Advances in Artificial Intelligence and Machine Learning; Research 3 (4) 1699-1719 Received 11-09-2023; Accepted 28-11-2023; Published 05-12-2023

Depth-wise Decomposition for Accelerating Separable Convolutions in Efficient Convolutional Neural Networks

Yihui He yihuihe.yh@gmail.com
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Jianing Qian jianingq@alumni.cmu.edu
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Cindy X. Le xl2738@columbia.edu
Columbia University
New York, NY 10027, USA

Congrui Hetang congruihetang@gmail.com
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Qi Lyu lyuqi1@msu.edu
Michigan State University
East Lansing, MI 48824, USA

Wenping Wang wenpingw@alumni.cmu.edu
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Tianwei Yue tyue@alumni.cmu.edu
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Corresponding Author: Congrui Hetang

Abstract

Very deep convolutional neural networks (CNNs) have been firmly established as the primary methods for many computer vision tasks. However, most state-of-the-art CNNs are large, which results in high inference latency. Depth-wise separable convolution has been proposed for image recognition tasks on platforms with limited computation power, such as robots and self-driving cars. Any regular deep CNN has a depth-wise separable counterpart, which is faster, but less accurate, when equally trained. In this paper, we propose a novel decomposition approach based on SVD, namely depth-wise decomposition, for converting regular convolutions into depth-wise separable convolutions post-training, while maintaining high accuracy. We show that our approach generalizes to the multi-channel and multi-layer cases, by applying Generalized Singular Value Decomposition (GSVD). We conduct thorough experiments with the ShuffleNet V2 model on a large-scale image recognition dataset: ImageNet. Our approach outperforms the baseline, channel decomposition. Moreover, our approach improves the Top-1 accuracy of ShuffleNet V2 by ~2%.

Keywords: Computer vision, Efficient convolutional neural networks, Neural network acceleration, Neural network compression.

1. INTRODUCTION

In recent years, very deep convolutional neural networks (CNNs) [1–3], have led to a series of breakthroughs in many image understanding problems [4–8], such as image recognition [9–13], object detection [14–22], semantic segmentation [23–26], and tracking [26–28]. However, most state-of-the-art CNNs have very high inference latency, limiting their applications on platforms with tight computational budget, such as smartphones, wearable systems [29, 30], surveillance cameras [31, 32], and self-driving cars [33, 34].

Driven by the increasing need for faster CNN models, research focus has been moving towards reducing CNN model size and computation cost while achieving acceptable accuracy instead of purely pursuing very high accuracy. One trend is to train CNNs with efficient architectures. For example, MobileNets [35], proposed a family of lightweight convolutional neural networks based on depth-wise separable convolution. ShuffleNets [36, 37], proposed channel shuffle to reduce the parameter number and FLOPs (floating point operations per second). Pushing further, ShiftNet [38], proposed shifting feature maps as an alternative to spatial convolution. The other trend is to compress large regular CNN models after training, shown in FIGURE 1 (a). For example, channel decomposition [39], and spatial decomposition [40], proposed to decompose regular convolutional layers, shown in FIGURE 1 (e) and FIGURE 1 (d). Channel pruning [41–43], proposed to prune channels of convolutional layers, shown in FIGURE 1 (c). Deep compression [44, 45], proposed to sparsify the connections of fully-connected layers, shown in FIGURE 1 (b).

Representative compact CNN model designs mentioned above share a common component: depth-wise separable convolution, initially introduced by [46]. As is pointed out in MobileNets [35], 3×3 depth-wise separable convolutions use between 8 to 9 times less computation than standard convolutions, with the same number of input and output channels. However, the accuracy is inevitably sacrificed. Although many efforts have been put on searching optimal hyper-parameters for the compact models [3, 47], it remains a question whether we could improve the performance of depth-wise separable convolution as it is.

Motivated by this, in this paper, we propose a method for mitigating the performance degradation of depth-wise separable convolution. Inspired by channel decomposition work [39], which decomposes a standard CNN post-training to a lighter version without much degradation (shown in FIGURE 1 (e)), we propose to decompose a regular convolution into a depth-wise separable convolution post-training, while minimizing the performance degradation. Specifically, we start from a ”heavy” CNN (created by replacing all depth-wise separable convolutions in a ”light” convolutional neural network, like ShuffleNet V2 [36], with regular convolutions). The ”heavy” CNN is trained from scratch on ImageNet. Then, the regular convolutions are decomposed by our method into depth-

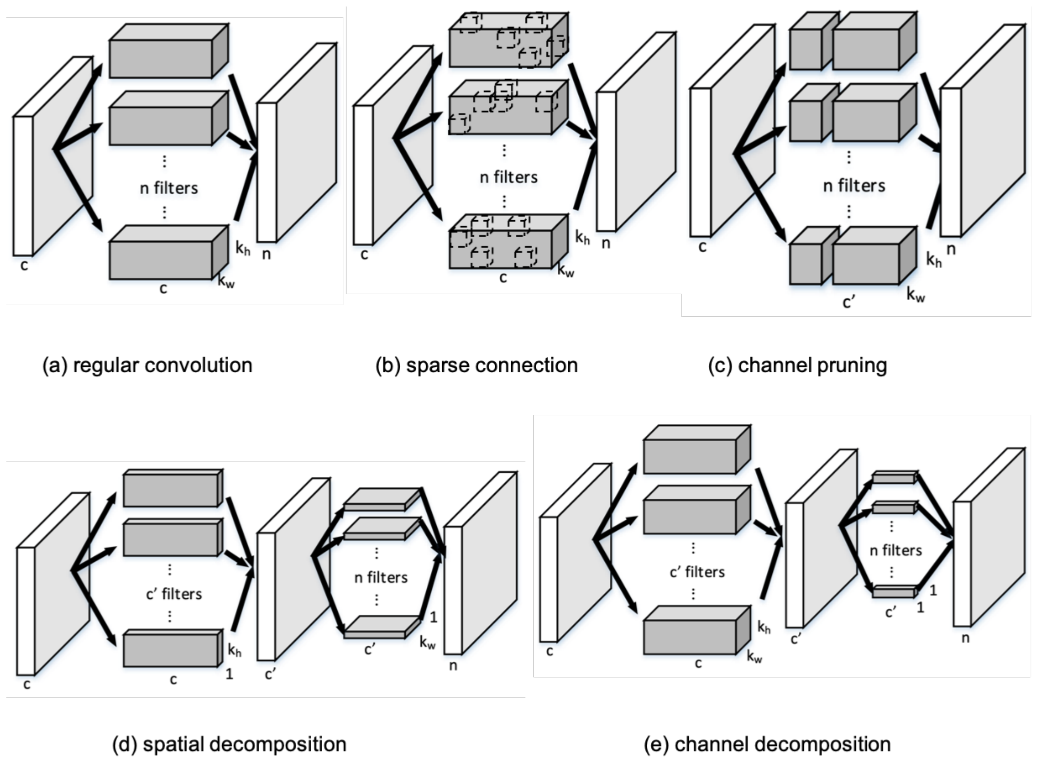


Figure 1: Different types of compression algorithms.

wise separable convolutions. This produces the same original ”light” architecture, yet we show it outperforms the ”light” CNN trained from scratch with the same data.

To show the generality of our approach, we conduct rich experiments on ShuffleNet V2 and Xception with the large-scale ImageNet dataset. With equally high speed-up ratio (9 times), our approach consistently outperforms the baseline, channel decomposition [39], in terms of accuracy, with different setups including single/multiple channels/layers. Moreover, a CNN with ShuffleNet V2 [36], architecture acquired by our decomposition method achieves $\sim 2\%$ better Top-1 accuracy than the original ShuffleNet V2.

We summarize our contributions as follow:

1) We propose a novel decomposition approach, namely Depth-wise Decomposition, for converting regular convolution into depth-wise separable convolution.

2) Our approach achieves a better speed/accuracy trade off than channel decomposition.

3) Our approach brings 2% Top-1 accuracy improvement to the original ShuffleNet V2 model.

2. RELATED WORK

Since LeCun *et al*. introduced ”optimal brain damage [48, 49]”, there has been a significant amount of works on accelerating CNNs [50]. Many of them fall into several categories: designing efficient architectures [35, 36], optimized implementation [51], quantization [52], and structured simplification [40].

2.1 Designing Efficient Architecture

Depth-wise separable convolution is widely used in efficient networks like MobileNets [35], ShuffleNets [37], and MnasNets [3]. [53] proved that a regular convolution could be approximated by a combination of several depth-wise separable convolutions. ShiftNets [38] proposed to use shift operation as an alternative to 3-by-3 convolution. AddressNets [54, 55] proposed three shift-based primitives for further improving performance on GPUs (Graphics Processing Units).

2.2 Sparse Connection

Shown in FIGURE 1 (b), connection pruning eliminates connections between neurons [56–60]. XNOR-Net [52], binarized the connections. [61] prunes connections based on weights magnitude. Deep compression [44], could accelerate fully connected layers up to 50×, in theory. However, in practice, the actual speed-up largely depends on implementation. The standard library like CUDNN [62], highly optimizes large dense matrx multiplication, limiting the headroom for sparcifying network connections.

2.3 Channel Pruning

Shown in FIGURE 1 (c), channel pruning aims at removing inter-channel redundancies of feature maps. There were several training-based approaches: [63–65], regularize networks to improve accuracy. Channel-wise SSL [64], reaches high compression ratio for the first few convolutional layers of LeNet [66], and AlexNet [67]. [65] could work well for fully connected layers. However, training-based approaches are more costly, and their effectiveness on very deep networks on large datasets is rarely exploited.

Inference-time channel pruning is challenging, as reported by previous works [68, 69]. Channel pruning [41], proposed to prune neural networks layer-by-layer using LASSO regression and linear least square reconstruction. Some works [70–73], focus on model size compression, which mainly operate the fully connected layers. Data-free approaches [74, 75], results for speed-up ratio (*e.g.*, 5×) have not been reported, and requires long retraining procedure. [75] select channels via over 100 random trials. However, it needs a long time to evaluate each trial on a deep network, which makes it infeasible to work on very deep models and large datasets. AMC [76, 77], improves general channel pruning approach by learning the speed-up ratio with reinforcement learning.

2.4 Tensor Decomposition

Tensor factorization methods [40, 78–80], aim to approximate the original convolutional layer weights with several pieces of decomposed weights. Consider a convolutional layer with 3×3 kernel size. As shown in FIGURE 1 (d), spatial decomposition [40], factorizes it into a 3×1 and 1×3 combination, driven by spatial feature map redundancy. As shown in FIGURE 1 (e), channel decomposition [39], factorizes it into the combination of a 3×3 layer with less output channels, and a 1×1 layer that restores the original channel number, driven by channel-wise feature map redundancy. [81–83] accelerate fully connected layers with truncated SVD.

Aforementioned methods first train regular neural networks, then decomposes them into lighter versions. It's straightforward to consider not doing so, but directly training the light version from scratch. In [39], this has been empirically verified to perform worse. This motivates us to decompose regular convolution into depth-wise separable convolution post-training, which may improve the performance of popular compact neural networks building on top of depth-wise separable convolution.

2.5 Implementation-Based Acceleration

Though convolution is a well-defined operation, the run-time can vary a lot depending on implementations. Optimized implementation methods [51, 84–87], accelerate convolution, with special convolution algorithms like FFT [85]. Quantization [52, 88], reduces floating point computational complexity, which is usually followed by fine-tuning and Huffman coding [44]. BinaryNet [88, 89], proposed to binarize both connections and weights. Recently, HAQ [90], automated this process, which further compressed deep neural networks. These methods also depend on the hardware and library implementation.

3. APPROACH

In this section, we first briefly review how a regular convolution layer and a depth-wise separable convolution layer works (Section 3.1). Then we introduce channel decomposition (Section 3.2), which our method is built upon. Following that, we describe our method in detail. We start from the special one-layer, single-input-channel case (Section 3.3), then generalize it to the multi-input-channel case (Section 3.4 3.5). Finally, we cover the multi-layer case, which is used to decompose an entire neural network (Section 3.6).

3.1 Regular and Depth-wise Separable Convolutions

FIGURE 2 (a) illustrates how a regular convolutional layer works. It takes a feature map X with shape $[c, h, w]$ as input, then applies a bank of filters W shaped $[n, c, k, k]$ to produce the output feature map Y, shaped $[n, h, w]$. Here, c and n are the number of input/output channels, respectively; h and w are feature map height/width in pixels; k is filter size. The coloring of the figure shows which input channel each filter is applied to, and how each output channel is produced.

The filters slide across the feature map X and process it patch-by-patch. Therefore, a regular convolutional layer can be seen as a matrix multiplication:

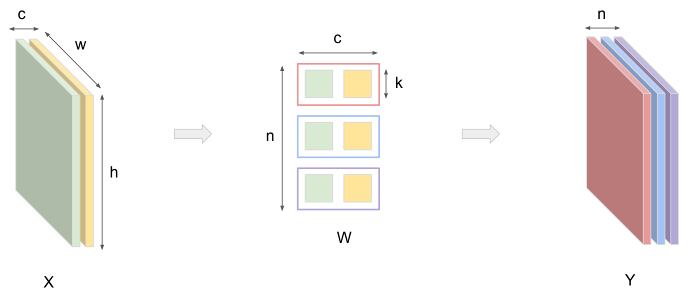
$$\begin{aligned} Y &= WX \\ [n, N] &= [n, (c, k, k)] \times [(c, k, k), N]; \end{aligned} \tag{1}$$

The shape of each tensor is annotated underneath for clarity. Here, N is the total number of patches. The dimensions in $(\cdot)$ are combined into one, so it's a matrix multiplication of $[n, ck^2] \times [ck^2, N]$. We rely on this notation to present our method. For simplicity, we assume filters are square, and omit paddings and bias.

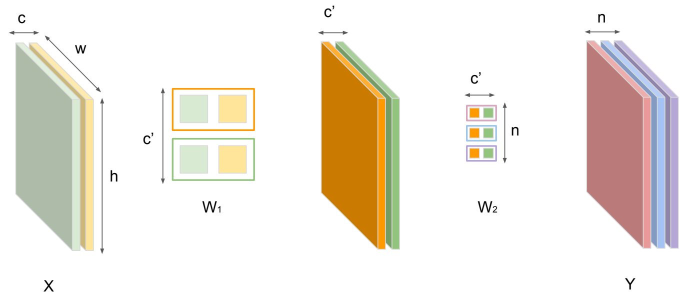
FIGURE 2 (c) illustrates the depth-wise separated version of FIGURE 2 (a). The regular convolutional layer is decomposed into a depth-wise convolutional layer with $[k, k]$ filter size, and a point-wise layer with $[1, 1]$ filter size. In the depth-wise layer, each of the c filters is applied to its own input channel, and produces one output channel (notice that this differs from a regular convolution). The point-wise layer then projects the c-channel feature map into the original n output channels. It's straightforward that the depth-wise separated version needs much less computation. The goal of our method is to decompose already-trained regular convolutions into depth-wise separable ones, achieving speed-up while maintaining accuracy.

3.2 A Brief Introduction of Channel Decomposition

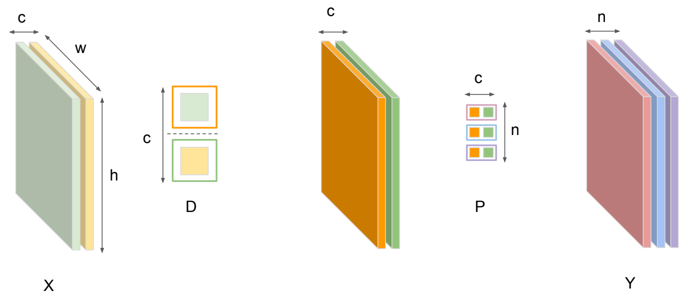
FIGURE 2 (b) illustrates channel decomposition, where a regular convolutional layer is decomposed into a smaller regular convolution layer (with fewer c' output channels) and a point-wise layer that restores the original n output channels. As described in [39], this is viable because regular CNNs has redundancy in their channels. Formally, Y mostly resides on a lower-dimension manifold:



(a) A regular convolutional layer.



(b) Channel decomposition.



(c) Depth-wise decomposition.

Figure 2: Illustration of a regular convolution and its decompositions. Best viewed in color. The colors of the filters indicate which channel they are applied to. The colors of the filter groups indicate which channel they produce.

$$\begin{aligned} Y &\approx MY \\ [n, N] &\approx [n, n] \times [n, N]; \\ \text{s.t. } &\text{rank}(M) \leq c' \end{aligned} \tag{2}$$

We assume that Y has its mean subtracted. Finding the optimal M equivalents to the following:

$$\begin{aligned} &\arg\min_{M} \|Y - MY\|; \\ &\text{s.t. } \text{rank}(M) \leq c' \end{aligned} \tag{3}$$

Where $\|\cdot\|$ denotes the sum of L2 distance. This can be solved by Singular Value Decomposition (SVD), or in fact Principal Component Analysis (PCA):

$$\begin{aligned} \text{SVD}(Y) &= USV^T \\ [n, N] &= [n, n] \times [n, N] \times [N, N]; \\ M &= U_{c'} U_{c'}^T \\ [n, n] &= [n, c'] \times [c', n]; \end{aligned} \tag{4}$$

Where $U_{c'}$ means the first c' columns of U. Plugging this into Eq. 2:

$$\begin{aligned} Y &\approx U_{c'}(U_{c'}^T W)X \\ [n, N] &\approx [n, c'] \times ([c', n] \times [n, (c, k, k)]) \times [(c, k, k), N]; \\ Y &\approx W_2 W_1 X \\ [n, N] &\approx [n, c'] \times [c', (c, k, k)] \times [(c, k, k), N]; \end{aligned} \tag{5}$$

Eq. 5 can be then interpreted as first applying a regular convolutional layer W_1 with c' output channels, followed by a pointwise layer W_2 that projects them into n channels, as shown in FIGURE 2 (b). c' is adjustable, which provides different speed-up ratios.

It's worth noting that, in the optimization problem of Eq. 3, the N dimension doesn't have to include all possible patches from the training dataset (which can be a forbiddingly large number). Instead, according to the channel decomposition work [39], it's enough to randomly sample 10 patches per image and 300 images in total. Therefore, $N = 300 \times 10$. This also applies to our approach.

3.3 Single-Layer Depth-Wise Decomposition: Single-Input-Channel Case

We now introduce our depth-wise decomposition method for a special case: a regular convolutional layer with $c = 1$ input channels and n output channels. It is converted into a single-channel depth-wise convolution D with $c = 1$ filter of $[k, k]$, followed by a $[1, 1]$ point-wise convolution projecting the one channel into n. Meanwhile, the output approximates the original Y:

$$\begin{aligned} Y &\approx PDX \\ [n, N] &\approx [n, 1] \times [1, (1, k, k)] \times [(1, k, k), N]; \end{aligned} \tag{6}$$

Given section 3.2, this case equivalents applying channel decomposition, with $c = 1$ input channels and $c' = 1$ intermediate channels.

3.4 Single-Layer Depth-Wise Decomposition: Multi-Input-Channel Case

Now consider a regular convolutional layer with multiple c input channels. In fact, this can be seen as c single-input-channel convolutions applied to each input channel individually, then having their results summed up:

$$\begin{aligned} Y_i &\approx P_i D_i X_i \\ [n, N] &\approx [n, 1] \times [1, (1, k, k)] \times [(1, k, k), N]; \\ Y &\approx \sum_{i=0}^{c} (Y_i) \end{aligned} \tag{7}$$

As a result, a naive solution for multi-input-channel depth-wise decomposition is simply applying the single-input-channel case for each input channel, as shown in algorithm 1.

Algorithm 1 Depth-wise Decomposition

X_i is the ith channel of the input feature map (shape: $[(1, k, k), N]$). W_i is the filters applied to X_i (shape: $[n, (1, k, k)]$).

```
Initialize $D$, the depth-wise filters shaped $[c, (1, k, k)]$, to all-zero.
initialize $P$, the point-wise weights shaped $[n, c]$, to all-zero.
**for** $i \leftarrow 0$ to $c - 1$ **do**
    $Y_i = W_i X_i$
    $U, S, V^T = SVD(Y_i)$
    $D[i, :] = U_1^T W_i$, $U_1$ is the first column of $U$.
    $P[:, i] = U_1$
**end for**
**return** $D$, $P$
```

3.5 Multi-Input-Channel Case With Inter-Channel Error Compensation

A caveat with the naive solution of algorithm 1 is the accumulation of error. Each channel contributes accumulation error $E_i = Y_i - P_i D_i X_i$. If not accounted for, the total error would be $\sum_{i=0}^{c}(E_i)$, which hurts accuracy of the decomposed network. Viewing the decomposition as reconstructing the output Y_i given X_i with sparser weights, we can address the problem by performing an asymmetric reconstruction, the target of which is $Y_i + E_i'$, with E_i' offsetting the accumulated error of previous channels. This is further explained by TABLE 3.5, where the accumulated error is confined.

Table 1: Correcting the inter-channel accumulated error by asymmetric reconstruction.

Iteration i	Reconstruction target	Reconstructed value	Accumulated error
0	Y_0	$Y_0 - E_0$	E_0
1	$Y_1 + E_0$	$(Y_1 + E_0) - E_1$	E_1
2	$Y_2 + E_1$	$(Y_2 + E_1) - E_2$	E_2

Acquiring E_i is practical, because the activation of the original network is always accessible. Accordingly, the optimization problem changes from Eq. 3 to:

$$\begin{aligned} \arg\min_M \|(Y_i + E_{i-1}) - MY_i\|; \\ \text{s.t. } \operatorname{rank}(M) = 1 \end{aligned} \tag{8}$$

This problem still has a closed-form solution, with Generalized SVD (GSVD) [39, 91]:

$$\begin{aligned} U, S, V^T &= \text{GSVD}(Y_i + E_{i-1}, Y_i); \\ M^* &= U_1 S_{11} V_1^T \\ [n, n] &= [n, 1] \times [1, 1] \times [1, n]; \end{aligned} \tag{9}$$

Where U, S, V are all shaped $[n, n]$, U_1, V_1 denotes their first column, and S_{11} denotes the top-left element of S. Replacing the SVD with GSVD in algorthim 1, we have multi-input-channel depth-wise decomposition with inter-channel error compensation, in alogrithm 2:

Algorithm 2 Depth-wise Decomposition with error compensation

X_i is the ith channel of the input feature map (shape: $[(1, k, k), N]$). W_i is the filters applied to X_i (shape: $[n, (1, k, k)]$). E_i is the error of each step.

```
Initialize $D$, the depth-wise filters shaped $[c, (1, k, k)]$, to all-zero.
initialize $P$, the point-wise weights shaped $[n, c]$, to all-zero.
for $i \leftarrow 1$ to $c$ do
    $Y_i = W_i X_i$
    $U, S, V^T = GSVD(Y_i + E_{i-1}, Y_i)$
    $D[i, :] = D_i = V_1^T W_i$
    $P[:, i] = P_i = U_1 S_{11}$
    $E_i = Y_i - P_i D_i X_i$
end for
return $D, P$
```

3.6 Multi-Layer Depth-Wise Decomposition

Finally, our approach can be applied to deep convolutional neural networks layer-by-layer (Experiments in Section 4.3). Notice that this also introduces accumulated error from each layer. Let the ith layer's activation of the original network be Y^l, the decomposed version then outputs $Y^l - E^l$, where E^l is the approximation error. This error further affects subsequent layers, which harms accuracy. To compensate for this error, asymmetric reconstruction can be applied similar with Section 3.5. Specifically, since the ground-truth Y^l is always accessible, regardless of how the input X^l of each layer l changes due to approximation error, GSVD can always find an optimal decomposition such that Y^l is reconstructed as close as possible, taking previous errors into account. Therefore, for the multi-layer case, we simply apply algorithm 2 to each layer of the network, while changing the optimization problem in Eq. 8 into:

$$\begin{aligned} &\arg\min_{M} \|(Y_i + E_{i-1}) - MW_i X_i'\|; \\ &\qquad\qquad \text{s.t. } \operatorname{rank}(M) = 1 \end{aligned} \tag{10}$$

Where X_i' is the changed input of the current layer due to the errors in previous layers. GSVD gives the closed-from solution for this in the same way as in Section 3.5.

3.7 Fine-Tuning

Following channel decomposition [39], after a deep CNN is fully decomposed, it can be fine-tuned for ten epochs with a small learning rate $1e^{-4}$ to obtain better accuracy (Section 4.3). In the experiment results, we specify it in the names when this is applied.

4. EXPERIMENTS

We conduct rich experiments on ImageNet [4], 2012 classification dataset. ImageNet is a very large image classification dataset which consists of 1000 classes. We use the 1.3 million training images to train our models, and evaluate them with with top-1 error rate and relative error on the 50,000 validation images. Our neural networks implementation is based on TensorFlow [92]. Our implementation of the baseline approach channel decomposition [39], is based on pure Python[1].

4.1 Sanity Check

To check the correctness of our implementation, we created a random weights matrix of size 128×64 and the corresponding random input feature of size 64×3000. The resulting output response is, therefore, a 128×3000 matrix. Our baseline is channel decomposition [39]. Our Depth-wise Decomposition decomposes a convolutional layer into a depth-wise convolution followed by a

[1] github.com/yihui-he/channel-pruning

Table 2: Sanity check. Using random data, our approach performs as good as channel decomposition [39]. Standard deviations and relative errors are obtained after 10 runs

single layer 9× acceleration with random data	
	relative error
Channel Decomposition [39]	$0.887 \pm 1.34e^{-6}$
Depth-wise Decomposition (ours)	$0.914 \pm 3.21e^{-7}$
Depth-wise Decomposition (ours) with compensation	$0.906 \pm 2.78e^{-7}$

point-wise convolution, which accelerates the convolutional layer by around 9 times. For a fair comparison, we adjust channel decomposition to get the same 9-times acceleration.

To measure the correctness of our approach, we measure the relative error of the resulting output activation between those of decomposing algorithms(including baseline [39], Depth-wise Decomposition and Depth-wise Decomposition with inter-channel compensation) and the ground-truth output response. As is shown in TABLE 4, our Depth-wise Decomposition is comparable to channel decomposition [39].

We further tested the reconstruction error of Depth-wise Decomposition with inter-channel error compensation. As expected, the relative error of this method is smaller than the basic approach.

4.2 Single Layer Decomposition

Firstly, we want to evaluate the reconstruction error of Depth-wise Decomposition for a single layer. We conduct experiments on decomposing five different convolutional layers of ShuffleNet V2 [36]. We decompose each 3-by-3 convolutional layer with the baseline method [39], Depth-wise Decomposition and Depth-wise Decomposition with error compensation. FIGURE 3 shows the relative error of reconstructing each of the five convolutional layers. As shown in the FIGURE, Depth-wise Decomposition results in lower reconstruction error for all five layers. Furthermore, the Depth-wise Decomposition with inter-channel compensation results in even smaller reconstruction error. This proves that both Depth-wise Decomposition and Depth-wise Decomposition with inter-channel error compensation are better at preserving the accuracy of the network while achieving the same level of effectiveness in terms of accelerating the network.

Secondly, we want to measure the effectiveness of Depth-wise Decomposition. Thus we measure how much the top-1 error of the resulting network has increased when decomposing a single convolutional layer in ShuffleNet V2 [36]. We demonstrate the effectiveness of Depth-wise Decomposition by showing the top-1 error of the networks resulting from decomposing each of the four convolutional layers in stage 4 of ShuffleNet V2 [36], along with the top-1 error achieved by the original ShuffleNet V2 [36], model . As shown in FIGURE 4, the baseline method [39], when accelerating one convolutional layer by around 9 times, largely increases the top-1 error of the resulting network. In contrast, Depth-wise Decomposition, along with Depth-wise Decomposition

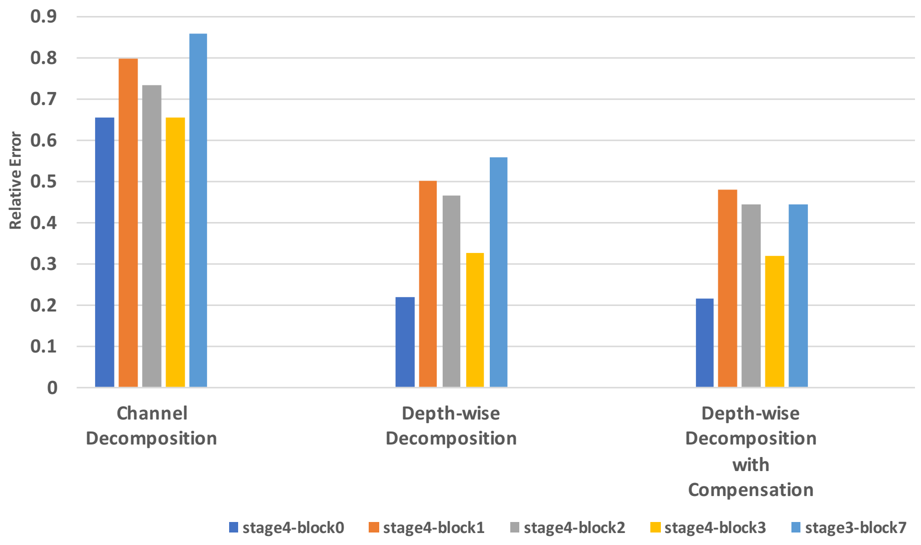


Figure 3: Relative error for decomposing a single conv layer in ShuffleNet V2. ShuffleNet consists of 4 stages, each stage contains a number of blocks. Here, we decompose the core 3x3 conv layer of a certain block. The appendix of [36], shows the detailed architecture.

with error compensation do not have such a significant impact on the top-1 error of the resulting network.

4.3 Whole Model Decomposition

Lastly we want to measure the effect of decomposing the whole model. First we train a "folded" ShuffleNet V2 [36]. In this model we replace all depth-wise separable convolutional layers with their regular convolution counterparts. It's dubbed as "folded" because the depthwise conv and channelwise conv layers are combined back together. We train this model with the exact same settings as the original ShuffleNets V2 paper. Our implementation is based on TensorPack [93][2]. Shown in TABLE 4.3, the Top-1 accuracy of "folded" ShuffleNet V2 is **3.1%** higher than the original ShuffleNet V2, which serves as the upper bound of our proposed algorithm.

Then we decompose each convolutional layers in the folded ShuffleNet V2 [36], using our multi-layer channel decomposition method with inter-channel error compensation (Section 3.6). This decomposed model has the exact same architecture as the original ShuffleNet V2. Then the decomposed model is further fine-tuned on ImageNet training dataset for 10 epochs (Section 3.7).

As a baseline, we perform channel decomposition [39], under the same $9\times$ acceleration ratio on ShuffleNet V2 and fine-tune for the same number of epochs as our model. Interestingly, channel decomposition does not work well for high acceleration ratio in our case. As shown in TABLE 4.3, it even performs worse than the original ShuffleNet V2. The inherently worse accuracy-speed trade-off of regular conv layers may have contributed to this.

[2] github.com/tensorpack/tensorpack

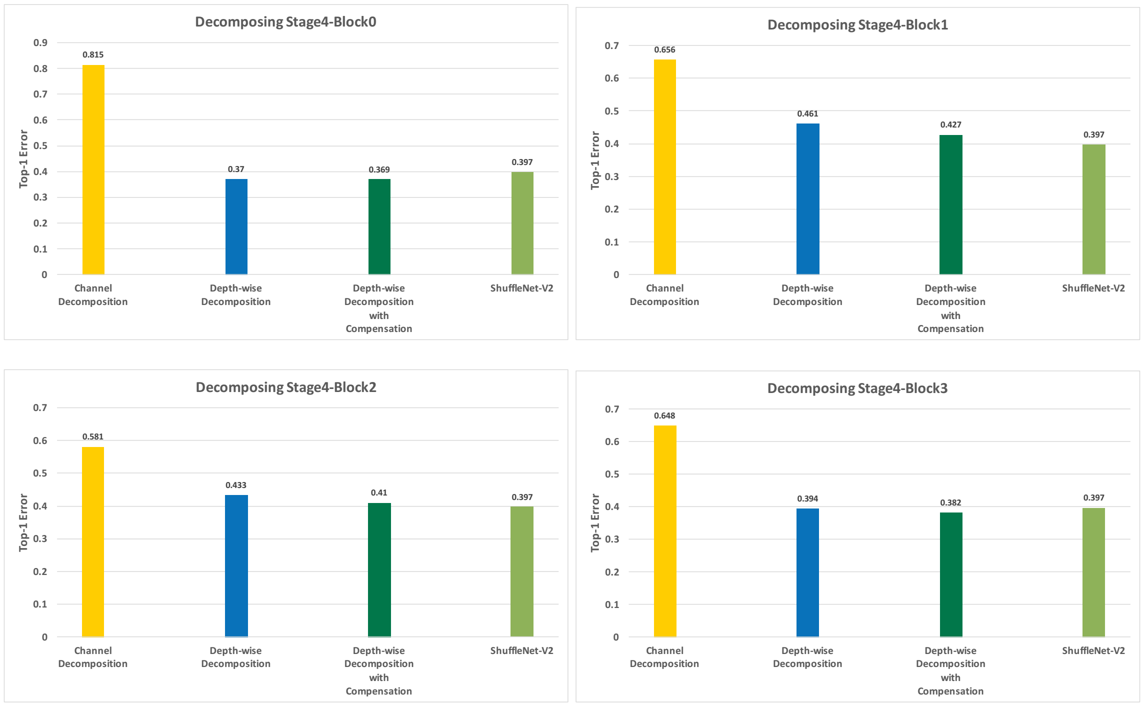


Figure 4: Top-1 testing error for decomposing a single convolutional layer in ShuffleNet V2 [36]

Table 3: Comparisons on compressing ”ShuffleNet V2 0.5x”, the main architecture of [36].

Whole Model Decomposition with ImageNet	
	top-1 error
Folded ShuffleNet V2 [36]	36.6%
ShuffleNet V2 [36]	39.7%
Fine-tuned Channel Decomposition (our impl.)	40.0%
Depth-wise Decomposition with compensation (ours)	43.9%
Fine-tuned Depth-wise Decomposition with compensation (ours)	**37.9%**

Table 4: Top-1 error for compressing ShuffleNet V2 family and Xception. ”Baseline” are their performance as originally proposed, ”ours” are the performance of the same architectures obtained from our depthwise decomposition method.

top-1 error	baseline	ours
ShuffleNet V2 0.5× [36]	39.7%	**37.9%**
ShuffleNet V2 1.0× [36]	30.6%	**28.0%**
ShuffleNet V2 2.0× [36]	25.1%	**23.4%**
Xception [1]	21.0%	**20.1%**

As is shown in TABLE 4.3, our Depth-wise Decomposition with inter-channel error compensation has ~ **2%** lower top-1 error than the original ShuffleNet V2 [36]. It proves that our method is able to achieve a better level of accuracy under the same computational complexity. Another benefit is that our method only needs fine-tuning, instead of training from scratch.

To test the generalizability of our approach, we further test on other ShuffleNet V2 architectures and Xception. Shown in TABLE 4.3, The results are consistent with ShuffleNet v2 0.5× in TABLE 4.3. For ShuffleNet v2 1×, our approach improves the Top-1 accuracy by **1.6%**. For ShuffleNet v2 2×, the Top-1 accuracy improvement is **1.7%**. For Xception, the model performs **1.6%** better. This shows our method is applicable to various architectures. In fact, any future network design with depthwise separable conv layers can benefit from this approach.

5. CONCLUSION

In conclusion, very deep convolutional neural networks (CNNs) are widely used by many computer vision applications, while many have headroom for latency improvement. This is especially important on computation-limited platforms. We propose a novel method to decompose regular convolutions into highly-efficient depth-wise separable convolutions. Given a pre-trained ”heavy” convolutional neural network, our method optimizes its latency with one-off decomposition and cheap fine-tuning, with minor accuracy loss. Through experiments with the ShuffleNet V2 model [36], on ImageNet [4], we demonstrate the method to achieve a superior acceleration/accuracy trade-off. This is particularly valuable for applications like autonomous vehicles, wearable devices, smart phones, and many more. For future work, our approach may be applied to other vision tasks, like object detection, semantic segmentation, keypoint detection, etc.

References

[1] Chollet F. Xception: Deep Learning With Depthwise Separable Convolutions. In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017:1251-1258.

[2] Zeiler MD, Fergus R. Visualizing and Understanding Convolutional Networks. In: European conference on computer vision. Springer. 2014:818-833.

[3] Tan M, Chen B, Pang R, Vasudevan V, Le QV, at al . Mnasnet: Platform-Aware Neural Architecture Search for Mobile. In proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019:2820-2828.

[4] Deng J, Dong W, Socher R, Li LJ, Li K, at al. Imagenet: A Large-Scale Hierarchical Image Database. In IEEE conference on computer vision and pattern recognition. 2009:248-255. In2009 IEEE conference on computer vision and pattern recognition 2009 Jun 20 (pp. 248-255). Ieee.

[5] Lin TY, Maire M, Belongie S, Hays J, Perona P, et al. Microsoft Coco: Common Objects in Context. In: European Conference on Computer Vision. Cham: Springer. 2014:740-755.

[6] Zhou B, Zhao H, Puig X, Fidler S, Barriuso A, et al. Scene Parsing Through ade20K Dataset. In: Proceedings of the Ieee Conference on Computer Vision and Pattern Recognition; 2017:633-641.

[7] Hetang C, Wang Y. Novel View Synthesis From a Single Rgbd Image for Indoor Scenes. 2023. ArXiv Preprint: https://arxiv.org/pdf/2311.01065.pdf

[8] Wang J, He Y. Physics-Aware 3D Mesh Synthesis. In: International Confer. IEEE Publications. 2019:502-512.

[9] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. In: Proceedings of the Ieee Conference on Computer Vision and Pattern Recognition. 2016:770-778.

[10] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, et al. Going Deeper With Convolutions. In: Proceedings of the Ieee Conference on Computer Vision and Pattern Recognition. 2015:1-9.

[11] He Y. Estimated Depth Map Helps Image Classification.2017. ArXiv Preprint: https://arxiv.org/pdf/1709.07077.pdf

[12] Song G, Leng B, Liu Y, Hetang C, Cai S, et al. Region-Based Quality Estimation Network for Large-Scale Person Re-Identification. In: Proceedings of the Thirty-Second Aaai Conference on Artificial Intelligence.2018;32.

[13] Walawalkar D, He Y, Pillai R. An Empirical Analysis of Deep Audio-Visual Models for Speech Recognition.2018. Arxiv Preprint :https//arxiv.org/abs/1812.09336

[14] He Y, Zhang X, Savvides M, Kitani K. Softer-Nms: Rethinking Bounding Box Regression for Accurate Object Detection. 2019:2888-2897. Arxiv Preprint: https://arxiv.org/pdf/1809.08545.pdf

[15] Zhu C, He Y, Savvides M. Feature Selective Anchor-Free Module for Single-Shot Object Detection. 2019: 840-849. ArXiv Preprint: https://arxiv.org/pdf/1903.00621.pdf

[16] He Y, Zhu C, Wang J, Savvides M, Zhang X, et al. Bounding Box Regression With Uncertainty for Accurate Object Detection. In: Proceedings of the Ieee/Cvf Conference on Computer Vision and Pattern Recognition. 2019:2883-2892.

[17] He Y, Wang J. Deep Multivariate Mixture of Gaussians for Object Detection Under occlusion.2019.

[18] He Y, Wang J. Deep Mixture Density Network for Probabilistic Object Detection. In: International Rsj, Editor Conference on Intelligent Robots and Systems (Iros). Ieee Publications. 2020:10550-10555.

[19] He Y, Yan R, Fragkiadaki K, Yu S I. Epipolar Transformers. In: Proceedings of the Ieee/Cvf Conference on Computer Vision and Pattern Recognition. 2020:7779-7788.

[20] Hetang C. Impression Network for Video Object Detection. In: 3rd International Conference on Information Technology, ICIBA. 2023;3:735-743.

[21] He Y, Yan R, Fragkiadaki K, Yu SI. Epipolar Transformer for Multi-View Human Pose Estimation. In: Proceedings of the Ieee/Cvf Conference on Computer Vision and Pattern Recognition Workshops. 2020:1036-1037.

[22] He Y. Object Detection With YOLO on Artwork Dataset. Advanced Computer Vision at Jiaotong University. 2016:22.

[23] Long J, Shelhamer E, Darrell T. Fully Convolutional Networks for Semantic Segmentation. In: Proceedings of the Ieee Conference on Computer Vision and Pattern Recognition. 2015:3431-3440.

[24] Hariharan B, Arbeláez P, Girshick R, Malik J. Hypercolumns for Object Segmentation and Fine-Grained Localization. In: Proceedings of the Ieee Conference on Computer Vision and Pattern Recognition. 2015:447-456.

[25] Liang Y, Yang Z, Zhang K, He Y, Wang J, et al. Single Image Super-Resolution via a Lightweight Residual Convolutional Neural Network. 2017. ArXiv Preprint: https://arxiv.org/pdf/1703.08173.pdf

[26] Wang J, He Y, Wang X, Xinjia Y, Chen X, et al. Prediction-Tracking Segmentation. 2019. ArXiv Preprint: https://arxiv.org/pdf/1904.03280.pdf

[27] Wang J, He Y. Motion Prediction in Visual Object Tracking. In: In Rsj, Editor Ternational Conference on Iros. 2020:10374-10379.

[28] Tangcongrui HE, Hongwei Q. Methods and apparatuses for recognizing video and training, electronic device and medium. 2021;909:380.

[29] Wang J, Xu Junkai, Shull PB. Vertical Jump Height Estimation Algorithm Based on Takeoff and Landing Identification via Foot-Worn Inertial Sensing. J Biomech Eng. 2018;140:034502.

[30] Xia H, Xu Junkai, Wang J, Hunt MA, Shull PB, et al. Validation of a Smart Shoe for Estimating Foot Progression Angle During Walking Gait. J Biomech. 2017;61:193-198.

[31] Ma X, He Y, Luo Xiapu, Li J, Zhao M, et al. Camera Placement Based on Vehicle Traffic for Better City Security Surveillance. IEEE Intelligent Systems. 2018;33:49-61.

[32] He Y, Ma X, Luo Xiapu, Li J, Zhao Mengchen, et al. Vehicle Traffic Driven Camera Placement for Better Metropolis Security Surveillance. 2017. ArXiv Preprint: https://arxiv.org/abs/1705.08508.

[33] Djuric N, Radosavljevic V, Cui Henggang, Nguyen T, Chou F-C, et al. Motion Prediction of Traffic Actors for Autonomous Driving Using Deep Convolutional Networks. 2018. ArXiv Preprint: https://arxiv.org/abs/1808.05819.

[34] Thibaux R, Silver DH, Hetang C. Stop location change detection. US Patent App. 2022;131:232.

[35] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, et al. Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017. ArXiv preprint: https://arxiv.org/pdf/1704.04861.pdf

[36] Ma N, Zhang X, Zheng H-T, Sun J. Shufflenet V2: Practical Guidelines for Efficient CNN Architecture Design. In: The European Conference on Computer Vision (ECC). 2018:116-131.

[37] Zhang X, Zhou X, Lin M, Sun J. Shufflenet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. 2018. ArXiv Preprint: https://arxiv.org/pdf/1707.01083.pdf

[38] Wu Bichen, Wan A, Yue X, Jin P, Zhao Sicheng, et al. Shift: A Zero Flop, Zero Parameter lter-native to Spatial Convolutions. Inproceedings of the Ieee Conference on Computer Vision and Pattern Recognition. 2017: 9127-9135. ArXiv Preprint: https://arxiv.org/pdf/1711.08141.pdf

[39] Zhang X, Zou J, He K, Sun J. Accelerating Very Deep Convolutional Networks for Classification and Detection. IEEE Trans Pattern Anal Mach Intell. 2016;38:1943-1955.

[40] Jaderberg M, Vedaldi A, Zisserman A. Speeding Up Convolutional Neural Networks With Low Rank Expansions. 2014. ArXiv Preprint: https://arxiv.org/pdf/1405.3866.pdf

[41] He Y, Zhang X, Sun J. Channel Pruning for Accelerating Very Deep Neural Networks. In: International Conference on Computer Vision.2017:1389-1397.

[42] He Y. Pruning Very Deep Neural Network Channels for Efficient Inference. 2022. ArXiv preprint: https://arxiv.org/pdf/2211.08339.pdf

[43] Zhang X, Yihui HE. Image processing method and apparatus, and computer readable storage medium. Filed: Jul 12, 2018 and Date of Patent: Jul 14, 2020. US Patent 10,713,533.

[44] Han S, Mao H, Dally WJ. Deep Compression: Compressing Deep Neural Net-work With Pruning, Trained Quantization and Huffman Coding. 2015. Arxiv Preprint: https://arxiv.org/pdf/1510.00149.pdf

[45] Koratana A, Kang D, Bailis P, Zaharia M. Lit: Block-Wise Intermediate Representation Training for Model Compression. 2018. Arxiv Preprint: https://arxiv.org/pdf/1810.01937.pdf

[46] Sifre L, Mallat S. Rigid-Motion Scattering for Image Classification. 2014. Arxiv Preprint: https://arxiv.org/pdf/1403.1687.pdf.

[47] Zoph B, Vasudevan V, Shlens L, Quoc V Le. Learning Transferable Architectures for Scalable Image Recognition. Inproceedings of the Ieee Conference on Computer Vision and Pattern Recognition. 2018:8697-8710.

[48] LeCun Y, Denker JS, Solla SA, Howard Re, Jackel Ld, et al. Optimal Brain Damage. NIPS. 1989;2:598-605.

[49] Hassibi B, Stork DG. Second Order Derivatives for Network Pruning: Optimal Brain Surgeon. Morgan Kaufmann. 1995;5.

[50] Cheng Y, Wang D, Zhou P, Zhang T. A Survey of Model Compression and Acceleration for Deep Neural Networks. 2017. ArXiv Preprint: https://arxiv.org/pdf/1710.09282.pdf.

[51] Bagherinezhad H, Rastegari M, Farhadi A. Lcnn: Lookup-Based Convolutional Neural Network. Inproceedings of the Ieee Conference on Computer Vision and Pattern Recognition. 2017:7120-7129. ArXiv Preprint arXiv: https://arxiv.org/pdf/1611.06473.pdf

[52] Rastegari M, Ordonez V, Redmon J, Farhadi A. Xnor-Net: Imagenet Classification Using Binary Convolutional Neural Networks. In: European Conference on Computer Vision. Springer. 2016:525-542.

[53] Guo J, Li Y, Lin W, Chen Y, Li J, et al. Network Decoupling: From Regular to Depthwise Separable Convolutions. 2018. ArXiv Preprint: https://arxiv.org/pdf/1808.05517.pdf

[54] He Y, Liu X, Zhong H, Ma Y. Addressnet: Shift-Based Primitives for Efficient Convolutional Neural Networks. In: Ieee Winter Conference on Applications of Computer Vision (WACV). IEEE Publications. 2019:1213-1222.

[55] Zhong H, Liu X, He Y, Ma Y. Shift-Based Primitives for Efficient Convolutional Neural Networks. 2018. ArXiv Preprint: https://arxiv.org/pdf/1809.08458.pdf

[56] Han S, Pool J, Tran J, Dally W. Learning Both Weights and Connections for Efficient Neural Network. Adv Neural Inf Process Syst. 2015;28:1135-1143.

[57] Liu B, Wang M, Foroosh H, Tappen M, Pensky M, et al. Sparse Convolutional Neural Networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015;806-814.

[58] Lebedev V, Lempitsky V. Fast Convnets Using Group-Wise Brain Damage. Inproceedings of the Ieee Conference on Computer Vision and Pattern Recognition. 2016:2554-2564. ArXiv Preprint: https://arxiv.org/pdf/1506.02515.pdf

[59] Han S, Liu X, Mao Huizi, Pu J, Pedram A, et al. Eie: Efficient Inference Engine on Compressed Deep Neural Network. In: Proceedings of the 43rd International Symposium on Computer Architecture. IEEE Publications. 2016;44:243-254.

[60] Guo Y, Yao A, Chen Y. Dynamic Network Surgery for Efficient Dnns. Adv Neural Inf Process Syst. 2016;29:1379-1387.

[61] Yang TJ, Chen YH, Sze V. Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning. Inproceedings of the Ieee Conference on Computer Vision and Pattern Recognition. 2017:5687-5695.

[62] Chetlur S, Woolley C, Vandermersch P, Cohen J, Tran J, et al. Cudnn: Efficient Primitives for Deep Learning. 2014. ArXiv Preprint: https://arxiv.org/pdf/1410.0759.pdf

[63] Alvarez JM, Salzmann M. Learning the Number of Neurons in Deep Networks. Adv Neural Inf Process Syst. 2016;29:2262-2270.

[64] Wen W, Wu Chunpeng, Wang Y, Chen Yiran, Li H, et al. Learning Structured Sparsity in Deep Neural Networks. Adv Neural Inf Process Syst. 2016;29:2074-2082.

[65] Zhou H, Alvarez JM, Porikli F. Less Is More: Towards Compact Cnns. In: European Conference on Computer Vision. Springer International Publishing. 2016:662-677.

[66] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-Based Learning Applied to Document Recognition. Proc IEEE. 1998;86:2278-2324.

[67] Krizhevsky A, Sutskever I, Hinton GE. Imagenet Classification With Deep Convolutional Neural Networks. Adv Neural Inf Process Syst. 2012;25:1097-1105.

[68] Anwar S, Hwang K, Sung W. Structured Pruning of Deep Convolutional Neural Networks. ACM Journal on Emerging Technologies in Computing Systems. 2015. ArXiv Preprint: https://arxiv.org/ftp/arxiv/papers/1512/1512.08571.pdf

[69] Polyak A, Wolf L. Channel-Level Acceleration of Deep Face Representations. IEEE Access. 2015;3:2163-2175.

[70] Srinivas S, Babu RV. Data-Free Parameter Pruning for Deep Neural Networks. 2015. ArXiv Preprint: https://arxiv.org/pdf/1507.06149.pdf

[71] Mariet Z, Sra S. Diversity Networks. Neural Network Compression Using Determinantal Point Processes. 2015. ArXiv Preprint: https://arxiv.org/pdf/1511.05077.pdf

[72] Zhou Z, Yue T, Liang C, Bai X, Chen D, et al. Unlocking Everyday Wisdom: Enhancing Machine Comprehension With Script Knowledge Integration. Appl Sci. 2023;13:9461.

[73] Hu H, Peng R, Tai Y W, Tang C K. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. 2016. ArXiv Preprint: https://arxiv.org/pdf/1607.03250.pdf

[74] Li H, Kadav A, Durdanovic I, Samet H, Graf HP, et al. Pruning Filters for Efficient Convnets. 2016. ArXiv Preprint: https://arxiv.org/pdf/1608.08710.pdf.

[75] Anwar S, Sung W. Compact deep convolutional neural networks with coarse pruning. 2016. ArXiv Preprint: https://arxiv.org/pdf/1610.09639.pdf

[76] He Y, Han S. Adc: Automated Deep Compression and Acceleration With Reinforcement Learning. 2018:784-800. ArXiv Preprint: https://arxiv.org/pdf/1802.03494.pdf

[77] He Y, Lin J, Liu Z, Wang H, Han S, et al. Amc: Automl for Model Compression and Acceleration on Mobile Devices. In: Proceedings of the European Conference on Computer Vision. 2018:784-800.

[78] Lebedev V, Ganin Y, Rakhuba M, Oseledets I, Lempitsky V, et al. Speeding-up Convolutional Neural Networks Using Fine-Tuned Cp-Decomposition. 2014. Arxiv Preprint: https://arxiv.org/pdf/1412.6553.pdf

[79] Gong Y, Liu L, Yang M, Bourdev L. Compressing Deep Convolutional Networks Using Vector Quantization. 2014. ArXiv Preprint: https://arxiv.org/pdf/1412.6115.pdf

[80] Kim Y D, Park E, Yoo S, Choi T, Yang L, et al. Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications.2015. Arxiv Preprint: https://arxiv.org/pdf/1511.06530.pdf

[81] Xue J, Li J, Gong Y. Restructuring of Deep Neural Network Acoustic Models With Singular Value Decomposition. In: Interspeech. 2013:2365-2369.

[82] Denton EL, Zaremba W, Bruna J, LeCun Y, Fergus R, et al. Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation. Adv Neural Inf Process Syst. 2014;27:1269-1277.

[83] Girshick R, Fast R-Cnn. Proceedings of the Ieee International Conference on Computer Vision. 2015:1440-1448.

[84] Mathieu M, Henaff M, LeCun Y. Fast Training of Convolutional Networks Through Ffts. 2013. ArXiv Preprint: https://arxiv.org/pdf/1312.5851.pdf

[85] Vasilache N, Johnson J, Mathieu M, Chintala S, Piantino S, et al. Fast Convolutional Nets With Fbfft: A Gpu Performance Evaluation. 2014. ArXiv Preprint: https://arxiv.org/pdf/1412.7580.pdf

[86] Zhang L, Wang W, Keyi Y, Huang J, Lyu Q, et al. Sliding-Bert: Striding Towards Conversational Machine Comprehension in Long Contex. Adv Artif Intell Mach Learn. 2023.

[87] Lavin A. Fast Algorithms for Convolutional Neural Networks. InProceedings of the IEEE conference on computer vision and pattern recognition. 2016:4013-4021.

[88] Courbariaux M, Bengio Y. Binarynet: Training Deep Neural Networks With Weights and Activations Constrained To+ 1 Or-1. 2016. Arxiv Preprint: https://arxiv.org/pdf/1602.02830.pdf

[89] Phan H, Huynh D, He Y, Savvides M, Shen Z, et al. Mobinet: A Mobile Binary Network for Image Classification. In: Ieee Winter Conference on Applications of Computer Vision (WACV). IEEE Publications. 2020:4353-3462.

[90] Wang K, Liu Z, Lin Y, Lin J, Han S, et al. Haq: Hardware-Aware Automated Quantization. 2018:8612-8620. ArXiv Preprint: https://arxiv.org/pdf/1811.08886.pdf

[91] Takane Y, Hwang H. Regularized Linear and Kernel Redundancy Analysis. Comp Stat Data Anal. 2007;52:394-405.

[92] Abadi M, Barham P, Chen J, Chen Z, Davis A, et al. Tensorflow: A System for Large-Scale Machine Learning. In12th Usenix Symposium on Operating Systems Design and Implementation. 2016:265-283.

[93] Wu Y et al. Tensorpack; 2016. Available from: https://github.com/tensorpack