Progressive Blockwise Knowledge Distillation for Neural Network Acceleration

Hui Wang[1,*], **Hanbin Zhao**[1,*], **Xi Li**[1,†], **Xu Tan**[2]
[1] Zhejiang University, Hangzhou, China
[2] 2012 Lab, Huawei Technologies, Hangzhou, China
{wanghui_17, zhaohanbin, xilizju, xutan}@zju.edu.cn, tanxu2@huawei.com

Abstract

As an important and challenging problem in machine learning and computer vision, neural network acceleration essentially aims to enhance the computational efficiency without sacrificing the model accuracy too much. In this paper, we propose a progressive blockwise learning scheme for teacher-student model distillation at the subnetwork block level. The proposed scheme is able to distill the knowledge of the entire teacher network by locally extracting the knowledge of each block in terms of progressive blockwise function approximation. Furthermore, we propose a structure design criterion for the student subnetwork block, which is able to effectively preserve the original receptive field from the teacher network. Experimental results demonstrate the effectiveness of the proposed scheme against the state-of-the-art approaches.

1 Introduction

Recent years have witnessed a great development of deep convolutional neural networks (DCNNs) and their various applications [Krizhevsky *et al.*, 2012; Simonyan and Zisserman, 2015; He *et al.*, 2016; Szegedy *et al.*, 2017]. Due to the resource limit of real world devices, DCNN compression and acceleration have emerged as a crucial and challenging problem in practice. Typically, the problem is resolved from the following four perspectives: 1) quantization and binarization [Hubara *et al.*, 2016; Rastegari *et al.*, 2016; Wu *et al.*, 2016; Zhou *et al.*, 2017]; 2) parameter pruning and sharing [Courbariaux *et al.*, 2015; Hu *et al.*, 2016; Li *et al.*, 2017a; Luo *et al.*, 2017; Molchanov *et al.*, 2017; Li *et al.*, 2017c]; 3) matrix factorization [Tai *et al.*, 2016; Lin *et al.*, 2016; Jaderberg *et al.*, 2014; Sainath *et al.*, 2013]; and 4) model distillation [Bucila *et al.*, 2006; Ba and Caruana, 2014; Hinton *et al.*, 2014; Romero *et al.*, 2015; Li *et al.*, 2017b]. In principle, the first three points focus on how to carry out an efficient network inference process using a variety of computational acceleration techniques with low memory usage. In contrast, the last one aims at distilling the original network model into a low-complexity network model in terms of the teacher-student learning strategy. Without sacrificing too much accuracy, the low-complexity network model naturally possesses the properties of high computational efficiency and low memory usage. However, the effectiveness of model distillation is often challenged in the aspects of teacher-student network optimization and student network structure design. Therefore, we mainly focus on constructing an effective network learning strategy with a structure-preserving criterion for model distillation in this paper.

More specifically, the process of model distillation usually involves two components: a teacher network with a complicated network structure as well as a simple student network. In essence, the process seeks for a feasible student network to mimic the output of the teacher network. Usually, conventional approaches (e.g., Knowledge Distillation [Hinton *et al.*, 2014]) rely on a non-convex joint network optimization strategy that converts the teacher network into the desired student network in a one-pass fashion. Implemented in a huge search space of the student network function with a wide variety of network configurations, the aforementioned non-convex joint optimization process is usually intractable and unstable in practice. Following the work [Hinton *et al.*, 2014], Nowak and Corso [Nowak and Corso, 2018] make an attempt to compress a subnetwork block of the teacher network into a student subnetwork block and then design different methods for initialization and training. Moreover, their design criterion for the student subnetwork block is to simply retain a convolution layer and directly remove the other two convolution layers from the teacher subnetwork block. Such a blockwise distillation strategy is simple and easy to optimize, but incapable of effectively modeling the sequential dependency relationships between layer-specific subnetwork blocks. In addition, the student subnetwork block design criterion is also incapable of well preserving the receptive field information on feature extraction.

Motivated by the observations above, we propose a blockwise learning scheme for progressive model distillation. Specifically, the proposed learning scheme converts a sequence of teacher subnetwork blocks into a sequence of student subnetwork blocks after progressive blockwise optimization. Additionally, we propose a structure-preserving criterion for student subnetwork design. This criterion allows us to transform the teacher subnetwork blocks into the student

[*] Indicates equal contribution
[†] Corresponding author

subnetwork blocks without changing the receptive field.

As a result, the proposed progressive learning scheme aims at distilling the knowledge of the entire teacher network by locally extracting the knowledge of each block in a progressive learning manner with the structure-preserving criterion of feature extraction. Therefore, the proposed scheme creates a novel network acceleration strategy in terms of progressive blockwise learning, and has the following advantages: 1) structure-preserving for each subnetwork block; 2) easy to implement for progressive blockwise optimization; 3) fast stagewise convergence; 4) flexible compatability with existing learning modules; and 5) good balance with high accuracy and competitive FLOPs reduction.

As a result, the main contributions of this work are two-fold:

- We propose a progressive blockwise learning scheme for model distillation, which is innovative in the area of neural network acceleration. The proposed learning scheme naturally converts the problem of network acceleration into that of progressive blockwise function approximation.
- We propose a structure design criterion for the student subnetwork block, which is able to effectively preserve the original receptive field from the teacher network.

2 Our Approach

2.1 Problem Definition

To better understand our representations, we provide the detailed explanations of the main notations and symbols used throughout this paper as shown in Tab. 1.

A neural network is mainly comprised of the convolution layers, the pooling layers, and the fully connected layers. The subnetwork between two adjacent pooling layers is defined as a subnetwork block.

Let a complicated network T be the teacher network, which is composed of N subnetwork blocks:

$$T = c \circ t_N \circ t_{N-1} \circ \cdots \circ t_1 \tag{1}$$

where t_i ($i \in \{1, 2, \ldots, N\}$) is the mapping function of the i-th block in the sequence and c is the mapping function of the classifier. To simplify the representation of the network, we shorten it as:

$$\prod_{i=1}^{N} \circ t_i = t_N \circ t_{N-1} \circ \cdots \circ t_1 \tag{2}$$

Therefore, T is rewritten as:

$$T = c \circ \prod\nolimits_{i=1}^{N} \circ t_i \tag{3}$$

The parameters of the teacher network are denoted as:

$$W_T = \left\{W_c, W_{t_N}, W_{t_{N-1}}, \ldots, W_{t_1}\right\} \tag{4}$$

where W_c and W_{t_i} ($i \in \{1, 2, \ldots, N\}$) are the parameters of c and t_i.

Our objective is to design a student network with high computational efficiency and low memory usage, and to learn the corresponding optimal parameters. The student network composed of N student subnetwork blocks can be written as:

Notation	Definition
T	The function that represents the initial teacher network
S	The function that represents the final student network
A^k	The auxiliary function that represents the intermediate network
s_i	The mapping function of the i-th block in S
t_i	The mapping function of the i-th block in T
c	The mapping function of the classifier
$\prod_{i=1}^{N} \circ$	A symbol to simplify the network representation
$W_T/W_S/W_{A^k}$	The parameters of $T/S/A^k$
$W_{t_i}/W_{s_i}/W_c$	The parameters of $t_i/s_i/c$

Table 1: Main notations and symbols used throughout the paper.

$$S = c \circ \prod\nolimits_{i=1}^{N} \circ s_i \tag{5}$$

where s_i denotes a student subnetwork block. The corresponding optimal parameters of the student network are denoted as:

$$W_S = \{W_c, W_{s_N}, W_{s_{N-1}}, \ldots, W_{s_1}\} \tag{6}$$

In essence, the problem is to design N student subnetwork blocks sequence S and optimize the corresponding parameters W_S using the prior knowledge of N teacher subnetwork blocks sequence T:

$$c \circ \prod\nolimits_{i=1}^{N} \circ t_i(x; W_T) \xrightarrow[\text{design } S]{\text{optimize } W_S} c \circ \prod\nolimits_{i=1}^{N} \circ s_i(x; W_S) \tag{7}$$

The main challenges to solve this problem are: 1) The joint optimization of the student network function with a wide variety of parameters is usually intractable and unstable in practice; 2) Designing a feasible student network structure from scratch is difficult. In Sectiom 2.2 and Section 2.3, we propose our solutions to these two challenges.

2.2 Progressive Blockwise Learning

To reduce the optimization difficulty described in Eq. (7), we propose a progressive blockwise learning scheme. As shown in Fig. 1, our blockwise learning scheme learns the sequence of student subnetwork blocks by N block learning stages, and only optimizes one block at each learning stage while keeping the other blocks fixed.

To better introduce our blockwise scheme, we use the auxiliary function A^k ($k \in \{0, 1, \ldots, N\}$) to represent our intermediate network at the k-th block learning stage:

$$A^k = c \circ (\prod_{i=k+1}^{N} \circ t_i) \circ (\prod_{j=1}^{k} \circ s_j) \tag{8}$$

where s_j is the optimized student network block and t_i is the teacher network block. The parameters of A^k are denoted as below:

$$W_{A^k} = \left\{W_c, W_{t_N}, \ldots, W_{t_{k+1}}, W_{s_k}, W_{s_{k-1}}, \ldots, W_{s_1}\right\} \tag{9}$$

As can be noted from the description of A^k, A^0 is the teacher network T and A^N is the optimized network S. Hence, the problem defined in Eq. (7) can be solved as below:

$$A^0 \xrightarrow{stage\ 1} A^1 \xrightarrow{stage\ 2} A^2 \xrightarrow{\cdots} A^{k-1} \xrightarrow{stage\ k} A^k \xrightarrow{\cdots} A^N \tag{10}$$

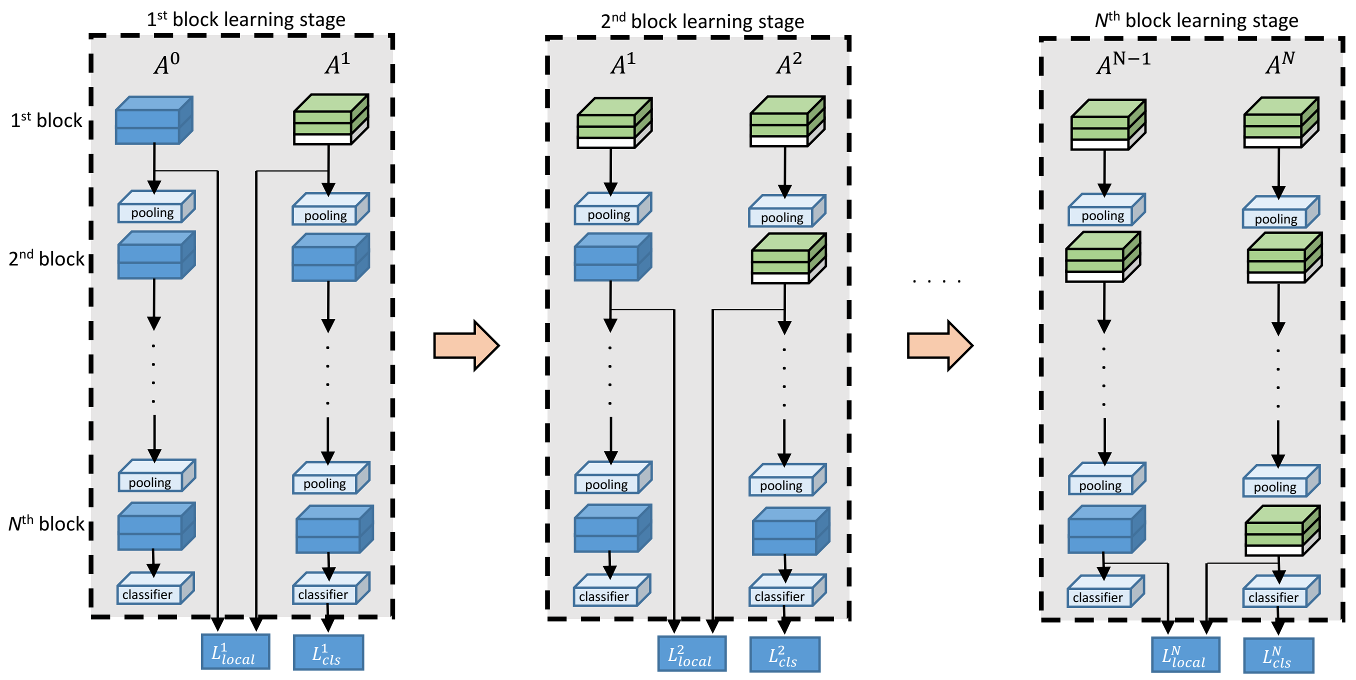

Figure 1: Illustration of our progressive blockwise learning scheme by the Bottom-Up optimization order. We use the auxiliary function A^k ($k \in \{0, 1, \dots, N\}$) to represent our intermediate network at each block learning stage (A^0 is the initial teacher network, A^N is the final student network). At the k-th learning stage, we build the auxiliary function A^k according to A^{k-1} and learn the optimal parameters of the k-th block by minimizing the loss. The loss is the weighted sum of L^k_{local} and L^k_{cls}.

More specifically, we use the teacher-student learning strategy at each block learning stage. At the block learning stage 1, we consider A^0/A^1 as the teacher/student network, and then learn A^1 from A^0. Similarly, we learn A^2 from A^1 at the block learning stage 2. By analogy, using such a progressive way, we will eventually solve the problem in Eq. (10).

In order to solve the teacher-student network optimization problem at each block learning stage, we use two terms to compose the objective loss function. Taking the k-th block learning stage for example, the first term of the loss compares the output of the block s_k with its corresponding block in the teacher network t_k, and is formulated as:

$$L^k_{local}(I; W_{s_k}) = \begin{aligned} &\tfrac{1}{2} \parallel (t_k \circ \textstyle\prod_{i=1}^{k-1} \circ s_i)(I; W_{t_k} \cup \{W_{s_i}\}_{i=1}^{k-1}) \\ &-(\textstyle\prod_{i=1}^{k} \circ s_i)(I; \{W_{s_i}\}_{i=1}^{k}) \parallel_F^2 \end{aligned} \tag{11}$$

where I is the input of the network (e.g., image) and $\| \cdot \|_F$ denotes the Frobenius norm.

The second classification loss term L^k_{cls} is to make the output of the student network approximate the ground truth, which is defined as:

$$L^k_{cls}(I, y; W_{s_k}) = softmax(A^k(I; W_{A^k}), y) \tag{12}$$

where y is the ground truth for I and $softmax(.)$ means the softmax loss between the network's output and y as described in [Simonyan and Zisserman, 2015].

Then the objective loss function for one training sample (I, y) at the k-th block learning stage is as below:

$$L^k(I, y; W_{s_k}) = \lambda_{local} L^k_{local}(I; W_{s_k}) + L^k_{cls}(I, y; W_{s_k}) \tag{13}$$

where λ_{local} is a hyper-parameter to balance these two terms of the loss function. Therefore, our objective loss function for the training data $\{(I^{(1)}, y^{(1)}), \dots, (I^{(M)}, y^{(M)})\}$ is:

$$L^k(W_{s_k}) = \frac{1}{M} \sum_{m=1}^{M} L^k(I^m, y^m; W_{s_k}) \tag{14}$$

By optimizing this loss function, the student subnetwork block can be trained under the ground truths and knowledge of the teacher subnetwork block. The details of our progressive blockwise learning scheme are shown in Alg. 1.

Moreover, the optimization order of our method can be flexible. Typically, we try three optimization orders: 1) Bottom-Up $(1, 2, \dots, N)$; 2) Top-Down $(N, N-1, \dots, 1)$; 3) Skipping-First $(2, 3, \dots, N, 1)$. Our experimental results show that optimizing the network in a Bottom-Up fashion gives better results than other orders. Therefore, we use the Bottom-Up optimization order as the default optimization order. The details are discussed in Section 3.

2.3 Student Subnetwork Block Design

We propose a blockwise design scheme. Based on our blockwise design scheme, the student subnetwork block we design should not change the input/output size of the next/previous block and maintain the receptive field at the same time. In addtion, this student subnetwork block is based on the teacher subnetwork block but contains fewer parameters and FLOPs (floating point operations).

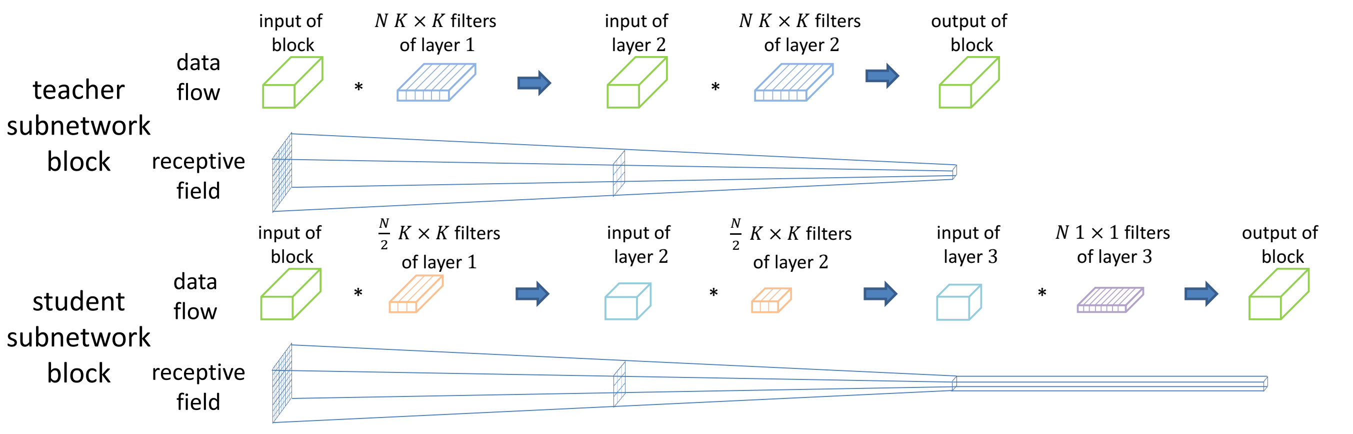

Figure 2: Illustration of the student subnetwork block design. Our student subnetwork block is based on the teacher subnetwork block with fewer convolution operations. First, we prune half of the filters of all teacher subnetwork block's convolution layers (layer 1 and layer 2) to reduce convolution operations. Then we use a 1×1 convolution layer (layer 3) to recover the output size. The figure also illustrates that our design criterion preserves the receptive field from the teacher subnetwork block.

Algorithm 1: Progressive Blockwise Learning

```
Input: A teacher network $T = t_N \circ t_{N-1} \circ \cdots \circ t_1$ and
       the corresponding parameters
       $W_T = \{W_c, W_{t_N}, W_{t_{N-1}}, \ldots, W_{t_1}\}$.
Output: The objective student network $S$ and the
        corresponding optimal parameters $W_S$.
1 Design the structure of $s_i$ by the structure of $t_i$ as
  described in Section 2.3 with the corresponding
  randomly initialized weights $W_{s_i}$ $(i = 1, 2, \ldots, N)$;
2 Build the auxiliary function $A^0$ and its parameters $W_{A^0}$
  by the teacher network $T$;
3 for $k$=1, 2, ..., $N$ do
      /* Learn $A^k$ from $A^{k-1}$ at the $k$-th
         stage                                   */
4     Build the auxiliary function $A^k$ and its parameters
      $W_{A^k}$ using Eq. (8) and Eq. (9);
5     Obtain the optimal parameters $W_{s_k}$ by minimizing
      Eq. (14);
6     Update $W_{A^k}$ using Eq. (9);
7 end
8 return $S = A^N$;
```

More specifically, we prune half of the channels of all convolution layers which belong to the teacher subnetwork blocks to construct a simple student subnetwork block. Then we apply one 1×1 convolution layer at the end of the block to expand the number of output channels to the original number. Fig. 2 shows our student subnetwork block design based on the teacher subnetwork block of two $K \times K$ convolution layers. The figure also illustrates that our design criterion preserves the receptive field from the teacher subnetwork block.

Blockwise Complexity Analysis

The main complexity of the model's computation costs lies in the convolution layers. For network acceleration, our method mainly focuses on the convolution layers. Taking the VGG-16 model [Simonyan and Zisserman, 2015] for example, we will analyze the FLOPs reduction of the general student subnetwork block in this section.

Let the FLOPs of a $K \times K$ $(K = 3)$ convolution layer be $Conv_{fl}$. If the teacher subnetwork block has L $(L \in \{2, 3\})$ layers described above, then its total FLOPs TB_{fl} is $L \times Conv_{fl}$.

As shown in Fig. 2, we have three types of convolution layers based on our design. The first type of convolution layer is the same input and half the number of filters as the original convolution layer (Layer 1 in Fig. 2). The second type of convolution layer is half the input and half the number of filters as the original convolution layer (Layer 2 in Fig. 2). The third type of convolution layer is half the input and the same number of filters as the original convolution layer (Layer 3 in Fig. 2), but its kernel size is 1×1. So the FLOPs SB_{fl} of the student subnetwork block which we design based on the previous teacher subnetwork block is:

$$SB_{fl} = \tfrac{1}{2}Conv_{fl} + (L-1) \times \tfrac{1}{4}Conv_{fl} + \tfrac{1}{2K^2}Conv_{fl} \tag{15}$$

In summary, the FLOPs reduction C_{fl} of the student subnetwork block we design is:

$$C_{fl} = \tfrac{TB_{fl}}{SB_{fl}} \approx 4 - \tfrac{4}{L+1} \tag{16}$$

3 Experiments

3.1 Datasets

CIFAR10 [Krizhevsky and Hinton, 2009] is a labeled subset of the 80 million tiny images dataset for object recognition. This dataset contains 60000 32×32 RGB images in 10 classes, with 5000 images per class for training and 1000 images per class for testing.

CIFAR100 [Krizhevsky and Hinton, 2009] is also a labeled subset of the 80 million tiny images dataset for object recognition. It contains 100 classes including 600 32×32 images each, with 500 images for training and 100 images for testing.

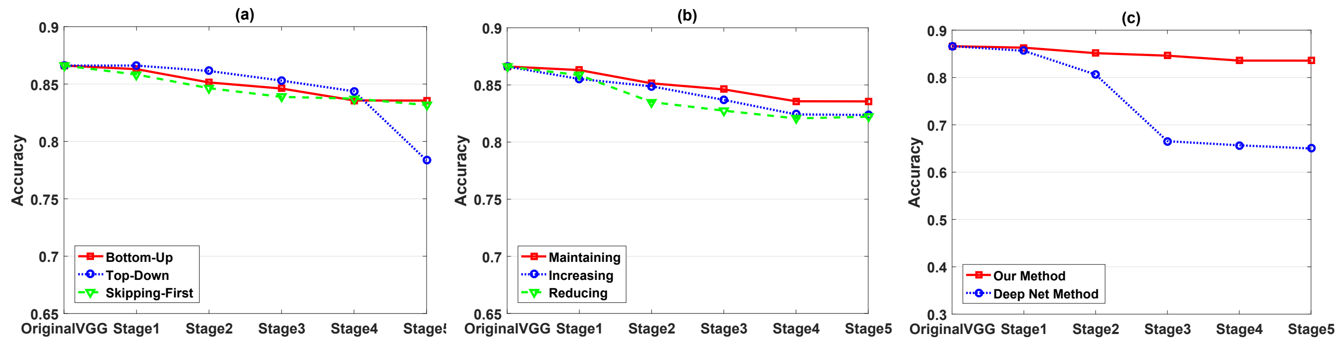

Figure 3: Ablation experiments' results on CIFAR10. Figure (a): Optimization Orders Comparison, the result is better and more stable using the Bottom-Up order. Figure (b): Receptive Fields Comparison, the compressed model's accuracy is the highest at each block learning stage by maintaining the receptive field of each block. Figure (c): Block Structure Comparison, the accuracy of our method is better than Deep Net's [Nowak and Corso, 2018] at every stage.

ImageNet [Krizhevsky *et al.*, 2012] is a dataset for ImageNet Large Scale Visual Recognition Challenge 2012. It contains 1.28 million training images and 50k validation images in 1000 classes.

3.2 Implementation Details

Network Architecture

In this paper, we utilize the widely used network VGG-16 [Simonyan and Zisserman, 2015] as the teacher network sample. We divide the VGG-16 network into five blocks using the pooling layers as the boundaries. For each block, we use our block design criterion described in Section 2.3 to obtain the student subnetwork block.

Data Preprocessing

On CIFAR10 and CIFAR100, the only preprocessing we do is subtracting the mean RGB value computed on the training data from each pixel. On ImageNet, we employ the data augmentation strategy as the practice in [Krizhevsky *et al.*, 2012; Simonyan and Zisserman, 2015], including the 224×224 random cropping, the horizontal flip, and the per-pixel mean subtracted.

Training Details

We implement our architecture using Caffe [Jia *et al.*, 2014] and use an NVIDIA TITAN X GPU to train the network. To validate our proposed learning scheme, we first conduct four ablation experiments on CIFAR10 which will be fully described in Section 3.3 in the following aspects: 1) Optimization Orders Comparison; 2) Receptive Fields Comparison; 3) Block Structure Comparison; 4) Block Training Strategy Comparison. Then we verify our method on CIFAR100 and ImageNet which will thoroughly be described in Section 3.4, and compare our proposed method to recent state-of-the-art approaches.

On CIFAR10 and CIFAR100, we use SGD with a mini-batch size of 100 at each block learning stage. The initial learning rate is set to 0.01 and is divided by 10 after 3 epochs. We train the network using a weight decay of 0.005 and a momentum of 0.9. From our experiments, we notice that each learning stage converges in less than 6 epochs. And so we terminate each learning stage after 6 epochs.

On ImageNet, we use SGD with a mini-batch size of 32 at each block learning stage. The momentum parameter is chosen as 0.9, the initial learning rate is set to 0.01 and the weight decay is 0.0005. It takes 50000 training iterations for our method to converge at every learning stage.

3.3 Ablation Experiments

In this section, we utilize four ablation experiments to validate our method on CIFAR10. And we obtain the original VGG-16 model by fine-tuning the pre-trained model [Simonyan and Zisserman, 2015] on CIFAR10.

Optimization Orders Comparison. We evaluate the performance of different optimization orders. We try three optimization orders: Bottom-Up, Top-Down, Skipping-First as described in Section 2.2. The results in Fig. 3 (a) show that the Bottom-Up order performs better than the other two optimization orders. For Top-Down optimization order, the accuracy of the model drops rapidly at the stage five of optimizing block one which indicates that the first-block features have a significant impact on the higher block features.

Receptive Fields Comparison. We evaluate the performance of the student networks with different receptive fields. The method of maintaining each block's receptive field is fully described in Section 2.3, and repeating/reducing one 3×3 convolutional layer on the basis of the former to increase/reduce the receptive field of each block. As shown in Fig. 3 (b), the model's accuracy is the highest by maintaining the receptive field of each block at every block learning stage. After all block learning stages, the final model's Top-1 accuracy is improved by 1% compared to the other two methods. This indicates that keeping the student subnetwork block's receptive field the same as that of the teacher network block is feasible.

Block Structure Comparison. We compare our student subnetwork block structure design's performance with Deep Net's [Nowak and Corso, 2018]. The Deep Net's method approximates the functions learned by the two or three layers with a single layer in each block. Fig. 3 (c) shows that the structure we design can improve the Top-1 accuracy by 17% over Deep Net's after all block learning stages.

Block Training Strategy Comparison. We evaluate the performance of the loss in Eq. (14). We employ two different training strategies at each block learning stage: 1) cls_loss, we train the block only with the classification loss; 2) cls_loss+local_loss, we train the block with the weighted

Dataset	Strategy	Top-1	Training Epochs
CIFAR10	OriginalVGG	86.61%	–
	cls_loss	72.41%	60
	cls_loss+local_loss	83.56%	30

Table 2: Performance of different block training strategies using progressive learning. The compressed model's accuracy using our strategy (cls_loss+local_loss) is higher than that of training the blocks with the classification loss only (cls_loss) and our strategy makes faster convergence.

Dataset	Method	Top-1	Top-5	#Param. ↓	#FLOPs ↓	f./b.(ms)
CIFAR100	OriginalVGG	60.74%	87.05%	34.00M	0.66B	–
	Ours	-2.22%	-1.89%	1.40×	2.69×	–
ImageNet	OriginalVGG	68.28%	88.36%	138.34M	30.94B	14.98/42.07
	APoZ-1	-2.08%	-0.76%	2.04×	≈1×	–
	APoZ-2	+1.99%	+1.33%	2.70×	≈1×	–
	Taylor-1	–	-1.36%	≈1×	2.68×	–
	Taylor-2	–	-3.86%	≈1×	**3.86**×	–
	ThiNet-Conv	+1.52%	+1.17%	1.05×	3.23×	–
	ThiNet-GAP	-0.94%	-0.44%	**16.63**×	3.31×	–
	Ours-Conv	**+2.00%**	**+1.36%**	1.04×	2.53×	9.90/41.21
	Ours-Conv-FC	+1.59%	+1.17%	4.22×	2.57×	7.92/36.29
	Ours-GAP	-0.55%	-0.02%	13.75×	2.58×	**7.36/35.24**

Table 3: Validation of our method on CIFAR100 and comparison of different methods on ImageNet. Here, the results of APoZ-1, APoZ-2, Taylor-1, Taylor-2, ThiNet-Conv and ThiNet-GAP are based on the original paper. (M/B means million/billion respectively, ↓ means the compression ratio of parameter numbers and FLOPs compared to the original model, f./b. denotes the forward/backward speed tested on one TITAN X GPU with batch size 1)

sum of the classification loss and the local loss described in Eq. (14). As shown in Tab. 2, the final model's accuracy is roughly 97% of the original model using our strategy and our strategy makes faster convergence than another strategy.

3.4 State-of-the-Art Performance Comparison

We evaluate the performance of our method on CIFAR100 and ImageNet, against the state-of-the-art methods, including APoZ-1, APoZ-2 [Hu *et al.*, 2016], Taylor-1, Taylor-2 [Molchanov *et al.*, 2017], ThiNet-Conv and ThinNet-GAP [Luo *et al.*, 2017]. In principle, the original VGG-16 model on CIFAR100 is obtained by fine-tuning the pre-trained model [Simonyan and Zisserman, 2015]; APoZ prunes the zero activation neurons to compress network; Taylor uses Taylor expansion to approximate the loss, and then only utilizes the first order gradient information; ThiNet reduces the number of convolution-layer channels by evaluating channel importance.

Subsequently, we design three variants based on our method: 1) Ours-Conv: it doesn't optimize the last three layers (conv5-1, conv5-2, conv5-3) of the original VGG-16 model similarly to ThiNet-Conv; 2) Ours-Conv-FC: it compresses the fully connected layers by truncated singular value decomposition [Girshick, 2015] on Ours-Conv; 3) Ours-GAP: it replaces the fully connected layers of Ours-Conv with a global average pooling layer [Lin *et al.*, 2014] similarly to ThiNet-GAP. We mainly focus on the VGG-16 model's acceleration, so we only compare our proposed method with the VGG-16 based frameworks as shown in Tab. 3.

In principle, our method is based on the progressive blockwise learning scheme with the structure-preserving design criterion. This criterion focuses on improving the accuracy

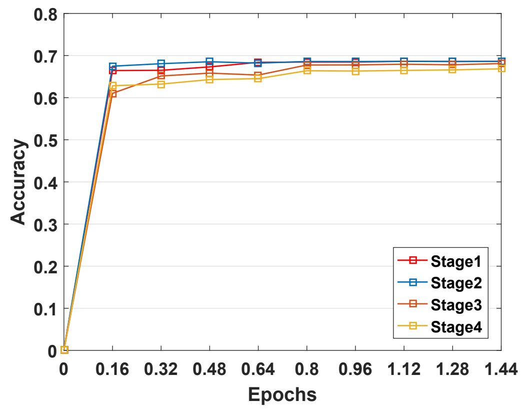

Figure 4: Illustration of the convergence performance of our learning method for each stage on ImageNet. As the epoch goes on, the performance improves rapidly to approximate the original model (0.6828) and then converges.

of the compressed model with a competitive FLOPs reduction. From the comparison of these state-of-the-art methods in Tab. 3, we can observe that the accuracy of our proposed method is the highest and our method's FLOPs reduction is comparable to those of others. In addition, Fig. 4 shows that our method makes fast convergence for different learning stages. Taking ThiNet-Conv [Luo *et al.*, 2017] for example, it demonstrates that ThiNet-Conv requires one or two epochs to converge for each layer, but we only need about 1.5 epochs to converge for each block containing two or three layers. Moreover, as shown in Tab. 3, the results of three variant experiments (Ours-Conv, Ours-Conv-FC, Ours-GAP) demonstrate that our method can be flexibly combined with other methods (e.g., fully connected layers decomposition [Girshick, 2015] and GAP [Lin *et al.*, 2014]) to further reduce the parameters and achieve feasible results.

In summary, our scheme sets up an effective and practical network acceleration pipeline from a new viewpoint of progressive blockwise learning. In contrast to other types of pipelines, ours is a powerful tool in the aspects of structure-preserving for knowledge distillation, easy implementation for progressive blockwise optimization, fast stagewise convergence, flexible compatability with existing learning modules, and good balance with high accuracy and competitive FLOPs reduction.

4 Conclusion

In this paper, we have proposed a novel progressive blockwise knowledge distillation scheme for neural network acceleration, which distills the teacher network model by progressively extracting the knowledge of each block in local optimization. We have also proposed a structure-preserving criterion for the student subnetwork block design. The proposed criterion is able to keep the original receptive field unchanged from the teacher network. Therefore, our proposed progressive blockwise learning scheme provides a novel theoretical perspective as well as a promising practical alternative for neural network acceleration. Moreover, the experimental results on several datasets also demonstrate the effectiveness of our scheme.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants (U1509206, 61472353, and 61751209), in part by the National Basic Research Program of China under Grant Grant 2015CB352302.

References

[Ba and Caruana, 2014] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NIPS*, pages 2654–2662, 2014.

[Bucila *et al.*, 2006] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. 2006.

[Courbariaux *et al.*, 2015] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, pages 3123–3131, 2015.

[Girshick, 2015] Ross Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[Hinton *et al.*, 2014] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NIPS Workshop*, 2014.

[Hu *et al.*, 2016] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[Hubara *et al.*, 2016] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *NIPS*, pages 4107–4115, 2016.

[Jaderberg *et al.*, 2014] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

[Jia *et al.*, 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, pages 675–678, 2014.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[Li *et al.*, 2017a] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2017.

[Li *et al.*, 2017b] Quanquan Li, Shengying Jin, and Junjie Yan. Mimicking very efficient network for object detection. In *CVPR*, pages 7341–7349, 2017.

[Li *et al.*, 2017c] Zefan Li, Bingbing Ni, Wenjun Zhang, Xiaokang Yang, and Wen Gao. Performance guaranteed network acceleration via high-order residual quantization. In *ICCV*, 2017.

[Lin *et al.*, 2014] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014.

[Lin *et al.*, 2016] Shaohui Lin, Rongrong Ji, Xiaowei Guo, Xuelong Li, et al. Towards convolutional neural networks compression via global error reconstruction. In *IJCAI*, pages 1753–1759, 2016.

[Luo *et al.*, 2017] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017.

[Molchanov *et al.*, 2017] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. In *ICLR*, 2017.

[Nowak and Corso, 2018] Theodore S Nowak and Jason J Corso. Deep net triage: Assessing the criticality of network layers by structural compression. *arXiv preprint arXiv:1801.04651*, 2018.

[Rastegari *et al.*, 2016] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pages 525–542. Springer, 2016.

[Romero *et al.*, 2015] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

[Sainath *et al.*, 2013] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *ICASSP*, pages 6655–6659, 2013.

[Simonyan and Zisserman, 2015] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[Szegedy *et al.*, 2017] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[Tai *et al.*, 2016] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. In *ICLR*, 2016.

[Wu *et al.*, 2016] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *CVPR*, pages 4820–4828, 2016.

[Zhou *et al.*, 2017] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *ICLR*, 2017.