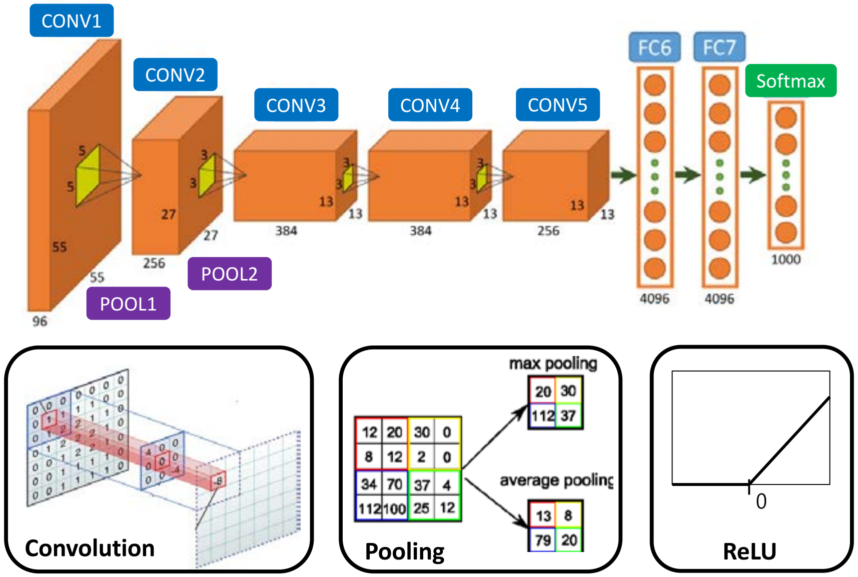
CSED703R: Deep Learning for Visual Recognition (2017F)

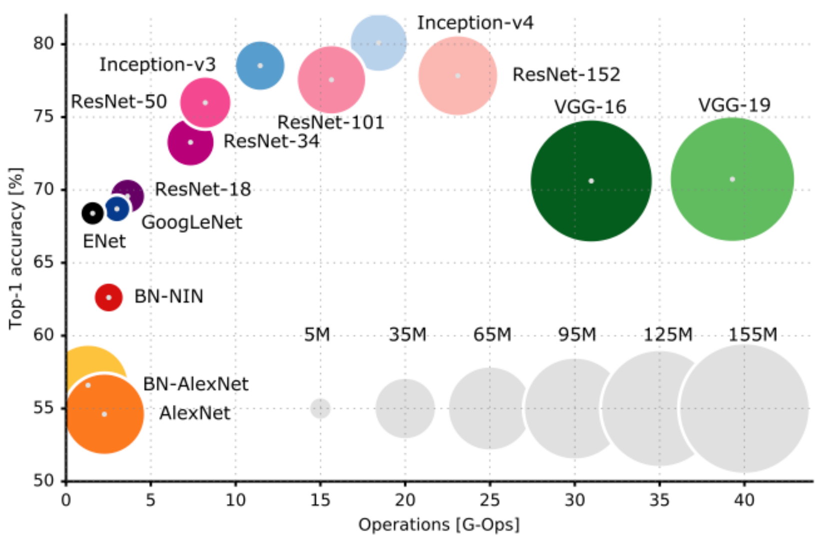
Lecture 9: CNN Optimization

Bohyung Han
Computer Vision Lab.
bhhan@postech.ac.kr

Convolutional Neural Networks



Complexity of CNNs



https://medium.com/towards-data-science/neural-network-architectures-156e5bad51ba

CNN Optimization

- Motivation: huge computational costs
  - Both time and space complexity are very large.
  - They are proportional to the number of parameters and layers, and the size of feature maps.
- Source of complexity
  - Convolutional layers
    - Slow due to many redundant operations (convolutions)
    - Easy to be parallelized
    - Filters are small in general but feature map sizes are large.
  - Fully connected layers
    - Fast since the operation can be implemented by matrix-vector multiplication
    - Large memory requirement: a large number of parameters

Operations in Convolutional Neural Networks

- Convolutional layers

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$\boldsymbol{X}$

$*$

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$\boldsymbol{F}$

$=$

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

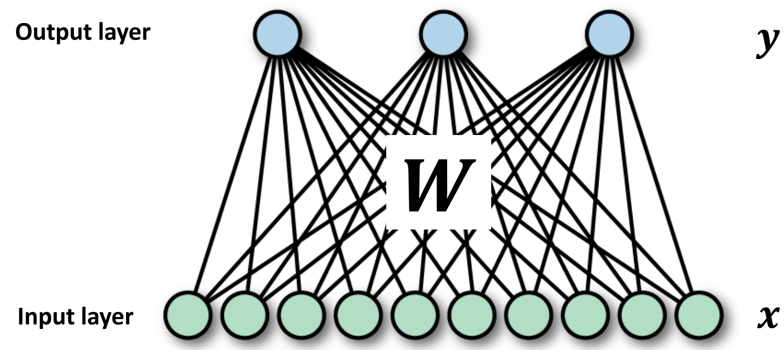
$\boldsymbol{Y}$

If $\boldsymbol{F} = \boldsymbol{VH}$, then $\boldsymbol{Y} = \boldsymbol{F} * \boldsymbol{X} = (\boldsymbol{VH}) * \boldsymbol{X} = \boldsymbol{V} * (\boldsymbol{H} * \boldsymbol{X})$.

Operations in Convolutional Neural Networks

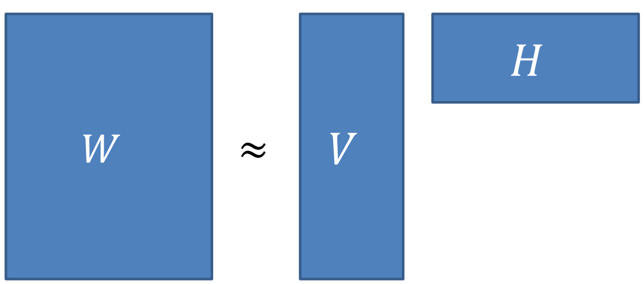
- Fully connected layers

$$\boldsymbol{y} = \boldsymbol{Wx}$$



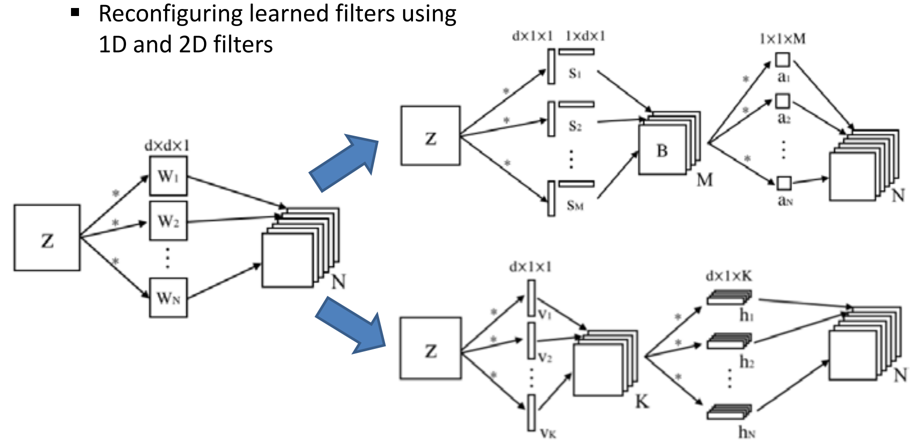
Low Rank Approximation

- Operations in CNNs
  - Convolutional layers: linear filtering with 3D tensors
  - Fully connected layers: simple matrix-vector multiplication
- CNN parameter approximation
  - Operations in both layers involve parameter matrices, which can be approximated by products of low-rank matrices.
  - Use of approximate matrices incur small differences in output layers.



Filter Bank Approximation

- Removing redundancy across filter banks
  - Reconstructing learned filters using a set of linearly separable filters
  - Reconfiguring learned filters using 1D and 2D filters

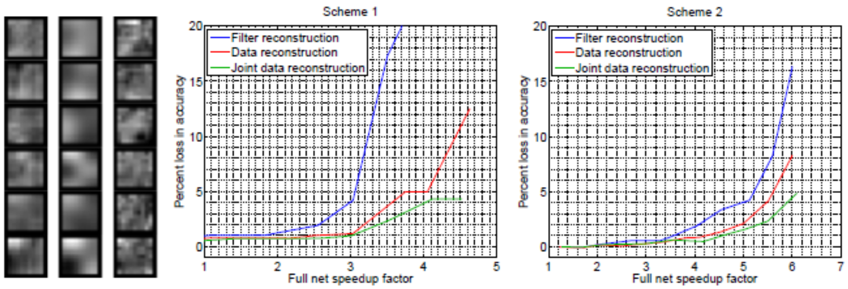


[Jaderberg14] M. Jaderberg, A. Vedaldi, A. Zisserman: **Speeding up Convolutional Neural Networks with Low Rank Expansions**. BMVC 2014

Filter Bank Approximation

- Objective
  - Filter reconstruction: optimizing filter itself
  - Data reconstruction: optimizing feature responses
  - Joint reconstruction: considering both filters and responses

2.5x speed-up with no loss in accuracy, 4.5x speed-up with <1% drop in accuracy



[Jaderberg14] M. Jaderberg, A. Vedaldi, A. Zisserman: **Speeding up Convolutional Neural Networks with Low Rank Expansions**. BMVC 2014

Non-Linear Filter Approximation

- Linear approximation
  - Output of a layer is approximated: $\boldsymbol{y}_i = \boldsymbol{W}\boldsymbol{x}_i$
  - Low-rank assumption of the output: $\boldsymbol{y}_i = \boldsymbol{M}\boldsymbol{W}\boldsymbol{x}_i$, where $\operatorname{rank}(\boldsymbol{M}) \le d'$
  - Output $\boldsymbol{y}$ is assumed to be on a low-dimensional manifold.

$$\min_{\boldsymbol{M}} \sum_i \|(\boldsymbol{y}_i - \bar{\boldsymbol{y}}) - \boldsymbol{M}(\boldsymbol{y}_i - \bar{\boldsymbol{y}})\|_2^2 \quad \text{such that} \quad \operatorname{rank}(\boldsymbol{M}) \le d'$$

- Non-linear approximation
  - Approximation of ReLU together

$$\min_{\boldsymbol{M},\boldsymbol{b}} \sum_i \|r(\boldsymbol{y}_i) - r(\boldsymbol{M}\boldsymbol{y}_i + \boldsymbol{b})\|_2^2 \quad \text{such that} \quad \operatorname{rank}(\boldsymbol{M}) \le d'$$

  - Relaxation

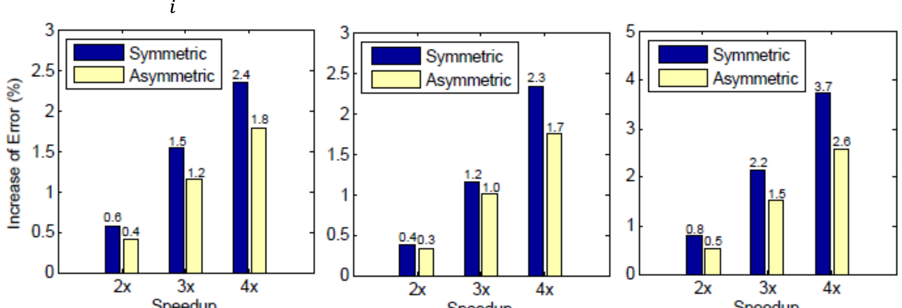
$$\min_{\boldsymbol{M},\boldsymbol{b},\{\boldsymbol{z}_i\}} \sum_i \|r(\boldsymbol{y}_i) - r(\boldsymbol{z}_i)\|_2^2 + \lambda\|\boldsymbol{z}_i - (\boldsymbol{M}\boldsymbol{y}_i + \boldsymbol{b})\|_2^2 \ \text{ such that } \operatorname{rank}(\boldsymbol{M}) \le d'$$

[Zhang15] X. Zhang, J. Zou, X. Ming, K. He, J. Sun: **Efficient and Accurate Approximations of Nonlinear Convolutional Networks**. CVPR 2015

Non-Linear Filter Approximation

- Multiple layer approximation
  - Layer-by-layer approximation: prone to accumulate error
  - Asymmetric reconstruction with noisy input $\hat{\boldsymbol{x}}_i$

$$\min_{\boldsymbol{M},\boldsymbol{b}} \sum_i \|r(\boldsymbol{W}\boldsymbol{x}_i) - r(\boldsymbol{M}\boldsymbol{W}\hat{\boldsymbol{x}}_i + \boldsymbol{b})\|_2^2 \quad \text{such that} \quad \operatorname{rank}(\boldsymbol{M}) \le d'$$

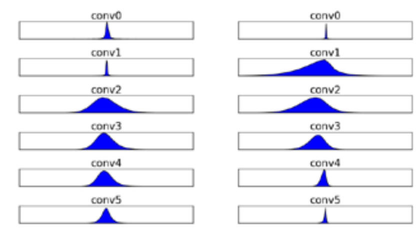


(a) 2–layer (Conv6 and 7) (b) 3–layer (Conv2, 3 and 4) (c) 3–layer (Conv5, 6 and 7)

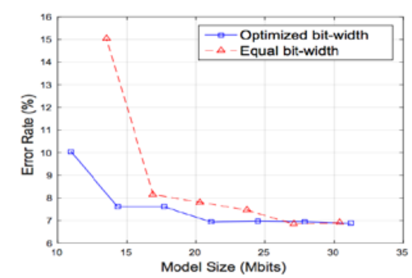
[Zhang15] X. Zhang, J. Zou, X. Ming, K. He, J. Sun: **Efficient and Accurate Approximations of Nonlinear Convolutional Networks**. CVPR 2015

Network Quantization

- Fixed point approximation
  - Quantizing both weights and activations
  - Identifying optimal fixed point bit-width allocation across layers
  - Data-driven bit-width and step size estimation: relying on Gaussian distribution assumption
  - Considering trade-off between overflow and quantization error
- Results
  - More than 20% reduction in the model size without any loss in accuracy on CIFAR-10



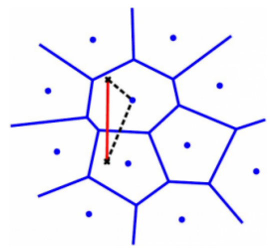
(a) Histogram of weights (b) Histogram of activations



[Lin16] D. D. Lin, S. S. Talathi, V. S. Annapureddy: **Fixed Point Quantization of Deep Convolutional Networks**. ICML 2016

Network Quantization

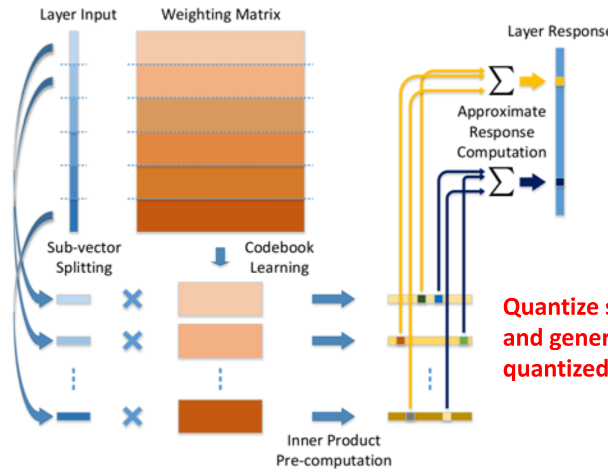
- Quantized CNN
  - Speed-up the computation
  - Reduce the storage and memory overhead of CNN models
  - Quantize convolutional filters and weight matrices of FC layers
  - Minimize the estimation error of each layer's response
  - 4-6× speed-up and 15-20× compression with 1% point loss of accuracy
- Main idea
  - Product quantization: initially proposed for approximate nearest neighbor search
  - Quantize subvectors independently and generate a large number of quantized vectorsusing Cartesian product of subvector quantizations
  - [Jegou2011TPAMI]



[Wu16] J. Wu, C. Leng, Y. Wang, Q. Hu, J. Cheng: **Quantized Convolutional Neural Networks for Mobile Devices**. CVPR 2016

Network Quantization

- Product quantization of weight matrix

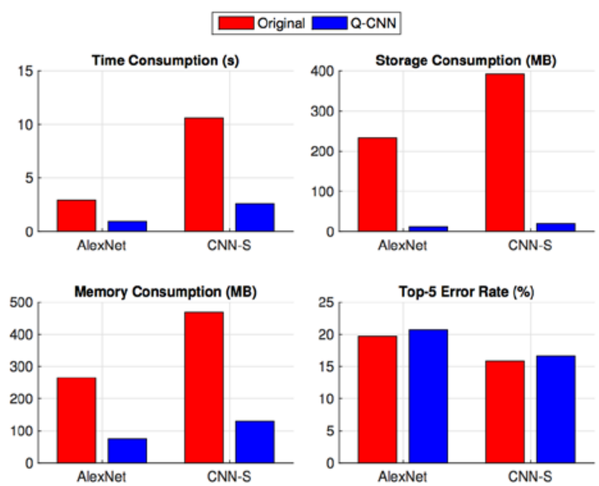


Quantize submatrix independently, and generate a large number of quantized weight vectors.

[Wu16] J. Wu, C. Leng, Y. Wang, Q. Hu, J. Cheng: **Quantized Convolutional Neural Networks for Mobile Devices**. CVPR 2016

Network Quantization

- Results



[Wu16] J. Wu, C. Leng, Y. Wang, Q. Hu, J. Cheng: **Quantized Convolutional Neural Networks for Mobile Devices**. CVPR 2016

Binary Networks

- Two versions
  - Binary-Weight-Networks: binary-valued filters
  - XNOR-Networks: binary filters and inputs for convolutions

	Network Variations		Operations used in Convolution	Memory Saving (Inference)	Computation Saving (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs 0.11 -0.21 ... -0.34 -0.25 0.61 ... 0.52	Real-Value Weights 0.12 -1.2 ... 0.41 -0.2 0.5 ... 0.68	+ , − , ×	1x	1x	%56.7
Binary Weight	Real-Value Inputs 0.11 -0.21 ... -0.34 -0.25 0.61 ... 0.52	Binary Weights 1 -1 ... 1 -1 1 ... 1	+ , −	~32x	~2x	%56.8
BinaryWeight Binary Input **(XNOR-Net)**	Binary Inputs 1 -1 ... -1 -1 1 ... 1	Binary Weights 1 -1 ... 1 -1 1 ... 1	XNOR , bitcount	~32x	~58x	%44.2

[Rastegari16] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi: **XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks**. ECCV 2016

Binary Networks

- Binary-Weight-Networks
  - Goal: $\boldsymbol{I} * \boldsymbol{W} \approx (\boldsymbol{I} \oplus \boldsymbol{B})\alpha$, where $\oplus$ is convolution without multiplications
  - Objective function

$$\underset{\boldsymbol{b},\alpha}{\operatorname{argmin}}\, J(\boldsymbol{b},\alpha) \equiv \|\boldsymbol{w} - \alpha \boldsymbol{b}\|^2$$

  - Optimization for $\boldsymbol{B}$

$$J(\boldsymbol{b},\alpha) = \alpha^2 \boldsymbol{b}^{\mathrm{T}}\boldsymbol{b} - 2\alpha \boldsymbol{w}^{\mathrm{T}}\boldsymbol{b} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} = -2\alpha \boldsymbol{w}^{\mathrm{T}}\boldsymbol{b} + (\text{constant})$$

$$\boldsymbol{b}^* = \underset{\boldsymbol{B}}{\operatorname{argmax}}\, \boldsymbol{w}^{\mathrm{T}}\boldsymbol{b} \quad \text{such that} \quad \boldsymbol{b} \in \{+1,-1\}^n \qquad \boldsymbol{b}^* = \operatorname{sign}(\boldsymbol{w})$$

  - Optimization for α

$$\frac{\partial}{\partial \alpha} J(\boldsymbol{B},\alpha) = 2\alpha \boldsymbol{b}^{\mathrm{T}}\boldsymbol{b} - 2\boldsymbol{w}^{\mathrm{T}}\boldsymbol{b} = 0$$

$$\alpha^* = \frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{b}}{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{b}} = \frac{\boldsymbol{w}^{\mathrm{T}}\operatorname{sign}(\boldsymbol{w})}{n} = \frac{1}{n}\|\boldsymbol{w}\|_1$$

[Rastegari16] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi: **XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks**. ECCV 2016

Binary Networks

- Training Binary-Weight-Networks
  - Binarize the weights during the forward pass and backward propagation
  - Weight updates in floating points to handle tiny changes effectively

Algorithm 1 Training an L-layers CNN with binary weights:

Input: A minibatch of inputs and targets $(\mathbf{I}, \mathbf{Y})$, cost function $C(\mathbf{Y}, \hat{\mathbf{Y}})$, current weight $\mathcal{W}^t$ and current learning rate η^t.
Output: updated weight $\mathcal{W}^{t+1}$ and updated learning rate η^{t+1}.

Binarizing weight filters:
for $l = 1$ to L **do**
  for k^{th} filter in l^{th} layer **do**
    $\mathcal{A}_{lk} = \frac{1}{n}\|\mathcal{W}_{lk}^t\|_{\ell 1}$
    $\mathcal{B}_{lk} = \operatorname{sign}(\mathcal{W}_{lk}^t)$
    $\widetilde{\mathcal{W}}_{lk} = \mathcal{A}_{lk}\mathcal{B}_{lk}$
$\hat{\mathbf{Y}} =$ **BinaryForward**$(\mathbf{I}, \mathcal{B}, \mathcal{A})$ // standard forward propagation except that convolutions are computed using equation 1 or 11
$\frac{\partial C}{\partial \widetilde{\mathcal{W}}} =$ **BinaryBackward**$(\frac{\partial C}{\partial \hat{\mathbf{Y}}}, \widetilde{\mathcal{W}})$ // standard backward propagation except that gradients are computed using $\widetilde{\mathcal{W}}$ instead of $\mathcal{W}^t$
$\mathcal{W}^{t+1} =$ **UpdateParameters**$(\mathcal{W}^t, \frac{\partial C}{\partial \widetilde{\mathcal{W}}}, \eta_t)$ // Any update rules (*e.g.*,SGD or ADAM)
$\eta^{t+1} =$ **UpdateLearningrate**(η^t, t) // Any learning rate scheduling function

[Rastegari16] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi: **XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks**. ECCV 2016

Binary Networks

- XNOR-Networks
  - Goal: $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W} \approx \beta \boldsymbol{H}^{\mathrm{T}} \alpha \boldsymbol{B}$
  - Objective function and solution

$$\underset{\boldsymbol{b},\alpha,\boldsymbol{h},\beta}{\operatorname{argmin}} \|\boldsymbol{x} \odot \boldsymbol{w} - \beta\alpha \boldsymbol{h} \odot \boldsymbol{b}\|^2 \equiv \underset{\boldsymbol{c},\gamma}{\operatorname{argmin}} \|\boldsymbol{y} - \gamma \boldsymbol{c}\|^2$$

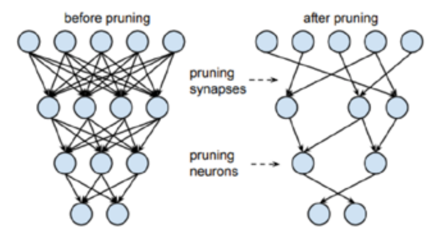
$$\boldsymbol{c}^* = \operatorname{sign}(\boldsymbol{y}) = \operatorname{sign}(\boldsymbol{x}) \odot \operatorname{sign}(\boldsymbol{w}) = \boldsymbol{h}^* \odot \boldsymbol{b}^*$$

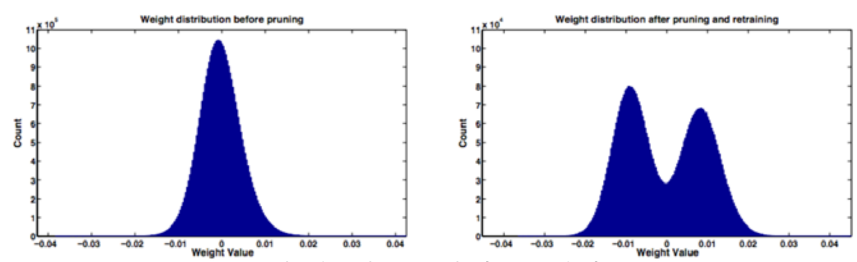
$$\gamma^* = \frac{1}{n}\|\boldsymbol{y}\|_1 \approx \left(\frac{1}{n}\|\boldsymbol{x}\|_1\right)\left(\frac{1}{n}\|\boldsymbol{w}\|_1\right) = \beta^* \alpha^*$$

[Rastegari16] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi: **XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks**. ECCV 2016

Pruning Low Magnitude Weights

- Simple approach
  - Pruning unimportant connections (with near zero weights)
  - Training network, pruning weights, retraining network (repeat)



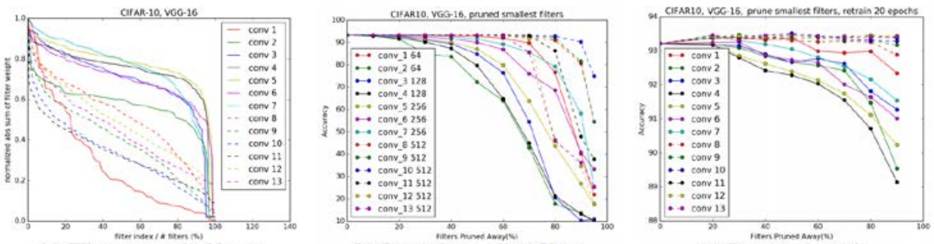


Weight distributions before and after pruning

[Han15] S. Han, J. Pool, J. Tran, W. J. Dally: **Learning both Weights and Connections for Efficient Neural Networks**. NIPS 2015

Channel Pruning

- Filter pruning
  - Motivation: reducing computational cost significantly
  - By identifying filters having a small effect on the output accuracy
- Main idea
  - Prune the channels corresponding to the filters with smallest magnitudes!
  - This idea is correlated to but better than activation-based pruning.
  - Filter pruning in the lower layers incurs filter updates in the upper ones.



(a) Filters are ranked by s_j (b) Prune the smallest filters (c) Prune and retrain

[Kadav17] H. Li, A. Kadav, I. Durdanovic, H. Samet, H. P. Graf: **Pruning Filters for Efficient ConvNets**. ICLR 2017

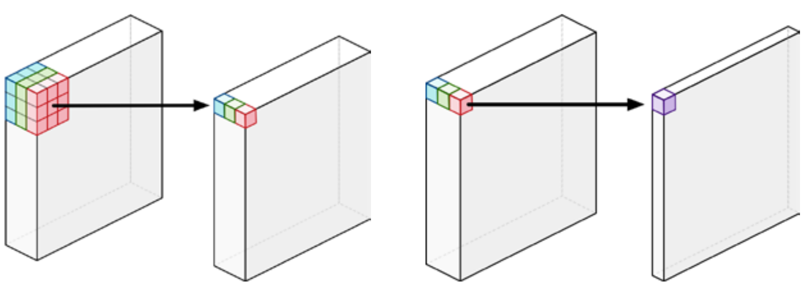
Eliminating Redundant Convolutions

- Motivation and main idea
  - Speeds up the bottleneck convolutional layers by skipping their evaluation in some of the spatial positions
  - Inspired by the loop perforation technique from source code optimization
  - Interpolates missing activations using nearest neighbors
  - Accelerates 2-4x in AlexNet and VGG
- Perforation mask
  - Marks positions for exact convolutions
  - Uniform: selects mask randomly and generates clusters (not desirable)
  - Grid:
  - Pooling structure: computes exact convolutions that are included in more pooling windows
  - Impact: estimates the impact of perforation of each position on the CNN loss function, and then removes the least important positions

[Figurnov16] M. Figurnov, A. Ibraimova, D. Vetrov, P. Kohli: **PerforatedCNNs: Acceleration through Elimination of Redundant Convolutions**. NIPS 2016

MobileNets

- Depthwise separable convolution
  - Factorizing standard convolution
    - Depthwise convolution: applies a single filter to each input channel.
    - Pointwise convolution: applies a 1×1 convolution to combine the outputs of the depthwise convolution.
  - Drastically reducing computation and model size



[Howard17] A. G. Howard, et al.: **MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications**. arXiv:1704.04861, 2017

MobileNets

- Two simple global hyperparameters
  - Width multiplier: thinner model
  - Resolution multiplier: reduced representation
  - Controlling trade off between latency and accuracy

[Howard17] A. G. Howard, et al.: **MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications**. arXiv:1704.04861, 2017