IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 June 2023, accepted 7 July 2023, date of publication 13 July 2023, date of current version 24 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3295344

RESEARCH ARTICLE

Sparsity-Aware Orthogonal Initialization of Deep Neural Networks

KIARA ESGUERRA[1], MUNEEB NASIR[1], TONG BOON TANG[1], (Senior Member, IEEE), AFIDALINA TUMIAN[2], AND ERIC TATT WEI HO[1]

[1]Department of Electrical and Electronics Engineering, Universiti Teknologi PETRONAS, Seri Iskandar, Perak 32610, Malaysia
[2]Petroliam Nasional Berhad (PETRONAS), Kuala Lumpur 50088, Malaysia

Corresponding author: Eric Tatt Wei Ho (hotattwei@utp.edu.my)

This work is supported by Yayasan Universiti Teknologi PETRONAS in collaboration with PETRONAS Research Sdn Bhd under grant 015LC0-403 and the APC was funded by the Petroleum Research Fund of Malaysia, grant number E.025.FOF.02021.0.14 awarded to PETRONAS Research Sdn Bhd and in collaboration with Universiti Teknologi PETRONAS.

ABSTRACT Deep neural networks have achieved impressive pattern recognition and generative abilities on complex tasks by developing larger and deeper models, which are increasingly costly to train and implement. There is in tandem interest to develop sparse versions of these powerful models by post-processing with weight pruning or dynamic sparse training. However, these processes require expensive train-prune-finetune cycles and compromise the trainability of very deep network configurations. We introduce sparsity-aware orthogonal initialization (SAO), a method to initialize sparse but maximally connected neural networks with orthogonal weights. SAO constructs a sparse network topology leveraging Ramanujan expander graphs to assure connectivity and assigns orthogonal weights to attain approximate dynamical isometry. Sparsity in SAO networks is tunable prior to model training. We compared SAO to fully-connected neural networks and demonstrated that SAO networks outperform magnitude pruning in very deep and sparse networks up to a thousand layers with fewer computations and training iterations. Convolutional neural networks are SAO networks with special constraints, while kernel pruning may be interpreted as tuning the SAO sparsity level. Within SAO framework, kernels may be pruned prior to model training based on a desired compression factor rather than post-training based on parameter-dependent heuristics. SAO is well-suited for applications with tight energy and computation budgets such as edge computing tasks, because it achieves sparse, trainable neural network models with fewer learnable parameters without requiring special layers, additional training, scaling, or regularization. The advantages of SAO networks are attributed to both its sparse but maximally connected topology and orthogonal weight initialization.

INDEX TERMS Sparse neural networks, dynamical isometry, Ramanujan expander graph, expander neural networks, model pruning, orthogonal neural networks.

I. INTRODUCTION

Deep neural networks (DNN) have demonstrated state-of-the-art performance in learning complex patterns in a variety of creative and choice-based tasks. A popularly accepted maxim is that task complexity can be managed by increasing the number of perceptrons (in width and depth) and the size of the training dataset. This idea emerged from the success of the deep CNN AlexNet in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which achieved significantly lower prediction error than the winners of the previous years, enabled by advancements in hardware that allowed efficient training of deep neural networks [1]. Due to this, the development of DNN design leans towards the construction of larger and deeper networks [2], [3], [4]. However, the continuously increasing size of DNNs to achieve higher accuracy drives up the computational and energy costs of model training and deployment [5], [6].

Model compression techniques are methods to reduce these costs without degrading the model performance [7]. For example, pruning removes insignificant weights from densely-connected neural network based on a significance

The associate editor coordinating the review of this manuscript and approving it for publication was Mu-Yen Chen.

criterion such as magnitude [8], [9] and dynamic sparse training seeks to simultaneously learn sparse connections and weights values together [10]. State-of-the-art model compression techniques can be computationally expensive, requiring a series of iterated train-prune cycles. Model compression may result in the loss of global connectivity because some neurons lose a connective path to other neurons [11]. Taken to the extreme, sparsification methods induce layer-collapse where entire layers are disconnected [12].

Another risk of removing connections is that it may destroy the dynamical isometry property of the network [13], [14]. This characteristic is essential to maintain stable training dynamics for very deep neural networks and confers benefits such as depth-independent learning rates [15] and successful training of networks with thousands of layers without specialized architectures like residual connections or techniques such as batch normalization [16]. Several theoretical studies support the importance of depth in neural network performance, attributing to it the increase in representation power [17], [18]. Early studies of deep convolutional neural networks show that deep networks outperformed shallow networks by a large margin [1] which subsequently spurred the success of deep convolutional neural networks in the ILSVRC challenge [19], [20], [21], [22], [23].

The performance of neural networks is influenced more by the graph expander-like properties of the topology rather than on the density of network connections and neurons [24], [25]. The benefits are attributed to the preservation of both local connectivity and global connectivity, where each layer is capable of sensing all of its inputs, and yet all information from the input reaches the output [11]. This is supported by studies which have incorporated expander properties in the formation of sparse neural networks such as X-Net [11] and RadiX-Net [26].

In this study, we combine the concepts of graph expander properties and dynamical isometry and introduce the method sparsity-aware orthogonal initialization (SAO), which allows the explicit construction of *a priori* sparsely connected deep neural networks. We accomplish this through the use of Ramanujan expander graphs to define the sparse connectivity of each neural layer whose weights are made orthogonal during initialization scheme. We say the orthogonal initialization is *aware* of the sparse topology of the network.

We were inspired to account for dynamical isometry in the construction of our sparse neural networks *a priori* to model training. Through SAO we can construct very sparse but maximally connected neural networks without needing to first train and adapt a dense network. SAO can also approach dynamical isometry at initialization without iterative training or regularization thus making sparse SAO instantiations of a network trainable up to very deep configurations. We investigated SAO on fully-connected and convolutional neural networks of various sizes and depths and report their advantages over the equivalent dense network baseline and sparse versions generated with various magnitude pruning configurations.

II. RELATED LITERATURE

A. EXPANDER GRAPHS AND SPARSE NEURAL NETWORKS

Several studies have incorporated the properties of expander graphs, which are sparse graphs with good connectivity, in the construction of sparse neural networks [11], [24], [26]. Since the layers are built to be sparse and not recovered from a dense model, these techniques save on the computational cost of training. The X-Net architecture [11] applied random d-regular bipartite expander graphs generated from Cayley graphs to form sparse layer connections with symmetric input and output sizes. They preserved connectivity in the model by ensuring that every input was connected to an output and by making the number of edges to scale proportionally to the product of the sizes of input and output vertices. X-Net achieved 4% higher accuracy on MobileNet than group convolutions for a given level of sparsity. RadiX-Net [26] generalizes X-Net for unequal input and output sizes by leveraging the concept of mixed-radix numeral systems. The trainability of sparse neural networks based on Cayley graphs is not explicitly addressed in the above-cited studies.

B. PRUNING DAMAGES DYNAMICAL ISOMETRY AND DEGRADES NEURAL NETWORK PERFORMANCE

Pruning weights in neural networks leads to loss of dynamical isometry, and this loss contributes to the degradation of accuracy by hindering gradient error and signal propagation during training [13], [14], [27], [28]. We can gain an intuitive understanding of how pruning destroys dynamical isometry by taking an orthogonal weight matrix, zeroing out some of the entries, and then finding that the gram matrix of $\mathbf{W}$ is no longer an identity matrix. This happens because the orthogonalization precedes the sparse structure instead of constructing specific sparse structures first and then assigning the appropriate values to achieve orthogonality [29].

Several studies have presented methods to recover dynamical isometry in pruned models [13], [14], [28]. Wang et al. [14] proposed finetuning the pruned model using an orthogonalized version of its weight matrix obtained with QR decomposition. The recovered neural network outperforms the pruned model, and the speed of recovery increased with the learning rate. Lee et al. [13] trained the pruned model with regularization of the weight matrix by optimizing $\min_{\mathbf{W}^l} ||(\mathbf{C}^l \odot \mathbf{W}^l)^\top (\mathbf{C}^l \odot \mathbf{W}^l) - \mathbf{I}^l||_F$, where $(\mathbf{C}^l \odot \mathbf{W}^l)$ is the pruned weight matrix. They found that layerwise dynamical isometry provides better results with models pruned at initialization as it improves connection sensitivity which is used as the saliency criterion. Orthogonality preserving pruning (OPP) [28] which adds a penalty term for the gram matrix $\mathbf{G} = \mathbf{W}\mathbf{W}^\top$ of the weights attempts to orthogonalize "important" filters and drive the rest to zero. They also penalized the learnable scale and offset parameters of batch normalization layers so that activations from pruned filters would not be propagated to deeper layers and harm dynamical isometry. These methods employ regularization and additional training on the pruned model, which introduces additional computation.

To preserve dynamical isometry in convolutional neural networks, Xiao et al. [16] proposed an orthogonal initialization algorithm appropriate for the block-circulant filtering matrix based on orthogonal wavelets. They dubbed the method Delta-Orthogonal Initialization and successfully trained a 10,000-layer vanilla convolutional neural network. Sedghi et al. [30] demonstrated how to explicitly compute the singular values of the convolutional weight matrix. They expressed the multichannel convolution as an operator defined by a doubly block-circulant matrix and showed how to derive its singular values from the 2D Fourier Transforms of the circulant matrices. Their work laid the foundation for explicitly constructed orthogonal convolutions [31], which we apply in this study.

III. PRELIMINARIES

Our work attempts to integrate both sparse weight connections and dynamical isometry into an *a priori* sparse neural network topology for both fully-connected and convolutional neural networks without introducing special layers. We aim to circumvent the difficulties associated with training sparsified models to convergence. In this section, we define the fully-connected deep neural network and convolutional deep neural network architectures, which serve as the basis for our experiments. Next, we describe the orthogonal weight initialization in the context of fully-connected and convolutional neural networks since we will later adopt orthogonal initialization to our sparse topology to preserve trainability. Finally, we describe several variants of magnitude pruning, which will be benchmarked against our method, sparsity-aware orthogonal initialization (SAO).

A. NOTATION

Variables are denoted by italicized letters n, vectors are denoted by bold small letters $\mathbf{v}$, matrices by bold capital letters $\mathbf{M}$, and tensors by bold calligraphic letters $\boldsymbol{\mathcal{T}}$. The weight matrix of a fully-connected layer is denoted by $\mathbf{W}_{fc} \in \mathbb{R}^{n_{out} \times n_{in}}$ and the weight tensor of a convolutional layer by $\boldsymbol{\mathcal{W}}_{conv} \in \mathbb{R}^{k \times k \times c_{out} \times c_{in}}$, where $\boldsymbol{\mathcal{W}}_{conv}[:, :, s, t]$ denotes the kernel sensing the t^{th} input for the s^{th} output channel. As we will deal with sparse matrices, let $\mathbf{M} \in \{0, 1\}^{n_{out} \times n_{in}}$ denote the pruning mask for a fully-connected layer, where $\mathbf{M}[i, j] = 1$ denotes the connection between the j^{th} input node the i^{th} output node, where $i \in [1, m]$ and $j \in [1, n]$. $\text{DFT}_{2D}(\cdot)$ denotes the 2D discrete Fourier transform (DFT) while $\text{IDFT}_{2D}(\cdot)$ denotes the inverse 2D DFT. The Jacobian of the weight matrix is denoted by $\mathbf{J}$ while $\sigma(\cdot)$ denotes the singular values of a matrix.

B. DYNAMICAL ISOMETRY IN NEURAL NETWORKS

Neural networks attain dynamical isometry when all singular values of the input-output or end-to-end Jacobian matrix $\mathbf{J}$ are unity, corresponding to factorizing the Jacobian into an orthogonal transformation via singular value decomposition. The input-output Jacobian matrix of a fully-connected layer is given by the weight matrix $\mathbf{W}$, while for convolutional layers, it is given by the associated doubly-block circulant matrix of all the kernels [32]. Given a function $f : \mathbb{R}^n \to \mathbb{R}^m$, the Jacobian matrix $\mathbf{J}$ of f is an $m \times n$ matrix defined by [33]:

$$\mathbf{J}_{i,j} = \frac{\partial f_i}{\partial x_j} \tag{1}$$

where $\mathbf{J}_{i,j}$ is the $(i, j)^{th}$ element of $\mathbf{J}$, and $\frac{\partial f_i}{\partial x_j}$ is the partial derivative of the i^{th} component of f with respect to the j^{th} variable. In other words, the Jacobian matrix $\mathbf{J}$ expresses the local linear approximation of f near a point x as a matrix transformation. Specifically, the Jacobian matrix $\mathbf{J}$ at a point x is given by:

$$\mathbf{J}(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{2}$$

The Jacobian matrix for a deep neural network with L layers with input $\mathbf{x_0} \in \mathbb{R}^n$ as $\mathbf{J} \in \mathbb{R}^{m \times n}$ may also be defined as follows [34]:

$$\mathbf{J} = \frac{x^L}{x^0} = \prod_{l=1}^{L} \mathbf{D}^l \mathbf{W}^l \tag{3}$$

where $\mathbf{D}^l_{ij} = \phi'(h^l_i)\delta_{ij}$ and δ_{ij} is the Kronecker delta function and $\mathbf{W}^l$ is the weight matrix of the l^{th} layer in the neural network. The singular values of $\mathbf{J}$ are computed from the singular value decomposition:

$$U, \Sigma, V^* = \text{SVD}(\mathbf{J}) \tag{4}$$

where the singular values are the diagonal elements of $\Sigma \in m \times n$ and $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are complex unitary matrices representing rotations or reflections.

For linear neural networks, dynamical isometry is perfectly attained through orthogonal weight initialization. Non-linear activation functions compromise dynamical isometry, but it was empirically observed that employing orthogonal weight initialization [13], [15] may still produce a well-conditioned Jacobian sufficient to mitigate vanishing and exploding gradients [35] when the singular values of the Jacobian closely approximate unity. ReLU networks do not attain *approximate dynamical isometry* with simple orthogonal weight initialization [34] but possible with shifted and smoothed variants [35].

C. FULLY-CONNECTED DEEP NEURAL NETWORKS

We benchmark against fully-connected neural network (FNN), where N specifies the number of layers between the input and the final output activations. We use L_0 to denote the 512×256 layer while L_1 and L_{N-1} denotes the first and last symmetric layers, respectively. We trained FNN models with several depths ranging from $N = 50$ to $N = 1000$ on MNIST with Stochastic Gradient Descent (SGD) with momentum $= 0.9$ and weight decay $= 10^{-4}$. We scaled the magnitude of the weights from all networks by a factor of 1.1.

TABLE 1. Architecture of fully connected neural network (FNN) with a selectable number of layers parameterized by *N*.

	Dimensions	Repeats
Input Layer	1024×512	1
Downscaling	512×256	1
Symmetric	256×256	N-1
Output Layer	256×10	1

TABLE 2. Learning rates are gradually decreased with increasing depth of Fully-connected neural networks (FNN).

Depth (N Layers)	60	200	400	600	800	1000
Learning Rate	10^{-2}	10^{-3}	10^{-3}	10^{-4}	10^{-4}	10^{-4}

TABLE 3. Learning rates are gradually decreased with increasing depth of Convolutional neural networks (CNN) for CIFAR-10.

Depth (N Layers)	32	128	256	512	768
Learning Rate	10^{-2}	10^{-3}	10^{-3}	10^{-4}	10^{-4}

The learning rates used are listed in Table 2 and are chosen to encourage the sparse network to train to convergence.

1) ORTHOGONAL WEIGHT INITIALIZATION FOR FULLY-CONNECTED DEEP NEURAL NETWORKS

The Jacobian of the weight matrix, $\mathbf{J}$ of a fully-connected layer is given by the transpose of the weight matrix $\mathbf{W}_{fc}$, such that orthogonalizing $\mathbf{W}_{fc}$ results in an orthogonal $\mathbf{J}$. [15].

D. CONVOLUTIONAL NEURAL NETWORKS

1) VANILLA CNN

We implemented a vanilla convolutional neural network which does not use any special layers or techniques such as batch normalization, residual connections and dropout to assist the training of the network and only consists of convolutional layers interleaved with either Tanh or ReLU activation function. All the layers use a kernel size of 3×3, zero-padding, and stride equal to 1, except for the second and third convolutional layers with stride = 2. Before the linear output layer, an average pooling layer is used. We trained models with $N = 32, 128, 256, 512$, and 768 convolutional layers on CIFAR-10 with SGD (momentum = 0.9, weight decay = 10^{-4}) and without weight rescaling. At a specific network depth, we experimented with layers with widths of $W = 32, 64, 128$ and 256. The initial learning rates are listed in Table 2 and are decayed by a factor of 0.1 at the 50^{th} and 150^{th} epochs. A learning rate warm-up (10 epochs) was used at $N = 768$ for ReLU and $N = 512$ and above for Tanh with Cosine Annealing. Deep vanilla CNNs are trainable to convergence only with orthogonal weight initialization, described in Section III-B above.

For CINIC-10 and CIFAR-100, the average pooling layer before the classifier was replaced with two convolutional layers with stride = 2. The same hyperparameters were used, except for the adoption of the Cosine Annealing scheduler with warm-up for 1 epoch for all values of N. A selection from networks with $N = 8, 16, 32$ and 128 were trained on CINIC-10, depending on the orthogonalization technique, with the same learning rates as Table 3 and 10^{-2} for $N < 32$. For CIFAR-100, a selection from networks with $N = 8, 16, 32$ were trained with the same learning rates. Only ReLU was used for these datasets. For all of the experiments, the training set was augmented using random crop, random horizontal flip, and AutoAugment [36].

TABLE 4. Architecture of LipConvNet [31] parameterized by depth, *N*.

	Input Size	Output Size	Convolution	Repeats
Block 1	24×24	12×12	conv[32, 3×3, 1]	(N/5 - 1)
			conv[64, 3×3, 2]	1
Block 2	12×12	6×6	conv[64, 3×3, 1]	(N/5 - 1)
			conv[128, 3×3, 2]	1
Block 3	6×6	3×3	conv[128, 3×3, 1]	(N/5 - 1)
			conv[256, 3×3, 2]	1
Block 4	3×3	1×1	conv[256, 3×3, 1]	(N/5 - 1)
			conv[512, 3×3, 3]	1
Block 5	1×1	1×1	conv[512, 1×1, 1]	(N/5 - 1)
			conv[1024, 1×1, 1]	1

2) LipConvNet-N

We adopt the 1-Lipschitz compliant network from [31] to apply an alternative weight orthogonalization method for CNNs. The complete network comprises of five blocks. Each block consists of a single-strided convolutional layer repeated $(N/5-1)$ times followed by a convolutional layer with stride, $s \neq 1$ to down-sample the feature space. The Max-Min activation function is applied after every convolutional layer. The first two blocks are initialized with Skew-Orthogonal Convolutions (SOC) from [37] while the last three blocks are initialized with Explicitly-Constructed Orthogonal Initialization (ECO) - see Section III-C.5 below. This construction yields better performance than purely ECO initialized layers as shown by [31] as dilated convolutions early on leads to deterioration of recognition capabilities. We trained the LipConvNet with Stochastic Gradient Descent for 200 epochs with a learning rate of 0.01 which is decayed by a factor of 0.1 in the 50^{th} and 150^{th} epochs, and weight decay 5×10^{-4}.

3) DELTA-ORTHOGONAL INITIALIZATION FOR CNNs

Delta-Orthogonal Initialization for convolutional neural networks enabled training of a 10,000 layer CNN without special layers like batch normalization and residual connections [16]. The initialization algorithm is summarized in Table 5. Two-dimensional kernels are initialized with a single, non-zero parameter at its center. The construction was adopted from the literature on generating block-circulant, random orthogonal matrices and is specific to CNNs with sigmoidal activation functions, *i.e* Tanh.

The same algorithm can be applied to CNNs with the ReLU activation function, only that the orthonormal matrix $\mathbf{H}$ is

TABLE 5. Delta-Orthogonal Initialization algorithm for CNN kernels.

Delta-Orthogonal Initialization for 2D CNN kernels
Require: k kernel size, c_{in} and c_{out} input and output channels **Return:** a $k \times k \times c_{in} \times c_{out}$ weight tensor $\mathcal{W}_{conv}$ **Step 1.** Randomly generate a $c_{in} \times c_{out}$ matrix $\mathbf{H}$ with orthonormal rows **Step 2.** Define a $k \times k \times c_{in} \times c_{out}$ tensor $\mathcal{W}_{conv}$ in the following way: for β, β' in $0, 1, ..., k-1, j = 0, ..., c_{out} - 1$, set $\mathrm{W}_{conv}(\beta, \beta', i, j) = \begin{cases} \mathbf{H}(i,j) & \text{if } \beta, \beta' = [k/2] \\ 0 & \text{otherwise} \end{cases}$

given by:

$$\mathbf{H}^l = \begin{bmatrix} \mathbf{H_0} & -\mathbf{H_0} \\ -\mathbf{H_0} & \mathbf{H_0} \end{bmatrix} \tag{5}$$

where $\mathbf{H_0}^l \in \mathbb{R}^{\frac{N_{l-1}}{2} \times \frac{N_l}{2}}$ is an orthonormal matrix [38].

4) EXPLICITLY-CONSTRUCTED ORTHOGONAL INITIALIZATION FOR CNNs

Orthogonal convolutional weights may be achieved by constraining all singular values of the Jacobian weight matrix to unity. Rather than explicitly orthogonalizing the block circulant weight matrix, orthogonality was imposed by initializing matrices with all-unity singular values in the Fourier domain and subsequently constructing the orthogonal weight matrix using the inverse Fourier Transform [30].

For an expanded convolution kernel $\mathcal{W}_{conv} \in \mathbb{R}^{n \times n \times c \times c}$, for each $p, q \in [n] \times [n]$, let $\mathcal{P}^{(p,q)}$ be the $c \times c$ matrix computed by:

$$\mathcal{P}^{(p,q)}[s,t] = (F_n^\top \mathcal{W}[:,:,s,t] F_n)_{p,q}, \forall s, t \in [c] \times [c] \tag{6}$$

Orthogonal convolution kernels may be obtained according to the following process:

1) Construct the orthogonal matrices $\mathcal{P}[p,q,:,:] = (\mathrm{DFT}_{2\mathrm{D}}\mathcal{W})[p,q,:,:]$
2) Recover $\mathcal{W}[:,:,s,t]$ through $\mathrm{IDFT}_{2\mathrm{D}}\mathcal{P}[:,:,s,t]$
3) Perform the dilated convolution with $\mathcal{W}$ with dilation n/k

E. NEURAL NETWORK SPARSITY THROUGH WEIGHT PRUNING

We benchmark the performance of sparse networks with several variants of pruning, namely: global magnitude pruning (GMP), local magnitude pruning (LMP), and local random pruning (LRP). In fully-connected neural networks, individual weights are pruned. In convolutional neural networks, two-dimensional kernels are removed en bloc. Fig. 1 illustrates the process of creating sparse neural networks with weight pruning applied after training a dense network, indicated by a suffix "T", *e.g.* LMP-T indicates performing local magnitude pruning on a trained dense network which is then finetuned for the same duration as in training. The suffix "S" implies that upon initialization, the model is immediately pruned without training, *e.g.* GMP-S implies global magnitude pruning performed on initialized weights prior to network training, such that the pruned model undergoes only half the total training epochs of GMP-T from initialization. Pruning was performed after orthogonal initialization but prior to network training in the spirit of the Lottery Ticket Hypothesis [39].

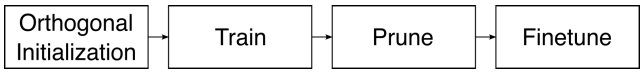

FIGURE 1. Creating a sparse neural network by weight pruning involves a four-step process (denoted by suffix -T). Models are also initialized as sparse according to Lottery Ticket Hypothesis [39] and the Train step prior to Pruning is omitted (denoted by suffix -S).

IV. METHODS

In this section, we formulate our SAO method and describe how we compare the performance of SAO construction to using conventional pruning methods when introducing sparsity to fully-connected and convolutional neural networks. We apply sparsity-aware orthogonal initialization (SAO) by forming a sparse, orthogonal matrix $\mathbf{S}$ and then using this as the orthogonal matrix from which the weight matrix will be derived from to attain orthogonal transformations, distinct for fully-connected neural networks and two different methods, ECO and Delta-Orthogonal Initialization, for convolutional neural networks. Forming the $\mathbf{S} \in \mathbb{R}^{m \times n}$ comprises two key steps: generating the sparse, Ramanujan binary matrix $\mathbf{M} \in \{0,1\}^{m \times n}$ and orthogonalizing $\mathbf{M}$ to generate the matrix $\mathbf{S} \in \mathbb{R}^{m \times n}$. The sparse structure comes first before the values are assigned, such that the orthogonal initialization is *aware* of the sparsity.

The sparsity of individual neural layers is controlled by setting the connectivity parameter of each neuron, d. Parameter d is equivalently the number of non-zero weighted connections emerging from each neuron to the next layer.

To understand if the advantages of SAO are due to topology alone or co-dependent on the dynamical isometry property, we also compare against neural networks initialized with only sparse Ramanujan topology (indicated with RG). RG uses the same sparse topology as SAO but without orthogonal weight initialization. We evaluated the RG layers with different weight initialization schemes: random normal (RG-N), random-uniform (RG-U), and random orthogonal weights (SAO).

A. SPARSE NEURAL LAYERS WITH RAMANUJAN EXPANDER GRAPH TOPOLOGY

In this work, we leverage (c, d)-regular bipartite graph expanders, *i.e.*, graphs whose input and output nodes have degree c and d, to form the sparse structure of the neural networks layers as in [11], but we use a specific type of expander graphs called Ramanujan graphs which are maximally sparse.

Each layer of a neural network may be represented as a bipartite graph. Bipartite graphs $G(U, V, E)$ are graphs with two disjoint sets of nodes U and V and edges E that connect nodes between but not within sets. Thus, U and V

are the neural input and output activations, respectively. The biadjacency matrix, $\mathbf{B} \in \{0,1\}^{u \times v}$ represents the connections between input and output neurons of a layer. Each entry of the matrix indicates the presence (1) or absence of (0) of an edge between nodes from set U and set V. The biadjacency matrix is the matrix transpose of the binary mask, $\mathbf{M}$, which describes the topology of the neural layer.

The full graph adjacency matrix [40] of each layer is defined as:

$$\mathbf{A} = \begin{bmatrix} 0_{u,u} & \mathbf{B} \\ \mathbf{B}^\top & 0_{v,v} \end{bmatrix} \tag{7}$$

All nodes have the same number of edges, d in a d-regular bipartite graph. For d-regular bipartite graphs to be expanders [11], the eigenvalues of the adjacency matrix, $\mathbf{A}$ obey $|\lambda_1 - \lambda_2| \leq 1 - \lambda_2/d$, where λ_1 and λ_2 are the largest eigenvalues. A graph is a Ramanujan expander if λ_2 also satisfies $\lambda_2 \leq 2\sqrt{d-1}$. For asymmetric neural layers where c and d are the degree of the input and output nodes, this condition generalizes to $\lambda_2 \leq \sqrt{c-1} + \sqrt{d-1}$.

We construct the mask $\mathbf{M}$ to be the transpose of a biadjacency matrix describing a (c,d)-regular graph that is a block matrix comprised of identical blocks or submatrices. We control the degree which corresponds to the smaller dimension of the graph by specifying the degree c if $m > n$, where $d = \frac{cn}{m}$ and d if $m < n$, where $c = \frac{dm}{n}$. With an equal number of input and output nodes, the graph simplifies to a d-regular graph.

We first construct the block $\mathbf{M}_1$ which comprises $\mathbf{M}$. For $m > n$, we construct the $(c,1)$-regular block matrix $\mathbf{M}_1 \in \{0,1\}^{m \times \frac{m}{c}}$ which is copied $\frac{cn}{m} - 1$ times and then concatenated in the vertical axis to get $\mathbf{M} = [\mathbf{M}_1 \mathbf{M}_2 \dots \mathbf{M}_{\frac{cn}{m}}]$. For $m < n$, we construct the $(1,d)$-regular block matrix $\mathbf{M}_1 \in \{0,1\}^{\frac{n}{d} \times n}$ which is copied $\frac{dm}{n} - 1$ times and then concatenated in the horizontal axis to get $\mathbf{M} = [\mathbf{M}_1; \mathbf{M}_2; \dots \mathbf{M}_{\frac{dm}{n}}]$. The number of blocks in $\mathbf{M}$ will be referred to as r in the following discussions. For $m = n$, any of these two construction methods can be used. The sparse, Ramanujan matrices that can be generated through this process has two constraints: the larger dimension between n and m should be divisible by the specified degree, and the ratio of n and m should be equal to the degree divided by a number $r \in \mathbb{Z}$ to guarantee a degree of at least 1 in the larger dimension.

Convolutional neural networks are (c,d)-regular by definition where the local $k \times k$ kernel are the (c,d)-sparse connections. Unlike fully connected networks, we modulate the sparsity of convolutional neural networks in increments of the kernel size $k \times k$, which is equivalent to pruning entire kernels in a channel. We illustrate this in Fig. 2, where $m = n$.

B. VISUALIZING INPUT-OUTPUT REACHABILITY OF SPARSE LAYERS

Sparsity in neural layers may restrict inputs from contributing to neural processing in deeper layers. It could compromise neural network accuracy by excluding sections of the input from local neural decision-making or by limiting the trainability of networks by constraining the paths for gradient error propagation to individual weights. We introduce a method to visualize the influence of sparse topologies on connectivity of information and gradient error propagation called reachability. The reachability matrix $\mathbf{R}_{i \to j} \in \mathbb{R}^{m_i \times n_i}$ is acquired through $(\prod_{i=i}^{j} \mathbf{M}_i^\top) \in F_2$ and we binarize the matrix, $\mathbf{M}$. When $\mathbf{R}_{i \to j}[p,q] = 1$, the p^{th} input node in L_i is connected to the q^{th} output node in L_j. Through $\mathbf{R}$, we can determine if all the inputs may reach and contribute to decision-making at all of the output nodes.

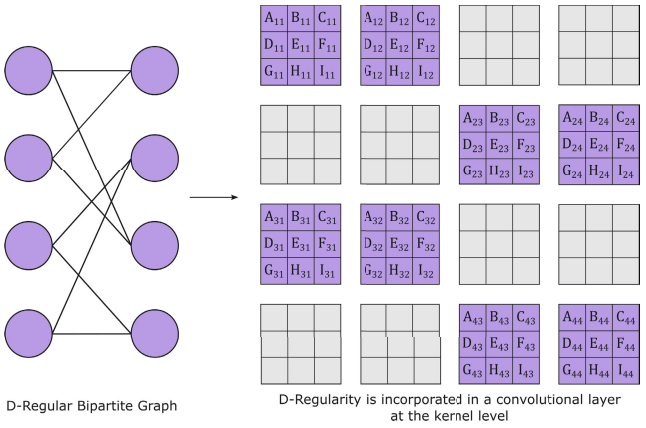

FIGURE 2. Neural networks can adopt d-regular bipartite graph connectivity (left) and in convolutional layers, this appears like kernel-level sparsity (right).

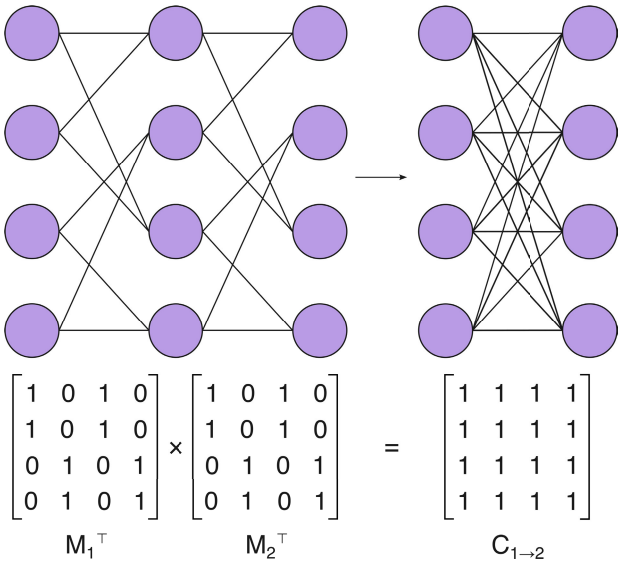

FIGURE 3. An example computation of the reachability matrix for two sparse layers based on 2-regular bipartite graphs illustrates that all inputs nodes are still able to influence all output nodes despite the regular sparse structure.

C. DYNAMICAL ISOMETRY FOR SPARSE SAO LAYERS

We construct the sparse-orthogonal matrix $\mathbf{S}$ by processing the mask matrix, $\mathbf{M}$ to have mutually orthogonal row vectors. This produces an orthogonal matrix for $m = n$, and a semi-orthogonal matrix when $m \neq n$. We generate $min\{m,n\}/r$ sets of orthogonal vectors where each set

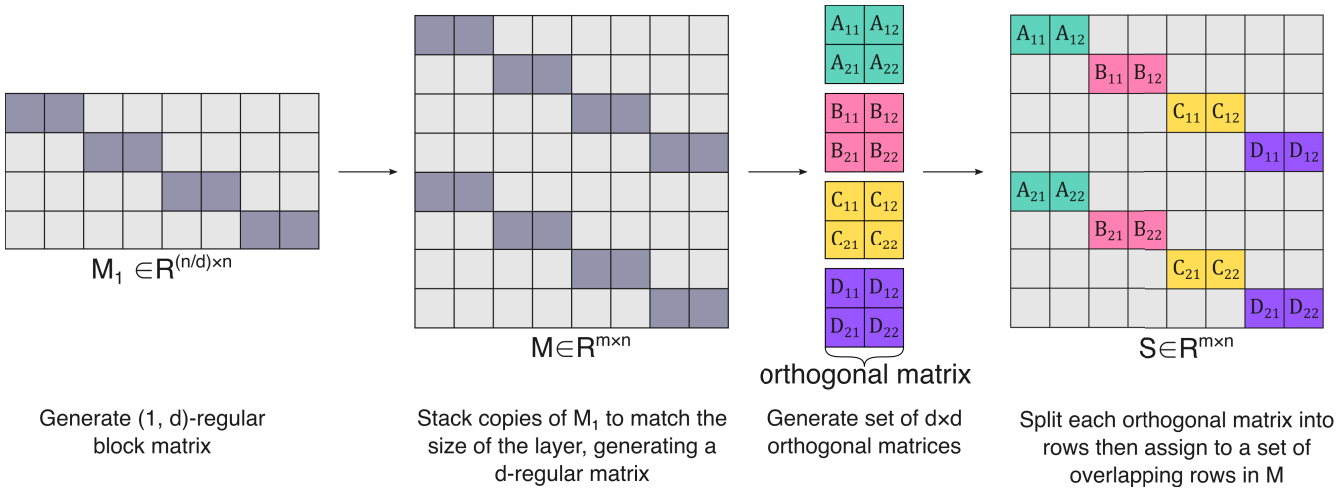

FIGURE 4. **This illustration depicts how a sparse-orthogonal matrix of size 8 × 8 based on 2-regular expander graph: i) an 8 × 4 binary block matrix is first generated and ii) duplicated to form the mask matrix M, where the shaded and unshaded pixels represent 1 and 0 respectively. iii) Four 2 × 2 orthogonal matrices are generated and iv) the rows of the orthogonal matrices are assigned to the block matrix, M.**

has r number of vectors with length equal to the specified degree. One set of orthogonal vectors correspond to one set of identical columns ($m > n$) or rows ($m < n$). The values of each vector are then assigned to the non-zeros of each row or column for each set. Fig. 4 illustrates the process of generating an $\mathbf{S} \in \mathbb{R}^{8\times 8}$ matrix with $c = d = 2$.

When SAO is applied to fully-connected layers, $\mathbf{S} \in \mathbb{R}^{m\times n}$ is simply assigned as the weight matrix. The density of a (c, d)-regular layer is given by $c/m = d/n$, where c and d refer to the degree of the input and output nodes, respectively.

We compared the performance of SAO to the baseline fully-connected model and variants pruned with one-shot magnitude global pruning with and without pretraining (GMP-T and GMP-S), local random pruning (LMP) as well as networks with *a priori* sparse Ramanujan construction with random uniform and Gaussian initialization (RG-U and RG-N).

SAO was applied to convolutional neural networks by adapting two methods for orthogonal weights initialization (i) Delta-Orthogonal Initialization (SAO-Delta) and (ii) Explicitly Constructed Orthogonal Initialization (SAO-ECO).

With the SAO-Delta scheme, for a convolutional layer with weight tensor $\mathcal{W} \in \mathbb{R}^{k\times k\times c_{in}\times c_{out}}$, we generate the sparse orthogonal matrix $\mathbf{S} \in \mathbb{R}^{c_{out}\times c_{in}}$ and then assign $\mathbf{S}[s, t]$ to $\mathcal{W}_{conv}[[\frac{k}{2}], [\frac{k}{2}], s, t]$ $\forall s, t$, as illustrated in Fig. 5. All non-center weights are zeroed at initialization but the weights are not removed. The weights initialized to zero can assume nonzero values during training as illustrated in Fig. 5. Sparsity in SAO-Delta is controlled by removing entire kernels, corresponding to kernel-level pruning.

With the SAO-ECO Scheme, all convolutional layers are constructed with the sparse orthogonal matrix, $\mathbf{S} \in \mathbb{R}^{c_{out}\times c_{in}}$ assigned to the Fourier domain kernel $\mathcal{P}[p, q, :, :]$. To keep the $\mathcal{W}_{conv}$ real, $\mathcal{P}[i, j, :, :] := \mathcal{P}[(k - i)\%k, (k - j)\%k, :, :]$ for each $(i, j) \in [k] \times [k]$ [31]. Due to this constraint, only L unique orthogonal matrices are generated, where $L = (k^2 + 1)/2$ if k is odd and $L = (k^2 + 4)/2$ if k is even. Sparsity in SAO is controlled by removing kernels, which corresponds to kernel-level pruning.

For layers where $c_{in} < c_{out}$, we set $\mathcal{W}_{conv} \in \mathbb{R}^{k\times k\times c_{out}\times c_{out}}$ then pad the input with zeros in the channel dimension to match the number of output channels as

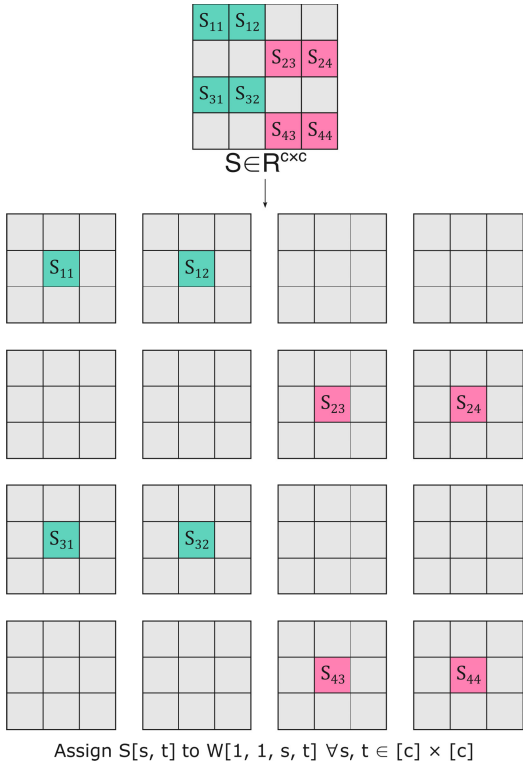

FIGURE 5. **This illustration depicts the SAO-Delta initialization of convolutional neural network weights with 3 × 3 kernels and 16 channels, are populated from a sparse orthogonal matrix, S.**

in [31]. For $c_{in} > c_{out}$, we perform the convolution with $k \times k \times c_{in} \times c_{in}$ then only select the first c_{out} channels. For the final block with input $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n \times n \times c_{in}}$, we apply dilation n/k and cyclic padding $d(k-1)/2$ on each side of the input as in [31]. When the stride s is greater than 1, we reshape the input tensor into $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{\frac{n}{s} \times \frac{n}{s} \times c_{in}s^2}$ and perform the convolution with $\boldsymbol{\mathcal{W}}_{conv} \in \mathbb{R}^{k \times k \times c_{in}s^2 \times c_{out}}$ with $s = 1$ [31], [41].

SAO-Delta and SAO-ECO CNNs are benchmarked against vanilla CNN with Delta-Orthogonal and ECO weight initialization and versions pruned using RG-S, LRP, LMP-T and LMP-S. The first convolutional and linear layers are excluded from pruning.

V. RESULTS AND DISCUSSION

Deep neural networks (DNN) are conventionally constructed with many more neurons and higher representation capacity than what is necessary for learning complex patterns in data. Neural network models are designed with excess neural capacity to remain trainable by gradient learning which enables convergence to a high-accuracy model. Oftentimes, the number of neurons can be reduced by almost a factor of 10 without noticeable degradation in performance based on longstanding results from network compression, pruning, and transfer learning methods.

It remains an open problem to determine the optimal topology and neural capacity of a DNN for a given task and training dataset, though there is emerging interest to develop DNNs that use computational resources efficiently. Optimal capacity DNN models consume less energy to deploy and develop, and therefore support climate sustainability goals. Optimal-capacity models are also more amenable to deployments in edge and mobile devices with limited compute resources without compromising inference latency.

Sparse DNN topologies are difficult to train from scratch as these do not easily or robustly converge to accurate models presumably due to reduced flexibility in evolving feature representations during gradient learning. Sparse topologies may impede the forward and backpropagation of error gradients during gradient learning. There are no known sparse topologies and learning algorithms that converge to accurate models as robustly as equivalent DNN models with overcapacity. Model reduction can still be achieved using repeated learning cycles to perturb the network to discover gradually sparser topologies with algorithms like weight pruning and dynamic sparse training at the cost of additional computations and learning time.

Our work investigates the feasibility of defining a sparse DNN topology *a priori* to gradient learning which generates robust convergence to an accurate model. We propose to adopt expander graph topologies to predefine sparse connectivity between neural layers. We complement the *a priori* topology with weight initialization to create dynamical isometry that mitigates vanishing and exploding gradients across all network layers.

A. SPARSE AWARE ORTHOGONALIZATION (SAO): A SPARSE TOPOLOGY WITH DYNAMICAL ISOMETRY FROM RAMANUJAN EXPANDER GRAPHS

Expander graphs are a type of graph topology with maximally connected and sparse nodes. Graphs possess the expander property if the topology fulfills the edge expansion, vertex expansion, or spectral expansion criteria. Generating expander graph topologies are more challenging, but several construction algorithms are known for selected expander graph families. Since expander graphs are maximally connected, we reason that this type of topology would not unduly impede the forward and backward propagation of gradients between neurons while the sparsity property achieves the fewest weights to connect layers.

Ramanujan graphs are a family of expander graphs with maximally sparse connectivity and are described as the optimal expanders [42]. We expect that Ramanujan topology would consist of the fewest edges (and hence the fewest number of weights in a neural layer) as compared to other expander graph families. Ramanujan graphs satisfy the spectral expansion condition $\lambda_2 \leq 2\sqrt{\lambda_1 - 1}$ where λ_1 is the largest eigenvalue of the binary adjancency matrix and λ_2 the second largest eigenvalue [43].

We designed the topology of each neural network layer to be a bipartite Ramanujan graph, achieved by constructing $\mathbf{M}$ as a (c, d)-regular block matrix with identical blocks as discussed in Section IV-A. We can prove that $\mathbf{M}$ is always Ramanujan through properties inherent to bipartite graphs and those which are specific to the construction.

The eigenvalues of an adjacency matrix $\mathbf{A}$ of a bipartite graph comes in pairs λ and $-\lambda$ where $|\lambda|$ is a singular value of the biadjacency matrix $\mathbf{B}$ [40]. To check for the Ramanujan property, we can evaluate the singular values of $\mathbf{B}$, which are equal to that of $\mathbf{M}$ as its transpose. Thus, we take interest in the singular values of $\mathbf{M}$ as it is the structure directly controlled in SAO.

We conjecture that $\mathbf{M}$, through elementary matrix operations, *i.e.* row and column interchanges, can be transformed into a block diagonal matrix $\mathbf{M}_{bd}$ comprised of blocks with dimension $c \times d$ with entries all equal to 1. For instance, $\mathbf{M}$ in Figure 4 can be transformed into a block diagonal matrix through row interchanges: $r_1 \leftrightarrow r_5$, $r_4 \leftrightarrow r_7$, $r_3 \leftrightarrow r_6$, and $r_4 \leftrightarrow r_5$. This allows ease of analysis of the singular values, especially as the blocks comprising its diagonal are the same and therefor have the same singular values.

As all the entries of the block are equal to 1, it has rank 1 (all the rows and columns are linearly dependent) [44], and Frobenius norm equal to $\sqrt{cd}$. Meanwhile, the L_2 norm of the singular values of a matrix is equal to the Fobrenius norm [45], and since each block has only one non-zero singular value as indicate by the rank [46], it should be equal to $\sqrt{cd}$. Then, all of the blocks comprising $\mathbf{M}_{bd}$ have singular values equal to $\sqrt{cd}$. By taking the union of these singular values, we deduce that the singular values of $\mathbf{M}_{bd}$, and consequently $\mathbf{M}$, consist only of $\sqrt{cd}$ and zero [44]. The eigenvalues of $\mathbf{A}$ are then comprised only of $\sqrt{cd}$, $-\sqrt{cd}$, and

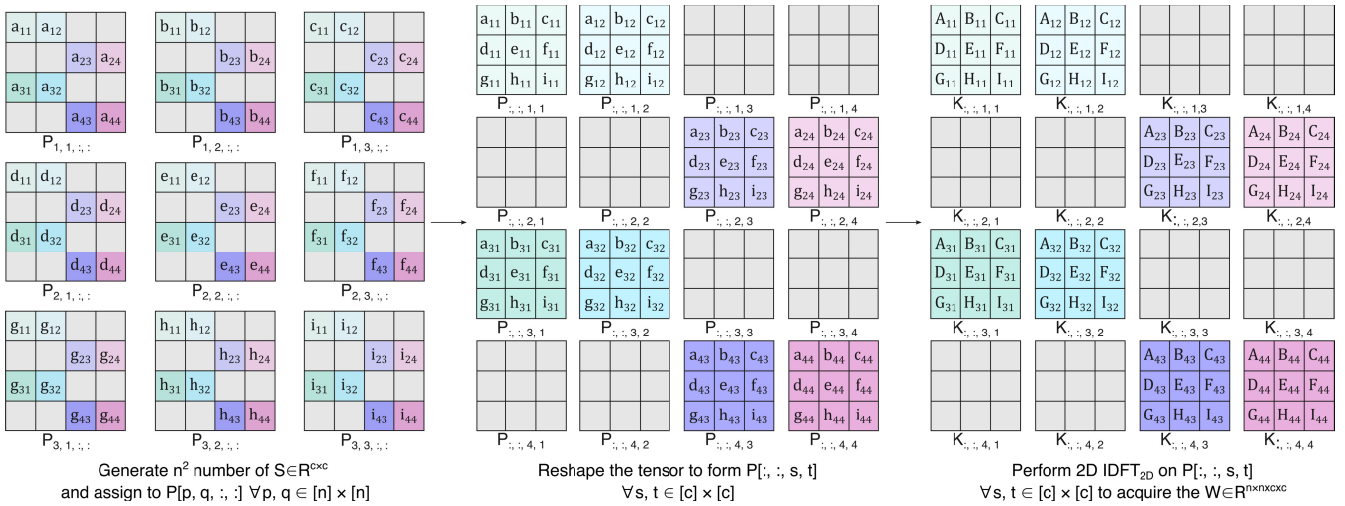

FIGURE 6. This illustration depicts the SAO-ECO initialization of convolutional neural network weights with 3 × 3 kernels and 16 channels. Four sparse matrices are generated and populated as sparse samples of the Fourier transform of the weights, $\mathcal{P}[p, q, :, :]$ $p, q \in [n] \times [n]$. The orthogonal convolutional weights are obtained by applying the 2D inverse Fourier Transform on selected slices of $\mathcal{P}$.

0 which guarantees that $\lambda_2 \leq 2\sqrt{\lambda_1 - 1}$ is always satisfied as λ_2 is always zero.

We verified this property by computing the distribution of eigenvalues of d-regular bipartite graphs with 128, 256, 512, and 1024 neurons each, as shown in Fig. 7. With this construction, the sparsity of the topology can be controlled by selecting the value of d, which corresponds to the number of weighted connections to each output neuron. These sparsely-connected neural layers can then be cascaded to form sparse deep neural networks prior to gradient learning.

We expect this structure to have advantages over standard pruning methods. First, as we use d-regular graphs, all input nodes have a path toward the output nodes of a layer, and all output nodes receive signals from the input nodes. This assures that there won't be any *dead* neurons in the neural network, *i.e.* neurons that do not contribute to the output of the model. Second, as the d-regular graphs we use are also expander graphs, we assure that every input node is sensed by every output node, thus providing *global connectivity* as proved in [11] in their work on expander neural networks. These two benefits point to the maximization of the connectivity, which is one of the factors affecting information flow in the network during training [11], [47]. Furthermore, as we use Ramanujan expanders, which are described as the optimal expanders, we are maximizing the benefits of expander properties in our construction of sparse neural networks.

The benefits of having Ramanujan layers are not limited to the connectivity within each layer. Reference [11] showed that using expander graphs for each layer can assure global connectivity for a given number of layers. They defined global connectivity as having all the inputs have a path towards the outputs. In our work, we refer to this quality as *reachability*. We illustrate this in Fig. 8, where we compare the reachability of a sparse neural network uncovered through magnitude pruning and one which was constructed using Ramanujan graphs. The figure labeled $N = 1$ is the transpose of the mask of a single 256 × 256 layer, where the colored pixels indicate the presence (1) or absence (0) of a connection between the input nodes (rows) and the output nodes (columns) of this specific layer. While the connections in magnitude pruning are sporadic, a pattern can be seen in the Ramanujan construction, coming from the d-regularity and in a characteristic specific to our construction, where the second half of the graph is identical to the first half. At $N = 5$, RG achieved full connectivity, *i.e.* all the input nodes *reach* the output nodes, in contrast to magnitude pruning where some input nodes are disconnected from some output nodes. The connectivity in RG is carried over in deeper layers, which agrees with the report of [11], thus proving that Ramanujan networks are able to achieve full connectivity despite the sparsity.

B. SAO LAYERS RETAIN TRAINING DYNAMICS OF FULLY-CONNECTED, ORTHOGONAL WEIGHT LAYERS

Linear, fully connected networks may attain dynamical isometry with orthogonally initialized weights [15]. All singular values of the weight Jacobian matrix are unity, and there are no dynamical modes that cause vanishing or exploding gradients during model training. Nonlinear activations compromise dynamical isometry [34] by introducing singular values that are both greater and less than unity, which engenders dynamical modes capable of propagating exploding and vanishing gradients, respectively, through deep networks. Sigmoidal activation functions may attain *approximate dynamical isometry* with the majority of singular values being arbitrarily close to unity [15]. ReLU did not achieve dynamical isometry due to the sizeable presence of zero-valued singular values which engender vanishing gradients, in contrast to its smoothed variant, SiLU [34].

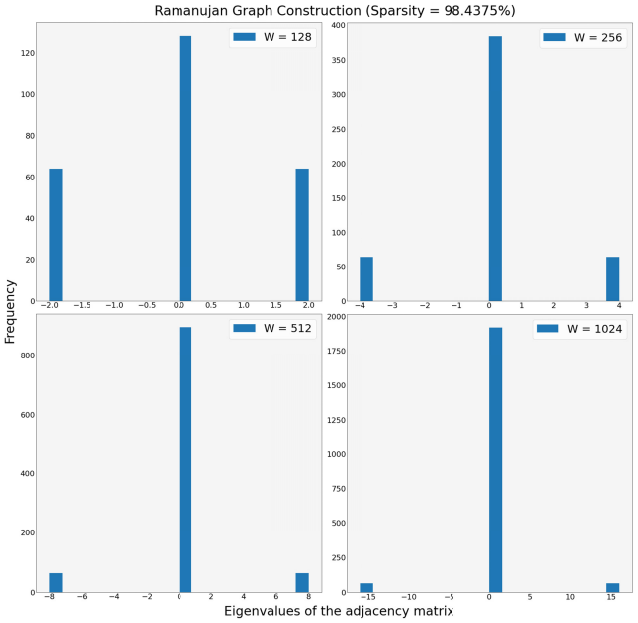

FIGURE 7. Neural network layers constructed according to our sparse d-regular graph topology always fulfill the Ramanujan graph spectral expansion condition. The distribution of eigenvalues always consist of d, $-d$, and 0, irrespective the number of neurons in the layer.

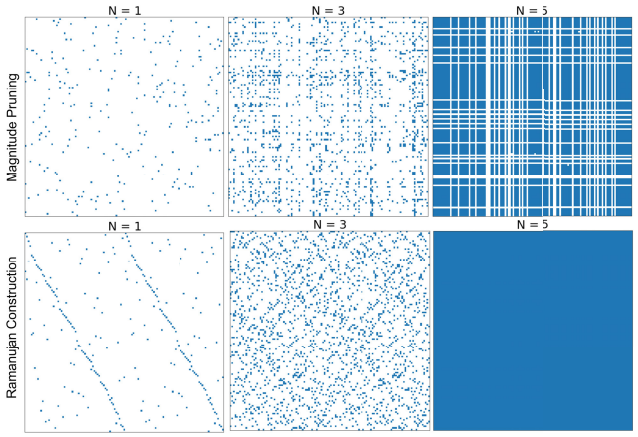

FIGURE 8. An illustration of the reachability matrix from sparse neural networks obtained through magnitude pruning and constructed using Ramanujan graphs shows that the Ramanujan construction achieved global connectivity by the 5^{th} layer, as indicated by the solid color. The colored pixels indicate the presence (0) or absence (0) of a connection between the input nodes (columns) and the output nodes (rows) at the N^{th} layer. For magnitude pruning, the white strips indicate some output nodes are disconnected from some input nodes.

To illustrate, we plotted the distribution of singular values of the weight Jacobian from a single layer of fully-connected neurons in Fig. 9. Local magnitude (LMP), local random pruning (LRP), and sparse Ramanujan layers up to 99.2% sparsity introduced null singular values into the training dynamics of fully-connected, linear layers and destroyed dynamical isometry. In contrast, SAO preserves the distribution of singular values of fully-connected layers as illustrated in Fig. 10 and Table 6. Thus, SAO retains the beneficial

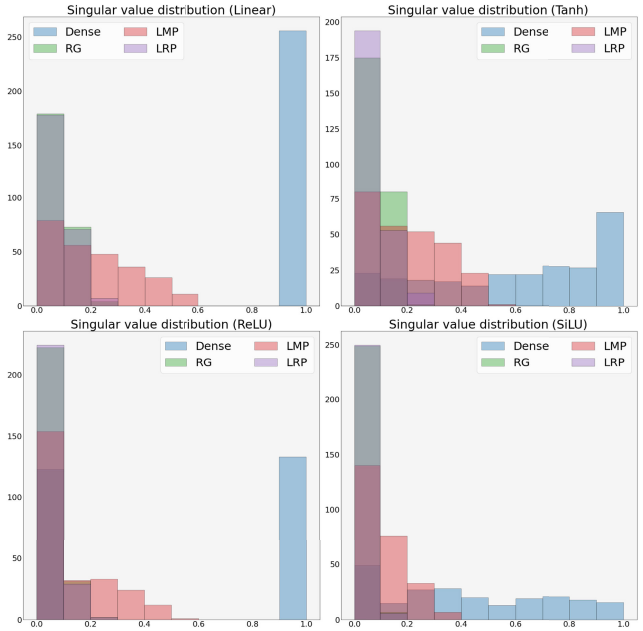

FIGURE 9. Plot of the singular value distribution of the Jacobian weight matrix from a single layer of fully-connected, orthogonally initialized neurons is overlayed with the distributions from sparse layers with 99.2% sparsity. With linear and non-linear activation functions, local pruning (LMP and LRP) and Ramanujan layers distort dynamical isometry by shifting singular values towards and to zero.

TABLE 6. The distribution of singular values between SAO, Ramanujan layers and local pruning are compared to fully-connected orthogonal layers with the Kolmogorov-Smirnov test. The singular values of Jacobian weight matrix from SAO layers are indistinguishable from dense, orthogonal layers. The p-values indicate that all other sparsity techniques produce significantly different singular value distributions.

Method	SAO	RG	LMP	LRP
Linear	1	4.23E-153	4.23E-153	4.23E-153
Tanh	0.83958	4.74E-107	8.92E-54	2.97E-102
ReLU	0.99998	2.07E-30	2.07E-30	2.07E-30
SiLU	0.41603	1.42E-96	1.53E-62	2.55E-93

training dynamics granted by orthogonal initialization on dense networks, along with the added advantage of sparse connectivity. We expect SAO layers to remain trainable to thousands of layers with fast convergence [15], [16].

C. SAO CREATES DEEP AND SPARSE NEURAL NETWORKS

Despite the sparsity of individual neural layers, all input nodes have a path towards all output nodes and provide *global connectivity* according to [11], which implies that all input nodes can influence every receptive field in the output layer. The maximization of input-output node connectivity is a factor that affects information flow in the network during training [11], [47]. We introduce the concept of a reachability matrix (Section III-D) to visualize the influence of input nodes on output nodes when propagated across deep layers. A dense reachability matrix implies that all input nodes can

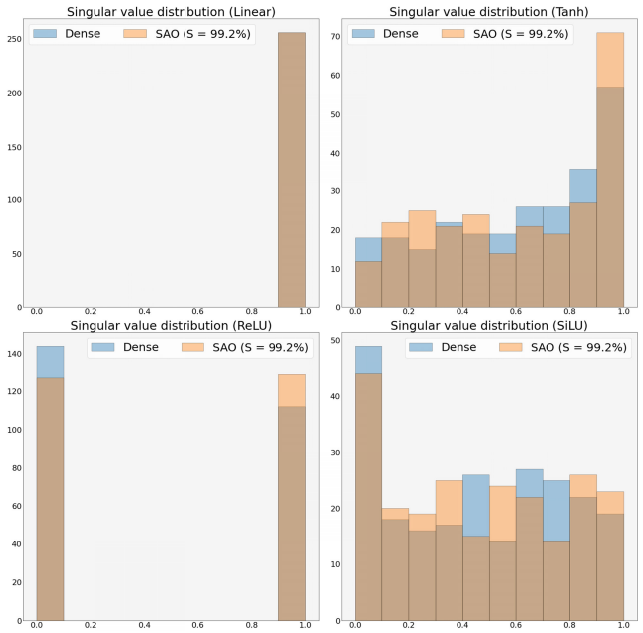

FIGURE 10. Despite achieving 99.2% sparsity over fully-connected neurons, SAO preserves the distribution of singular values with linear and non-linear activations.

contribute information toward the computation of the output receptive fields.

Table 7 compares the classification accuracy achieved by deep sparse networks against deep, dense fully-connected networks between 50 to 1000 layers for the MNIST classification task. We compared pruning via global magnitude pruning (GMP) with training sparse, deep networks configured *a priori* using Ramanujan graph topology with random uniform (RG-U), random Gaussian (RG-N) and random orthogonal (SAO) initialization. Random orthogonal initialization encourages dynamical isometry, and we dub the combination of d-regular graph topology and random orthogonal weight initialization as Sparsity-Aware Orthogonalization (SAO).

Sparse, deep networks cannot be robustly obtained from global magnitude pruning (GMP) of dense layers because the algorithm does not train to convergence. The distribution of singular values of the input-output Jacobian for each layer approaches 0 and indicates that GMP suffers from vanishing gradients [15], which impedes gradient learning (see Fig. 11).

Utilizing the Ramanujan graph topology to define *a priori* neural layers helps these sparse, deep networks (RG-U and RG-N) train towards convergence but still appears insufficient to achieve good accuracy consistently (see Table 7). The singular value distributions of $\mathbf{J}$ in RG-U and RG-N have a mean close to 1 but large variance (see Fig. 11), implying the presence of very large and very small singular values. The extreme singular values indicate that the network occasionally suffers from vanishing and exploding gradients which hinder gradient learning and necessitates more training epochs to converge [34]. Only SAO reliably converges to a high-accuracy model for deep, sparse networks of 50 up

TABLE 7. The advantage of SAO over standard pruning methods and Ramanujan pruning proves the importance of sparse-orthogonality and is demonstrated by comparing the accuracy of tanh fully-connected networks (Dense) on MNIST with pruned, Ramanujan topology and SAO with $d = 4$ at increasing network depth, N. Only RG-U, RG-N, and SAO remained trainable up to 1000 layers, while only SAO did not experience drastic accuracy degradation.

N	50	200	400	600	800	1000
Dense	98.41	98.40	98.21	96.64	97.90	97.58
GMP-S			10.00			
GMP-T			10.00			
RG-S			10.00			
RG-T			10.00			
RG-N	94.39	94.66	92.51	91.80	10.00	10.00
RG-U	95.99	95.83	92.97	93.20	10.00	10.00
SAO	94.47	94.91	95.21	95.94	95.71	95.93

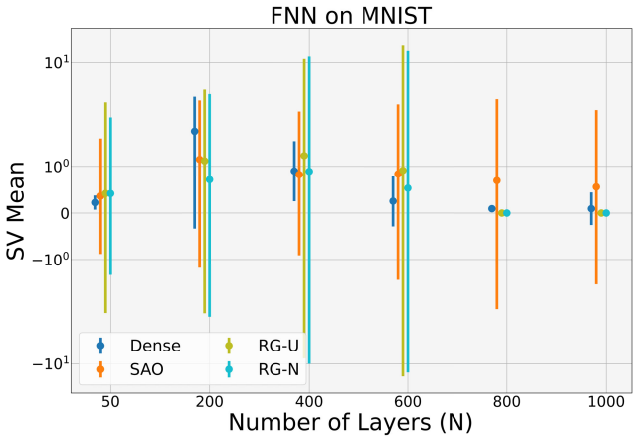

FIGURE 11. Plot of the mean and variance of singular values of the Jacobian from networks trained on MNIST with increasing depth, N, shows that SAO keeps the distribution of singular values relatively close to unity just like fully connected networks. Ramanujan topology alone does not preserve approximate dynamical isometry since RG-U, and RG-N have high variance indicating the presence of very small and large singular values, while dense and SAO have relatively low variance indicating approximate dynamical isometry.

to 1000 layers within the same 100 training epochs [16] as indicated in Table 7. The distribution of singular values of $\mathbf{J}$ are close to unity with tight variance [15] as illustrated in Fig. 11.

The implications of these results are: 1) the Ramanujan property is a prerequisite that promotes sparsity and reachability of gradients but is not enough to guarantee trainability in very deep sparse neural networks, and 2) dynamical isometry can be created in deep, sparse networks to engender convergence to high accuracy models. This evidence suggests that SAO, which is comprised of both the sparse Ramanujan graph topology and orthogonal weight initialization to enforce dynamical isometry, grants trainability in very deep, sparse neural network. Sparse but regular connectivity assures that sparsity can be increased in a principled way, and yet no layer or neuron will be entirely disconnected. The maximal connectivity property retains pathways in the sparsely connected network for information to propagate

from input to output nodes. As the network grows in depth, we observe that more input nodes have pathways to influence the choice at every output node despite topological sparsity. Meanwhile, orthogonality not only conditions the Jacobian to avoid vanishing or exploding gradients but also reduces the interdependence of weights and thereby improve the efficiency and efficacy of learned feature representations.

D. SAO PROVIDES A UNIFYING FRAMEWORK TO TUNE SPARSITY IN ORTHOGONALLY INITIALIZED CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNNs) are locally connected networks (not full) with the same set of weights applied across all neurons in each channel of a neural layer. CNN models reduce the number of model parameters for a regularizing effect [48], which makes the model less prone to overfitting. CNNs have been especially successful in machine vision applications compared to fully connected networks. The computation is implemented by convolving small weight matrices across the input image to produce neural feature maps [49]. Repeated weight parameters may be efficient for computation, storage, and model convergence. However, it is challenging to impose orthogonality on weight vectors that are shifted versions with repeated values [32], [50] and more so if the weight vectors must also be sparse. Fig. 12 shows a visual illustration of the 2-dimensional convolution operation expressed as a doubly block-circulant Toeplitz (DBT) matrix where the 2-dimensional weight "filter" has been unwrapped into a basis of 1-dimensional vectors comprising of spatially shifted versions of the same vectors.

Achieving orthogonal weights has been attempted through regularization [32] and workarounds to remove spatial overlaps of shifted vectors such as the Delta-Orthogonal Initialization [16]. Sparsity has been achieved on orthogonally initialized CNNs through rescaling methods. Dynamical isometry was restored in pruned networks by rescaling weights using a scaling factor derived from the weights, width, and the pruning mask [51]. Such rescaling presented up to 4.25% higher accuracy than just magnitude pruning at 98% sparsity on ResNet-104 trained on CIFAR-10. Regularization was applied during model training to scale the trace of the gram matrix of the pruned weight matrix towards identity to restore dynamical isometry [13], while the diagonal entry which corresponds to the unimportant channels are driven to zero [28]. [13] showed the benefits of such regularization on the accuracy of various CNNs such as VGG and ResNet, where the pruned models that were trained with regularization after pruning achieved up to 0.70% accuracy improvement than without regularization. Meanwhile, [28] showed up to 1.30% higher accuracy compared to standard pruning methods on ResNet-56 trained on the CIFAR-10 dataset at 95% sparsity. None of these studies implemented their methods on a very deep sparse vanilla convolutional neural networks without means of maintaining trainability such as batch normalization and residual connections.

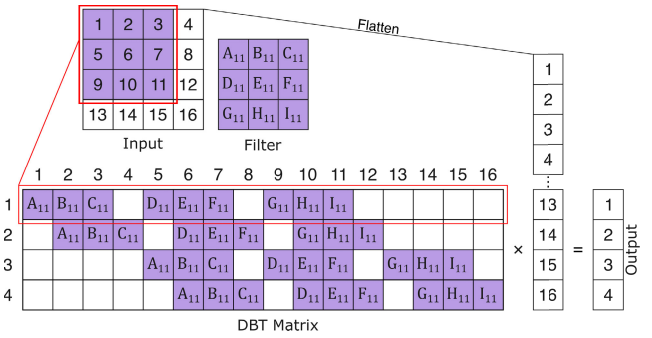

FIGURE 12. The single-strided 2D convolution operation on a 4 × 4 image with 3 × 3 filter can be represented as a Doubly block-circulant Toeplitz (DBT) matrix applied to the vectorized image. Each row of the DBT matrix represents the application of the filter at one spatial location. To achieve orthogonal CNNs, all row vectors in the DBT matrix must be mutually orthogonal. This is difficult to explicitly orthogonalize because each row is a shifted version of the same vector. Delta-Orthogonal Initialization circumvents this difficulty by setting all values in the kernel to zero except for E_{11}.

Viewed from the lens of SAO, CNN layers are inherently d-regular graphs with additional constraints that edges are localized to spatially-connected neighbor nodes and edge weights are repeated. Therefore CNNs are compliant with the Ramanujan property. By extension, any method that provides orthogonal weight initialization (which is not the same as having orthogonal kernels) with CNN may be interpreted as SAO.

To adapt SAO for CNNs, we build upon two explicitly-orthogonal convolution schemes: first, the Delta-Orthogonal Weight Initialization [16], [34], [38] and second, the Explicitly-Constructed Orthogonal (ECO) convolutions [31]. While both methods are already forms of SAO by definition, we apply the SAO framework to generalize the concept of tunable sparsity levels and refer to these two implementations as SAO-Delta and SAO-ECO, respectively.

The Delta-Orthogonal Initialization [16] is similar to initializing the CNN with depth-wise convolutions [52] but allows the network to learn wide filters during model training. Since each 2D kernel has only a single non-zero weight, controlling the sparsity of the d-regular Ramanujan graph is achieved by removing the 2D kernel of individual channels, which is structured (kernel-level) pruning.

We report the performance of SAO-Delta CNN for the CIFAR-10 classification task using sigmoidal and linear unit activations, *i.e.* Tanh and ReLU, in Table 8, with a variety of depth, width, and sparsity (50% and $(1 - \frac{4}{C_{in}})\%$) configurations. We benchmark the sparse SAO-Delta CNN against an equivalent vanilla CNN (Dense) baseline. In our experiments, the vanilla CNN required Delta-Orthogonal Initialization to remain trainable to a practical accuracy beyond a depth of 128 layers. We also attempted to compare local magnitude pruned (LMP-S and LMP-T) version of the vanilla CNN and CNN with Ramanujan-only constructions (RG-S and RG-T). However, only SAO remained trainable for all but one case at $(1 - \frac{4}{C_{in}})\%$ sparsity and on 256 layers and above at 50% sparsity.

To test the generalization capabilities of SAO on larger datasets, we analyze the performance of SAO-Delta CNN on CINIC-10. This dataset bridges the gap in complexity between ImageNet and CIFAR-10 by combining their samples, resulting in a dataset 450% larger than CIFAR-10 [53]. Table 9 reports the accuracy of SAO-Delta on CINIC-10 for a variety of depth and sparsity levels. The models utilize the ReLU activation function with a fixed layer width of 128, *i.e.*, $C_{in} = C_{out} = 128$. We compared SAO-Delta to the dense baseline network and to LMP-S. Similar to the results of CIFAR-10, SAO can maintain trainability at large depths and sparsity, in contrast to the standard pruning method LMP-S which failed at high sparsity levels at shallow depths, and even at low sparsity levels at greater depths. Notably, among depths of $N = 8, 16, 32$, the robustness of SAO-Delta against sparsity increased with depth. This could be attributed to the increase in model capacity with depth, just before reaching a point where training complications start to arise. At $N = 128$, the accuracy for both the dense baseline and SAO-Delta decreased and could be attributed to the presence of very large and small singular values relative to unity which is exacerbated in deeper networks [16]. At $N = 128$, a lower learning rate is required to maintain trainability and slows down convergence [16]. The SAO-Delta model achieved higher accuracy than that of $N = 8$ at 96.875% sparsity; it is evidence that SAO improves robustness of training in very sparse networks with greater depth.

SAO-Delta was also tested on the CIFAR-100 dataset, which has 10 times more classes and 10 times fewer training images per class as compared to CIFAR-10. The performance of SAO-Delta on CIFAR-100 is limited by the baseline dense model and are summarized in Table 10. At very high sparsity levels, SAO stabilizes and improves the model performance with increasing network depth.

SAO-Delta does not yet achieve comparable performance with state-of-the-art CNN networks with special layers, but it contributes an incremental improvement towards sparse and energy-efficient deep CNN. State-of-the-art ResNet-101, which incorporates batch normalization and skip connections with customized training algorithms, achieves accuracy in excess of 93% on CIFAR-10 [54]. Meanwhile, other network variations such as ResNet-104 achieves 75.24% on CIFAR-100 [51], while ResNet-18 achieves 90.27% accuracy on CINIC-10 [53]. In a selected few SAO-Delta configurations with improved performance, we attribute the slight accuracy boost to the regularizing effect of sparsity and optimal sizing of network width and depth [55].

The explicitly constructed orthogonalization (ECO) scheme was first created as one of several adaptions to CNNs to ensure trainability by preserving the gradient norm. CNNs compliant to 1-Lipschitz constraints [56] learn well-conditioned and improved feature representations with better generalization capabilities on unseen data [57]. These more interpretable gradient errors facilitate tracing sources of error or biases in the model [58] and enhanced robustness against adversarial attacks since variations in the network output are bounded by the magnitude of perturbations to the input [59]. To meet 1-Lipschitz constraint, the modified CNN dubbed LipConvNet [31], [60] utilized the gradient norm preserving MaxMin activation function [61] and orthogonalized the input-output Jacobian matrix with the weight initialization schemes, SOC and ECO. ECO cleverly defines orthogonal kernels in the Fourier domain to generate orthogonal convolutions in the spatial domain.

Our SAO-ECO makes a novel contribution by showing how spatial domain sparsity can be achieved by sparsifying the Fourier kernels. If a sparse orthogonal matrix $\mathbf{S}$ is used to generate slices of the Fourier domain kernel $\mathcal{P}[p, q, :, :]\ \forall p, q$, there will be several $\mathcal{P}[:, :, s, t]$ comprised

TABLE 8. SAO-Delta helps sparse and deep CNN converge to models with good performance. On CIFAR-10 classification task, SAO and CNN are trainable up to 768 layers with Delta-Orthogonal Initialization. With 256 or deeper layers, magnitude pruned CNN was not trainable while RG-U and RG-N CNN did not exceed 25% accuracy.

Method	Params.	Tanh	ReLU
$N = 32, C_{in} = C_{out} = 32$			
Dense	2.87E+05	78.27	82.77
LMP-S	1.44E+05	79.57	81.56
LMP-T	1.44E+05	80.84	83.38
LMP-T	3.66E+04	74.45	10.00
RG-S	1.44E+05	78.87	10.00
SAO	1.44E+05	79.24	82.59
SAO	3.66E+04	78.46	78.71
$N = 32, C_{in} = C_{out} = 256$			
Dense	1.83E+07	78.90	87.72
LMP-S	9.15E+06	80.15	88.60
LMP-T	9.15E+06	80.89	88.60
SAO	9.15E+06	78.99	88.10
SAO	2.93E+05	80.65	84.10
$N = 128, C_{in} = C_{out} = 32$			
Dense	1.17E+06	73.28	76.50
LMP-S	5.86E+05	69.98	75.64
LMP-T	5.86E+05	78.34	77.31
SAO	5.86E+05	74.76	75.01
SAO	1.47E+05	74.20	76.20
$N = 128, C_{in} = C_{out} = 128$			
Dense	1.87E+07	75.00	85.34
LMP-S	9.37E+06	73.78	74.70
LMP-T	9.37E+06	76.84	86.47
SAO	9.37E+06	77.63	84.50
SAO	5.89E+05	73.67	81.96
$N = 256, C_{in} = C_{out} = 32$			
Dense	2.35E+06	74.35	72.56
SAO	1.18E+06	74.38	63.33
SAO	2.95E+05	69.88	70.96
$N = 512, C_{in} = C_{out} = 32$			
Dense	4.71E+06	77.56	73.76
SAO	2.36E+06	76.35	74.00
SAO	5.90E+05	68.40	59.86
$N = 768, C_{in} = C_{out} = 32$			
Dense	6.70E+06	76.78	56.60
SAO	3.35E+06	76.23	60.69
SAO	8.38E+05	68.95	54.54

TABLE 9. SAO-Delta facilitates convergence of sparse and deep CNN models to achieve high performance on the larger CINIC-10 dataset with configurations comparable to the dense baseline model, in contrast to standard pruning methods.

Sparsity	Method	N = 8	N = 16	N = 32	N = 128
0%	Dense	81.32	81.23	81.47	77.94
50.00%	SAO	79.78	80.35	80.37	75.87
	LMP-S	79.54	80.32	80.12	75.97
75.00%	SAO	77.08	77.16	78.83	73.72
	LMP-S	76.90	78.25	78.03	10.00
87.50%	SAO	73.57	75.70	76.63	71.05
	LMP-S	72.94	74.11	10.00	10.00
96.875%	SAO	64.66	66.51	68.45	65.48
	LMP-S	10.00	10.00	10.00	10.00

TABLE 10. At high sparsity, the performance of SAO-Delta on CIFAR-100 dataset stabilizes and improves accuracy with increasing depth, *N* and shows a similar trend with the CINIC-10 dataset but its performance is limited by the dense baseline model.

Sparsity	Method	N = 8	N = 16	N = 32
0%	Dense	67.35	66.52	63.10
50.00%	SAO	66.49	64.17	62.23
75.00%	SAO	63.12	63.47	61.10
96.875%	SAO	51.61	53.75	54.23

entirely of zeros. Consequently, the spatial kernel $\mathcal{W}[:, :, s, t]$, which will be recovered through the inverse Discrete Fourier Transform will also be sparse. Therefore, defining sparse orthogonal Fourier kernels is equivalent to kernel-level pruning, where entire spatial-domain filter kernels are zeroed out. Kernel-level sparsity in the spatial domain can be directly related to d-parameterized sparsity of the orthogonal Fourier domain vectors.

We compare SAO-ECO to pruned versions of LipConvNet on CIFAR-10. At lower sparsity levels, the pruning methods outperformed SAO, but only SAO was able to maintain trainability at 93.75% sparsity and above for all N. LipConvNet initialized with sparse RG-S and RG-T topology also failed to train to convergence beyond $> 50\%$ sparsity. We examined the singular values of the Jacobian from all pruned networks, which failed to converge, and found that mean singular values approached zero as training progressed. Pruning destroys orthogonality in LipConvNet and compromises the trainability of the CNN [34].

We also implemented SAO-ECO on a vanilla CNN with $C_{in} = C_{out} = 128$ on the CINIC-10 dataset. Similar to SAO-Delta, deeper SAO-ECO networks are also found to be more robust at high levels of sparsity as reported in Table 12.

For both SAO-Delta and SAO-ECO, we show that the trainability of deep CNN can be maintained with tunable levels of sparsity without requiring special layers. Absent the use of special layers like residual connections [21] and batch normalization [62], our results for CNNs are consistent with behaviors observed in the fully connected networks

TABLE 11. SAO-ECO preserves the performance and trainability of CNNs on CIFAR-10 especially for very sparse configurations. The gradient limiting advantages of LipConvNet did not preserve the performance of pruned networks or Ramanujan layers in deep and sparse CNN networks.

LipConvNet-N				
Sparsity	Method	N = 5	N = 10	N = 15
0%	Dense	80.15	78.75	79.04
50.00%	LMP-S	79.04	79.10	78.68
	LMP-T	79.36	79.37	79.08
	RG-S	78.38	78.56	74.13
	RG-T	79.84	79.73	79.20
	SAO	78.81	78.44	77.35
87.50%	LMP-T	75.46	10.00	10.00
	RG-T	77.19	10.00	10.00
	SAO	76.77	75.24	75.26
93.75%	LMP-T	73.27	10.00	10.00
	SAO	75.86	73.44	73.53
96.875%	LMP-T	70.80	10.00	10.00
	SAO	73.39	72.51	72.00
98.4375%	SAO	72.45	70.01	70.79

TABLE 12. SAO-ECO preserves the trainability of vanilla CNNs with ReLU activation on CINIC-10 especially with very sparse configurations.

Sparsity	Method	N = 8	N = 16
0%	Dense	79.10	77.85
50.00%	SAO	77.92	75.24
96.875%	SAO	68.45	69.87

in Section IV-A, where both Ramanujan connectivity and weight orthogonality are integral to maintaining trainability of deep and sparse neural networks.

We argue that the SAO framework for tunable sparsity adds value to applying dynamical isometry to CNNs. SAO preserves trainability while deep CNN is made sparse without additional parameter or computational overheads. SAO does not introduce additional model parameters [62] and provably minimizes uncontrolled gradients [63] through dynamical isometry unlike batch normalization nor does it impose additional gradient computations as in the case of residual connections. Without the benefits of sparsity, the advantages of dynamical isometry in CNNs are less clear. Similar to [34], we observed no clear performance benefits when increasing the depth of CNNs beyond 200 layers. Furthermore, CNN with ReLU activations do not possess dynamical isometry but are still favored for image recognition tasks [64] and outperform CNNs with sigmoidal activations. We observe these results empirically, and ReLU CNNs are trainable to convergence.

E. LIMITATIONS OF SAO

SAO permits sparse weight and model topology initializations without reference to a dense model but has several limitations. Firstly, SAO imposes dimensional constraints where the inputs, outputs, and their connectivity must obey the condition in Section IV-A. The (c, d)-regular graphs to achieve Ramanujan property also imposes discrete increments in

sparsity levels in each layer. Second, it is observed from experiments that the performance of the baseline dense model serves as the ceiling on the performance of SAO networks in most cases. As there are not yet CNN and MLP models that achieve state-of-art performance without special layers like batch normalization and skip connections, SAO models are not yet competitive accuracy-wise. Finally, SAO cannot leverage the many, publicly available foundational and pre-trained dense models as a good starting point because of differences in network topology. However, SAO networks may still benefit through knowledge distillation approaches albeit with the SAO network acting as a student network to the dense model as teacher network.

VI. CONCLUSION

We introduced the method sparsity-aware orthogonal initialization (SAO), which explicitly initializes a neural network with sparse connections and orthogonal weights prior to model training. SAO networks simultaneously achieve sparse construction and approximate dynamical isometry with a principled approach to tuning the sparsity level. SAO networks achieve comparable accuracy to their dense equivalents with far fewer trainable parameters whilst allowing network configurations with very deep layers to converge during model training and outperform popular network pruning methods. For a plain 1000-layer fully-connected neural network on MNIST, SAO minimized the accuracy loss to 1.65% at 98.4% sparsity, in contrast to conventional pruning methods and other Ramanujan constructions which did not train past 10.00% accuracy. SAO attains this using fewer computations than pruning or dynamic sparse training because no special layers or additional training iterations are required. SAO is attractive for applications with tight computation or energy budgets, such as edge computing applications.

Seen through the SAO framework, CNNs already possess a sparse d-regular bipartite graph topology. Adding orthogonal weight initialization converts the network to SAO with added constraints - spatially-localized connections with repeated weight values. SAO contributes new insight to unify the interpretation of kernel pruning on CNNs employing different weight orthogonalization schemes. Introducing sparsity to Delta-Orthogonal initialization and explicitly constructed orthogonalization is equivalent to kernel pruning. The number of pruned kernels may be determined from the outset prior to model training and conditioned on the desired sparsity level of the CNN rather than on weight-dependent heuristics. SAO showed greater robustness against sparsity and depth in plain convolutional neural networks, achieving comparable accuracy to the dense baseline network on certain configurations and maintaining trainability up to 768 layers, in contrast to other methods which rendered the networks untrainable beyond 128 layers.

ACKNOWLEDGMENT

The authors thank Yayasan Universiti Teknologi PETRONAS in collaboration with PETRONAS Research Sdn Bhd to support open access article publishing charges. The author Kiara Esguerra thanks the Department of Science and Technology-Science Education Institute (DOST-SEI), Philippines, for a graduate scholarship.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, Jun. 2017, doi: 10.1145/3065386.

[2] S. Liang and R. Srikant, "Why deep neural networks for function approximation?" in *Proc. IEEE Workshop Neural Netw. Signal Process.*, Oct. 2016, pp. 21–29.

[3] D. Rolnick and M. Tegmark, "The power of deeper networks for expressing natural functions," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–14.

[4] J. Guo, W. Liu, W. Wang, C. Yao, J. Han, R. Li, Y. Lu, and S. Hu, "AccUDNN: A GPU memory efficient accelerator for training ultra-deep neural networks," in *Proc. IEEE 37th Int. Conf. Comput. Design (ICCD)*, Nov. 2019, pp. 65–72.

[5] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," 2020, *arXiv:2007.05558*.

[6] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," 2021, *arXiv:2104.10350*.

[7] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Process. Mag.*, vol. 35, no. 1, Jan. 2018, doi: 10.1109/MSP.2017.2765695.

[8] Y. L. Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. 2nd Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Cambridge, MA, USA: MIT Press, Jan. 1989, pp. 598–605.

[9] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, Oct. 2021.

[10] J. Liu, Z. Xu, R. Shi, R. C. C. Cheung, and H. K. H. So, "Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–14.

[11] A. Prabhu, G. Varma, and A. Namboodiri, "Deep expander networks: Efficient deep networks from graph theory," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, in Lecture Notes in Computer Science, vol. 11217, 2018, pp. 20–36.

[12] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 6377–6389.

[13] N. Lee, T. Ajanthan, S. Gould, and P. H. S. Torr, "A signal propagation perspective for pruning neural networks at initialization," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–16.

[14] H. Wang, C. Qin, Y. Bai, and Y. Fu, "Dynamical isometry: The missing ingredient for neural network pruning," 2021, *arXiv:2105.05916*.

[15] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–22.

[16] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. S. Schoenholz, and J. Pennington, "Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, vol. 12, 2018, pp. 8581–8590.

[17] M. Telgarsky, "Benefits of depth in neural networks," *J. Mach. Learn. Res.*, vol. 49, pp. 1517–1539, Jun. 2016.

[18] S. Sun, W. Chen, L. Wang, X. Liu, and T. Y. Liu, "On the depth of deep neural networks: A theoretical view," in *Proc. 30th AAAI Conf. Artif. Intell. (AAAI)*, 2016, pp. 2066–2072.

[19] M. Thoma, C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–14.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[22] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. 30th IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.
[23] K.-H. Tan and B. P. Lim, "The artificial intelligence renaissance: Deep learning and the road to human-level machine intelligence," *APSIPA Trans. Signal Inf. Process.*, vol. 7, no. 1, pp. 1–19, 2018.
[24] A. Bourely, J. P. Boueri, and K. Choromonski, "Sparse neural networks topologies," 2017, *arXiv:1706.05683*.
[25] B. Pal, A. Biswas, S. Kolay, P. Mitra, and B. Basu, "A study on the Ramanujan graph property of winning lottery tickets," in *Proc. 39th Int. Conf. Mach. Learn.*, 2022, pp. 17186–17201.
[26] J. Kepner and R. Robinett, "RadiX-Net: Structured sparse matrices for deep neural networks," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2019, pp. 268–274.
[27] J. Fischer, A. Gadhikar, and R. Burkholz, "Lottery tickets with nonzero biases," 2021, *arXiv:2110.11150*, doi: 10.48550/arXiv.2110.11150.
[28] H. Wang and Y. Fu, "Trainability preserving neural pruning," 2023, *arXiv:2207.12534*, doi: 10.48550/arXiv.2207.12534.
[29] G.-S. Cheon, S.-G. Hwang, S.-H. Rim, B. L. Shader, and S.-Z. Song, "Sparse orthogonal matrices," *Linear Algebra Appl.*, vol. 373, pp. 211–222, Nov. 2003.
[30] H. Sedghi, V. Gupta, and P. M. Long, "The singular values of convolutional layers," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–12.
[31] T. Yu, J. Li, Y. Cai, and P. Li, "Constructing orthogonal convolutions in an explicit manner," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–14.
[32] J. Wang, Y. Chen, R. Chakraborty, and S. X. Yu, "Orthogonal convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11502–11512.
[33] M. Borel and G. Vénizélos, "Vector calculus," in *Movement Equations 2: Mathematical and Methodological Supplements*. London, U.K.: Wiley, 2017, doi: 10.1002/9781119379065.ch1.
[34] J. Pennington, S. S. Schoenholz, and S. Ganguli, "Resurrecting the sigmoid in deep learning through dynamical isometry: Theory and practice," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4786–4796.
[35] J. Pennington, S. S. Schoenholz, and S. Ganguli, "The emergence of spectral universality in deep networks," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2018, pp. 1924–1932.
[36] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning augmentation strategies from data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 113–123.
[37] S. Singla and S. Feizi, "Skew orthogonal convolutions," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 1–11.
[38] R. Burkholz and A. Dubatovka, "Initialization of ReLUs for dynamical isometry," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.
[39] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–42.
[40] G. Brito, I. Dumitriu, and K. D. Harris, "Spectral gap in random bipartite biregular graphs and applications," *Combinatorics, Probab. Comput.*, vol. 31, no. 2, pp. 229–267, Mar. 2021.
[41] K. D. Maduranga, K. E. Helfrich, and Q. Ye, "Complex unitary recurrent neural networks using scaled Cayley transform," in *Proc. 33rd AAAI Conf. Artif. Intell. (AAAI), 31st Innov. Appl. Artif. Intell. Conf. (IAAI), 9th AAAI Symp. Educ. Adv. Artif. Intell. (EAAI)*, 2019, pp. 4528–4535.
[42] A. Lubotzky, "Expander graphs in pure and applied mathematics," *Bull. Amer. Math. Soc.*, vol. 49, no. 1, pp. 113–162, Jan. 2012.
[43] M. R. Murty, "Ramanujan graphs: An introduction," *Indian J. Discrete Math.*, vol. 6, no. 2, 2020.
[44] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1985.
[45] M. Dahleh, M. A. Dahleh, and G. Verghes, "Matrix norms and singular value decomposition," in *Lectures on Dynamic Systems and Control*. Massachusetts Institute of Technology, MIT OpenCourseware Course 6.241J, 2004, ch. 4. [Online]. Available: https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Book%3A_Dynamic_Systems_and_Control_(Dahleh_Dahleh_and_Verghese)
[46] D. A. Simovici, "Singular values," in *Linear Algebra Tools for Data Mining*. World Scientific Publishers, 2012, pp. 565–617, doi: 10.1142/8360.
[47] S. S. Schoenholz, J. Gilmer, S. Ganguli, and J. Sohl-Dickstein, "Deep information propagation," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–18.
[48] Z. Li, Y. Zhang, and S. Arora, "Why are convolutional nets more sample-efficient than fully-connected nets?" in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–11.
[49] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. [Online]. Available: http://ieeexplore.ieee.org/document/726791/
[50] A. Trockman and J. Z. Kolter, "Orthogonalizing convolutional layers with the Cayley transform," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–21. [Online]. Available: https://github.com/locuslab/orthogonal-convolutions
[51] S. Hayou, J.-F. Ton, A. Doucet, and Y. W. Teh, "Robust pruning at initialization," 2020, *arXiv:2002.08797*.
[52] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[53] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey, "CINIC-10 is not ImageNet or CIFAR-10," 2018, *arXiv:1810.03505*.
[54] J. Frankle, D. J. Schwab, and A. S. Morcos, "Training BatchNorm and only BatchNorm: On the expressive power of random features in CNNs," in *Proc. Int. Conf. Learn. Represent.*, vol. 10, 2021, pp. 1–28.
[55] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Jun. 2019, pp. 10691–10700.
[56] J. Heinonen, "Lectures on Lipschitz analysis," 14th Jyvaskyla Summer School, Dept. Math., Univ. Michigan, Aug. 2004.
[57] P. L. Bartlett, D. J. Foster, and M. Telgarsky, "Spectrally-normalized margin bounds for neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6241–6250.
[58] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–24.
[59] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, vol. 2, 2017, pp. 1423–1432.
[60] S. Singla and S. Feizi, "Skew orthogonal convolutions," 2021, *arXiv:2105.11417*.
[61] C. Anil, J. Lucas, and R. Grosse, "Sorting out Lipschitz function approximation," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Jun. 2019, pp. 432–452.
[62] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, vol. 1, Jul. 2015, pp. 448–456.
[63] K. Hu. (2016). *Revisiting Exploding Gradient: A Ghost That Never Leaves*. [Online]. Available: https://www.andrew.cmu.edu/user/kaihu/Revisiting_Exploding_Gradient.pdf
[64] T. Szandała, "Review and comparison of commonly used activation functions for deep neural networks," in *Bio-Inspired Neurocomputing*. Singapore: Springer, 2020, doi: 10.1007/978-981-15-5495-7.

KIARA ESGUERRA was born in Tanza, Navotas, Philippines. She received the B.S. degree in applied physics majoring in instrumentation from the University of Santo Tomas, Manila, Philippines, in 2019. She is currently pursuing the M.Sc. degree in electrical and electronics engineering with Universiti Teknologi PETRONAS, Malaysia. From 2020 to 2021, she was a Research Assistant with the Research Center for Natural and Applied Sciences, University of Santo Tomas. Her research interest includes deep learning model compression.

MUNEEB NASIR received the Bachelor of Engineering degree (Hons.) in electrical and electronics engineering, majoring in instrumentation and control from Universiti Teknologi PETRONAS, Malaysia, in 2020. He is currently pursuing the M.Sc. degree in electrical and electronics engineering with Universiti Teknologi PETRONAS. His final year project was on self-supervised image segmentation. During 2021–2023, he is a Graduate Research Assistant with Universiti Teknologi PETRONAS sponsored by PETRONAS. His research interests include deep learning, model compression, self-supervised learning, and indoor localization.

TONG BOON TANG (Senior Member, IEEE) was born in Johor, Malaysia. He received the B.Eng. (Hons.) and Ph.D. degrees from The University of Edinburgh. He is currently a Professor and the Institute Director of health and analytics with Universiti Teknologi PETRONAS. His research interest includes biomedical instrumentation, from device and measurement to data fusion. He received the Lab on Chip Award, in 2006, the IET Nanobiotechnology Premium Award, in 2008, the IET Mountbatten Medal, in 2020, and the Top Research Scientists Malaysia, in 2021. He served as the Secretary for the Higher Centre of Excellence (HICoE) Council and the Chair for the IEEE Circuits and Systems Society Malaysia Chapter. He is also a Chartered Engineer.

AFIDALINA TUMIAN received the engineering degree in electronics engineering majoring in computer, the M.Sc. degree from the University of Surrey, and the Ph.D. degree from the Life Sciences Interface Doctoral Training Centre Program, University of Oxford. She started her career in academia as a Tutor with the Faculty of Engineering, Multimedia University, in 2003. She joined the Kulliyyah of Engineering, International Islamic University Malaysia (IIUM), as a Lecturer. She is currently the Manager (Technology Discipline) in group research and technology with PETRONAS Research Sdn Bhd. She leads several successful developments of AI-ML data-driven technology from the research and development phase up to pilot testing. During her academic stint, she underwent a six-month industrial attachment with Cisco Systems (Malaysia) Sdn Bhd under the Ministry of Education Malaysia CEO@Faculty Program. Having completed her thesis in bioinformatics, in 2013, some of her work was published in *Science*, *Nature*, and *eLife*.

ERIC TATT WEI HO received the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 2007 and 2013, respectively. He was a Visiting Scholar with Deakin University, Australia, in 2023. He is currently a Senior Lecturer with the Department of Electrical and Electronics Engineering, Universiti Teknologi Petronas, Malaysia. His research interests include the application of deep learning to imaging systems, renewable energy systems, and brain network analysis.