Channel Pruning and Other Methods for Compressing CNN

Yihui He

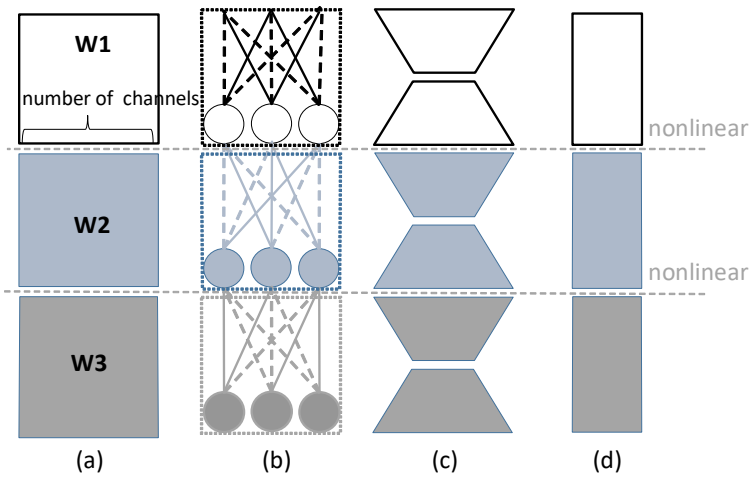
Xi'an Jiaotong Univ.

October 12, 2017

Background

Recent CNN acceleration works fall into three categories:
(1) optimized implementation (e.g., FFT, dictionary)
(2) quantization (e.g., BinaryNet)
(3) structured simplification (convert CNN structure into compact one)
We focuses on the last one.

Background



Structured simplification methods that accelerate CNNs: (a) a network with 3 conv layers. (b) sparse connection deacti-vates some connections between channels. (c) tensor factorization factorizes a convolutional layer into several pieces. (d) channel pruning reduces number of channels in each layer (focus of this paper).

Sparse Connection

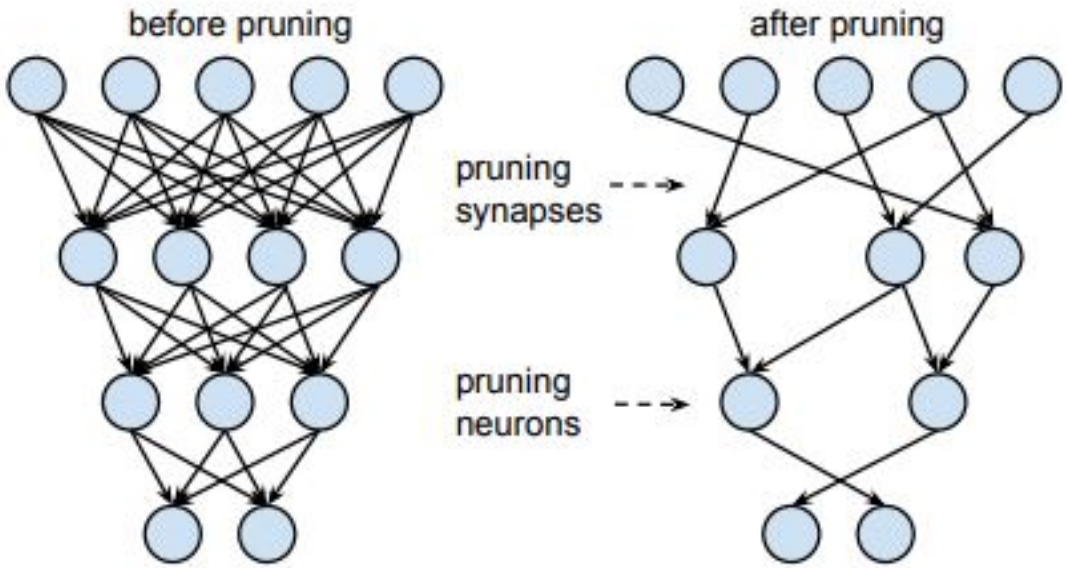


Figure 3: Synapses and neurons before and after pruning.

S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems, pages 1135–1143, 2015

Tensor factorization

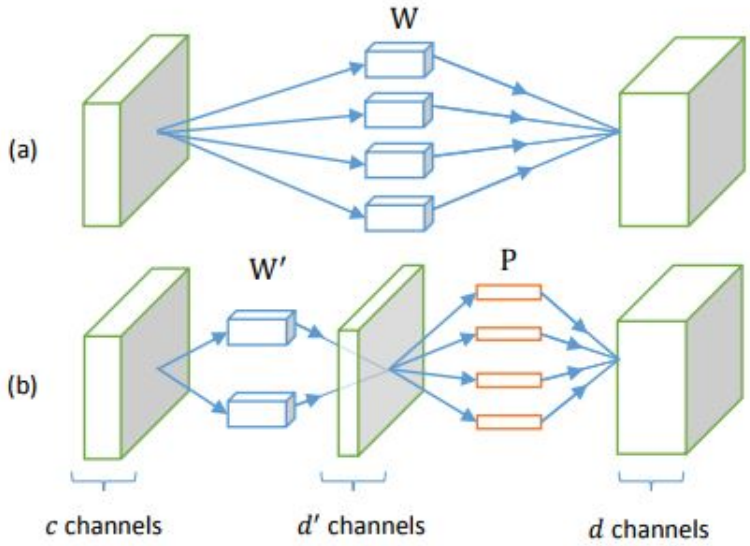
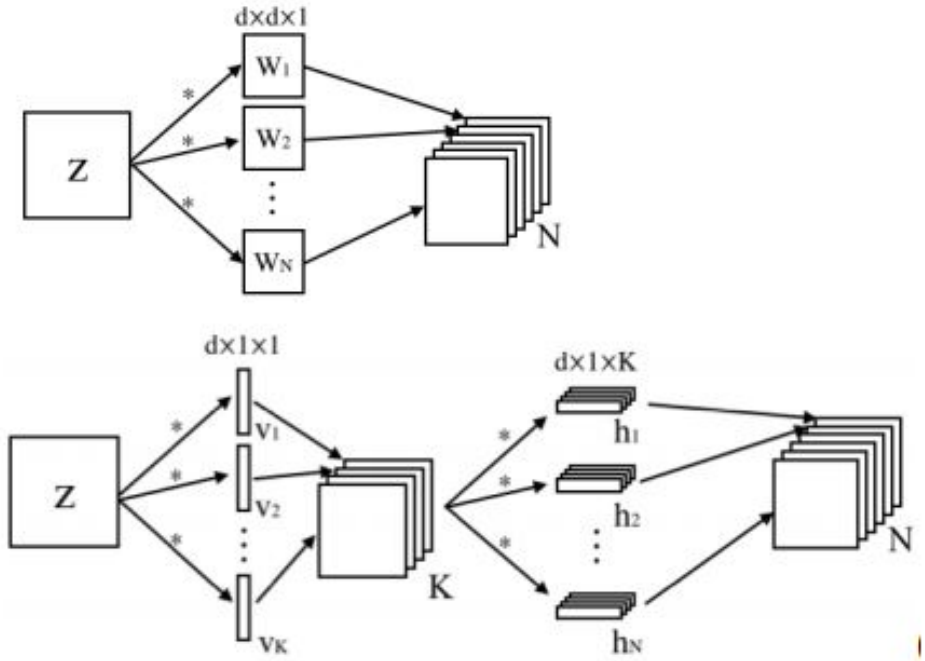


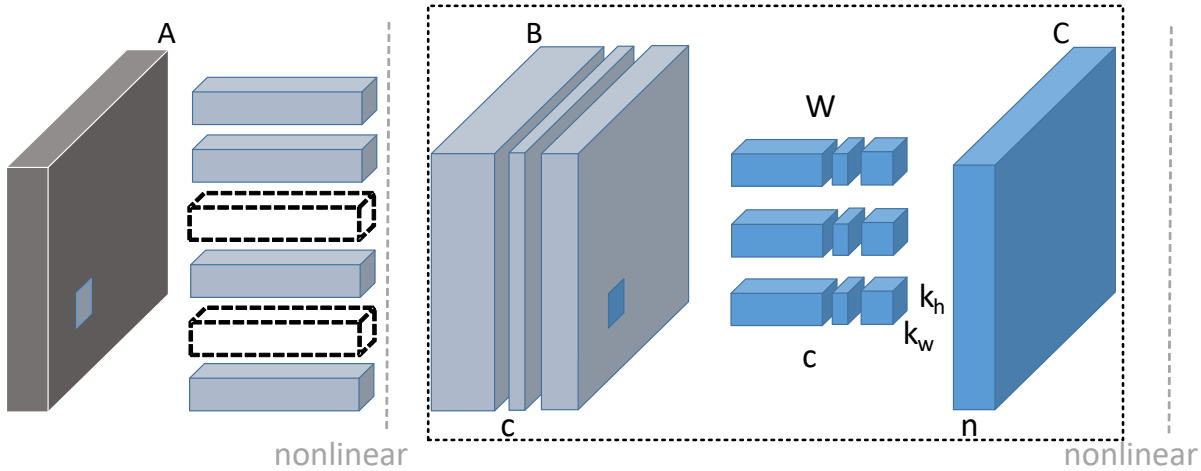
Figure 1: Illustration of the decomposition. (a) An original layer with complexity $O(dk^2c)$. (b) An approximated layer with complexity reduced to $O(d'k^2c) + O(dd')$.



Accelerating very deep convolutional networks for classification and detection.

Speeding up convolutional neural networks with low rank expansions.

Our approach



We aim to reduce the width of feature map B, while minimizing the reconstruction error on feature map C. Our optimization algorithm performs within the dotted box, which does not involve nonlinearity. This figure illustrates the situation that two channels are pruned for feature map B. Thus corresponding channels of filters W can be removed. Furthermore, even though not directly optimized by our algorithm, the corresponding filters in the previous layer can also be removed (marked by dotted filters). c, n: number of channels for feature maps B and C, $k_h \times k_w$: kernel size.

Optimization

Formally, to prune a feature map with c channels, we consider applying $n \times c \times k_h \times k_w$ convolutional filters W on $N \times c \times k_h \times k_w$ input volumes X sampled from this feature map, which produces $N \times n$ output matrix Y. Here, N is the number of samples, n is the number of output channels, and k_h, k_w are the kernel size. For simple representation, bias term is not included in our formulation. To prune the input channels from c to desired c' ($0 \leq c' \leq c$), while minimizing reconstruction error, we formulate our problem as follow:

$$\begin{aligned} &\arg\min_{\boldsymbol{\beta}, \mathrm{W}} \frac{1}{2N} \left\| \mathrm{Y} - \sum_{i=1}^{c} \beta_i \mathrm{X_i} \mathrm{W_i}^\top \right\|_F^2 \\ &\text{subject to } \|\boldsymbol{\beta}\|_0 \leq c' \end{aligned} \tag{1}$$

$\|\cdot\|_F$ is Frobenius norm. $\mathrm{X_i}$ is $N \times k_h k_w$ matrix sliced from ith channel of input volumes X, $i = 1, ..., c$. $\mathrm{W_i}$ is $n \times k_h k_w$ filter weights sliced from ith channel of W. $\boldsymbol{\beta}$ is coefficient vector of length c for channel selection, and β_i is ith entry of $\boldsymbol{\beta}$. Notice that, if $\beta_i = 0$, $\mathrm{X_i}$ will be no longer useful, which could be safely pruned from feature map. $\mathrm{W_i}$ could also be removed.

Optimization

Solving this ℓ_0 minimization problem in Eqn. 1 is NP-hard. we relax the ℓ_0 to ℓ_1 regularization:

$$\begin{aligned} &\arg\min_{\boldsymbol{\beta},\mathrm{W}} \frac{1}{2N} \left\| \mathrm{Y} - \sum_{i=1}^{c} \beta_i \mathrm{X_i} \mathrm{W_i}^\top \right\|_F^2 + \lambda \left\| \boldsymbol{\beta} \right\|_1 \\ &\text{subject to } \left\| \boldsymbol{\beta} \right\|_0 \leq c', \forall i \left\| \mathrm{W_i} \right\|_F = 1 \end{aligned} \tag{2}$$

λ is a penalty coefficient. By increasing λ, there will be more zero terms in $\boldsymbol{\beta}$ and one can get higher speed-up ratio. We also add a constrain $\forall i \left\| \mathrm{W_i} \right\|_F = 1$ to this formulation, which avoids trivial solution. Now we solve this problem in two folds. First, we fix W, solve $\boldsymbol{\beta}$ for channel selection. Second, we fix $\boldsymbol{\beta}$, solve W to reconstruct error.

Optimization

(i) The subproblem of β: In this case, W is fixed. We solve $\boldsymbol{\beta}$ for channel selection.

$$\hat{\boldsymbol{\beta}}^{LASSO}(\lambda) = \arg\min_{\boldsymbol{\beta}} \frac{1}{2N} \left\| \mathrm{Y} - \sum_{i=1}^{c} \beta_i \mathrm{Z_i} \right\|_F^2 + \lambda \|\boldsymbol{\beta}\|_1 \quad (3)$$

$$\text{subject to } \|\boldsymbol{\beta}\|_0 \leq c'$$

Here $\mathrm{Z_i} = \mathrm{X_i}\mathrm{W_i}^\top$ (size $N \times n$). We will ignore ith channels if $\beta_i = 0$.

(ii) The subproblem of W: In this case, $\boldsymbol{\beta}$ is fixed. We utilize the selected channels to minimize reconstruction error. We can find optimized solution by least squares:

$$\arg\min_{\mathrm{W}'} \left\| \mathrm{Y} - \mathrm{X}'(\mathrm{W}')^\top \right\|_F^2 \quad (4)$$

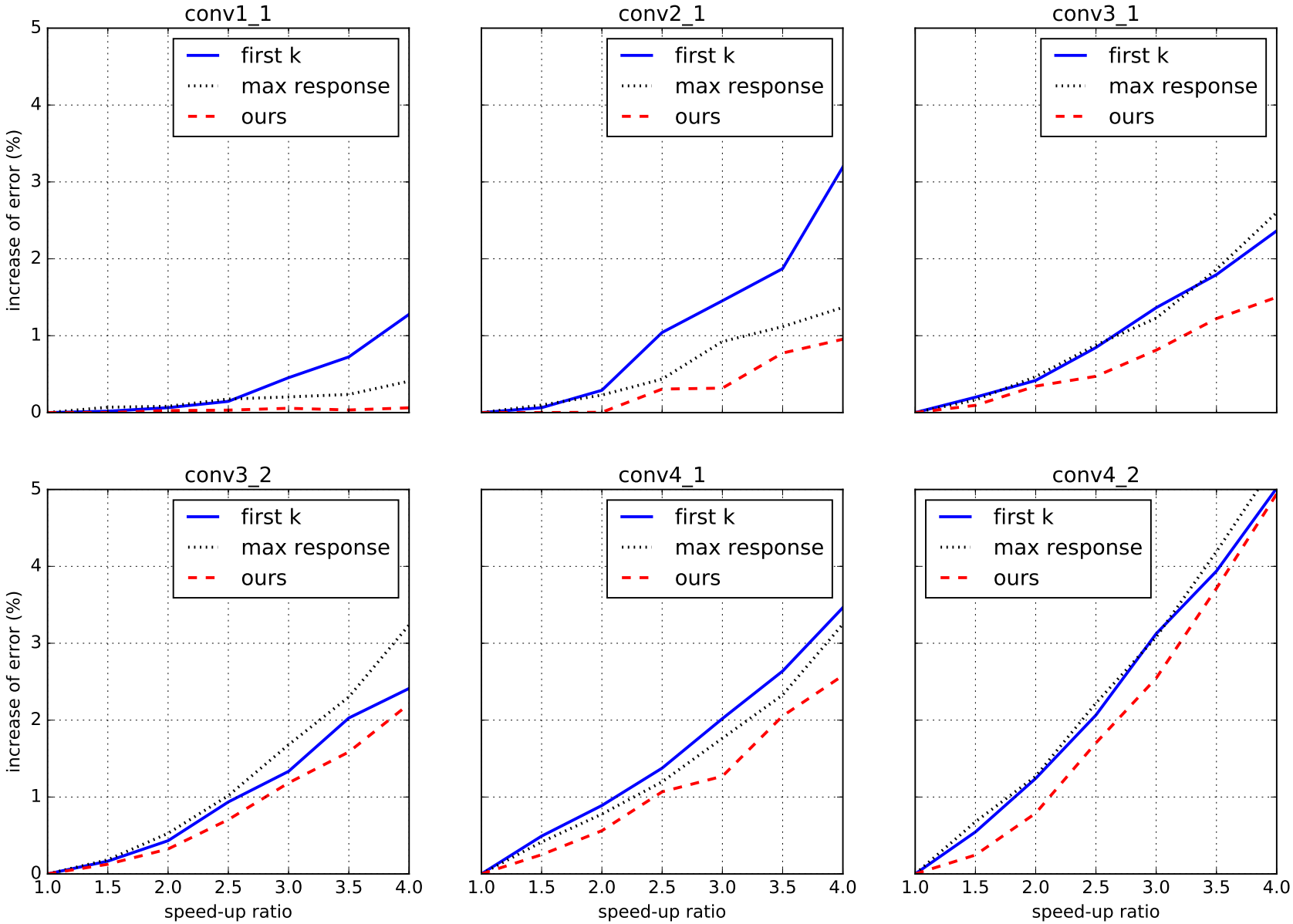
Here $\mathrm{X}' = [\beta_1\mathrm{X_1}\ \beta_2\mathrm{X_2}\ \ldots\ \beta_i\mathrm{X_i}\ \ldots\ \beta_c\mathrm{X_c}]$ (size $N \times ck_hk_w$). W' is $n \times ck_hk_w$ reshaped W, $\mathrm{W}' = [\mathrm{W_1}\ \mathrm{W_2}\ \ldots\ \mathrm{W_i}\ \ldots\ \mathrm{W_c}]$. After obtained result W', it is reshaped back to W. Then we assign $\beta_i \leftarrow \beta_i \|\mathrm{W_i}\|_F$, $\mathrm{W_i} \leftarrow \mathrm{W_i}/\|\mathrm{W_i}\|_F$. Constrain $\forall i \|\mathrm{W_i}\|_F = 1$ satisfies.

Analysis

TABLE 1
The VGG-16 architecture. The column "complexity" is portion of the theoretical time complexity each layer contributes. The column "PCA energy" shows feature map PCA Energy (top 50% eigenvalues).

	# channels	# filters	output size	complexity (%)	PCA energy (%)
conv1_1	64	64	224	0.6	99.8
conv1_2	64	64	224	12	99.0
pool1			112		
conv2_1	64	128	112	6	96.7
conv2_2	128	128	112	12	92.9
pool2			56		
conv3_1	128	256	56	6	94.8
conv3_2	256	256	56	12	92.3
conv3_3	256	256	56	12	89.3
pool3			28		
conv4_1	256	512	28	6	89.9
conv4_2	512	512	28	12	86.5
conv4_3	512	512	28	12	81.8
pool4			14		
conv5_1	512	512	14	3	83.4
conv5_2	512	512	14	3	83.1
conv5_3	512	512	14	3	80.8

Analysis


conv1_1
conv2_1
conv3_1
conv3_2
conv4_1
conv4_2
first k
max response
ours
increase of error (%)
speed-up ratio
0
1
2
3
4
5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Analysis

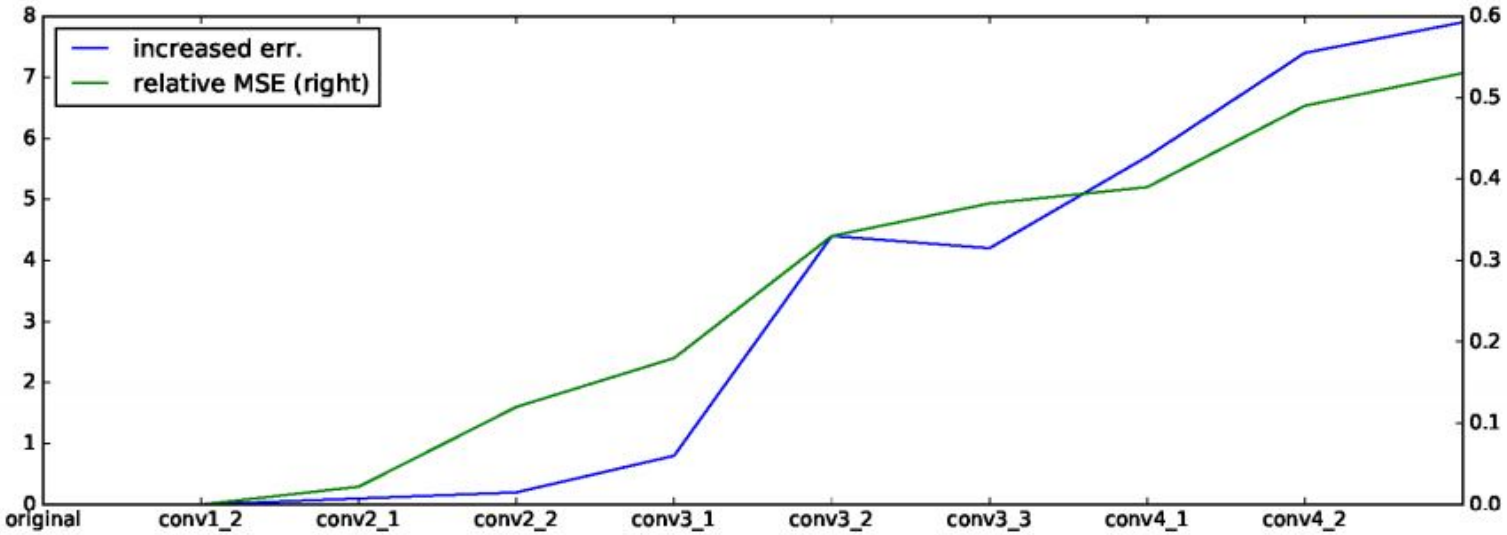


Fig. 6. Accumulated layerwise pruning error for accelerating VGG-16 under $4\times$. "relative MSE" is the relative mean square error. After fine-tuning, the Top-5 drops is 1.0%.

Results

Increase of top-5 error (1-view, baseline 89.9%)			
Solution	2×	4×	5×
Jaderberg (Zhang 2015's impl.)	-	9.7	29.7
Asym.	0.28	3.84	-
Filter pruning (fine-tuned, our impl.)	0.8	8.6	14.6
Ours (without fine-tune)	2.7	7.9	22.0
Ours (fine-tuned)	0	1.0	1.7

Table 1: Accelerating the VGG-16 model using a speedup ratio of 2×, 4×, or 5× (*smaller is better*).

Results

Increase of top-5 error (1-view, 89.9%)		
Solution	4×	5×
Asym. 3D	0.9	2.0
Asym. 3D (fine-tuned)	0.3	1.0
Our 3C	0.7	1.3
Our 3C (fine-tuned)	**0.0**	**0.3**

Table 2: Performance of combined methods on the VGG-16 model using a speed-up ratio of 4× or 5×. Our 3C solution outperforms previous approaches (*smaller is better*).

Absolute performance

Model	Solution	Increased err.	GPU time/ms
VGG-16	-	0	8.144
VGG-16 (4×)	Jaderberg *et al.* [22] ([53]'s impl.)	9.7	8.051 (1.01×)
	Asym. [53]	3.8	5.244 (1.55×)
	Asym. 3D [53]	0.9	8.503 (0.96×)
	Asym. 3D (fine-tuned) [53]	**0.3**	8.503 (0.96×)
	Ours (fine-tuned)	1.0	**3.264** (2.50×)

Table 3. GPU acceleration comparison. We measure forward-pass time per image. Our approach generalizes well on GPU (smaller is better).

Acc Detection

Speedup	mAP	Δ mAP	mmAP	Δ mmAP
Baseline	68.7	-	36.7	-
2×	68.3	0.4	36.7	0.0
4×	66.9	1.8	35.1	1.6

2×, 4× acceleration for Faster R-CNN detection. mmAP is AP at IoU=.50:.05:.95 (primary challenge metric of COCO [32]).