Accelerator Architectures for Machine Learning

Lecture 4: Pruning & Sparsity

Tuesday: 3:30 – 6:20 pm

Classroom: ED-302

Acknowledgements and Disclaimer

- Slides was developed in the reference with
  Joel Emer, Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, ISCA 2019 tutorial
  Efficient Processing of Deep Neural Network, Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, Morgan and Claypool Publisher, 2020
  Yakun Sophia Shao, EE290-2: Hardware for Machine Learning, UC Berkeley, 2020
  CS231n Convolutional Neural Networks for Visual Recognition, Stanford University, 2020
- 6.5940, TinyML and Efficient Deep Learning Computing, MIT
- NVIDIA, Precision and performance: Floating point and IEEE 754 Compliance for NVIDIA GPUs, TB-06711-001_v8.0, 2017

Outline

- Neural Network Pruning
- Pruning granularity
- Pruning criterion
- Pruning ratio
- Fine-tune/train pruned neural network

Pruning Happens in Human Brain

- **Neural Network Pruning**
  - Reduce the network connections
  - Small weight while maintaining training accuracy

50 Trillion Synapses

New born

1000 Trillion Synapses

1 year old

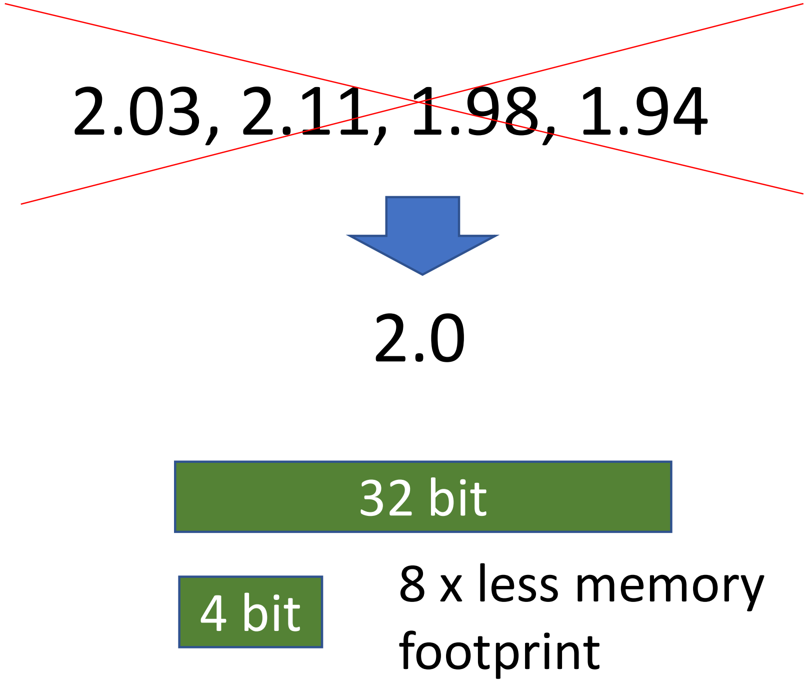
500 Trillion Synapses

Teenager

Christopher A Walsh, Peter Huttenlocher (1931 - 2013). Nature, 502(7470), 2013

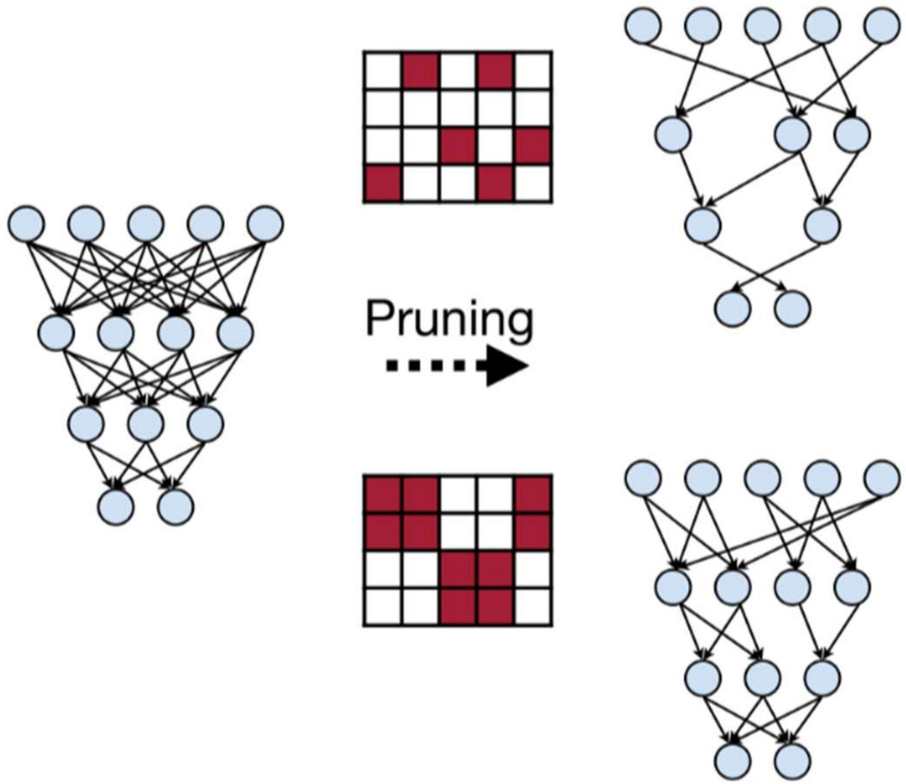
Approaches to Reduce Model Sizes

- **Weight sharing**
  - Trained quantization
- **Quantization**
  - Quantizing the weight and activation
  - Fine-tune in float format
  - Reduce to fixed-point format



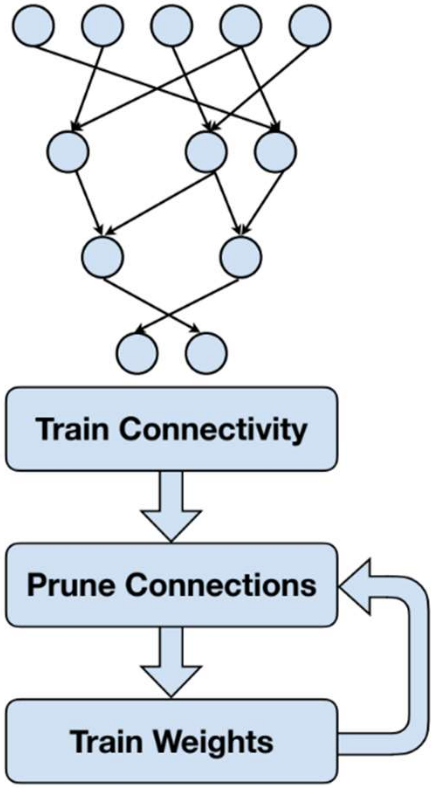
What is Neural Network Pruning ?

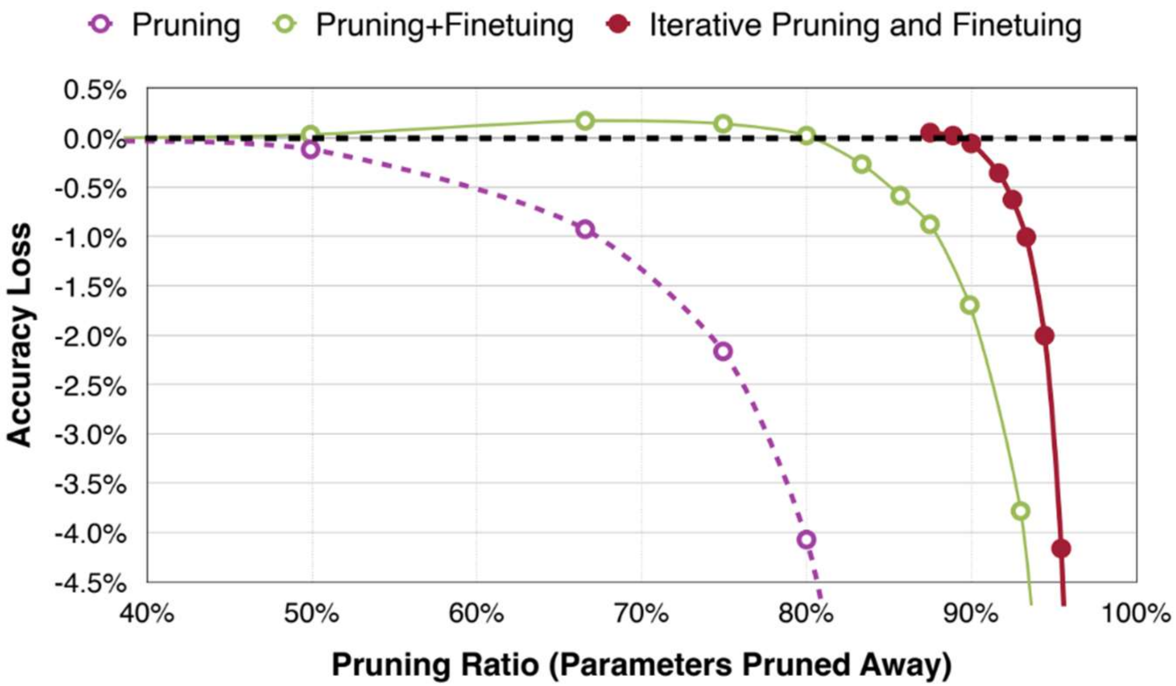
- **Neural Network Pruning**
  - Reducing the **parameter counts** of neural networks
  - Decreasing the **storage requirements**
  - Improving **computation efficiency** of neural network



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

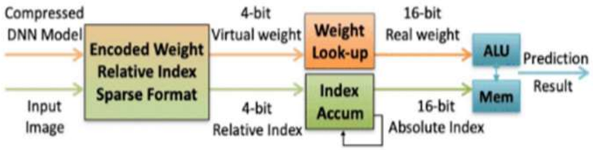
- Make neural network smaller by removing synapses and neurons

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 ×

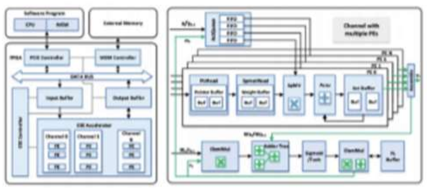
Efficient Methods and Hardware for Deep Learning [Han S., Stanford University]

Pruning in the Industry

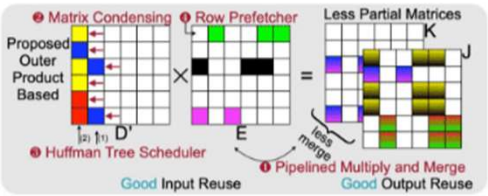
- Hardware support for sparsity



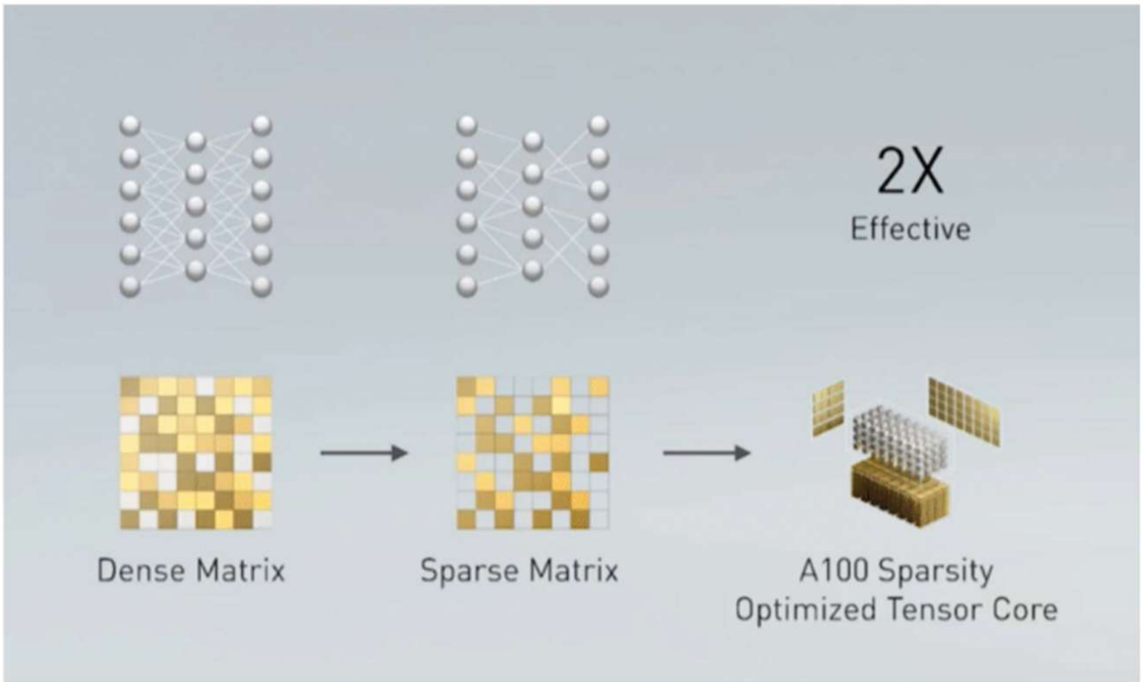
EIE [Han *et al.*, ISCA 2016]



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

Neural Network Pruning

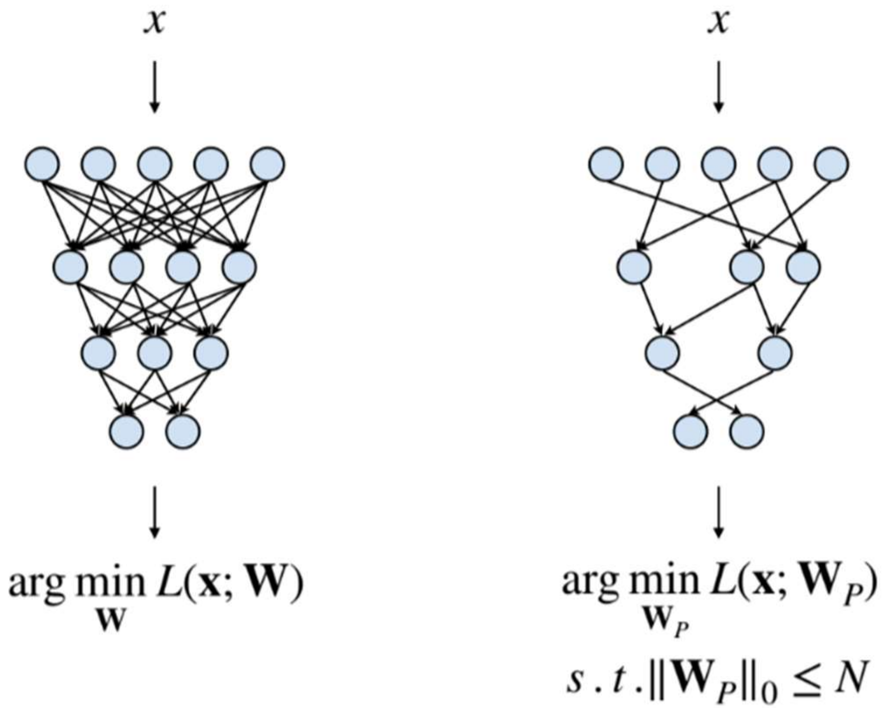
- In general, we could formulate the pruning as follows:

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

subject to

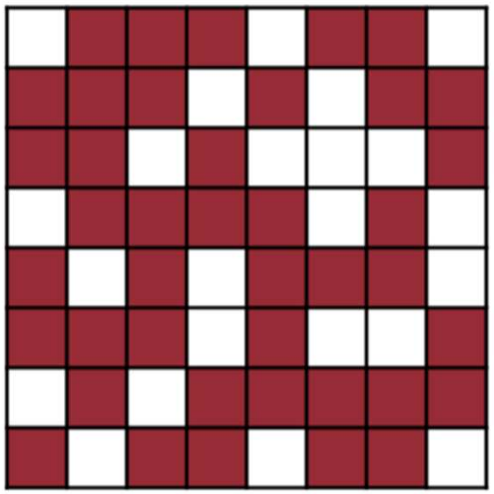
$$\|\mathbf{W}_p\|_0 < N$$

  - 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.



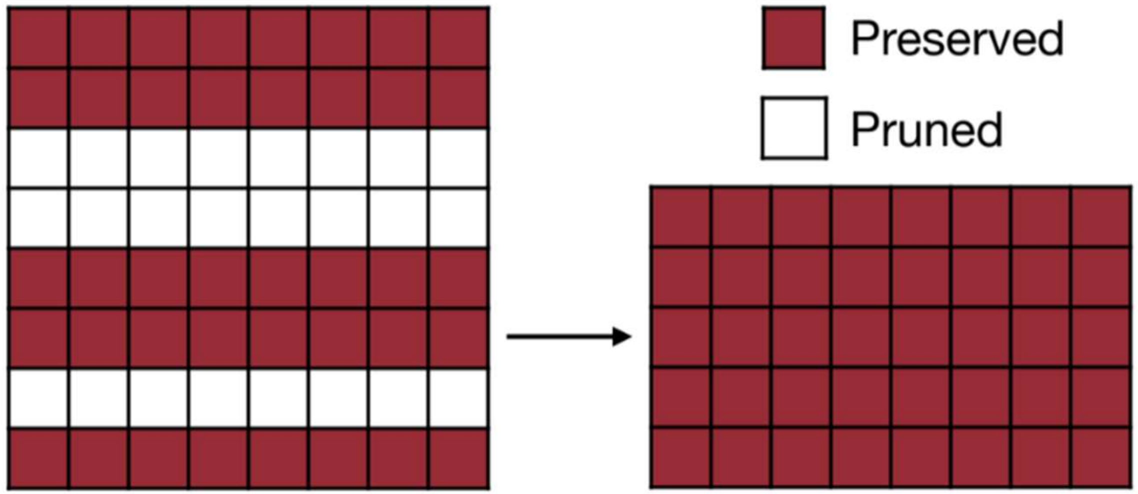
Pruning at Different Granularities

- 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

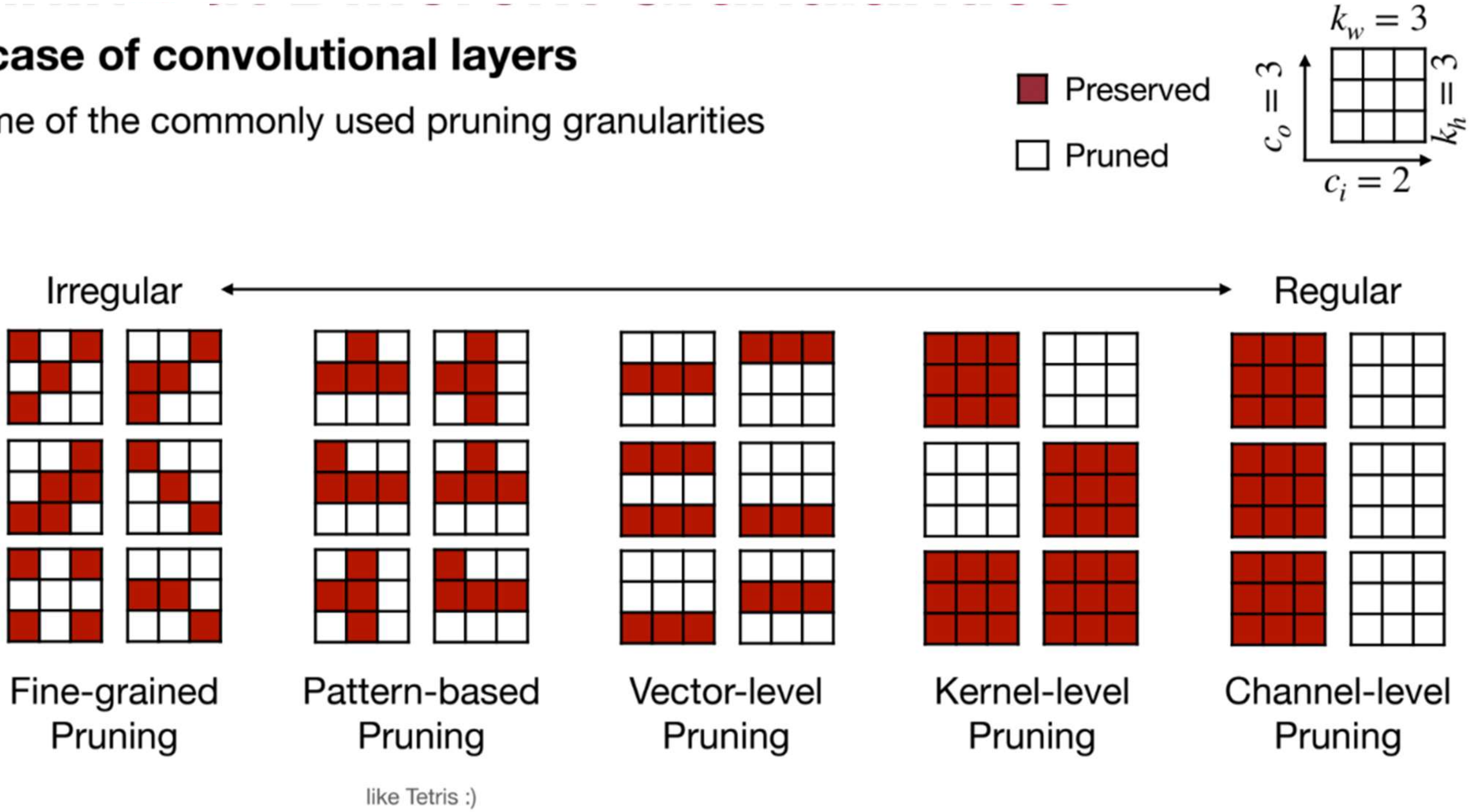
- The weights of convolutional layers have 4 dimensions $[c_o, c_i, k_h, k_w]$:
  - c_i: input channels (or channels)
  - c_o: output channels (or filters)
  - k_h: kernel size height
  - k_w: kernel size width

- 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



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

Pruning at Different Granularities

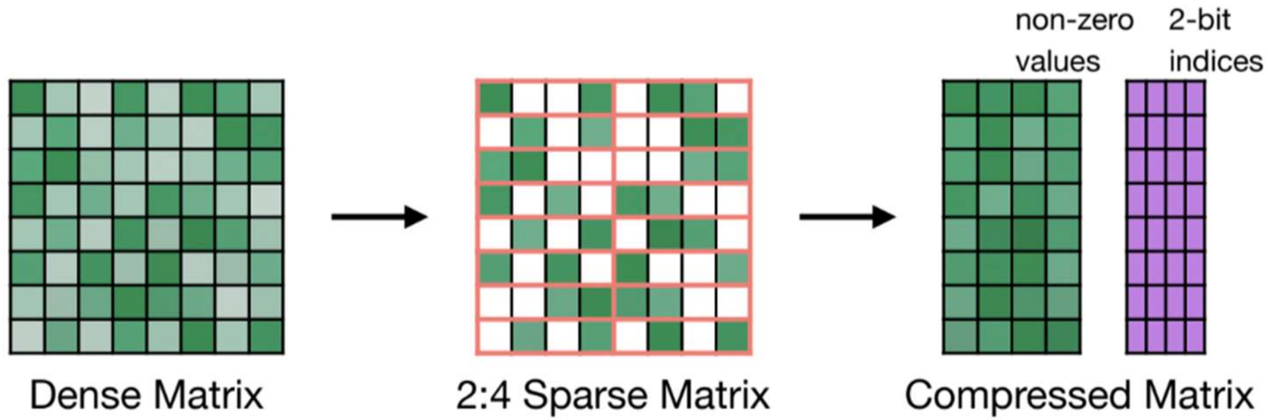
- **Fine-grained pruning**
  - Flexible pruning indices
  - Large compression ratio (flexibly find redundant weight)
  - Can deliver speedup on some customized hardware (EIE), but not GPU

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

- **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, 2X speedup



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

Pruning at Different Granularities

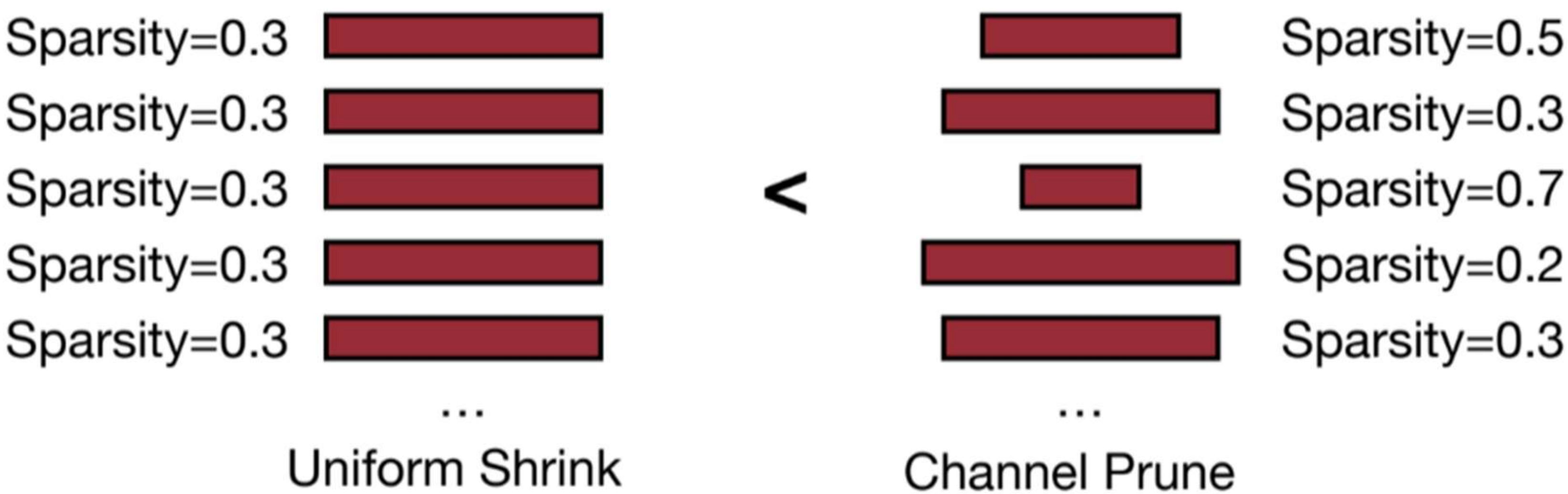
- **Pattern-based pruning: N:M sparsity**
  - Usually maintains accuracy

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

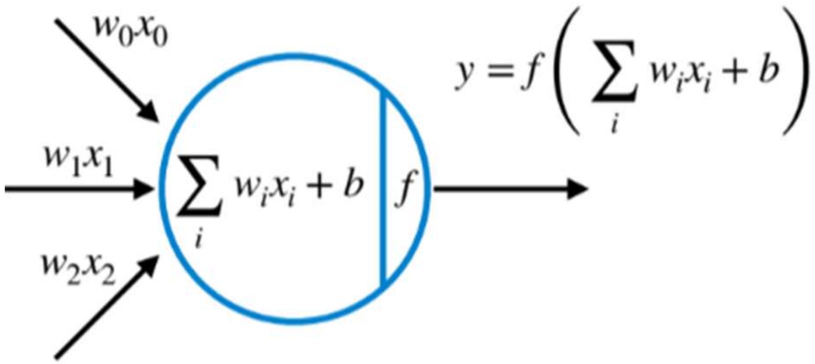
Pruning at Different Granularities

- **Channel pruning**
  - Reduce channel numbers (leading to an neural network with smaller # of channels) -> speedup
  - Con: smaller compression ratio



Pruning Criterion

- What synapses and neurons should we prune ?
  - The less important parameters should be removed
  - What is the less important parameter in a neural network?



Example

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

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

- If one weight will be removed, which one?

Magnitude-based Pruning

- **Magnitude-based pruning**
  - Considers weights with **large absolute values** are more important than other weights
  - Remove weights with small magnitudes

$$\boxed{Importance = |W|}$$

$$\underset{\text{fmap}}{\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}} * \underset{\text{filter}}{\begin{bmatrix} -8 & 3 & 2 \\ 1 & -3 & -2 \\ 1 & 1 & 1 \end{bmatrix}} = -4$$

Without Pruning

$$\underset{\text{fmap}}{\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}} * \underset{\text{filter}}{\begin{bmatrix} -8 & 3 & 2 \\ 0 & -3 & -2 \\ 0 & 0 & 0 \end{bmatrix}} = -8$$

Error = -4

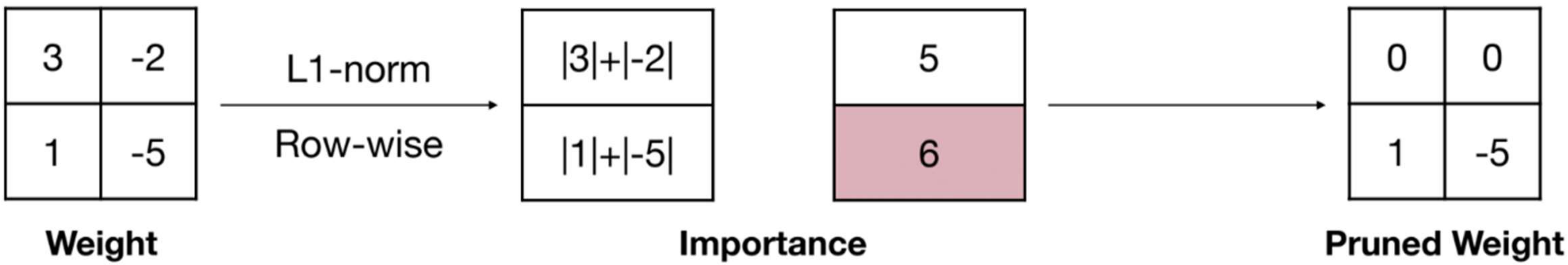
Magnitude-based Pruning

Magnitude-based Pruning

- **Row-wise pruning**
    - The L1-norm magnitude can be defined as

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

Example

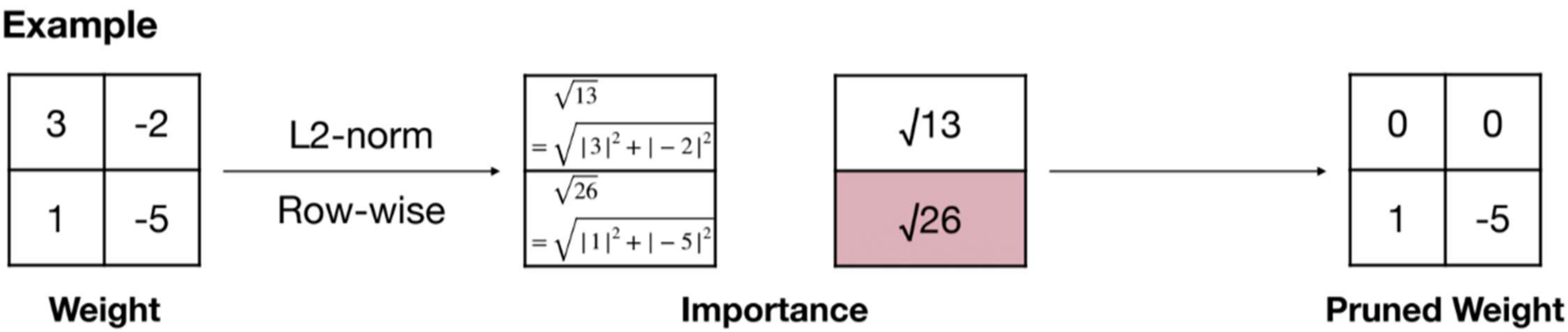


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Magnitude-based Pruning

- **A heuristic pruning criterion**
  - The Lp-norm magnitude can be defined as

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



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

Feature-Based Pruning

- **Feature-based pruning**
  - Pruning based on the impact of the output feature map
  - Achieve higher accuracy than magnitude-based pruning
  - Complex evaluating the impact of the weights

$$\overset{\text{fmap}}{\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}} * \overset{\text{filter}}{\begin{bmatrix} -8 & 3 & 2 \\ 1 & -3 & -2 \\ 1 & 1 & 1 \end{bmatrix}} = -4$$

Without Pruning

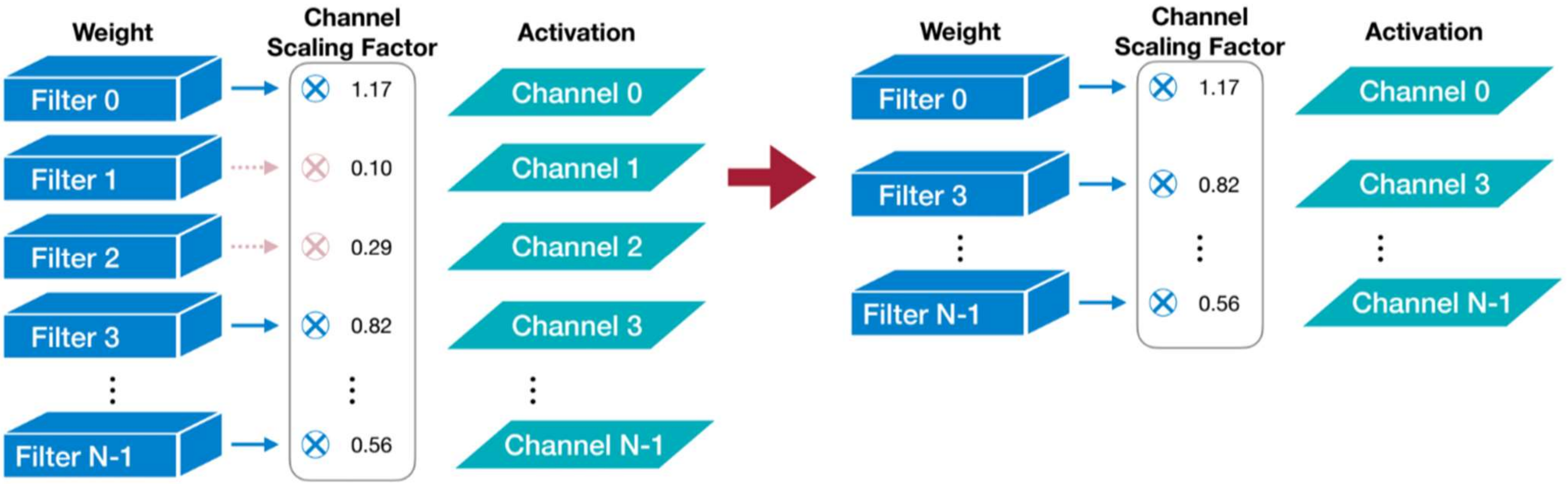
$$\overset{\text{fmap}}{\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}} * \overset{\text{filter}}{\begin{bmatrix} -8 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}} = -4$$

Error = 0

Feature-based Pruning

Scaling-based Pruning

- **A scaling factor**
  - Associated with each filter in convolutional layers
  - Trainable parameter
  - The filters/output channels with small scaling factor magnitude will be pruned

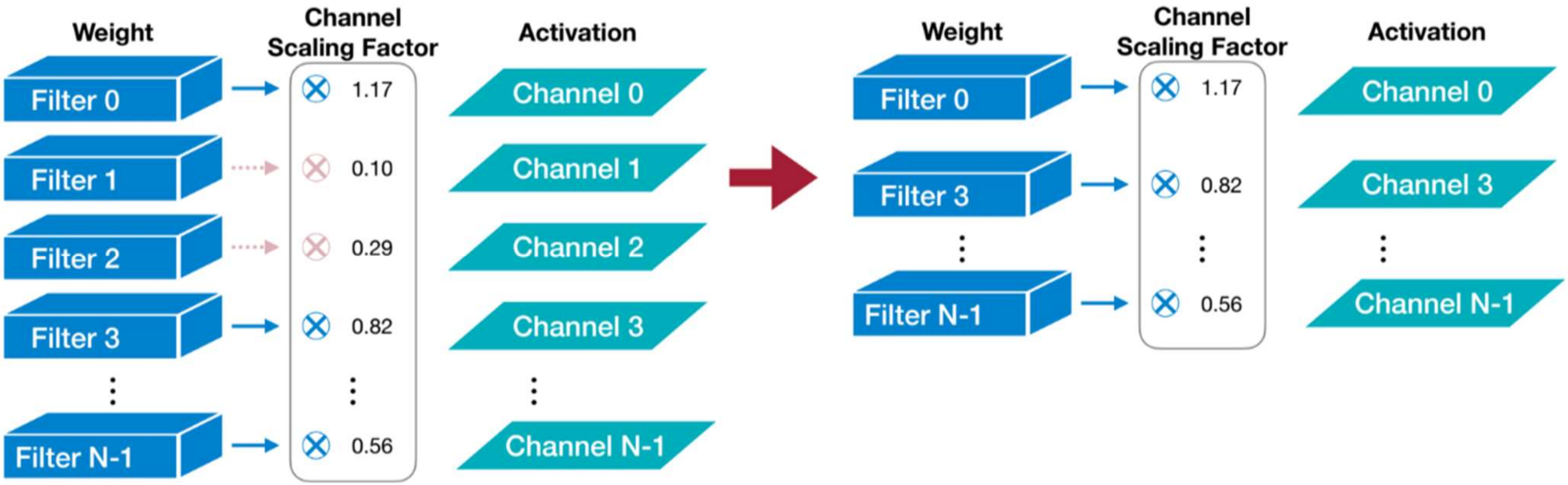


Learning Efficient Convolutional Networks through Network Slimming [Liu *et al.*, ICCV 2017]

Scaling-based Pruning

- **A scaling factor**
  - The scaling factor can be used from batch normalization layer

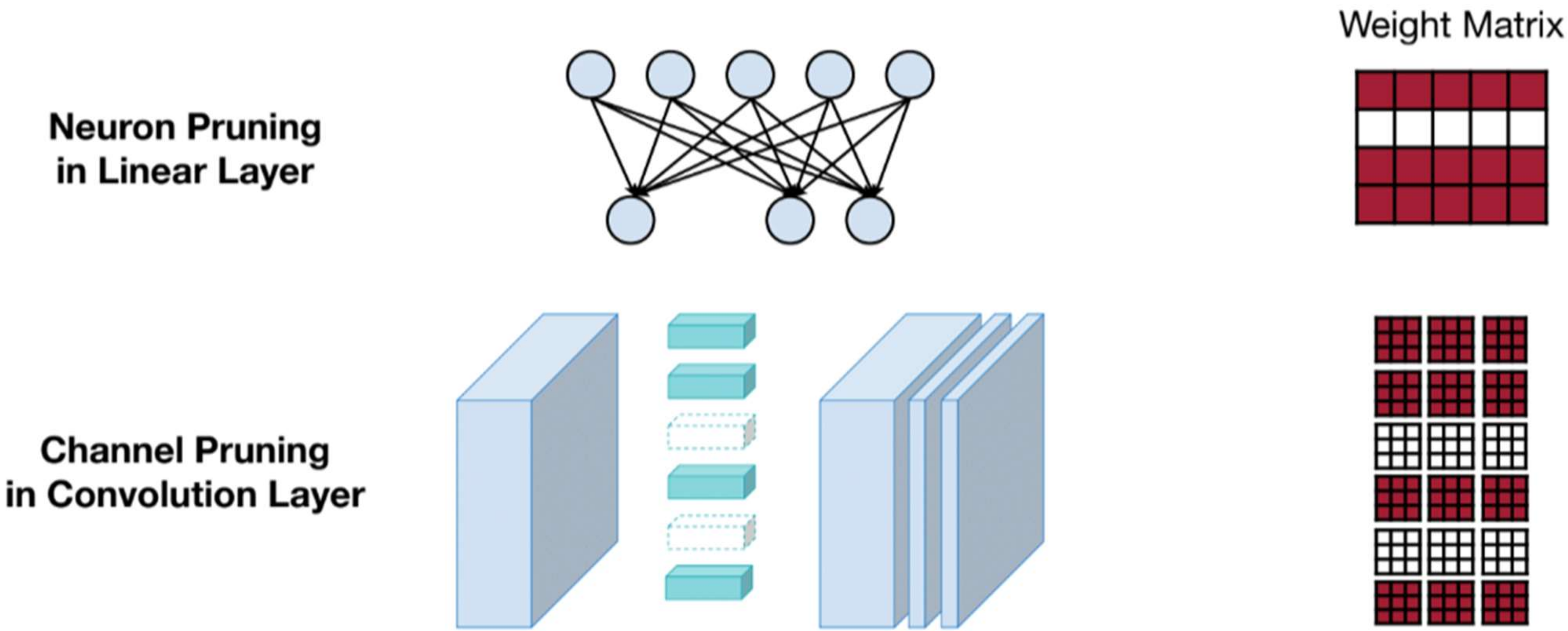
$$\mathbf{z}_o = \gamma \frac{\mathbf{z}_i - \mu_{\mathscr{B}}}{\sqrt{\sigma_{\mathscr{B}}^2 + \epsilon}} + \beta$$



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

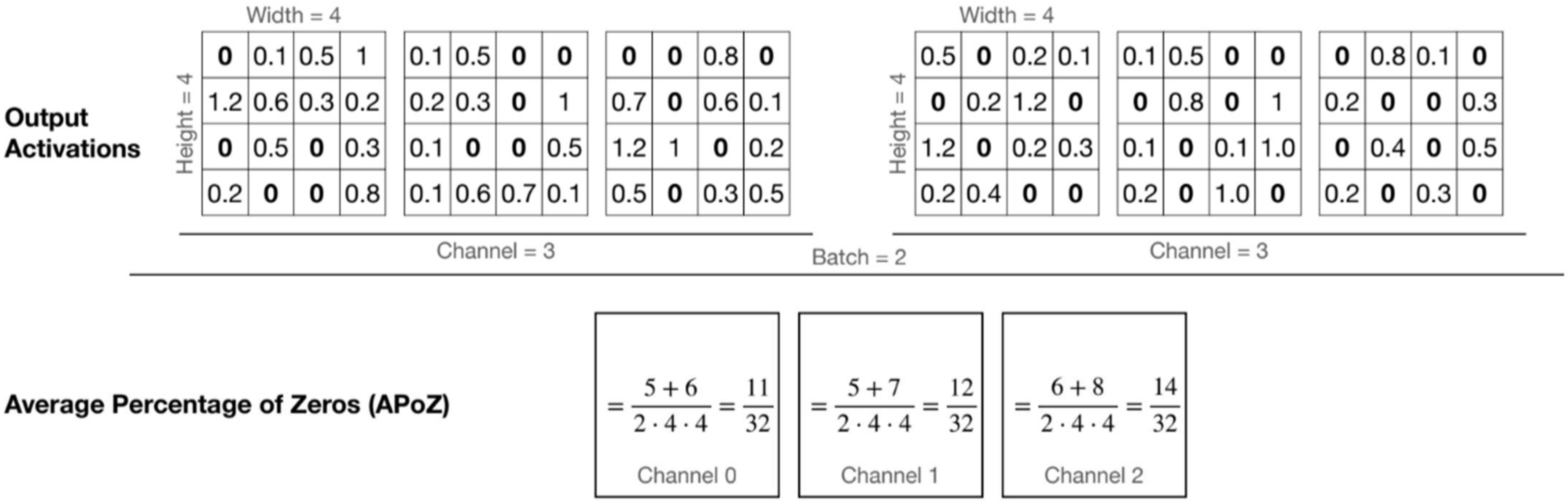
Pruning Neurons

- When removing neurons from a neural network model
  - The less useful neurons are removed



Percentage-of-Zero-Based Pruning

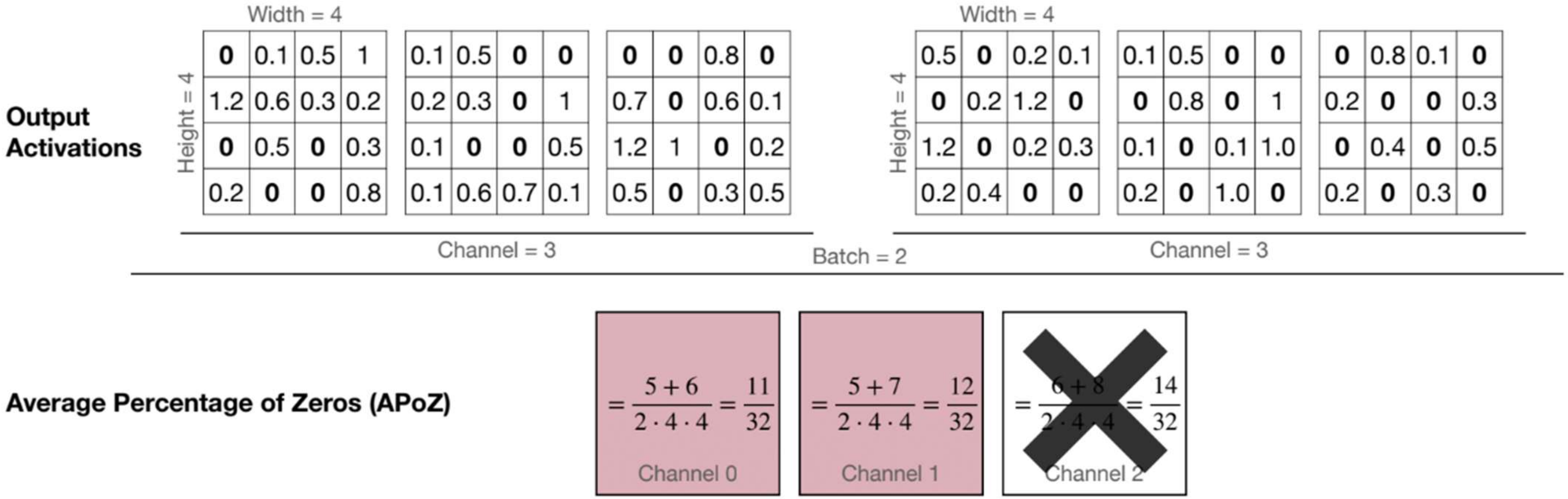
- ReLU activation will generate zeros in the output activation
- The Average Percentage of Zero activations (APoZ) can be exploited to measure the importance of the neurons



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

Percentage-of-Zero-Based Pruning

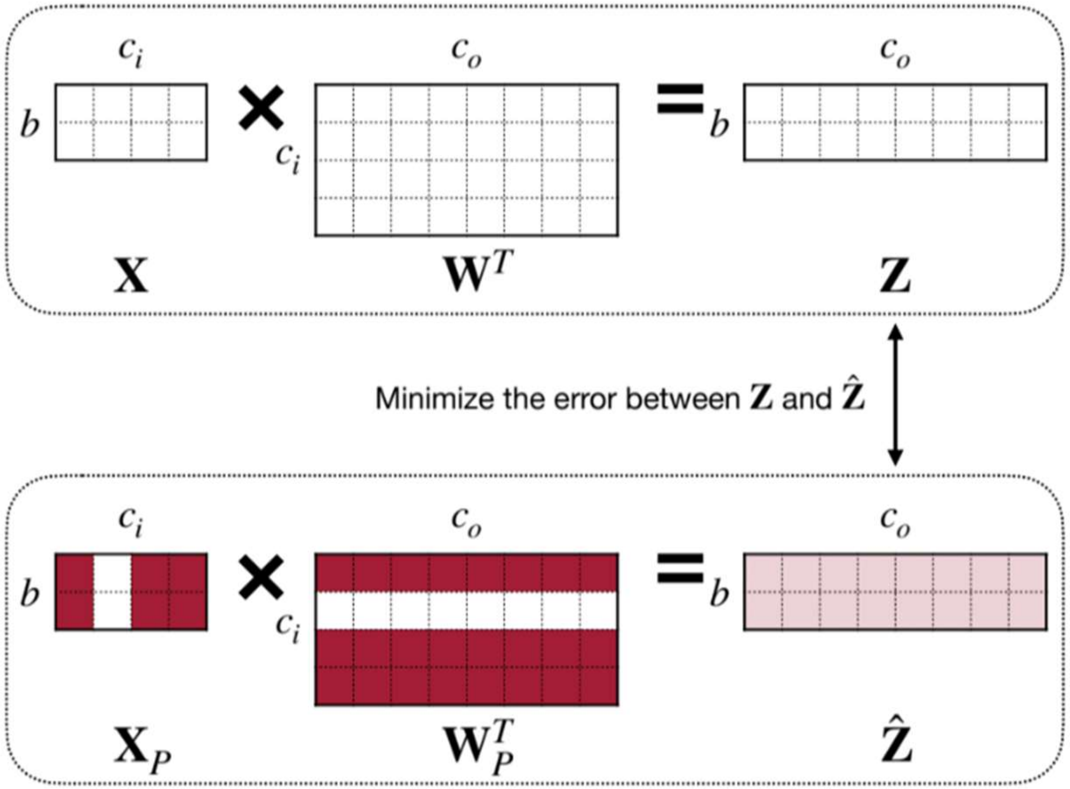
- The Average Percentage of Zero activations (APoZ) can be exploited to measure the importance of the neurons
- The neuron with smaller APoZ is more important



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



Regression-based Pruning

- Let

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

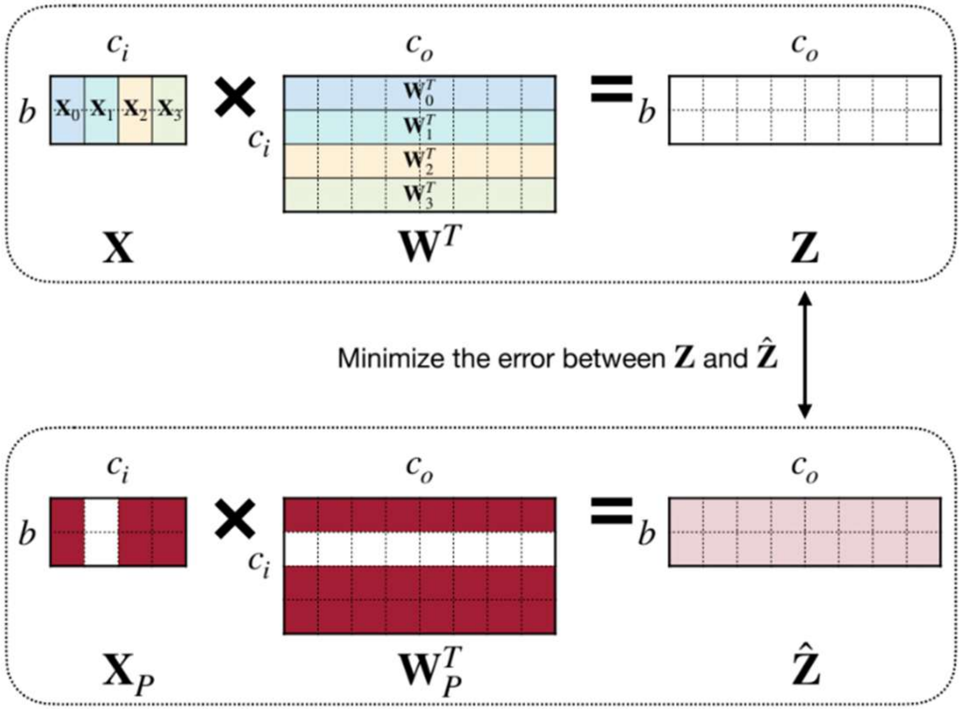
- The problem can be formulate as

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

  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.
- 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]

Takeaway Questions

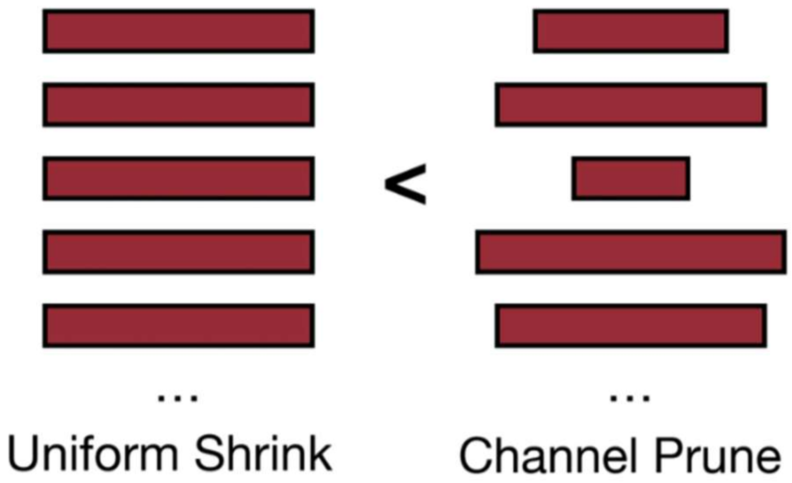
- How does feature-based pruning work?
  - (A) Removing weights with small magnitudes
  - (B) Pruning through complex evaluation
  - (C) Removing inputs with small magnitudes
- What are goals of neural network pruning ?
  - Less number of weights
  - Less number of inputs
  - Less bits per weights

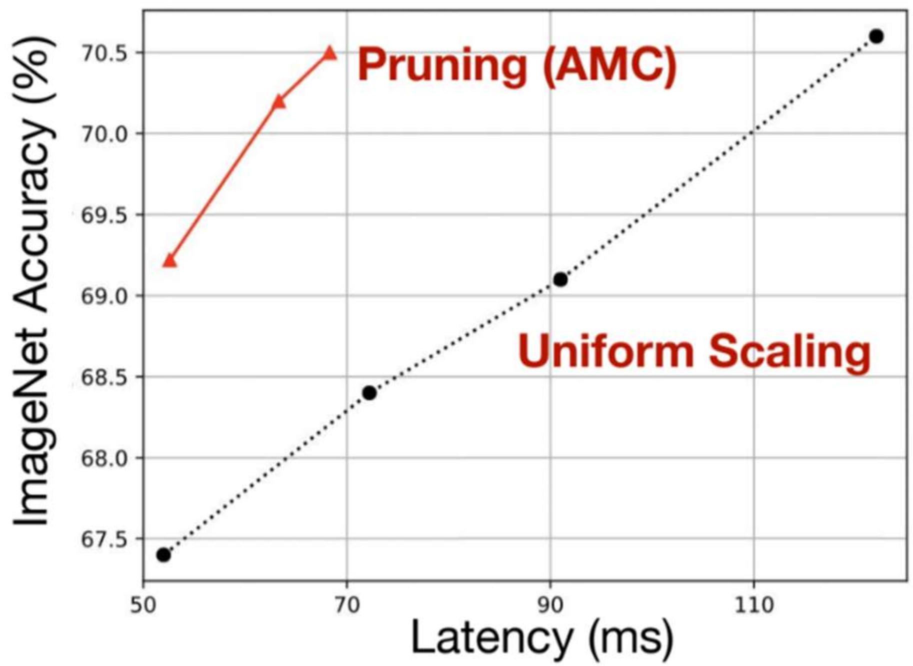
Takeaway Questions

- What are benefits of network pruning ?
  - Reduce the size of input data
  - Small size of filter data
  - Shorten the time to complete the DNN model inference

Pruning Ratio

- How should we find per-layer pruning ratios ?
  - Non-uniform pruning is better than uniform shrinking



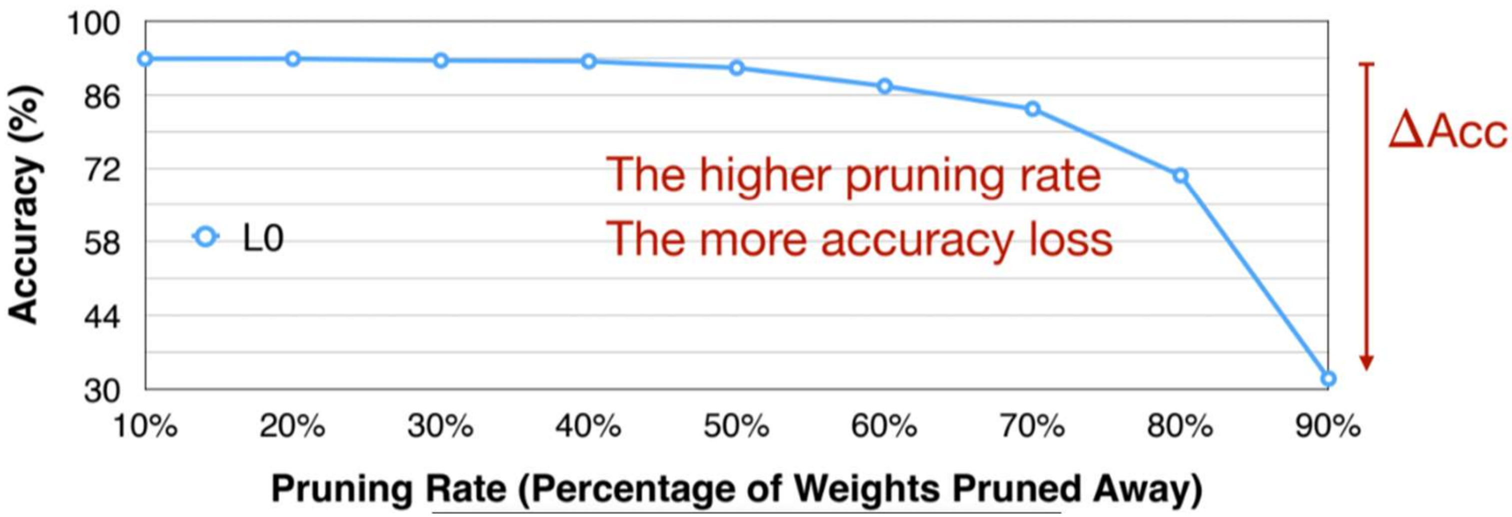


Finding Pruning Ratios

- **Analyze the sensitivity of each layer**
  - Pruning ratios are varied across different layers
  - Some layers are more sensitive (e.g., first layer, why?)
  - Some layers are more redundant
  - Need to perform sensitivity analysis to determine the per-layer pruning ratio

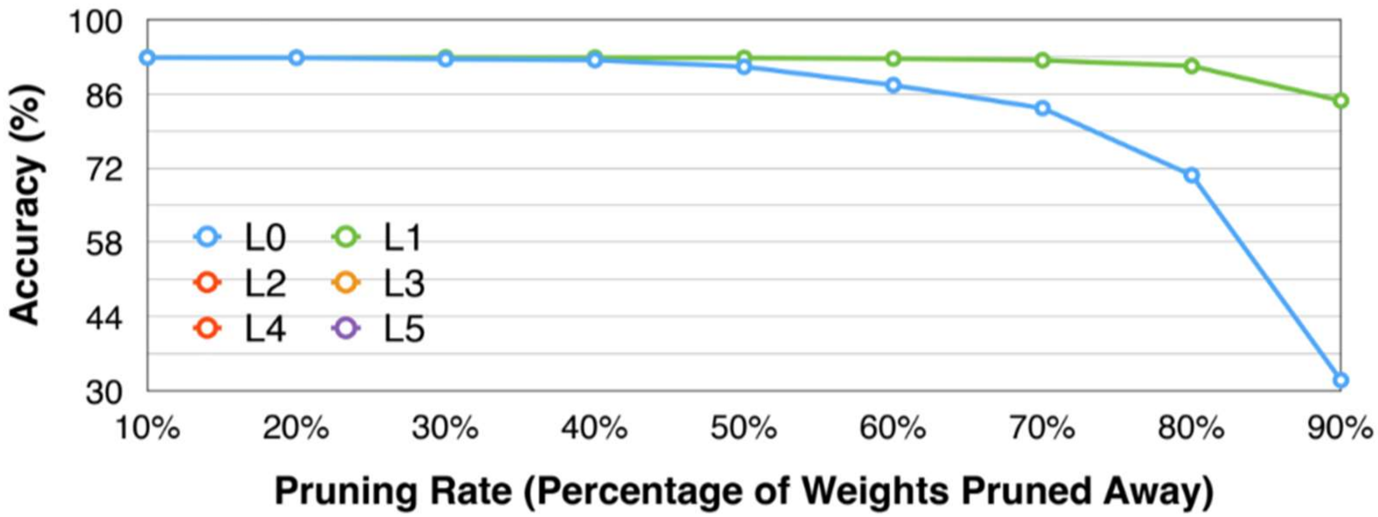
Finding Pruning Ratios

- The process of Sensitivity Analysis (* VGG-11 on CIFAR-10 dataset)
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, ..., 0.9\}$ (or other strides)
    - Observe the accuracy degrade $\Delta \mathrm{Acc}_r^i$ for each pruning ratio



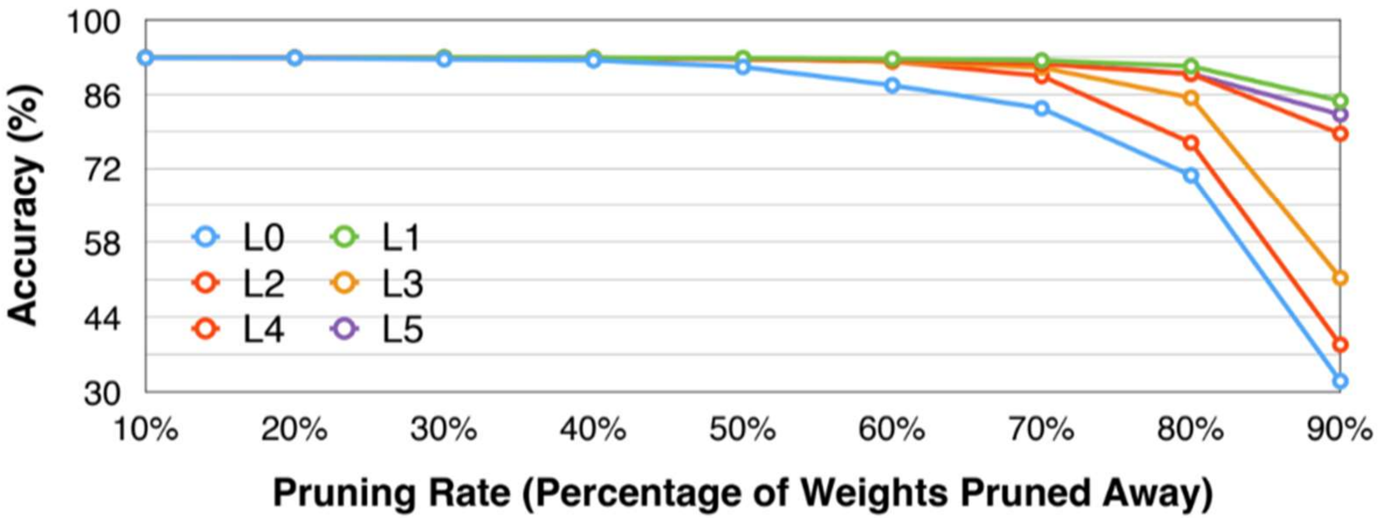
Finding Pruning Ratios

- The process of Sensitivity Analysis (* VGG-11 on CIFAR-10 dataset)
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, ..., 0.9\}$ (or other strides)
    - Observe the accuracy degrade $\Delta \mathrm{Acc}_r^i$ for each pruning ratio
  - Repeat the process for all layers



Finding Pruning Ratios

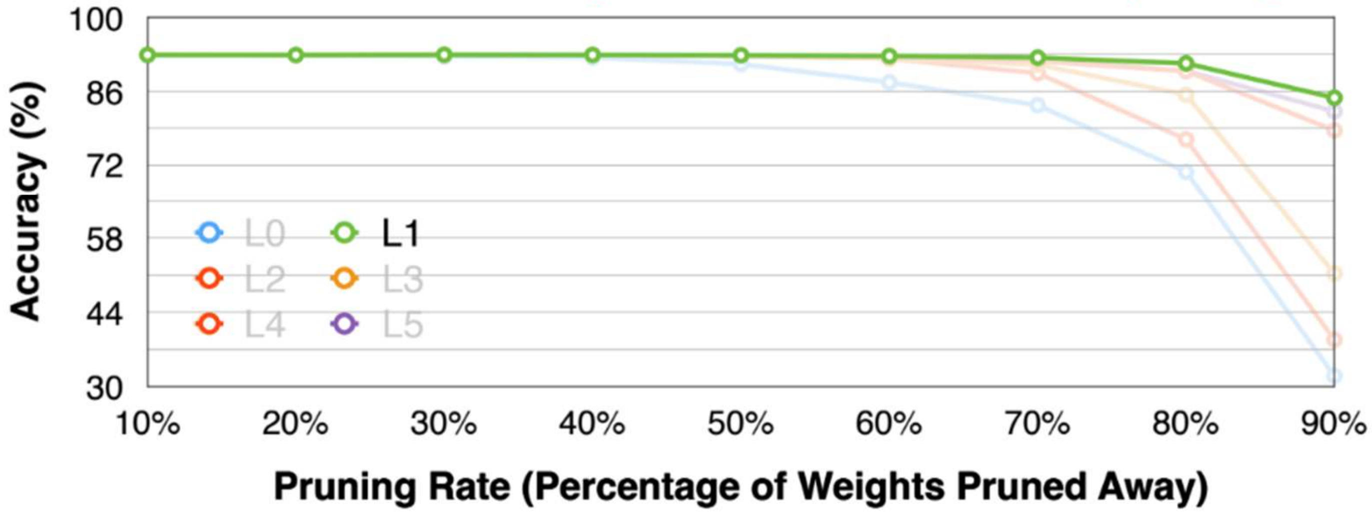
- The process of Sensitivity Analysis (* VGG-11 on CIFAR-10 dataset)
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0,0.1,0.2,...,0.9\}$ (or other strides)
    - Observe the accuracy degrade $\Delta \mathrm{Acc}_r^i$ for each pruning ratio
  - Repeat the process for all layers



Finding Pruning Ratios

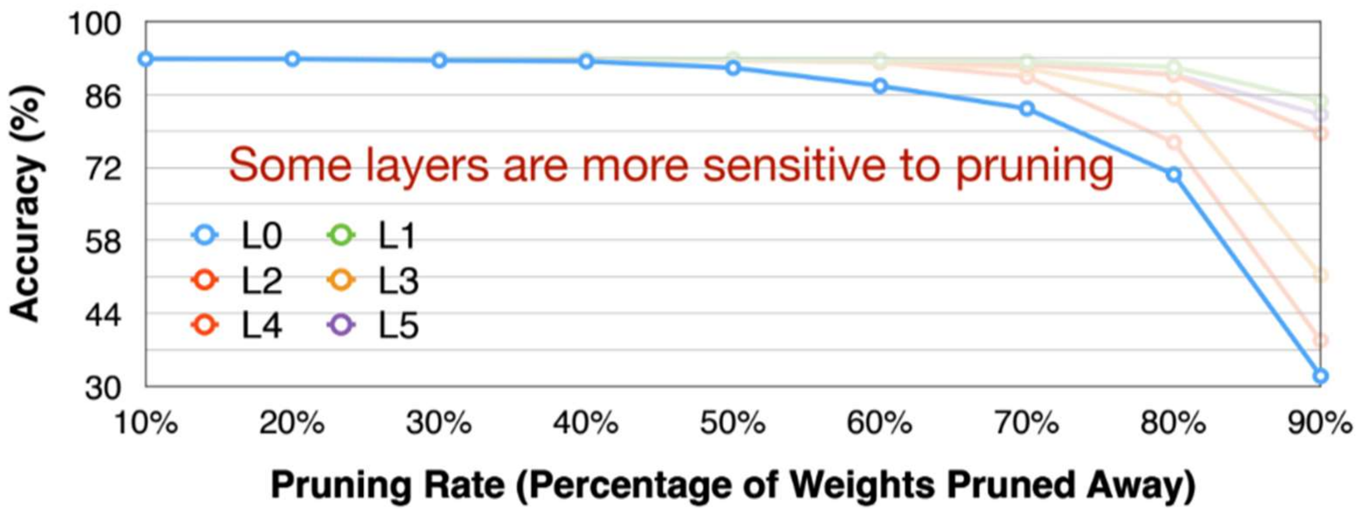
- The process of Sensitivity Analysis (* VGG-11 on CIFAR-10 dataset)
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0,0.1,0.2,...,0.9\}$ (or other strides)
    - Observe the accuracy degrade $\Delta \mathrm{Acc}_r^i$ for each pruning ratio
  - Repeat the process for all layers

Some layers are less sensitive to pruning



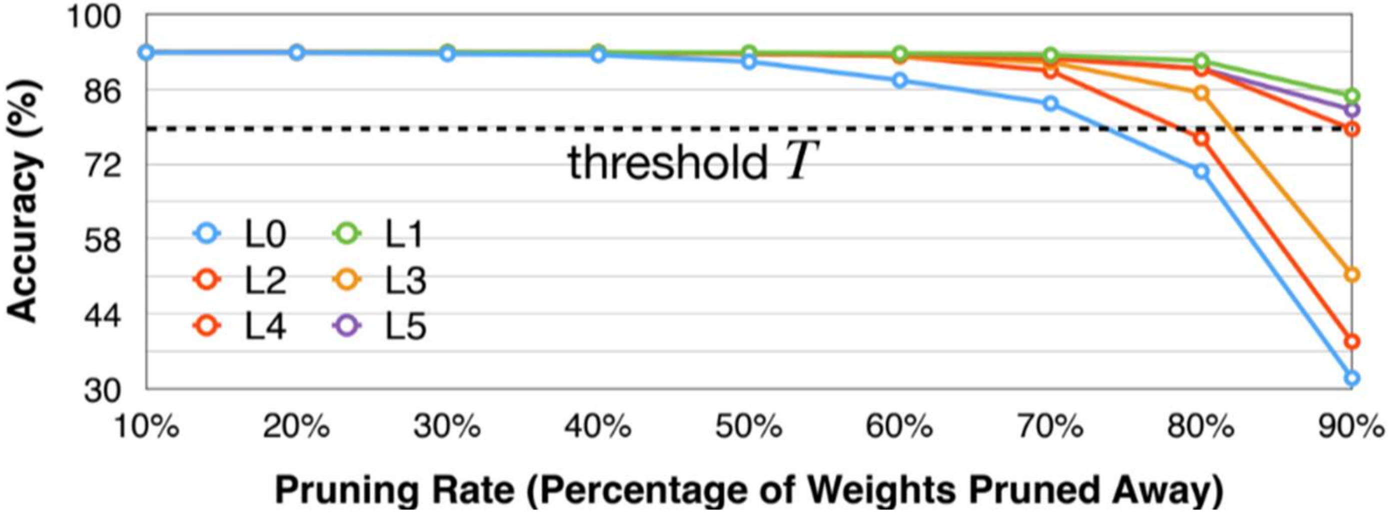
Finding Pruning Ratios

- The process of Sensitivity Analysis (* VGG-11 on CIFAR-10 dataset)
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, ..., 0.9\}$ (or other strides)
    - Observe the accuracy degrade $\Delta \mathrm{Acc}_r^i$ for each pruning ratio
  - Repeat the process for all layers



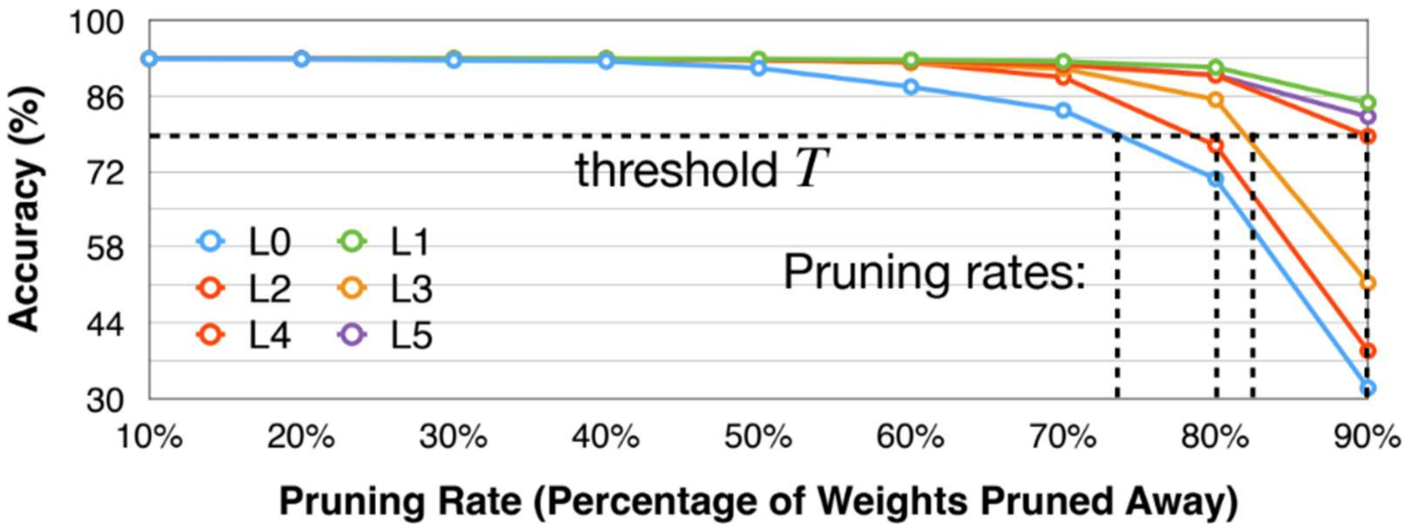
Finding Pruning Ratios

- The process of Sensitivity Analysis (* VGG-11 on CIFAR-10 dataset)
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, ..., 0.9\}$ (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

- The process of Sensitivity Analysis (* VGG-11 on CIFAR-10 dataset)
  - Pick a layer L_i in the model
    - Prune the layer L_i with pruning ratio $r \in \{0, 0.1, 0.2, ..., 0.9\}$ (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

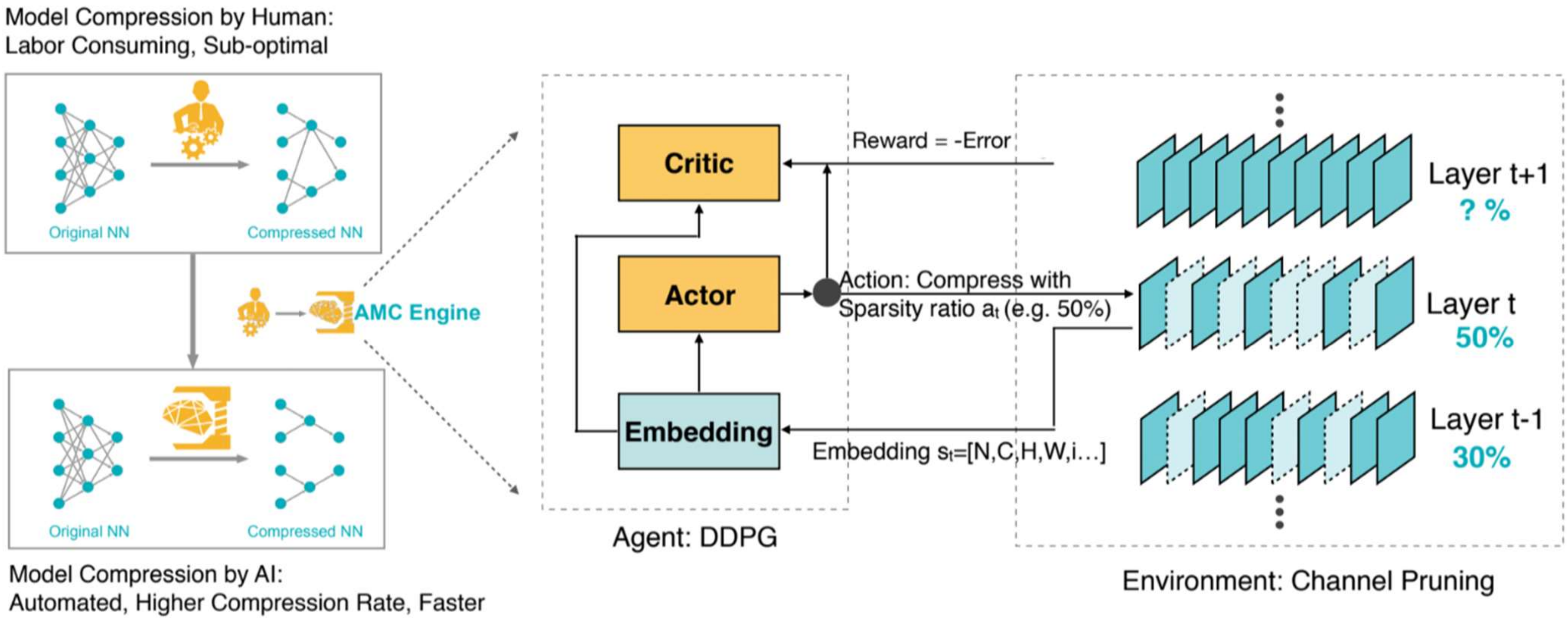


Automatic Pruning

- Given an **overall** compression ratio, how do we **choose per-layer** pruning ratios ?
  - Sensitivity analysis ignores the interaction between layers
  - Conventionally, such process relies on human expertise and trails and errors

AMC: AutoML for Model Compression

- Pruning as a reinforcement learning problem



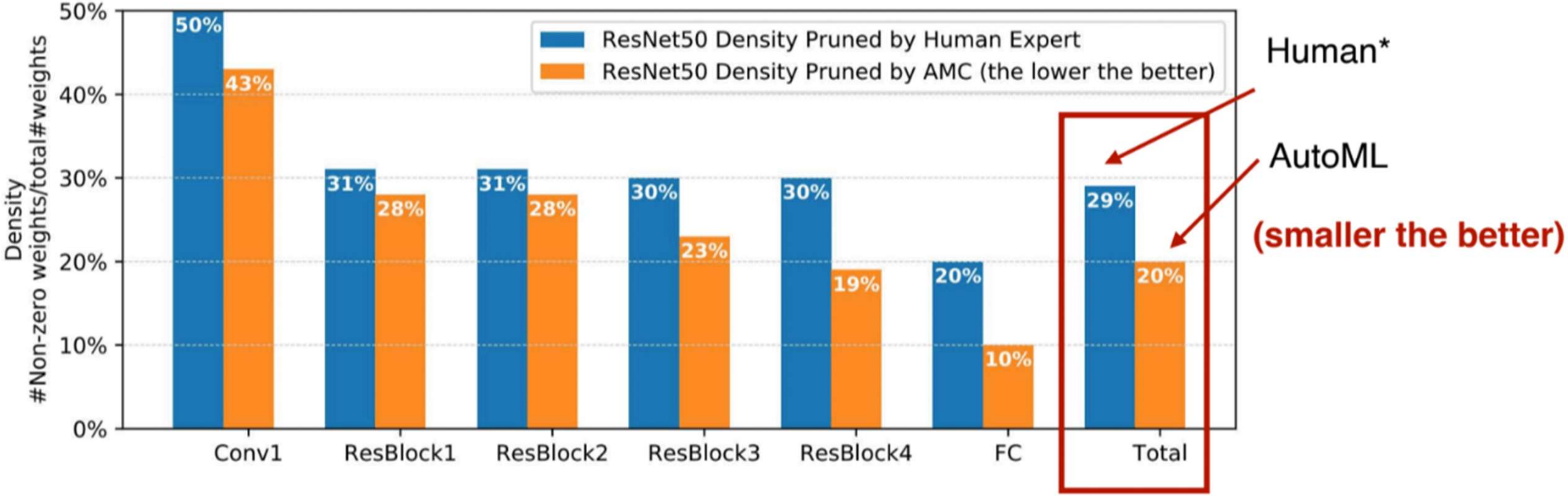
AMC: AutoML for Model Compression and Acceleration on Mobile Devices [He *et al.*, ECCV 2018]

AMC: AutoML for Model Compression

- AMC uses the following steps for the reinforcement learning problem
  - **State**: 11 features (including layer indices, channel numbers, kernel sizes, FLOPs, …)
  - **Action:** A continuous number (pruning ratio) $a \in [0,1)$
  - **Agent:** Deep Deterministic Policy Gradient (DDPG) agent, because it supports continuous action output
  - **Reward:**

$$R = \begin{cases} -\text{Error}, & \text{if satisfies constrains} \\ -\infty, & \text{if not} \end{cases}$$

AMC: AutoML for Model Compression



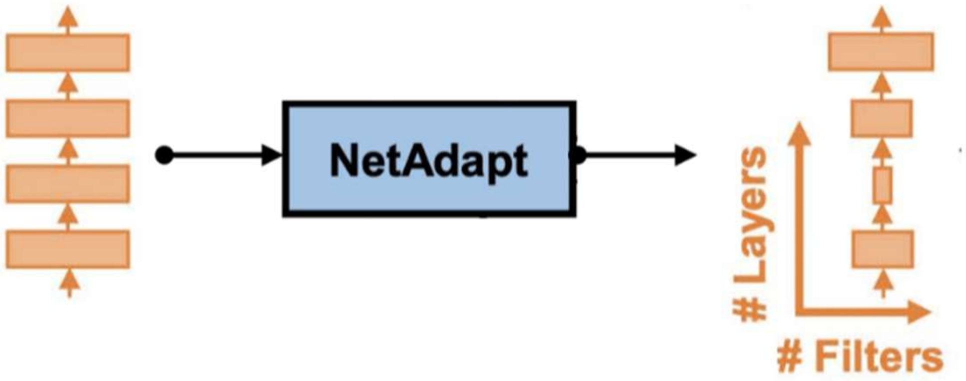
AMC: AutoML for Model Compression

Model	MAC	Top-1	Latency*	Speedup	Memory
1.0 MobileNet	569M	70.6%	119.0ms	1x	20.1MB
AMC (50% **FLOPs**)	285M	**70.5%**	64.4ms	**1.8x**	14.3MB
AMC (50% **Time**)	272M	**70.2%**	59.7ms	**2.0x**	13.2MB
0.75 MobileNet	325M	68.4%	69.5ms	1.7x	14.8MB

* Measured with TF-Lite on Samsung Galaxy S7 Edge, which has Qualcomm Snapdragon SoC
Single core, Batch size = 1(mobile, latency oriented)

NetAdapt

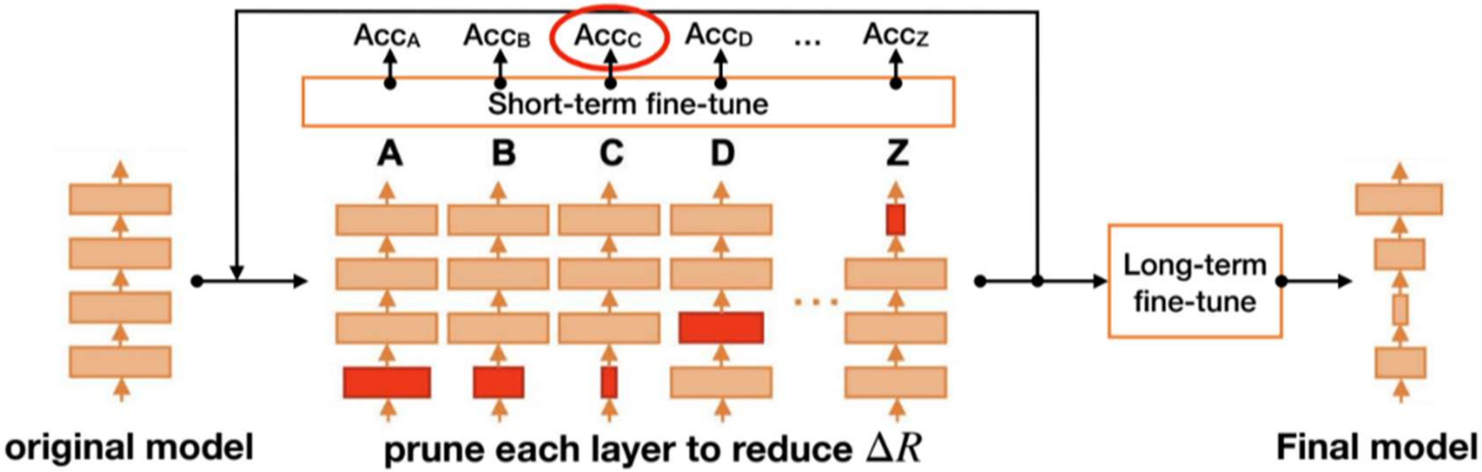
- **A rule-based iterative/progressive method**
  - Aim to find a per-layer pruning ratio to meet a global resource constraint (e.g., latency, energy, …)
  - The process is done iteratively



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

NetAdpt

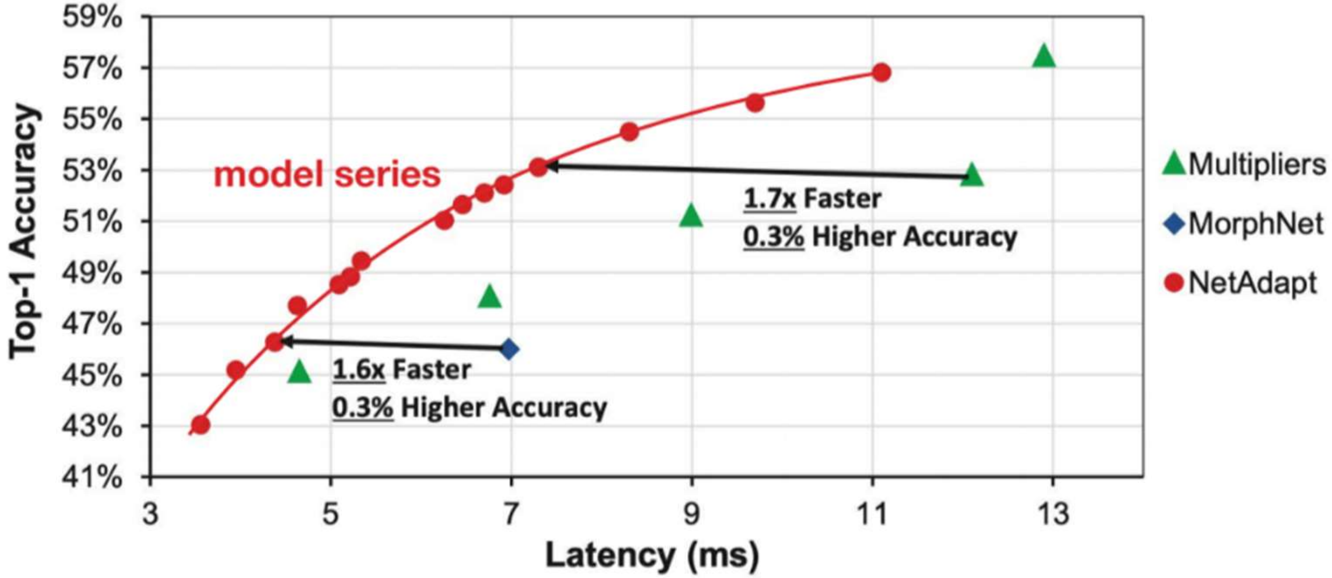
- For each iteration, we aim to reduce the latency by a certain amount ΔR (manually defined)
  - For each layer L_k (k in A-Z in the figure)
    - Prune the layer s.t. the latency reduction meets ΔR (based on a pre-built lookup table)
    - Short-term fine-tune model (10k iterations); measure accuracy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

NetAdpt

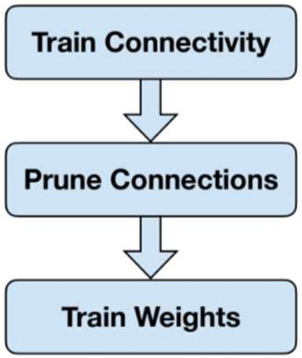
- The iterative nature allows us to obtain **a serial of** models with different costs
  - # of models = # of iterations

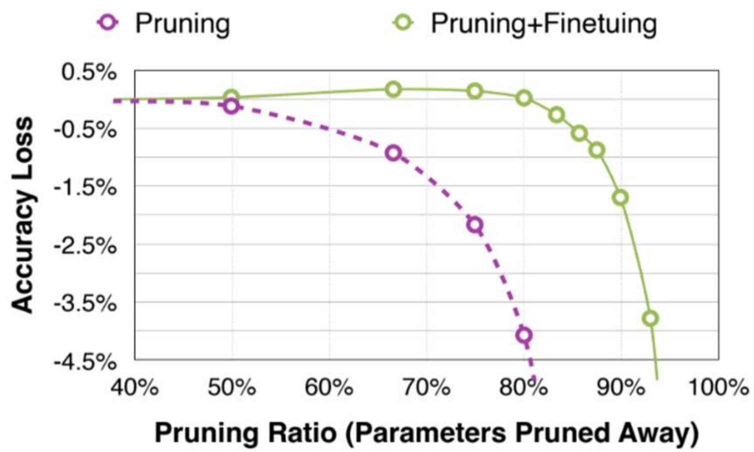


NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications [Yang *et al.*, ECCV 2018]

Fine-tuning Pruned Neural Networks

- How to improve performance of sparse (pruned) models ?
  - Fine-tuning the pruned neural networks will help recover the accuracy and push the pruning ratio higher
  - **Learning rate for fine-tuning** is usually 1/100 or 1/10 of the original learning rate

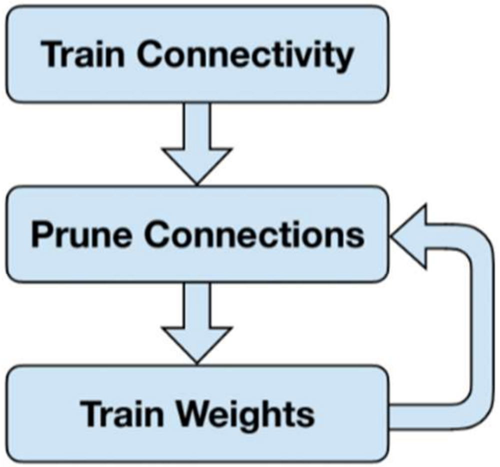


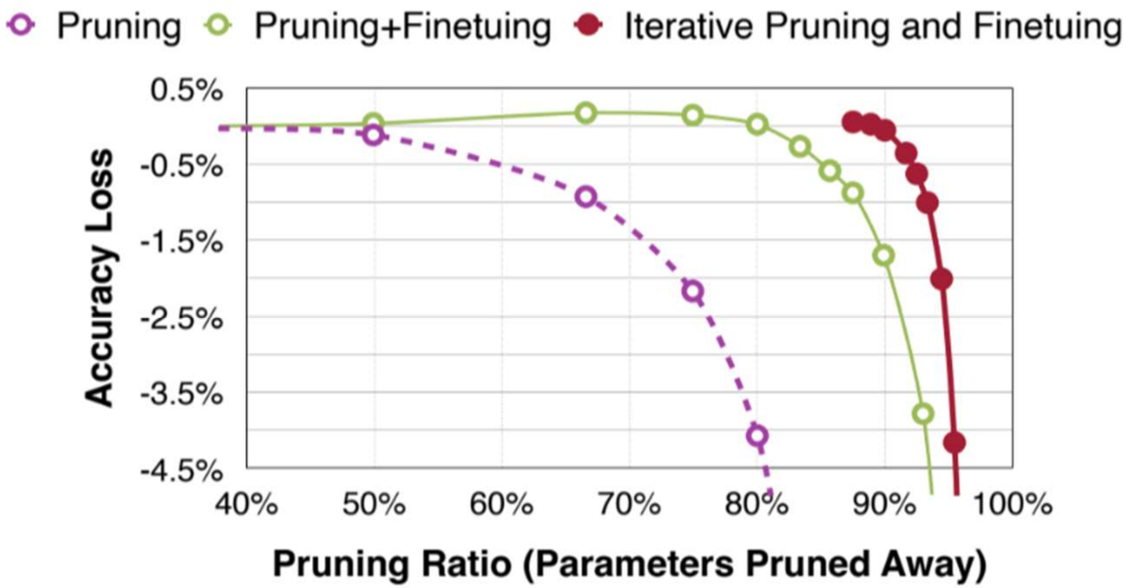


Learning Both Weights and Connections for Efficient Neural Network [Han *et al.*, NeurIPS 2015]

Iterative Pruning

- Iterative pruning gradually increases the target sparsity in each iteration
  - Iterative pruning and fine-tuning resists to the large pruning ratio





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

Regularization

- When training neural networks or fine-tuning quantized neural network, regularization is added
  - Penalized 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

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

Summary of Neural Network Pruning

- **Introduction to pruning**
  - What is the purpose of pruning ?
- **Determine the pruning granularity**
  - Fine-grain, channel-level pruning ….
- **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 to improve performance of pruned models

Takeaway Questions

- How to find prune ratios appropriately ?
  - (A) Randomly guess
  - (B) Sensitivity analysis
  - (C) Refer to the ratio in the batch normalization
- What are potential techniques used by automatic pruning ?
  - Word embedding
  - Iterative training
  - Reinforcement learning