Correlation Based Node Partitioning to Sparse Multilayer Perceptron

Cem Benar [1] and Ali Naci Akansu [2]

[1]New Jersey Institute of Technology
[2]Affiliation not available

October 31, 2023

Abstract

A signal-dependent, correlation-based pruning algorithm is proposed to sparsify inter-layer weight matrices of a Multilayer Perceptron (MLP). The method measures correlations of node outputs for an input or hidden layer. The nodes are partitioned, accordingly. The nodes of a partition with relatively higher correlations are bundled to be the inputs of a node in the next layer. Such partitioning improves subspace representation of nodes in the network. The numerical performances for various MLP architectures and input (training) signal statistics for the two-class classification problem are presented. The results provide insights on the relationships between signal statistics, node and layer behavior, network dimension, depth, sparsity, and system performance. We show convincing evidence in the paper that the model design should track input statistics and transformations through the building blocks to sparsify the network for improved performance and computational efficiency. The proposed pruning method may also be used to design a self-reconfiguring network architecture with weight and node sparsities.

Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version XX Month, XXXX.

Digital Object Identifier 10.1109/XXXX.2022.1234567

Correlation Based Node Partitioning to Sparse Multilayer Perceptron

CEM BENAR (Graduate Student Member, IEEE) and ALI N. AKANSU (Fellow, IEEE)

New Jersey Institute of Technology, Newark, NJ 07102 USA

CORRESPONDING AUTHOR: CEM BENAR (email: cb427@njit.edu).

ABSTRACT A signal-dependent, correlation-based pruning algorithm is proposed to sparsify inter-layer weight matrices of a Multilayer Perceptron (MLP). The method measures correlations of node outputs for an input or hidden layer. The nodes are partitioned, accordingly. The nodes of a partition with relatively higher correlations are bundled to be the inputs of a node in the next layer. Such partitioning improves subspace representation of nodes in the network. The numerical performances for various MLP architectures and input (training) signal statistics for the two-class classification problem are presented. The results provide insights on the relationships between signal statistics, node and layer behavior, network dimension, depth, sparsity, and system performance. We show convincing evidence in the paper that the model design should track input statistics and transformations through the building blocks to sparsify the network for improved performance and computational efficiency. The proposed pruning method may also be used to design a self-reconfiguring network architecture with weight and node sparsities.

INDEX TERMS Explainable neural network, pdf re-shaping, misaligned node, node compression ratio, layer compression ratio, sparsity, AR(1) source model.

I. INTRODUCTION

Over-parameterized deep neural networks perform exceptionally well in applications such as natural language processing and many others [1]. The best performing network architecture is designed mostly by trial-and-error. This procedure lacks the framework to explain model performance [2]. The development of new methods to design architecture and optimization for under-determined systems is an active research topic.

The state-of-the-art deep neural networks are heavily under-determined (over-parameterized) [3]. However, increased model complexity (more parameters) may not lead to reduced model generalizability. This behavior is in contrast to the classical statistical learning theory [3], [4], [5], [6], [7], [8], [9], [10], [11]. Experimental studies claim that overfitting training data and higher generalization performance can occur at the same time if the network is sufficiently over-parameterized [3], [4], [5], [6], [7], [8], [9], [10], [11]. When network parameter set or the number of training epochs (training time) is increased, network performance may initially improve, then worsen, and it may improve again. This phenomenon is known as *deep double-descent curve* or *double-descent hypothesis* [6], [7], [8], [9], [10], [11]. *Deep double-descent curve* is divided into three stages [6], [7], [8], [9], [10], [11]. They are called as a. *under-fitting* stage where the number of model parameters P is smaller than the number of training data points M ($P < M$), b. *critical-fitting* stage when $P \cong M$, and c. *overfitting* stage with $P > M$.

A common strategy is to train an over-parameterized neural network by using some form of inductive bias and to learn more compact representations (better approximation) with increased generalization performance. Deep neural networks tend to be biased towards low-rank solutions to reduce complexity and to enhance generalization performance, known as *implicit regularization*. The *benign overfitting* phenomenon and *implicit regularization* as observed in specific architectures and various real-world data sets suggest to over-parameterize neural networks judiciously and learn compressed representations (lower rank approximation) with improved performance [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22].

The modern neural networks are mostly dense and over-parameterized [3]. They have built-in inductive bias of architectural choices (e.g., convolutional layers, pooling methods, positional-encoding mechanism), regularizations (e.g., L0, L1, or L2 norm), optimizer types, and initialization methods [1]. Using inductive bias is intuitive to enhance the model approximation. Some of them may be computation-

ally demanding to train the model [23]. Their redundancy may be reduced by incorporating sparsity methods during model optimization [23]. Sparse neural networks achieve higher efficiency in the forms of speed, storage, and energy consumption with better (or marginally lower) performance than their dense counterparts [23]. The advantages of sparse neural networks are highlighted in Section II and quantified in Section V.

The regularization effect of sparsity in optimization is discussed in Section II. A Multilayer Perceptron (MLP) is briefly presented in Section III-C. We introduce the node and layer performance metrics for explainability of network behavior in Section III-E. They relate variations of entropy (compression) through the nodes and layers to network performance. The motivation and mathematical framework for the proposed (correlation-based) weight pruning method to sparsify a network is given in Section IV-A. Its algorithmic and implementation details are explained in Section IV-B and Section IV-C, respectively. The data sets used in the experiments are described in Section IV-D. The performances of fully-connected and sparse MLPs (pruned with various techniques) for AR(1) and CIFAR-10 data sets are compared in Section V. The conclusions are given in Section VI.

II. SPARSITY REGULARIZATION

There is redundancy among model parameters (features) of over-parameterized deep neural networks since representations within each hidden layer (subspace) and between layers are usually correlated [20]. Redundant features provide little or no additional information beyond what is already expressed by others. Learning redundant features leads to the undesired multi-collinearity with increased complexity without any improvement in model approximation. The multi-collinearity also exists between vectors (filters) of a trained weight matrix [16], [17], [18], [19]. The redundancy of trained weight matrices for several deep neural networks (e.g., VGG, ResNet, DenseNet) is shown in [16], [17], [18], [19]. Multi-collinearity and redundancy have several drawbacks as outlined below.

Firstly, redundant features are more sensitive to a noisy environment [24]. It is harder for a model to learn generalizable patterns from distorted training signals [24].

Secondly, the existence of redundant features deteriorates the interpretability and trustworthiness of a model. It becomes harder to identify the most relevant features in a model [25]. Trustworthiness is critical for most applications such as in medicine, autonomous driving, and finance [25]. In fact, the explainable artificial intelligence (XAI) pursues solutions with maximized trustworthiness [25]. Although XAI is attractive, some researchers favor interpretable models over explainable ones since it is very difficult and complex to explain heavily over-parameterized networks by using available frameworks.

Thirdly, heavily over-parameterized dense networks are mostly very large models. They require significant computational resources to train and perform evaluations. In contrast, sparse neural networks are more efficient to implement and run faster with mostly increased (or negligibly reduced) performance [26].

Lastly, neurological studies show a link between sparsity and intelligence in human brain. It is found that human brain is not densely connected [27]. In fact, sparsely connected human brains have higher intelligence and computational efficiency than dense ones [27]. As a supporting evidence, infant brains are more densely connected with higher activity than adult brains [28]. During adulthood, the removal of connections between neurons efficiently is essential to achieve higher intelligence [28].

Sparsity in deep learning is achieved by reducing the number of model parameters based on available sparsity schedules and selection criteria [23]. In this study, we primarily focus on the weight selection criteria and propose a signal dependent, correlation-based weight pruning algorithm to sparsify MLP as presented in Section IV-B. We compare the performance of the proposed method with the magnitude and random pruning methods. We also rank the classification performances of the fully-connected and sparsified MLPs for AR(1) (auto-regressive order one) model and CIFAR-10 data sets for the two-class case.

The contributions of the paper are summarized as follows.

1) The impact of signal statistics ($\Delta\rho$ and SNR) on network performance is demonstrated.
2) The two metrics called node compression ratio *(NCR)* and layer compression ratio *(LCR)* are proposed to explain the inner workings of model optimization at the node level. Their values are related to the *implicit regularization* forced by the optimizer. The invaluable insights for weight and node sparsity are gained from network simulations.
3) A correlation-based signal dependent network pruning method is proposed. Its superior performance over the magnitude and random pruning methods is shown for two different data sets.

The impact of weight pruning (sparsity) and input statistics on MLP performance are presented next in Section II-A.

A. PRUNING METHODS

The two popular pruning methods along with the proposed one are summarized in this section.

1) RANDOM PRUNING

Random pruning method does not consider any statistics in the network [23], [29]. It randomly selects the discarded weights [29], [30]. It is generally used as a baseline pruning method for comparisons [29], [30].

2) MAGNITUDE PRUNING

Magnitude pruning removes weights smaller than the predefined threshold to sparsify the network [23], [29], [31], [32], [33]. The pruning of a dense network is followed by the re-training of the sparse model (fine-tuning) [34]. The selection of magnitude threshold is important [23]. The optimum magnitude threshold is determined by cross-validation or learned during training [23], [35]. The main factor determining the magnitude threshold is the resource budget for implementation [23]. It is noted that smaller weights may be important for certain cases [36], [37]. The magnitude pruning does not consider input or any other node statistics in the network [23]. It was reported that signal-dependent (data-driven) sparsity methods are more effective than signal-independent (data-free) techniques [23], [38], [39], [40].

3) CORRELATION BASED PRUNING

Correlation-based pruning is a signal-dependent weight sparsity method [23]. It considers statistics of the input and hidden node outputs to sparsify a network [23]. The Hebbian learning inspired a number of correlation-based pruning algorithms [23], [37], [41], [42]. If two neurons simultaneously fire (high correlation between their firing rates), the strength of their synaptic connection is increased [43]. The correlation-based weight pruning method calculates Pearson correlation matrix for the node outputs of the current and previous layers [41]. Then, it removes connections between weakly correlated neurons and keeps ones with strong correlations [41]. The same idea is also tested by using the cosine similarity metric [42]. Weight matrices are sparsified starting from the last layer to the previous layers in sequence [41]. The entire neural network is re-trained (fine-tuned) each time after a weight matrix is sparsified [41]. Increasing sparsity ratio gradually decreases the probability of getting stuck in a bad local minimum [41]. Over-parameterized neural networks possess equally many similar local minima that are as (approximately) good as the global minimum [44]. In other words, the loss landscapes of smaller neural networks include a higher number of bad local minima [44]. Hence, layer by layer sparsifying and re-training the whole network progressively decreases the possibility of being stuck at a bad local minimum and improves performance [41]. This sparsity schedule is explained in detail later in Section IV-B.

B. IMPACT OF SIGNAL STATISTICS

The over-parameterized neural networks have high expressivity (redundancy) [45]. Experimental observations suggest that over-parameterized neural networks are able to fit randomly labeled training data or unstructured random noise due to their high expressivity. And they can still simultaneously generalize structured (correlated) data as well [45]. The generalization (approximation) performance of a model is tied to the labels randomization ratio (label noise) or signal-to-noise ratio *(SNR)* [12]. This finding is also true for relatively smaller feed-forward neural networks where the proposed complexity measure for Boolean functions highly correlates with the generalization error [46]. As the complexity measure of Boolean functions increases, the generalization error also increases [46]. Similarly, generalization abilities of various network architectures as a function of intrinsic dimension of data, called *task rank,* are tested [47]. Shallower networks perform better for higher task rank (data with higher intrinsic dimension). In contrast, deeper networks are more successful for lower task rank (data with lower intrinsic dimension) [47]. Thus, the signal statistics plays a crucial role in determining the best network architecture [47]. The proposed metrics to measure generalization capability of a network model are related to the correlation strength of data set and its signal-to-noise ratio. Besides, the empirical studies demonstrate that signal statistics should be taken into consideration when the generalization ability of a model is evaluated [12], [45], [46], [47]. The impact of signal statistics on network performance favors the use of signal-dependent sparsity techniques over signal-independent ones to sparsify a network optimally. We empirically relate the signal statistics at nodes and layers of MLP based classifier to its accuracy in Section V.

III. MATHEMATICAL OVERVIEW

We review some basic concepts of signal processing in the following sections that are utilized to develop the empirical framework used in this study.

A. AR(1) SOURCE MODEL

Autoregressive discrete process of order one, AR(1), provides a coarse approximation to natural signals like images [48]. AR(1) source generates signal sequence as

$$x(n) = \rho x(n-1) + \xi(n) + c \tag{1}$$

where ρ is the first order correlation coefficient, n is the discrete-time variable, c is a constant, and $\xi(n)$ is white noise with zero-mean and variance σ_ξ^2 [48]. The auto-correlation of $\xi(n)$ is expressed as

$$E\{\xi(n)\xi(n+k)\} = \sigma_\xi^2 \delta_{n-k} \tag{2}$$

where $E\{\cdot\}$ is the expectation and δ_k is the Kronecker delta function. The mean of $x(n)$ is found as [48]

$$\mu_x = E\{x(n)\} = \frac{c}{1-\rho} \tag{3}$$

The variance of $x(n)$ is calculated as

$$\sigma_x^2 = E\{x(n)^2\} - \mu_x^2 = \frac{\sigma_\xi^2}{1-\rho^2} \tag{4}$$

The auto-correlation sequence of $x(n)$ is written as

$$R_{xx}(k) = E\{x(n)x(n+k)\} = \sigma_x^2 \rho^{|k|};\ k = 0, \pm 1, \pm 2, \ldots \tag{5}$$

and ρ is found as

$$\rho = \frac{R_{xx}(1)}{R_{xx}(0)} = \frac{E\{x(n)x(n+1)\}}{E\{x(n)x(n)\}} \tag{6}$$

where $-1 < \rho < 1$ [48]. The auto-correlation matrix of AR(1) process is exponential and *Toeplitz* of size $N \times N$ as

$$R_x = \sigma_x^2 \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{N-1} \\ \rho & 1 & \rho & \cdots & \rho^{N-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{N-1} & \rho^{N-2} & \rho^{N-3} & \cdots & 1 \end{bmatrix} \tag{7}$$

The *signal-to-noise ratio (SNR)* of AR(1) source as a function of ρ is found as

$$SNR(\rho) = \frac{\sigma_x^2}{\sigma_\xi^2} = \frac{1}{1-\rho^2} \tag{8}$$

One of the two data types used in this study is generated by using AR(1) model as explained in Section IV-D1.

B. EIGEN SUBSPACE OF R_x

The eigen-decomposition of the matrix R_x in (7) is written as [48]

$$R_x = \Phi_{KLT}^T \Lambda \Phi_{KLT} = \sum_{k=0}^{N-1} \lambda_k \phi_k \phi_k^T \tag{9}$$

where ϕ_k is the k^{th} eigenvector of R_x, λ_k is the corresponding eigenvalue, and $\Lambda = diag(\lambda_k)$.

The Karhúnen-Loeve transform (KLT), also called principal component analysis (PCA), is the optimal orthonormal transform [48]. Its transform matrix for the given (correlation) matrix R_x is defined as

$$\Phi_{KLT} = [\phi_0 \ \phi_1 \ ... \ \phi_{(N-1)}]^T \tag{10}$$

The rows of Φ_{KLT} are eigenvectors of R_x as defined in (9) where $\Phi_{KLT}\Phi_{KLT}^{-1} = \Phi_{KLT}\Phi_{KLT}^T = I$, and I is $N \times N$ identity matrix.

N dimensional vector process $X = [x_0 \ x_1 \ ... \ x_{(N-1)}]^T$ is mapped onto its eigen-subspace through the forward KLT

$$\Theta_{KLT} = \Phi_{KLT} X \tag{11}$$

where Θ_{KLT} is the KLT *transform coefficients vector* as

$$\Theta_{KLT} = [\theta_0 \ \theta_1 \ \ldots \ \theta_{(N-1)}]^T \tag{12}$$

The inverse transformation perfectly reconstructs the signal vector X from Θ_{KLT} as

$$X = \Phi_{KLT}^{-1}\Theta_{KLT} = \Phi_{KLT}^T \Theta_{KLT} \tag{13}$$

The signal energy σ_x^2 is preserved and optimally repacked among transform coefficients (dimensions) in the eigen-subspace expressed as [48]

$$E\{\Theta_{KLT}^T \Theta_{KLT}\} = \sum_{k=0}^{N-1} E\{\theta_k^2\} = \sum_{k=0}^{N-1} \sigma_{\theta_k}^2 = N\sigma_x^2 \tag{14}$$

where $\sigma_{\theta_k}^2$ is the variance of the k^{th} coefficient θ_k. Note that $\{\lambda_k = \sigma_{\theta_k}^2\} \ \forall_k$ for KLT. One can show that $R_\theta = E\{\Theta_{KLT}\Theta_{KLT}^T\} = diag\{\lambda_k\}$.

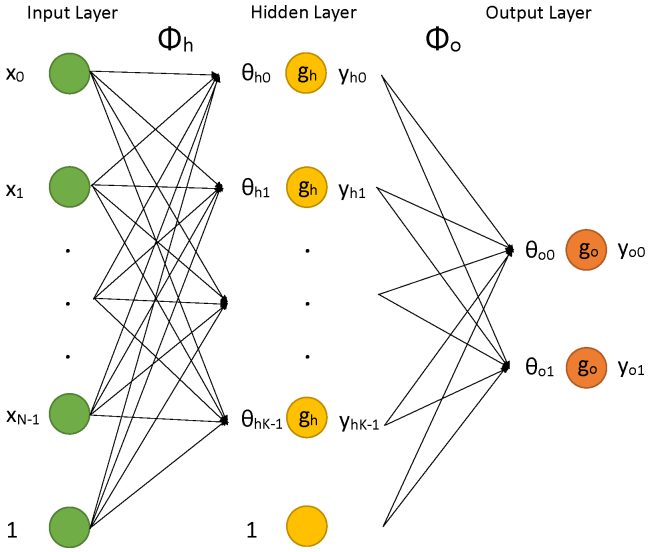

FIGURE 1. A single hidden layer Multilayer Perceptron (MLP) for the two-class case.

The subspace performance of KLT for AR(1) process as a function of ρ and N is displayed in Section IV-A as the theoretical basis for the proposed correlation-based node partitioning and network sparsification method in Section IV-B.

C. MULTILAYER PERCEPTRON

A shallow MLP consists of one or two hidden layers [1]. The two mappings of a typical shallow network with one hidden layer are $M_h = f[\Phi_h, g_h(.)]$ and $M_o = f[\Phi_o, g_o(.)]$ where Φ_h and Φ_o are $N \times K$ and $K \times L$ weight (transform) matrices, respectively. B_h and B_o are defined as the bias vectors of the hidden and output layers, respectively. The mapping functions $g_h(.)$ and $g_o(.)$ are the activation functions of the hidden and output layers, respectively. Therefore, $Y_h = g_h(\Theta_h)$, $\Theta_h = \Phi_h X + B_h$ and $Y_o = g_o(\Theta_o)$, $\Theta_o = \Phi_o Y_h + B_o$. These mappings are depicted in Fig. 1.

Alternatively, the forward propagation of a typical shallow neural network with one hidden layer comprised of K nodes is expressed in the following equations. Let X be N dimensional random vector process

$$X = [x_0 \ x_1 \ ... \ x_{(N-1)}]^T \tag{15}$$

where x_n is the n^{th} component (random variable) of X, $0 \le n \le N-1$, with zero-mean and unit variance. We can populate the rows of weight (transform) matrix Φ_h between the input and hidden layers with a set of K real vectors, discrete-time sequences $\{\phi_k(n)\}$, as

$$\Phi_h = [\phi_0 \ \phi_1 \ ... \ \phi_{(K-1)}]^T \tag{16}$$

The bias vector B_h is defined as

$$B_h = [\beta_0 \ \beta_1 \ ... \ \beta_{(K-1)}]^T \tag{17}$$

Then, the input of the hidden layer is obtained as

$$\Theta_h = \Phi_h X + B_h \tag{18}$$

FIGURE 2. A single node of a neural network. Activation function $g(\cdot)$ maps input x to output y.

where $\Theta_h = [\theta_0 \ \theta_1 \ \dots \ \theta_{(K-1)}]^T$. The components of Θ_h individually go through the same mapping (activation) function $g_h(.)$. The hidden layer output vector is populated by

$$Y_h = g_h(\Theta_h) \tag{19}$$

where $M_h = f[\Theta_h, g_h(.)]$ represents the hidden layer mapping from input X to the output Y_h. Similarly, the output vector of the hidden layer Y_h goes through the output layer mapping $M_o = f[\Phi_o, g_o(.)]$ as

$$\Theta_o = \Phi_o Y_h + B_o \tag{20}$$

and

$$Y_o = g_o(\Theta_o) \tag{21}$$

The components of the vector Y_o are L outputs of the network. The decisions of the network are made based on the component values of Y_o.

A typical MLP architecture with P hidden layers is built by concatenating hidden layers hierarchically [1]. The forward propagation of a MLP with P hidden layers is expressed as

$$Y_p = g_p(\Phi_p g_{p-1}(\Phi_{p-1} \cdots g_1(\Phi_1 X + B_1) \cdots + B_{p-1}) + B_p) \tag{22}$$

where $\Phi_p \in \mathbb{R}^{N_p \times N_{(p-1)}}$, N_p is the p^{th} layer dimension (width), and the mapping functions of hidden layers are $g_p(.)$, $1 \leq p \leq P$. Similarly, the output mapping is described as

$$Y_o = g_o(\Phi_o Y_p + B_o) \tag{23}$$

where $\Phi_o \in \mathbb{R}^{N_o \times N_p}$ is the weight matrix between the last hidden and the output layers. We use various MLP architectures and compare their performances for two different data sets in Section V.

D. NODE STATISTICS

A single node of a neural network is displayed in Fig. 2 with input X and output Y.

The activation function $g(\cdot)$ is monotonic and nonlinear. The node output is expressed as

$$Y = g(X) \tag{24}$$

The input random variable X of a node is a weighted sum of multiple random variables (node outputs from the previous layer). The probability density function *(pdf)* of node output $f_Y(y)$ is calculated from its input *pdf* $f_X(x)$ as [49]

$$f_Y(y) = \frac{f_X(x)}{|\frac{\partial Y}{\partial X}|} \tag{25}$$

These two relationships dictate the statistics of node output while weight coefficients are optimized during the training of a network. The transcendental nature of such optimization process is the root-cause of built-in explainability concern in learning networks. Additionally, joint statistics of node outputs play a central role in system performance. We observe that optimization process reshapes the marginal and joint statistics of nodal inputs and outputs during training while searching for optimal weights. This process forces some node inputs to be misaligned (mean and/or variance mismatch) with their activation functions. Such nodes have entropy compression from their inputs to outputs. We quantify this *implicit regularization* phenomenon by using the two entropy based metrics introduced in Section III-E and relate them to the system performance. Similar observations discussed in Section I also support the existence of *implicit regularization* by reducing the number of free model parameters during the performance optimization of an underdetermined system. Sparsifying an over-parameterized neural network is another way to reduce the number of optimized parameters that we also investigate in this study.

The (differential) entropy of a continuous information source with *pdf* $f_X(x)$ is calculated as [50]

$$E(x) = -\int_{-\infty}^{+\infty} f_X(x)\, ln[f_X(x)]dx \tag{26}$$

Similarly, entropy of a discrete information source with N symbols $\{x_i\}$ with probabilities $\{p_i\}$ is calculated (in *bits)* as [50]

$$E_x = -\sum_{i=1}^{N} p_i log_2 p_i \tag{27}$$

Entropies of node inputs and outputs in a network are measured. They are used to calculate node and layer compression ratios as discussed in Section III-E.

E. ENTROPY COMPRESSION IN NODES AND LAYERS

We introduce entropy compression metrics for nodes and layers of a network and highlight their empirical relationships with the classifier accuracy later in the paper.

1) NODE COMPRESSION RATIO

We introduce the *node compression ratio (NCR)* to measure the information-theoretic behavior of a node as [24]

$$\eta_{E_i} = \frac{E^i_{in}}{E^i_{out}} \tag{28}$$

where E^i_{in} and E^i_{out} are the input and output entropies of node i, respectively. A node with high *node compression ratio* has a misalignment between its input *pdf* $f_X(x)$ and the activation function $g(\cdot)$. It causes significant information loss (entropy reduction) at the node output.

The input and output *pdfs* including E^i_{in} and E^i_{out} with Sigmoid activation for the aligned (high-energy) and misaligned (low-energy) node examples are displayed in Figs. 3 and 4, respectively. These figures show the dramatic differences between *pdf* shapes of those nodes caused in

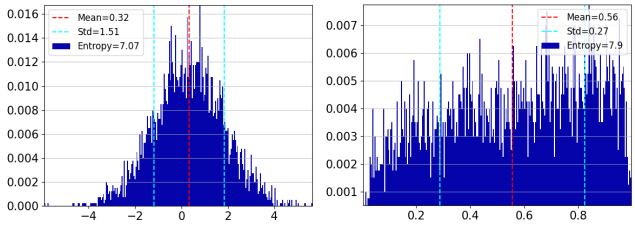

FIGURE 3. The input and output *pdfs* of an *aligned (high-energy) node* along with their input entropy E^i_{in} and output entropy E^i_{out} for Sigmoid activation function.

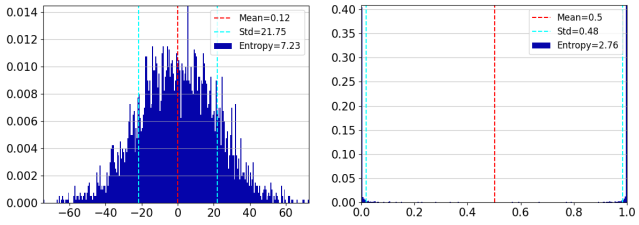

FIGURE 4. The input and output *pdfs* of a *misaligned (low-energy) node* along with their input entropy E^i_{in} and output entropy E^i_{out} for Sigmoid activation function.

principle by the optimizer. The latter is a good example of *implicit regularization* (node sparsity) dictated by the optimization process.

2) LAYER COMPRESSION RATIO

Similarly, the *layer compression ratio (LCR)* is defined as the average of *NCRs* in a layer

$$\eta_E = \frac{1}{N}\sum_{i=0}^{N-1}\eta_{E_i} \tag{29}$$

Node and layer compression ratios are used to quantify the combined impact of activation and optimizer as implicit regularizer in the optimization of under-determined system as discussed in Section V.

IV. CORRELATION BASED NODE PARTITIONING AND WEIGHT SPARSITY

A. EIGEN SUBSPACE PERFORMANCE

The gain of transform coding (G_{TC}) over pulse code modulation (PCM) is an information-theoretic metric to assess subspace performance. It measures the merit of representing a signal in a subspace rather than signal space. It is defined as [48]

$$G^N_{TC} = \frac{\frac{1}{N}\sum_{k=0}^{N-1}\sigma^2_{\theta_k}}{[\prod_{k=0}^{N-1}\sigma^2_{\theta_k}]^{\frac{1}{N}}} \tag{30}$$

where $\sigma^2_{\theta_k}$ is the variance of the k^{th} transform coefficient (see (11) and (14)) and N is the dimension [48]. It is noted that $\{\lambda_k = \sigma^2_{\theta_k}\}\ \forall_k$ for eigen-subspace representation (KLT) of signals (see Section III-B).

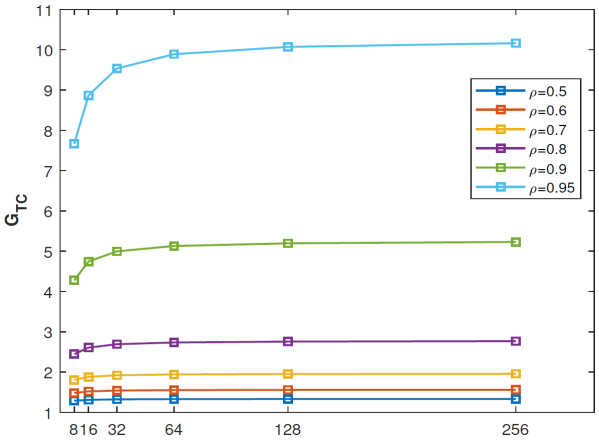

FIGURE 5. G_{TC} performance of eigen subspace (KLT) for AR(1) signal of various correlation coefficients, $\rho = \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}$and dimensions.

Source: [51].

G_{TC} performances of eigen-subspace representation (KLT) for AR(1) signal with various correlation coefficients (ρ) and dimensions (N) are displayed in Fig. 5. It is observed from the figure that G_{TC} increases as dimension increases and then saturates at some N value [48], [51]. It is also noted that higher values of ρ show higher G_{TC} performance [48], [51]. Most importantly, Fig. 5 demonstrates that G_{TC} performance of eigen subspace (KLT) for a lower-dimensional and higher correlated AR(1) is better than higher-dimensional and lower correlated one (compare G_{TC} values for $\{\rho = 0.9, N = 8\}$ and $\{\rho = 0.8, N = 256\}$ in Fig. 5) [51]. This fact is the theoretical basis to develop a method for correlation-based partitioning (clustering) of nodes in a layer prior to connecting them blindly to the next layer nodes. Such a method divides N nodes of a layer into L partitions where intra-partition correlations are increased. Consequently, partitions possess their selected subsets of higher correlated nodes. These node partitions with higher G_{TC} are connected to different nodes of the next layer as their inputs [51]. It is more effectively used when random variables of input vector process are not highly correlated. This idea was successfully implemented for the design of eigenportfolios with improved financial performance [51]. We are extending it to develop the correlation-based pruning method to sparsify network models as explained in the next section.

B. CORRELATION BASED PRUNING

Inter-layer mappings of a deep neural network are viewed as subspace representations. An input data vector propagates through each layer and transformed onto a new subspace. Each layer learns a representation (weight matrix) of its input data vector based on node outputs of the previous layer and a specific cost function. It is noted that node outputs of a hidden or input layer in a network are not always highly correlated. The subspace representation is quite poor for such a case as discussed in Section IV-A. We develop the

novel weight pruning method to sparsify MLP by adopting a correlation-based partitioning of nodes [51]. It increases intra-partition node correlations. The outputs of such a node partition are the only inputs for a single node in the next layer.

The dense MLP with P hidden layers is fully trained to the convergence. This pruning method partitions nodes of each layer successively by using Pearson correlation matrix of layer output vectors and a pre-defined correlation threshold. The pseudo-code for the correlation-based partitioning of node outputs in a layer is shown in Algorithm 1. The optimal correlation threshold is signal-dependent for each $\{\rho_0, \rho_1\}$ pair set ($\Delta\rho$ and *SNR* data sets as tabulated in Table 1). This process increases intra-partition correlations while decreasing inter-partition ones. After this step, all nodes within a partition are fully connected to a node in the next layer as its only inputs. The node outputs of other partitions are not inputs to that node anymore. It results in a sparse inter-layer weight matrix for each layer. Note that the weight matrix between the last hidden and output layers is not sparsified in this study.

All inter-layer weight matrices of multiple hidden layer MLP are progressively sparsified by using this pruning method. The MLP is re-trained to fine tune the model after each weight matrix is sparsified. The progressive nature of sparsification process reduces the chance of getting stuck in a poorly performing local minimum point in the search region [41], [44]. Note that this pruning method may also be applied to all weight matrices at once and network is re-trained only once if more efficient implementation is desired.

We define the *sparsity ratio* of the weight matrix between the p^{th} and $(p+1)^{th}$ layers of a MLP as

$$SR_p = 100 \times \left[1 - \frac{\sum_{i=0}^{N_{(p+1)}-1} \sum_{j=0}^{N_p-1} S(i,j)}{N_{(p+1)} N_p}\right] \quad (31)$$

where $S \in \mathbb{R}^{N_{(p+1)} \times N_p}$ is the binary *sparsity mask* comprised of 1's and 0's. $S(i,j) = 1$ indicates a connection between the node pair (i,j) of layers $p+1$ and p. The parameter N_p is the dimension of the p^{th} layer (width).

C. IMPLEMENTATION

The training is performed by using mini-batch gradient descent with the batch size of 100 data vectors. Weight matrices are initialized by uniformly distributed random values in the range -1 and 1. The mean squared error *(MSE)* is used as the loss function. AR(1) and CIFAR-10 data sets as explained in Section IV-D for the two-class case are used for experiments. Input and hidden layer dimensions of all MLPs are assumed to be the same. Accuracies of MLPs are calculated with respect to various architectural core parameters (layer dimension, network depth, and network sparsity) with Sigmoid activation functions in the hidden and output layer nodes. The correlation-based pruning method as explained in Section IV-B sparsifies weight matrices of a MLP layer-by-layer followed by a re-training step. A MLP with P hidden layers is sparsified in P re-training steps. Magnitude and random pruning methods sparsify all weight matrices at once proceeded by a single re-training step. Sparsifying a fully trained network (with P hidden layers) in one single step is the most common schedule type [23]. Note that all these pruning methods perform one single re-training step to sparsify a single hidden layer MLP. The weight matrix between the last hidden layer and the output layer is kept dense for all experiments.

The magnitude and random pruning methods cannot dynamically change the nodal architecture of network. Therefore, we did not implement node sparsity and kept the dimension of input and hidden layers fixed in this study. We kept redundant nodes as necessary in the next layer in order to avoid node sparsity. This forced node redundancy degrades performance and prevents the correlation-based pruning to reach its full potential. The self-reconfiguring network design is beyond the scope of this paper.

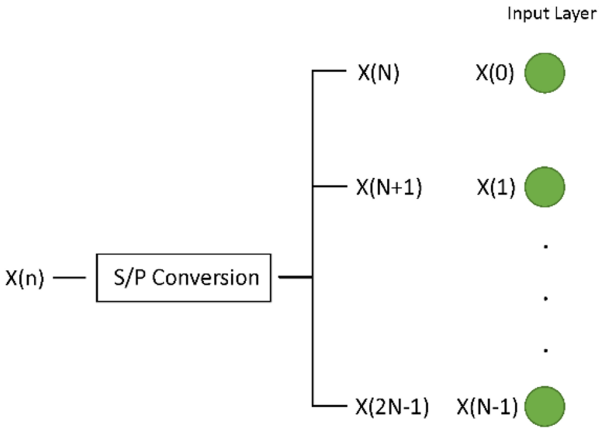

FIGURE 6. Serial to Parallel (S/P) conversion to generate data vectors of dimension N from a sequence *X(n)*.

D. DATA SETS

1) AR(1) DATA SET

We simulated a classifier for the two-class case using AR(1) model generated training and test data sets. The two classes (class 0 and class 1) are defined by their correlation coefficients $\{\rho_0, \rho_1\}$ for the AR(1) model, respectively. We followed the steps given below to generate each *class data* with a distinct *class correlation coefficient*.

1) The i^{th} AR(1) class sequence is generated for the given class correlation coefficient ρ_i, $i = \{0, 1\}$.
2) 8-bit uniform quantization of the i^{th} AR(1) class sequence is performed to generate one byte per sample resolution.
3) The serial to parallel conversion (S/P) is applied to generate N-dimensional data vectors for the i^{th} AR(1) class sequence, see Fig. 6.
4) The steps above are repeated to generate data for all classes with distinct class correlation coefficient.

The same correlation coefficient pair $\{\rho_0, \rho_1\}$ is used to generate the training and test data sets for each experiment. AR(1) data vectors of two classes are concatenated to form the training and test data sets. Each class has $10,000$ training and $2,000$ test data vectors. In total, each $\{\rho_0, \rho_1\}$ pair of AR(1) data set experiment has $20,000$ training and $4,000$ test data vectors of dimension N. Labels for each class are created and the data sets and labels are randomly shuffled. Each data set is normalized to zero mean and unit variance in each dimension (random variable).

The *signal-to-noise ratio (SNR)* of training and test data set for a two-class problem is expressed as

$$SNR(\rho) = \frac{\sigma_x^2}{\sigma_\xi^2} = \frac{1}{1-\rho^2} \tag{32}$$

where ρ is the average correlation coefficient of the two classes as

$$\rho = \frac{\rho_0 + \rho_1}{2} \tag{33}$$

We look at the relationship between accuracy and *SNR* range of $\{\rho_0, \rho_1\}$ pair as well as their correlation distance $\Delta\rho = |\rho_0 - \rho_1|$ in Section V. These two distinct data sets of $\{\rho_0, \rho_1\}$ pairs for two classes are generated as described next.

a: CORRELATION COEFFICIENT PAIRS FOR TWO CLASSES

We generate two distinct data sets of $\{\rho_0, \rho_1\}$ pairs to relate the accuracy of the two-class classifier as a function of AR(1) signal statistics. The first one is called $\Delta\rho$ *set* for $\{\rho_0, \rho_1\}$ pair with correlation distance $\Delta\rho = |\rho_0 - \rho_1|$ between the two classes. We examine the relationship between accuracy and the correlation distance $\Delta\rho$. The second one is the *SNR set* for various $\{\rho_0, \rho_1\}$ pairs with their *SNRs* for fixed correlation distance $\Delta\rho$. We look at the relationship between accuracy and *SNR* of $\{\rho_0, \rho_1\}$ pair. *SNR* data sets emphasize the significance of input signal correlation on performance. The empirical performance for both data sets are presented in Section V-A. These results show the benefit of tracing signal statistics in the network and their relations to classifier performance. This approach leads to the development of signal-dependent framework to design neural networks.

2) CIFAR-10 DATA SET

The CIFAR-10 data set includes $60,000$ 32×32 pixels size color images. The data set consists of 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). Each class has $5,000$ training and $1,000$ test images. We convert CIFAR-10 color images to gray-scale images of $1,024$ pixels. We randomly select 20 class pairs from CIFAR-10 data set to show the relationship between input signal statistics ($\Delta\rho$ and *SNR*) and classifier performance. Each class pair is comprised of $10,000$ training and $2,000$ test images. Each data set is normalized to zero mean and unit variance in each dimension. The classifier accuracies for CIFAR-10 class pairs along with their measured $\{\rho_0, \rho_1\}$, $\Delta\rho$, *SNR*, and $SR_1(\%)$ values are presented and discussed in Section V-B.

TABLE 1. AR(1) $\Delta\rho$ and *SNR Data Sets* of Various Correlation Coefficient Pairs for *Two-Class* Case and Their *SNRs*

$\Delta\rho$ Data Sets			SNR Data Sets		
$\{\rho_0, \rho_1\}$	$\Delta\rho$	SNR	$\{\rho_0, \rho_1\}$	$\Delta\rho$	SNR
0, 0	0	1	0.15, 0.05	0.1	1.01
0.1, 0	0.1	1	0.25, 0.15	0.1	1.04
0.2, 0	0.2	1.01	0.35, 0.25	0.1	1.1
0.3, 0	0.3	1.02	0.45, 0.35	0.1	1.19
0.4, 0	0.4	1.04	0.55, 0.45	0.1	1.33
0.5, 0	0.5	1.07	0.65, 0.55	0.1	1.56
0.6, 0	0.6	1.1	0.75, 0.65	0.1	1.96
0.7, 0	0.7	1.14	0.85, 0.75	0.1	2.78
0.8, 0	0.8	1.19	0.95, 0.85	0.1	5.26
0.9, 0	0.9	1.25			
0.99, 0	0.99	1.32			

V. PERFORMANCE COMPARISONS

We simulated MLP based two-class classifier to measure its accuracy for the dense and weight sparsed network architectures. The results for the AR(1) model and CIFAR-10 data set types are presented in the following sections.

A. ACCURACY FOR AR(1) DATA

Tables 2 and 3 display the two-class accuracies of the *fully-* and *sparsely-connected* single hidden layer MLP based classifier for several $\Delta\rho$ and *SNR* data sets with $N = 1,024$, respectively. The results are consistent that the sparsified MLPs do not malignantly overfit and they outperform their fully-connected counterparts. It is also observed that the correlation-based pruning of MLP consistently outperforms the magnitude- and the random-pruning based designs.

The results show the relationships between data statistics ($\Delta\rho$ and *SNR*) and accuracy. It is shown in Tables 3, 5, 6, 7 and 8 that accuracy increases as the correlation distance $\Delta\rho = |\rho_1 - \rho_0|$ of the class pair increases. It is also seen that the accuracies decrease as the signal-to-noise ratio *(SNR)* gets smaller even when $\Delta\rho$ is constant (see Tables 2 and 4).

It is noted from Tables 2 and 3 that the *accuracy* and the *layer compression ratio* $\eta_E^{h_1}$ of the hidden layer are inversely correlated with each other.

1) IMPACT OF DIMENSION

We present accuracies of the single hidden layer *fully-* and *sparsely-connected* MLPs for the two-class case with dimensions, $N = \{16, 64, 256, 1024\}$, in Table 4. The performance of sparsified MLP with correlation-based pruning increases as its layer dimension (width) N increases. However, the performance of the fully-connected MLP and sparsified MLPs with the magnitude- and random-based methods deteriorate

TABLE 2. The *Accuracy* and *Layer Compression Ratio (LCR)* Measurements of the *Single Hidden Layer* MLPs with *Layer Dimension (Width) N=1,024* for the *SNR Data Sets* of Various Correlation Coefficient Pairs

SNR Data Sets			Fully-Connected			Correlation		Magnitude		Random	
$\{\rho_0, \rho_1\}$	$\Delta\rho$	SNR	Acc	$\eta_E^{h_1}$	$SR_1(\%)$	Acc	$\eta_E^{h_1}$	Acc	$\eta_E^{h_1}$	Acc	$\eta_E^{h_1}$
0.15, 0.05	0.1	1.01	50.95	2.26	99.58	**79.68**	0.97	49.58	1.09	50.25	0.95
0.25, 0.15	0.1	1.04	52.02	2.26	99.48	**80.65**	0.97	49.52	1.09	50.92	0.95
0.35, 0.25	0.1	1.1	50.7	2.26	99.4	**82.48**	0.96	50.2	1.08	50.15	0.95
0.45, 0.35	0.1	1.19	50.95	2.26	99.31	**83.48**	0.96	49.7	1.08	50.45	0.95
0.55, 0.45	0.1	1.33	50.75	2.26	99.25	**87.25**	0.96	49.55	1.08	51.78	0.95
0.65, 0.55	0.1	1.56	49.98	2.26	99.16	**90.02**	0.96	50.08	1.09	52.05	0.95
0.75, 0.65	0.1	1.96	49.6	2.27	99.05	**94.75**	0.95	50.5	1.1	57.88	0.95
0.85, 0.75	0.1	2.78	50.75	2.27	98.76	**97.32**	0.94	52.85	1.1	69.03	0.94
0.95, 0.85	0.1	5.26	50.2	2.29	98.54	**99.78**	0.93	83.88	1.1	96	0.93

TABLE 3. The *Accuracy* and *Layer Compression Ratio (LCR)* Measurements of the *Single Hidden Layer* MLPs with *Layer Dimension (Width) N=1,024* for the $\Delta\rho$ *Data Sets* of Various Correlation Coefficient Pairs

$\Delta\rho$ Data Sets			Fully-Connected			Correlation		Magnitude		Random	
$\{\rho_0, \rho_1\}$	$\Delta\rho$	SNR	Acc	$\eta_E^{h_1}$	$SR_1(\%)$	Acc	$\eta_E^{h_1}$	Acc	$\eta_E^{h_1}$	Acc	$\eta_E^{h_1}$
0, 0	0	1	**51.08**	2.26	99.63	49.6	0.97	50.32	1.1	49.85	0.95
0.1, 0	0.1	1	51.1	2.26	99.46	**76.92**	0.97	50.42	1.08	49.55	0.95
0.2, 0	0.2	1.01	50.72	2.26	99.25	**94.75**	0.95	49.75	1.07	53.78	0.95
0.3, 0	0.3	1.02	52.08	2.26	98.95	**98.8**	0.94	49.42	1.07	61.62	0.94
0.4, 0	0.4	1.04	51.72	2.26	98.6	**99.68**	0.93	52.2	1.08	70.2	0.94
0.5, 0	0.5	1.07	51.88	2.26	98.33	**99.75**	0.92	53.5	1.1	76.88	0.94
0.6, 0	0.6	1.1	51.35	2.26	97.92	**99.95**	0.92	55.22	1.13	84.3	0.94
0.7, 0	0.7	1.14	51.82	2.26	97.48	**99.98**	0.92	55.7	1.16	88.22	0.95
0.8, 0	0.8	1.19	51.1	2.26	96.96	**99.98**	0.91	60.68	1.22	92.78	0.97
0.9, 0	0.9	1.25	50.82	2.26	95.35	**99.92**	0.91	70.75	1.33	91.28	1.02
0.99, 0	0.99	1.32	67.22	2.23	84.59	**99.98**	0.9	69.8	1.73	98.7	1.28

after the layer dimension of $N = 64$. The correlation-based pruning shows superior performance to the other two methods for all test scenarios considered in the paper. The superiority of the correlation-based pruning method is more visible in larger dimensions, $N = \{256, 1024\}$. This performance improvement is due to the efficient partitioning of node outputs for the AR(1) generated input data sets with exponential correlation model, see (7). It is observed from Table 4 that magnitude and random pruning methods perform comparable when *SNR* is lower. In contrast, the performance of random pruning is significantly better than the magnitude pruning and the dense network when *SNR* and N are higher.

B. ACCURACY FOR CIFAR-10 DATA SET

We tabulated accuracy of single hidden layer, dense and sparsified (with the three pruning methods) MLPs for CIFAR-10 class pairs in Table 5. The table also includes the measured $\{\rho_0, \rho_1\}$, $\Delta\rho$, *SNR*, and $SR_1(\%)$ values for each pair. Note that the three pruning methods are tuned for the same $SR_1(\%)$ in each case. The sparse MLP with correlation-based pruning is superior to the other two methods for the CIFAR-10 data set. The correlation-based pruning works more effectively for weakly correlated data sets. The CIFAR-10 class pairs have high correlation (high *SNR*) and small correlation distance $\Delta\rho$. As stated in Section IV-C, the inefficient implementation of correlation-based pruning in this study inserts nodal redundancy to avoid node sparsity. It causes performance degradation. One should exploit both node and weight sparsities when using this pruning technique for a typical network design problem in real world. It is emphasized that the sparse implementations increase accuracy and they significantly reduce computational cost of implementation.

The superiority of the correlation-based pruning method is more significant for the AR(1) set than the CIFAR-10 data. Although AR(1) model is a rough approximation to real world signals, the values of correlation coefficients for the CIFAR-10 pairs are still related to MLP performance. Table 5 demonstrates the fact that accuracy also increases as the $\Delta\rho$ of a CIFAR-10 pair gets larger.

C. REMARKS

The experimental results for AR(1) and CIFAR-10 data sets measure the impact of signal statistics ($\Delta\rho$ and *SNR*) on MLP performance. Our observations on empirical performance results are summarized as follows.

Remark 1:
The accuracy of MLP improves when $\Delta\rho$ increases (see Tables 3, 5, 6, 7 and 8).

TABLE 4. The Impact of the *Layer Dimension (Width) N* on the *Accuracy* of the *Single Hidden Layer* MLPs for the *SNR Data Sets* of Various Correlation Coefficient Pairs

SNR Data Sets		N=16					N=64					N=256					N=1,024				
$\{\rho_0, \rho_1\}$	SNR	Dense	Corr	Mag	Rand	$SR_1(\%)$	Dense	Corr	Mag	Rand	$SR_1(\%)$	Dense	Corr	Mag	Rand	$SR_1(\%)$	Dense	Corr	Mag	Rand	$SR_1(\%)$
0.15, 0.05	1.01	**55.8**	55.58	53.88	53.82	61.72	53.28	**57.42**	50.22	50.85	92.87	50.32	**68.75**	51.58	52.15	98.4	50.95	**79.68**	49.58	50.25	99.58
0.25, 0.15	1.04	**55.8**	55.75	53.35	53.48	74.22	52.18	**59.28**	51.7	50.82	93.41	48.98	**70.35**	51.62	53.02	98.19	52.02	**80.65**	49.52	50.92	99.48
0.35, 0.25	1.1	**56.22**	56.1	54.25	55.05	61.72	52.7	**59**	51.6	51.45	92.53	49.08	**70.65**	50.7	54.3	97.92	50.7	**82.48**	50.2	50.15	99.4
0.45, 0.35	1.19	**56.75**	55.85	55.85	54.8	64.84	53.35	**59.22**	53.82	52.85	92.53	50.52	**73.82**	51.6	54.75	97.89	50.95	**83.48**	49.7	50.45	99.31
0.55, 0.45	1.33	58.48	**59.62**	58.5	57.25	64.84	55.28	**62**	55.8	55.45	91.8	50.25	**76.3**	52.32	55	97.59	50.75	**87.25**	49.55	51.78	99.25
0.65, 0.55	1.56	60.42	**61.65**	60.92	59.12	64.84	59.25	**61.65**	58.9	59.38	91.26	50.42	**80.9**	54.52	58.82	97.27	49.98	**90.02**	50.08	52.05	99.16
0.75, 0.65	1.96	**65.05**	64.6	63.9	63.55	64.84	**64.8**	60.92	63.02	64.45	90.72	50.52	**87.02**	55.8	64.8	96.89	49.6	**94.75**	50.5	57.88	99.05
0.85, 0.75	2.78	72.45	72.2	**73.1**	70.6	64.84	74.3	**78.25**	77.5	74.38	91.16	50.95	**95.4**	71.12	80.68	96.26	50.75	**97.32**	52.85	69.03	98.76
0.95, 0.85	5.26	87.98	**89.8**	89.55	86.45	49.22	95.58	**98.55**	93.72	96.5	91.55	85.18	**99.68**	94.7	97.42	97.07	50.2	**99.78**	83.88	96	98.54

TABLE 5. Performance Comparison of the *Weight Pruning Methods* for a series of *CIFAR-10 Class Pairs* and the *Single Hidden Layer* MLPs with *Layer Dimension (Width) N=1,024*

	$\{\rho_0, \rho_1\}$	$\Delta\rho$	SNR	Dense	Corr	Mag	Rand	$SR_1(\%)$
Cat, Dog	0.868, 0.871	0.003	4.1	**63.9**	62.1	63.5	60.5	90.86
Cat, Horse	0.868, 0.858	0.01	3.92	77.9	**79.5**	77.25	77.4	89.01
Dog, Horse	0.871, 0.858	0.013	3.96	75.8	**77.3**	74.85	75.25	77.04
Bird, Cat	0.852, 0.868	0.016	3.84	68.55	**69.65**	68.1	69.1	17.3
Bird, Dog	0.852, 0.871	0.019	3.88	**72.65**	72.45	71.65	72.25	92.81
Airplane, Dog	0.893, 0.871	0.022	4.5	84.5	**86.2**	83.85	84.4	89.27
Airplane, Cat	0.893, 0.868	0.025	4.45	**84.1**	83.45	83.05	82.55	74.42
Airplane, Ship	0.893, 0.924	0.031	5.73	76.55	**77.6**	76.25	74.95	66.77
Bird, Frog	0.852, 0.821	0.031	3.33	74	75.05	**75.85**	**75.85**	35.5
Airplane, Horse	0.893, 0.858	0.035	4.28	84.65	**86.15**	84.25	84.55	69.03
Frog, Horse	0.821, 0.858	0.037	3.39	84.4	**86.2**	84.05	84.9	79.3
Airplane, Bird	0.893, 0.852	0.041	4.19	73.8	**76.6**	76.15	**76.6**	72.97
Cat, Frog	0.868, 0.821	0.047	3.49	73.05	**75.25**	71.75	72.6	84.16
Dog, Frog	0.871, 0.821	0.05	3.52	78.7	**80.1**	78.6	79.85	95.4
Dog, Ship	0.871, 0.924	0.053	5.14	87.9	**89.35**	87.35	87.95	74.23
Cat, Ship	0.868, 0.924	0.056	5.07	86.25	**87.5**	86.55	85.85	79.08
Horse, Ship	0.858, 0.924	0.066	4.85	87.9	**89.8**	86.25	86.1	77.1
Bird, Ship	0.852, 0.924	0.072	4.73	86	86.65	87.55	**87.95**	58.31
Airplane, Frog	0.893, 0.821	0.072	3.77	84.7	**84.75**	84.1	83.7	59.34
Frog, Ship	0.821, 0.924	0.103	4.19	87.1	**88.05**	86.45	87.25	61.24
Average				79.62	**80.69**	79.37	79.48	72.16

The shape of the *deep double-descent curve* is regulated by the input data statistics such as the label noise, signal-to-noise ratio, and the effective rank of the given data set [6], [47], [52]. Less parameterized neural networks work better in high effective rank (weakly correlated) data sets [47].

Remark 2:
The accuracy of MLP rises when the *SNR* of data set increases (see Tables 2 and 4).

The optimization of the fully-connected single hidden layer MLPs falls in the under-parameterized regime with dimensions $N = \{16, 64\}$ for the AR(1) sets. On the other hand, the optimization improves performance in the over-parameterized regime with dimensions, $N = \{256, 1024\}$. We observe from the results that the performance of a dense MLP increases until $N = 64$ (see Table 4). Then, the performance drops for $N > 64$. This behavior supports the classical *bias-variance* trade-off [53]. The *benign overfitting* phenomenon is not observed in the over-parameterized regime, $N = \{256, 1024\}$ for various MLP architectures with the AR(1) sets. However, the *benign overfitting* phenomenon does not claim that the larger model size is always better for all neural network architectures and all data sets [2], [6], [11], [47], [52]. It is not universal and depends on the network architecture and input signal statistics under consideration [2], [6], [11], [47], [52].

Remark 3:
Benign overfitting phenomenon is not observed for the fully-connected MLPs with dimensions $N = \{256, 1024\}$ on AR(1) sets (see Table 4).

The correlation strengths of the AR(1) sets are not only dictated by *SNR* but also by dimension N of the input. The input correlation matrices R_x of AR(1) sets include much weaker correlations especially for larger dimensions like $N = \{256, 1024\}$ due to the nature of the exponential correlation given in (7). An over-parameterized, fully-connected MLP is less impacted negatively by the malign overfitting when it is trained on a data set with higher *SNR* (see the correlation coefficient pair of $\{0.95, 0.85\}$) for the fully-connected MLP with dimension $N = 256$ in Table 4). The performances of the over-parameterized, dense MLPs with layer dimensions $N = \{256, 1024\}$ are lower than the performances of the less parameterized, dense MLPs with dimensions $N = \{16, 64\}$ and their sparse counterparts (see Table 4).

Remark 4:
Malign overfitting is more devastating in the weakly-correlated AR(1) sets (with lower *SNRs* and/or higher dimensions N).

Less parameterized MLPs are more desirable especially for the weakly-correlated data sets (see Table 4).

These results and observations are in support of the theoretical intuition with respect to signal correlation, layer dimension N, and reduction of model parameters in the optimization of an under-determined system. Over-parameterizing neural networks by increasing layer width or network depth should be decided carefully considering data statistics and available resources for implementation [11], [47], [52].

Remark 5:

TABLE 6. The Impact of the *Network Depth* on the *Layer Compression Ratio (LCR)* and *Accuracy* of the *Fully-Connected* MLPs with *Layer Dimension (Width) N=1,024* for the *Single*, *Double*, and *Triple Hidden Layer* Cases and the $\Delta\rho$ *Data Sets* of Various Correlation Coefficient Pairs

$\Delta\rho$ Data Sets			Single		Double			Triple			
$\{\rho_0, \rho_1\}$	$\Delta\rho$	SNR	Acc	$\eta_E^{h_1}$	Acc	$\eta_E^{h_1}$	$\eta_E^{h_2}$	Acc	$\eta_E^{h_1}$	$\eta_E^{h_2}$	$\eta_E^{h_3}$
0, 0	0	1	51.08	2.26	49.9	2.26	4.44	50.9	2.25	3.45	12.55
0.1, 0	0.1	1	51.1	2.26	50	2.26	4.33	50.4	2.25	3.40	11.43
0.2, 0	0.2	1.01	50.72	2.26	51.7	2.26	4.11	49.92	2.25	3.40	9.95
0.3, 0	0.3	1.02	52.08	2.26	51.12	2.26	3.87	51.32	2.25	3.37	12.42
0.4, 0	0.4	1.04	51.72	2.26	53.8	2.26	4.13	52.48	2.25	3.40	10.74
0.5, 0	0.5	1.07	51.88	2.26	55.78	2.26	4.24	53	2.25	3.42	10.39
0.6, 0	0.6	1.1	51.35	2.26	60.68	2.26	4.27	54.5	2.25	3.39	11.16
0.7, 0	0.7	1.14	51.82	2.26	69.82	2.26	4.36	56.78	2.25	3.40	10.80
0.8, 0	0.8	1.19	51.1	2.26	83.55	2.26	4.12	62.05	2.25	3.46	11.45
0.9, 0	0.9	1.25	50.82	2.26	95.88	2.26	4.3	74.55	2.25	3.50	13.23
0.99, 0	0.99	1.32	67.22	2.23	99.7	2.25	4.05	97.5	2.22	3.90	12.78

TABLE 7. The Impact of the *Network Depth* on the *Layer Compression Ratio (LCR)* and *Accuracy* of the *Sparsely-Connected* MLPs with *Layer Dimension (Width) N=1,024* using *Correlation-Based Pruning* for the *Single*, *Double*, and *Triple Hidden Layer* Cases and the $\Delta\rho$ *Data Sets* of Various Correlation Coefficient Pairs

$\Delta\rho$ Data Sets			Single			Double			
$\{\rho_0, \rho_1\}$	$\Delta\rho$	SNR	Acc	$SR_1(\%)$	$\eta_E^{h_1}$	Acc	$SR_1(\%)$	$SR_2(\%)$	$\eta_E^{h_1}$	$\eta_E^{h_2}$
0, 0	0	1	49.6	99.63	0.97	49.52	99.63	93.89	0.95	1.1
0.1, 0	0.1	1	76.92	99.46	0.97	69.1	99.46	94.43	0.97	1.03
0.2, 0	0.2	1.01	94.75	99.25	0.95	93.5	99.25	94.81	0.96	1
0.3, 0	0.3	1.02	98.8	98.95	0.94	97.82	98.95	93.71	0.94	1
0.4, 0	0.4	1.04	99.68	98.6	0.93	98.9	98.6	94.14	0.93	1
0.5, 0	0.5	1.07	99.75	98.33	0.92	99.35	98.33	91.88	0.93	1
0.6, 0	0.6	1.1	99.95	97.92	0.92	99.58	97.92	94.54	0.92	1
0.7, 0	0.7	1.14	99.98	97.48	0.92	99.95	97.48	93.85	0.91	1
0.8, 0	0.8	1.19	99.98	96.96	0.91	99.98	96.96	94.4	0.9	1
0.9, 0	0.9	1.25	99.92	95.35	0.91	99.82	95.35	93.19	0.88	1.02
0.99, 0	0.99	1.32	99.98	84.59	0.9	100	84.59	79.98	0.87	1.05

$\Delta\rho$ Data Sets			Triple						
$\{\rho_0, \rho_1\}$	$\Delta\rho$	SNR	Acc	$SR_1(\%)$	$SR_2(\%)$	$SR_3(\%)$	$\eta_E^{h_1}$	$\eta_E^{h_2}$	$\eta_E^{h_3}$
0, 0	0	1	50.02	99.63	94.38	97.97	0.98	1.04	0.98
0.1, 0	0.1	1	74.3	99.46	93.38	96.05	1.02	1.01	0.96
0.2, 0	0.2	1.01	94.2	99.25	92.87	96.99	0.97	1	0.98
0.3, 0	0.3	1.02	98.15	98.95	95.34	97.81	0.95	0.99	0.99
0.4, 0	0.4	1.04	98.92	98.6	95.34	97.86	0.94	0.99	0.99
0.5, 0	0.5	1.07	99.48	98.33	94.29	97.66	0.93	0.99	0.99
0.6, 0	0.6	1.1	99.55	97.92	94.42	96.48	0.92	1	0.99
0.7, 0	0.7	1.14	99.65	97.48	95.02	94.79	0.92	1	1
0.8, 0	0.8	1.19	99.9	96.96	94.37	94.64	0.92	1	1
0.9, 0	0.9	1.25	99.95	95.35	90.25	93.97	0.91	1	0.99
0.99, 0	0.99	1.32	99.98	84.59	79.99	93.99	0.87	1.06	0.99

A node with high *node compression ratio* is somehow dysfunctional. It plays a less significant role in the next layer(s) for the prediction of desired output. It is a good candidate to prune its output connections to the next layer nodes. This point was also reported in [23], [24], [32], [47], [54], [55].

Table 6 demonstrates that the existence of the *implicit regularization* and its impact on performance for the fully-connected MLP. It indicates that *LCRs* of deeper layers are higher than *LCRs* of earlier layers. It also shows that the *built-in sparsity* caused by the *implicit regularization* is not effective enough to improve the network performance for the correlation pairs $\{\rho_0, \rho_1\}$ of smaller $\Delta\rho$.

TABLE 8. The *Accuracy* Measurements of the *Fully-* and *Sparsely-Connected* MLPs with *Layer Dimension (Width) N=1,024* for the *Double*, and *Triple Hidden Layer* Cases and the $\Delta\rho$ *Data Sets* of Various Correlation Coefficient Pairs

$\Delta\rho$ Data Sets			Double					
$\{\rho_0, \rho_1\}$	$\Delta\rho$	SNR	Dense	Corr	Mag	Rand	$SR_1(\%)$	$SR_2(\%)$
0, 0	0	1	49.9	49.52	**50.58**	50.38	99.63	93.89
0.1, 0	0.1	1	50	**69.1**	50.42	51.75	99.46	94.43
0.2, 0	0.2	1.01	51.7	**93.5**	52.9	58.4	99.25	94.81
0.3, 0	0.3	1.02	51.12	**97.82**	55.42	65.47	98.95	93.71
0.4, 0	0.4	1.04	53.8	**98.9**	59.85	73.9	98.6	94.14
0.5, 0	0.5	1.07	55.78	**99.35**	64.53	80.97	98.33	91.88
0.6, 0	0.6	1.1	60.68	**99.58**	73.05	89.52	97.92	94.54
0.7, 0	0.7	1.14	69.82	**99.95**	80.6	94.08	97.48	93.85
0.8, 0	0.8	1.19	83.55	**99.98**	88.8	97.6	96.96	94.4
0.9, 0	0.9	1.25	95.88	**99.82**	94.15	99.3	95.35	93.19
0.99, 0	0.99	1.32	99.7	**100**	99.32	99.7	84.59	79.98

$\Delta\rho$ Data Sets			Triple						
$\{\rho_0, \rho_1\}$	$\Delta\rho$	SNR	Dense	Corr	Mag	Rand	$SR_1(\%)$	$SR_2(\%)$	$SR_3(\%)$
0, 0	0	1	50.9	50.02	49.98	**51.22**	99.63	94.38	97.97
0.1, 0	0.1	1	50.4	**74.3**	51.22	50.92	99.46	93.38	96.05
0.2, 0	0.2	1.01	49.92	**94.2**	53.28	57	99.25	92.87	96.99
0.3, 0	0.3	1.02	51.32	**98.15**	56.58	64.38	98.95	95.34	97.81
0.4, 0	0.4	1.04	52.48	**98.92**	59.82	71.6	98.6	95.34	97.86
0.5, 0	0.5	1.07	53	**99.48**	67.53	80	98.33	94.29	97.66
0.6, 0	0.6	1.1	54.5	**99.55**	73.12	87.98	97.92	94.42	96.48
0.7, 0	0.7	1.14	56.78	**99.65**	81.18	91.98	97.48	95.02	94.79
0.8, 0	0.8	1.19	62.05	**99.9**	87.1	96.95	96.96	94.37	94.64
0.9, 0	0.9	1.25	74.55	**99.95**	92.98	99.08	95.35	90.25	93.97
0.99, 0	0.99	1.32	97.5	**99.98**	99.02	99.8	84.59	79.99	93.99

Remark 6:
The optimization modulates the degree of parameterization for MLP to prevent under-fitting or malign overfitting via Sigmoid activation function.

The simulation results and points made above provide convincing evidence to sparsify dense MLPs in order to (potentially) improve performance and computational efficiency. Tables 2, 3 and 4 show the fact that the under-determined system optimization incorporating signal dependent pruning exploits the potential benefits of model size reduction.

Remark 7:
The use of sparsity as an explicit regularizer in model optimization brings significantly higher performance gains and efficiency than the *implicit regularization* performed by the optimizer itself (see Table 4 and compare Tables 6 and 7). It was reported that signal-dependent sparsity methods may bring performance improvements over signal-independent methods [23], [38], [39], [40]. The superior performance of the correlation-based pruning also validates this point. Its superiority is consistent for the single, double, and triple hidden layers MLPs tested in this study (see Tables 3, 4 and 5 for the single hidden layer case and Table 8 for the double and triple hidden layer cases).

VI. CONCLUSIONS

Mathematical nature of the under-determined system optimization requires parameter reduction methods (regularizers) to improve search performance. We propose a signal dependent, correlation-based pruning method to sparsify networks. It is intuitive and simulations show that performance improves when network sparsity is tied to input signal and node statistics. The proposed correlation-based pruning outperforms magnitude and random pruning methods. It is shown that model design benefits from the use of such sparsity method during training.

Two information-theoretic metrics called node compression *(NCR)* and layer compression ratios *(LCR)* are introduced. They explain the inner workings of model optimization at the node level and provide hints for network performance. *NCR* measures the entropy compression of a node dictated by the alignment between its input *pdf* and activation function. Its value increases when the mean and/or variance misalignment become more significant. It is argued that such misalignment is exploited as an implicit regularizer during optimization of an under-determined system. This phenomenon causes *built-in sparsity*.

The signal dependent pruning may also be used for Neural Architecture Search (NAS) and automated Machine Learning (AutoML) where a self re-configuring and adaptive network architecture with node and weight sparsities is realized.

APPENDIX

The pseudo-code of the correlation-based node partitioning introduced in Section IV-B is given in Algorithm 1.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[3] P. L. Bartlett, A. Montanari, and A. Rakhlin, "Deep learning: A statistical viewpoint," *Acta Numerica*, vol. 30, pp. 87–201, 2021.

[4] B. Neyshabur, R. Tomioka, and N. Srebro, "In search of the real inductive bias: On the role of implicit regularization in deep learning," 2014, *arXiv:1412.6614*.

[5] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro, "Towards understanding the role of over-parametrization in generalization of neural networks," 2018, *arXiv:1805.12076*.

[6] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, no. 12, p. 124003, 2021.

[7] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias-variance trade-off," in *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, 2019, pp. 15 849–15 854.

[8] S. Spigler, M. Geiger, S. d'Ascoli, L. Sagun, G. Biroli, and M. Wyart, "A jamming transition from under- to over-parametrization affects generalization in deep learning," *Journal of Physics A: Mathematical and Theoretical*, vol. 52, no. 47, p. 474001, 2019.

[9] M. Geiger *et al.*, "Jamming transition as a paradigm to understand the loss landscape of deep neural networks," *Physical Review E*, vol. 100, no. 1, p. 012115, 2019.

[10] M. S. Advani, A. M. Saxe, and H. Sompolinsky, "High-dimensional dynamics of generalization error in neural networks," *Neural Networks*, vol. 132, pp. 428–446, 2020.

[11] M. Geiger *et al.*, "Scaling description of generalization with number of parameters in deep learning," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2020, no. 2, p. 023401, 2020.

[12] G. Valle-Perez, C. Q. Camargo, and A. A. Louis, "Deep learning generalizes because the parameter-function map is biased towards simple functions," 2018, *arXiv:1805.08522*.

[13] G. Yang and H. Salman, "A fine-grained spectral perspective on neural networks," 2019, *arXiv:1907.10599*.

[14] D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro, "The implicit bias of gradient descent on separable data," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 2822–2878, 2018.

[15] Y. Zhang, A. M. Saxe, M. S. Advani, and A. A. Lee, "Energy-entropy competition and the effectiveness of stochastic gradient descent in machine learning," *Molecular Physics*, vol. 116, no. 21-22, pp. 3214–3223, 2018.

[16] C. H. Martin, T. Peng, and M. W. Mahoney, "Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data," *Nature Communications*, vol. 12, no. 1, p. 4122, 2021.

[17] C. H. Martin and M. W. Mahoney, "Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 7479–7551, 2021.

[18] ——, "Heavy-tailed universality predicts trends in test accuracies for very large pre-trained deep neural networks," in *Proceedings of the SIAM International Conference on Data Mining*, 2020, pp. 505–513.

[19] M. W. Mahoney and C. H. Martin, "Traditional and heavy tailed self regularization in neural network models," in *Proceedings of International Conference on Machine Learning*, 2019, pp. 4284–4293.

[20] N. O. Hodas and P. Stinis, "Doing the impossible: Why neural networks can be trained at all," *Frontiers in Psychology*, vol. 9, p. 1185, 2018.

[21] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," 2017, *arXiv:1703.00810*.

[22] A. M. Saxe *et al.*, "On the information bottleneck theory of deep learning," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124020, 2019.

[23] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 10 882–11 005, 2021.

[24] C. Benar and A. Akansu, "On explainability of a simple classifier for AR(1) source," in *Proceedings of the IEEE 56th Annual Conference on Information Sciences and Systems (CISS)*, 2022, pp. 275–280.

[25] C. Molnar, *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. Durham, NC, USA: Lulu Press, 2020.

[26] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," in *Proceedings of the IEEE*, vol. 105, no. 12, 2017, pp. 2295–2329.

[27] E. Genç *et al.*, "Diffusion markers of dendritic density and arborization in gray matter predict differences in intelligence," *Nature Communications*, vol. 9, no. 1, p. 1905, 2018.

[28] S. Herculano-Houzel, B. Mota, P. Wong, and J. H. Kaas, "Connectivity-driven white matter scaling and folding in primate cerebral cortex," in *Proceedings of the National Academy of Sciences*, vol. 107, no. 44, 2010, pp. 19 008–19 013.

[29] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," 2019, *arXiv:1902.09574*.

[30] S. Changpinyo, M. Sandler, and A. Zhmoginov, "The power of sparsity in convolutional neural networks," 2017, *arXiv:1702.06257*.

[31] D. Blalock, J. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" in *Proceedings of Machine Learning and Systems*, vol. 2, 2020, pp. 129–146.

[32] M. Hagiwara, "Removal of hidden units and weights for back propagation networks," in *Proceedings of 1993 International Joint Conference on Neural Networks*, vol. 1, 1993, pp. 351–354.

[33] G. Thimm and E. Fiesler, "Evaluating pruning methods," in *Proceedings of the International Symposium on Artificial Neural Networks*, 1995, pp. 20–25.

[34] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[35] A. Kusupati *et al.*, "Soft threshold weight reparameterization for learnable sparsity," in *Proceedings of the International Conference on Machine Learning*, 2020, pp. 5544–5555.

[36] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," 2018, *arXiv:1802.00124*.
[37] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Proceedings of the International Conference on Neural Information Processing Systems*, vol. 1, 1988.
[38] V. Sanh, T. Wolf, and A. Rush, "Movement pruning: Adaptive sparsity by fine-tuning," in *Proceedings of the International Conference on Neural Information Processing Systems*, vol. 33, 2020, pp. 20 378–20 389.
[39] E. Tartaglione, S. Lepsøy, A. Fiandrotti, and G. Francini, "Learning sparse neural networks via sensitivity-driven regularization," in *Proceedings of the International Conference on Neural Information Processing Systems*, vol. 31, 2018, pp. 3878–3888.
[40] S. Wang, Z. Chen, S. Du, and Z. Lin, "Learning deep sparse regularizers with applications to multi-view clustering and semi-supervised classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5042–5055, 2021.
[41] Y. Sun, X. Wang, and X. Tang, "Sparsifying neural network connections for face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4856–4864.
[42] Z. Atashgahi, J. Pieterse, S. Liu, D. C. Mocanu, R. Veldhuis, and M. Pechenizkiy, "A brain-inspired algorithm for training highly sparse neural networks," *Machine Learning*, vol. 111, no. 12, pp. 4411–4452, 2022.
[43] D. O. Hebb, *The Organization of Behavior*. New York, NY, USA: Wiley, 1949.
[44] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Proceedings of Artificial Intelligence and Statistics*, 2015, pp. 192–204.
[45] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.
[46] L. Franco, "Generalization ability of boolean functions implemented in feedforward neural networks," *Neurocomputing*, vol. 70, no. 1-3, pp. 351–361, 2006.
[47] M. Huh, H. Mobahi, R. Zhang, B. Cheung, P. Agrawal, and P. Isola, "The low-rank simplicity bias in deep networks," 2021, *arXiv:2103.10427*.
[48] A. N. Akansu and R. A. Haddad, *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets*. New York, NY, USA: Academic Press, 1992.
[49] A. Papoulis, *Probability, Random Variables and Stochastic Processes*. New York, NY, USA: McGraw Hill, 1984.
[50] T. Berger, *Rate-Distortion Theory*. New York, NY, USA: Wiley, 2003.
[51] A. Akansu, M. Avellaneda, and A. Xiong, "Quant investing in cluster portfolios," *Journal of Investment Strategies*, vol. 9, no. 4, pp. 61–78, 2020.
[52] K. Wen, J. Teng, and J. Zhang, "Benign overfitting in classification: Provably counter label noise with larger models," in *Proceedings of the 11th International Conference on Learning Representations*, 2023.
[53] J. Franklin, "The elements of statistical learning: Data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.
[54] J. Sietsman, "Neural net pruning-why and how," in *Proceedings of the IEEE International Conference on Neural Networks*, 1988, pp. 1325–333.
[55] M. Hagiwara, "A simple and effective method for removal of hidden units and weights," *Neurocomputing*, vol. 6, no. 2, pp. 207–218, 1994.

CEM BENAR (Graduate Student Member, IEEE) received the B.S. degree in electrical and electronics engineering with a double major in computer science from Özyeğin University, Istanbul, Turkey, in 2016, and the M.S. degree in neuroscience from Bilkent University, Ankara, Turkey, in 2019. He is currently working toward the Ph.D. degree with the New Jersey Institute of Technology, Newark, NJ, USA. His research interests include signal processing, machine learning, and deep learning, with emphasis on understanding the inner-workings of neural networks.

ALI N. AKANSU (Fellow, IEEE) received his B.S. degree from the Technical University of Istanbul, Turkey, M.S. and Ph.D. degrees from the Polytechnic University, Brooklyn, New York, all in electrical engineering. He has been with the New Jersey Institute of Technology since 1987 where he is Professor of Electrical & Computer Engineering. He was a Founding Director of the New Jersey Center for Multimedia Research and NSF Industry-University Cooperative Research Center for Digital Video & New Media. Dr. Akansu was the Vice President for R&D of IDT Corporation [NYSE:IDT]. He was the founding President and CEO of PixWave, Inc., and Senior VP for Technology Development of TV.TV (IDT Entertainment subsidiaries). He did sit on the boards of start-up companies, and an investment fund. He visited David Sarnoff Research Center, IBM T.J. Watson Research Center, GEC-Marconi Electronic Systems Corp., and Courant Institute of Mathematical Sciences at NYU. His current research interests are signals and transforms, quantitative finance and algorithmic trading, explainable machine learning methods and data engineering, including FPGA & GPU computing.

Algorithm 1 Correlation-Based Partitioning of Layer Node Outputs in a Multilayer Perceptron (MLP).

▷ Y_k: k^{th} *layer node outputs of size* $N \times M$
N: k^{th} *layer dimension (number of nodes)*
M: *the number of training samples*
R_k: *Pearson correlation matrix of node outputs* Y_k *of the* k^{th} *layer (the* 1^{st} *layer refers to input layer)*
$thre$: *correlation threshold of the* k^{th} *layer*
P: *a list of node partitions (*P_m *is* m^{th} *partition)*
C: *a set of node indices belonging to a partition (partitioned node indices)*
U: *a set of node indices not belonging to any partition (unpartitioned node indices)*
I: *a list of node pair indices* (a, b) *whose correlation coefficients* $R[a, b]$ *are greater than correlation* $thre$ ◁

function PARTITIONNODES($R_k, thre, Y_k$)
  ▷ *partitioning nodes with positive correlations* ◁
  $P_{pos}, U_{pos} \leftarrow$ PARTITIONCOMBINE($R_k, thre, Y_k$)
  ▷ *partitioning nodes with negative correlations* ◁
  $P_{neg}, U_{neg} \leftarrow$ PARTITIONCOMBINE($-R_k[U_{pos}, U_{pos}]$, $thre, Y_k[:, U_{pos}]$)
  $P \xleftarrow{+} P_{pos}$ ▷ *append partitions of nodes with positive correlations*
  $P \xleftarrow{+} P_{neg}$ ▷ *append partitions of nodes with negative correlations*
  $P \xleftarrow{+} U_{neg}$ ▷ *append each unpartitioned node in* U_{neg} *as a separate (single node) partition*
  return P

function PARTITIONCOMBINE($R_k, thre, Y_k$)
  $R'_k \leftarrow R_k$
  $Y'_k \leftarrow Y_k$
  while LEN(R'_k) > 1 **do**
    $P, U \leftarrow$ PARTITIONSELECT($R'_k, thre$)
    $L \leftarrow$ LEN(P) ▷ *number of partitions*
    for $l = 1, \ldots, L$ **do** ▷ l: *partition id*
      ▷ *add node outputs within the same partition* l ◁
      $Y'_k[:, l] \leftarrow$ SUM($Y'_k[:, P(l)]$)
    ▷ *calculate Pearson correlation of* Y'_k ◁
    $R'_k \leftarrow$ CORRCOEF(Y'_k)
  return P, U

function PARTITIONSELECT($R_k, thre$)
  $I \leftarrow ()$ ▷ *initialized as an empty list*
  for $a = 1, \ldots, N$ **do** ▷ *number of rows of* R_k
    for $b = 1, \ldots, N$ **do** ▷ *number of columns of* R_k
      if $a > b$ **then**
        ▷ *only check node pair indices in non-diagonal, lower triangular part of* R_k ◁
        if $R_k(a, b) \geq thre$ **then**
          $I \xleftarrow{+} (a, b)$
  ▷ *sort node pair indices in* I *in the descending order based on* R_k ◁
  SORT(I, R_k)
  $P \leftarrow ()$ ▷ *initialize as an empty list*
  $C \leftarrow \{\}$ ▷ *initialize as an empty set*
  ▷ *number of thresholded node pair indices* I ◁
  $L \leftarrow$ LEN(I)
  for $l = 1, \ldots, L$ **do**
    $(a, b) \leftarrow I(l)$ ▷ *indices of* l^{th} *node pair in* I
    if $a \notin C$ & $b \notin C$ **then**
      ▷ *none of them belong to any existing partition* ◁
      $P \xleftarrow{+} \{a, b\}$ ▷ *add it as a separate partition*
      $C \xleftarrow{+} a$ ▷ *node* a *is partitioned*
      $C \xleftarrow{+} b$ ▷ *node* b *is partitioned*
    else if $a \in C$ & $b \notin C$ **then**
      ▷ *node* a *belongs to a partition (e.g.,* P_m*) check if node* b *is eligible to be in the same partition* ◁
      if MEAN($R_k[b, P_m]$) $\geq thre$ **then**
        $P_m \xleftarrow{+} b$ ▷ *add node* b *into partition* P_m
        $C \xleftarrow{+} b$ ▷ *node* b *is partitioned*
    else if $a \notin C$ & $b \in C$ **then**
      ▷ *node* b *belongs to a partition (e.g.,* P_m*) check if node* a *is eligible to be in the same partition* ◁
      if MEAN($R_k[a, P_m]$) $\geq thre$ **then**
        $P_m \xleftarrow{+} a$ ▷ *add node* a *into partition* P_m
        $C \xleftarrow{+} a$ ▷ *node* a *is partitioned*
  $U \leftarrow \{1, 2, \ldots, N\} \setminus C$ ▷ *unpartitioned node indices*
  return P, U