Accelerating Very Deep Convolutional Networks for Classification and Detection

Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun

Abstract—This paper aims to accelerate the test-time computation of convolutional neural networks (CNNs), especially very deep CNNs [1] that have substantially impacted the computer vision community. Unlike previous methods that are designed for approximating linear filters or linear responses, our method takes the nonlinear units into account. We develop an effective solution to the resulting nonlinear optimization problem without the need of stochastic gradient descent (SGD). More importantly, while previous methods mainly focus on optimizing one or two layers, our nonlinear method enables an asymmetric reconstruction that reduces the rapidly accumulated error when multiple (e.g., $\geq$10) layers are approximated. For the widely used very deep VGG-16 model [1], our method achieves a whole-model speedup of 4$\times$ with merely a 0.3 percent increase of top-5 error in ImageNet classification. Our 4$\times$ accelerated VGG-16 model also shows a graceful accuracy degradation for object detection when plugged into the Fast R-CNN detector [2].

Index Terms—Convolutional neural networks, acceleration, image classification, object detection

1 Introduction

The accuracy of convolutional neural networks (CNNs) [3], [4] has been continuously improving [1], [5], [6], [7], [8], but the computational cost of these networks also increases significantly. For example, the very deep VGG models [1], which have witnessed great success in a wide range of recognition tasks [2], [9], [10], [11], [12], [13], [14], are substantially slower than earlier models [4], [5]. Real-world systems may suffer from the low speed of these networks. For example, a cloud service needs to process thousands of new requests per seconds; portable devices such as phones and tablets may not afford slow models; some recognition tasks like object detection [2], [7], [10], [11] and semantic segmentation [12], [13], [14] need to apply these models on higher-resolution images. It is thus of practical importance to accelerate test-time performance of CNNs.

There have been a series of studies on accelerating deep CNNs [15], [16], [17], [18]. A common focus of these methods is on the decomposition of one or a few layers. These methods have shown promising speedup ratios and accuracy on one or two layers and whole (but shallower) models. However, few results are available for accelerating *very deep* models (e.g., $\geq$ 10 layers). Experiments on complex datasets such as ImageNet [19] are also limited - e.g., the results in [16], [17], [18] are about accelerating a *single* layer of the shallower AlexNet [4]. Moreover, performance of the accelerated networks as generic feature extractors for other recognition tasks [2], [12] remain unclear.

It is nontrivial to speed up *whole*, *very deep* models for *complex* tasks like ImageNet classification. Acceleration algorithms involve not only the decomposition of layers, but also the optimization solutions to the decomposition. Data (response) reconstruction solvers [17] based on stochastic gradient descent (SGD) and backpropagation work well for simpler tasks such as character classification [17], but are less effective for complex ImageNet models (as we will discussed in Section 4). These SGD-based solvers are sensitive to initialization and learning rates, and might be trapped into poorer local optima for regressing responses. Moreover, even when a solver manages to accelerate a single layer, the *accumulated* error of approximating multiple layers grow rapidly, especially for very deep models. Besides, the layers of a very deep model may exhibit a great diversity in filter numbers, feature map sizes, sparsity, and redundancy. It may not be beneficial to uniformly accelerate all layers.

In this paper, we present an accelerating method that is effective for very deep models. We first propose a response reconstruction method that takes into account the nonlinear neurons and a low-rank constraint. A solution based on Generalized Singular Value Decomposition (GSVD) is developed for this nonlinear problem, without the need of SGD. Our explicit treatment of the nonlinearity better models a nonlinear layer, and more importantly, enables an *asymmetric* reconstruction that accounts for the error from previous approximated layers. This method effectively reduces the accumulated error when multiple layers are approximated sequentially. We also present a rank selection method for adaptively determining the acceleration of each layer for a whole model, based on their redundancy.

In experiments, we demonstrate the effects of the nonlinear solution, asymmetric reconstruction, and whole-model acceleration by controlled experiments of a 10-layer model on ImageNet classification [19]. Furthermore, we apply our method on the publicly available **VGG-16** model [1], and achieve a 4$\times$ speedup with merely a **0.3** percent increase of top-5 center-view error.

- *X. Zhang and J. Zou are with the Xi'an Jiaotong University, Xi'an, China. E-mail: xyz.clx@stu.xjtu.edu.cn, jhzou@sei.xjtu.edu.cn.*
- *K. He and J. Sun are with Microsoft Research, Beijing, China. E-mail: {kahe, jiansun}@microsoft.com.*

Manuscript received 25 May 2015; revised 9 Oct. 2015; accepted 17 Nov. 2015. Date of publication 19 Nov. 2015; date of current version 12 Sept. 2016. Recommended for acceptance by A. Vedaldi.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2015.2502579

The impact of the ImageNet dataset [19] is not merely on the specific 1000-class classification task; deep models pre-trained on ImageNet have been actively used to replace hand-engineered features, and have showcased excellent accuracy for challenging tasks such as object detection [2], [9], [10], [11] and semantic segmentation [12], [13], [14]. We exploit our method to accelerate the very deep VGG-16 model for Fast R-CNN [2] object detection. With a 4× speedup of all convolutions, our method has a graceful degradation of 0.8 percent mAP (from 66.9 to 66.1 percent) on the PASCAL VOC 2007 detection benchmark [20].

A preliminary version of this manuscript has been presented in a conference [21]. This manuscript extends the initial version from several aspects to strengthen our method. (1) We demonstrate compelling acceleration results on very deep VGG models, and are among the first few works accelerating very deep models. (2) We investigate the accelerated models for transfer-learning-based object detection [2], [9], which is one of the most important applications of ImageNet pre-trained networks. (3) We provide evidence showing that a model trained from scratch and sharing the same structure as the accelerated model is inferior. This discovery suggests that a very deep model can be accelerated not simply because the decomposed network architecture is more powerful, but because the acceleration optimization algorithm is able to digest information.

2 Related Work

Methods [15], [16], [17], [18] for accelerating test-time computation of CNNs in general have two components: (i) a layer decomposition design that reduces time complexity, and (ii) an optimization scheme for the decomposition design. Although the former ("decomposition") attracts more attention because it directly addresses the time complexity, the latter ("optimization") is also essential because not all decompositions are similarly easy to fine good local optima.

The method of Denton et al., [16] is one of the first to exploit low-rank decompositions of filters. Several decomposition designs along different dimensions have been investigated. This method does not explicitly minimize the error of the activations after the nonlinearity, which is influential to the accuracy as we will show. This method presents experiments of accelerating a single layer of an OverFeat network [6], but no whole-model results are available.

Jaderberg et al., [17] present efficient decompositions by separating $k \times k$ filters into $k \times 1$ and $1 \times k$ filters, which was earlier developed for accelerating generic image filters [22]. Channel-wise dimension reduction is also considered. Two optimization schemes are proposed: (i) "filter reconstruction" that minimizes the error of filter weights, and (ii) "data reconstruction" that minimizes the error of responses. In [17], conjugate gradient descent is used to solve filter reconstruction, and SGD with backpropagation is used to solve data reconstruction. Data reconstruction in [17] demonstrates excellent performance on a character classification task using a four-layer network. For ImageNet classification, their paper evaluates a single layer of an OverFeat network by "filter reconstruction". But the performance of whole, very deep models in ImageNet remains unclear.

Concurrent with our work, Lebedev et al., [18] adopt "CP-decomposition" to decompose a layer into five layers of lower complexity. For ImageNet classification, only a single-layer acceleration of AlexNet is reported in [18]. Moreover, Lebedev et al., report that they "failed to find a good SGD learning rate" in their fine-tuning, suggesting that it is nontrivial to optimize the factorization for even a single layer in ImageNet models.

Despite some promising preliminary results that have been obtained in the above works [16], [17], [18], the *whole-model* acceleration of *very deep* networks for *ImageNet* is still an open problem.

Besides the research on decomposing layers, there have been other streams on improving train/test-time performance of CNNs. FFT-based algorithms [23], [24] are applicable for both training and testing, and are particularly effective for large spatial kernels. On the other hand, it is also proposed to train "thin" and deep networks [25], [26] for good trade-off between speed and accuracy. Besides reducing running time, a related issue involving memory conservation [27] has also attracted attention.

3 Approaches

Our method exploits a low-rank assumption for decomposition, following the stream of [16], [17]. We show that this decomposition has a closed-form solution (SVD) for linear neurons, and a slightly more complicated solution (GSVD [28], [29], [30]) for nonlinear neurons. The simplicity of our solver enables an asymmetric reconstruction method for reducing accumulated error of very deep models.

3.1 Low-Rank Approximation of Responses

Our assumption is that the filter response at a pixel of a layer approximately lies on a low-rank subspace. A resulting low-rank decomposition reduces time complexity. To find the approximate low-rank subspace, we minimize the reconstruction error of the responses.

More formally, we consider a convolutional layer with a filter size of $k \times k \times c$, where k is the spatial size of the filter and c is the number of input channels of this layer. To compute a response, this filter is applied on a $k \times k \times c$ volume of the layer input. We use $\mathbf{x} \in \mathbb{R}^{k^2c+1}$ to denote a vector that reshapes this volume, where we append one as the last entry for the sake of the bias. A response $\mathbf{y} \in \mathbb{R}^d$ at a position of a layer is computed as:

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

where W is a d-by-(k^2c+1) matrix, and d is the number of filters. Each row of W denotes the reshaped form of a $k \times k \times c$ filter with the bias appended.

Under the assumption that the vector $\mathbf{y}$ is on a low-rank subspace, we can write $\mathbf{y} = \mathrm{M}(\mathbf{y} - \bar{\mathbf{y}}) + \bar{\mathbf{y}}$, where M is a d-by-d matrix of a rank $d' < d$ and $\bar{\mathbf{y}}$ is the mean vector of responses. Expanding this equation, we can compute a response by:

$$\mathbf{y} = \mathrm{MW}\mathbf{x} + \mathbf{b}, \tag{2}$$

where $\mathbf{b} = \bar{\mathbf{y}} - \mathrm{M}\bar{\mathbf{y}}$ is a new bias. The rank-d' matrix M can be decomposed into two d-by-d' matrices P and Q such that $\mathrm{M} = \mathrm{PQ}^\top$. We denote $\mathrm{W}' = \mathrm{Q}^\top\mathrm{W}$ as a d'-by-(k^2c+1) matrix,

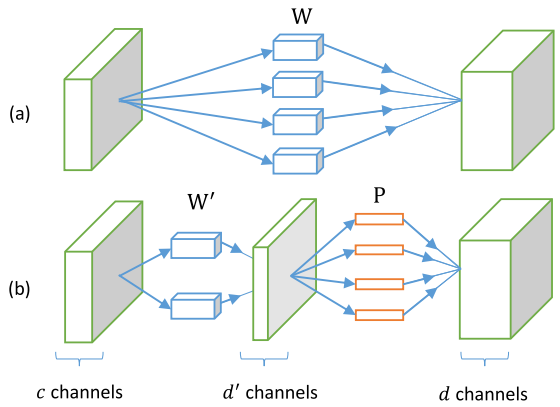


Fig. 1. Illustration of the decomposition. (a) An original layer with complexity $O(dk^2c)$. (b) An approximated layer with complexity reduced to $O(d'k^2c) + O(dd')$.

which is essentially a new set of d' filters. Then we can compute (2) by:

$$\mathbf{y} = \mathrm{P}\mathrm{W}'\mathbf{x} + \mathbf{b}. \tag{3}$$

The complexity of using Eqn. (3) is $O(d'k^2c) + O(dd')$, while the complexity of using Eqn. (1) is $O(dk^2c)$. For many typical models/layers, we usually have $O(dd') \ll O(d'k^2c)$, so the computation in Eqn. (3) will reduce the complexity to about d'/d.

Fig. 1 illustrates how to use Eqn. (3) in a network. We replace the original layer (given by W) by two layers (given by W′ and P). The matrix W′ is actually d' filters whose sizes are $k \times k \times c$. These filters produce a d'-dimensional feature map. On this feature map, the d-by-d' matrix P can be implemented as d filters whose sizes are $1 \times 1 \times d'$. So P corresponds to a convolutional layer with a 1×1 spatial support, which maps the d'-dimensional feature map to a d-dimensional one.

Note that the decomposition of $\mathrm{M} = \mathrm{P}\mathrm{Q}^\top$ can be arbitrary. It does not impact the value of $\mathbf{y}$ computed in Eqn. (3). A simple decomposition is the Singular Value Decomposition (SVD) [31]: $\mathrm{M} = \mathrm{U}_{d'}\mathrm{S}_{d'}\mathrm{V}_{d'}{}^\top$, where $\mathrm{U}_{d'}$ and $\mathrm{V}_{d'}$ are d-by-d' column-orthogonal matrices and $\mathrm{S}_{d'}$ is a d'-by-d' diagonal matrix. Then we can obtain $\mathrm{P} = \mathrm{U}_{d'}\mathrm{S}_{d'}^{1/2}$ and $\mathrm{Q} = \mathrm{V}_{d'}\mathrm{S}_{d'}^{1/2}$.

In practice the low-rank assumption does not strictly hold, and the computation in Eqn. (3) is approximate. To find an approximate low-rank subspace, we optimize the following problem:

$$\min_{\mathrm{M}} \sum_i \|(\mathbf{y}_i - \bar{\mathbf{y}}) - \mathrm{M}(\mathbf{y}_i - \bar{\mathbf{y}})\|_2^2, \quad s.t. \quad rank(\mathrm{M}) \le d'. \tag{4}$$

Here $\mathbf{y}_i$ is a response sampled from the feature maps in the training set. This problem can be solved by SVD [31] or actually Principal Component Analysis (PCA): let Y be the d-by-n matrix concatenating n responses with the mean subtracted, compute the eigen-decomposition of the covariance matrix $\mathrm{Y}\mathrm{Y}^\top = \mathrm{U}\mathrm{S}\mathrm{U}^\top$ where U is an orthogonal matrix and S is diagonal, and $\mathrm{M} = \mathrm{U}_{d'}\mathrm{U}_{d'}{}^\top$ where $\mathrm{U}_{d'}$ are the first d' eigenvectors. With the matrix M computed, we can find $\mathrm{P} = \mathrm{Q} = \mathrm{U}_{d'}$.

How good is the low-rank assumption? We sample the responses from a CNN model (with seven convolutional layers, detailed in Section 4) trained on ImageNet. For the responses of each layer, we compute the eigenvalues of their covariance matrix and then plot the sum of the largest eigenvalues (Fig. 2). We see that substantial energy is in a small portion of the largest eigenvectors. For example, in the Conv2 layer ($d = 256$) the first 128 eigenvectors contribute over 99.9 percent energy; in the Conv7 layer ($d = 512$), the first 256 eigenvectors contribute over 95 percent energy. This indicates that we can use a fraction of the filters to precisely approximate the original filters.

The low-rank behavior of the responses $\mathbf{y}$ is because of the low-rank behaviors of the filter weights W and the inputs $\mathbf{x}$. Although the low-rank assumptions about filter weights W have been adopted in recent work [16], [17], we further adopt the low-rank assumptions about the filter inputs $\mathbf{x}$, which are local volumes and have correlations. The responses $\mathbf{y}$ will have lower rank than W and $\mathbf{x}$, so the approximation can be more precise. In our optimization (4), we directly address the low-rank subspace of $\mathbf{y}$.

3.2 Nonlinear Case

Next we investigate the case of using nonlinear units. We use $r(\cdot)$ to denote the nonlinear operator. In this paper we focus on the Rectified Linear Unit (ReLU) [32]: $r(\cdot) = \max(\cdot, 0)$.

Driven by Eqn.(4), we minimize the reconstruction error of the nonlinear responses:

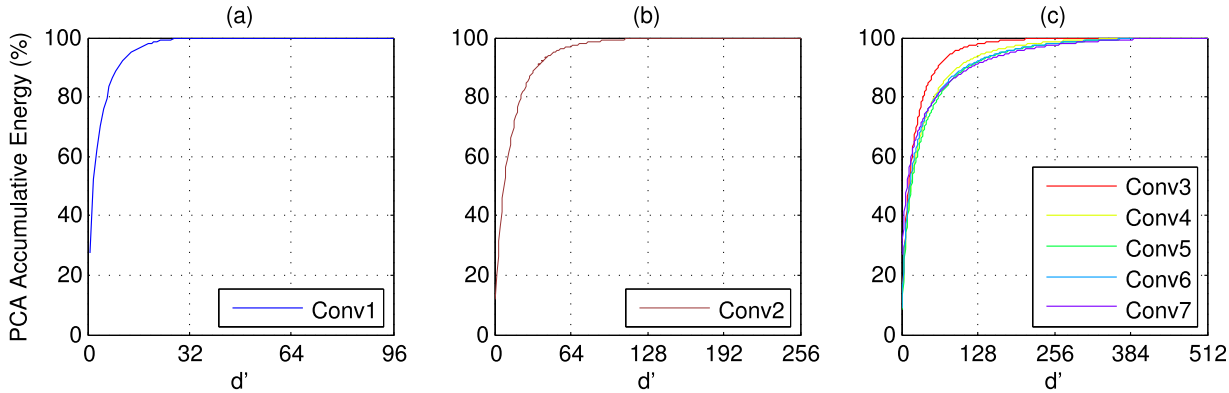


Fig. 2. PCA accumulative energy of the responses in each layer, presented as the sum of largest d' eigenvalues (relative to the total energy when $d' = d$). Here the filter number d is 96 for Conv1, 256 for Conv2, and 512 for Conv3-7 (detailed in Table 1). These figures are obtained from 3,000 randomly sampled training images.

$$\min_{\mathrm{M},\mathbf{b}} \sum_i \|r(\mathbf{y}_i) - r(\mathrm{M}\mathbf{y}_i + \mathbf{b})\|_2^2, \quad s.t. \quad rank(\mathrm{M}) \leq d'. \tag{5}$$

Here $\mathbf{b}$ is a new bias to be optimized, and $r(\mathrm{M}\mathbf{y}+\ \mathbf{b}) = r(\mathrm{MW}\mathbf{x} + \mathbf{b})$ is the nonlinear response computed by the approximated filters.

The above optimization problem is challenging due to the nonlinearity and the low-rank constraint. To find a feasible solution, we relax it as:

$$\min_{\mathrm{M},\mathbf{b},\{\mathbf{z}_i\}} \sum_i \|r(\mathbf{y}_i) - r(\mathbf{z}_i)\|_2^2 + \lambda\|\mathbf{z}_i - (\mathrm{M}\mathbf{y}_i + \mathbf{b})\|_2^2 \quad s.t. \quad rank(\mathrm{M}) \leq d'. \tag{6}$$

Here $\{\mathbf{z}_i\}$ is a set of auxiliary variables of the same size as $\{\mathbf{y}_i\}$. λ is a penalty parameter. If $\lambda \to \infty$, the solution to (6) will converge to the solution to (5) [33]. We adopt an alternating solver, fixing $\{\mathbf{z}_i\}$ and solving for M, $\mathbf{b}$ and vice versa.

(i) **The subproblem of M, $\mathbf{b}$.** In this case, $\{\mathbf{z}_i\}$ are fixed. It is easy to show that $\mathbf{b}$ is solved by $\mathbf{b} = \bar{\mathbf{z}} - \mathrm{M}\bar{\mathbf{y}}$ where $\bar{\mathbf{z}}$ is the mean vector of $\{\mathbf{z}_i\}$. Substituting $\mathbf{b}$ into the objective function, we obtain the problem involving M:

$$\min_{\mathrm{M}} \sum_i \|(\mathbf{z}_i - \bar{\mathbf{z}}) - \mathrm{M}(\mathbf{y}_i - \bar{\mathbf{y}})\|_2^2, s.t. \quad rank(\mathrm{M}) \leq d'. \tag{7}$$

This problem appears similar to Eqn. (4) except that there are two sets of responses.

This optimization problem also has a closed-form solution by Generalized SVD [28], [29], [30]. Let Z be the d-by-n matrix concatenating the vectors of $\{\mathbf{z}_i - \bar{\mathbf{z}}\}$. We rewrite the above problem as:

$$\min_{\mathrm{M}} \|\mathrm{Z} - \mathrm{MY}\|_{\mathrm{F}}^2, \quad s.t. \quad rank(\mathrm{M}) \leq d'. \tag{8}$$

Here $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm. A problem in this form is known as Reduced Rank Regression [28], [29], [30]. This problem belongs to a broader category of *procrustes* problems [28] that have been adopted for various data reconstruction problems [34], [35], [36]. The solution is as follows (see [30]). Let $\hat{\mathrm{M}} = \mathrm{ZY}^\top(\mathrm{YY}^\top)^{-1}$. GSVD [30] is applied on $\hat{\mathrm{M}}$: $\hat{\mathrm{M}} = \mathrm{USV}^\top$, such that U is a d-by-d orthogonal matrix satisfying $\mathrm{U}^\top\mathrm{U} = \mathrm{I}_d$ where I_d is a d-by-d identity matrix, and V is a d-by-d matrix satisfying $\mathrm{V}^\top\mathrm{YY}^\top\mathrm{V} = \mathrm{I}_{\mathrm{d}}$ (called *generalized orthogonality*). Then the solution M to (8) is given by $\mathrm{M} = \mathrm{U}_{d'}\mathrm{S}_{d'}\mathrm{V}_{d'}{}^\top$ where $\mathrm{U}_{d'}$ and $\mathrm{V}_{d'}$ are the first d' columns of U and V and $\mathrm{S}_{d'}$ are the largest d' singular values. One can show that if $\mathrm{Z} = \mathrm{Y}$ (so the problem in (7) becomes (4)), this GSVD solution becomes SVD, i.e., eigen-decomposition of $\mathrm{YY}^\top$.

(ii) **The subproblem of $\{\mathbf{z}_i\}$.** In this case, M and $\mathbf{b}$ are fixed. Then in this subproblem each element z_{ij} of each vector $\mathbf{z}_i$ is independent of any other. So we solve a 1-dimensional optimization problem as follows:

$$\min_{z_{ij}} (r(y_{ij}) - r(z_{ij}))^2 + \lambda(z_{ij} - y'_{ij})^2, \tag{9}$$

where y'_{ij} is the j-th entry of $\mathrm{M}\mathbf{y}_i + \mathbf{b}$. By separately considering $z_{ij} \geq 0$ and $z_{ij} < 0$, we obtain the solution as follows: let

$$z_0 = \min(0, y'_{ij}), \tag{10}$$

$$z_1 = \max\left(0, \frac{\lambda \cdot y'_{ij} + r(y_{ij})}{\lambda + 1}\right), \tag{11}$$

then $z_{ij} = \arg\min_{z_0, z_1} (r(y_{ij}) - r(z_{ij}))^2 + \lambda(z_{ij} - y'_{ij})^2$. Our method is also applicable for other types of nonlinearities. The subproblem in (9) is a one-dimensional nonlinear least squares problem, so can be solved by gradient descent for other $r(\cdot)$.

We alternatively solve (i) and (ii). The initialization is given by the solution to the linear case (4). We warm up the solver by setting the penalty parameter $\lambda = 0.01$ and run 25 iterations. Then we increase the value of λ. In theory, λ should be gradually increased to infinity [33]. But we find that it is difficult for the iterative solver to make progress if λ is too large. So we increase λ to 1, run 25 more iterations, and use the resulting M as our solution. As before, we obtain P and Q by SVD on M.

In experiments, we find that it is sufficient to randomly sample 3,000 images to solve Eqn.(5). It only takes our method **2-5 minutes** in MATLAB solving a layer. This is much faster than SGD-based solvers.

3.3 Asymmetric Reconstruction for Multi-Layer

When each layer is approximated independently, the error of shallower layers will be rapidly accumulated and affect deeper layers. We propose an asymmetric reconstruction method to alleviate this problem.

We apply our method sequentially on each layer, from the shallower layers to the deeper ones. Let us consider a layer whose input feature map is not precise due to the approximation of the previous layer/layers. We denote the approximate input to the current layer as $\hat{\mathbf{x}}$. For the training data, we can still compute its non-approximate responses as $\mathbf{y} = \mathrm{W}\mathbf{x}$. So we can optimize an "asymmetric" version of (5):

$$\min_{\mathrm{M},\mathbf{b}} \sum_i \|r(\mathrm{W}\mathbf{x}_i) - r(\mathrm{MW}\hat{\mathbf{x}}_i + \mathbf{b})\|_2^2, \quad s.t. \quad rank(\mathrm{M}) \leq d'. \tag{12}$$

In the first term $r(\mathrm{W}\mathbf{x}) = r(\mathbf{y})$ is the non-approximate output of this layer. In the second term, $\hat{\mathbf{x}}_i$ is the approximated input to this layer, and $r(\mathrm{MW}\hat{\mathbf{x}}_i + \mathbf{b})$ is the approximated output of this layer. In contrast to using $\mathbf{x}$ (or $\hat{\mathbf{x}}$) for both terms, this asymmetric formulation faithfully incorporates the two actual terms before/after the approximation of this layer. The optimization problem in (12) can be solved using the same algorithm as for (5).

3.4 Rank Selection for Whole-Model Acceleration

In the above, the optimization is based on a target d' of each layer. d' is the only parameter that determines the complexity of an accelerated layer. But given a desired speedup ratio of the *whole model*, we need to determine the proper rank d' used for each layer. One may adopt a uniform speedup ratio

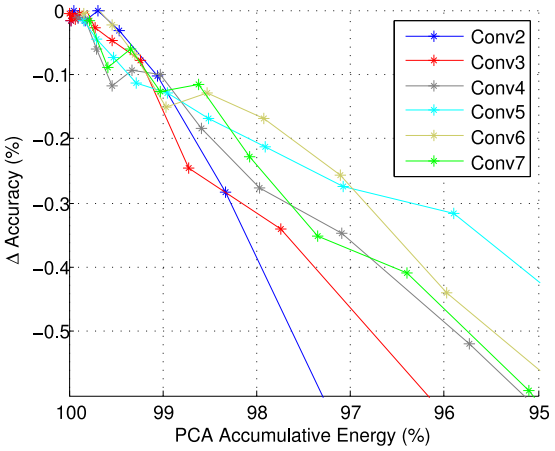


Fig. 3. PCA accumulative energy and the accuracy rates (top-5). Here the accuracy is evaluated using the linear solution (the nonlinear solution has a similar trend). Each layer is evaluated independently, with other layers not approximated. The accuracy is shown as the difference to no approximation.

for each layer. But this is not an optimal solution, because the layers are not equally redundant.

We empirically observe that the PCA energy after approximations is roughly related to the classification accuracy. To verify this observation, in Fig. 3 we show the classification accuracy (represented as the difference to no approximation) versusthe PCA energy. Each point in this figure is empirically evaluated using a reduced rank d'. 100 percent energy means no approximation and thus no degradation of classification accuracy. Fig. 3 shows that the classification accuracy is roughly linear on the PCA energy.

To simultaneously determine the reduced ranks of all layers, we further assume that the whole-model classification accuracy is roughly related to the product of the PCA energy of all layers. More formally, we consider this objective function:

$$\mathcal{E} = \prod_l \sum_{a=1}^{d'_l} \sigma_{l,a}. \tag{13}$$

Here $\sigma_{l,a}$ is the a-th largest eigenvalue of the layer l, and $\sum_{a=1}^{d'_l} \sigma_{l,a}$ is the PCA energy of the largest d'_l eigenvalues in the layer l. The product $\prod_l$ is over all layers to be approximated. The objective $\mathcal{E}$ is assumed to be related to the accuracy of the approximated whole network. Then we optimize this problem:

$$\max_{\{d'_l\}} \mathcal{E}, \qquad s.t. \quad \sum_l \frac{d'_l}{d_l} C_l \leq C. \tag{14}$$

Here d_l is the original number of filters in the layer l, and C_l is the original time complexity of the layer l. So $\frac{d'_l}{d_l} C_l$ is the complexity after the approximation. C is the total complexity after the approximation, which is given by the desired speedup ratio. This optimization problem means that we want to maximize the accumulated energy subject to the time complexity constraint.

The problem in (14) is a combinatorial problem [37]. So we adopt a greedy strategy to solve it. We initialize d'_l as d_l, and consider the set $\{\sigma_{l,a}\}$. In each step we remove an eigenvalue σ_{l,d'_l} from this set, chosen from a certain layer l. The relative reduction of the objective is $\triangle\mathcal{E}/\mathcal{E} = \sigma_{l,d'}/\sum_{a=1}^{d'_l} \sigma_{l,a}$, and the reduction of complexity is $\triangle C = \frac{1}{d_l} C_l$. Then we define a measure as $\frac{\triangle\mathcal{E}/\mathcal{E}}{\triangle C}$. The eigenvalue σ_{l,d'_l} that has the smallest value of this measure is removed. Intuitively, this measure favors a small reduction of $\triangle\mathcal{E}/\mathcal{E}$ and a large reduction of complexity $\triangle C$. This step is greedily iterated, until the constraint of the total complexity is achieved.

3.5 Higher-Dimensional Decomposition

In our formulation, we focus on reducing the channels (from d to d'). There are algorithmic advantages of operating on the channel dimension. Firstly, this dimension can be easily controlled by the rank constraint $rank(\mathrm{M}) \leq d'$. This constraint enables closed-form solutions, e.g., SVD or GSVD. Secondly, the optimized low-rank projection M can be exactly decomposed into low-dimensional filters (P and Q). These simple and closed-form solutions can produce good results using a very small subset of training images (3,000 out of one million).

On the other hand, compared with decomposition methods that operate on multiple dimensions (spatial and channel) [17], our method has to use a smaller d' to approach a given speedup ratio, which might limit the accuracy of our method. To avoid d' being too small, we further propose to combine our solver with Jaderberg et al., spatial decomposition. Thanks to our asymmetric reconstruction, our method can effectively alleviate the accumulated error for the multi-decomposition.

To determined the decomposed architecture (but not yet the weights), we first use our method to decompose all conv layers of a model. This involves the rank selection of d' for all layers. Then we apply Jaderberg et al., method to further decompose the resulting $k \times k$ layers ($k > 1$) into $k \times 1$ and $1 \times k$ filters. The first $k \times 1$ layer has d'' output channels depending on the speedup ratio. In this way, an original layer of ($k \times k$, d) is decomposed into three layers of ($k \times 1$, d''), ($1 \times k$, d'), and (1×1, d). For a speedup ratio r, we let each method contribute a speedup of $\sqrt{r}$.

With the decomposed architecture determined, we solve for the weights of the decomposed layers. Given their order as above, we first optimize the ($k \times 1$, d'') and ($1 \times k$, d) layers using "filter reconstruction" [17] (we will discuss "data reconstruction" later). Then we adopt our solution on the ($1 \times k$, d) layer and optimize for the ($1 \times k$, d') and (1×1, d) layers. We use our asymmetric reconstruction in Eqn.(12). In the $r(\mathrm{M}\mathbf{\hat{x}} + \mathbf{b})$ term, $\hat{x}$ is the approximated input to this $1 \times k$ layer, and the $r(\mathrm{W}\mathbf{x}) = r(\mathbf{y})$ term is still the true response of the original $k \times k$ layer without any decomposition. The approximation error of the spatial decomposition will also be addressed by our asymmetric reconstruction, which is important to alleviate accumulated error. We term this as "asymmetric (3d)" in the following.

3.6 Fine-Tuning

With any approximated whole model, we may "fine-tune" this model end-to-end in the ImageNet training data. This process is similar to training a classification network with the approximated model as the initialization.

However, we empirically find that fine-tuning is very sensitive to the initialization (given by the approximated

TABLE 1
The Architecture of the SPP-10 Model [7]

layer	filter size	# channels	# filters	stride	output size	complexity (%)	# of zeros
Conv1	7×7	3	96	2	109×109	3.8	0.49
Pool1	3×3			3	37×37		
Conv2	5×5	96	256	1	35×35	17.3	0.62
Pool2	2×2			2	18×18		
Conv3	3×3	256	512	1	18×18	8.8	0.60
Conv4	3×3	512	512	1	18×18	17.5	0.69
Conv5	3×3	512	512	1	18×18	17.5	0.69
Conv6	3×3	512	512	1	18×18	17.5	0.68
Conv7	3×3	512	512	1	18×18	17.5	0.95

It has seven conv layers and three fc layers Each layer (except the last fc) is followed by ReLU. The final conv layer is followed by a spatial pyramid pooling layer [7] that have four levels ($\{6 \times 6, 3 \times 3, 2 \times 2, 1 \times 1\}$, totally 50 bins). The resulting 50×512-d is fed into the 4096-d fc layer (fc6), followed by another 4096-d fc layer (fc7) and a 1,000-way softmax layer. The column "complexity" is the theoretical time complexity, shown as relative numbers to the total convolutional complexity. The column "# of zeros" is the relative portion of zero responses, which shows the "sparsity" of the layer.

model) and the learning rate. If the initialization is poor and the learning rate is small, the fine-tuning is easily trapped in a poor local optimum and makes little progress. If the learning rate is large, the fine-tuning process behaves very similar to training the decomposed architecture "from scratch" (as we will discuss later). A large learning rate may jump out of the initialized local optimum, and the initialization appears to be "forgotten".

Fortunately, our method has achieved very good accuracy even without fine-tuning as we will show by experiments. With our approximated model as the initialization, the fine-tuning with a sufficiently small learning rate is able to further improve the results. In our experiments, we use a learning rate of 1e-5 and a mini-batch size of 128, and fine-tune the models for 5 epochs in the ImageNet training data.

We note that in the following the results are **without** fine-tuning unless specified.

4 Experiments

We comprehensively evaluate our method on two models. The first model is a 10-layer model of "SPPnet (OverFeat-7)" in [7], which we denote as "**SPP-10**". This model (detailed in Table 1) has a similar architecture to the OverFeat model [6] but is deeper. It has 7 conv layers and 3 fc layers. The second model is the publicly available **VGG-16** model [1][1] that has 13 conv layers and 3 fc layers. **SPP-10** won the 3-rd place and **VGG-16** won the 2-nd place in ILSVRC 2014 [19].

We evaluate the "top-5 error" using single-view testing. The view is the center 224×224 region cropped from the resized image whose shorter side is 256. The single-view error rate of SPP-10 is 12.51 percent on the ImageNet validation set, and VGG-16 is 10.09 percent in our testing (which is consistent with the number reported by [1]).[2] These numbers serve as the references for the increased error rates of our approximated models.

4.1 Experiments with SPP-10

We first evaluate the effect of our each step on the SPP-10 model by a series of controlled experiments. Unless specified, we do not use the 3-d decomposition.

Single-Layer: Linear versus Nonlinear

In this section we evaluate the single-layer performance. When evaluating a single approximated layer, the remaining layers are unchanged and not approximated. The speedup ratio (involving that single layer only) is shown as the theoretical ratio computed by the complexity.

In Fig. 4 we compare the performance of our linear solution (4) and nonlinear solution (6). The performance is displayed as *increase of error rates* (decrease of accuracy) versus the speedup ratio of that layer. Fig. 4 shows that the nonlinear solution consistently performs better than the linear solution. In Table 1, we show the sparsity (the portion of zero activations after ReLU) of each layer. A zero activation is due to the truncation of ReLU. The sparsity is over 60 percent for Conv2-7, indicating that the ReLU takes effect on a substantial portion of activations. This explains the discrepancy between the linear and nonlinear solutions. Especially, the Conv7 layer has a sparsity of 95 percent, so the advantage of the nonlinear solution is more obvious.

Fig. 4 also shows that when accelerating only a single layer by $2\times$, the increased error rates of our solutions are rather marginal or negligible. For the Conv2 layer, the error rate is increased by < 0.1 percent; for the Conv3-7 layers, the error rate is increased by $\approx 0.2\%$.

We also notice that for Conv1, the degradation is negligible near $2\times$ speedup ($1.8\times$ corresponds to $d' = 32$). This can be explained by Fig. 2a: the PCA energy has little loss when $d' \geq 32$. But the degradation can grow quickly for larger speedup ratios, because in this layer the channel number $c = 3$ is small and d' needs to be reduced drastically to achieve the speedup ratio. So in the following whole-model experiments of SPP-10, we will use $d' = 32$ for Conv1.

Multi-Layer: Symmetric versus Asymmetric

Next we evaluate the performance of asymmetric reconstruction as in the problem (12). We demonstrate approximating two layers or three layers. In the case of two layers, we show the results of approximating Conv6 and 7; and in the case of three layers, we show the results of approximating Conv5-7 or Conv2-4. The comparisons are consistently observed for other cases of multi-layer.

We sequentially approximate the layers involved, from a shallower one to a deeper one. In the asymmetric version (12), $\hat{\mathbf{x}}$ is from the output of the previous approximated layer (if any), and $\mathbf{x}$ is from the output of the previous non-approximate layer. In the symmetric version (5), we use $\mathbf{x}$ for both terms. We have also tried another symmetric

1. www.robots.ox.ac.uk/vgg/research/very_deep/
2. http://www.vlfeat.org/matconvnet/pretrained/

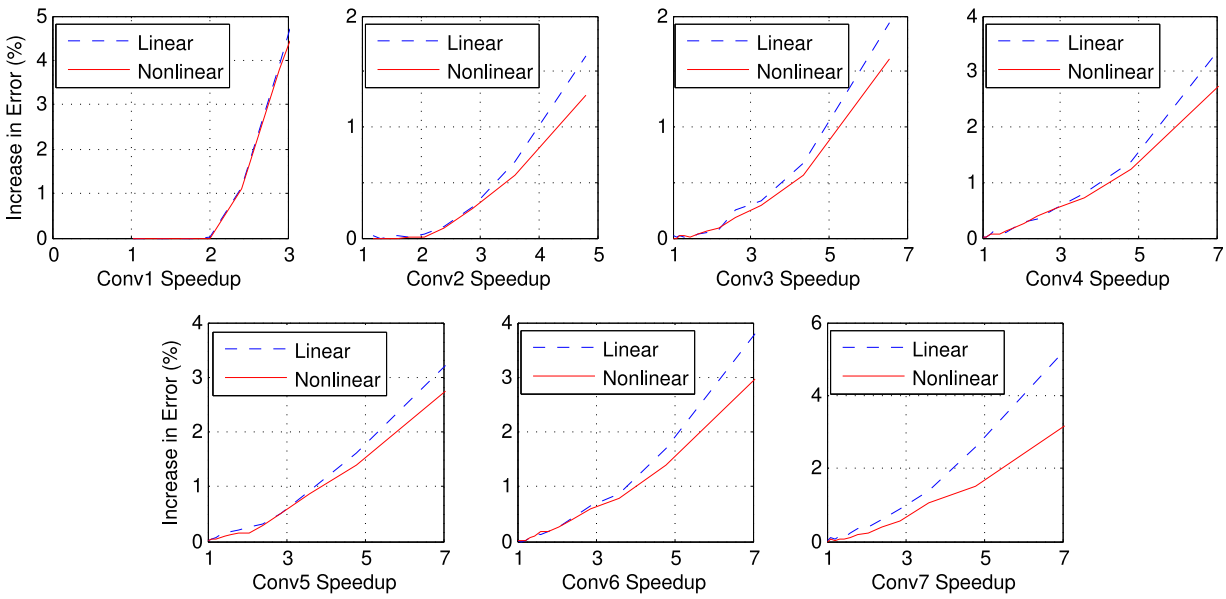


Fig. 4. **Linear versus Nonlinear** for SPP-10: single-layer performance of accelerating Conv1 to Conv7. The speedup ratios are computed by the theoretical complexity of that layer. The error rates are top-5 single-view, and shown as the increase of error rates compared with no approximation (*smaller is better*).

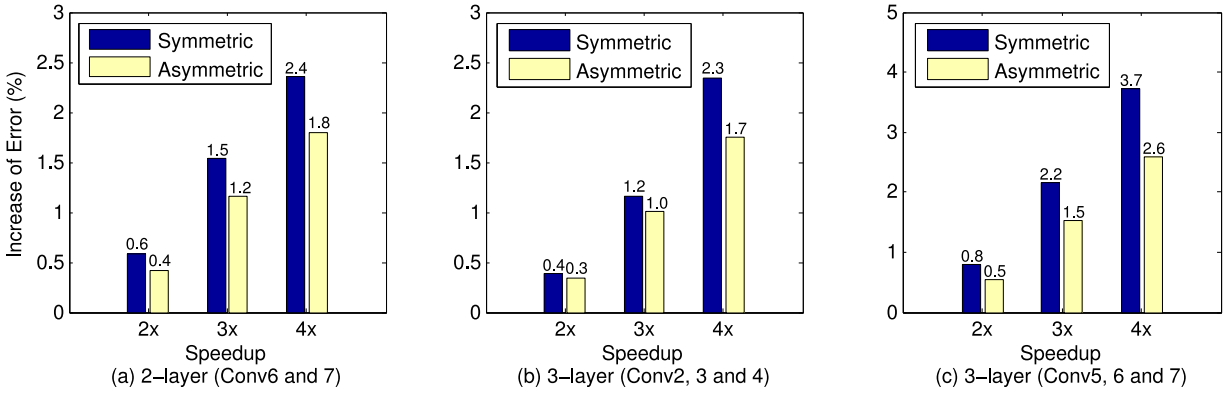


Fig. 5. **Symmetric versus Asymmetric** for SPP-10: the cases of two-layer and three-layer approximation. The speedup is computed by the complexity of the layers approximated. (a) Approximation of Conv6 & 7. (b) Approximation of Conv2, 3, & 4. (c) Approximation of Conv5, 6, & 7.

TABLE 2
Whole-Model Acceleration with/without Rank Selection for SPP-10

speedup	rank sel.	Conv1	Conv2	Conv3	Conv4	Conv5	Conv6	Conv7	err. ↑ %
2×	no	32	110	199	219	219	219	219	1.18
2×	**yes**	32	83	182	211	239	237	253	**0.93**
2.4×	no	32	96	174	191	191	191	191	1.77
2.4×	**yes**	32	74	162	187	207	205	219	**1.35**
3×	no	32	77	139	153	153	153	153	2.56
3×	**yes**	32	62	138	149	166	162	167	**2.34**
4×	no	32	57	104	115	115	115	115	4.32
4×	**yes**	32	50	112	114	122	117	119	**4.20**
5×	no	32	46	83	92	92	92	92	6.53
5×	**yes**	32	41	94	93	98	92	90	**6.47**

The solver is the asymmetric version. The speedup ratios shown here involve all convolutional layers (Conv1-Conv7). We fix $d' = 32$ in Conv1. In the case of no rank selection, the speedup ratio of each other layer is the same. Each column of Conv1-7 shows the rank d' used, which is the number of filters after approximation. The error rates are top-5 single-view, and shown as the increase of error rates compared with no approximation.

version of using $\hat{\mathbf{x}}$ for both terms, and found this symmetric version is even worse.

Fig. 5 shows the comparisons between the symmetric and asymmetric versions. The asymmetric solution has significant improvement over the symmetric solution. For example, when only three layers are approximated simultaneously (like Fig. 5c), the improvement is over 1.0 percent when the speedup is 4×. This indicates that the

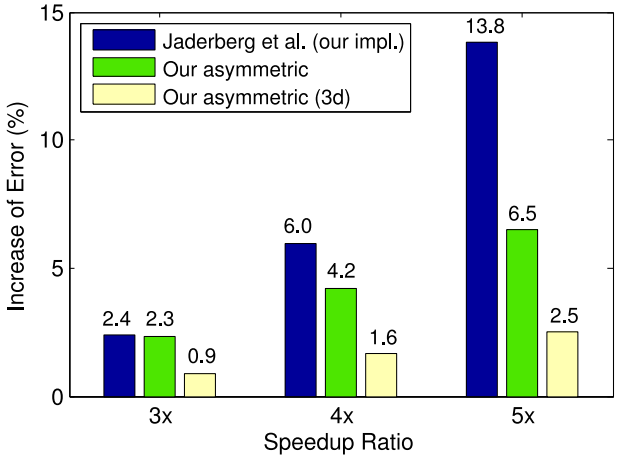


Fig. 6. Comparisons with Jaderberg et al. spatial decomposition method [17] for SPP-10. The speedup ratios are theoretical speedups of the whole model. The error rates are top-5 single-view, and shown as **the increase of error rates** compared with no approximation (*smaller is better*).

accumulative error rate due to multi-layer approximation can be effectively reduced by the asymmetric version.

When more and all layers are approximated simultaneously (as below), if without the asymmetric solution, the error rates will increase more drastically.

Whole-Model: with/without Rank Selection

In Table 2 we show the results of whole-model acceleration. The solver is the asymmetric version. For Conv1, we fix $d' = 32$. For other layers, when the rank selection is not used, we adopt the same speedup ratio on each layer and determine its desired rank d' accordingly. When the rank selection is used, we apply it to select d' for Conv2-7. Table 2 shows that the rank selection consistently outperforms the counterpart without rank selection. The advantage of rank selection is observed in both linear and nonlinear solutions.

In Table 2 we notice that the rank selection often chooses a higher rank d' (than the no rank selection) in Conv5-7. For example, when the speedup is $3\times$, the rank selection assigns $d' = 167$ to Conv7, while this layer only requires $d' = 153$ to achieve $3\times$ single-layer speedup of itself. This can be explained by Fig. 2c. The energy of Conv5-7 is less concentrated, so these layers require higher ranks to achieve good approximations.

As we will show, the rank selection is more prominent for VGG-16 because of its diversity of layers.

Comparisons with Jaderberg et al. method [17]

We compare with Jaderberg et al., method [17], which is a recent state-of-the-art solution to efficient evaluation.

TABLE 4
Comparisons with the Same Decomposed Architecture Trained from Scratch

model	top-5 err. (1-view)	increased err. (1-view)
SPP-10 [7]	12.5	-
our asym. 3d ($4\times$)	14.1	1.6
from scratch	16.9	4.4

Although our *decomposition* shares some high-level motivations as [17], we point out that our *optimization* strategy is different with [17] and is important for accuracy, especially for *very deep* models that previous acceleration methods rarely addressed.

Jaderberg et al., method [17] decomposes a $k \times k$ spatial support into a cascade of $k \times 1$ and $1 \times k$ spatial supports. A channel-dimension reduction is also considered. Their optimization method focuses on the linear reconstruction error. In the paper of [17], their method is only evaluated on a single layer of an OverFeat network [6] for ImageNet.

Our comparisons are based on our implementation of [17]. We use the *Scheme 2* decomposition in [17] and its "filter reconstruction" version (as we explain below), which is used for ImageNet as in [17]. Our reproduction of the filter reconstruction in [17] gives a $2\times$ single-layer speedup on Conv2 of SPP-10 with 0.2 percent increase of error. As a reference, in [17] it reports 0.5 percent increase of error on Conv2 under a $2\times$ single-layer speedup, evaluated on another OverFeat network [6] similar to SPP-10.

It is worth discussing our implementation of Jaderberg et al., [17] "data reconstruction" scheme, which was suggested to use SGD and backpropagation for optimization. In our reproduction, we find that data reconstruction works well for the *character classification* task as studied in [17]. However, we find it nontrivial to make data reconstruction work for large models trained for ImageNet. We observe that the learning rate needs to be carefully chosen for the SGD-based data reconstruction to converge (as also reported independently in [18] for another decomposition), and when the training starts to converge, the results are still sensitive to the initialization (for which we have tried Gaussian distributions of a wide range of variances). We conjecture that this is because the ImageNet dataset and models are more complicated, and using SGD to regress a single layer may be sensitive to multiple local optima. In fact, Jaderberg et al., [17] only report "filter reconstruction" results of a single layer on ImageNet. For these reasons, our implementation of Jaderberg et al., method on ImageNet

TABLE 3
Comparisons of Absolute Performance of SPP-10

model	speedup solution	top-5 err. (1-view)	CPU (ms)	GPU (ms)
SPP-10 [7]	-	12.5	930	7.67
SPP-10 ($4\times$)	Jaderberg et al. [17] (our impl.)	18.5	278 ($3.3\times$)	2.41 ($3.2\times$)
	our asym.	16.7	271 ($3.4\times$)	2.62 ($2.9\times$)
	our asym. (3d)	14.1	267 ($3.5\times$)	2.32 ($3.3\times$)
	our asym. (3d) FT	**13.8**	267 ($3.5\times$)	2.32 ($3.3\times$)
AlexNet [4]	-	18.8	273	2.37

The top-5 error is the absolute value. The running time is a single view on a CPU (single thread, with SSE) or a GPU. The accelerated models are those of $4\times$ theoretical speedup (Fig. 6). On the brackets are the actual speedup ratios.

TABLE 5
The Architecture of the VGG-16 Model [1]

layer	filter size	# channels	# filters	stride	output size	complexity (%)	# of zeros
$Conv1_1$	3×3	3	64	1	224×224	0.6	0.48
$Conv1_2$	3×3	64	64	1	224×224	12.0	0.32
Pool1	3×3			2	112×112		
$Conv2_1$	3×3	64	128	1	112×112	6.0	0.35
$Conv2_2$	3×3	128	128	1	112×112	12.0	0.52
Pool2	2×2			2	56×56		
$Conv3_1$	3×3	128	256	1	56×56	6.0	0.48
$Conv3_2$	3×3	256	256	1	56×56	12.1	0.48
$Conv3_3$	3×3	256	256	1	56×56	12.1	0.70
Pool3	2×2			2	28×28		
$Conv4_1$	3×3	256	512	1	28×28	6.0	0.65
$Conv4_2$	3×3	512	512	1	28×28	12.1	0.70
$Conv4_3$	3×3	512	512	1	28×28	12.1	0.87
Pool4	2×2			2	14×14		
$Conv5_1$	3×3	512	512	1	14×14	3.0	0.76
$Conv5_2$	3×3	512	512	1	14×14	3.0	0.80
$Conv5_3$	3×3	512	512	1	14×14	3.0	0.93

It has 13 conv layers and 3 fc layers. The column "complexity" is the theoretical time complexity, shown as relative numbers to the total convolutional complexity. The column "# of zeros" is the relative portion of zero responses, which shows the "sparsity" of the layer.

TABLE 6
Whole-Model Acceleration with/without Rank Selection for VGG-16

speedup	rank sel.	$C1_1$	$C1_2$	$C2_1$	$C2_2$	$C3_1$	$C3_2$	$C3_3$	$C4_1$	$C4_2$	$C4_3$	$C5_1$	$C5_2$	$C5_3$	err. ↑ %
2×	no	64	28	52	57	104	115	115	209	230	230	230	230	230	0.99
2×	**yes**	64	18	41	50	94	96	116	207	213	260	467	455	442	**0.28**
3×	no	64	19	34	38	69	76	76	139	153	153	153	153	153	3.25
3×	**yes**	64	15	31	34	68	64	75	134	126	146	312	307	294	**1.66**
4×	no	64	14	26	28	52	57	57	104	115	115	115	115	115	6.38
4×	**yes**	64	11	25	28	52	46	56	104	92	100	232	224	214	**3.84**

The solver is the asymmetric version. The speedup ratios shown here involve all convolutional layers. We do not accelerate $Conv1_1$. In the case of no rank selection, the speedup ratio of each other layer is the same. Each column of $C1_2$-$C5_3$ shows the rank d' used, which is the number of filters after approximation. The error rates are top-5 single-view, and shown as the increase of error rates compared with no approximation.

models is based on filter reconstruction. We believe that these issues have not be settled and need to be investigated further, and accelerating deep networks does not just involve *decomposition* but also the way of *optimization*.

In Fig. 6 we compare our method with Jaderberg et al., [17] for whole-model speedup. For whole-model speedup of [17], we implement their method sequentially on Conv2-7 using the same speedup ratio.[3] The speedup ratios are the theoretical complexity ratios involving all convolutional layers. Our method is the asymmetric version and with rank selection. Fig. 6 shows that when the speedup ratios are large (4× and 5×), our method outperforms Jaderberg et al., method significantly. For example, when the speedup ratio is 4×, the increased error rate of our method is 4.2 percent, while Jaderberg et al. is 6.0 percent. Jaderberg et al. result degrades quickly when the speedup ratio is getting large, while ours degrades slowly. This suggests the effects of our method for reducing accumulative error.

3. We do not apply Jaderberg et al., method [17] on Conv1, because this layer has a small number of input channels (3), and the first $k \times 1$ decomposed layer can only have a very small number of filters (e.g., 5) to approach a speedup ratio (e.g., 4×). Also note that the speedup ratio is about all conv layers, and because Conv1 is not accelerated, other layers will have a slightly larger speedup.

We further compare with our asymmetric version using 3d decomposition (Section 3.5). In Fig. 6 we show the results "*asymmetric (3d)*". Fig. 6 shows that this strategy leads to significantly smaller increase of error. For example, when the speedup is 5×, the error is increased by only 2.5 percent. Our asymmetric solver effectively controls the accumulative error even if the multiple layers are decomposed extensively, and the 3d decomposition is easier to achieve a certain speedup ratio.

For completeness, we also evaluate our approximation method on the *character classification* model released by [17]. Our asymmetric (3d) solution achieves 4.5× speedup with

TABLE 7
Accelerating the **VGG-16** Model [1] Using a Speedup Ratio of 3×, 4×, or 5×

	increase of top-5 error (1-view)		
speedup ratio	3×	4×	5×
Jaderberg et al. [17] (our impl.)	2.3	9.7	29.7
our asym. (3d)	0.4	0.9	2.0
our asym. (3d) FT	**0.0**	**0.3**	**1.0**

The top-5 error rate (1-view) of the VGG-16 model is 10.1 percent. This table shows the increase of error on this baseline.

TABLE 8
Absolute Performance of Accelerating the **VGG-16** Model [1]

model	speedup solution	top-5 error (1-view)	CPU (ms)	GPU (ms)
VGG-16 [1]	-	10.1	3287	18.60
VGG-16 (4×)	Jaderberg et al. [17] (our impl.)	19.8	875 (3.8×)	6.40 (2.9×)
	our asym.	13.9	875 (3.8×)	7.97 (2.3×)
	our asym. (3d)	11.0	860 (3.8×)	6.30 (3.0×)
	our asym. (3d) FT	**10.4**	858 (3.8×)	6.39 (2.9×)

The top-5 error is the absolute value. The running time is a single view on a CPU (single thread, with SSE) or a GPU. The accelerated models are those of 4× theoretical speedup (Table 7). On the brackets are the actual speedup ratios.

only a drop of 0.7 percent in classification accuracy, which is better than the 1 percent drop for the same speedup reported by [17].

Comparisons with Training from Scratch

The architecture of the approximated model can also be trained "*from scratch*" on the ImageNet dataset. One hypothesis is that the underlying architecture is sufficiently powerful, and the acceleration algorithm might be not necessary. We show that this hypothesis is premature.

We directly train the model of the same architecture as the decomposed model. The decomposed model is much deeper than the original model (each layer replaced by three layers), so we adopt the initialization method in [38] otherwise it is not easy to converge. We train the model for 100 epochs. We follow the common practice in [7], [39] of training ImageNet models.

The comparisons are in Table 4. The accuracy of the model trained from scratch is worse than that of our accelerated model by a considerable margin (2.8 percent). These results indicate that the accelerating algorithms can effectively digest information from the trained models. They also suggest that the models trained from scratch have much redundancy.

Comparisons of Absolute Performance

Table 3 shows the comparisons of the absolute performance of the accelerated models. We also evaluate the AlexNet [4] which is similarly fast as our accelerated 4× models. The comparison is based on our re-implementation of AlexNet. Our AlexNet is the same as in [4] except that the GPU splitting is ignored. Our re-implementation of this model has top-5 single-view error rate as 18.8 percent (10-view top-5 16.0 percent and top-1 37.6 percent). This is better than the one reported in [4].[4]

The models accelerated by our asymmetric (3d) version have 14.1 and 13.8 percent top-5 error, without and with fine-tuning. This means that the accelerated model has 5.0 percent lower error than AlexNet, while its speed is nearly the same as AlexNet.

Table 3 also shows the actual running time per view, on a C++ implementation and Intel i7 CPU (2.9 GHz) or Nvidia K40 GPU. In our CPU version, our method has actual speedup ratios (3.5×) close to theoretical speedup ratios (4.0×). This overhead mainly comes from the fc and other layers. In our GPU version, the actual speedup ratio is about 3.3×. An accelerated model is less easy for parallelism in a GPU, so the actual ratio is lower.

4. In [4] the 10-view error is top-5 18.2 percent and top-1 40.7 percent.

4.2 Experiments with VGG-16

The very deep VGG models [1] have substantially improved a wide range of visual recognition tasks, including object detection [2], [9], [10], [11], semantic segmentation [12], [13], [14], [40], [41], image captioning [42], [43], [44], video/action recognition [45], image question answering [46], texture recognition [47], etc. Considering the big impact yet slow speed of this model, we believe it is of practical significance to accelerate this model.

Accelerating VGG-16 for ImageNet Classification

Firstly we discover that our whole-model *rank selection* is particularly important for accelerating VGG-16. In Table 6 we show the results without/with rank selection. No 3d decomposition is used in this comparison. For a 4× speedup, the rank selection reduces the increased error from 6.38 to 3.84 percent. This is because of the **greater diversity of layers** in VGG-16 (Table 5). Unlike SPP-10 (or other shallower models [4], [5]) that repeatedly applies 3×3 filters on the same feature map size, the VGG-16 model applies them more evenly on five feature map sizes (224, 112, 56, 28, and 14). Besides, as the filter numbers in Conv5_1-5_3 are not increased, the time complexity of Conv5_1-5_3 is smaller than others. The selected ranks d' in Table 6 show their adaptivity - e.g., the layers Conv5_1 to Conv5_3 keep more filters, because they have small time complexity and it is not a good trade-off to compactly reduce them. The whole-model rank selection is a key to maintain a high accuracy for accelerating VGG-16.

In Table 7 we evaluate our method on VGG-16 for ImageNet classification. Here we evaluate our asymmetric 3d version (without or with fine-tuning). We evaluate challenging speedup ratios of 3×, 4× and 5×. The ratios are those of the theoretical speedups of all 13 conv layers.

Somewhat surprisingly, our method has demonstrated compelling results for this very deep model, even without fine-tuning. Our no-fine-tuning model has a 0.9 percent increase of one-view top-5 error for a speedup ratio of 4×. On the contrary, the previous method [17] suffers greatly from the increased depth because of the rapidly accumulated error of multiple approximated layers. After fine-tuning, our model has a 0.3 percent increase of one-view top-5 error for a 4× speedup. This degradation is even lower than that of the shallower model of SPP-10. This suggests that the information in the very deep VGG-16 model is highly redundant, and our method is able to effectively digest it.

Table 8 and Fig. 7 shows the actual versustheoretical speedup ratios of VGG-16 using CPU and GPU implementations. The CPU speedup ratios are very close to the theoretical ratios. The GPU implementation, which is based on the

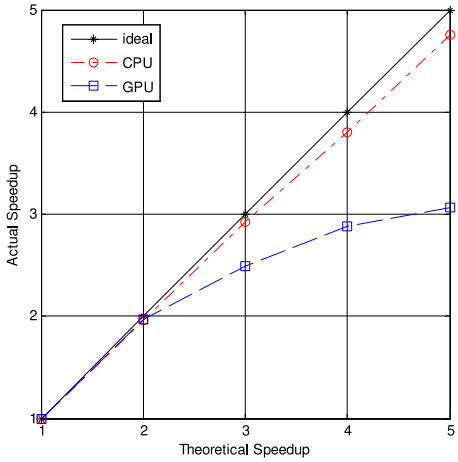


Fig. 7. Actual versus theoretical speedup ratios of VGG-16 using CPU and GPU implementations.

standard Caffe library [48], exhibits a gap between actual versus theoretical ratios (as is also witnessed in [49]). GPU speedup ratios are more sensitive to specialized implementation, and the generic Caffe kernels are not optimized for some layers (*e. g.*, 1×1, 1×3, and 3×1 convolutions). We believe that a more specially engineered implementation will increase the actual GPU speedup ratio.

Figurnov et al., work [49] is one of few existing works that present results of accelerating the whole model of VGG-16. They report increased top-5 one-view error rates of 3.4 and 7.1 percent for actual CPU speedups of 3× and 4× (for 4× theoretical speedup they report a 3.8× actual CPU speedup). Thus our method is substantially more accurate than theirs. Note that results in [49] are after fine-tuning. This suggests that fine-tuning is not sufficient for whole-model acceleration; a good optimization solver for the decomposition is needed.

Accelerating VGG-16 for Object Detection

Current state-of-the-art object detection results [2], [9], [10], [11] mostly rely on the VGG-16 model. We evaluate our accelerated VGG-16 models for object detection. Our method is based on the recent Fast R-CNN [2].

We evaluate on the PASCAL VOC 2007 object detection benchmark [20]. This dataset contains 5k trainval images and 5k test images. We follow the default setting of Fast R-CNN using the publicly released code.[5] We train Fast R-CNN on the trainval set and evaluate on the test set. The accuracy is evaluated by mean Average Precision (mAP).

In our experiments, we first approximate the VGG-16 model on the ImageNet classification task. Then we use the approximated model as the pre-trained model for Fast R-CNN. We use our asymmetric 3d version with fine-tuning. Note that unlike image classification where the conv layers dominate running time, for Fast R-CNN detection the conv layers consume about 70 percent actual running time [2]. The reported speedup ratios are the theoretical speedups about the conv layers only.

Table 9 shows the results of the accelerated models in PASCAL VOC 2007 detection. Our method with a 4× convolution speedup has a graceful degradation of **0.8** percent in mAP. We believe this trade-off between accuracy and speed is of practical importance, because even with the recent advance of fast object detection [2], [7], the feature extraction running time is still considerable.

5. https://github.com/rbgirshick/fast-rcnn

TABLE 9
Object Detection mAP on the PASCAL VOC 2007 Test Set

conv speedup	mAP	ΔmAP
baseline	66.9	–
3×	66.9	0.0
4×	66.1	−0.8
5×	65.2	−1.7

The detector is Fast R-CNN [2] using the pre-trained VGG-16 model.

5 Conclusion

We have presented an acceleration method for very deep networks. Our method is evaluated under whole-model speedup ratios. It can effectively reduce the accumulated error of multiple layers thanks to the nonlinear asymmetric reconstruction. Competitive speedups and accuracy are demonstrated in the complex ImageNet classification task and PASCAL VOC object detection task.

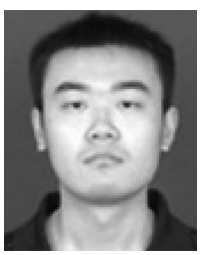
Acknowledgments

Kaiming He is the corresponding author.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015.
[2] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015.
[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.
[4] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.
[5] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional neural networks," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.
[6] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2014.
[7] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 346–361.
[8] 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. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 1–9.
[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 580–587.
[10] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun, "Object detection networks on convolutional feature maps," *arXiv: 1504.06066*, 2015.
[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015.

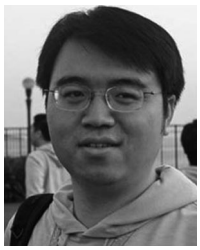
[12] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3431–3440.
[13] J. Dai, K. He, and J. Sun, "Convolutional feature masking for joint object and stuff segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3992–4000.
[14] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 447–456.
[15] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Proc. Deep Learn. Unsupervised Feature Learn. Workshop*, 2011.
[16] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1269–1277.
[17] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proc. British Mach. Vis. Conf.*, 2014.
[18] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," in *Proc. Int. Conf. Learn. Represent.*, 2015.
[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., "Imagenet large scale visual recognition challenge," *arXiv:1409.0575*, 2014.
[20] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge 2007 (VOC2007) results," 2007.
[21] X. Zhang, J. Zou, X. Ming, K. He, and J. Sun, "Efficient and accurate approximations of nonlinear convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, 1984–1992.
[22] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 2754–2761.
[23] N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun, "Fast convolutional nets with FBFFT: A GPU performance evaluation," in *Proc. Int. Conf. Learn. Represent.*, 2015.
[24] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through ffts," *arXiv:1312.5851*, 2013.
[25] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 5353–5360.
[26] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *Proc. Int. Conf. Learn. Represent.*, 2015.
[27] M. D. Collins and P. Kohli, "Memory bounded deep convolutional networks," *arXiv:1412.1442*, 2014.
[28] J. C. Gower and G. B. Dijksterhuis, *Procrustes Problems*, vol. 3. London, U.K.: Oxford Univ. Press, 2004.
[29] Y. Takane and S. Jung, "Generalized constrained redundancy analysis," *Behaviormetrika*, vol. 33, pp. 179–192, 2006.
[30] Y. Takane and H. Hwang, "Regularized linear and kernel redundancy analysis," *Comput. Statist. Data Anal.*, vol. 52, no. 1, pp. 394–405, 2007.
[31] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.
[32] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[33] Y. Wang, J. Yang, W. Yin, and Y. Zhang, "A new alternating minimization algorithm for total variation image reconstruction," *SIAM J. Imaging Sci.*, vol. 1, no. 3, pp. 248–272, 2008.
[34] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2011, pp. 2916–2929.
[35] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2014, pp. 744–755.
[36] Y. Xia, K. He, P. Kohli, and J. Sun, "Sparse projections for high-dimensional binary codes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3332–3339.
[37] C. R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*. Hoboken, NJ, USA: Wiley, 1993.
[38] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *arXiv:1502.01852*, 2015.
[39] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *Proc. British Mach. Vis. Conf.*, 2014.
[40] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in *Proc. Int. Conf. Learn. Represent.*, 2015.
[41] J. Dai, K. He, and J. Sun, "Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation," *arXiv:1503.01640*, 2015.
[42] H. Fang, S. Gupta, F. Iandola, R. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. Platt, et al., "From captions to visual concepts and back," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 1473–1482.
[43] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3128–3137.
[44] X. Chen and C. L. Zitnick, "Learning a recurrent visual representation for image caption generation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 2422–2431.
[45] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using LSTMs," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 843–852.
[46] M. Ren, R. Kiros, and R. Zemel, "Image question answering: A visual semantic embedding model and a new dataset," in *ICML Deep Learn. Workshop*, 2015.
[47] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep convolutional filter banks for texture recognition and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3828–3836.
[48] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv:1408.5093*, 2014.
[49] M. Figurnov, D. Vetrov, and P. Kohli, "PerforatedCNNs: Acceleration through elimination of redundant convolutions," *arXiv:1504.08362*, 2015.



Xiangyu Zhang is currently working toward the PhD degree at the Xi'an Jiaotong University. He is also working as an intern at Microsoft Research Asia. His research interests include computer vision and deep learning.

Jianhua Zou received the Bachelor's, Master's, and Doctor's degrees from the Huazhong University of Science in 1984, 1987, and 1991, respectively. He is currently Professor and Vice President and Deputy Director with the Telecommunications and Systems Engineering Institute, Xi'an Jiaotong University. His main research areas include: control systems and computer networks, multimedia, cognition and knowledge discovery, high voltage insulation monitoring, and complex system analysis. Since 1991, he has been the project leader and has completed about 20 research projects, including National Natural Science projects.

Kaiming He received the BS degree from the Tsinghua University in 2007, and the PhD degree from the Chinese University of Hong Kong in 2011. He is a lead researcher at Microsoft Research Asia. He joined Microsoft Research Asia in 2011. His current research interests are deep learning for visual recognition, including image classification, object detection, and semantic segmentation. He has won the Best Paper Award at CVPR 2009.



Jian Sun received the BS degree, MS degree and PhD degree from the Xian Jiaotong University in 1997, 2000, and 2003, respectively. He is a principal researcher at Microsoft Research Asia. He joined Microsoft Research Asia in July 2003. His current major research interests are computer vision, computational photography, and deep learning. He has won the Best Paper Award at CVPR 2009.