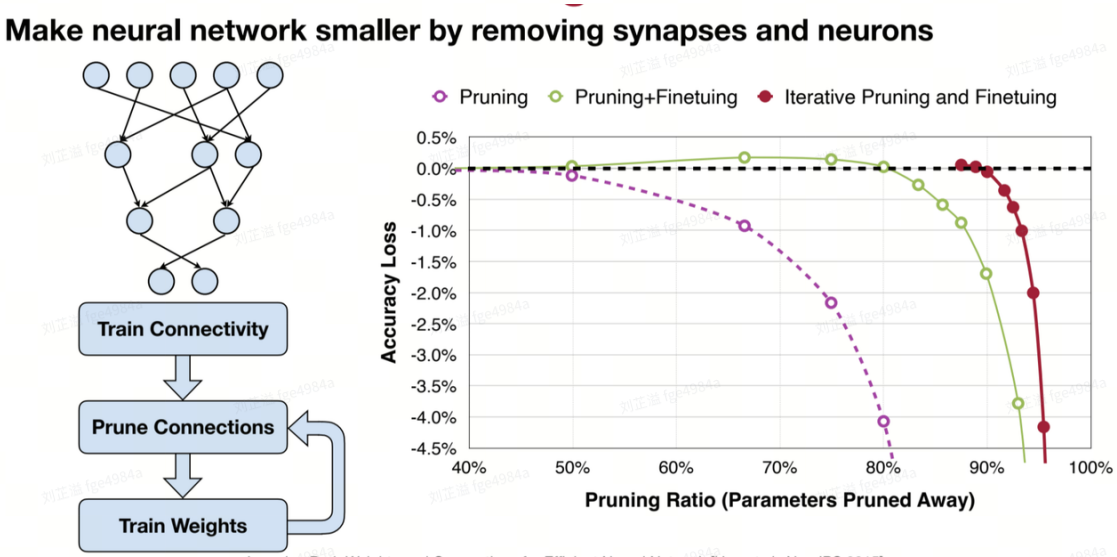
Pruning and Sparsity

- 剪枝和稀疏性
- 减小权重矩阵的大小

Neural Network Pruning

- 迭代剪枝和训练权重
  - 重新训练pruning后保存下来的weights比训练再次初始化的weights更好



- 剪枝问题实际上是将W权重矩阵尽可能变成稀疏矩阵

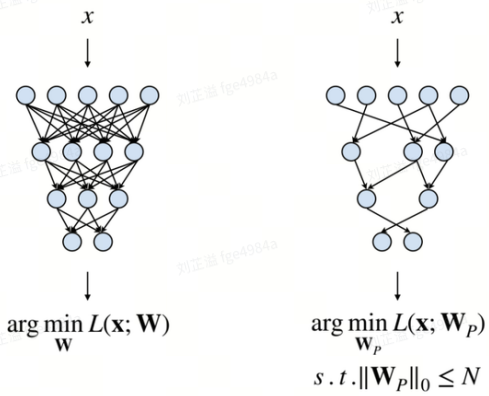
- In general, we could formulate the pruning as follows:

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

subject to

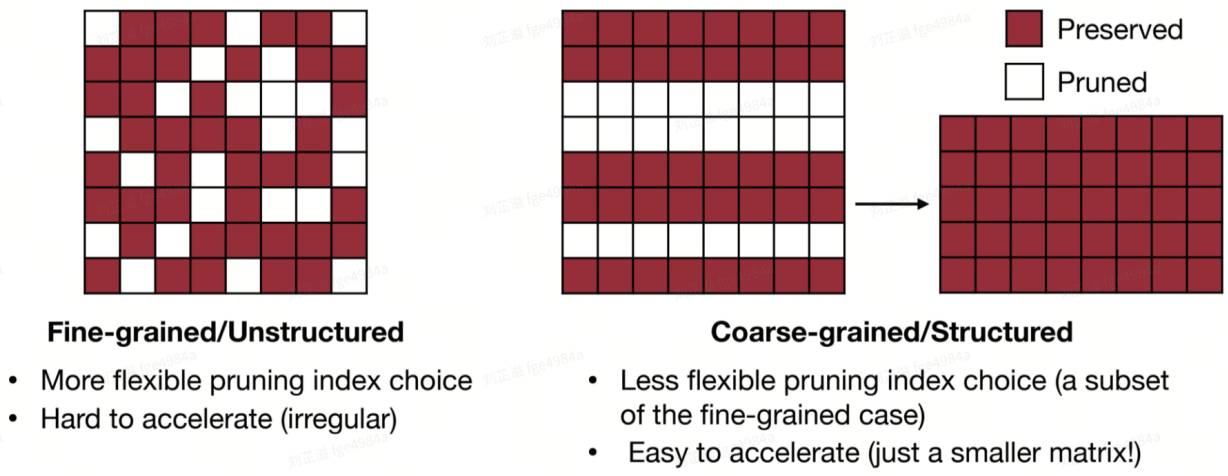
$$\|\mathbf{W}_p\|_0 < N$$

  - 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.

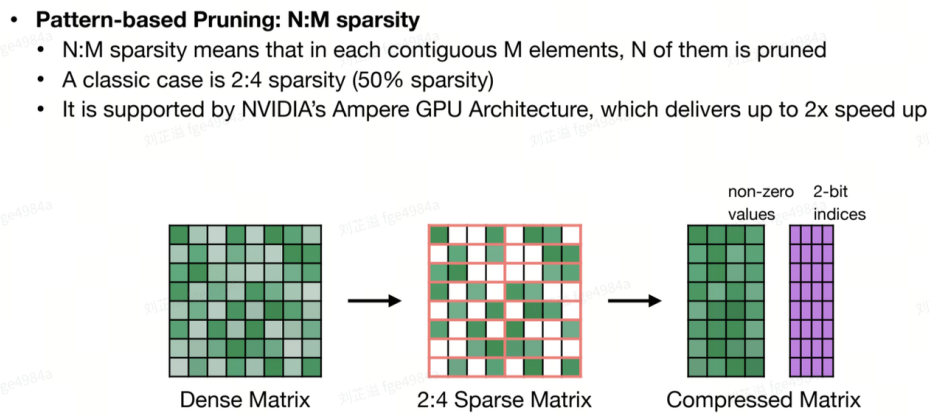


Pruning Granularity

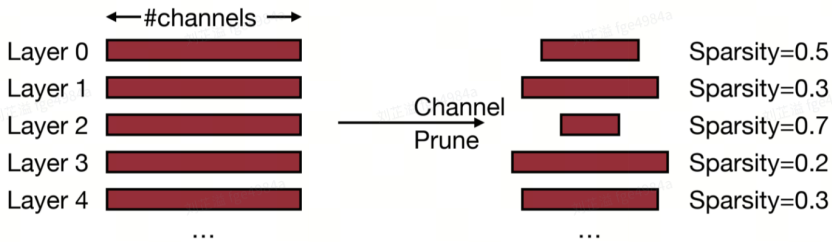
- 粗粒度：不灵活但是容易计算
- 细粒度：灵活但是很难去计算矩阵

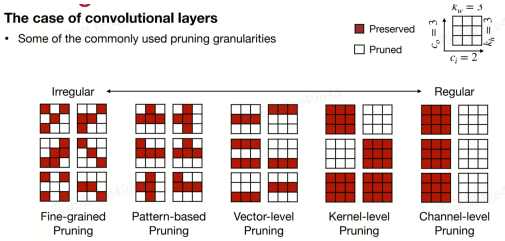


- Fine-grained：只能在定制化的硬件上实现很好地加速；在GPU上则不行
- Pattern-based Pruning: N:M sparsity：每连续M个元素中，就有N个元素被剪枝



- Channel Pruning：直接减少通道数量来加速；较小的压缩比





Pruning Criterion

- 尽可能移除对网络最不重要的参数

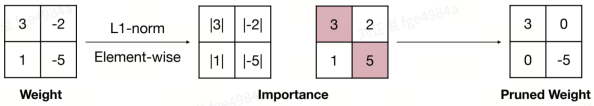
Magnitude-based Pruning

- 基于幅度的剪枝，一种启发式的剪枝方法
- 该方法认为**更大的权重绝对值**对网络越重要；使用L1、L2、Lp范数进行W的比较
  - 可以**逐元素剪枝**
  - 可以按行剪枝

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

$$Importance = |W|$$

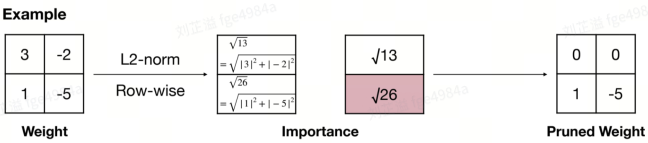
- **Example**



- For row-wise pruning, the L2-norm magnitude can be defined as,

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

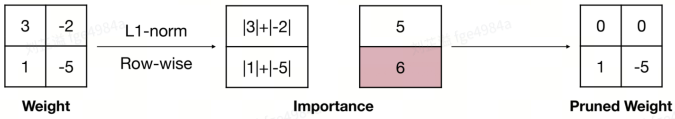
- **Example**



- For row-wise pruning, the L1-norm magnitude can be defined as,

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

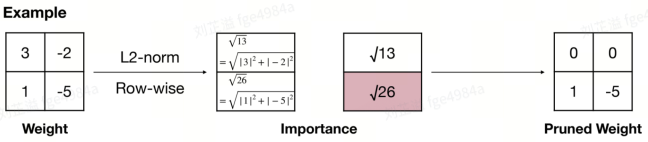
- **Example**



- Magnitude is also known as L_p-norm defined as,

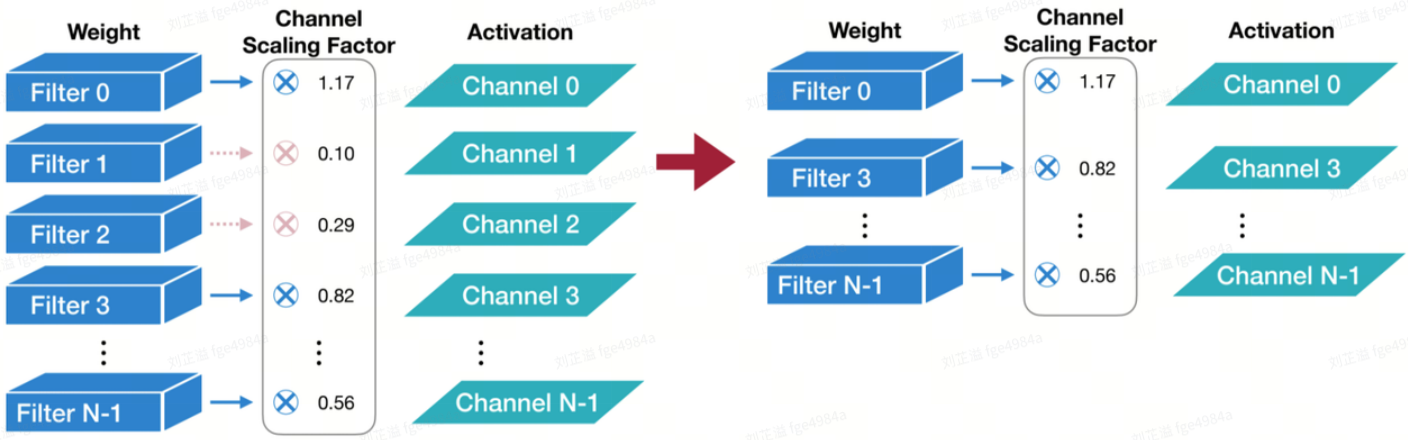
$$\|\mathbf{W}^{(S)}\|_p = \left(\sum_{i \in S} |w_i|^p \right)^{\frac{1}{p}}, \text{ where } \mathbf{W}^{(S)} \text{ is a structural set of parameters}$$

- **Example**



Scaling-based Pruning

- 基于缩放的剪枝；缩放因子与每个卷积核相关；该因子与输出的channel相乘；**缩放因子是可训练的参数**
- **小比例缩放因子的通道被剪枝**
- The filters/output channels with small scaling factor magnitude will be pruned



- 该因子可以在**批归一化层**使用
- The scaling factors can be reused from batch normalization layer

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

Second-Order-based Pruning

- 基于二阶的剪枝
  - 损失函数L是二阶的，最后一项忽略
  - 神经网络训练收敛，第一项可以被忽略
  - 导致错误的每个参数是独立的，忽略交叉项

Minimize the error on loss function introduced by pruning synapses

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

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

where

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

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

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

- 有较小L的神经元被移除

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

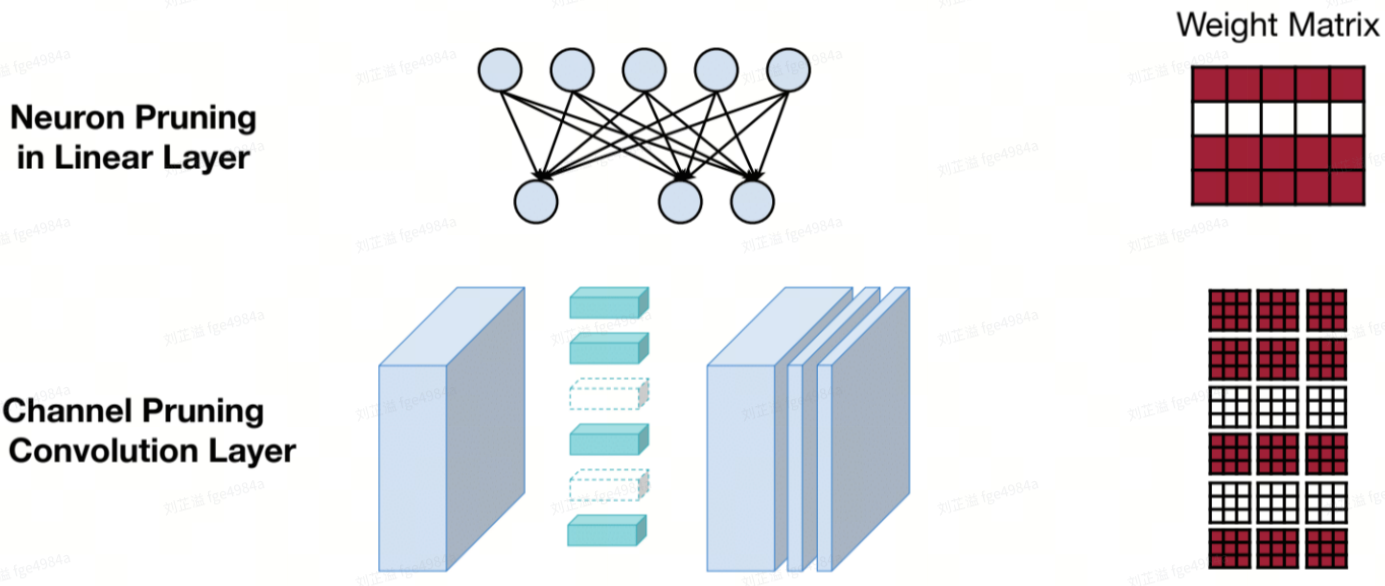
- The synapses with smaller induced error $|\delta L_i|$ will be removed; that is to say,

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

* h_{ii} is no[illegible]negative

Hessian Matrix *H* is difficult to compute.

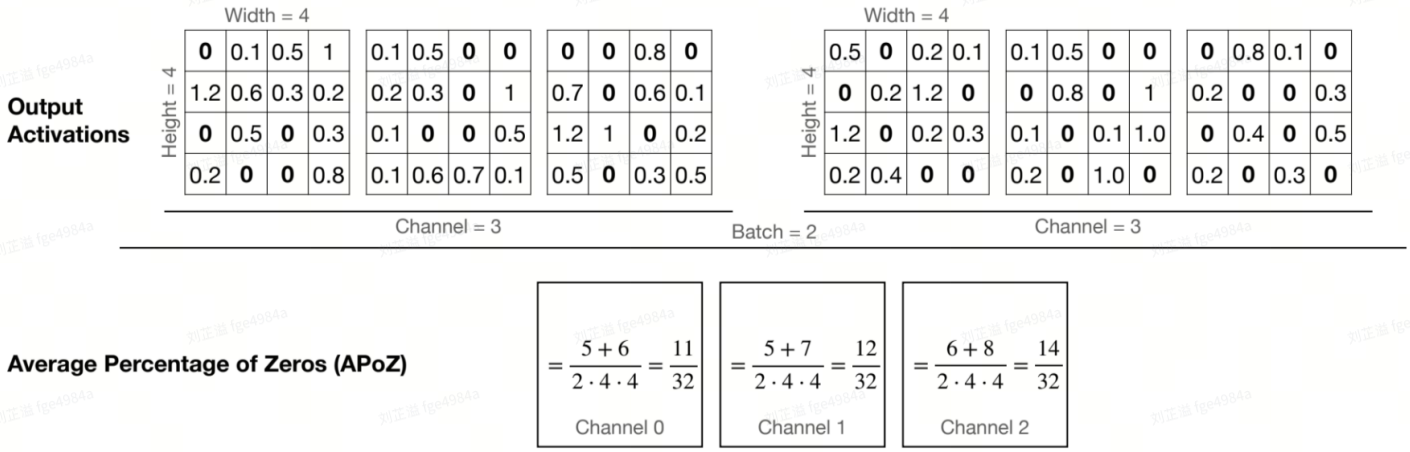
- 神经元剪枝是粗粒度的剪枝



Percentage-of-Zero-Based Pruning

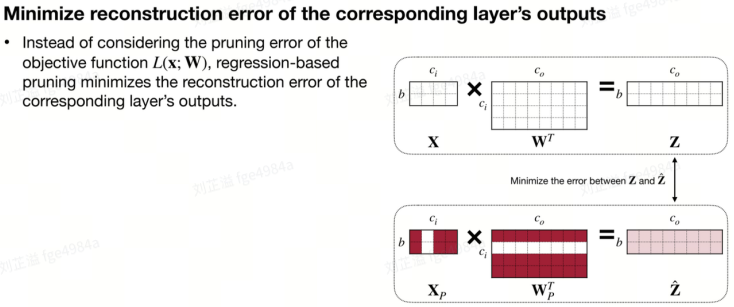
- 从零开始的百分比剪枝；利用Zero激活的百分比来判定神经元的重要性

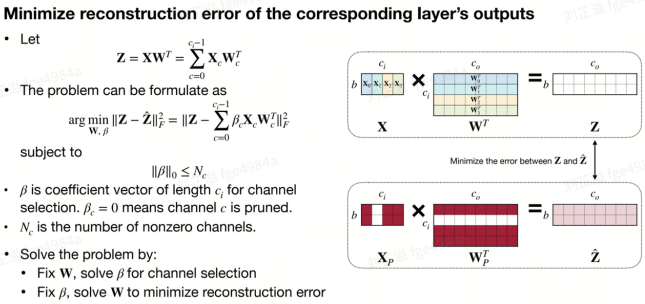
- 越小越重要



Regression-based Pruning

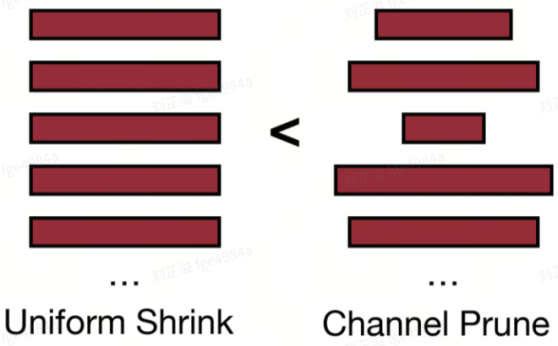
- 最小化相应层的输出；不考虑目标函数的剪枝误差，而是尽可能减少相应层的**输出误差**

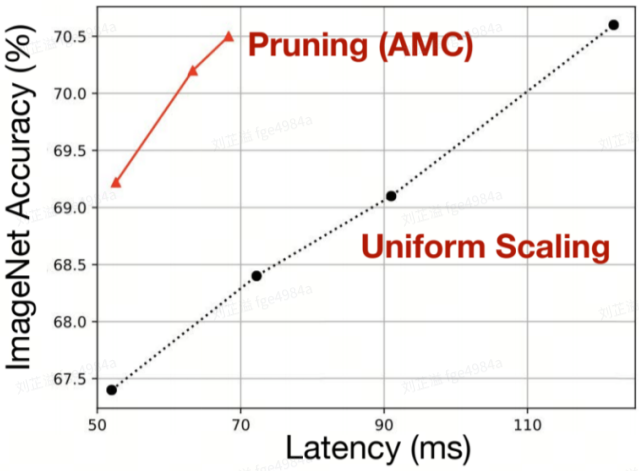




Pruning Ratio

- 非均匀剪枝优于均匀收缩





Analyze the sensitivity of each layer

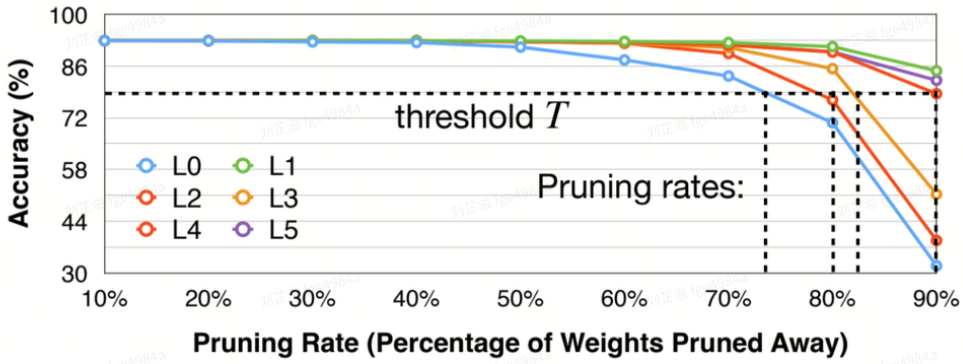
- 对每一层的剪枝率进行敏感性分析
- 对每一层进行不同ratio的剪枝；观察下降趋势

- 有的层对剪枝不敏感，如L1
- 有的层对剪枝敏感，如L0

- 设定阈值T，交点处为不同层对应的剪枝率
- **注意：剪枝率大的层，参数量不一定大，可能对参数的降低没有很大贡献**

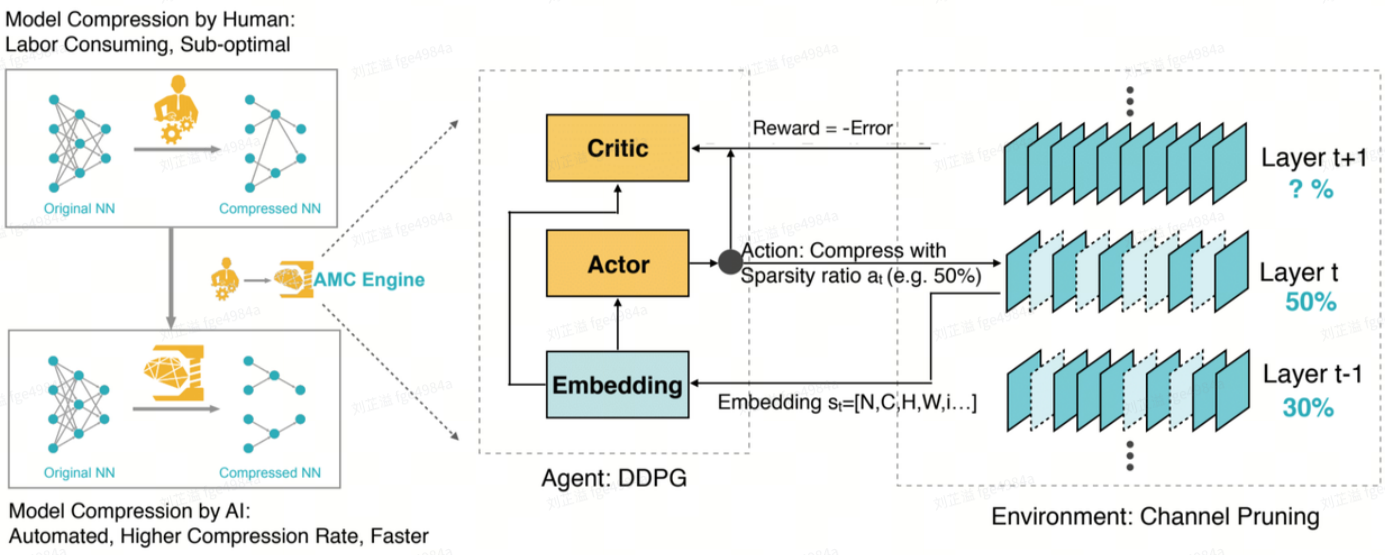
Analyze the sensitivity of each layer

- 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

- 敏感性分析忽略了层与层之间的交互；结果可能是次优的
- AMC：AutoML for Model Compression



- 通过把模型压缩看作一个强化学习的问题

- AMC uses the following setups 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**: DDPG agent, since it supports continuous action output
  - **Reward**: $R = \begin{cases} -\text{Error}, & \text{if satisfies constrains} \\ -\infty, & \text{if not} \end{cases}$
    - We can also optimize latency constraints with a pre-built lookup table (LUT)

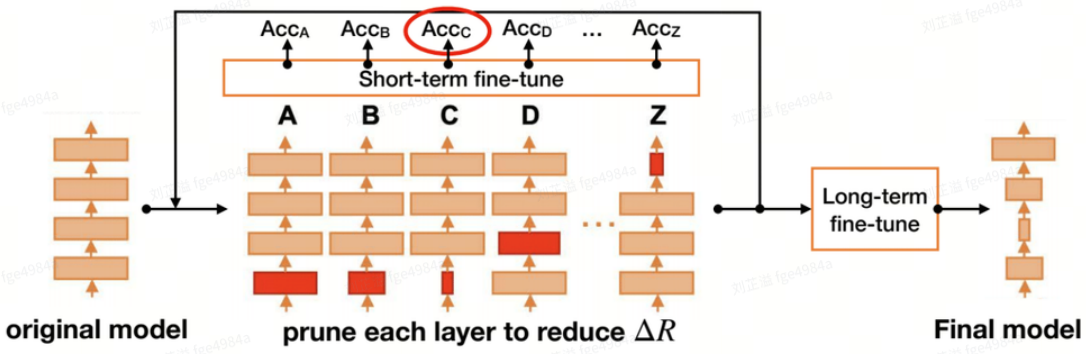
- **卷积层channel减小，flops增加不是线性的**

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

NetAdapt: A rule-based iterative/progressive method

- 找到一个每层剪枝率来实现全局资源限制；是一个迭代的过程
- 短期迭代后更新模型；长期迭代后进行权重训练

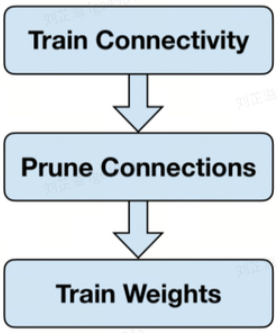
- For each iteration, we aim to reduce the latency by a certain amount ΔR (manually defined)
  - For each layer L_k (k in A-Z in the figure)
    - Prune the layer s.t. the latency reduction meets ΔR (based on a pre-built lookup table)
    - Short-term fine-tune model (10k iterations); measure accuracy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

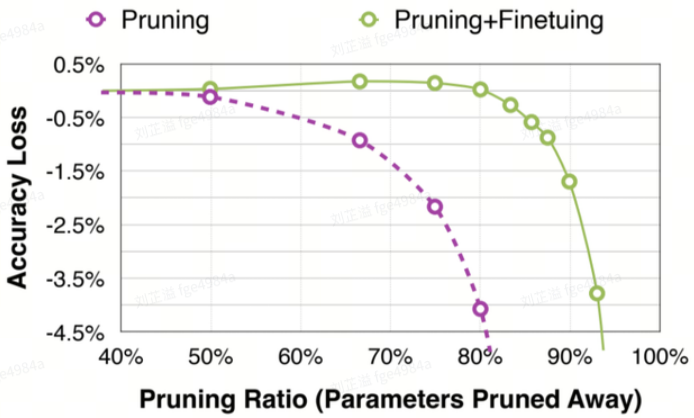


Fine-tuning/Training

- 剪枝后微调可以提升识别率；lr设置为original lr的1/100或1/10

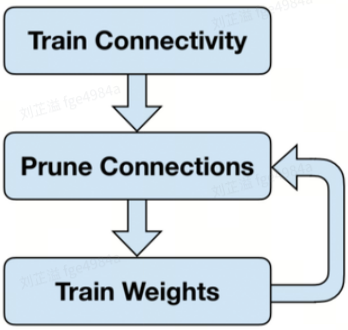
- After pruning, the model may decrease, especially for larger pruning ratio.
- Fine-tuning the pruned neural networks will help recover the accuracy and push the pruning ratio higher.
  - **Learning rate for fine-tuning is usually 1/100 or 1/10 of the original learning rate.**

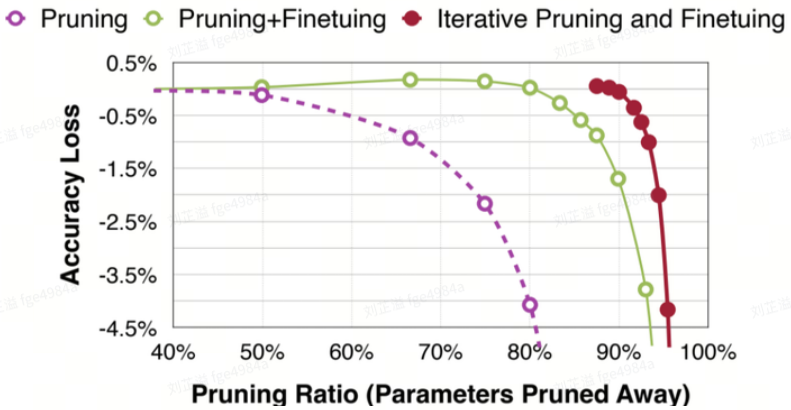




- 使用迭代剪枝；**一次剪枝和微调看作一次迭代**

  - 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.





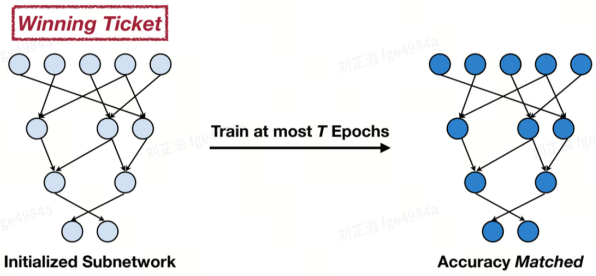
Regularization

- **Fine-grained pruning applies L2 regularization**
- **Network slimming applies L1 regularization on channel scaling factors**

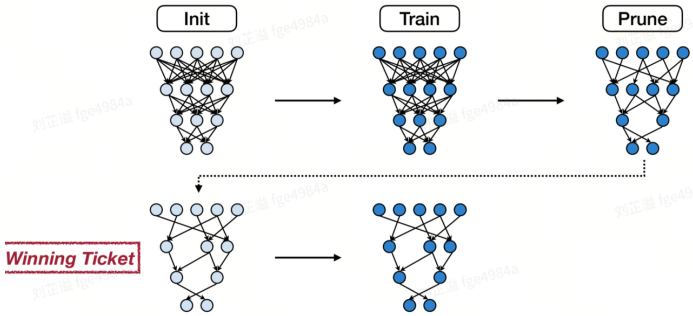
The Lottery Ticket Hypothesis

- 一个随机初始化的网络的**子网络**通过若干次迭代可以达到**原来网络的精度**

*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



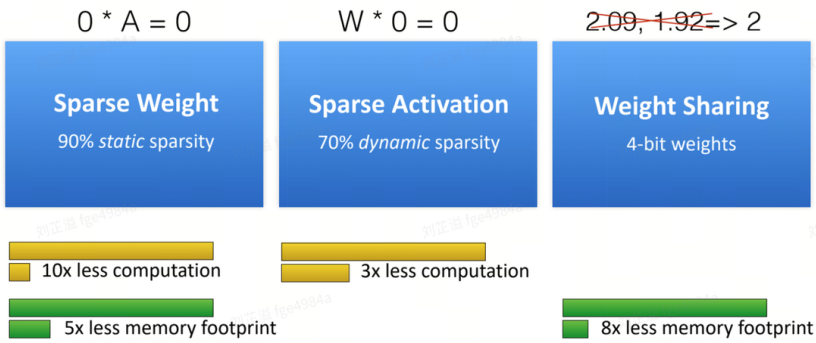
Iterative Magnitude Pruning



System Support for Sparsity

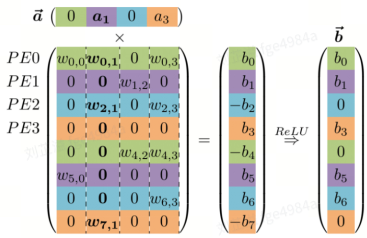
- 稀疏权重矩阵
- 稀疏激活函数
- 权重共享

The First DNN Accelerator for Sparse, Compressed Model



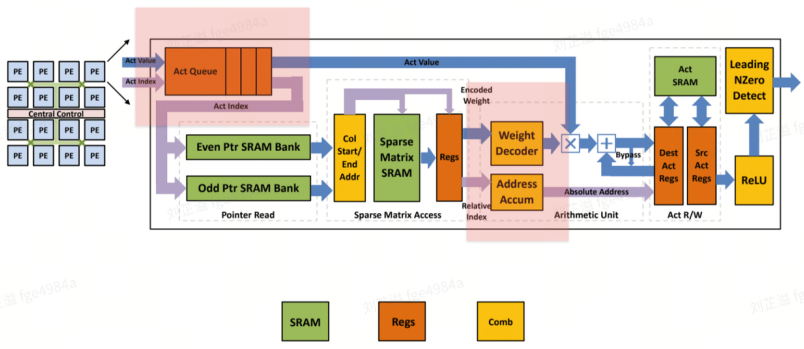
EIE: Parallelization on Sparsity

logically



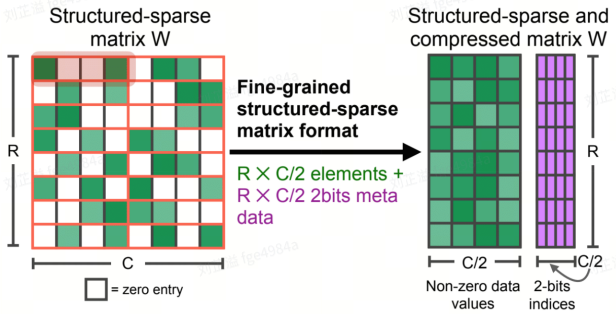
physically

Virtual Weight	$W_{0,0}$	$W_{0,1}$	$W_{4,2}$	$W_{0,3}$	$W_{4,3}$
Relative Index	0	1	2	0	0
Column Pointer	0	1	2	3	5

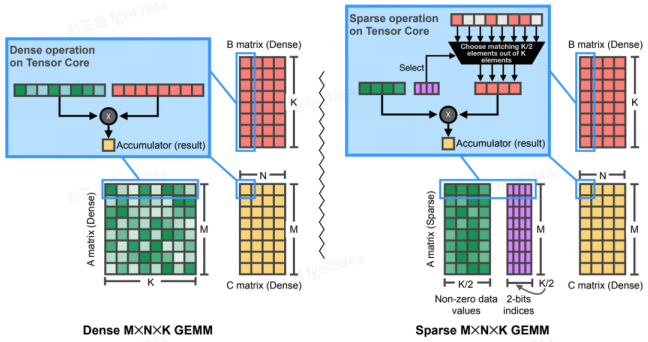


NVIDIA Tensor Core: M:N Weight Sparsity

- 4位数值，2位索引

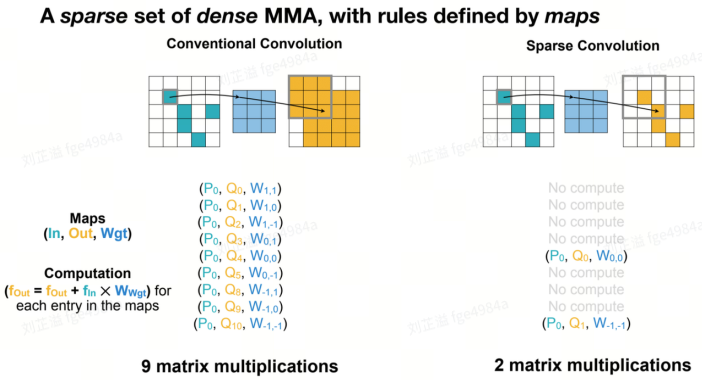


Two weights are nonzero out of four consecutive weights (2:4 sparsity).



The indices are used to mask out the inputs. Only 2 multiplications will be done out of four.

Sparse convolution on sparse inputs



- 为了解决计算0的冗余过多以及计算不规律问题；采用分组