Decomposition and Preconditioning of Deep Convolutional Neural Networks for Training Acceleration

Linyan Gu, Wei Zhang, Jia Liu, and Xiao-Chuan Cai

1 Introduction

Deep convolutional neural networks (DCNNs) [7] have brought significant improvements to the field of computer vision for a wide range of problems [3, 7]. Larger models and larger datasets have led to breakthroughs in accuracy; however, it results in much longer training time and memory intensity, which negatively impact the development of CNNs [1, 9]. There are two ways to parallelize training: model parallelism and data parallelism [1]. Model parallelism partitions the network into pieces, and different processors train different pieces. In model parallelism, frequent communication between different processors is needed since the calculation of the next layer usually requires the outputs of the previous layer. In data parallelism, the dataset is partitioned into parts stored in each processor which has a local copy of the network with its parameters. However, scaling the training to a large number of processors means an increase in the batch size, which results in poor generalization. New training methods are developed to avoid this problem [1, 9].

In this paper, we propose a method to parallelize the training of DCNNs by decomposing and preconditioning DCNNs motivated by the idea of domain decomposition methods [8]. Domain decomposition methods are a family of highly parallel methods for solving partial differential equations on large scale computers, which is based on the divide and conquer philosophy for solving a problem defined on a global domain by iteratively solving subproblems defined on smaller subdomains [8]. The advantages of domain decomposition methods consist of the straightforward

Linyan Gu, Wei Zhang, Jia Liu
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, 518055, China, e-mail: ly.gu@siat.ac.cn,wei.zhang@siat.ac.cn,jia.liu@siat.ac.cn

Xiao-Chuan Cai
Faculty of Science and Technology, University of Macau, Avenida da Universidade, Taipa, Macao, China, e-mail: xccai@um.edu.mo

applicability for parallel computing [4] and the localized treatment for the specificity of subdomain problems [8].

First, motivated by the domain decomposition methods, a DCNN (also called a global network) is decomposed into sub-networks by partitioning the width of the network while keeping the depth constant. All the sub-networks are individually trained, in parallel without any interprocessor communication, with the corresponding decomposition of the input samples. Then, following the idea of nonlinear preconditioning of Newton's method [5] that replaces the standard initial guess by an initial guess that satisfies some of the constraints locally, we propose a sub-network transfer learning strategy in which the weights of the trained sub-networks are composed to initialize the global network, which is then further trained. There are some differences between the proposed preconditioning of DCNNs and the standard nonlinear domain decomposition preconditioners. For example, we use the stochastic gradient descent (SGD) method instead of Newton's method. Besides, the nonlinear preconditioner of DCNNs (i.e., compositing the sub-networks to initialize the global network) is applied only once, while the nonlinear preconditioner is applied in every iteration (or some iterations) in the nonlinear preconditioning of Newton's method.

The rest of this paper is organized as follows. Section 2 proposes a new method to parallelize the training of DCNNs by decomposing and preconditioning DCNNs. Section 3 provides some experiments, followed by our conclusions in Section 4. Additionally, we have submitted some parts of this work to a special issue of Electronic Transaction on Numerical Analysis [2], where more details, additional theoretical discussions and more experimental results are included.

2 Proposed approaches

In this section, we propose and study a new method to parallelize the training of DCNNs by decomposing and preconditioning DCNNs. We consider a DCNN for classification consisting of some convolutional layers and some fully connected (FC) layers, followed by a classification module which is usually a softmax layer.

Notations. Denote a L-layer DCNN as $F(\boldsymbol{x};\Theta)$ with input $\boldsymbol{x}$ and the set of parameters Θ. The output of each layer is called feature map and is a 3D tensors, where the third dimension of the tensors is the number of independent maps, and the first and the second are the height and the width, respectively. The kernel of the l-th layer is a 4D tensor and can be denoted by $\boldsymbol{w}^l \in \mathbb{R}^{t_1^l \times t_2^l \times c_{\text{in}}^l \times c_{\text{out}}^l}$, where c_{in}^l and c_{out}^l are the number of input and output channels, respectively, and t_1^l, t_2^l are the kernel widths. A FC layer can be regarded as a special case of a convolutional layer.

Assume $\boldsymbol{x} \in \mathbb{R}^{H\times W\times D}$ is a 3D tensor with element $x_{i,j,k}$ where $(i,j,k) \in \Omega$, $\Omega = [H]\times[W]\times[D]$ with $[H] = \{1,\cdots,H\}$. Given a Cartesian product $\tilde{\Omega} \subset \Omega$, we call $\tilde{\boldsymbol{x}} = \mathcal{D}_{\tilde{\Omega}}(\boldsymbol{x})$ a subdomain of $\boldsymbol{x}$ with element $\tilde{x}_{\tilde{i},\tilde{j},\tilde{k}} = x_{i,j,k}$ for all $(i,j,k) \in \tilde{\Omega}$, where the elements of $\mathcal{D}_{\tilde{\Omega}}(\boldsymbol{x})$ remain order-preserving (cf. [2]). Given a set of Cartesian products $\{\Omega_k\}_{k=1}^K$ satisfying

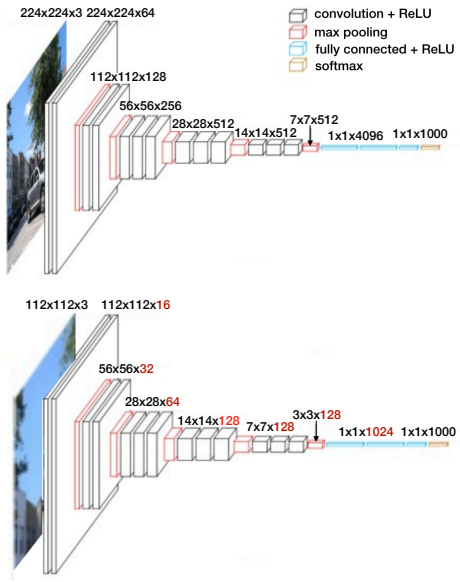

Fig. 1: Illustration of the decomposition of DCNN. The global network and input is uniformly decomposed into 4 partitions. The architecture of VGG16 (top) and one of sub-networks (bottom); cf. [2].

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

we call $\{\tilde{\boldsymbol{x}}_k = \mathcal{D}_{\Omega_k}(\boldsymbol{x}) \mid k \in [K]\}$ a complete decomposition of $\boldsymbol{x}$. Besides, given 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 $\tilde{\Omega}$; see [2] for more formal details.

2.1 Decomposing a DCNN into sub-networks

We consider a global network for a classification task with samples $X = \{\boldsymbol{x}_i\}_i$ and their corresponding labels. Given a set of Cartesian products $\{\Omega_k\}_{k=1}^K$, the samples are decomposed into K subdomain denoted by $X_k = \{\mathcal{D}_{\Omega_k}(\boldsymbol{x}_i)\}_i$ for $k \in [K]$. For a natural RGB image (i.e., $D = 3$), we decompose $\boldsymbol{x}_i$ in the first and second dimensions but not the third dimension. Correspondingly, a global network is decomposed into K sub-networks by partitioning the **width** (i.e., along the channel dimension) of the network while keeping the depth constant; see more formal details in [2]. Then, the samples of each subdomain and their ground truth are used to train the corresponding sub-network. The trainings of sub-networks can be performed completely independently on parallel computers. Compared with existing distributed

training methods for DCNNs, the parallel training of sub-networks is rather related to model parallelism than to data parallelism. However, the sub-networks are trained completely independently and the communication between different sub-networks in the proposed approach only occurs in the initialization of the global network, which is different from the current model parallelism that suffers from excessive inter-GPU communication since the model part trained by one processor usually requires output from a model part trained by another processor.

Fig. 1 shows the decomposition of VGG16 [7], where the VGG16 and the input samples are uniformly decomposed into $K = 4$ partitions; uniformly means that the decomposition of inputs is a complete decomposition, and the channel number and the input are partitioned uniformly.

The number of floating point operations (FLOPs) [6] is used to estimate the computational complexity of a network, in which each multiplication or addition is counted as one FLOP. Assume that the global network and the inputs are uniformly decomposed into $K = n^2$ partitions. Generally, the FLOPs of each sub-network is approximately $1/K^3$ of that of the global network, since the convolutional layers contain the vast majority of computations; see [2] for more details.

2.2 Preconditioning of DCNNs

We propose an algorithm for composing the trained sub-networks to initialize the global network, which we call the sub-network transfer learning strategy, and then the global network is further trained. In a reverse process of the decomposition, the weights of the sub-networks are composed along the channel dimension but with additional connections between the sub-networks initialized to zero. Taking VGG16 as an example, Fig. 2 shows how to compose 4 sub-networks into one global network. More formally, denoting the weights in l-th layer of the global network and the k-th sub-networks by $\boldsymbol{w}^l$ and $\{\boldsymbol{w}_k^l\}_k$, respectively, $\boldsymbol{w}^l$ is initialized as follows (cf. [2]):

- For the first layer,

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

- For the first FC layer,

$$\begin{aligned} \mathcal{D}_{\Omega_k'}(\boldsymbol{w}^l) &= \boldsymbol{w}_k^l, \text{ for } k \in [K], \mathcal{D}_{\Omega''}(\boldsymbol{w}^l) = \boldsymbol{0}, \\ \Omega_k' &= \big(\mathcal{G}_{F,l-1}(\Omega_k) \cap ([t_1^l] \times [t_2^l] \times \{1 + \sum_{i=1}^{k-1} c_{i,\text{in}}^l : \sum_{i=1}^{k} c_{i,\text{in}}^l\})\big) \times \{1 + \sum_{i=1}^{k-1} c_{i,\text{out}}^l : \sum_{i=1}^{k} c_{i,\text{out}}^l\}, \\ \Omega'' &= \big([t_1^l] \times [t_2^l] \times [c_{\text{in}}^l] \times [c_{\text{out}}^l]\big) \setminus \cup_{k=1}^{K} \Omega_k'. \end{aligned} \tag{2}$$

- For the last FC layer,

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

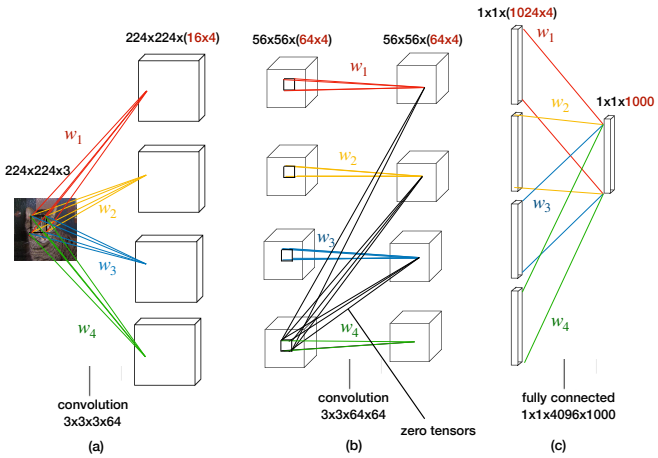

Fig. 2: Illustration of preconditioning the global network by composing 4 sub-networks. $\boldsymbol{w}_k$ denotes the weights of the k-th sub-network (using the same notation for different layers for simplicity). (a) The first convolutional layer; (b) One of the intermediate convolutional layers. Note that some of the connections with zero weights are omitted for simplicity; (c) The last FC layer; cf. [2].

$$\Omega_k^{'} = [t_1^L] \times [t_2^L] \times \{1 + \sum_{i=1}^{k-1} c_{i,\text{in}}^L : \sum_{i=1}^{k} c_{i,\text{in}}^L\} \times [C]. \tag{3}$$

- For other convolutional layers and other FC layers,

$$\begin{gathered}\mathcal{D}_{\Omega_k^{'}}(\boldsymbol{w}^l) = \boldsymbol{w}_k^l, \text{ for } k \in [K], \mathcal{D}_{\Omega^{''}}(\boldsymbol{w}^l) = \boldsymbol{0}, \\ \Omega_k^{'} = [t_1^l] \times [t_2^l] \times \{1 + \sum_{i=1}^{k-1} c_{i,\text{in}}^l : \sum_{i=1}^{k} c_{i,\text{in}}^l\} \times \{1 + \sum_{i=1}^{k-1} c_{i,\text{out}}^l : \sum_{i=1}^{k} c_{i,\text{out}}^l\}, \\ \Omega^{''} = ([t_1^l] \times [t_2^l] \times [c_{\text{in}}^l] \times [c_{\text{out}}^l]) \setminus \cup_{k=1}^{K} \Omega_k^{'}. \end{gathered} \tag{4}$$

3 Experiments

In this section, some experiments on image classification tasks are carried out to evaluate the proposed approach by observing the training time and the classification accuracy. The experiments are carried out using the TensorFlow library on a workstation with 4 NVIDIA Tesla V100 32G GPUs. We compare the performances 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-networks 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. The sub-networks are trained in parallel by a multiprocessing strategy, with GPUs uniformly assigned to sub-

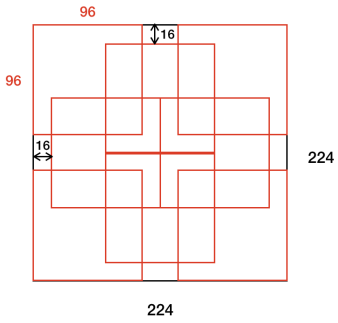

Fig. 3: Illustration of the way of partitioning each image into 8 sub-images; cf. [2].

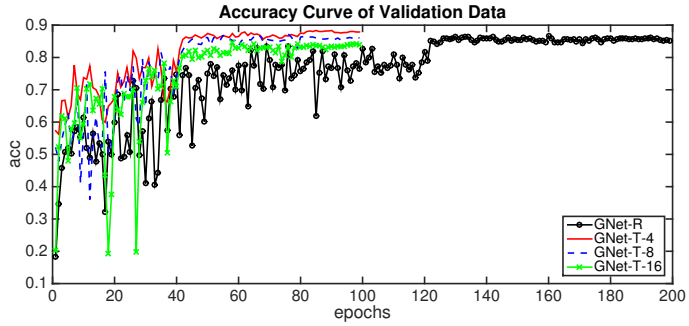

Fig. 4: Classification accuracy curves for validation data during training with varying partition numbers. Comparisons between the two training strategies (i.e., "GNet-R" and "GNet-T").

networks. In addition, the numbers of training iterations in the two strategies are the same. For the strategy 1), the GNet-R is trained for 200 epochs. For the strategy 2), the SNets are trained for 100 epochs and then the GNet-T is trained for 100 epochs.

The experiments are carried out on the dataset which contains 4323 images of flowers in 5 categories[1]. The dataset is split into training, validation and testing sets in the ratio of about 70:15:15, and the images are all resized to 224×224. We use a residual network of 18 layers in [3]. Additionally, the network and the input images are decomposed into 4, 8, and 16 partitions. For 4 (or 16) partitions, the input images are cropped 24 pixels on the boundaries, which are then decomposed into 4 (or 16) sub-images of size 140×140 (or 70×70) by decomposing into 2 (or 4) partitions in both the width and height dimensions and then applying overlap between each pair of neighbouring subdomains; for 8 partitions, the decomposition is illustrated as Fig. 3.

Table 1 shows the FLOPs and the number of parameters of the global network and one sub-network, and the training times of the two training strategies, which

[1] https://www.kaggle.com/alxmamaev/flowers-recognition.

Table 1: The FLOPs and the number of parameters of the global network ("GNet") and one sub-network ("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. "M" means 10^6.

	GNet	SNet-4	SNet-8	SNet-16
# param	17.22M	1.08M	0.27M	0.07M
FLOPs	5015.56M	154.57M	19.69M	4.05M
training time	289 s	137 s	101 s	82 s

Table 2: The comparison of the classification accuracy of the testing data between the global networks of the two training strategies (i.e., "GNet-T" and "GNet-R") with varying partition numbers. "Initialized" means that the global networks are initialized by the sub-network transfer learning strategy (i.e., "GNet-T") or randomly initialized (i.e., "GNet-R") without being further trained, and "Trained" means that the global networks are trained.

Initialized (%)				Trained (%)			
GNet-R	GNet-T-4	GNet-T-8	GNet-T-16	GNet-R	GNet-T-4	GNet-T-8	GNet-T-16
17.39	77.73	57.45	49.46	82.80	83.56	82.49	79.26

indicates that 1) the number of parameters of the sub-network is approximately $1/K^2$ of that of the corresponding global network, 2) for 4 partitions, the computation of the sub-network is approximately $1/2^5$ of the corresponding global network; for 8 and 16, this ratio decreases to $1/2^8$ and $1/2^{10}$, and 3) for the same number of iterations, the training time of K sub-networks is less than $1/2$ of that of the global network; thus, the sub-network transfer learning strategy saves more than $1/4$ of the training time.

Fig. 4 and Table 2 show the comparisons of the classification accuracy between the two training strategies, which shows that 1) in general, as the number of partitions increases, the initialization seem to be worse and the accuracy of GNet-T after further training decreases, and 2) after further training, the sub-network transfer learning strategy shows almost no loss of accuracy, except for the case of 16 partitions. These results indicate that a decomposition into too many partitions may reduce the quality of the initialization and also perform poorly after further training.

4 Conclusion

In this paper, inspired by the idea of domain decomposition methods and nonlinear preconditioning, we propose and study a new method of decomposing and preconditioning DCNNs for the purpose of parallelizing the training of DCNNs. The global network is firstly decomposed into sub-networks that are trained independently without any interprocessor communication, which are then recomposed to initialize the global network via the transfer learning strategy. The experimental results show that the proposed approach 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.

References

1. 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. *arXiv preprint arXiv:1706.02677*, 2017.
2. L. Gu, W. Zhang, J. Liu, and X.-C. Cai. Decomposition and composition of deep convolutional neural networks and training acceleration via sub-network transfer learning. *submitted to ETNA*, 2020.
3. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.
4. 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. *Int. J. Numer. Methods Fluids*, 89(9):343–361, 2019.
5. 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.*, 399:108926, 2019.
6. P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. In *ICLR*, 2017.
7. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
8. A. Toselli and O. B. Widlund. *Domain Decomposition Methods-Algorithms and Theory*. Springer, 2005.
9. Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer. Imagenet training in minutes. In *ICPP*, pages 1–10, 2018.