Network Decoupling: From Regular to Depthwise Separable Convolutions

Jianbo Guo[1]
jianboguo@outlook.com
Yuxi Li[2]
lyxok1@sjtu.edu.cn
Weiyao Lin[2]
hellomikelin@gmail.com
Yurong Chen[3]
yurong.chen@intel.com
Jianguo Li[3]
jianguo.li@intel.com

[1] Institute for Interdisciplinary Information Sciences, Tsinghua University, China
[2] Shanghai Jiaotong University, China
[3] Intel Labs China

Abstract

Depthwise separable convolution has shown great efficiency in network design, but requires time-consuming training procedure with full training-set available. This paper first analyzes the mathematical relationship between regular convolutions and depthwise separable convolutions, and proves that the former one could be approximated with the latter one in closed form. We show depthwise separable convolutions are principal components of regular convolutions. And then we propose network decoupling (ND), a training-free method to accelerate convolutional neural networks (CNNs) by transferring pre-trained CNN models into the MobileNet-like depthwise separable convolution structure, with a promising speedup yet negligible accuracy loss. We further verify through experiments that the proposed method is orthogonal to other training-free methods like channel decomposition, spatial decomposition, etc. Combining the proposed method with them will bring even larger CNN speedup. For instance, ND itself achieves about 2× speedup for the widely used VGG16, and combined with other methods, it reaches 3.7× speedup with graceful accuracy degradation. We demonstrate that ND is widely applicable to classification networks like ResNet, and object detection network like SSD300.

1 Introduction

Convolutional neural networks (CNNs) demonstrate great success in various computer vision tasks, such as image classification [18], object detection [7], image segmentation [24], etc. However, they suffer from high computation cost when deployed to resource-constrained devices. Many efforts have been devoted to optimize/accelerate the inference speed of CNNs, which could be roughly divided into three categories. *First*, design-time network optimization considers designing efficient network structures from scratch in a handcraft way or automatic search way. Typical handcraft based works include Xception [4], MobileNet [13],

and networks with channel interleaving/shuffle [30, 32], while typical works on automatic network architecture search are NASNet [33], PNASNet [20].

Second, training-time network optimization takes pre-defined network structures as input, and refines the structures through regularized retraining or fine-tuning or even knowledge distilling [12]. Typical works involve weight pruning [8, 9], structure (filters/channels) pruning [19, 22, 25, 29], weight hashing/quantization [1], low-bit networks [3, 26].

Third, deploy-time network optimization takes pre-trained CNN models as input, and replaces some redundant and less-efficient CNN structures with efficient ones in a training-free way. Low-rank decomposition [5], spatial decomposition [14], and channel decomposition [31] fall into this category.

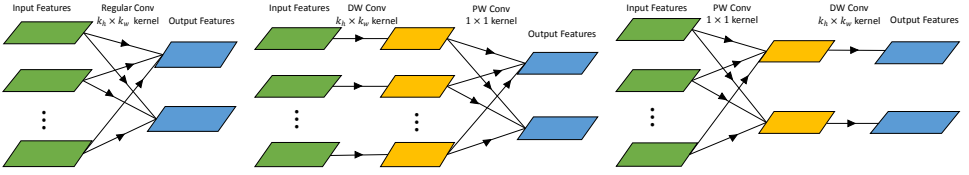
Methods in the first two categories require time-consuming training procedure to produce desired outputs, with full training-set available. On the contrary, methods in the third category may not require training-set at all, or in some cases require a small calibration-set (e.g., 5,000 images) to tune some parameters. The optimization procedure can typically be done within dozens of minutes. Hence, it is of great value when software/hardware vendors assist their customers to optimize CNN based solutions in case that either the time budget is so tight that training based solutions are not feasible, or the customer data are unavailable due to privacy or confidential issues. Therefore, there is a strong demand for modern deep learning frameworks or hardware (GPU/ASIC/FPGA, etc) vendors to provide deploy-time model optimization tools.

Meanwhile, handcraft designed structures such as depthwise separable convolution [2, 13, 32] have shown great efficiency over regular convolution, while still keeping high accuracy. To the best of our knowledge, the mathematical relationship between regular convolutions and depthwise separable convolutions is not yet studied and unknown to the public. Our motivation is to show their relationship, and present a solution to decouple the regular convolutions into depthwise separable convolutions in a training-free way for deploy-time network optimization/acceleration. Our main contributions are summarized as below:

- We are the first to analyze and disclose the mathematical relationship between regular convolutions and depthwise separable convolutions. This theoretic result enables a lot of possibilities for future studies.
- We present a closed-form and data-free tensor decomposition to decouple regular convolutions into depthwise separable convolutions, and show that network decoupling (ND) enables noticeable speedup for different CNN models, such as VGG16 [28], ResNet [10], as well as object detection network SSD [21].
- We demonstrate that ND is complementary to other training-free methods like channel decomposition [31], spatial decomposition [14], and channel pruning [11]. For instance, network decoupling itself achieves about 1.8× speedup for VGG16. When combined with other training-free methods, it achieves 3.7× speedup.
- We show extremely decoupled network is friendly to fine-tuning. The extremely decoupled network will bring more speedup but larger accuracy drop. For instance, 4× speedup will bring >50% accuracy drop, while 2× speedup has <1% drop. We show the larger accuracy drop can be recovered with just several epochs of fine-tuning.

2 Related Work

Here we only discuss related works on deploy-time network optimization. Low-rank decomposition [5] exploits low-rank nature within CNN layers, and shows that fully-connected



(a) Regular Convolution (b) DW+PW Convolution (c) PW+DW Convolution

Figure 1: Regular convolutions vs. depthwise separable convolutions. (a) Regular convolution; (b) depthwise separable convolution in the DW+PW form (depthwise followed by pointwise); (c) depthwise separable convolution in the PW+DW form.

layers can be efficiently compressed and accelerated with low-rank decomposition, while convolutional layers can not. Spatial decomposition [14] takes a single channel filter as input, and do per-channel spatial decomposition on regular convolution for each input/output channel, i.e., factorizing the $k_h \times k_w$ filter into $1 \times k_w$ and $k_h \times 1$ filters.

Channel decomposition [31] decomposes one conv-layer into two conv-layers, where the first one has the same filter-size but with fewer channels, and the second one is a 1×1 convolution. Channel pruning [11] develops a training free method to prune useless filter channels by minimizing the response reconstruction error with a small-size calibration set.

Our motivation is different from all these methods, since we consider the possibility to decompose regular convolutions into depthwise separable convolutions.

3 Theory

Our key insight is that different filter channels in regular convolutions are strongly coupled, and may involve plenty of redundancy. Our analysis shows that this coupling induced redundancy is corresponding to some kind of low-rank assumption, with similar spirit of [5, 14, 31]. Here, we first analyze and disclose the mathematical relationship between regular convolutions and depthwise separable convolutions.

3.1 Regular vs. Depthwise Separable Convolutions

A regular convolution kernel (Figure 1a) is tasked to build both cross-channel correlation and spatial correlations. Formally, we consider a convolution layer represented by a 4D tensor $\mathbf{W} \in \mathbb{R}^{n_o \times n_i \times k_h \times k_w}$, where n_o and n_i are the number of output and input channels respectively, and k_h and k_w are the spatial height and width of the kernel respectively. When the filter is applied to an input patch $\mathbf{x}$ with size $n_i \times k_h \times k_w$, we obtain a response vector $\mathbf{y} \in \mathbb{R}^{n_o}$ as

$$\mathbf{y} = \mathbf{W} * \mathbf{x}, \tag{1}$$

where $y_o = \sum_{i=1}^{n_i} W_{o,i} * x_i, o \in [n_o], i \in [n_i]$, and $*$ means convolution operation. $W_{o,i} = \mathbf{W}[o,i,:,:]$ is a tensor slice along the i-th input and o-th output channels, $x_i = \mathbf{x}[i,:,:]$ is a tensor slice along the i-th channel of 3D tensor $\mathbf{x}$. And the computational complexity for patch $\mathbf{x}$ is $O(n_o \times n_i \times k_h \times k_w)$. It is easy to extend the complexity from patch level to feature map level. Given the feature map size $H \times W$, the complexity is $O(H \times W \times n_o \times n_i \times k_h \times k_w)$.

Compared with the regular convolution, a depthwise separable convolution consists of a depthwise (DW) convolution followed by a pointwise (PW) convolution, where DW focuses on spatial relationship modeling with 2D channel-wise convolutions, and PW focuses on cross-channel relationship modeling with 1×1 convolution across channels. This factorization form, denoted by DW+PW, is shown in Figure 1b. To ensure the same shape output as the regular convolution $\mathbf{W}$, we set the DW convolution kernel tensor $\mathbf{D} \in \mathbb{R}^{n_i \times 1 \times k_h \times k_w}$, and

the PW convolution tensor $\mathbf{P} \in \mathbb{R}^{n_o \times n_i \times 1 \times 1}$. When applying it to the input patch $\mathbf{x}$, we can obtain the corresponding response vector $\mathbf{y}'$ as

$$\mathbf{y}' = (\mathbf{P} \circ \mathbf{D}) * \mathbf{x}, \tag{2}$$

where $y'_o = \sum_{i=1}^{n_i} P_{o,i}(D_i * x_i)$, $\circ$ is the compound operation, $P_{o,i} = \mathbf{P}[o,i,:,:]$ and $D_i = \mathbf{D}[i,:,:,:]$. And the computational complexity for the whole feature map is $O(H \times W \times (n_i \times k_h \times k_w + n_i \times n_o))$.

Alternatively, we could put PW convolution before DW and obtain another factorization form PW+DW as shown in Figure 1c. In this case, $\mathbf{P} \in \mathbb{R}^{n_o \times n_i \times 1 \times 1}$, $\mathbf{D} \in \mathbb{R}^{n_o \times 1 \times k_h \times k_w}$. When it is applied to the input patch $\mathbf{x}$, the response vector y'' is

$$\mathbf{y}'' = (\mathbf{D} \circ \mathbf{P}) * \mathbf{x}, \tag{3}$$

where $y''_o = D_o * \left(\sum_{i=1}^{n_i} P_{o,i} x_i\right)$, and $D_o = \mathbf{D}[o,:,:,:]$. Here, the computational complexity is $O(H \times W \times (n_i \times n_o + n_o \times k_h \times k_w))$. It is obvious that DW+PW and PW+DW are more efficient than regular convolutions according to the computational complexity.

3.2 Relationship

We have shown regular convolutions model the spatial correlation and cross-channel correlation simultaneously with one tensor kernel, while depthwise separable convolutions model these two correlations in a decoupling way. Is there any relationship between these two convolution formulations? Is it possible to approximate regular convolutions with depthwise separable convolutions precisely? We give the following theorem to answer these questions.

Theorem 1. *Regular convolutions can be losslessly expanded to a sum of several depthwise separable convolutions, without the increase of computational complexity. Formally,* $\forall$ $\mathbf{W}$ *with spatial kernel size* $k_h \times k_w$, $\exists$ $\{\mathbf{P}^k, \mathbf{D}^k\}_{k=1}^K$,

$$s.t.\ \ (a) K \leq k_h k_w;$$
$$(b) \mathbf{W} = \begin{cases} \sum_{k=1}^K \mathbf{P}^k \circ \mathbf{D}^k & \textit{for DW+PW} \\ \sum_{k=1}^K \mathbf{D}^k \circ \mathbf{P}^k & \textit{for PW+DW.} \end{cases} \tag{4}$$

Proof. This problem is similar to the Kronecker product decomposition problem [16, 23], which factorizes a tensor into a linear combination of tensor/matrix Kronecker products. We adopt similar techniques to prove the above theorem. As DW+PW and PW+DW cases are similar, for simplicity, we only discuss the DW+PW case below.

In the DW+PW case, $\mathbf{D}^k \in \mathbb{R}^{n_i \times 1 \times k_h \times k_w}$ and $\mathbf{P}^k \in \mathbb{R}^{n_o \times n_i \times 1 \times 1}$. Given an input patch $\mathbf{x}$, the response difference between regular convolution and DW+PW convolution is

$$\begin{aligned} \|\mathbf{y} - \mathbf{y}'\|_2^2 &= \|(\mathbf{W} - \sum_{k=1}^K \mathbf{P}^k \circ \mathbf{D}^k) * x\|_2^2 = \sum_{o=1}^{n_o} (\sum_{i=1}^{n_i} W_{o,i} * x_i - \sum_{k=1}^K \sum_{i=1}^{n_i} P_{o,i}^k (D_i^k * x_i))^2 \\ &\leq \sum_{o=1}^{n_o} \sum_{i=1}^{n_i} ((W_{o,i} - \sum_{k=1}^K P_{o,i}^k D_i^k) * x_i)^2 \leq \sum_{o=1}^{n_o} \sum_{i=1}^{n_i} \|W_{o,i} - \sum_{k=1}^K P_{o,i}^k D_i^k\|_F^2 \cdot \|x_i\|_F^2, \end{aligned}$$

where $P_{o,i}^k = \mathbf{P}^k[o,i,:,:]$, $D_i^k = \mathbf{D}^k[i,:,:,:]$ are tensor slices, and $\|\cdot\|_F$ is the Frobenius norm. For the rightmost term,

$$\sum_{o=1}^{n_o} \sum_{i=1}^{n_i} \|W_{o,i} - \sum_{k=1}^K P_{o,i}^k D_i^k\|_F^2 = \sum_{i=1}^{n_i} \|\widetilde{W}_{:,i} - \sum_{k=1}^K \widetilde{P_{:,i}^k} \widetilde{D_i^k}^\mathsf{T}\|_F^2,$$

where $\widetilde{W}_{:,i} = \widetilde{\mathbf{W}}[:,i,:]$ is a slice of the reshaped tensor $\widetilde{\mathbf{W}} \in \mathbb{R}^{n_o \times n_i \times (k_h k_w)}$ from $\mathbf{W}$, $\widetilde{D_i^k} = Vec(D_i^k)$ is the vector view of D_i^k, and $\widetilde{P_{:,i}^k} = \mathbf{P}^k[:,i,:,:]$ is a tensor fiber. $\widetilde{W}_{:,i}$ can be viewed as a matrix of size $n_o \times (k_h k_w)$ with $rank(\widetilde{W}_{:,i}) \leq \min\{n_o, k_h k_w\}$.

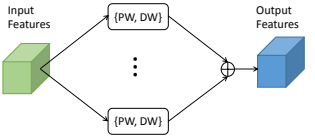


Figure 2: Approximating regular convolution by the sum of depthwise separable convolutions. {PW, DW} can be either PW+DW or DW+PW.

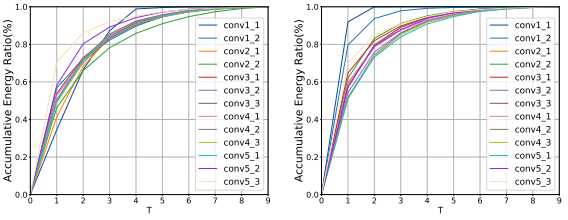


Figure 3: Accumulative energy ratio in the DW+PW case (left) and in the PW+DW case (right).

Suppose $\widetilde{W}_{:,i} = USV^\intercal$ is the singular value decomposition. Let $\widetilde{P^k_{:,i}} = U_k S(k)$ where U_k is the k-th column of U and $S(k)$ is the k-th singular value, $\widetilde{D^k_i} = V_k$ where V_k is the k-th column of V. When we set $K = \max_i rank(\widetilde{W}_{:,i})$, the rightmost term equals to zero. And then $\|W_{o,i} - \sum_{k=1}^{K} P^k_{o,i} D^k_i\|_F^2 = 0$. Hence $\mathbf{W} = \sum_{k=1}^{K} \mathbf{P}^k \circ \mathbf{D}^k$. As $n_o \gg k_h k_w$ generally holds, $K = \max_i rank(\widetilde{W}_{:,i}) \leq \min\{n_o, k_h k_w\}$. Therefore, $K \leq k_h k_w$ holds.

The computational complexity of this expansion is thus $O(K \times H \times W \times (n_i \times k_h k_w + n_i \times n_o))$, where $H \times W$ is the resolution of current feature map. The computing cost ratio of the regular convolution to this expansion is $r = {k_h k_w}/{K(k_h k_w/n_o + 1)}$. As $n_o \gg k_h k_w$, $r \approx {k_h k_w}/{K}$. Since $K \leq k_h k_w$, $r \geq 1$ holds. That means the lossless expansion does not increase the computational complexity over the regular convolution. □

4 Network Decoupling

Theorem 1 actually presents a closed-form tensor decomposition to decouple regular convolutions into depthwise separable convolutions. We name this solution as *exact network decoupling*, and K as the *decoupling rank* of $\mathbf{W}$ which reflects the coupling induced redundancy in $\mathbf{W}$. When K is low, for instance, $K = 1$, there is significant redundancy in $\mathbf{W}$, so that *exact network decoupling* can bring great computation cost reduction. When $K = k_h k_w$, there is no redundancy, and hence no benefit with *exact network decoupling*.

4.1 Approximated Network Decoupling

For less redundant CNN layers, the *exact network decoupling* may bring unsatisfied speedup. Due to the decomposition nature, the energy is not equally distributed among the K depthwise separable convolution (DSC) blocks. In fact, substantial energy is concentrated in a fraction of those K DSC blocks. Hence, we can realize the *approximated network decoupling* for better CNN speedup based on the following corollary deduced from Theorem 1.

Corollary 1. *Given a regular convolution tensor* $\mathbf{W}$ *with spatial kernel size* $k_h \times k_w$*, we can approximate it with top-*T *(*$\leq K$*) depthwise separable convolutions as*

$$\mathbf{W} \approx \begin{cases} \sum_{k=1}^{T} \mathbf{P}^k \circ \mathbf{D}^k & \text{for DW+PW} \\ \sum_{k=1}^{T} \mathbf{D}^k \circ \mathbf{P}^k & \text{for PW+DW}, \end{cases} \tag{5}$$

and the acceleration over the original regular convolution is $k_h k_w / T$.

Figure 2 illustrates our depthwise separable convolution approximation to the regular convolution. Note that the proposed network decoupling does not require any training data. We will further study the possibility to combine it with other existing training-free methods for even larger CNN speedup in Section 4.2.

How good is the approximated decoupling? Let's take VGG16 for example, in which $k_h k_w = 9$ for all the convolutional layers. For the DW+PW case, we compute the singular values of $\widetilde{W}_{:,i}$ ($i \in [n_i]$), and average the ratio of the square sum of the top-T largest singular values to the total square sum. The same thing is done for PW+DW. Figure 3 plots the average energy ratio for both cases. We can see that substantial energy is from several top singular vectors. For example, in both cases, the top-4 singular vectors contribute over 90% energy in all the layers except the conv2_2 layer in DW+PW case. Especially, in the conv1_2 layer by PW+DW, the top-4 singular vectors account for over 99% energy. This indicates that we can only use a fraction of depthwise separable convolutions to precisely approximate the original regular convolutions. Note that energy in PW+DW is more concentrated than that in DW+PW, which means PW+DW can realize the same quality approximation with fewer DSC blocks, and thus yields a better speedup. One possible reason is that for the layer with tensor kernel $n_i \times k_w \times k_h \times n_o$, the PW+DW case will produce n_o separate DW channels, while the DW+PW case only has n_i DW channels. When $n_o > n_i$, the PW+DW case will have more parameters than the DW+PW case so that PW+DW may have better approximation. We will verify this result by experiments later.

4.2 Complementary Methods

ND focuses on decomposing a regular convolution into a sum of T depthwise separable convolutions. The method not only has a closed-form solution, but also is training-data free.

Besides our work, there are some works considering network decomposition from different perspectives, like channel decomposition [31], spatial decomposition [14], and channel pruning [11]. All these methods require a small calibration dataset (for instance 5000 images from 1.2 million ImageNet images for ImageNet models) to reduce possible accuracy loss. Different from these methods, the proposed network decoupling does not involve any channel reduction and spatial size reduction, which implies our method should be complementary to them. Hence we propose to combine network decoupling with these methods to further accelerate deep CNN models. Additionally, this combination even provides us the possibility to reduce the accumulated error with the calibration set.

Let's take the combination of ND and channel decomposition as an example. For a layer with tensor kernel $n_i \times k_w \times k_h \times n_o$, we first apply CD to decompose it into two layers, where the first layer has tensor kernel $n_i \times k_w \times k_h \times d$, and the second layer is 1×1 conv-layer with tensor kernel $d \times 1 \times 1 \times n_o$. CD sets $d < n_o$ to ensure acceleration. We then apply ND to the layer with kernel $n_i \times k_w \times k_h \times d$ to decouple it into one point-wise convolution layer and one depthwise convolution layer. We process the next layer in the original network with the same procedure. As is known, CD will minimize the reconstruction error of the responses between original networks and decomposed one with a small calibration set. When apply CD to next layer, it will compensate the accumulated error somewhat from two successive approximations in previous layer. This procedure is adopted sequentially until all the layers in the original network are processed. We will show in the experiments that the combined solution can bring significantly better CNN model acceleration than any single methods.

5 Experiments

We conduct extensive experiments to evaluate the proposed method on different network structures such as VGG16 [28] and ResNet18 [10] pre-trained on ImageNet [2] with Caffe

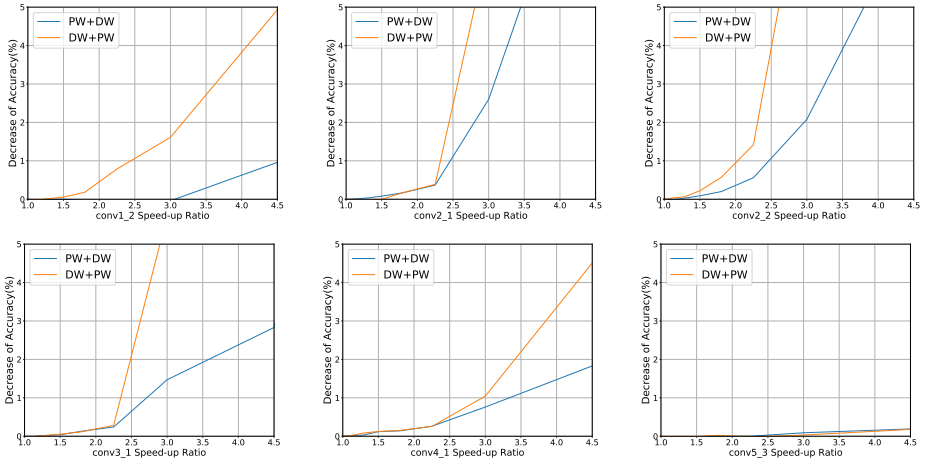


Figure 4: **DW+PW vs. PW+DW** decoupling for VGG16: single layer performance under different speedup ratios, measured by decrease of top-5 accuracy on ImageNet (*smaller is better*). The speedup ratios are computed by the theoretical complexity of that layer.

[15]. The top-5 accuracy (measured by single center-crop test) of VGG16 is 88.66% with 15.35G FLOPs, and ResNet18 is 88.09% with 1.83G FLOPs. We also evaluate the proposed method for the object detection framework SSD300 [21] (with VGG16 as backbone) on PASCAL VOC 2007 benchmark [6]. We further study how fine-tuning can help extremely decoupled networks. All these studies are conducted **without** fine-tuning unless specified.

5.1 Single Layer Decoupling

We first evaluate the single layer acceleration performance using our network decoupling method. In this study, we decouple one given layer with all the remaining layers unchanged. The speedup ratio reported only involves that single layer, which is shown as the theoretical ratio computed by the complexity (FLOPs). Figure 4 illustrates the speedup vs. the accuracy drop for different layers of VGG16 under both the DW+PW and the PW+DW decoupling.

We can see that when speeding up a single layer by 2×, the accuracy drop is rather marginal or negligible, especially in the PW+DW case. In this case, there is no accuracy drop for layer conv1_2 and conv5_3, and less than 0.5% accuracy loss for all other layers. It also shows that PW+DW decoupling consistently outperforms the DW+PW case. This result can be explained by Figure 3: to accumulate the same energy, PW+DW decoupling needs smaller T, hence better acceleration. In the later experiments, we only use PW+DW decoupling. We also find that decoupling brings less speedup for the intermediate layers compared with shallower and deeper (front and end) layers. It implies shallower and deeper layers have much more redundancy. This property is different from channel decomposition [31] and channel pruning [11], where redundancy concentrates only in the shallower layers. This verifies network decoupling and channel decomposition/pruning are intrinsically complementary, and we can combine our decoupling with them to speed up CNNs further.

5.2 Whole Model Decoupling of VGG16

5.2.1 Experiments on Single Methods

Next, we evaluate the acceleration performance of PW+DW decoupling on the whole VGG16 model. We sequentially approximate the layers involved (conv1_2 to conv5_3). The first

Method	FLOPs	top-1 drop (%)
Original VGG16	15.35G	0
CD[31]	6.52G	2.10
SD[14]	7.20G	1.96
CP[11]	9.89G	1.68
Ours	8.61G	1.55

(a) Single methods

Method	FLOPs	
	without ND	with ND (ND+X)
CD [31]	6.52G	4.72G
SD [14]	7.20G	**4.15G**
CP [11]	9.89G	8.49G
CD+SD [31]	4.32G	4.28G
CD+CP	7.16G	5.45G
CD+SD+CP [11]	4.92G	4.70G

(b) Combined methods

Table 1: Acceleration performance on VGG16 with (a) single method and (b) combined method. "with ND" means the experiments are conducted in combination with our network decoupling (ND). The number of FLOPs is computed by the theoretical complexity of the approximated model. Except for original VGG16, all the other models are tuned with fixed 1.0% top-5 accuracy drop. (a) also lists the corresponding top-1 accuracy drops.

layer conv1_1 is not decoupled, since it only contributes 0.3% computation. Guided by the single layer experiments above, we decouple more aggressively for both shallower layer and deeper layer, by using smaller T. The computing cost (in FLOPs) of the decoupled model is reported with fixed 1% top-5 accuracy drop, where the corresponding top-1 accuracy drops range from 1.5% to 2.1%. As shown in Table 1a, our decoupling method can reduce about 45% of the total FLOPs, which demonstrates significant redundancy inside VGG16.

We also compare our decoupling method with other design-time network optimization methods like channel decomposition (CD) [31], spatial decomposition (SD) [14] and channel pruning (CP) [11], which are recent state-of-the-art training/fine-tuning free solutions for CNNs acceleration. The comparison is based on the re-implementation of [11, 31]. Note that all these three compared methods require a small portion of training set (i.e., 5000 images) in their optimization procedure. Even though, our method still outperforms the data-driven based channel pruning as shown in Table 1a, which indicates that decoupling convolutions is promising for CNN acceleration. Although our method performs somewhat worse than CD and SD, we should emphasize that our method is totally data-free, while all these three are data-driven methods. Moreover, we will show below that our network decoupling combined with these methods will bring state-of-the-art CNN acceleration in the training-free setting.

5.2.2 Experiments on Combined Methods

As discussed in Section 4.2 and 5.1, our network decoupling is intrinsically complementary to channel decomposition, spatial decomposition and channel pruning. In this part, we will study the performance of combined methods. We not only test the performance of combining our decoupling with each of the above methods separately, but also conduct the experiments of the existing compound methods with/without our decoupling scheme. Note that these combined models require a small portion of training dataset for data-driven based optimization. For fair comparison, we randomly pick up 5000 images out of ImageNet training set (1.2 million images) and use them for all the evaluated methods.

Table 1b shows the results. Clearly, combined with our network decoupling, each of the above methods has significant improvement, which verifies that our decoupling exploits a new cardinality. In the best result, our combined method (ND+SD) could reduce about 73% of the original FLOPs (3.7× speedup), with only 1% top-5 accuracy drop.

Interestingly, we find that CD+CP performs worse than CD alone (FLOPs increase from 6.52G to 7.16G). We speculate that channel pruning and channel decomposition are not complementary to some extent. They both exploit the inter-channel redundancy and reduce the number of channels. Furthermore, both CD and CP captures more redundancy in the shallower layers according to [11, 31]. Therefore, their combination may yield conflicts

Method	top-5 Accuracy	FLOPs
Original ResNet18	88.09	1.83G
CD [31]	83.69	1.32G
SD [14]	85.20	1.25G
Ours	86.68	1.20G

Table 2: Training-free acceleration for ResNet18.

Method	mAP	FLOPs
Original SSD300	82.29	31.37G
CD [31]	80.99	18.69G
Ours	81.41	20.27G
Ours+CD	80.71	15.01G

Table 3: Training-free acceleration for SSD300.

between them. Compared with CD and CP, our network decoupling can capture redundancy from both shallower layers and deeper layers (see Figure 4). Hence, we can enhance both of them. This result also holds for the combination with spatial decomposition.

5.3 Decoupling ResNet

Modern networks like ResNet [10] are designed for both efficiency and high accuracy, which are usually not easy to accelerate in the training-free setting. In this part, we evaluate network decoupling on ResNet18, which has a VGG16 comparable top-5 accuracy (88.09) but with much lower computation cost (1.83G FLOPs).

Table 2 shows that our network decoupling alone can reduce about 34% of total FLOPs (1.5× speedup) with 1.4% drop of top-5 accuracy. This result is not as graceful as that of VGG16, since modern structures tend to have less redundancy by design. As a comparison, CD is not a data-free solution, which only reduces 28% total FLOPS (1.36× speedup), and brings 4.4% top-5 accuracy drop. Other methods like [11] handle the accuracy drop with a limited epoch fine-tuning. We will explore the benefit of this solution later.

5.4 Decoupling SSD300

Object detection suffers from even higher computing cost than image classification due to its relatively high-resolution input. We evaluate our network decoupling for one of widely used object detection framework SSD300 [21] on the PASCAL VOC 2007 [6]. The backbone of SSD300 is based on VGG16, which is popular in many object detection frameworks [27]. The performance is measured by mean Average Precision (mAP) and total FLOPs of the detection networks.

Different from [11, 31], we extract backbone from the pre-trained SSD model, decompose the filters inside the backbone, and then use the approximated backbone to take the detection task, where no fine-tuning is involved. From Table 3, we observe that network decoupling alone achieves 35% FLOPs reduction with mAP drop less than 1.0%, which is acceptable in most scenes. Further, if combining our approach with other training-free methods like channel decomposition, we could at most reduce the FLOPs to 48% of the original model (2.1× speedup) with only 1.58% loss in mAP, which is also acceptable. Note although the backbone is based on VGG16, SSD300 has different speedup performance from that of the VGG16 classification network. This is because SSD300 changes the model parameters of backbone network during training procedure due to different target loss functions and different inputs & resolutions, so that detection backbone has less redundance than that of classification network. Similar phenomena have been observed by [11, 31].

5.5 Extremely decoupling with Fine-tuning

Here we study the possibility of combining network decoupling with fine-tuning for even better speedup like [11, 17]. We first perform extremely or aggressively network decoupling, which will usually bring large accuracy drop. We then fine-tune the extremely decoupled

Model	Epochs	top-5 Accuracy	Speedup
Original VGG16	100	88.66	1.00
ND (auto-tuned T)	0	87.66(−1.00)	1.8×
ND(T=2)+VGG16	10	88.10(−0.56)	3.9×
ThiNet [25]	32	88.14(−0.52)	3.3×
CP+finetune [11]	10	87.66(−1.00)	4.0×
Original ResNet18	100	88.09	1.00
ND (auto-tuned T)	0	86.68(−1.41)	1.5×
ND(T=3)+ResNet18	6	88.22(+0.13)	2.0×

Table 4: Results of fine-tuning of extremely decoupled VGG16 (top part) and ResNet18 (bottom part). Digits in the bracket show accuracy change compared with original models. "ND+" means we apply our method to aggressively decouple the original model and fine-tune it for given epochs. On VGG16, we also list results by other two fine-tuning based network optimization methods as comparison.

models with initial learning rate 0.0001, and decrease the learning rate 1/10 every 4 epochs. The results are shown in Table 4.

We aggressively decouple VGG16 with $T = 2$, which yields a model with top-5 accuracy 19.4%. We recover the top-5 accuracy of this model to 88.1% (-0.56% to baseline) with just 10 epochs of fine-tuning, yielding 3.9× speedup (3.96G FLOPs). We aggressively decouple ResNet18 with $T = 3$, which yields a model with top-5 accuracy 29.2%. We recover the top-5 accuracy to 88.22% (+0.13% to baseline) with just 6 epochs of fine-tuning, yielding 2.1× speedup (0.89G FLOPs). Note that the number of epochs used here is less than $1/10$ of the training from scratch solutions. In comparison, the fine-tuning-free network decoupling only provides 1.8× speedup for VGG16 with top-5 accuracy 87.66%, and 1.5× speedup for ResNet18 with top-5 accuracy only 86.68%. It is obvious that extremely decoupling plus fine-tuning provides not only better speedup but also much higher accuracy.

Besides, we compare our results with the state-of-the-art training-time network optimization methods (with fine-tuning on VGG16). ThiNet [25] is a pure fine-tuning based network optimization method, which requires 32 epochs of fine-tuning to obtain 3.3× speedup with 0.52% top-5 accuracy drop. It shows that our method achieves better speedup over ThiNet (3.9 vs. 3.3), while with much fewer fine-tuning epochs (10 vs. 32). Channel pruning(+fine-tuning) [11] requires 10 epochs of fine-tuning to obtain about 4× speedup with 1.0% top-5 accuracy drop. It shows that our method achieves similar speedup (3.9 vs. 4.0), while with less accuracy drop (0.52% vs. 1.0%).

6 Conclusion

This paper analyzes the mathematical relationship between regular convolutions and depth-wise separable convolutions, and proves that the former one can be approximated with the latter one precisely. We name the solution *network decoupling* (ND), and demonstrate its effectiveness on VGG16, ResNet as well as object detection network SSD300. We further show that ND is complementary to existing training-free methods, and can be combined with them for splendid acceleration. ND could be an indispensable module for deploy-time network optimization, as well as provides theoretic supports for possible future studies.

Acknowledgement: Jianbo Guo is supported in part by the National Basic Research Program of China Grant 2015CB358700, the NSFC Grant 61772297, 61632016, 61761146003. Weiyao Lin is supported in part by the NSFC Grant 61471235 and the Shanghai "The Belt and Road" Young Scholar Grant (17510740100). Jianguo Li is the corresponding author.

References

[1] Wenlin Chen, James Wilson, et al. Compressing neural networks with the hashing trick. In *ICML*, 2015.

[2] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[3] M. Courbariaux and Y. Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. In *ICLR*, 2016.

[4] J. Deng, W. Dong, et al. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[5] Emily Denton, Zaremba, et al. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014.

[6] Mark Everingham, Luc Van Gool, et al. The pascal visual object classes (voc) challenge. *IJCV*, 88(2), 2010.

[7] Ross Girshick, Jeff Donahue, et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[8] Song Han, Jeff Pool, et al. Learning both weights and connections for efficient neural network. In *NIPS*, 2015.

[9] Song Han, Huizi Mao, and Bill Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *NIPS*, 2016.

[10] K. He, X. Zhang, et al. Deep residual learning for image recognition. In *CVPR*, 2016.

[11] Y. He, X. Zhang, and Others. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017.

[12] G. Hinton, O. Vinyals, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[13] Andrew G Howard, Menglong Zhu, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[14] M. Jaderberg, A. Vedaldi, et al. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

[15] Yangqing Jia, Evan Shelhamer, et al. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, 2014.

[16] B. Kim and N. Wong. A constructive arbitrary-degree kronecker product decomposition of tensors. *Numerical Linear Algebra with Applications*, 24(5), 2017.

[17] Y. Kim, E. Park, et al. Compression of deep convolutional neural networks for fast and low power mobile applications. In *ICLR*, 2016.

[18] A. Krizhevsky and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[19] Hao Li, Asim Kadav, et al. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[20] C. Liu, B. Zoph, et al. Progressive neural architecture search. *arXiv preprint arXiv:1712.00559*, 2017.

[21] Wei Liu, Dragomir Anguelov, et al. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[22] Zhuang Liu, Jianguo Li, et al. Learning efficient convolutional networks through network slimming. *arxiv preprint*, 1708, 2017.

[23] CF Van Loan. The ubiquitous kronecker product. *Journal of computational and applied mathematics*, 123, 2000.

[24] Jonathan Long, Evan Shelhamer, et al. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[25] J. Luo, J. Wu, et al. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017.

[26] M. Rastegari, V. Ordonez, et al. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016.

[27] S. Ren, K. He, et al. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE TPAMI*, 39(6), 2017.

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

[29] W. Wen, C. Wu, et al. Learning structured sparsity in deep neural networks. In *NIPS*, 2016.

[30] T. Zhang, G. Qi, et al. Interleaved group convolutions. In *ICCV*, 2017.

[31] X. Zhang, J. Zou, et al. Accelerating very deep convolutional networks for classification and detection. *IEEE TPAMI*, 38(10), 2016.

[32] X. Zhang, X. Zhou, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.

[33] B. Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.