CHAMP, a hardware-aware magnitude pruning technique for coherent IPNNs;
- LTPrune, a hardware-aware pruning technique based on the lottery ticket hypothesis to obtain power- and energy-efficient IPNNs;
- Comparing the sparsity of the pruned models obtained using CHAMP and LTPrune on IPNNs with different footprints;
- Analyzing the trade-off between the sparsity of the PhS in a pruned IPNN and its sensitivity to random uncertainties in phase angles when the pruned PhS are power-gated or removed;
- HybridPrune, an in-field pruning technique to obtain sparse general-purpose IPNNs with few retraining epochs.

The rest of the paper is organized as follows. Section II presents the fundamentals of IPNNs and motivates the need for pruning IPNNs to improve the power-efficiency and reduce the area overhead. Furthermore, we highlight the challenges in pruning IPNNs using existing hardware-unaware DNN pruning approaches. In Section III, we propose CHAMP and LTPrune, two novel hardware-aware IPNN pruning techniques. We present simulation results using three example IPNNs in Section IV. In Section V, we present a hybrid pruning approach that achieves a high PhS sparsity with few retraining epochs by combining CHAMP and LTPrune. We draw conclusions in Section VI.

II. Background and Related Prior Work

A. Mach-Zehnder Interferometers (MZIs)

MZIs are used to determine the relative phase difference between collimated optical signals [4], [11]. Fig. 1 shows the MZI structure considered in our work. Each MZI consists of two tunable PhS (ϕ and θ in Fig. 1) – these are used to apply configurable phase shifts and obtain varying degrees of interference between the input optical signals. They can be implemented using thermal microheaters [12], where the refractive index of the underlying waveguide changes with temperature (i.e., thermo-optic effect), altering the phase of the optical signal traversing the waveguide. Each phase shifter in the MZI is followed by a passive 50:50 2×2 beam splitter (BeS). Each BeS can be designed using directional couplers, where a fraction (defined by transmittance) of the optical signal at an input port is transmitted to an output port, and the remaining (defined by the reflectance) is coupled to the other output port with a phase shift of $\frac{\pi}{2}$. For symmetric 50:50 BeS, both transmittance and reflectance coefficients are $\frac{1}{\sqrt{2}}$. As a result, the transfer matrix for an MZI with two PhS and two 50:50 BeS (see Fig. 1) can be modeled as [11]:

$$T_{MZI}(\theta,\phi) = U_{BeS}\cdot U_{PhS}(\theta)\cdot U_{BeS}\cdot U_{PhS}(\phi) = \begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22}\end{pmatrix} = \begin{pmatrix} \frac{e^{i\phi}}{2}(e^{i\theta}-1) & \frac{i}{2}(e^{i\theta}+1) \\ \frac{ie^{i\phi}}{2}(e^{i\theta}+1) & -\frac{1}{2}(e^{i\theta}-1)\end{pmatrix}, \quad (1)$$

where U_{BeS} (U_{PhS}) is the BeS (PhS) transfer matrix.

B. MZI-based Coherent IPNNs

A multi-layer perceptron (MLP)-based DNN consists of several consecutive layers of interconnected neurons. Post feature-extraction, the input features $(X_1,\ldots,X_{N_1})$ are fed into a series of fully connected layers, followed by a final LogSoftMax activation layer to obtain the probability of each output class $(Y_1,\ldots,Y_{N_2})$. Each connection between the neurons is assigned a weight that represents its synaptic plasticity and each neuron is tasked with a multiply-and-accumulate (MAC) operation followed by passing the resultant output through a non-linear activation function (f_{NAU}). By introducing non-linearity in the network, the activation functions (e.g., sigmoid, $\tanh$, and Rectified Linear Unit) enable the DNNs to learn complex non-linear relationships [2]. During each training iteration, the weight of each connection in a DNN is incrementally updated to minimize the loss function that quantifies the difference between the expected and the obtained DNN output.

Consider an $N_2 \times N_1$ weight matrix L_m representing the edge weights connecting a layer with N_1 neurons with a layer with N_2 neurons. Using singular value decomposition (SVD) and considering Fig. 1, we have $L_m = U\Sigma V^H$, where U and V are unitary matrices with dimensions $N_2 \times N_2$ and $N_1 \times N_1$, respectively. Moreover, V^H denotes the Hermitian transpose of V_m, and Σ is a diagonal matrix consisting of the eigenvalues of L_m.

Reck *et al* [3] first demonstrated that any unitary transformation between optical channels can be realized using a triangular mesh of MZIs. However, Clements *et al* proposed an alternative arrangement of MZIs (see Fig. 1) to implement unitary transformations with half the physical footprint of the Reck design and a lower optical loss [4]. Therefore, for a given weight matrix $W_m = U_m \Sigma_m V_m^H$, this paper assumes the Clements design to represent the unitary matrices U_m and V_m^H. The diagonal matrix Σ_m can be realized using an array of MZIs to attenuate each channel separately without mixing by terminating one input and one output of each MZI (Σ in Fig. 1). As MZIs can only attenuate optical signals, a global optical amplification is necessary on each output to represent arbitrary diagonal matrices [13]. This scaling factor is realized using the optical gain unit (OGU) G (see Fig. 1) that includes semiconductor optical amplifiers [14].

C. Motivation for Pruning IPNNs

With the increasing applications of DNNs to complex problems, software-based magnitude pruning approaches are being increasingly explored to minimize the resources necessary for the storage and training of large-scale DNNs. In particular, such approaches are crucial for edge applications, where the network parameters (weights in the case of DNNs) may need to be updated in-field when the application workload changes.

The phase shifters in IPNNs with MZI arrays consume a significant portion of the network area and power. In particular, the size of the constituent thermo-optic PhS in an MZI determines the size of the device (see Fig. 1). For example, the state-of-the-art 2×2 MZI proposed in [8] is ≈ 300 μm long, in which each PS has a length of 135 μm (i.e., $\approx 90\%$ of the length of the MZI considering the two PhS). Moreover, as discussed in Section II-A, the required phase shift in a PS ($\Delta\phi$) is directly proportional to its length (L) and tuning power consumption (P): $\Delta\phi \propto L{\cdot}P$. Even power-efficient PhS can consume up to ≈ 25 mW tuning power for a phase shift of π [6]. As a result, low accuracy-loss pruning approaches are essential in IPNNs to identify prunable PhS, thereby reducing the footprint of the network and the tuning power consumption during inferencing. Note that the phase angles

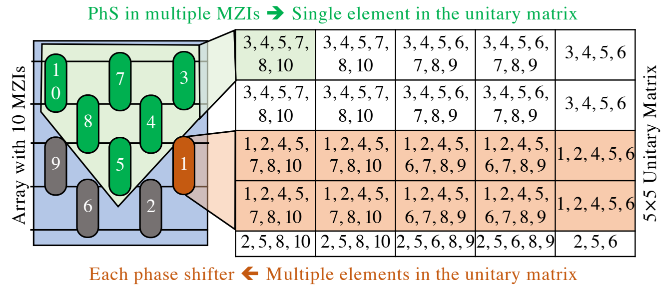


Fig. 2: An example of the bidirectional many-to-one association (BMA) between the elements of the weight matrix and the MZI array for a 5×5 unitary matrix. The numbers in each cell of the unitary matrix denote the MZIs that affect the corresponding matrix element.

are adjusted dynamically during software training. However, we only consider the static tuning power dissipated in the PhS during inferencing to maintain the trained phase angles in the hardware platform. Additionally, as lower phase shifts require lower ΔT ($\Delta\phi \propto \Delta T$), mutual thermal crosstalk between PhS can be minimized by pruning. The problem of explicitly reducing thermal crosstalk is beyond the scope of this paper.

Note that recent work has shown that, as an alternative to pruning the redundant PhS, high-radix matrix multiplications can be mapped to multiple cascaded small-radix photonic tensor cores [15]. Simulation results have shown that using such tensorized decomposition, the number of MZIs in an IPNN can be reduced by up to 79×. While this approach can lead to a considerable reduction in the footprint, such networks cannot be easily reconfigured to perform a different matrix multiplication (e.g., where the dimensions of the synaptic interconnections change). However, the redundant components can be power-gated (and not removed) during pruning; consequently, the MZI arrays can be easily reconfigured when the application workload changes.

D. Challenges in Pruning IPNNs

Hardware-unaware software pruning methods in DNNs aim at obtaining a sparse weight matrix [9]. A binary mask M^k is maintained for each DNN layer L^k. An element of the mask, say $M^k_{i,j}$, is 0 (1) iff the corresponding weight $L^k_{i,j}$ is zero (non-zero). In each pruning iteration, a fraction of the non-zero weights—typically those with a smaller magnitude—in each layer is clamped to zero, and the corresponding mask elements are updated. During backpropagation in retraining, the gradient of each weight is multiplied with its respective mask element, ensuring that the zero weights are not updated.

Unfortunately, conventional software pruning approaches fail to achieve high sparsity in IPNNs. This can be attributed to the *bidirectional many-to-one association (BMA)* between the elements of the weight matrix of the linear layers in the IPNN and the phase angles in the MZI arrays. Each weight matrix element is mapped to multiple PhS and, conversely, each PhS in an MZI array affects multiple weight matrix elements [16] [17]. Fig. 2 shows an example of this BMA for a 5×5 unitary matrix and its corresponding mapped MZI array. Observe also that the bidirectional dependence is largely asymmetric in nature—certain MZIs (e.g., MZI# 8) affect several matrix

This article has been accepted for publication in IEEE Journal of Selected Topics in Quantum Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTQE.2023.3242992

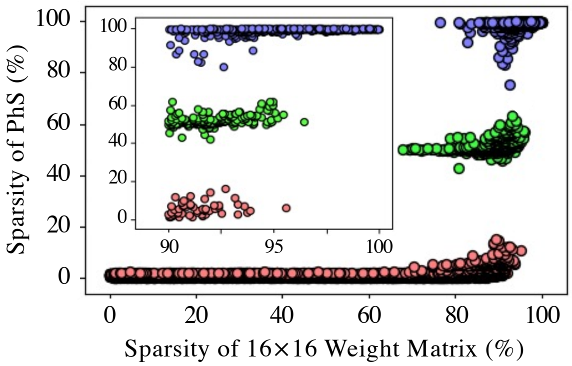


Fig. 3: Scatter plot comparing the sparsity of 10000 randomly generated 16×16 unitary matrices and the sparsity of their corresponding mapped MZI arrays. The inset shows a similar comparison for highly sparse (>90% sparsity) unitary matrices. The red, green, and blue clusters denote the unitary matrices with low, medium, and high PhS sparsity.

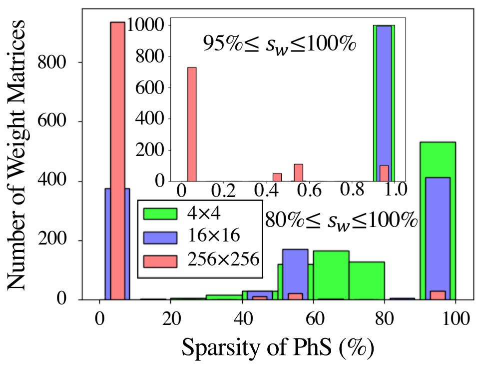


Fig. 4: Histogram comparing the sparsity of the PhS in the mapped MZI arrays for 3000 randomly generated weight matrices of different dimensions (1000 of each dimension) with sparsity s_w, where 80%≤ s_w ≤100%. The inset shows a similar plot for 95%≤ s_w ≤100%.

elements (20 in this case), whereas others (e.g., MZI# 1) affect fewer elements (10 in this case). Similarly, some weight matrix elements depend on fewer phase angles than others. As a result of the asymmetric BMA, a sparse weight matrix (obtained using software pruning approaches) may not always be mapped to a sparse MZI array (where many PhS have zero tuned phase angle) and vice versa. In Fig. 3, we compare the weight matrix sparsity of 10000 randomly generated 16×16 unitary matrices with the phase sparsity of their corresponding mapped MZI arrays. The inset shows a magnified view of the unitary matrices with matrix sparsity >90%. Observe that for many sparse weight matrices, the corresponding MZI arrays have sparsity below 20%.In fact, in some cases, the MZI array sparsity can be as low as 0%. Indeed, this discrepancy between the weight matrix sparsity and the phase sparsity is observed for unitary matrices of all dimensions, as we show in Fig. 4. We also observe that this discrepancy is more pronounced for larger unitary matrices (compare the red and green bars in Fig. 4). Therefore, it is expected that the effectiveness of software pruning will reduce as the size and complexity of IPNNs scale. In order to minimize the tuning power and area overhead of IPNNs, pruning approaches should be hardware-aware and should directly target the tuned phase angles.

In [10], the authors showed that no more than 30% of the phase angles can be pruned post-training without a significant accuracy loss (≈10%) in IPNNs. To address this, [10] proposed a pruning-friendly *non-SVD-based* C-IPNN architecture that leverages block-circulant matrix representation and performs matrix-vector multiplication using optical fast Fourier transform (FFT). However, even this alternative architecture could achieve a sparsity of only up to 45%. Unlike in IPNNs, hardware-unaware software pruning is applicable to noncoherent IPNNs. In [18], pruned noncoherent IPNNs demonstrate a classification accuracy of up to 93.49% on the MNIST dataset. Using layer-wise pruning and weight clustering, the noncoherent photonic accelerator proposed in [19] obtains a sparsity of up to 50%.

III. Hardware-Aware Pruning Approaches for IPNNs

As discussed above, efficient pruning approaches for IPNNs, that can minimize the associated tuning power and area overhead must be photonic hardware-aware and should target the tuned phase angles (not the software weights). However, to minimize the accuracy loss while pruning, we should ideally prune only the benign PhS—these are the phase angles that affect the weights with lower saliency. Prior work has shown that, in conventional DNNs, the saliency of a weight and its magnitude are correlated [9]. Therefore, magnitude-based approaches, where the low-magnitude weights in each layer are pruned, are able to achieve high sparsity with a low accuracy loss in electronic DNNs. However, due to the non-linear dependence between the phase angles and weights in IPNNs, weights with large magnitude (and hence, high saliency) can be mapped to smaller phase angles. As a result, pruning smaller phase angles based on their magnitude can lead to degraded accuracy. Recall also that, due to the BMA between the phase angles and the weight matrix elements, it is unlikely that a single phase shifter will exclusively affect low-saliency weights. However, our simulation results show that magnitude-pruning, if applied in a hardware-aware fashion, where we target the phase angles themselves, can still lead to a high sparsity, especially in easy-to-prune over-parameterized IPNNs. We propose this hardware-aware magnitude pruning as a baseline approach, CHAMP. Next, we present an improved pruning technique, LTPrune, where we use the lottery ticket hypothesis to prune a significant fraction of phase angles with negligible accuracy loss. LTPrune is particularly efficient for compact and difficult-to-prune IPNNs.

A. *CHAMP: Coherent Hardware-Aware Magnitude Pruning of IPNNs*

In conventional hardware-unaware pruning techniques, the weights in each layer are sorted based on their magnitude and a fraction of the weights with small magnitude are pruned. To recover the lost inferencing accuracy due to pruning, the network is then retrained while clamping the pruned phase angles to zero. During this step, the phase sparsity is maintained by zeroing out the gradients corresponding to the pruned phase angles. However, based on our discussion in

This article has been accepted for publication in IEEE Journal of Selected Topics in Quantum Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTQE.2023.3242992

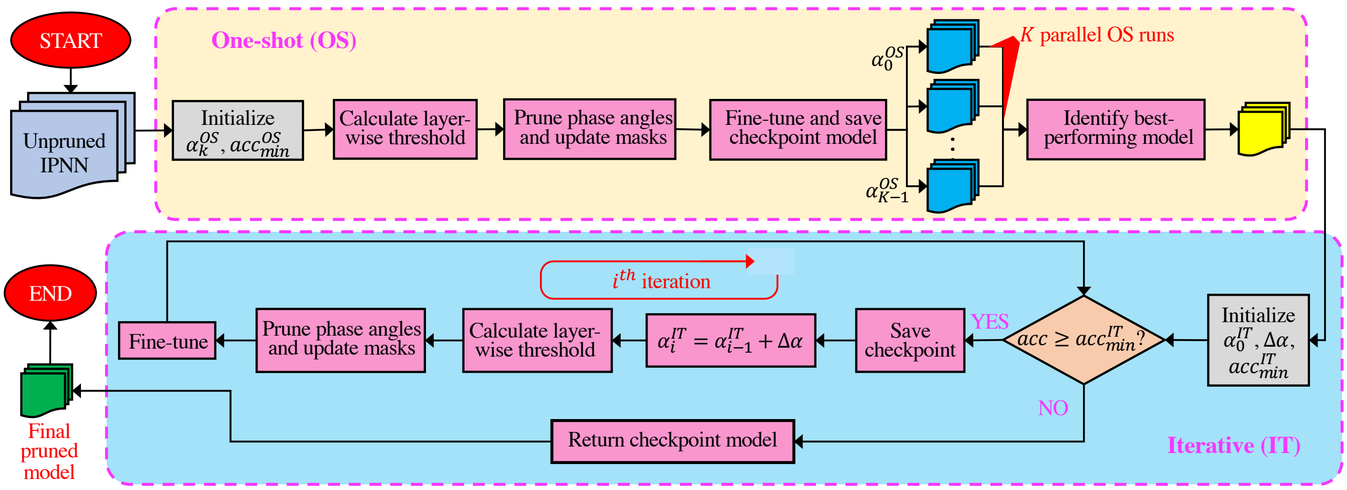


Fig. 5: An overview of the proposed CHAMP method.

Section II.D, the software approach may not necessarily lead to sparse phase angles. Given that the advantages associated with pruning (e.g., lower tuning power and area overhead) are dependent on the sparsity of the phase angles (and not that of the weight matrix elements), hardware-unaware techniques are largely inefficient [10].

The main difference between existing hardware-unaware magnitude pruning techniques and CHAMP lies in the training approach. As discussed earlier, conventionally to obtain the tuned phase angles in an IPNN, a DNN is first trained in software and the trained weights are then mapped to the phase angles using singular value decomposition-based approaches (e.g., [4]). Observe that, in this training flow, we only have control over the weight matrix elements and not the phase angles. This is because the phase angles are deterministically obtained from the trained weight matrices. Consequently, this flow is unlikely to obtain sparse PhS. On the other hand, in CHAMP, we propose a *photonic* training approach. In each training epoch, we calculate the gradients of the loss with respect to the phase angles themselves and iteratively tune them based on this gradient and the learning rate. This gives us increased control over the phase angles and allows us to obtain sparse PhS (instead of sparse weight matrices). We apply CHAMP on an IPNN trained using this photonic approach – in each pruning round, the phase angles below a threshold are pruned while the remaining phase angles are retrained to recover the inferencing accuracy. This threshold can be determined in different ways; however, a common approach is to consider a fraction, say α of the standard deviation of the phase angles in the layer. This takes into consideration the distribution of the phase angles, and we have used this approach for thresholding for our simulation results. We have considered two different variants of CHAMP – in the one-shot (OS) approach, all the phase angles below a threshold (corresponding to some α) are pruned at once after which retraining (a.k.a. fine-tuning) is performed. Alternatively, in the iterative (IT) variant, we perform pruning over several steps by gradually increasing α. Each pruning step is followed by a round of retraining to recover the inferencing accuracy.

OS CHAMP is typically faster than the IT variant as it involves a single pruning step and fewer retraining epochs. However, due to the lack of extensive retraining, the maximum sparsity that can be achieved without a significant accuracy loss is also lower than that of IT CHAMP. Therefore, we propose a hybrid approach where OS CHAMP is first used to quickly ramp up the sparsity. This is followed by IT CHAMP, where the sparsity is gradually increased further, with intermittent retraining. Fig. 5 presents a flowchart of the CHAMP approach. In addition to generating the input model for the IT variant, OS CHAMP also provides a starting point for α for the subsequent iterative flow. The inputs to the OS flow include the trained IPNN, the minimum acceptable inferencing accuracy acc_{min}^{OS}, and K different values of α's (α_k^{OS}, $k = 0, 1, \ldots, K-1$). Note that the K different OS runs are mutually independent and can, therefore, be launched in parallel. Also, we consider the same initialization and the same model architecture for the K runs. After the K OS CHAMP-pruned models are obtained, we identify the "best-performing" model – this is the one that has the maximum phase sparsity while having an inferencing accuracy greater than acc_{min}^{OS}. The inputs to the IT CHAMP flow include this best-performing model along with the initial α (α_0^{IT}), the increment in α in each iteration ($\Delta\alpha$), and the minimum acceptable inferencing accuracy (acc_{min}^{IT}). Note that, for our simulations, we have considered equal increments in the α in each iteration: $\alpha_i^{IT} = \alpha_{i-1}^{IT} + \Delta\alpha$, where i denotes the iteration number. However, we can also use non-uniform step sizes, especially if we find that the sparsity or inferencing accuracy requirements are not satisfied. For example, if the accuracy remains low even after retraining, we may reduce the rate by which α is incremented.

B. LTPrune: Pruning IPNNs using LTH

Efforts on improving the efficiency of neural networks have largely focused on the inferencing step. This is because training such networks is a one-time operation and the associated costs are amortized over high-volume production. However, with the advent of general-purpose accelerators, which are often incrementally retrained in-field, there has been considerable interest in minimizing the training time. It has been shown that training a magnitude-pruned model from

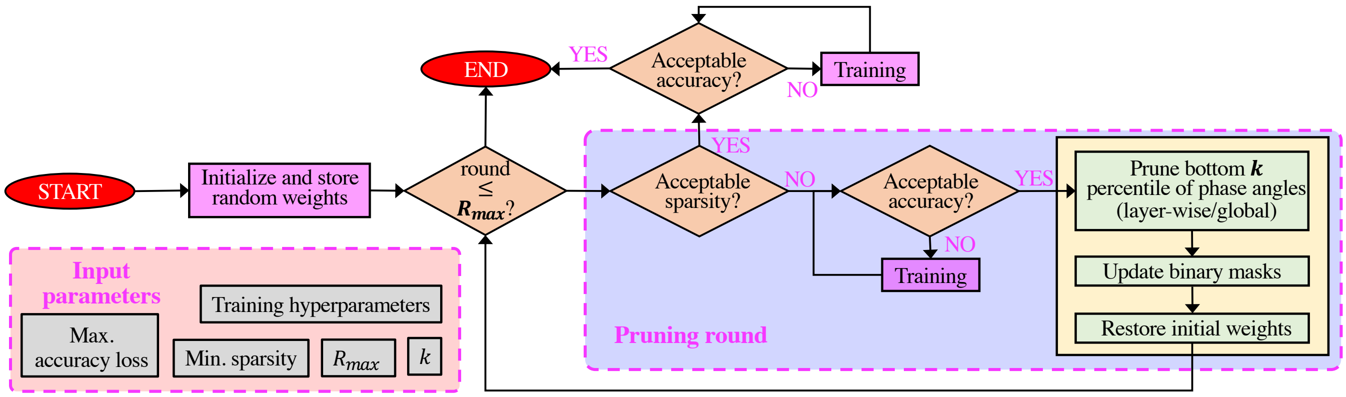


Fig. 6: A flowchart of the proposed LTPrune method.

scratch is considerably difficult and achieves lower accuracy compared to the unpruned model [9].

The lottery ticket hypothesis (LTH) postulates that for any neural network, there exists a sparse subnetwork that, when trained from scratch, can achieve similar accuracy as that of the unpruned network in a significantly lower number of training epochs [20]. For any given network, such easy-to-train sub-networks (a.k.a. *winning tickets*) can be obtained using a modified magnitude pruning approach. In each pruning iteration, after the weights with a magnitude below a threshold are clamped to zero, the remaining weights are reset back to their initial values (before the onset of training). Following this, the network is retrained to recover the inferencing accuracy while ensuring that the pruned weights remain zero. In LTH-based pruning, the obtained sparse sub-network depends on the architecture of the original network, the dataset on which the network is trained, and the initial values of the weights.

In this paper, we propose LTPrune, a novel technique where LTH-based pruning is extended to IPNNs. Note that, LTPrune is different from the conventional LTH-based pruning proposed in [20]. Given the BMA between the weights and the phase angles in IPNNs, LTPrune is hardware-aware and targets the phase angles and not the weights by leveraging photonic training (similar to CHAMP). Fig. 6 presents a flowchart for LTPrune. Given the network architecture (number of layers, neuron count in each layer, etc.), we initialize the phase angles and store their values in a database. In each of the R_{max} pruning rounds, we check whether the IPNN has acceptable sparsity and accuracy. To increase the sparsity, we first retrain the IPNN to obtain a sufficiently high inferencing accuracy. In our simulations, retraining is performed till we obtain an inferencing accuracy within 5% of the nominal accuracy. Post retraining, a fraction of phase angles are pruned and the remaining are reset to their initial values. We consider two ways to identify the phase angles to be pruned: in layer-wise LTPrune, the bottom k-percentile of the phase angles (with the lowest magnitudes) in each layer are pruned, whereas, in global LTPrune, the bottom k-percentile of the phase angles in the entire IPNN are pruned. Note that, while global LTPrune can lead to different sparsity in each IPNN layer, the layer-wise approach ensures uniform sparsity levels across all layers. In Section IV.A, we will show that this uniform distribution of the pruned PhS across all the layers allows us to obtain

TABLE I: Description of the IPNNs considered in our simulation results. FC(x,y): Fully connected layer with x inputs and y outputs, SP: Softplus activation, LSM: LogSoftMax activation.

Model	Architecture
Network-1	FC(16,16)-SP-FC(16,16)-SP-FC(16,10)-LSM
Network-2	FC(64,256)-SP-FC(256,100)-SP-FC(100,10)-LSM
Network-3	FC(64,256)-SP-FC(256,256)-SP-FC(256,128)-SP-FC(128,128)-SP-FC(128,100)-SP-FC(100,10)-LSM

a higher overall sparsity level. On the other hand, in global LTPrune, a majority of the pruned PhS are from a few layers, and this leads to a lower overall sparsity (compared to layer-wise LTPrune) for the same accuracy loss. Once the phase angles are pruned, the binary masks associated with the phase angles are updated – the mask elements corresponding to the pruned (unpruned) phase angles are updated to 0 (1). Consequently, during retraining in the next round, when the gradients computed from backpropagation are multiplied with these binary masks, the pruned phase angles are not updated. This iterative process continues for R_{max} rounds or till we reach the target sparsity.

Recall that, due to the BMA between the phase angles and the weights of the linear layers in the IPNNs, pruning a single phase angle can affect several weight matrix elements, some of which may have high saliency. Therefore, post pruning, we often observe a significant loss in the inferencing accuracy. The hyperparameters for retraining, which are inputs to the LTPrune flow, may need to be adjusted in each round to ensure that acceptable accuracy is achieved. In the next section, we will demonstrate that LTPrune can efficiently identify highly sparse sub-networks in IPNNs and can outperform CHAMP, especially for difficult-to-prune compact IPNNs.

IV. Simulation Results

We consider three fully-connected feedforward IPNNs with different footprints (see Table I) to demonstrate the performance of CHAMP and LTPrune. Network-1 and Network-2 are trained on the MNIST dataset while Network-3 is trained on the CIFAR-10 dataset. Network-1 has a smaller footprint with an input layer with 16 neurons, two hidden layers consisting of 16 neurons each, and an output layer with

This article has been accepted for publication in IEEE Journal of Selected Topics in Quantum Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTQE.2023.3242992

10 neurons (corresponding to the 10 classes in the MNIST dataset). Note that, in addition to the MZIs in the OIUs, N standalone MZIs are necessary before the $V^H_{N\times N}$ and after the $U_{N\times N}$ multipliers to realize the weights in an $N \times N$ fully-connected layer. The inferencing accuracy drops significantly when the PhS in these MZIs are pruned; therefore, we do not consider them as prunable PhS in our simulations. In total, Network-1 has 1380 prunable PhS. Each 28×28=784 dimensional MNIST image is compressed to 16-dimensional compressed feature vectors by considering the 4×4 region at the center of its shifted fast Fourier transform. For the larger Network-2 (with 155,214 prunable PhS), we use a 64-dimensional complex feature vector by considering the 8×8 region at the center of the frequency spectrum. Consequently, Network-2 consists of a 64-neuron input layer, followed by two hidden layers with 256 and 100 neurons, and an output layer with 10 neurons. We convert the images in the CIFAR-10 dataset to gray-scale and compress them to a 64-dimensional feature vector using shifted fast Fourier transform. Network-3 (with 351,822 prunable PhS) consists of a 64-neuron input layer, followed by five hidden layers with 256, 256, 128, 128, and 100 neurons, and an output layer with 10 neurons.

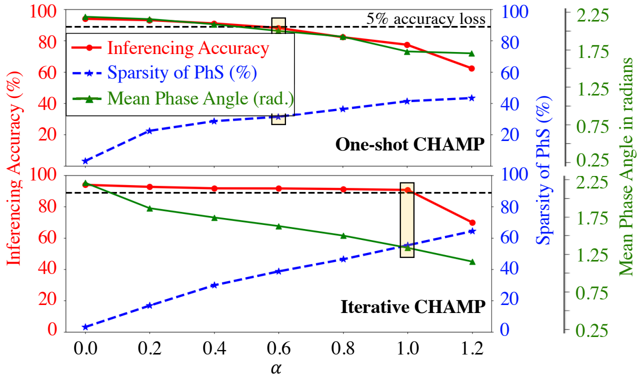
A. Sparsity of Pruned IPNNs

Fig. 7(a)-(c) shows the simulation results when the one-shot and iterative CHAMP methods are applied to Network-1, Network-2, and Network-3. In each case, the pruning threshold, which is the magnitude below which all phase angles are pruned, is given by $\alpha \cdot \sigma_{layer}$. Here, α is a user-defined constant, and σ_{layer} denotes the standard deviation of the non-zero phase angles in each layer. We consider both the one-shot and the iterative pruning approaches – this is because, for the smaller Network-1, the one-shot approach performs better than the iterative approach for some cases (e.g., for $\alpha = 0.2$ in Fig. 7(a)). For the larger Network-2, while iterative CHAMP outperforms the one-shot approach in terms of sparsity, we use the one-shot approach to ramp up the initial values of α quickly. The best-performing one-shot model (yellow rectangle corresponding to $\alpha = 2.5$ in Fig. 7(b)) is used as an input model for the iterative CHAMP. Similarly, for Network-3, the best-performing one-shot model ($\alpha = 3.0$ in Fig. 7(c)) has a sparsity of 77.3% and is used as the input model for the iterative flow.

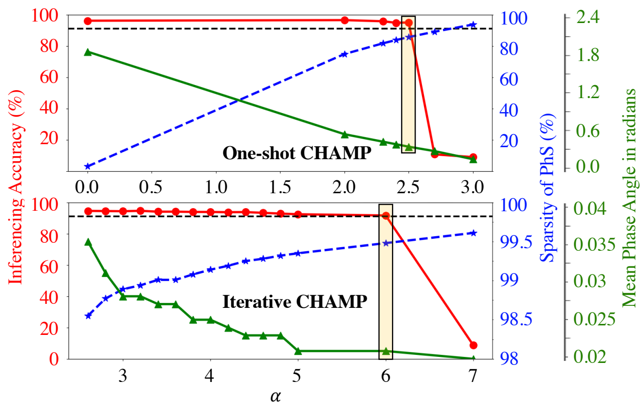
For all these cases, the pruning threshold in each layer increases with increasing α; consequently, the sparsity of the PhS increases, and the mean phase angle—averaged over the 1380 PhS in Network-1, the 155,214 PhS in Network-2, and the 351,822 PhS in Network-3, to which the weight parameters are mapped— decreases. Note, however, that in the one-shot case, the accuracy drops steeply with increasing α. In fact, for an allowable accuracy loss of 5%, one-shot CHAMP leads to a maximum sparsity of 31.1% in Network-1, 85.7% in Network-2, and 77.3% in Network-3. On the other hand, with iterative CHAMP, the accuracy loss is less than 5% up to α =1 (for Network-1), α =6 (for Network-2), and α =5.6 for Network-3. Consequently, for a 5% accuracy loss, 55% PhS in Network-1, 99.48% PhS in Network-2, and 91.3% PhS in Network-3 can be pruned. We obtain a significantly higher sparsity using iterative CHAMP as the models are pruned and retrained over several epochs gradually, compared to the drastic pruning and few fine-tuning epochs in the one-shot case. Note also that using the same approach and with the same allowable accuracy loss, the sparsity obtained in Network-2 is higher than that in Network-1. This can be attributed to the fact that while both the networks are trained for the same task, Network-2 has a significantly higher number of PhS, and is therefore highly over-parameterized. However, despite the larger PhS count in Network-3, the maximum sparsity achieved is lower than that of Network-2. This is indeed expected, as the CIFAR-10 task is more complex than MNIST, and, hence, necessitates more parameters for learning. We also observe that while, in most cases, the mean phase angle decreases with increasing PhS sparsity, the reverse occurs from $\alpha = 1.0$ to $\alpha = 1.2$ in Fig. 7(a). This is due to the fine-tuning step where some of the few remaining non-zero phase angles may increase in magnitude to account for the additional pruning. However, our simulations show that such scenarios are, indeed, quite rare and even when they occur, the increase in the mean phase angle is negligible.

Fig. 8(a)-(c) show the fine-tuned inferencing accuracy, sparsity of the PhS, mean phase angle, when layer-wise and global LTPrune are applied to the three networks. The performance of LTPrune depends on how the pruning rate, k, is scheduled over the different pruning rounds. For the smaller Network-1, we found that a high PhS sparsity at a low accuracy loss is achieved when we start with a low k for the first few rounds, followed by aggressive pruning (high k) in the final few rounds. The top and bottom subfigures in Fig. 8(a)-(c) show the simulation results when we apply layer-wise LTPrune and global LTPrune, respectively. For Network-1, we use k =10% for the first ten rounds and k =25% for the remaining rounds. However, for Network-2, we obtained the best pruning performance by using a moderate k =25% for the first five rounds, followed by a high k =50% in the next four rounds, and then a low k =10% for the remaining rounds. Clearly, the optimal schedule of the pruning rate k can vary based on the network architecture and the dataset. For Network-3, we use k =30% for the first three rounds, k =25% for the next five rounds, followed by k =10% for the remaining rounds. A trial-and-error-based approach is, therefore, necessary to find the optimal k schedule for a given IPNN.

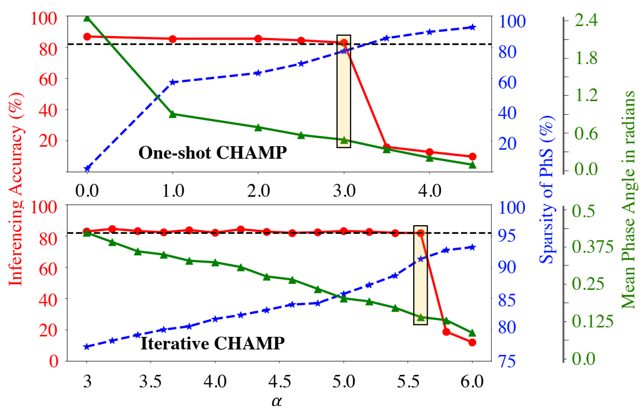
We observe that for both layer-wise and global LTPrune (across all networks), the inferencing accuracy is only ≈10% in the first round. This is because the accuracy in each round is recorded before the training in that round (see Fig. 6). After the training in the first round, the accuracy improves in all four cases. As pruning progresses, the mean phase angle decreases, and the sparsity of the phase angles increases, as expected. Recall that the tuning power consumption in the PhS is directly proportional to the phase angle; therefore, a reduction in the mean phase angle signifies a proportional reduction in the tuning power. We find that with global LTPrune, we reach the maximum achievable sparsity in a few pruning rounds (eight for Network-1, five for Network-2, and ten for Network-3). Beyond this point, the inferencing accuracy drops steeply. Using global LTPrune, we obtain a maximum sparsity of 57% for Network-1, 68% for Network-2, and 73% for Network-3.



(a) Network-1

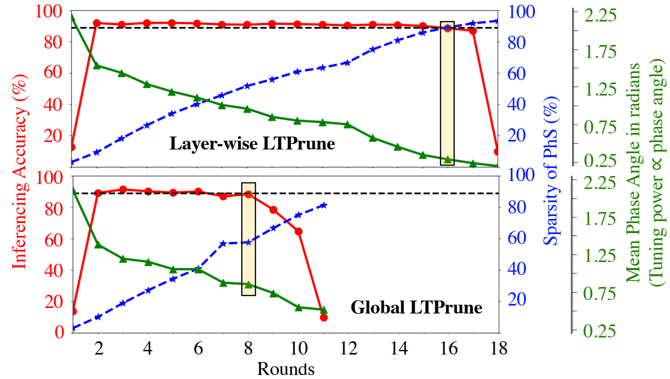


(b) Network-2

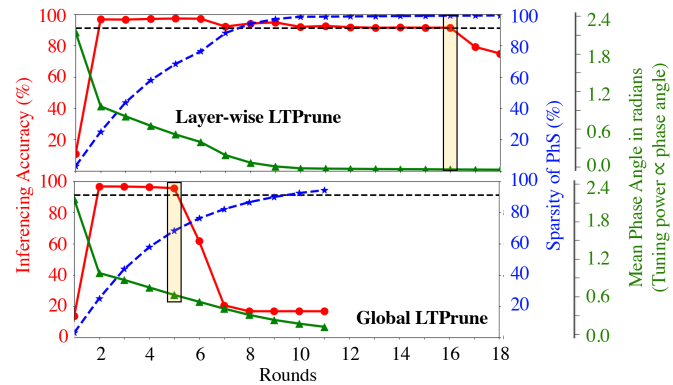


(c) Network-3

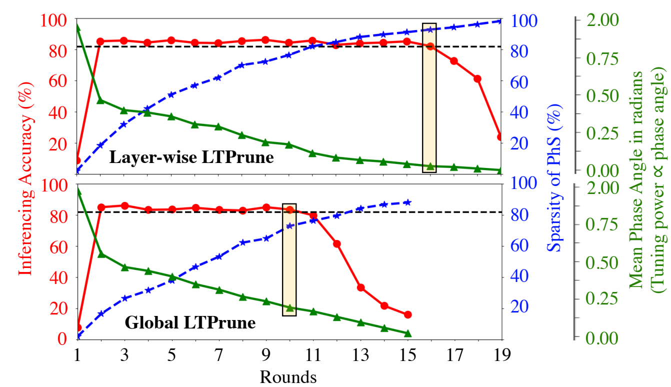
Fig. 7: Fine-tuned inferencing accuracy, sparsity of PhS, and mean phase angle for one-shot and iterative CHAMP when applied to (a) Network-1, (b) Network-2, and (c) Network-3 for different values of α. The black-dashed lines show a 5% accuracy loss and the yellow rectangles highlight the best-performing models (maximum sparsity with accuracy loss <5%) in each case.



(a) Network-1



(b) Network-2



(c) Network-3

Fig. 8: Fine-tuned inferencing accuracy, sparsity of PhS, and mean phase angle for layer-wise and global LTPrune when applied to (a) Network-1, (b) Network-2, (c) Network-3.

In contrast, with layer-wise LTPrune, the inferencing accuracy remains within 5% of the nominal accuracy for 16 pruning rounds for all three networks. Using this approach, we can prune up to 89% of the PhS in Network-1, 99% of the PhS in Network-2, and 93% of the PhS in Network-3.

Fig. 9 shows the results of iterative CHAMP and layer-wise LTPrune, when applied to Network-3 where we only use the inferencing accuracy of 10k images (separate from the test dataset) for validation. The best-performing models obtained using iterative CHAMP and layer-wise LTPrune demonstrate inferencing accuracies of 80.12% and 79.73%, respectively, for the remaining 10k test images. These correspond to a less than 7% drop in the inferencing accuracy compared to that of the unpruned model (86.67%). Therefore, CHAMP and LTPrune are able to obtain sparse IPNNs that offer high test accuracy, even when we use a separate validation dataset.

The global LTPrune performs worse compared to the layer-wise pruning as it is biased towards layers with smaller phase angles, i.e., it is possible that most phase angles in such layers are pruned away in the initial rounds. This is highlighted in Fig. 10 where we show the sparsity of phase angles in the three (one input and two hidden) layers of layer-wise and global LTPruned models, for Network-1 and Network-

This article has been accepted for publication in IEEE Journal of Selected Topics in Quantum Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTQE.2023.3242992

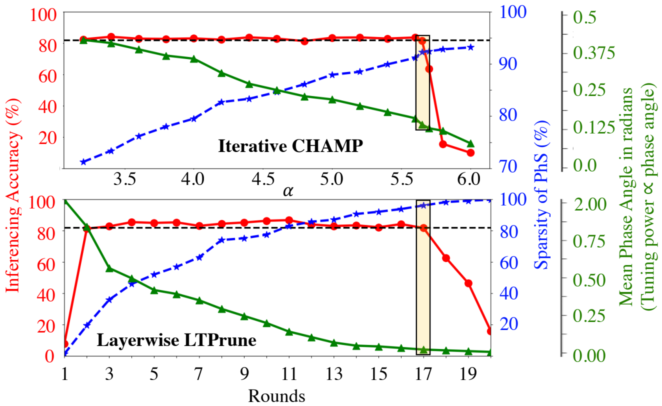
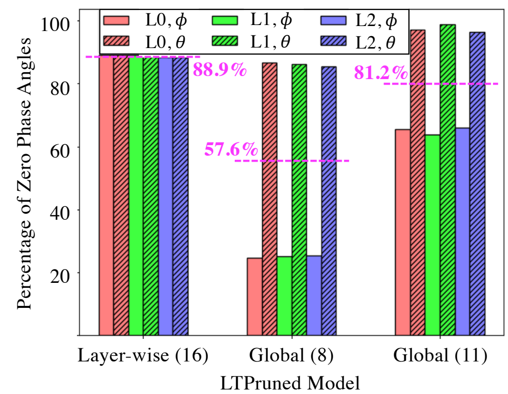


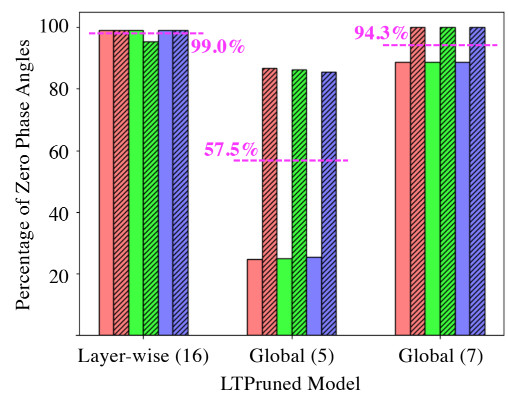
Fig. 9: Fine-tuned inferencing accuracy, sparsity of PhS, and mean phase angle for iterative CHAMP and layer-wise LTPrune on Network-3 where we only use the inferencing accuracy of 10k images (out of the 20k total test dataset) for validation. The test accuracies for the best-performing models (yellow rectangles) are 80.12% and 79.73%, respectively. The validation accuracies are 82.34% and 81.14%, respectively.

2. For Network-1, the global model with the best trade-off between accuracy and sparsity (eight rounds of pruning, 88.9% accuracy, and 57.6% mean sparsity), the percentage of pruned (i.e., zero) θ phase angles is considerably higher than ϕ. In the global model with maximum sparsity (11 rounds of pruning and 81.2% mean sparsity), up to 98.8% of θ phase angles are pruned. This holds for the LTPruned models for Network-2 as well – 99.93% of the θ phase angles are pruned in the global model with maximum sparsity. Extreme sparsity in certain layers can potentially hinder training and lead to exploding loss. In contrast, layer-wise pruning (see Fig. 11(a)) ensures that the sparsity of phase angles is uniform across the different layers. This leads to a lesser likelihood of exploding loss (and consequently, exploding gradient) of layer-wise LTPrune, even at higher levels of sparsity.

Next, we present a comparative study of the performance of CHAMP and LTPrune when applied to Network-1 and Network-2. Fig. 11(a)-(b) compares the histogram distributions of the phase angles of the best-performing models obtained using iterative CHAMP and layer-wise LTPrune with that of the unpruned IPNN. Observe that LTPrune outperforms CHAMP by a significant margin for Network-1, we obtain a sparsity of 55% using CHAMP compared to a sparsity of 89% using LTPrune. However, their performance for the larger Network-2 is similar. In fact, CHAMP offers a slightly higher sparsity (99.48%) compared to that of the best-performing LTPrune model (99.02%). This indicates that while LTPrune outperforms CHAMP when applied to difficult-to-prune compact networks, their effectiveness is similar while pruning highly overparameterized IPNNs. Recall also that, in LTPrune, all the unpruned phase angles are reset to their initial values and retrained from scratch after every pruning round; consequently, each round of LTPrune necessitates a higher number of retraining epochs. Therefore, taking the higher retraining time associated with LTPrune, it is recommended that for large over-parameterized IPNNs, we should preferentially use CHAMP. In contrast, for compact IPNNs, where retraining is



(a) Network-1



(b) Network-2

Fig. 10: Phase-angle sparsity (ϕ and θ) in the three layers (L0, L1, and L2) of different LTPruned models for (a) Network-1 and (b) Network-2. The x-axis shows the variant of LTPrune used with the number of rounds in parentheses. The magenta-dashed lines indicate the mean sparsity over all the layers.

faster, we can use LTPrune for higher sparsity.

Fig. 11(c)-(d) presents compares the accuracy and sparsity of pruned IPNNs (Network-1 and Network-2, respectively) obtained using different methods. Here, we only consider those models where the accuracy loss (from the respective unpruned models) is less than 5%. The magenta data points in each figure denote the unpruned models. From Fig. 11(c), it is clear that for Network-1, only layer-wise LTPrune can offer a sparsity greater than 60%. When very low accuracy loss (<1%) is acceptable after pruning, CHAMP can be considered, which achieves a maximum sparsity of 22% under this constraint. Similarly, for Network-2 (Fig. 11(d)), layer-wise LTPrune offers the maximum sparsity of 96.91% for a <1% accuracy loss. Observe also that contrary to Network-1, in the case of Network-2, both iterative CHAMP and layer-wise LTPrune offer a high sparsity for a <5% accuracy loss.

B. Characterizing Pruned IPNNs under Uncertainties

In [21] and [22], we have shown that the IPNN inferencing accuracy is sensitive to several imperfections such as uncertainties in the MZIs, insertion loss, and low-precision drivers. In particular, expected levels of uncertainties in the phase angles due to fabrication process variations and thermal crosstalk have a catastrophic impact on performance. As an IPNN is gradually pruned, we essentially discard the redundant

This article has been accepted for publication in IEEE Journal of Selected Topics in Quantum Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTQE.2023.3242992

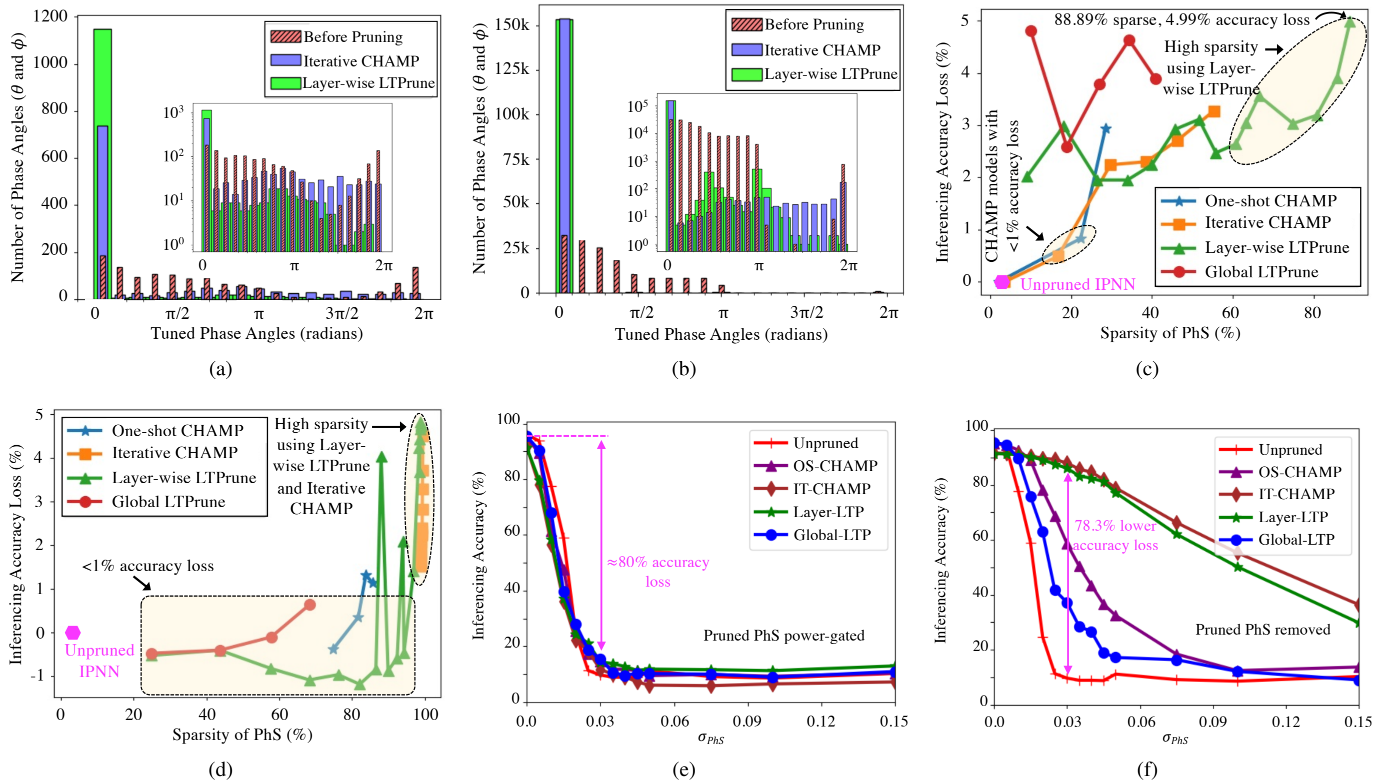


Fig. 11: (a)-(b) Histogram distribution of the phase angles in Network-1 and Network-2 with CHAMP and LTPrune pruning (inset shows the same plot with a logarithmic scale on the y-axis). (c)-(d) Comparison between the accuracy loss and sparsity of PhS in pruned Network-1 and Network-2 models obtained using different methods. (e)-(f) Accuracy of the unpruned, best-performing one-shot (OS)-CHAMP, best performing iterative (IT)-CHAMP, best-performing layer-wise LTP, and best-performing global LTP models under random phase uncertainties when the pruned PhS are (e) power-gated and (f) removed.

phase angles. Consequently, the non-zero phase angles in the obtained sparse IPNN have a high saliency, and small uncertainties in these components can lead to a large accuracy loss. To demonstrate this, we consider the larger IPNN (Network-2) and inject uncertainties in the phase angles of the unpruned IPNN and those of the best-performing models obtained from one-shot CHAMP, iterative CHAMP, layer-wise LTPrune, and global LTPrune. We perform 1000 Monte Carlo (MC) iterations; in each iteration, the uncertainties are sampled from a zero-mean Gaussian distribution with a standard deviation of $\sigma_{PS} \cdot \pi$. Fig. 11(e)-(f) shows the mean inferencing accuracy—over 1000 MC iterations—for the five models for different values of σ_{PS}.

Note that the redundant PhS identified during pruning can be handled in two ways: (1) they can be power-gated (turned-off) and left in the network (Fig. 11(e)), or 2) they can be altogether removed from the IPNN (Fig. 11(f)). We observe that in the first case (power-gated PhS), compared to the unpruned model, the pruned networks are slightly more susceptible to phase uncertainties because even small uncertainties in otherwise zero phase angles lead to a large relative deviation in the MZI operation. For all models, the accuracy drops steeply with σ_{PhS}. In contrast, removing pruned PhS reduces the number of uncertainty-susceptible components and leads to significantly higher accuracy (up to 78.3%) under uncertainties. Additionally, removing the redundant PhS leads to a lower area overhead, and lower optical loss (due to reduced network depth). Therefore, in situations where physical modifications in the IPNN are feasible, the pruned PhS should be removed. Alternatively, for general-purpose photonic accelerators, which may need to be retrained when the application workload changes, hardware modifications are infeasible. In such cases, it must be ensured that the phase uncertainties are minimized.

V. HybridPrune: In-field Pruning using a CHAMP-LTPrune Hybrid Approach

While layer-wise LTPrune offers a high sparsity, it necessitates a large number of retraining epochs to recover the inferencing accuracy in the pruned IPNNs. This is because, in each LTPrune iteration, in addition to all the pruned weights being clamped to zero, the unpruned weights in each layer are reset back to their initial (pre-training) values. As a result, the correlation between the features and the labels is learned slowly (compared to conventional training). Note that this is in contrast to prior observations in DNNs where LTPruned networks have been shown to be more easily trainable. This can potentially be attributed to the BMA between the phase angles and the weights in IPNNs; LTPrune in IPNNs generates sparse phase angles and not easily trainable sparse subnetworks with few non-zero weights. The large training time and computational overhead associated with LTPrune are typically not a concern for application-specific IPNNs; pruning such networks is a one-time operation and the associated costs will be amortized by the reduction in tuning power

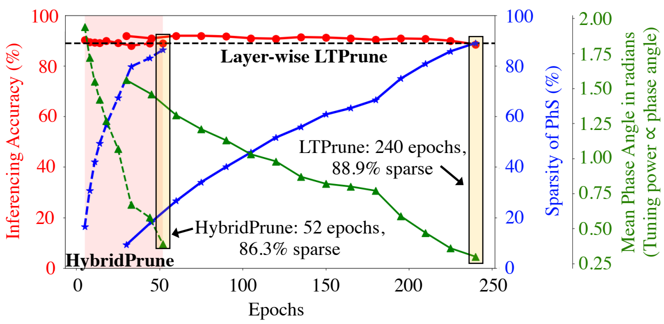


Fig. 12: Comparison of the fine-tuned inferencing accuracy, sparsity of PhS, and mean phase angle when Network-1 is pruned using layer-wise LTPrune (solid lines) and the CHAMP-LTPrune hybrid approach (dashed lines, shaded region). The X-axis shows the number of retraining epochs necessary to fine-tune the inferencing accuracy. Simulation results show that with the hybrid approach, up to 86.3% sparsity can be obtained with only 52 retraining epochs. While layer-wise LTPrune offers a slightly higher sparsity of 88.9%, it necessitates significantly more ($\approx$ 240) retraining epochs.

and area overhead in high-volume production. However, for general-purpose IPNNs, which can be reused for multiple application workloads, pruning needs to be repeated in-field based on the respective tuned phase angles of each application. Consequently, pruning approaches for general-purpose IPNNs must be fast and have a low computational overhead.

Note that, contrary to LTPrune, the unpruned weights in each iteration in CHAMP are not reset back to their initial values. Consequently, retraining in CHAMP requires fewer epochs compared to that in LTPrune. For example, in our simulations, iterative CHAMP required a maximum of 10 retraining epochs across all iterations. On the other hand, layer-wise LTPrune required at least 15 retraining epochs across the different rounds. Clearly, while LTPrune is likely to offer a higher phase sparsity, CHAMP is the faster of the two. Therefore, this tradeoff between the performance and the run time needs to be explored while determining the optimal pruning approach for a given IPNN. It is expected that for larger (over-parameterized) IPNNs, that are typically easy to prune, CHAMP and LTPrune will offer similar PhS sparsity. In fact in our simulations results for Network-2, the maximum sparsity obtained using CHAMP (99.48%) is marginally higher than that obtained using LTPrune (99.02%). Therefore, in such IPNNs, it is recommended to use CHAMP to perform fast in-field pruning. However, for small IPNNs, the sparsity obtained using LTPrune is higher than that from CHAMP; consequently, analyzing the erstwhile tradeoff between sparsity and pruning run time becomes crucial. For example, while CHAMP can prune only up to 55% of the PhS in Network-1, we obtain a sparsity of up to 89% using LTPrune.

To enable efficient in-field pruning of small (difficult-to-prune) IPNNs, we propose HybridPrune, a novel approach that leverages both CHAMP and LTPrune. In HybridPrune, we first ramp up the PhS sparsity using iterative CHAMP. Note that, as we are using CHAMP, each iteration here typically requires only a few retraining epochs. Iterative CHAMP is repeated till the finetuned inferencing accuracy falls below 5% of the nominal accuracy. The best-performing CHAMP model, with maximum PhS sparsity and a $<$5% accuracy loss, is then used as an input for layer-wise layer-wise LTPrune. We run LTPrune for multiple rounds, till the inferencing accuracy falls below 5% of the nominal accuracy. While each LTPrune round will necessitate several retraining epochs, the number of such rounds will be limited given that the input model to LTPrune had a high initial sparsity (obtained using CHAMP). Note also that, based on the IPNN architecture and the tuned phase angles, we may need to perform more than one run of CHAMP and LTPrune each. In other words, instead of the CHAMP-LTPrune sequence mentioned above, we may need to perform CHAMP-LTPrune-CHAMP-LTPrune or another similar sequence. Essentially, the idea is that CHAMP should be used for a quick ramp-up in the sparsity of easy-to-prune networks, whereas the time-intensive LTPrune should be used for the difficult-to-prune cases. Therefore, as the prunability of the intermediate IPNN changes during pruning, we can determine whether to use CHAMP or LTPrune. The number of retraining epochs can be further reduced by keeping track of the inferencing accuracy in each iteration/round; using this information, the retraining in each iteration/round can be stopped once a threshold inferencing accuracy is obtained.

Fig. 12 compares the inferencing accuracy, PhS sparsity, and the mean phase angle for different numbers of retraining epochs when Network-1 is pruned using the HybridPrune and layer-wise LTPrune. In HybridPrune, we first perform iterative CHAMP with $\alpha = 0.2$ for six iterations. In each iteration, the retraining is done till an inferencing accuracy within 5% of the nominal accuracy (93.86%) is obtained. Based on this policy, we obtain a PhS sparsity of 67.27% with only 25 retraining epochs distributed over the six pruning iterations. This model is then used as an input to the LTPrune phase of HybridPrune where we perform three rounds of pruning with $k = 25\%$. Using layer-wise LTPrune, we obtain a sparsity of up to 86.34% with only 27 additional retraining epochs spread over the three rounds. In total, HybridPrune requires only 52 retraining epochs to achieve a sparsity of 86.34%. This translates to a 78.3% reduction in the run time compared to the standalone layer-wise LTPrune that necessitates 240 retraining epochs to obtain a sparsity of 88.9%.

In Section IV.B, we show that the pruned PhS can either be power-gated or removed from the IPNN. In both these cases, the tuning power consumption is reduced in the pruned network; however, the area overhead is reduced only when the PhS are removed. Also, recall from simulation results in Fig. 11(e)-(f) that the reliability of the pruned IPNNs under random phase uncertainties improves when the pruned PhS are removed, whereas it remains similar to the nominal IPNN when the PhS are power-gated. For the general-purpose IPNNs, which should be trainable for different application workloads, the pruned PhS can not be removed. Consequently, HybridPrune for such IPNNs does not lead to a reduced area or improved reliability under uncertainties. In spite of these drawbacks, HybridPrune is a promising in-field alternative to the standalone CHAMP and LTPrune due to the reduction in the retraining time, especially for difficult-to-prune IPNNs.

This article has been accepted for publication in IEEE Journal of Selected Topics in Quantum Electronics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSTQE.2023.3242992

VI. Conclusion

Due to the bidirectional many-to-one mapping between the software weights and the phase angles in IPNNs, conventional DNN pruning techniques, when applied to IPNNs, prove to be inefficient. We propose CHAMP and LTPrune, two novel hardware-aware pruning techniques for IPNNs, and show that, for large IPNNs, we can achieve more than 99% sparsity with an accuracy loss of less than 5% using these methods. We have also shown that for smaller IPNNs, CHAMP can be used to obtain a moderate sparsity (up to 22%) and ultra-low accuracy loss ($<$1%) and LTPrune can achieve ultra-high sparsity (up to 89%) with an acceptable accuracy loss ($<$5%). Using these approaches, we can improve the power efficiency (by up to 98.2%) and enhance the robustness of IPNNs under uncertainties in the phase angles. While LTPrune offers the maximum sparsity, it necessitates several retraining epochs to recover the inferencing accuracy. To address this, we propose HybridPrune where we combine CHAMP and LTPrune to obtain highly sparse IPNNs with up to 78% fewer retraining epochs compared to standalone layer-wise LTPrune.

References

[1] Q. Cheng *et al.*, "Silicon photonics codesign for deep learning," *Proceedings of the IEEE*, vol. 108, no. 8, pp. 1261–1282, 2020.

[2] F. P. Sunny *et al.*, "A survey on silicon photonics for deep learning," *arXiv preprint arXiv:2101.01751*, 2021.

[3] M. Reck *et al.*, "Experimental realization of any discrete unitary operator," *Physical Review Letters*, vol. 73, no. 1, p. 58, 1994.

[4] W. R. Clements *et al.*, "Optimal design for universal multiport interferometers," *Optica*, vol. 3, no. 12, pp. 1460–1465, 2016.

[5] M. Jacques *et al.*, "Optimization of thermo-optic phase-shifter design and mitigation of thermal crosstalk on the SOI platform," *Optics Express*, vol. 27, no. 8, pp. 10 456–10 471, 2019.

[6] N. C. Harris *et al.*, "Efficient, compact and low loss thermo-optic phase shifter in silicon," *Optics express*, vol. 22, no. 9, pp. 10 487–10 493, 2014.

[7] S. Banerjee *et al.*, "Optimizing coherent integrated photonic neural networks under random uncertainties," *IEEE/OSA Optical Fiber Conference (OFC)*, 2021.

[8] F. Shokraneh *et al.*, "Theoretical and experimental analysis of a 4× 4 reconfigurable mzi-based linear optical processor," *Journal of Lightwave Technology*, vol. 38, no. 6, pp. 1258–1267, 2020.

[9] S. Han *et al.*, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *ICLR*, 2016.

[10] J. Gu *et al.*, "Towards area-efficient optical neural networks: An FFT-based architecture," in *IEEE/ACM ASP-DAC*, 2020.

[11] M. Y. S. Fang *et al.*, "Design of optical neural networks with component imprecisions," *Optics Express*, pp. 14 009–14 029, 2019.

[12] M. Bahadori *et al.*, "Thermal rectification of integrated microheaters for microring resonators in silicon photonics platform," *IEEE Journal of Lightwave Technology*, vol. 36, no. 3, pp. 773–788, 2017.

[13] M. J. Connelly, *Semiconductor Optical Amplifiers*. Springer Science & Business Media, 2007.

[14] B. Haq *et al.*, "Micro-transfer-printed iii-v-on-silicon c-band semiconductor optical amplifiers," *Laser & Photonics Reviews*, vol. 14, no. 7, p. 1900364, 2020.

[15] X. Xiao *et al.*, "Large-scale and energy-efficient tensorized optical neural networks on III–V-on-silicon MOSCAP platform," *APL Photonics*, vol. 6, no. 12, p. 126107, 2021.

[16] S. Banerjee *et al.*, "CHAMP: Coherent hardware-aware magnitude pruning of integrated photonic neural networks," *IEEE/Optica Optical Fiber Communications Conference and Exhibition (OFC)*, 2022.

[17] ——, "Pruning coherent integrated photonic neural networks using the lottery ticket hypothesis," *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2022.

[18] Q. Zhang *et al.*, "Implementation of pruned backpropagation neural network based on photonic integrated circuits," in *Photonics*, vol. 8, no. 9, 2021, p. 363.

[19] F. Sunny *et al.*, "SONIC: A sparse neural network inference accelerator with silicon photonics for energy-efficient deep learning," *arXiv preprint arXiv:2109.04459*, 2021.

[20] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[21] S. Banerjee *et al.*, "Modeling silicon-photonic neural networks under uncertainties," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.

[22] ——, "Characterizing coherent integrated photonic neural networks under imperfections," *Journal of Lightwave Technology*, 2022.

Sanmitra Banerjee received the B.Tech. degree from the Indian Institute of Technology, Kharagpur, in 2018, and the M.S. and Ph.D. degrees from Duke University, Durham, NC, in 2021 and 2022, respectively. He is currently a Senior DFX Methodology Engineer at NVIDIA Corporation, Santa Clara, CA. His research interests include machine learning-based DFX techniques, and fault modeling and optimization of emerging AI accelerators under process variations and manufacturing defects.

Mahdi Nikdast (S'10–M'14–SM'19) is an Assistant Professor in the Department of Electrical and Computer Engineering at Colorado State University (CSU), Fort Collins. He received his Ph.D. in Electronic and Computer Engineering from The Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2014. From 2014 to 2017, he was a post-doctoral fellow jointly at McGill University and Polytechnique Montreal, Canada. He is the director of the Electronic-PhotoniC System Design (ECSyD) Laboratory at CSU. His research interests include various topics related to integrated photonics and high-performance computing. Prof. Nikdast currently serves as an Associate Editor for IEEE Transactions on Very Large Scale Integration Systems (IEEE TVLSI). He was the recipient of various awards, including the National Science Foundation (NSF) CAREER Award in 2021. Prof. Nikdast is a Senior Member of IEEE.

Sudeep Pasricha (M'02, SM'13) received his Ph.D. degree in computer science from the University of California at Irvine, USA, in 2008. He is currently a Professor and Chair of Computer Engineering in the Department of Electrical and Computer Engineering at Colorado State University. His current research interests include optical computing, chip-scale network and memory architectures, hardware-software co-design for machine learning, and optimizations for energy, reliability, and security in embedded systems. He has received 15 Best Paper Awards and Nominations at various IEEE and ACM conferences, including at DAC, ASPDAC, NOCS, GLSVLSI, SLIP, AICCSA, and ISQED. Other notable awards include: the 2019 George T. Abell Outstanding Research Faculty Award, the 2016-2018 University Distinguished Monfort Professorship, 2016-2019 Walter Scott Jr. College of Engineering Rockwell-Anderson Professorship, 2018 IEEE-CS/TCVLSI Mid-Career Research Achievement Award, the 2015 IEEE/TCSC Award for Excellence for a Mid-Career Researcher, the 2014 George T. Abell Outstanding Mid-Career Faculty Award, and the 2013 AFOSR Young Investigator Award. He is currently serving as the Vice Chair of ACM SIGDA and a Senior Associate Editor for the ACM Journal of Emerging Technologies in Computing (JETC).

Krishnendu Chakrabarty received the B. Tech. degree from the Indian Institute of Technology, Kharagpur, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively. He is now the Fulton Professor of Microelectronics in the School of Electrical, Computer and Energy Engineering at Arizona State University (ASU). Before moving to ASU, he was the John Cocke Distinguished Professor of Electrical and Computer Engineering (ECE) and Department Chair of ECE at Duke University. Prof. Chakrabarty is a recipient of the National Science Foundation CAREER award, the Office of Naval Research Young Investigator award, the Humboldt Research Award from the Alexander von Humboldt Foundation, Germany, the IEEE Transactions on CAD Donald O. Pederson Best Paper Award (2015), the IEEE Transactions on VLSI Systems Prize Paper Award (2021), the ACM Transactions on Design Automation of Electronic Systems Best Paper Award (2017), multiple IBM Faculty Awards and HP Labs Open Innovation Research Awards, and over a dozen best paper awards at major conferences. He is also a recipient of the IEEE Computer Society Technical Achievement Award (2015), the IEEE Circuits and Systems Society Charles A. Desoer Technical Achievement Award (2017), the IEEE Circuits and Systems Society Vitold Belevitch Award (2021), the Semiconductor Research Corporation (SRC) Technical Excellence Award (2018), the SRC Aristotle Award (2022), the IEEE-HKN Asad M. Madni Outstanding Technical Achievement and Excellence Award (2021), and the IEEE Test Technology Technical Council Bob Madge Innovation Award (2018). He is a Research Ambassador of the University of Bremen (Germany) and he was a Hans Fischer Senior Fellow at the Institute for Advanced Study, Technical University of Munich, Germany during 2016-2019. He is a 2018 recipient of the Japan Society for the Promotion of Science (JSPS) Invitational Fellowship in the "Short Term S: Nobel Prize Level" category.Prof. Chakrabarty's current research projects include: design-for-testability of 3D integrated circuits; AI accelerators; microfluidic biochips; hardware security; AI for healthcare; neuromorphic computing systems. He is a Fellow of ACM, IEEE, and AAAS, and a Golden Core Member of the IEEE Computer Society. He was a Distinguished Visitor of the IEEE Computer Society (2005-2007, 2010-2012), a Distinguished Lecturer of the IEEE Circuits and Systems Society (2006-2007, 2012-2013), and an ACM Distinguished Speaker (2008-2016). Prof. Chakrabarty served as the Editor-in-Chief of IEEE Design Test of Computers during 2010-2012, ACM Journal on Emerging Technologies in Computing Systems during 2010-2015, and IEEE Transactions on VLSI Systems during 2015-2018.