香港中文大學
The Chinese University of Hong Kong

CENG5030
Part 2-5: CNN Inaccurate Speedup-3 —- Pruning

Bei Yu

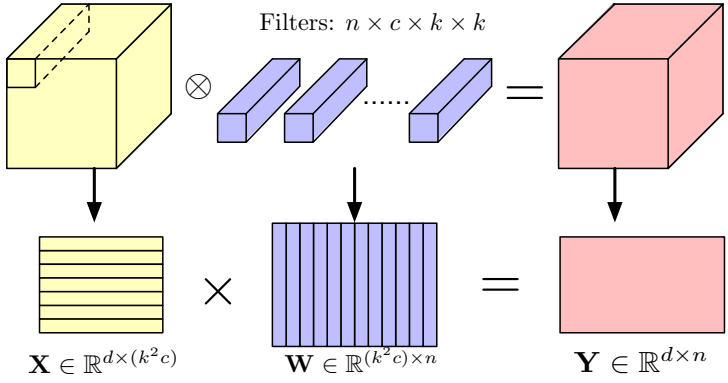
(Latest update: March 26, 2019)

Spring 2019

These slides contain/adapt materials developed by

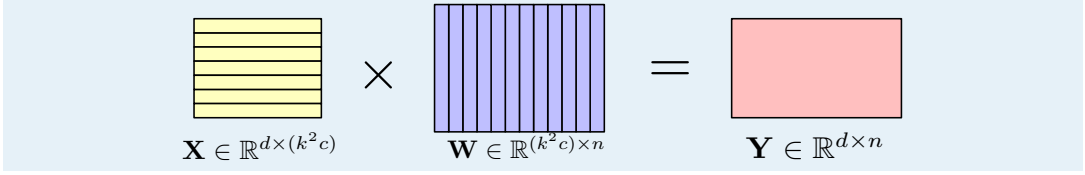
- Yuzhe Ma et al. (2018). “A Unified Approximation Framework for Non-Linear Deep Neural Networks”. In: *arXiv preprint arXiv:1807.10119*
- Yihui He, Xiangyu Zhang, and Jian Sun (2017). “Channel Pruning for Accelerating Very Deep Neural Networks”. In: *Proc. ICCV*

`Im2col` (Image2Column) Convolution



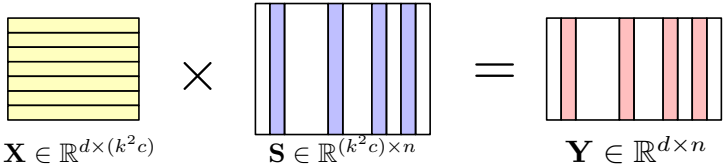
- Transform convolution to matrix multiplication
- Unified calculation for both convolution and fully-connected layers

Matrix Approximation or Matrix Regression?



- Matrix approximation: $\boldsymbol{W} \approx \boldsymbol{W'}$
- Matrix regression: $\boldsymbol{Y} = \boldsymbol{W} \cdot \boldsymbol{X} \approx \boldsymbol{W'} \cdot \boldsymbol{X}$

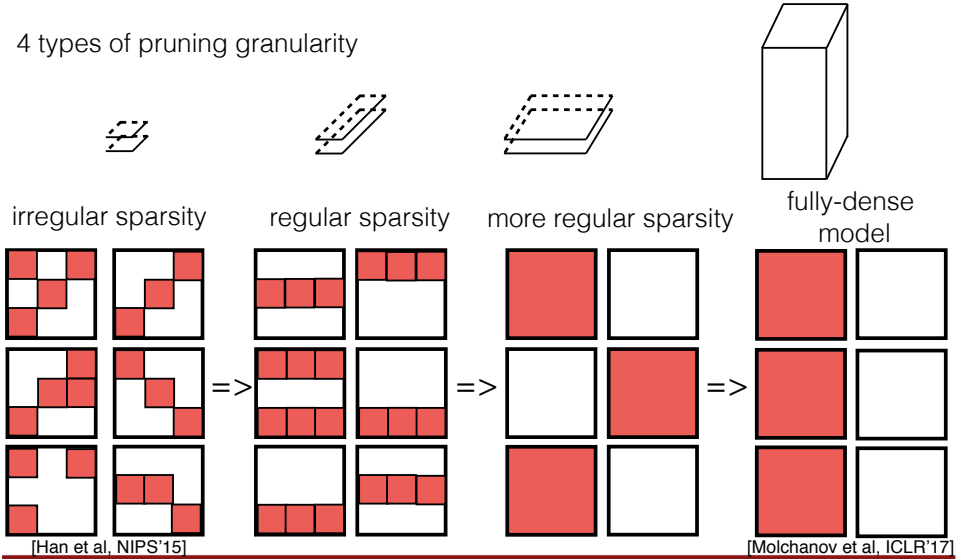
Compression Approach 1: Sparsity



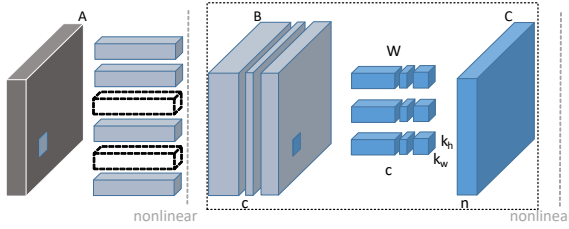
Sparse DNN

- *Sparsification*: weight pruning;
- *Compression*: compressed sparse format for storage;
- *Potential acceleration*: sparse matrix multiplication algorithm.

Exploring the Granularity of Sparsity that is Hardware-friendly



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 $\boldsymbol{\beta}$: In this case, W is fixed. We solve $\boldsymbol{\beta}$ for channel selection.

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

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

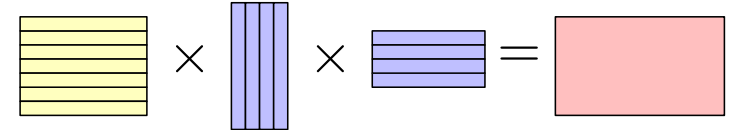
Here $\mathrm{Z_i} = \mathrm{X_i W_i}^\top$ (size $N \times n$). We will ignore ith channels if $\beta_i = 0$.

(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:

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

Here $\mathrm{X'} = [\beta_1 \mathrm{X_1}\ \beta_2 \mathrm{X_2}\ ...\ \beta_i \mathrm{X_i}\ ...\ \beta_c \mathrm{X_c}]$ (size $N \times ck_hk_w$). $\mathrm{W'}$ is $n \times ck_hk_w$ reshaped W, $\mathrm{W'} = [\mathrm{W_1}\ \mathrm{W_2}\ ...\ \mathrm{W_i}\ ...\ \mathrm{W_c}]$. After obtained result $\mathrm{W'}$, 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.

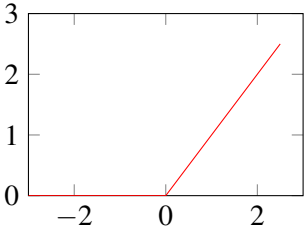
Compression Approach 2: Low-Rank



Low-rank DNN

- *Low-rank approximation*: matrix decomposition or tensor decomposition.
- *Compression and acceleration*: less storage required and less FLOP in computation.

Non-linearity Approximation



ReLU

- Activation unit: `ReLU`
- Error more sensitive to positive response;
- Enlarge the solution space.

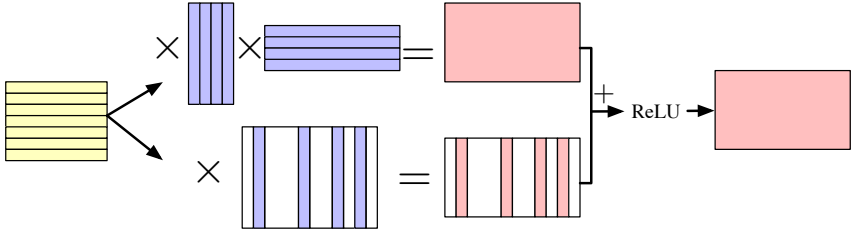
$$\min_{\boldsymbol{W}} \sum_{i=1}^{N} \|\boldsymbol{W}\boldsymbol{X}_i - \boldsymbol{Y}_i\|_F \rightarrow \min_{\boldsymbol{W}} \sum_{i=1}^{N} \|r(\boldsymbol{W}\boldsymbol{X}_i) - \boldsymbol{Y}_i\|_F$$

- $\boldsymbol{X}$: input feature map
- $\boldsymbol{Y}$: output feature map

Reading List

- Xiangyu Zhang et al. (2015). “Efficient and accurate approximations of nonlinear convolutional networks”. In: *Proc. CVPR*, pp. 1984–1992
- Hao Zhou, Jose M Alvarez, and Fatih Porikli (2016). “Less is more: Towards compact cnns”. In: *Proc. ECCV*, pp. 662–677
- Yihui He, Xiangyu Zhang, and Jian Sun (2017). “Channel Pruning for Accelerating Very Deep Neural Networks”. In: *Proc. ICCV*
- Xiyu Yu et al. (2017). “On compressing deep models by low rank and sparse decomposition”. In: *Proc. CVPR*, pp. 7370–7379

Proposed Unified Structure



- Simultaneous low-rank approximation and network sparsification;
- Non-linearity is taken into account.
- Acceleration is achieved with structured sparsity.

Formulation

Given a pre-trained network, the goal is to minimize the reconstruction error of the response in each layer after activation, using sparse component and low-rank component.

$$\begin{aligned} \min_{\boldsymbol{A},\boldsymbol{B}} \ & \sum_{i=1}^{N} \|\boldsymbol{Y}_i - r((\boldsymbol{A}+\boldsymbol{B})\boldsymbol{X}_i)\|_F, \\ \text{s.t.} \ & \|\boldsymbol{A}\|_0 \leq S, \\ & \text{rank}(\boldsymbol{B}) \leq L. \end{aligned}$$

- $\boldsymbol{X}$: input feature map
- $\boldsymbol{Y}$: output feature map

Not easy to solve: l_0 minimization and rank minimization are NP-hard.

Relaxation

$$\min_{\boldsymbol{A},\boldsymbol{B}} \sum_{i=1}^{N} \|\boldsymbol{Y}_i - r((\boldsymbol{A}+\boldsymbol{B})\boldsymbol{X}_i)\|_F^2 + \lambda_1 \|\boldsymbol{A}\|_{2,1} + \lambda_2 \|\boldsymbol{B}\|_*$$

- The l_0 constraint is relaxed by $l_{2,1}$ norm such that the zero elements in $\boldsymbol{A}$ appear column-wise;
- The rank constraint on $\boldsymbol{B}$ is relaxed by nuclear norm of $\boldsymbol{B}$, which is the sum of the singular values;
- Apply alternating direction method of multipliers (ADMM) to solve it;

Alternating Direction Method of Multipliers (ADMM)

Reformulating the problem with an auxiliary variable $\boldsymbol{M}$,

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{M}} \sum_{i=1}^{N} \|\boldsymbol{Y}_i - r(\boldsymbol{M}\boldsymbol{X}_i)\|_F^2 + \lambda_1 \|\boldsymbol{A}\|_{2,1} + \lambda_2 \|\boldsymbol{B}\|_*,$$
$$\text{s.t. } \boldsymbol{A} + \boldsymbol{B} = \boldsymbol{M}.$$

Then the augmented Lagrangian function is

$$\begin{aligned} & L_t(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{M}, \boldsymbol{\Lambda}) \\ = & \sum_{i=1}^{N} \|\boldsymbol{Y}_i - r(\boldsymbol{M}\boldsymbol{X}_i)\|_F^2 + \lambda_1 \|\boldsymbol{A}\|_{2,1} + \lambda_2 \|\boldsymbol{B}\|_* + \langle \boldsymbol{\Lambda}, \boldsymbol{A} + \boldsymbol{B} - \boldsymbol{M} \rangle + \frac{t}{2} \|\boldsymbol{A} + \boldsymbol{B} - \boldsymbol{M}\|_F^2, \end{aligned}$$

Alternating Direction Method of Multipliers (ADMM)

Iteratively solve with following rules. All of them can be solved efficiently.

$$
\begin{cases}
\boldsymbol{A}_{k+1} = \underset{\boldsymbol{A}}{\operatorname{argmin}}\ \lambda_1 \|\boldsymbol{A}\|_{2,1} + \dfrac{t}{2}\left\|\boldsymbol{A} + \boldsymbol{B}_k - \boldsymbol{M}_k + \dfrac{\boldsymbol{\Lambda}_k}{t}\right\|_F^2, \\
\boldsymbol{B}_{k+1} = \underset{\boldsymbol{B}}{\operatorname{argmin}}\ \lambda_2 \|\boldsymbol{B}\|_{*} + \dfrac{t}{2}\left\|\boldsymbol{B} + \boldsymbol{A}_{k+1} - \boldsymbol{M}_k + \dfrac{\boldsymbol{\Lambda}_k}{t}\right\|_F^2, \\
\boldsymbol{M}_{k+1} = \underset{\boldsymbol{M}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\boldsymbol{Y}_i - r(\boldsymbol{M}\boldsymbol{X}_i)\|_F^2 + \langle \boldsymbol{\Lambda}_k, \boldsymbol{A}_{k+1} + \boldsymbol{B}_{k+1} - \boldsymbol{M}\rangle + \dfrac{t}{2}\|\boldsymbol{A}_{k+1} + \boldsymbol{B}_{k+1} - \boldsymbol{M}\|_F^2, \\
\boldsymbol{\Lambda}_{k+1} = \boldsymbol{\Lambda}_k + t(\boldsymbol{A}_{k+1} + \boldsymbol{B}_{k+1} - \boldsymbol{M}_{k+1}).
\end{cases}
$$

Solving $l_{2,1}$-norm

$$\min_{\boldsymbol{A}} \lambda_1 \|\boldsymbol{A}\|_{2,1} + \frac{t}{2} \left\| \boldsymbol{A} + \boldsymbol{B}_k - \boldsymbol{M}_k + \frac{\boldsymbol{\Lambda}_k}{t} \right\|_F^2$$

Closed Form Update Rule[1]

$$\boldsymbol{A}_{k+1} = \text{prox}_{\frac{\lambda_1}{t}\|\cdot\|_{2,1}}(\boldsymbol{M}_k - \boldsymbol{B}_k - \frac{\boldsymbol{\Lambda}_k}{t}),$$

$$\boldsymbol{C} = \boldsymbol{M}_k - \boldsymbol{B}_k - \frac{\boldsymbol{\Lambda}_k}{t},$$

$$[\boldsymbol{A}_{k+1}]_{:,i} = \begin{cases} \frac{\|[\boldsymbol{C}]_{:,i}\|_2 - \frac{\lambda_1}{t}}{\|[\boldsymbol{C}]_{:,i}\|_2}[\boldsymbol{C}]_{:,i}, & \text{if } \|[\boldsymbol{C}]_{:,i}\|_2 > \frac{\lambda_1}{t}; \\ 0, & \text{otherwise.} \end{cases}$$

[1] G. Liu et al., "*Robust recovery of subspace structures by low-rank representation*", TPAMI, 2013.

Solving nuclear norm

$$\min_{\boldsymbol{B}} \lambda_2 \|\boldsymbol{B}\|_* + \frac{t}{2} \left\| \boldsymbol{B} + \boldsymbol{A}_{k+1} - \boldsymbol{M}_k + \frac{\boldsymbol{\Lambda}_k}{t} \right\|_F^2$$

Closed Form Update Rule[2]

$$\boldsymbol{B}_{k+1} = \text{prox}_{\frac{\lambda_2}{t}\|\cdot\|_*}(\boldsymbol{M}_k - \boldsymbol{A}_{k+1} - \frac{\boldsymbol{\Lambda}_k}{t}),$$

$$\boldsymbol{D} = \boldsymbol{M}_k - \boldsymbol{A}_{k+1} - \frac{\boldsymbol{\Lambda}_k}{t},$$

$$\boldsymbol{B}_{k+1} = \boldsymbol{U}\mathcal{D}_{\frac{\lambda_2}{t}}(\boldsymbol{\Sigma})\boldsymbol{V}, \quad \text{where } \mathcal{D}_{\frac{\lambda_2}{t}}(\boldsymbol{\Sigma}) = \text{diag}(\{(\sigma_i - \frac{\lambda_2}{t})_+\}).$$

[2] J-F. Cai et al., "*A singular value thresholding algorithm for matrix completion*", SIOPT, 2010.

Comparison on *CIFAR-10* dataset

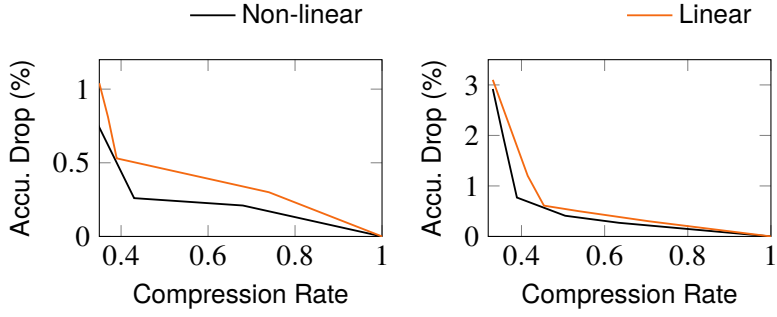
Model	Method	Accuracy ↓	CR	Speed-up
VGG-16	Original	0.00%	1.00	1.00
	ICLR'17[3]	**0.06%**	2.70	1.80
	Ours	0.40%	**4.44**	**2.20**
NIN	Original	0.00%	1.00	1.00
	ICLR'16[4]	1.43%	1.54	1.50
	IJCAI'18[5]	1.43%	1.45	-
	Ours	**0.41%**	**2.77**	**1.70**

[3] Hao Li et al. (2017). "Pruning filters for efficient convnets". In: *Proc. ICLR*.

[4] Cheng Tai et al. (2016). "Convolutional neural networks with low-rank regularization". In: *Proc. ICLR*.

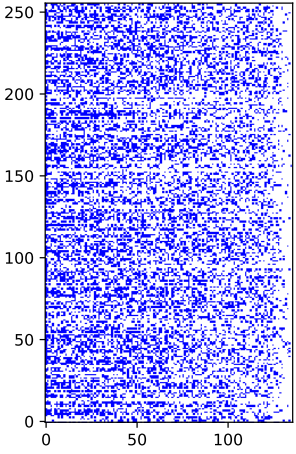
[5] Shiva Prasad Kasiviswanathan, Nina Narodytska, and Hongxia Jin (2018). "Network Approximation using Tensor Sketching". In: *Proc. IJCAI*, pp. 2319–2325.

Preliminary Results

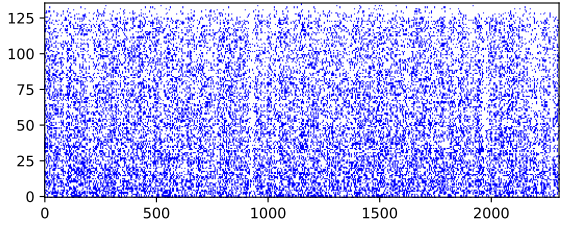


Comparison of reconstructing linear response and non-linear response: (a) layer `conv2-1`; (b) layer `conv3-1`.

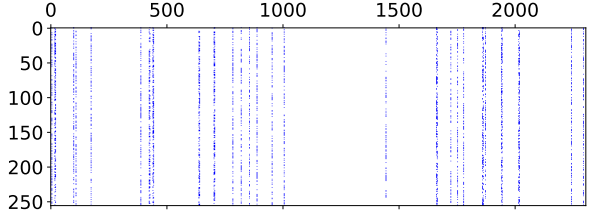
Approximation Example



(a)



(b)



(c)

Approximated filters of `conv3-1`. Blue dots have non-zero values. Low-rank filter $\boldsymbol{B}$ with rank 136 is decomposed into $\boldsymbol{UV}$, both of which have rank 136. (a) Matrix $\boldsymbol{U}$; (b) Matrix $\boldsymbol{V}$. (c) Column-wise sparse filter $\boldsymbol{A}$.

Comparison on *ImageNet* dataset

Model	Method	Top-5 Accu.↓	CR	Speed-up
AlexNet	Original	0.00%	1.00	1.00
	ICLR'16[6]	**0.37%**	5.00	**1.82**
	ICLR'16[7]	1.70%	5.46	1.81
	CVPR'18[8]	1.43%	1.50	-
	Ours	1.27%	**5.56**	1.10
GoogleNet	Original	0.00%	1.00	1.00
	ICLR'16[12]	0.42%	2.84	1.20
	ICLR'16[13]	0.24%	1.28	1.23
	CVPR'18[14]	0.21%	1.50	-
	Ours	**0.00%**	**2.87**	**1.35**

[6] Cheng Tai et al. (2016). "Convolutional neural networks with low-rank regularization". In: *Proc. ICLR*.

[7] Yong-Deok Kim et al. (2016). "Compression of deep convolutional neural networks for fast and low power mobile applications". In: *Proc. ICLR*.

[8] Ruichi Yu et al. (2018). "NISP: Pruning networks using neuron importance score propagation". In: *Proc. CVPR*.