CMSC 5743
Efficient Computing of Deep Neural Networks

Model 01: Pruning

Bei Yu
CSE Department, CUHK
byu@cse.cuhk.edu.hk

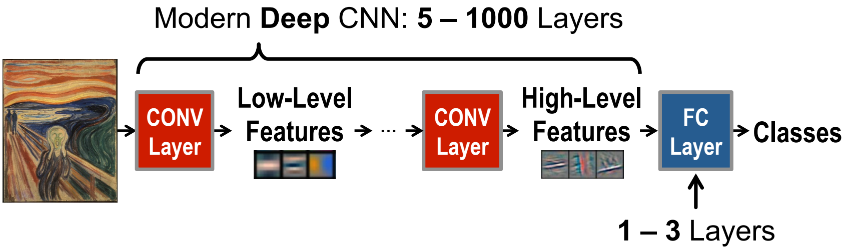
(Latest update: October 7, 2024)

2024 Fall

These slides contain/adapt materials developed by

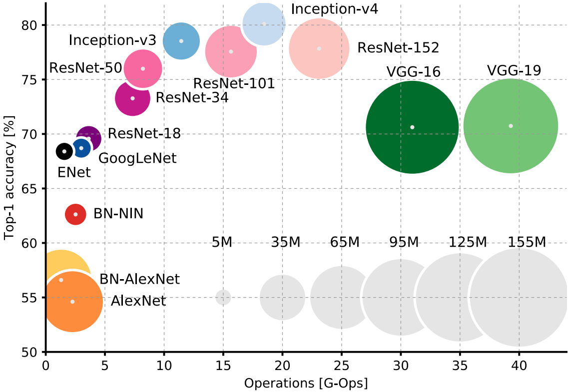
- Wei Wen et al. (2016). "Learning structured sparsity in deep neural networks". In: *Proc. NIPS*, pp. 2074–2082
- Yihui He, Xiangyu Zhang, and Jian Sun (2017). "Channel Pruning for Accelerating Very Deep Neural Networks". In: *Proc. ICCV*
- Ruichi Yu et al. (2018). "NISP: Pruning networks using neuron importance score propagation". In: *Proc. CVPR*, pp. 9194–9203
- Shijin Zhang et al. (2016). "Cambricon-x: An accelerator for sparse neural networks". In: *Proc. MICRO*. IEEE, pp. 1–12
- Jorge Albericio et al. (2016). "Cnvlutin: Ineffectual-neuron-free deep neural network computing". In: *ACM SIGARCH Computer Architecture News* 44.3, pp. 1–13

Deeper and Larger Networks



- Researchers design deeper and larger networks to ensure model performance.
- ☺ VGG-16, 16 parameter layers
- ☺ VGG-19, 19 parameter layers
- ☺ GoogLeNet, 22 parameter layers
- ☺ ResNet : -18, -34, -50, -101, -152 layers

Memory and Computations



- The size of the blob is proportional to the number of network parameters.
- More than millions of parameters and billions of operations.
- Challenges in **memory** and **energy**, finally affect the performance.

Overview

2. Sparse Regression

3. Pruning

Sparse Regression

Input

- $\boldsymbol{y} = (y_1, \ldots, y_N)^\top$: N samples to measure performance
- $\boldsymbol{X} = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)})^\top$: N parameters, where $\boldsymbol{x}^{(i)} = (x_1^{(i)}, \ldots, x_p^{(i)})^\top$ is parameter vector for sample y_i

Output

- $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)^\top$: linear regression model coefficients, s.t. $\boldsymbol{y} \approx \boldsymbol{X}\boldsymbol{\beta}$

$$
\begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_N \end{bmatrix} \approx \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \ldots & x_p^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \ldots & x_p^{(2)} \\ \ldots & \ldots & \ldots & \ldots \\ x_1^{(N)} & x_2^{(N)} & \ldots & x_p^{(N)} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \ldots \\ \beta_p \end{bmatrix}
$$

Input

- $\boldsymbol{y} = (y_1, \dots, y_N)^\top$: N samples to measure performance
- $\boldsymbol{X} = (\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)})^\top$: N parameters, where $\boldsymbol{x}^{(i)} = (x_1^{(i)}, \dots, x_p^{(i)})^\top$ is parameter vector for sample y_i

Output

- $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots, \beta_p)^\top$: linear regression model coefficients, s.t. $\boldsymbol{y} \approx \boldsymbol{X}\boldsymbol{\beta}$

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix} \approx \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_p^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_p^{(2)} \\ \dots & \dots & \dots & \dots \\ x_1^{(N)} & x_2^{(N)} & \dots & x_p^{(N)} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_p \end{bmatrix}$$

Objective

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2$$

Challenges in Linear Regression

$$\boldsymbol{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_p^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_p^{(2)} \\ \dots & \dots & \dots & \dots \\ x_1^{(N)} & x_2^{(N)} & \dots & x_p^{(N)} \end{bmatrix}$$

N: sample #
p: parameter #

- Time consuming to run simulation or measure $\rightarrow$ sample# N is limited
- If $N <$ parameter# p, $\rightarrow$ no unique solutions
- Overfitting problem
- Should reduce parameter#

Local Analysis

$$S_i = \frac{f(x_1, \cdots, x_i + \Delta x_i, \cdots, x_K) - f(x_1, \cdots, x_K)}{\Delta x_i}$$

- ☺ Computationally efficient
- ☹ Only take into account local variation around nominal value

Least Squares

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \quad \rightarrow \quad \boldsymbol{\beta} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

- ☺ Global view
- ☹ Too complicated model after analysis
- ☹ Need large simulation size ($N > p$)
- ☹ Otherrwise $\boldsymbol{X}^\top \boldsymbol{X}$ may be singular (difficult to invert)

Please prove:

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\beta\|_2^2 \quad \rightarrow \quad \boldsymbol{\beta} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

Vector Norms

Definition

$$\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{1/p}$$

Definition

$$\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{1/p}$$

- L-0 norm: $\|\boldsymbol{x}\|_0 =$ number of non-zero values

Definition

$$\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{1/p}$$

- L-0 norm: $\|\boldsymbol{x}\|_0 =$ number of non-zero values
- L-1 norm: $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$

Vector Norms

Definition

$$\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{1/p}$$

- L-0 norm: $\|\boldsymbol{x}\|_0$ = number of non-zero values
- L-1 norm: $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$
- L-2 norm: $\|\boldsymbol{x}\|_2 = \sqrt{\sum_i |x_i|^2} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$

Vector Norms

Definition

$$\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{1/p}$$

- L-0 norm: $\|\boldsymbol{x}\|_0 =$ number of non-zero values
- L-1 norm: $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$
- L-2 norm: $\|\boldsymbol{x}\|_2 = \sqrt{\sum_i |x_i|^2} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$
- L-infinity norm: $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$

Which of the following is L0?

- A. x1 + x2 + x3
- B. max{x1, x2, x3}
- C. |x1| + |x2| + |x3|
- D. none of them

ℓ_0-Norm Regularization

$$
\begin{aligned}
\text{minimize} \quad & \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|, \\
\text{subject to} \quad & \|\boldsymbol{\beta}\|_0 \leq \lambda.
\end{aligned}
$$

- ☺ Global view
- ☹ $\mathcal{NP}$-hard
- Orthogonal matching pursuit (OMP): iterative heuristics
- ☹ Computational expensive

Why we dont use L0 as regularization term?

- A. complicated inference
- B. no direct gradient obtained
- C. none of above
- D. both of above

Ridge Regression

$$\arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} |\beta_j|^2$$

Ridge Regression

$$\arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} |\beta_j|^2$$

$$\rightarrow \quad \boldsymbol{\beta} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

Lasso

$$\arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

- "ℓ_1 penalty" (Lasso)
- $\boldsymbol{\beta}$ optimally solved by Coordinate Descent [Friedman+,AOAS'07]
- λ: nonnegative regularization parameter

Which of the following regularization term can introduce sparsity?

- A. L1
- B. L2
- C. both of them
- D. None of them

Which regularization is used to reduce the over fit problem?

- A. L1
- B. L2
- C. both of them
- D. None of them

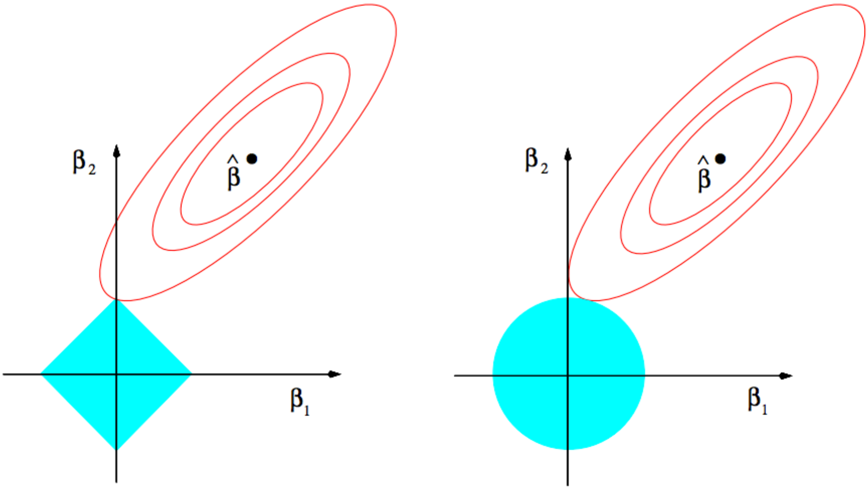
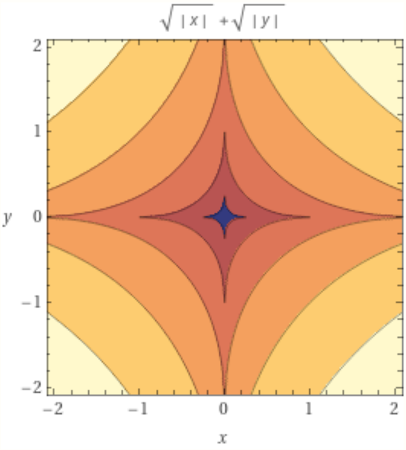


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

Can we introduce a more sparse regularization term? Why

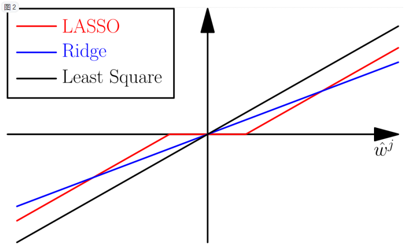
For example, $|x|^{1/2} + |y|^{1/2}$.



Provide closed-form solution to single variable Lasso problem:

$$\min_{x} L(x) = \frac{1}{2}(x - w)^2 + \lambda |x|$$

Visualization For Single Variable Lasso



Note: Lasso solution

- If $x > 0$, then $\omega - \lambda > 0(\omega > \lambda) \rightarrow x = \omega - \lambda$
- If $x < 0$, then $\omega + \lambda < 0(\omega < -\lambda) \rightarrow x = \omega + \lambda$
- $x = 0$, others

Coordinate Descent

- The idea behind coordinate descent is, simply, to optimize a target function with respect to a single parameter at a time, iteratively cycling through all parameters until convergence is reached
- Coordinate descent is particularly suitable for problems, like the lasso, that have a simple closed form solution in a single dimension but lack one in higher dimensions

- Let us consider minimizing Q with respect to β_j, while temporarily treating the other regression coefficients $\boldsymbol{\beta}_{-j}$ as fixed:

$$Q(\beta_j|\boldsymbol{\beta}_{-j}) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \sum_{k\neq j} x_{ij}\beta_k - x_{ij}\beta_j)^2 + \lambda|\beta_j| + \text{Constant}$$

- Let

$$\tilde{r}_{ij} = y_i - \sum_{k\neq j} x_{ik}\widetilde{\beta}_k$$

$$\tilde{z}_j = n^{-1}\sum_{i=1}^{n} x_{ij}\tilde{r}_{ij},$$

where $\{\tilde{r}_{ij}\}_{i=1}^n$ are the partial residuals with respect to the j^{th} predictor, and $\tilde{z}_j$ is the OLS estimator based on $\{\tilde{r}_{ij}, x_{ij}\}_{i=1}^n$

- We have already solved the problem of finding a one-dimensional lasso solution; letting $\widetilde{\beta}_j$ denote the minimizer of $Q(\beta_j|\widetilde{\boldsymbol{\beta}}_{-j})$,

$$\widetilde{\beta}_j = S(\tilde{z}_j|\lambda)$$

- This suggests the following algorithm:

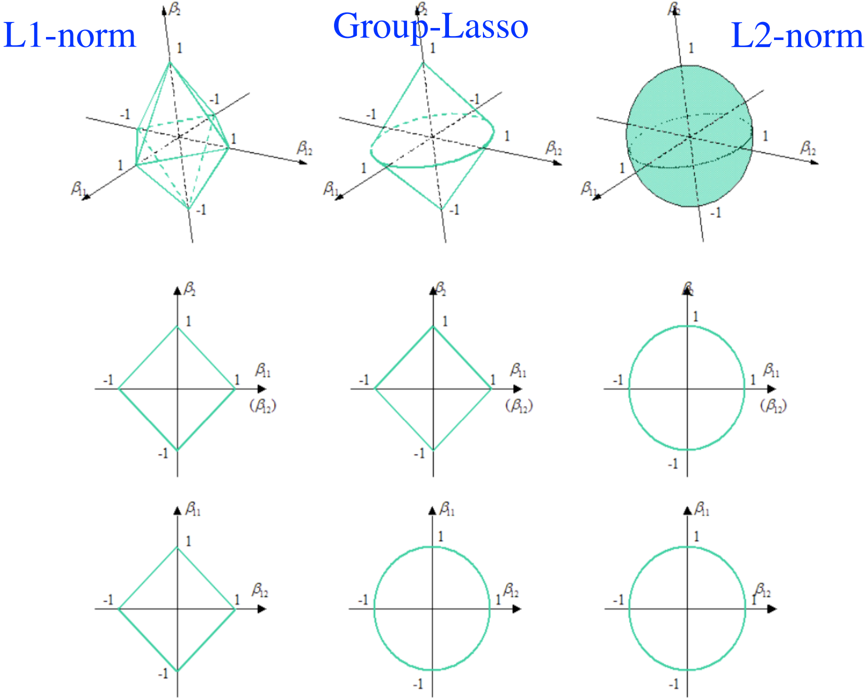
repeat
  for $j = 1, 2, \ldots, p$
    $\tilde{z}_j = n^{-1}\sum_{i=1}^{n} x_{ij} r_i + \widetilde{\beta}_j^{(s)}$
    $\widetilde{\beta}_j^{(s+1)} \leftarrow S(\tilde{z}_j|\lambda)$
    $r_i \leftarrow r_i - (\widetilde{\beta}_j^{(s+1)} - \widetilde{\beta}_j^{(s)}) x_{ij}$ for all i.
until convergence

Group Lasso

- We denote $\boldsymbol{X}$ as being composed of J groups $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_J$
- $\boldsymbol{X}\boldsymbol{\beta} = \sum_j \boldsymbol{X}_j \boldsymbol{\beta}_j$, where $\boldsymbol{\beta}_j$ represents the coefficients belonging to the jth group

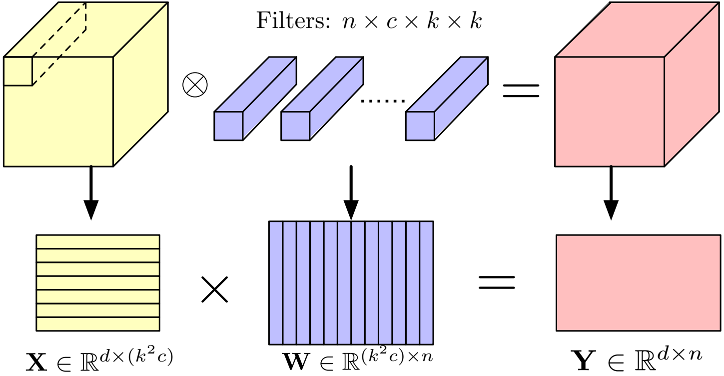
$$
\begin{aligned}
&\arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \sum_j \lambda_j \|\boldsymbol{\beta}_j\| \\
= &\arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \sum_j \boldsymbol{X}_j \boldsymbol{\beta}_j\|_2^2 + \sum_j \lambda_j \|\boldsymbol{\beta}_j\|
\end{aligned}
$$

Example:

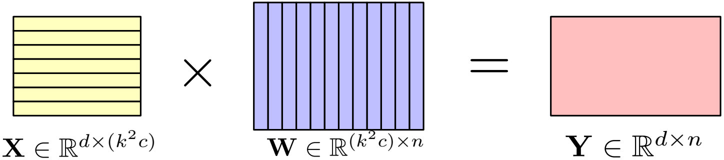

L1-norm
Group-Lasso
L2-norm
β_2
β_{11}
β_{12}
(β_{12})
1
-1

Pruning

Im2col (Image2Column) Convolution



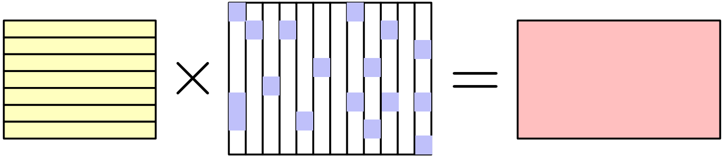
- Transform convolution to matrix multiplication
- Unified calculation for both convolution and fully-connected layers



Which is better?

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

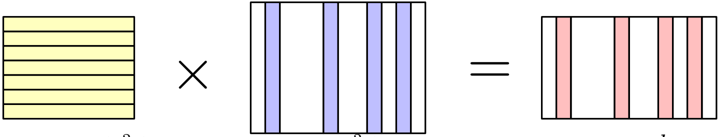
Compression Approach 1: Random Sparsity



$$\mathbf{X} \in \mathbb{R}^{d \times (k^2 c)} \qquad \times \qquad \mathbf{S} \in \mathbb{R}^{(k^2 c) \times n} \qquad = \qquad \mathbf{Y} \in \mathbb{R}^{d \times n}$$

Sparse DNN

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



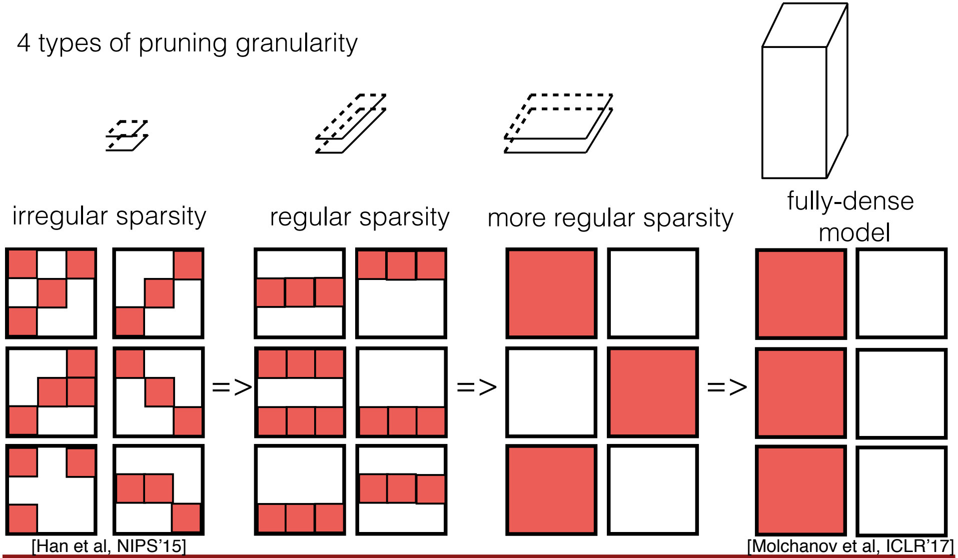
$\mathbf{X} \in \mathbb{R}^{d\times(k^2c)}$ $\times$ $\mathbf{S} \in \mathbb{R}^{(k^2c)\times n}$ $=$ $\mathbf{Y} \in \mathbb{R}^{d\times n}$

Structured Sparse DNN

- *Potential acceleration*: GEMM or directed convolution.

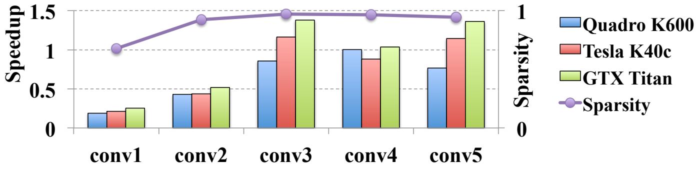
Exploring the Granularity of Sparsity that is Hardware-friendly

4 types of pruning granularity



Structured Sparsity Learning[1]

Random sparsity, theoretical Speedup $\neq$ practical Speedup



Forwarding speedups of AlexNet on GPU platforms and the sparsity. Baseline is GEMM of cuBLAS. The sparse matrixes are stored in the format of Compressed Sparse Row (CSR) and accelerated by cuSPARSE.

Random sparsity → Irregular memory access → Poor cache locality → No or trivial speedup

Hardcoding nonzero weights in source code in B. Liu, etc., CVPR 2015

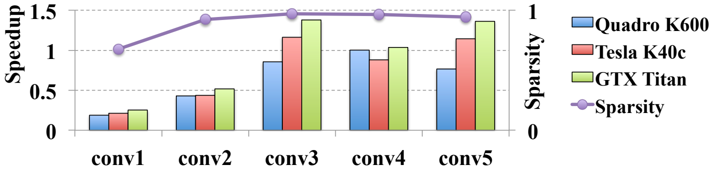
Software customization

Hardware customization

Customizing an EIE chip accelerator for compressed DNN in S. Han ISCA 2017

[1] Wei Wen et al. (2016). "Learning structured sparsity in deep neural networks". In: *Proc. NIPS*, pp. 2074–2082.

Structural Sparsity



Forwarding speedups of AlexNet on GPU platforms and the sparsity. Baseline is GEMM of cuBLAS. The sparse matrixes are stored in the format of Compressed Sparse Row (CSR) and accelerated by cuSPARSE.

Structured sparsity → Regular memory access → Good cache locality → Great speedup → Higher speedup with software or hardware customization

Dense matrix to block sparse matrix

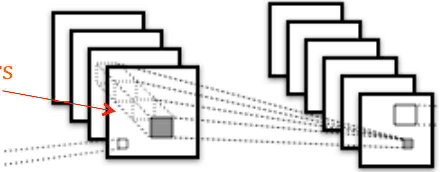
0.2	0.1	0.2	-0.6	0.1	0.4	-0.1	0.6
0.4	-0.3	0.4	0.1	0.2	-0.4	0.1	0.5
0.7	-0.1	-0.3	0.1	0.5	-0.1	0.5	0.1
-0.1	0.6	-0.5	0.3	-0.4	-0.2	0.3	0.6

$\Rightarrow$

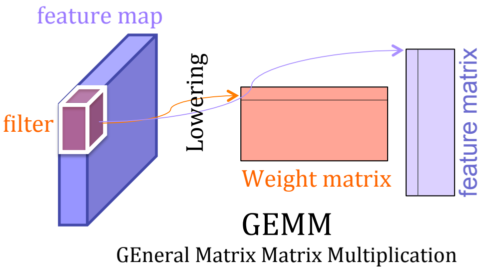
		0.2	-0.6			-0.1	0.6
		0.4	0.1			0.1	0.5
0.7	-0.1					0.5	0.1
-0.1	0.6					0.3	0.6

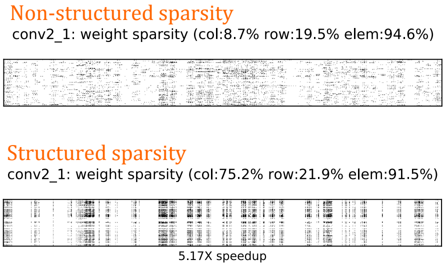
Removing 2D filters in convolution (2D-filter-wise sparsity)

3D filter =
stacked 2D filters



Removing rows/columns in GEMM (row/column-wise sparsity)





Group Lasso Regularization

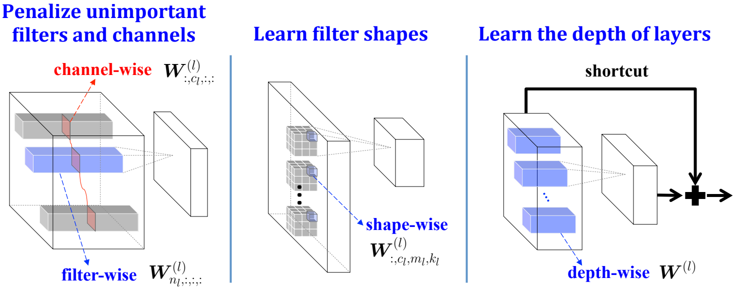
- $E_D(W)$ is the loss on data.
- $R(\cdot)$ is non-structured regularization applying on every weight, *e.g.*, ℓ_2-norm.
- $R_g(\cdot)$ is the structured sparsity regularization for G groups on each layer:

$$R_g(w) = \sum_{g=1}^{G} \|w^{(g)}\|_g.$$

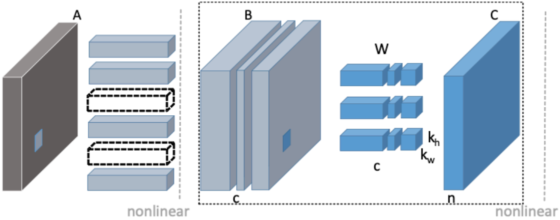
- Here $\|\cdot\|_g$ is group lasso, or $\|w^{(g)}\|_g = \sqrt{\sum_{i=1}^{|w^{(g)}|}(w_i^{(g)})^2}$, where $|w^{(g)}|$ is the number of weights in $w^{(g)}$.

Group Lasso Regularization

Learned structured sparsity is determined by the way of splitting groups.



$$E(W) = E_D(W) + \lambda \cdot R(W) + \lambda_g \sum_{l=1}^{L} R_g(W^{(l)})$$



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.

[2] Yihui He, Xiangyu Zhang, and Jian Sun (2017). "Channel Pruning for Accelerating Very Deep Neural Networks". In: *Proc. ICCV*.

Channel Pruning[2]

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, $i = 1, \ldots, c$. $\mathrm{W_i}$ 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.

[2] Yihui He, Xiangyu Zhang, and Jian Sun (2017). "Channel Pruning for Accelerating Very Deep Neural Networks". In: *Proc. ICCV*.

Channel Pruning[2]

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

$$\underset{\boldsymbol{\beta},\mathrm{W}}{\arg\min} \frac{1}{2N} \left\| \mathrm{Y} - \sum_{i=1}^{c} \beta_i \mathrm{X_i} \mathrm{W_i}^{\top} \right\|_F^2 + \lambda \|\boldsymbol{\beta}\|_1 \tag{2}$$

$$\text{subject to } \cancel{\|\boldsymbol{\beta}\|_0 \le c'}, \forall i \, \|\mathrm{W_i}\|_F = 1$$

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 $\forall i \, \|\mathrm{W_i}\|_F = 1$ 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.

[2]Yihui He, Xiangyu Zhang, and Jian Sun (2017). "Channel Pruning for Accelerating Very Deep Neural Networks". In: *Proc. ICCV*.

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

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

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

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:

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

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.

[2]Yihui He, Xiangyu Zhang, and Jian Sun (2017). “Channel Pruning for Accelerating Very Deep Neural Networks”. In: *Proc. ICCV*.