ECE-GY 9143
Introduction to High Performance Machine Learning

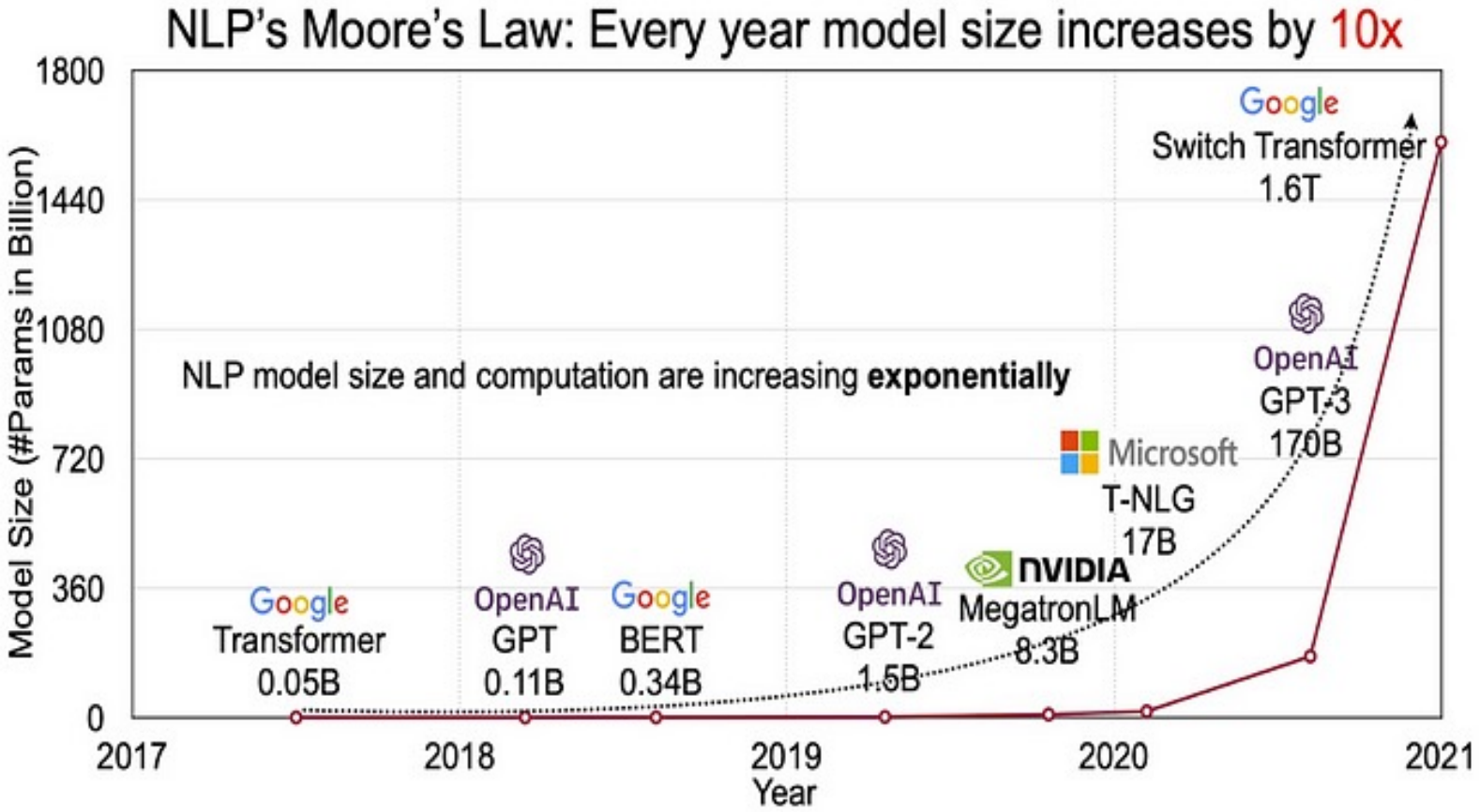
Lecture 10

Parijat Dube

Kaoutar El Maghraoui

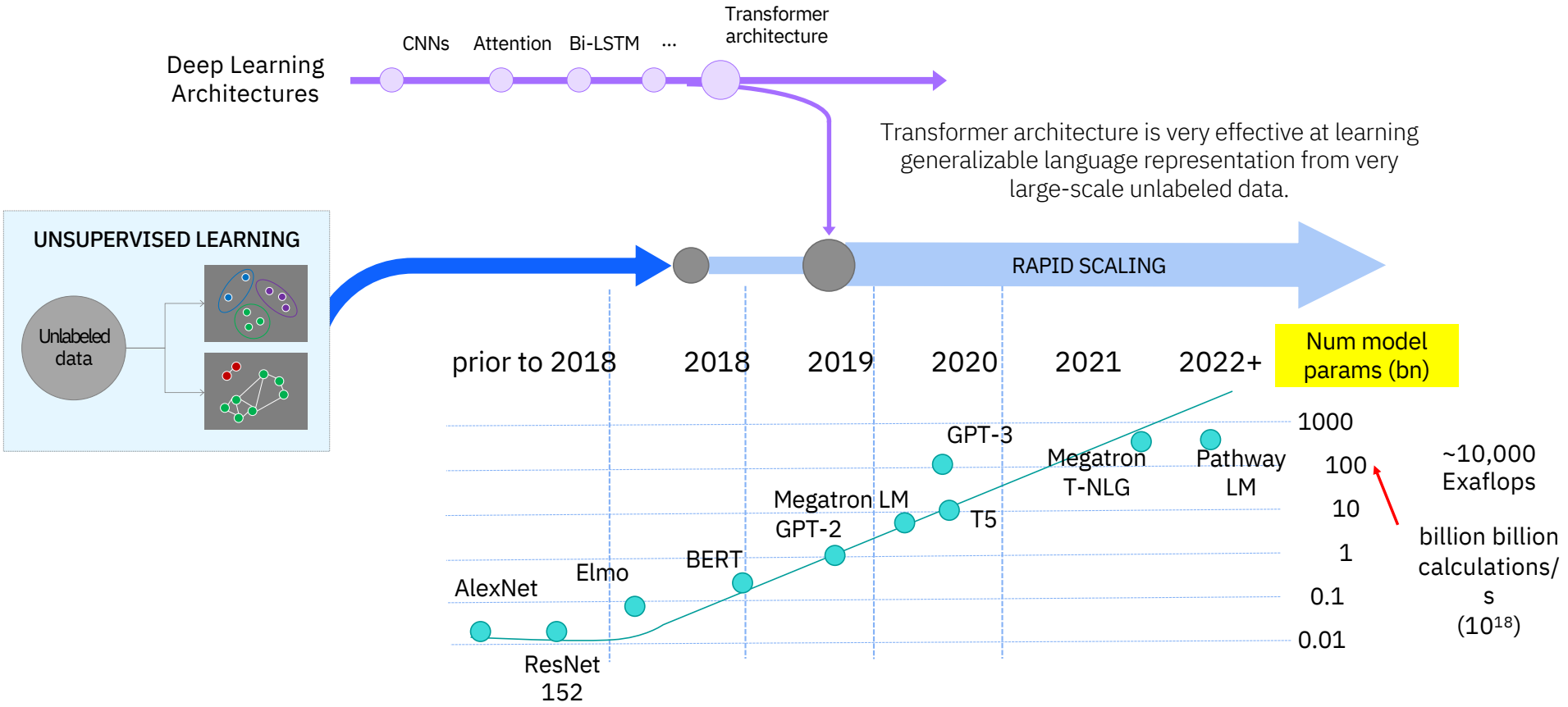
Neural Network Sparsity and Pruning

Today's AI is too BIG!



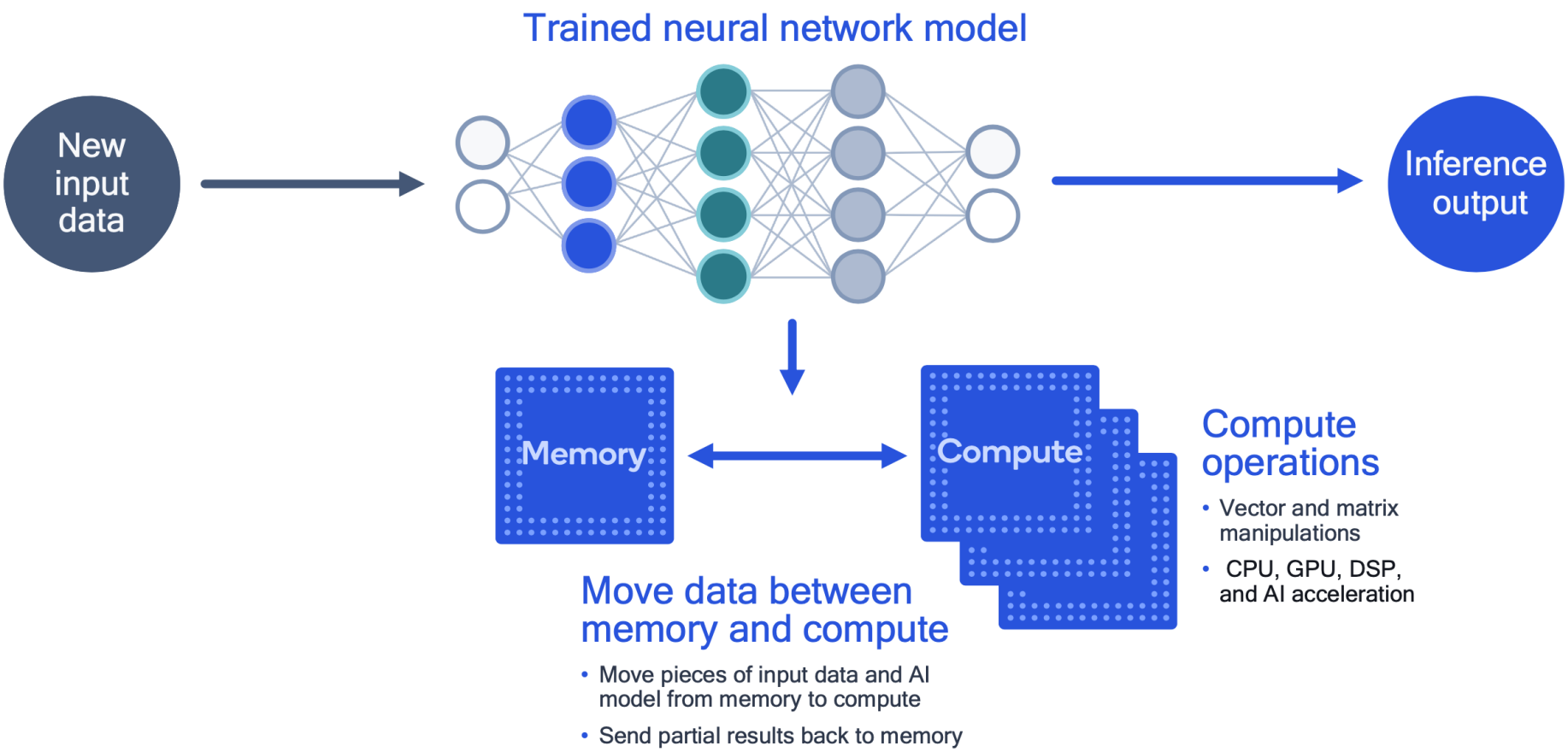
https://efficientml.ai

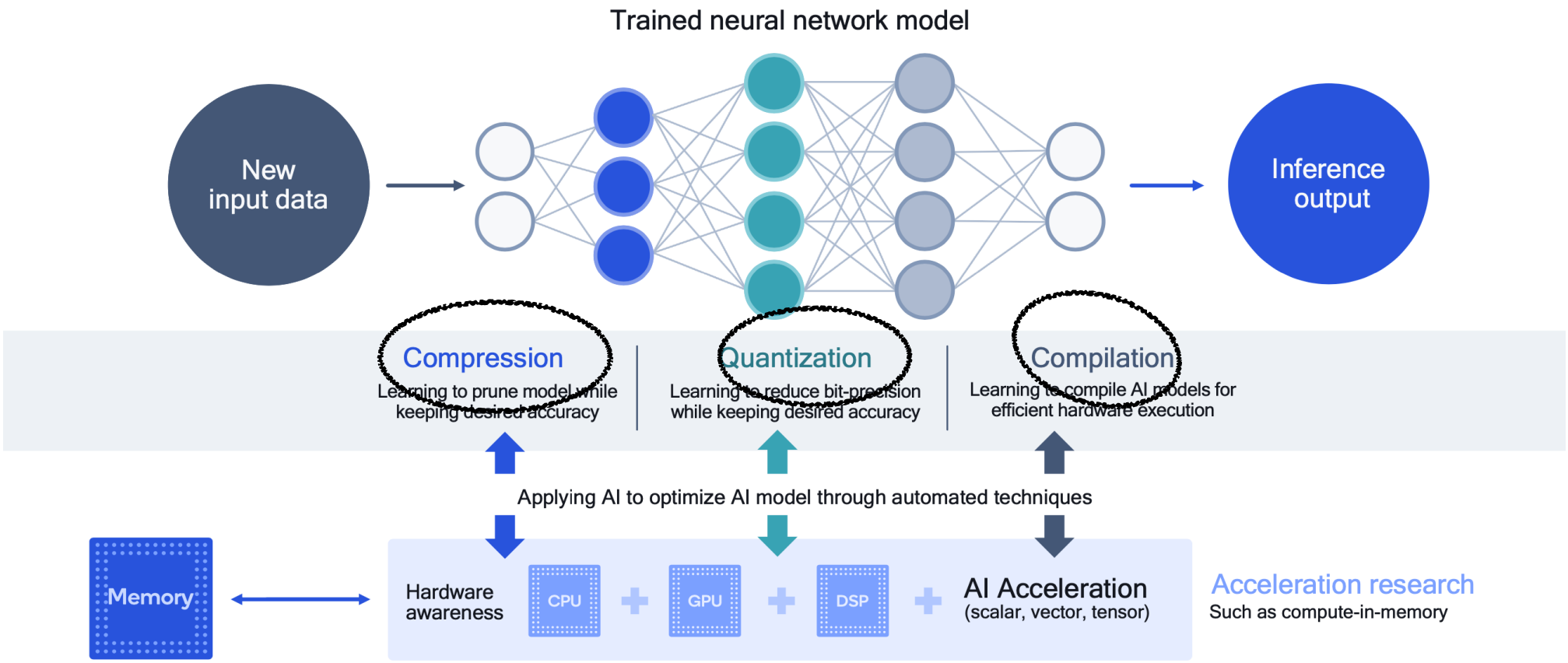
Foundation models are blowing up compute requirements

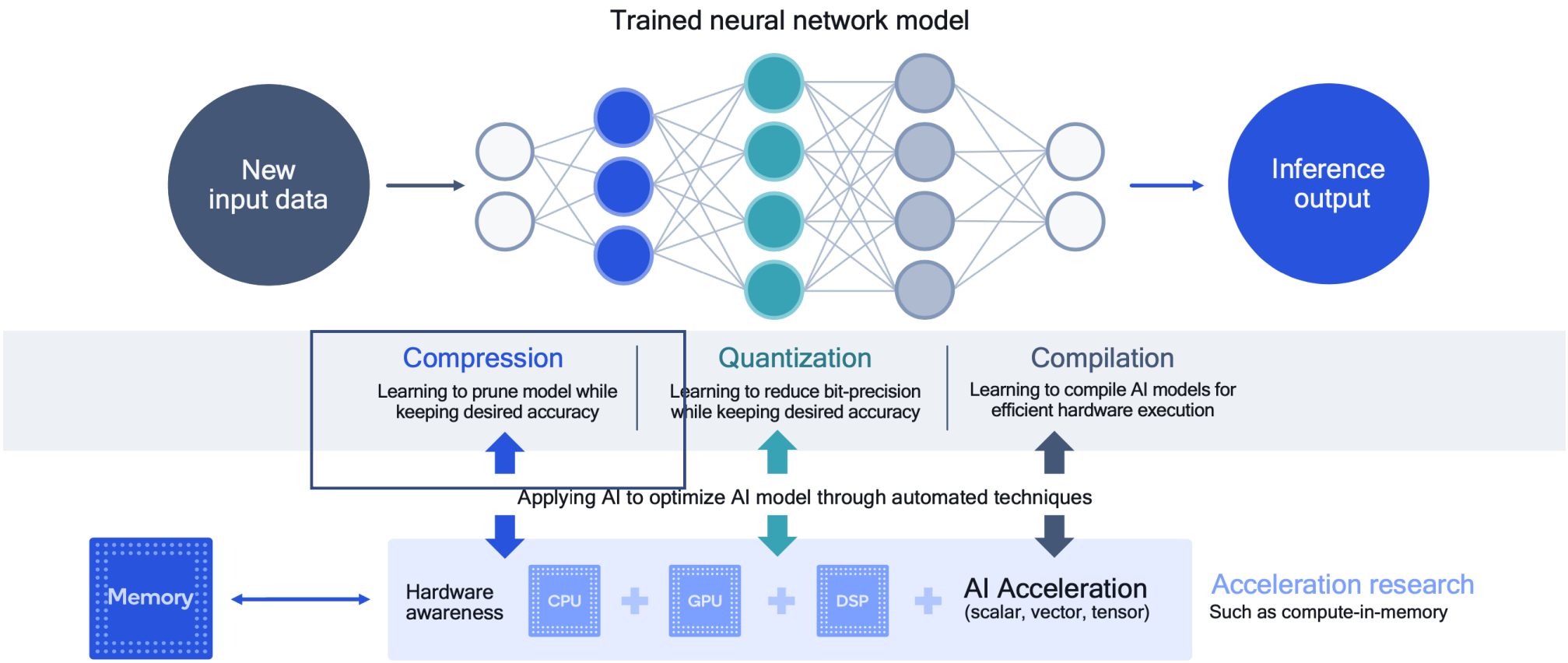


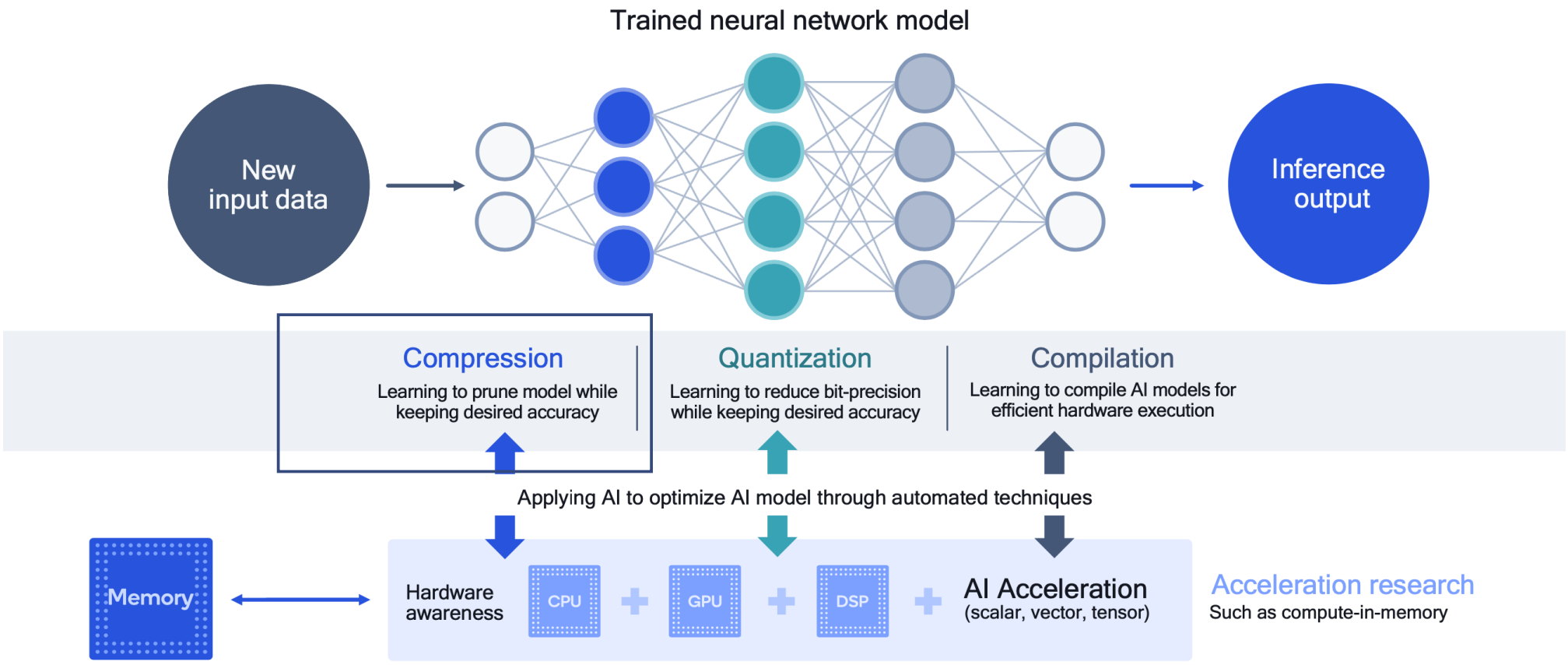
© 2023 IBM Corporation

Advancing AI research to increase power efficiency



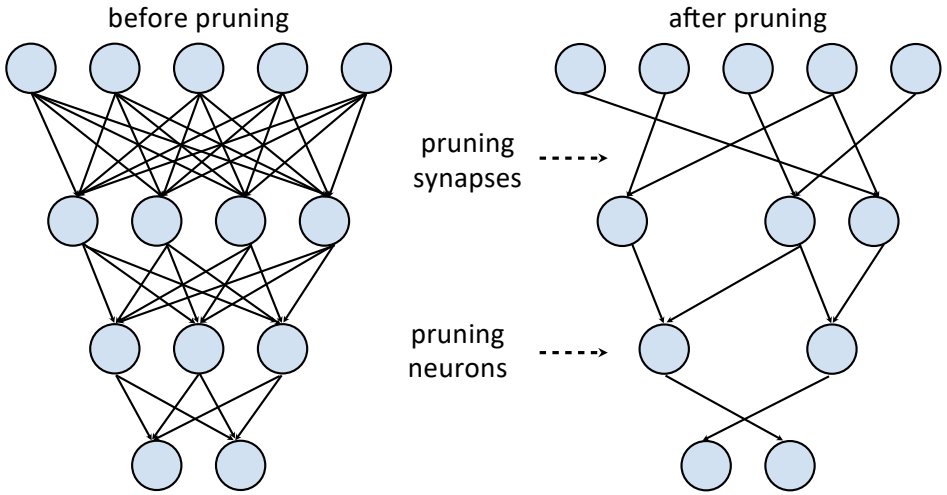

Trained neural network model
New input data
Inference output
Compression
Learning to prune model while keeping desired accuracy
Quantization
Learning to reduce bit-precision while keeping desired accuracy
Compilation
Learning to compile AI models for efficient hardware execution
Applying AI to optimize AI model through automated techniques
Memory
Hardware awareness
CPU
GPU
DSP
AI Acceleration
(scalar, vector, tensor)
Acceleration research
Such as compute-in-memory


Trained neural network model
New input data
Inference output
Compression
Learning to prune model while keeping desired accuracy
Quantization
Learning to reduce bit-precision while keeping desired accuracy
Compilation
Learning to compile AI models for efficient hardware execution
Applying AI to optimize AI model through automated techniques
Memory
Hardware awareness
CPU
GPU
DSP
AI Acceleration
(scalar, vector, tensor)
Acceleration research
Such as compute-in-memory


Trained neural network model
New input data
Inference output
Compression
Learning to prune model while keeping desired accuracy
Quantization
Learning to reduce bit-precision while keeping desired accuracy
Compilation
Learning to compile AI models for efficient hardware execution
Applying AI to optimize AI model through automated techniques
Memory
Hardware awareness
CPU
GPU
DSP
AI Acceleration
(scalar, vector, tensor)
Acceleration research
Such as compute-in-memory

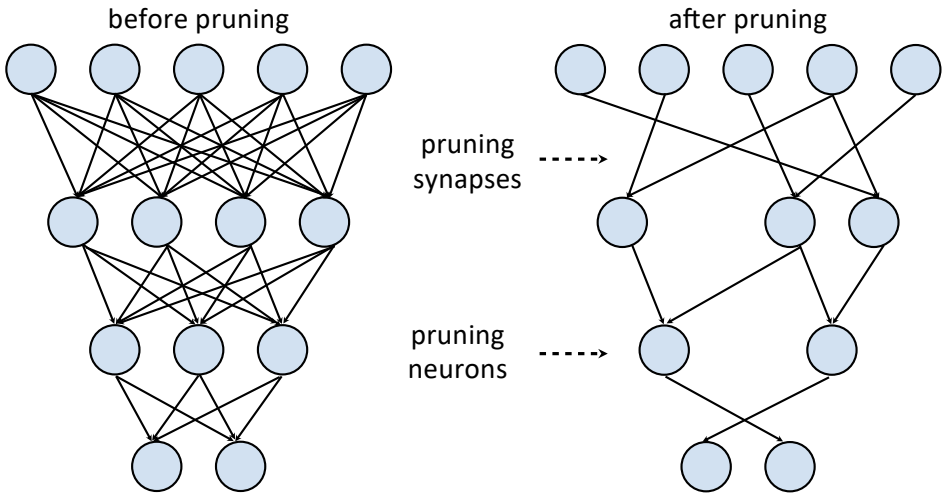
Agenda

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



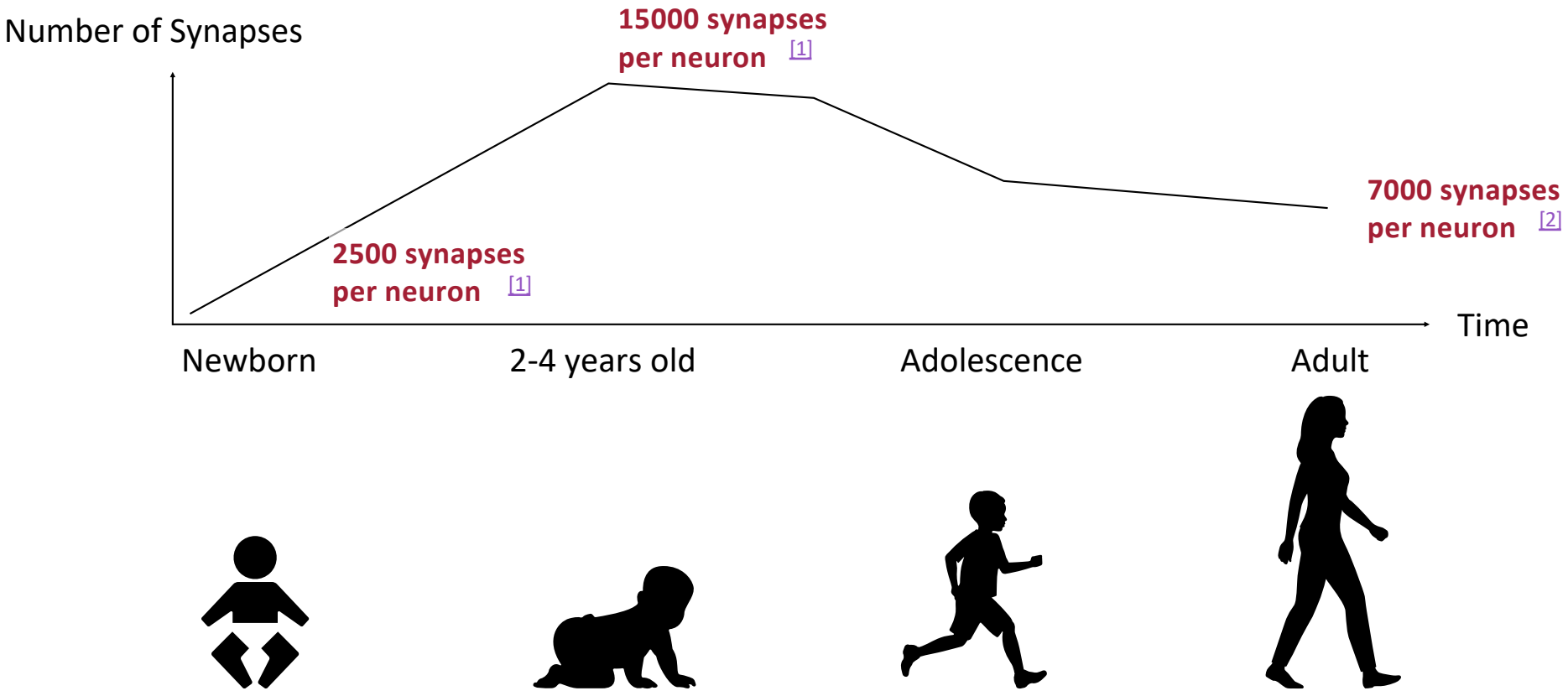
Neural Network Pruning

- **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

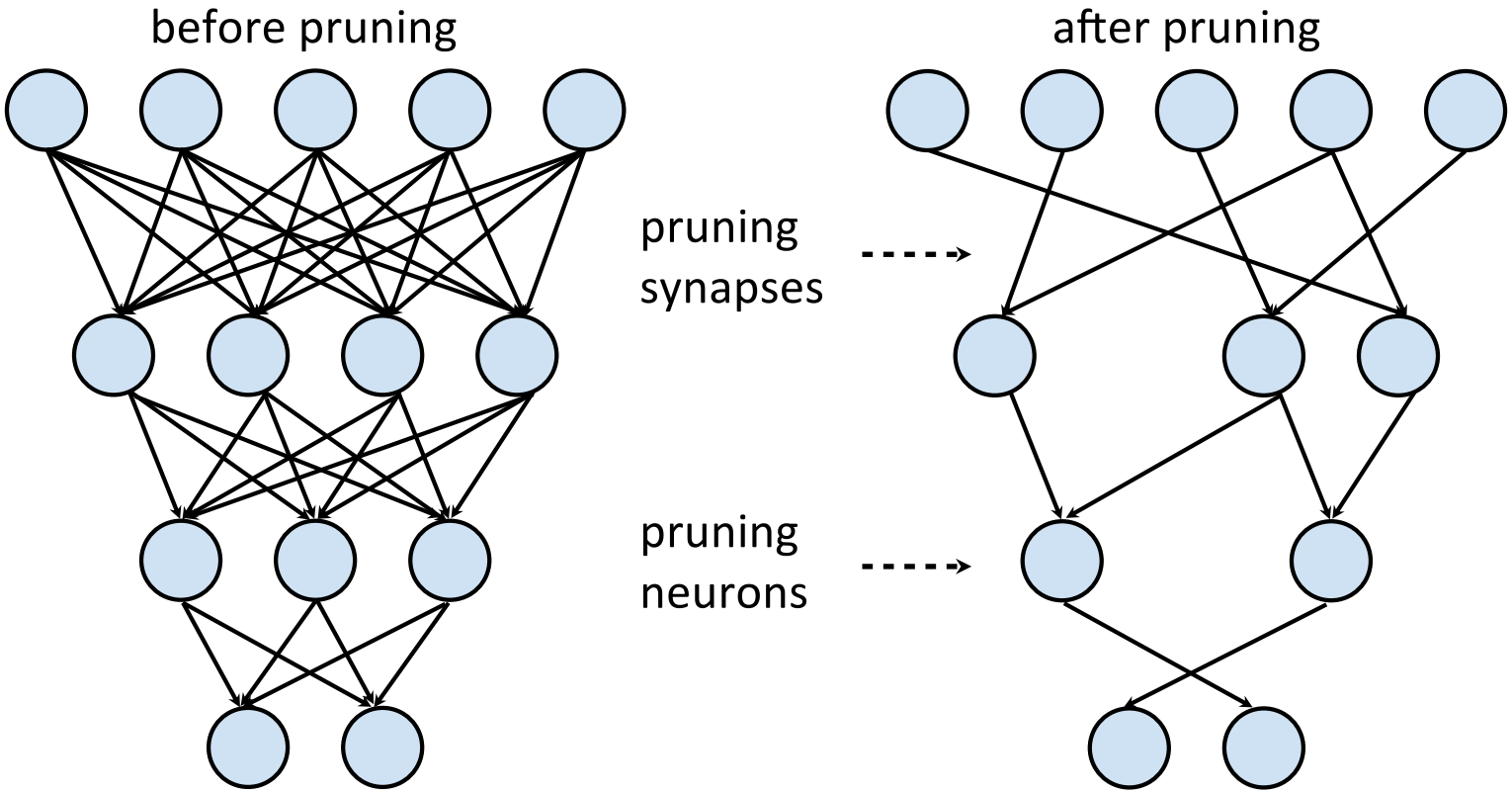


Do We Have Brain to Spare? [Drachman DA, Neurology 2004]
Peter Huttenlocher (1931–2013) [Walsh, C. A., Nature 2013]
https://efficientml.ai

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

Neural Network Pruning

- Make neural network smaller by removing synapses and neurons

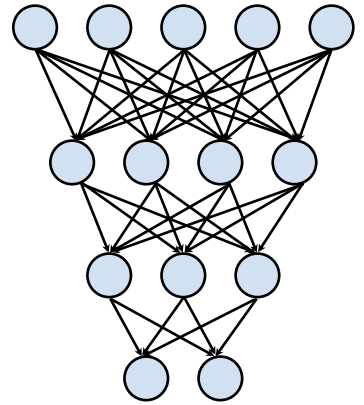


Optimal Brain Damage [LeCun *et al.*, NeurIPS 1989]

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

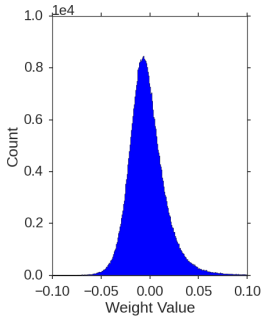
Neural Network Pruning

Make neural network smaller by removing synapses and neurons



Train Connectivity

Baseline accuracy



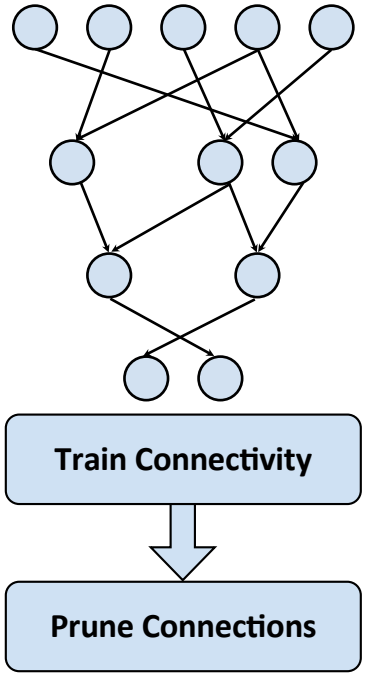
we can reduce weights which are close to zero

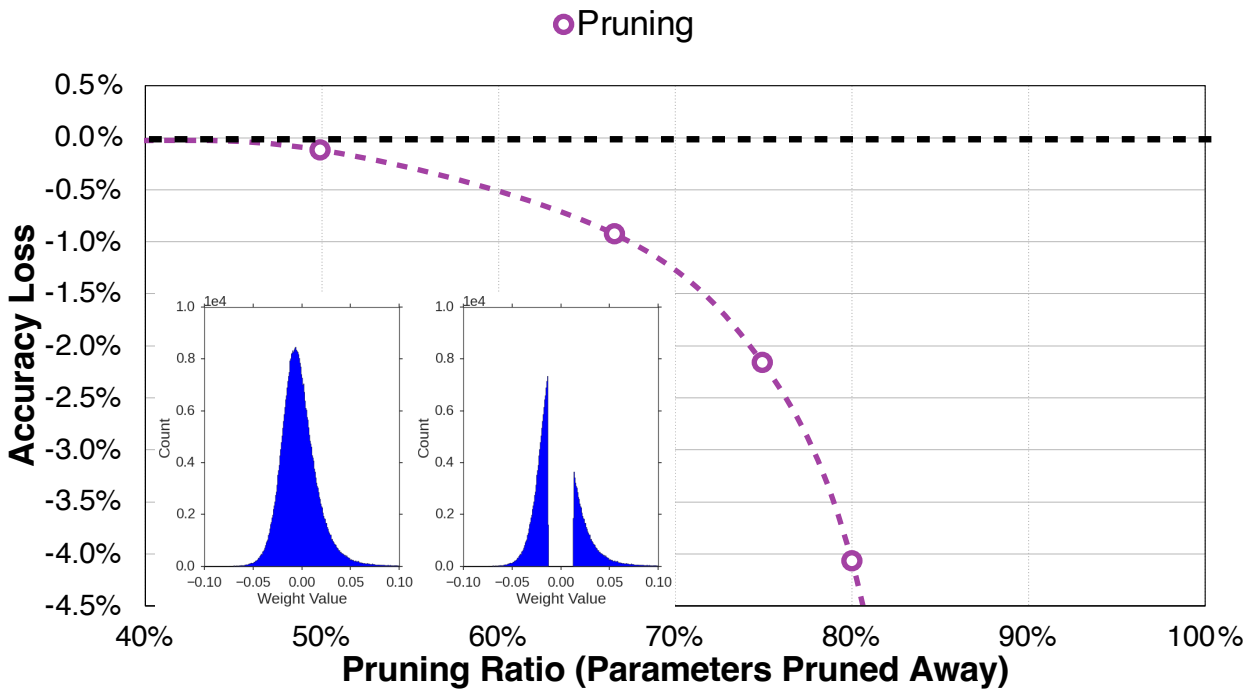
this is one of the various methods to prune

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



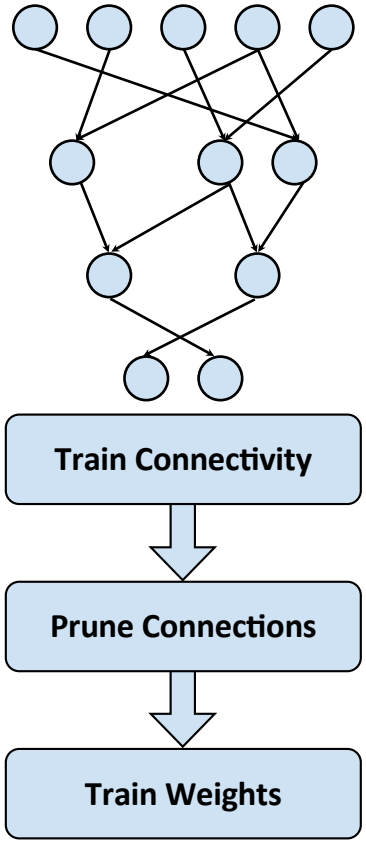


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

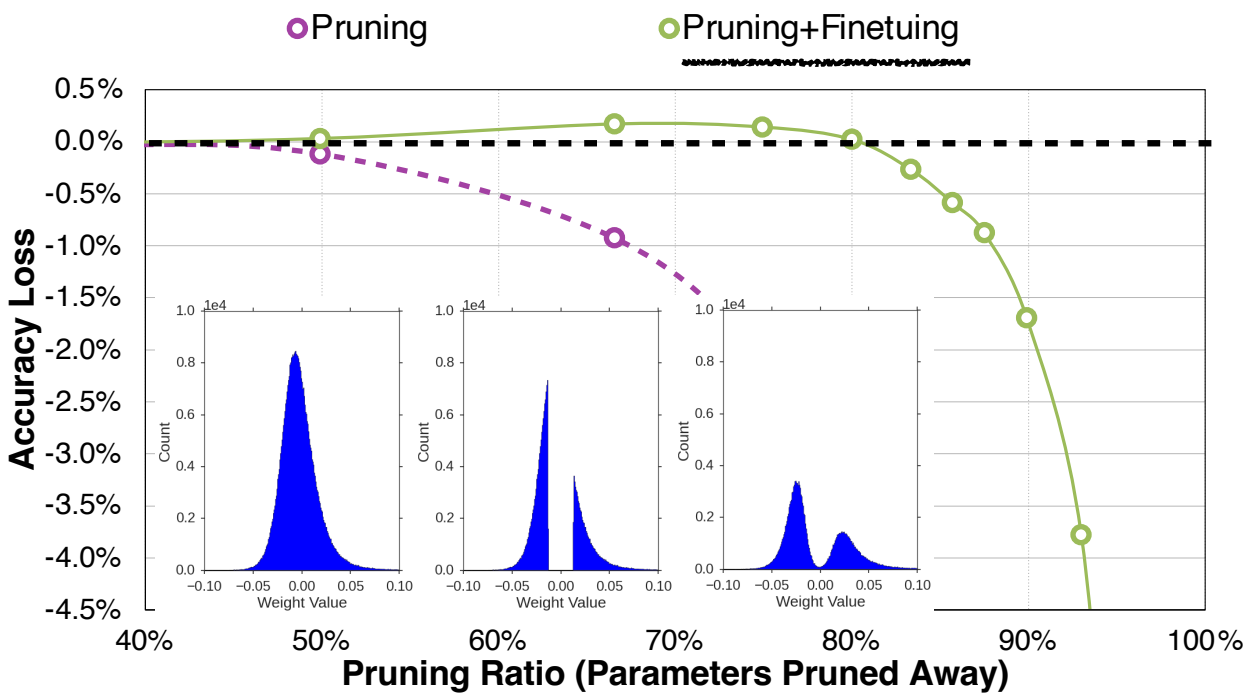
Important thing - weights are centered around zero

Neural Network Pruning

Make neural network smaller by removing synapses and neurons



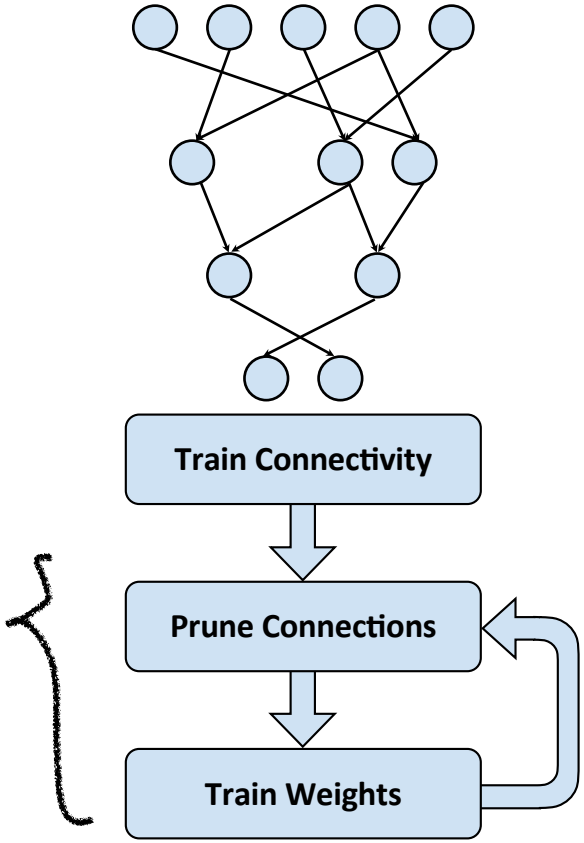
we fine-tune it small learning rate

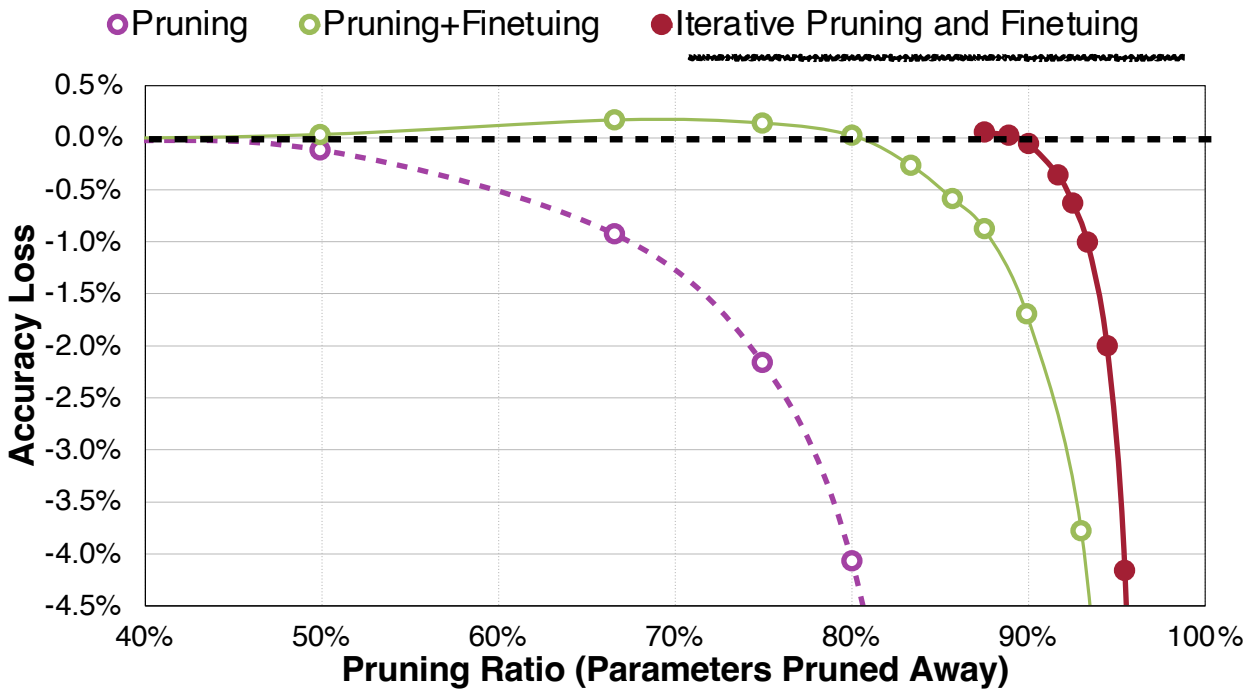


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

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

why is compute doesn't reduce by same factor as pruning

Make neural network smaller by removing synapses and neurons

Neural Network	#Parameters			MACs compute
	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 ×

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.

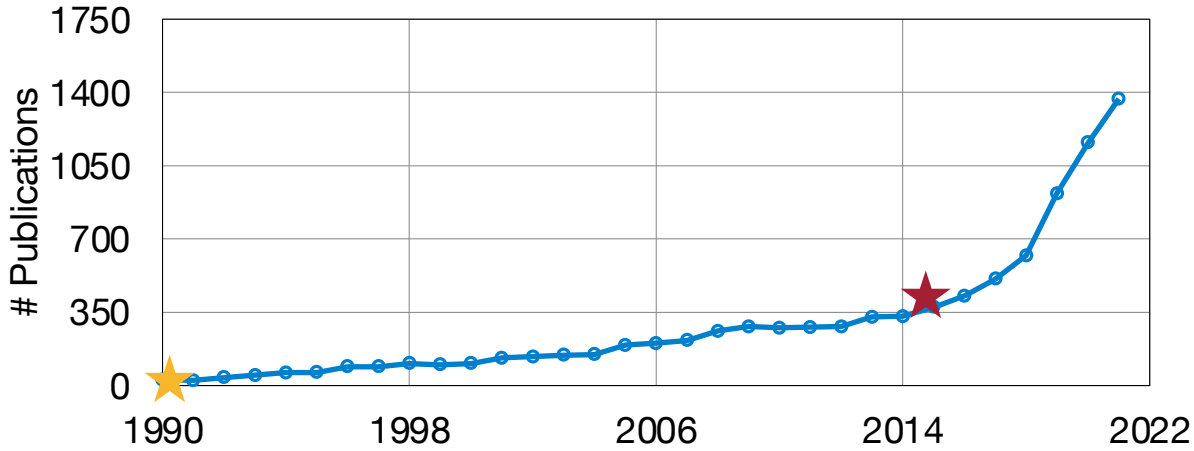
Pruned 90%: a man in a wetsuit is riding a wave on a beach.

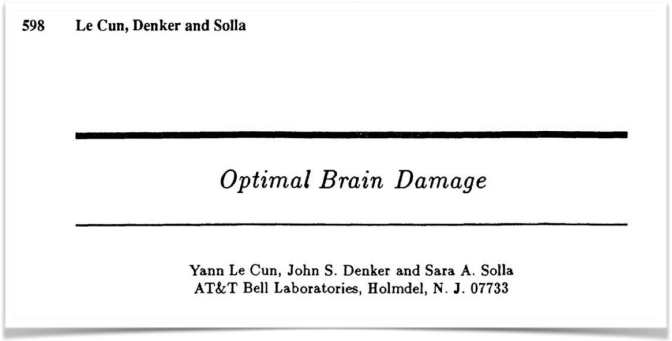
Baseline: a soccer player in red is running in the field.

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

Neural Network Pruning

Make neural network smaller by removing synapses and neurons

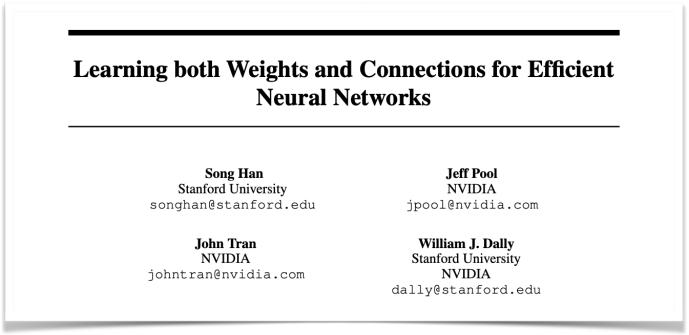




598 Le Cun, Denker and Solla

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

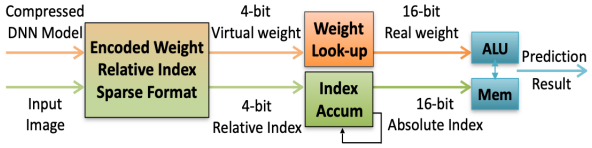
John Tran
NVIDIA
johntran@nvidia.com

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

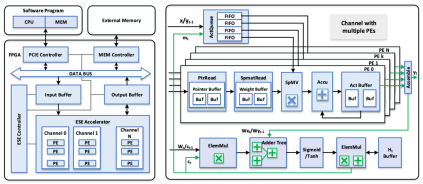
Souce: Dimension.ai

Pruning in the Industry

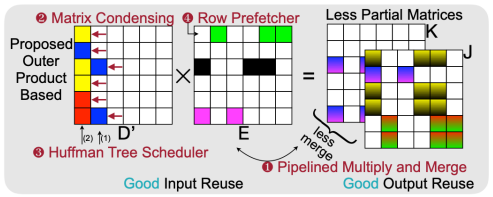
Hardware support for sparsity



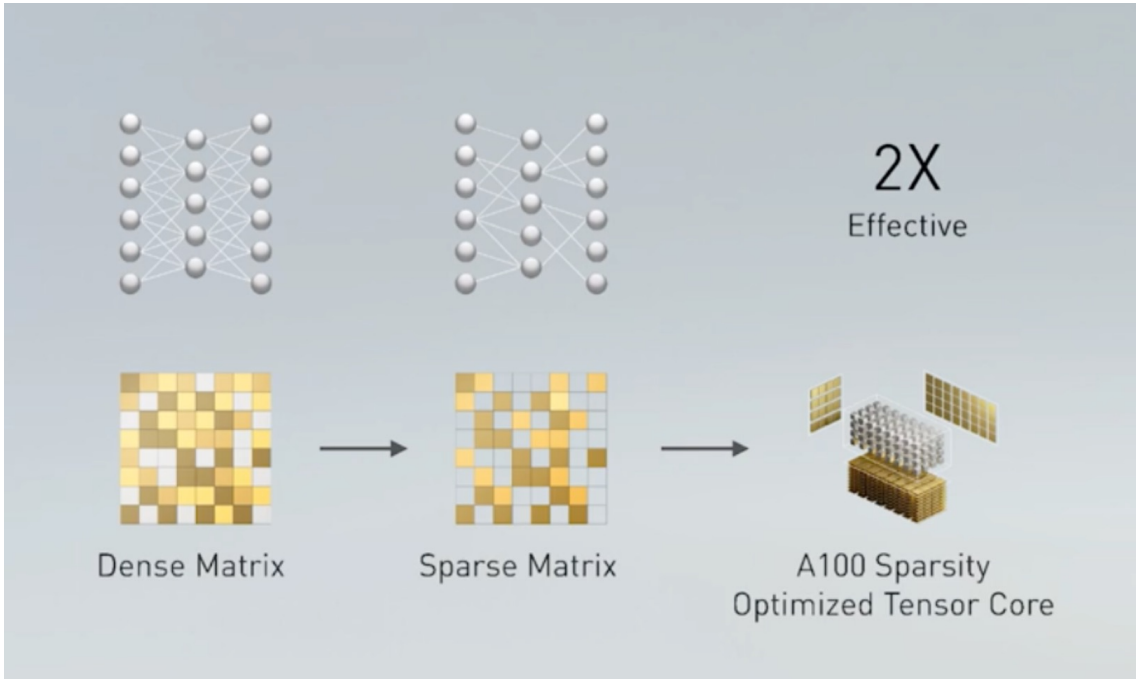
EIE [Han *et al.*, ISCA 2016]



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



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

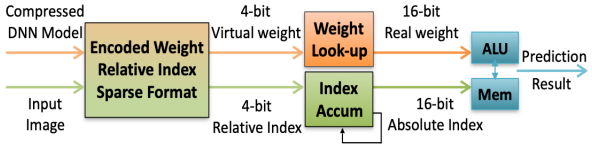


2:4 sparsity in A100 GPU
2X peak performance, 1.5X measured BERT speedup

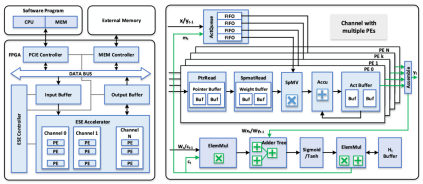
https://efficientml.ai

Pruning in the Industry

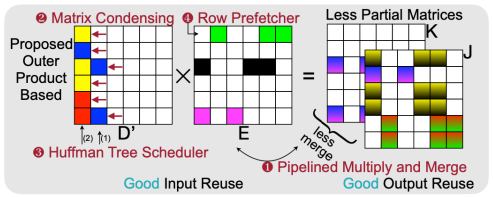
Hardware support for sparsity



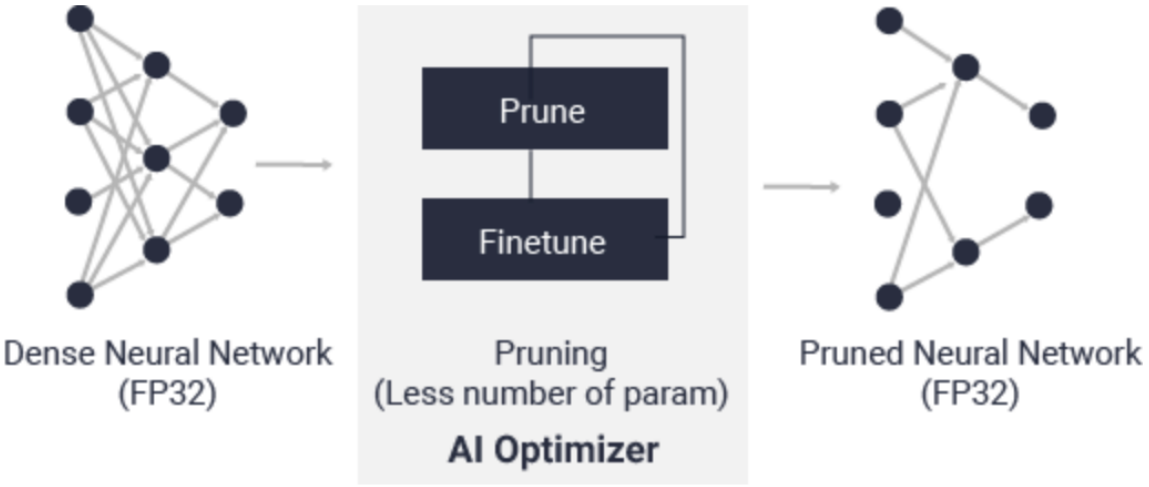
EIE [Han *et al.*, ISCA 2016]



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



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

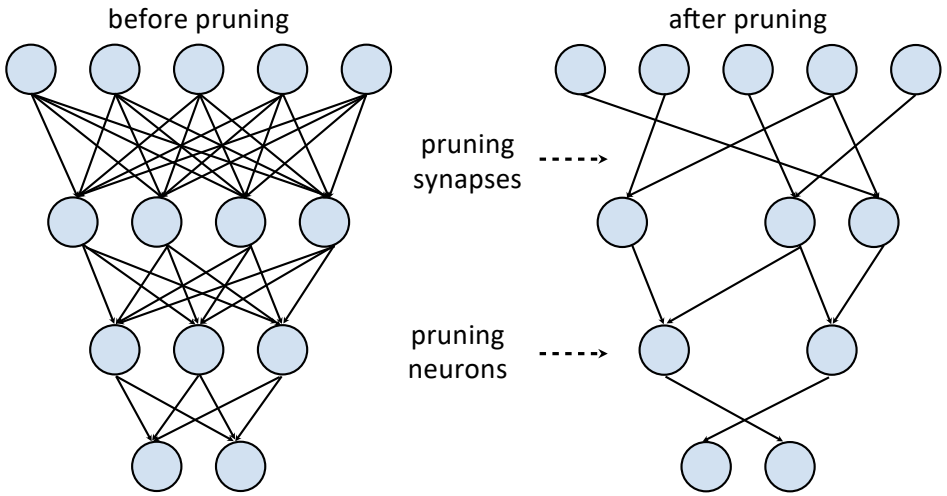


Reduce model complexity by 5x to 50x with minimal accuracy impact

https://efficientml.ai

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

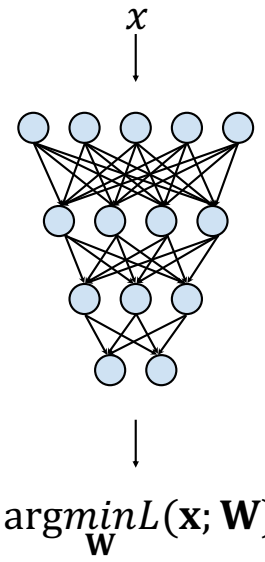
- In general, we could formulate the pruning as follows:

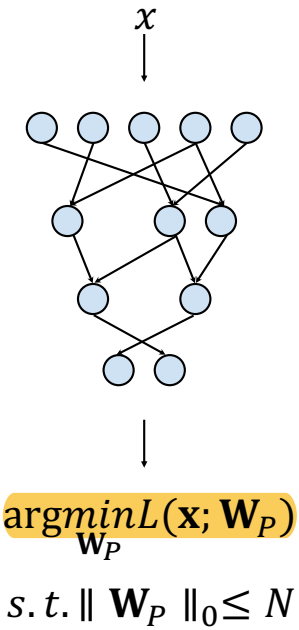
$$\arg\min_{\mathbf{W}_P} L(\mathbf{x}; \mathbf{W}_P)$$

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

number of non-zero weights

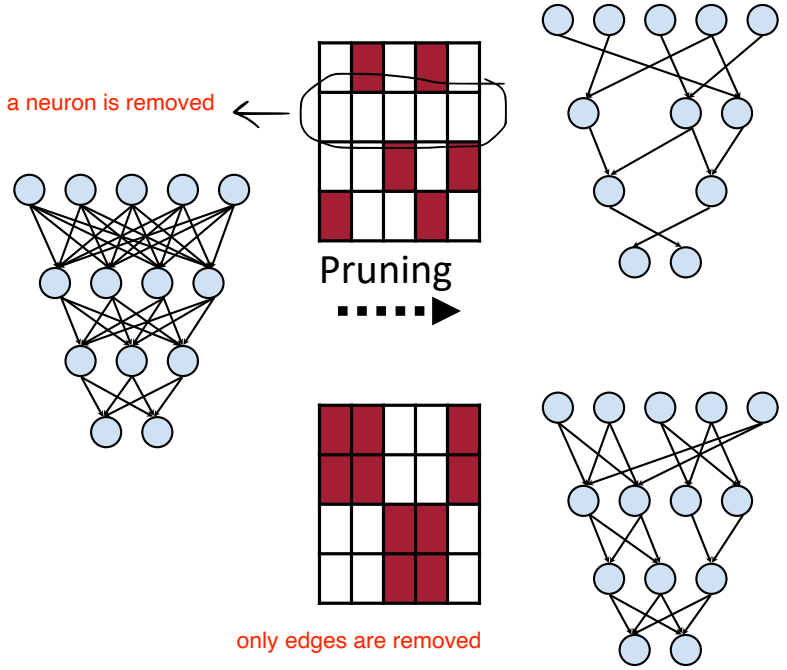
  - L represents the objective function for neural network training;
  - $\mathbf{x}$ is input, $\mathbf{W}$ is original weights, $\mathbf{W}_P$ is pruned weights;
  - $\| \mathbf{W}_p \|_0$ calculates the #nonzeros in W_P, and N is the target #nonzeros.





Neural Network Pruning

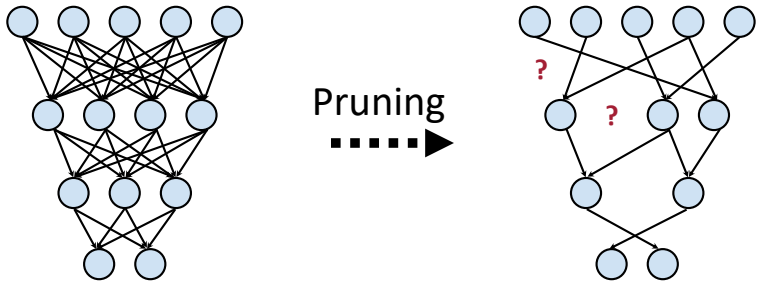
- **Introduction to Pruning**
  - What is pruning?
  - How should we formulate pruning?
- **Determine the Pruning Granularity**
  - In what pattern should we prune the neural network?
- **Determine the Pruning Criterion**
  - What synapses/neurons should we prune?
- **Determine the Pruning Ratio**
  - 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?

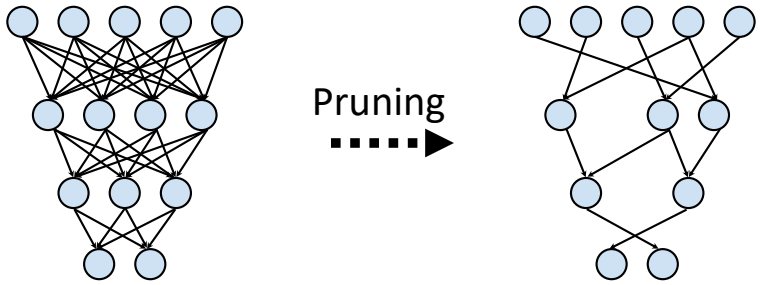


which synapses?
which neurons?

- 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%?

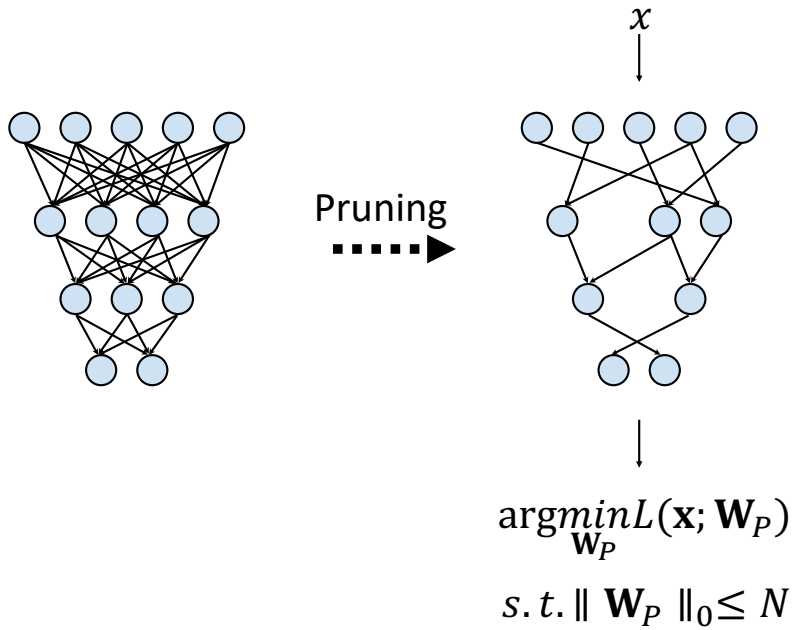
prune 50%?

prune 70%?

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

Neural Network Pruning

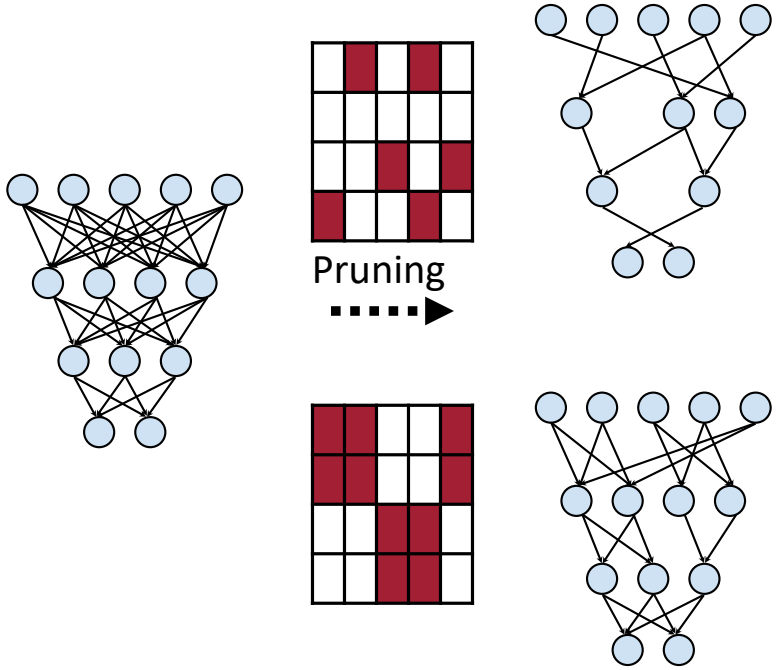
- **Introduction to Pruning**
  - What is pruning?
  - How should we formulate pruning?
- **Determine the Pruning Granularity**
  - In what pattern should we prune the neural network?
- **Determine the Pruning Criterion**
  - What synapses/neurons should we prune?
- **Determine the Pruning Ratio**
 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



- 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?



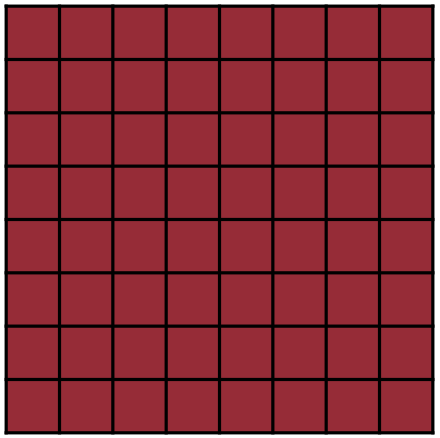
- Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Section 2: Pruning Granularity

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

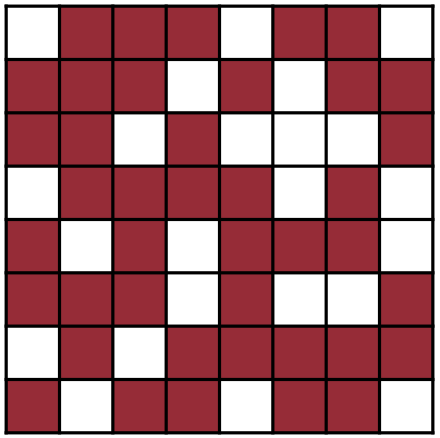
Pruning at Different Granularities

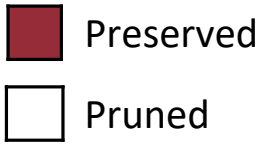
A simple example of 2D weight matrix



Pruning at Different Granularities

A simple example of 2D weight matrix





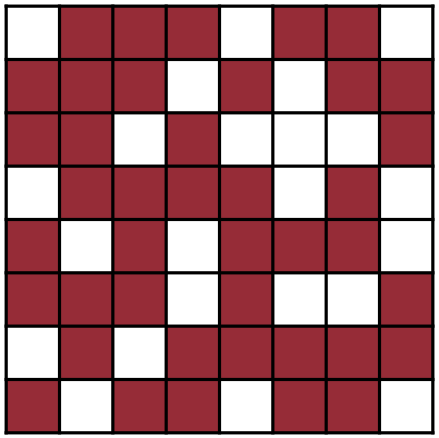
Fine-grained/Unstructured

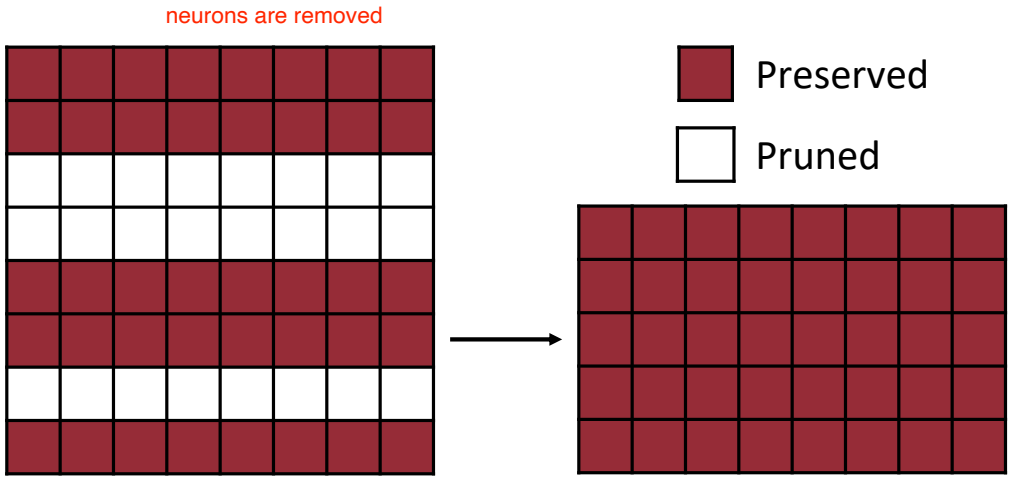
- More flexible pruning index choice
- Hard to accelerate (irregular data expression)

Bad for 

Pruning at Different Granularities

A simple example of 2D weight matrix





Fine-grained/Unstructured

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

Bad for 

Coarse-grained/Structured

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

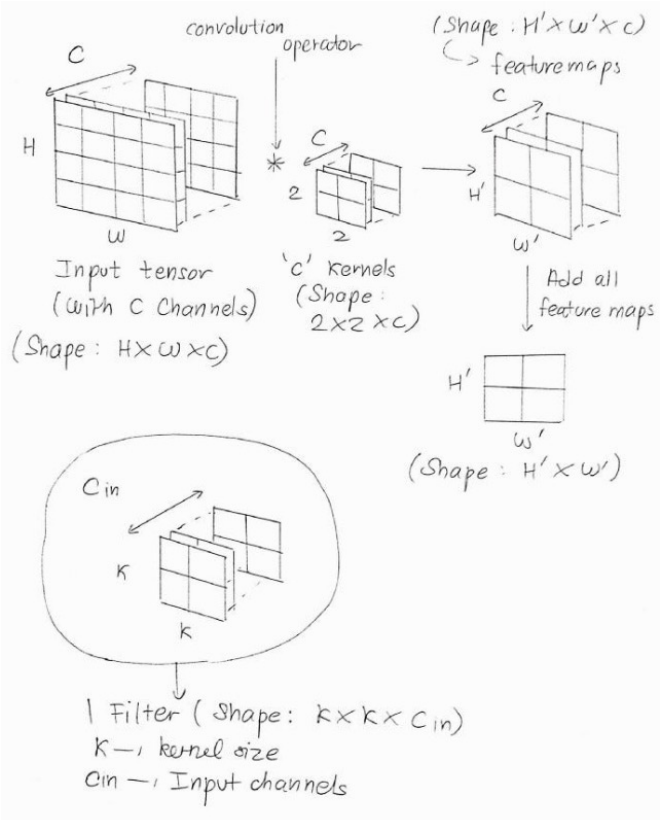
Pruning at Different Granularities

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) ex - R G B
  - c_o: output channels (or filters) # filters
  - k_h: kernel size height
  - k_w: kernel size width

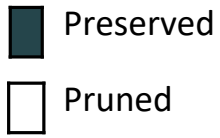
- The 4 dimensions give us more choices to select pruning granularities

Filter

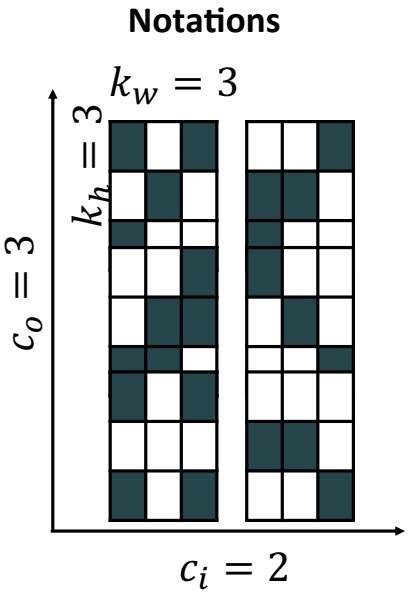

convolution operator
(Shape : H' × w' × c)
feature maps
C
H
w
Input tensor
(with C Channels)
(Shape : H × w × C)
C
2
2
'c' kernels
(Shape : 2×2×C)
C
H'
w'
Add all feature maps
H'
w'
(Shape : H' × w')
Cin
K
K
1 Filter (Shape : K × K × Cin)
K — kernel size
Cin — Input channels

Pruning at Different Granularities

The case of convolutional layers



- Some of the commonly used pruning granularities



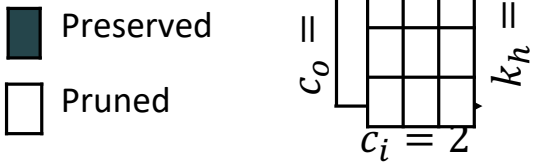
- The weights of convolutional layers have 4 dimensions $[c_o, c_i, k_h, k_w]$:
  - c_i: input channels (or channels)
  - c_o: output channels (or filters)
  - k_h: kernel size height
  - k_w: kernel size width
- The 4 dimensions give us more choices to select pruning granularities

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

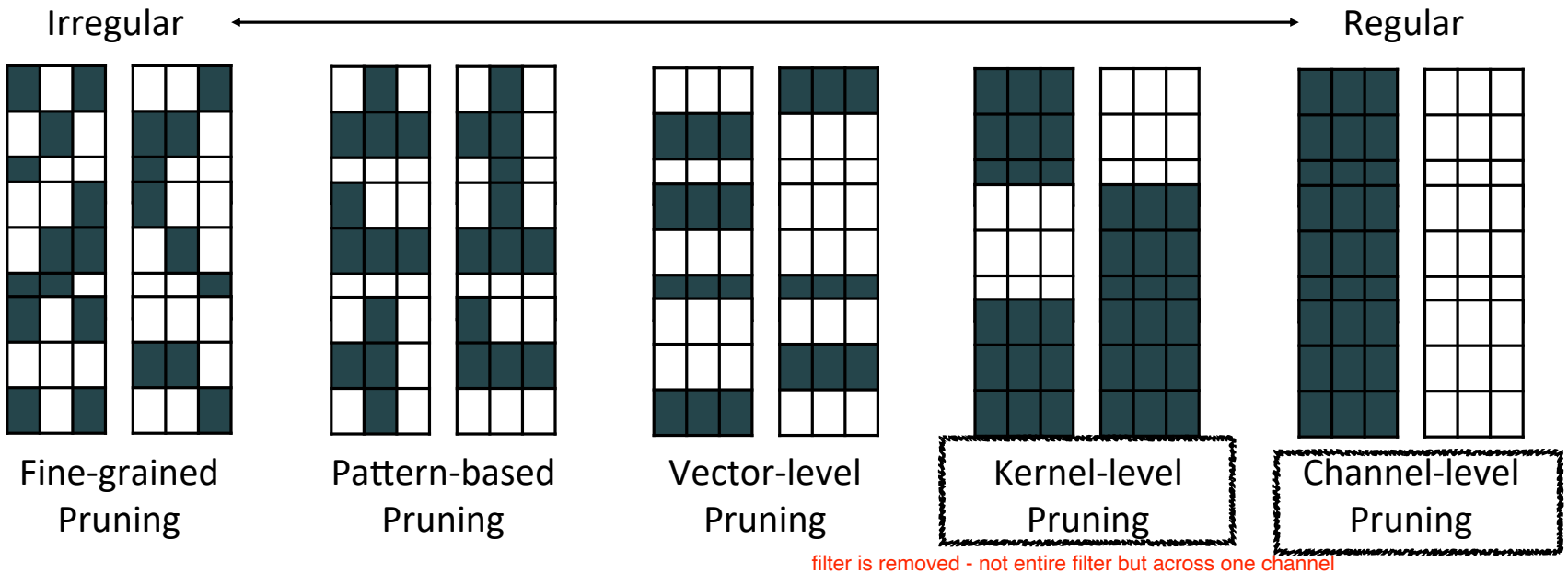
Pruning at Different Granularities

The case of convolutional layers

- Some of the commonly used pruning granularities



filter dimmension - 3x3x2



like Tetris :)

filter is removed - not entire filter but across one channel

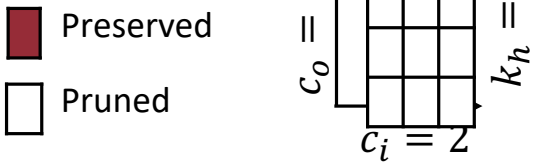
Exploring the granularity of sparsity in convolutional neural networks [Mao *et al.*, CVPR-W]

a channel is removed across all the filters

Pruning at Different Granularities

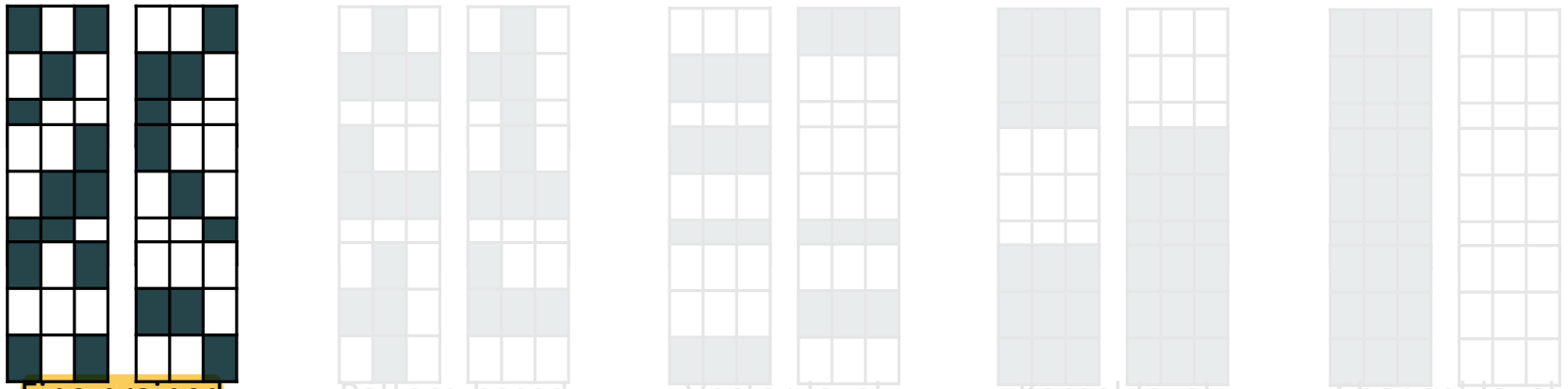
The case of convolutional layers

- Some of the commonly used pruning granularities



Pros? **Cons?**

Irregular ⟷ Regular



Fine-grained Pruning

Pattern-based Pruning

Vector-level Pruning

Kernel-level Pruning

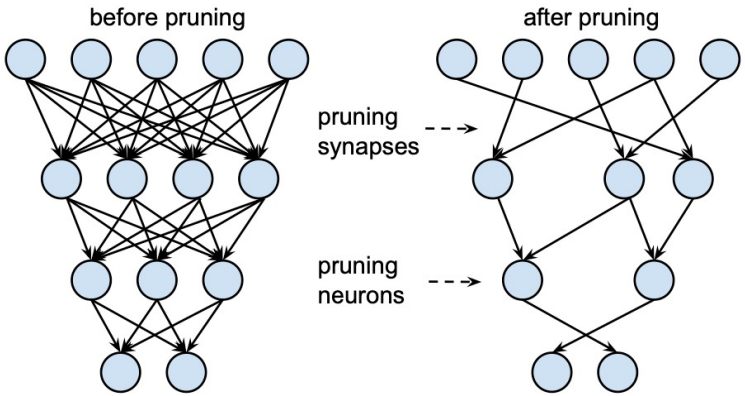
Channel-level Pruning

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**×

Pruning at Different Granularities

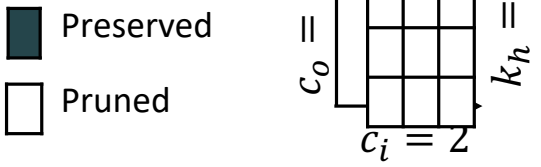
Let's look into some cases

- **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

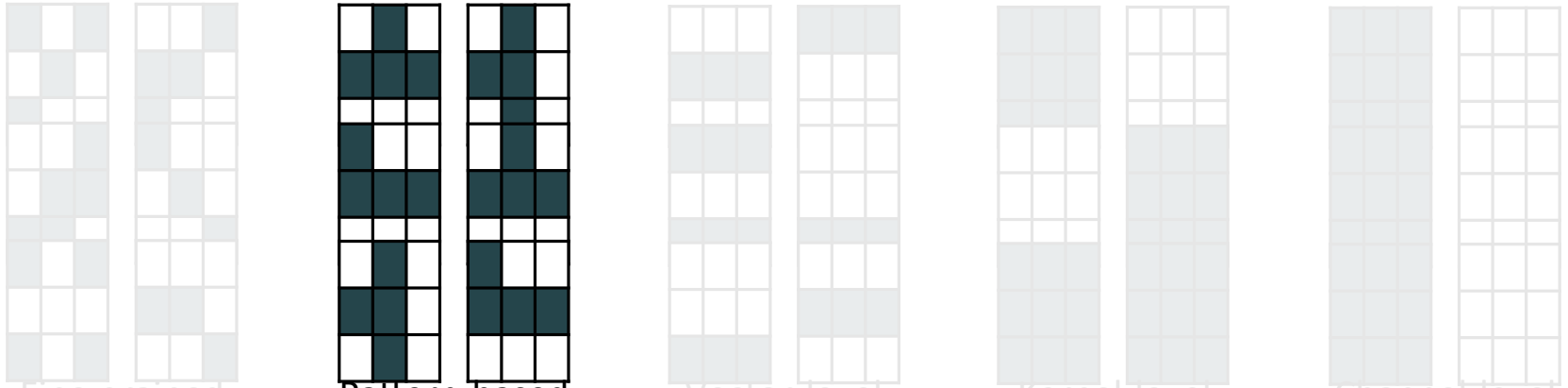
The case of convolutional layers

- Some of the commonly used pruning granularities



Pros? **Cons?**

Irregular ⟷ Regular



Fine-grained Pruning | Pattern-based Pruning | Vector-level Pruning | Kernel-level Pruning | Channel-level Pruning

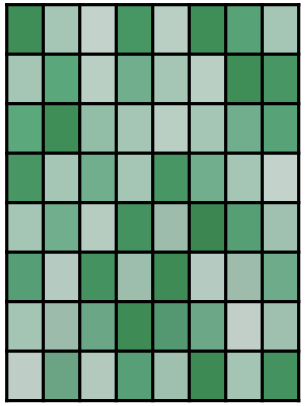
Exploring the granularity of sparsity in convolutional neural networks [Mao *et al.*, CVPR-W]

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

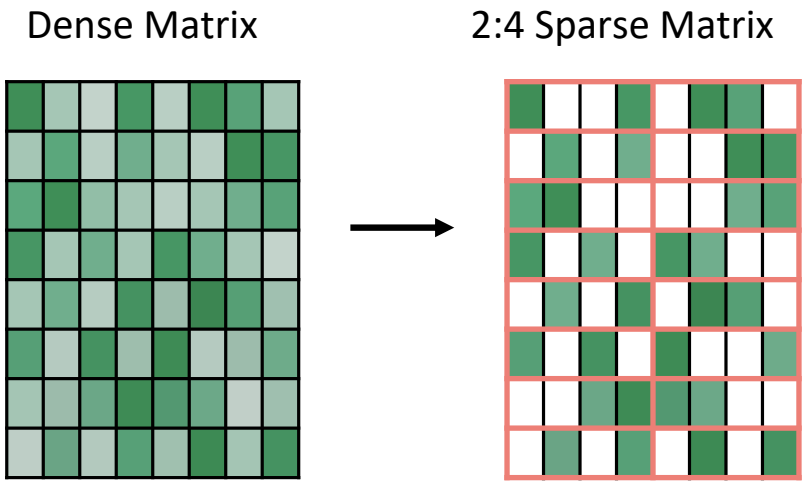


Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT

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) (available in A100 GPUs)

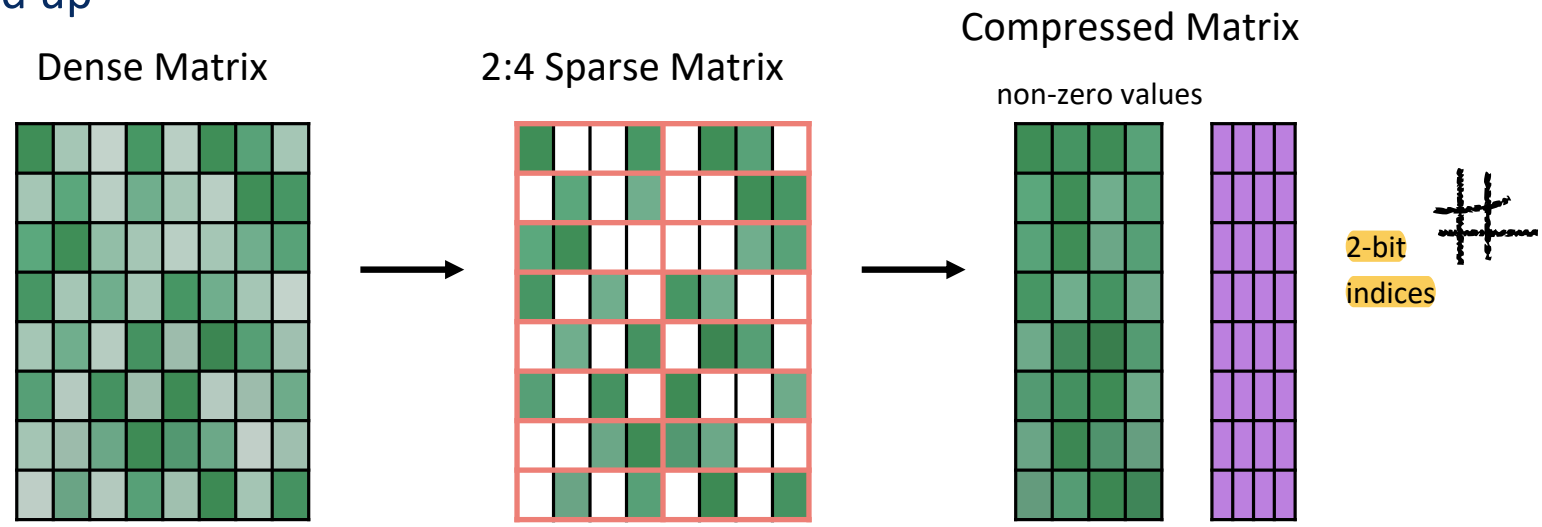


Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT

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)

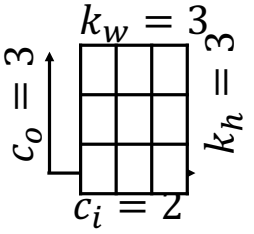
Network	Data Set	Metric	Dense FP16	Sparse FP16
ResNet-50	ImageNet	Top-1	76.1	76.2
ResNeXt-101_32x8d	ImageNet	Top-1	79.3	79.3
Xception	ImageNet	Top-1	79.2	79.2
SSD-RN50	COCO2017	bbAP	24.8	24.8
MaskRCNN-RN50	COCO2017	bbAP	37.9	37.9
FairSeq Transformer	EN-DE WMT'14	BLEU	28.2	28.5
BERT-Large	SQuAD v1.1	F1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

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

Pruning at Different Granularities

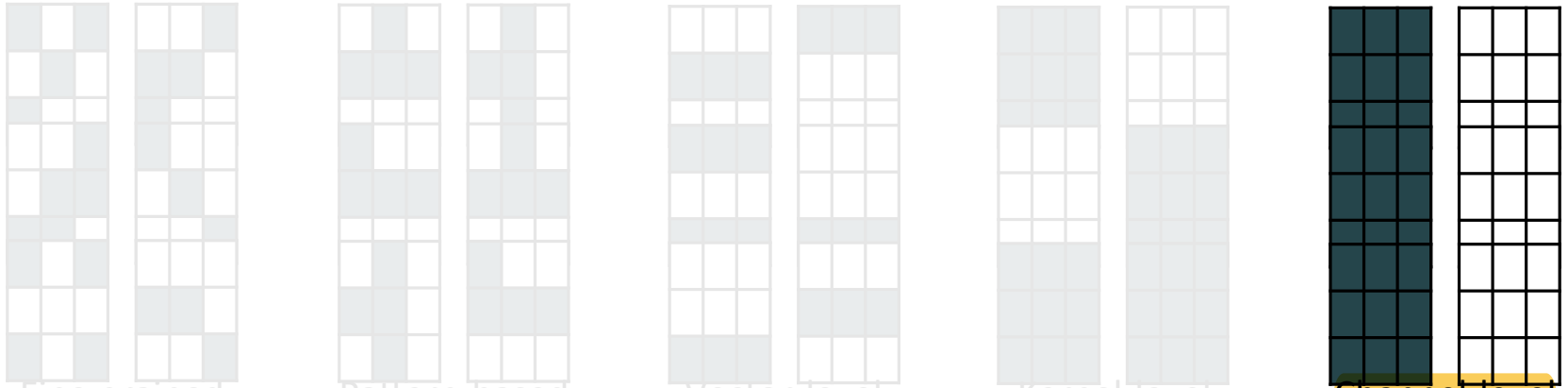
The case of convolutional layers

- Some of the commonly used pruning granularities



Pros? **Cons?**

Irregular ⟵ ⟶ Regular



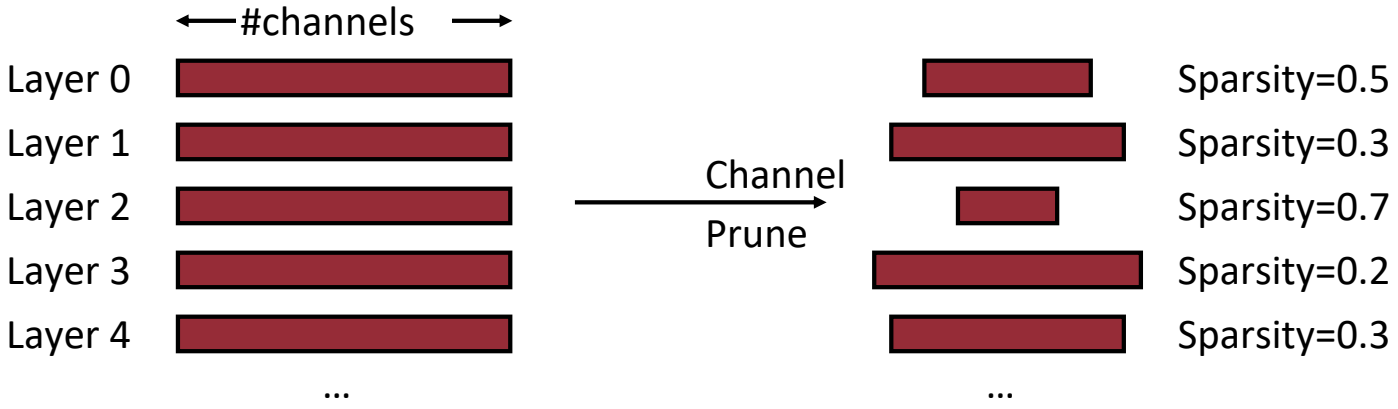
Fine-grained Pruning | Pattern-based Pruning | Vector-level Pruning | Kernel-level Pruning | Channel-level Pruning

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
  - Most popular on CPUs and off the shelf hardware

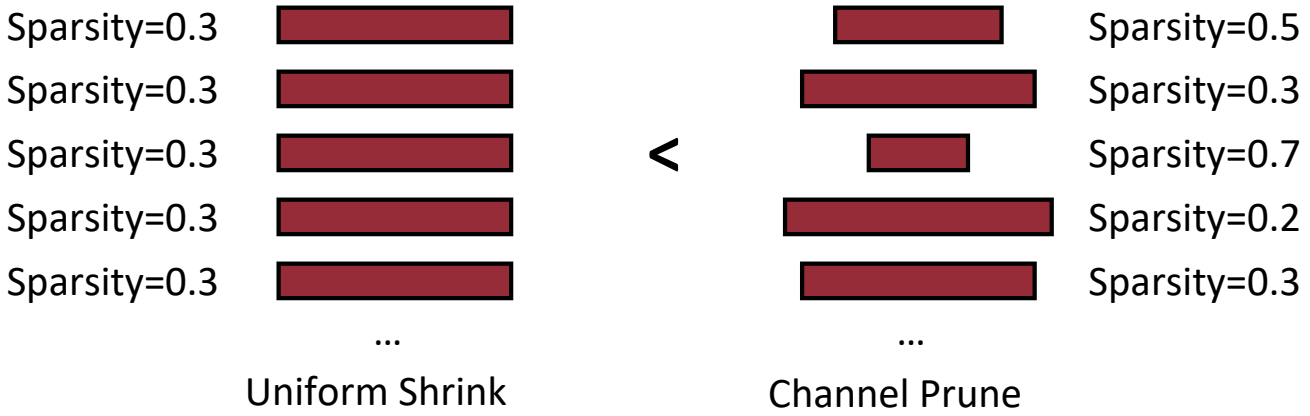


Pruning at Different Granularities

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
  - Most popular on CPUs and off the shelf hardware

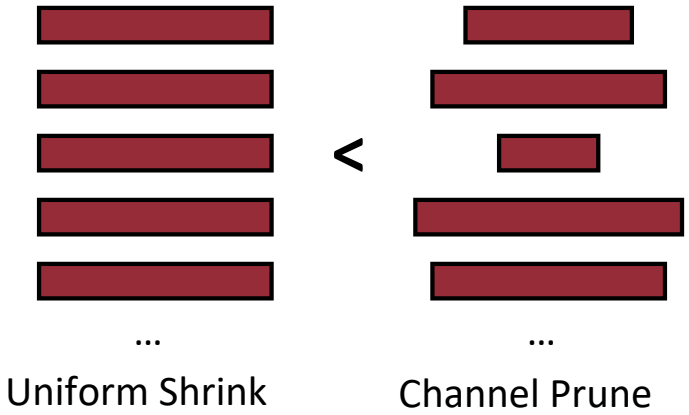
We will later discuss how to find sparsity ratios

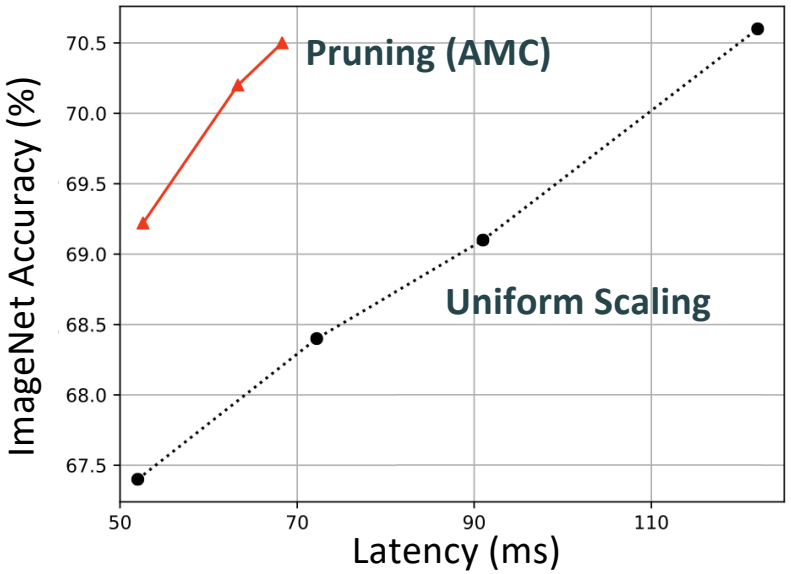


Pruning at Different Granularities

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

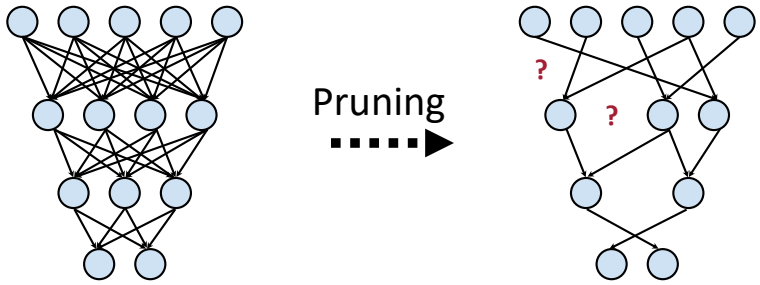




AMC: Automl for Model Compression and Acceleration on Mobile Devices [He *et al.*, ECCV 2018]

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?



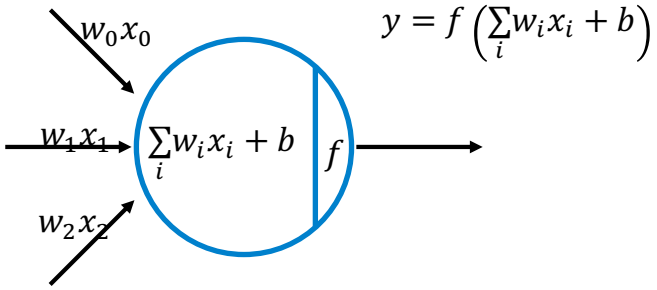
- Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

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

$$f(\cdot) = ReLU(\cdot), W = [10, -8, 0.1]$$
$$y = ReLU(10x_0 - 8x_1 + 0.1x_2)$$

- If only one weight will be removed, which one? Why?
- If only one weight will be kept, which one? Why?

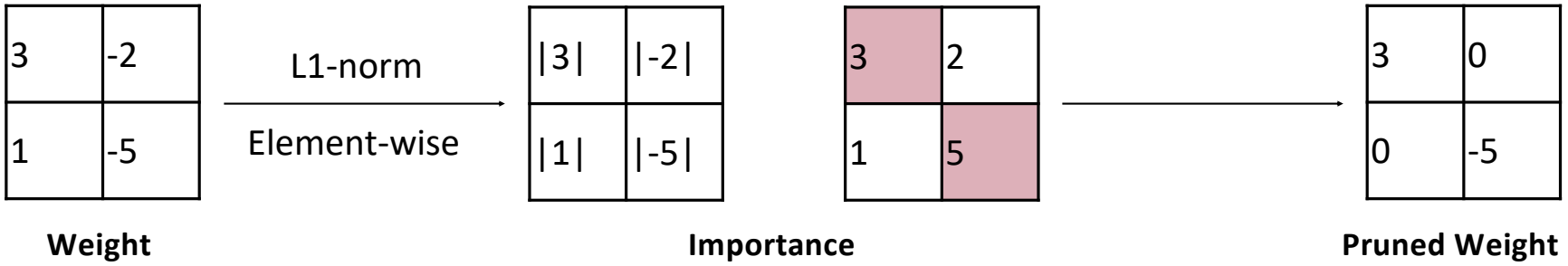
Magnitude-based Pruning

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]

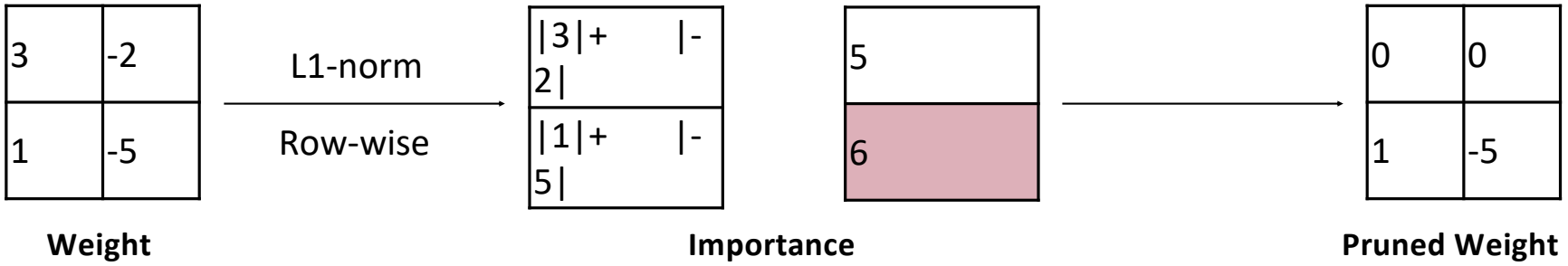
Magnitude-based Pruning

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,

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

- **Example**



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

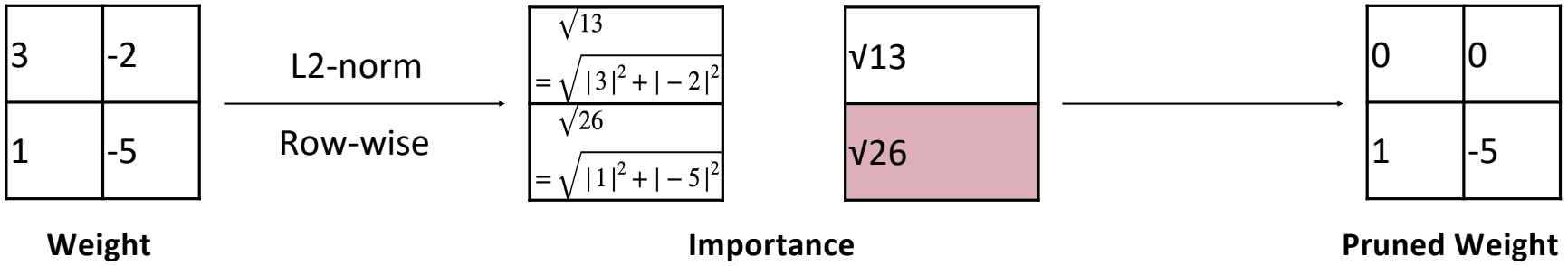
Magnitude-based Pruning

A heuristic pruning criterion

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

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

- **Example**



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

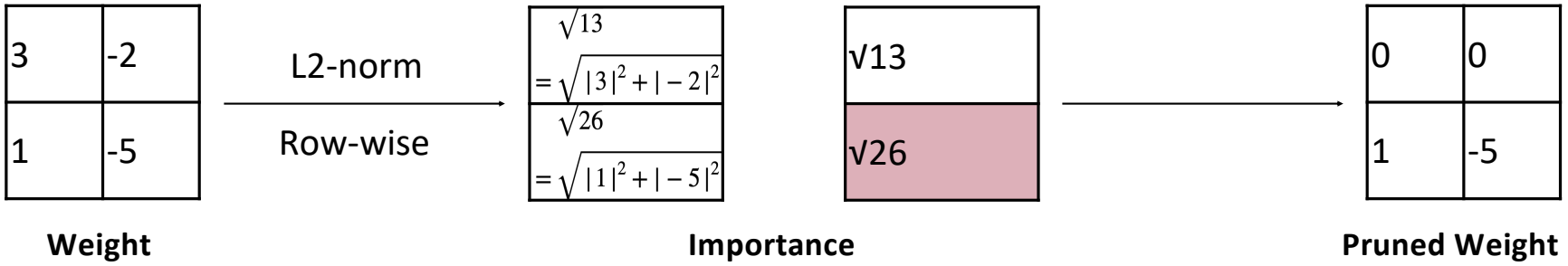
Magnitude-based Pruning

A heuristic pruning criterion

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

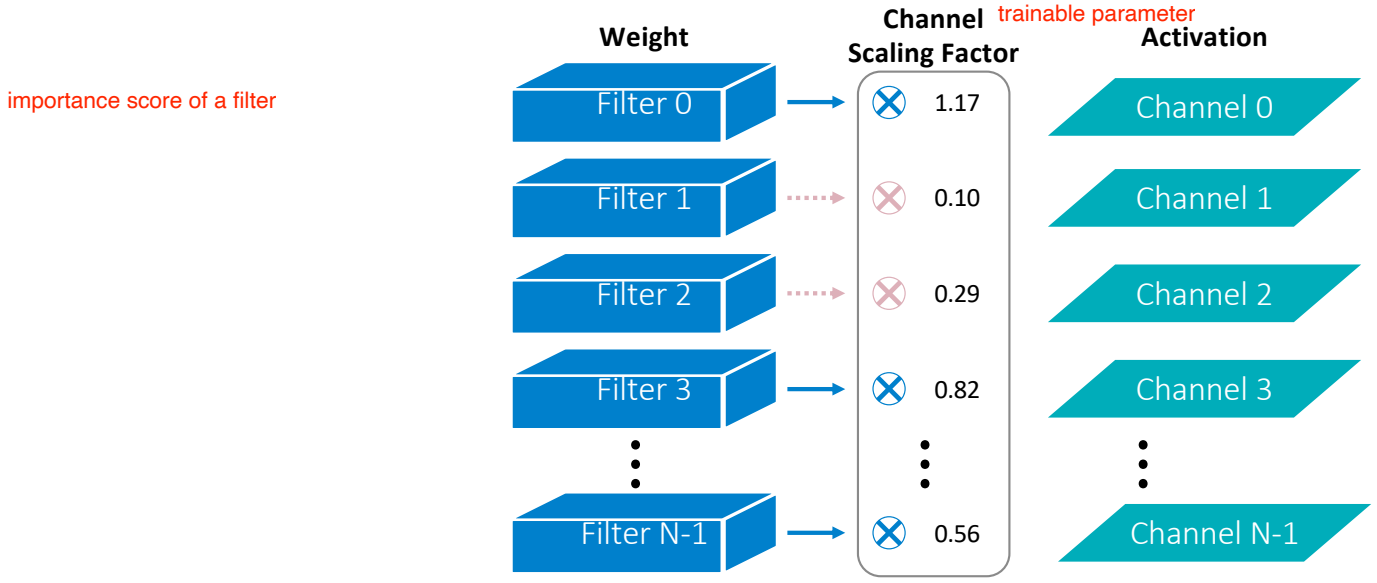


Learning Structured Sparsity in Deep Neural Networks [Wen *et al.,* NeurIPS 2016]

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
- The filters/output channels with small scaling factor magnitude will be pruned

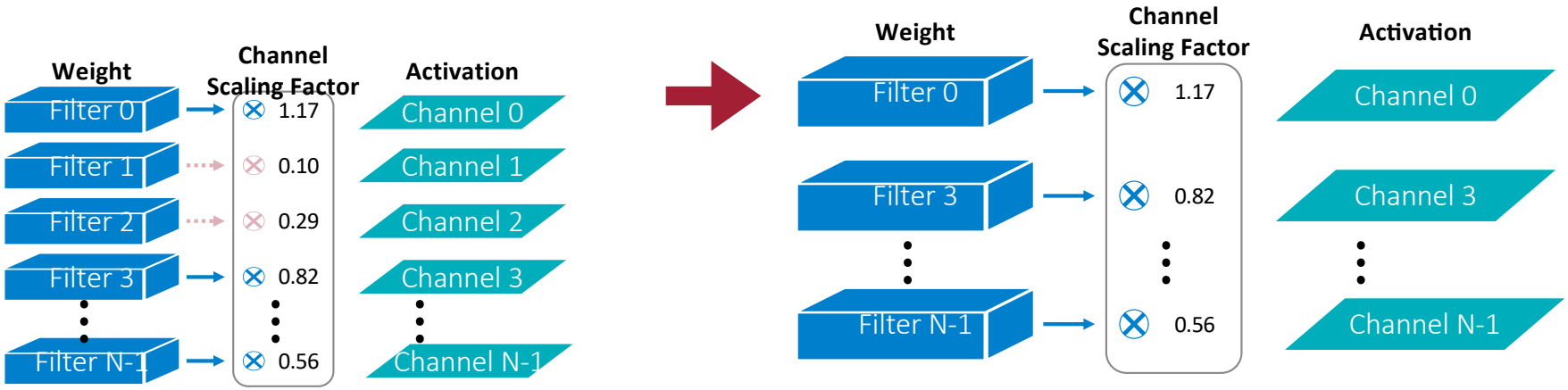


Learning Efficient Convolutional Networks through Network Slimming [Liu *et al.*, ICCV 2017]

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 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

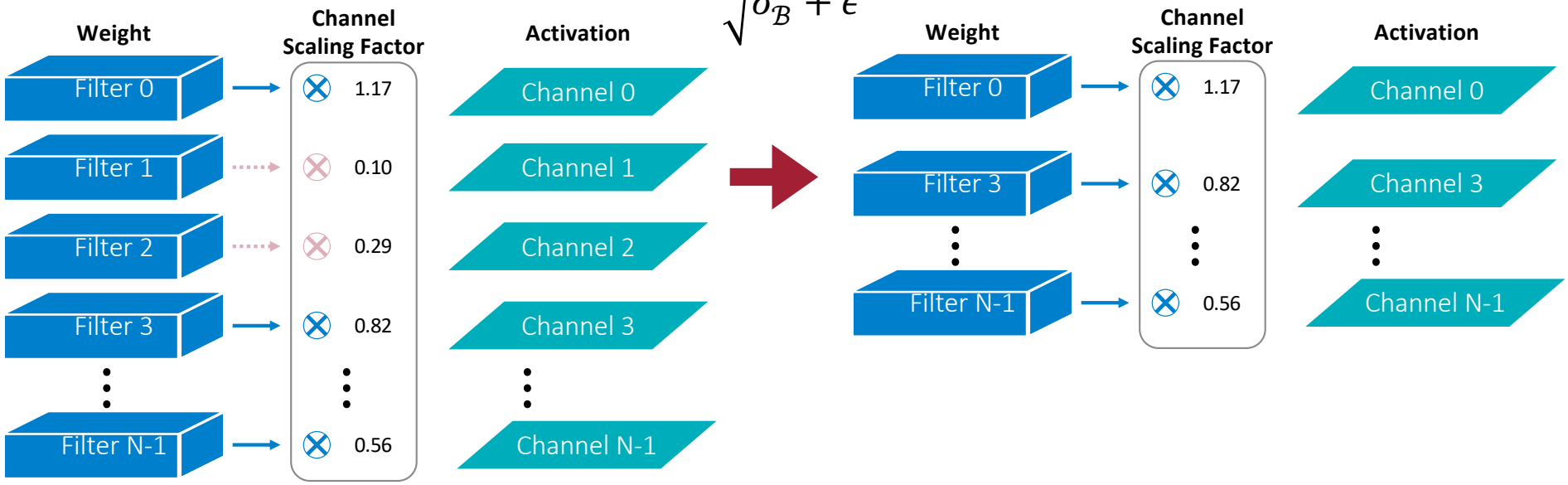
Pruning criterion for filter pruning

- 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

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

batch level statistics

gamma, beta -> trainable param



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

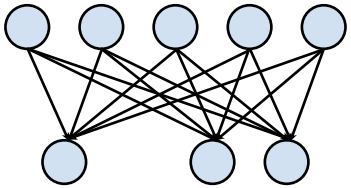
Selection of Neurons to Prune

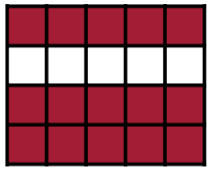
- When removing neurons from a neural network model,
  - ***the less useful*** the neurons being removed are,
  - the better the performance of pruned neural network is.

Recall: Neuron pruning is coarse-grained pruning indeed.

Weight Matrix

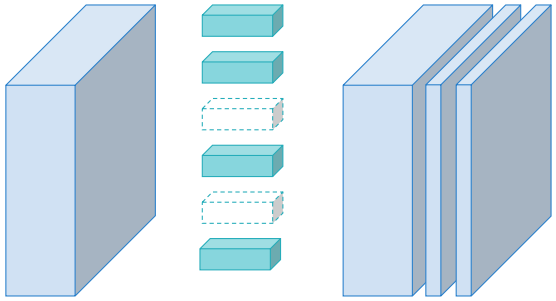
Neuron Pruning in Linear Layer

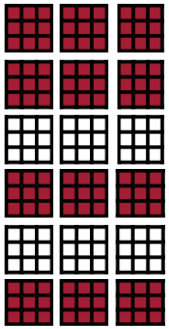




Input feature map Several filters Output channel map

Channel Pruning in Convolution Layer

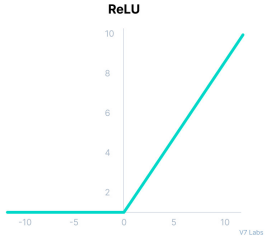




https://efficientml.ai

Percentage-of-Zero-Based Pruning (POZ)

- ReLU activation will generate zeros in the output activation.



Output Activations

Features of Image 1

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

Features of Image 2

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

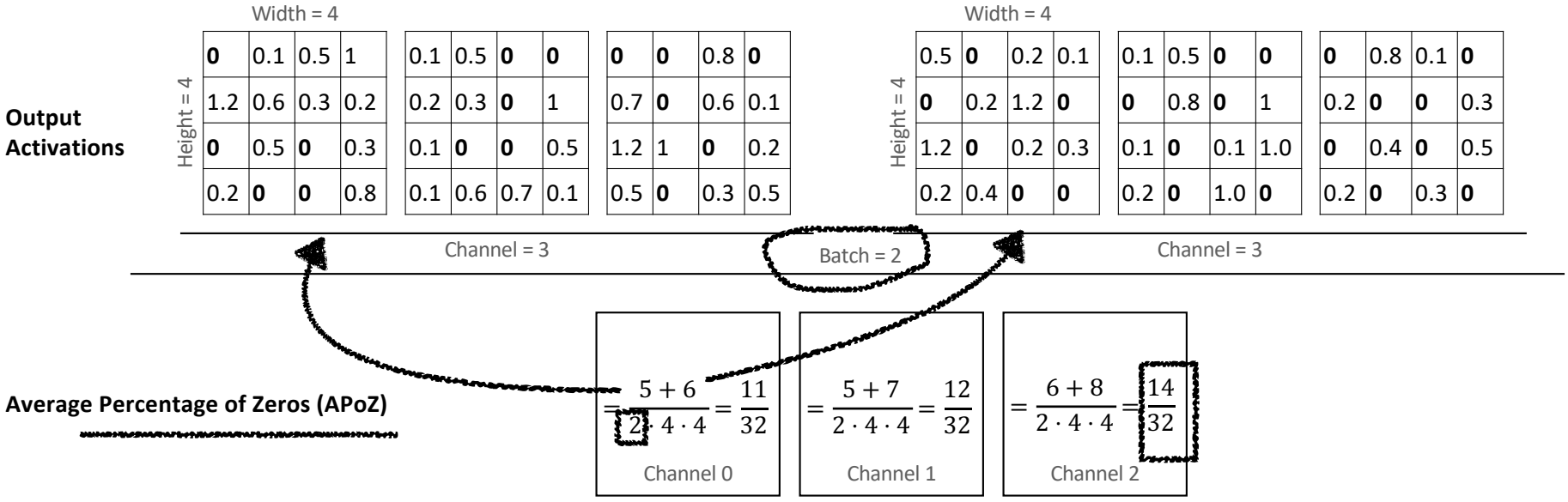
Batch = 2

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

https://efficientml.ai

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.



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

https://efficientml.ai

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 = 2

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

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

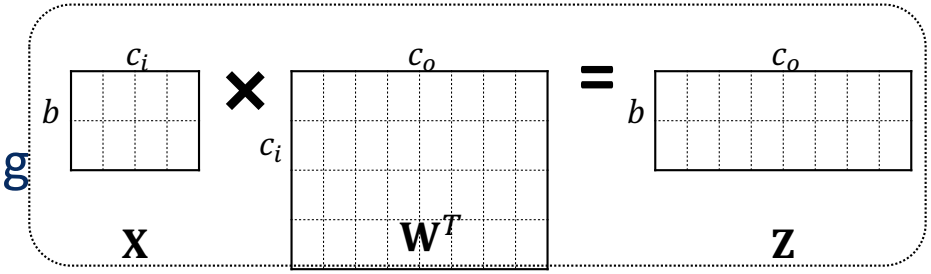
Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures [Hu *et al.,* ArXiv 2017]

https://efficientml.ai

Regression-based Pruning

Minimize reconstruction error of the corresponding layer's outputs

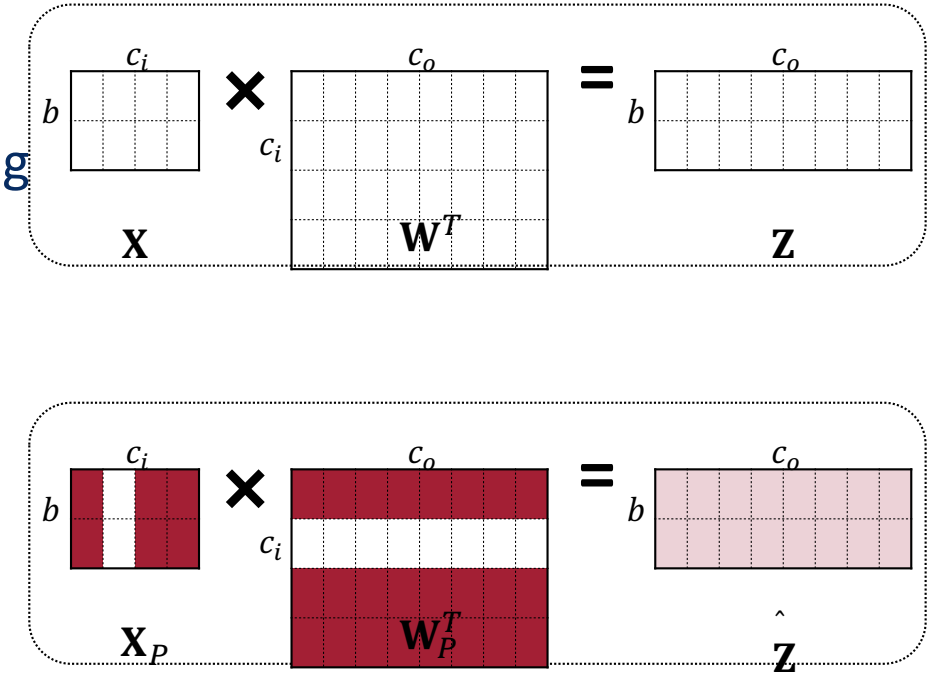
- Instead of considering the pruning error of the objective function $L(\mathbf{x}; \mathbf{W})$, regression-based pruning minimizes the reconstruction error of the corresponding layer's outputs.



Regression-based Pruning

Minimize reconstruction error of the corresponding layer's outputs

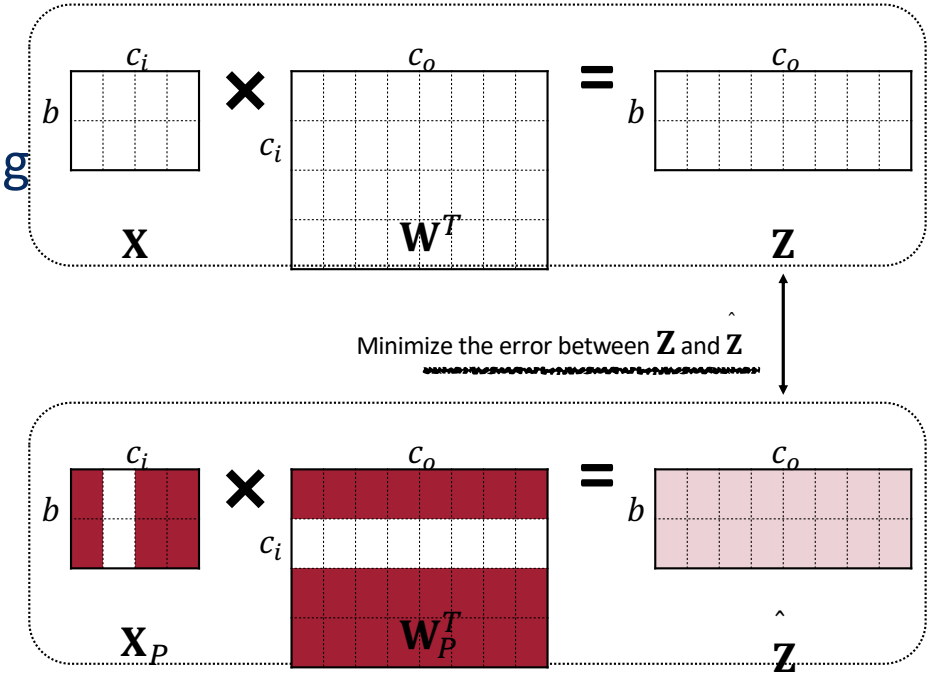
- Instead of considering the pruning error of the objective function $L(\mathbf{x}; \mathbf{W})$, regression-based pruning minimizes the reconstruction error of the corresponding layer's outputs.



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.

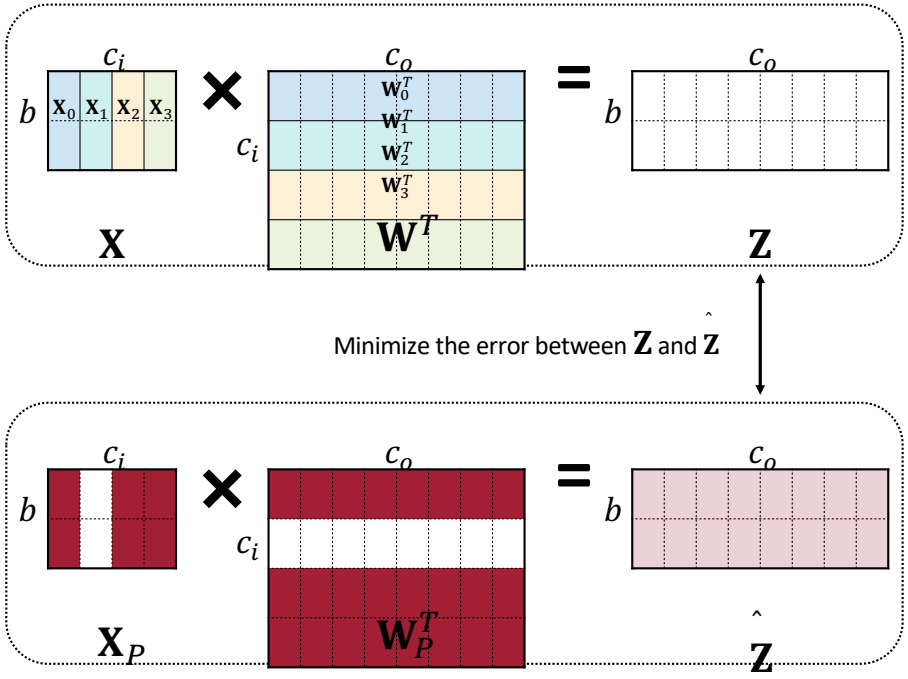


Regression-based Pruning

Minimize reconstruction error of the corresponding layer's outputs

- Let

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



beta is a vector

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

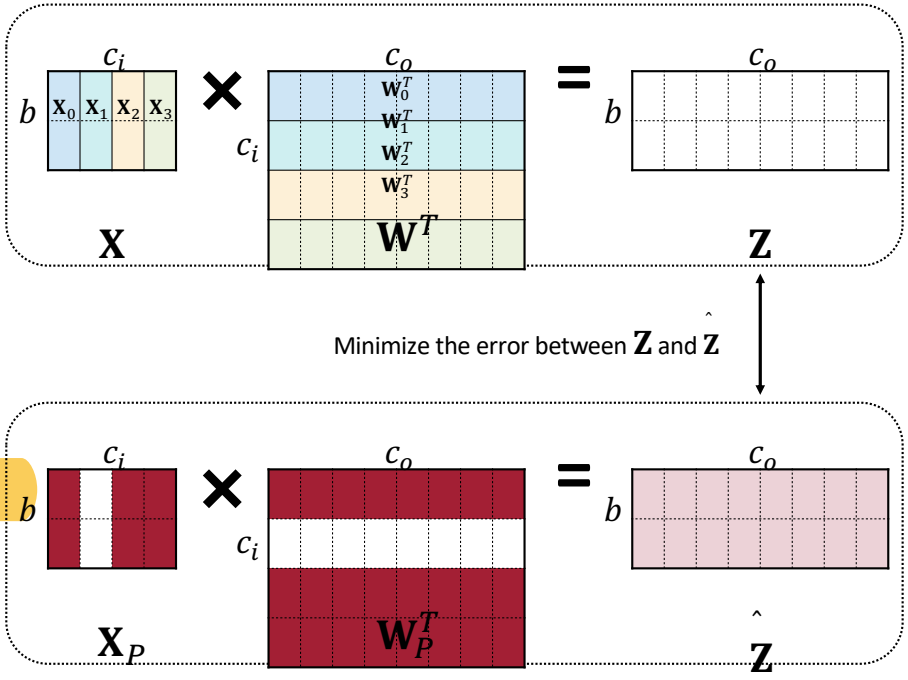
$$\underset{\mathbf{W},\beta}{\operatorname{argmin}} \| \mathbf{Z} - \hat{\mathbf{Z}} \|_F^2 = \| \mathbf{Z} - \sum_{c=0}^{c_i-1} \beta_c \mathbf{X}_c \mathbf{W}_c^T \|_F^2$$

subject to

L0 norm

No. of non-zero entries in beta $\| \beta \|_0 \leq N_c$ tells how much will be pruned

- 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.



Nc should be less than total number of channels

Regression-based Pruning

Minimize reconstruction error of the corresponding layer's outputs

- Let

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

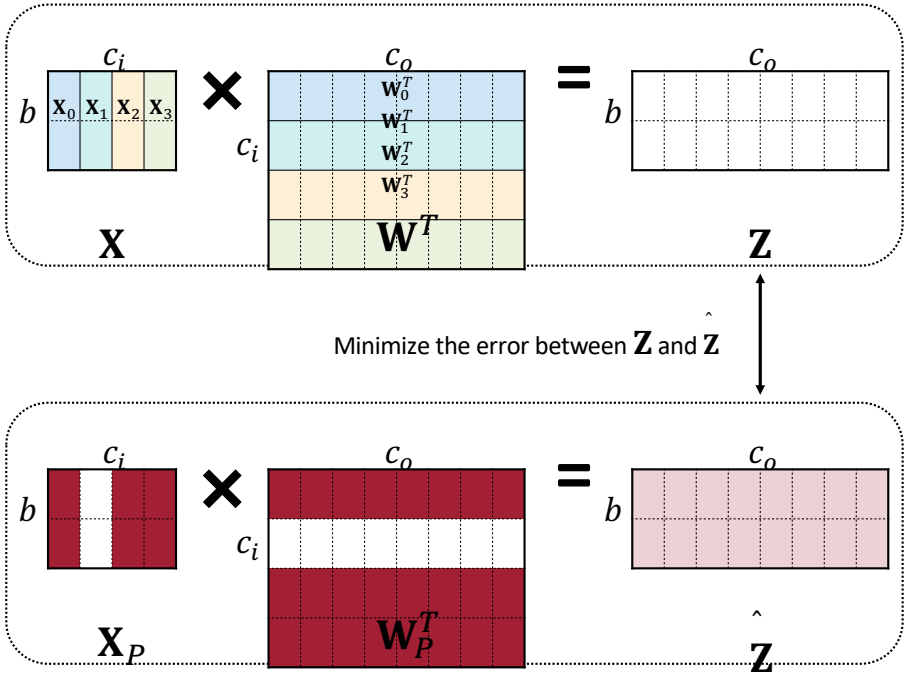
- The problem can be formulate as

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

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]

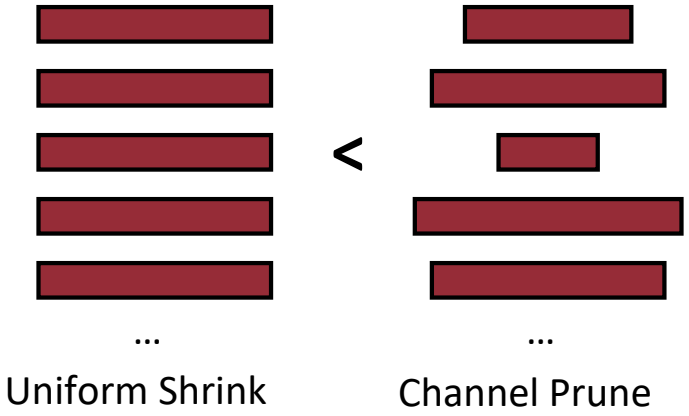
https://efficientml.ai

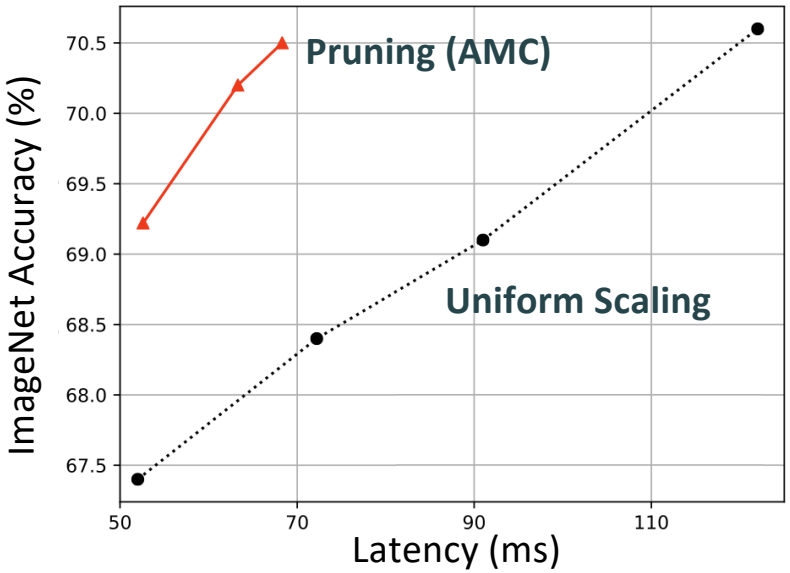
Pruning Ratio

How should we find per-layer pruning ratios?

Recap

Non-uniform pruning is better than uniform shrinking

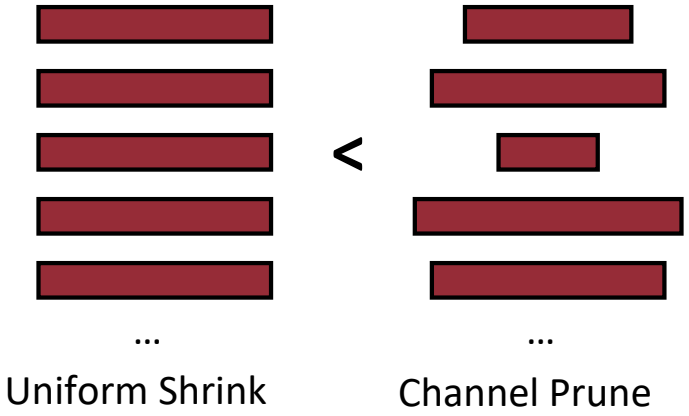


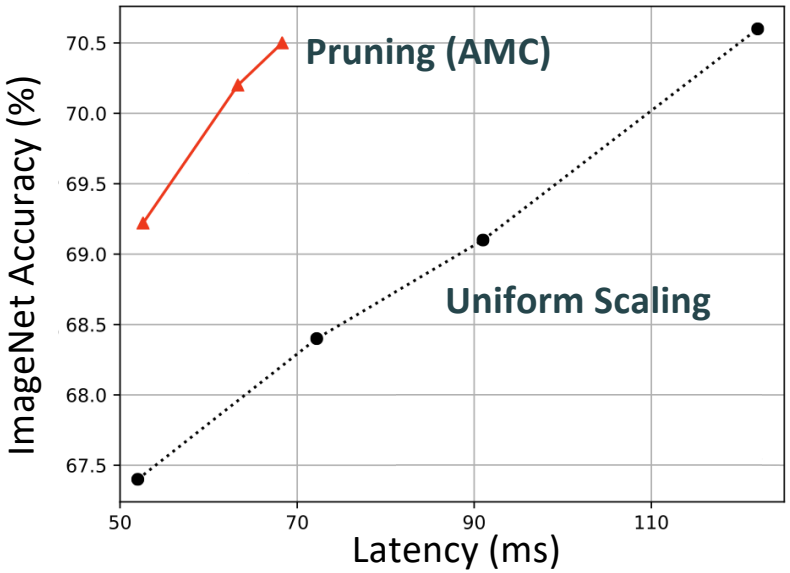


AMC: Automl for Model Compression and Acceleration on Mobile Devices [He *et al.*, ECCV 2018]

Recap

Non-uniform pruning is better than uniform shrinking





Question: how should we find ratios for each layer?

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

Finding Pruning Ratios

Analyze the sensitivity of each layer

- We need different pruning ratios for each layer since different layers have different **sensitivity**
  - Some layers are more sensitive (e.g., first layer)
  - Some layers are more redundant

Finding Pruning Ratios

Analyze the sensitivity of each layer

- We need different pruning ratios for each layer since different layers have different **sensitivity**
  - Some layers are more sensitive (e.g., first layer)
  - Some layers are more redundant
- We can perform **sensitivity analysis** to determine the per-layer pruning ratio

Finding Pruning Ratios

Analyze the sensitivity of each layer

（* VGG-11 on CIFAR-10 dataset)

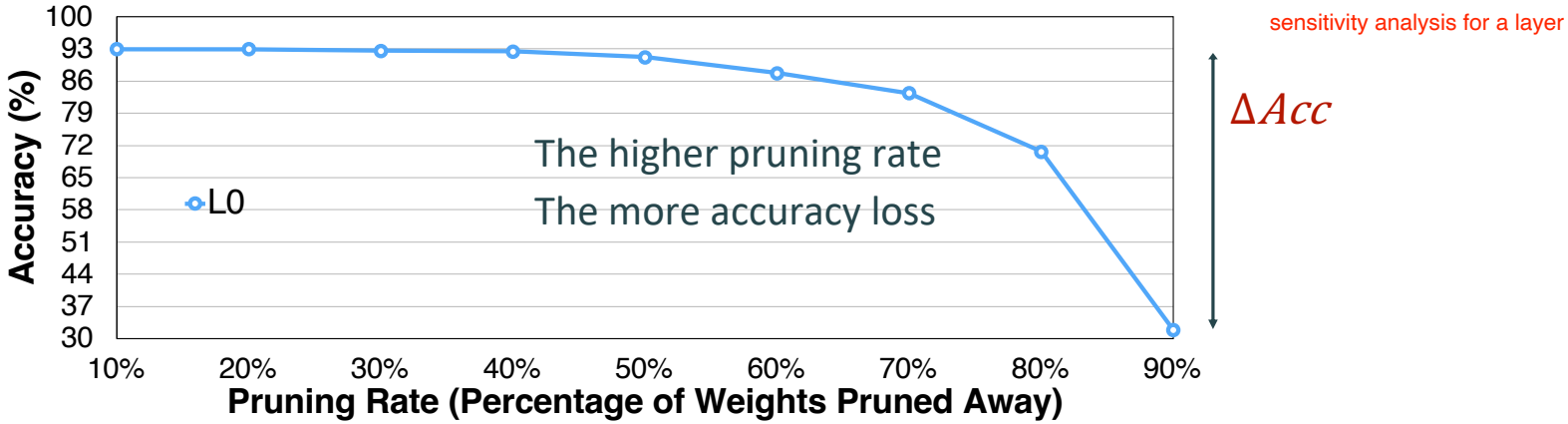
- The process of Sensitivity Analysis
  - Pick a layer L_i in the model

Finding Pruning Ratios

Analyze the sensitivity of each layer

(* VGG-11 on CIFAR-10 dataset)

- The process of Sensitivity Analysis
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, \ldots, 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio

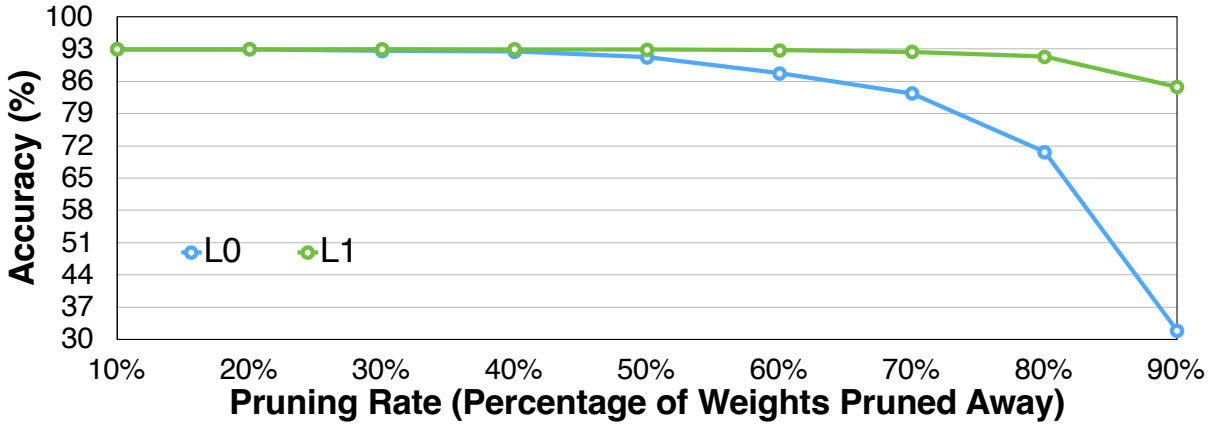


Finding Pruning Ratios

Analyze the sensitivity of each layer

(* VGG-11 on CIFAR-10 dataset)

- The process of Sensitivity Analysis
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, \ldots, 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio
  - Repeat the process for all layers

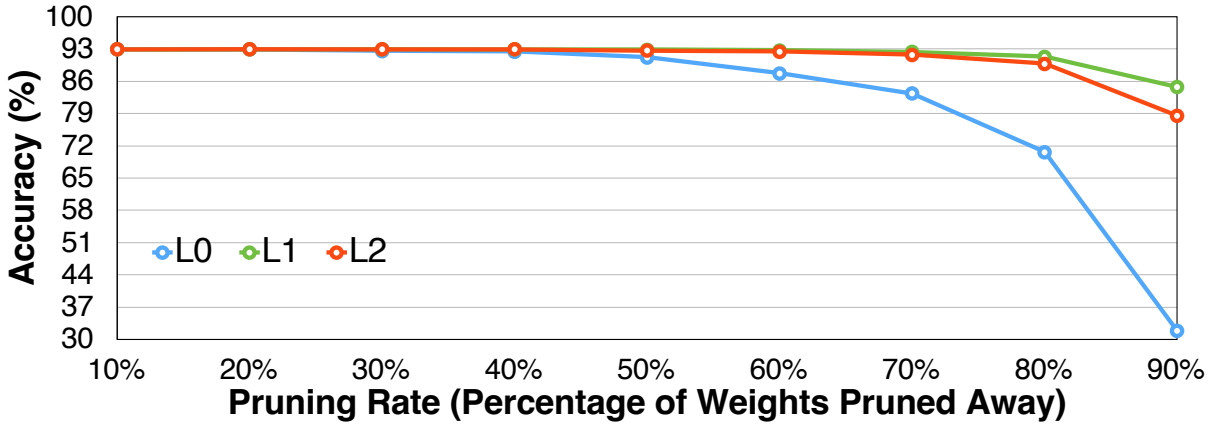


Finding Pruning Ratios

Analyze the sensitivity of each layer

(* VGG-11 on CIFAR-10 dataset)

- The process of Sensitivity Analysis
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, \ldots, 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio
  - Repeat the process for all layers

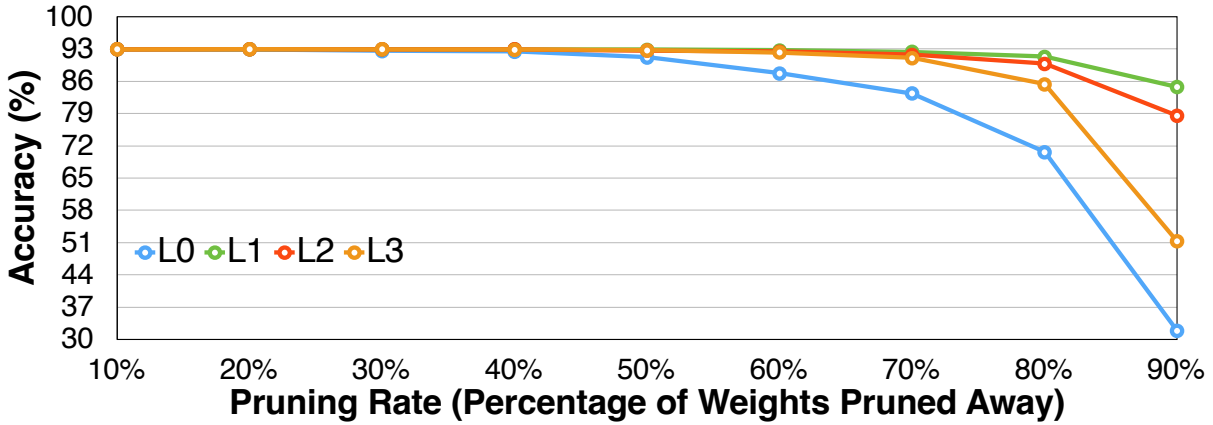


Finding Pruning Ratios

Analyze the sensitivity of each layer

(* VGG-11 on CIFAR-10 dataset)

- The process of Sensitivity Analysis
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, \ldots, 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio
  - Repeat the process for all layers

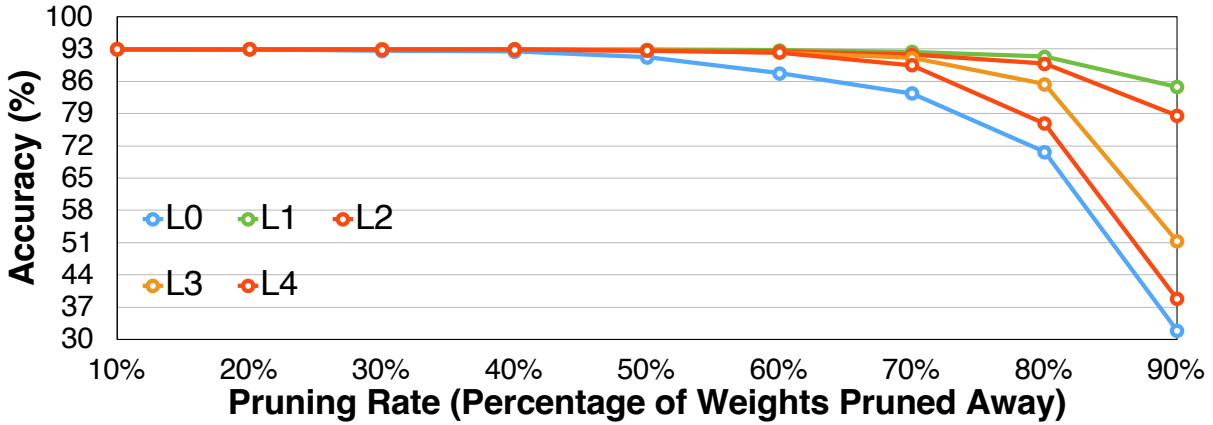


Finding Pruning Ratios

Analyze the sensitivity of each layer

(* VGG-11 on CIFAR-10 dataset)

- The process of Sensitivity Analysis
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, \ldots, 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio
  - Repeat the process for all layers

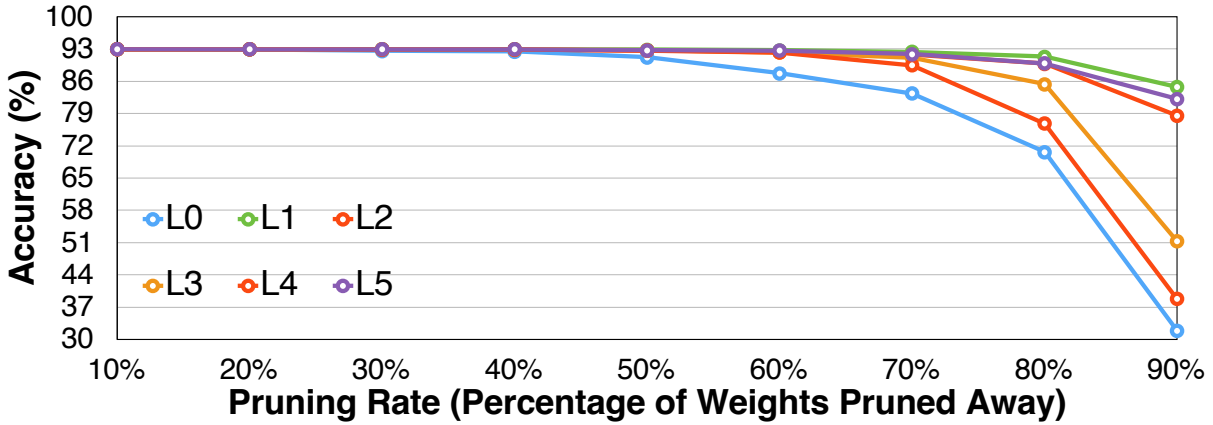


Finding Pruning Ratios

Analyze the sensitivity of each layer

(* VGG-11 on CIFAR-10 dataset)

- The process of Sensitivity Analysis
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, \ldots, 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio
  - Repeat the process for all layers



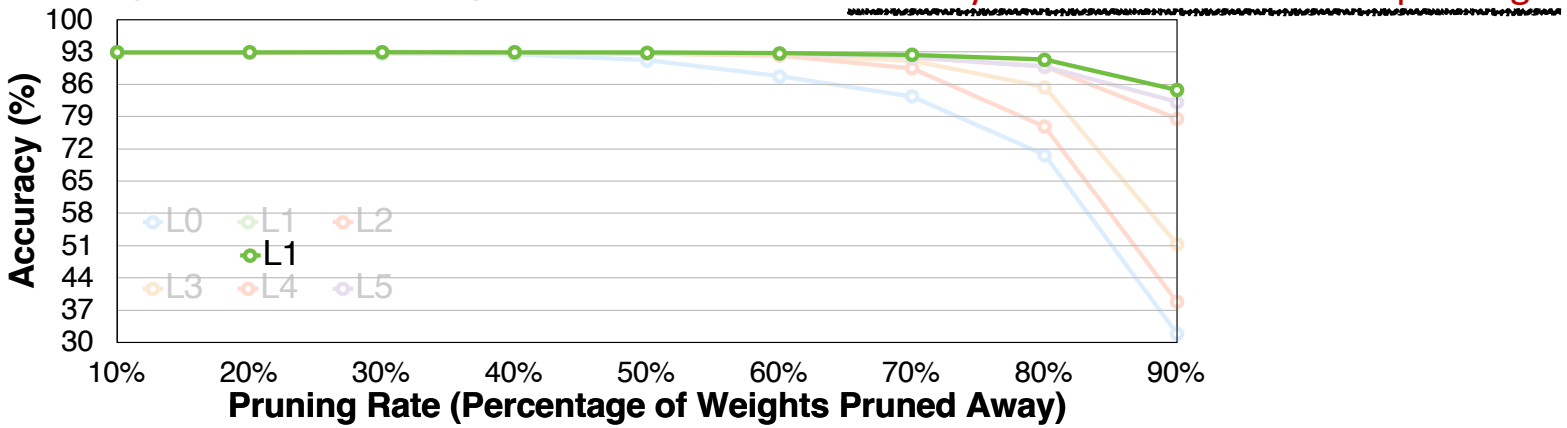
Finding Pruning Ratios

Analyze the sensitivity of each layer

(* VGG-11 on CIFAR-10 dataset)

- The process of Sensitivity Analysis
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, \ldots, 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio
  - Repeat the process for all layers

Some layers are less sensitive to pruning



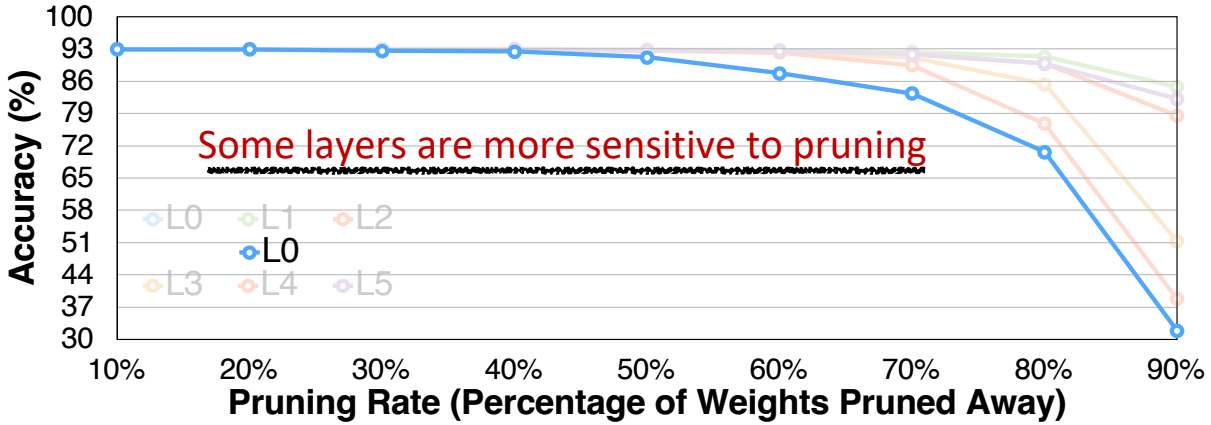
It means that accuracy is not changing much on pruning them - so in a way they are redundant

Finding Pruning Ratios

Analyze the sensitivity of each layer

(* VGG-11 on CIFAR-10 dataset)

- The process of Sensitivity Analysis
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, \ldots, 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio
  - Repeat the process for all layers

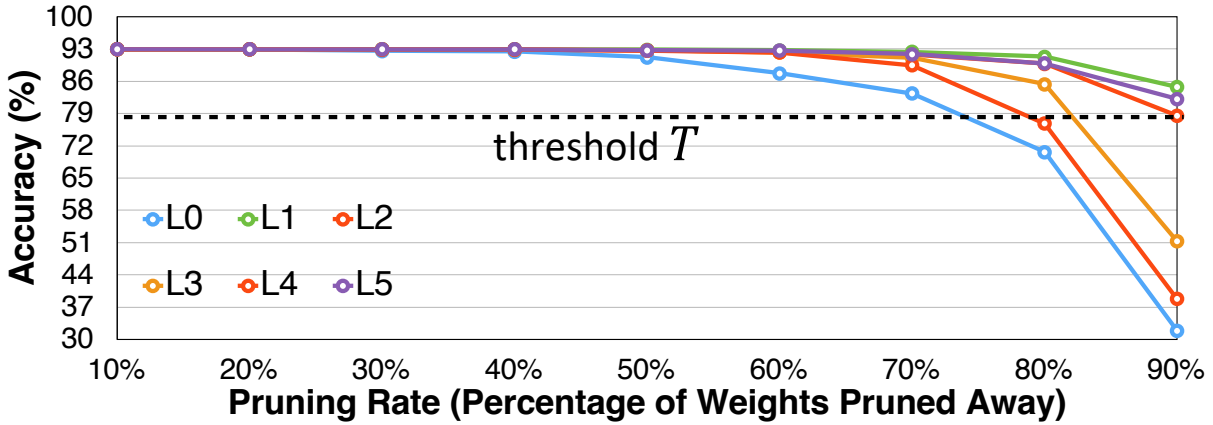


Finding Pruning Ratios

Analyze the sensitivity of each layer

(* VGG-11 on CIFAR-10 dataset)

- The process of Sensitivity Analysis
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, \ldots, 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio
  - Repeat the process for all layers
  - Pick a degradation threshold T such that the overall pruning rate is desired



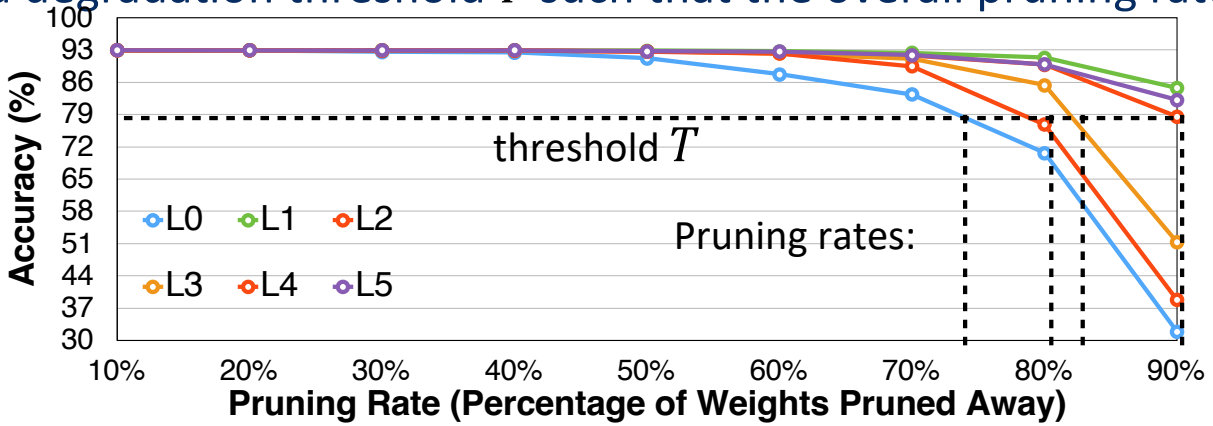
for the lowest acceptable accuracy

Finding Pruning Ratios

Analyze the sensitivity of each layer

(* VGG-11 on CIFAR-10 dataset)

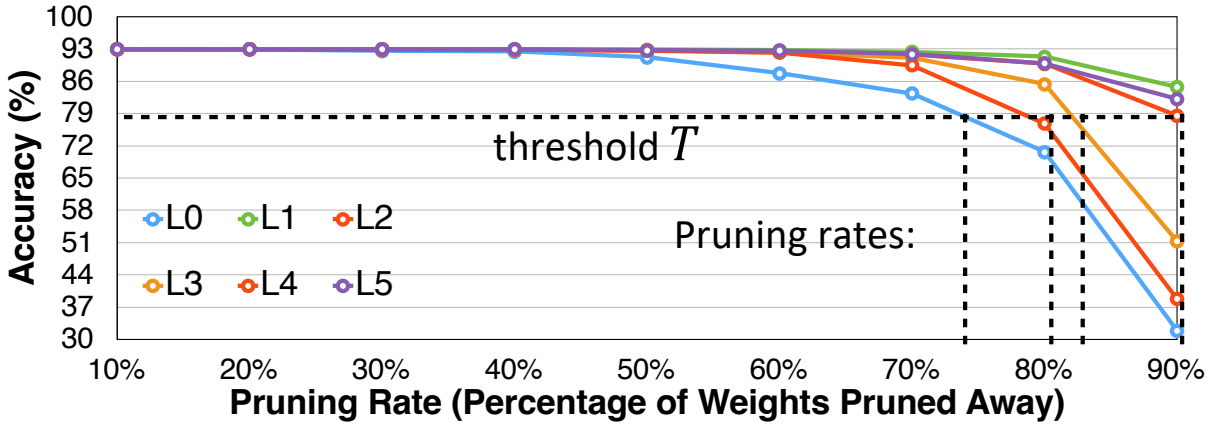
- The process of Sensitivity Analysis
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, \ldots, 0.9\}$ (or other strides)
    - Observe the accuracy degrade ΔAcc_r^i for each pruning ratio
  - Repeat the process for all layers
  - Pick a degradation threshold T such that the overall pruning rate is desired



Finding Pruning Ratios

Analyze the sensitivity of each layer

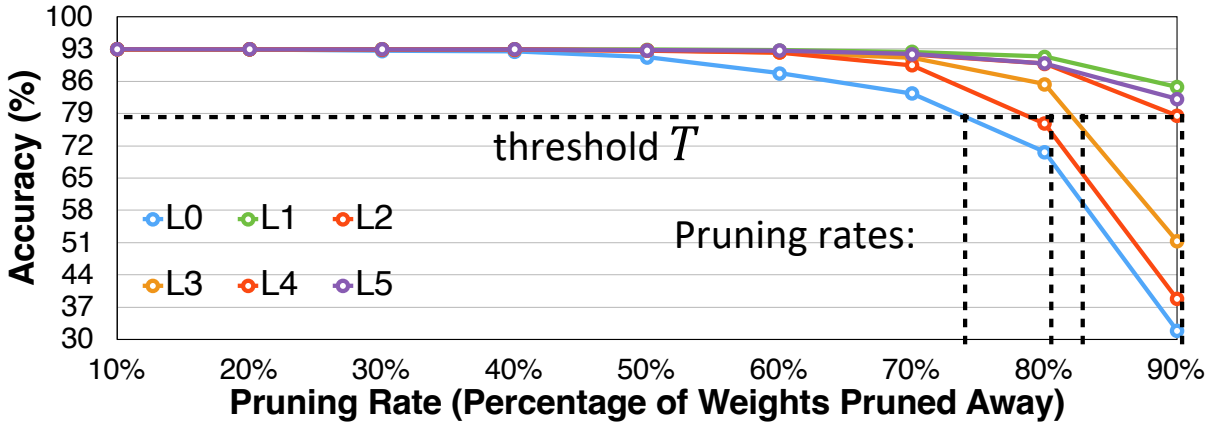
- Is this optimal?



Finding Pruning Ratios

Analyze the sensitivity of each layer

- Is this optimal?
  - Maybe not. We do not consider the interaction between layers.
- Can we go beyond the heuristics?
  - Yes!



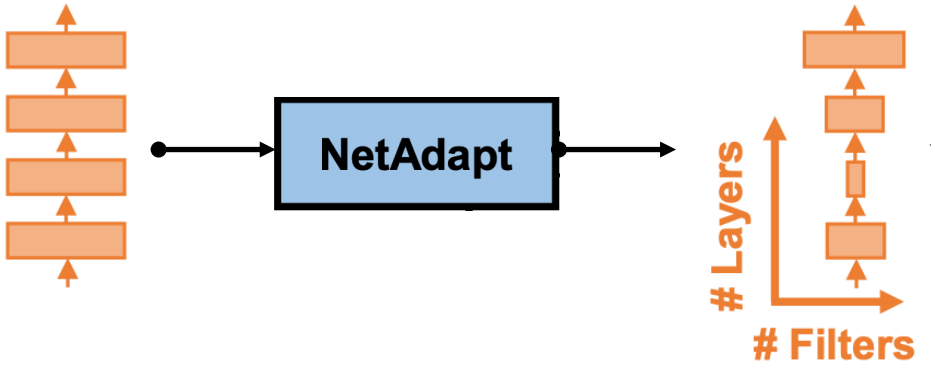
Automatic Pruning

- Given an **overall** compression ratio, how do we choose **per-layer** pruning ratios?
  - Sensitivity analysis ignores the interaction between layers -> sub-optimal

NetAdapt

A rule-based iterative/progressive method

- The goal of NetAdapt is to find a per-layer pruning ratio to meet a global resource constraint (e.g., latency, energy, …)
- The process is done iteratively
- We take **latency** constraint as an example

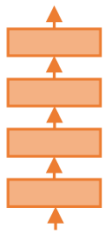


NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications [Yang *et al.*, ECCV 2018]

https://efficientml.ai

NetAdapt

- For each iteration, we aim to reduce the latency by a certain amount ΔR (manually defined)

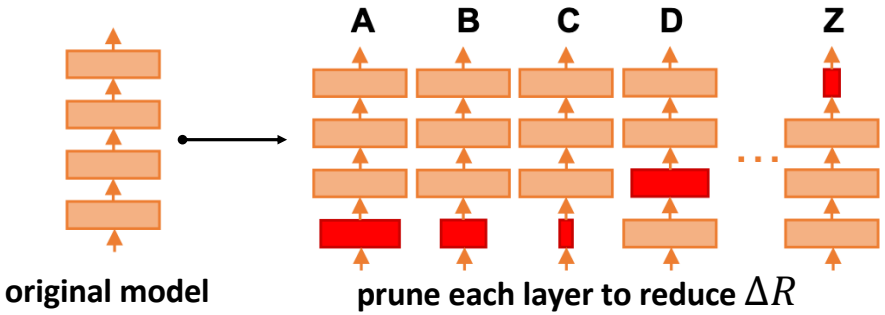


original model

NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications [Yang *et al.*, ECCV 2018]
https://efficientml.ai

NetAdapt

- For each iteration, we aim to reduce the latency by a certain amount ΔR (manually defined)
  - For each layer L_k (k in A-Z in the figure)
    - Prune the layer s.t. the latency reduction meets ΔR (based on a pre-built lookup table)

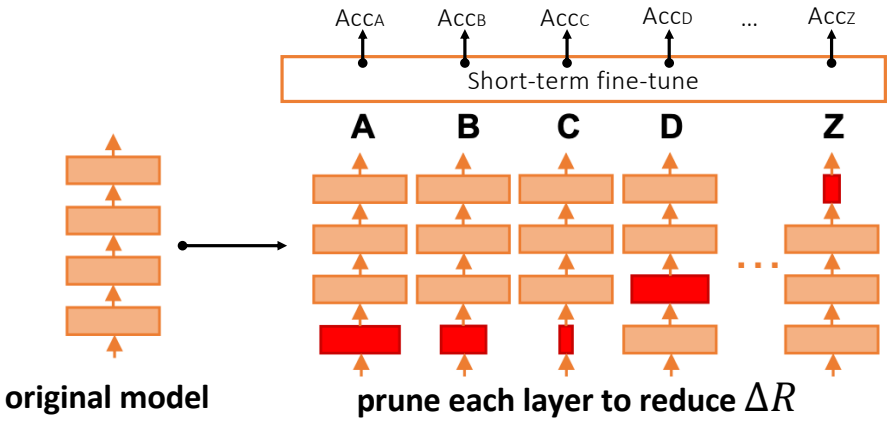


NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications [Yang *et al.*, ECCV 2018]

https://efficientml.ai

NetAdapt

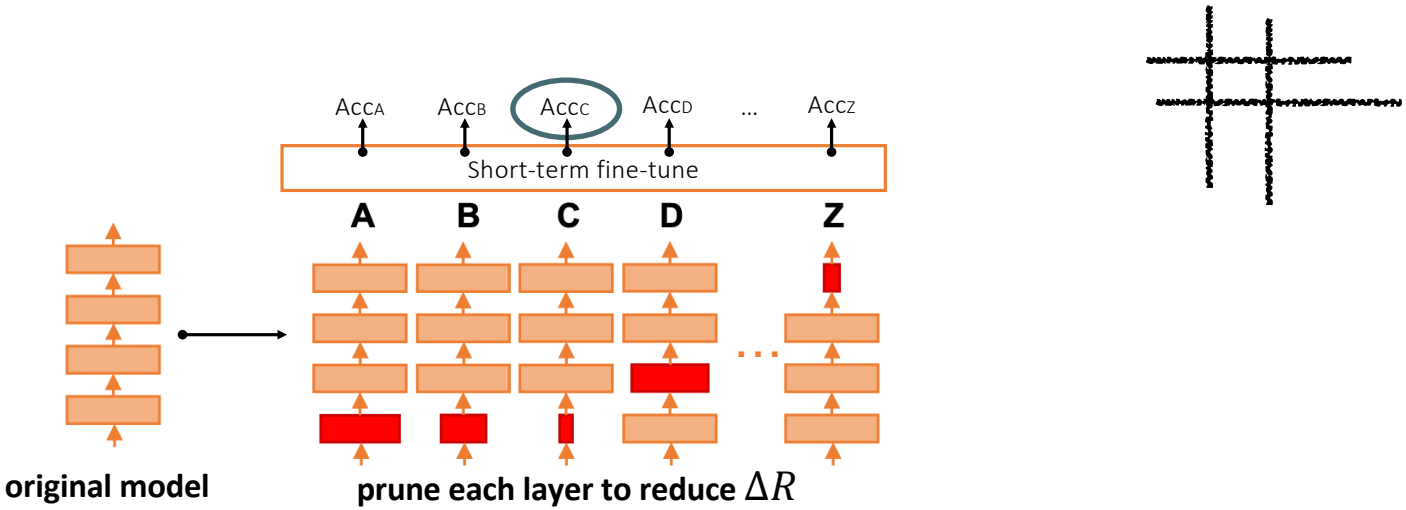
- For each iteration, we aim to reduce the latency by a certain amount ΔR (manually defined)
  - For each layer L_k (k in A-Z in the figure)
    - Prune the layer s.t. the latency reduction meets ΔR (based on a pre-built lookup table)
    - Short-term fine-tune model (10k iterations); measure accuracy after fine-tuning



NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications [Yang *et al.*, ECCV 2018]
https://efficientml.ai

NetAdapt

- For each iteration, we aim to reduce the latency by a certain amount ΔR (manually defined)
  - For each layer L_k (k in A-Z in the figure)
    - Prune the layer s.t. the latency reduction meets ΔR (based on a pre-built lookup table)
    - Short-term fine-tune model (10k iterations); measure accuracy after fine-tuning
  - Choose and prune the layer with the highest accuracy

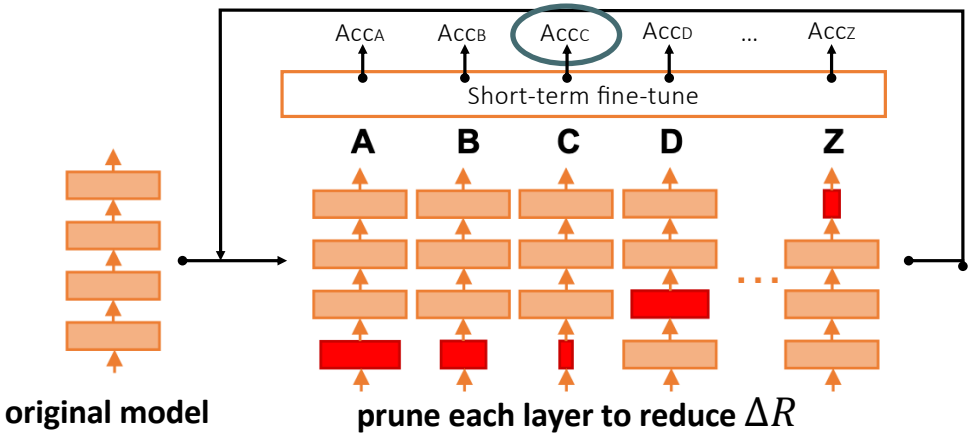


NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications [Yang *et al.*, ECCV 2018]

https://efficientml.ai

NetAdapt

- For each iteration, we aim to reduce the latency by a certain amount ΔR (manually defined)
  - For each layer L_k (k in A-Z in the figure)
    - Prune the layer s.t. the latency reduction meets ΔR (based on a pre-built lookup table)
    - Short-term fine-tune model (10k iterations); measure accuracy after fine-tuning
  - Choose and prune the layer with the highest accuracy
- Repeat until the total latency reduction satisfies the constraint

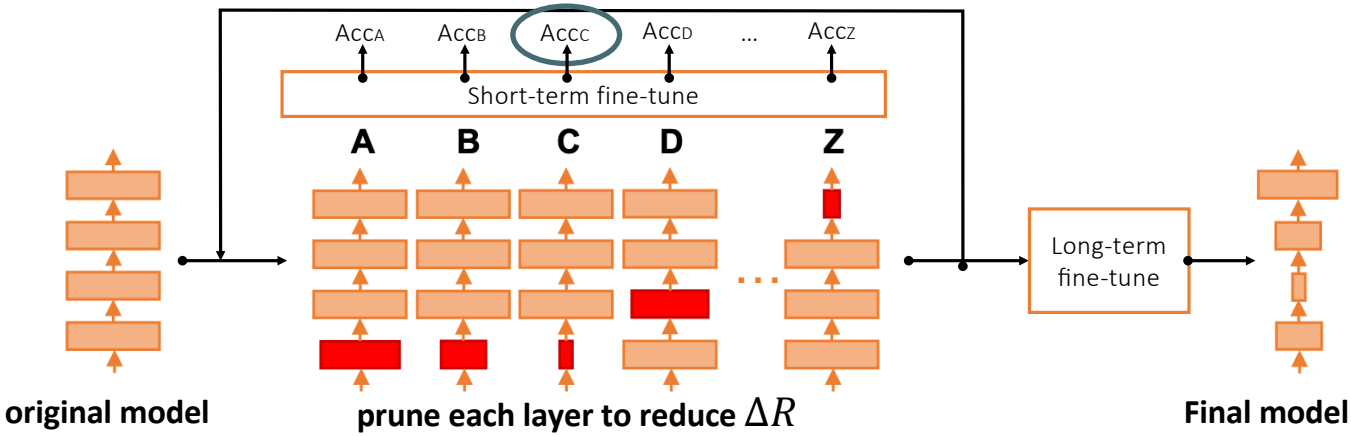


NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications [Yang *et al.*, ECCV 2018]

https://efficientml.ai

NetAdapt

- For each iteration, we aim to reduce the latency by a certain amount ΔR (manually defined)
  - For each layer L_k (k in A-Z in the figure)
    - Prune the layer s.t. the latency reduction meets ΔR (based on a pre-built lookup table)
    - Short-term fine-tune model (10k iterations); measure accuracy after fine-tuning
  - Choose and prune the layer with the highest accuracy
- Repeat until the total latency reduction satisfies the constraint
- Long-term fine-tune to recover accuracy

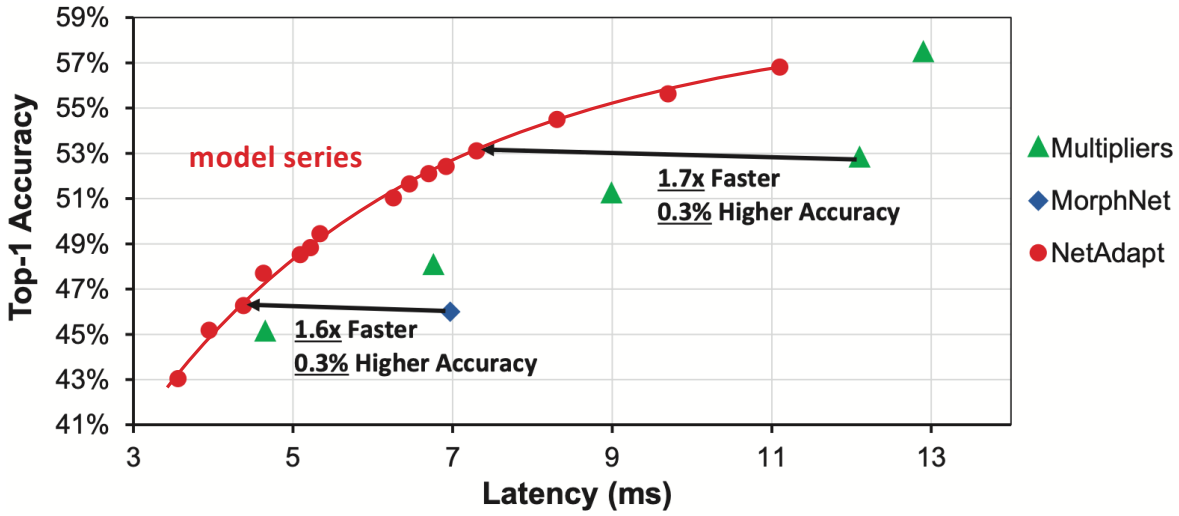


NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications [Yang *et al.*, ECCV 2018]

https://efficientml.ai

NetAdapt

- The iterative nature allows us to obtain **a serial of** models with different costs
  - #models = #iterations



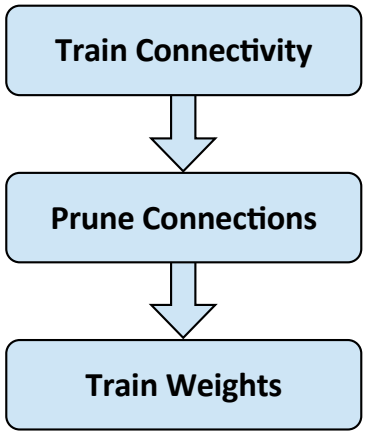
NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications [Yang *et al.*, ECCV 2018]
https://efficientml.ai

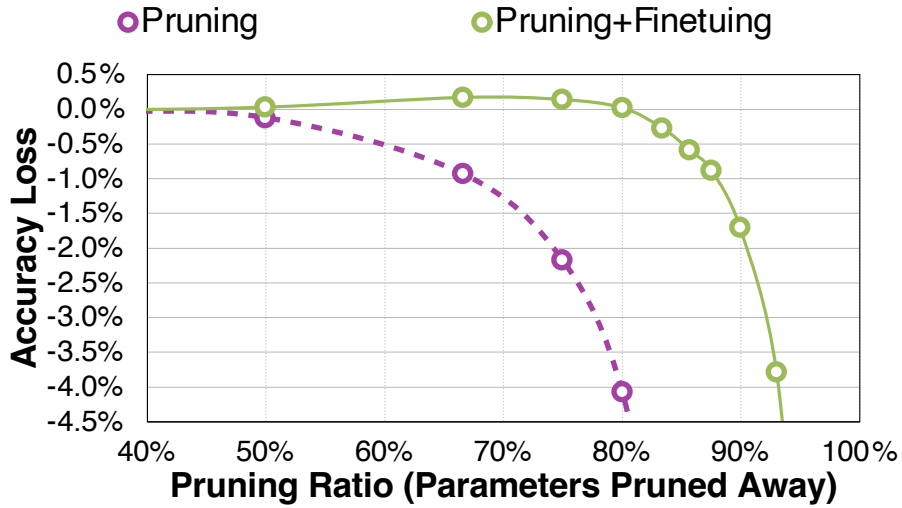
Fine-tuning / Training

How should we improve performance of sparse models?

Finetuning Pruned Neural Networks

- After pruning, the model may decrease, especially for larger pruning ratio.
- Fine-tuning the pruned neural networks will help recover the accuracy and push the pruning ratio higher. as we don't want to drift much from our original learned model. As if LR is high, our model might unlearn various things
  - **Learning rate for fine-tuning is usually 1/100 or 1/10 of the original learning rate.**

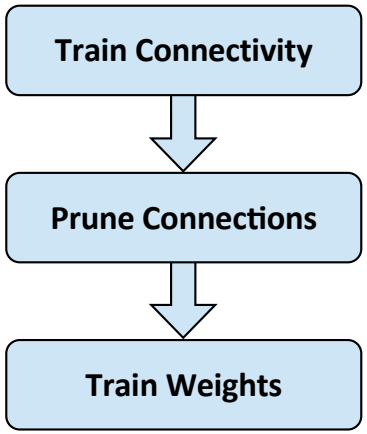


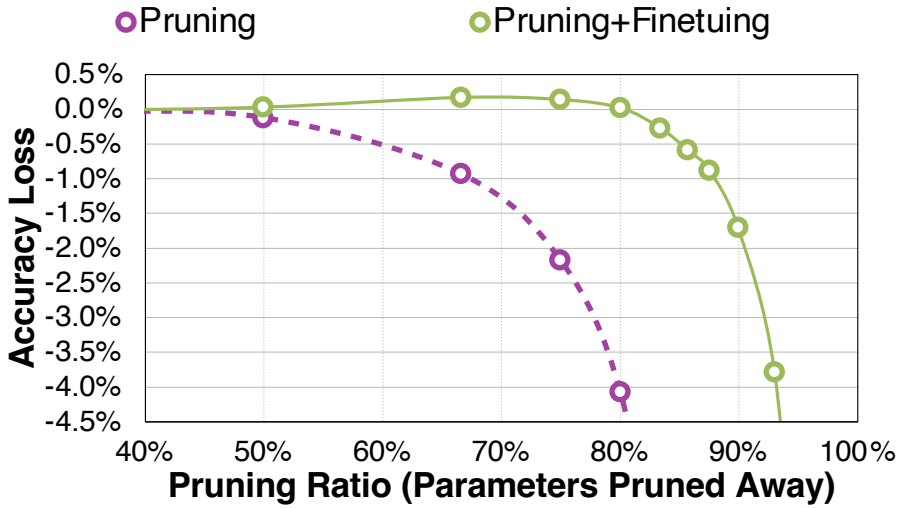


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Iterative Pruning

- Consider pruning followed by a fine-tuning is one iteration.
- Iterative pruning gradually increases the target sparsity in each iteration.

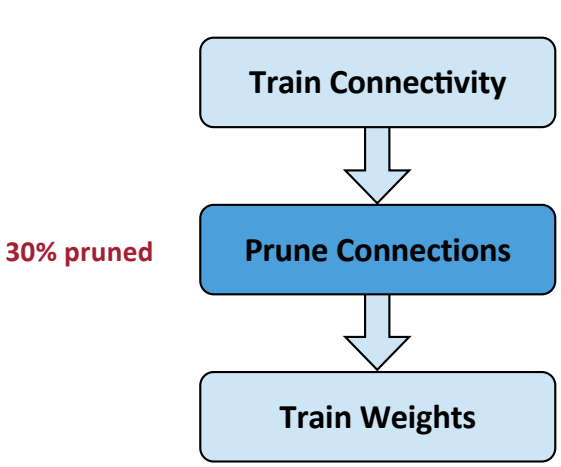


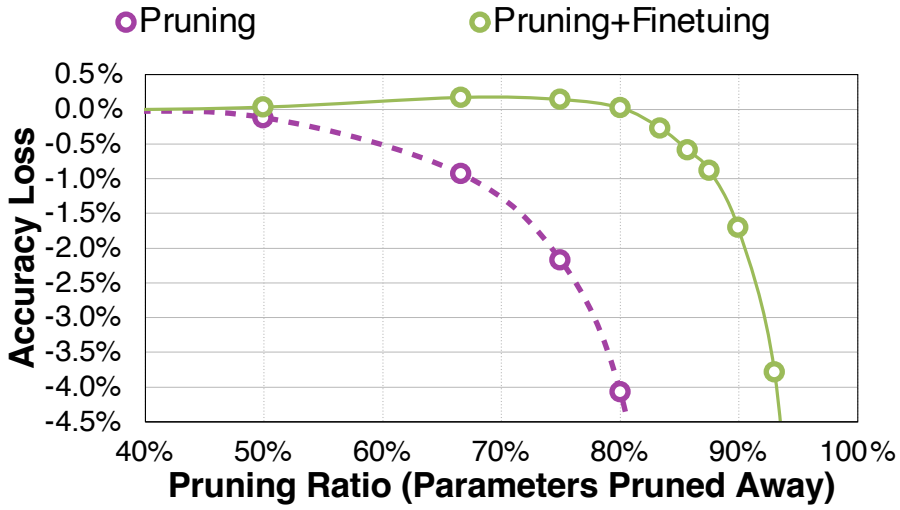


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Iterative Pruning

- Consider pruning followed by a fine-tuning is one iteration.
- Iterative pruning gradually increases the target sparsity in each iteration.

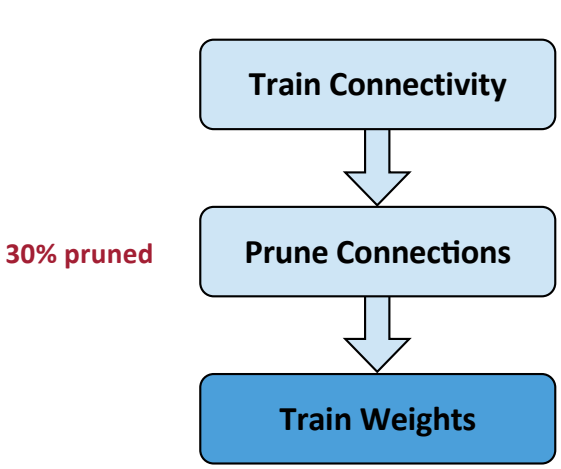


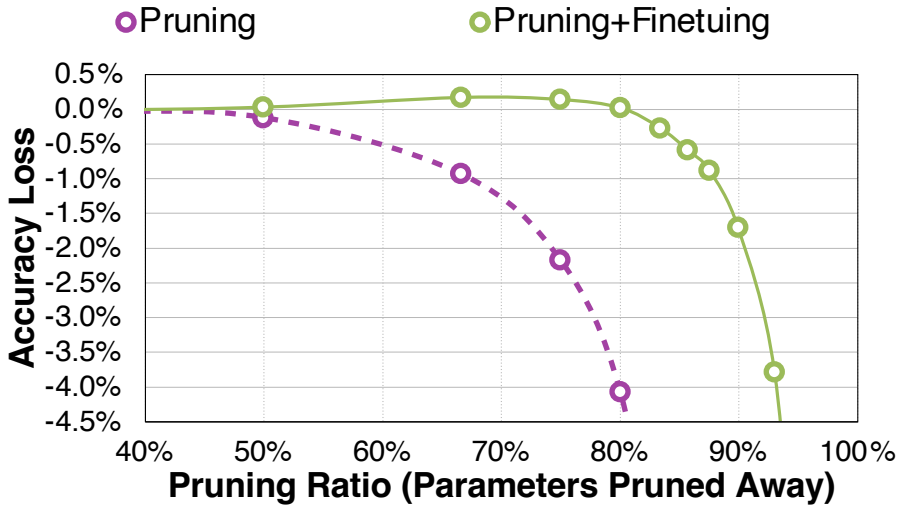


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Iterative Pruning

- Consider pruning followed by a fine-tuning is one iteration.
- Iterative pruning gradually increases the target sparsity in each iteration.

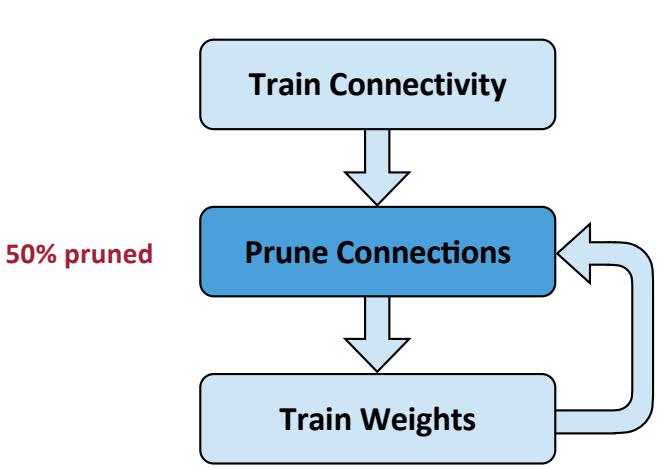


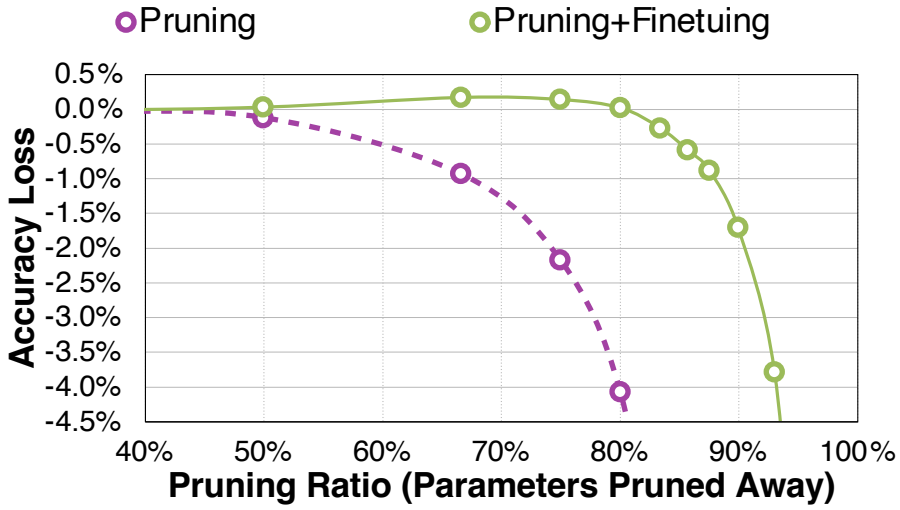


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Iterative Pruning

- Consider pruning followed by a fine-tuning is one iteration.
- Iterative pruning gradually increases the target sparsity in each iteration.

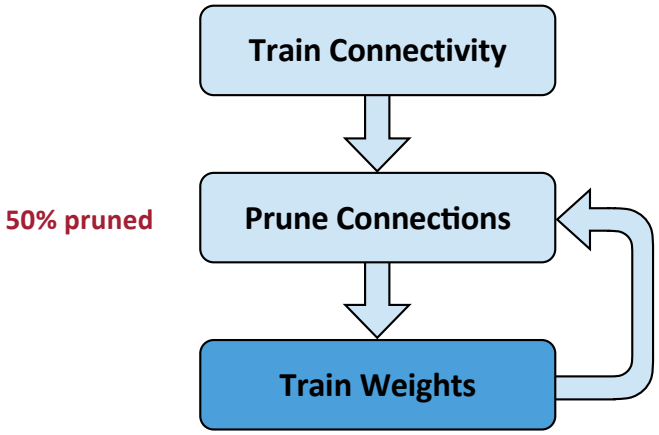


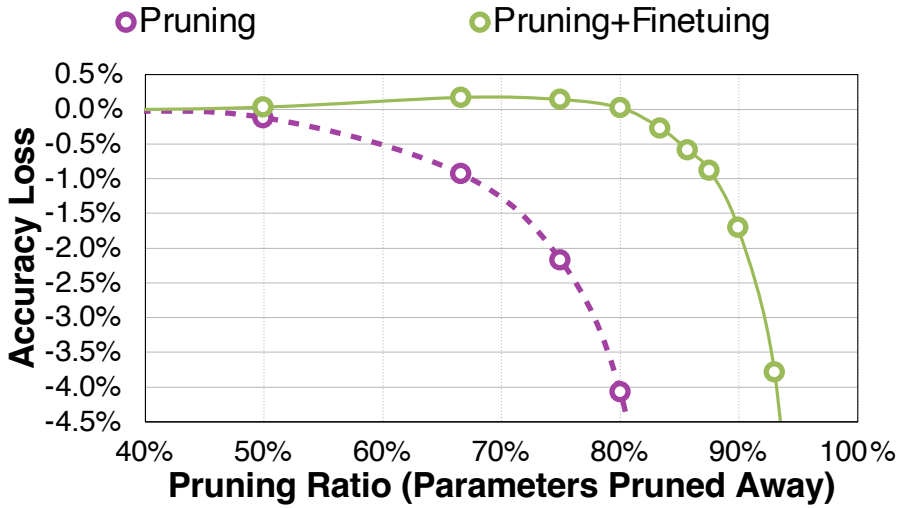


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Iterative Pruning

- Consider pruning followed by a fine-tuning is one iteration.
- Iterative pruning gradually increases the target sparsity in each iteration.

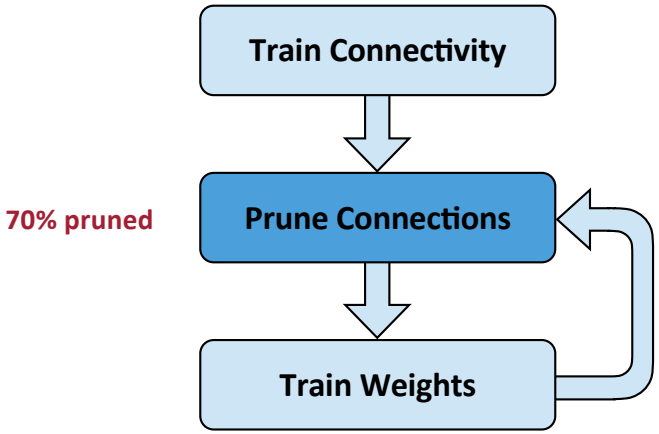


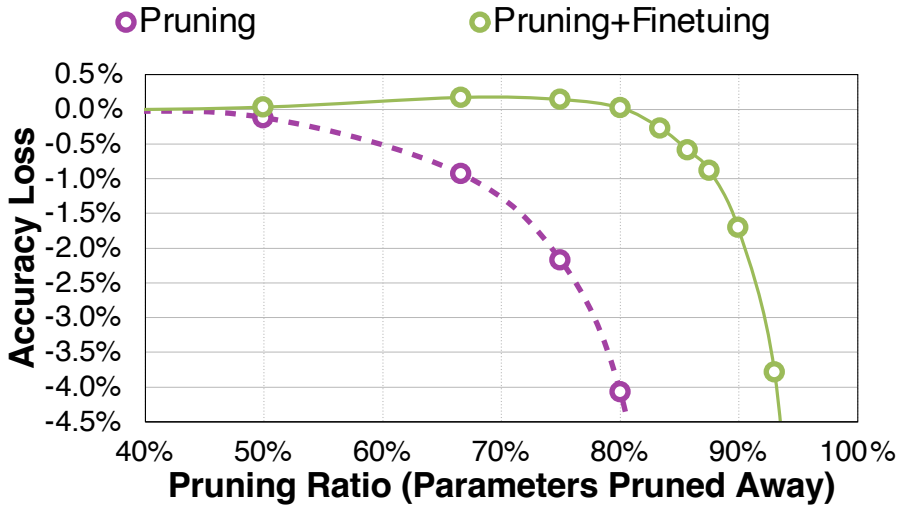


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Iterative Pruning

- Consider pruning followed by a fine-tuning is one iteration.
- Iterative pruning gradually increases the target sparsity in each iteration.

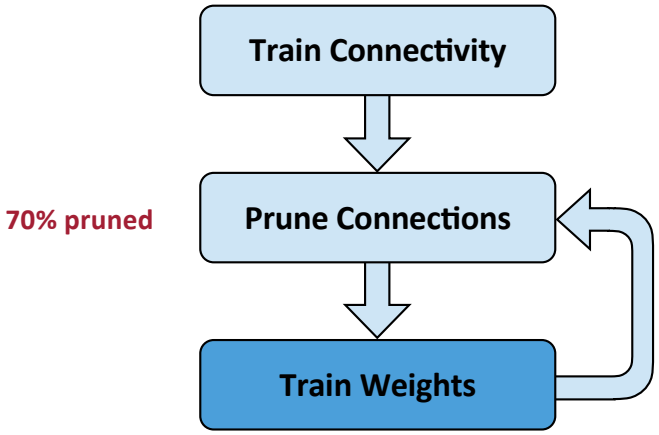


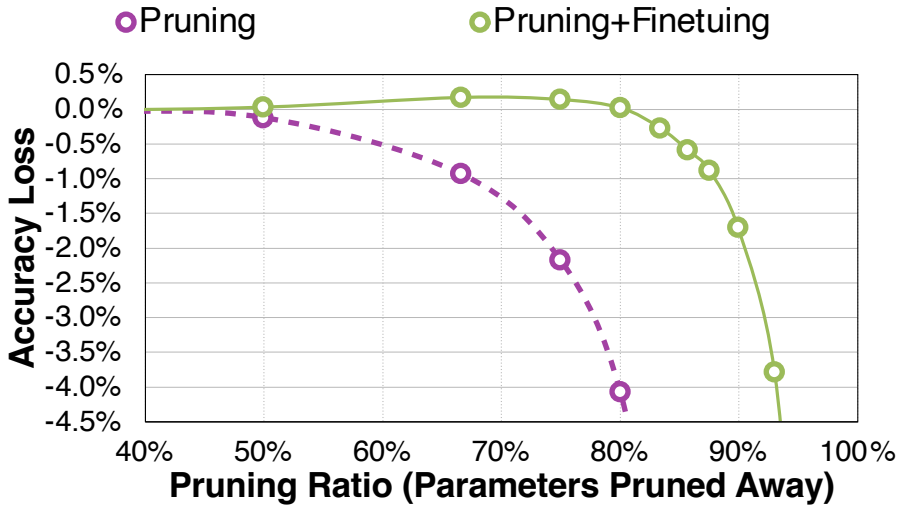


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Iterative Pruning

- Consider pruning followed by a fine-tuning is one iteration.
- Iterative pruning gradually increases the target sparsity in each iteration.

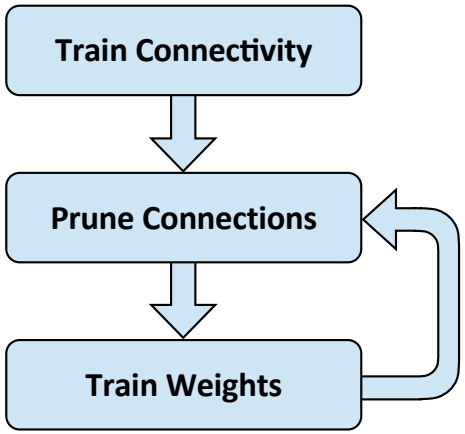


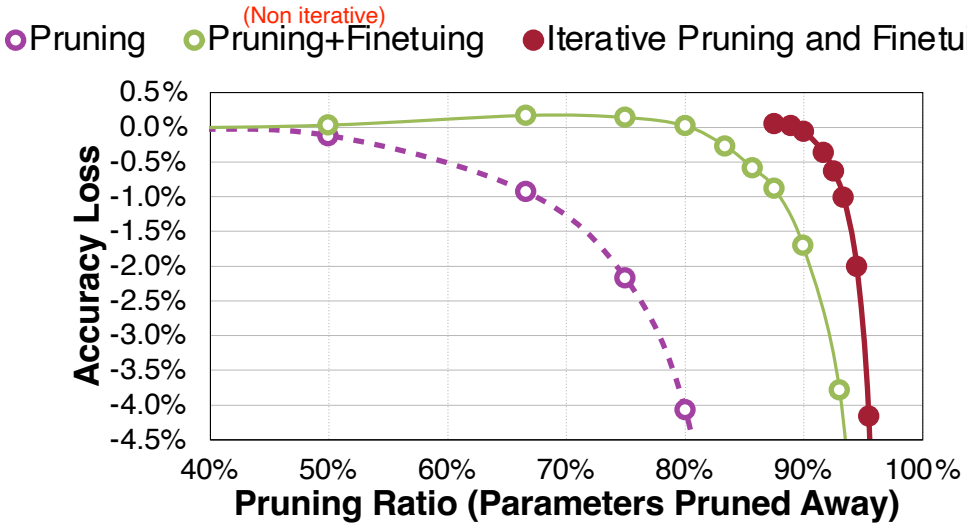


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Iterative Pruning

- Consider pruning followed by a fine-tuning is one iteration.
- Iterative pruning gradually increases the target sparsity in each iteration.
  - boost pruning ratio from 5× to 9× on AlexNet compared to single-step aggressive pruning.





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

Regularization

- When training neural networks or fine-tuning quantized neural networks, regularization is added to the loss term to
  - ***penalize non-zero parameters***
  - ***encourage smaller parameters***
- The most common regularization for improving performance of pruning is L1/L2 regularization.
  - L1-Regularization

$$L' = L(\mathbf{x}; \mathbf{W}) + \lambda |\mathbf{W}|$$

  - L2-Regularization

$$L' = L(\mathbf{x}; \mathbf{W}) + \lambda \parallel \mathbf{W} \parallel^2$$

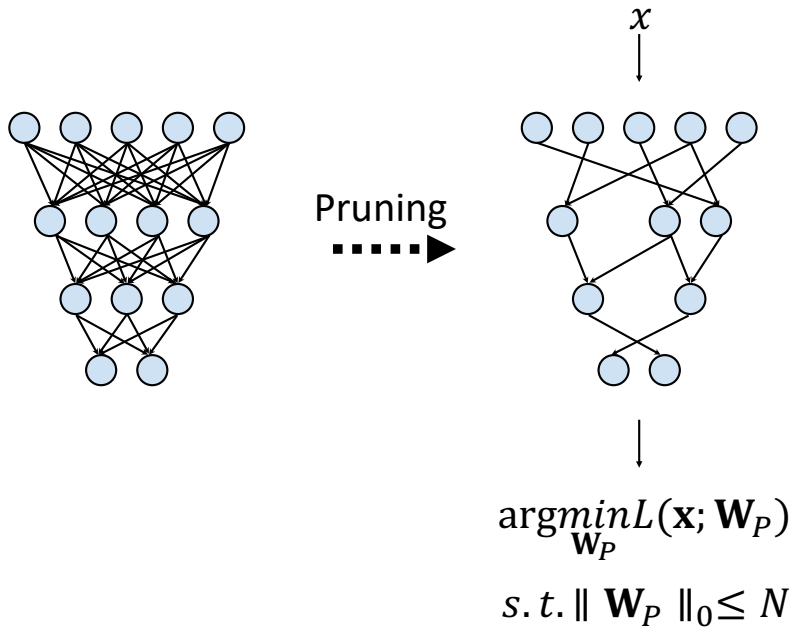
  - **Examples:**
    - Magnitude-based Fine-grained Pruning applies L2 regularization on weights
    - Network Slimming applies smooth-L1 regularization on channel scaling factors.

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

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]

Lottery Ticket Hypothesis

Can we train a sparse neural network from scratch?

can we get a pruned network while building from scratch?

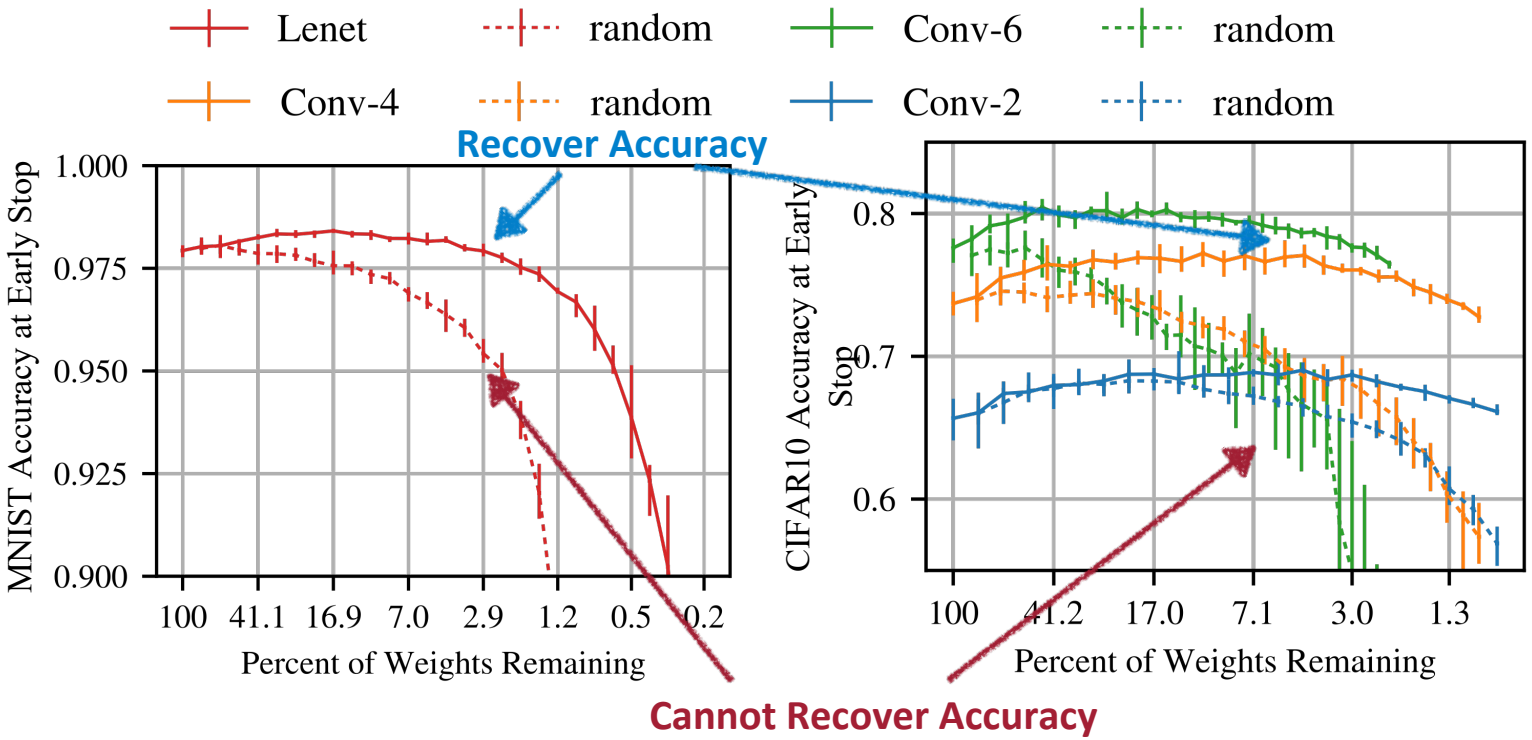
Train Sparse Neural Network From Scratch

- Neural Network Pruning shows that
  - **a neural network can be reduced in size.**

- Question: **Can we directly train this sparse neural network from scratch?**

- Contemporary experience tells us that
  - **the architectures uncovered by pruning are harder to train from the start,**
  - **reaching lower accuracy than the original networks**

The Lottery Ticket Hypothesis



- Dashed Lines: randomly sampled sparse networks
- Solid Lines: correct sparsitiy mask (found by training) & same initialization.

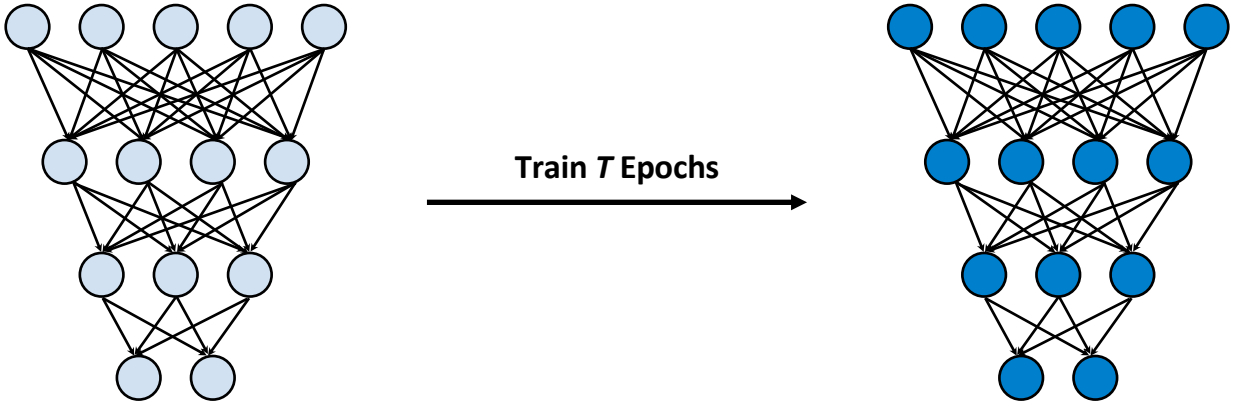
The Lottery Ticket Hypothesis

A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations.

—The Lottery Ticket Hypothesis

the is because, after pruning the original model, we can come up with a sparse network with same accuracy

So, the hypothesis is we can find that subnetwork beforehand and train it from scratch we can achieve same accuracy as that of big model



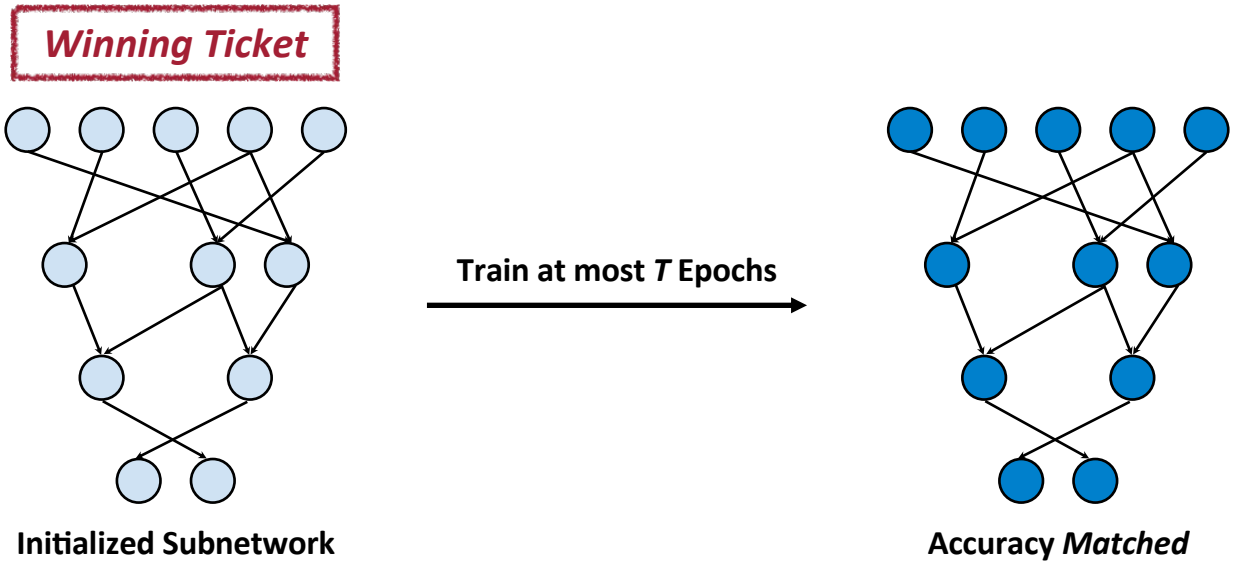
Randomly-Initialized

The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]

https://efficientml.ai

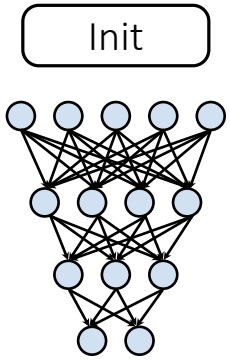
The Lottery Ticket Hypothesis

*A randomly-initialized, dense neural network contains **a subnetwork** that is initialized such that—when **trained in isolation**—it can **match the test accuracy** of the original network after training for **at most the same number of iterations**.*
—The Lottery Ticket Hypothesis



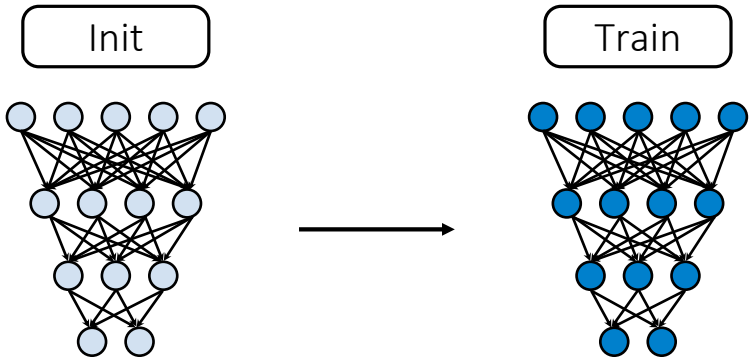
The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]
https://efficientml.ai

Iterative Magnitude Pruning



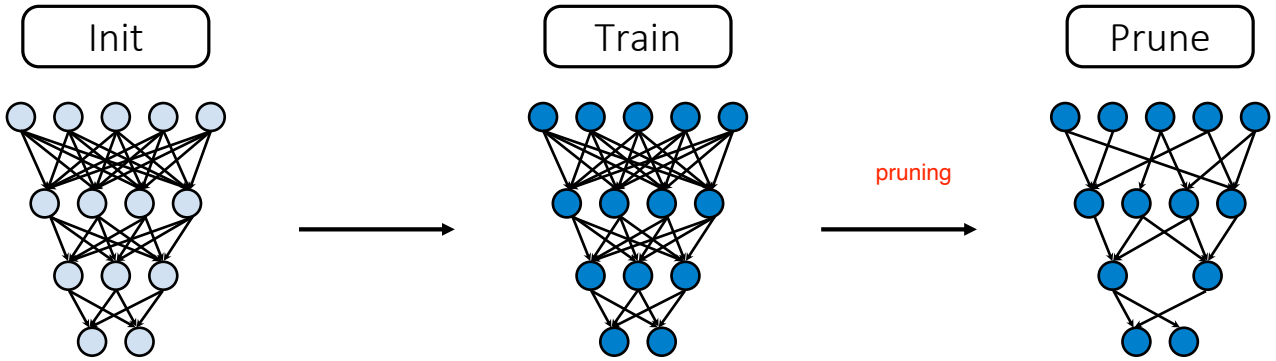
original dense neural network

Iterative Magnitude Pruning



The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]

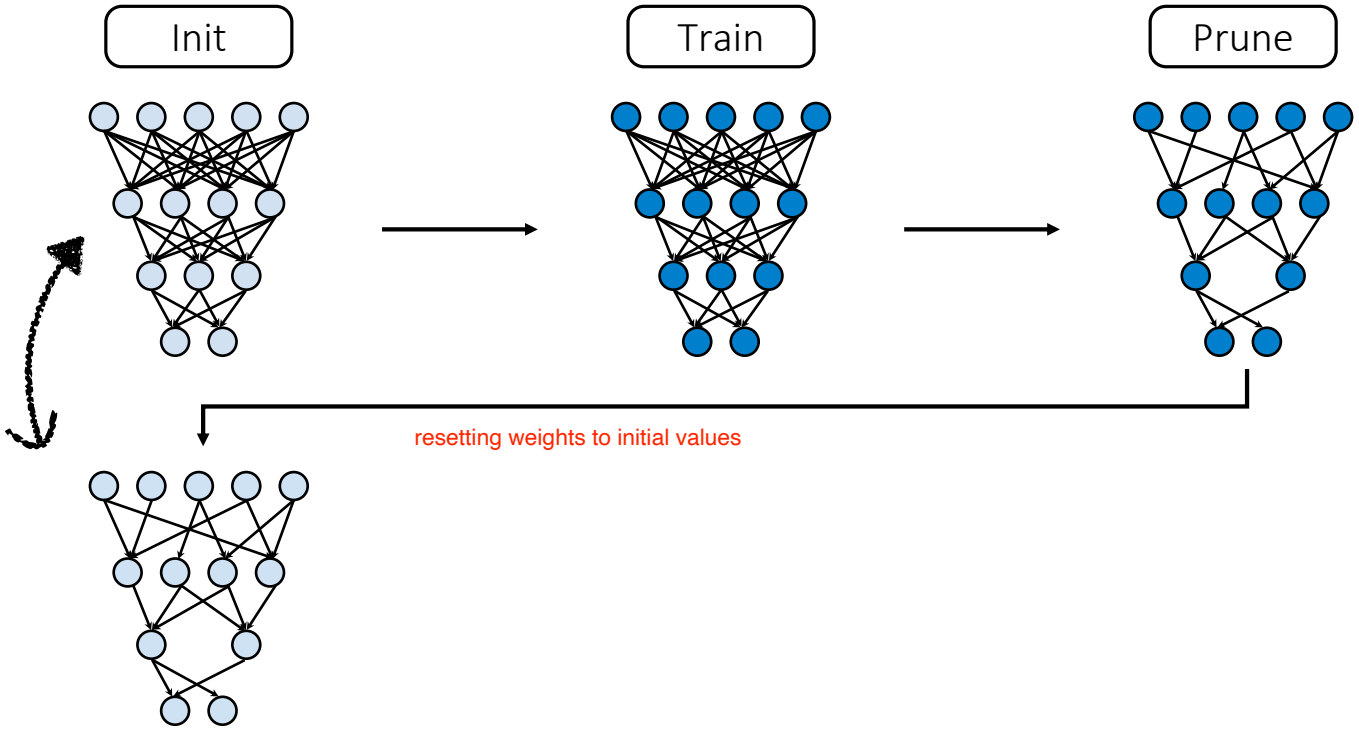
Iterative Magnitude Pruning



The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]

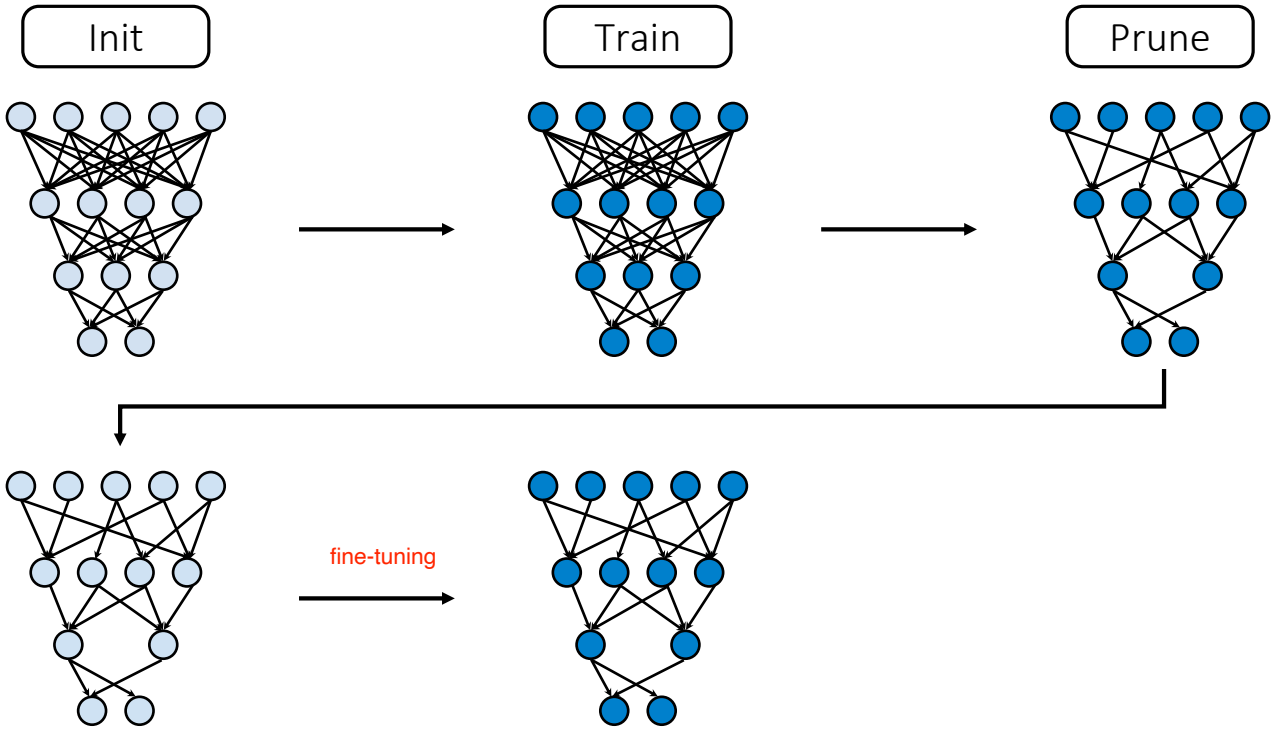
Iterative Magnitude Pruning

here, colour indicates the state of weights



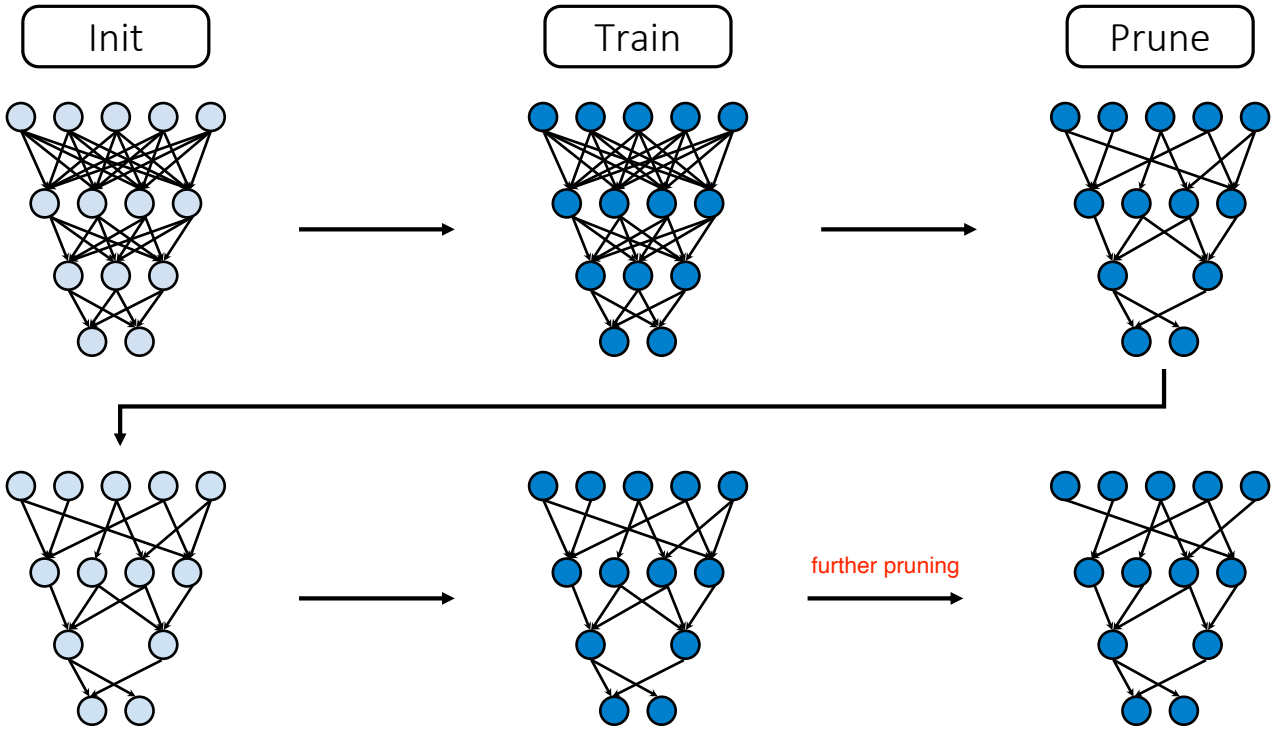
The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]

Iterative Magnitude Pruning



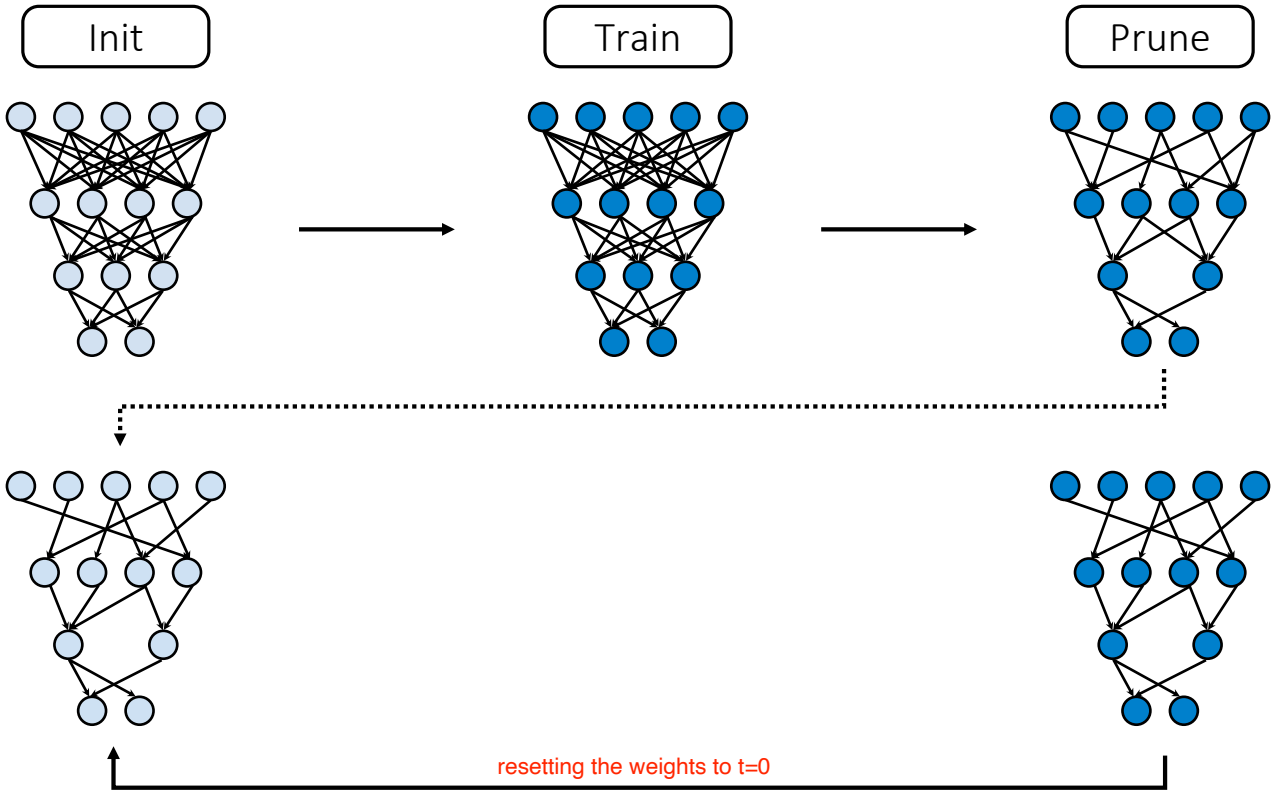
The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]

Iterative Magnitude Pruning



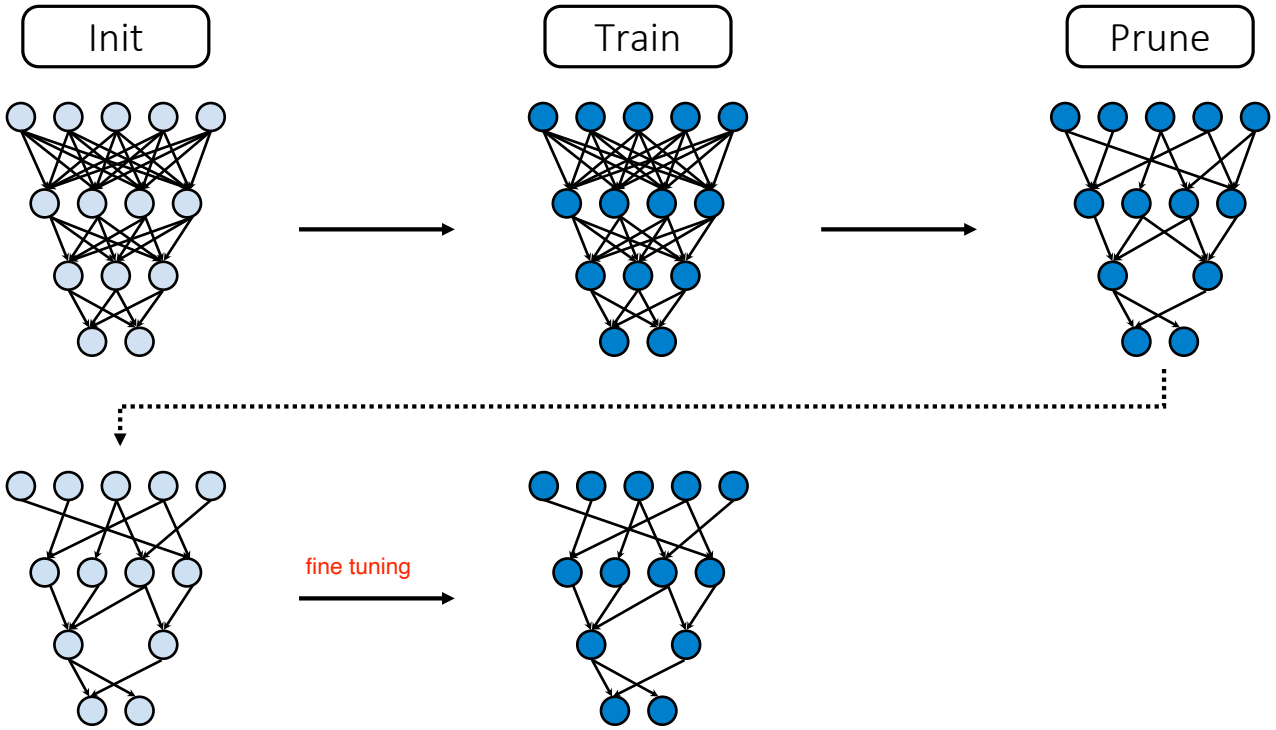
The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]

Iterative Magnitude Pruning



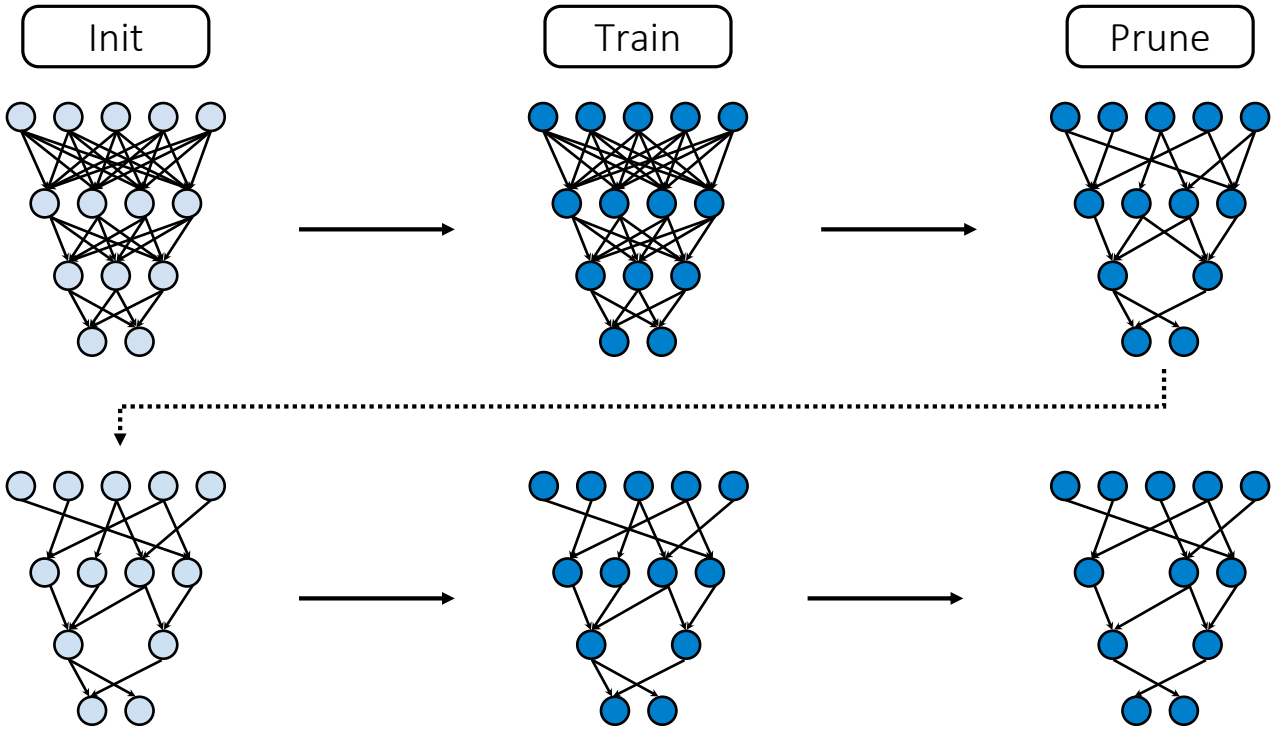
The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]

Iterative Magnitude Pruning



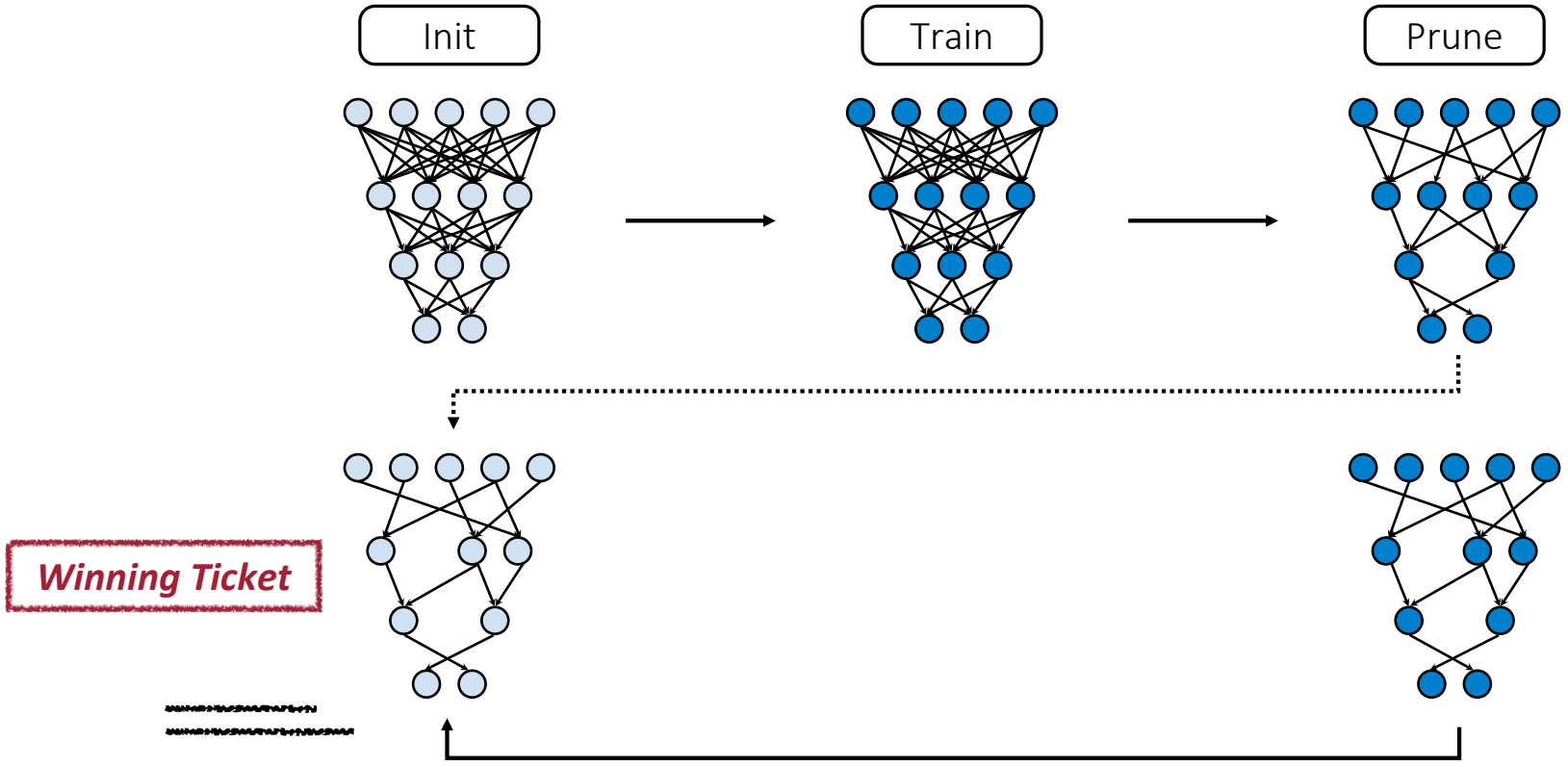
The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]

Iterative Magnitude Pruning



The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]

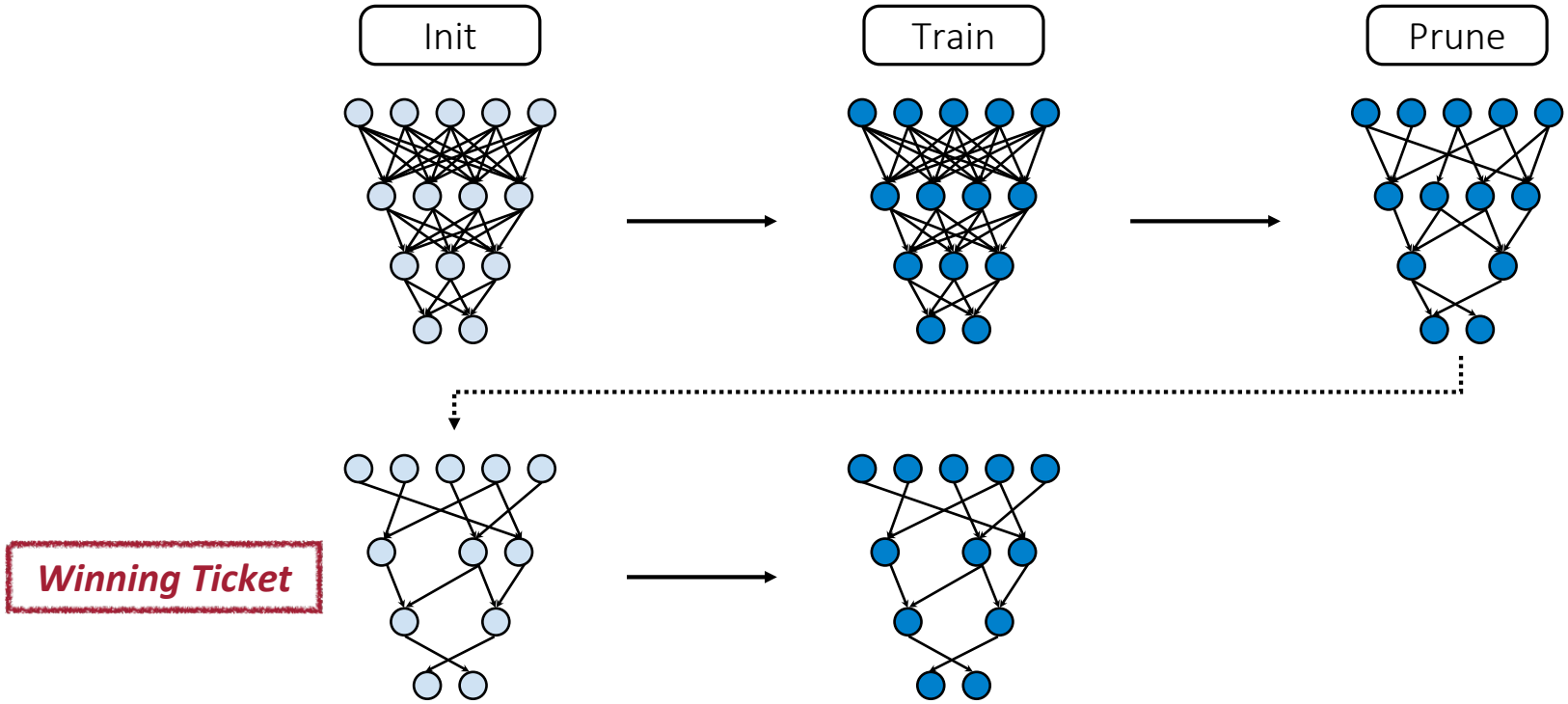
Iterative Magnitude Pruning



The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]

https://efficientml.ai

Iterative Magnitude Pruning

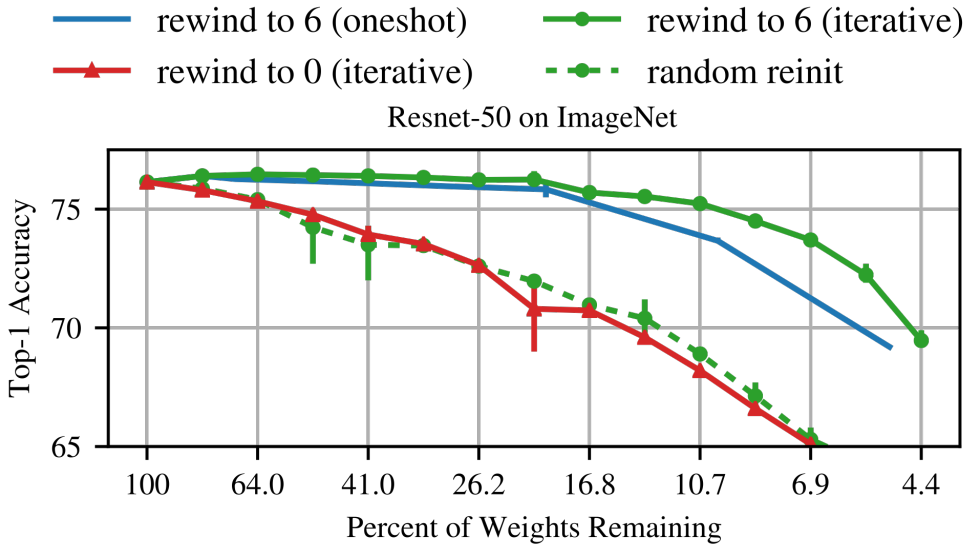


The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]
https://efficientml.ai

Scaling Limitation

- Resetting the weights to the very initial value $\mathbf{W}_{t=0}$ works for small-scale tasks such as MNIST and CIFAR-10, and fails on deep networks.

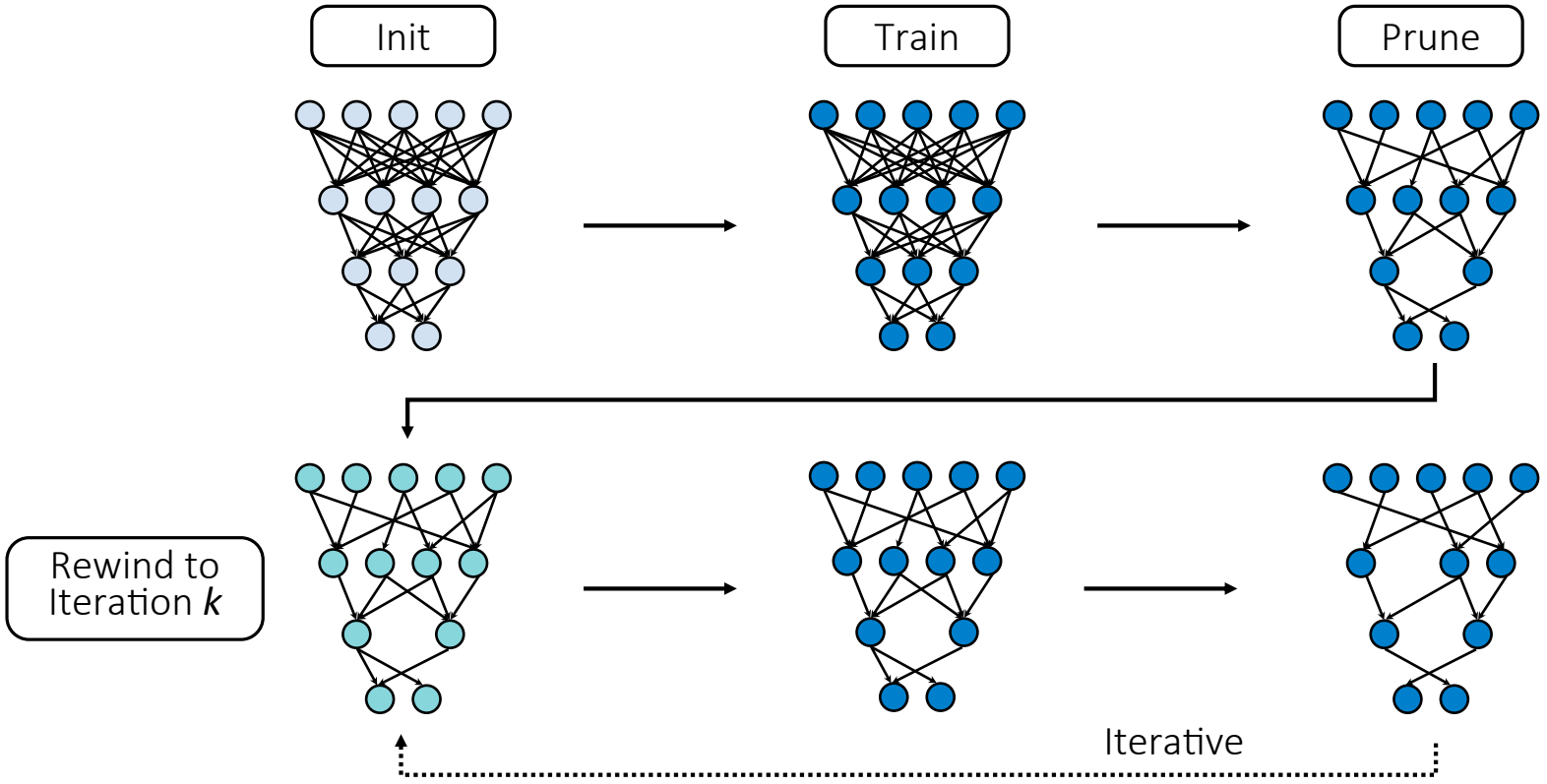
- Instead, it is possible to robustly obtain pruned subnetworks by resetting the weights to the values after ***a small number of k training iterations***, that is $\mathbf{W}_{t=k}$.



Total Training Epochs: $T = 90$
Rewinding to: $k = 6$

Stabilizing the Lottery Ticket Hypothesis [Frankle *et al.*, arXiv 2019]

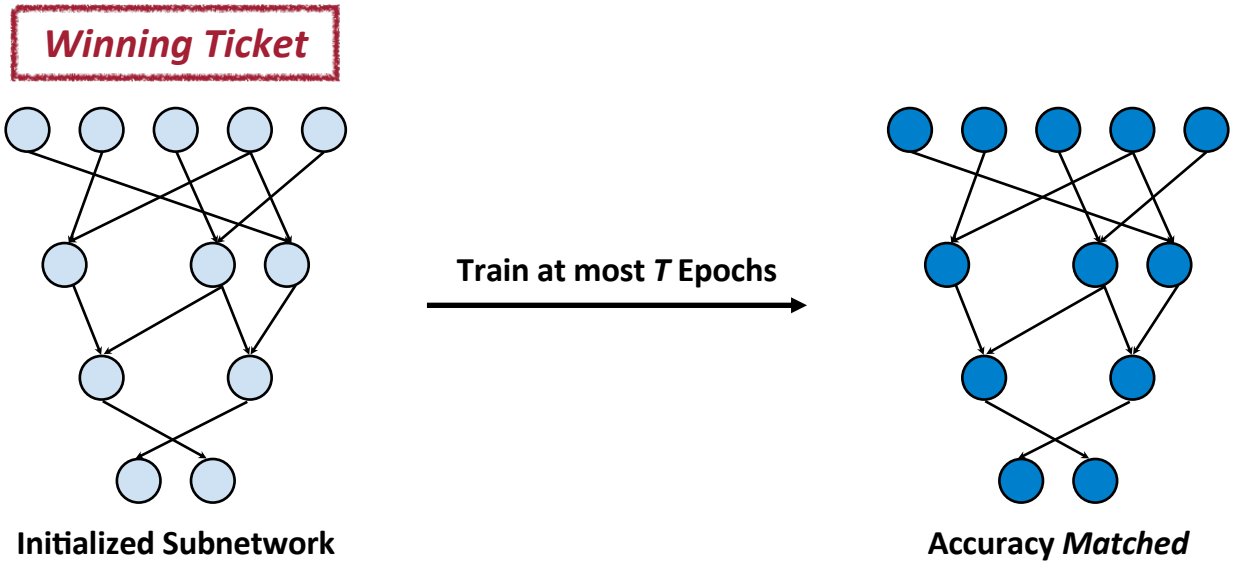
Iterative Magnitude Pruning with Rewinding



Stabilizing the Lottery Ticket Hypothesis [Frankle *et al.*, arXiv 2019]

The Lottery Ticket Hypothesis

*A randomly-initialized, dense neural network contains **a subnetwork** that is initialized such that—when **trained in isolation**—it can **match the test accuracy** of the original network after training for **at most the same number of iterations**.*
—The Lottery Ticket Hypothesis



The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks [Frankle *et al.*, ICLR 2019]
https://efficientml.ai

References

1. Learning Both Weights and Connections for Efficient Neural Network [Han et al., NeurIPS 2015]
2. Exploring the granularity of sparsity in convolutional neural networks [Mao et al., CVPR-W]
3. Learning Structured Sparsity in Deep Neural Networks [Wen et al., NeurIPS 2016]
4. Learning Efficient Convolutional Networks through Network Slimming [Liu et al., ICCV 2017]
5. A Systematic DNN Weight Pruning Framework using Alternating Direction Method of Multipliers [Zhang et al., ECCV 2018]
6. AMC: Automl for Model Compression and Acceleration on Mobile Devices [He et al., ECCV 2018]
7. Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT
8. EIE: Efficient Inference Engine on Compressed Deep Neural Network [Han et al., ISCA 2016]
9. ESE: Efficient Speech Recognition Engine with Sparse LSTM on FPGA [Han et al., FPGA 2017]
10. Block Sparse Format [NVIDIA, 2021]
11. Accelerating Sparse Deep Neural Networks [Mishra et al., arXiv 2021]

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]

Acknowledgments:

Part of this material has been adapted from the MIT efficient Deep Learning class by Prof. Song Han, Quantization Neural Network white paper (https://arxiv.org/pdf/2106.08295.pdf), and Floran de Putter lecture