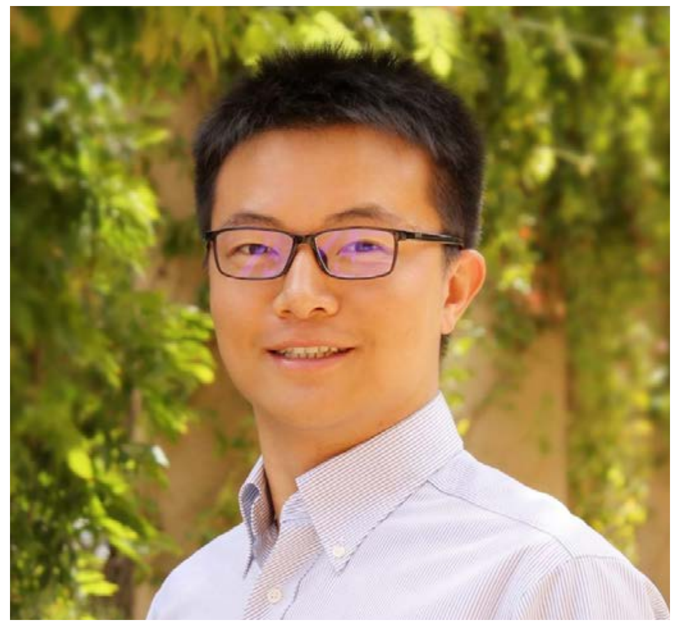
MIT HAN LAB

EfficientML.ai Lecture 03: Pruning and Sparsity

Part I

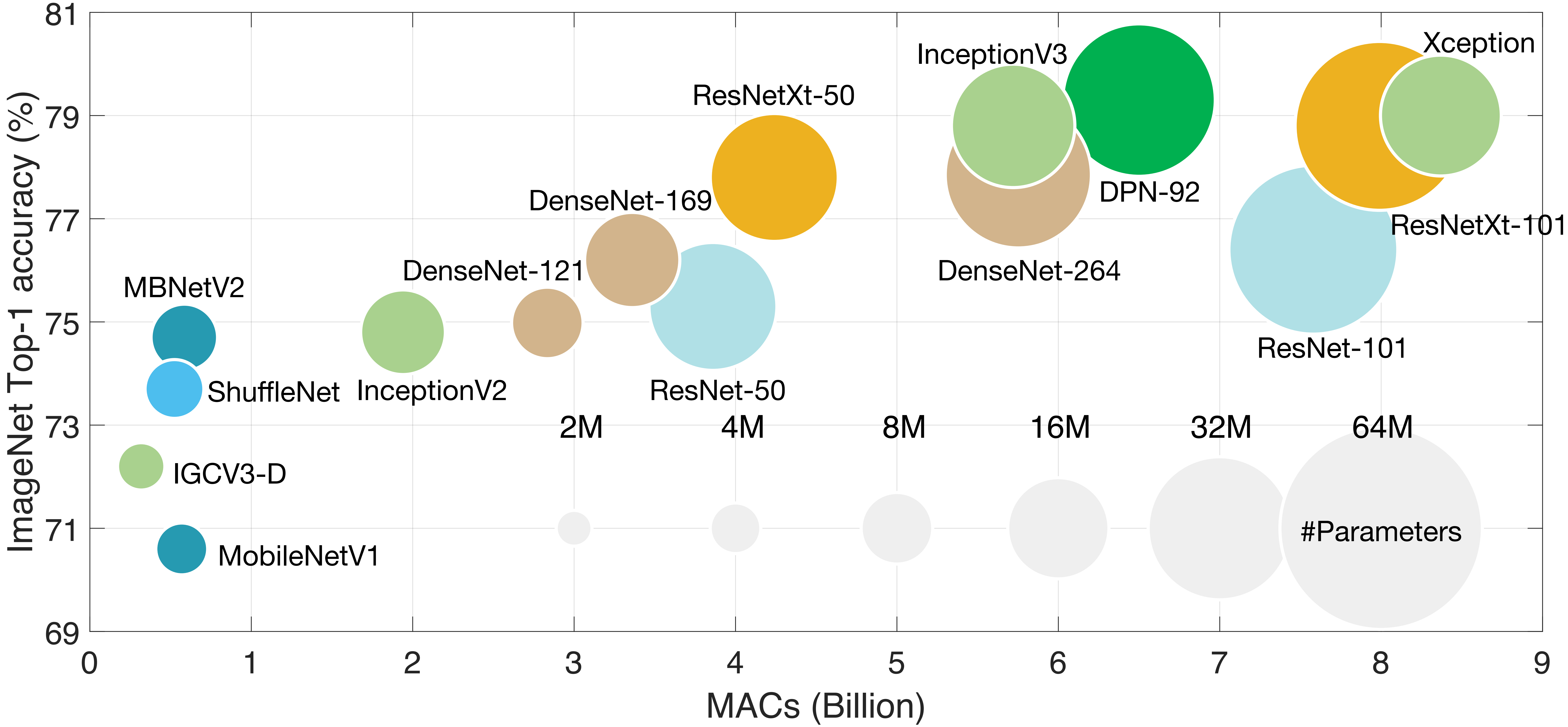


Song Han

Associate Professor, MIT
Distinguished Scientist, NVIDIA

@SongHan_MIT

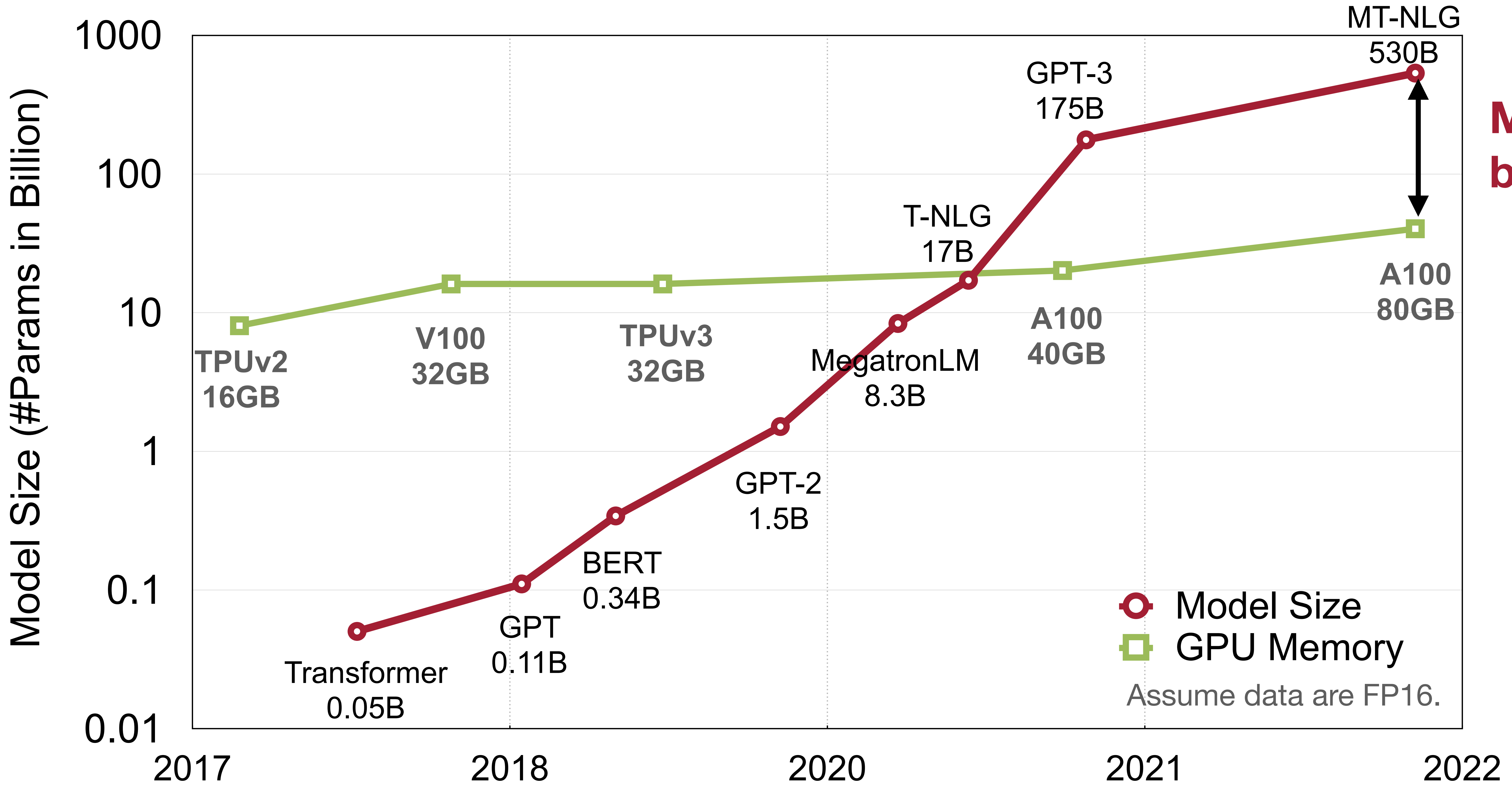
Today's AI is too BIG!



Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey [Deng *et al.*, IEEE 2020]

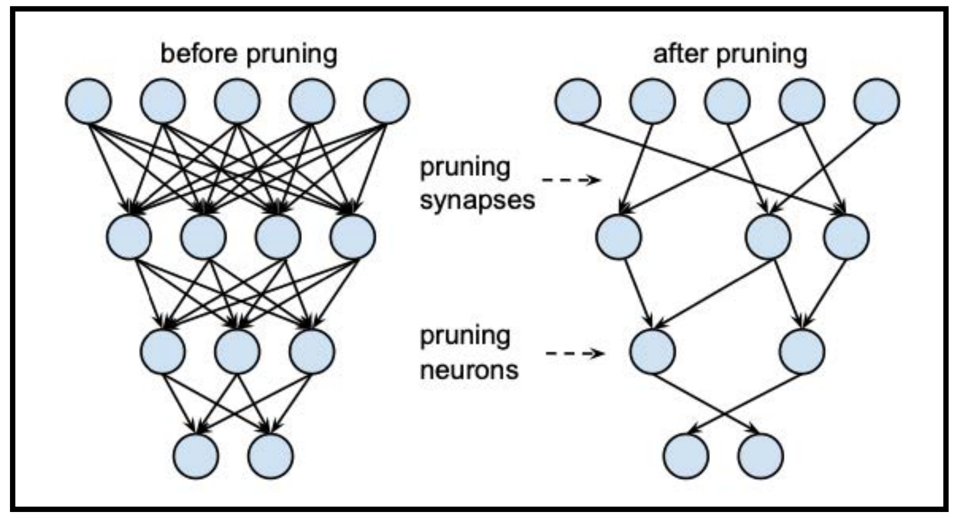
Efficient Deep Learning Techniques are Essential

Bridges the Gap between the Supply and Demand of Computation

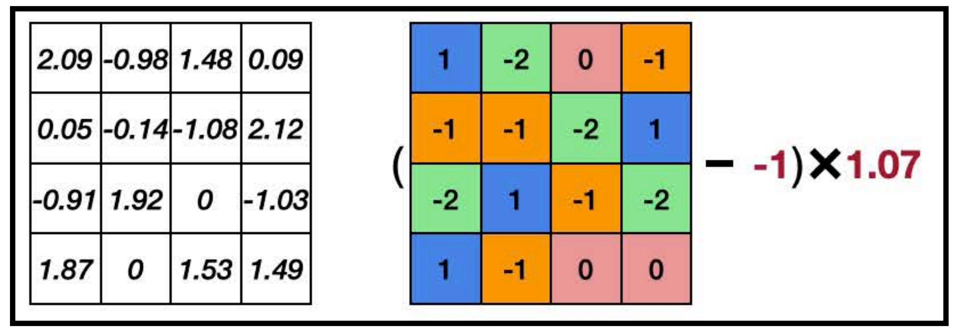


Model compression bridges the gap.

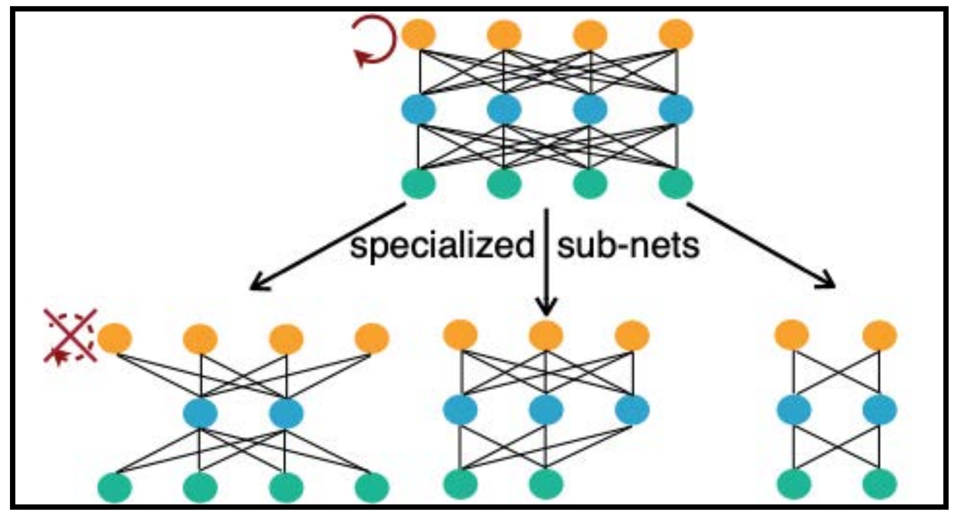
Part 1 of This Course: Efficient Inference



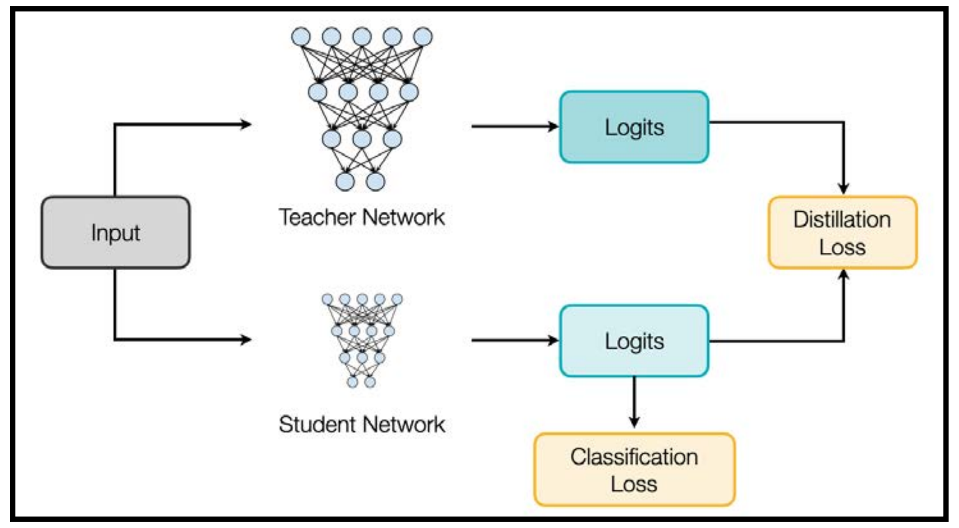
Pruning



Quantization



Neural Architecture Search



Knowledge Distillation

9/14	Lecture 3: **Pruning and Sparsity (Part I)** [slides] [video] [video (live)]	
9/19	Lecture 4: **Pruning and Sparsity (Part II)** [slides] [video] [video (live)]	Lab 1 out
9/21	Lecture 5: **Quantization (Part I)** [slides] [video] [video (live)]	
9/26	Lecture 6: **Quantization (Part II)** [slides] [video] [video (live)]	
9/28	Lecture 7: **Neural Architecture Search (Part I)** [slides] [video] [video (live)]	Lab 1 due, Lab 2 out
10/3	Lecture 8: **Neural Architecture Search (Part II)** [slides] [video] [video (live)]	
10/5	Lecture 9: **Knowledge Distillation** [slides] [video] [video (live)]	Lab 3 out
10/10	Student Holiday — No Class	
10/12	Lecture 10: **MCUNet** [slides] [video] [video (live)]	Lab 2 due
10/17	Lecture 11: **TinyEngine** [slides] [video] [video (live)]	

MLPerf (the Olympic Game for AI Computing)

Closed Division vs Open Division

- The open division submission on BERT: more than 4x while maintaining 99% accuracy.

	Closed Division	Open Division	Speedup
Offline samples/sec	1029	4609	4.5x

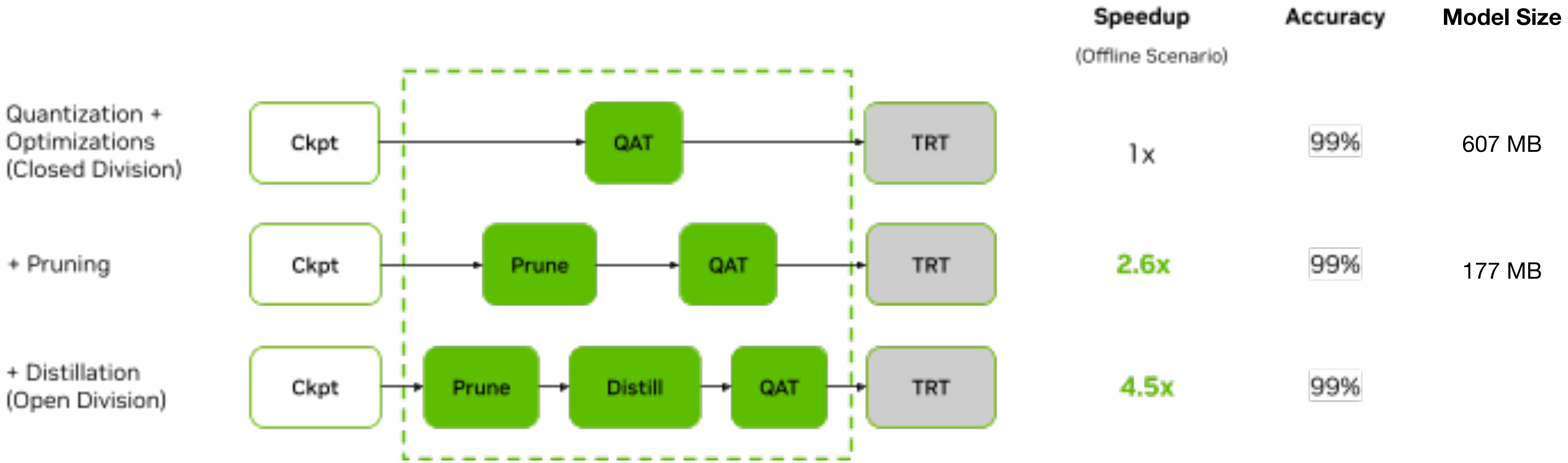
BERT Large performance metrics for both closed division and open division

Leading MLPerf Inference v3.1 Results with NVIDIA GH200 Grace Hopper Superchip Debut

MLPerf (the Olympic Game for AI Computing)

Key techniques: pruning, distillation, quantization

- The open division submission on BERT: more than 4x while maintaining 99% accuracy.

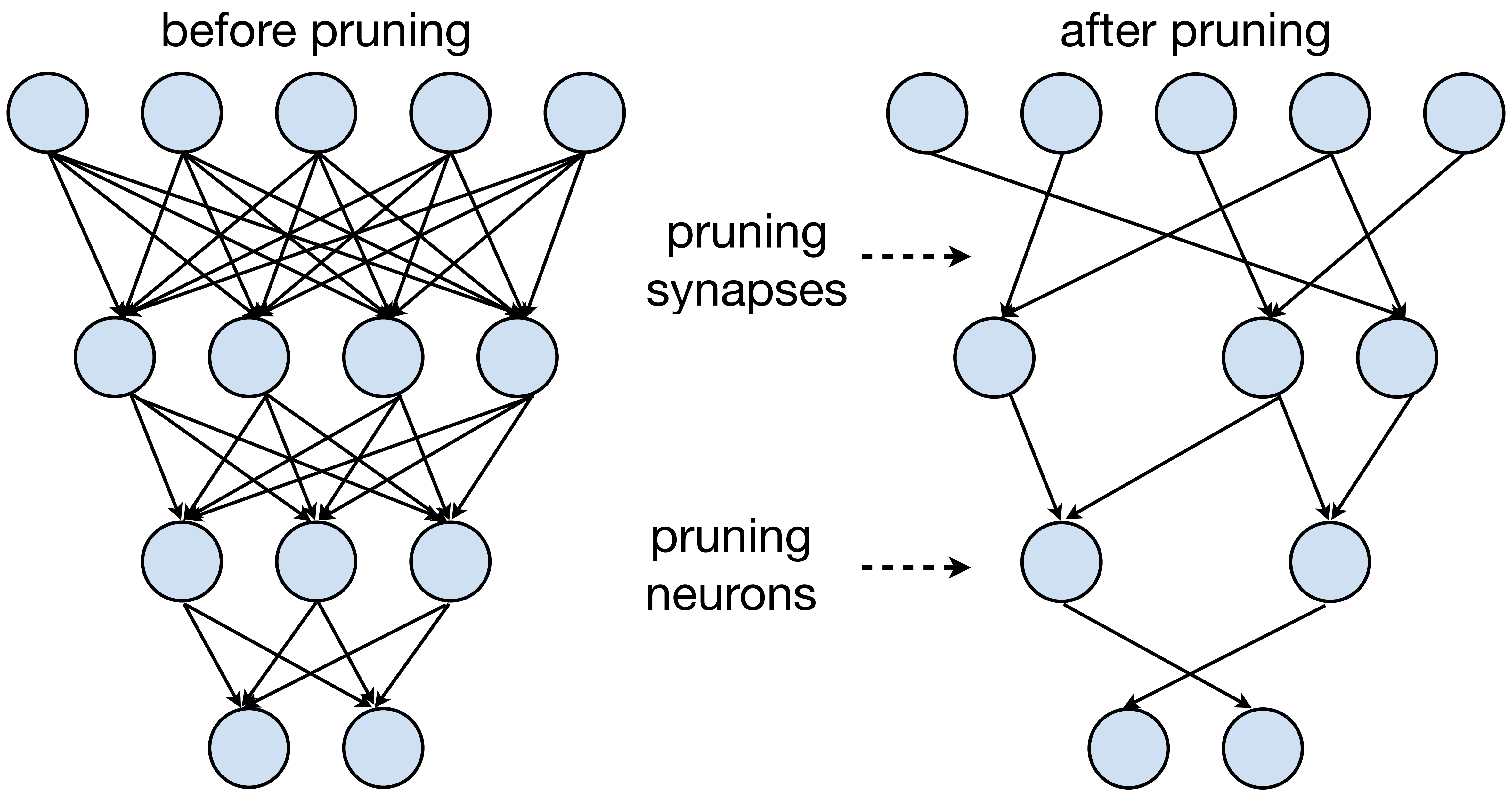


Leading MLPerf Inference v3.1 Results with NVIDIA GH200 Grace Hopper Superchip Debut

Lecture Plan

Today we will:

1. Introduce ***neural network pruning*** which can reduce the parameter counts of neural networks by more than 90%, decreasing the storage requirements and improving computation efficiency of neural networks.
2. Go through all steps of pruning, and introduce different ***granularities*** and ***criteria*** of neural network pruning.



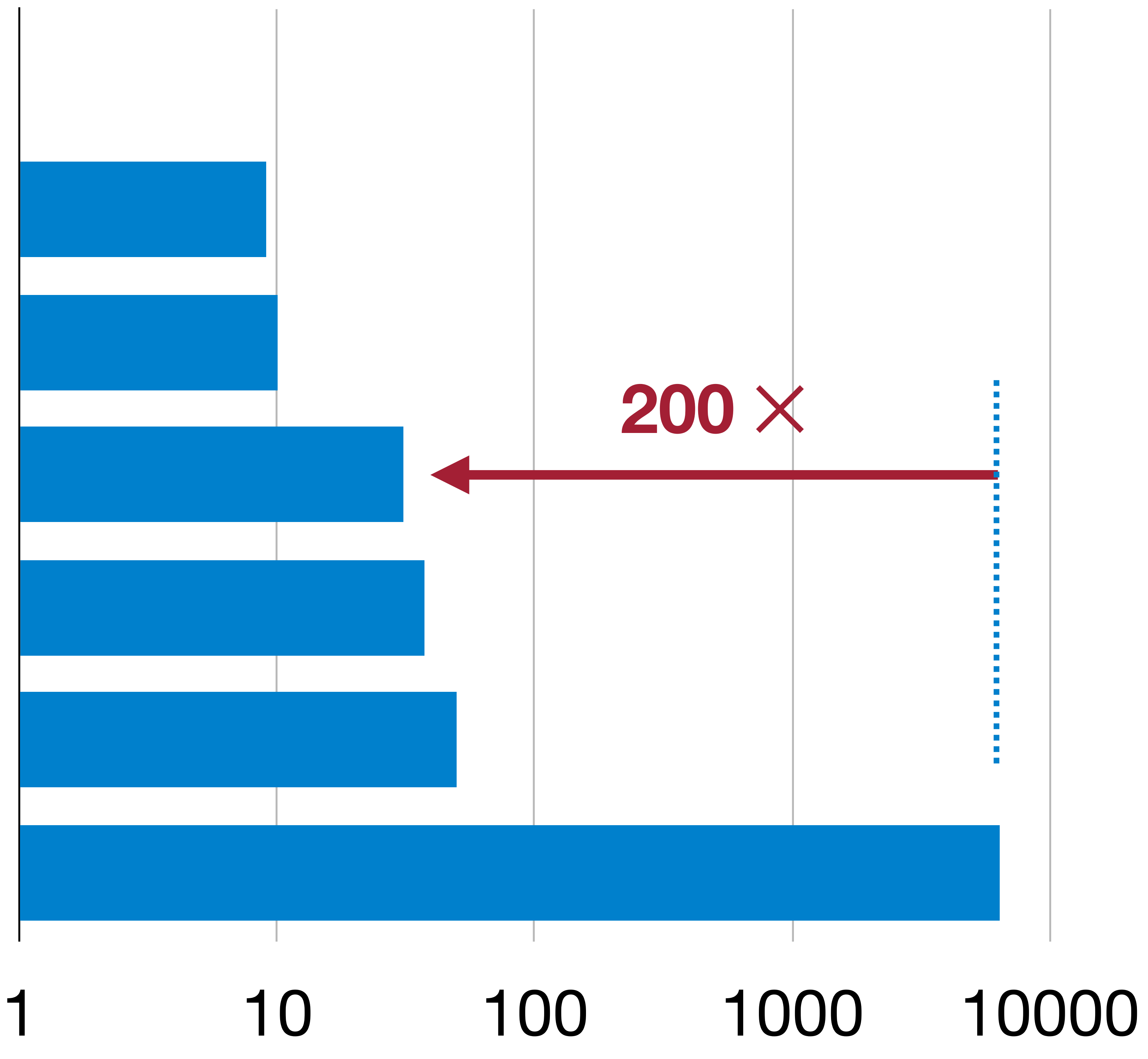
Memory is Expensive

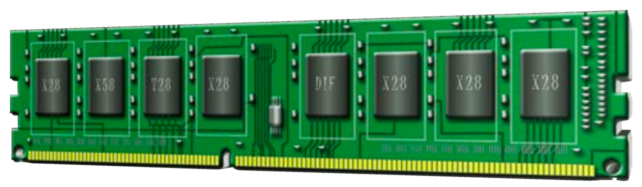
Data Movement → More Memory Reference → More Energy

Operation	Energy [pJ]
32 bit int ADD	0.1
32 bit float ADD	0.9
32 bit Register File	1
32 bit int MULT	3.1
32 bit float MULT	3.7
32 bit SRAM Cache	5
32 bit DRAM Memory	640

Rough Energy Cost For Various Operations in 45nm 0.9V

Relative Energy Cost



1  = 200 ×+

This image is in the public domain

Computing's Energy Problem (and What We Can Do About it) [Horowitz, M., IEEE ISSCC 2014]

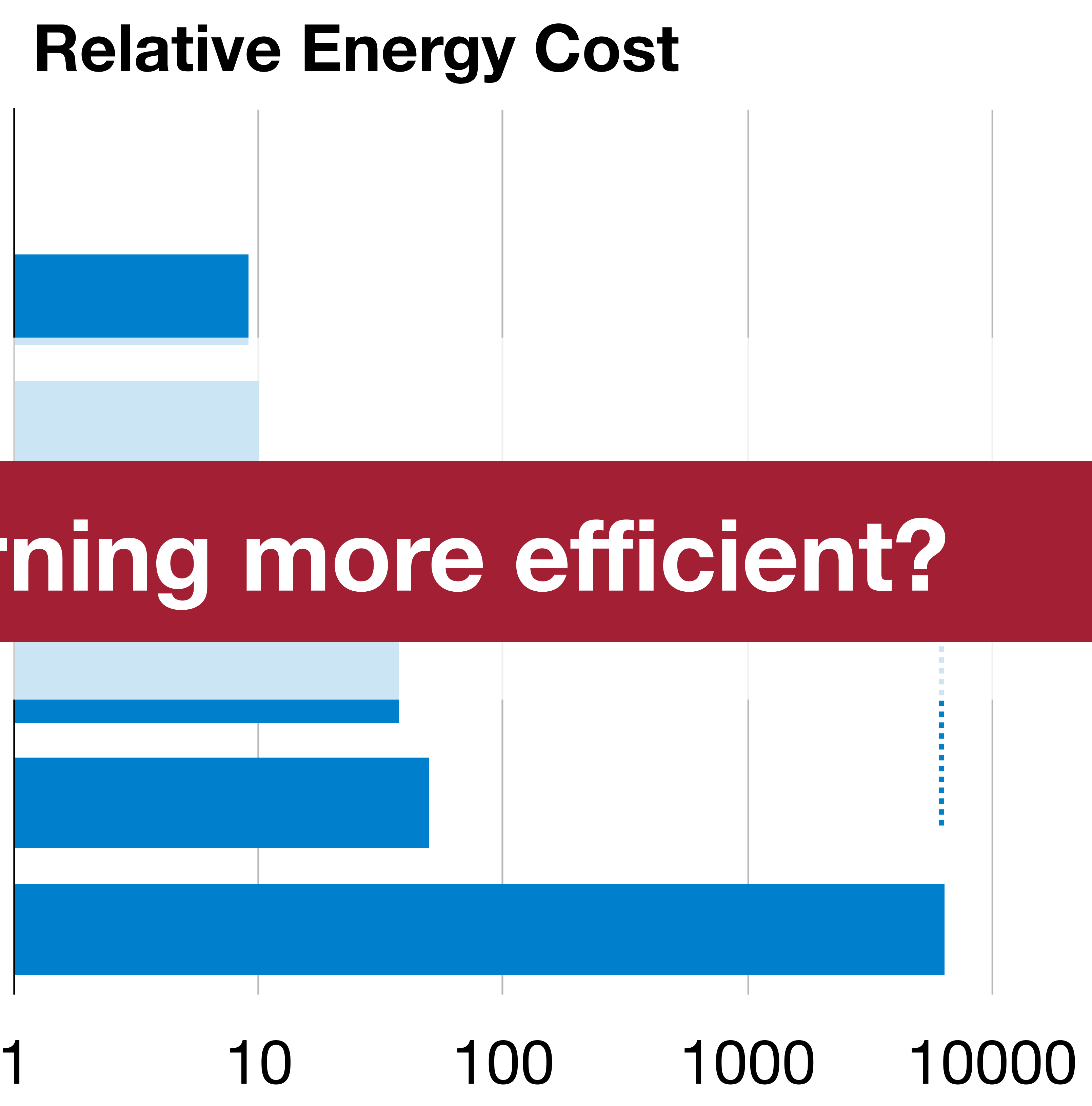
Memory is Expensive

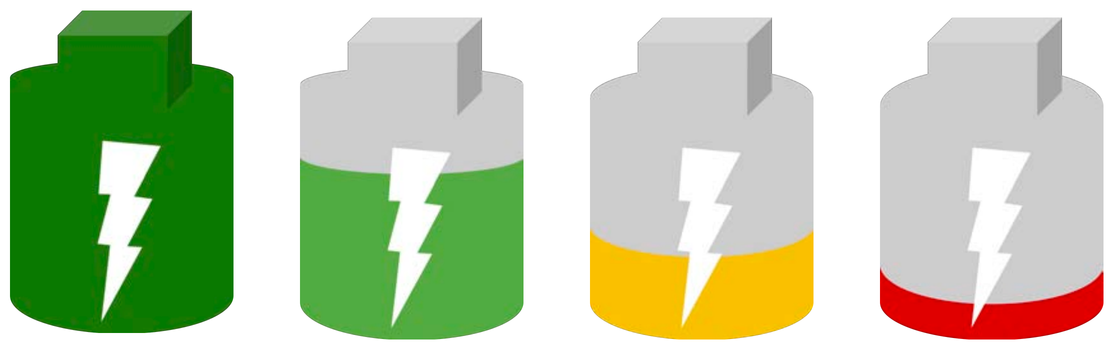
Data Movement → More Memory Reference → More Energy

Operation	Energy [pJ]
32 bit int ADD	0.1
32 bit float ADD	0.9
32 bit Register File	1
32 bit float MULT	3.7
32 bit SRAM Cache	5
32 bit DRAM Memory	640

Rough Energy Cost For Various Operations in 45nm 0.9V

How should we make deep learning more efficient?



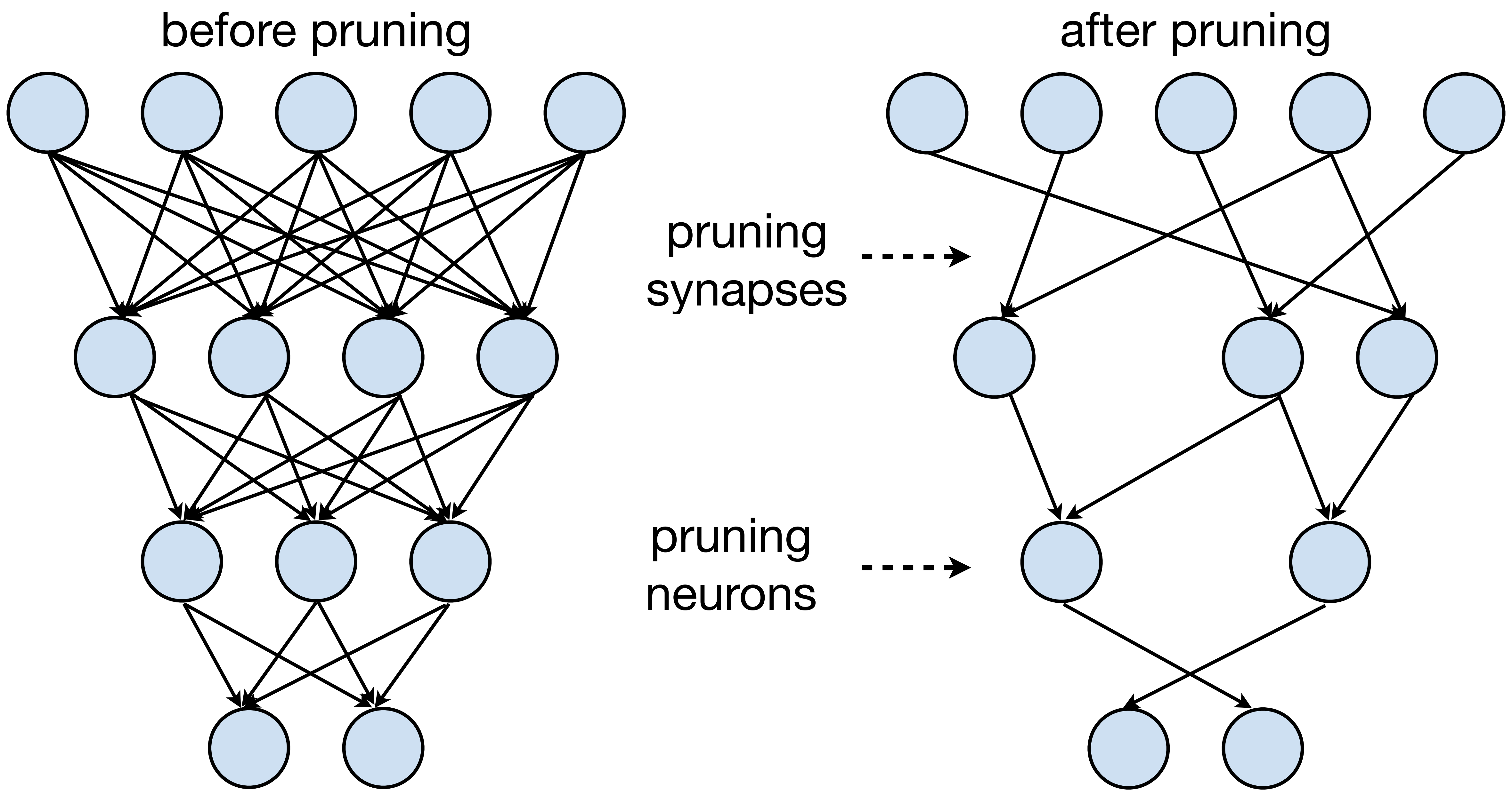


Battery images are in the public domain
Image 1, image 2, image 2, image 4

Computing's Energy Problem (and What We Can Do About it) [Horowitz, M., IEEE ISSCC 2014]

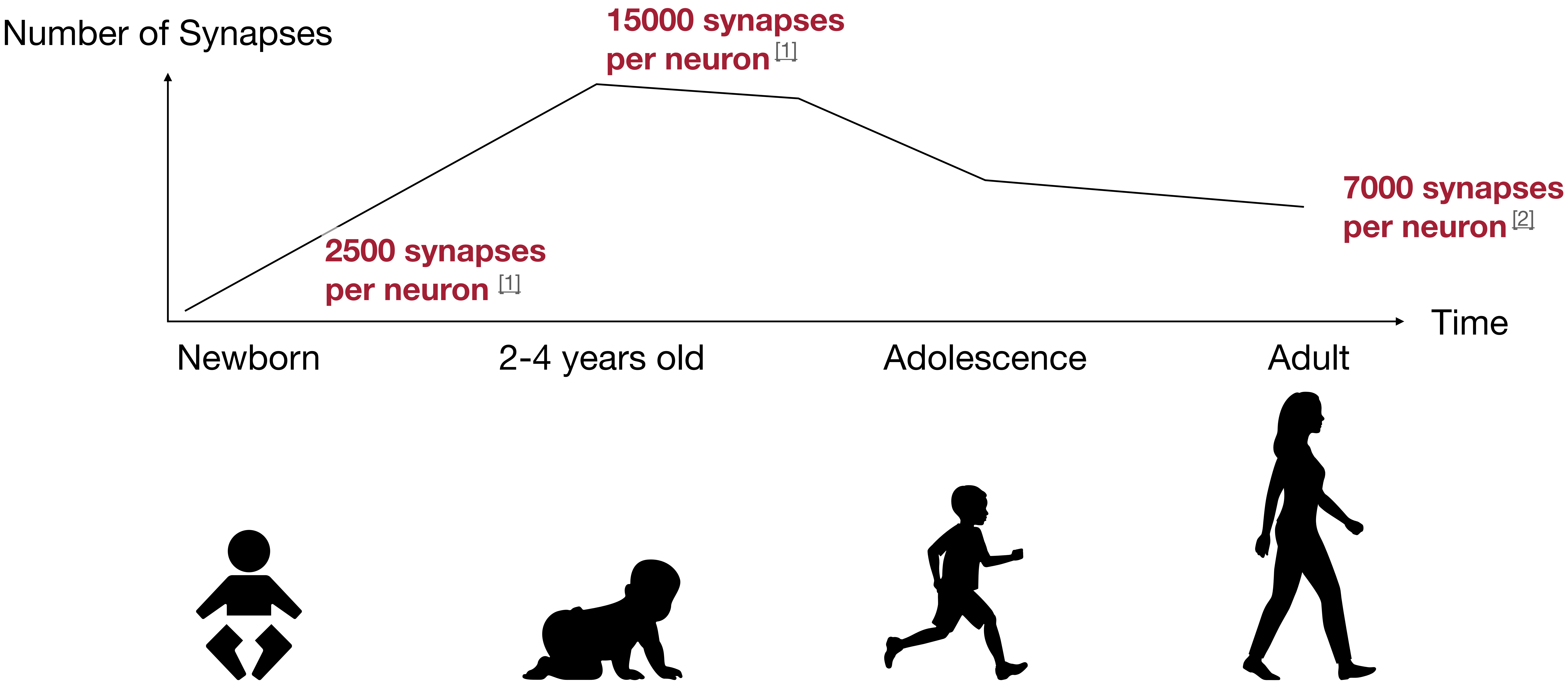
Neural Network Pruning

- **Introduction to Pruning**
  - What is pruning?
  - How should we formulate pruning?
- **Determine the Pruning Granularity**
  - In what pattern should we prune the neural network?
- **Determine the Pruning Criterion**
  - What synapses/neurons should we prune?
- **Determine the Pruning Ratio**
  - What should target sparsity be for each layer?
- **Fine-tune/Train Pruned Neural Network**
  - How should we improve performance of pruned models?



Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Pruning Happens in Human Brain

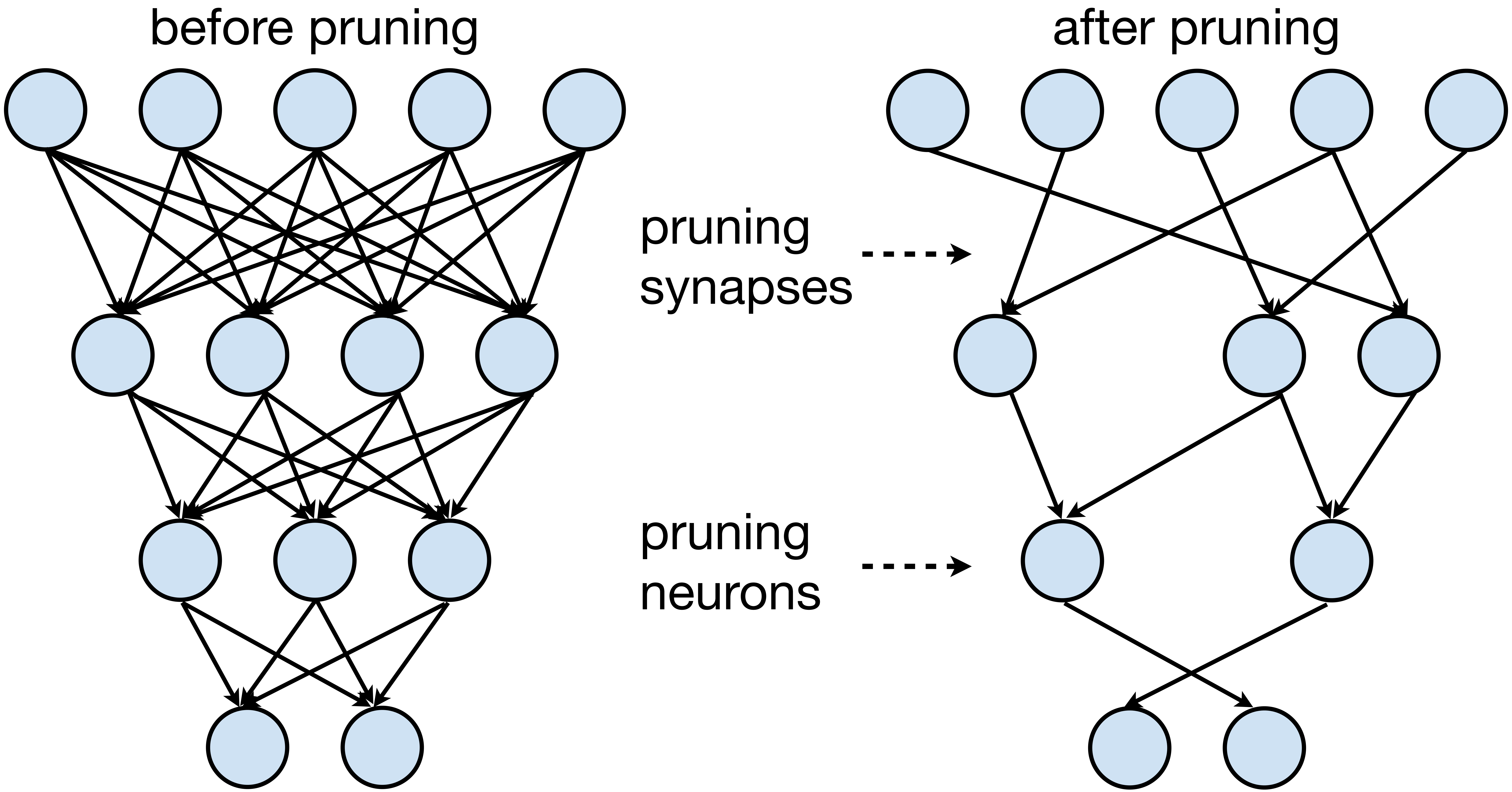


Do We Have Brain to Spare? [Drachman DA, Neurology 2004]
Peter Huttenlocher (1931–2013) [Walsh, C. A., Nature 2013]

Data Source: 1, 2
Slide Inspiration: Alila Medical Media

Neural Network Pruning

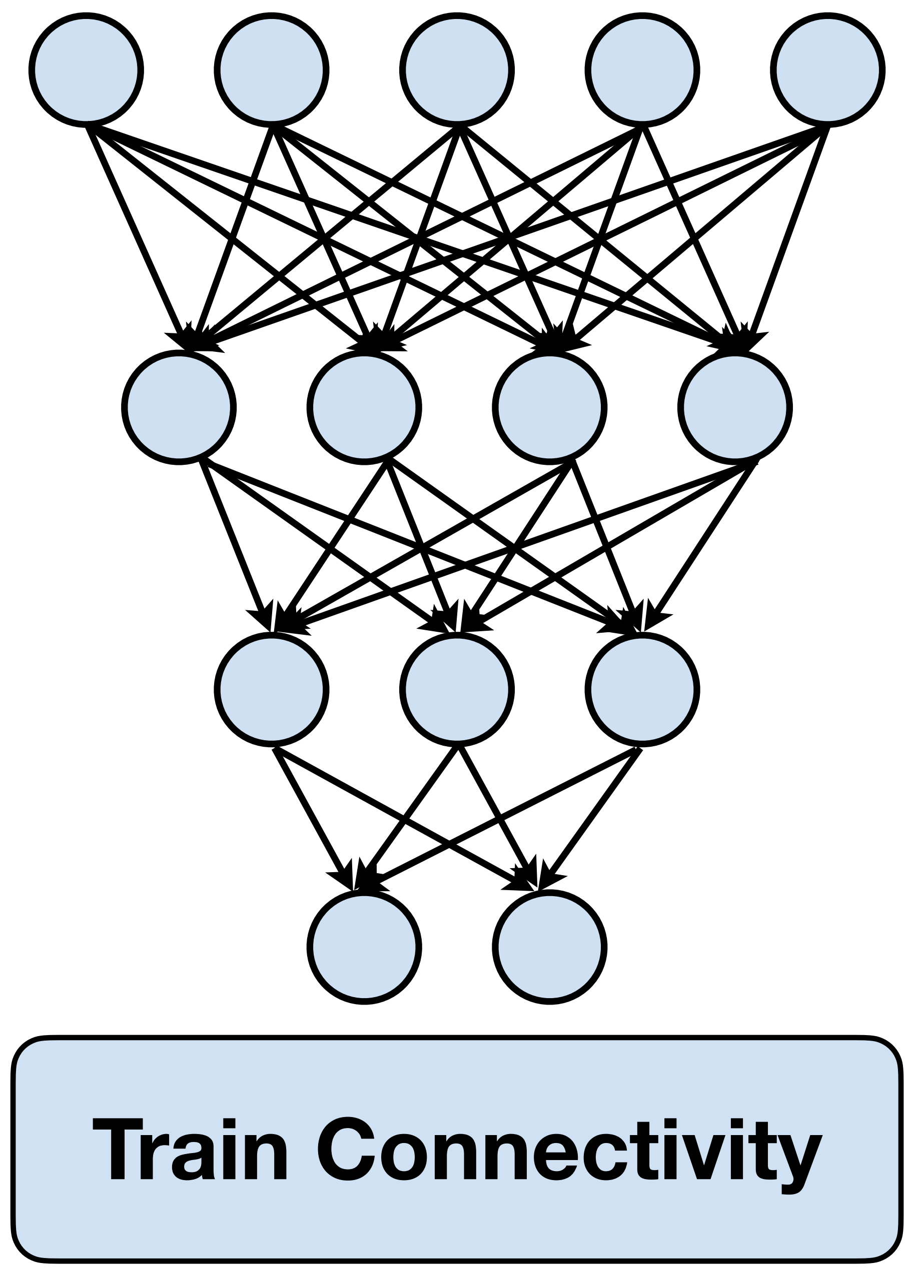
Make neural network smaller by removing synapses and neurons

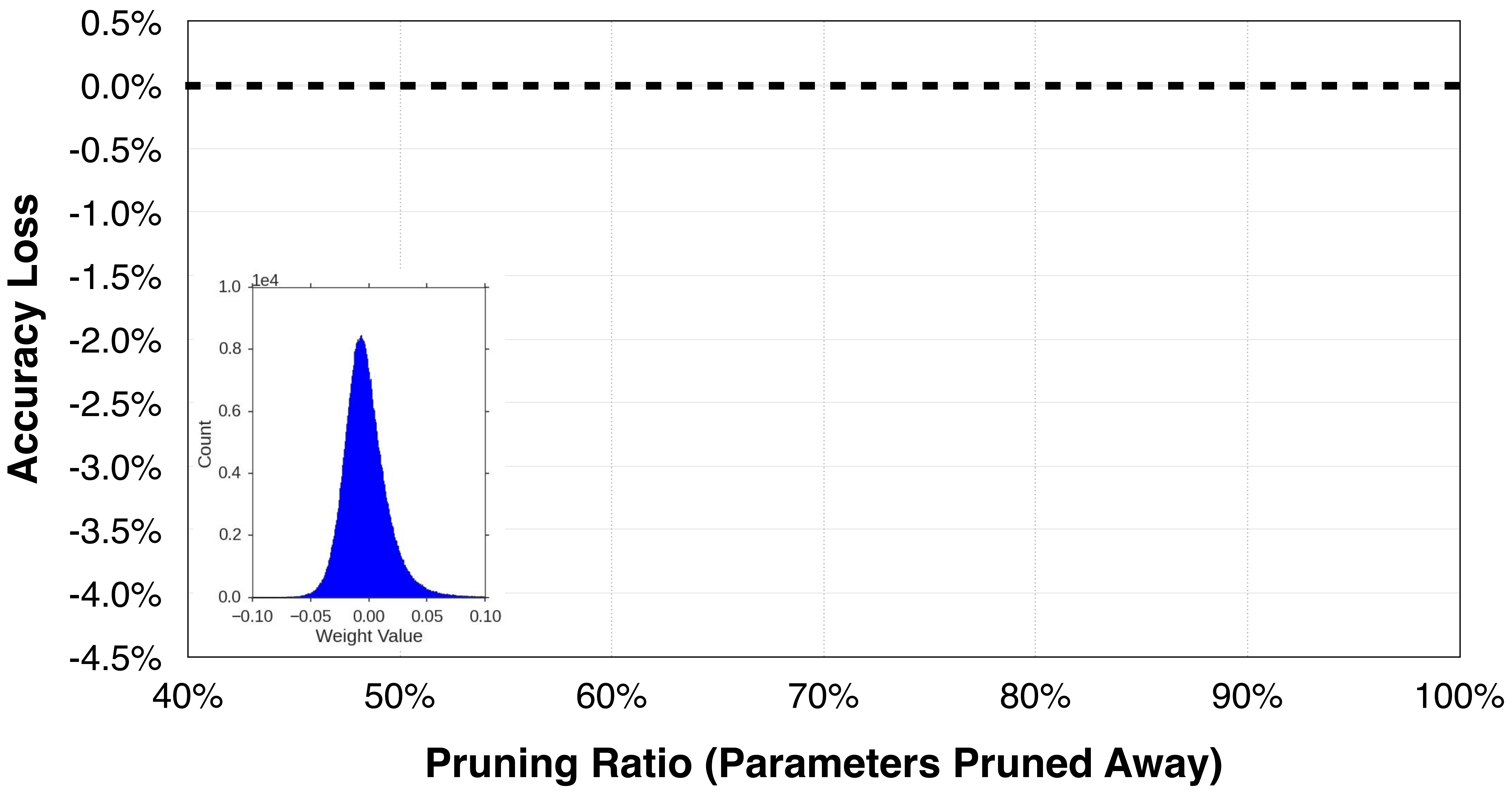


Optimal Brain Damage [LeCun *et al.,* NeurIPS 1989]
Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Neural Network Pruning

Make neural network smaller by removing synapses and neurons

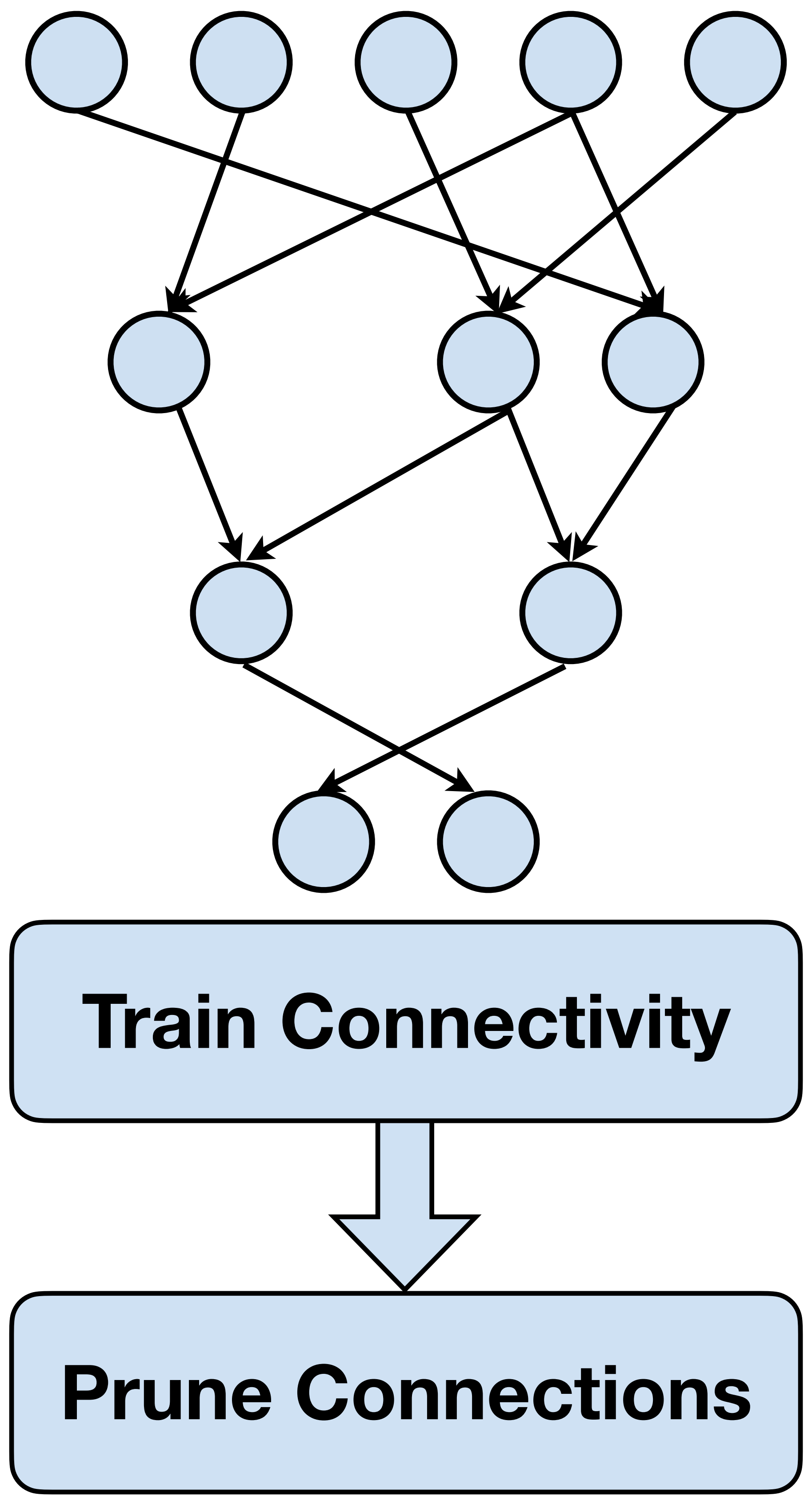


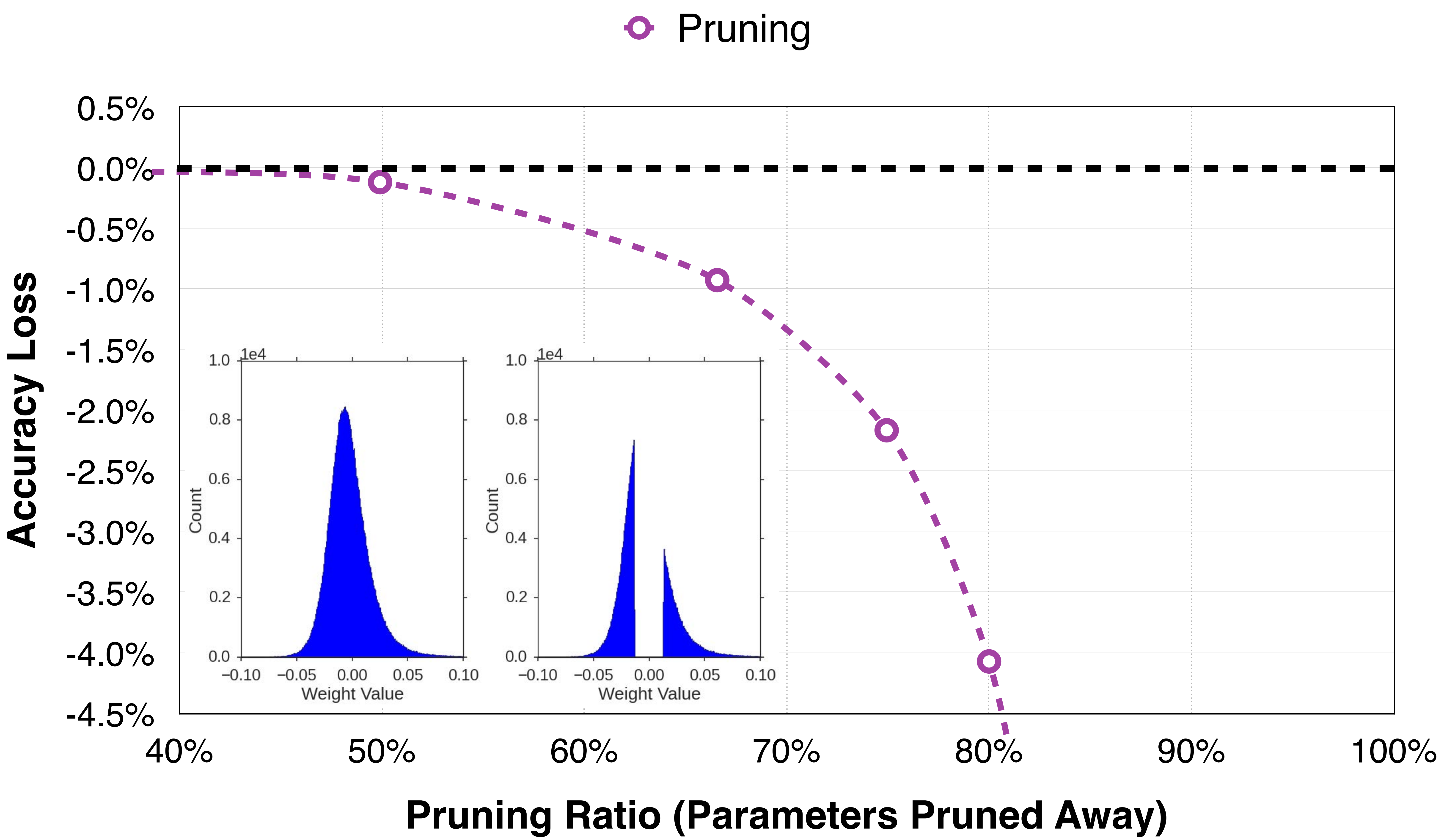


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Neural Network Pruning

Make neural network smaller by removing synapses and neurons

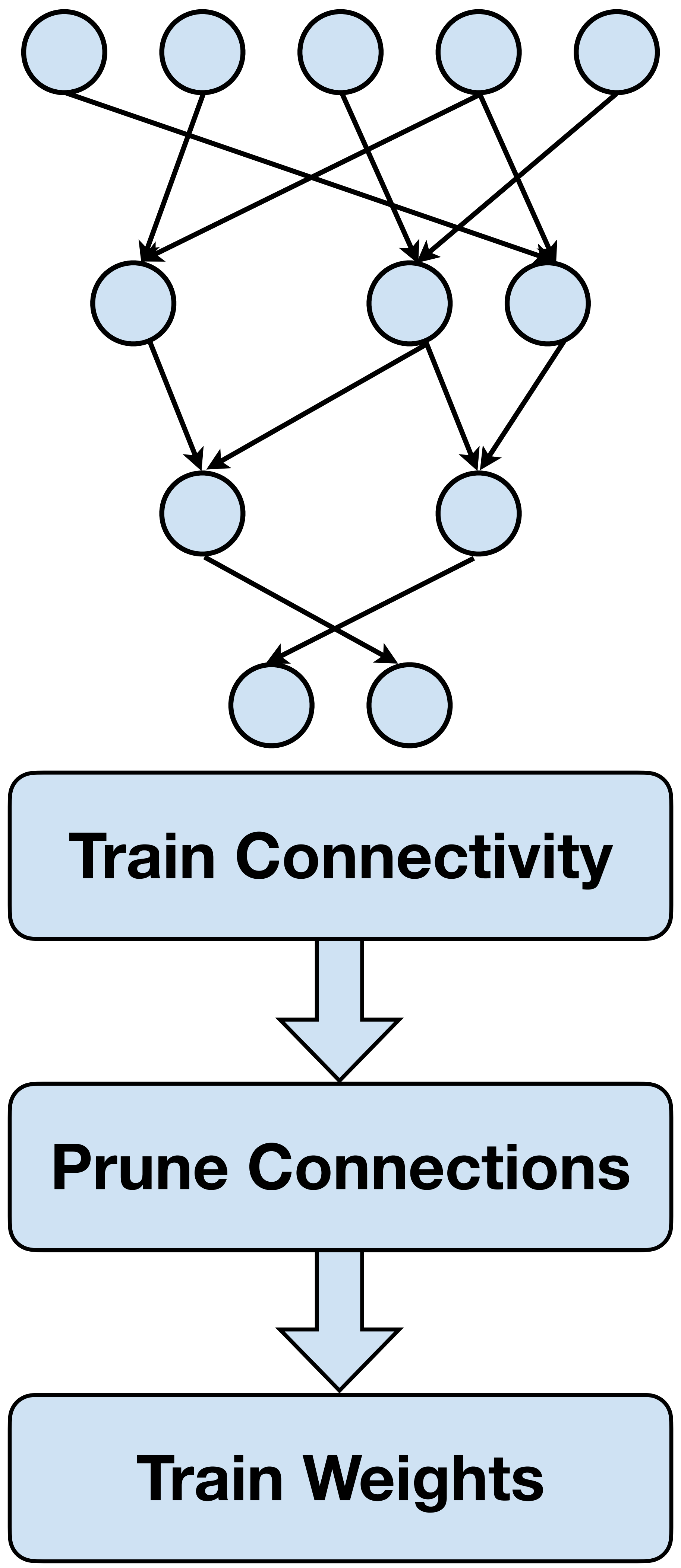


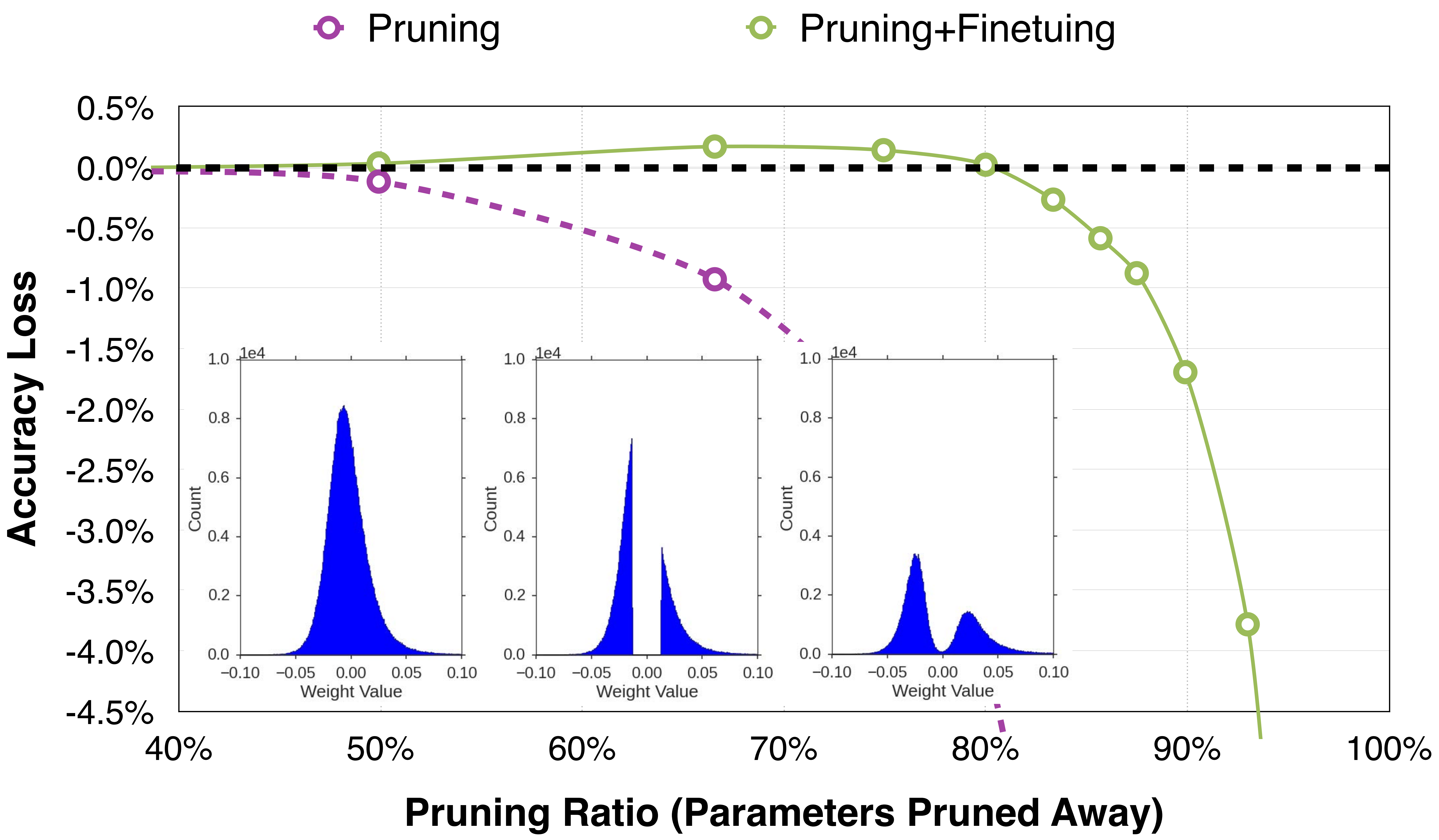


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Neural Network Pruning

Make neural network smaller by removing synapses and neurons

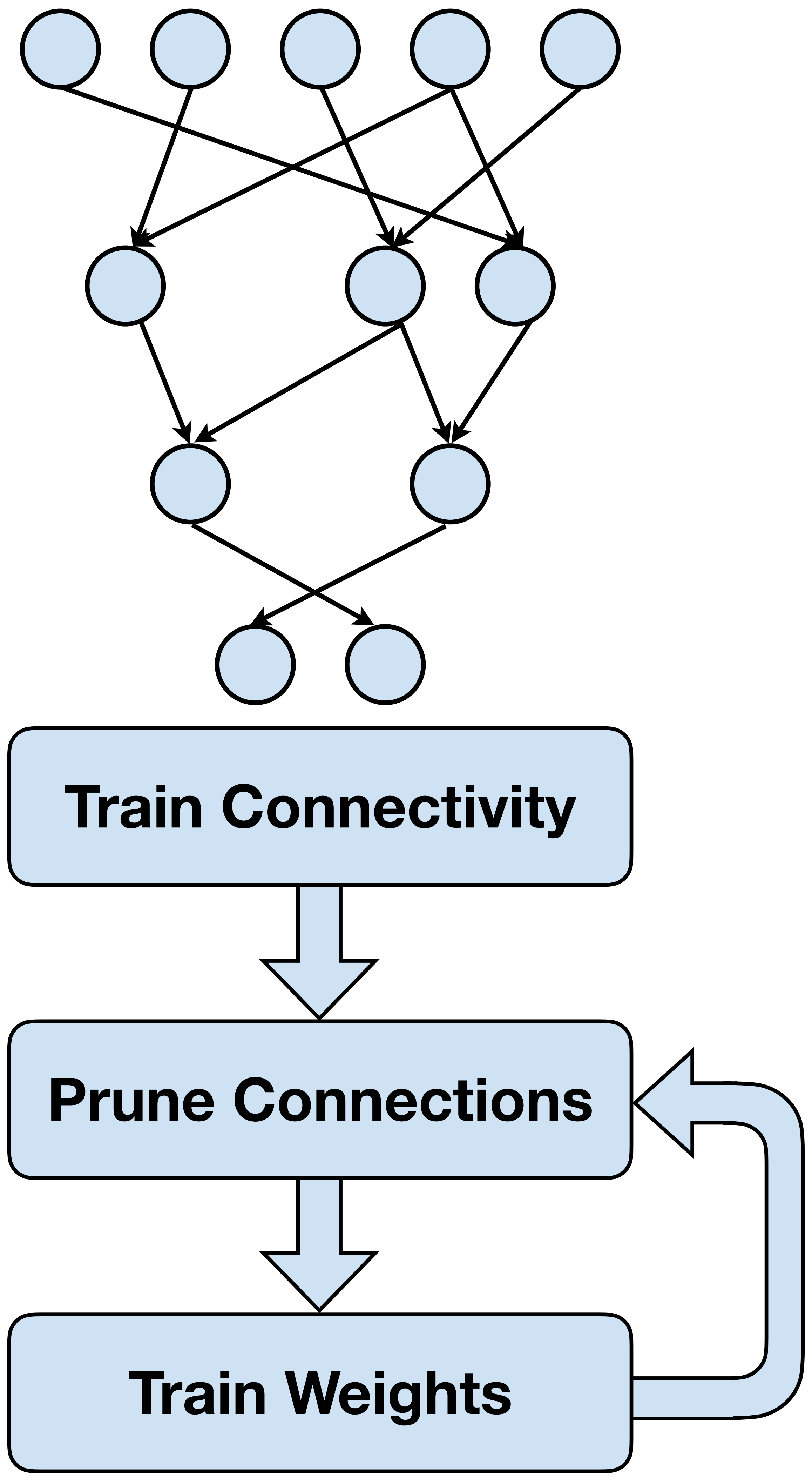


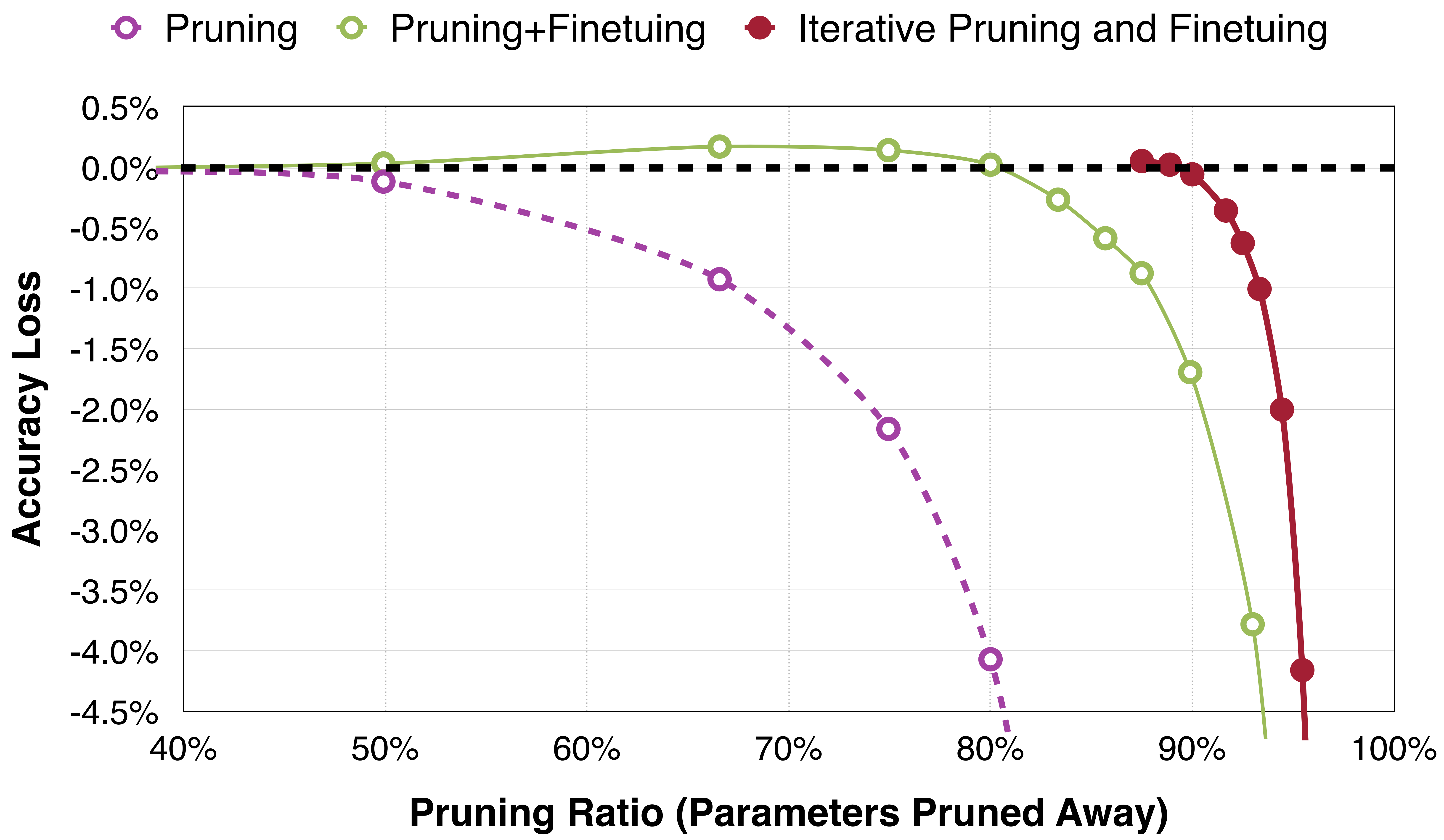


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Neural Network Pruning

Make neural network smaller by removing synapses and neurons





Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Neural Network Pruning

Make neural network smaller by removing synapses and neurons

Neural Network	#Parameters			MACs
	Before Pruning	After Pruning	Reduction	Reduction
AlexNet	61 M	6.7 M	**9 ×**	3 ×
VGG-16	138 M	10.3 M	**12 ×**	5 ×
GoogleNet	7 M	2.0 M	**3.5 ×**	5 ×
ResNet50	26 M	7.47 M	**3.4 ×**	6.3 ×
SqueezeNet	1 M	0.38 M	**3.2 ×**	3.5 ×

Efficient Methods and Hardware for Deep Learning [Han S., Stanford University]

Neural Network Pruning

Pruning the NeuralTalk LSTM does not hurt image caption quality.

Baseline: a basketball player in a white uniform is playing with a ball .

Pruned 90%: a basketball player in a white uniform is playing with a basketball.

Baseline: a brown dog is running through a grassy field.

Pruned 90%: a brown dog is running through a grassy area.

Baseline: a man is riding a surfboard on a wave.

Pruned 90%: a man in a wetsuit is riding a wave on a beach.

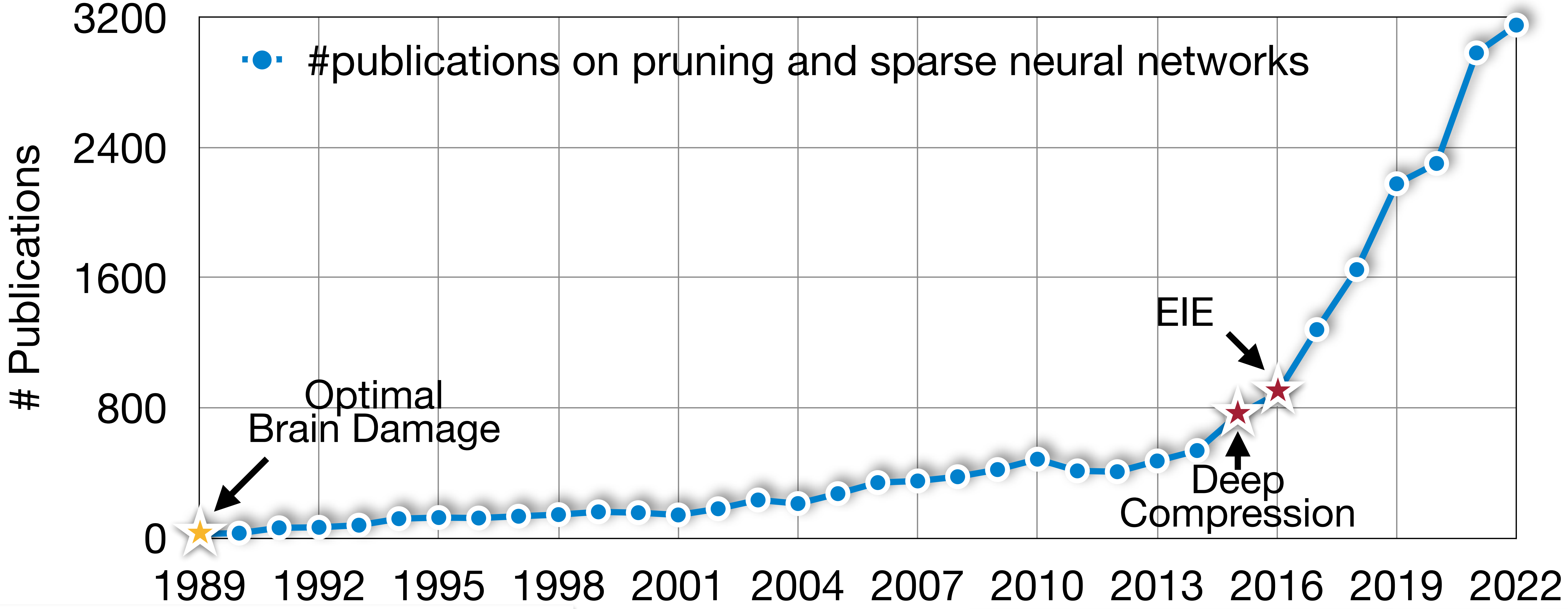
Baseline: a soccer player in red is running in the field.

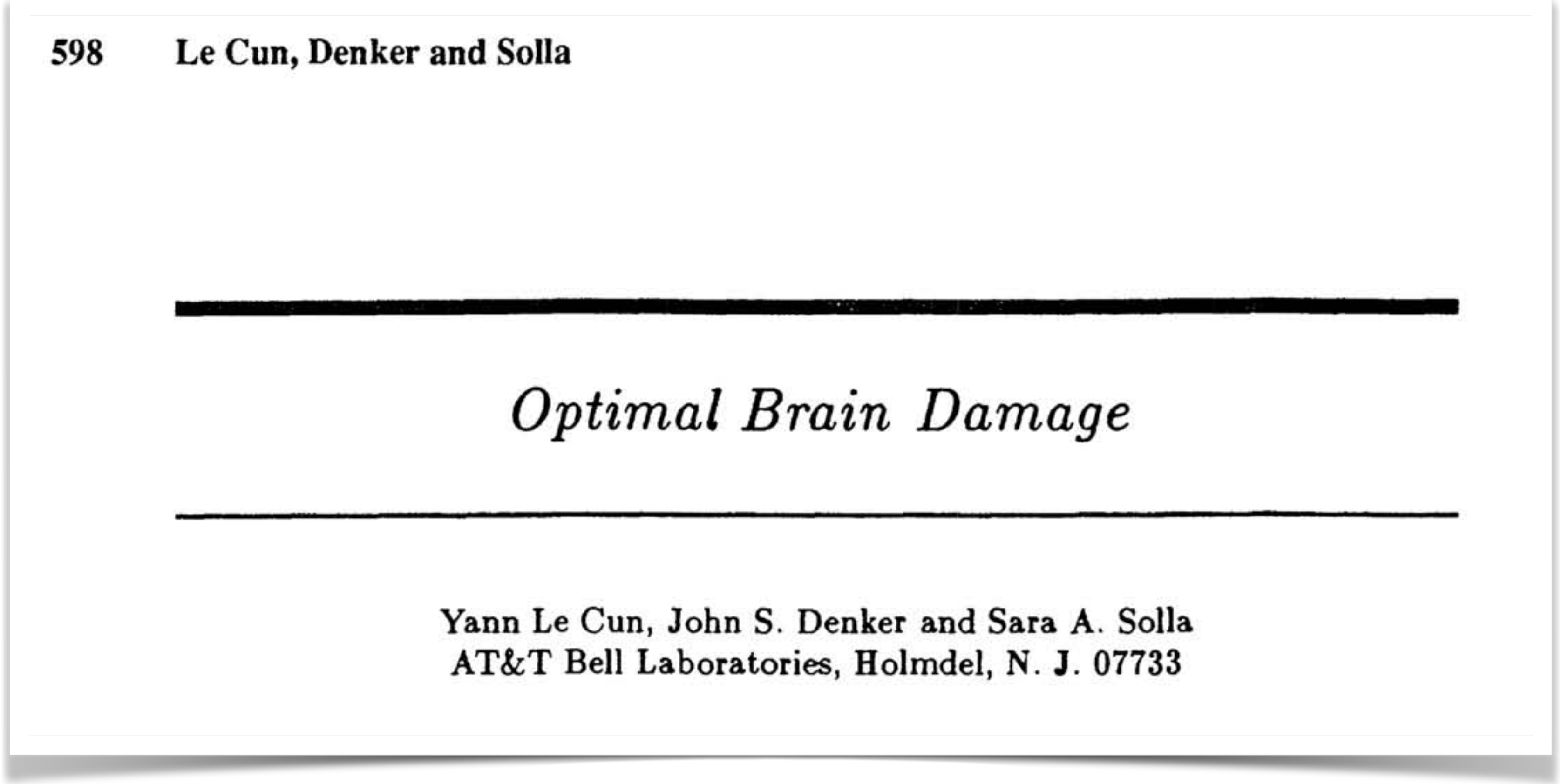
Pruned <u>95%</u>: a man in a red shirt and black and white black shirt is running through a field.

Efficient Methods and Hardware for Deep Learning [Han S., Stanford University]

Neural Network Pruning

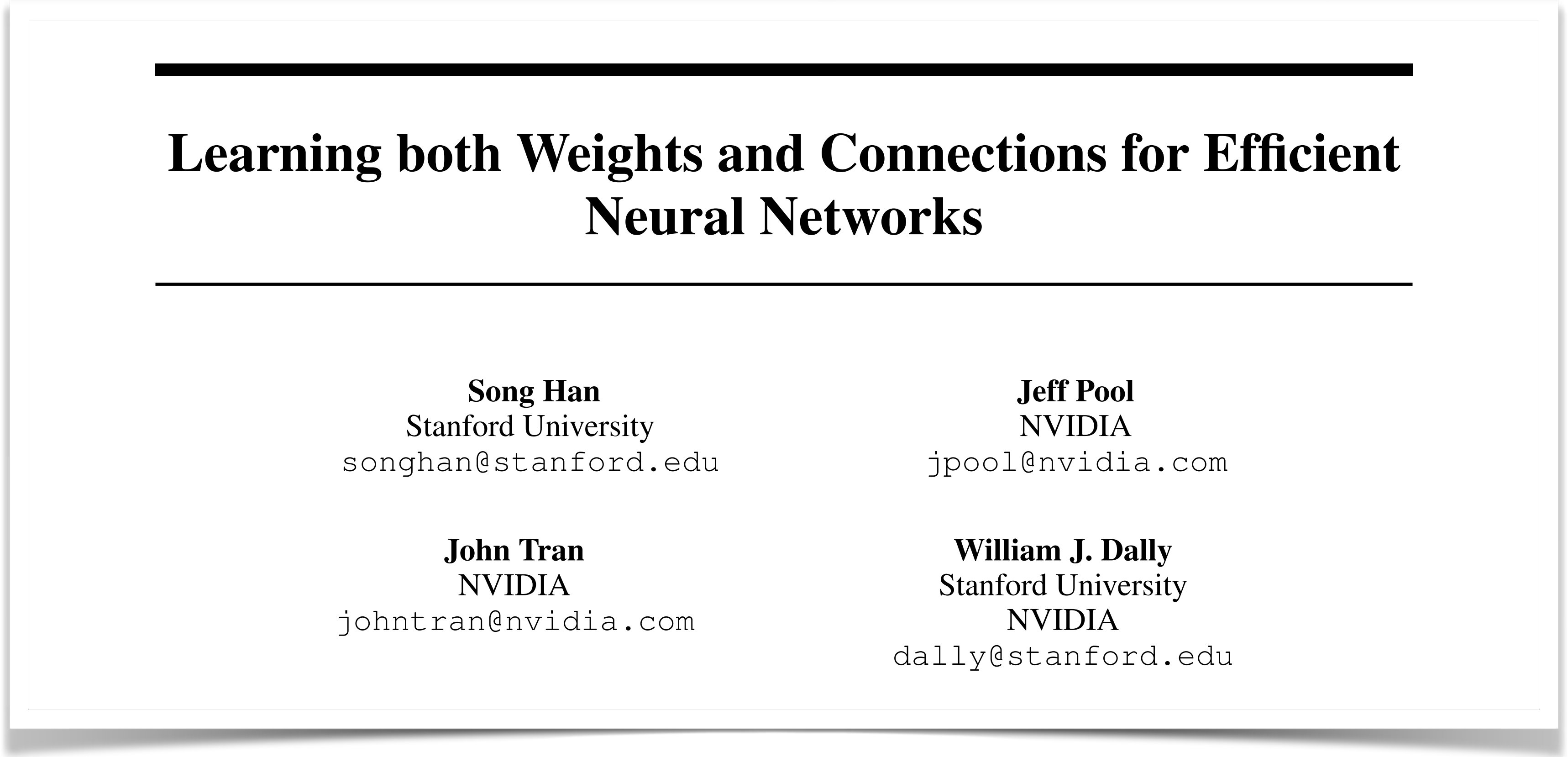
Make neural network smaller by removing synapses and neurons




598 Le Cun, Denker and Solla

Optimal Brain Damage

Yann Le Cun, John S. Denker and Sara A. Solla
AT&T Bell Laboratories, Holmdel, N. J. 07733


Learning both Weights and Connections for Efficient Neural Networks

Song Han
Stanford University
songhan@stanford.edu

Jeff Pool
NVIDIA
jpool@nvidia.com

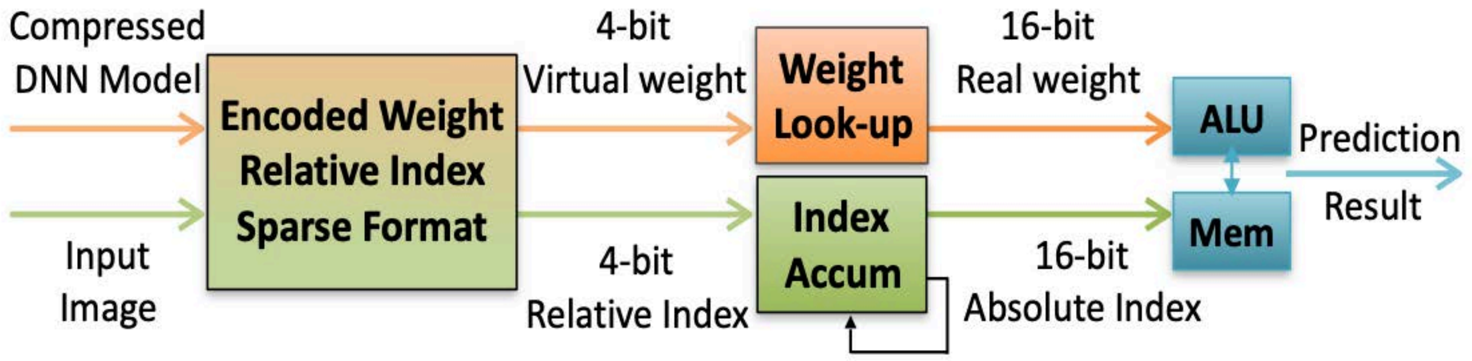
John Tran
NVIDIA
johntran@nvidia.com

William J. Dally
Stanford University
NVIDIA
dally@stanford.edu

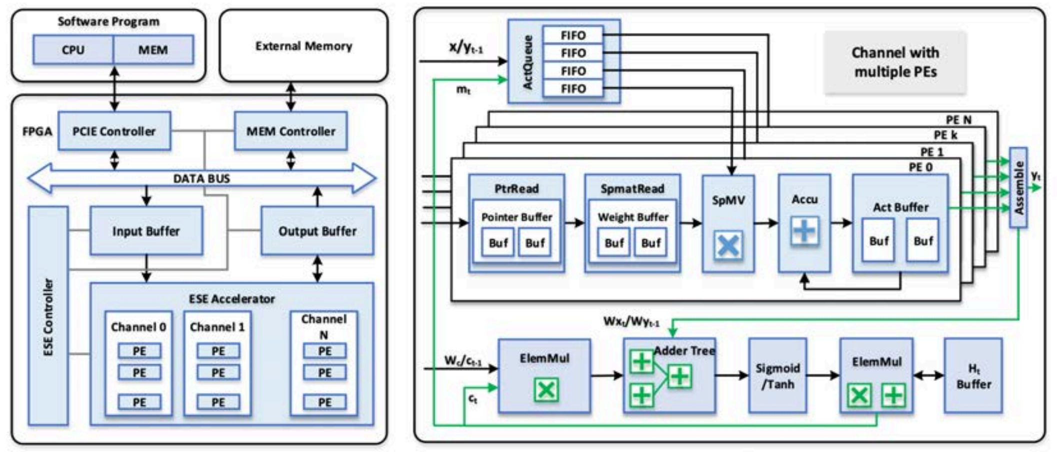
Souce: https://github.com/mit-han-lab/pruning-sparsity-publications

Pruning in the Industry

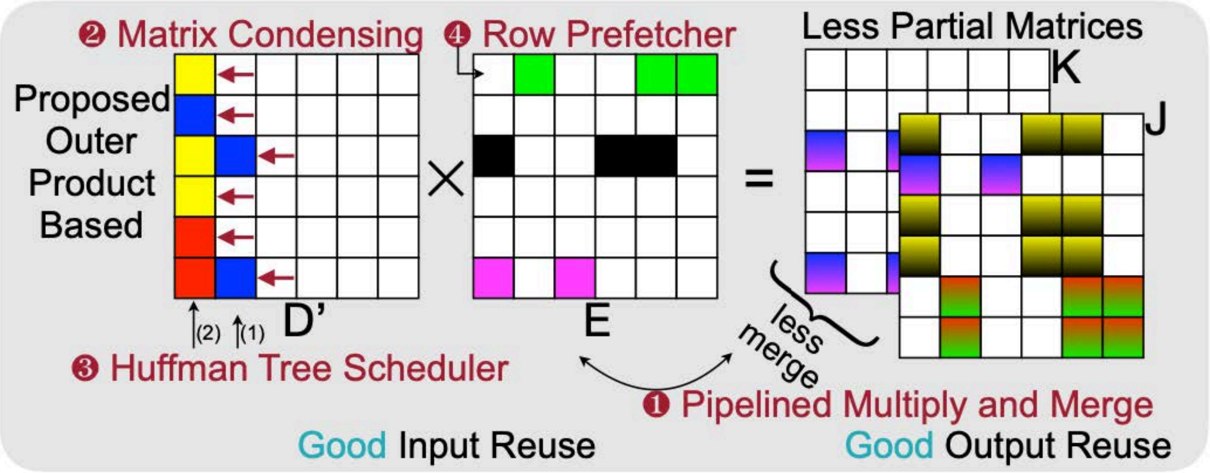
Hardware support for sparsity



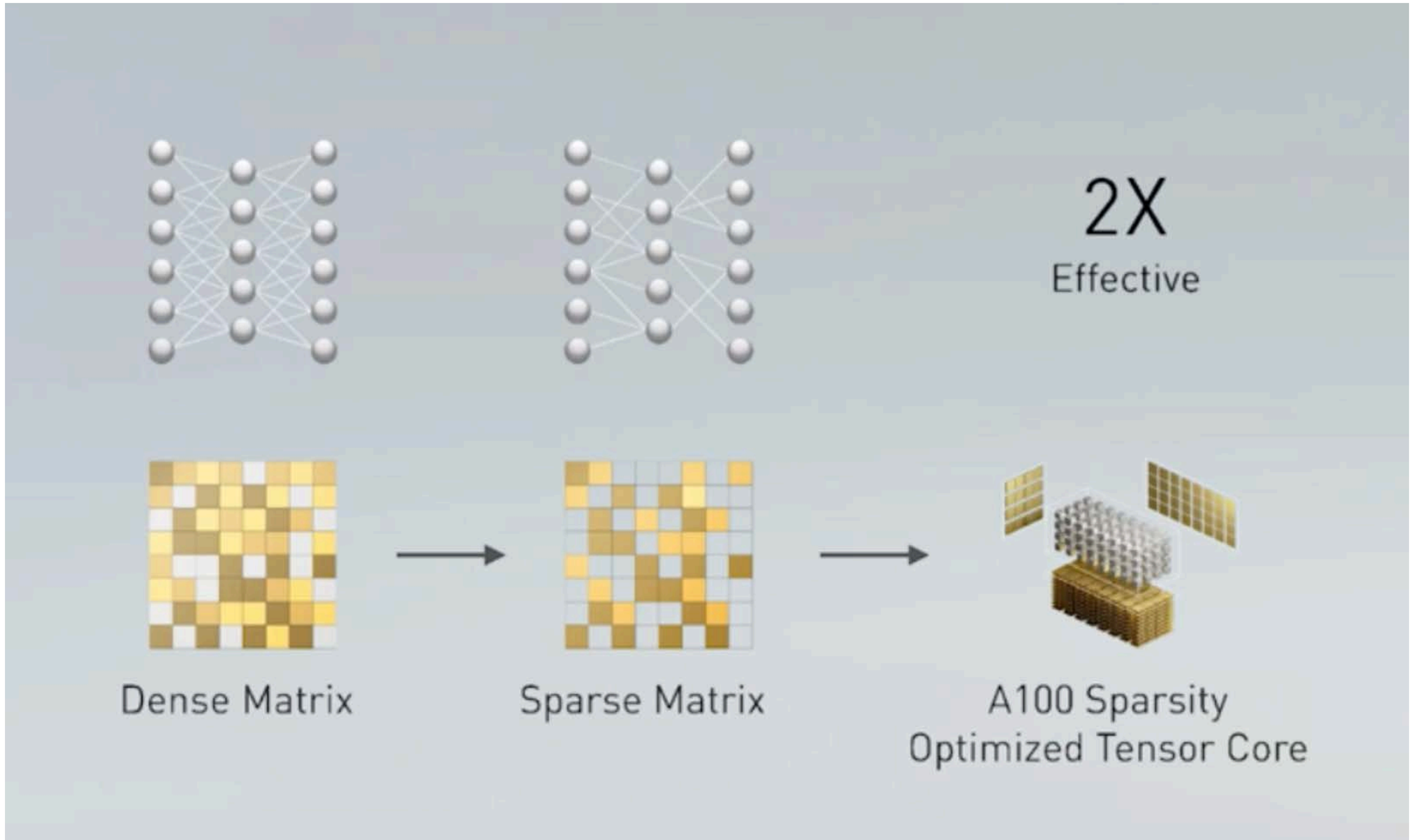
EIE [Han *et al.*, ISCA 2016]



ESE [Han *et al.*, FPGA 2017]



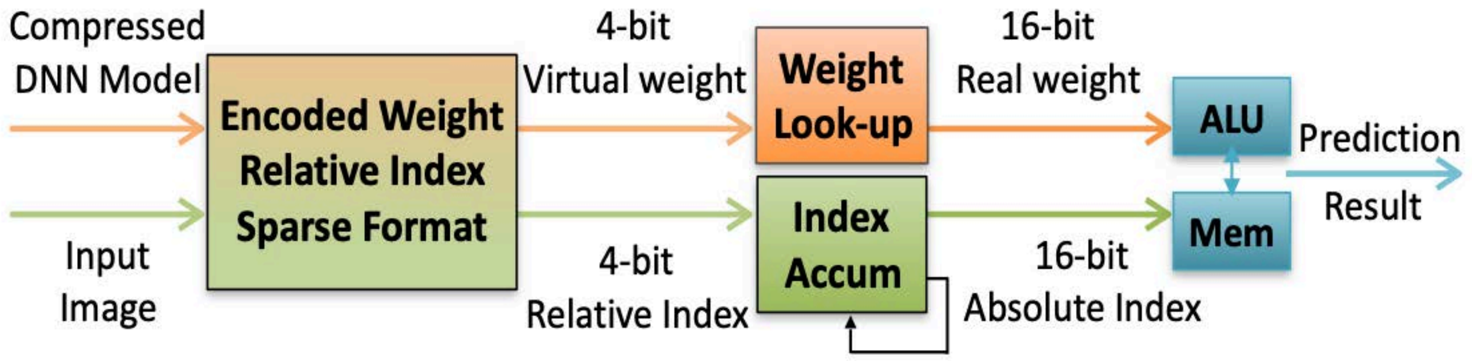
SpArch [Zhang *et al.*, HPCA 2020]
SpAtten [Wang *et al.*, HPCA 2021]



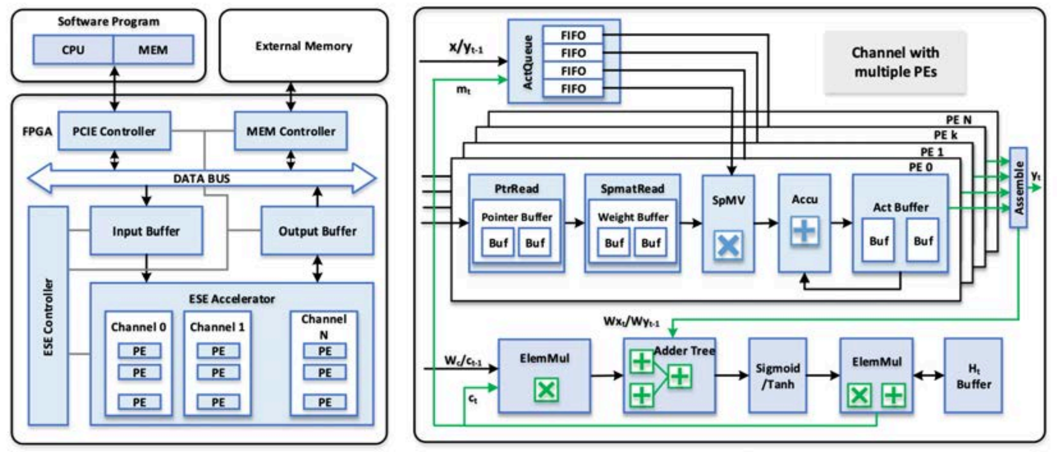
2:4 sparsity in A100 GPU
2X peak performance, 1.5X measured BERT speedup

Pruning in the Industry

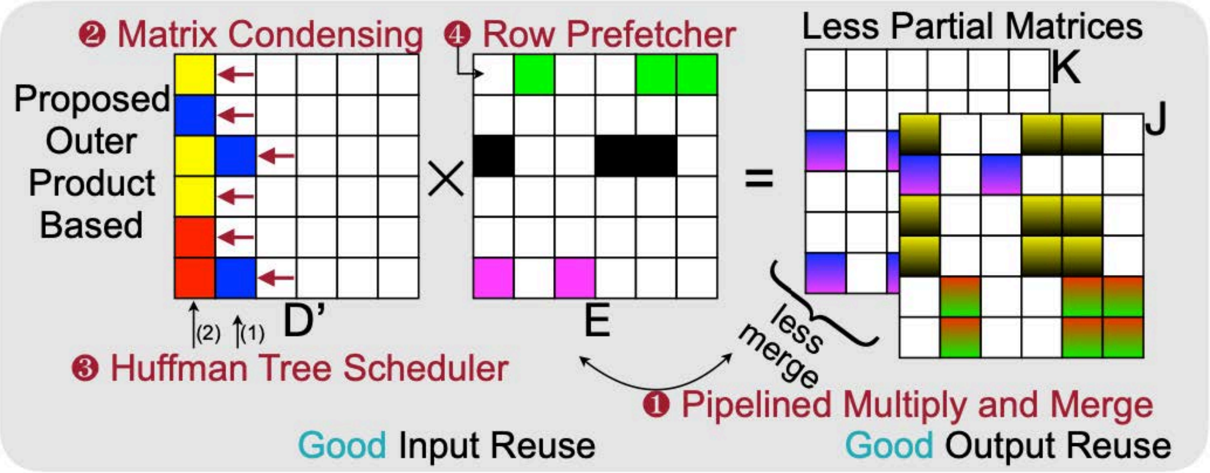
Hardware support for sparsity



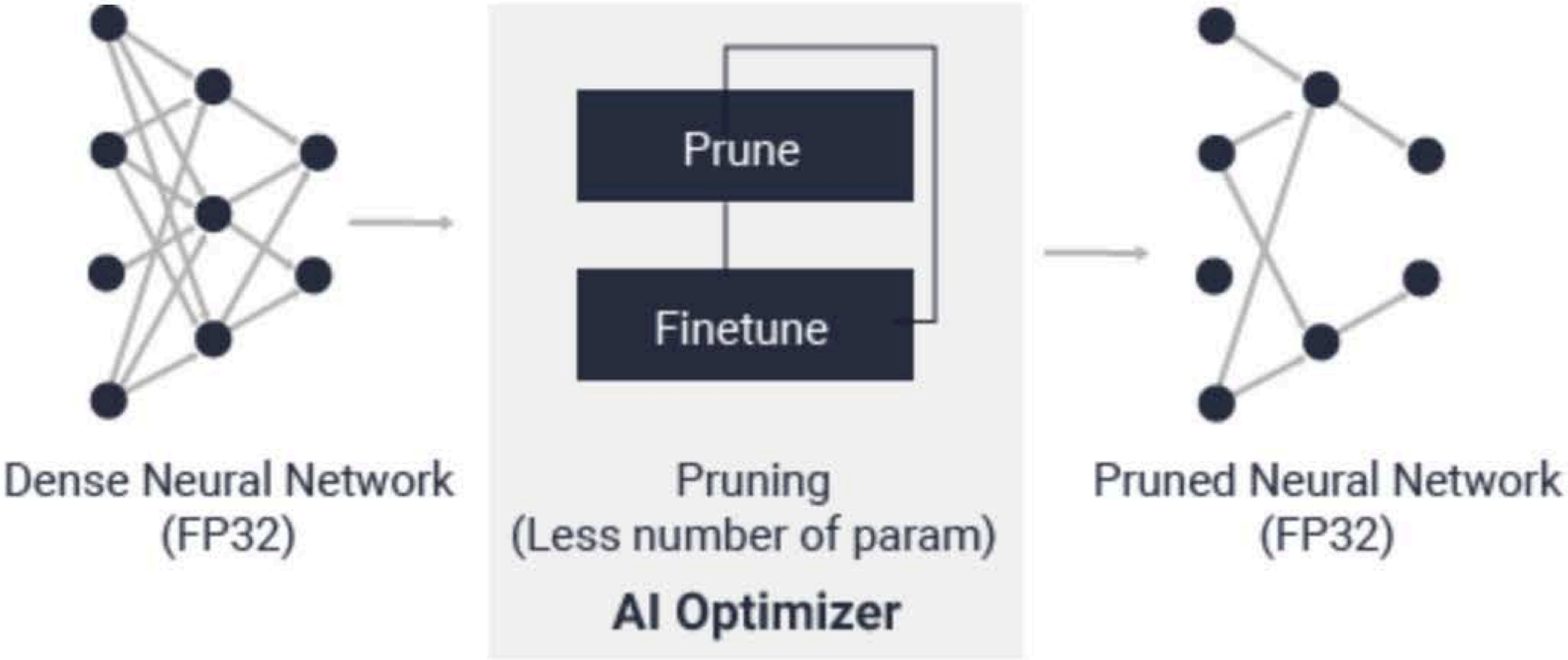
EIE [Han *et al.*, ISCA 2016]



ESE [Han *et al.*, FPGA 2017]



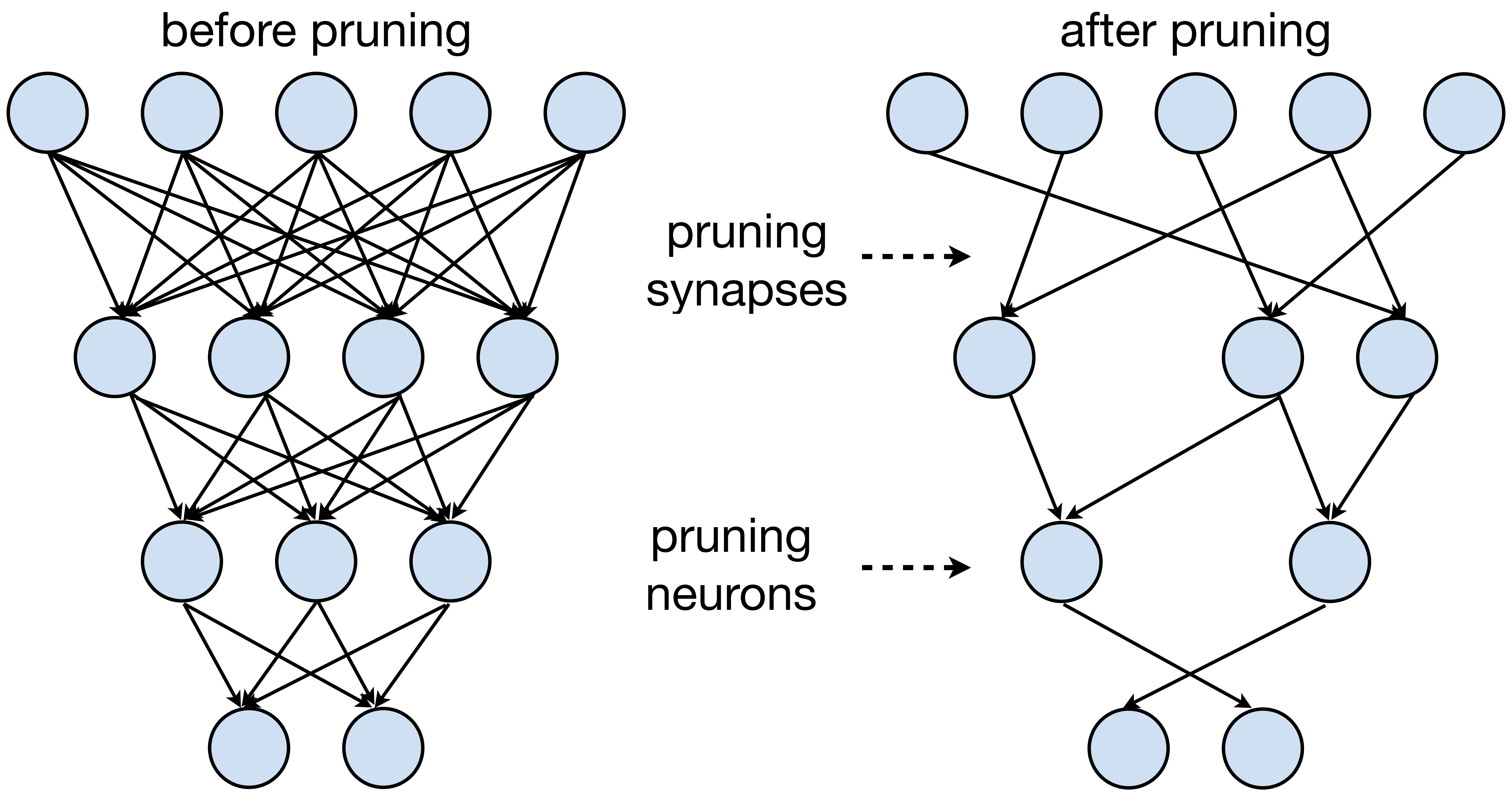
SpArch [Zhang *et al.*, HPCA 2020]
SpAtten [Wang *et al.*, HPCA 2021]



Reduce model complexity by 5x to 50x with minimal accuracy impact

Neural Network Pruning

- **Introduction to Pruning**
  - What is pruning?
  - How should we formulate pruning?
- **Determine the Pruning Granularity**
  - In what pattern should we prune the neural network?
- **Determine the Pruning Criterion**
  - What synapses/neurons should we prune?
- **Determine the Pruning Ratio**
  - What should target sparsity be for each layer?
- **Fine-tune/Train Pruned Neural Network**
  - How should we improve performance of pruned models?



Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Neural Network Pruning

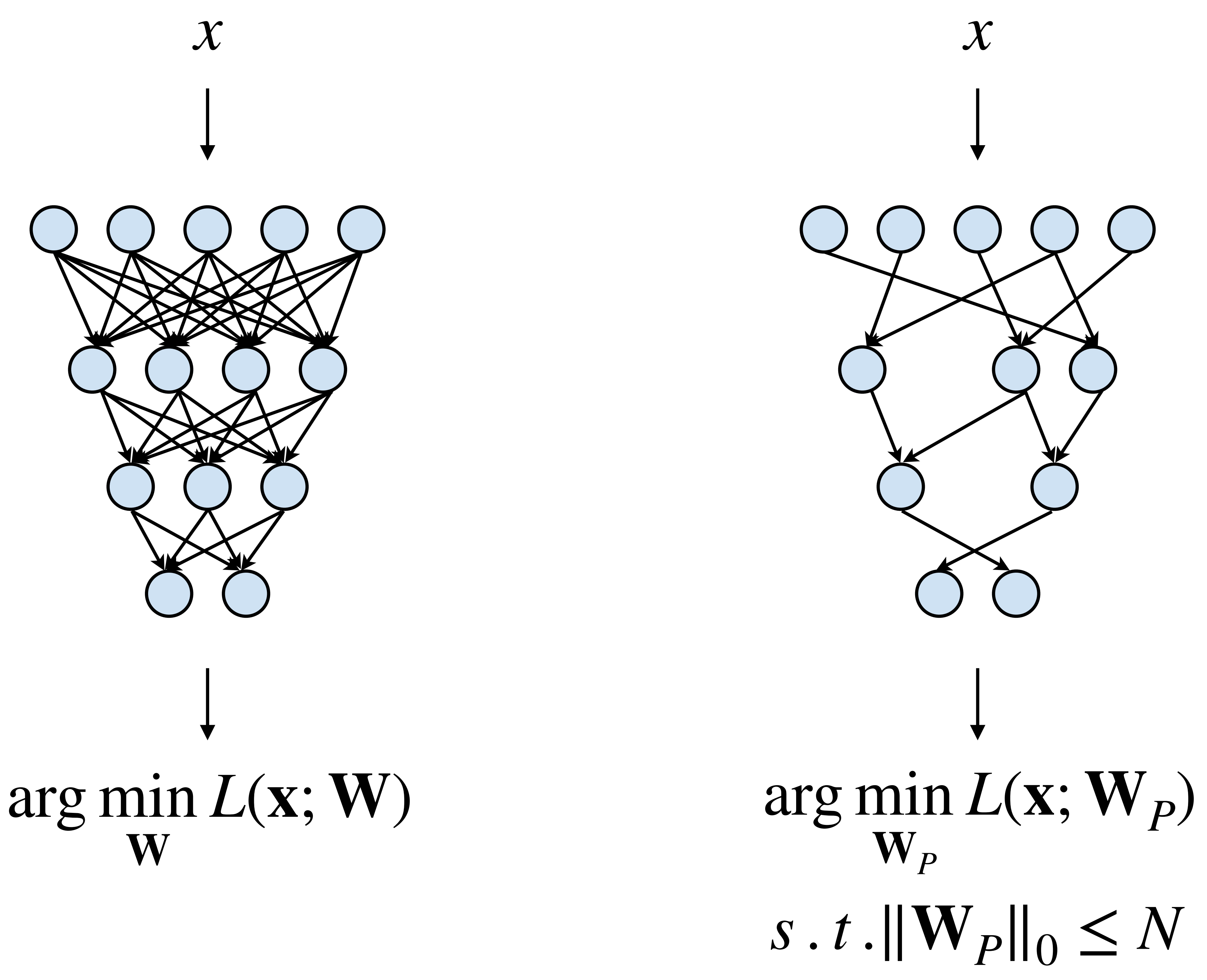
- In general, we could formulate the pruning as follows:

$$\arg\min_{\mathbf{W}_P} L(\mathbf{x}; \mathbf{W}_P)$$

subject to

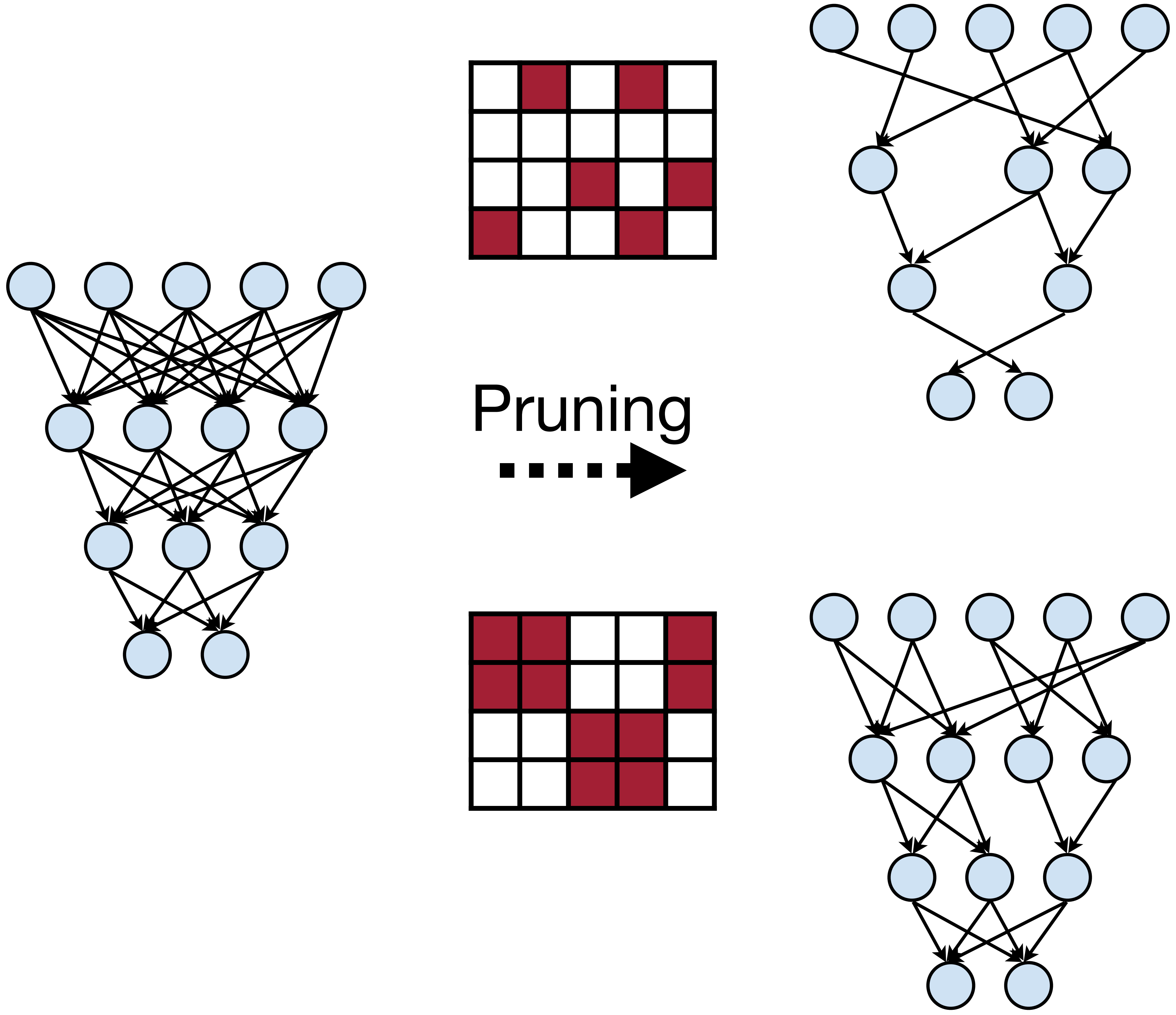
$$\|\mathbf{W}_p\|_0 < N$$

  - L represents the objective function for neural network training;
  - $\mathbf{x}$ is input, $\mathbf{W}$ is original weights, $\mathbf{W}_P$ is pruned weights;
  - $\|\mathbf{W}_p\|_0$ calculates the #nonzeros in W_P, and N is the target #nonzeros.



Neural Network Pruning

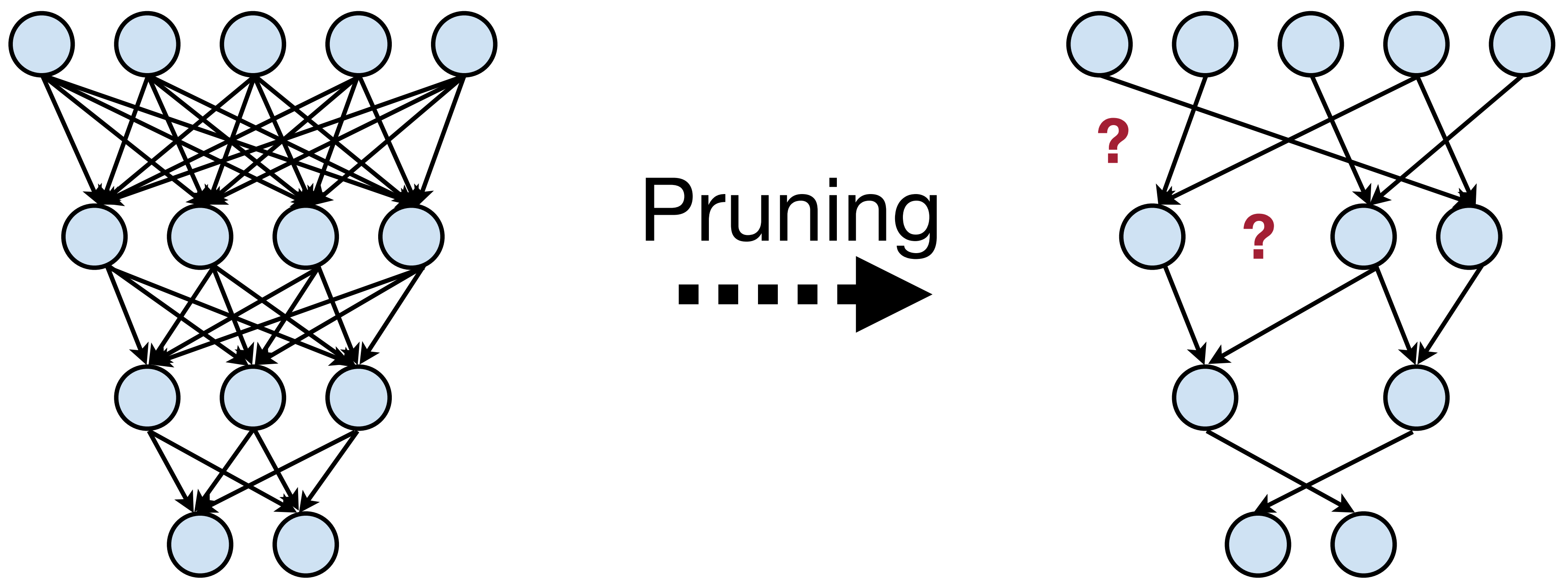
- **Introduction to Pruning**
  - What is pruning?
  - How should we formulate pruning?
- **Determine the Pruning Granularity**
  - In what pattern should we prune the neural network?
- **Determine the Pruning Criterion**
  - What synapses/neurons should we prune?
- **Determine the Pruning Ratio**
  - What should target sparsity be for each layer?
- **Fine-tune/Train Pruned Neural Network**
  - How should we improve performance of pruned models?



Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Neural Network Pruning

- **Introduction to Pruning**
  - What is pruning?
  - How should we formulate pruning?
- **Determine the Pruning Granularity**
  - In what pattern should we prune the neural network?
- **Determine the Pruning Criterion**
  - What synapses/neurons should we prune?
- **Determine the Pruning Ratio**
  - What should target sparsity be for each layer?
- **Fine-tune/Train Pruned Neural Network**
  - How should we improve performance of pruned models?

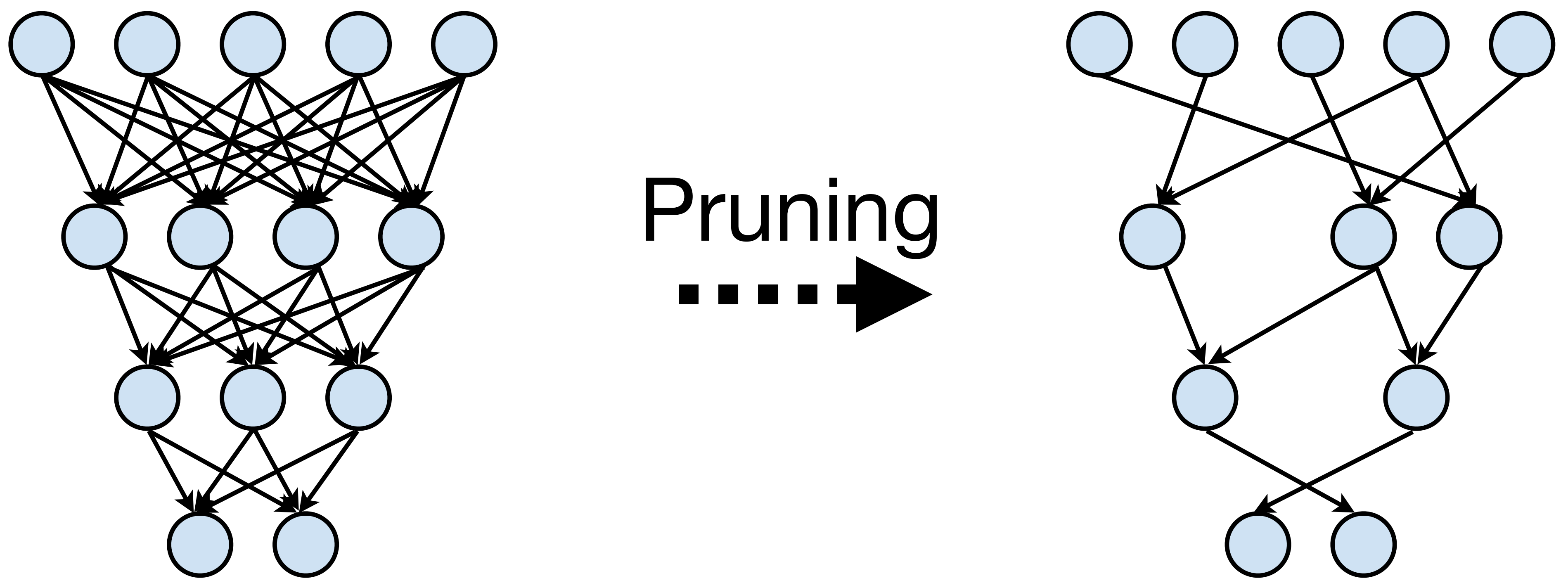


which synapses?
which neurons?

Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Neural Network Pruning

- **Introduction to Pruning**
  - What is pruning?
  - How should we formulate pruning?
- **Determine the Pruning Granularity**
  - In what pattern should we prune the neural network?
- **Determine the Pruning Criterion**
  - What synapses/neurons should we prune?
- **Determine the Pruning Ratio**
  - What should target sparsity be for each layer?
- **Fine-tune/Train Pruned Neural Network**
  - How should we improve performance of pruned models?



prune 30%?

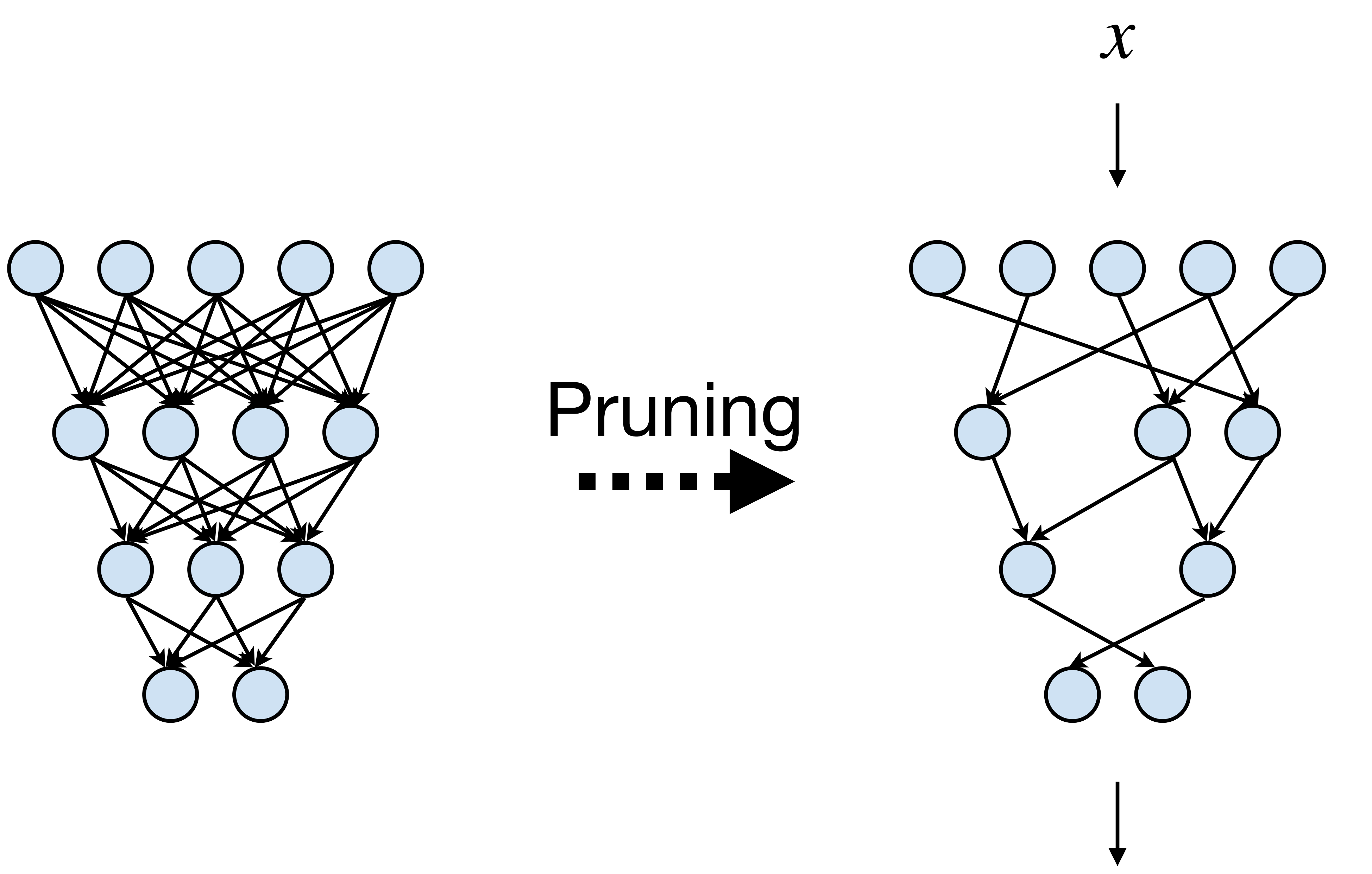
prune 50%?

prune 70%?

Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Neural Network Pruning

- **Introduction to Pruning**
  - What is pruning?
  - How should we formulate pruning?
- **Determine the Pruning Granularity**
  - In what pattern should we prune the neural network?
- **Determine the Pruning Criterion**
  - What synapses/neurons should we prune?
- **Determine the Pruning Ratio**
  - What should target sparsity be for each layer?
- **Fine-tune/Train Pruned Neural Network**
  - How should we improve performance of pruned models?



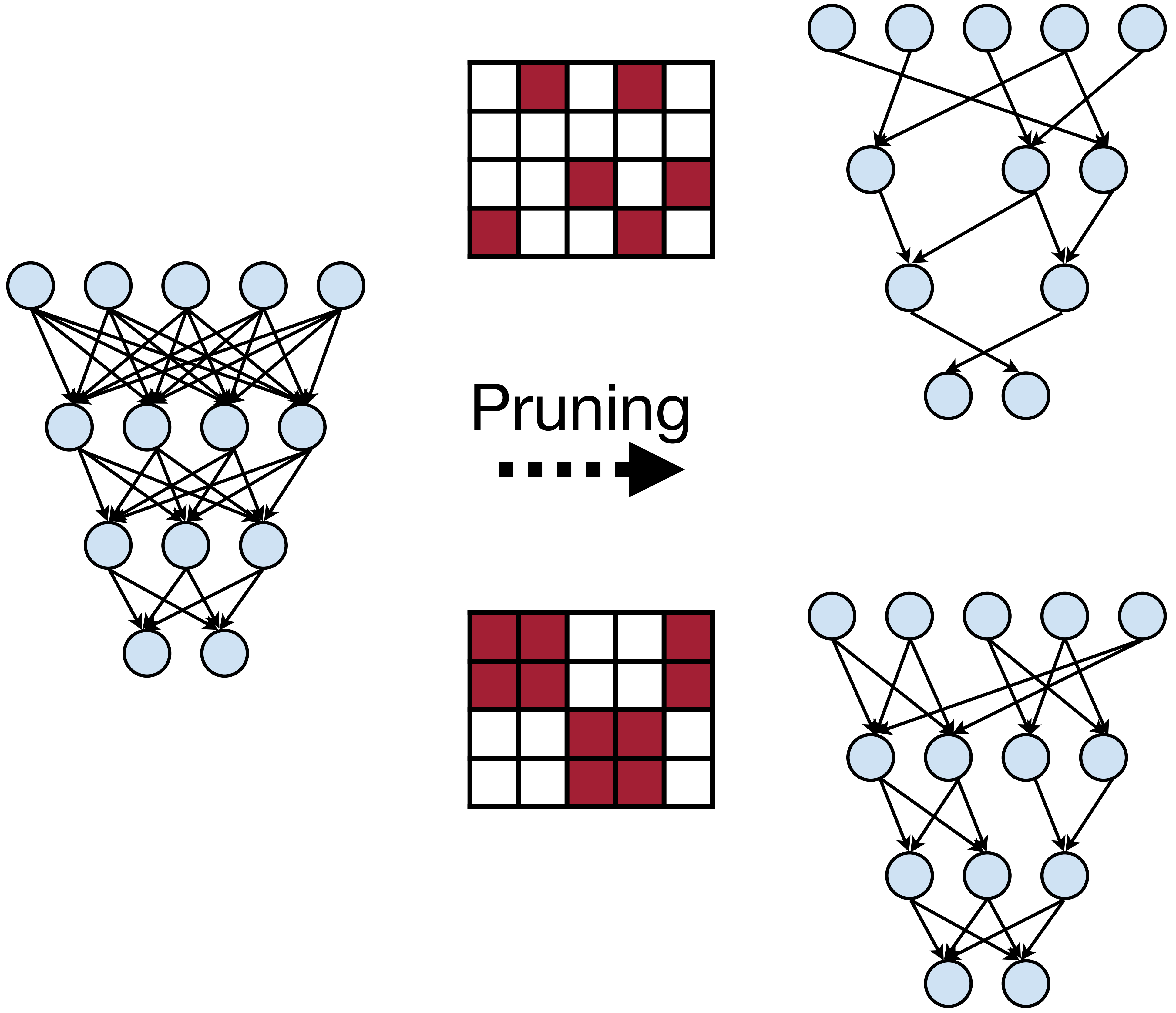
$$\arg\min_{\mathbf{W}_P} L(\mathbf{x}; \mathbf{W}_P)$$

$$s.t. \|\mathbf{W}_P\|_0 \leq N$$

Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Neural Network Pruning

- **Introduction to Pruning**
  - What is pruning?
  - How should we formulate pruning?
- **Determine the Pruning Granularity**
  - In what pattern should we prune the neural network?
- **Determine the Pruning Criterion**
  - What synapses/neurons should we prune?
- **Determine the Pruning Ratio**
  - What should target sparsity be for each layer?
- **Fine-tune/Train Pruned Neural Network**
  - How should we improve performance of pruned models?



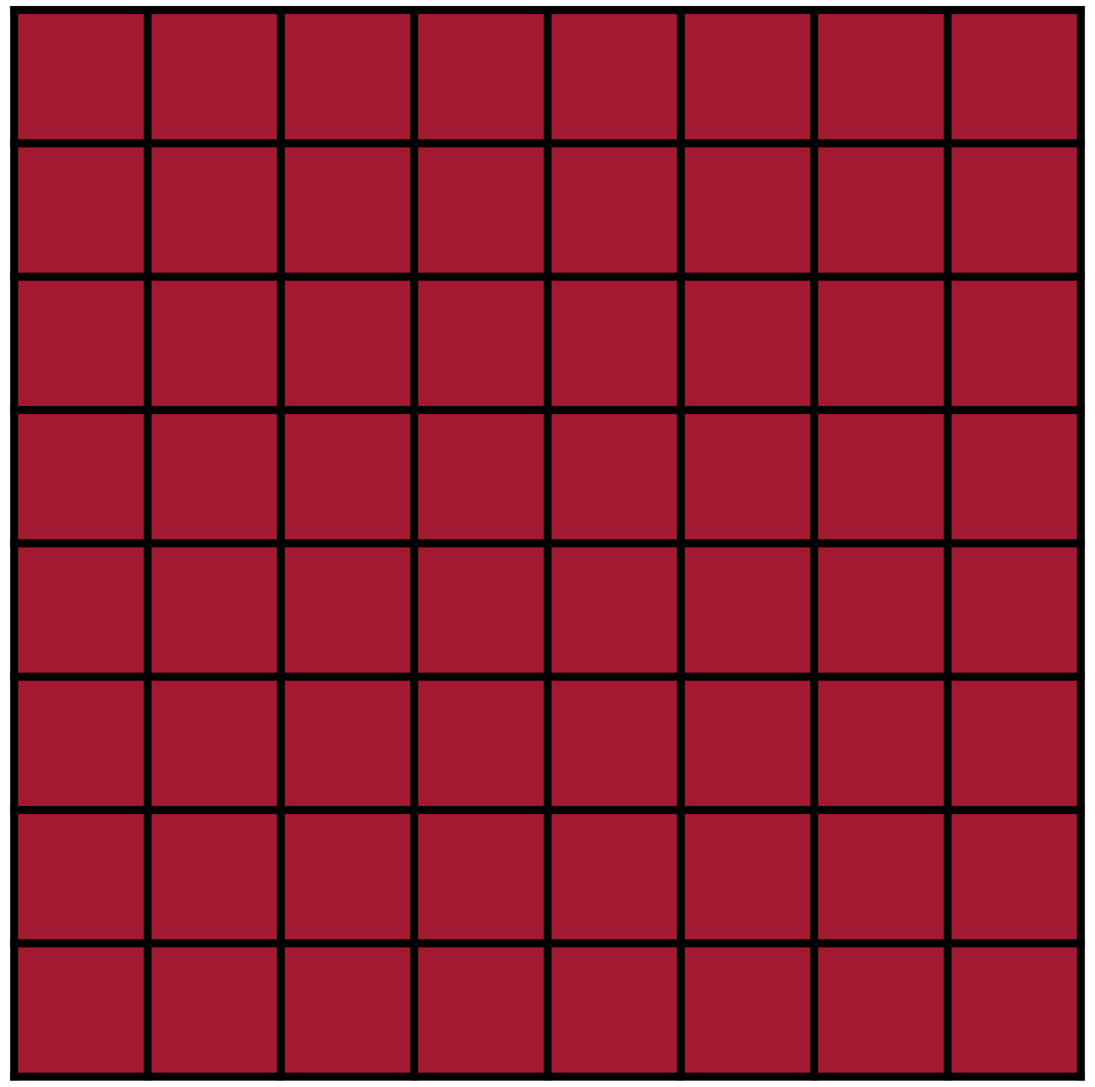
Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Section 2: Pruning Granularity

Pruning can be performed at different granularities, from structured to non-structured.

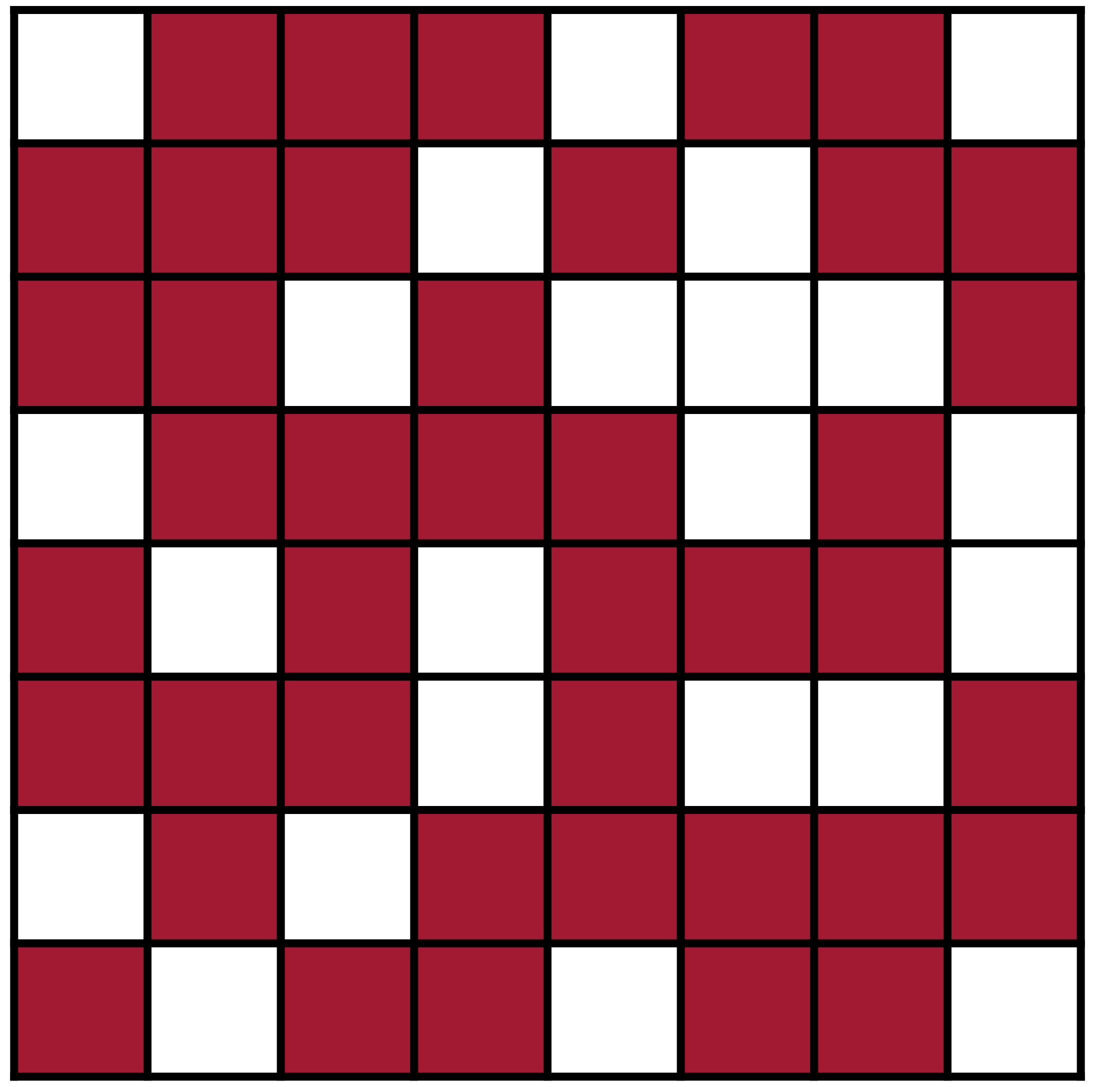
Pruning at Different Granularities

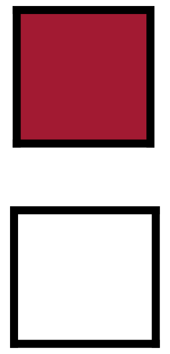
A simple example of 2D weight matrix



Pruning at Different Granularities

A simple example of 2D weight matrix



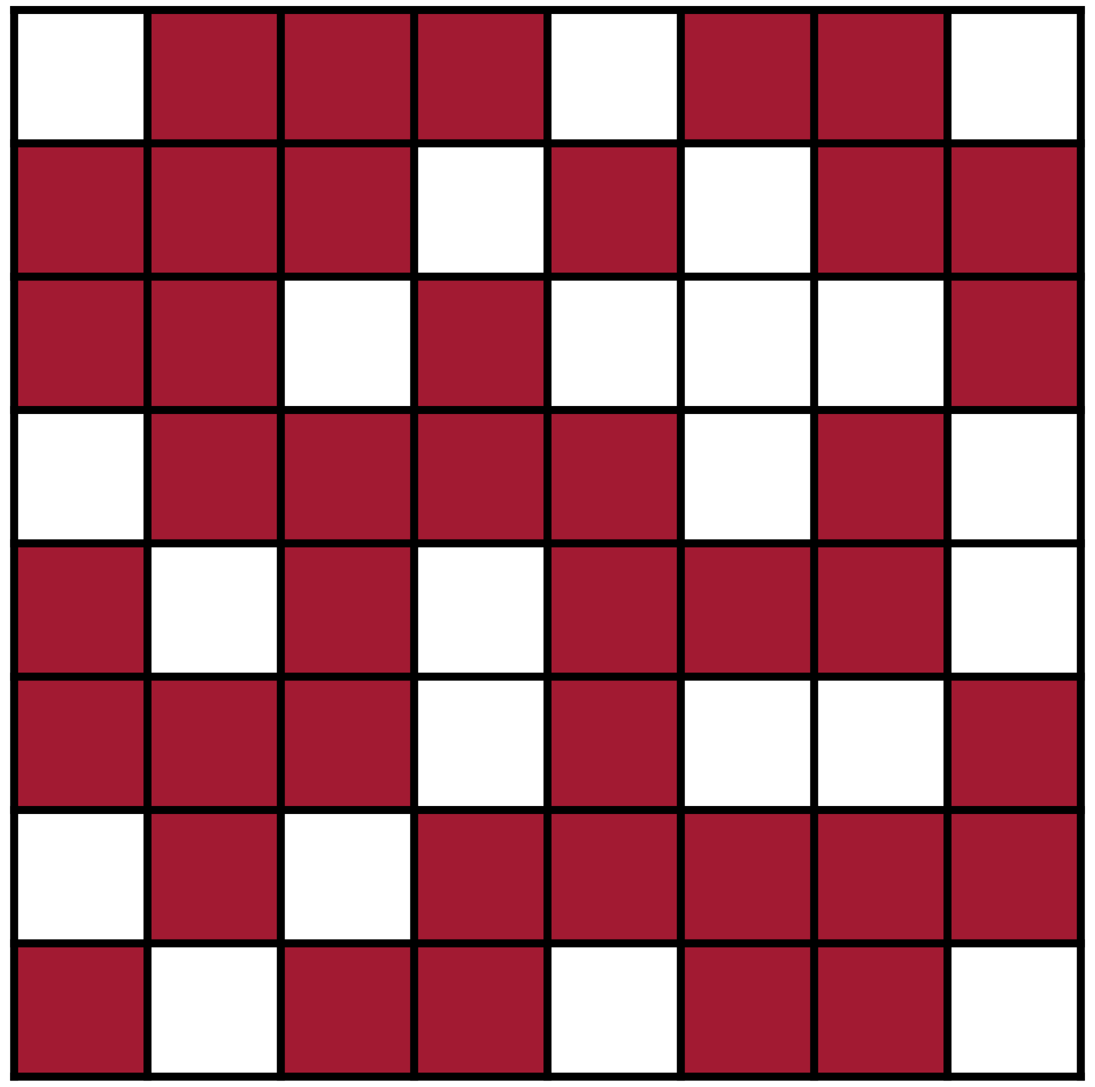


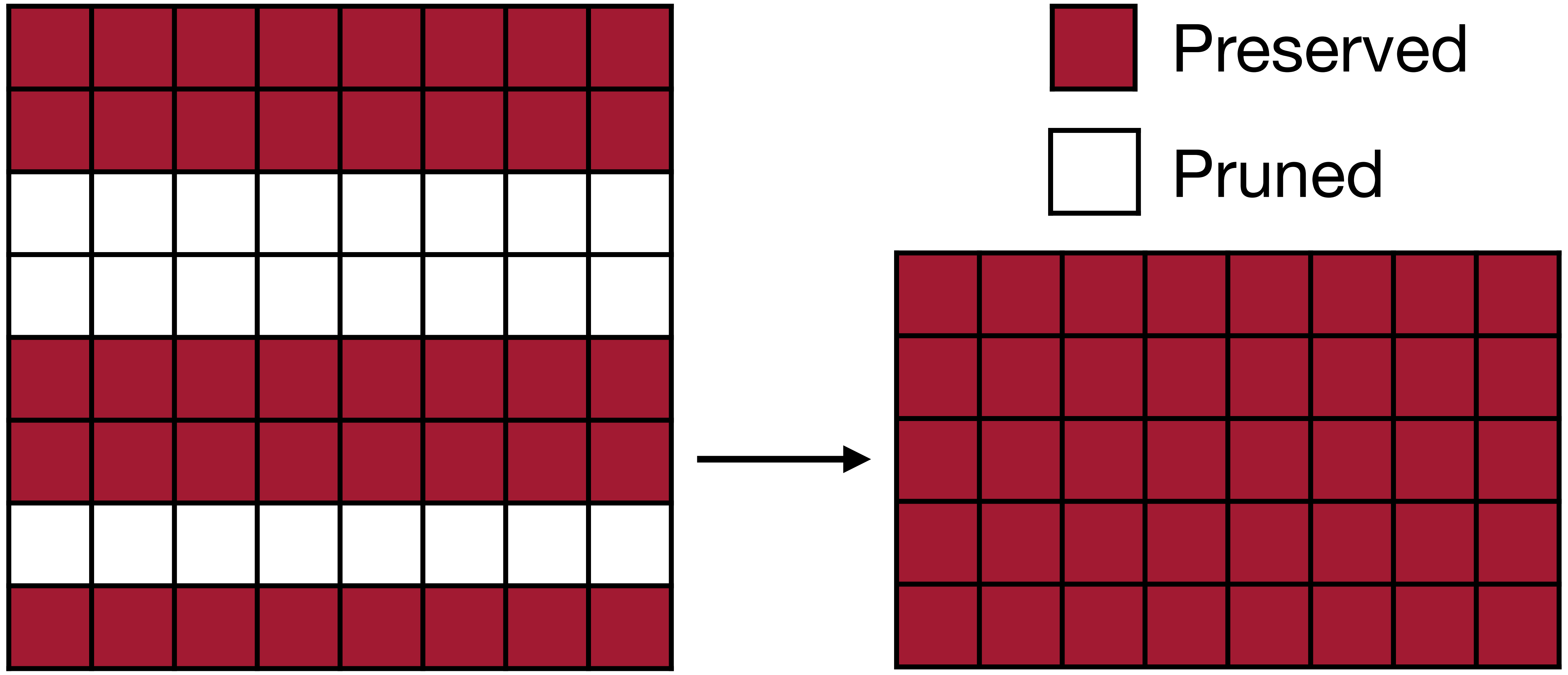
Fine-grained/Unstructured

- More flexible pruning index choice
- Hard to accelerate (irregular)

Pruning at Different Granularities

A simple example of 2D weight matrix





Fine-grained/Unstructured

- More flexible pruning index choice
- Hard to accelerate (irregular)

Coarse-grained/Structured

- Less flexible pruning index choice (a subset of the fine-grained case)
- Easy to accelerate (just a smaller matrix!)

Pruning at Different Granularities

The case of convolutional layers

- The weights of convolutional layers have 4 dimensions $[c_o, c_i, k_h, k_w]$:
  - c_i: input channels (or channels)
  - c_o: output channels (or filters)
  - k_h: kernel size height
  - k_w: kernel size width

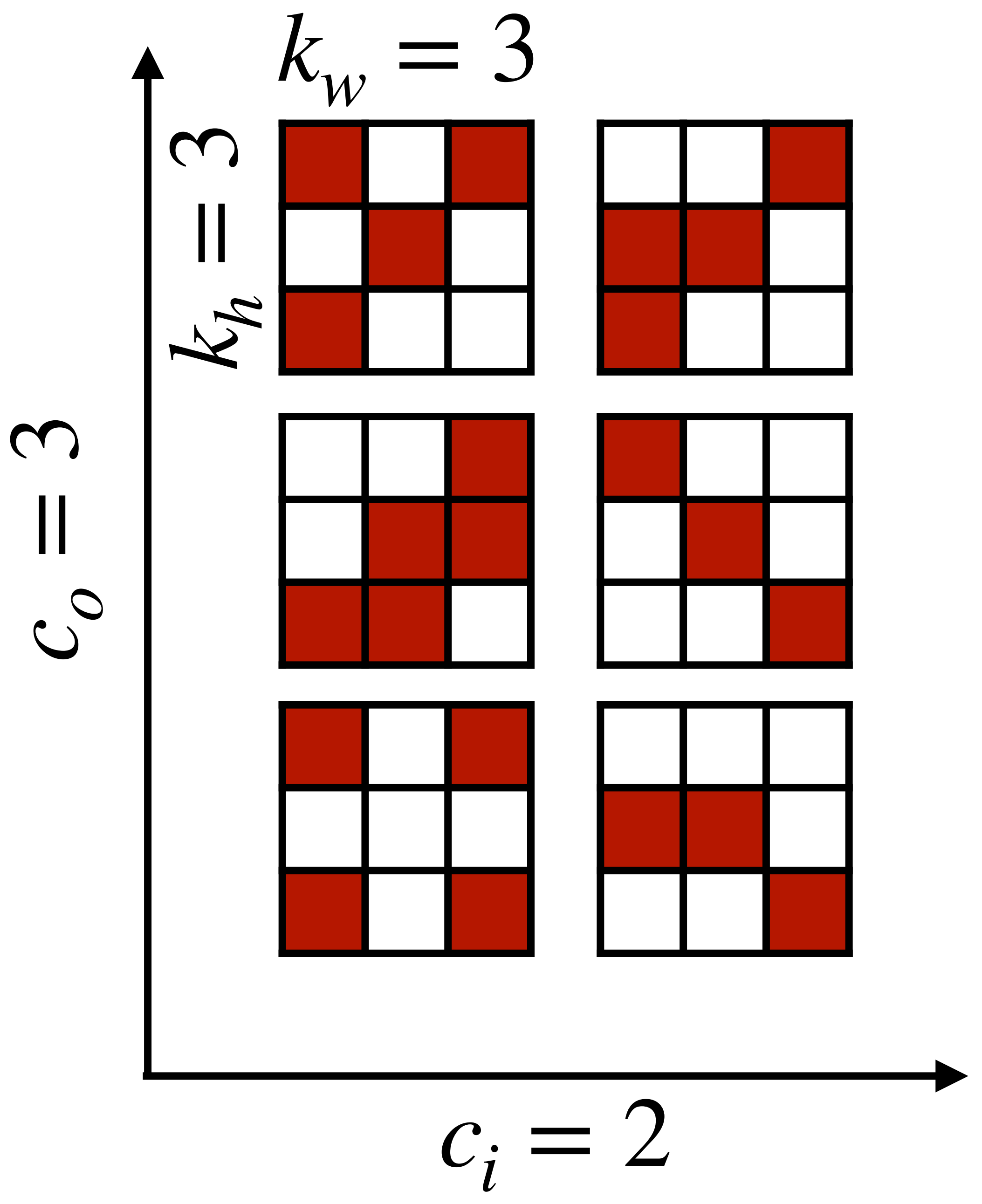
- The 4 dimensions give us more choices to select pruning granularities

Pruning at Different Granularities

The case of convolutional layers

- Some of the commonly used pruning granularities

Notations

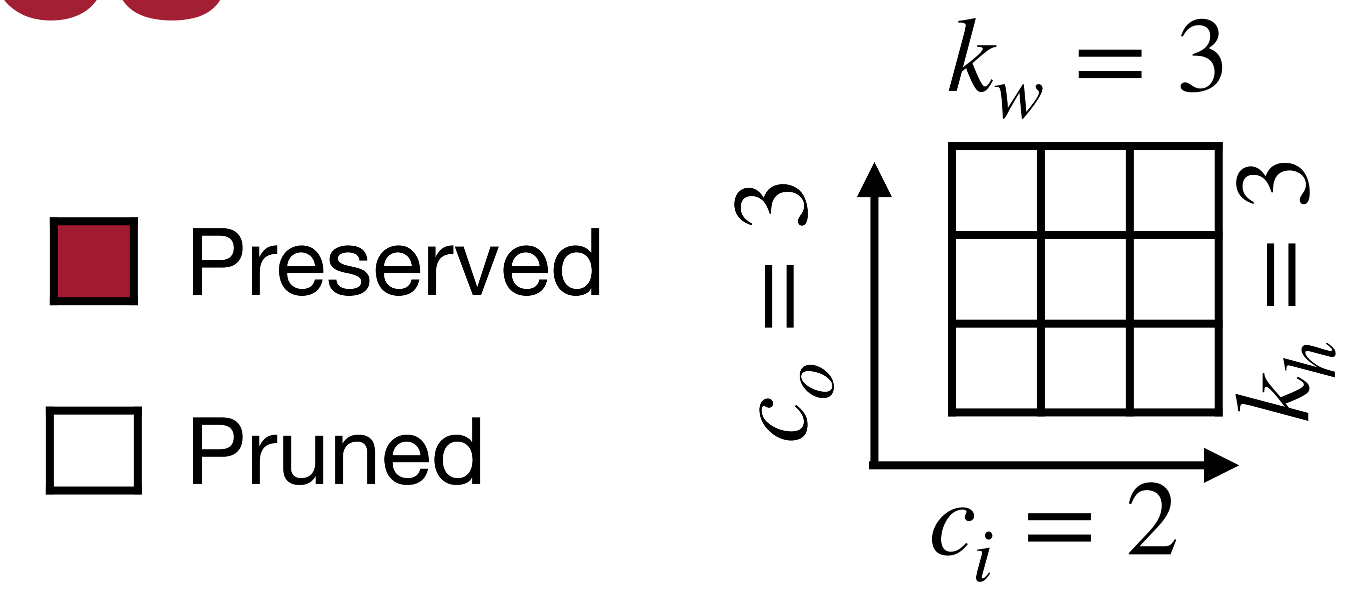


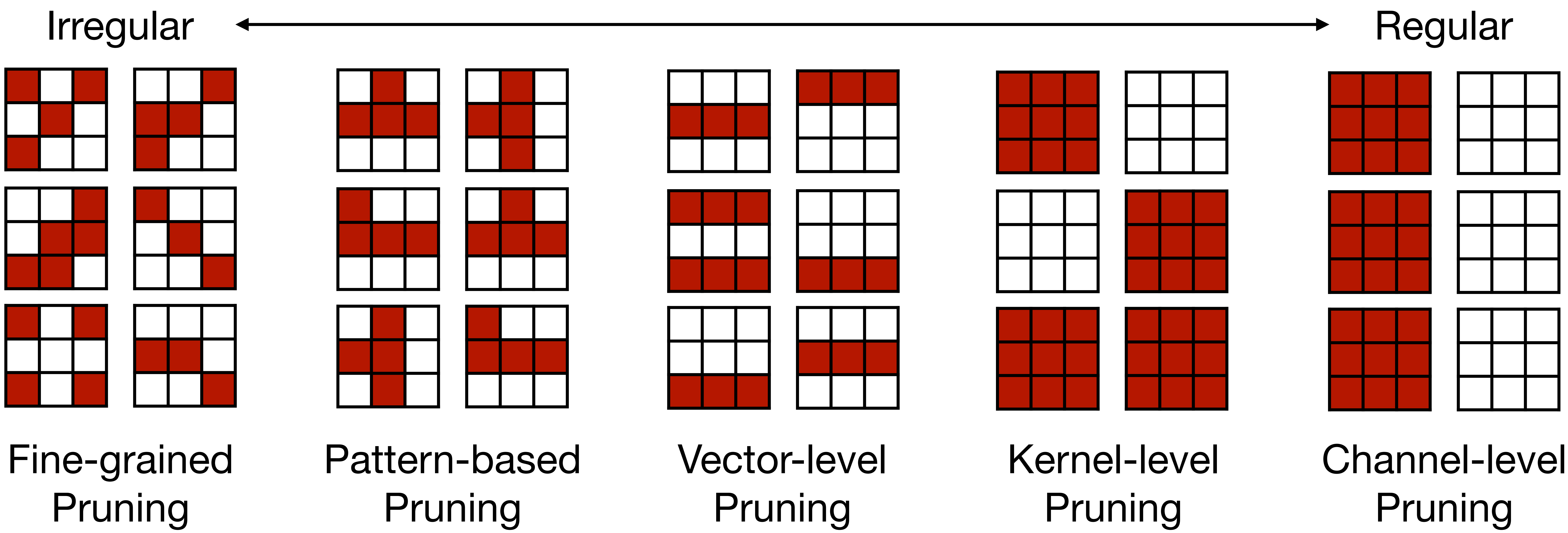
Exploring the granularity of sparsity in convolutional neural networks [Mao *et al.*, CVPR-W]

Pruning at Different Granularities

The case of convolutional layers

- Some of the commonly used pruning granularities





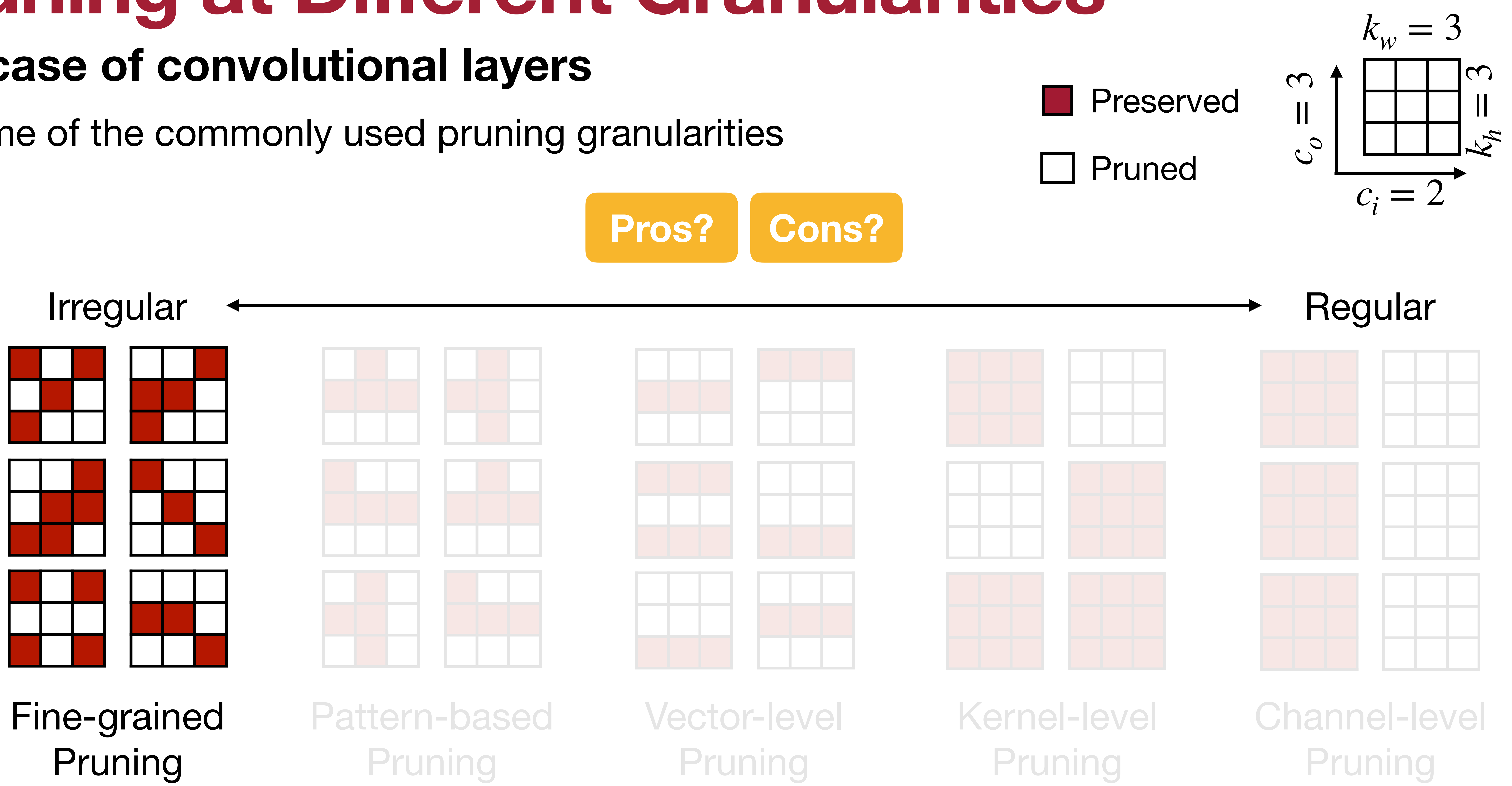
like Tetris :)

Exploring the granularity of sparsity in convolutional neural networks [Mao *et al.*, CVPR-W]

Pruning at Different Granularities

The case of convolutional layers

- Some of the commonly used pruning granularities

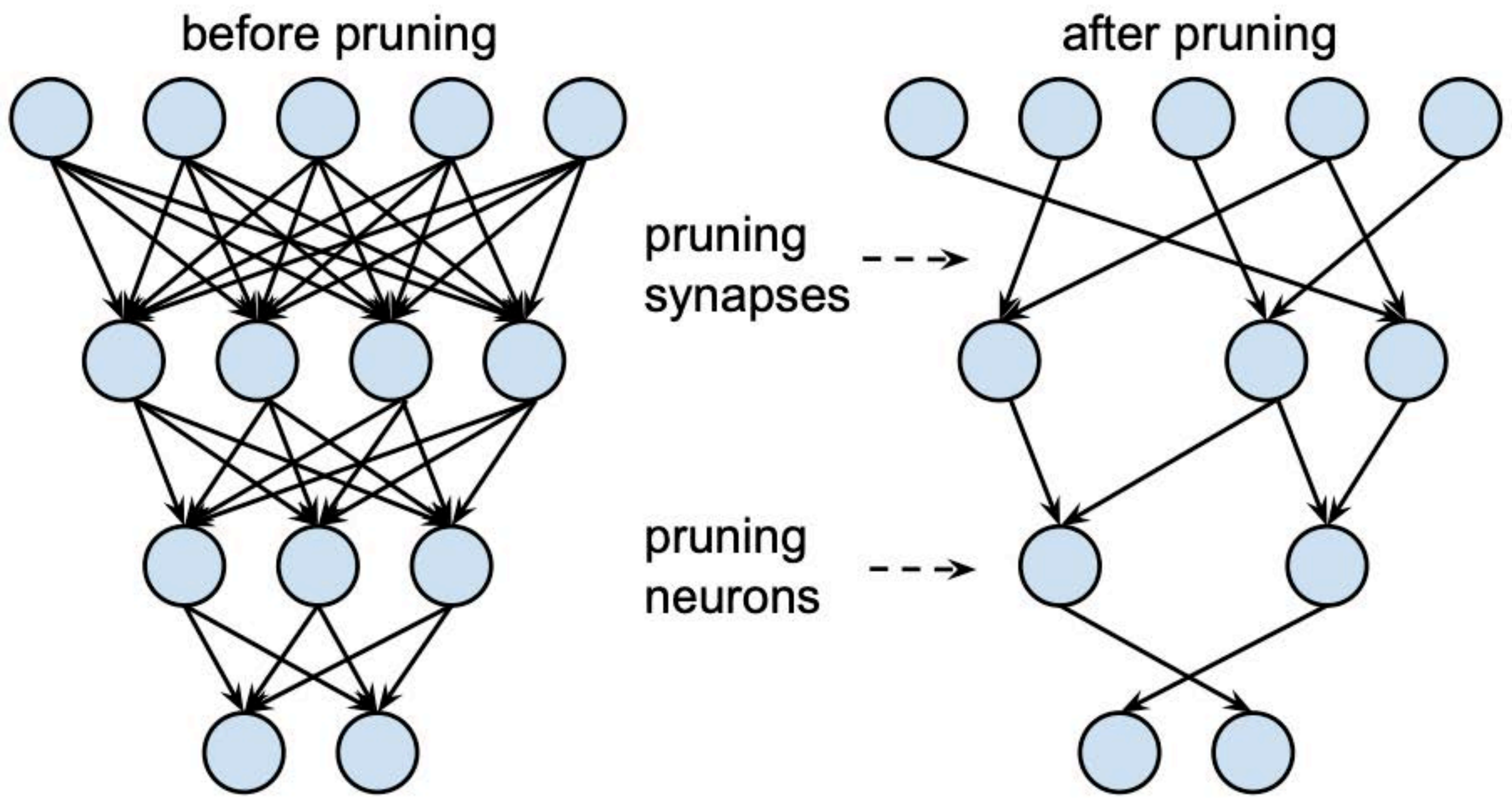


Exploring the granularity of sparsity in convolutional neural networks [Mao *et al.*, CVPR-W]

Pruning at Different Granularities

Let’s look into some cases

- **Fine-grained Pruning** (the case we show before)
  - Flexible pruning indices



Learning Both Weights and Connections for Efficient Neural Network [Han *et al*., NeurIPS 2015]

Pruning at Different Granularities

Let's look into some cases

- **Fine-grained Pruning** (the case we show before)
  - Flexible pruning indices
  - Usually larger compression ratio since we can flexibly find "redundant" weights (we will later discuss how we find them)

Neural Network	#Parameters		
	Before Pruning	After Pruning	Reduction
AlexNet	61 M	6.7 M	**9** ×
VGG-16	138 M	10.3 M	**12** ×
GoogleNet	7 M	2.0 M	**3.5** ×
ResNet50	26 M	7.47 M	**3.4** ×

Efficient Methods and Hardware for Deep Learning [Han S., Stanford University]

Pruning at Different Granularities

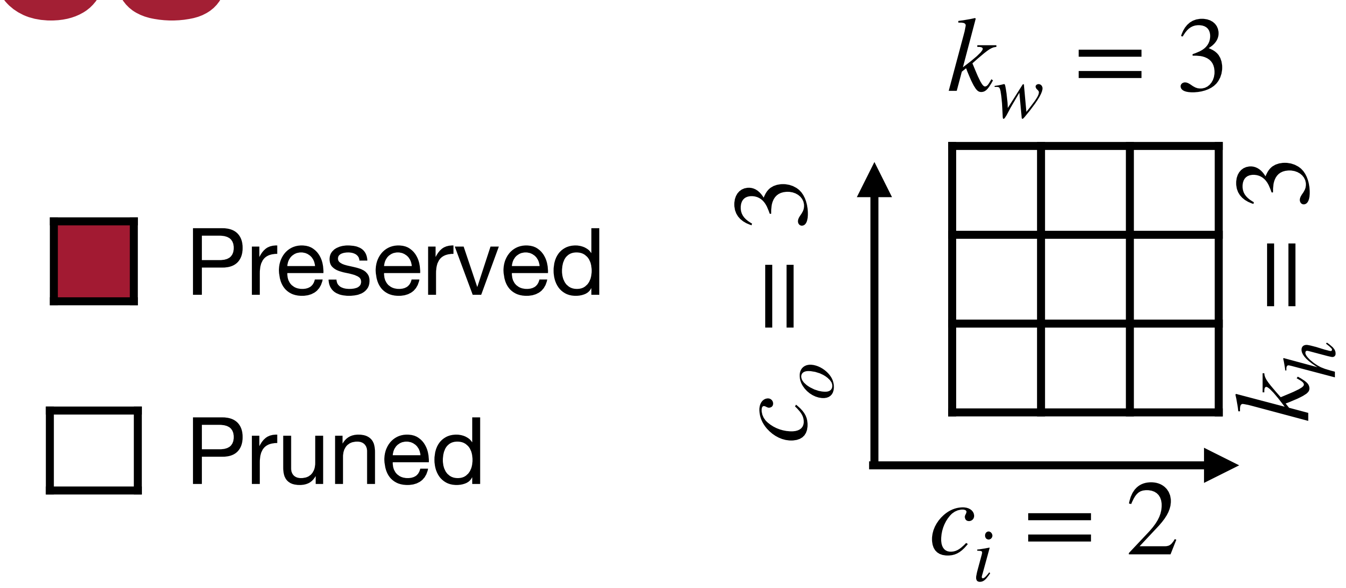
Let's look into some cases

- **Fine-grained Pruning** (the case we show before)
  - Flexible pruning indices
  - Usually larger compression ratio since we can flexibly find "redundant" weights (we will later discuss how we find them)
  - Can deliver speed up on some custom hardware (e.g., EIE) but not GPU (easily)

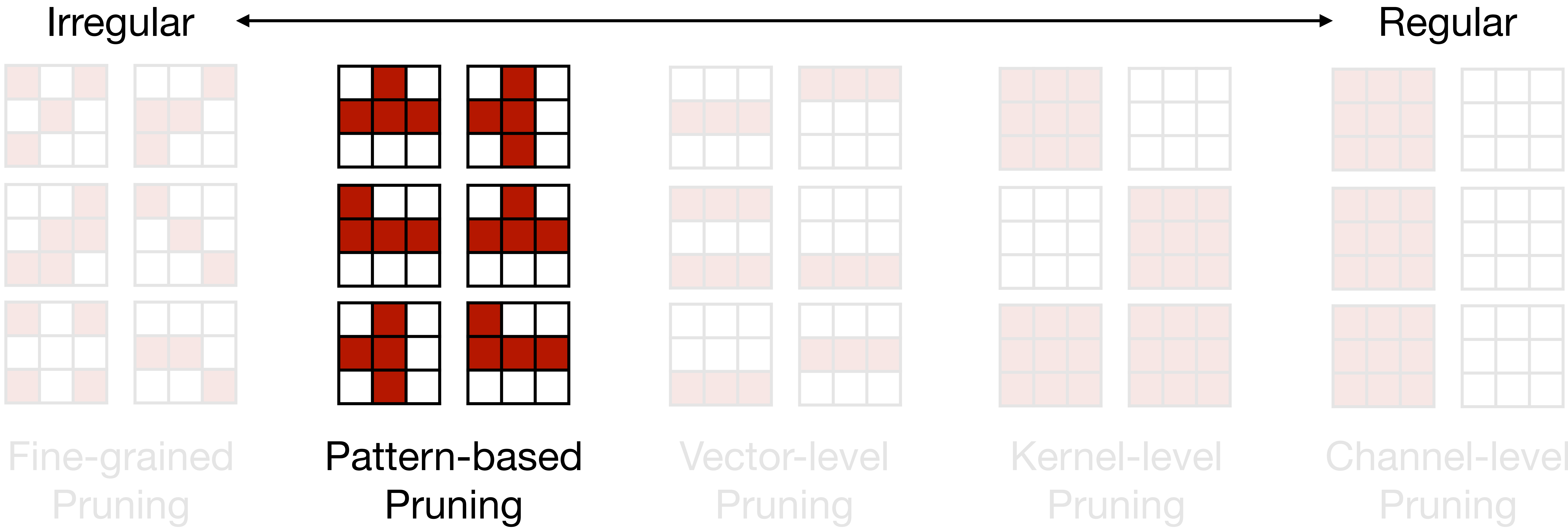
Pruning at Different Granularities

The case of convolutional layers

- Some of the commonly used pruning granularities



Pros? Cons?

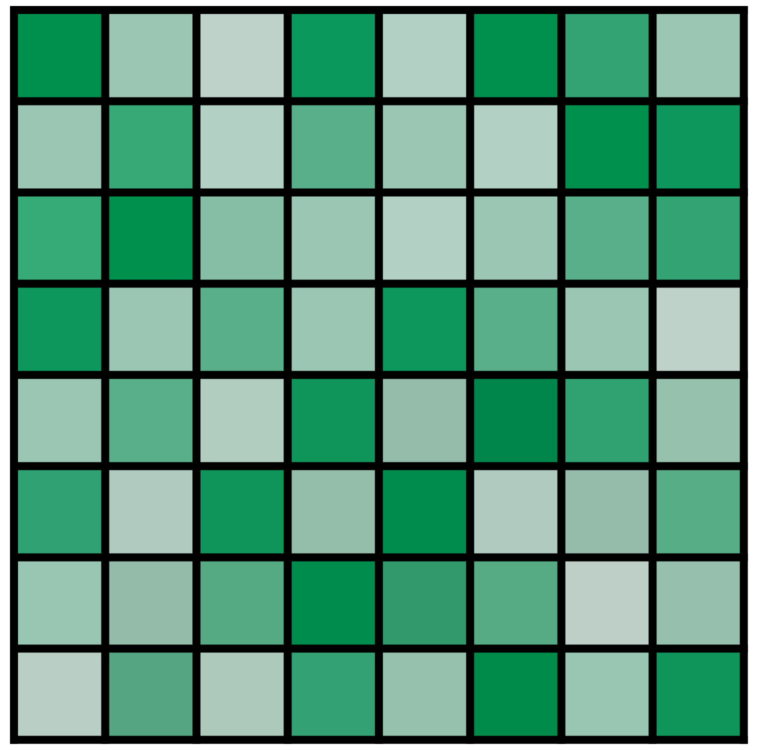


Exploring the granularity of sparsity in convolutional neural networks [Mao *et al.*, CVPR-W]

Pruning at Different Granularities

Let’s look into some cases

- **Pattern-based Pruning: N:M sparsity**
  - N:M sparsity means that in each contiguous M elements, N of them is pruned



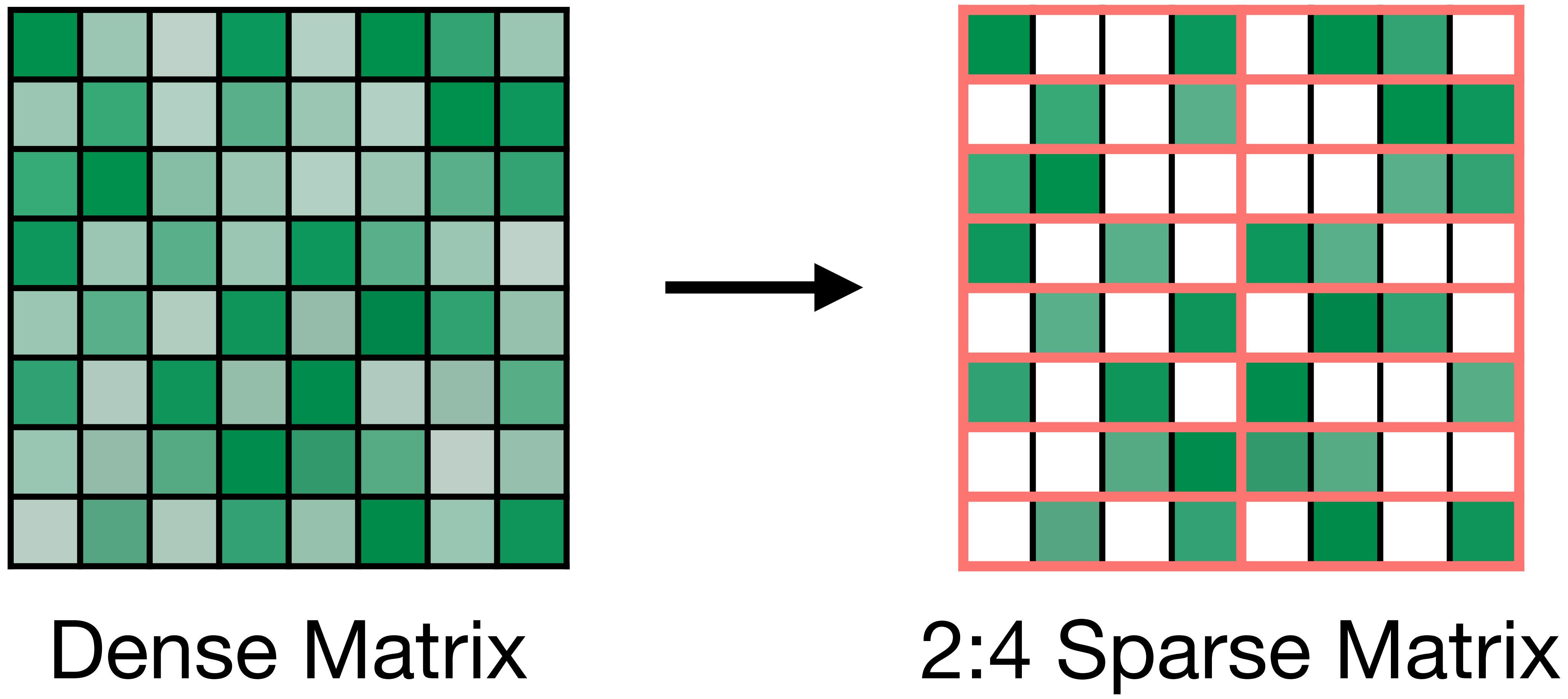
Dense Matrix

Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT

Pruning at Different Granularities

Let's look into some cases

- **Pattern-based Pruning: N:M sparsity**
  - N:M sparsity means that in each contiguous M elements, N of them is pruned
  - A classic case is 2:4 sparsity (50% sparsity)

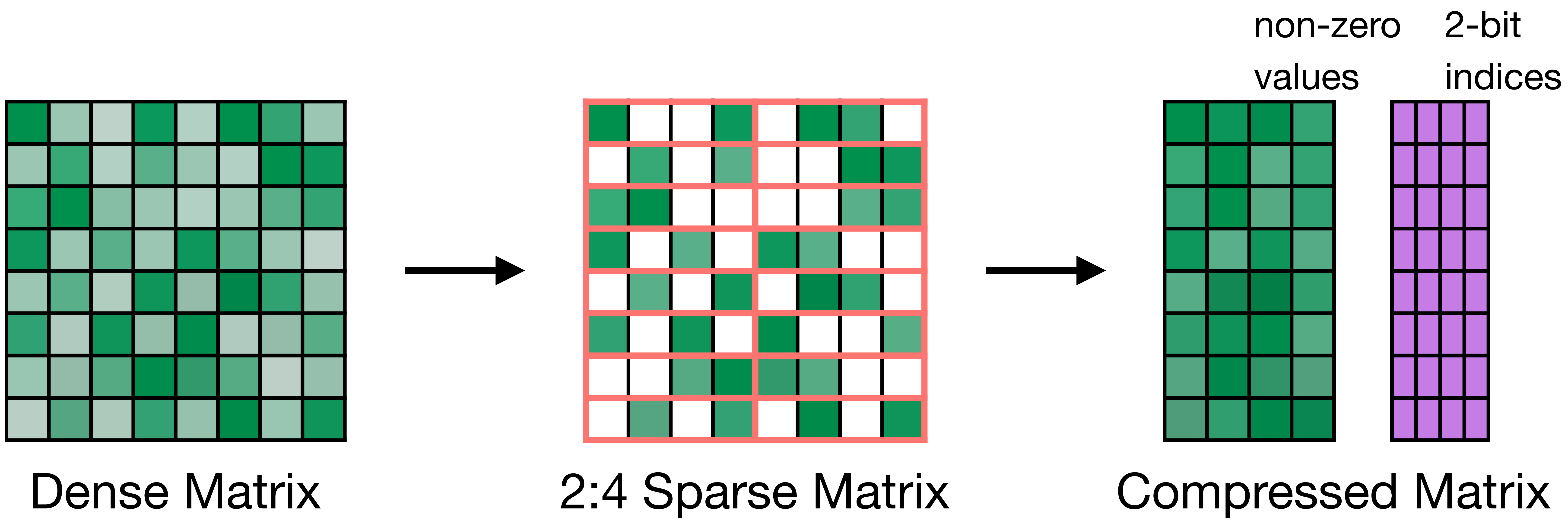


Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT

Pruning at Different Granularities

Let's look into some cases

- **Pattern-based Pruning: N:M sparsity**
  - N:M sparsity means that in each contiguous M elements, N of them is pruned
  - A classic case is 2:4 sparsity (50% sparsity)
  - It is supported by NVIDIA's Ampere GPU Architecture, which delivers up to 2x speed up



Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT

Pruning at Different Granularities

Let's look into some cases

- **Pattern-based Pruning: N:M sparsity**
  - N:M sparsity means that in each contiguous M elements, N of them is pruned
  - A classic case is 2:4 sparsity (50% sparsity)
  - It is supported by NVIDIA's Ampere GPU Architecture, which delivers ~2x speed up
  - Usually maintains accuracy (tested on varieties of tasks)

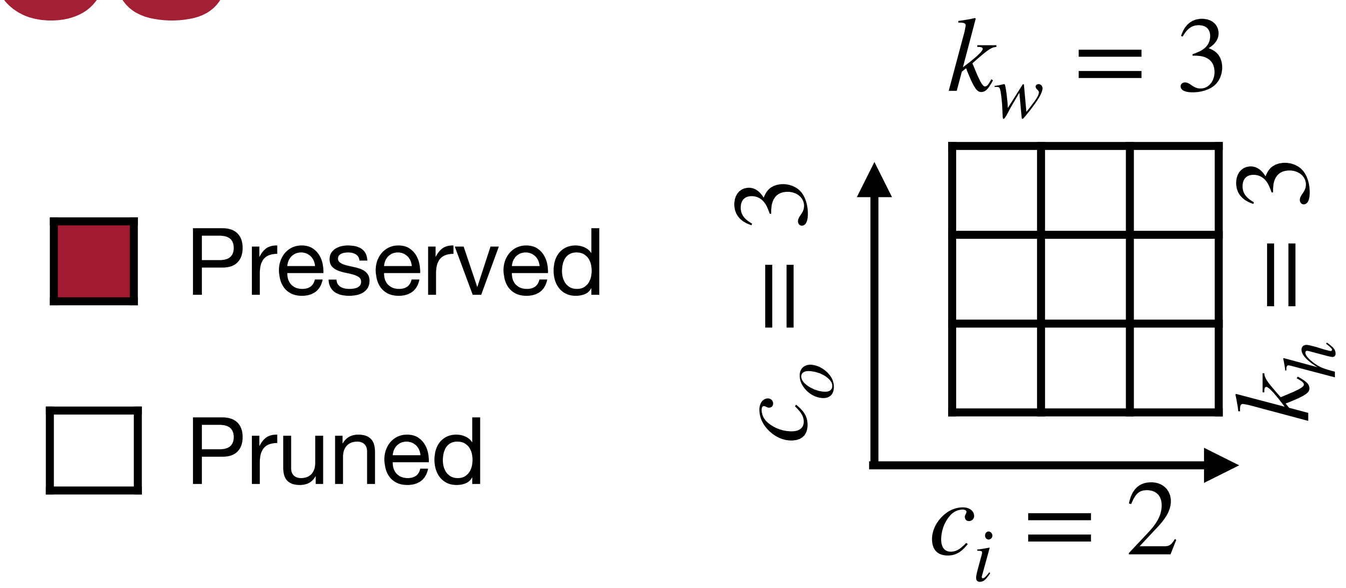
Network	Data Set	Metric	Dense FP16	Sparse FP16
ResNet-50	ImageNet	Top-1	76.1	76.2
ResNeXt-101_32x8d	ImageNet	Top-1	79.3	79.3
Xception	ImageNet	Top-1	79.2	79.2
SSD-RN50	COCO2017	bbAP	24.8	24.8
MaskRCNN-RN50	COCO2017	bbAP	37.9	37.9
FairSeq Transformer	EN-DE WMT'14	BLEU	28.2	28.5
BERT-Large	SQuAD v1.1	F1	91.9	91.9

Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT

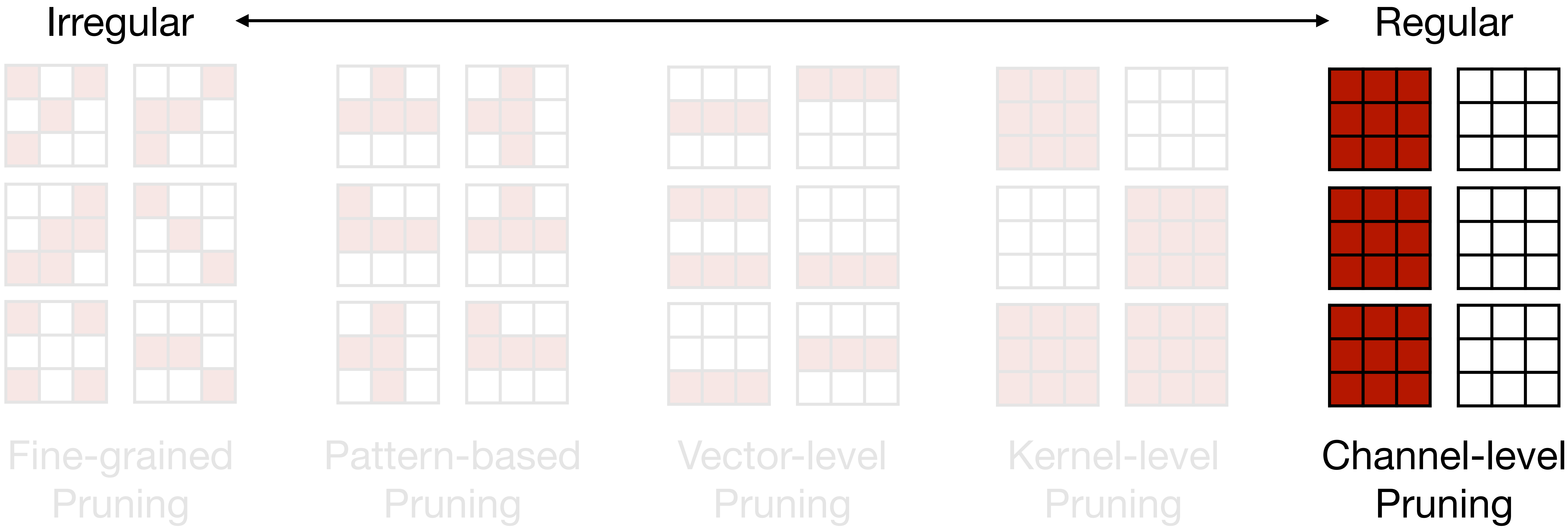
Pruning at Different Granularities

The case of convolutional layers

- Some of the commonly used pruning granularities



Pros? **Cons?**

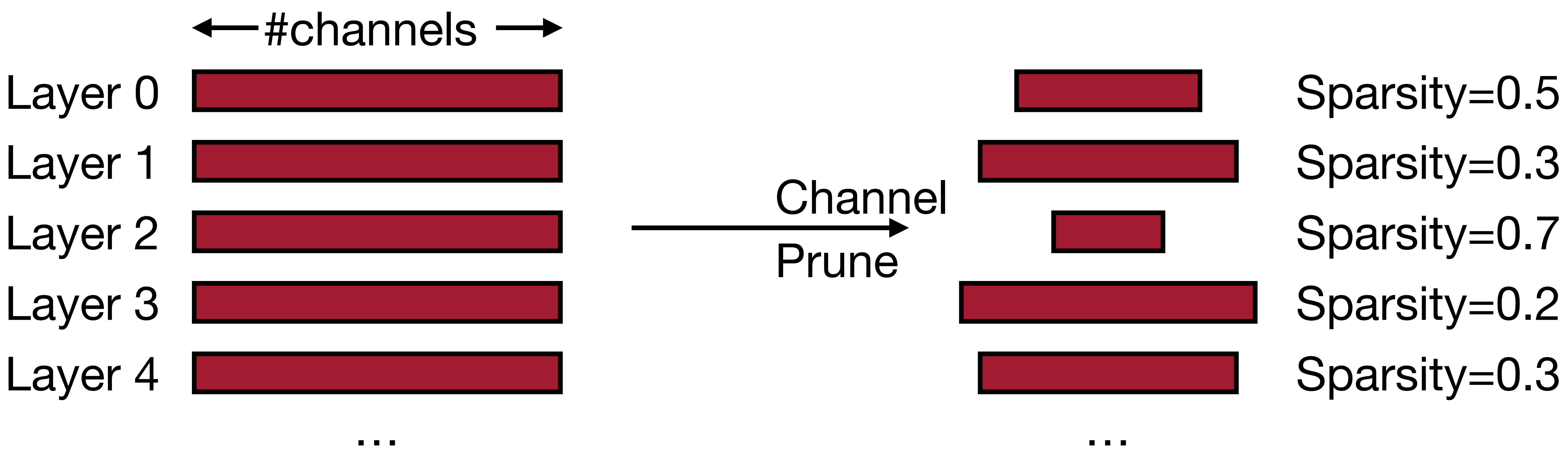


Exploring the granularity of sparsity in convolutional neural networks [Mao *et al.*, CVPR-W]

Pruning at Different Granularities

Let's look into some cases

- **Channel Pruning**
  - Pro: Direct speed up due to reduced channel numbers (leading to an NN with smaller #channels)
  - Con: smaller compression ratio

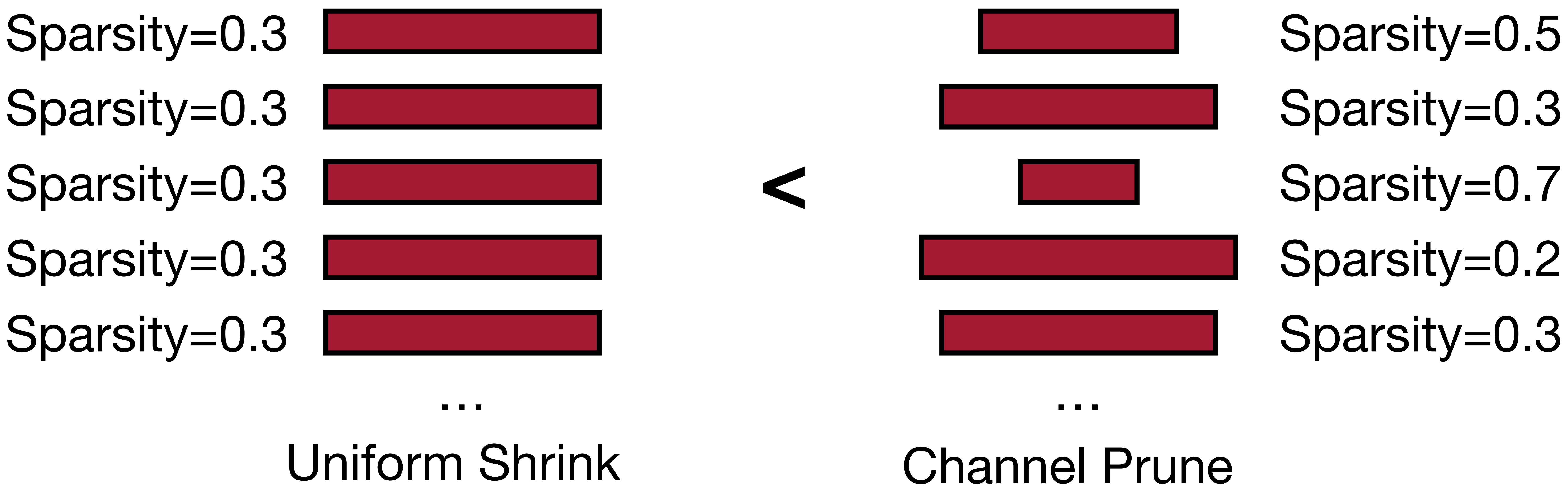


Pruning at Different Granularities

Let's look into some cases

- **Channel Pruning**
  - Pro: Direct speed up due to reduced channel numbers (leading to an NN with smaller #channels)
  - Con: smaller compression ratio

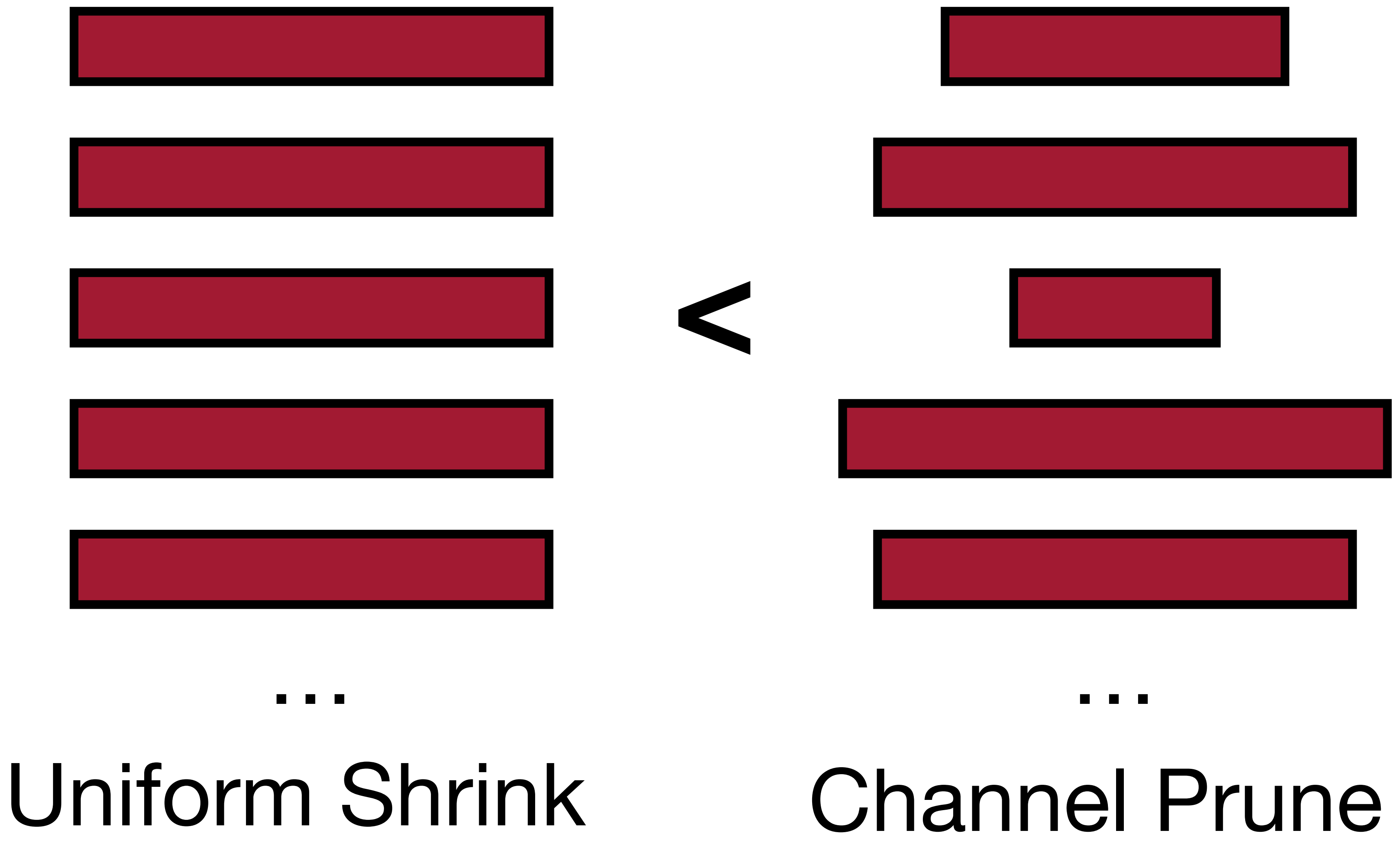
We will later discuss how to find sparsity ratios

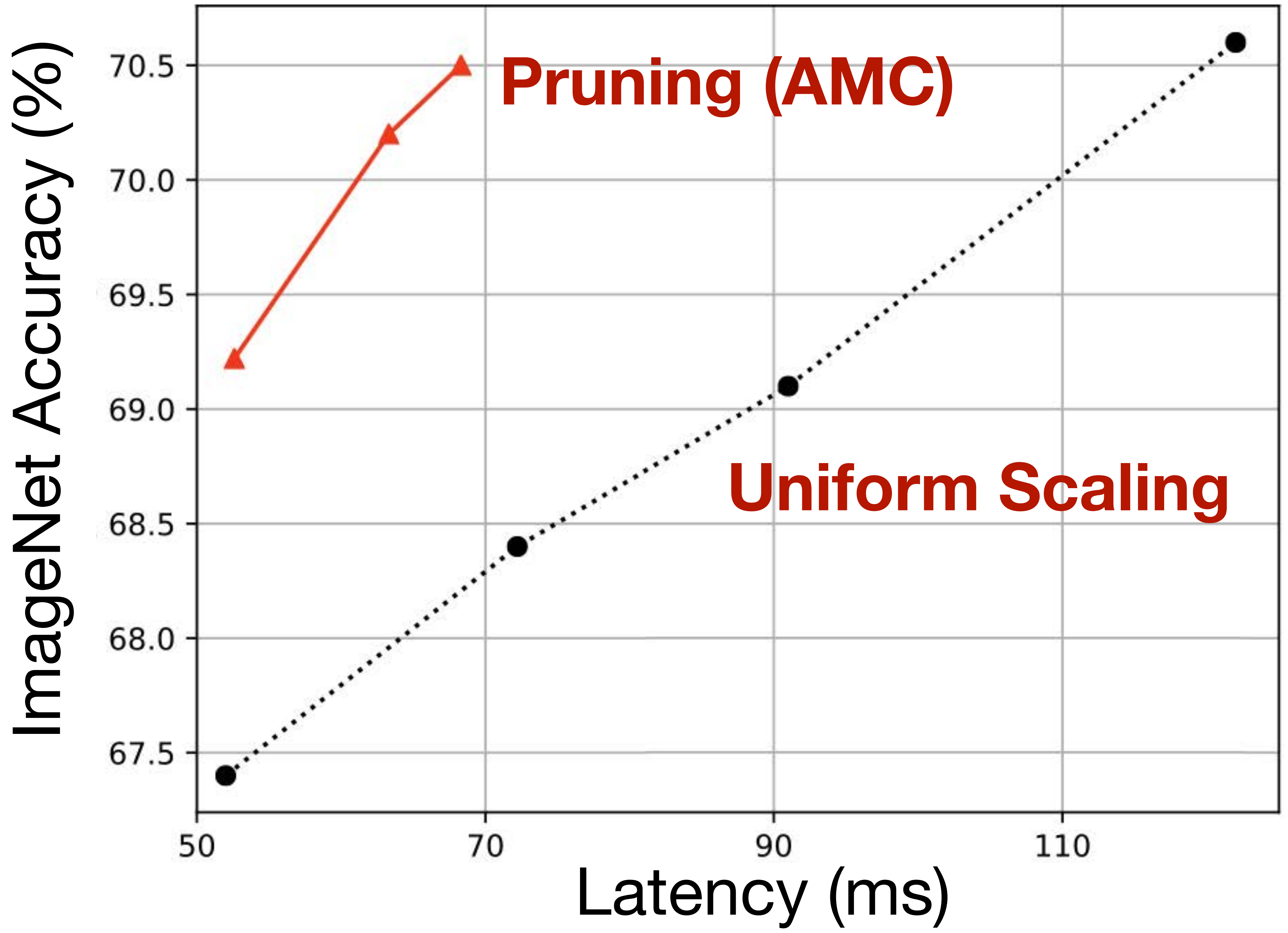


Pruning at Different Granularities

Let's look into some cases

- **Channel Pruning**
  - Pro: Direct speed up due to reduced channel numbers (leading to an NN with smaller #channels)
  - Con: smaller compression ratio

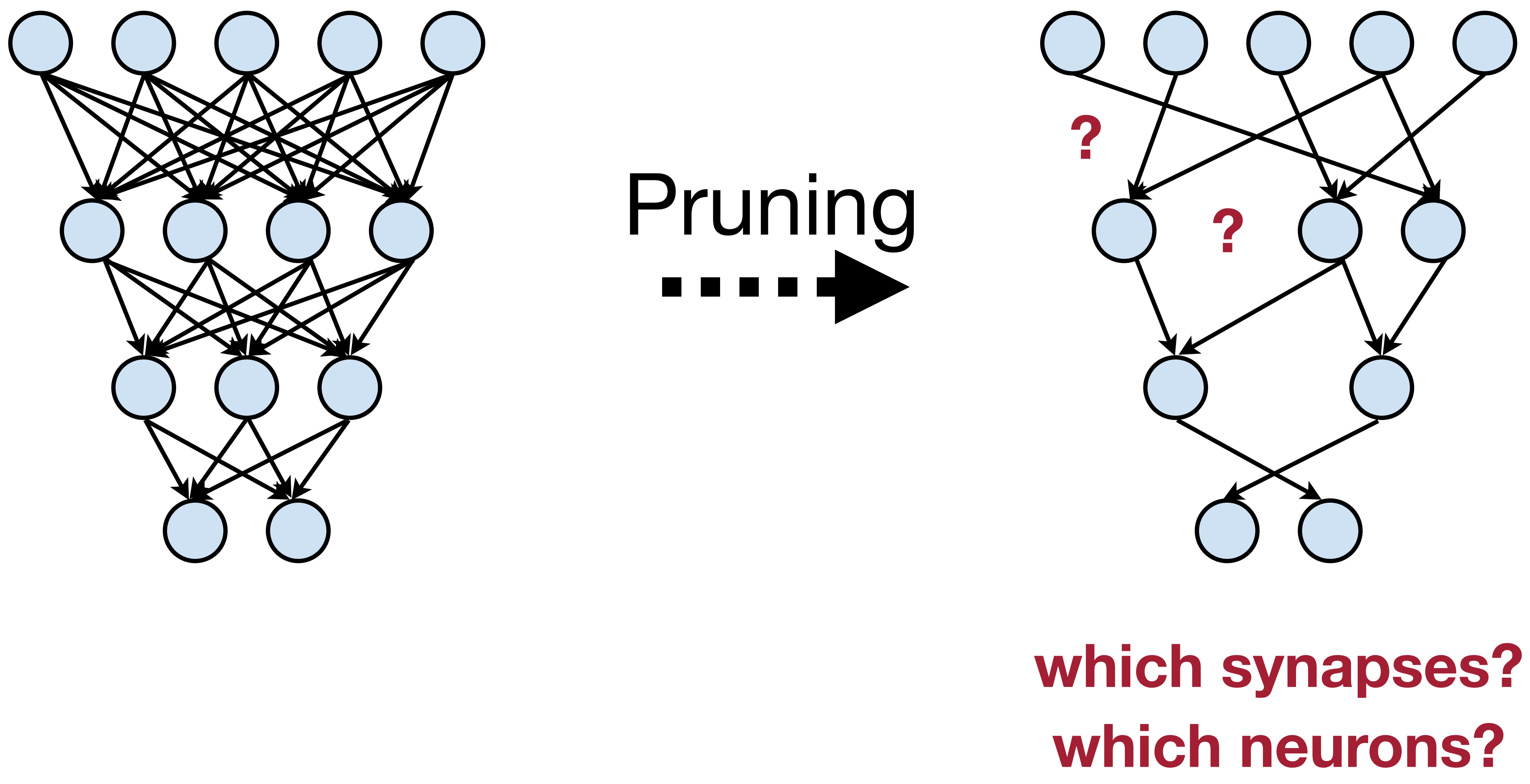




AMC: Automl for Model Compression and Acceleration on Mobile Devices [He *et al.*, ECCV 2018]

Neural Network Pruning

- **Introduction to Pruning**
  - What is pruning?
  - How should we formulate pruning?
- **Determine the Pruning Granularity**
  - In what pattern should we prune the neural network?
- **Determine the Pruning Criterion**
  - What synapses/neurons should we prune?
- **Determine the Pruning Ratio**
  - What should target sparsity be for each layer?
- **Fine-tune/Train Pruned Neural Network**
  - How should we improve performance of pruned models?



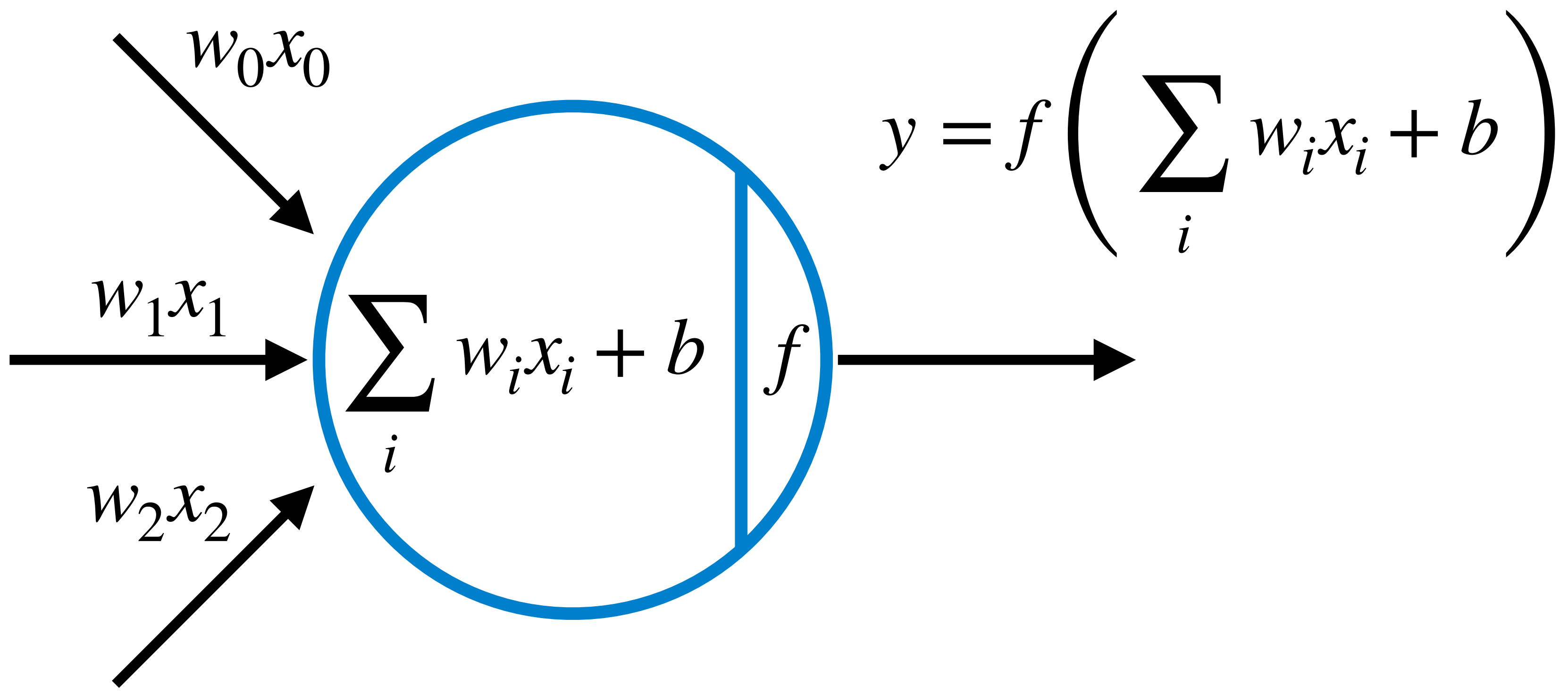
Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Section 3: Pruning Criterion

What synapses and neurons should we prune?

Selection of Synapses to Prune

- When removing parameters from a neural network model,
  - ***the less important*** the parameters being removed are,
  - the better the performance of pruned neural network is.



Example

$$f(\cdot) = \text{ReLU}(\cdot),\quad W = [10, -8, 0.1]$$

➡ $y = \text{ReLU}(10x_0 - 8x_1 + 0.1x_2)$

- If one weight will be removed, which one?

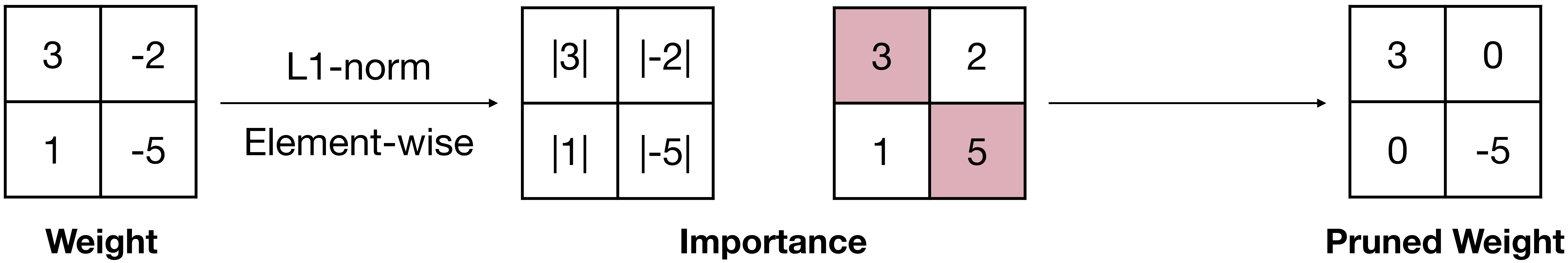
Magnitude-based Pruning

A heuristic pruning criterion

- Magnitude-based pruning considers weights with ***larger absolute values*** are more important than other weights.
  - For element-wise pruning,

$$Importance = |W|$$

- **Example**



Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Magnitude-based Pruning

A heuristic pruning criterion

- Magnitude-based pruning considers weights with ***larger absolute values*** are more important than other weights.
  - For row-wise pruning, the L1-norm magnitude can be defined as,

$$Importance = \sum_{i \in S} |w_i|, \quad \text{where } \mathbf{W}^{(S)} \text{ is the structural set } S \text{ of parameters } \mathbf{W}$$

- **Example**

$$\begin{bmatrix} 3 & -2 \\ 1 & -5 \end{bmatrix} \xrightarrow[\text{Row-wise}]{\text{L1-norm}} \begin{bmatrix} |3|+|-2| \\ |1|+|-5| \end{bmatrix} \quad \begin{bmatrix} 5 \\ 6 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 0 \\ 1 & -5 \end{bmatrix}$$

Weight **Importance** **Pruned Weight**

Learning Both Weights and Connections for Efficient Neural Network [Han *et al*., NeurIPS 2015]

Magnitude-based Pruning

A heuristic pruning criterion

- Magnitude-based pruning considers weights with ***larger absolute values*** are more important than other weights.
  - For row-wise pruning, the L2-norm magnitude can be defined as,

$$Importance = \sqrt{\sum_{i \in S} |w_i|^2}, \text{ where } \mathbf{W}^{(S)} \text{ is the structural set } S \text{ of parameters } \mathbf{W}$$

- **Example**

Weight

3	-2
1	-5

L2-norm

Row-wise

Importance

$\sqrt{13} = \sqrt{\lvert 3 \rvert^2 + \lvert -2 \rvert^2}$
$\sqrt{26} = \sqrt{\lvert 1 \rvert^2 + \lvert -5 \rvert^2}$

√13
√26

Pruned Weight

0	0
1	-5

Learning Both Weights and Connections for Efficient Neural Network [Han *et al*., NeurIPS 2015]

Magnitude-based Pruning

A heuristic pruning criterion

- Magnitude-based pruning considers weights with ***larger absolute values*** are more important than other weights.
- Magnitude is also known as L_p-norm defined as,

$$\|\mathbf{W}^{(S)}\|_p = \left(\sum_{i \in S} |w_i|^p\right)^{\frac{1}{p}}$$, where $\mathbf{W}^{(S)}$ is a structural set of parameters

- **Example**

Weight

3	-2
1	-5

$\xrightarrow[\text{Row-wise}]{\text{L2-norm}}$

Importance

$\sqrt{13} = \sqrt{|3|^2 + |-2|^2}$
$\sqrt{26} = \sqrt{|1|^2 + |-5|^2}$

$\sqrt{13}$
$\sqrt{26}$

$\longrightarrow$

Pruned Weight

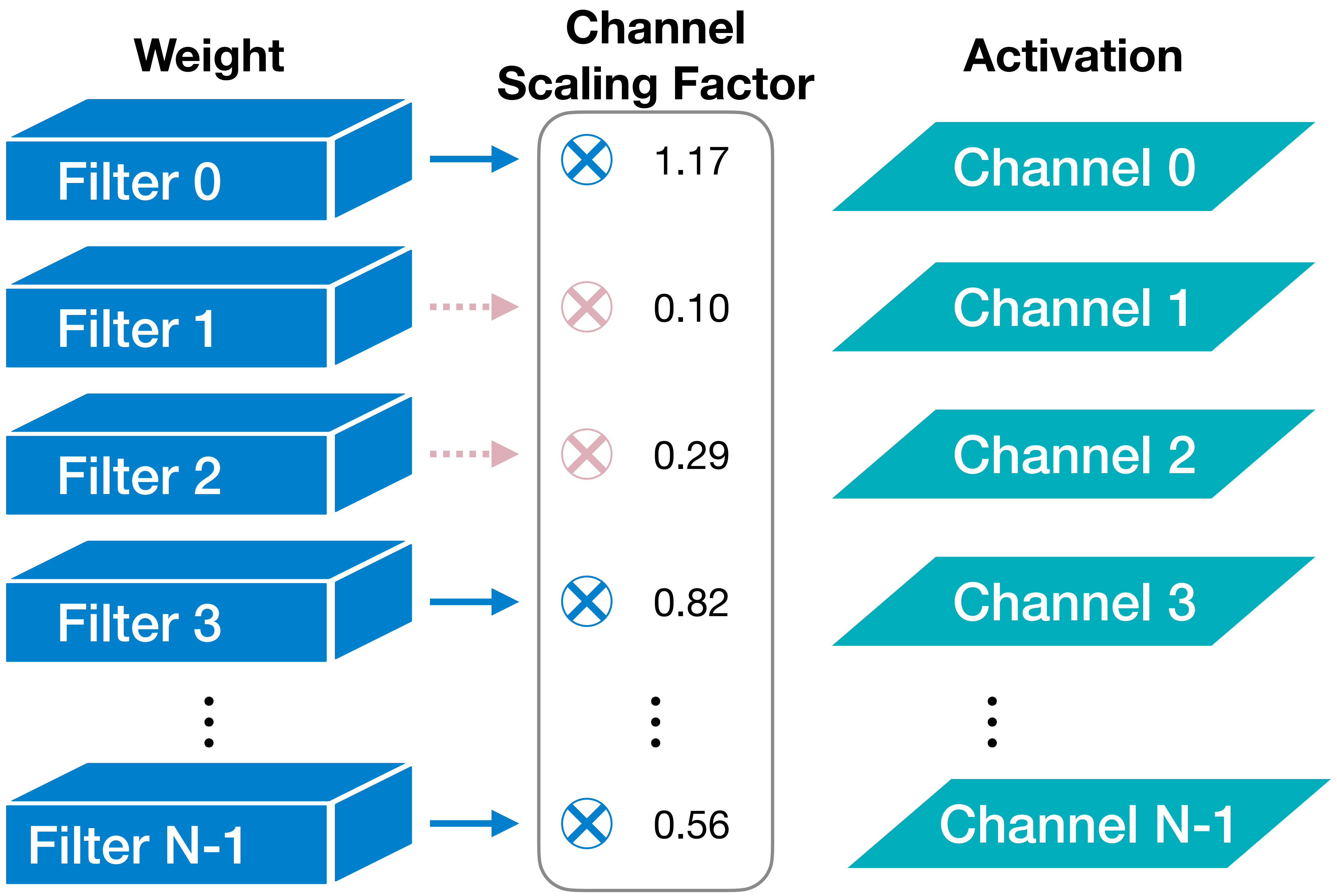
0	0
1	-5

Learning Structured Sparsity in Deep Neural Networks [Wen *et al.*, NeurIPS 2016]

Scaling-based Pruning

Pruning criterion for filter pruning

- A scaling factor is associated with each filter (*i.e.*, output channel) in convolutional layers
  - The scaling factor is multiplied to the output of that channel
  - The scaling factors are trainable parameters

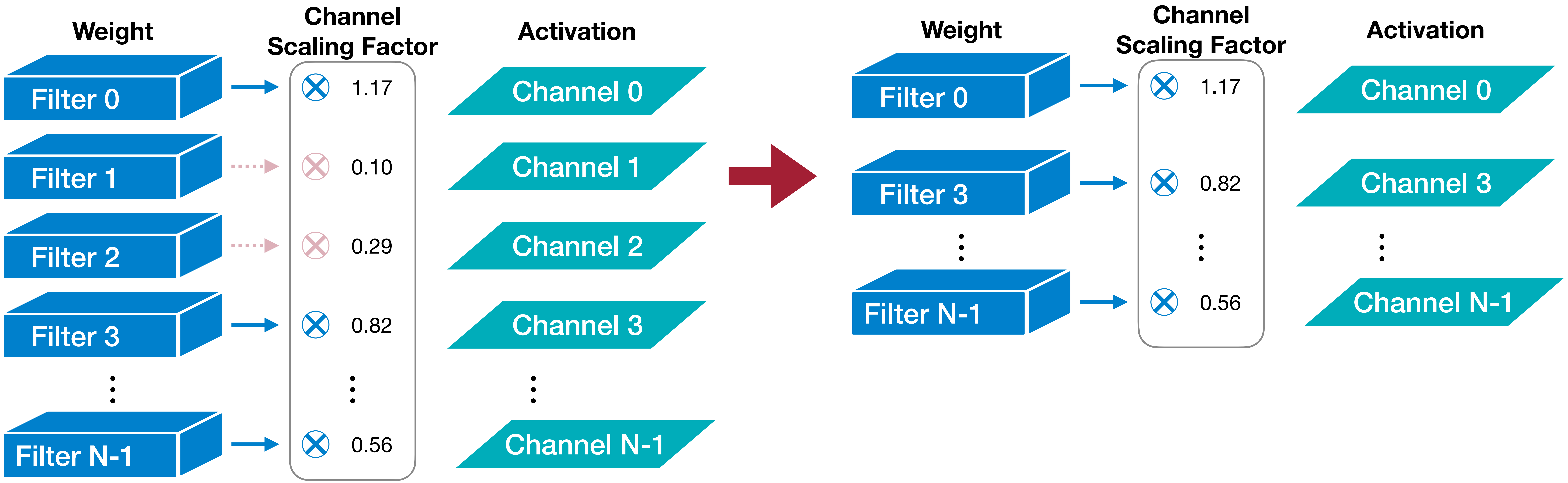


Learning Efficient Convolutional Networks through Network Slimming [Liu *et al.*, ICCV 2017]

Scaling-based Pruning

Pruning criterion for filter pruning

- A scaling factor is associated with each filter (*i.e.*, output channel) in convolutional layers
  - The scaling factor is multiplied to the output of that channel
  - The scaling factors are trainable parameters
- The filters/output channels with small scaling factor magnitude will be pruned



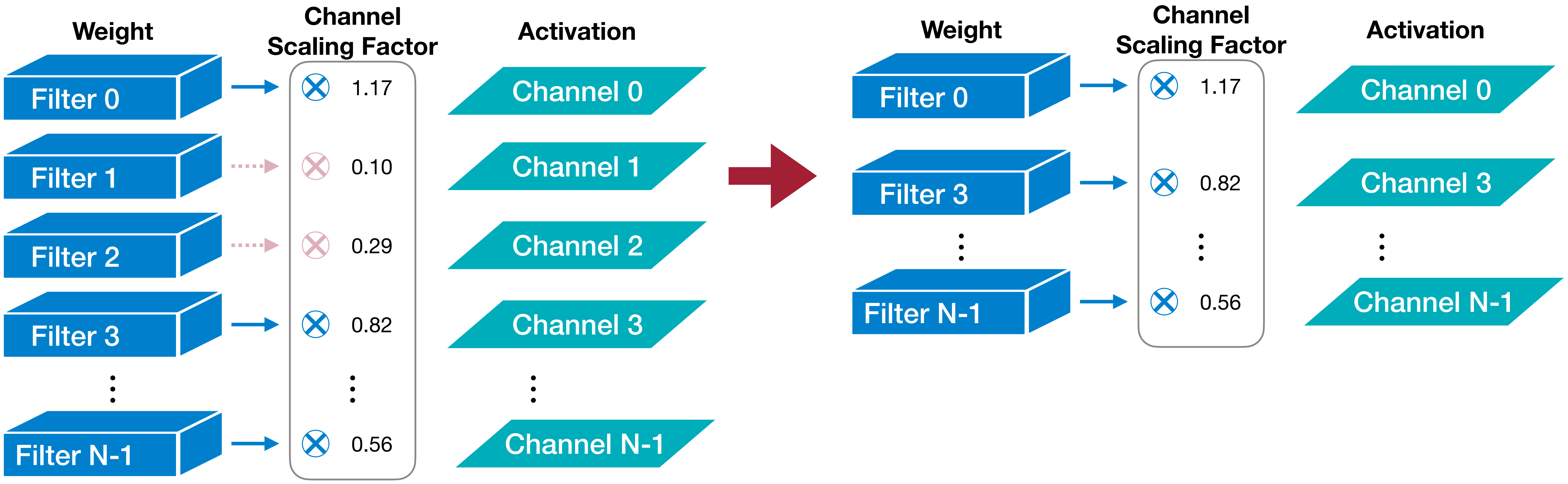
Learning Efficient Convolutional Networks through Network Slimming [Liu *et al.*, ICCV 2017]

Scaling-based Pruning

Pruning criterion for filter pruning

- A scaling factor is associated with each filter (*i.e.*, output channel) in convolutional layers
- The scaling factors can be reused from batch normalization layer

$$\mathbf{z}_o = \gamma \frac{\mathbf{z}_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \beta$$



Learning Efficient Convolutional Networks through Network Slimming [Liu *et al.*, ICCV 2017]

Second-Order-based Pruning

Minimize the error on loss function introduced by pruning synapses

- The induced error can be approximated by a Taylor series.

$$\delta L = L(\mathbf{x};\mathbf{W}) - L(\mathbf{x};\mathbf{W}_P = \mathbf{W} - \delta\mathbf{W}) = \sum_i g_i \delta w_i + \frac{1}{2}\sum_i h_{ii}\delta w_i^2 + \frac{1}{2}\sum_{i \neq j} h_{ij}\delta w_i \delta w_j + O(\|\delta\mathbf{W}\|^3)$$

where

$$g_i = \frac{\partial L}{\partial w_i}, \ h_{i,j} = \frac{\partial^2 L}{\partial w_i \partial w_j}$$

- Optimal Brain Damage assumes that

Optimal Brain Damage [LeCun *et al.*, NeurIPS 1989]

Second-Order-based Pruning

Minimize the error on loss function introduced by pruning synapses

- The induced error can be approximated by a Taylor series.

$$\delta L = L(\mathbf{x}; \mathbf{W}) - L(\mathbf{x}; \mathbf{W}_P = \mathbf{W} - \delta\mathbf{W}) = \sum_i g_i \delta w_i + \frac{1}{2}\sum_i h_{ii}\delta w_i^2 + \frac{1}{2}\sum_{i \neq j} h_{ij}\delta w_i \delta w_j + \cancel{O(\|\delta\mathbf{W}\|^3)}$$

  where

$$g_i = \frac{\partial L}{\partial w_i}, \; h_{i,j} = \frac{\partial^2 L}{\partial w_i \partial w_j}$$

- Optimal Brain Damage assumes that
  - The objective function L is nearly quadratic: the last term is neglected

Optimal Brain Damage [LeCun *et al.*, NeurIPS 1989]

Second-Order-based Pruning

Minimize the error on loss function introduced by pruning synapses

- The induced error can be approximated by a Taylor series.

$$\delta L = L(\mathbf{x}; \mathbf{W}) - L(\mathbf{x}; \mathbf{W}_P = \mathbf{W} - \delta\mathbf{W}) = \sum_i \cancel{g_i \delta w_i} + \frac{1}{2}\sum_i h_{ii}\delta w_i^2 + \frac{1}{2}\sum_{i\neq j} h_{ij}\delta w_i \delta w_j + O(\cancel{\|\delta\mathbf{W}\|^3})$$

where

$$g_i = \frac{\partial L}{\partial w_i}, \; h_{i,j} = \frac{\partial^2 L}{\partial w_i \partial w_j}$$

- Optimal Brain Damage assumes that
  - The objective function L is nearly quadratic: the last term is neglected
  - The neural network training has converged: first-order terms are neglected

Optimal Brain Damage [LeCun *et al.*, NeurIPS 1989]

Second-Order-based Pruning

Minimize the error on loss function introduced by pruning synapses

- The induced error can be approximated by a Taylor series.

$$\delta L = L(\mathbf{x}; \mathbf{W}) - L(\mathbf{x}; \mathbf{W}_P = \mathbf{W} - \delta\mathbf{W}) = \sum_i \cancel{g_i \delta w_i} + \frac{1}{2}\sum_i h_{ii}\delta w_i^2 + \frac{1}{2}\sum_{i \neq j} \cancel{h_{ij}\delta w_i \delta w_j} + \cancel{O(\|\delta\mathbf{W}\|^3)}$$

  where

$$g_i = \frac{\partial L}{\partial w_i}, \; h_{i,j} = \frac{\partial^2 L}{\partial w_i \partial w_j}$$

- Optimal Brain Damage assumes that
  - The objective function L is nearly quadratic: the last term is neglected
  - The neural network training has converged: first-order terms are neglected
  - The error caused by deleting each parameter is independent: cross terms are neglected

Optimal Brain Damage [LeCun *et al.*, NeurIPS 1989]

Second-Order-based Pruning

Minimize the error on loss function introduced by pruning synapses

- The induced error can be approximated by a Taylor series.

$$\delta L = L(\mathbf{x}; \mathbf{W}) - L(\mathbf{x}; \mathbf{W}_P = \mathbf{W} - \delta\mathbf{W}) = \sum_i \cancel{g_i \delta w_i} + \frac{1}{2}\sum_i h_{ii}\delta w_i^2 + \frac{1}{2}\sum_{i \neq j} \cancel{h_{ij}\delta w_i} \delta w_j + \cancel{O(\|\delta \mathbf{W}\|^3)}$$

where

$$g_i = \frac{\partial L}{\partial w_i}, \; h_{i,j} = \frac{\partial^2 L}{\partial w_i \partial w_j}$$

- Optimal Brain Damage assumes that
  - The objective function L is nearly quadratic: the last term is neglected
  - The neural network training has converged: first-order terms are neglected
  - The error caused by deleting each parameter is independent: cross terms are neglected

$$\delta L_i = L(\mathbf{x}; \mathbf{W}) - L(\mathbf{x}; \mathbf{W}_P \,|\, w_i = 0) \approx \frac{1}{2} h_{ii} w_i^2$$

Optimal Brain Damage [LeCun *et al.,* NeurIPS 1989]

Second-Order-based Pruning

Minimize the error on loss function introduced by pruning synapses

- Optimal Brain Damage assumes that
  - The objective function L is nearly quadratic
  - The neural network training has converged
  - The error caused by deleting each parameter is independent

$$\delta L_i = L(\mathbf{x}; \mathbf{W}) - L(\mathbf{x}; \mathbf{W}_P \,|\, w_i = 0) \approx \frac{1}{2} h_{ii} w_i^2, \quad \text{where} \quad h_{ii} = \frac{\partial^2 L}{\partial w_i \partial w_j}$$

  - The synapses with smaller induced error $|\delta L_i|$ will be removed; that is to say,

$$importance_{w_i} = |\delta L_i| = \frac{1}{2} h_{ii} w_i^2$$

* h_{ii} is non-negative

Hessian Matrix *H* is difficult to compute.

Optimal Brain Damage [LeCun *et al.*, NeurIPS 1989]

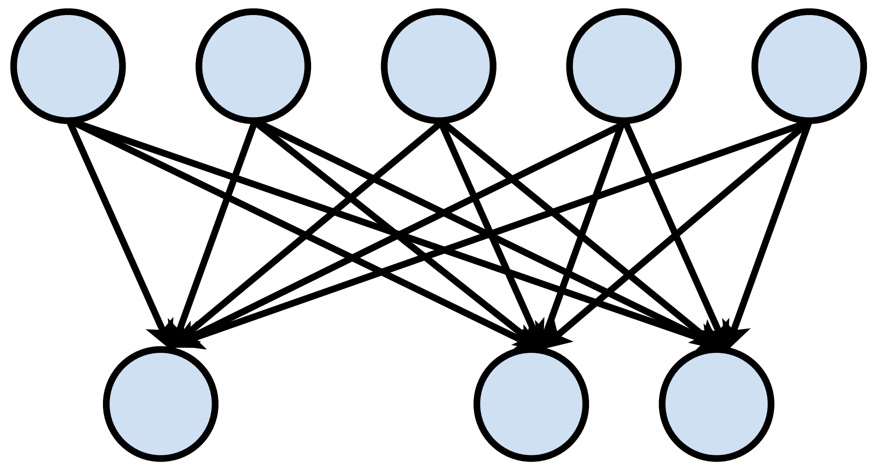
Selection of Neurons to Prune

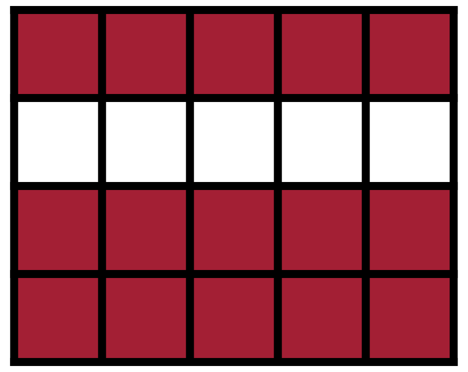
- When removing neurons from a neural network model,
  - ***the less useful*** the neurons being removed are,
  - the better the performance of pruned neural network is.

Neuron pruning is coarse-grained weight pruning

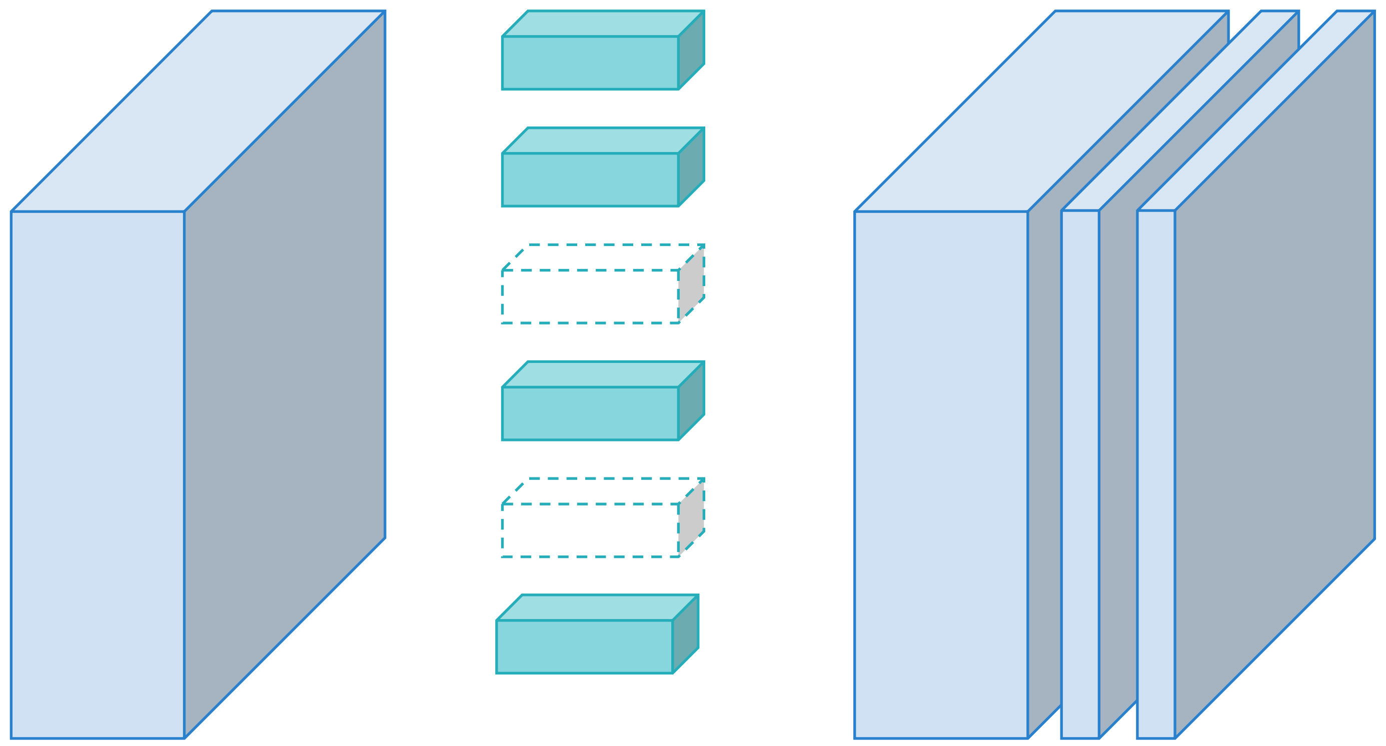
Weight Matrix

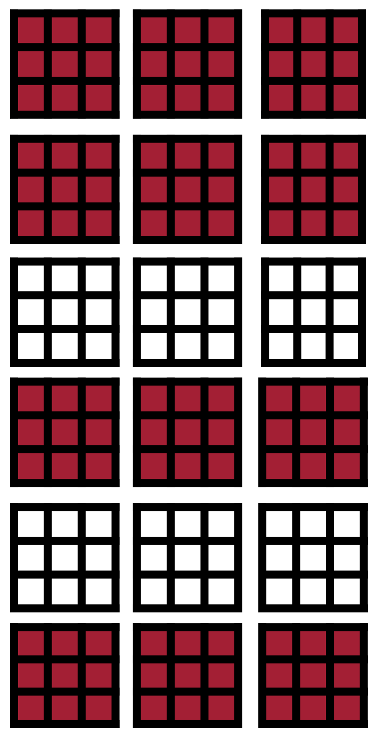
Neuron Pruning in Linear Layer





Channel Pruning in Convolution Layer





Percentage-of-Zero-Based Pruning

- ReLU activation will generate zeros in the output activation.

Output Activations

Width = 4

Height = 4

0	0.1	0.5	1
1.2	0.6	0.3	0.2
0	0.5	**0**	0.3
0.2	**0**	**0**	0.8

0.1	0.5	**0**	**0**
0.2	0.3	**0**	1
0.1	**0**	**0**	0.5
0.1	0.6	0.7	0.1

0	**0**	0.8	**0**
0.7	**0**	0.6	0.1
1.2	1	**0**	0.2
0.5	**0**	0.3	0.5

Channel = 3

Width = 4

Height = 4

0.5	**0**	0.2	0.1
0	0.2	1.2	**0**
1.2	**0**	0.2	0.3
0.2	0.4	**0**	**0**

0.1	0.5	**0**	**0**
0	0.8	**0**	1
0.1	**0**	0.1	1.0
0.2	**0**	1.0	**0**

0	0.8	0.1	**0**
0.2	**0**	**0**	0.3
0	0.4	**0**	0.5
0.2	**0**	0.3	**0**

Channel = 3

Batch = 2

Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures [Hu *et al.*, ArXiv 2017]

Percentage-of-Zero-Based Pruning

- ReLU activation will generate zeros in the output activation.
- Similar to magnitude of weights, the Average Percentage of Zero activations (APoZ) can be exploited to measure the importance of the neurons.

Output Activations

Batch = 2

Batch 0 (Width = 4, Height = 4, Channel = 3)

0	0.1	0.5	1
1.2	0.6	0.3	0.2
0	0.5	**0**	0.3
0.2	**0**	**0**	0.8

0.1	0.5	**0**	**0**
0.2	0.3	**0**	1
0.1	**0**	**0**	0.5
0.1	0.6	0.7	0.1

0	**0**	0.8	**0**
0.7	**0**	0.6	0.1
1.2	1	**0**	0.2
0.5	**0**	0.3	0.5

Batch 1 (Width = 4, Height = 4, Channel = 3)

0.5	**0**	0.2	0.1
0	0.2	1.2	**0**
1.2	**0**	0.2	0.3
0.2	0.4	**0**	**0**

0.1	0.5	**0**	**0**
0	0.8	**0**	1
0.1	**0**	0.1	1.0
0.2	**0**	1.0	**0**

0	0.8	0.1	**0**
0.2	**0**	**0**	0.3
0	0.4	**0**	0.5
0.2	**0**	0.3	**0**

Average Percentage of Zeros (APoZ)

Channel 0: $$= \frac{5+6}{2 \cdot 4 \cdot 4} = \frac{11}{32}$$

Channel 1: $$= \frac{5+7}{2 \cdot 4 \cdot 4} = \frac{12}{32}$$

Channel 2: $$= \frac{6+8}{2 \cdot 4 \cdot 4} = \frac{14}{32}$$

Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures [Hu *et al.*, ArXiv 2017]

Percentage-of-Zero-Based Pruning

- ReLU activation will generate zeros in the output activation.
- Similar to magnitude of weights, the Average Percentage of Zero activations (APoZ) can be exploited to measure the importance of the neurons.
- The smaller APoZ is, the more importance the neuron has.

Output Activations

Width = 4, Height = 4, Channel = 3 (Batch 1)

0	0.1	0.5	1	0.1	0.5	**0**	**0**	**0**	**0**	0.8	**0**
1.2	0.6	0.3	0.2	0.2	0.3	**0**	1	0.7	**0**	0.6	0.1
0	0.5	**0**	0.3	0.1	**0**	**0**	0.5	1.2	1	**0**	0.2
0.2	**0**	**0**	0.8	0.1	0.6	0.7	0.1	0.5	**0**	0.3	0.5

Width = 4, Height = 4, Channel = 3 (Batch 2)

0.5	**0**	0.2	0.1	0.1	0.5	**0**	**0**	**0**	0.8	0.1	**0**
0	0.2	1.2	**0**	**0**	0.8	**0**	1	0.2	**0**	**0**	0.3
1.2	**0**	0.2	0.3	0.1	**0**	0.1	1.0	**0**	0.4	**0**	0.5
0.2	0.4	**0**	**0**	0.2	**0**	1.0	**0**	0.2	**0**	0.3	**0**

Batch = 2

Average Percentage of Zeros (APoZ)

$$= \frac{5+6}{2 \cdot 4 \cdot 4} = \frac{11}{32}$$
Channel 0

$$= \frac{5+7}{2 \cdot 4 \cdot 4} = \frac{12}{32}$$
Channel 1

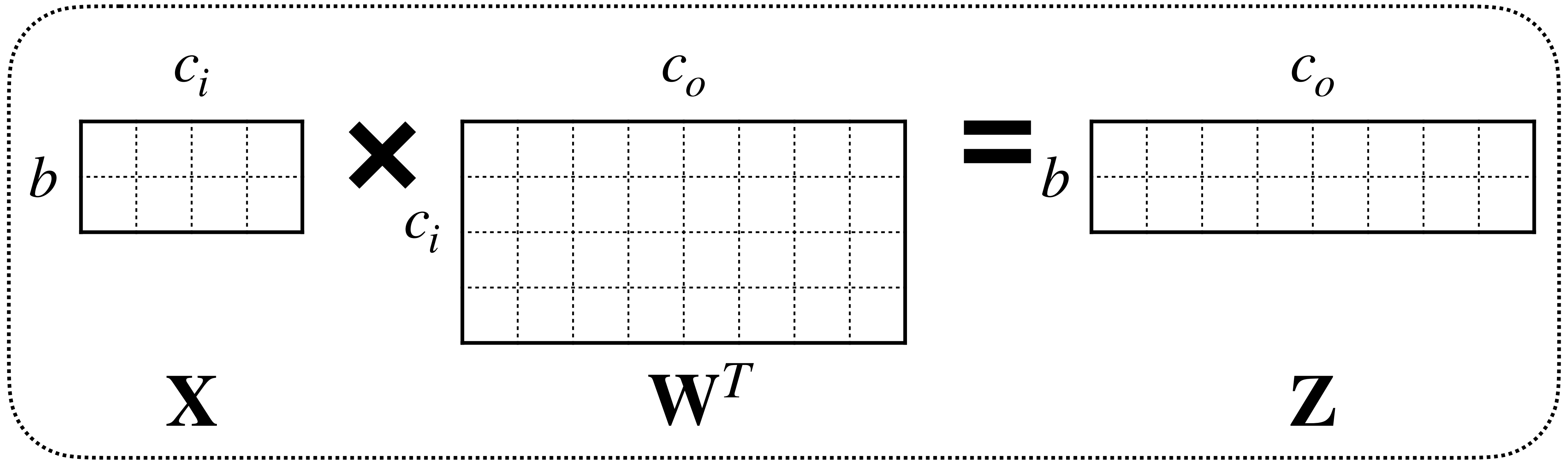
$$= \frac{6+8}{2 \cdot 4 \cdot 4} = \frac{14}{32}$$
Channel 2

Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures [Hu *et al.*, ArXiv 2017]

Regression-based Pruning

Minimize reconstruction error of the corresponding layer's outputs

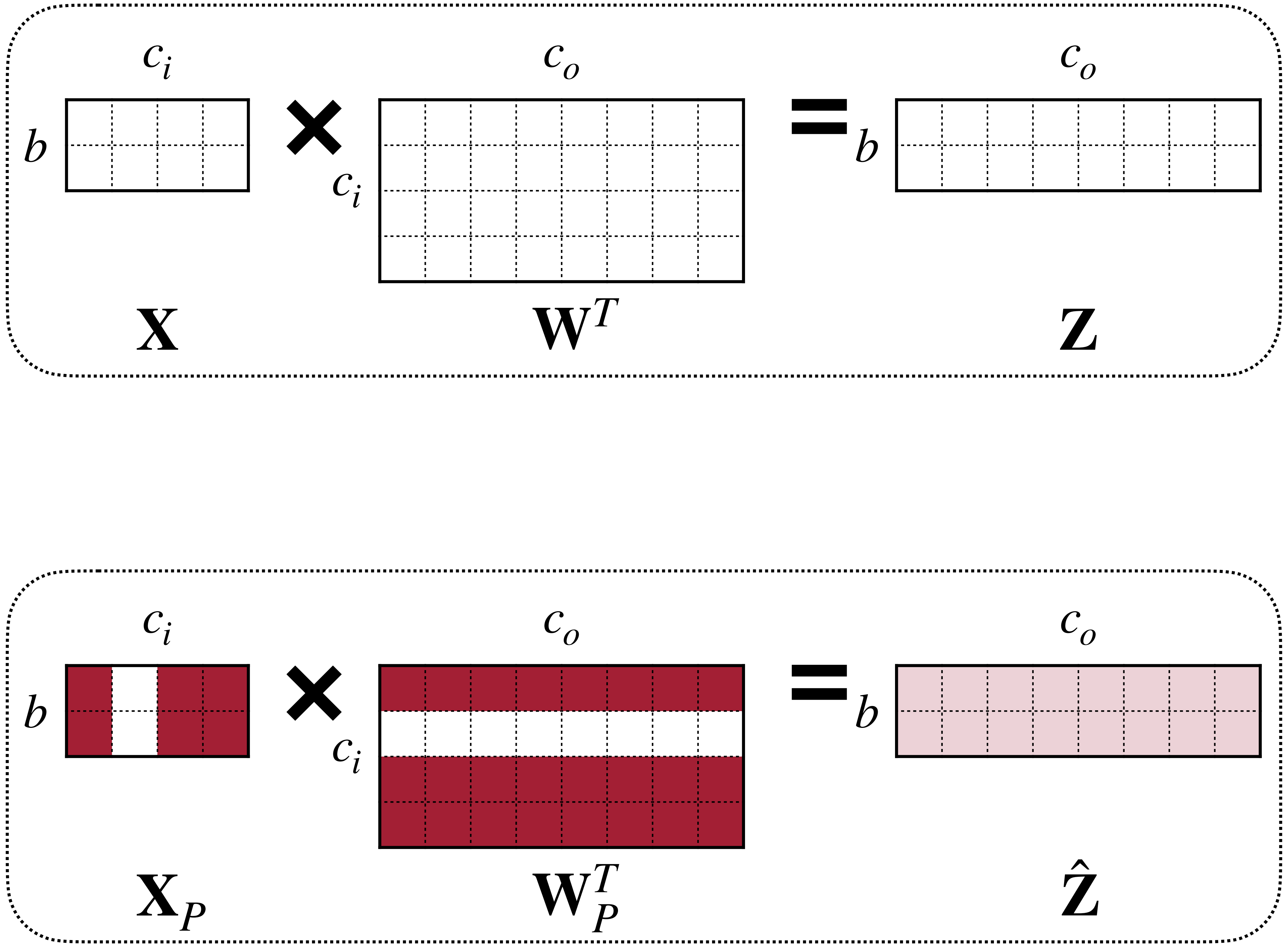
- Instead of considering the pruning error of the objective function $L(\mathbf{x}; \mathbf{W})$, regression-based pruning minimizes the reconstruction error of the corresponding layer's outputs.



Regression-based Pruning

Minimize reconstruction error of the corresponding layer's outputs

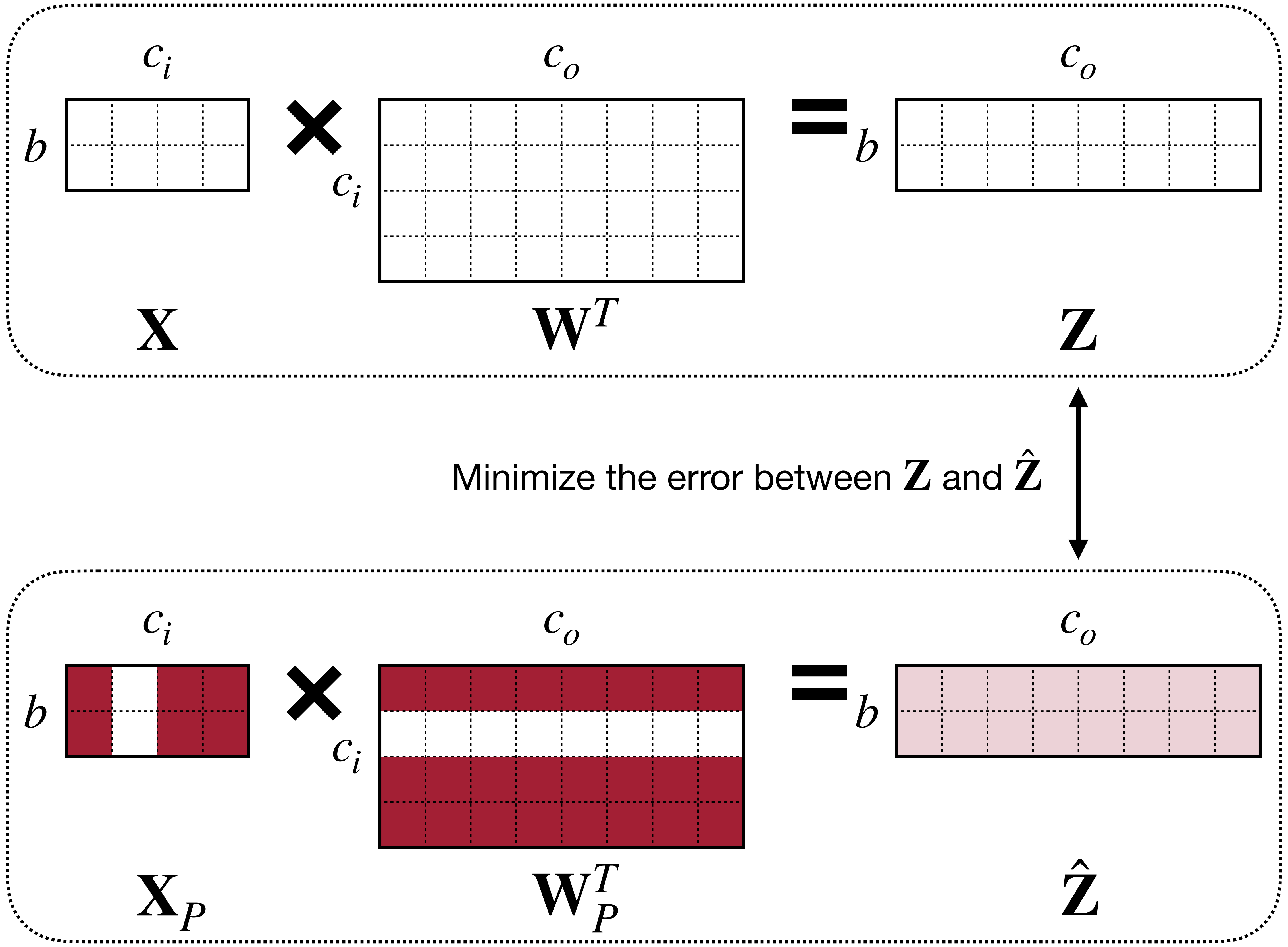
- Instead of considering the pruning error of the objective function $L(\mathbf{x}; \mathbf{W})$, regression-based pruning minimizes the reconstruction error of the corresponding layer's outputs.



Regression-based Pruning

Minimize reconstruction error of the corresponding layer's outputs

- Instead of considering the pruning error of the objective function $L(\mathbf{x}; \mathbf{W})$, regression-based pruning minimizes the reconstruction error of the corresponding layer's outputs.

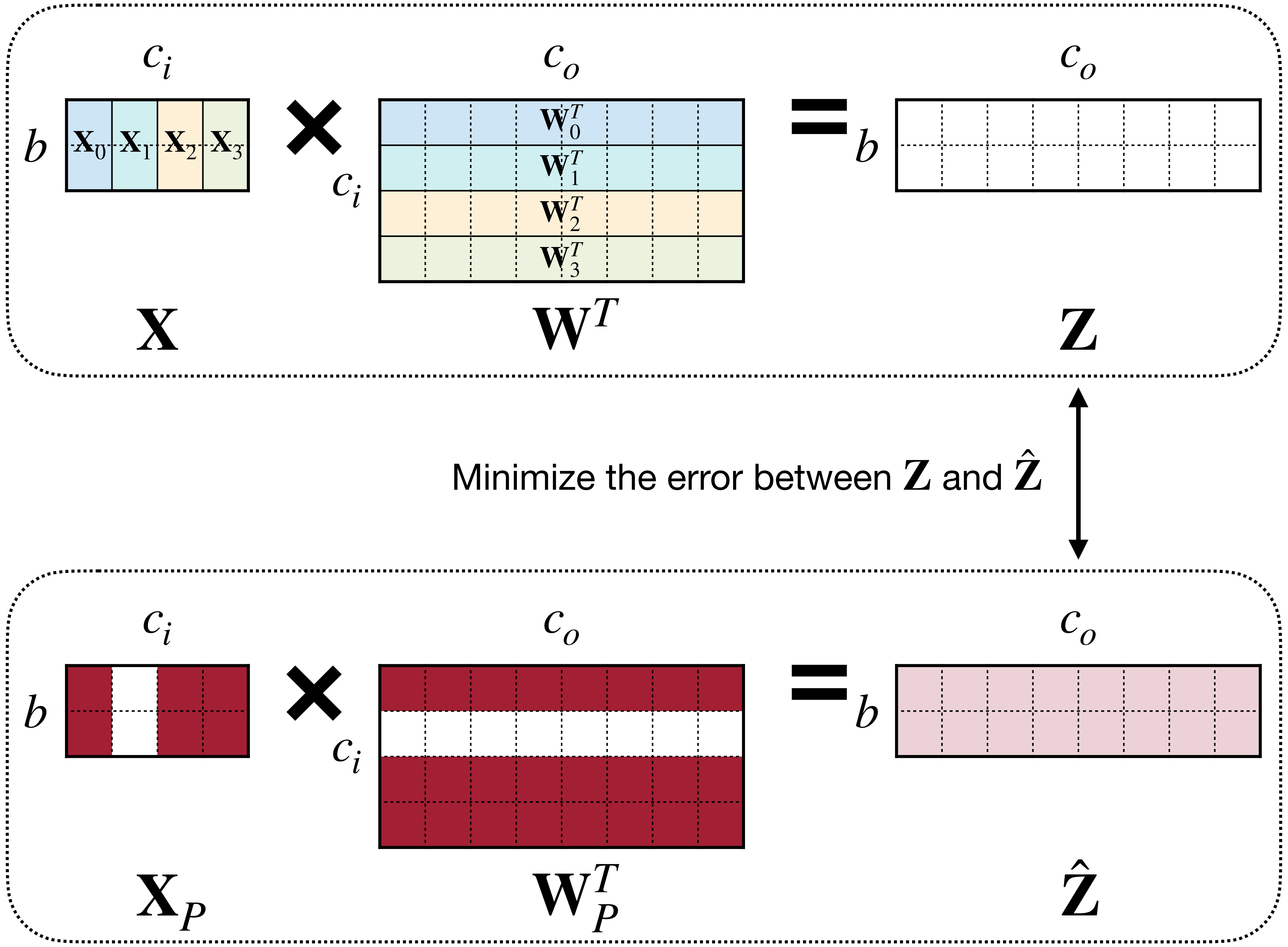


Regression-based Pruning

Minimize reconstruction error of the corresponding layer's outputs

- Let

$$\mathbf{Z} = \mathbf{X}\mathbf{W}^T = \sum_{c=0}^{c_i-1} \mathbf{X}_c \mathbf{W}_c^T$$



Regression-based Pruning

Minimize reconstruction error of the corresponding layer's outputs

- Let

$$\mathbf{Z} = \mathbf{X}\mathbf{W}^T = \sum_{c=0}^{c_i-1} \mathbf{X}_c \mathbf{W}_c^T$$

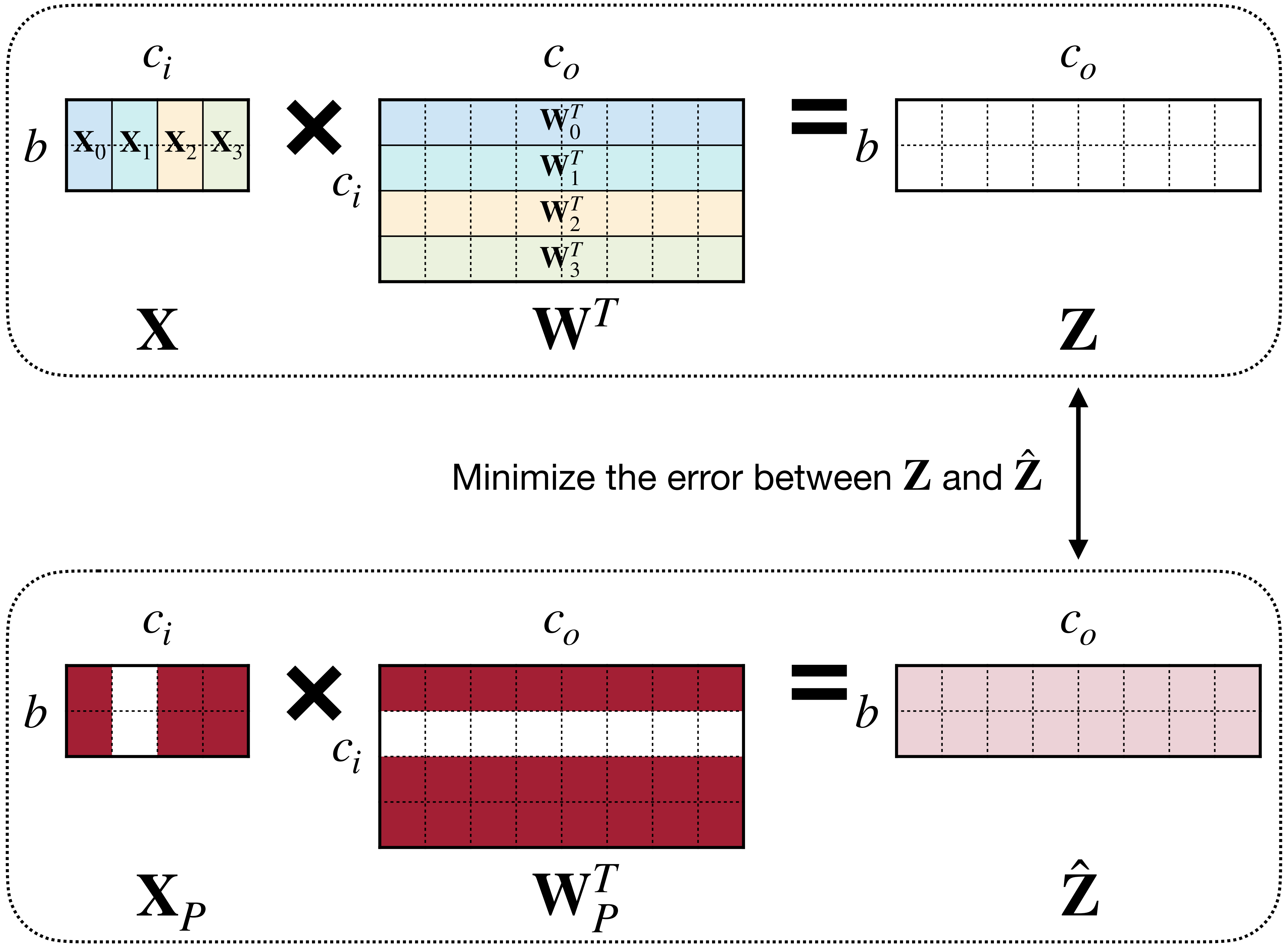
- The problem can be formulate as

$$\arg\min_{\mathbf{W},\, \beta} \|\mathbf{Z} - \hat{\mathbf{Z}}\|_F^2 = \|\mathbf{Z} - \sum_{c=0}^{c_i-1} \beta_c \mathbf{X}_c \mathbf{W}_c^T\|_F^2$$

  subject to

$$\|\beta\|_0 \leq N_c$$

- β is coefficient vector of length c_i for channel selection. $\beta_c = 0$ means channel c is pruned.
- N_c is the number of nonzero channels.



Channel Pruning for Accelerating Very Deep Neural Networks [He *et al.*, ICCV 2017]

Regression-based Pruning

Minimize reconstruction error of the corresponding layer's outputs

- Let

$$\mathbf{Z} = \mathbf{X}\mathbf{W}^T = \sum_{c=0}^{c_i-1} \mathbf{X}_c \mathbf{W}_c^T$$

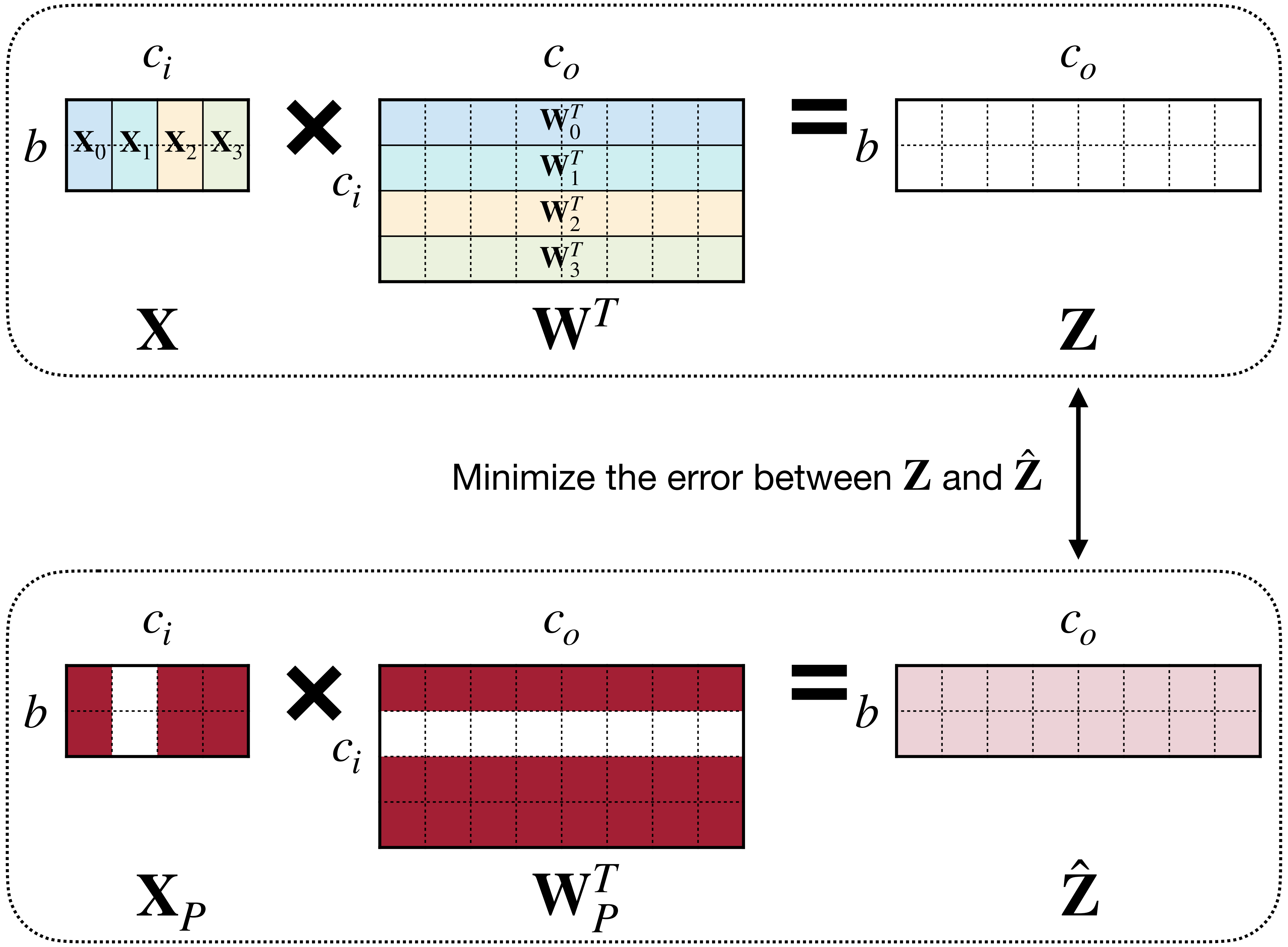
- The problem can be formulate as

$$\underset{\mathbf{W},\,\beta}{\arg\min} \|\mathbf{Z} - \hat{\mathbf{Z}}\|_F^2 = \|\mathbf{Z} - \sum_{c=0}^{c_i-1} \beta_c \mathbf{X}_c \mathbf{W}_c^T\|_F^2$$

subject to

$$\|\beta\|_0 \leq N_c$$

- β is coefficient vector of length c_i for channel selection. $\beta_c = 0$ means channel c is pruned.
- N_c is the number of nonzero channels.
- Solve the problem by:
  - Fix $\mathbf{W}$, solve β for channel selection
  - Fix β, solve $\mathbf{W}$ to minimize reconstruction error



Channel Pruning for Accelerating Very Deep Neural Networks [He *et al.*, ICCV 2017]

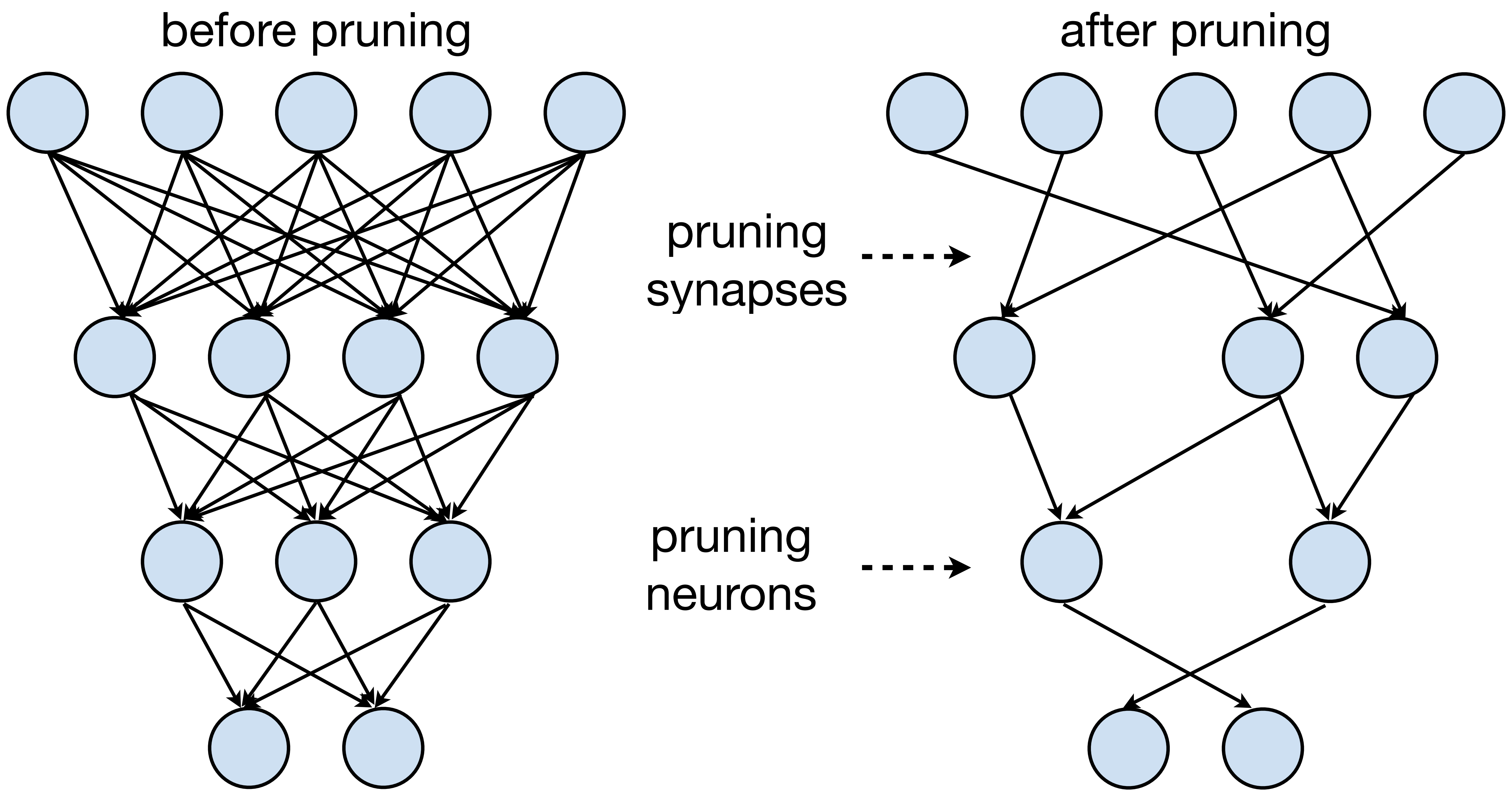
Summary of Today's Lecture

Pruning Demo

In this lecture, we introduced:

- What is pruning
- Granularities of pruning
- Criteria to select weights to prune

- **We will cover in the next lecture:**
  - How to find pruning ratio for each layer
  - How to train/fine-tune the pruned layer
  - Automated ways to find pruning ratios
  - Lottery ticket hypothesis
  - System support for different granularities



References

1. Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey [Deng et al., IEEE 2020]
2. Computing's Energy Problem (and What We Can Do About it) [Horowitz, M., IEEE ISSCC 2014]
3. Optimal Brain Damage [LeCun et al., NeurIPS 1989]
4. Learning Both Weights and Connections for Efficient Neural Network [Han et al., NeurIPS 2015]
5. Efficient Methods and Hardware for Deep Learning [Han S., Stanford University]
6. Peter Huttenlocher (1931–2013) [Walsh, C. A., Nature 2013]
7. Exploring the granularity of sparsity in convolutional neural networks [Mao et al., CVPR-W]
8. Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT
9. AMC: Automl for Model Compression and Acceleration on Mobile Devices [He et al., ECCV 2018]
10. Learning Structured Sparsity in Deep Neural Networks [Wen et al., NeurIPS 2016]
11. Learning Efficient Convolutional Networks through Network Slimming [Liu et al., ICCV 2017]
12. Pruning Convolutional Filters with First Order Taylor Series Ranking [Wang M.]
13. Importance Estimation for Neural Network Pruning [Molchanov et al., CVPR 2019]
14. Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures [Hu et al., ArXiv 2017]
15. Pruning Convolutional Neural Networks for Resource Efficient Inference [Molchanov et al., ICLR 2017]
16. Channel Pruning for Accelerating Very Deep Neural Networks [He et al., ICCV 2017]
17. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression [Luo et al., ICCV 2017]
18. SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot [Elias Frantar, Dan Alistarh, ArXiv 2023]