Electronic Transactions on Numerical Analysis.
Volume 56, pp. 157–186, 2022.

ISSN 1068–9613.
DOI: 10.1553/etna_vol56s157

ETNA
Kent State University and
Johann Radon Institute (RICAM)

DECOMPOSITION AND COMPOSITION OF DEEP CONVOLUTIONAL NEURAL NETWORKS AND TRAINING ACCELERATION VIA SUB-NETWORK TRANSFER LEARNING*

LINYAN GU[†‡], WEI ZHANG[†‡], JIA LIU[†‡], AND XIAO-CHUAN CAI[§]

Abstract. Deep convolutional neural network (DCNN) has led to significant breakthroughs in deep learning. However, larger models and larger datasets result in longer training times slowing down the development progress of deep learning. In this paper, following the idea of domain decomposition methods, we propose and study a new method to parallelize the training of DCNNs by decomposing and composing DCNNs. First, a global network is decomposed into several sub-networks by partitioning the width of the network (i.e., along the channel dimension) while keeping the depth constant. All the sub-networks are individually trained, in parallel without any interprocessor communication, with the corresponding decomposed samples from the input data. Then, following the idea of nonlinear preconditioning, we propose a sub-network transfer learning strategy in which the weights of the trained sub-networks are recomposed to initialize the global network, which is then trained to further adapt the parameters. Some theoretical analyses are provided to show the effectiveness of the sub-network transfer learning strategy. More precisely speaking, we prove that (1) the initialized global network can extract the feature maps learned by the sub-networks; (2) the initialization of the global network can provide an upper bound and a lower bound for the cost function and the classification accuracy with the corresponding values of the trained sub-networks. Some experiments are provided to evaluate the proposed methods. The results show that the sub-network transfer learning strategy can indeed provide good initialization and accelerate the training of the global network. Additionally, after further training, the transfer learning strategy shows almost no loss of accuracy and sometimes the accuracy is higher than if the network is initialized randomly.

Key words. deep convolutional neural networks, decomposition and composition, parallel training, transfer learning, domain decomposition

AMS subject classifications. 68W10, 68W40

1. Introduction.

Deep convolutional neural network (DCNN) [17] is one of the most successful models in deep learning [16] and it has brought significant improvements to the field of computer vision for a wide range of problems, including image classification [7, 14], object detection [21, 27], semantic image segmentation [3, 23, 28], and many other applications. DCNN is capable of integrating multilevel of features from the dataset [7], and by increasing the depth of the network more accuracy can be obtained [29, 31]. Another approach to making networks more expressive is to increase the network width, such as GoogLeNet in [31, 32]. Larger (deeper and wider) models and larger datasets have led to breakthroughs in accuracy. However, it results in much longer training time and memory intensity negatively impacting the development of DCNNs [6, 36] in many important applications, such as computational finance, autonomous driving, and medical imaging [36]. Therefore, reducing the training time of DCNN is critical. Below we provide an overview of existing methods for accelerating the network training.

Distributed training. Generally speaking, there are two ways to parallelize training: model parallelism and data parallelism [6, 13]. On a multi-processor computer, model parallelism partitions the network into pieces, and different processors train different pieces. In

*Received December 07, 2020. Accepted April 07, 2021. Published online on March 2, 2022. Recommended by Axel Klawonn. This research was partially supported by the Cooperation on Scientific and Technological Innovation in Hong Kong, Macao and Taiwan as Part of National Key R&D Programs: 2021YFE0204300, the National Natural Science Foundation of China under Grant No. 12101589, and the Guangdong Natural Science Foundation under Grant No. 2020A1515110951. Corresponding author: Xiao-Chuan Cai.

[†]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, 518055, China.
[‡]Shenzhen Key Laboratory for Exascale Engineering and Scientific Computing, Shenzhen, 518000, China.
[§]Faculty of Science and Technology, University of Macau, Avenida da Universidade, Taipa, Macao, China (xccai@um.edu.mo).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

model parallelism, frequent communication between different processors is needed since the calculation of the next layer usually requires all the outputs of the previous layer; that is, different processors need to synchronize, which is costly. In data parallelism, the dataset is partitioned into parts stored in each processor, and each processor has a local copy of the network with its parameters. Distributed synchronous stochastic gradient descent (SGD) is a widely used method in data parallelism. It divides the dataset into minibatches and then maps them onto a pool of parallel processors [6, 36]. However, in synchronous SGD, scaling the training to a large number of processors means an increase in the batch size used in each iteration, which causes convergence problems in solving the optimization problem and results in poor generalization [6, 9, 36]. To avoid the problem, new training algorithms are developed to increase the batch size in synchronous SGD without loss of accuracy. For example, with ResNet-50 [7] and the dataset ImageNet-1K [5], the batch size and number of processors have been successfully increased from batch size of 1K on 128 processors [9] to batch size of 32K on 1024 processors [36]. In addition, in synchronized data parallelism, the communication is needed in every iteration in order to obtain the sum of local gradients and to broadcast the global parameters [36].

Network pruning. Network pruning is a technique of model compression for saving computation and storage [4]. The method explores the redundancy in the model parameters and attempts to remove the redundant and uncritical connections. The criterion of pruning is to evaluate the importance or contribution of weights; thus, the less important weights can be pruned. For example, in [18], neurons with small magnitude of weights are pruned. Molchanov et al. [25, 26] used Taylor expansions to approximate a filter's contribution. Yu et al. [37] utilized a neuron importance score propagation algorithm to propagate the importance scores to every weight. Liu et al. [22] imposed a scaling factor and facilitated one channel-level pruning during the training process. In addition, there is also growing interest in training compact DCNNs with sparsity constraints [8, 22, 35].

In the present paper, we propose a new method to accelerate the training of DCNNs by decomposing and composing DCNNs motivated by the idea of domain decomposition methods for solving partial differential equations on large scale computers [30].

Domain decomposition. Domain decomposition methods are a family of highly parallel methods for solving partial differential equations [30, 34]. They are based on the divide and conquer philosophy for solving a problem defined on a global domain by iteratively solving subproblems defined on smaller subdomains [34]. The advantage of domain decomposition methods is the straightforward applicability for parallel computing [2, 19]. Additional advantages also include localized treatment for the specificity of subdomain problems [34].

Following the domain decomposition methods, firstly, a DCNN (also called a global network in this paper) is decomposed into K sub-networks by partitioning the width of the network (i.e., along the channel dimension) while keeping the depth constant. Correspondingly, each sample is decomposed into K subsamples, which are then used to train the corresponding sub-networks. Note that the trainings of sub-networks are completely independent of each other and can be performed on parallel computers without any interprocessor communication. The width-based partitioning of DCNNs is based on the intuitive assumption that there is information redundancy when using global networks to deal with subsamples. In addition, since the sub-networks focus on the subdomains of the samples, it may help to boost the models' ability to capture fine details, which is of crucial importance for dense prediction tasks such as semantic segmentation. Secondly, after the sub-networks are trained, we propose a sub-network transfer learning strategy in which the weights of the trained sub-networks are composed to initialize the global network. The composition is almost the reverse process of the decomposition but with the weights connected between different sub-networks set as zero

ETNA
Kent State University and
Johann Radon Institute (RICAM)

tensors. The global network with such initialization is then further trained to learn long-range contextual information that cannot be learned by the sub-networks.

Compared with existing distributed training methods for DCNNs, the new approach is different in the following ways: 1) across the data dimension, we decompose each sample into K subsamples, and the subsamples on one subdomain are used to train one sub-network, which is different than partitioning the dataset into parts in the current data parallel approaches; 2) across the model dimension, we decompose a global network into sub-networks that can be trained completely independently, which is different from the current model parallelism in which the computation in one processor depends on that in the other processors; 3) the communication between different sub-networks only occurs in the initialization of the global network; that is, the weights of the trained sub-networks are composed to initialize the global network.

Similar to network pruning, the decomposition is based on the intuition that there is information redundancy in the process of computing the global network. The difference is that we decompose the global network into sub-networks for the purpose of parallelization of the training, and then the sub-networks are composed to initialize the global network that will be further trained.

Some theoretical analyses are developed to show the effectiveness of the initialization provided by the sub-network transfer learning strategy. First, we show that the initialized global network can extract the feature maps learned by the trained sub-networks. Second, it is proven that the initialization of the global network offers an upper bound and a lower bound for the cost function and the classification accuracy. The bounds depend on those of the trained sub-networks. Finally, the proposed approach is evaluated by some experiments involving image classification tasks. The experiments show that the sub-network transfer learning strategy can provide a good initialization and accelerate the training of the global DCNNs. Additionally, after further training, the global network shows almost no loss of accuracy; sometimes the global network initialized by the sub-network transfer learning strategy offers higher accuracy than those initialized randomly.

This paper is organized as follows. In Section 2, we first present a method to decompose a global network into sub-networks, along with some notations and preliminaries (Section 2.1). Secondly, a sub-network transfer learning strategy in which the weights of the trained sub-networks are composed to initialize the global network is proposed (Section 2.2). Third, we present a theory to show the effectiveness of the sub-network transfer learning strategy (Section 2.3). Section 3 provides some experiments to evaluate the proposed approach. Finally, Section 4 gives a concluding summary of this work.

2. Proposed approaches. In this section, we first introduce a way to decompose a DCNN into sub-DCNNs and then a way to compose several subnetworks into a global DCNN. Furthermore, some theoretical results about the effectiveness of the sub-network transfer learning strategy are derived by analyzing the intermediate features, cost function, and classification accuracy of the global network. For the simplicity of presentation, some assumptions about the architecture of the networks are made in this section. For example, we assume 2D DCNNs are used for classification. Note that the proposed approaches are applicable to general DCNNs without requiring these assumptions.

2.1. Decomposition of DCNNs.

2.1.1. Notations and preliminaries. In this section, we assume that a DCNN for classification is a network consisting of some convolutional layers and some fully connected (FC) layers, followed by a classification module. A convolutional layer consists of a convolutional operation, a nonlinearity operation, and a feature pooling operation. The form of an FC

ETNA
Kent State University and
Johann Radon Institute (RICAM)

layer can be regarded as a special case of a convolutional layer with the receptive field as the whole input of this layer. The classification module is usually a softmax layer. Some formal descriptions of the operations are given below.

Conventions. The input and output of each layer are 3D tensors called *feature maps*, where the third dimension of the tensors is the number of independent feature maps, and the first and the second are the height and the width, respectively. Assume a classification task with samples $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ and their corresponding ground truth $Y = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N\}$, where $\boldsymbol{y}_i \in \{0,1\}^C$ and the integer C denotes the number of classes and $\boldsymbol{y}_i(c) = 1$ indicates that the ground truth of $\boldsymbol{x}_i$ is in the c-th class.

Multi-channel convolution. Assume that the input $\boldsymbol{V}$ is a 3D tensor with element $V_{i,j,l}$ within channel l at row i and column j. Let the convolutional kernel be a 4D kernel tensor $\boldsymbol{K}$ with element $K_{m,n,l,k}$ giving the connection strength between a unit in channel k of the output and a unit in channel l of the input, with an offset of m rows and n columns between the output unit and the input unit. Then, the output $\boldsymbol{Z}$ is produced by convolving $\boldsymbol{K}$ across $\boldsymbol{V}$, that is, $\boldsymbol{Z} = \boldsymbol{V} * \boldsymbol{K}$ and

$$Z_{i,j,k} = \sum_{m,n,l} V_{i+m-1,j+n-1,l} K_{m,n,l,k}. \tag{2.1}$$

Note that it is assumed in this section that there are no skip connections in the DCNN and each layer contains only one multichannel convolution if not specified otherwise.

Stacking multi-layers. Denote a L-layer DCNN as $F(\boldsymbol{x}; \Theta)$, where $\boldsymbol{x}$ is the input of the network and $\Theta = \{\theta_1, \ldots, \theta_L\}$ is the set of parameters. In addition, denote $\Theta^l = \{\theta_1, \ldots, \theta_l\}$ as the set of parameters of the first l layers, and denote the output of the l-th layer by

$$\boldsymbol{x}^l = f^l(\boldsymbol{x}^{l-1}; \theta^l) = F^l(\boldsymbol{x}^0; \Theta^l) \tag{2.2}$$

with $\boldsymbol{x}^0 = \boldsymbol{x}$, where f^l denotes a convolutional layer or an FC layer. By regarding the FC layer as a special case of a convolutional layer, the kernel of the l-th layer is denoted by $\boldsymbol{w}^l \in \mathbb{R}^{t_1^l \times t_2^l \times c_{\text{in}}^l \times c_{\text{out}}^l}$ (as defined in (2.1)), where c_{in}^l and c_{out}^l are the number of channels of the input and output of this layer, respectively, and t_1^l, t_2^l are the kernel widths. Generally, for the FC layers, as the receptive field is the whole input, the kernel width of the first FC layer is adapted to the size of the input of this layer. For the other FC layers, the kernel width is 1. The output of the network is denoted by $\boldsymbol{x}^L = F(\boldsymbol{x}; \Theta) \in \mathbb{R}^C$, where C is the number of classes, and we denote $\boldsymbol{x}^L = (\boldsymbol{x}^L(1), \ldots, \boldsymbol{x}^L(C))^\top$.

Classification module. A classification module is a softmax function that normalizes the input (i.e., $\boldsymbol{x}^L$) into a probability distribution; that is, the output of the classification module is given by

$$\hat{\boldsymbol{y}}(c) = \frac{e^{\boldsymbol{x}^L(c)}}{\|e^{\boldsymbol{x}^L}\|_1}, \quad c = 1, \ldots, C, \tag{2.3}$$

where $\hat{\boldsymbol{y}}(c)$ is the posterior probability of the c-th class. The predicted class $\hat{c}_{\boldsymbol{x}}$ is $\hat{c}_{\boldsymbol{x}} = \arg\max_c \hat{\boldsymbol{y}}(c) = \arg\max_c \boldsymbol{x}^L(c)$. Then, for sample $\boldsymbol{x}$ and its ground truth $\boldsymbol{y}$, its cost function is defined as the cross entropy loss:

$$J(\boldsymbol{x}; \boldsymbol{y}, \Theta) = -\sum_{c=1}^{C} \boldsymbol{y}(c) \log \hat{\boldsymbol{y}}(c). \tag{2.4}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

With observation samples $\{X, Y\}$, the network is trained to solve the optimization problem:

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{N} J(\boldsymbol{x}_i; \boldsymbol{y}_i, \Theta). \tag{2.5}$$

Notations of tensors. Assume $\boldsymbol{x} \in \mathbb{R}^{H\times W\times D}$ is a 3D tensor with element $x_{i,j,k}$ where $(i,j,k) \in \Omega$, Ω is a Cartesian product and $\Omega = [H] \times [W] \times [D]$ with $[H] = \{1, \dots, H\}$. Given $a, b \in \mathbb{Z}^+$ with $a \le b$, denote by

$$[a : b] := \{a, a+1, \dots, b\}.$$

Given a Cartesian product $\tilde{\Omega} \subset \Omega$, that means, there exist $\tilde{\mathcal{H}}, \tilde{\mathcal{W}}, \tilde{\mathcal{D}} \subset [H], [W], [D]$ such that $\tilde{\Omega} = \tilde{\mathcal{H}} \times \tilde{\mathcal{W}} \times \tilde{\mathcal{D}}$, we define some bijective mappings:

$$\tau_h : \tilde{\mathcal{H}} \to [|\tilde{\mathcal{H}}|], \tau_w : \tilde{\mathcal{W}} \to [|\tilde{\mathcal{W}}|], \text{ and } \tau_d : \tilde{\mathcal{D}} \to [|\tilde{\mathcal{D}}|],$$

where $|\cdot|$ denotes the cardinality of a set. In addition, the mappings, τ_h, τ_w, and τ_d, are order-preserving mappings; that is,

$$\tau_h(i_1) < \tau_h(i_2), \tau_w(j_1) < \tau_w(j_2), \tau_d(k_1) < \tau_d(k_2), \text{ for any } i_1 < i_2, j_1 < j_2, k_1 < k_2.$$

Then we denote a subdomain of $\boldsymbol{x}$ with respect to $\tilde{\Omega}$ by

$$\tilde{\boldsymbol{x}} = \mathcal{D}_{\tilde{\Omega}}(\boldsymbol{x}), \text{with element } \tilde{x}_{\tilde{i},\tilde{j},\tilde{k}} = x_{i,j,k}$$

where

$$(i,j,k) \in \tilde{\Omega},\ \tilde{i} = \tau_h(i),\ \tilde{j} = \tau_w(i),\ \text{and } \tilde{k} = \tau_d(k).$$

In this paper, the Cartesian product used to decompose a tensor is assumed to satisfy that

$$\begin{gathered} h_{i+1} - h_i = 1, w_{j+1} - w_j = 1, \text{ and } d_{k+1} - d_k = 1, \text{ with} \\ \tilde{\mathcal{H}} = \{h_1, \dots, h_n\}, \tilde{\mathcal{W}} = \{w_1, \dots, w_n\}, \text{ and } \tilde{\mathcal{D}} = \{d_1, \dots, d_n\}. \end{gathered}$$

With this assumption, the decomposition is "neighbor preserving"; that is, for any $i_1, i_2 \in \tilde{\mathcal{H}}$, $j_1, j_2 \in \tilde{\mathcal{W}}$, and $k_1, k_2 \in \tilde{\mathcal{D}}$, we have

$$\begin{cases} \tau_h(i_1) - \tau_h(i_2) = 1 & \Rightarrow i_1 - i_2 = 1, \\ \tau_w(j_1) - \tau_w(j_2) = 1 & \Rightarrow j_1 - j_2 = 1, \\ \tau_d(k_1) - \tau_d(k_2) = 1 & \Rightarrow k_1 - k_2 = 1. \end{cases}$$

Given a set of Cartesian products $\{\Omega_k\}_{k=1}^K$ where

$$\Omega_i \cap \Omega_j = \emptyset, \bigcup_{k=1}^{K} \Omega_k = \Omega, \tag{2.6}$$

we call $\{\tilde{\boldsymbol{x}}_k = \mathcal{D}_{\Omega_k}(\boldsymbol{x}) \mid k = 1, \dots, K\}$ a complete decomposition of $\boldsymbol{x}$.

Activation fields of subdomains. Given a DCNN F and a subdomain $\tilde{\Omega}$ of the input, the activation field of $\tilde{\Omega}$ in the l-th layer, which is denoted by $\mathcal{G}_{F,l}(\tilde{\Omega})$, represents the largest subdomain of the output of this layer that only responds to $\tilde{\Omega}$. More formally, let

$$\Lambda = \{\bar{\Omega} \mid \mathcal{D}_{\bar{\Omega}}(F^l(\boldsymbol{x}_1; \Theta^l)) = \mathcal{D}_{\bar{\Omega}}(F^l(\boldsymbol{x}_2; \Theta^l)), \forall\, \Theta, \boldsymbol{x}_1, \boldsymbol{x}_2 \text{ with } \mathcal{D}_{\tilde{\Omega}}(\boldsymbol{x}_1) = \mathcal{D}_{\tilde{\Omega}}(\boldsymbol{x}_2)\},$$

then it holds that $\mathcal{G}_{F,l}(\tilde{\Omega}) \in \Lambda$ and $\bar{\Omega} \subset \mathcal{G}_{F,l}(\tilde{\Omega})$ for any $\bar{\Omega} \in \Lambda$.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

2.1.2. Decomposing a DCNN into sub-networks. Assume that we have a L-layer global network designed for a classification task with samples X and labels Y where each sample $\boldsymbol{x}_i \in \mathbb{R}^{H\times W\times D}$, and a set of Cartesian products $\{\Omega_k\}_{k=1}^K$ where $\Omega_k \subset \Omega$ and $\Omega = [H]\times[W]\times[D]$. For example, let the sample $\boldsymbol{x}_i$ be a natural RGB image (i.e., $D = 3$), then we decompose $\boldsymbol{x}_i$ in the first and second dimensions but not the third dimension. The samples of the k-th subdomain can be denoted as $X_k = \{\mathcal{D}_{\Omega_k}(\boldsymbol{x}_1), \mathcal{D}_{\Omega_k}(\boldsymbol{x}_2), \ldots, \mathcal{D}_{\Omega_k}(\boldsymbol{x}_n)\}$. Then, for these K subdomains, we construct their corresponding sub-networks by decomposing the global network into K sub-networks. The global network is decomposed by partitioning the *width* (i.e., along the channel dimension) of the global network while keeping the depth constant. Hence, the architecture of the sub-networks is almost the same as that of the global network except for the channel number of the feature maps and the first FC layer where the kernel width is adapted to the size of the input of this layer. Formally, following the notations in Section 2.1.1, denote $\boldsymbol{w}^l \in \mathbb{R}^{t_1^l\times t_2^l\times c_{\text{in}}^l\times c_{\text{out}}^l}$ and $\boldsymbol{w}_k^l \in \mathbb{R}^{t_{1,k}^l\times t_{2,k}^l\times c_{k,\text{in}}^l\times c_{k,\text{out}}^l}$ as the convolutional kernel of the l-th layer in the global network and the k-th sub-network, where c_{in}^l, $c_{k,\text{in}}^l$ (c_{out}^l, $c_{k,\text{out}}^l$) are the number of channels of the input (output) feature map, and t_1^l, t_2^l ($t_{1,k}^l$, $t_{2,k}^l$) are the kernel width. In addition, let H^l, W^l (H_k^l, W_k^l) be the height and width of channels of the input feature map in the l-th layer of the global network (the k-th sub-network). Then, we have

$$\begin{cases} \sum_{k=1}^K c_{k,\text{in}}^l = c_{\text{in}}^l, & l = 2,\ldots,L,\\ \sum_{k=1}^K c_{k,\text{out}}^l = c_{\text{out}}^l, & l = 1,\ldots,L-1,\\ c_{k,\text{in}}^1 = c_{\text{in}}^1, c_{k,\text{out}}^L = c_{\text{out}}^L, & \\ t_{1,k}^l = t_1^l, t_{2,k}^l = t_2^l, & \text{except the first FC layer},\\ t_1^l = H^l, t_2^l = W^l, & \text{the first FC layer},\\ t_{1,k}^l = H_k^l, t_{2,k}^l = W_k^l, & \text{the first FC layer}. \end{cases} \tag{2.7}$$

We employ the well-known VGG16 [29] as an example. Figure 2.1 and Table 2.1 illustrate the architecture of VGG16 and the sub-networks decomposed from VGG16. The VGG16 is *uniformly* decomposed into $K = 4$ partitions, and the input samples are also *uniformly* decomposed into $K = 4$ partitions (with $K = n^2$, i.e., $n = 2$ partitions in both the first and second dimension); here, *uniformly* means that the decomposition of inputs is a complete decomposition satisfying (2.6), and

$$\begin{cases} H_k^1 = \lfloor H^1/n\rfloor, W_k^1 = \lfloor W^1/n\rfloor, & \text{for inputs},\\ c_{k,\text{in}}^l = c_{\text{in}}^l/K, & l = 2,\ldots,L,\\ c_{k,\text{out}}^l = c_{\text{out}}^l/K, & l = 1,\ldots,L-1,\\ H_k^l = \lfloor H^l/n\rfloor, W_k^l = \lfloor W^l/n\rfloor, & \text{for convolutional layers and the first FC layer}, \end{cases} \tag{2.8}$$

where the last equality holds when using the same padding in convolutional operations.

Then, the samples of each subdomain and their ground truth are used to train the corresponding sub-network $F_k(\mathcal{D}_{\Omega_k}(\boldsymbol{x}_i);\Theta_k)$ by optimizing the subtasks: for each k,

$$\Theta_k^* = \arg\min_{\Theta_k} \sum_{i=1}^N J_k(\mathcal{D}_{\Omega_k}(\boldsymbol{x}_i);\boldsymbol{y}_i,\Theta_k). \tag{2.9}$$

Note that the trainings of sub-networks are completely independent of each other and can be performed on parallel computers without any interprocessor communication.

The goal of decomposing the subsamples into K subdomains is to save the computational time of the sub-networks and thus accelerate the training. The idea of the width-based

ETNA
Kent State University and
Johann Radon Institute (RICAM)

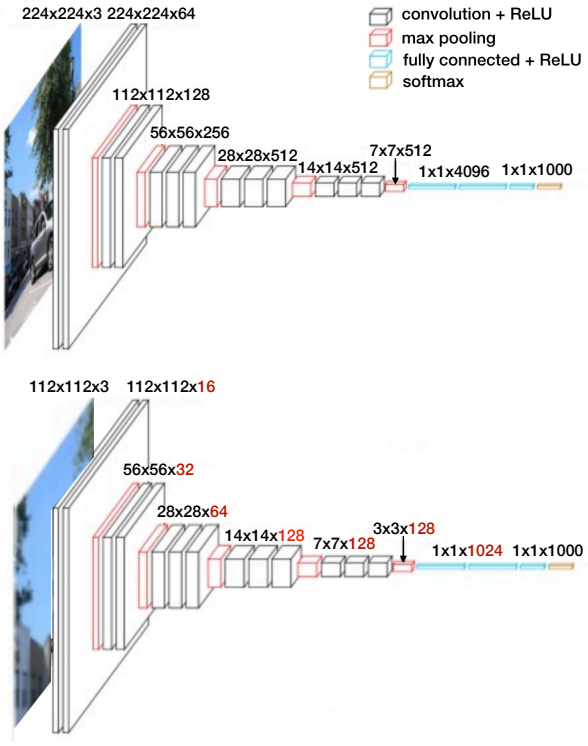

FIG. 2.1. *Illustration of the decomposition of DCNN by the proposed method. The global network (VGG16) is uniformly decomposed into* 4 *sub-networks by partitioning the "width" of the global network while keeping the depth constant. The input is also decomposed into* 4 *partitions. (top) The architecture of VGG16. (bottom) The architecture of one of the sub-networks.*

partitioning is based on the assumption that there is information redundancy when using global networks to deal with the decomposed samples. In addition, by keeping the depth constant, we can compose the sub-networks into the global network in a reverse process discussed in Section 2.2. Note that although the lack of information in the subsample and the reduction in the number of feature maps of sub-networks may reduce the accuracy of the sub-networks, the sub-networks are composed to initialize a global network that will be trained further.

Computational complexity and memory usage analysis of global and sub-networks. We use the number of floating point operations (FLOPs) [26], which is regarded as a strong predictor of the energy usage and latency of a network [33], to estimate the computational complexity of a network. As defined in [26], for a multichannel convolution operation (2.1) with stride 1 and the same padding (without bias), its FLOPs are given by

$$\text{FLOPs} = HW(2c_{\text{in}}t_1t_2 - 1)c_{\text{out}}, \tag{2.10}$$

where each multiplication or addition is counted as one floating point operation; H, W, and c_{in} are the height, width, and number of channels of the input feature map, respectively, t_1 and t_2 are the kernel width, and c_{out} is the number of output channels. Note that, as mentioned before, an FC layer can be regarded as a special case of a convolutional layer; thus, its FLOPs can be obtained similarly. In addition, we denote one GFLOPs to be one billion (i.e., 10^9) FLOPs. Note that, in this paper, the FLOPs of a network refers to the FLOPs of one forward propagation of the network with the batch size of 1.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 2.1
The output size and the kernel size of each layer in the VGG16 (global network) and in the sub-networks, when the global network and the input image are uniformly decomposed into 4 partitions.

	output size		kernel size	
	global network	sub-network	global network	sub-network
input layer	$224 \times 224 \times 3$	$112 \times 112 \times 3$	-	-
layer 1	$224 \times 224 \times 64$	$112 \times 112 \times 16$	$3 \times 3 \times 3 \times 64$	$3 \times 3 \times 3 \times 16$
layer 2	$224 \times 224 \times 64$	$112 \times 112 \times 16$	$3 \times 3 \times 64 \times 64$	$3 \times 3 \times 16 \times 16$
layer 3	$112 \times 112 \times 128$	$56 \times 56 \times 32$	$3 \times 3 \times 64 \times 128$	$3 \times 3 \times 16 \times 32$
layer 4	$112 \times 112 \times 128$	$56 \times 56 \times 32$	$3 \times 3 \times 128 \times 128$	$3 \times 3 \times 32 \times 32$
layer 5	$56 \times 56 \times 256$	$28 \times 28 \times 64$	$3 \times 3 \times 128 \times 256$	$3 \times 3 \times 32 \times 64$
layer 6-7	$56 \times 56 \times 256$	$28 \times 28 \times 64$	$3 \times 3 \times 256 \times 256$	$3 \times 3 \times 64 \times 64$
layer 8	$28 \times 28 \times 512$	$14 \times 14 \times 128$	$3 \times 3 \times 256 \times 512$	$3 \times 3 \times 64 \times 128$
layer 9-10	$28 \times 28 \times 512$	$14 \times 14 \times 128$	$3 \times 3 \times 512 \times 512$	$3 \times 3 \times 128 \times 128$
layer 11-13	$14 \times 14 \times 512$	$7 \times 7 \times 128$	$3 \times 3 \times 512 \times 512$	$3 \times 3 \times 128 \times 128$
layer 14	$1 \times 1 \times 4096$	$1 \times 1 \times 1024$	$7 \times 7 \times 512 \times 4096$	$3 \times 3 \times 128 \times 1024$
layer 15	$1 \times 1 \times 4096$	$1 \times 1 \times 1024$	$1 \times 1 \times 4096 \times 4096$	$1 \times 1 \times 1024 \times 1024$
layer 16	$1 \times 1 \times 1000$	$1 \times 1 \times 1000$	$1 \times 1 \times 4096 \times 1000$	$1 \times 1 \times 1024 \times 1000$

We compare the computational complexity between the global network and the sub-networks. Assume that the input images are uniformly decomposed into $K = n^2$ partitions, and the width of the global network is uniformly decomposed into the same number of partitions. That is, (2.7) and (2.8) hold. We also assume that the convolution in the l-th layer of the global network has the input size $H^l \times W^l \times c_{\text{in}}^l$ and the kernel has the size $t_1^l \times t_2^l \times c_{\text{in}}^l \times c_{\text{out}}^l$. According to (2.10), its computational complexity is given by $O(H^l W^l t_1^l t_2^l c_{\text{in}}^l c_{\text{out}}^l)$. Then, for the layers of the sub-networks except the first convolutional layer and the FC layers, since (2.7) and (2.8) hold, the complexity is $O(H^l W^l t_1^l t_2^l c_{\text{in}}^l c_{\text{out}}^l / K^3)$, which is $1/K^3$ of that of the global network. In the first FC layer, since the kernel width is adapted to the size of the input of this layer, the complexity of each sub-network is given by $O((H^l W^l)^2 c_{\text{in}}^l c_{\text{out}}^l / K^4)$, which is $1/K^4$ of that of the global network. Similarly, these ratios in the first convolutional layer and the FC layers (except the first FC layer and the last layer) are $1/K^2$, and this ratio in the last FC layer is $1/K$. Generally speaking, for a DCNN (e.g., ResNet50 or ResNet101 [7]) in which all the FC layers (except the last layer) are replaced by a single global average pooling layer [20] (global average pooling outputs the spatial average of each feature map at the last convolutional layer), the convolutional layers contain the vast majority of computations; thus the computational complexity of each sub-network is approximately $1/K^3$ of that of the global network. For example, Table 2.2 displays the FLOPs of some global networks (some well-known DCNNs: VGG16, ResNet50, and ResNet101 [7]) and their corresponding sub-networks when the number of partitions $K = 4$, from which we can see that the computational complexity in the forward-propagation step of each sub-network is approximately $1/60$ of that of the global network; and the back-propagation step is similar.

Next, we consider the memory usage of the sub-networks during training, which contains mainly the memory for the parameters, the gradient of the parameters (and the first-order or second-order momentum of the parameters when using the momentum term in the gradient descent algorithm), and the outputs of all layers of networks. 1) For the parameters, in the first layer and the last layer, the number of parameters (or their gradients, momentums) in each sub-network is $1/K$ of that of the global network. In the first FC layer, since the kernel width is adapted to the size of the input of this layer, this ratio is approximately $1/K^3$. In the other layers, this ratio is $1/K^2$. For example, Table 2.2 lists the number of parameters of some global networks (VGG16, ResNet50, and ResNet101) and their corresponding sub-networks when the number of partitions is $K = 4$, from which we can see that the number of parameters

ETNA
Kent State University and
Johann Radon Institute (RICAM)

of each sub-network is approximately $1/13$ of that of the global network for ResNet50 and ResNet101, and this ratio is approximately $1/33$ for VGG. 2) For the outputs of each layer, in the convolutional layers, the memory usage of each sub-network is $1/K^2$ of that of the global network, and this ratio for the FC layers (except the last layer) is $1/K$. In general, the convolutional layers contain the vast majority of the outputs of all layers; thus, during training, the memory usage of the outputs of all layers of the sub-network is about $1/K^2$ of that of the global network.

TABLE 2.2
The floating point operations (FLOPs, with input size of $214 \times 214 \times 3$*) and the number of parameters ('M' means* 10^6*, i.e., one million) of the global networks (VGG16, ResNet50, and ResNet101) and their corresponding sub-networks when the global networks and the input images are uniformly decomposed into 4 partitions.*

network	GFLOPs	# param
VGG16	31.65	138.35M
VGG16-subNet	0.52	4.17M
ResNet50	7.10	25.56M
ResNet50-subNet	0.13	1.99M
ResNet101	14.62	44.55M
ResNet101-subNet	0.25	3.19M

2.2. Composition of DCNNs via the sub-network transfer learning strategy. In this section, we propose an algorithm for composing the trained sub-networks to initialize the global network via the sub-network transfer learning strategy. Since the sub-networks are obtained by partitioning the width of the global network while keeping the depth of the DCNN constant, an intuitive idea is to compose the weights in a reverse process of the decomposition but with the weights connected between different sub-networks set to be zero tensors. Taking VGG16 as an example, the illustration of how to compose 4 sub-networks into one global network is shown in Figure 2.2. More formally, by using the same notations, the weights $\boldsymbol{w}^l$ of the global network are initialized by composing the weights $\{\boldsymbol{w}_k^l\}_k$ of the trained sub-networks as follows:

- For the first layer, by (2.7), we have $t_1^1 = t_{1,k}^1$, $t_2^1 = t_{2,k}^1$, $c_{\text{in}}^1 = c_{k,\text{in}}^1$, and $c_{\text{out}}^1 = \sum_k c_{k,\text{out}}^1$, then let

$$\mathcal{D}_{\Omega_k'}(\boldsymbol{w}^1) = \boldsymbol{w}_k^1, \text{ for } k \in [K],$$

$$\Omega_k' = [t_1^1] \times [t_2^1] \times [c_{\text{in}}^1] \times \{1 + \sum_{i=1}^{k-1} c_{i,\text{out}}^1 : \sum_{i=1}^{k} c_{i,\text{out}}^1\}. \tag{2.11}$$

- For the other convolutional layers and the FC layers except the first and last layers, by (2.7), we have $t_1^l = t_{1,k}^l$, $t_2^l = t_{2,k}^l$, $c_{\text{in}}^l = \sum_k c_{k,\text{in}}^l$, $c_{\text{out}}^l = \sum_k c_{k,\text{out}}^l$, then let

$$\mathcal{D}_{\Omega_k'}(\boldsymbol{w}^l) = \boldsymbol{w}_k^l, \text{ for } k \in [K], \mathcal{D}_{\Omega''}(\boldsymbol{w}^l) = \mathbf{0},$$

$$\Omega_k' = [t_1^l] \times [t_2^l] \times \{1 + \sum_{i=1}^{k-1} c_{i,\text{in}}^l : \sum_{i=1}^{k} c_{i,\text{in}}^l\} \times \{1 + \sum_{i=1}^{k-1} c_{i,\text{out}}^l : \sum_{i=1}^{k} c_{i,\text{out}}^l\},$$

$$\Omega'' = \left([t_1^l] \times [t_2^l] \times [c_{\text{in}}^l] \times [c_{\text{out}}^l]\right) \setminus \cup_{k=1}^{K} \Omega_k', \tag{2.12}$$

where $\mathbf{0}$ is a zero tensor.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

- For the first FC layer, by (2.7), we have $c_{\text{in}}^l = \sum_k c_{k,\text{in}}^l$, $c_{\text{out}}^l = \sum_k c_{k,\text{out}}^l$ and t_1^l, t_2^l are adapted to the size of the input of this layer, and

$$\mathcal{D}_{\Omega_k^{'}}(\boldsymbol{w}^l) = \boldsymbol{w}_k^l, \text{ for } k \in [K], \mathcal{D}_{\Omega^{''}}(\boldsymbol{w}^l) = \mathbf{0},$$

$$\Omega_k^{'} = \big(\mathcal{G}_{F,l-1}(\Omega_k) \cap ([t_1^l] \times [t_2^l] \times \{1 + \sum_{i=1}^{k-1} c_{i,\text{in}}^l : \sum_{i=1}^{k} c_{i,\text{in}}^l\})\big) \times \{1 + \sum_{i=1}^{k-1} c_{i,\text{out}}^l : \sum_{i=1}^{k} c_{i,\text{out}}^l\},$$

$$\Omega^{''} = \big([t_1^l] \times [t_2^l] \times [c_{\text{in}}^l] \times [c_{\text{out}}^l]\big) \setminus \cup_{k=1}^{K} \Omega_k^{'}. \tag{2.13}$$

  It is assumed that valid padding is used in convolutional operations; the case of other kinds of padding can be obtained similarly but considering the boundary of $\Omega_k^{'}$.
- For the last FC layer, we have $t_1^L = t_{1,k}^L$, $t_2^L = t_{2,k}^L$, $c_{\text{in}}^L = \sum_k c_{k,\text{in}}^L$ and $c_{\text{out}}^L = c_{k,\text{out}}^L = C$, then let

$$\mathcal{D}_{\Omega_k^{'}}(\boldsymbol{w}^L) = \boldsymbol{w}_k^L, \text{ for } k \in [K],$$

$$\Omega_k^{'} = [t_1^L] \times [t_2^L] \times \{1 + \sum_{i=1}^{k-1} c_{i,\text{in}}^L : \sum_{i=1}^{k} c_{i,\text{in}}^L\} \times [C]. \tag{2.14}$$

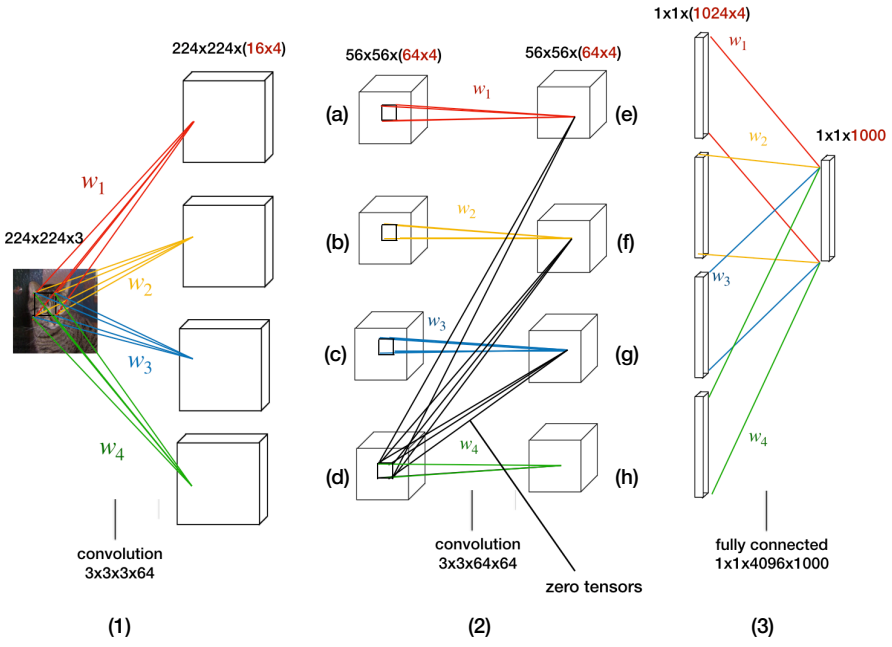

FIG. 2.2. *Illustration of composing 4 sub-networks into one global network by the proposed method. Three layers of the example VGG16 are shown. The weights in the corresponding layers of the k-th sub-network are denoted by $\boldsymbol{w}_k$ (note that, for simplicity, we use the same notation for different layers). (1) The first convolutional layer. (2) One of the intermediate convolutional layers. The connections with zero weights from feature maps (d) to (e), (f), and (g) are shown. Other connections with zero weights (from (a) to (f), (g), and (h); from (b) to (e), (g), and (h); from (c) to (e), (f), and (h)) are omitted for simplicity. (3) The last FC layer.*

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Then, the global network is optimized according to (2.5). In summary, the training of a global network by the decomposition and composition method is described in Algorithm 1.

The proposed decomposition and composition method is motived by the nonlinear preconditioning of Newton's method. The training of the sub-networks and the global network corresponds to a nonlinear process that is local to the subdomains and the outer loop corresponds to the global Newton iteration, respectively. The initialization of the global network by the sub-network transfer learning strategy corresponds to the nonlinear preconditioner. However, compared to the standard nonlinear domain decomposition preconditioner, the stochastic gradient descent method is used instead of Newton's method in the training of the sub-networks and the global network. Besides, the nonlinear preconditioner of DCNNs is applied only once, while the nonlinear preconditioner is applied in every iteration (or some iterations) in the nonlinear preconditioning of Newton's method.

Algorithm 1 Training of a global network by the decomposition and composition method

Input: A classification task with samples $\{X, Y\}$ with $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and ground truth $Y = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$;
A set of Cartesian product $\{\Omega_k\}_{k=1}^K$.

Output: A trained global network $F(\boldsymbol{x}; \boldsymbol{y}, \Theta^*)$ with parameters Θ^*.

1: Decompose samples X to $X_k = \{\mathcal{D}_{\Omega_k}(\boldsymbol{x}_1), \ldots, \mathcal{D}_{\Omega_k}(\boldsymbol{x}_n)\}$ for $k \in [K]$.
2: Construct the corresponding sub-networks $F_k(\mathcal{D}_{\Omega_k}(\boldsymbol{x}_i); \Theta_k)$ by decomposing the width of the global network, as described in Section 2.1.2.
3: Use the gradient descent algorithm to solve the optimization problem for the sub-networks: $\Theta_k^* = \arg\min_{\Theta_k} \sum_{i=1}^N J(\mathcal{D}_{\Omega_k}(\boldsymbol{x}_i); \boldsymbol{y}_i, \Theta_k)$.
4: Compose $\{\Theta_k^*\}_{k=1}^K$ to initialize Θ according to (2.11)–(2.14).
5: Use the gradient descent algorithm to solve the optimization problem: $\Theta^* = \arg\min_{\Theta} \sum_{i=1}^N J(\boldsymbol{x}_i; \boldsymbol{y}_i, \Theta)$, with the initial value Θ.

2.3. Some analysis of the sub-network transfer learning strategy. In this section, we provide some theoretically analysis of the effectiveness of the sub-network transfer learning strategy in which the trained sub-networks are composed to initialize the global network. First, we show that the initialized global network can extract the feature maps learned by the trained sub-networks. Second, it is proven that the initialization of the global network provided by the sub-network transfer learning strategy offers upper and lower bounds for the cost function and the classification accuracy. The bounds depend on those of the trained sub-networks. Note that, for simplicity, it is assumed that the DCNN contains L layers (some convolutional layers and some FC layers), followed by a softmax layer; valid padding is used in convolutional operations, and the decomposition of the inputs is a complete decomposition defined in (2.6).

First, due to the properties of convolutional operations and the stacking architecture of DCNNs, we show that the initialized global network can extract the feature maps learned by the trained sub-networks. In addition, the output of the initialized global network is the sum of those of the trained sub-networks.

THEOREM 2.1. *For samples $\{X, Y\}$ and a global network $F(\boldsymbol{x}; \Theta)$ of L layers where $\boldsymbol{x} \in X$, given a set of Cartesian products $\{\Omega_k\}_{k=1}^K$ satisfying (2.6) and the corresponding trained sub-networks $F_k(\mathcal{D}_{\Omega_k}(\boldsymbol{x}); \Theta_k^*)$ decomposed from the global network, if Θ is initialized by (2.11)–(2.14), then it holds that:*

ETNA
Kent State University and
Johann Radon Institute (RICAM)

(1) for the first $L-1$ layers, we have that,

$$\mathcal{D}_{\Omega_k^l}(F^l(\boldsymbol{x};\Theta^l)) = F_k^l(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*l}), l = 1,\dots,L-1,$$

(2.15)

$$\text{for } \Omega_k^l = \{(\tau_1,\tau_2,\tau_3) \mid (\tau_1,\tau_2,\tau_3) \in \mathcal{G}_{F,l}(\Omega_k), \tau_3 \in \{1+\sum_{i=1}^{k-1} c_{i,\text{out}}^l : \sum_{i=1}^{k} c_{i,\text{out}}^l\}\},$$

where F^l and F_k^l denote the output of the l-th layer of the global network and the k-th sub-network, as defined by (2.2);
(2) for the L-th layer, the outputs satisfy

$$F^L(\boldsymbol{x};\Theta^L) = \sum_{k=1}^{K} F_k^L(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*L}); \tag{2.16}$$

(3) the output of the softmax layer satisfies

$$\hat{\boldsymbol{y}} = \frac{e^{\sum_{k=1}^K F_k^L(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*L})}}{\|e^{\sum_{k=1}^K F_k^L(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*L})}\|_1}. \tag{2.17}$$

Proof. For simplicity, we assume that there are only convolutional operations (without biases) and FC operations in the DCNNs, and there is no pooling and nonlinearity operation; that is, $\theta^l = \boldsymbol{w}^l$ (and $\theta_k^{*l} = \boldsymbol{w}_k^l$). The case of more general DCNNs can be proven similarly.

Following (2.2), denote the input of the l-th layer as $F^{l-1}(\boldsymbol{x};\Theta^{l-1})$ for $l = 2,\dots,L-1$, $\boldsymbol{x}^0 = \boldsymbol{x}$ for $l = 1$, and the output of the l-th layer as $F^l(\boldsymbol{x};\Theta^l) = F^{l-1}(\boldsymbol{x};\Theta^{l-1}) * \boldsymbol{w}^l$. Similarly, for the k-th sub-network F_k, the output of the l-th layer is

$$F_k^l(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*l}) = F_k^{l-1}(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*l-1}) * \boldsymbol{w}_k^l.$$

By representing Ω_k^l in (2.15) as $\Omega_k^l = \tau_{k,l}^1 \times \tau_{k,l}^2 \times \tau_{k,l}^3$, let $\tilde{\Omega}_k^l = [|\tau_{k,l}^1|] \times [|\tau_{k,l}^2|] \times \tau_{k,l}^3$. Then, it is easy to see that

$$\mathcal{D}_{\tilde{\Omega}_k^l}(\mathcal{D}_{\mathcal{G}_{F,l}(\Omega_k)}) = \mathcal{D}_{\Omega_k^l}.$$

(1) We use mathematical induction. For $l = 1$, we have

$$\mathcal{D}_{\Omega_k^1}(F^1(\boldsymbol{x};\Theta^1)) = \mathcal{D}_{\Omega_k^1}(\boldsymbol{x} * \boldsymbol{w}^1) = \mathcal{D}_{\tilde{\Omega}_k^1}(\mathcal{D}_{\mathcal{G}_{F,1}(\Omega_k)}(\boldsymbol{x} * \boldsymbol{w}^1)) = \mathcal{D}_{\Omega_k}(\boldsymbol{x}) * \boldsymbol{w}_k^1,$$

where the third equality holds according to (2.11). Thus (2.15) holds for $l = 1$.

Assume that (2.15) holds for the $(l-1)$-th layer, that is,

$$\mathcal{D}_{\Omega_k^{l-1}}(F^{l-1}(\boldsymbol{x};\Theta^{l-1})) = F_k^{l-1}(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*l-1}).$$

For the l-th layer, we have

$$\begin{aligned}
\mathcal{D}_{\Omega_k^l}(F^l(\boldsymbol{x};\Theta^l)) &= \mathcal{D}_{\Omega_k^l}(F^{l-1}(\boldsymbol{x};\Theta^{l-1}) * \boldsymbol{w}^l) \\
&= \mathcal{D}_{\tilde{\Omega}_k^l}(\mathcal{D}_{\mathcal{G}_{F,l}(\Omega_k)}(F^{l-1}(\boldsymbol{x};\Theta^{l-1}) * \boldsymbol{w}^l)) \\
&= \mathcal{D}_{\tilde{\Omega}_k^{l-1}}(\mathcal{D}_{\mathcal{G}_{F,l-1}(\Omega_k)}(F^{l-1}(\boldsymbol{x};\Theta^{l-1}))) * \boldsymbol{w}_k^l \\
&= \mathcal{D}_{\Omega_k^{l-1}}(F^{l-1}(\boldsymbol{x};\Theta^{l-1})) * \boldsymbol{w}_k^l \\
&= F_k^{l-1}(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*l-1}) * \boldsymbol{w}_k^l = F_k^l(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*l}),
\end{aligned} \tag{2.18}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where the third equality holds according to (2.12); or (2.13) for the first FC layer. Therefore, (2.15) holds for $l = 1, \ldots, L-1$.
(2) For the layer $L-1$, from (1), we have

$$\mathcal{D}_{\Omega_k^{L-1}}(F^{L-1}(\boldsymbol{x}; \Theta^{L-1})) = F_k^{L-1}(\mathcal{D}_{\Omega_k}(\boldsymbol{x}); \Theta_k^{*\,L-1}).$$

Then, for the layer L, according to (2.14), it holds that

$$\begin{aligned}(F^{L-1}(\boldsymbol{x}; \Theta^{L-1})) * \boldsymbol{w}^L &= \sum_{k=1}^{K} \mathcal{D}_{\Omega_k^{L-1}}(F^{L-1}(\boldsymbol{x}; \Theta^{L-1})) * \boldsymbol{w}_k^L \\ &= \sum_{k=1}^{K} F_k^{L-1}(\mathcal{D}_{\Omega_k}(\boldsymbol{x}); \Theta_k^{*\,L-1}) * \boldsymbol{w}_k^L.\end{aligned}$$

That is, (2.16) holds.
(3) It can be directly deduced from the definition (2.3) of the softmax module and (2.16). ∎

In the following theorem, we show that the initialization of the global network by the sub-network transfer learning provides an upper bound and a lower bound for the cost function and the classification accuracy with the corresponding values of the trained sub-networks.

THEOREM 2.2. *For samples $\{X, Y\}$, with $X = \{\boldsymbol{x}_i\}_i^N$ and $Y = \{\boldsymbol{y}_i\}_i^N$, and a global network $F(\boldsymbol{x}_i; \Theta)$ of L layers, given a set of Cartesian products $\{\Omega_k\}_{k=1}^K$ satisfying (2.6) and the corresponding trained sub-networks $F_k(\mathcal{D}_{\Omega_k}(\boldsymbol{x}_i); \Theta_k^*)$ decomposed from the global network, let Θ be initialized by (2.11)–(2.14). Denote the cost function of $F_k(\mathcal{D}_{\Omega_k}(\boldsymbol{x}_i); \Theta_k^*)$ and $F(\boldsymbol{x}_i; \Theta)$ as*

$$r_k = \sum_{i=1}^{N} r_k^i \ \text{ for } r_k^i = J_k(\mathcal{D}_{\Omega_k}(\boldsymbol{x}_i); \boldsymbol{y}_i, \Theta_k^*), \ \text{ and } r = \sum_{i=1}^{N} J(\boldsymbol{x}_i; \boldsymbol{y}_i, \Theta),$$

where J and J_k are defined as in (2.4). In addition, the sets of samples correctly classified by $F_k(\mathcal{D}_{\Omega_k}(\boldsymbol{x}_i); \Theta_k^*)$ and $F(\boldsymbol{x}_i; \Theta)$ are denoted as $\tilde{X}_k \subset X$ and $\tilde{X} \subset X$, respectively. Then, it holds that $\bigcap_{k=1}^K \tilde{X}_k \subset \tilde{X}$ and

$$J(\boldsymbol{x}_i; \boldsymbol{y}_i, \Theta) \le J_k(\mathcal{D}_{\Omega_k}(\boldsymbol{x}_i); \boldsymbol{y}_i, \Theta_k^*), \ k = 1, \ldots, K, \text{ for } \boldsymbol{x}_i \in \bigcap_{k=1}^{K} \tilde{X}_k.$$

More generally,

$$r \le \sum_{i=1}^{N} \Big(\min_{k \in K_i^+} r_k^i + \sum_{k \in K_i^-} r_k^i \Big), \tag{2.19}$$

where $K_i^+ = \{k \mid \boldsymbol{x}_i \in \tilde{X}_k\}$ and $K_i^- = \{k \mid \boldsymbol{x}_i \in X \setminus \tilde{X}_k\}$.

Proof. Denote $\boldsymbol{x}_i^{L,k} = (\boldsymbol{x}_i^{L,k}(1), \ldots, \boldsymbol{x}_i^{L,k}(C))^\top$ and $\boldsymbol{x}_i^L = (\boldsymbol{x}_i^L(1), \ldots, \boldsymbol{x}_i^L(C))^\top$ as the outputs of $F_k(\mathcal{D}_{\Omega_k}(\boldsymbol{x}_i); \Theta_k^*)$ and $F(\boldsymbol{x}_i; \Theta)$, respectively, in the L-th layer. For any $\boldsymbol{x}_i \in \tilde{X}_k$ and any $j \ne c_{\boldsymbol{x}_i}$, where $\boldsymbol{y}_i(c_{\boldsymbol{x}_i}) = 1$ (i.e., the ground truth of $\boldsymbol{x}_i$ is the $c_{\boldsymbol{x}_i}$-th class), we have that $\boldsymbol{x}_i^{L,k}(j) \le \boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i})$. That is, for any $\boldsymbol{x}_i \in \bigcap_{k=1}^K \tilde{X}_k$, $\boldsymbol{x}_i^{L,k}(j) \le \boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i})$, $\forall j \ne c_{\boldsymbol{x}_i}$, for $k = 1, \ldots, K$. From (2.16) in Theorem 2.1, we have that $\boldsymbol{x}_i^L(j) \le \boldsymbol{x}_i^L(c_{\boldsymbol{x}_i})$, $\forall j \ne c_{\boldsymbol{x}_i}$. Thus, $\boldsymbol{x}_i \in \tilde{X}$, that is, $\bigcap_{k=1}^K \tilde{X}_k \subset \tilde{X}$.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

According to (2.3) and (2.17), the output of the softmax layer of the initialized global network is given by

(2.20)
$$\hat{\boldsymbol{y}}(c') = \frac{\exp(\sum_{k=1}^{K} \boldsymbol{x}_i^{L,k}(c'))}{\|\exp(\sum_{k=1}^{K} \boldsymbol{x}_i^{L,k})\|_1} = \frac{\exp(\sum_{k=1}^{K} \boldsymbol{x}_i^{L,k}(c'))}{\sum_{c=1}^{C} \exp(\sum_{k=1}^{K} \boldsymbol{x}_i^{L,k}(c))}, \text{ for } c' = 1, \dots, C.$$

Since the ground truth of $\boldsymbol{x}_i$ is the $c_{\boldsymbol{x}_i}$-th class, then $\boldsymbol{y}(c) = 1$ if $c = c_{\boldsymbol{x}_i}$ and 0 otherwise. Then, for any $\boldsymbol{x}_i \in \bigcap_{k=1}^{K} \tilde{X}_k$ and $\tilde{k} \in [K]$, according to (2.4), (2.17), and (2.20), we have

(2.21)
$$\begin{aligned}
J(\boldsymbol{x}_i; \boldsymbol{y}_i, \Theta) &= -\sum_{c=1}^{C} \boldsymbol{y}(c) \log \hat{\boldsymbol{y}}(c) = -\log \hat{\boldsymbol{y}}(c_{\boldsymbol{x}_i}) \\
&= -\log \frac{\exp(\sum_{k=1}^{K} \boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i}))}{\sum_{c=1}^{C} \exp(\sum_{k=1}^{K} \boldsymbol{x}_i^{L,k}(c))} \\
&= -\log \frac{\exp(\boldsymbol{x}_i^{L,\tilde{k}}(c_{\boldsymbol{x}_i}))}{\sum_{c=1}^{C} \exp(\sum_{k=1}^{K} \boldsymbol{x}_i^{L,k}(c) - \sum_{k\neq\tilde{k}} \boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i}))} \\
&= -\log \frac{\exp(\boldsymbol{x}_i^{L,\tilde{k}}(c_{\boldsymbol{x}_i}))}{\sum_{c=1}^{C} \exp(\boldsymbol{x}_i^{L,\tilde{k}}(c) + \sum_{k\neq\tilde{k}} (\boldsymbol{x}_i^{L,k}(c) - \boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i})))} \\
&\le -\log \frac{\exp(\boldsymbol{x}_i^{L,\tilde{k}}(c_{\boldsymbol{x}_i}))}{\sum_{c=1}^{C} \exp(\boldsymbol{x}_i^{L,\tilde{k}}(c))} = J(\mathcal{D}_{\Omega_{\tilde{k}}}(\boldsymbol{x}_i); \boldsymbol{y}_i, \Theta_{\tilde{k}}^*),
\end{aligned}$$

where the inequality holds since $\boldsymbol{x}_i^{L,k}(c) \le \boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i})$ for any $k \in [K]$.

Assume that $K_i^+ \neq \emptyset$ and denote $\tilde{k}^i = \arg\min_{k\in K_i^+} r_k^i$. Similar to (2.21), it holds that

$$\begin{aligned}
&J(\boldsymbol{x}_i; \boldsymbol{y}_i, \Theta) \\
&= -\log \frac{\exp\left(\boldsymbol{x}_i^{L,\tilde{k}^i}(c_{\boldsymbol{x}_i})\right)}{\sum_{c=1}^{C} \exp\left(\boldsymbol{x}_i^{L,\tilde{k}^i}(c) + \sum_{\substack{k\neq\tilde{k}^i \\ k\in K_i^+}} \left(\boldsymbol{x}_i^{L,k}(c) - \boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i})\right) + \sum_{k\in K_i^-} \left(\boldsymbol{x}_i^{L,k}(c) - \boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i})\right)\right)} \\
&\le -\log \frac{\exp\left(\boldsymbol{x}_i^{L,\tilde{k}^i}(c_{\boldsymbol{x}_i})\right)}{\sum_{c=1}^{C} \exp\left(\boldsymbol{x}_i^{L,\tilde{k}^i}(c) + \sum_{k\in K_i^-} \left(\boldsymbol{x}_i^{L,k}(c) - \boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i})\right)\right)} \\
&\le -\log \frac{\exp\left(\boldsymbol{x}_i^{L,\tilde{k}^i}(c_{\boldsymbol{x}_i})\right)}{\left(\sum_{c=1}^{C} \exp(\boldsymbol{x}_i^{L,\tilde{k}^i}(c))\right)\left(\sum_{c=1}^{C} \exp\left(\sum_{k\in K_i^-} (\boldsymbol{x}_i^{L,k}(c) - \boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i}))\right)\right)} \\
&= \min_{k\in K_i^+} r_k^i - \log \frac{1}{\sum_{c=1}^{C} \exp\left(\sum_{k\in K_i^-} (\boldsymbol{x}_i^{L,k}(c) - \boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i}))\right)} \\
&\le \min_{k\in K_i^+} r_k^i - \log \frac{1}{\prod_{k\in K_i^-} \left(\sum_{c=1}^{C} \exp(\boldsymbol{x}_i^{L,k}(c) - \boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i}))\right)} \\
&= \min_{k\in K_i^+} r_k^i - \sum_{k\in K_i^-} \log \frac{\exp\left(\boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i})\right)}{\sum_{c=1}^{C} \exp\left(\boldsymbol{x}_i^{L,k}(c)\right)} = \min_{k\in K_i^+} r_k^i + \sum_{k\in K_i^-} r_k^i.
\end{aligned}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

If $K_i^+ = \emptyset$, then

$$J(\boldsymbol{x}_i; \boldsymbol{y}_i, \Theta) \leq -\log \frac{\Pi_{k=1}^K \exp(\boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i}))}{\Pi_{k=1}^K (\sum_{c=1}^C \exp(\boldsymbol{x}_i^{L,k}(c)))}$$
$$= \sum_{k \in K_i^-} -\log \frac{\exp(\boldsymbol{x}_i^{L,k}(c_{\boldsymbol{x}_i}))}{\sum_{c=1}^C \exp(\boldsymbol{x}_i^{L,k}(c))} = \sum_{k \in K_i^-} r_k^i.$$

Thus, (2.19) holds. ∎

From Theorem 2.2, we know that the global network initialized by the trained sub-networks has a good initialization in terms of the cost function and the classification accuracy if the samples are reasonably decomposed and the sub-networks are well trained. In fact, in our experiments in Section 3, the sub-network transfer learning strategy can provide a good initialization for the global network when the samples are just uniformly decomposed. Sometimes the experimental results are stronger than the theoretical estimates in Theorem 2.2. Furthermore, some assumptions are made to simplify the analysis and to make the proof rigorous; however, the results of our experiments imply that the sub-network transfer learning strategy can also provide a good initialization for the global network that has more general architectures; for example, that contains global average pooling layers or skip connections.

2.4. Extension to Other Types of Neural Networks. In this section, we will extend the proposed approach of decomposition and composition of DCNNs for classification to other types of network architectures, and to other types of data, e.g., 1D and 3D data.

Fully connected neural networks. For 1D input data, fully connected neural networks or 1D CNNs are usually used. For 1D CNNs [11], the 1D convolution of size k can be regarded as a special 2D convolution of size $k \times 1$. Thus, the proposed approach and the theoretical results also apply to 1D CNNs. For fully connected neural networks, the fully connected layer can be similarly regarded as a special case of a convolutional operation with the kernel width to be 1, with regarding the number of neurons as the channel number. Since the input sample is decomposed along the channel, we modify the composition of the weights in the first layer (2.11) to

$$\mathcal{D}_{\Omega_k'}(\boldsymbol{w}^1) = \boldsymbol{w}_k^1, \text{ for } k \in [K],$$
$$\Omega_k' = [t_1^1] \times [t_2^1] \times \Omega_k \times \{1 + \sum_{i=1}^{k-1} c_{i,\text{out}}^1 : \sum_{i=1}^{k} c_{i,\text{out}}^1\},$$

where $\{\Omega_k\}_{k=1}^K$ is the set of Cartesian products used to decompose the input samples. With such modification, the proposed approach and the theoretical results are also applicable to fully connected neural networks.

Regression tasks. Generally, classification tasks are used to predict the discrete values, and regression tasks are used to predict the continuous values. The network architecture for regression tasks can be set to be almost the same as that for classification tasks, except the last FC layer (i.e., the L-th layer) and the soft-max layer [15]. More specifically, we remove the soft-max layer, and replace the last FC layer with an FC layer with linear activations equal in number to the dimension of the target space. For a regression task with sample $X = \{\boldsymbol{x}_i\}_{i=1}^N$ and their corresponding ground truth $Y = \{\boldsymbol{y}_i\}_{i=1}^N$, assume that a network for this regression

ETNA
Kent State University and
Johann Radon Institute (RICAM)

task outputs the prediction $\hat{Y} = \{\hat{\boldsymbol{y}}_i\}_{i=1}^N$, then its cost function is defined by

$$D(Y, \hat{Y}) = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\|,$$

where $\|\cdot\|$ denotes some kind of norm. This cost function measures the mean absolute error (MAE) or the mean squared error (MSE) when the norm is the L_1 or the square of the L_2 norm. Since the first $L-1$ layers are the same as those of the network for classification tasks, the proposed approach and the theoretical results for the first $L-1$ layers are applicable to the networks for regression tasks. Then for the L-th layer, we first divide the weights of the sub-networks by K before composing these weights, that is, (2.14) is modified to

$$\mathcal{D}_{\Omega'_k}(\boldsymbol{w}^L) = \frac{1}{K} \boldsymbol{w}_k^L, \text{ for } k \in [K].$$

With such modification, the result (2.16) in Theorem 2.1 satisfies

$$F^L(\boldsymbol{x}; \Theta^L) = \frac{1}{K} \sum_{k=1}^{K} F_k^L(\mathcal{D}_{\Omega_k}(\boldsymbol{x}); \Theta_k^{*L}). \tag{2.22}$$

That is, assume that the k-th sub-networks outputs the prediction $\hat{Y}_k = \{\hat{\boldsymbol{y}}_{k,i}\}_{i=1}^N$ for $k = 1, \ldots, K$, then the outputs of the global network initialized by these sub-networks satisfy

$$\hat{\boldsymbol{y}}_i = \frac{1}{K} \sum_{k=1}^{K} \hat{\boldsymbol{y}}_{k,i}, \text{for } i = 1, \ldots, N. \tag{2.23}$$

Therefore, denoting the cost function of the sub-networks by $D(Y, \hat{Y}_k)$, the cost function of the global network satisfies

$$D(Y, \hat{Y}) = \frac{1}{N} \sum_{i=1}^{N} \|\frac{1}{K} \sum_{k=1}^{K} \hat{\boldsymbol{y}}_{k,i} - \boldsymbol{y}_i\| \leq \frac{1}{N} \sum_{i=1}^{N} \frac{1}{K} \sum_{k=1}^{K} \|\hat{\boldsymbol{y}}_{k,i} - \boldsymbol{y}_i\| = \frac{1}{K} \sum_{k=1}^{K} D(Y, \hat{Y}_k).$$

Therefore, the cost function of the global network is bounded from above by the average of the cost function of the trained sub-networks.

Auto-encoders. A typical auto-encoder is a feedforward neural network that is built by a stack of fully connected layers, and the output layer has the same number of neurons as the input layer. The idea is that the auto-encoder is trained to reconstruct the input pattern at the output of the network by minimizing the difference between the input and the output. As mentioned above, by regarding the fully connected layer as a special case of a convolutional operation, the composition of the auto-encoder can be set to (2.12) and (2.11). Then, (2.15) in Theorem 2.1 holds for all the layer of the auto-encoder. Thus, the output of the global network initialized by the sub-networks is the concatenation of the outputs of the sub-networks. More formally, for an input vector $\boldsymbol{x}$ and its decomposition $\boldsymbol{x}_k = \mathcal{D}_{\Omega_k}(\boldsymbol{x})$, denote the output of the k-th sub-network by $\hat{\boldsymbol{x}}_k$, then the output $\hat{\boldsymbol{x}}$ of the global network initialized by the sub-networks satisfies

$$\mathcal{D}_{\Omega_k}(\hat{\boldsymbol{x}}) = \hat{\boldsymbol{x}}_k.$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Thus, the reconstruction error satisfies

$$D(\boldsymbol{x},\hat{\boldsymbol{x}}) = \|\boldsymbol{x}-\hat{\boldsymbol{x}}\| = \|\sum_{k=1}^{K}(\mathcal{D}_{\Omega_k}(\boldsymbol{x}) - \mathcal{D}_{\Omega_k}(\hat{\boldsymbol{x}}))\|$$
$$\leq \sum_{k=1}^{K}\|\mathcal{D}_{\Omega_k}(\boldsymbol{x}) - \mathcal{D}_{\Omega_k}(\hat{\boldsymbol{x}})\| = \sum_{k=1}^{K} D(\boldsymbol{x}_k,\hat{\boldsymbol{x}}_k).$$

Therefore, the cost function of the global network is upper bounded by the sum of the cost function of the sub-networks.

Residual neural networks. As proposed in [7], a residual block of a residual neural network is defined as

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x},\{W_i\}) + \boldsymbol{x}, \tag{2.24}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the input and output of the layers considered, and $\mathcal{F}(\boldsymbol{x},\{W_i\})$ represents the residual mapping, which is usually a stack of convolutions. It is assumed that the output channel of $\mathcal{F}(\boldsymbol{x},\{W_i\})$ and $\boldsymbol{x}$ are the same. Besides, as mentioned above, we assume that a valid padding is used in convolutional operations. Thus, to ensure the sum in (2.24) is well-defined, it is assumed that $\boldsymbol{x}$ is cropped on the boundaries before adding such that its size is the same as $\mathcal{F}(\boldsymbol{x},\{W_i\})$. Without loss of generality, let $\mathcal{F}(\boldsymbol{x},\{W_i\})$ be a convolutional operation with a convolutional kernel W. The cropping of $\boldsymbol{x}$ can be regarded as convolving a convolutional kernel K_W across $\boldsymbol{x}$, where K_W is of the same size as W and

$$(K_W)_{\lceil (w+1)/2\rceil,\lceil (h+1)/2\rceil,k,k} = 1 \text{ and } 0 \text{ otherwise, for all output channel } k, \tag{2.25}$$

where w and h denote the widths of the kernel. Therefore, (2.24) can be rewritten as

$$\boldsymbol{y} = \boldsymbol{x} * W + \boldsymbol{x} * K_W, \tag{2.26}$$

where the activation function and the biases are omitted for simplicity since the element-wise operations do not affect the decomposition and composition operations. Besides, for the case that $\mathcal{F}(\boldsymbol{x},\{W_i\})$ contains more than one convolutional layers, the second term in (2.26) is a stack of convolutions that correspond to the convolutions of $\mathcal{F}(\boldsymbol{x},\{W_i\})$.

The proposed decomposition and composition method is applicable to DCNNs containing residual blocks, which is performed along the channel dimension of the weights. Then, it can be proven that (2.15) also holds for residual neural networks. From (2.26), we have

$$F_k^l(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*l}) = F_k^{l-1}(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*l-1}) * \boldsymbol{w}_k^l + F_k^{l-1}(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*l-1}) * K_k^l$$
$$F^l(\boldsymbol{x};\Theta^l) = F^{l-1}(\boldsymbol{x};\Theta^l) * \boldsymbol{w}^l + F^{l-1}(\boldsymbol{x};\Theta^l) * K^l,$$

where K_k^l and K^l, that correspond to $\boldsymbol{w}_k^l$ and $\boldsymbol{w}^l$, respectively, are defined as (2.25). Assuming that (2.15) holds for the $(l-1)$-th layer, from (2.18), we have

$$\mathcal{D}_{\Omega_k^l}(F^{l-1}(\boldsymbol{x};\Theta^{l-1}) * \boldsymbol{w}^l) = F_k^{l-1}(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*l-1}) * \boldsymbol{w}_k^l,$$
$$\mathcal{D}_{\Omega_k^l}(F^{l-1}(\boldsymbol{x};\Theta^l) * K^l) = F_k^{l-1}(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*l-1}) * K_k^l.$$

Thus,

$$\mathcal{D}_{\Omega_k^l}(F^l(\boldsymbol{x};\Theta^l)) = \mathcal{D}_{\Omega_k^l}(F^{l-1}(\boldsymbol{x};\Theta^l) * \boldsymbol{w}^l) + \mathcal{D}_{\Omega_k^l}(F^{l-1}(\boldsymbol{x};\Theta^l) * K^l)$$
$$= F_k^l(\mathcal{D}_{\Omega_k}(\boldsymbol{x});\Theta_k^{*l}).$$

Thus, (2.15) is applicable to residual neural networks, then other results in Theorems 2.1 and 2.2 can also be proven in the case of classification tasks or regression tasks.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Recurrent neural networks. A recurrent neural network (RNN) is a type of artificial neural network that treats sequential data or time-series data. We consider a typical RNN that contains a stack of blocks shown in Figure 2.3, that is, a standard fully connected neural network plus added loops. In RNNs, the hidden state is dependent on the previous computations and the weights are shared across all time steps. More formally, we denote the input, the hidden state, and the output at time step t and the l-th block by $\boldsymbol{x}_t^l$, $\boldsymbol{s}_t^l$, and $\boldsymbol{o}_t^l$, respectively, which satisfy

$$\boldsymbol{s}_t^l = \sigma(U^l \boldsymbol{x}_t^l + W^l \boldsymbol{s}_{t-1}^l + \boldsymbol{b}^l), \tag{2.27}$$

$$\boldsymbol{o}_t^l = \sigma(V^l \boldsymbol{s}_t^l + \boldsymbol{c}^l), \tag{2.28}$$

$$\boldsymbol{x}_t^{l+1} = \boldsymbol{o}_t^l,$$

with $\boldsymbol{x}_t = \boldsymbol{x}_t^1$ denoting the input of the RNN at time step t. By regarding the fully connected layer as a special case of a convolutional operation, the proposed approaches of decomposition and composition of DCNNs are applicable to RNNs. Note that, due to (2.27), the decomposition of the output channel of U^l and W^l need to be the same to ensure that the sum of the terms corresponding to the previous hidden state and the input is well-defined.

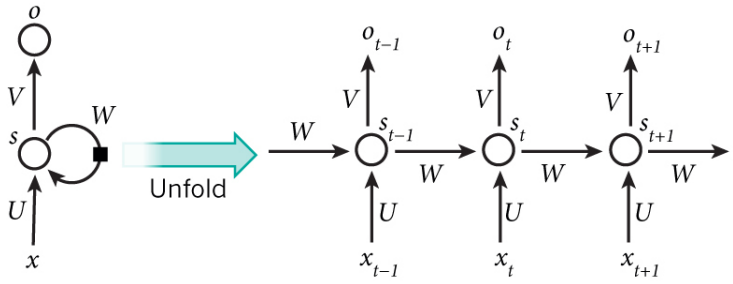

FIG. 2.3. *A block of a recurrent neural network and the unfolding in time of the computation involved in its forward computation, where $\boldsymbol{x}_t$, $\boldsymbol{s}_t$, and $\boldsymbol{o}_t$ denote the input, the hidden state, and the output at time step t, respectively.*

For the theoretical analysis, (2.15) can be shown to be true for RNNs. For simplicity, with the input $\boldsymbol{x}_{t,k}^1 := \mathcal{D}_{\Omega_k}(\boldsymbol{x}_t)$, we denote the input, the hidden state, and the output at time step t and the l-th block of the k-th sub-network by $\boldsymbol{x}_{t,k}^l$, $\boldsymbol{s}_{t,k}^l$, and $\boldsymbol{o}_{t,k}^l$, respectively, and $\boldsymbol{x}_t^l$, $\boldsymbol{s}_t^l$, and $\boldsymbol{o}_t^l$ for the global network initialized by the sub-networks. Besides, similar to the case of DCNNs, we denote the term Ω_k^l in (2.15) by $\Omega_{k,i}^l$, $\Omega_{k,h}^l$, and $\Omega_{k,o}^l$, which correspond to the input, the hidden state, and the output, respectively. Then, we prove

$$\mathcal{D}_{\Omega_{k,i}^l}(\boldsymbol{x}_t^l) = \boldsymbol{x}_{t,k}^l,\ \mathcal{D}_{\Omega_{k,h}^l}(\boldsymbol{s}_t^l) = \boldsymbol{s}_{t,k}^l,\ \mathcal{D}_{\Omega_{k,o}^l}(\boldsymbol{o}_t^l) = \boldsymbol{o}_{t,k}^l, \text{ for all } t \text{ and } l. \tag{2.29}$$

Assuming that the time-series data starts from time step 0, with fixing $t = 0$, the RNN is actually a standard fully connected neural network, then (2.29) holds when $t = 0$. Additionally, $\mathcal{D}_{\Omega_{k,i}^l}(\boldsymbol{x}_t^l) = \boldsymbol{x}_{t,k}^l$ holds for all t when $l = 1$. Therefore, by using mathematical induction, with assuming that $\mathcal{D}_{\Omega_{k,i}^l}(\boldsymbol{x}_{t+1}^l) = \boldsymbol{x}_{t+1,k}^l$ and $\mathcal{D}_{\Omega_{k,h}^l}(\boldsymbol{s}_t^l) = \boldsymbol{s}_{t,k}^l$, we only need to prove that $\mathcal{D}_{\Omega_{k,h}^l}(\boldsymbol{s}_{t+1}^l) = \boldsymbol{s}_{t+1,k}^l$ and $\mathcal{D}_{\Omega_{k,o}^l}(\boldsymbol{o}_{t+1}^l) = \boldsymbol{o}_{t+1,k}^l$. As mentioned above, σ, $\boldsymbol{b}^l$, and $\boldsymbol{c}^l$ are omitted in the following derivation for simplicity. First, we have

$$\mathcal{D}_{\Omega_{k,h}^l}(\boldsymbol{s}_{t+1}^l) = \mathcal{D}_{\Omega_{k,h}^l}(U^l \boldsymbol{x}_{t+1}^l + W^l \boldsymbol{s}_t^l) = \mathcal{D}_{\Omega_{k,h}^l}(U^l \boldsymbol{x}_{t+1}^l) + \mathcal{D}_{\Omega_{k,h}^l}(W^l \boldsymbol{s}_t^l).$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Then, since the decomposition of the output channel of U^l and W^l is the same as mentioned above, and $\mathcal{D}_{\Omega^l_{k,i}}(\boldsymbol{x}^l_{t+1}) = \boldsymbol{x}^l_{t,k+1}$, $\mathcal{D}_{\Omega^l_{k,h}}(\boldsymbol{s}^l_t) = \boldsymbol{s}^l_{t,k}$, then, from (2.18), we have

$$\mathcal{D}_{\Omega^l_{k,h}}(U^l\boldsymbol{x}^l_{t+1}) = U^l_k\boldsymbol{x}^l_{t+1,k},\ \mathcal{D}_{\Omega^l_{k,h}}(W^l\boldsymbol{s}^l_t) = W^l_k\boldsymbol{s}^l_{t,k},$$

where U^l_k and W^l_k denote the weights that correspond to the k-th sub-network. Thus,

$$\mathcal{D}_{\Omega^l_{k,h}}(\boldsymbol{s}^l_{t+1}) = U^l_k\boldsymbol{x}^l_{t+1,k} + W^l_k\boldsymbol{s}^l_{t,k} = \boldsymbol{s}^l_{t+1,k}.$$

From (2.28), the connection between $\boldsymbol{o}^l_{t+1}$ and $\boldsymbol{s}^l_{t+1}$ is actually a standard fully connected layer, thus $\mathcal{D}_{\Omega^l_{k,o}}(\boldsymbol{o}^l_{t+1}) = \boldsymbol{o}^l_{t+1,k}$ holds. Additionally, for the output and upper bound of the cost function, the theoretical results can be proven for classification tasks or regression tasks.

Besides, for 3D image data, 3D CNNs are usually used, to which the proposed approach and the theoretical results are applicable by simply replacing 2D convolutions with 3D convolutions.

3. Experiments. In this section, the proposed approach is evaluated by some experiments with some image classification tasks. In addition, some visualization techniques are implemented to visualize the initialization obtained by the composition of the sub-networks. The results demonstrate that the sub-network transfer learning strategy can indeed provide a good initialization for the global network and accelerate the training of the global network. All the experiments are carried out using the TensorFlow 1.2 library on a workstation with 4 NVIDIA Tesla V100 32G GPUs. We compare the performance between two training strategies: 1) to train the global network with the parameters randomly initialized (referred to as "GNet-R"), 2) to train the sub-networks (referred to as "SNets") in parallel and then further train the global network initialized by the sub-network transfer learning method (referred to as "GNet-T"). The sub-networks and the global networks are trained using the same computing resources. The global networks are trained using the data-parallel strategy on a workstation with 4 GPUs. The sub-networks are trained in parallel by a multiprocessing strategy, in which GPUs are uniformly assigned to sub-networks; for example, for the case of 4 (or 8, 16) sub-networks, one GPU is assigned to one (or 2, 4) sub-network. Besides, all the models are optimized using the mini-batch Adam optimization algorithm.

3.1. Image classification tasks.

3.1.1. Experiments on CIFAR-10/CIFAR-100. We test the proposed method on the CIFAR-10/CIFAR-100 dataset [12] consisting of 50k training images and 10k validation images in 10/100 classes. The images of the dataset are RGB images of size 32×32.

Experimental setting. We use the residual networks [7]. Networks of three different numbers of layers are used, all of which are a stack of convolutional layers (with a 3×3 convolutional operation, a batch normalization (BN) [10] operation, and a ReLU function) that end with a global average pooling, a 10-way (or 100-way) FC layer, followed by a softmax layer. In these networks, the first layer is a convolutional layer, followed by a stack of $6n$ layers with convolutions on the feature maps of sizes $\{32, 16, 8\}$, with $2n$ convolutional layers for each feature map size and with the filter numbers of $\{64, 128, 256\}$. That is, there are $6n + 2$ stacked weighted layers in total. In addition, shortcut connections are connected to the pairs of 3×3 layers; i.e., a total of $3n$ shortcuts. Note that we experiment on the cases of $n = 5, 9, 18$ (i.e., $32, 56, 110$ layers), which are denoted by ResNet32, ResNet56, and ResNet110, respectively. In our experiments, we decompose the images into $K = 4$ sub-images. The input images are decomposed into 4 sub-images with and without overlap by decomposing into 2 partitions in both the width and height dimensions. The sub-images

ETNA
Kent State University and
Johann Radon Institute (RICAM)

without and with overlap are of size 16×16 and 20×20, respectively. The networks are uniformly decomposed into $K = 4$ sub-networks. The input data are normalized to have a mean of zero and then scaled to $[0, 1]$. In addition, the training data are augmented by randomly horizontal flipped and randomly shifted in an on-the-fly way during the training.

We compare the training time and the classification accuracy between the two training strategies mentioned before. When evaluating the classification accuracy, the randomly initialized global networks are trained for 200 epochs starting with a learning rate of 0.001, which is divided by 10 at 80, 140, and 160 epochs and then divided by 2 at 180 epochs. The sub-networks are trained for 120 epochs also starting with a learning rate of 0.001, which and is divided by 10 at 60, 80, and 100 epochs. For the global networks initialized by the sub-network transfer learning method, they are trained for 80 epochs, and the learning rate starts from 0.0001 and is divided by 10 at 20 and 40 and then divided by 2 at 60 epochs. That is, in these two training strategies, the numbers of epochs of training of the strategies are the same; i.e., 200. Also the learning rate is the same for the last 80 epochs. The batch size is set to 32 in all the cases. In addition, an adaptive learning rate schedule is used, in which the learning rate is divided by $10^{1/2}$ if it makes no improvement in the classification accuracy for validation data in 5 epochs.

Training time analysis. We compare the training time between the global network and four sub-networks using the same computing resources. Table 3.1 shows the FLOPs and the number of parameters of the global network and one sub-network; with the dataset Cifar10. Since their network architecture is almost the same, the number of parameters of the sub-network in the overlapping case is the same as that in the non-overlapping case, which is approximately $1/16$ of that of the global network. Besides, in the case of overlap and non-overlap, the computational complexity of each sub-network is approximately $1/40$ and $1/60$ of that of the global network, respectively. Table 3.2 shows the training times for 10 epochs with different batch sizes of the global network and four sub-networks, taking ResNet32 for Cifar10 as an example. From the results, we observe that for the two training strategies, 1) without data augmentation, the training time of the global network is on average 2.33 (or 2.70) times that of four sub-networks in the overlapping (or non-overlapping) case; 2) with data augmentation, the training time of the global network is on average 2.19 (or 2.34) times that of four sub-networks in the overlapping (or non-overlapping) case; 3) the memory usage of the global network is approximately 4 times that of four sub-networks in the non-overlapping case, which is consistent with the analysis in Section 2.1.2. Note that in the case of data augmentation, the parallel efficiency of the sub-networks decreases due to images processing by the CPUs, especially when the batch size becomes larger. This can be improved by using GPUs (NVIDIA Data Loading Library), but this is beyond the scope of this paper.

TABLE 3.1

The FLOPs and the number of parameters of global networks with different depth (for dataset Cifar10) and one of their corresponding sub-networks with and without (inside the brackets) overlap in the sub-images. The sub-images without and with overlap are of size 16×16 *and* 20×20*, respectively. The "ratio" refers to the ratio of the FLOPs or the number of parameters of the global network to that of the sub-network. The global networks and the input images are uniformly decomposed into 4 partitions.*

	ResNet32			ResNet56			ResNet110		
	global-net	sub-net	ratio	global-net	sub-net	ratio	global-net	sub-net	ratio
# param	7.44M	0.47M	15.83	13.65M	0.86M	15.87	27.62M	1.74M	15.87
GFLOPs	2.24	0.056 (0.037)	40 (60.54)	4.08	0.103 (0.067)	39.61 (60.90)	8.23	0.207 (0.135)	39.76 (60.96)

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 3.2

Taking ResNet32 for Cifar10 as an example, the training times for 10 epochs (with different batch sizes) of the global network and four sub-networks with and without data augmentation, when the global networks are uniformly decomposed into 4 partitions, and the input images are decomposed into 4 partitions with and without (inside the brackets) overlap. The "ratio" means the ratio of the training time of the global network to that of the sub-network. Note that "—" means that there is not enough storage space when training the networks.

	batch size	32	128	512	2048	8192	16384	32768
w/o data augmentation	global network	1395 s	461 s	228 s	182 s	173 s	—	—
	4 sub-networks	620 s (565s)	180 s (170 s)	94 s (83 s)	83 s (67 s)	77 s (60 s)	80 s (64 s)	68 s (—)
	ratio	2.25 (2.46)	2.56 (2.71)	2.42 (2.75)	2.19 (2.71)	2.25 (2.88)	—	—
w/ data augmentation	global network	1484 s	472 s	216 s	174 s	169 s	—	—
	4 sub-networks	627 s (588 s)	191 s (181)	101 s (89 s)	88 s (87 s)	84 s (79 s)	94 s (88 s)	82 s (—)
	ratio	2.37 (2.52)	2.47 (2.60)	2.14 (2.43)	1.98 (2.00)	2.01 (2.14)	—	—

TABLE 3.3

The classification accuracy rates of the validation data of the global networks initialized by the sub-network transfer learning strategy ("GNet-T") and those randomly initialized ("GNet-R") and that of the 4 trained sub-networks ("SNet-1" to "SNet-4") with and without (inside the brackets) overlap in the sub-images. The sub-images without and with overlap are of size 16×16 and 20×20, respectively. Suffix-trained means that the global networks are trained, otherwise, it indicates the global networks are initialized but not further trained.

	Cifar10 (%)			Cifar100 (%)		
	ResNet32	ResNet56	ResNet110	ResNet32	ResNet56	ResNet110
SNet-1	80.49 (68.92)	81.02 (69.27)	81.50 (69.08)	52.30 (40.11)	53.40 (40.35)	52.36 (39.73)
SNet-2	80.33 (68.18)	80.08 (68.64)	81.53 (68.09)	51.55 (44.01)	52.91 (39.23)	52.00 (37.49)
SNet-3	80.63 (68.68)	81.08 (68.79)	80.10 (68.69)	52.58 (40.04)	52.74 (39.65)	53.21 (36.55)
SNet-4	78.97 (68.40)	79.52 (68.18)	79.81 (67.92)	52.21 (40.47)	52.16 (39.73)	51.46 (37.97)
GNet-T	**88.33** (81.11)	**88.28** (80.65)	**87.87** (80.54)	**57.70** (42.74)	**60.44** (43.70)	**56.00** (38.85)
GNet-T-trained	**94.40** (94.24)	**94.76** (94.08)	**94.44** (94.27)	**74.24** (73.13)	**74.49** (72.94)	**71.91** (70.72)
GNet-R-trained	93.26	93.12	92.37	73.68	72.53	70.68

Classification accuracy analysis. Figure 3.2 displays the accuracy curves of GNet-R and GNet-T during training. Figure 3.3 shows the comparison of the training curves between the overlapping case and the non-overlapping case. Table 3.3 presents the classification accuracy of the sub-networks, the global network initialized by the sub-network transfer learning method (with and without further training), and the global network randomly initialized. Figure 3.1 displays the training time of the two training strategies. According to the results, we conclude that: for both the overlapping and non-overlapping cases, 1) the sub-network transfer learning strategy can provide a good initialization for the global network in terms of classification accuracy; 2) before further training, the global network initialized by the trained sub-networks can obtain much higher classification accuracy than all of the trained sub-networks. It is stronger than the theoretical results in Theorem 2.2, which only justifies that the intersection of the set correctly identified by the trained sub-networks can be correctly identified by the global networks initialized by the sub-networks; 3) the sub-network transfer learning strategy can accelerate the training saving about $1/4$ of the training time; 4) after further training, the global networks initialized by the proposed approach usually offers higher accuracy than those initialized randomly. Additionally, Table 3.3 and Figure 3.3 illustrate that, the accuracy of the sub-networks, as well as that of the global network initialized by the trained sub-networks before and after further training, is higher in the case of overlap than in the case of non-overlap.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

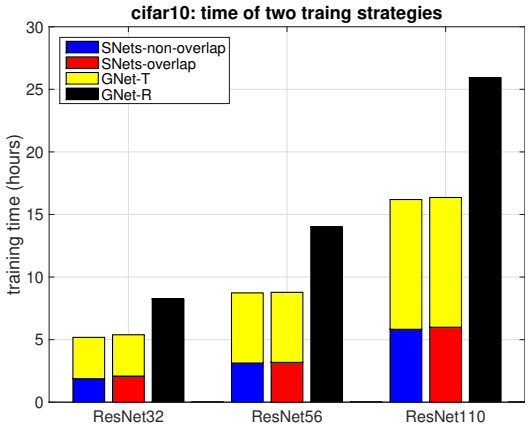

FIG. 3.1. *Training times of the two training strategies. Note that "SNets-overlap" and 'SNets-non-overlap" refer to the cases that the input images are decomposed with and without overlap, respectively.*

3.1.2. Decomposing into more partitions. For the proposed network decomposition method, there is a trade-off between the expressive power of the sub-networks and the parallel efficiency of the sub-networks training. More specifically, as the number of decompositions increases, the parallel efficiency of sub-network training improves, but the expressive ability of the sub-networks may be inadequate due to the reduced number of convolutional kernels. In addition, as the number of decompositions of input samples increases, the samples contain less information. Next, we will perform comparative experiments using different numbers of partitions.

Experimental setting. We experiment on the dataset "Food101" [1], which contains 101 food categories, with 101,000 RGB images of size 224×224. For each class, 750 training images and 250 validation images are provided. We use the well-known ResNet50 and ResNet101 [7], which end with a 1024-way FC layer, a 101-way FC layer, followed by a softmax layer. To investigate the performance of decomposing networks into more partitions, we decompose the networks and the input images into 4, 8, and 16 partitions. For input images, in the case of 4 partitions, the input images are decomposed into 4 sub-images of size 140×140 by decomposing into 2 partitions in both the width and height dimensions and then applying overlap between each pair of neighboring subdomains; for 8 partitions, the decomposition is illustrated as Figure 3.4; for 16 partitions, the input images are cropped to images of size 176×176 by cropping 24 pixels on the boundaries, which are then decomposed into 16 sub-images of size 70×70 by decomposing into 4 partitions in both the width and height dimensions with overlap. The motivation of these decompositions of input images is based on the assumption that the information at the center of the image usually contributes more to the classification task. The input data are normalized to have a mean of zero and then scaled to $[0, 1]$. In addition, the training data are augmented by randomly rotated, zoomed, horizontal flipped, and shifted in an on-the-fly way during the training. In all the experiments, the batch size is set to 128, and we apply 300 iterations in each epoch.

When comparing the classification accuracy between the two training strategies, the numbers of iterations of training in the two strategies are the same; namely, the epoch number of the global network randomly initialized (i.e., "GNet-R") is equal to the sum of the epoch number of one sub-network (i.e., "SNet") and that of the global network initialized by the composition of the trained sub-networks (i.e., "GNet-T"). More specifically, GNet-R is trained for 200 epochs, the learning rate starts from 0.001, and is divided by 10 at 120 and 180 epochs.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

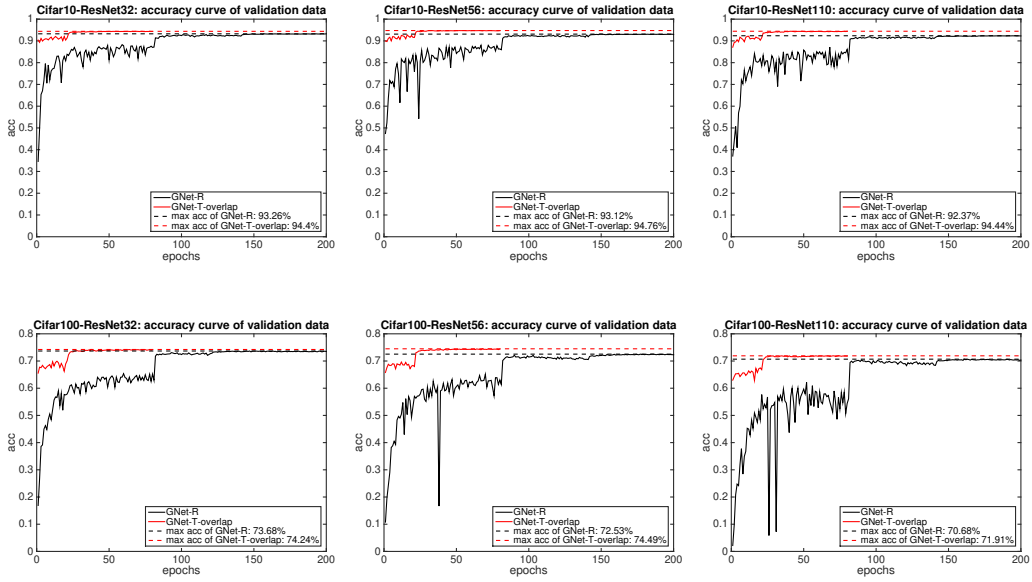

FIG. 3.2. *Classification accuracy for validation data during training for different networks and datasets Cifar10 (top) and Cifar100 (bottom). The comparison between the global network initialized by the sub-network transfer learning strategy ("GNet-T-overlap") and the global network randomly initialized ("GNet-R"). The suffix "overlap" in "GNet-T-overlap" refers to the decomposition of the input images with overlap. Note that only the case of overlap is shown since the curves of the non-overlapping case are similar to those of the overlapping case. From left to right are the networks with 32, 56, and 110 layers, respectively.*

The sub-networks are trained for 100 epochs with learning rate starting from 0.001, divided by 10 at 80 epochs. Then GNet-T is trained for 100 epochs, learning rate starting again from 0.001, but this time divided by 10 at 60 and 80 epochs. In addition, an adaptive learning rate schedule is used, in which the learning rate is divided by $10^{1/2}$ if it makes no improvement in the classification accuracy for validation data in 5 epochs.

Experimental results. Table 3.4 shows the FLOPs and the number of parameters of the global network and one sub-network, which indicates that 1) the number of parameters of the sub-network is approximately $1/K^2$ of that of the corresponding global network where K denotes the number of sub-networks that the global network is decomposed into, and 2) in the case of 4 partitions, the computation of the sub-network is approximately $1/33$ of the corresponding global network; as the number of partitions increases to 8 and 16, this ratio decreases to $1/(33 \times 8)$ and $1/(33 \times 8 \times 5)$. In addition, Table 3.4 and Figure 3.5 display the training times of the two training strategies. It indicate that, when using 4 GPUs, for the same number of iterations and batch size, the training time of K sub-networks is approximately $1/4$ of that of the global network; thus, the sub-network transfer learning strategy accelerates the training, which saves about $3/8$ of the training time; as shown in Figure 3.5. Note that in theory, the training time of the sub-networks should decrease as the partition number increases, since the computation of the sub-networks decreases exponentially as the partition number increases; but in our experiments, the parallel efficiency of 8 (or 16) sub-networks is reduced since 2 (or 4) sub-networks share one GPU. Additionally, Table 3.4 also shows the training time of using one GPU. When using one GPU, the training of the sub-network also shows an acceleration compared with the training of the global network. But this speedup is smaller than the case of using 4 GPUs.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

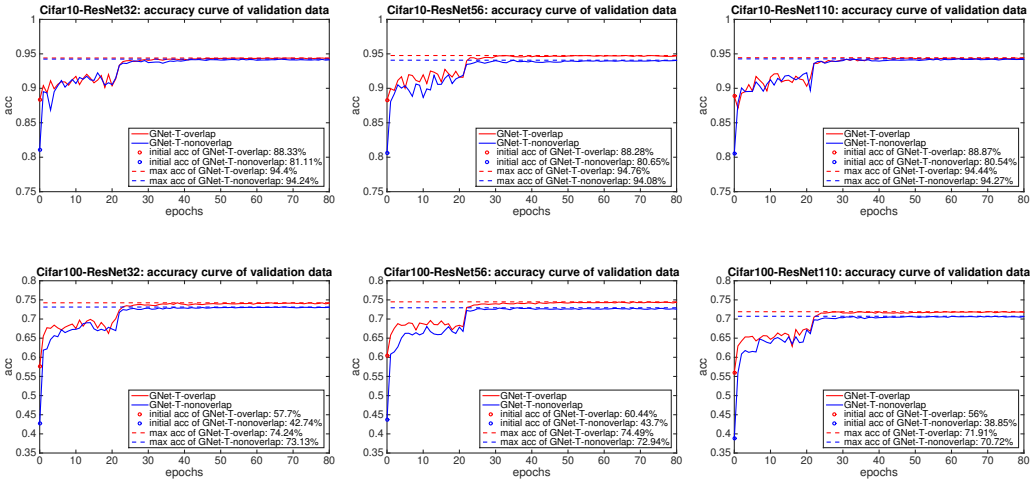

FIG. 3.3. *Classification accuracy for validation data during training for different networks and datasets Cifar10 (top) and Cifar100 (bottom). The comparison between the global network initialized by the sub-networks trained by overlapping and non-overlapping sub-images. "CNet-T-overlap" and "CNet-T-nonoverlap" refer to the cases there the input images are decomposed with and without overlap, respectively. From left to right are the networks with 32, 56, and 110 layers, respectively.*

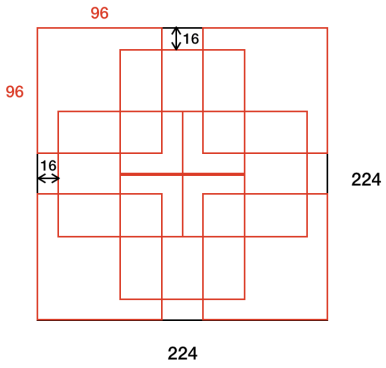

FIG. 3.4. *Illustration of the way of partitioning each image in dataset "Food101" into* 8 *sub-images.*

Figure 3.6 and Table 3.5 show comparisons of the classification accuracy between the two training strategies. According to the results, we conclude that 1) as the number of partitions increases, the initialization provided by the trained sub-networks seem to be worse, 2) in general, after further training, the accuracy of the global network initialized by the sub-network transfer learning strategy decreases as the number of partitions increases, and 3) after further training, the sub-network transfer learning strategy shows almost no loss of accuracy, except for the case when ResNet101 is decomposed into 16 partitions. These results indicate that decomposition into too many partitions reduces the quality of the initialization and also performs poorly after training.

3.1.3. Visualization of the initialization obtained by sub-network composition. Both the theoretical analysis in Section 2.3 and the experimental results in Section 3.1.1 and 3.1.2 show the effectiveness of the initialization provided by the sub-network transfer learning

ETNA
Kent State University and
Johann Radon Institute (RICAM)

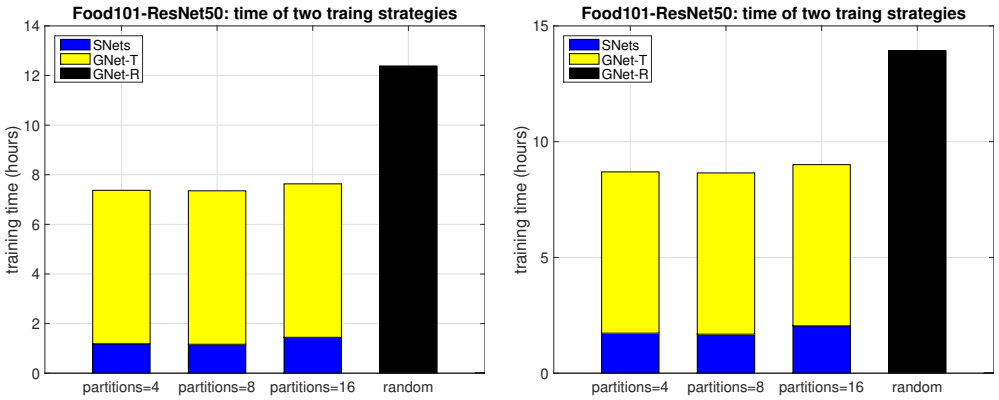

FIG. 3.5. *Training times of the two training strategies for dataset Food101 and ResNet50, ResNet101. And "4", "8", and "16" indicate that the global networks are decomposed into 4, 8, and 16 partitions. In all the cases four GPUs are used. Thus, one GPU is assigned to 1, 2, and 4 sub-network in the case of 4, 8, and 16 partitions, respectively.*

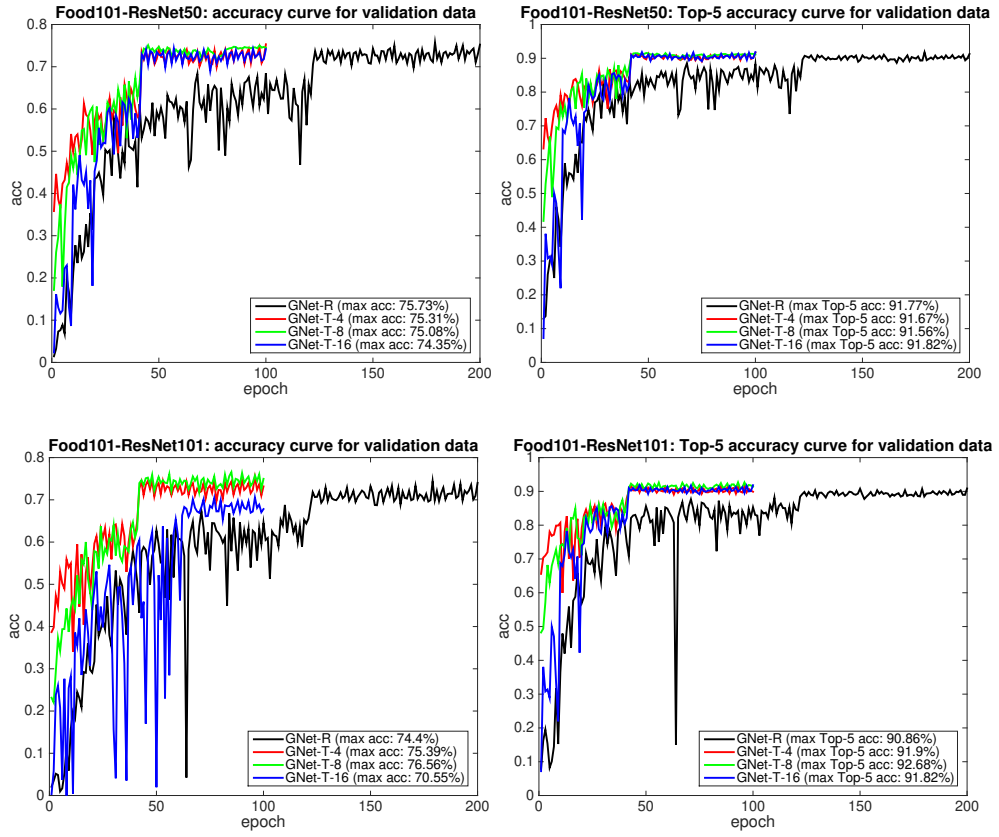

FIG. 3.6. *Classification accuracy for validation data during training for different networks and dataset Food101. Comparisons between the global network initialized by the sub-network transfer learning strategy ("GNet-T", and "4", "8", and "16" indicate that the global networks are decomposed into 4, 8, and 16 partitions.) and the global network randomly initialized ("GNet-R"). (Rows) ResNet50 and ResNet101. (Columns) Top-1 and top-5 accuracy curves.*

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 3.4
The FLOPs and the number of parameters of the global networks ("GNet") and one of their corresponding sub-networks ("SNet", and "4", "8", and "16" mean 4, 8, and 16 partitions, respectively); the training time of the global networks and 4 (or 8, 16) sub-networks for 10 epochs. The global networks and the input images are decomposed into 4, 8, 16 partitions as described in the experimental setting. We consider the cases of using four GPUs and one GPU. When using four GPUs, one GPU is assigned to 1, 2, and 4 sub-network in the case of 4, 8 and 16 partitions, respectively.

		ResNet50				ResNet101			
		GNet	SNet-4	SNet-8	SNet-16	GNet	SNet-4	SNet-8	SNet-16
# param		25.79M	1.66M	0.43M	0.12M	44.86M	2.87M	0.74M	0.20M
GFLOPs		7.79	0.23	0.028	0.0056	15.26	0.42	0.050	0.0096
training time	4 GPUs	2227 s	427 s	420 s	521 s	2507 s	623 s	607 s	736 s
of 10 epochs	1 GPU	2723 s	1347 s	987 s	1277 s	2920 s	1650 s	1533 s	2210 s

TABLE 3.5
The classification accuracy (Top-1 and Top-5) of the validation data of Food-101 by using ResNet50 and ResNet101. The comparison between the global networks that are initialized by the sub-network transfer learning strategy ("GNet-T", and "4", "8", and "16" mean 4, 8, and 16 partitions, respectively) and the randomly initialized ("GNet-R"). Suffix-trained means that the global networks are trained; otherwise, it indicates that the global networks are initialized but not further trained.

	ResNet50		ResNet101	
	Top-1(%)	Top-5(%)	Top-1(%)	Top-5(%)
GNet-T-4	**49.09**	**74.58**	**54.69**	**79.43**
GNet-T-8	25.59	51.11	26.80	54.45
GNet-T-16	13.65	37.03	10.76	31.85
GNet-T-4-trained	75.31	91.67	75.39	91.90
GNet-T-8-trained	75.08	91.67	**76.56**	**92.68**
GNet-T-16-trained	74.34	**91.82**	70.55	91.82
GNet-R-trained	**75.72**	91.77	74.40	90.86

strategy. Next, we visualize the initialization provided by the composition of the trained sub-networks. More specifically, two visualization techniques are used 1) to show the feature maps of the trained sub-networks and those of the global network initialized by the sub-networks, and 2) for both the sub-networks and the global network, to use the class activation mapping (CAM) technique [38], in which the predicted class score is mapped back to the previous convolutional layer to produce a coarse localization map highlighting the important regions in the image.

We experiment on the dataset "Dogs vs. Cats" with images of cats or dogs by using an 11-layer DCNN that consists of 10 convolutional layers (with one max-pooling for every 2 convolutional layers) and ends with a global average pooling layer and a 2-way FC layer with softmax. The DCNN is uniformly decomposed into 4 sub-networks by the proposed method. Each sample (of size 224×224) is decomposed into 4 sub-images of size 140×140 by decomposing into 2 partitions in both the width and height dimension with overlap.

For technique 1), Figure 3.7 displays some feature maps of the trained sub-networks and subdomains of the feature maps of the global network initialized by the sub-networks, where the subdomain Ω_k^l is defined as in (2.15) in Theorem 2.1, and the feature maps in the 1st, 3rd, and 5th layers are shown. It can be seen that the subdomains of the feature maps extracted by the global network are the same as the feature maps extracted by the corresponding sub-network, which is consistent with the result (1) in Theorem 2.1.

For technique 2), we use the CAM technique [38] to produce a coarse localization map highlighting the important regions in the image. For example, denote $\{X_n\}_n$ as the feature maps of the final convolutional layer, and denote w_n as the weight corresponding to class 'cat'

ETNA
Kent State University and
Johann Radon Institute (RICAM)

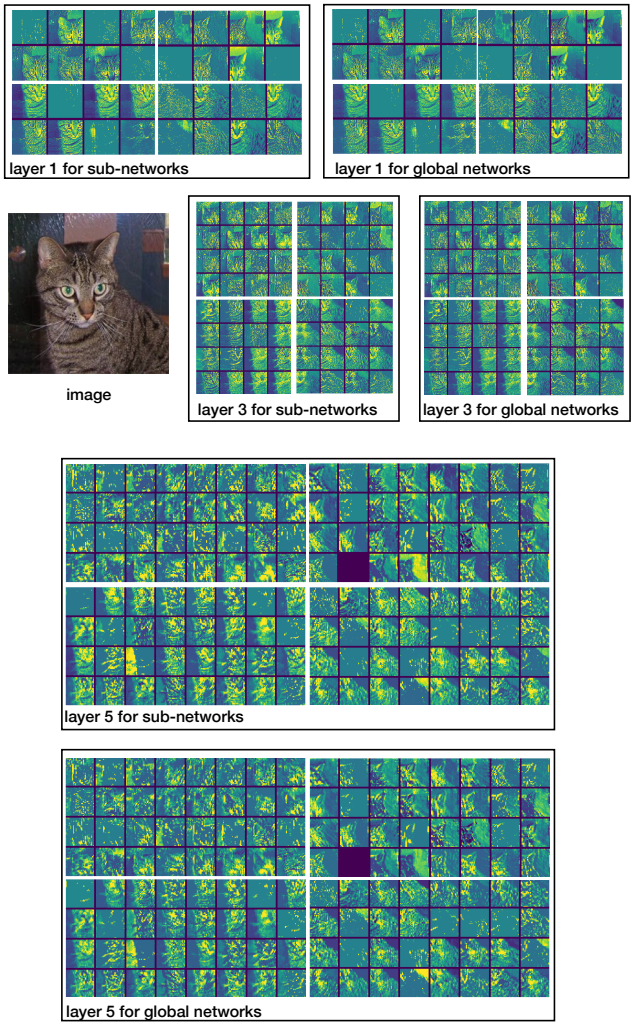

FIG. 3.7. *Some feature maps in the 1st, 3rd, and 5th layers of the trained sub-networks and subdomains of the feature maps of the global network initialized by the sub-networks, where the subdomain* Ω_k^l *is defined as in* (2.15).

for the global average pooling unit n. Then, the class activation map for class 'cat' is given by $M = \sum_n w_n X_n$, which is then upsampled to the size of the input image. Figure 3.8 displays such localization maps for predicting the ground truth category, which contains the CAM visualizations of the trained sub-networks, the global network initialized by the sub-networks with or without further training, and the global network randomly initialized with or without further training. In general, even without further training, the global network initialized by the trained sub-networks can highlight the important regions for prediction; for example, it can localize the cat regions for the images containing cats. But for the global network randomly initialized without training, its CAM visualization appears to be haphazard. Besides, after further training, both the CAM visualizations of the global network initialized by the sub-networks and that randomly initialized can locate the important regions for prediction.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

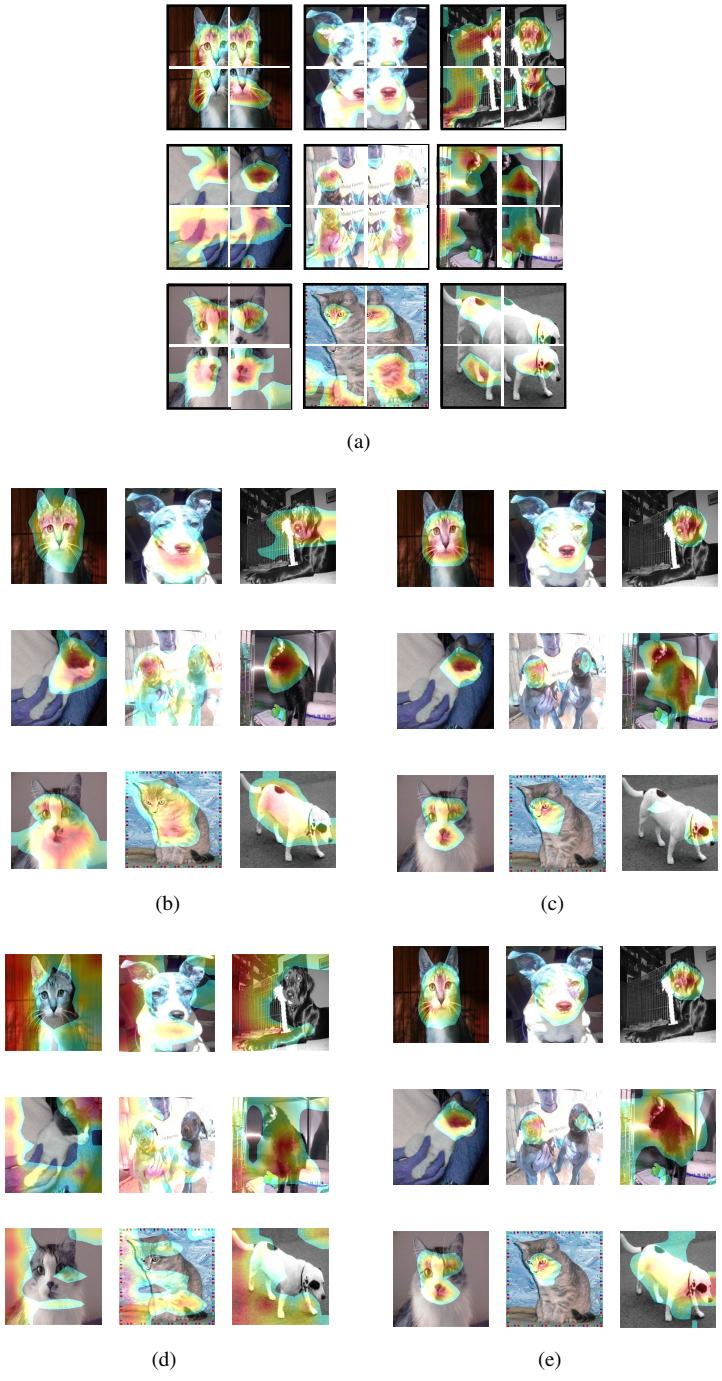

FIG. 3.8. *The CAM visualization of: the trained sub-networks (a), the global network initialized by the trained sub-networks without and with further training ((b) and (c) for without and with, respectively), and the global network randomly initialized without and with training ((d) and (e) for without and with, respectively). The red regions correspond to high scores. Even without further training, the global network initialized by the trained sub-networks can highlight the important regions for prediction.*

ETNA
Kent State University and
Johann Radon Institute (RICAM)

4. Conclusion. In this paper, following the idea of domain decomposition methods, we propose and study a deep convolutional neural network decomposition and composition technique for the purpose of paralleling the training of DCNNs. In the decomposition stage, the global network is decomposed into sub-networks that are trained completely independently with the decomposition of the samples. Then, following the idea of nonlinear preconditioning of Newton's method [24], the sub-networks are composed to initialize the global network by the sub-network transfer learning strategy. Similar to domain decomposition methods, the advantages of the proposed approaches include the applicability for parallelization of the computations and boosting the models' ability to capture fine details. The theoretical analysis and experiments indicate that the proposed approaches can provide a good initialization for DCNNs and accelerate the training. Only 2D, low resolution images are studied in this paper. The technique is expected to show more advantages for more difficult problems such as high resolution 3D images, and larger clusters of GPUs.

REFERENCES

[1] L. BOSSARD, M. GUILLAUMIN, AND L. VAN GOOL, *Food-101–mining discriminative components with random forests*, in European Conference on Computer Vision 2014, D. Fleet, T. Pajdla., B. Schiele, T. Tuytelaars, eds., Springer, Cham, 2014, pp. 446–461.

[2] X.-C. CAI AND Y. SAAD, *Overlapping domain decomposition algorithms for general sparse matrices*, Numer. Linear Algebra Appl, 3 (1996), pp. 221–237.

[3] L. CHEN, Y. ZHU, G. PAPANDREOU, F. SCHROFF, AND H. ADAM, *Encoder-decoder with atrous separable convolution for semantic image segmentation*, in European Conference on Computer Vision 2018, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds., Springer, Cham, 2018, pp. 801–818.

[4] Y. CHENG, D. WANG, P. ZHOU, AND T. ZHANG, *A survey of model compression and acceleration for deep neural networks*, e-print arXiv:1710.09282, October 2017. https://arXiv.org/abs/1710.09282

[5] J. DENG, W. DONG, R. SOCHER, L. J. LI, K. LI, AND F.-F. LI, *Imagenet: A large-scale hierarchical image database*, in IEEE Conference on Computer Vision and Pattern Recognition 2009, IEEE Conference Proceedings, Los Alamitos, 2009, pp. 248–255.

[6] P. GOYAL, P. DOLLÁR, R. GIRSHICK, P. NOORDHUIS, L. WESOLOWSKI, A. KYROLA, A. TULLOCH, Y. JIA, AND K. HE, *Accurate, large minibatch SGD: Training imagenet in 1 hour*, e-print arXiv:1706.02677, June 2017. https://arXiv.org/abs/1706.02677

[7] 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, IEEE Conference Proceedings, Los Alamitos, 2016, pp. 770–778.

[8] G. HUANG, S. LIU, L. V. DER MAATEN, AND K. Q. WEINBERGER, *Condensenet: An efficient densenet using learned group convolutions*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018, IEEE Conference Proceedings, Los Alamitos, 2018, pp. 2752–2761.

[9] F. N. IANDOLA, M. W. MOSKEWICZ, K. ASHRAF, AND K. KEUTZER, *Firecaffe: near-linear acceleration of deep neural network training on compute clusters*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016, IEEE Conference Proceedings, Los Alamitos, 2016, pp. 2592–2600.

[10] S. IOFFE AND C. SZEGEDY, *Batch normalization: accelerating deep network training by reducing internal covariate shift*, e-print arXiv:1502.03167, February 2015. https://arXiv.org/abs/1502.03167.

[11] S. KIRANYAZ, O. AVCI, O. ABDELJABER, T. INCE, M. GABBOU, AND D. J. INMAN, *1D convolutional neural networks and applications: a survey*, Mech. Syst. Signal Process., 151 (2021), Art. 107398 (21 pages).

[12] A. KRIZHEVSKY, *Learning multiple layers of features from tiny images*, Technical report, University of Toronto, Toronto, 2009. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[13] ———, *One weird trick for parallelizing convolutional neural networks*, e-print arXiv:1404.5997, April 2014. https://arXiv.org/abs/1404.5997

[14] A. KRIZHEVSKY, I. SUTSKEVER, AND G. E. HINTON, *Imagenet classification with deep convolutional neural networks*, in Advances in Neural Information Processing Systems 25 (NIPS2012), F. Pereira, C. J. C. Burges, L. Bottou, and K.Q. Weinberger, eds., NIPS Proceedings, 2012, pp. 1097–1105.

[15] S. LATHUILIÈRE, P. MESEJO, X. ALAMEDA-PINEDA, AND R. HORAUD, *A comprehensive analysis of deep regression*, IEEE Trans. Pattern Anal. Mach. Intell., 42 (2019), pp. 2065–2081.

[16] Y. LECUN, Y. BENGIO, AND G. E. HINTON, *Deep learning*, Nature, 521 (2015), pp. 436–444.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proc. IEEE, 86 (1998), pp. 2278–2324.
[18] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, *Pruning filters for efficient convnets*, e-print arXiv:1608.08710, March 2017. https://arXiv.org/abs/1608.08710
[19] Z.-J. Liao, R. Chen, Z. Yan, and X.-C. Cai, *A parallel implicit domain decomposition algorithm for the large eddy simulation of incompressible turbulent flows on 3d unstructured meshes*, Internat. J. Numer. Methods Fluids, 89 (2019), pp. 343–361.
[20] M. Lin, Q. Chen, and S. Yan, *Network in network*, e-print arXiv:1312.4400, March 2014. https://arXiv.org/abs/1312.4400
[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, *Ssd: Single shot multibox detector*, in European Conference on Computer Vision 2016, B. Leibe, J. Matas, N. Sebe, M. Welling, eds., Springer, Cham, 2016, pp. 21–37.
[22] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, *Learning efficient convolutional networks through network slimming*, in Proceedings of the IEEE International Conference on Computer Vision 2017, IEEE Conference Proceedings, Los Alamitos, 2017, pp. 2755–2763.
[23] J. Long, E. Shelhamer, and T. Darrell, *Fully convolutional networks for semantic segmentation*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015, IEEE Conference Proceedings, Los Alamitos, 2015, pp. 3431–3440.
[24] L. Luo, W. Shiu, R. Chen, and X.-C. Cai, *A nonlinear elimination preconditioned inexact Newton method for blood flow problems in human artery with stenosis*, J. Comput. Phys., 399 (2019), Art. 108926 (20 pages).
[25] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, *Importance estimation for neural network pruning*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2019, IEEE Conference Proceedings, Los Alamitos, 2019, pp. 11256–11264.
[26] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, *Pruning convolutional neural networks for resource efficient inference*, e-print arXiv:1611.06440, June 2017. https://arXiv.org/abs/1611.06440
[27] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, in Advances in Neural Information Processing Systems 28 (NIPS2015), C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds., NIPS Proceedings, 2015, pp. 91–99.
[28] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, in Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015), N. Navab, J. Hornegger, W. Wells, and A. Frangi, eds., Lecture Notes in Computer Science 9351, Springer, Cham, 2015, pp. 234–241.
[29] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, e-print arXiv:1409.1556, September 2014. https://arXiv.org/abs/1409.1556
[30] B. Smith, P. Bjorstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, 2004.
[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, *Going deeper with convolutions*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015, IEEE Conference Proceedings, Los Alamitos, pp. 1–9.
[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, *Rethinking the inception architecture for computer vision*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016, IEEE Conference Proceedings, Los Alamitos, 2016, pp. 2818–2826.
[33] R. Tang, W. Wang, Z. Tu, and J. Lin, *An experimental analysis of the power consumption of convolutional neural networks for keyword spotting*, in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE Conference Proceedings, Los Alamitos, 2018, pp. 5479–5483.
[34] A. Toselli and O. B. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer, Berlin, 2005.
[35] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, *Learning structured sparsity in deep neural networks*, in Advances in Neural Information Processing Systems 29 (NIPS2016), D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett, eds., NIPS Proceedings, 2015, pp. 2074–2082.
[36] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, *Imagenet training in minutes*, in ICPP 2018: Proceedings of the 47th International Conference on Parallel Processing, ACM, New York, 2018, pp. 1–10.
[37] R. Yu, A. Li, C. Chen, J. Lai, V. I. Morariu, Han X, M. Gao, C. Lin, and L. S. Davis, *NISP: Pruning networks using neuron importance score propagation*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018, IEEE Conference Proceedings, Los Alamitos, 2018, pp. 9194–9203.
[38] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, *Learning deep features for discriminative localization*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016, IEEE Conference Proceedings, Los Alamitos, 2016, pp. 2921–2929.