

# Detachable Second-order Pooling: Towards High Performance First-order Networks

Lida Li, Jiangtao Xie, Peihua Li, *Member, IEEE* and Lei Zhang, *Fellow, IEEE*

**Abstract**—Second-order pooling has proved to be more effective than its first-order counterpart in visual classification tasks. However, second-order pooling suffers from the high demand of computational resource, limiting its use in practical applications. In this work, we present a novel architecture, namely detachable second-order pooling network, to leverage the advantage of second-order pooling by first-order networks while keeping the model complexity unchanged during inference. Specifically, we introduce second-order pooling at the end of a few auxiliary branches and plug them into different stages of a convolutional neural network. During the training stage, the auxiliary second-order pooling networks assist the backbone first-order network to learn more discriminative feature representations. When training is completed, all auxiliary branches can be removed and only the backbone first-order network is used for inference. Experiments conducted on CIFAR-10, CIFAR-100 and ImageNet datasets clearly demonstrated the leading performance of our network, which achieves even higher accuracy than second-order networks but keeps the low inference complexity of first-order networks.

**Index Terms**—Image classification, second-order pooling, first-order networks.

## I. INTRODUCTION

DEEP convolutional neural networks (CNNs) have been widely used in tackling various computer vision problems, including visual object recognition [1], [2], [3], face recognition [4], [5], [6], person re-identification [7], [8], [9] and scene understanding [10], [11], [12], among others. Tremendous efforts have been devoted to the design of CNN architectures for boosting performance. It is widely acknowledged that deeper and/or wider networks, such as ResNet [13], Inception [14] and ResNeXt [15], could have higher representation learning capability. However, the increase of network depth/width will also bring more overhead and difficulties for network deployment.

Another factor affecting the learning capability of neural networks is the pooling strategy. In recent years, global second-order pooling (GSoP) networks [16], [17], [18], [19], [20], [21] have attracted a lot of attentions. By replacing the classical global average pooling (GAP) with covariance pooling at the end of CNNs, significant improvement has been reported on large-scale visual recognition tasks. For example, ResNet-50 with GSoP surpasses ResNet-152 [21]. The GAP [22] calculates the first-order statistics (i.e., mean) of individual channels without considering the interactions

between channels, while the global covariance pooling computes the second-order statistics of high-level convolutional features by exploiting the pair-wise channel correlations, leading to stronger statistical modeling capability. Though the use of covariance matrix to represent image statistics enhances the nonlinear learning capability of networks, the required computational complexity increases quadratically, significantly higher than its first-order counterpart.

Either increasing the width/depth of networks or employing covariance pooling will consume much more computational resources. One interesting question is whether we can leverage the advantage of second-order pooling in the first-order networks while keeping the model complexity unchanged. This work attempts to solve this challenging problem. Inspired by the knowledge distillation method [23], we propose a novel architecture, called detachable second-order pooling network (DSoP-Net), where the covariance pooling networks assist the first-order network to learn more discriminative representations during training; however, during the inference stage, the covariance pooling networks can be removed and only the trained first-order network is deployed. The proposed DSoP-Net achieves significant performance gains without introducing any additional cost. In particular, on the large-scale ImageNet dataset [1], DSoP-Net achieves a top-1 error rate of 21.15% with a single ResNet-50 network.

The key idea of DSoP-Net lies in that a weak pooling method (student) can learn from stronger ones (teachers). Existing methods of this kind, such as knowledge distillation [23], [24], [25], often explicitly minimize the discrepancy between features produced by one or more teacher networks and the student network. The success of such methods largely relies on a pre-trained, high performance teacher network as well as the skillful design of metrics to measure the discrepancy differences so that the knowledge can be well transferred. In contrast, in this paper we employ a simple yet effective regularization term by applying the same criterion used in the original first-order pooling to the second-order one. As a result, no extra metric is needed and the knowledge induced by covariance pooling can easily flow into the first-order network.

Figure 1 presents an overview of DSoP-Net. During training, auxiliary branches are employed, each with a covariance matrix based second-order pooling and an output header. This allows us to learn spatial information at intermediate layers by adjusting the channel correlations with deep supervision. Once these auxiliary branches are plugged into the backbone architecture at different stages, they actively cooperate with the first-order pooling and its corresponding output header. The

L. Li and L. Zhang are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: {cslli, csliang}@comp.polyu.edu.hk.

J. Xie and P. Li are with the School of Information and Communication Engineering, Dalian University of Technology, China. E-mail: jiangtaoxie@mail.dlut.edu.cn, peihuali@dlut.edu.cn

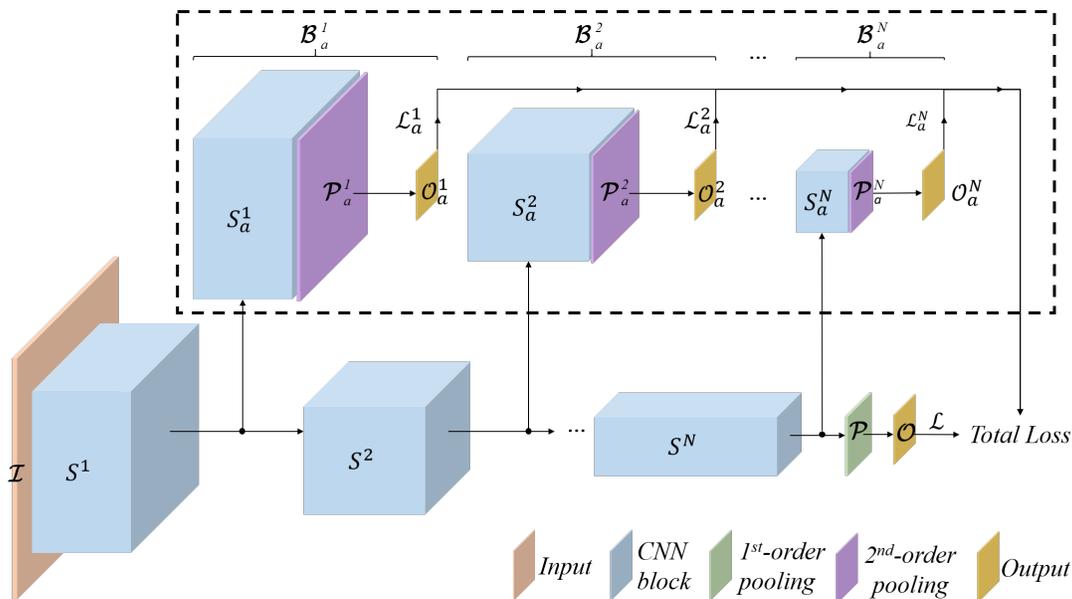


Fig. 1: The proposed auxiliary second-order pooling networks (subnetworks within the dashed rectangular box) assist the discriminative representation learning of the backbone first-order network. When deployment, we obtain a high-performance first-order network by detaching the auxiliary branches.

network is optimized with the loss function that is composed of the first-order output header and all extra second-order ones. When training is completed, all auxiliary branches are removed and only the backbone first-order network is preserved for inference. The resulting DSoP-Net is very powerful, even outperforming its second-order counterparts but with much lower model complexity.

The rest of the paper is organized as follows. Section II reviews some related works. Section III and Section IV introduce our methods and present implementation details, respectively. Section V shows experimental results of DSoP-Net and Section VI concludes the paper.

## II. RELATED WORK

Section II-A briefly reviews global pooling methods in literature. Section II-B presents the use of auxiliary networks in the training of backbone network. In Section II-C, we discuss squeeze-and-excitation networks and its differences from our DSoP-Net.

### A. Global pooling

The CNN models learn discriminative features in an end-to-end manner. At the end of the network, a global pooling of convolutional features is often performed to represent the whole image for classification. We briefly review the first-order pooling methods [1], [22] and the high-order ones [16], [20], [19], [17], [18], [21], [26], respectively.

First-order global pooling methods apply a unary operator to each feature map and concatenate all outputs as the final output. Lin *et al.* [22] first performed Global Average Pooling (GAP) in a network by averaging final convolutional features to obtain a vector descriptor. Thanks to this design, the

cost of high-dimensional dense layers in networks such as AlexNet [1] and VGGNet [27] can be largely reduced. GAP is widely adopted in mainstream CNN architectures, including ResNet [13], DenseNet [28], ResNeXt [15], MobileNet [29], and Inception networks [30], [31], [32]. Statistically, GAP summarizes the first-order statistics (i.e., mean) of high-level convolutional features, neglecting the higher-order statistics.

High-order global pooling algorithms aim at more discriminative image representation. Most works in this category exploit pairwise correlations between channels while some ones, e.g., [26], further consider higher-order interactions of features. Bilinear CNN (B-CNN) [16], [20] and DeepO<sub>2</sub>P [33] are pioneering works. Both of them compute covariance matrix (or second-order moments) as the global image representations. MPN-COV [19] and its fast version [21] (i.e., iSQRT-COV) have reported compelling performance on large-scale visual recognition and fine-grained classification, significantly outperforming the first-order networks. Unfortunately, the covariance representations are of hundreds of thousands of dimensions. In addition, computing high-order statistics is time-consuming at both training and test stages compared to their first-order counterparts. This limits the practical applications of second-order networks, especially on resource limited devices.

Several methods have been proposed to improve the efficiency of high-order global pooling methods. Compact Bilinear Pooling [34] compresses the full bilinear pooling and achieves comparable performance with significantly reduced parameters. In [19], [21], dimensionality reduction is performed prior to second-order pooling. Moreover, [21] only carries out basic matrix operations suitable for GPU to speed up. Nevertheless, these second-order based methods are still much slower than their first-order counterparts.

## B. Auxiliary networks

Recently, a few methods have been proposed to employ an auxiliary network for backbone network training. To the best of our knowledge, the Inception network architecture [30] is among the first works that utilize auxiliary branches with carefully crafted design for classification and detection tasks. Similarly, knowledge distillation in deep CNNs is an effective approach to transferring the knowledge from more powerful models into weaker ones for inference [23], [24]. Hinton *et al.* [23] proposed to transfer knowledge from an ensemble of acoustic models into a smaller, distilled one for easier deployment. Gupta *et al.* [24] transferred the learned representations from a well labeled domain, obtaining large performance boost by learning rich representations on the unseen domain. Recently, Yim *et al.* [25] introduced a sequential flow between layers to distill knowledge. Furlanello *et al.* [35] trained student models parameterized identically to their teachers so that students can even outperform their teachers in some small scale vision tasks.

## C. Squeeze-and-excitation networks

Based on the prior that it is important to fuse both spatial and channel-wise information at each layer, a lightweight block, called squeeze-and-excitation (SE) Networks (short for “SE-Net” in the rest of this paper), has been recently developed in [36]. It first introduces extra GAP layers followed by convolutional and non-linear activation layers. Then, the spatial features are adaptively adjusted along the channel dimension regarding to the results computed from the last step. Though SE-Net can improve much the performance, computation of the correlations is required so that the extra layers cannot be removed in the inference stage.

We argue that it is possible to design a detachable version of SE-Net which can achieve equivalent or even better performance. Actually, the re-calibration operation at the last step of the original SE block can be approximated and replaced by incorporating additional gradients computed based on channel correlations during training, which are not required and can be omitted in inference. In this paper, DSoP-Net is proposed as the first attempt to reach this goal.

## III. PROPOSED METHOD

Section III-A introduces our DSoP-Net in detail, and Section III-B presents how we solve the issue of dimensionality reduction in second-order pooling methods.

### A. DSoP-Net

Inspired by the works of knowledge distillation [23], [25], we propose to improve the first-order pooling network without introducing extra parameters and computational cost during inference. Our idea is to transfer the knowledge of second-order pooling networks (teachers) to the first-order pooling network (student) in the training stage, while the teacher networks can be detached from the student in the inference stage. To achieve this goal, existing knowledge distillation frameworks often employ one or more metrics to properly

measure the discrepancy between the output of student and teacher networks in the total loss; however, existing metrics, such as  $p$ -norm, Kullback-Leibler (KL) divergence, Jensen-Shannon (JS) divergence and Wasserstein divergence, are not suitable to directly compute the distance between first- and second-order pooling outputs.

We propose a simple yet effective solution without the need of extra metrics required in knowledge distillation methods. The auxiliary branch networks are introduced and attached to the first-order pooling network to form a multiple output network, where all outputs are identical to the original output of the first-order network in one task. This allows us to reuse the same criterion, e.g. cross-entropy loss in the classification task, to measure the distance between output of any auxiliary branch and the label. By summing the losses of auxiliary branches into the total loss, knowledge and expertise of the teacher networks can be taught to the student network by computing the gradients of auxiliary branches *w.r.t.* the first-order network.

To clearly illustrate the structure of DSoP-Net, we begin with the structure of an auxiliary branch designed in DSoP-Net, followed by the total loss and some discussions.

1) *Structure of an auxiliary branch:* Denote by  $y_l$  the output of the  $l$ -th layer of the backbone model. The  $i$ -th auxiliary branch in DSoP-Net is made up of three parts, including a number of convolution and non-linear activation layers  $S_a^i$ , a covariance-based second-order pooling layer  $\mathcal{P}_a^i$ , and task-dependent output  $O_a^i$  (please refer to Fig. 1). Before introducing the general case of auxiliary branches, we first discuss the case when there is only one auxiliary branch inserted after the  $l$ -th layer during training, as shown in Fig. 2(c).

The first part of our auxiliary branch consists of convolutional layers and non-linear activation layers  $S_a^1$ , which are used to extract features from  $y_l$  for second-order pooling. We reuse the building block of the first-order pooling network, such as bottlenecks of ResNet [13] or its variants [15], [37]. No down-sampling operation is performed so that the output of this part has the same height and width as those of  $y_l$  but the channel number can be determined as a hyper-parameter. If we train DSoP-Net without this part in the auxiliary branch, there is a clear performance drop under the same experimental settings. This is because the intermediate features, especially those of layers closer to the input, are not discriminative enough to fulfill a task.

The second part of our auxiliary branch is a covariance matrix based second-order pooling layer  $\mathcal{P}_a$ . The covariance matrix describes the channel correlations of the output of  $S_a$ . The element  $(i,j)$  of the covariance matrix is obtained by computing the inner production of the  $i$ -th and the  $j$ -th channels after they are vectorized. Once the covariance matrix is ready, we proceed to normalization, such as matrix logarithm [38] and matrix power [19], [21], etc. The upper- or lower-triangular matrix of the results is re-arranged to form a vector, regarded as the output of the second part.

<sup>1</sup>Superscript is omitted as there is only one branch in this case.

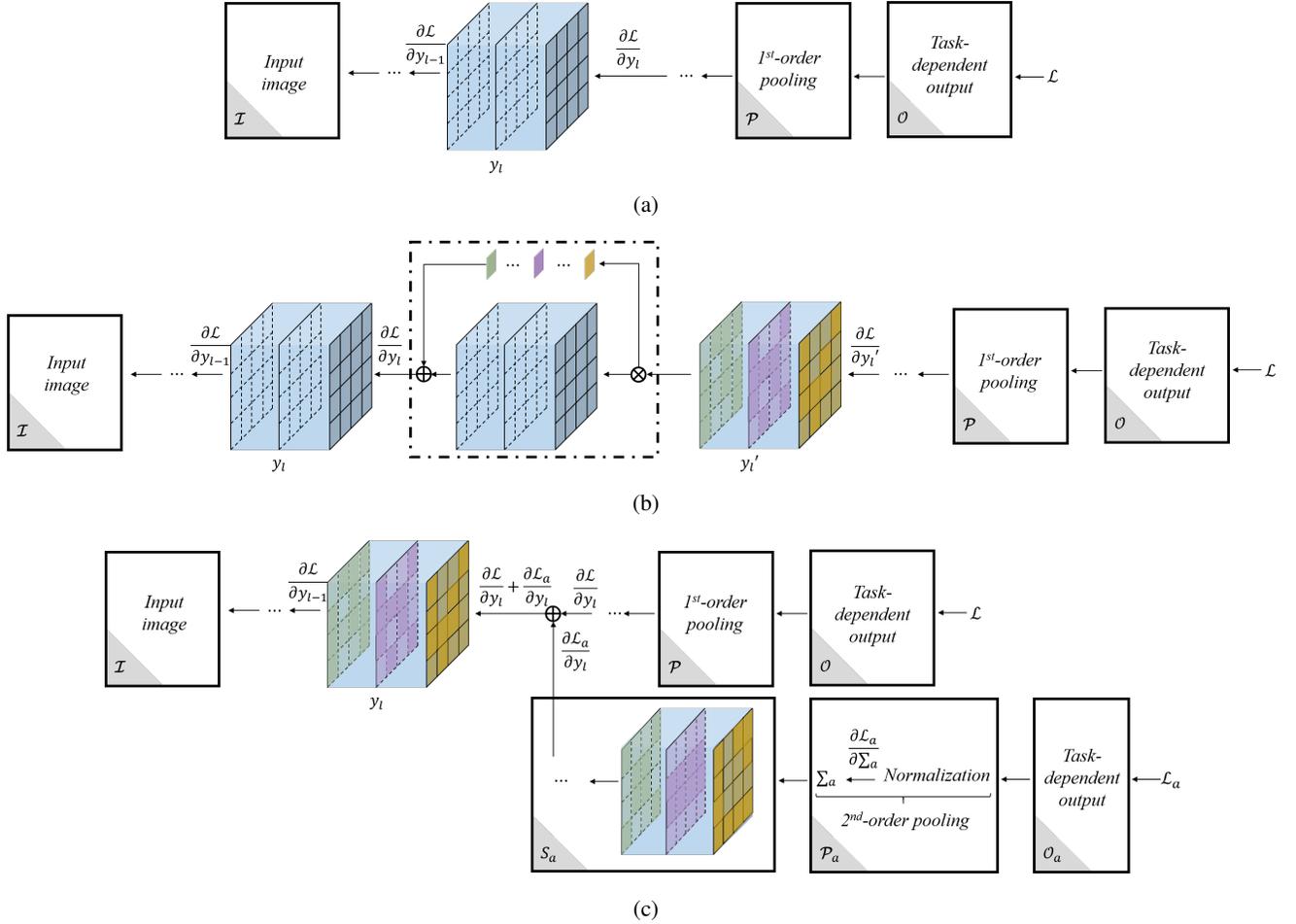


Fig. 2: Comparison of back propagation at the  $l$ -th layer of a first-order pooling network with different architectures, including (a): a normal first-order pooling network, (b): a first-order pooling network with the  $l$ -th layer modified as a squeeze-and-excitation block [36] (dash-dotted rectangle), and (c): DSoP-Net with an auxiliary branch plugged after the  $l$ -th layer (here we assume  $\mathcal{L}$  and  $\mathcal{L}_a$  are equally contributed in the total loss). A “ $\oplus$ ” symbol stands for the operation of element-wise plus and a “ $\otimes$ ” symbol stands for channel-wise multiplication between a scalar and its corresponding feature map in back propagation.  $y'_l$  denotes the output of the squeeze-and-excitation block related to the  $l$ -th layer.

The third part is to produce task-dependent output  $\mathcal{O}_a$ . It can be easily adapted from the corresponding structure in the backbone model. Since in this work we focus on the classification task, we leverage a fully-connected layer as a linear classifier. Meanwhile, it allows us to reuse the cross-entropy loss in classification tasks to effectively update the weights of an auxiliary branch in DSoP-Net during training. The weights of the  $l$ -th layer can be updated by minimizing the loss of the branch and the loss of the backbone first-order pooling network together. From the perspective of back propagation, the gradients related to channel correlations in the branch are merged into the gradients in the first-order pooling network.

2) *The total loss:* In the general case, we suppose that there are  $N$  auxiliary branches plugged into a first-order pooling network during training. Let  $\mathcal{B}_a^1, \dots, \mathcal{B}_a^N$  denote these branches in order as shown in Fig. 1. It always holds that  $\mathcal{B}_a^i$  is inserted closer to the input than  $\mathcal{B}_a^j$  in DSoP-Net,  $\forall i < j, i, j = 1, \dots, N$ . Let  $\mathcal{L}$  denote the loss computed for

the original output of the backbone model  $\mathcal{O}$ .  $\mathcal{O}_a^1, \dots, \mathcal{O}_a^N$  are the outputs of the auxiliary branches, sharing the same shape and meaning as  $\mathcal{O}$ . We employ the same criterion to compute the losses of auxiliary branches,  $\mathcal{L}_a^1, \dots, \mathcal{L}_a^N$ .

We sum up  $\mathcal{L}_a^1, \dots, \mathcal{L}_a^N$  together with  $\mathcal{L}$  as the total loss of the DSoP-Nets to minimize:

$$\mathcal{L}_{DSoP-Net} = \alpha \mathcal{L} + \sum_{i=1}^N \beta_i \mathcal{L}_a^i, \quad (1)$$

where  $\alpha$  and  $\{\beta_i\}_{i=1}^N$  are weights used to balance the contribution of each term. In this paper, we set  $\alpha = \beta_1 = \dots = \beta_N = 1$  by default except stated otherwise.

For any layer of the backbone model between insertion point of  $\mathcal{B}^k$  (included) and the insertion point of  $\mathcal{B}^{k-1}$  (excluded), its gradients are determined by outputs of the original first-order pooling and all the auxiliary branches inserted after that layer. The gradients of the  $l$ -th layer of the backbone model

can be written as

$$\frac{\partial \mathcal{L}_{DSoP-Net}}{\partial y_l} = \frac{\partial (\alpha \mathcal{L} + \sum_{i=k}^N \beta_i \mathcal{L}_a^i)}{\partial y_l}, \quad (2)$$

where  $y_l$  is the output of the  $l$ -th layer. For those outputs of branches plugged before the specific layer, it can be clearly observed that they are irrelevant to the update of the  $l$ -th layer's weights.

It should also be noted that a layer in the backbone model will not be updated with additional second-order statistics if there is no auxiliary branch inserted after that layer. In practice, we always insert the last auxiliary branch  $\mathcal{B}^N$  right before the first-order pooling layer to make sure that each layer of the backbone model (except the original pooling layer and its classifier) can acquire extra cues during training.

After training is completed, we remove all auxiliary branches so that the first-order network has the same parameters and computational overhead as it originally has during test.

3) *Discussions*: Once optimized, DSOP-Net can work without extra parameters and computation of channel-information in auxiliary branches. As we will see in the section of experimental results, DSOP-Net exhibits highly competitive performance with second-order pooling networks. This leads to an interesting question: how does channel-based information contribute to a CNN model during training and test?

It is important to fuse spatial and channel-based cues of a CNN model to boost model performance. As the channel-based cues can be directly computed from the spatial responses, optimizing spatial responses *w.r.t.* channel-based information can improve the latter one in return. Therefore, there is no need to explicitly compute channel-based cues during test as the channel-based cues have been determined once the spatial responses are given. In other words, when we jointly optimize the spatial and channel-based cues during training, the spatial cues can be further enhanced.

It is critical to ensure that the backbone network used for training is equivalent to the one used for testing after we detach all the auxiliary branches from it. When two CNNs have the same weights and the same computational graph, they will obtain the same output for the same input. As detaching the branches does not affect the value of any weight, all weights of DSOP-Net remain unchanged in the testing stage. As for the computational graph, though all auxiliary branches are detached, no operations are removed or added in the computational graph of the remained DSOP-Net. Therefore, the entire route from the input to the desired output  $\mathcal{O}$  is the same before and after the detachment. The DSOP-Net used for training is equivalent to the one used for testing.

To further explain the detachable property of DSOP-Net, in Fig. 2 we compare DSOP-Net with the original first-order pooling network and SE-Net [36] from the perspective of back propagation. One can see that gradients at the  $l$ -th layer of the original first-order pooling network are computed only regarding to the original loss  $\mathcal{L}$ . In contrast, gradients obtained at the  $l$ -th layer of both DSOP-Net and SE-Net can be written as the weighted summation of two terms corresponding to the spatial and channel-based information. In the dash-dotted

rectangle, it can be observed that the weight of one term in SE-Net is partially determined by the output of the other one according to the definition of channel-wise multiplication. However, weights of both terms in DSOP-Net are independent to each other. They are pre-defined by the relative contributions of  $\mathcal{L}$  and  $\mathcal{L}_a$  in the total loss, which is concise and easy to compute. It remains unsolved in [36] whether the channel-wise information plays a more important role than the spatial information as an SE block can be hardly partitioned for in-depth study. With DSOP-Net, however, we are able to unveil that it is merely important to explicitly compute channel-wise information during test when the spatial information is well learned. It can be seen that at the test stage DSOP-Net can be regarded as a special SE-Net whose channel-wise information is always trivial, e.g. **1s**. Besides, it can be found from the experimental results in Section V that DSOP-Net can achieve equivalent or even better performance than SE-Net.

### B. Progressive supervised dimensionality reduction

The output feature dimension of covariance matrix based second-order pooling method is proportional to the square of input channels. As a result, dimensionality reduction (DR) is required prior to a second-order pooling layer to save computational overhead and prevent over-fitting. However, a large channel reduction ratio often leads to significant performance drop. The DR operator in [19], [21], for example, consists of a  $1 \times 1$  convolution, followed by BN and ReLU layers. When it reduces the channel number from 2048 to 64 so that the dimension of second-order representation is comparable to that of the vanilla ResNet-50 model (with GAP) at 2K-d, the performance gain almost fades out while it costs 24% more inference time to perform GSoP. Clearly, it is of vital importance to develop a compact yet effective DR operator.

In this paper, we propose a compound DR operator which consists of a sequence of lightweight DR operators, called progressive supervised dimensionality reduction (PSDR). PSDR shares the same design principle of DSOP. Their main difference lies in where they locate in a CNN model. A CNN model can be regarded as a multi-step transform that maps its input domain to the desired output domain. Both PSDR and DSOP aim to introduce the second-order statistics from the detachable branches to actively guide and improve the learning of each step, i.e., they both exploit the second-order statistics. A straight of  $M$  intermediate layers are created for PSDR during training. Each intermediate layer gently reduces the channel to some extent without decreasing the performance, and all these layers work together to reduce the channel to the desired dimension. In addition, we incorporate auxiliary branches to better transfer the expertise and knowledge in high-dimensional domain to low-dimensional domain at all intermediate layers. Specifically, an auxiliary branch is inserted after each intermediate layer during training, and there are  $M$  auxiliary branches  $\mathbb{B}^1, \dots, \mathbb{B}^M$ . Similar to  $\mathcal{B}^1, \dots, \mathcal{B}^N$ , each auxiliary branch used in PSDR is composed of three parts. The first part is formed by a  $1 \times 1$  convolutional layer followed by batch normalization and non-linear activation layer, e.g. ReLU, reducing the given number of channels to the desired number.



TABLE I: DSoP-Net with modified ResNet architecture. The input image size is  $224 \times 224 \times 3$ .  $1^{st}$ - and  $2^{nd}$ -PDS are short terms for  $1^{st}$ - and  $2^{nd}$ -order pooling, dense layer and softmax, respectively. Figures in bracket indicate kernel size and number of channels used for pooling.

Layer Name	Output Size	18-layer	34-layer	50-layer	101-layer
conv1	$112 \times 112$	$7 \times 7, 64, \text{stride } 2$			
conv2_x	$56 \times 56$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
a_conv2_x	$56 \times 56$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	N.A.
a_pds2	$1 \times 1$	$2^{nd}\text{-PDS}(64)$	$2^{nd}\text{-PDS}(64)$	$2^{nd}\text{-PDS}(64)$	N.A.
conv3_x	$28 \times 28$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
a_conv3_x	$28 \times 28$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	N.A.
a_pds3	$1 \times 1$	$2^{nd}\text{-PDS}(128)$	$2^{nd}\text{-PDS}(128)$	$2^{nd}\text{-PDS}(256)$	N.A.
conv4_x	$14 \times 14$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$
a_conv4_x	$14 \times 14$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 13$	N.A.
a_pds4	$1 \times 1$	$2^{nd}\text{-PDS}(256)$	$2^{nd}\text{-PDS}(256)$	$2^{nd}\text{-PDS}(256)$	N.A.
conv5_x	$14 \times 14$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
a_conv5_x	$14 \times 14$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	N.A.
a_pds5	$1 \times 1$	$2^{nd}\text{-PDS}(256)$	$2^{nd}\text{-PDS}(256)$	$2^{nd}\text{-PDS}(256)$	N.A.
		$1^{st}\text{-PDS}(512)$		$1^{st}\text{-PDS}(2K)$	
GFLOPs (Train.)		7.26	9.81	13.45	10.01
GFLOPs (Test)		3.06	5.61	6.27	10.01
#Param (Train)		100.6M	110.7M	181.9M	44.5M
#Param (Test)		11.7M	21.8M	25.6M	44.5M

than training its backbone model with the same experimental configuration. The extra time comes from two parts: auxiliary branches and second-order pooling layers. As the total number of convolutional layers in our auxiliary branches are by default the same as that of the backbone model, the auxiliary convolutional layers roughly double the training time. The extra training time costed by the second-order layers depends on the number of auxiliary branches. For example, with DSoP-Net for ImageNet, it takes approximately 74, 97, and 158 hours to train a ResNet-18, ResNet-34 and ResNet-50, respectively.

### B. Organization of PSDR

To balance precision and speed of PSDR, we reduce half of the channels at the first intermediate layer in our implementation. For the rest of intermediate layers, we introduce a

decay factor  $\delta \in (0, 1)$  for gentle DR. The number of output channels at an intermediate layer can be computed by

$$C_a^i = \begin{cases} 0.5C_a^0, & i = 1, \\ \lfloor \delta C_a^{i-1} \rfloor, & \text{otherwise,} \end{cases} \quad (4)$$

where  $C_a^0$  is the number of given feature maps for PSDR. We set  $\delta = 0.25$  in the remaining of this paper unless otherwise specified. For example, given  $C_a^0=2048$  feature maps, we set up 2 intermediate layers for PSDR and they are associated with 1024 and 256 channels, respectively. Besides, we directly reduce the number of current channels to the desired number in each auxiliary branch before we perform second-order pooling in case of limited GPU memory.

TABLE II: Comparison of error rates (%) achieved by different methods with ResNet-50 on ImageNet.

Method	Pooling type (Train.)	Pooling type (Test)	GFLOPs	Extra GFLOPs	#Param. (M.)	Extra #param. (M.)	Top-1	Top-5
ResNet-50-s2 [13]	1 <sup>st</sup>	1 <sup>st</sup>	3.86	N.A.	25.6	N.A.	23.85	7.13
ResNet-50-s1 [13]	1 <sup>st</sup>	1 <sup>st</sup>	6.27	N.A.	25.6	N.A.	23.57	6.85
Knowledge distillation [23] w/ ResNet-50-s1	1 <sup>st</sup>	1 <sup>st</sup>	3.86	0	25.6	0	23.73	7.03
Knowledge distillation [23] w/ ResNet-101	1 <sup>st</sup>	1 <sup>st</sup>	3.86	0	25.6	0	23.54	6.70
Knowledge distillation [23] w/ SE-ResNeXt-101	1 <sup>st</sup>	1 <sup>st</sup>	3.86	0	25.6	0	23.46	6.66
FBN [39]	1 <sup>st</sup>	1 <sup>st</sup>	N.A.	N.A.	N.A.	N.A.	24.00	7.10
SORT [40]	1 <sup>st</sup>	1 <sup>st</sup>	N.A.	N.A.	N.A.	N.A.	23.82	6.72
SE-Net [36]	1 <sup>st</sup>	1 <sup>st</sup>	3.87	0.01	28.1	2.5	23.29	6.62
CBAM [41]	1 <sup>st</sup>	1 <sup>st</sup>	3.86	0.004	28.1	2.5	22.66	6.31
MPN-COV [19]	2 <sup>nd</sup>	2 <sup>nd</sup>	6.43	0.16	56.9	31.3	22.74	6.54
iSQRT-COV [21]	2 <sup>nd</sup>	2 <sup>nd</sup>	6.43	0.16	56.9	31.3	22.14	6.22
<b>ResNet-50-s2</b> <b>w/ DSoP-Net (ours)</b>	1 <sup>st</sup> &2 <sup>nd</sup>	1 <sup>st</sup>	3.86	0	25.6	0	23.14	6.56
<b>ResNet-50-s1</b> <b>w/ DSoP-Net (ours)</b>	1 <sup>st</sup> &2 <sup>nd</sup>	1 <sup>st</sup>	6.27	0	25.6	0	<b>21.15</b>	<b>5.70</b>

## V. EXPERIMENTS

We evaluate the proposed method on large-scale image classification dataset ImageNet, as well as CIFAR-10 and CIFAR-100 datasets. All experiments are conducted on a machine equipped with dual Intel Xeon Gold 6136@3.0GHz CPUs, 128G DDR4 2666MHz RAM, 1T nvme m.2 SSD and 8 NVIDIA Tesla P100 GPU cards. We implement our method by using PyTorch [42] compatible with CUDA and cuDNN.

### A. Datasets

We adopt ImageNet LSVRC2012 dataset [43] with 1,000 classes for large-scale image classification task. The dataset contains over 1.2 million images for training, 50 thousand images for validation, and 100 thousand images for testing. As labels of the test images are not released, we follow [31], [13] and compare methods on the validation set.

We also evaluate the generalization capability of the proposed DSoP-Net on the CIFAR-10 dataset and the CIFAR-100 dataset [44]. Both datasets are well balanced, consisting of 60,000  $32 \times 32$  colour images from 10 and 100 classes, respectively. For each dataset, 50,000 images are used for training and the remaining 10,000 are adopted for testing.

### B. Experimental Settings

We closely follow standard experimental settings on ImageNet as well as CIFAR-10 and CIFAR-100 datasets for fair comparison. Details are presented below.

**Experimental setting for ImageNet.** In the training phase, we first resize each image so that its shorter side is randomly sampled on [256, 512] [14]. Then, a fixed-size  $224 \times 224$  patch is randomly cropped from the down-scaled image or its horizontally flipped version. Finally, we normalize each patch by subtracting the dataset mean and dividing it by the dataset standard deviation. In the testing phase, we resize each test image so that its shorter side is 256 and a single  $224 \times 224$

center crop is applied for inference. We use SGD [45] with a mini-batch of 256 for optimization and set weight decay to  $1 \times 10^{-4}$  and momentum to 0.9. We train DSoP-Net from scratch for 90 epochs. Learning rate starts at 0.1, and is reduced to 0.01 and 0.001 at Epoch 30 and 60, respectively.

**Experimental setting for CIFAR-10 and CIFAR-100.** Standard data augmentation strategy [22], [46], [36] is adopted in training, where images are horizontally flipped at random and zero-padded on each side with 4 pixels before conducting a random  $32 \times 32$  crop. For evaluation, we report the error computed on the test images of original size.

### C. Evaluation on ImageNet

We compare our DSoP-Net with 4 categories of competing networks, including: (1) networks with quadratic transformation instead of just linear convolutions, such as FBN [39] and SORT [40]; (2) vanilla ResNet-50 trained with deeper or wider models in terms of knowledge distillation [23], which plays the role of student network jointly optimized with a teacher network, e.g. a modified ResNet-50 that uses stride=1 for all convolutional layers at stage 5, ResNet-101 and SE-ResNeXt-101; (3) architectures developed with fixed design such as SE-Net [36], and CBAM [41]; and (4) networks that use GSoP at the network end, such as MPN-COV [19] and iSQRT-OV [21]. We compare the results reported in the original papers for methods in categories (1), (3) and (4). We run the methods in category (2) with PyTorch official implementation of ResNet family models<sup>2</sup> and third-party<sup>3</sup> implementation of SE-ResNeXt-101, with which we achieved very close results to ResNet family models reported in literature [47]. Due to limited computational resources, we adapted the final size of a random image crop to  $256 \times 256$  pixels so that experiments can be conducted with a single set of 8-way GPU server;

<sup>2</sup><https://github.com/pytorch/vision>

<sup>3</sup><https://github.com/Cadene/pretrained-models.pytorch>

meanwhile, we reduced the mini-batch size from 1024 to 256 for SE-ResNeXt-101 as the original setting reported in [36] is highly in favor of the distributed training system. Accordingly, we reduced the initial learning rate from 0.6 to 0.15. Following the settings in [19], [21], to obtain higher resolution feature maps, we further changed the value of stride at stage 5 from 2 to 1. We use the original ResNet-50 architecture (stride=2) and the modified one (stride=1) as two baselines for fair comparison, and denote by ResNet-50-s2 and ResNet-50-s1 the two baselines, respectively.

Table II compares the performance of DSoP-Net with the state-of-the-arts on ImageNet. We have three observations articulated below. First, the proposed ResNet-50-s1 w/ DSoP-Net achieves top-1/5 error rates at 21.15%/5.70% on ImageNet, significantly outperforming all the competing methods. It is even better than those methods that explicitly use GSoP at the network end, including MPN-COV [19] and iSQRT-COV [21]. Second, with equivalent number of parameters used in training, ResNet-50-s2 w/ DSoP-Net outperforms joint training with a single deeper and/or wider teacher model. It shows that a student model is not able to well fuse the channel correlations and the spatial information without explicit modelling during training. However, it is hard to design metrics to directly measure the discrepancy of channel correlations between the student and the teacher models. In contrast, DSoP-Net is free of this issue and it allows us to easily transfer knowledge and expertise of second-order statistics to the student model. Third, DSoP-Net uses the same architecture as the two baseline models during inference, making it the most lightweight one among all competing methods. However, it can still obtain equivalent or even better performance than the heavier models, such as SE-Net [36].

We also study the performance of the latest state-of-the-art architectures with our DSoP-Net. EfficientNet-b0 [48] is selected as the backbone model. Without a TPU cluster in hand, we made a few changes of the original experimental settings to fit our own GPU server. Specifically, the mini-batch size is reduced from 2048 to 768, and the initial learning rate is accordingly set to 0.048 for RMSProp. First, we used a third-party implementation<sup>4</sup> and managed to achieve a top-1/5 accuracy of EfficientNet-b0 at 76.81%/93.32% on ImageNet with baseline ResNet preprocessing. The results we obtained are almost the same as the official TPU implementation<sup>5</sup> under similar settings (top-1 accuracy at 76.8%). Then, we replaced the original first-order pooling of EfficientNet-b0 with the second-order pooling structure by reducing the number of channels from 1280 to 64 with a sequence of conv1 × 1, BN and swish activation [49]. To prevent over-fitting, we inserted a dropout layer before the last linear classifier and set its dropout rate to 0.2. It turns out that the EfficientNet-b0 with second-order pooling obtains a top-1/5 accuracy at 77.02%/93.38%. Finally, we constructed the DSoP-Net for EfficientNet-b0 by inserting two auxiliary branches with second-order pooling. One was inserted at the middle of the backbone model and the other was inserted at the end. With DSoP-Net, the top-1/5

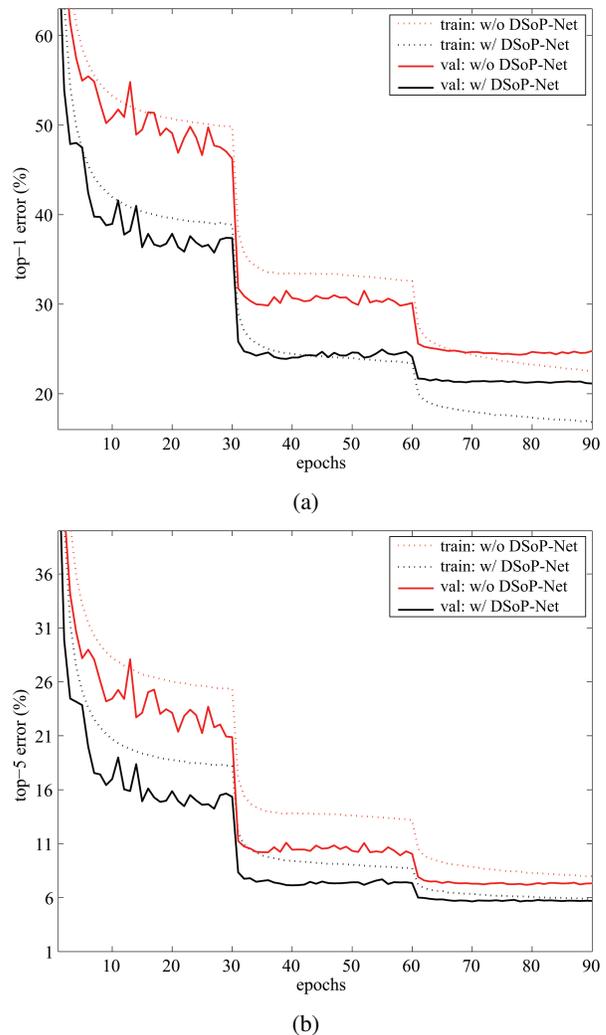


Fig. 4: Convergence curves regarding (a) top-1 and (b) top-5 error rates achieved by ResNet-50-s1 with and without DSoP-Net on ImageNet.

accuracy of the original first-order pooling head is further improved to 77.12%/93.56%. This validates the complementary nature of DSoP-Net to state-of-the-art CNN architectures such as EfficientNet.

Figure 4 compares the convergence curves of our DSoP-Net (output with GAP) and those of ResNet-50-s1. We can see that our DSoP-Net consistently outperforms ResNet-50-s1 by a large margin. This shows that knowledge acquired from the auxiliary classifiers with second-order pooling improves the original classifier with first-order pooling throughout the whole training process. Figure 5 presents the convergence curves of different output headers. We have three observations. First, the first-order output achieves very close performance to its teacher output in the last detachable branch. Second, output of a branch inserted at a later layer always performs better than that of a branch inserted at an earlier layer. The first-order output has even better results than those second-order outputs of the branches inserted at earlier layers. Third, compared to the original backbone model, the benefit gained

<sup>4</sup><https://github.com/rwightman/pytorch-image-models>

<sup>5</sup><https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet>

TABLE III: Comparison of error rates (%) achieved by different training policies with DSoP-Net on ImageNet.

Training policy	#Epoch	a_pds2 Top-1/5	a_pds3 Top-1/5	a_pds4 Top-1/5	a_pds5 Top-1/5	GAP Top-1/5
One-phase (default)	90	34.59/15.24	25.44/8.33	21.77/6.04	21.11/5.57	21.15/5.70
Two-phase	90+15	43.52/22.36	30.00/11.30	24.68/7.54	23.34/6.50	23.57/6.85

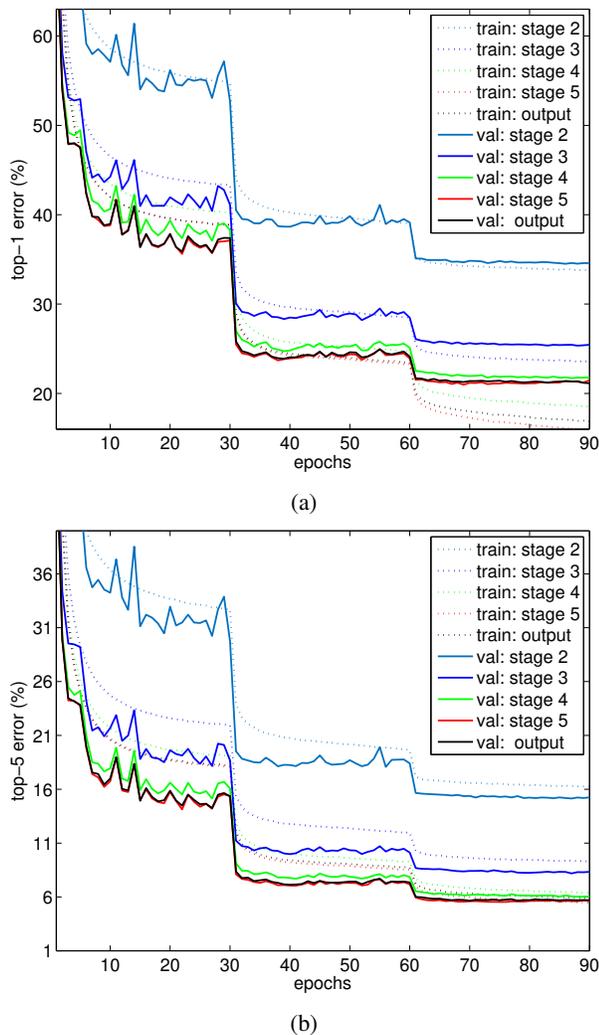


Fig. 5: Convergence curves regarding (a) top-1 and (b) top-5 error rates achieved by all output headers of DSoP-Net under ResNet-50 architecture on ImageNet, including the original one and the four obtained in the auxiliary branches.

by DSoP-Net during the very first epochs is significant while later the gap between the curves gradually reduces. This raises an interesting question regarding to the effect of early removal of auxiliary branches in DSoP-Net. We employed ResNet-50-s1 as the backbone and inserted auxiliary branches at the beginning of training on ImageNet. All experimental settings remain unchanged except that auxiliary branches are removed at epoch 30, 45, 60, respectively, to train the DSoP-Net. Experimental results show that the top-1/5 accuracies are decreased by 0.87%/0.31%, 0.41%/0.14% and 0.33%/0.11%,

respectively, compared to adopting auxiliary branches during the whole training. That is, early removal of auxiliary branches in DSoP-Net harms the final performance of the backbone model.

We continue to investigate the importance of engaging second-order statistics at earlier layers of first-order pooling networks. The default training policy of our DSoP-Net can be regarded as one-phase where first- and second-order statistics jointly help to update the weights of the backbone model. For comparison reasons, we define a two-phase training policy where first- and second-order statistics are independently computed in order. Specifically, we first attach the same 4 auxiliary branches, including a\_pds2,3,4,5, to the baseline ResNet-50 model (stride=1) after it converges under default training settings so that it has the same structure as DSoP-Net. Then, we proceed to fine-tune the parameters in auxiliary branches and fix those of the backbone model for 15 epochs on ImageNet. Learning rate starts at 0.1, and is reduced to 0.01 and 0.001 after 5 and 10 epochs, respectively. From Table III, one can see that the performance of the outputs at the 4 auxiliary branches achieved by the two-phase training policy can be largely improved by the default training policy. This suggests that optimizing networks simultaneously with first- and second-order statistics lead to better performance at earlier layers than computing the two kinds of statistics separately during training. Besides, with the default training policy of DSoP-Net, the strengthened intermediate layers help to improve the classification performance of the backbone model.

#### D. Evaluation on CIFAR-10 and CIFAR-100 Datasets

We further implement DSoP-Net with other modern backbone models to evaluate its generalization ability on CIFAR-10 and CIFAR-100 datasets, including two Pre-activation ResNet models [13], *i.e.* ResNet-110 and ResNet-164, two Wide Residual Networks (WRN) [37] models, *i.e.* WRN-22-10 and WRN-28-10, as well as two ResNeXt [15] models, *i.e.* ResNeXt-64-8 and ResNeXt-64-16. During training, for all models, we used SGD with momentum as optimizer and set the batch size as 128 and momentum as 0.9. For Pre-activation ResNet models, we set the weight decay as  $1 \times 10^{-4}$  and the number of training epochs as 110. The learning rate starts from 0.3, and it is divided by 10 at 80 and 95 epochs. For WRN models, we set the weight decay as  $5 \times 10^{-4}$  and the number of training epochs as 200. The learning rate begins from 0.1, and is divided by 5 at epochs 60, 120 and 160. For ResNeXt models, we set the weight decay as  $5 \times 10^{-4}$  and the number of training epochs as 300. The initial learning rate is 0.1, and is divided by 10 at 150 and 225 epochs. To construct auxiliary

TABLE IV: Summary of DSoP-Net with different backbone models on CIFAR-10 and CIFAR-100 datasets.

Backbone	#Block per stage	#Branch per stage	#Total branches	#Conv. per stage (Backbone+Branch)	Stage specification
ResNet-110	18	6	18	36+36	$[0,0,3] \times 6$
ResNet-164	18	6	18	54+54	$[0,0,3] \times 6$
WRN-22-10	3	2	6	6+6	$[1,0,2]$
WRN-28-10	4	2	6	8+8	$[0,2,0,2]$
ResNeXt-64-8	3	2	6	9+9	$[1,0,2]$
ResNeXt-64-16	3	2	6	9+9	$[1,0,2]$

branches for each model, definition of the same building block is reused at the stage where one auxiliary branch is attached. Meanwhile, we keep the total number of building blocks at one stage the same as that of the blocks used in the backbone model. Therefore, there are the same number of convolutional layers in the backbone model and its auxiliary branches.

Table IV summarizes some key statistics of DSoP-Net with different backbone models on CIFAR-10 and CIFAR-100 datasets. Take ResNet-110 as an example. We uniformly insert 18 branches with second-order pooling into the backbone network. The pattern of each stage is specified as  $[0, 0, 3] \times 6$ , where the non-zero values suggest the number of auxiliary convolutional layers inserted at the corresponding position while the zero values mean there are no auxiliary branches inserted at that position. To this end, there is a detachable branch at the end of every 3 bottlenecks; 6 auxiliary branches are attached to each stage; and there are  $6 \times 3 = 18$  auxiliary branches in total. For each branch, it adopts the same network structure as its associated 3 bottlenecks before the second-order pooling layer. As each original/auxiliary bottleneck depicts 2 convolutional layers,  $36 + 36 = 72$  convolutional layers are used at each stage during training. The target number of channels are set to 128 for all second-order pooling layers, including DSoP-Nets and iSQRT-COV [21] for comparison. For DR, we follow [19], [21] and use the direct DR operator. To avoid over-fitting, we incorporate a dropout layer with dropout rate as 0.5 before a dense layer in an auxiliary branch if 128 channels are used for second-order pooling. In this way, the image representation in auxiliary branches is  $2K-d$ ,  $4K-d$ , and  $4K-d$  for branches attached to Stage 1, Stage 2, and Stage 3, respectively.

To adapt to the CNN models on CIFAR-10 and CIFAR-100 datasets, we fixed the weight of original loss as  $\alpha = 1$  in Eqn. 1. Weights of losses connected with auxiliary outputs at Stage 1, Stage 2 and Stage 3 of all models are empirically set as 0.1, 0.2 and 0.3, respectively. This significantly helps to prevent gradient exposure accumulated at earlier layers in back-propagation. Meanwhile, this also strengthens the role of the original prediction of backbone model. We observed that all losses are basically on the same order of magnitude; thus, the desired loss of the original prediction given by the 1<sup>st</sup>-order pooling layer only takes up a very small portion ( $\sim 5\%$ ) if all components are equally weighted in the total loss, challenging the performance of the backbone model in inference. In contrast, with our implementation, loss of the original prediction actually takes up  $30\% \sim 50\%$  in the total loss to train more effective CNN model.

TABLE V: Comparison of error rates (%) achieved by different backbone models on CIFAR-10.

Backbone	w/o DSoP-Net (Original)	w/ DSoP-Net (Ours)	Gain
ResNet-110	6.37	5.73	0.64
ResNet-164	5.46	4.55	0.91
WRN-22-10	4.44	3.91	0.53
WRN-28-10	4.17	3.81	0.36
ResNeXt-64-8	3.65	3.44	0.21
ResNeXt-64-16	3.58	3.20	0.38

TABLE VI: Comparison of error rates (%) achieved by different backbone models on CIFAR-100.

Backbone	w/o DSoP-Net (Original)	w/ DSoP-Net (Ours)	Gain
ResNet-110	26.88	25.43	1.45
ResNet-164	24.33	21.06	3.27
WRN-22-10	20.75	18.91	1.84
WRN-28-10	20.50	18.50	2.00
ResNeXt-64-8	17.77	16.34	1.43
ResNeXt-64-16	17.31	16.23	1.08

To prevent over-fitting caused by auxiliary branches, a direct DR operator is inserted before the 2<sup>nd</sup>-order pooling layer to keep at most 128 channels if there are more channels for GSoP. Meanwhile, a dropout layer ( $p = 0.5$ ) is plugged between the second-order pooling layer and the dense layer in each branch. Consequently, the dimension of image representation produced by any auxiliary branch on CIFAR-10 and CIFAR-100 datasets is equal or less than  $128 \times (128+1) \times 0.5 \times 0.5 = 4K$ .

Table V and Table VI demonstrate the results achieved by different backbone models with and without DSoP-Net on CIFAR-10 and CIFAR-100 datasets. One can clearly see that models with DSoP-Net effectively outperforms the same architecture without DSoP-Net on both CIFAR-10 and CIFAR-100. Besides, it can be observed that DSoP-Net is more useful when training a narrow network. For example, the performance gain on CIFAR-10 for the narrowest model, ResNet, reaches  $0.64\% \sim 0.91\%$  while those for the WRN models, and the widest ResNeXt models, are reduced to  $0.36\% \sim 0.53\%$ , and  $0.21\% \sim 0.38\%$ , respectively.

### E. Ablation Study

1) *DSoP-Net*: We conduct ablation study of DSoP-Net from three aspects, as presented in detail as follows.

**Pooling methods in auxiliary branches.** We study the impact of different pooling methods in auxiliary branches.

TABLE VII: Comparison of error rates (%) achieved by different pooling methods in the auxiliary branches on ImageNet. ResNet-50-s1 is employed as the backbone model.

With DSoP-Net	Aux. Pool. Method	Order	Top-1/5
✗	N.A.	N.A.	23.57/6.85
✓	GAP [22] iSQRT-COV [21]	1 <sup>st</sup> 2 <sup>nd</sup>	22.74/6.61 <b>21.15/5.70</b>

TABLE VIII: Comparison of error rates (%) achieved by different number of channels used for second-order pooling at a\_pds2,3,4,5 on ImageNet. ResNet-18-s1 is employed as the backbone model. An asterisk symbol in superscript indicates a DR operation before 2<sup>nd</sup>-order pooling.

With DSoP-Net	Target #Chnl	#Chnl	GFLOPs / #Param.(M)	Top-1/5
✗	N.A.	N.A.	3.06/11.7	30.11/10.78
✓	64	64,64*,64*,64*	5.69/32.7	29.19/10.12
	128	64,128,128*,128*	5.95/51.3	28.88/9.92
	256	64,128,256,256*	7.26/100.6	<b>28.69/9.85</b>

ResNet-50-s1 is employed as the backbone model and we modify the network architecture of auxiliary branches during training. All layers used for channel reduction are removed and the second part is changed from iSQRT-COV to GAP while the other parts remain unchanged. We train the backbone model with the modified auxiliary branches from scratch with the default training strategy. We compare it to default architecture defined in DSoP-Net under the same backbone model. Table VII presents the results. Compared with the baseline, we can see that the introduction of auxiliary branches with first-order pooling method improves top-1 error rates by 0.8%. If we use the covariance matrix based second-order pooling as the second part of an auxiliary branch, it further boosts the result by nearly 1.4%. It shows that the second-order channel correlations are difficult to be directly modelled by first-order pooling such as GAP. Instead, knowledge and expertise from the second-order pooling layer in an auxiliary branch are effective to adjust the spatial responses in the backbone model.

**Number of channels for GSoP.** Covariance matrix as an

TABLE IX: Comparison of top-1 error rates by different optimizers (%).

(a) ResNet-20 on CIFAR-10.

Optimizer	w/o DSoP-Net	w/ DSoP-Net
SGD [45]	7.92	<b>7.59</b>
ADAHESIAN [50]	<b>7.87</b>	7.70
RAdam [51]	8.62	8.07

(b) ResNet-18-s1 on ImageNet.

Optimizer	w/o DSoP-Net	w/ DSoP-Net
SGD [45]	30.11	<b>28.69</b>
ADAHESIAN [50]	<b>29.86</b>	29.55
RAdam [51]	32.12	31.29

image representation quadratically increases the computational complexity of the second-order pooling method. It also affects the number of parameters of the dense layer in an auxiliary branch. To study the impact, in each auxiliary branch, we decrease the number of channels for second-order pooling from 256 to 128 and 64, respectively. ResNet-18 (stride=1), denoted by ResNet-18-s1, is employed as the backbone model. Table VIII compares the results by using different number of channels. One can see that the total number of parameters used in training decreases sharply from 100.6M to 32.7M with only 64 channels. It reduces the computational complexity by approximately 21.6%. Meanwhile, top-1/5 error rates increase moderately by 0.5% and 0.27%. Even so, DSoP-Nets still reduces the top-1 error rate by almost 1% with the same backbone inference architecture.

**Insertion points of the auxiliary branches.** To study the impact of different insertion points, we employ ResNet-18 model as backbone and fix the total number of ResNet bottlenecks used in all auxiliary branches as 16. Specifically, we remove one or more auxiliary branches based on the default architecture of DSoP-Net under the ResNet-18-s1 architecture (please refer to Table I), where each auxiliary branch contains four ResNet bottlenecks. We uniformly distribute the bottlenecks to the remaining branches. Table X presents the results. One can see that performance of first-order pooling can be improved by inserting more branches at different stages. With the same number of branches, plugging branches at the layers closer to the first-order pooling benefits performance boost at the price of more parameters and computations in training, but not during the inference stage.

**Choice of optimizer.** It has been shown [52], [53] that SGD is a very stable and effective network optimizer for various CNN architectures, especially on large scale datasets such as ImageNet. It is interesting to investigate whether other optimizers can further improve the performance of DSoP-Net than SGD. We test two recent optimizers: ADAHESSIAN [50] and Rectified Adam (a.k.a. RAdam) [51]. We use the official implementation of ADAHESSIAN<sup>6</sup> and RAdam<sup>7</sup>. In the experiments, we opt to SGD [45] as the baseline and evaluated ResNet-20 on CIFAR-10 and ResNet-18-s1 on ImageNet, which are widely used to compare different optimizers (especially those designed for CNN models). We use the same hyper parameter settings reported in the original paper of each optimizer, including weight decay, initial learning rate, total epochs, learning rate schedule, etc.

Table IX summarizes the results. One can have the following three observations. First, for each optimizer, model trained with DSoP-Net outperforms the same model trained without DSoP-Net on both CIFAR-10 and ImageNet. This shows that our method is effective with different optimizers. Second, compared to the baseline optimizer, DSoP-Net slightly reduces the gap between RAdam and SGD on CIFAR-10 from 0.70% to 0.48%. However, the gap between RAdam and SGD on ImageNet is enlarged from 2.01% to 2.60%. Third, ADAHESSIAN results in better performance than SGD for the original

<sup>6</sup><https://github.com/amirgholami/adahessian>

<sup>7</sup><https://github.com/LiyuanLucasLiu/RAdam>

TABLE X: Comparison of error rates(%) achieved by different insertion points with ResNet-18 on ImageNet.

#Conv. Layers in a_conv2,3,4,5_x	With DSoP-Net	#Branch	GFLOPs	#Param. (M)	Top-1	Top-5
0,0,0,0	✗	N.A.	3.06	11.7	30.11	10.78
16,0,0,0	✓	1	4.94	94.8	29.09	10.04
0,0,0,16	✓		11.23	127.4	28.88	9.92
8,8,0,0	✓	2	4.34	95.4	29.06	10.03
8,0,0,8	✓		8.47	108.7	29.00	9.98
0,0,8,8	✓		8.50	112.0	28.77	9.90
4,4,4,4	✓	4	7.26	100.6	28.69	9.85

TABLE XI: Comparison of error rates (%) achieved by different DR methods with ResNet-50-s1 on ImageNet in reducing the 2048 channels.

Method	Target #Channel	$\delta$	w/ Aux. Branches	Top-1	Top-5
Direct DR [19], [21]	64	N.A.	N.A.	23.73	6.99
	128	N.A.	N.A.	22.78	6.43
	256	N.A.	N.A.	22.14	6.22
PSDR (Ours)	64	0.25	✗	23.30	6.86
	128	0.25	✗	22.18	6.09
	64	0.5	✗	23.24	6.72
	128	0.5	✗	22.02	6.15
	256	0.5	✗	21.84	5.99
	64	0.5	✓	22.71	6.42
	128	0.5	✓	21.89	6.11
	256	0.5	✓	<b>21.65</b>	<b>6.00</b>

CNN models on both datasets, but it is not as good as SGD for models with DSoP-Net. We believe that the performance loss is caused by the approximation of the Hessian matrix as a diagonal operator. Such an approximation may be suitable for the sequential models (such as the vanilla CNN models), but for multiple-header models like DSoP-Net, it is critical to consider the correlations among parameters in the backbone model and those in the auxiliary model. As a result, some second-order cues in the auxiliary branches may be implicitly lost during the optimization with ADAHESSIAN, and the performance becomes worse than the DSoP-Net optimized with SGD. Besides, SGD is friendly to computational resources in terms of both memory and overhead, and hence it is easier to apply to deeper or wider CNN architectures.

2) *PSDR*: We employ ResNet-50-s1 as the backbone model in the ablation study of PSDR. In addition, we focus on the parameter selection of  $\delta$  in Eqn. 4 and the introduction of auxiliary branches for DR. Table XI demonstrates the results achieved by the direct DR operator adopted in [19], [21] and the proposed PSDR in reducing the 2048 channels of the last convolutional layer. It can be observed that PSDR outperforms direct DR when the same reduction ratio is applied. A larger  $\delta$  allows more intermediate layers, leading to a clear performance boost. Meanwhile, introduction of auxiliary branches enables transfer of second-order statistics from high-dimension domain to low-dimension domain, which also improves the DR operation. Particularly, PSDR significantly improves the direct DR when the given 2048 channels are reduced to 64

channels. We achieve a top-1/5 error rate of 22.71%/6.42%, even better than when keeping 128 channels with direct DR.

## VI. CONCLUSION

We proposed a novel method which significantly improves the performance of first-order CNNs in image classification. Auxiliary branches were carefully designed to transfer knowledge to the backbone first-order networks during training, which are however removable at the testing stage. As a result, the proposed method leverages the advantages of second-order pooling networks while keeping similar complexity to first-order networks during inference. To the best of our knowledge, this is the first attempt to make use of higher-order statistics in knowledge distillation. Experiments conducted on ImageNet as well as CIFAR-10 and CIFAR-100 datasets demonstrated the effectiveness of our network. In particular, we achieved a top-1 error rate of 21.15% with single center-crop using ResNet-50 network. In the future, we will adapt DSoP-Net to solve more tasks, including object detection, image segmentation, etc.

## ACKNOWLEDGMENT

This work is partially supported by Hong Kong RGC GRF project (PolyU 152135/16E) and NSF of China (under grant number 61971086). The authors would like to thank the reviewers for their insightful comments.

## REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NeurIPS*, 2012.
- [2] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.
- [3] Z. Zhao, P. Zheng, S. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–21, 2019.
- [4] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.
- [5] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard, "The megaface benchmark: 1 million faces for recognition at scale," in *Proc. CVPR*, 2016.
- [6] B. Cao, N. Wang, J. Li, and X. Gao, "Data augmentation-based joint learning for heterogeneous face recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 6, pp. 1731–1743, June 2019.
- [7] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, "Person re-identification by multi-channel parts-based CNN with improved triplet loss function," in *Proc. CVPR*, 2016.

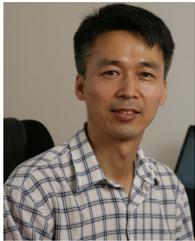
- [8] H. Zhao, M. Tian, S. Sun, J. Shao, J. Yan, S. Yi, X. Wang, and X. Tang, "Spindle Net: Person re-identification with human body region guided feature decomposition and fusion," in *Proc. CVPR*, 2017.
- [9] X. Yang, P. Zhou, and M. Wang, "Person reidentification via structural deep metric learning," *IEEE Trans. Neural Netw. Learn. Syst.*, 2018.
- [10] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva, "Sun database: Exploring a large collection of scene categories," *Int. J. Comput. Vis.*, vol. 119, no. 1, pp. 3–22, 2016.
- [11] J. Li, X. Mei, D. Prokhorov, and D. Tao, "Deep neural network for structural prediction and lane detection in traffic scene," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 690–703, March 2017.
- [12] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, 2018.
- [13] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. ECCV*. Springer, 2016.
- [14] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, Inception-Resnet and the impact of residual connections on learning," in *AAAI*, 2017. [Online]. Available: <https://arxiv.org/abs/1602.07261>
- [15] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. CVPR*. IEEE, 2017.
- [16] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear CNN models for fine-grained visual recognition," in *Proc. ICCV*, 2015.
- [17] T.-Y. Lin and S. Maji, "Improved bilinear pooling with CNNs," in *Proc. BMVC*, 2017.
- [18] Q. Wang, P. Li, and L. Zhang, "G<sup>2</sup>DeNet: Global gaussian distribution embedding network and its application to visual recognition," in *Proc. CVPR*, 2017.
- [19] P. Li, J. Xie, Q. Wang, and W. Zuo, "Is second-order information helpful for large-scale visual recognition?" in *Proc. ICCV*, 2017.
- [20] T. Lin, A. RoyChowdhury, and S. Maji, "Bilinear convolutional neural networks for fine-grained visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1309–1322, 2018.
- [21] P. Li, J. Xie, Q. Wang, and Z. Gao, "Towards faster training of global covariance pooling networks by iterative matrix square root normalization," in *Proc. CVPR*, 2018.
- [22] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. ICLR*, 2014. [Online]. Available: <https://arxiv.org/abs/1312.4400>
- [23] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
- [24] S. Gupta, J. Hoffman, and J. Malik, "Cross modal distillation for supervision transfer," in *Proc. CVPR*, 2016.
- [25] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. CVPR*, 2017.
- [26] S. Cai, W. Zuo, and L. Zhang, "Higher-order integration of hierarchical convolutional activations for fine-grained visual categorization," in *Proc. CVPR*, 2017.
- [27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015. [Online]. Available: <https://arxiv.org/abs/1409.1556>
- [28] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. CVPR*, 2017.
- [29] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," in *Proc. CVPR*, 2017.
- [30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR*, 2015.
- [31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015. [Online]. Available: <https://arxiv.org/abs/1502.03167>
- [32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. CVPR*, 2016.
- [33] C. Ionescu, O. Vantzos, and C. Sminchisescu, "Matrix backpropagation for deep networks with structured layers," in *Proc. ICCV*, 2015.
- [34] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, "Compact bilinear pooling," in *Proc. CVPR*, 2016.
- [35] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *Proc. ICML*, 2018.
- [36] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. CVPR*, 2018. [Online]. Available: <https://arxiv.org/abs/1709.01507>
- [37] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. BMVC*, 2016. [Online]. Available: <https://arxiv.org/abs/1605.07146>
- [38] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, "Geometric means in a novel vector space structure on symmetric positive-definite matrices," *SIMAX*, vol. 29, no. 1, pp. 328–347, 2007.
- [39] Y. Li, N. Wang, J. Liu, and X. Hou, "Factorized bilinear models for image recognition," in *Proc. ICCV*, 2017.
- [40] Y. Wang, L. Xie, C. Liu, S. Qiao, Y. Zhang, W. Zhang, Q. Tian, and A. L. Yuille, "SORT: Second-order response transform for visual recognition," in *Proc. ICCV*, 2017.
- [41] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "CBAM: Convolutional block attention module," in *Proc. ECCV*, 2018.
- [42] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *Proc. NeurIPS-W*, 2017.
- [43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*. IEEE, 2009.
- [44] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.
- [45] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. ICML*, 2013.
- [46] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. ECCV*. Springer, 2016.
- [47] A. Mishra and D. Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," in *Proc. ICLR*, 2018.
- [48] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Prof. ICML*, 2019. [Online]. Available: <https://arxiv.org/abs/1905.11946>
- [49] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.
- [50] Z. Yao, A. Gholami, S. Shen, K. Keutzer, and M. W. Mahoney, "ADA-HESSIAN: An adaptive second order optimizer for machine learning," *arXiv preprint arXiv:2006.00719*, 2020.
- [51] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *Proc. ICLR*, 2020. [Online]. Available: <https://arxiv.org/abs/1908.03265>
- [52] N. S. Keskar and R. Socher, "Improving generalization performance by switching from Adam to SGD," *arXiv preprint arXiv:1712.07628*, 2017.
- [53] L. Luo, Y. Xiong, Y. Liu, and X. Sun, "Adaptive gradient methods with dynamic bound of learning rate," in *Proc. ICLR*, 2019. [Online]. Available: <https://arxiv.org/abs/1902.09843>



**Lida Li** received the B.S. and M.Sc. degrees from the School of Software Engineering, Tongji University, Shanghai, China, in 2013 and 2016, respectively. He is now pursuing his Ph.D. degree at the Department of Computing, The Hong Kong Polytechnic University. His research interests are machine learning and face analysis.

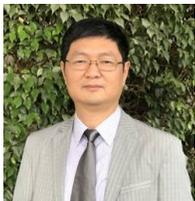


**Jiangtao Xie** is a Ph.D. candidate of School of Information and Communication Engineering, Dalian University of Technology. His research interests include computer vision and deep learning. He has published several papers in top tier international journal IEEE TPAMI and top conferences of CVPR/ICCV/NIPS. As a key member of DLUT\_VLG team, he achieved 1/59 in iNaturalist Challenge 2018 at Fine-Grained Visual Categorization (FGVC) 5 in conjunction with CVPR2018.



**Peihua Li** received Ph.D degree from Harbin Institute of Technology in 2003, Harbin, China, and then worked for one year as a postdoctoral fellow at INRIA/IRISA, France. He achieved the honorary nomination of National Excellent Doctoral dissertation in China. He is currently a professor of Dalian University of Technology, Dalian, China. He was supported by Program for New Century Excellent Talents in University of Chinese Ministry of Education. His team won the first place in large-scale iNaturalist Challenge spanning 8000 species at FGVC5

CVPR2018, second place in Alibaba Large-scale Image Search Challenge 2015, and fourth place in Noisy Iris Challenge Evaluation I. His research topics include deep learning, computer vision and pattern recognition, focusing on image/video recognition, object detection and semantic segmentation. He has published papers in top journals such as IEEE TPAMI/TIP/TCSVT and top conferences including ICCV/CVPR/ECCV.



**Lei Zhang** (M'04, SM'14, F'18) received his B.Sc. degree in 1995 from Shenyang Institute of Aeronautical Engineering, Shenyang, P.R. China, and M.Sc. and Ph.D degrees in Control Theory and Engineering from Northwestern Polytechnical University, Xian, P.R. China, in 1998 and 2001, respectively. From 2001 to 2002, he was a research associate in the Department of Computing, The Hong Kong Polytechnic University. From January 2003 to January 2006 he worked as a Postdoctoral Fellow in the Department of Electrical and Computer Engineering,

McMaster University, Canada. In 2006, he joined the Department of Computing, The Hong Kong Polytechnic University, as an Assistant Professor. Since July 2017, he has been a Chair Professor in the same department. His research interests include Computer Vision, Image and Video Analysis, Pattern Recognition, and Biometrics, etc. Prof. Zhang has published more than 200 papers in those areas. As of 2020, his publications have been cited more than 57,000 times in literature. Prof. Zhang is a Senior Associate Editor of *IEEE T-IP*, and is/was an Associate Editor of *IEEE T-PAMI*, *SIIMS*, *IEEE T-CSVT*, and *IVC*, etc. He is a "Clarivate Analytics Highly Cited Researcher" from 2015 to 2020. More information can be found in his homepage <http://www4.comp.polyu.edu.hk/~cszhang/>.