Pruning Coherent Integrated Photonic Neural Networks Using the Lottery Ticket Hypothesis

Sanmitra Banerjee*, Mahdi Nikdast†, Sudeep Pasricha†, and Krishnendu Chakrabarty*

*Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA
†Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523, USA

***Abstract*—Coherent integrated photonic neural network (C-IPNN) architectures enable light-speed and ultra-low-energy accelerators for rapidly growing artificial intelligence applications. Nevertheless, C-IPNNs have a large footprint and suffer from high tuning power consumption for both training and inference. Pruning C-IPNNs using model compaction to reduce the number of weight parameters can potentially alleviate these problems. However, prior attempts at pruning singular-value-decomposition-based C-IPNNs (SC-IPNNs) have shown that very few parameters can be removed without significantly degrading the network accuracy. In this paper, we present the first hardware-aware pruning method for SC-IPNNs based on the lottery ticket hypothesis (LTH). We also discuss the challenges associated with pruning SC-IPNNs and show that, in addition to the classification accuracy, model-compaction techniques should be guided by a reliability assessment of the pruned networks. As a case study, we prune a multi-layer perceptron-based SC-IPNN with two hidden layers and show that up to 89% of the phase angles, which correspond to weight parameters in SC-IPNNs, can be pruned with a negligible loss in accuracy (smaller than 5%) while reducing the network (i.e., tuning) power consumption by up to 86%. Therefore, the proposed pruning method paves the way for realizing compact and energy-efficient photonic neural networks.**

I. Introduction

Silicon photonics can enable compact, ultra-fast, and ultra-low-energy artificial intelligence (AI) accelerators, realizing a promising framework for the emerging class of information processing machines [1]. In particular, leveraging the inherently parallel nature and high-speed of optical-domain computation, coherent integrated photonic neural networks (C-IPNNs), which operate with a single wavelength, can reduce the computational complexity of matrix multiplication—the most compute-intensive operation in deep neural networks (DNNs)—from $O(N^2)$ to $O(1)$ [2]. Using singular value decomposition (SVD), several C-IPNNs (referred to as SC-IPNNs in this paper) have been recently proposed [1].

As shown in Fig. 1(a), SC-IPNNs use arrays of Mach–Zehnder interferometers (MZIs) as their building block. The phase angles on each MZI can be adjusted based on the weights in the network and by using training algorithms [3]. Recent work has shown up to a 70% accuracy loss in SC-IPNNs due to uncertainties in MZI phase settings [4], and found that such an accuracy loss is mostly due to the uncertainties in MZIs with higher adjusted phase angles. Moreover, SC-IPNNs suffer from large area and tuning power consumption. In particular, the underlying MZI devices in SC-IPNNs employ lengthy phase

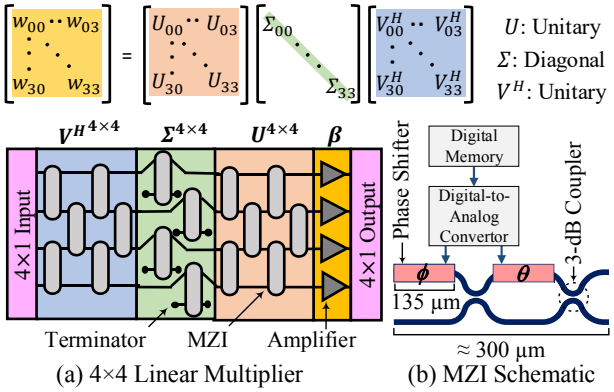


Fig. 1: (a) A 4×4 linear multiplier realized using MZI arrays, representing the weights of a four-neuron layer connected to another four-neuron layer. (b) An MZI with two phase shifters. Here, ϕ and θ denote the phase angles. The MZI footprint is limited by the length of its phase shifters (MZI dimensions obtained from [5]).

shifters (e.g., as long as 135 μm [5]) and consume high power (e.g., ≈25 mW per MZI, dominated by the tuning power in MZI phase shifters [6]). A potential solution to alleviate these problems is to prune SC-IPNNs to reduce the number of bulky components (e.g., phase shifters in MZIs—see Fig. 1(b)) and minimize the phase settings in the network.

While software pruning of weights has shown promising results in electronic implementations of DNNs [7], its applicability to SC-IPNN model compaction is significantly limited. This is because of the complex mapping between the weights in the fully connected layer (in software) and the phase angles (in hardware) in SC-IPNNs: i.e., each weight is mapped to multiple phase angles and each phase angle is used to realize multiple weights. Consequently, it is extremely challenging to selectively prune the phase angles that only affect the non-critical (low saliency) weights without deviating the critical weights. Prior efforts on pruning SC-IPNNs using existing techniques have shown that only up to 30% of the phase angles can be pruned without significant degradation in the network accuracy [8].

In this paper, we present the first efficient hardware-aware pruning method for SC-IPNNs based on the lottery ticket hypothesis (LTH). This hypothesis states that given any randomly initialized, dense, feed-forward DNN, there exists a sub-network that—when trained from scratch—can match the test accuracy of the original DNN [9]. Recent work on model compaction of software DNNs has empirically demonstrated the existence of such sub-networks (*winning tickets*), which are 10–20% of the size of the original network. By leveraging

This research was supported in part by the National Science Foundation (NSF) under grant numbers CCF-1813370, CCF-2006788, and CNS-2046226.

2159-3477/22/$31.00 ©2022 IEEE
DOI 10.1109/ISVLSI54635.2022.00035

2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI) | 978-1-6654-6605-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISVLSI54635.2022.00035

insights from LTH, our pruning method identifies a small subset of phase angles in SC-IPNNs that are critical for maintaining the classification accuracy. As a result, we can reduce the footprint and tuning power consumption in SC-IPNNs either by removing or power-gating the redundant phase shifters in SC-IPNNs. We consider an SC-IPNN case study and show that we can prune up to 89% of the 1374 phase shifters in the original network, and achieve an 86% reduction in the tuning power consumption.

Pruning of SC-IPNNs with low accuracy loss is a considerably more challenging problem compared to that in electronic implementations of DNNs. To the best of our knowledge, the proposed method is the first to achieve highly sparse SC-IPNNs with less than 5% accuracy loss. The example SC-IPNN considered as our case study is compact even in its unpruned form (only two hidden layers with 16 neurons each). Therefore, we allow an accuracy loss of up to 5% to show that our method achieves significant sparsity even in such difficult-to-prune networks. The main contributions of this paper are:

- Identifying the challenges associated with pruning SC-IPNNs and the limitations of conventional hardware-unaware software pruning;
- Developing the first hardware-aware pruning method based on LTH to generate power- and area-efficient SC-IPNNs;
- Exploring the trade-off between the sparsity of the phase angles in a pruned SC-IPNN and its sensitivity to random uncertainties in the (remaining) non-zero phase angles.

The rest of the paper is organized as follows. Section II covers the fundamentals of SC-IPNNs and LTH and Section III highlights the challenges in pruning SC-IPNNs. We then describe a hardware-aware magnitude-based (baseline) pruning and the proposed LTH-based pruning. We present simulation results in Section IV and draw conclusions in Section V.

II. Background and Motivation

A. Coherent Integrated Photonic Neural Networks

C-IPNNs operate with a single wavelength and employ optical phase-change mechanisms in on-chip interferometric devices (i.e., MZIs) to imprint weight parameters onto the electrical field amplitude of optical signals [1]. Using SVD, the weight matrices in linear layers of multi-layer perceptrons can be factorized into two unitary and one diagonal matrices. Using the Clements design [10] and considering Fig. 1(a), the weight matrix of a linear layer can be realized using three MZI arrays as $W = U\Sigma V^H$, where V^H is the Hermitian transpose of V. An $N \times N$ unitary and diagonal matrix can be implemented by adjusting the phase angles in an array with $N(N-1)/2$ and N MZIs, respectively. In addition, global optical amplification (layer β in Fig. 1(a)) is necessary on each output [11]. The non-linear activation can be performed using opto-electronic units [12], not shown in Fig. 1(a) for the sake of brevity.

Fig. 1(b) shows a schematic of a 2×2 MZI with two phase shifters (PSes), where ϕ and θ are the phase angles. PSes are used to determine the relative phase difference between the two optical signals traversing the MZI arms and can be implemented using microheaters that work based on the thermo-optic effect in silicon [3]. In a thermo-optic PS, the temperature-induced phase shift ($\Delta\phi$ or $\Delta\theta$) is proportional to the temperature change (ΔT) based on $\Delta\phi = \left(\frac{2\pi L}{\lambda_0}\right) \cdot \left(\frac{dn}{dT}\right) \cdot \Delta T$. Here, L is the length of the PS and λ_0 is the optical wavelength [13]. Also, $\frac{dn}{dT} \approx 1.8 \cdot 10^{-4}$ K^{-1} is the thermo-optic coefficient of silicon at $\lambda_0 = 1550$ nm and temperature $T = 300$ K. The parameter ΔT can be controlled by applying a DC voltage using a digital-to-analog converter (see Fig. 1(b)). The tuning power consumption in the PSes, P, is directly proportional to ΔT: $\Delta T \propto P$ [13]. The MZI also includes two directional couplers with a nominal splitting ratio of 50:50 (3-dB couplers).

Software training of SC-IPNNs can be performed either in a hardware-unaware or in a *photonic* hardware-aware manner. In the hardware-unaware approach, the optimal DNN weight matrices are first obtained using training and are then mapped to different phase angles in the MZIs. In contrast, in *photonic* hardware-aware software training, backpropagation is performed on the phase angles that are adjusted based on the computed gradients. We employ this approach as it offers more control on the phase angles during software training. This is essential for efficient pruning of SC-IPNNs (see Section III).

B. Pruning SC-IPNNs: Motivation

Effective pruning of neural networks can reduce the infrastructure costs associated with the storage and computation of enormous DNNs, thereby enabling their deployment in resource-constrained environments. Additionally, pruning SC-IPNNs can improve their area- and power-efficiency, as discussed next.

The phase shifters in SC-IPNNs with MZI arrays consume a significant portion of the network area and power. In particular, the size of the constituent thermo-optic PSes in an MZI determines the size of the device (see Fig. 1(b)). For example, the state-of-the-art 2×2 MZI proposed in [5] is ≈ 300 μm long, in which each PS has a length of 135 μm (i.e., ≈ 90% of the length of the MZI considering the two PSes). 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 PSes 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 SC-IPNNs to identify prunable PSes, thereby reducing the footprint of the network and the tuning power consumption during inferencing. Note that the phase angles are adjusted dynamically during software training. However, we only consider the static tuning power dissipated in the PSes 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 PSes can be minimized by pruning. The problem of explicitly reducing thermal crosstalk is beyond the scope of this paper.

C. Lottery Ticket Hypothesis (LTH)

Recent studies have shown that training a pruned model from scratch is considerably difficult and it often achieves lower accuracy compared to the original (unpruned) model [7].

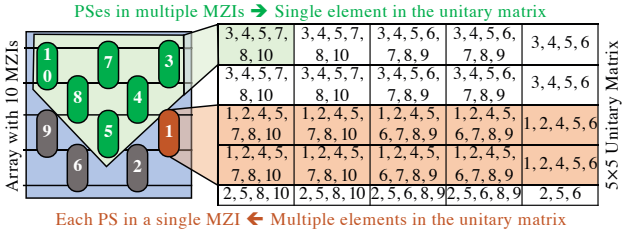


Fig. 2: An example of the bidirectional many-to-one association (BMA) between the elements of the weight matrix and the MZI array for a 5×5 unitary matrix. The numbers in each cell of the unitary matrix denote the MZIs that affect the corresponding matrix element.

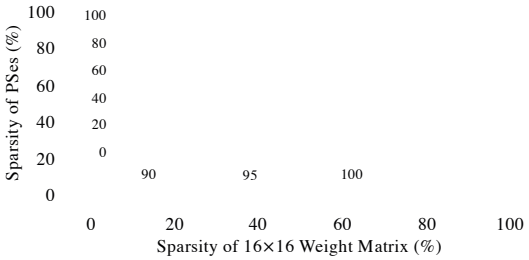


Fig. 3: Comparison between the sparsity of 10000 randomly generated 16×16 unitary matrices and the sparsity of the PSes (when phase angles are zero) in the respective MZI devices. Inset: Comparison of the matrix sparsity with the sparsity of PSes for 1000 highly sparse (matrix sparsity >90%) 16×16 unitary matrices. The red, green, and blue clusters indicate unitary matrices that are mapped to MZI arrays with low, medium, and high sparsity in the PSes, respectively.

Nevertheless, LTH has established that for a given randomly initialized network, we can always find a smaller sub-network that—when trained from scratch—can match the accuracy of the original network within a few training iterations [9]. These high-performing trainable sub-networks, called *the winning tickets*, can be identified by a modified magnitude-based pruning approach. After the smallest-magnitude weights (below a pre-determined threshold) are pruned, the remaining non-zero parameters are reset back to their original values (before the onset of training). This step is followed by retraining to recover the network accuracy. Experimental results show that the *winning ticket* for a network varies based on the initial weight values [9]. One possible explanation for this is that the *winning ticket* initialization can potentially land in a region-of-the-loss landscape that enables quick optimization. Model-compaction techniques have shown that stochastic-gradient descent seeks out and trains a sub-network. However, LTH highlights that there exist multiple such sub-networks unique to different initializations (see [9] for more details on LTH).

III. Enabling Efficient Pruning in SC-IPNNs

A. Challenges in Pruning SC-IPNNs

Hardware-unaware software pruning methods in DNNs aim at obtaining a sparse weight matrix [7]. 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 in each layer are not updated.

Unfortunately, there are several problems with applying such software pruning techniques to SC-IPNNs. In particular, each element of the weight matrix of a linear layer in SC-IPNNs is mapped to multiple phase angles, and each phase angle in an MZI array affects multiple elements of the weight matrix. Fig. 2 shows an example of this *bidirectional many-to-one association (BMA)* between a 5×5 unitary matrix and its corresponding MZI array. Due to this BMA, if a phase angle in an MZI is updated to prune a non-critical (low-magnitude) weight, it can also affect another potentially critical weight, thereby leading to significant accuracy losses. Moreover, because of BMA, a sparse weight matrix may not necessarily lead to *sparsity in the PSes of the corresponding MZI devices, i.e., when one or both of the phase angles (ϕ and θ) are zero and the PSes can be then removed or power-gated*. Fig. 3 compares the sparsity of 10000 randomly generated 16×16 unitary matrices with the sparsity of their mapped PSes in the corresponding MZIs. Observe that a highly sparse (>90%) weight matrix does not always lead to sparsity in PSes. While software pruning focuses on a sparse weight matrix, SC-IPNN model compaction should minimize and prune MZI phase angles to reduce area overhead and tuning power consumption. The discrepancy between these two objectives indicates the ineffectiveness of software pruning in SC-IPNNs and the critical need for hardware-aware pruning.

By leveraging hardware-unaware software pruning, where the weight matrices are first pruned in software and then mapped to MZI arrays, [8] showed that no more than 30% of the phase angles can be pruned without a significant accuracy loss (≈10%) in SC-IPNNs. To address this, [8] 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%.

B. Hardware-Aware Pruning in SC-IPNNs

As discussed in Section III-A, pruning in SC-IPNNs must be hardware-aware to ensure sparsity in phase angles. However, as we will show in Section IV, the nonlinear dependence between the phase angles and the weights in SC-IPNNs—e.g., weights with large magnitude can be mapped to smaller phase angles—makes it challenging to identify non-critical weights that can be safely pruned, even using hardware-aware techniques. Consequently, simply pruning phase angles based on their magnitude can be ineffective. In this section, we first present a baseline approach where we apply conventional magnitude-based pruning by considering the magnitude of the adjusted phase angles. Next, we present the LTH-based pruning technique and show that it can prune a significant fraction of phase angles with a negligible accuracy loss.

1) Baseline Method: Phase-Angle-based Magnitude Pruning

In magnitude-based pruning, all the weights in a layer having magnitudes smaller than a threshold are set to zero. Next, in post-pruning, the network is retrained to recover the lost accuracy. However, the pruned weights are kept at zero during the retraining. There are several ways to determine the threshold for a layer. For example, in magnitude pruning based on the mean (standard deviation), the threshold is considered to be a factor—say α—of the mean (standard deviation) of the non-zero weights in a layer. Magnitude pruning can be performed either in one shot or in an iterative manner. In one-shot pruning, all the weights below a threshold are pruned in a single step after which retraining (a.k.a. fine-tuning) is performed. In iterative pruning, weights are gradually pruned over multiple steps with each step followed by few rounds of fine-tuning. While extending magnitude pruning to SC-IPNNs, we implement both the one-shot and the iterative approaches. We consider *photonic* hardware-aware software training, and therefore, during backpropagation, gradients are calculated for each phase angle (and not layer edge weights). Consequently, the binary masks used to suppress the gradients for the pruned phase angles are also maintained for the PSes in the MZI array corresponding to the weight matrix of each linear layer.

2) Proposed Method: Using LTH to Prune SC-IPNNs

Pruning based on LTH is similar to magnitude pruning, but with a significant difference: after the weights are pruned, the remaining non-zero weights are set back to their initial values, which were stored before training began for the first time. This step is followed by retraining to recover the lost accuracy. Fig. 4 presents an overview of the proposed LTH-based pruning for SC-IPNNs. The inputs are the hyperparameters (learning rate and epochs) for the retraining step, the maximum acceptable accuracy loss, minimum sparsity, maximum number of pruning rounds (R_{max}), and the pruning rate (k). As a pre-processing step, we initialize the weights and store their values in a database. In each round, we check whether the network sparsity is acceptable, in which case we exit the pruning round and retrain to achieve acceptable classification accuracy. Otherwise, we enter the round and retrain the network. Note that the training hyperparameters should be adjusted in each round to obtain a sufficiently high accuracy ($< 5\%$ accuracy loss—considered as an example in this paper) before the phase angles are pruned. After retraining, the bottom k percentile of the phase angles with small magnitude are pruned and the respective binary masks are updated. The k-percentile pruning step can be performed in two ways: in layer-wise LTH pruning, the bottom k percentile weights in each tier are pruned, whereas, in global pruning, the bottom k percentile weights in the entire SC-IPNN are pruned. After pruning, we set the non-zero weights back to their initial values before proceeding to the next round. We use LTH to identify the best-performing *winning ticket* in an iterative manner over multiple rounds as it has been shown to yield more sparse subnetworks compared to the one-shot (single-round) approach [9].

Due to the unique challenges with the pruning of SC-IPNNs

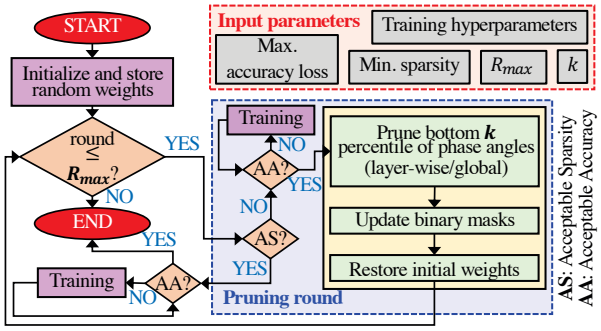


Fig. 4: An overview of the proposed LTH-based hardware-aware pruning method for SC-IPNNs. The input parameters are user-defined.

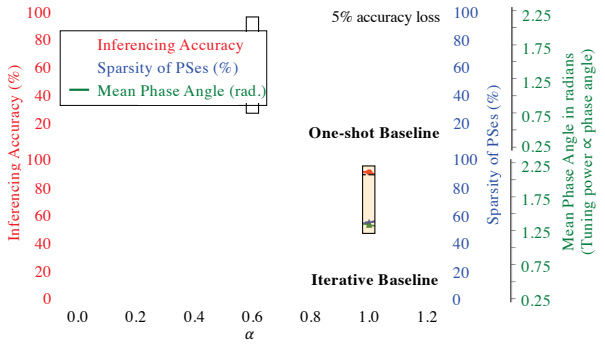
(see Section III-A), we encounter a significant loss in accuracy after each pruning iteration. However, by resetting the non-zero phase angles to their initial values after each iteration, our approach gradually identifies the best solution (with acceptable sparsity and accuracy) out of the *winning tickets* in each iteration. In accordance with LTH, this ensures that the pruned models obtained using our method can recover the accuracy loss (due to pruning) more effectively compared to a pruned model of similar sparsity obtained using the baseline method. As a result, we are able to achieve significantly higher sparsity with a negligible accuracy loss, as shown in the next section. Note that the proposed pruning method should be performed offline and only once per an SC-IPNN design.

IV. Simulation and Evaluation Results

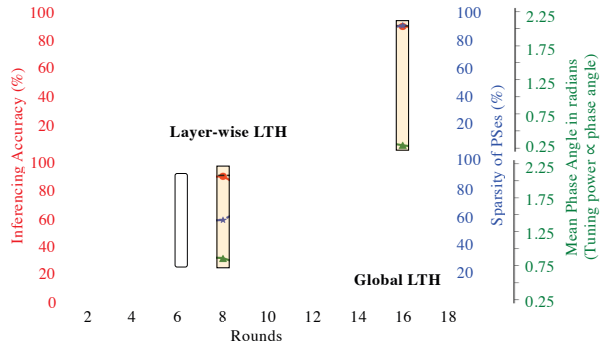
We use a fully connected feedforward SC-IPNN with two hidden layers (i.e., 32 neurons and 1374 PSes) to demonstrate the performance of the baseline and the proposed LTH-based pruning methods. Each linear layer is implemented using the Clements design [10] and is followed by a nonlinear Softplus function. To model intensity measurement, a modulus squared nonlinearity is applied after the output layer. This is followed by a final LogSoftMax layer to obtain a probability distribution. We use a cross-entropy loss function [14] to train the SC-IPNN on the MNIST dataset. Similar to [4], we use shifted FFTs to convert each 28×28 MNIST image to a 16-dimensional complex feature vector. The nominal inferencing accuracy (on the test dataset) of the unpruned SC-IPNN is 93.86%. Note that the proposed method is agnostic to the network (number and size of linear layers, non-linear activation, and loss functions).

A. Pruning Analysis for SC-IPNNs

Fig. 5(a) shows the simulation results for the one-shot (top) and iterative (bottom) phase-angle magnitude-based pruning with the baseline method. In each case, we consider standard-deviation-based pruning (and not the mean-based one) to calculate the threshold as it considers the distribution of the phase angles in each layer. The threshold is given by $\alpha \cdot \sigma_{layer}$. Here, α is a user-defined constant (see Section III-B1) and σ_{layer} denotes the standard deviation of the non-zero phase angles in a layer. We consider both one-shot and iterative approaches as in a few cases (e.g., for $\alpha = 0.2$), the former performs better. For iterative pruning, we approach this



(a) Baseline method



(b) Proposed LTH-based pruning method

Fig. 5: Fine-tuned accuracy, PS sparsity, and mean phase angle for (a) one-shot phase-angle magnitude-based (baseline) pruning (top) and iterative baseline pruning (bottom) with different values of α; and (b) different rounds of layer-wise (top) and global LTH-based pruning (bottom). The black-dashed lines show a 5% accuracy loss and the yellow rectangles highlight the best-performing models (maximum sparsity with accuracy loss <5%) for each pruning method.

threshold in incremental steps of $\alpha \cdot \sigma_{layer}/10$. As can be seen, in both cases the overall sparsity of the PSes increases with α. Also, the mean phase angle—averaged over the 1374 PSes to which the weight parameters are mapped—and hence the tuning power (see Section II-B) decrease as α increases. However, the inferencing accuracy drops significantly for the one-shot pruning as α increases. In contrast, in iterative pruning, the accuracy loss is less than 5% up to α =1 (see the black-dashed line in Fig. 5(a)). This is because of the gradual pruning and fine-tuning in iterative pruning, compared to the drastic pruning and few fine-tuning iterations in the one-shot case. If one allows for an accuracy loss of 5%, up to 55% (31%) PSes can be pruned using the iterative (one-shot) approach.

For the LTH-based pruning method, we found that the performance is better when we start with a low pruning rate (k) for the first few rounds, before aggressively pruning (high k) in the final rounds. Accordingly, we consider a pruning rate of k =10% for the first ten rounds and then increase it to k =25% for the remaining rounds. To show results over several rounds, we do not constrain the maximum accuracy loss, minimum sparsity, and R_{max} parameters, and tune the training hyperparameters to maximize the accuracy. Fig. 5(b) shows the results for the proposed layer-wise (top) and global (bottom) LTH-based pruning. Recall that in layer-wise (global) LTH-based pruning, the non-zero phase angles in the bottom k percentile of each layer (the entire SC-IPNN) are pruned. In both cases, the inferencing accuracy is significantly low in the first round as it is computed before the network is trained (see Fig. 4). The accuracy also drops sharply in the final round (round 18 for layer-wise and 11 for global) as the loss function explodes when a large fraction of phase angles are pruned, thereby leading to erroneous gradient propagation. As expected, the PS sparsity increases and the mean phase angle as well as the tuning power consumption (see Section II-B) decrease with more rounds of pruning. Yet, the accuracy loss is smaller than 5% (above the black-dashed line in Fig. 5(b)) for many rounds, especially for the layer-wise pruning. In fact, up to 89% of the PSes can be pruned with 86% reduction in the mean phase angles and tuning power consumption using 16 rounds of layer-wise LTH-based pruning. Similarly, up to 57% of phase angles can be pruned using the global LTH pruning.

The global LTH-based pruning 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. 6(a) where we show the sparsity of phase angles in the three (one input and two hidden) layers of layer-wise and global LTH-pruned models. In 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. Extreme sparsity in certain layers can potentially hinder training and lead to exploding loss. In contrast, layer-wise pruning (see Fig. 6(a)) ensures that the sparsity of phase angles is uniform across the different layers.

When considering both high sparsity and low accuracy loss, the LTH-based pruning outperforms hardware-aware magnitude pruning (baseline). Fig. 6(b) compares the accuracy loss and sparsity of pruned SC-IPNNs obtained using different methods. Here, we consider only those models where the accuracy loss due to pruning is less than 5%. The unpruned SC-IPNN has 3.03% sparsity based on the initialization (see the magenta data point). From Fig. 6(b) it is clear that only layer-wise LTH pruning can offer a sparsity greater than 60%. When very low accuracy loss (<1%) is acceptable after pruning, the hardware-aware magnitude pruning (baseline) can be considered, which achieves a maximum sparsity of 22% under this constraint. Fig. 6(c) compares the histograms of the phase angles of the best-performing models obtained using the baseline and the LTH-based pruning with that of the unpruned SC-IPNN. Layer-wise LTH-based pruning not only achieves high sparsity with negligible accuracy loss but minimizes the phase angles, resulting in significant savings in tuning power consumption (phase angle and tuning power are linearly proportional).

B. Noise Sensitivity of Pruned SC-IPNNs

As the redundant parameters in a DNN are gradually discarded during model compaction, the pruned DNN becomes more sensitive to uncertainties in the (few) remaining parame-

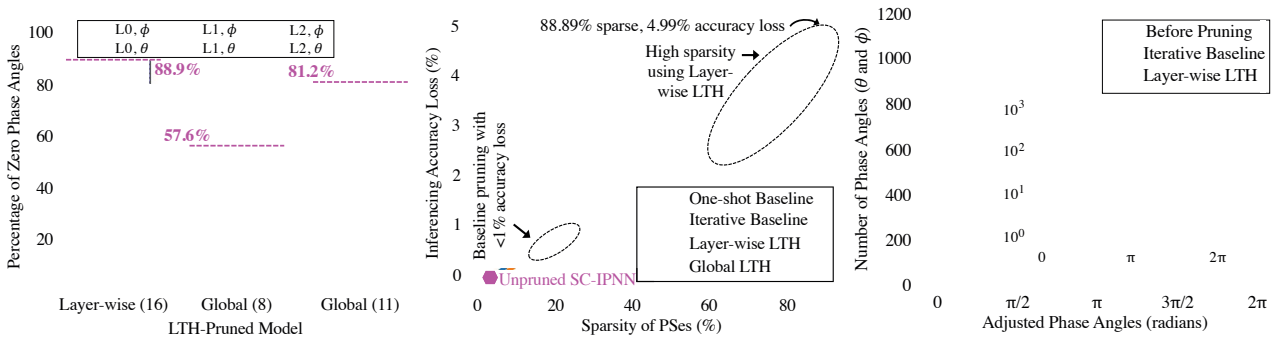


Fig. 6: (a) Phase-angle sparsity (ϕ and θ) in the three layers (L0, L1, and L2) of different LTH-pruned models. The x-axis shows the variant of LTH pruning used with the number of rounds in parentheses. The magenta-dashed lines indicate the mean sparsity over all the layers. (b) Comparison between the accuracy loss and PS sparsity in pruned models obtained using different methods. (c) Histogram distribution of the phase angles in the SC-IPNN with baseline and LTH-based pruning (inset shows the same plot with a logarithmic scale on the y-axis).

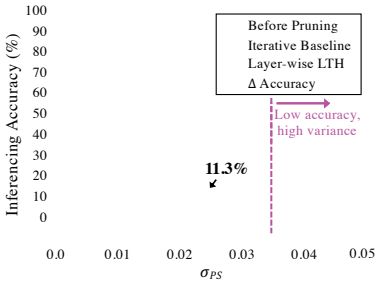


Fig. 7: Inferencing accuracy of the unpruned SC-IPNN and the best-performing pruned models obtained using the baseline and the LTH-based pruning methods. ΔAccuracy denotes the difference in the accuracy between the unpruned and the LTH-pruned models.

ters. This is indeed critical for sparse SC-IPNNs because even original and unpruned SC-IPNNs are sensitive to uncertainties, especially those in the adjusted phase angles in the network [4]. Using the SC-IPNN case study in this paper, we consider 1000 Monte Carlo iterations and inject random uncertainties in the phase angles of the unpruned SC-IPNN and those of the best-performing models obtained from the iterative baseline and the layer-wise LTH-based pruning methods. In each iteration, the uncertainties are sampled from a zero-mean Gaussian distribution with a standard deviation of $\sigma_{PS} \cdot \pi$. Fig. 7 shows the mean inferencing accuracy—over 1000 Monte Carlo iterations—for the three models for different values of σ_{PS}. We observe that while all three models are sensitive to uncertainties, the degradation in accuracy is slightly higher in sparse networks. The black-triangle line in Fig. 7 shows that the difference in the accuracy of the unpruned and the LTH-pruned model can be up to 11.3%. Note that in the absence of phase uncertainties ($\sigma_{PS} = 0$), the accuracy of the unpruned network is only 5% greater than the LTH-pruned model. Under uncertainties, the baseline-pruned model has slightly higher accuracy (0.4% on average) than the LTH-pruned model as it is less sparse. Mitigating uncertainties in SC-IPNNs often imposes high power-consumption overhead (e.g., when using tuning mechanisms [2], [3]). Fortunately, LTH-based pruning offers significant savings in power consumption, facilitating the deployment of uncertainty-mitigation methods in SC-IPNNs.

V. Conclusion

Pruning is challenging in SC-IPNNs because of the bidirectional many-to-one association between the edge weights of the linear layer and the phase angles. This paper is the first effort at hardware-aware pruning in SC-IPNNs to minimize their area overhead and tuning power consumption. We have presented two hardware-aware pruning methods, including a conventional magnitude-pruning-based approach for moderate sparsity (up to 22%) and ultra-low accuracy loss (<1%), and a novel pruning method based on the lottery ticket hypothesis to achieve ultra-high sparsity (up to 89%) with an acceptable accuracy loss (<5%). The insights derived from this paper pave the way for enabling advanced hardware-software-assisted design-optimization solutions for realizing compact and energy-efficient integrated photonic neural networks.

References

[1] F. Sunny *et al.*, "A survey on silicon photonics for deep learning," *ACM JETC*, vol. 17, no. 4, pp. 1–57, 2021.

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

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

[4] S. Banerjee *et al.*, "Modeling silicon-photonic neural networks under uncertainties," in *IEEE/ACM DATE*, 2021, pp. 98–101.

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

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

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

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

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

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

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

[12] I. A. Williamson *et al.*, "Reprogrammable electro-optic nonlinear activation functions for optical neural networks," *IEEE JSTQE*, vol. 26, no. 1, pp. 1–12, 2019.

[13] 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.

[14] T. M. Cover *et al.*, *Elements of Information Theory 2nd Edition*. Wiley-Interscience, 2006.