Microsoft

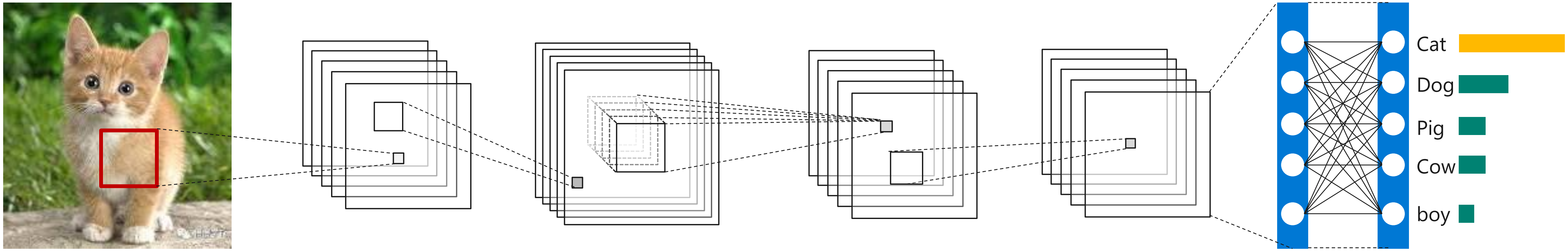
人工智能系统 System for AI

神经网络的稀疏化
Efficiency via compression and sparsity

主要内容

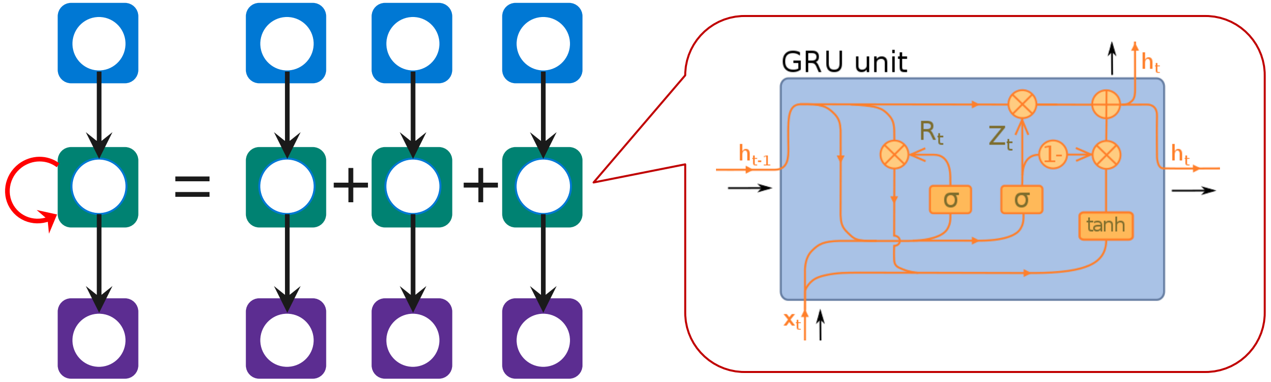
- 背景介绍
  - 模型的增长
  - 计算力的瓶颈
  - 模型压缩与稀疏的重要性
- 神经网络的稀疏性
  - 权重稀疏
  - 激活稀疏
  - 梯度稀疏
  - 稀疏与正则化
  - 量化

深度神经网络：多层的互联网络结构

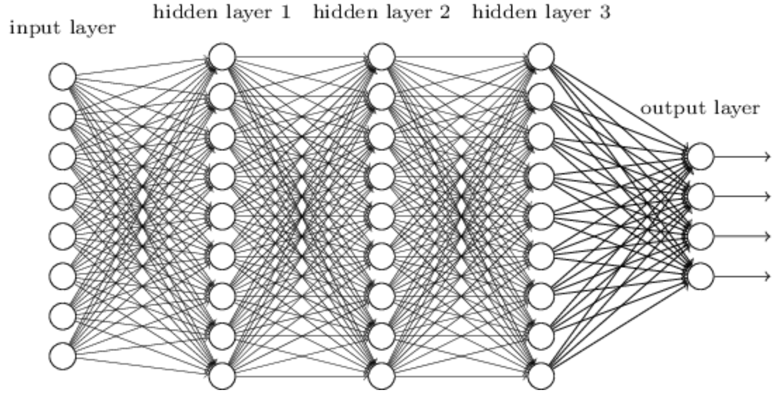


卷积神经网络

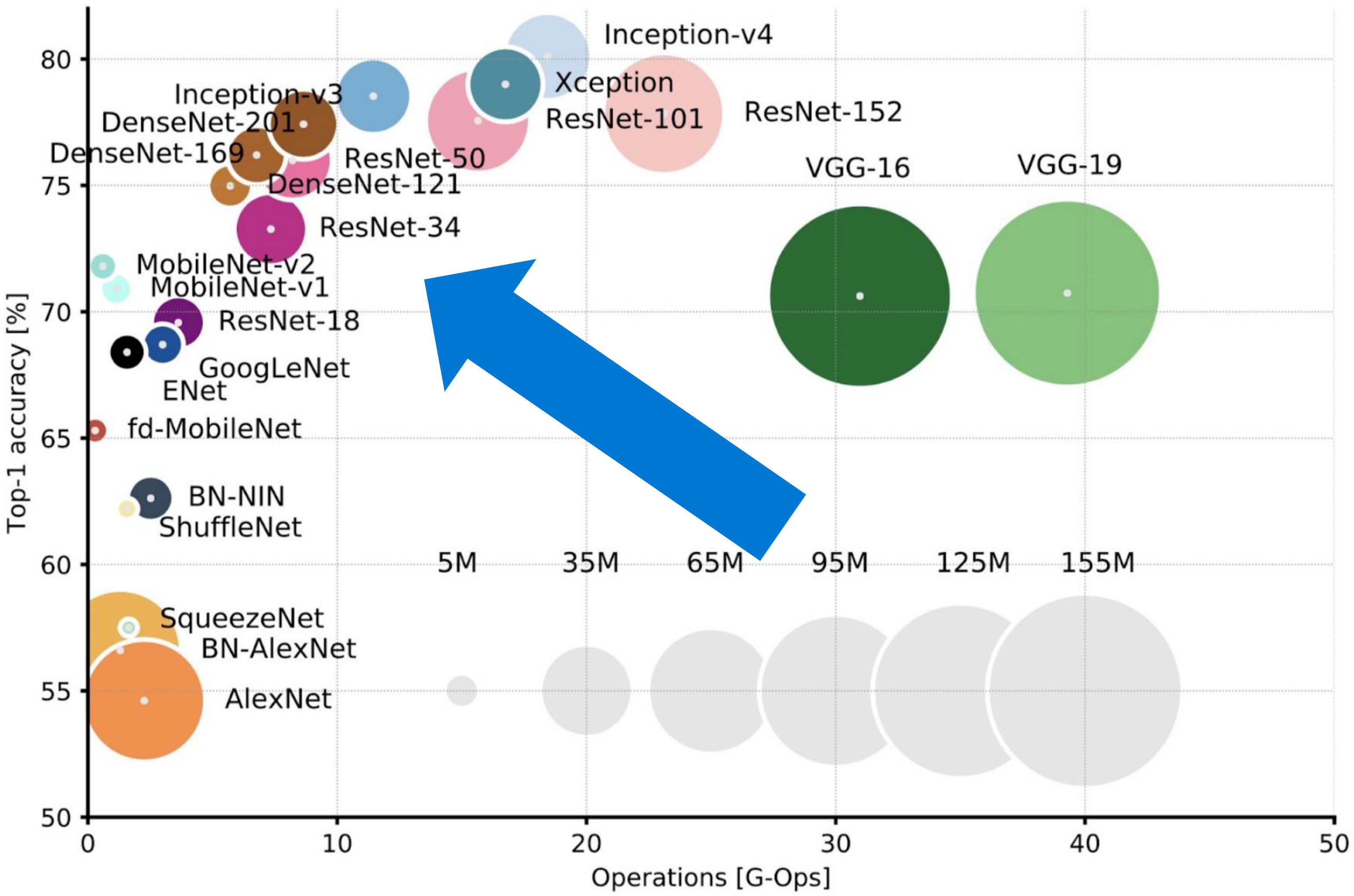
循环神经网络



全连接神经网络

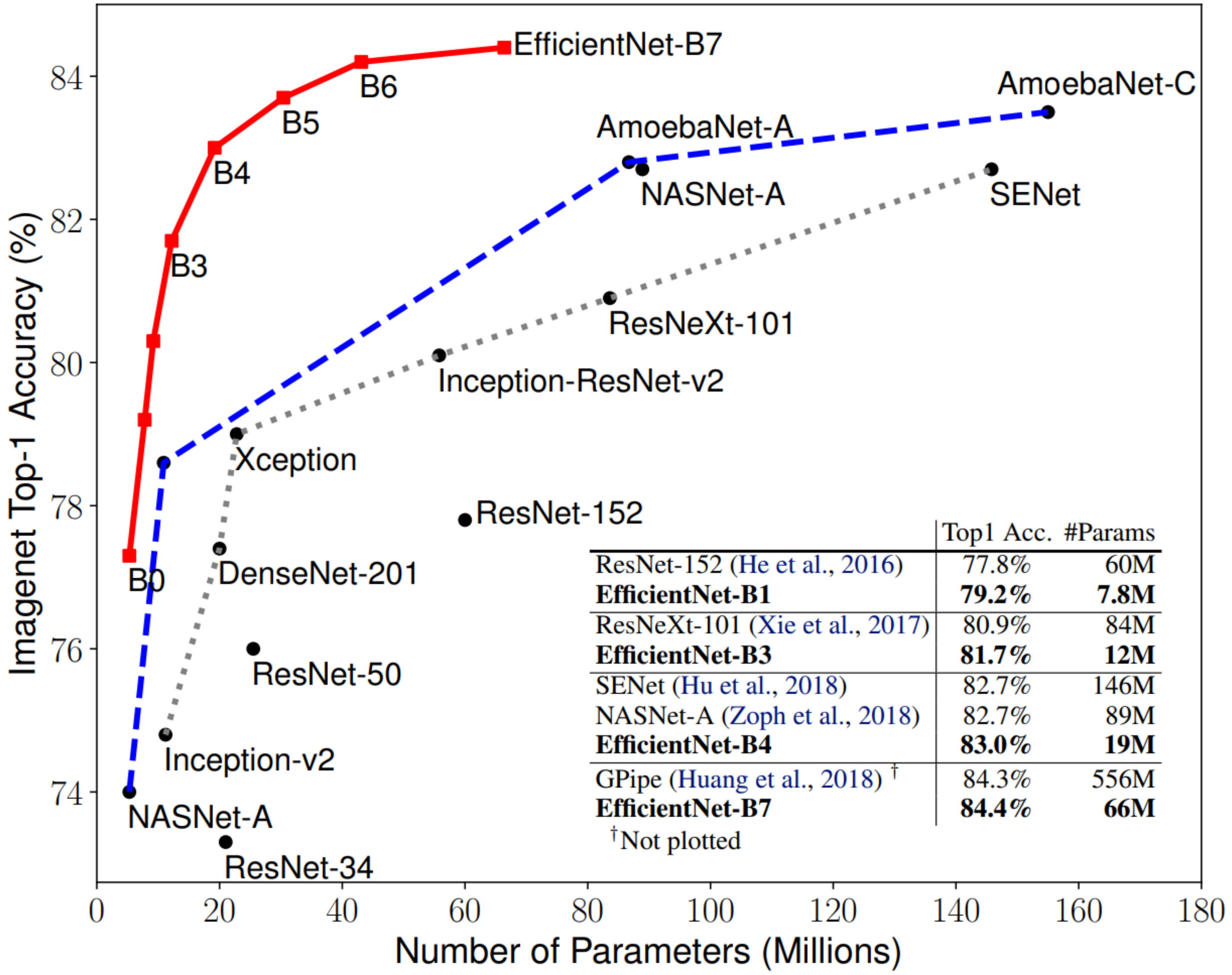


结构优化：寻找更高效的模型



增加模型维度是提高精度的主要手段

深度、宽度、分辨率



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

[1] EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

模型越来越大

BERT
Language

- 64 TPUv2
- 4 Days
- 1000 GB

or

- 8 P100
- 365 Days
- 1000 GB

Wavenet
Speech

- 2 P100
- 6 Days
- 16 GB

Deformable CNN
Vision

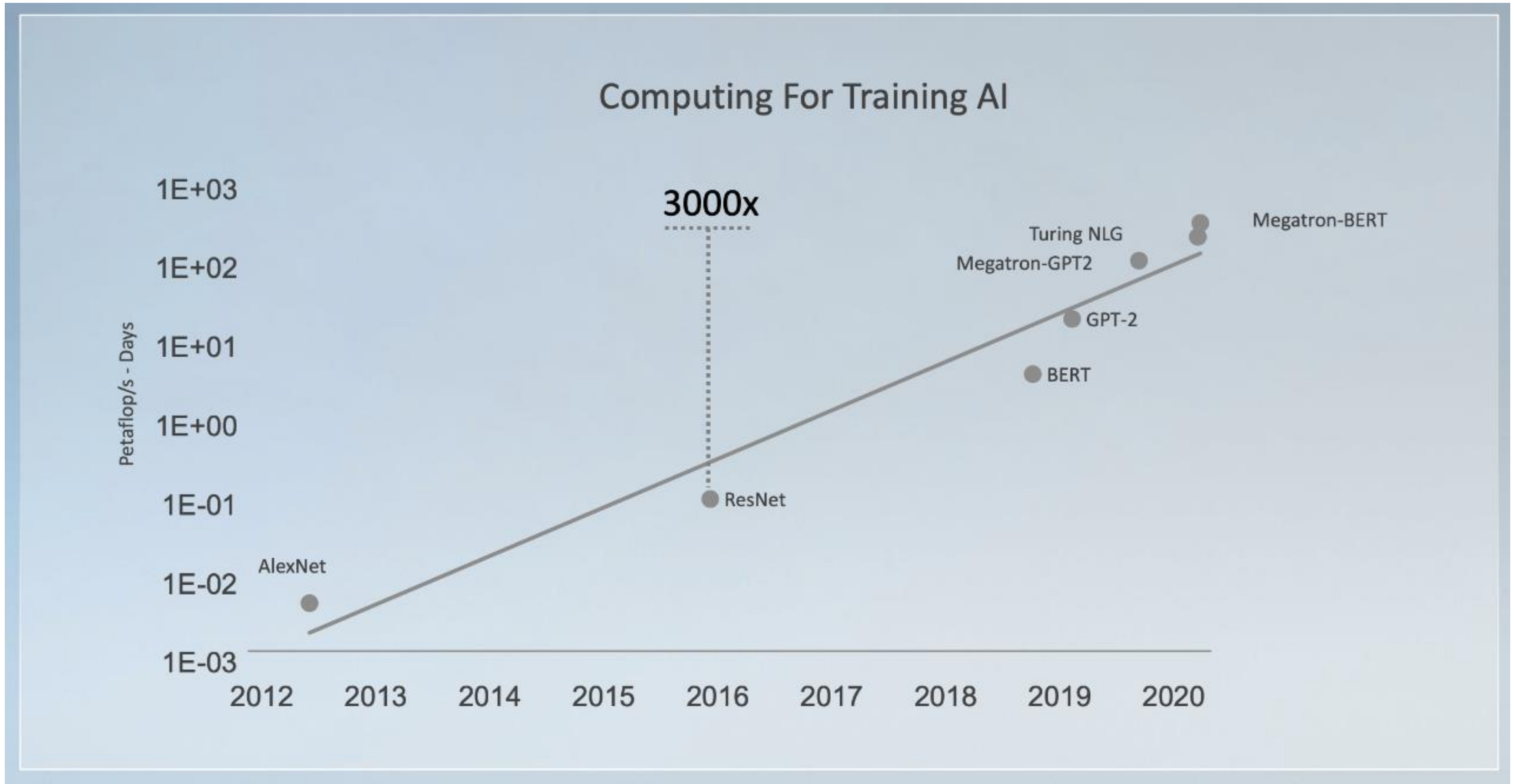
- 8 P100
- 10 Days
- 64 GB

MoE
Language

- 64 K80
- 6 Days
- 1500 GB

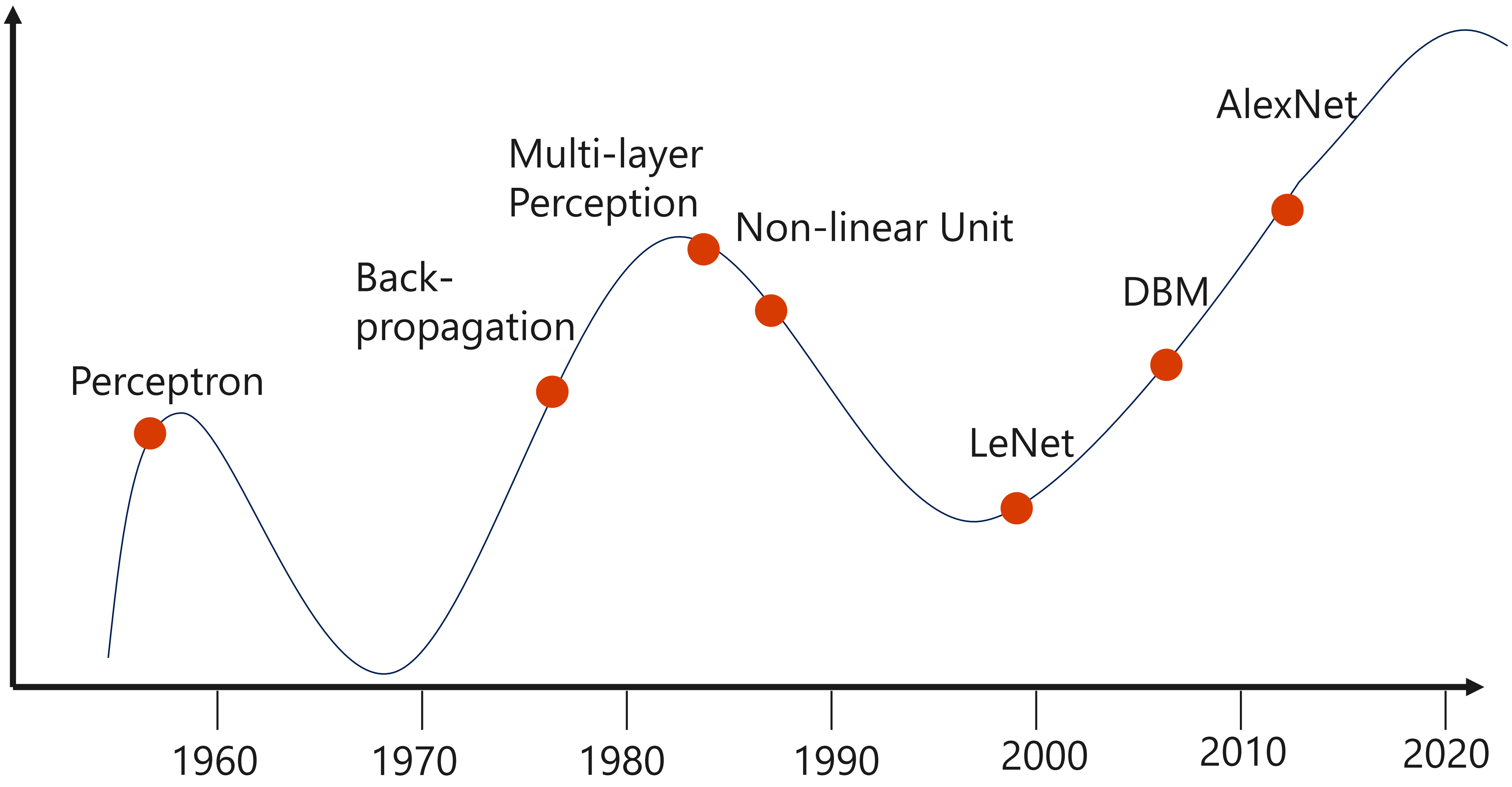
Volta GPU发布以来，大模型训练的算力要求增长了3000倍

- GPT-3：一个亿人民币的模型

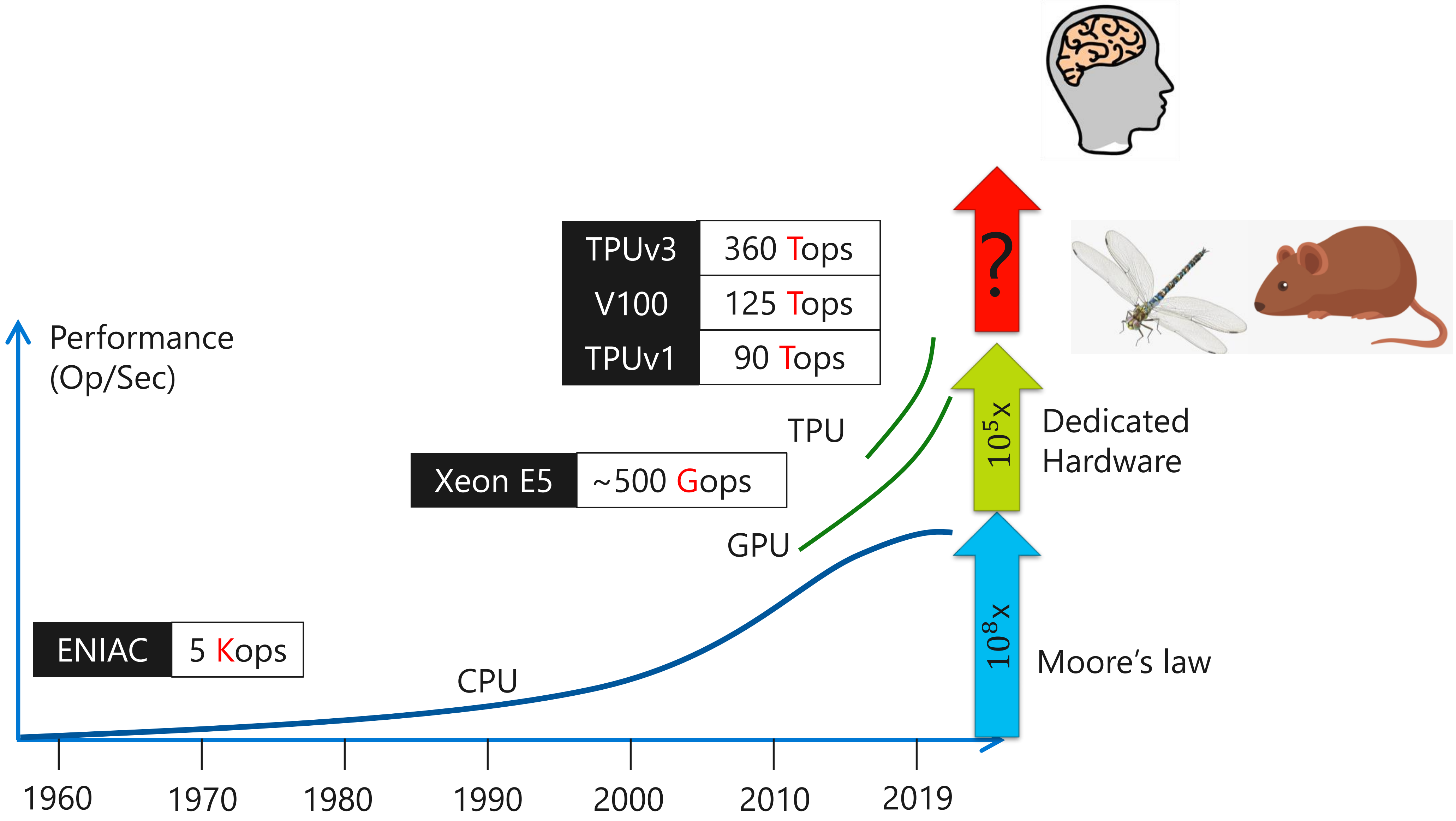


3000X Higher Compute Required to Train Largest Models Since Volta

计算力是AI在二十一世纪初叶复兴的核心动力

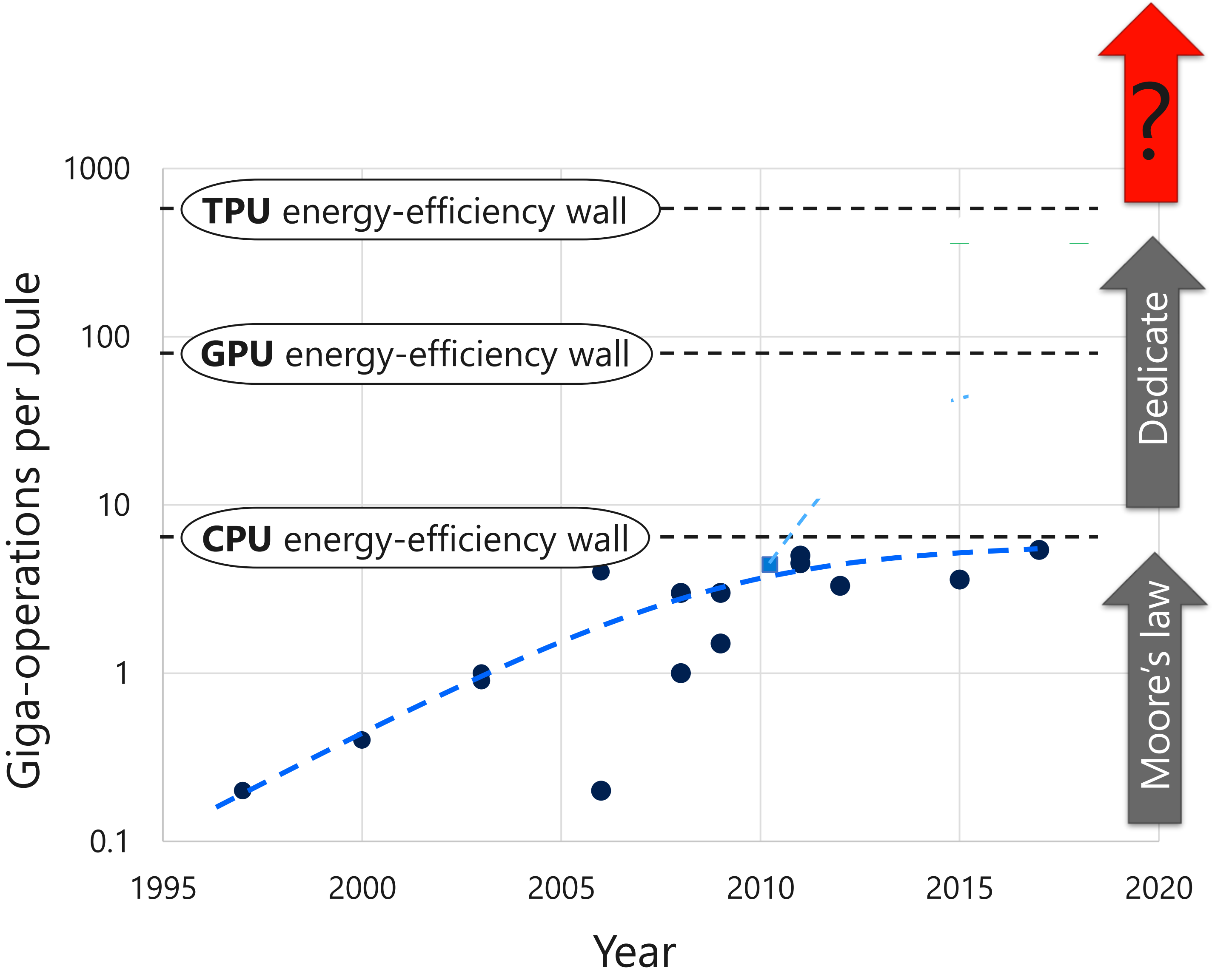


计算力瓶颈



Performance
(Op/Sec)
TPUv3
360 Tops
V100
125 Tops
TPUv1
90 Tops
Xeon E5
~500 Gops
TPU
GPU
ENIAC
5 Kops
CPU
?
10^5x
10^8x
Dedicated
Hardware
Moore's law
1960
1970
1980
1990
2000
2010
2019

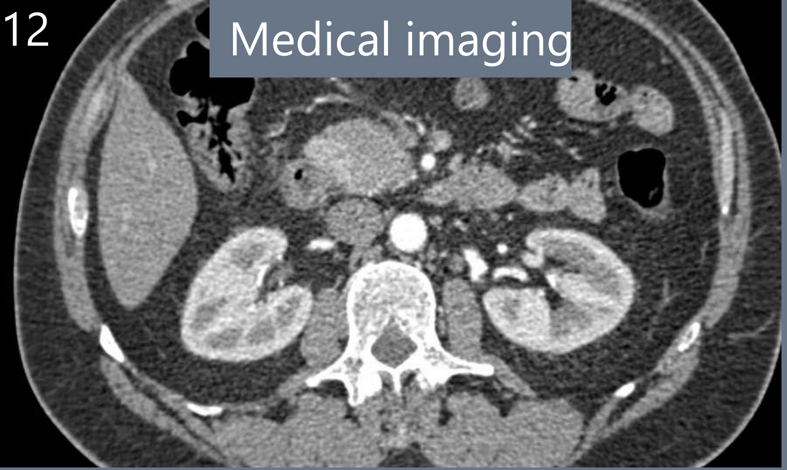
成本瓶颈

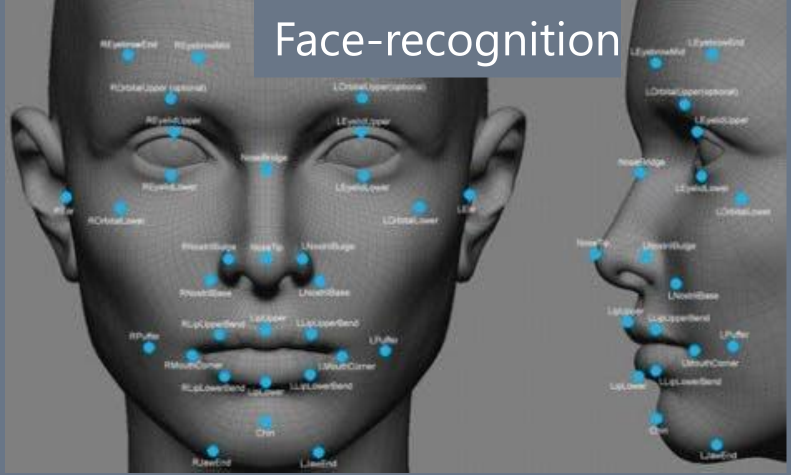


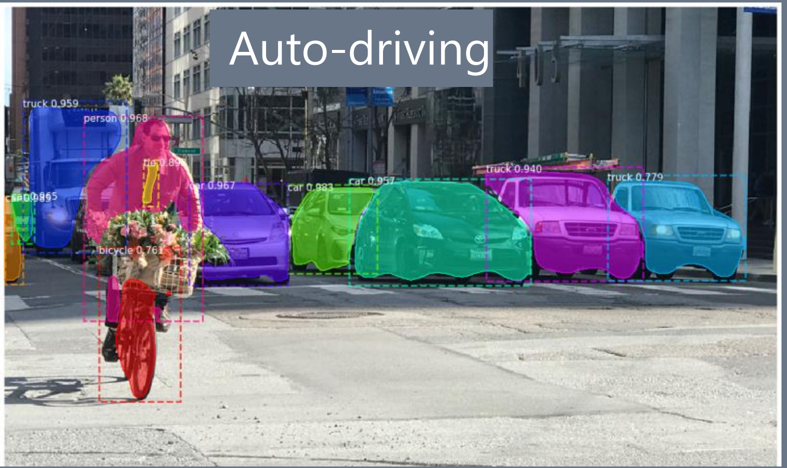
$$\frac{10^{17\sim20}}{20\ Watt}$$

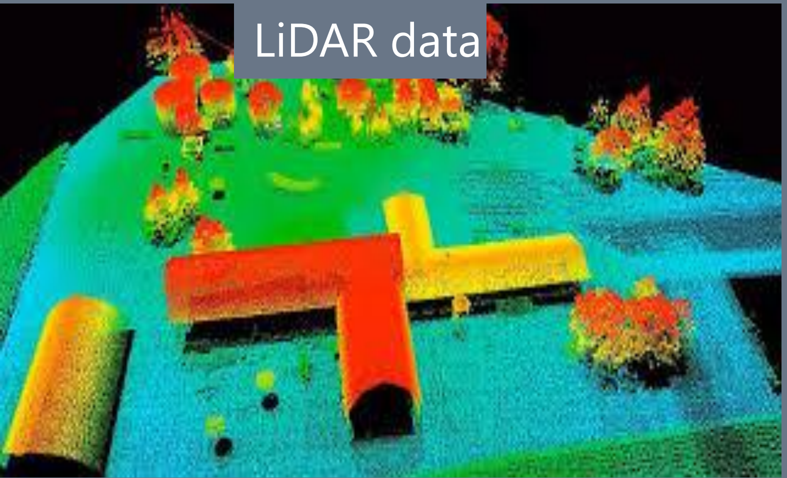
模型压缩与稀疏化
（更高效的算法，软硬件协同）

更大更强的处理器
（锗基材料，量子计算）



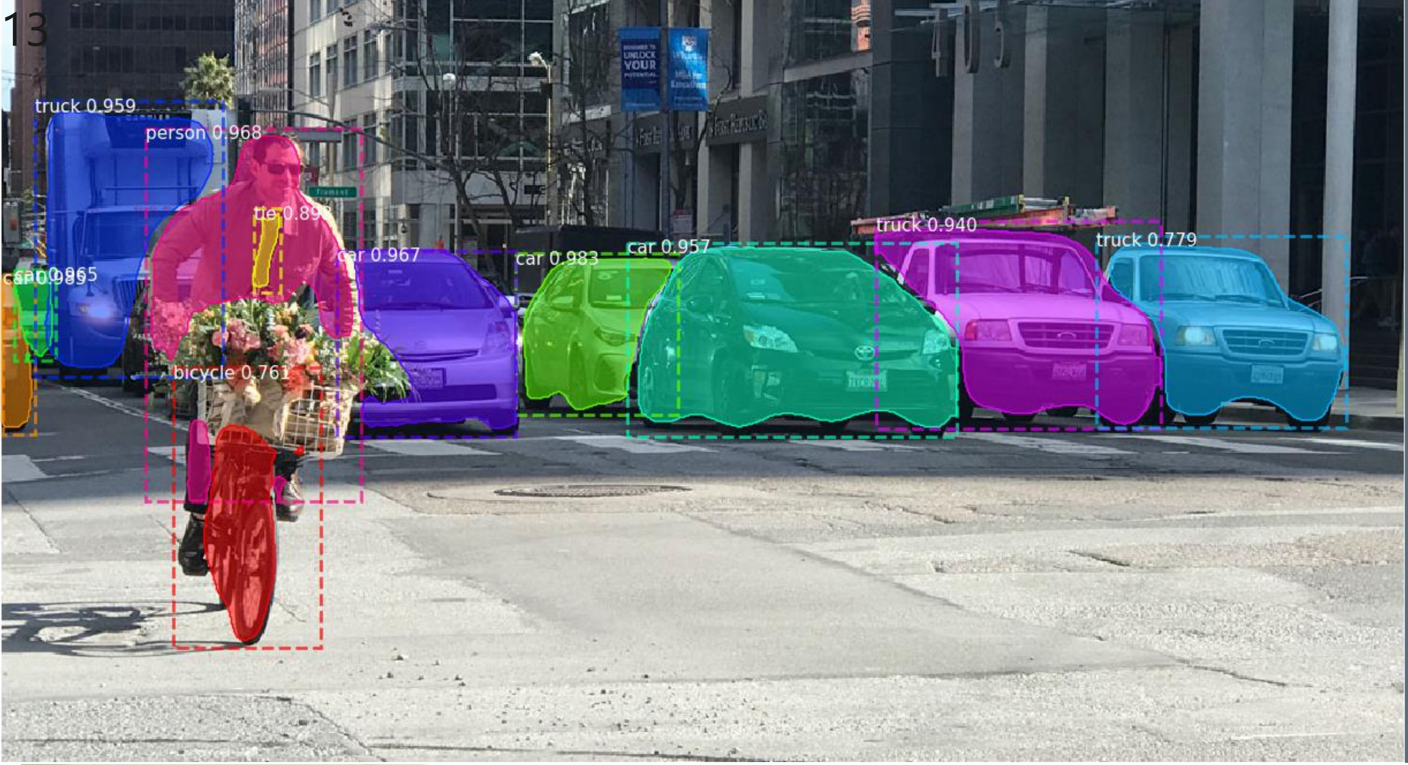






稀疏的来源：高度结构化的数据

- 我们被无数的数据包围
- 每一种数据都有其内在结构性
- (*自动地*) 学习数据内生的结构性是人工智能算法的核心
- 数据的结构性带来信息表达中的稀疏性
- 高效的人工智能系统应该充分的利用稀疏性。



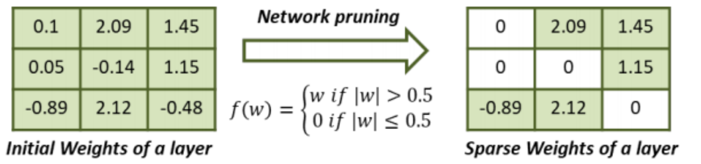
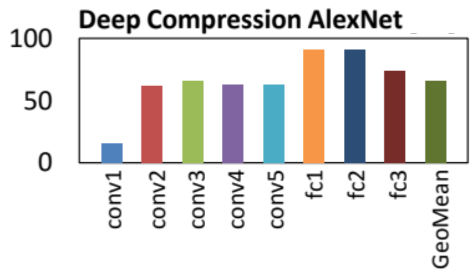
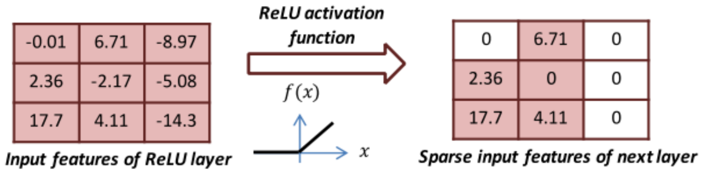
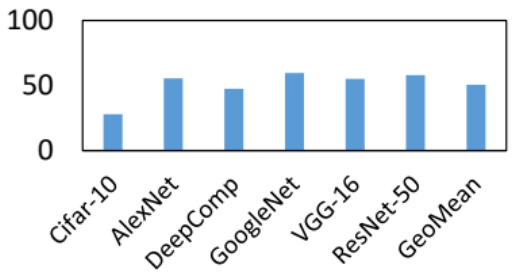
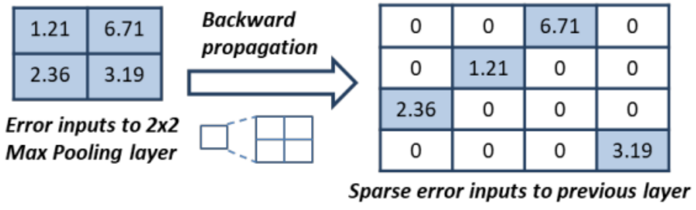
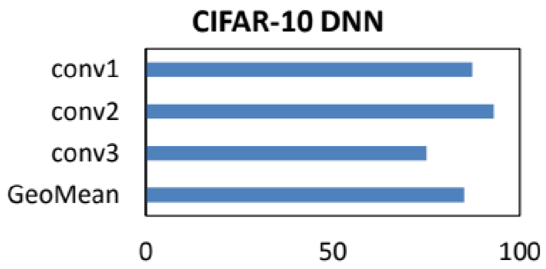
当人类看到眼前物体时，我们的神经系统：

- 不会处理所有像素；
- 关注主要物体；

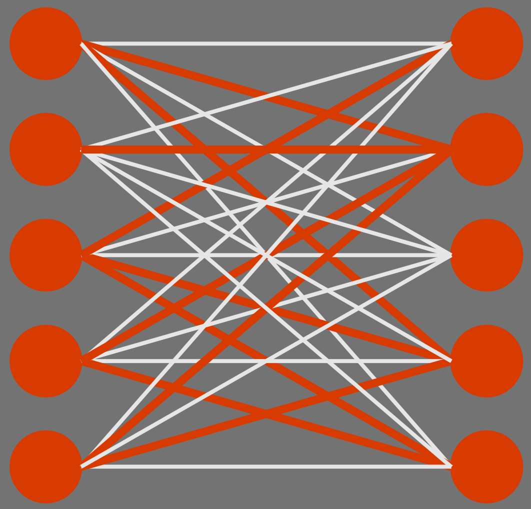
当人类识别一只猫时，我们：

- 不会仔细检查每一个毛发的纹理；
- 简单的几何边缘足以识别；

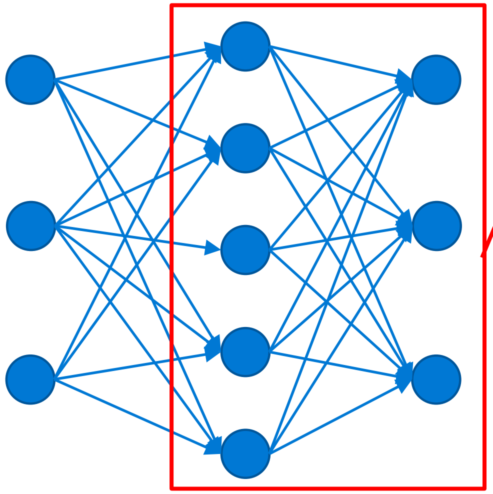
神经网络的稀疏性

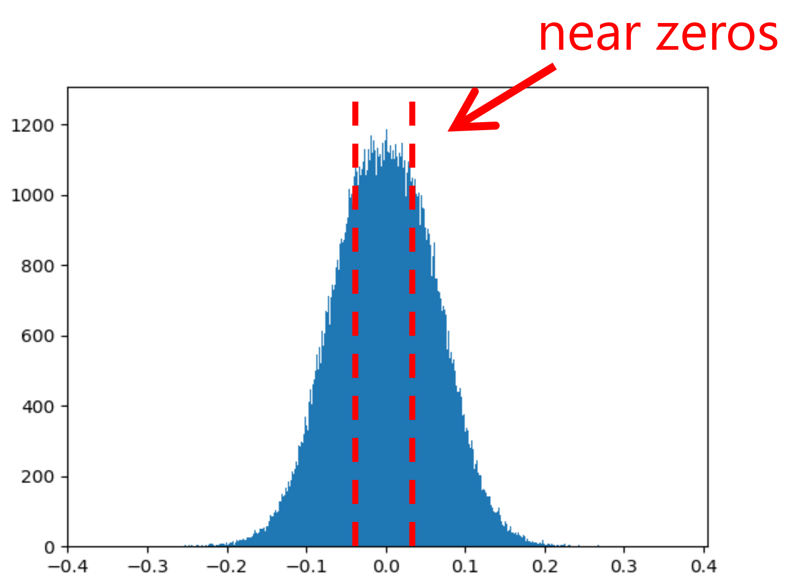
<table>
<tr><th>Sparsity Types</th><th>Data Structure</th><th>Sources</th><th>Percentage of Sparsity</th><th>Speedup Potential</th></tr>
<tr><td>Static Sparsity</td><td>Weight</td><td>
</td><td>
</td><td>1x~50x</td></tr>
<tr><td rowspan="2">Dynamic Sparsity</td><td>Activation</td><td>
</td><td>
</td><td>1x~10x</td></tr>
<tr><td>Gradient</td><td>
</td><td>
</td><td>1x~10x</td></tr>
</table>

权重稀疏



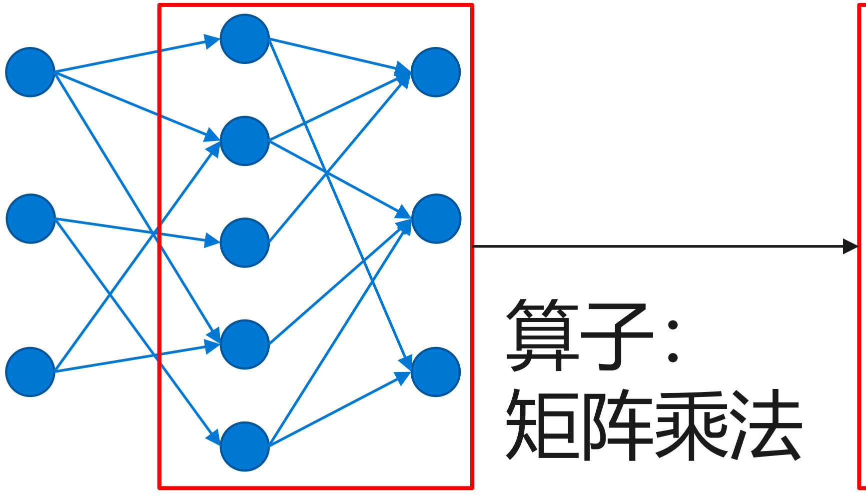
神经网络的权重（weight）稀疏



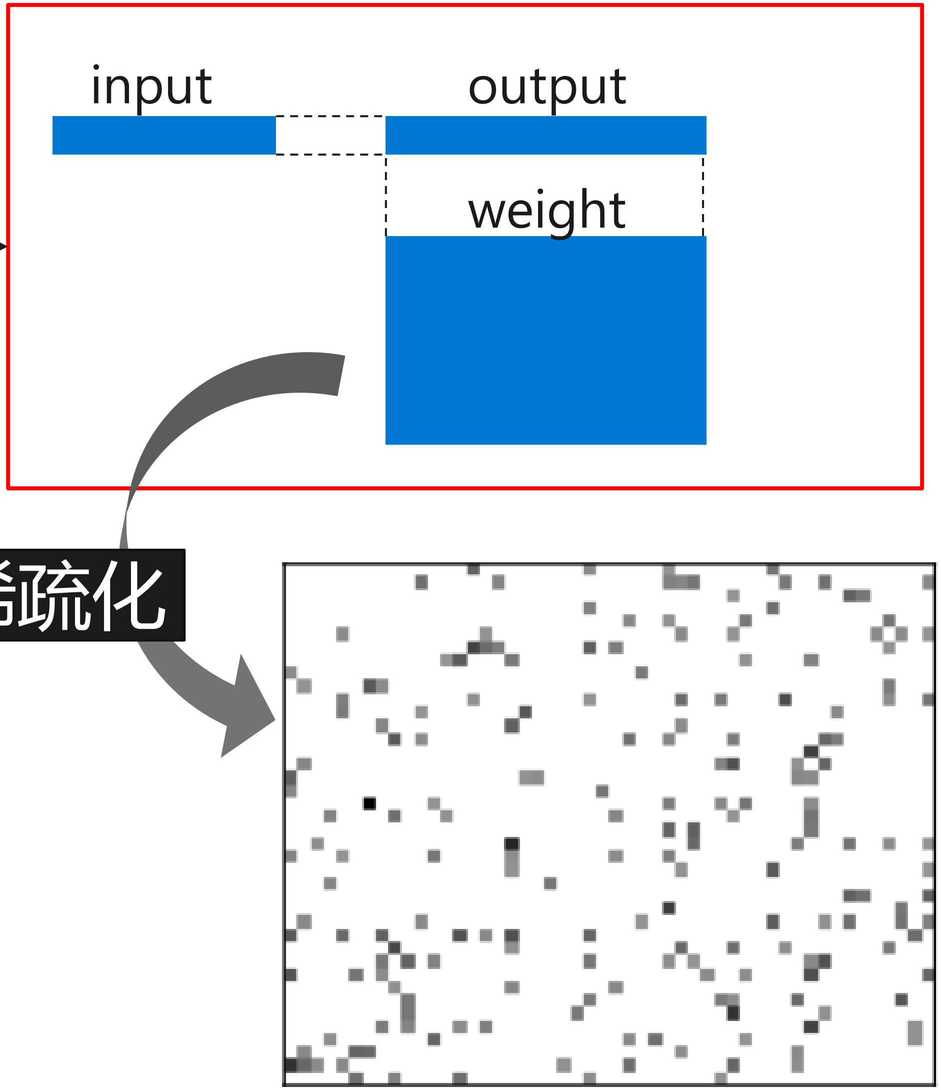


Data set: PTB, a language model
1 Million training words,
73k validation words,
82k test words
Model: 2-layer LSTM model,
LSTM size = 1500

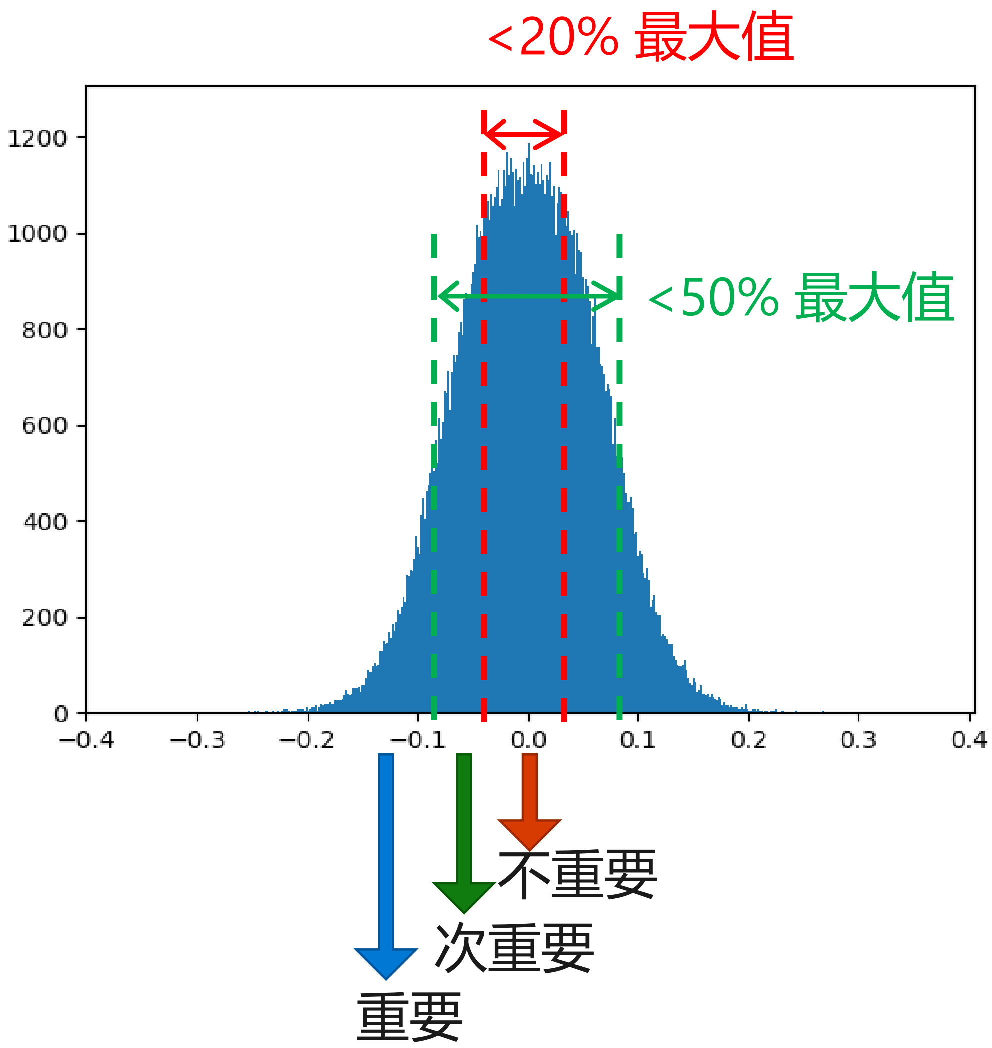
稀疏化的矩阵乘法表示



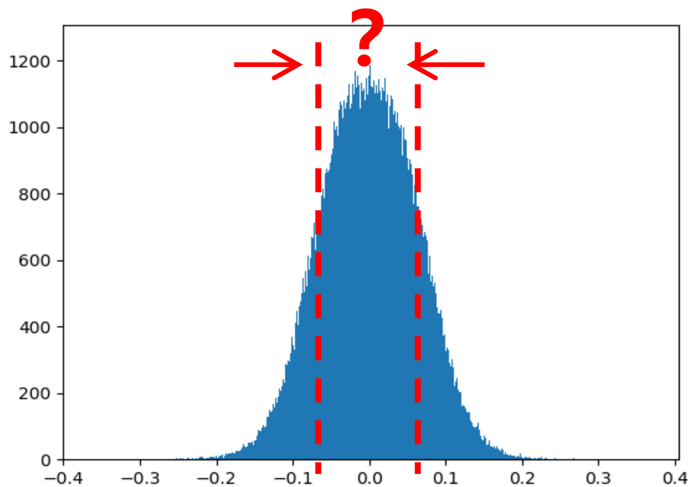
算子：
矩阵乘法

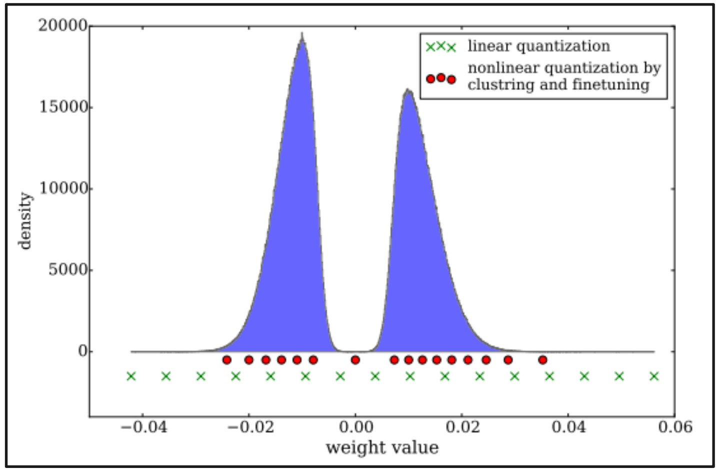


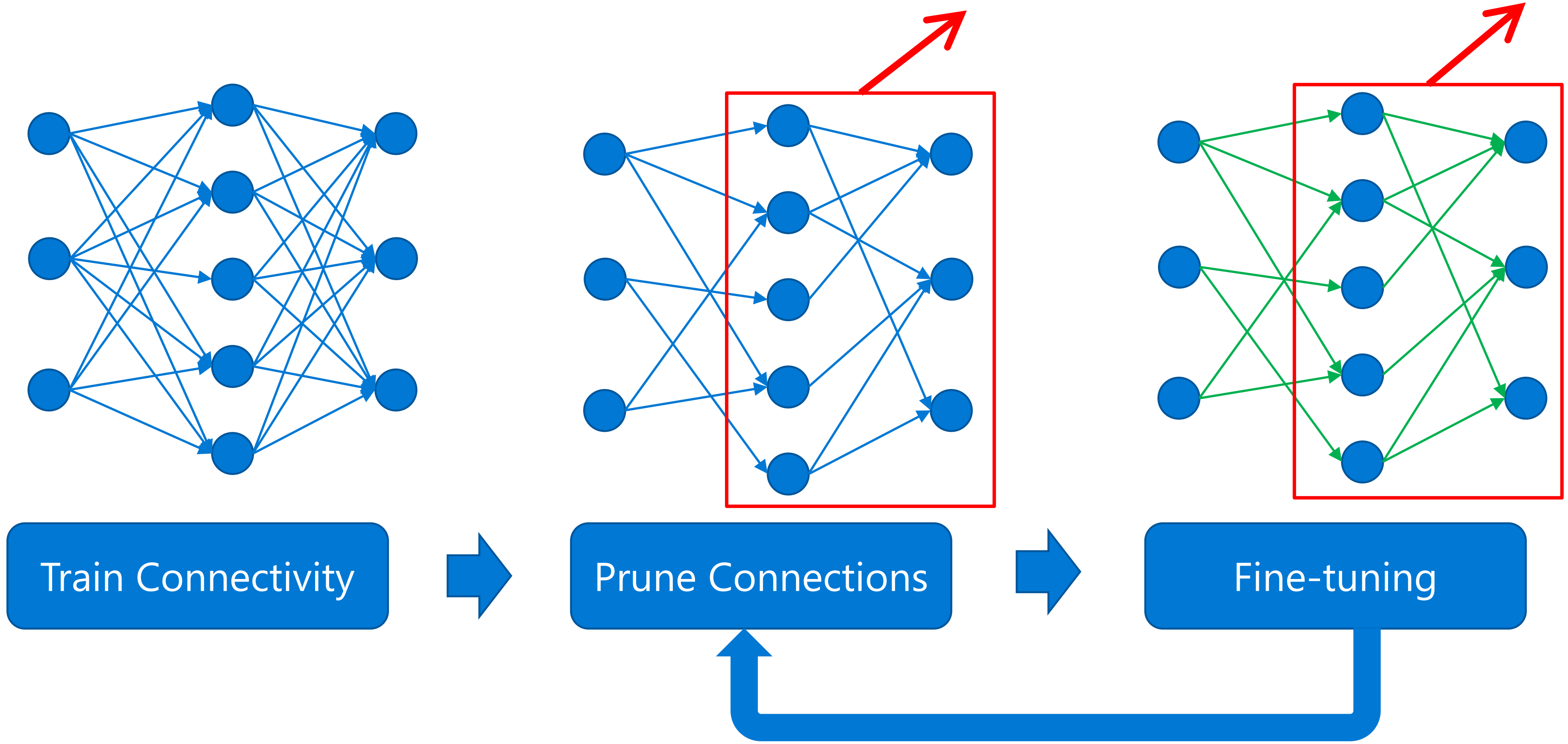
剪枝阈值


<20% 最大值
<50% 最大值
1200
1000
800
600
400
200
0
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
不重要
次重要
重要

权重稀疏化的三个步骤





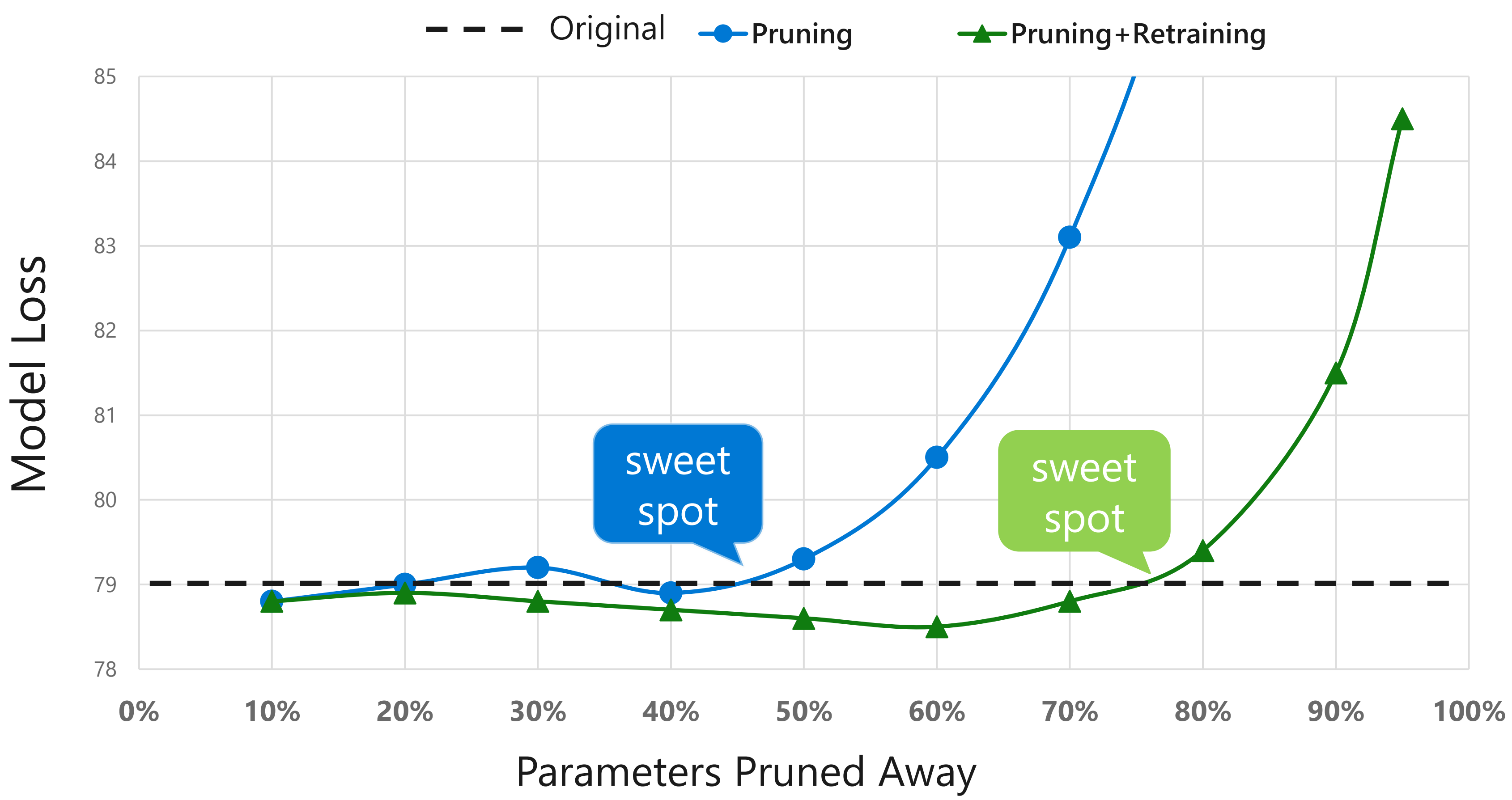


$$\widehat{w} \leftarrow abs(w)$$

$$if\ \widehat{w_i} \le Threshold$$

$$w_i \leftarrow 0$$

通过精调（fine-tuning）提高模型稀疏度

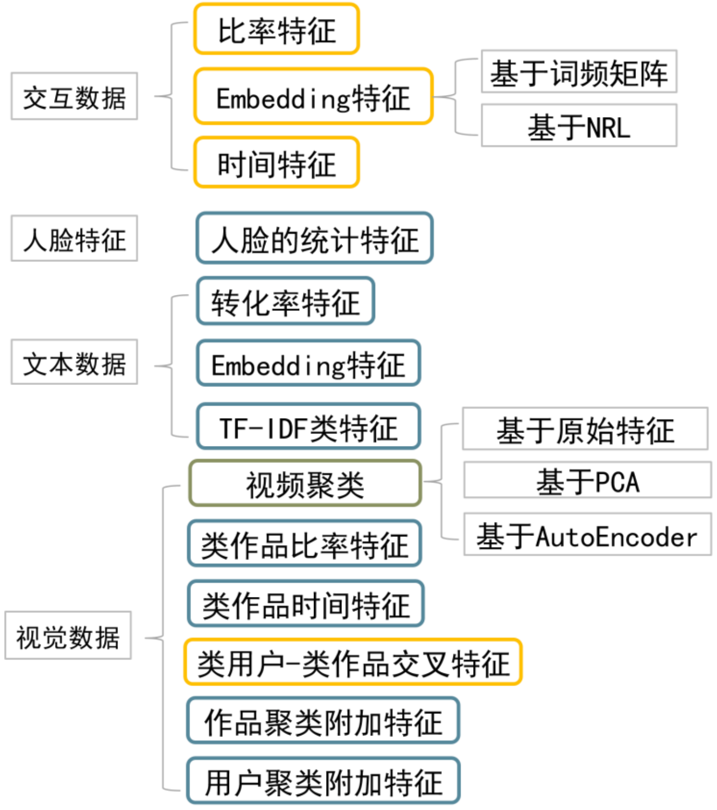


思考题

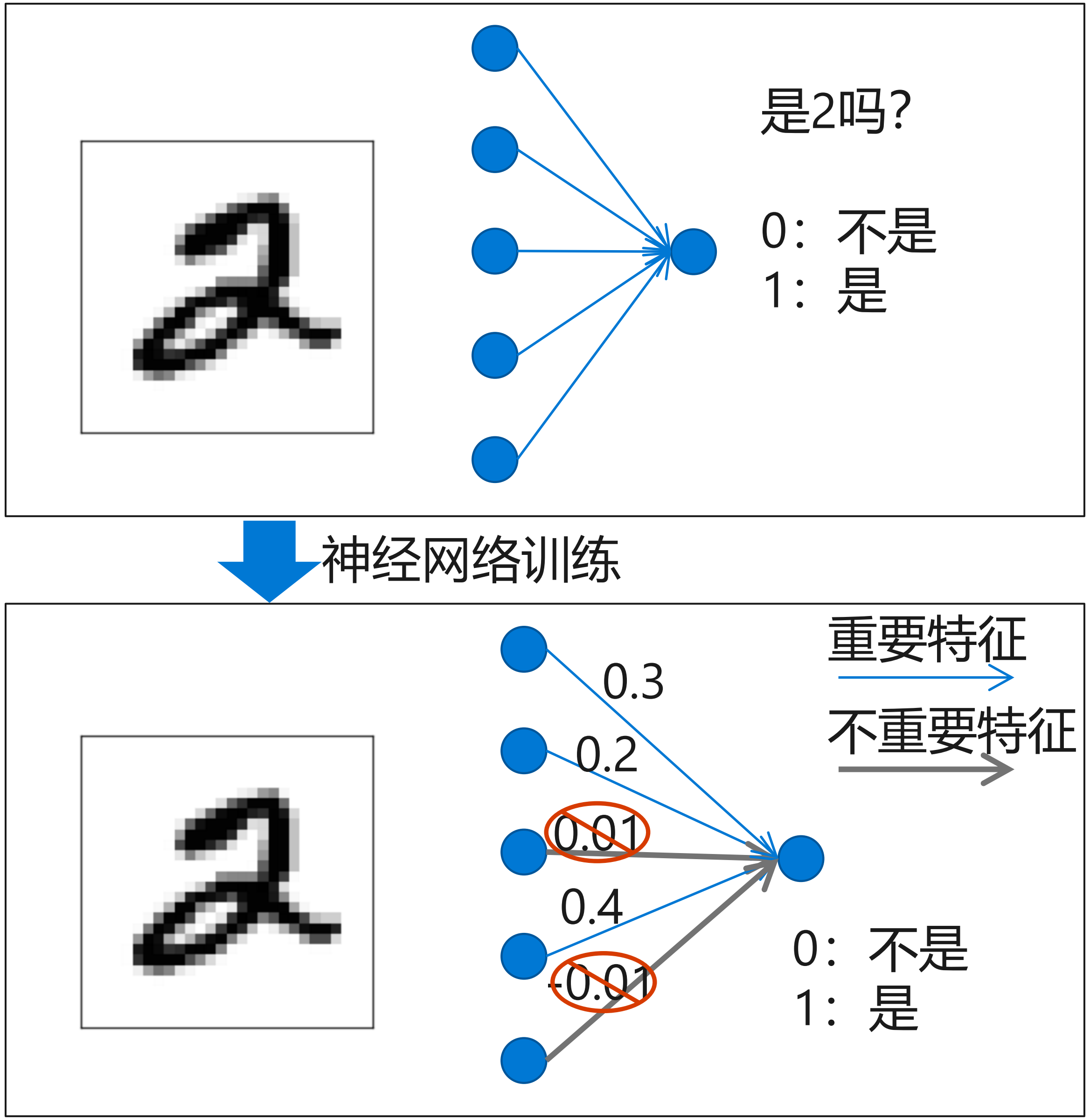
- 权重稀疏的本质是什么？

从特征工程到深度学习

传统方法：人工选取特征

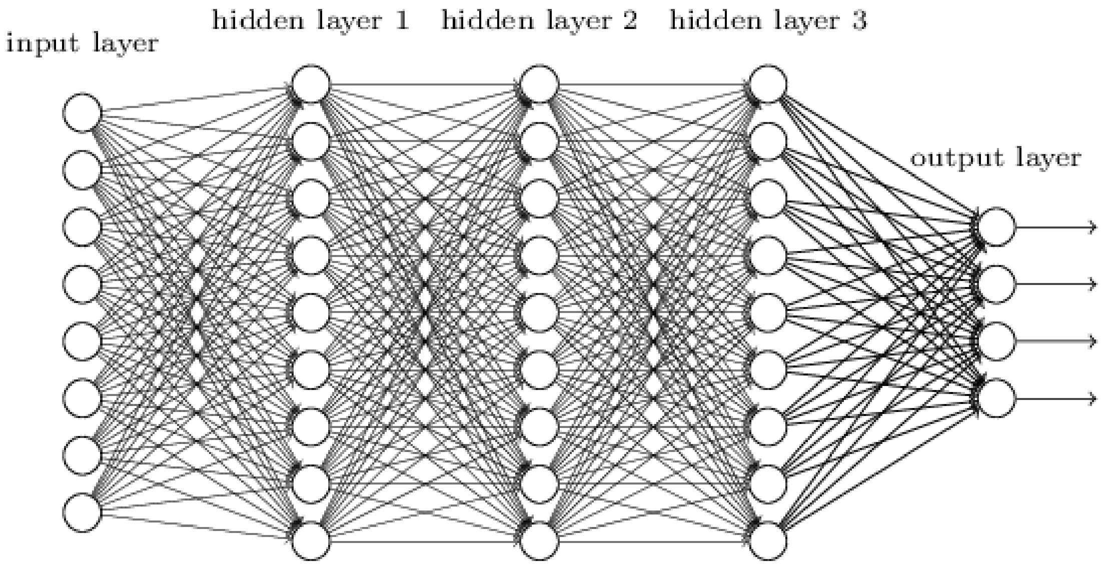


深度学习：自动学习特征

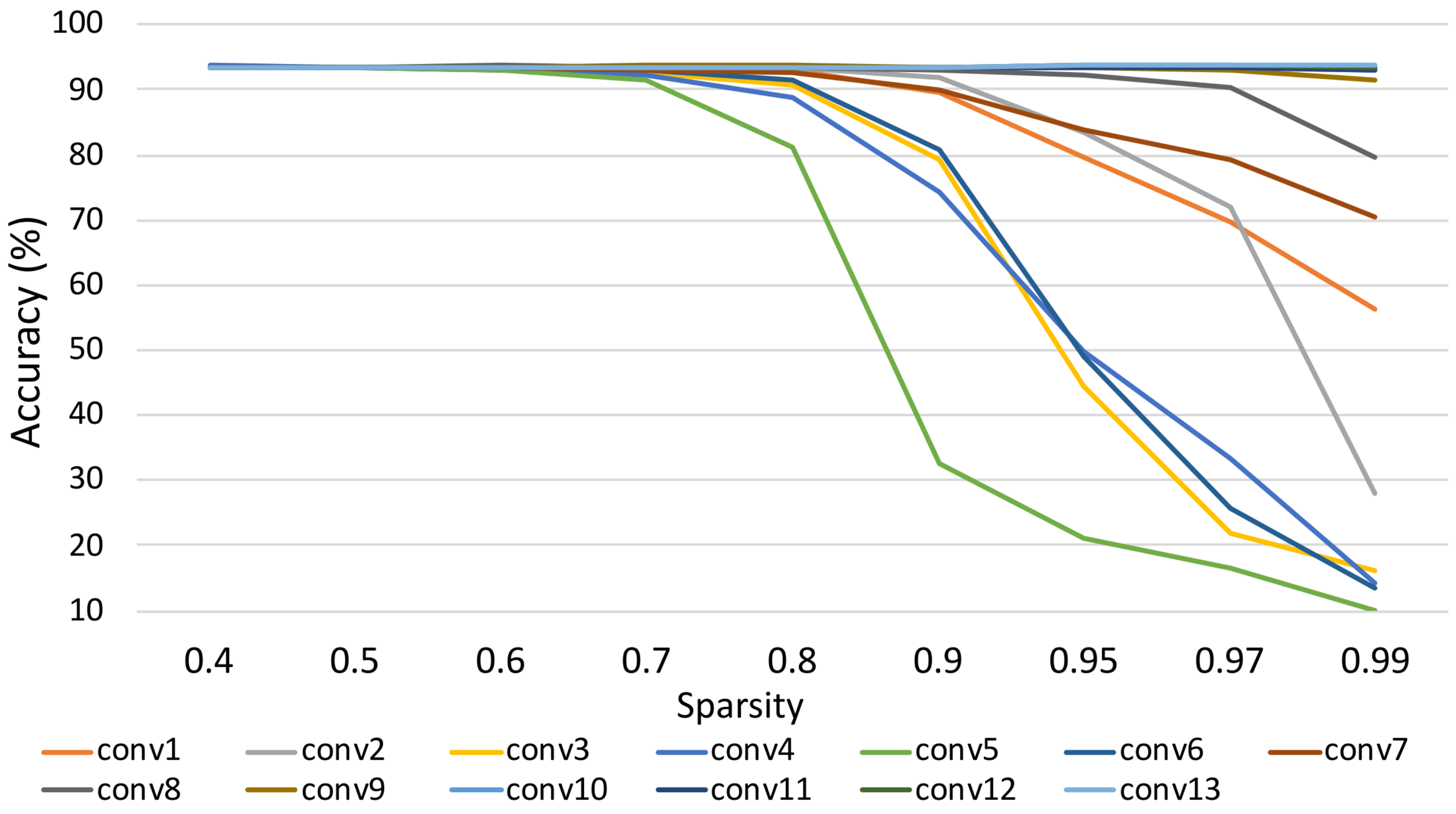


思考题

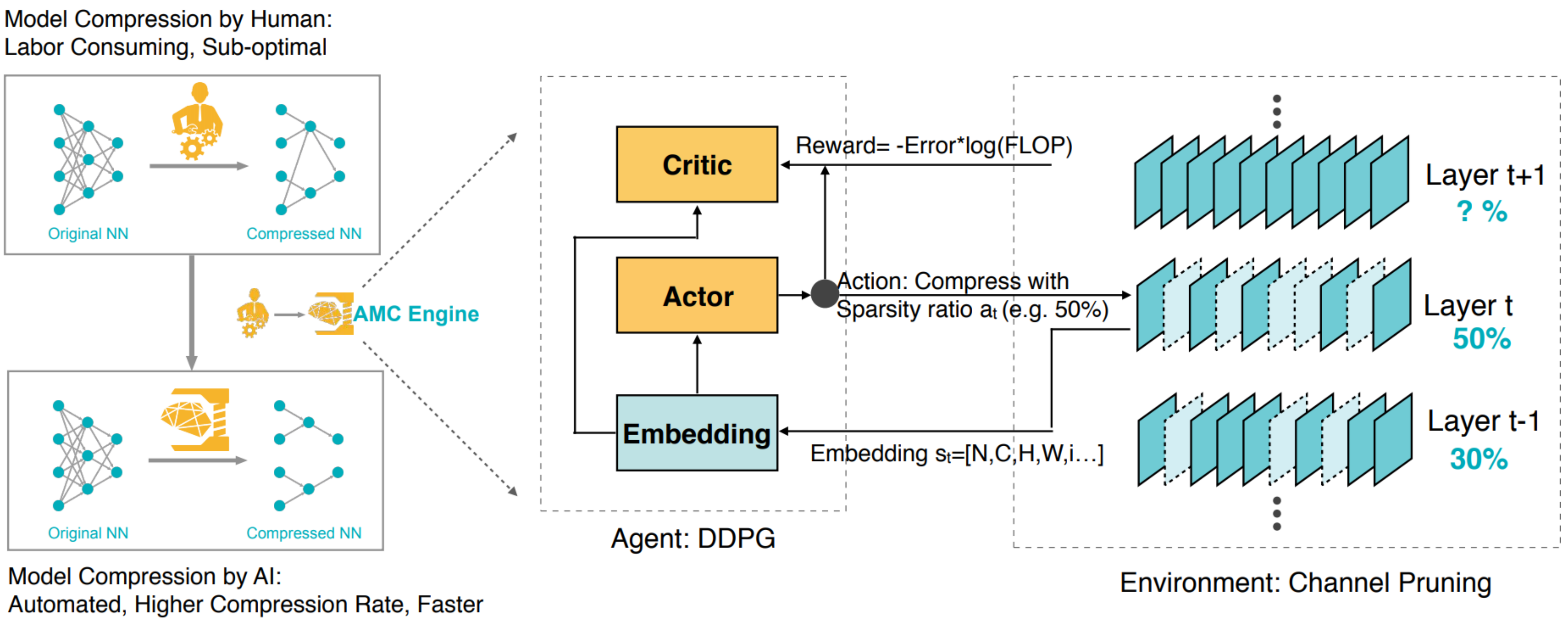
- 神经网络有许多层，每一层的稀疏度都一样吗？



稀疏的敏感性：VGG-16



研究课题 1：各层最优稀疏度



[3] AMC: AutoML for Model Compression and Acceleration on Mobile Devices

思考题

- 各层稀疏度和调参过程中的特征图数量的关系是什么？
- 如果某一层特别稀疏，是不是意味着该层特征图太多了？

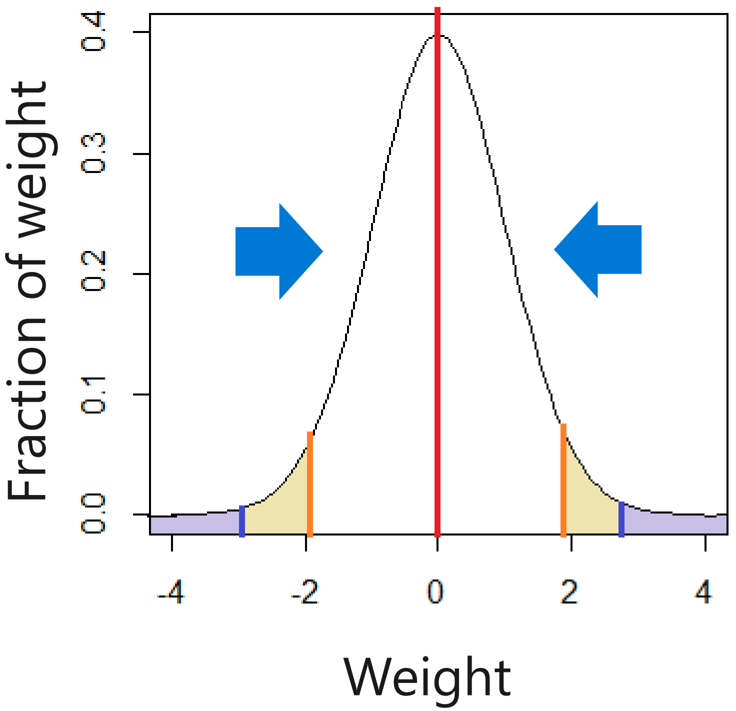
研究课题 2：稀疏与正则化

$L1$:

$$C = L(W,b) + \boxed{\lambda \sum_{w} |w|}$$

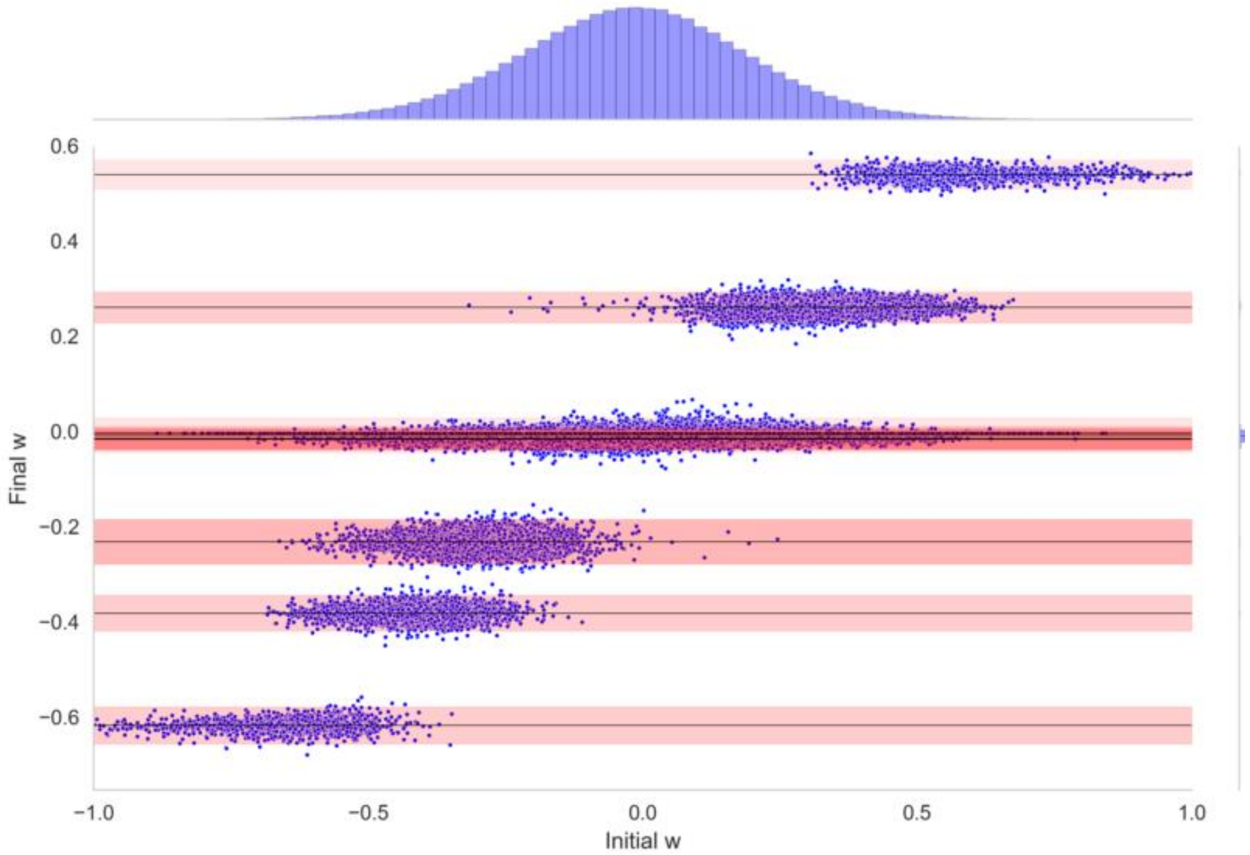
$L2$:

$$C = L(W,b) + \boxed{\lambda \sum_{w} w^2}$$



混合高斯模型

$$\mathcal{L}(q(\mathbf{w}), \mathbf{w}) = \underbrace{\mathbb{E}_{q(\mathbf{w})}\left[-\log p(\mathcal{D}|\mathbf{w})\right]}_{\mathcal{L}^E} + \boxed{\underbrace{\mathrm{KL}(q(\mathbf{w})||p(\mathbf{w}))}_{\mathcal{L}^C}}$$



$$\boxed{\mathbb{E}_{q(\mathbf{w})}\left[-\log p(\mathbf{w})\right] - \boxed{H(q(\mathbf{w}))}}$$

$$\boxed{= -\int_{\mathbb{R}^I} \mathcal{N}(\mathbf{w}|\mathbf{0}, \sigma\mathbf{I}) \log \mathcal{N}(\mathbf{w}|\mathbf{0}, \sigma\mathbf{I})}$$

[2] Soft Weight-Sharing for Neural Network Compression

思考题

- 对稀疏来说，L1和L2正则化的区别是什么？
- 还有哪些正则化方法？

研究课题 3：硬件友好的稀疏化方法

1. Train Connectivity

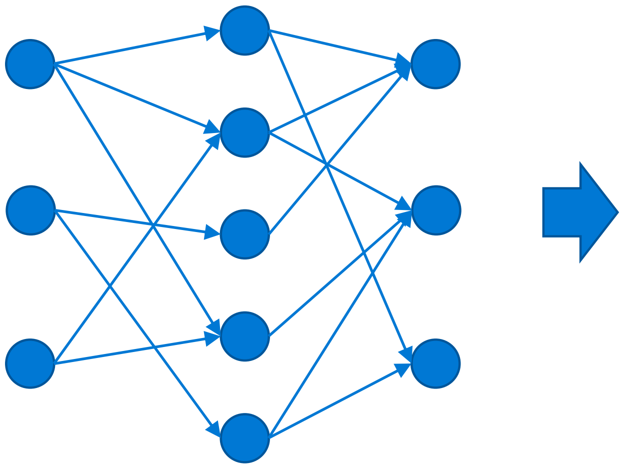
2. Prune Connections

$$\hat{w} \leftarrow abs(w)$$

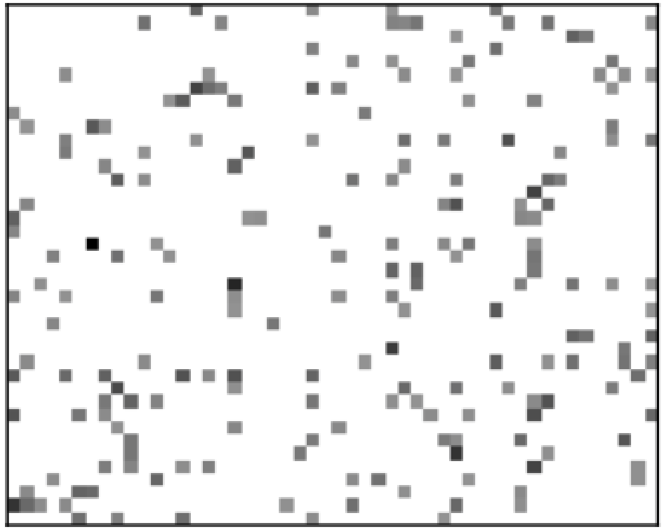
$$if\ \widehat{w_i} \leq Threshold$$

$$w_i \leftarrow 0$$

3. Train Weights

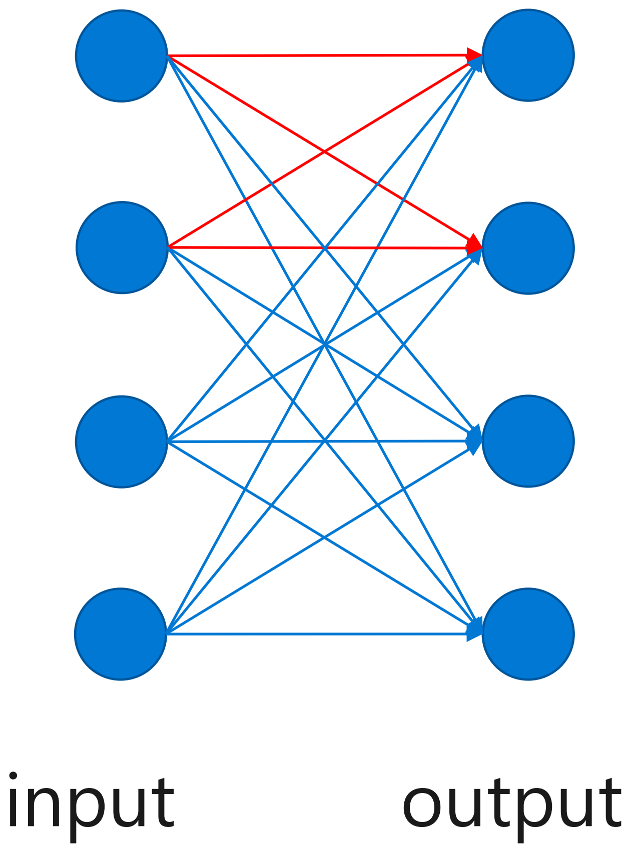


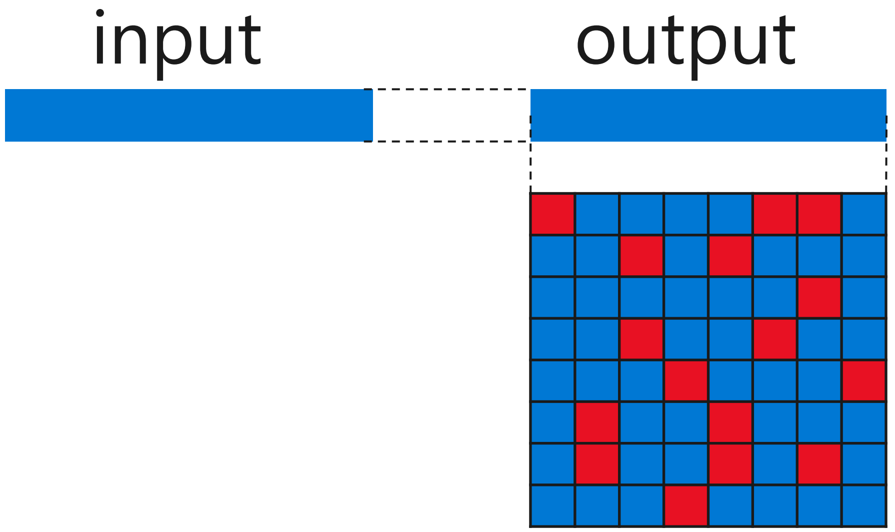
Sparsity = 50%~90%

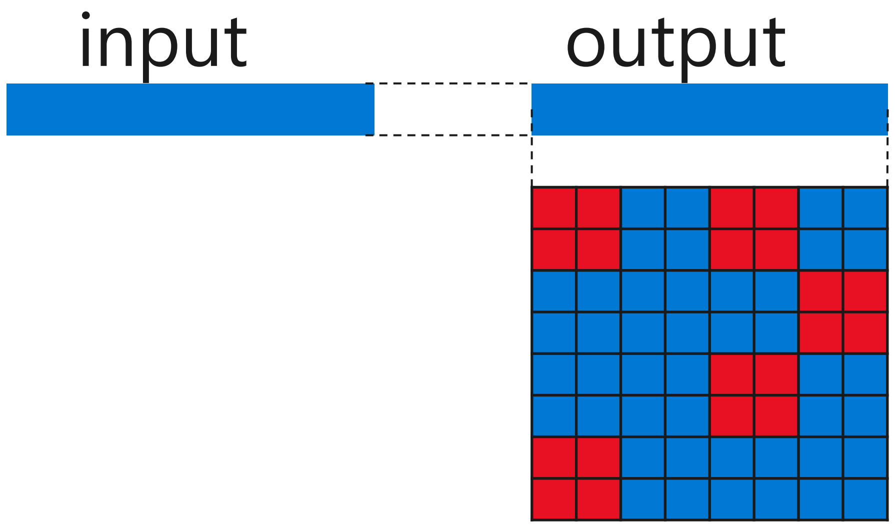


高度非规则的矩阵形状

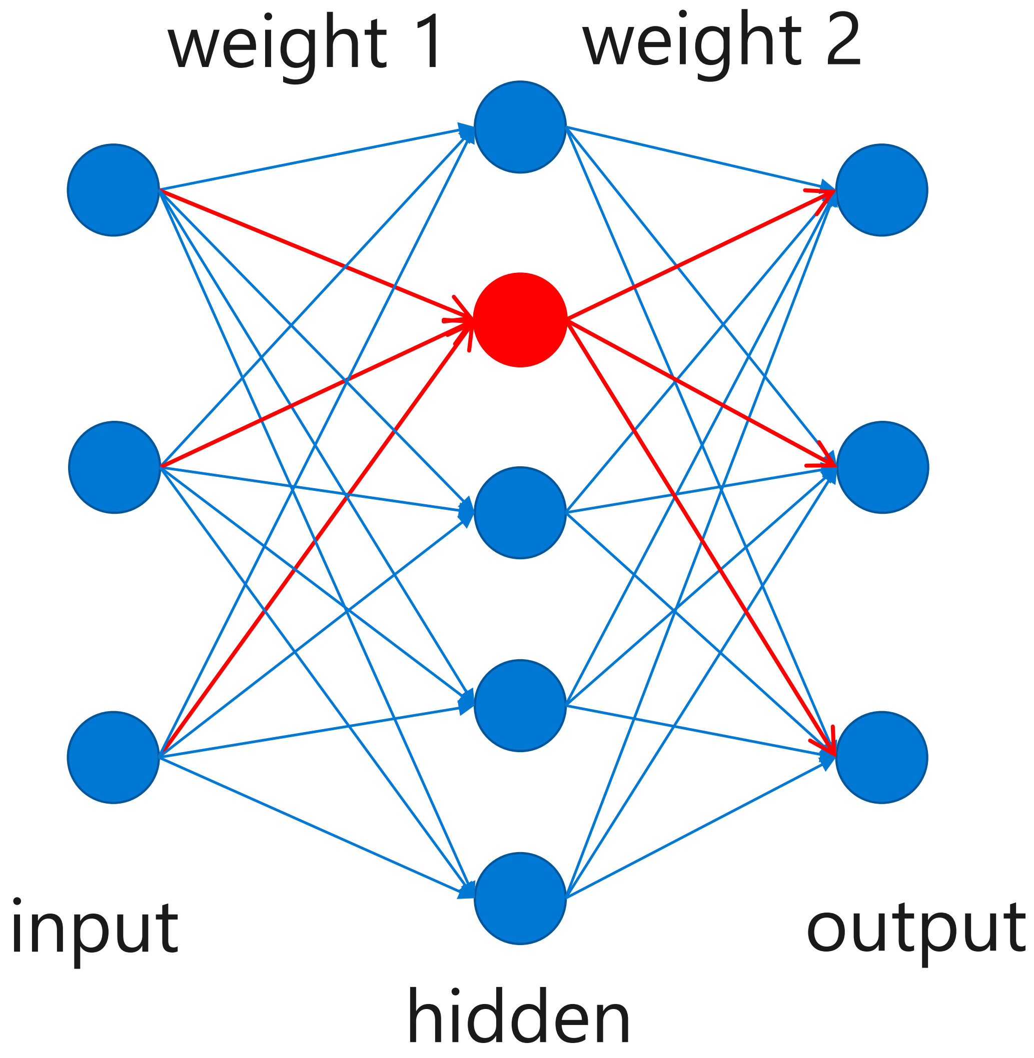
细粒度稀疏与粗粒度稀疏

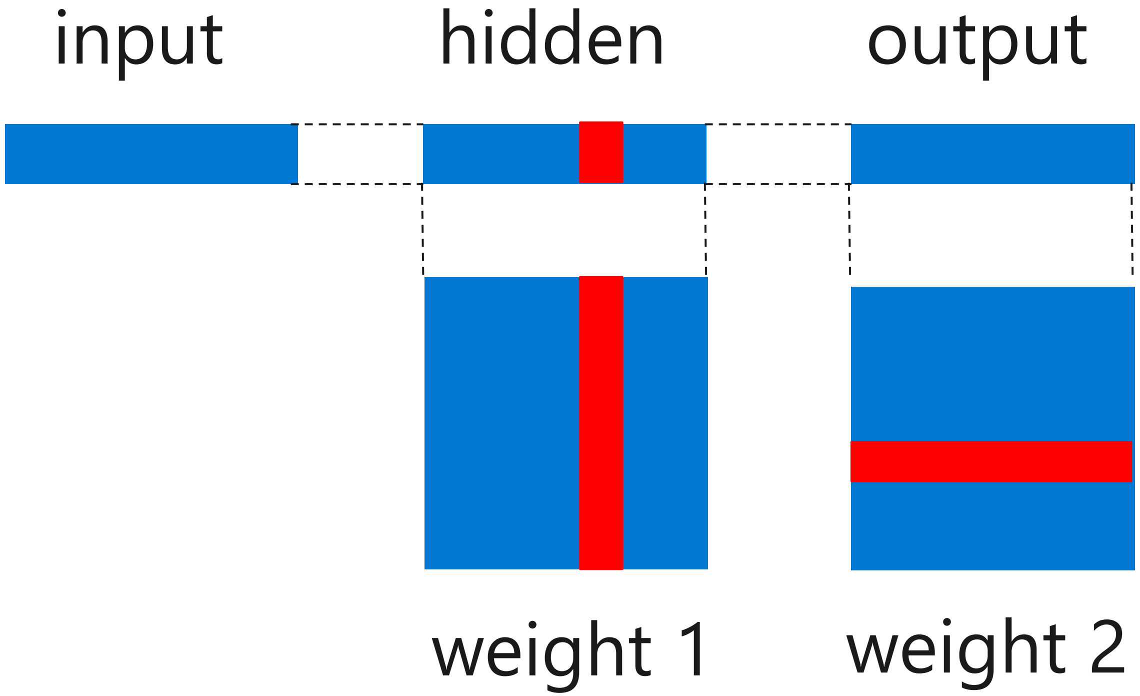




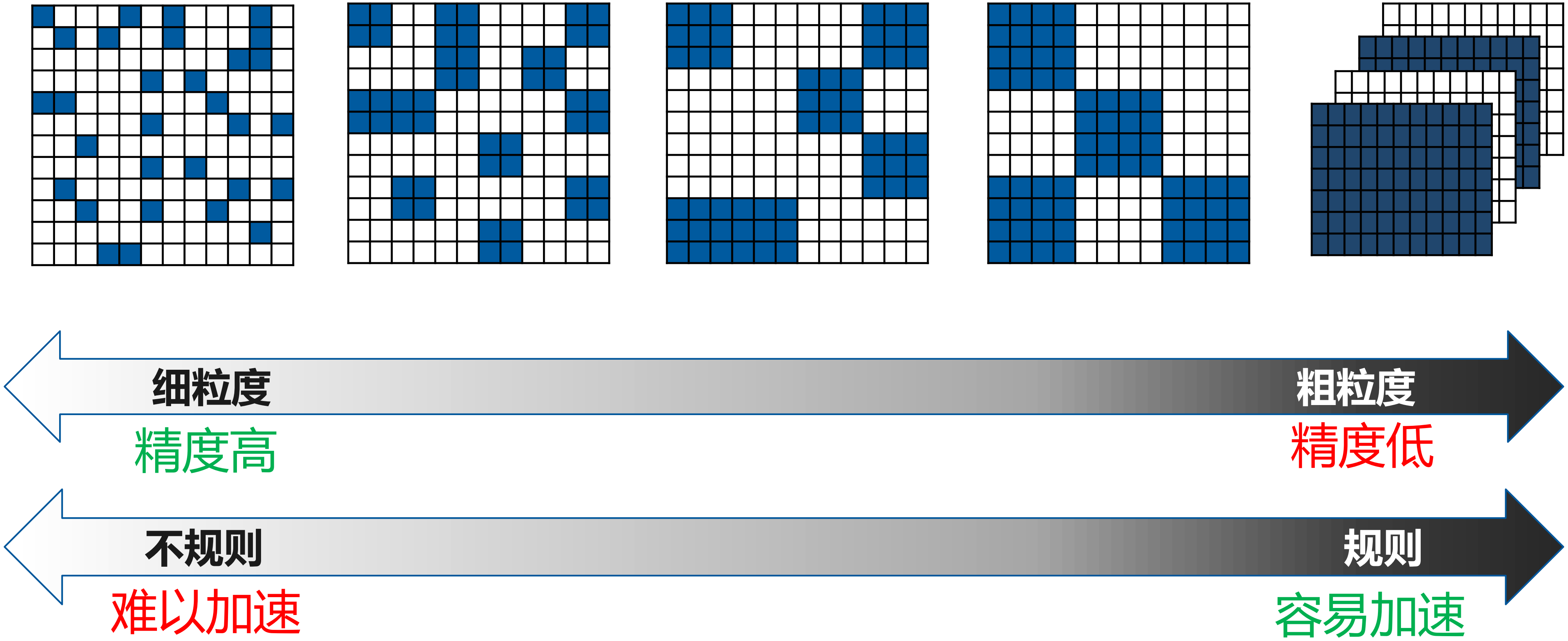


通道剪枝（Channel Pruning）

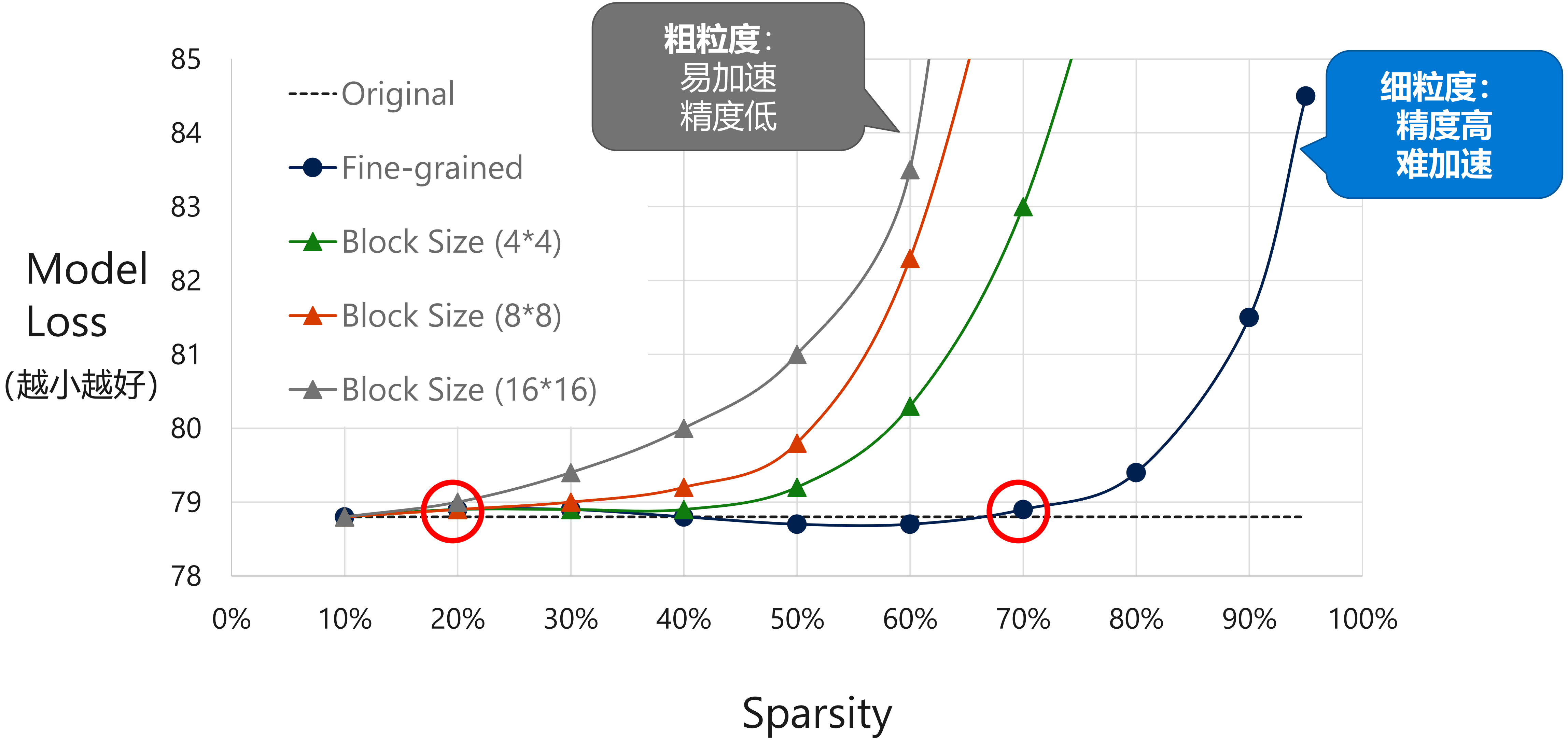




加速与精度的权衡

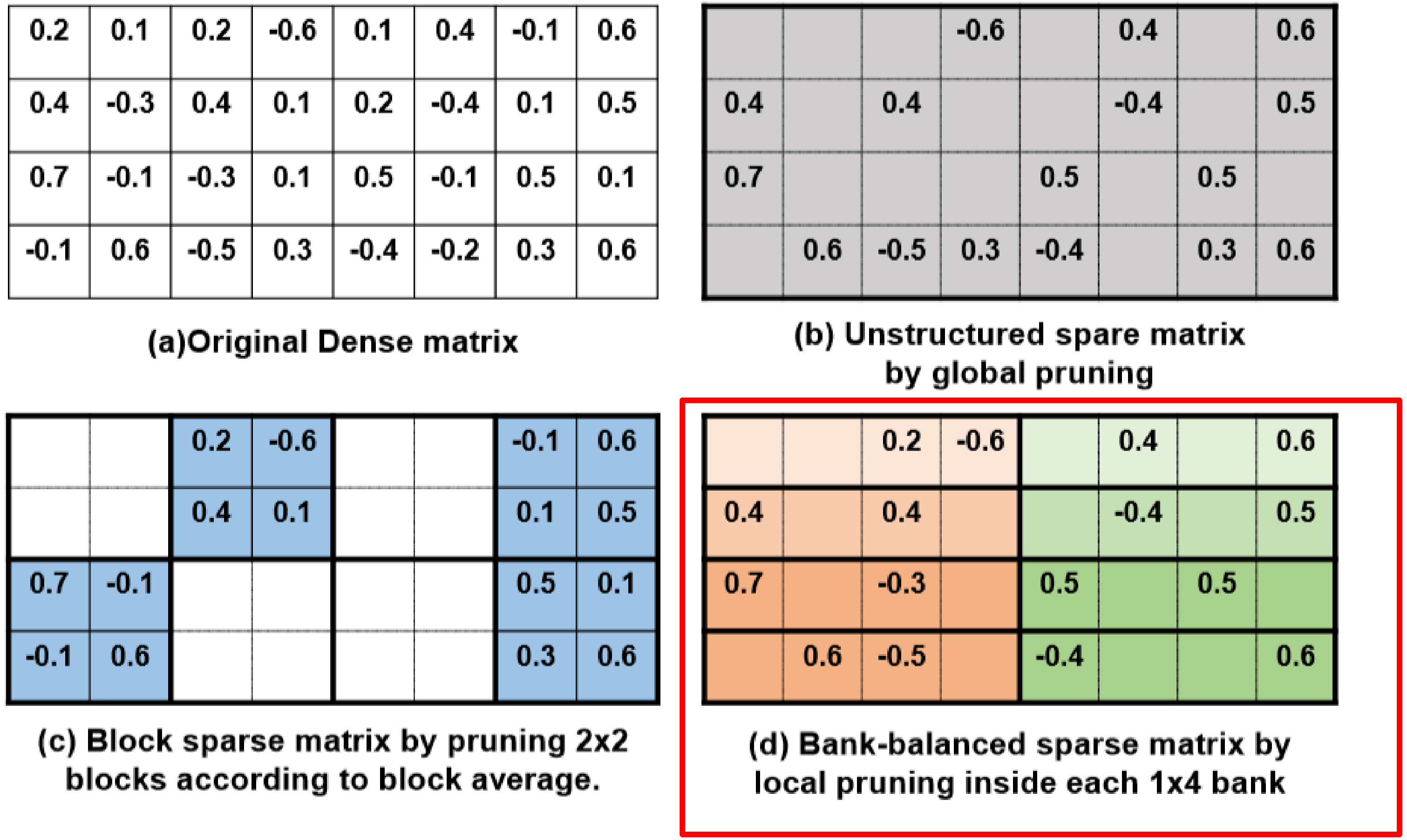


加速与精度的权衡



粗粒度：
易加速
精度低
细粒度：
精度高
难加速
Model Loss
(越小越好)
Original
Fine-grained
Block Size (4*4)
Block Size (8*8)
Block Size (16*16)
85
84
83
82
81
80
79
78
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Sparsity

FPGA’19 和 AAAI’19



(a)Original Dense matrix

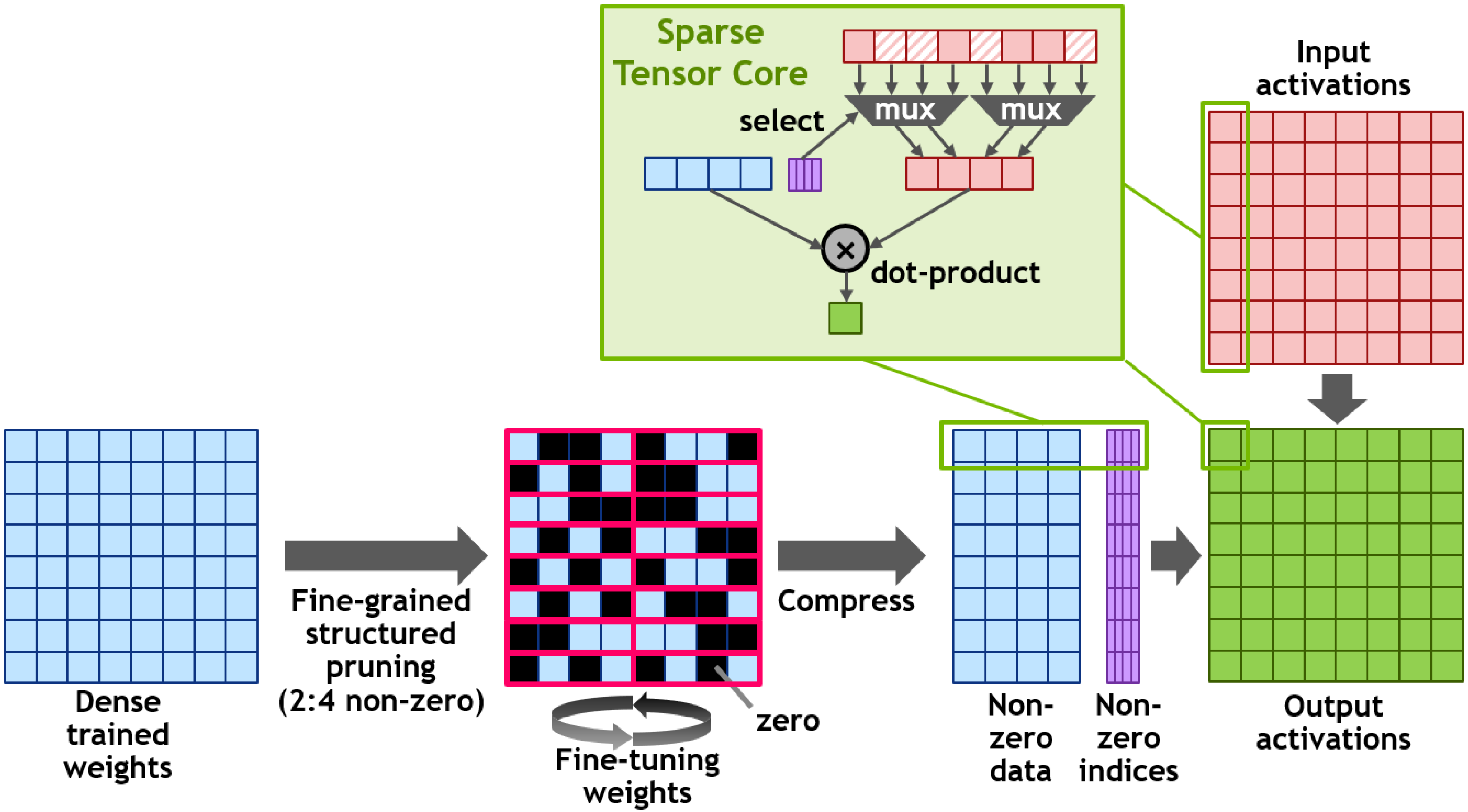
(b) Unstructured spare matrix by global pruning

(c) Block sparse matrix by pruning 2x2 blocks according to block average.

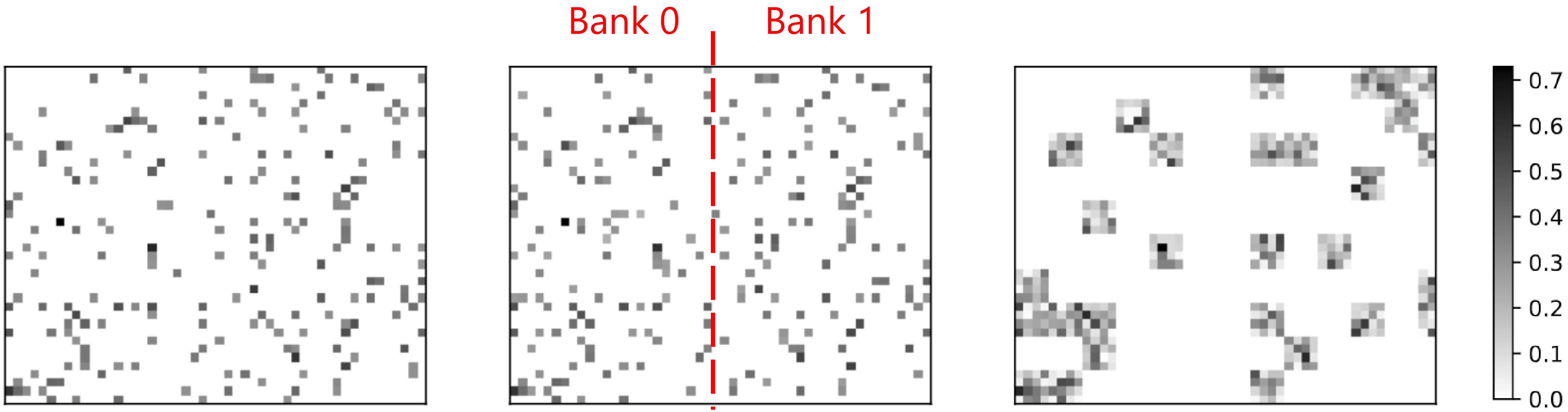
(d) Bank-balanced sparse matrix by local pruning inside each 1x4 bank

Efficient and effective sparse LSTM on FPGA with bank-balanced sparsity
Balanced sparsity for efficient DNN inference on GPU

英伟达 A100 GPU：sparse tensor core

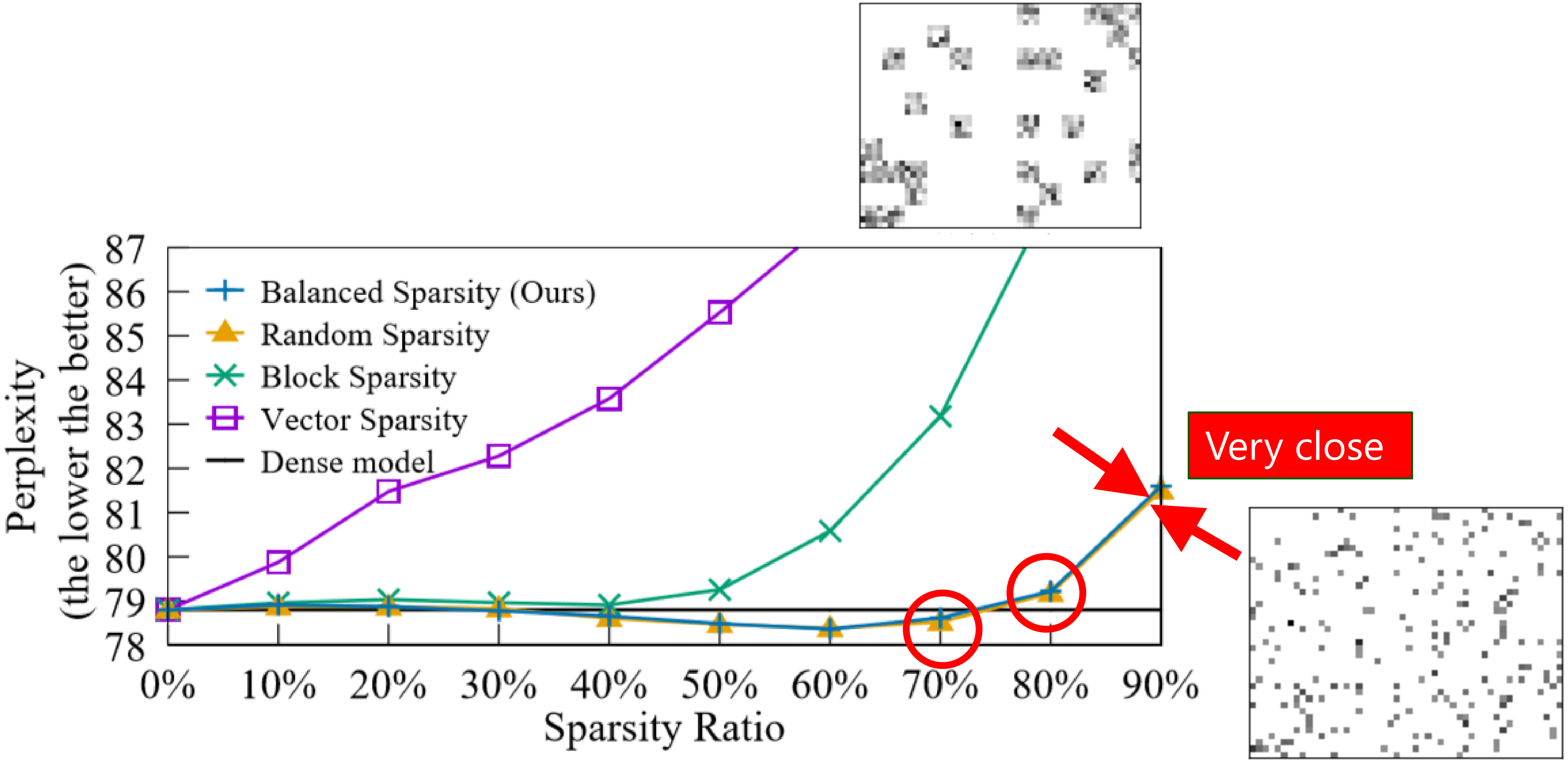


剪枝效果可视化



(a) 细粒度剪枝　(b) BBS (我们的方法)　(c) 粗粒度剪枝

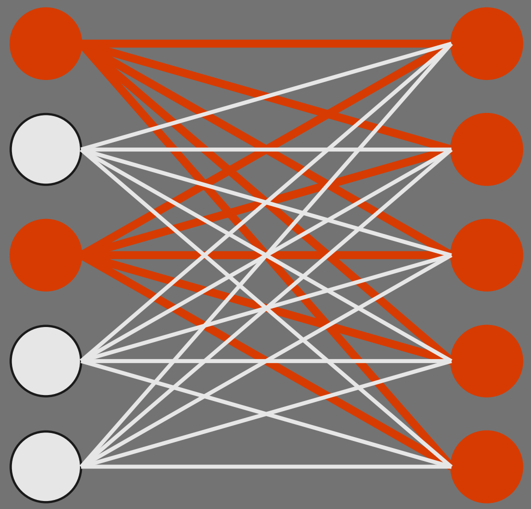
模型准确度对比


Perplexity
(the lower the better)
87
86
85
84
83
82
81
80
79
78
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
Sparsity Ratio
Balanced Sparsity (Ours)
Random Sparsity
Block Sparsity
Vector Sparsity
Dense model
Very close

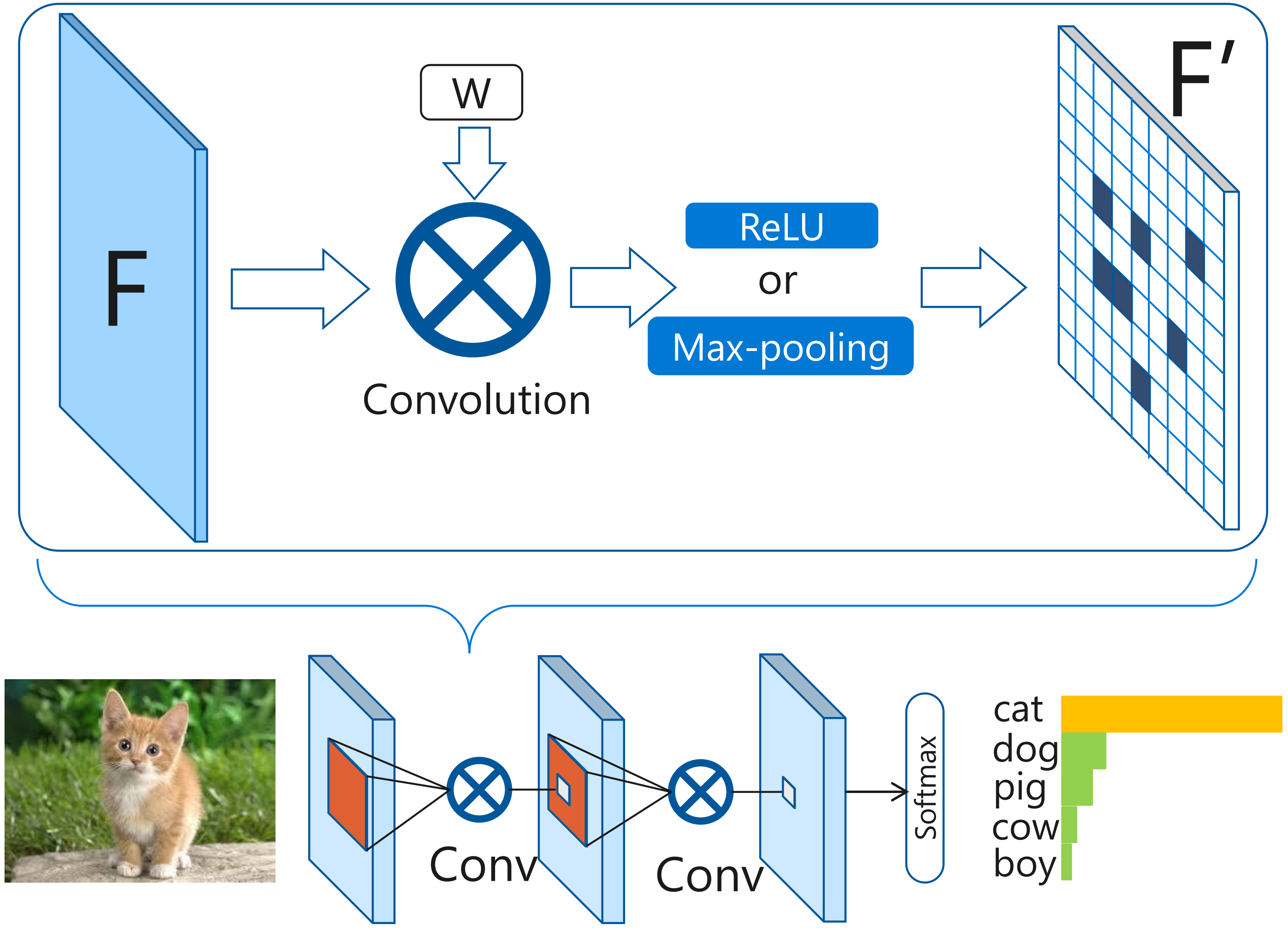
思考题

- 卷积神经网络上，可以有多少种不同粒度的剪枝？
- 剪枝和架构搜索（architecture search）的关系是什么？

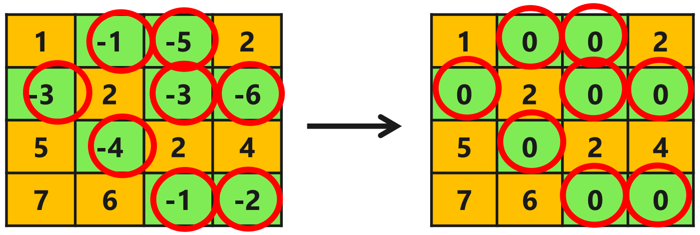
激活稀疏



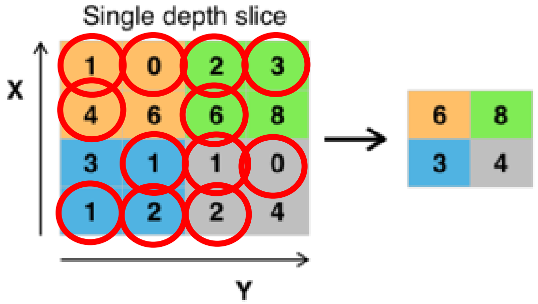
神经网络的激活（activation）稀疏



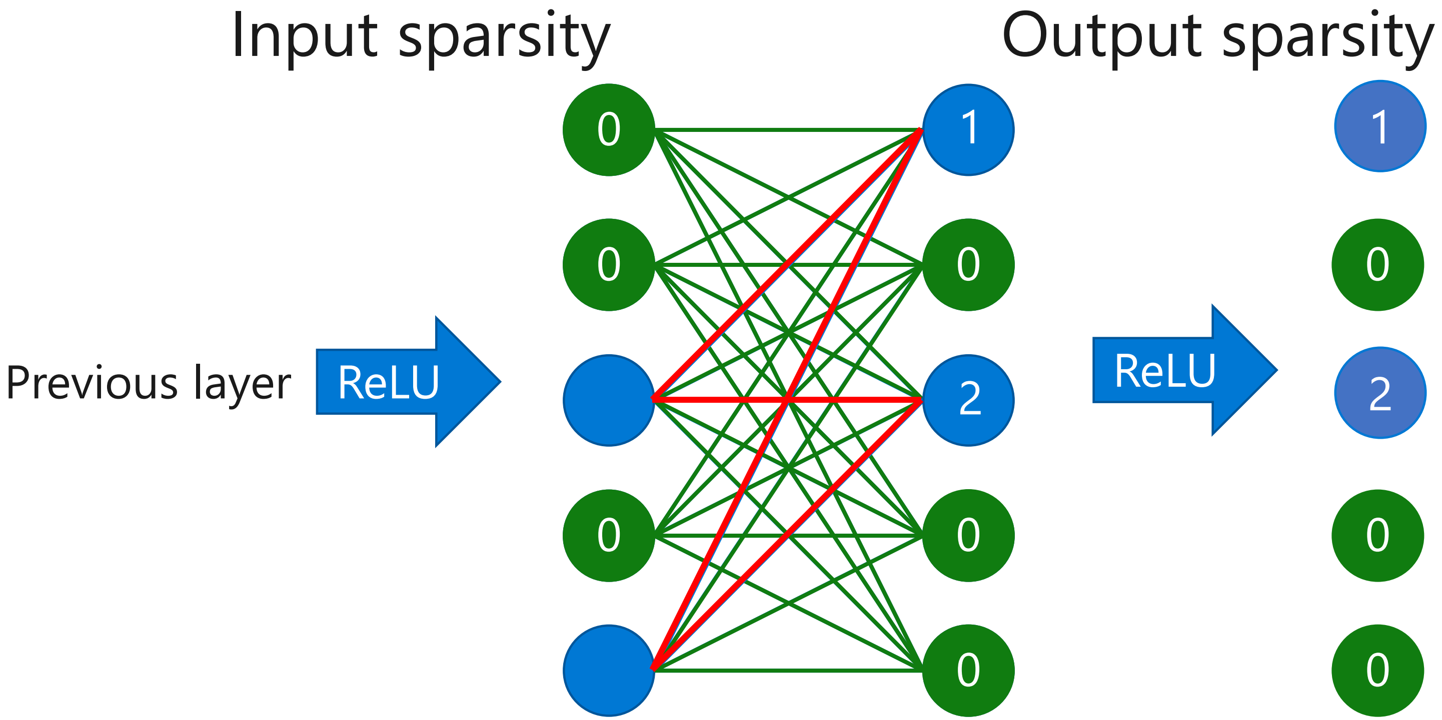
- ReLU
  - $y = max(0,x)$



- Max-pooling
  - $y = max(x_i \mid i=\{1,2,\ldots,n\})$



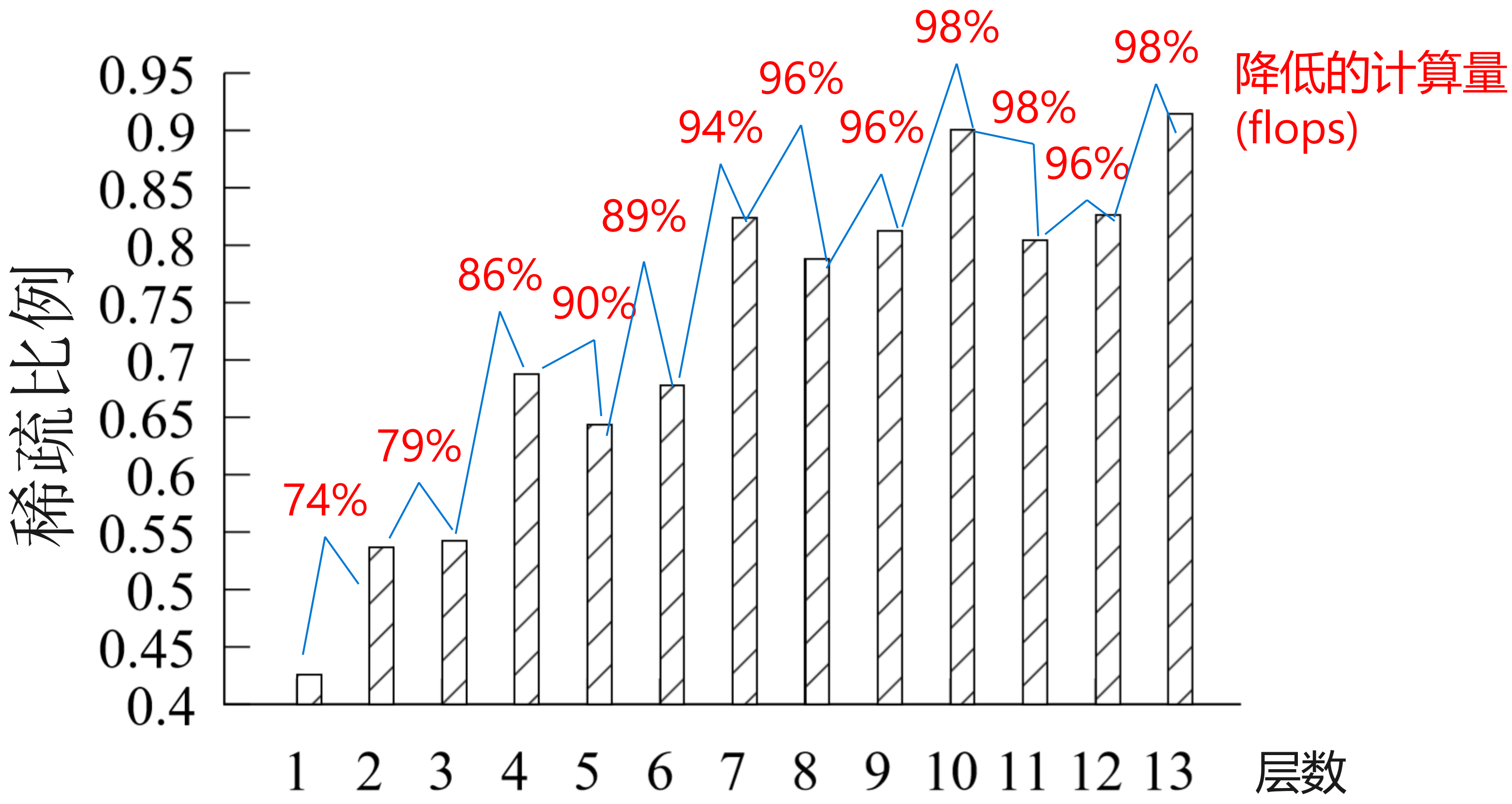
激活稀疏的加速机会



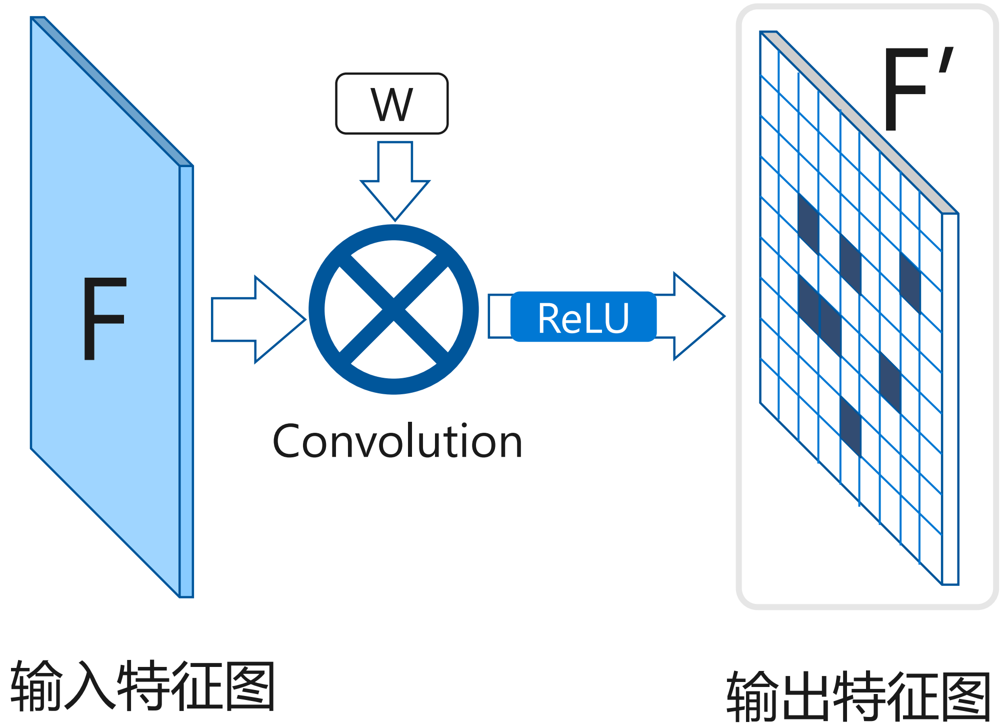
Effective computation = 4/25

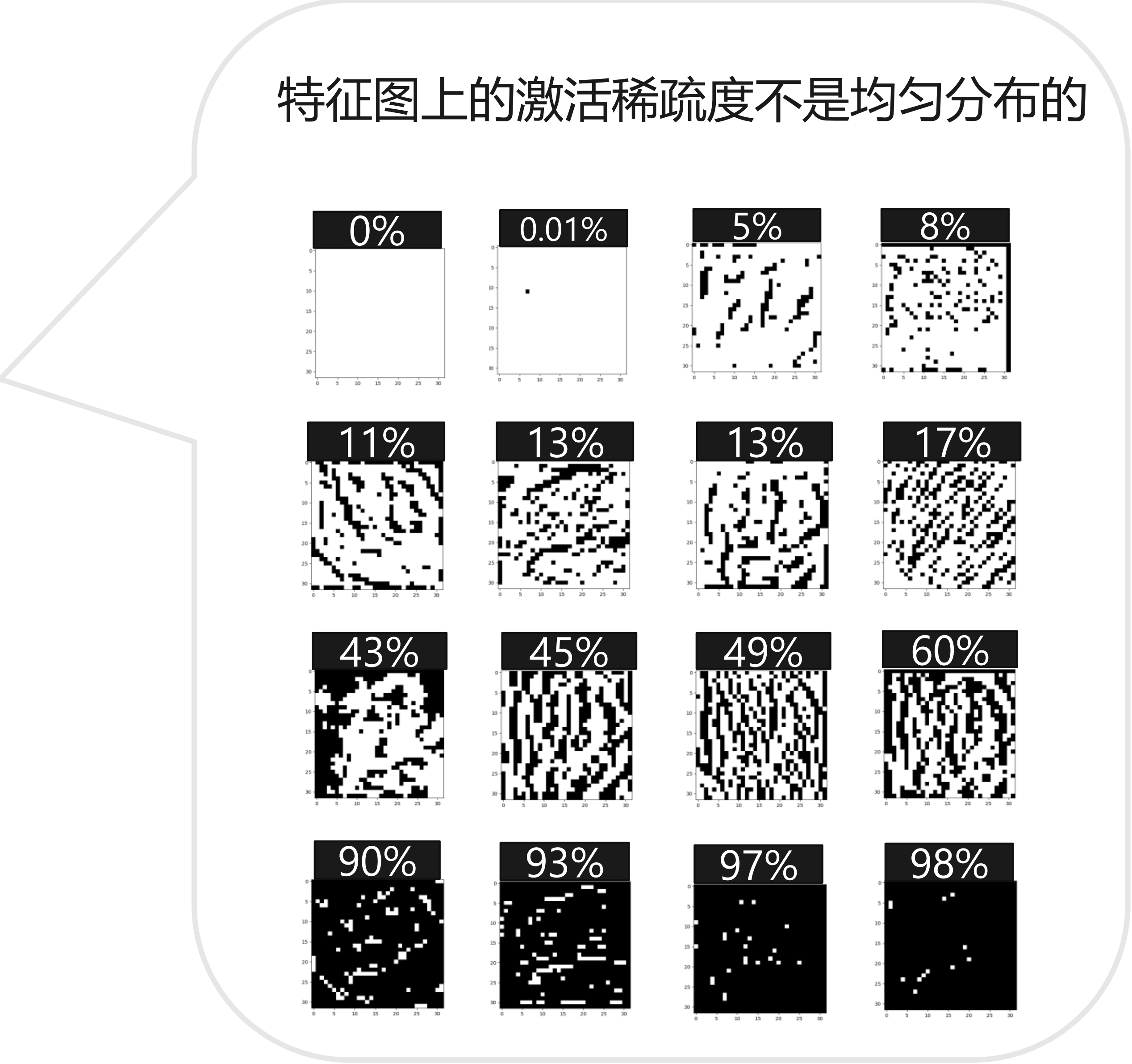
Resnet-16的激活稀疏

- 稀疏度 : 40% ~ 90%
- 稀疏性同时存在于输入和输出特征图



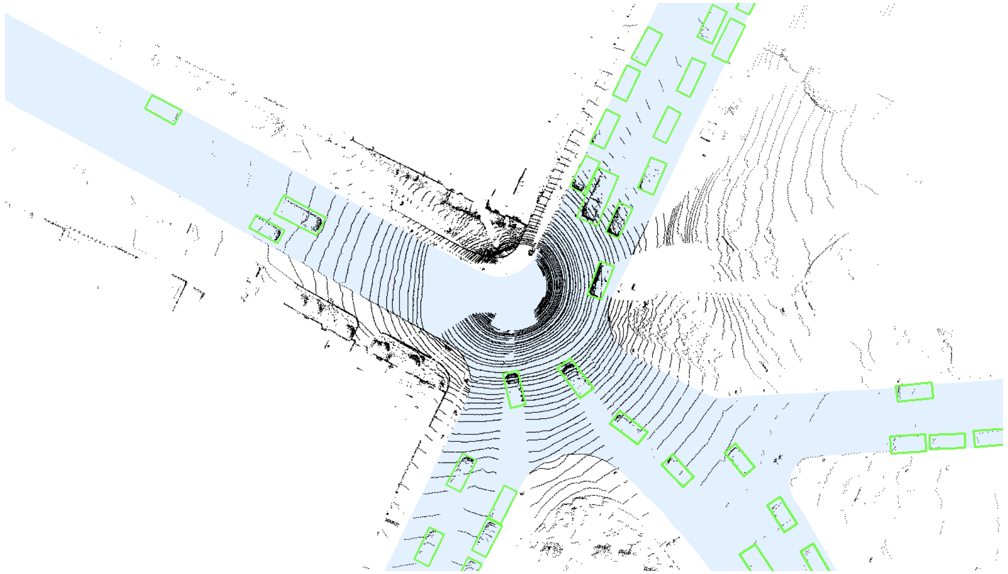
真实的例子：ResNet-16 layer 2

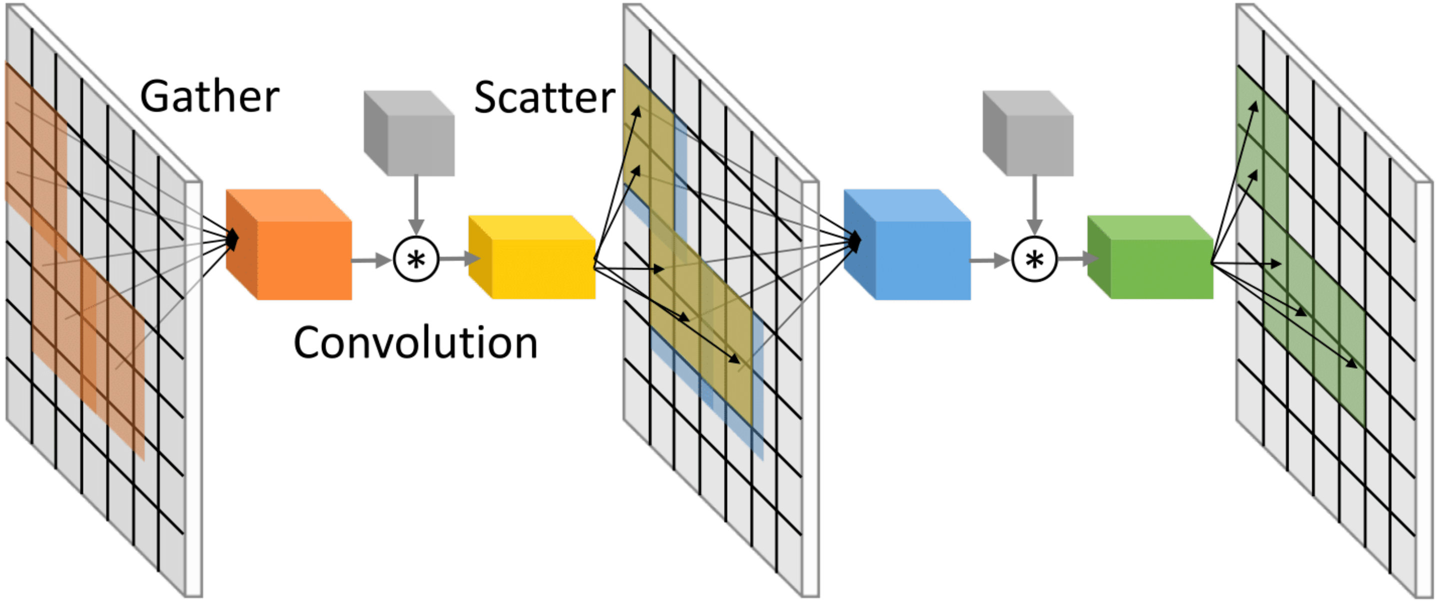




研究课题 4：结构化特征图稀疏

SBnet: 借助外部信息给特征图施加方形掩码



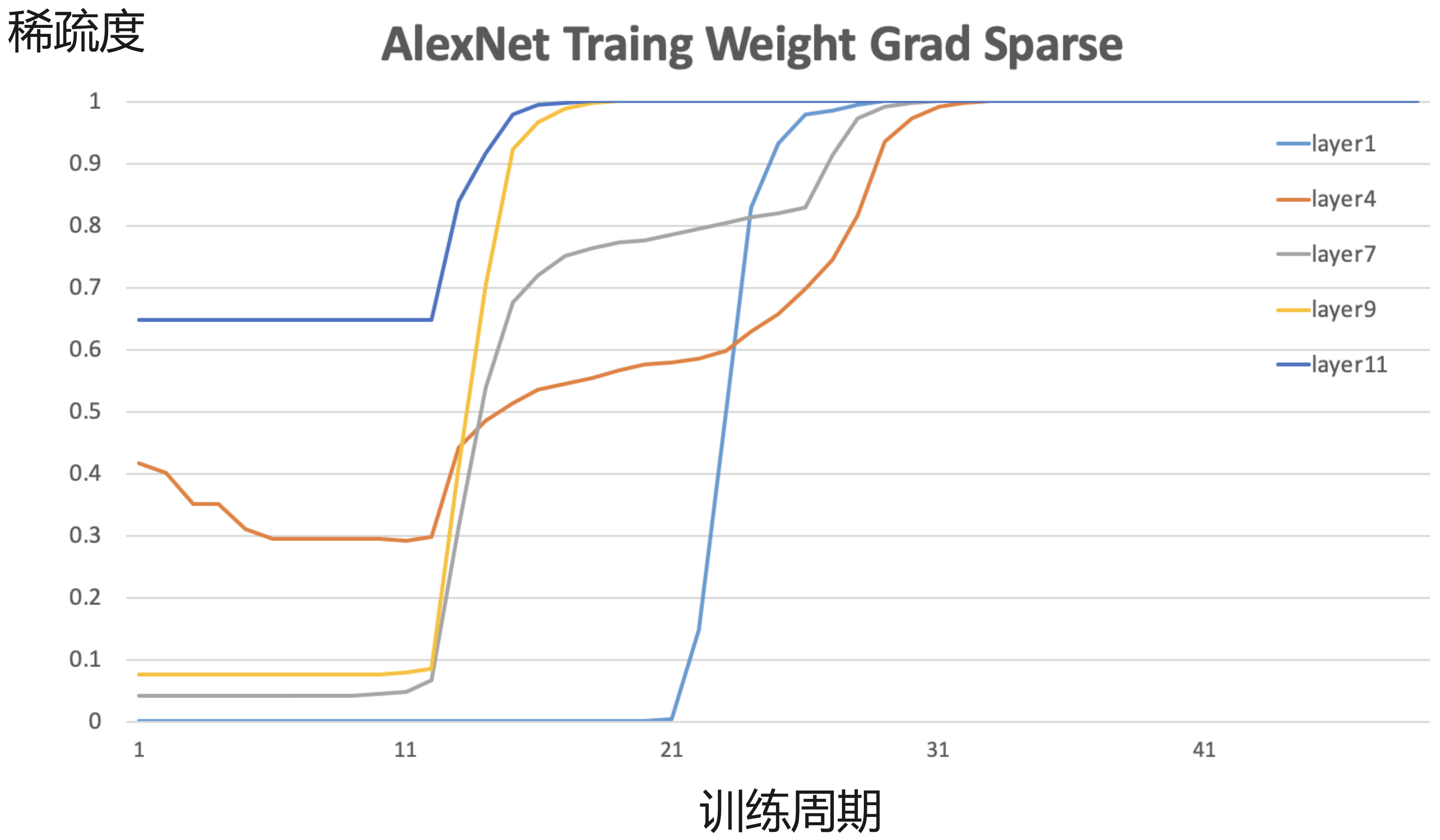


思考题：

- 从数据依赖角度看，权重稀疏与激活稀疏的关系是什么？
- 能否把权重稀疏和激活稀疏结合考虑取得更大的加速比？

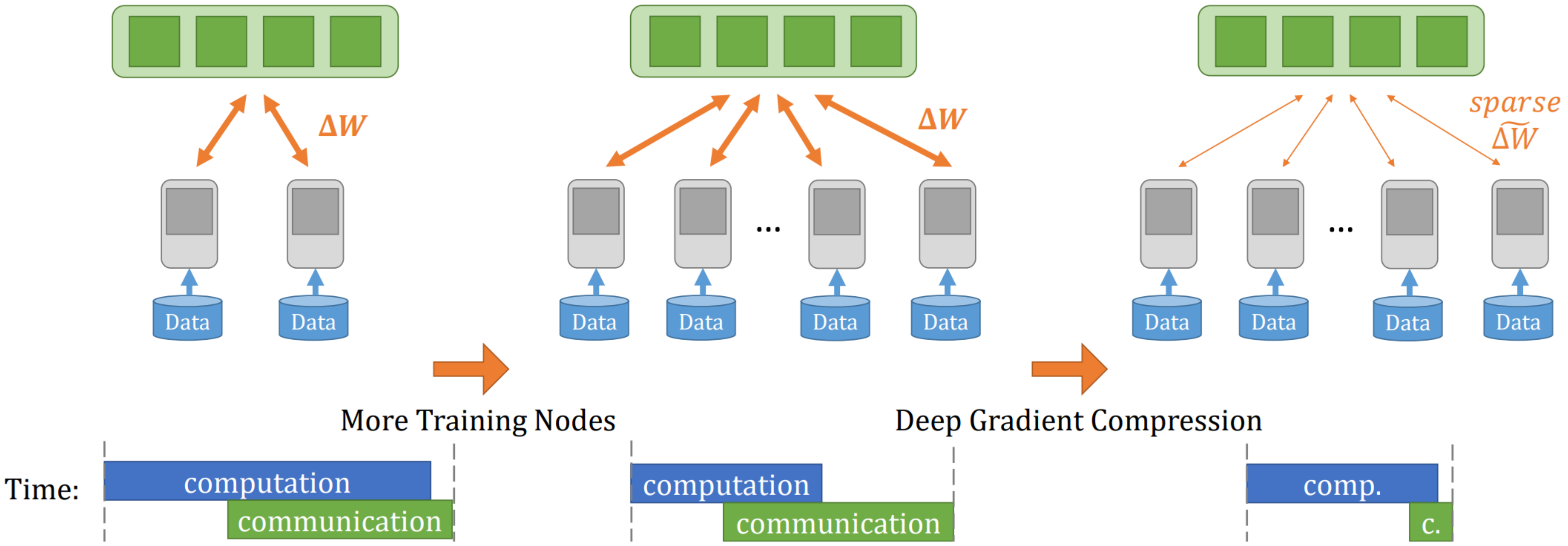
梯度稀疏

梯度稀疏：AlexNet



研究课题 5：加速训练

通过梯度稀疏降低分布式训练的通信代价



[] Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training

量化/低比特运算

整形与浮点

整形

符号位	数值位
1-bit	31-bit

最小值：-2^{31}　　最大值：$2^{31}-1$

浮点
(单精度)

符号位	指数位	尾数位
1-bit	8-bit	23-bit

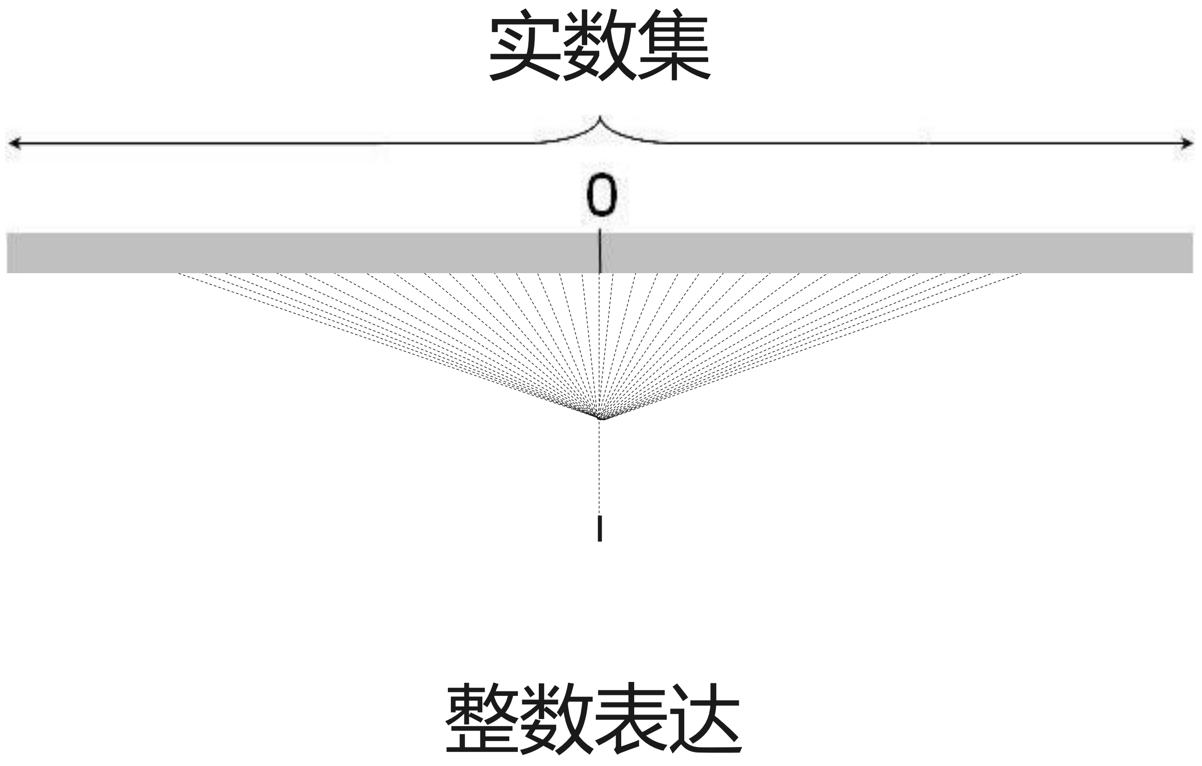
最小值：

$-2^{-126} \times 1.0$

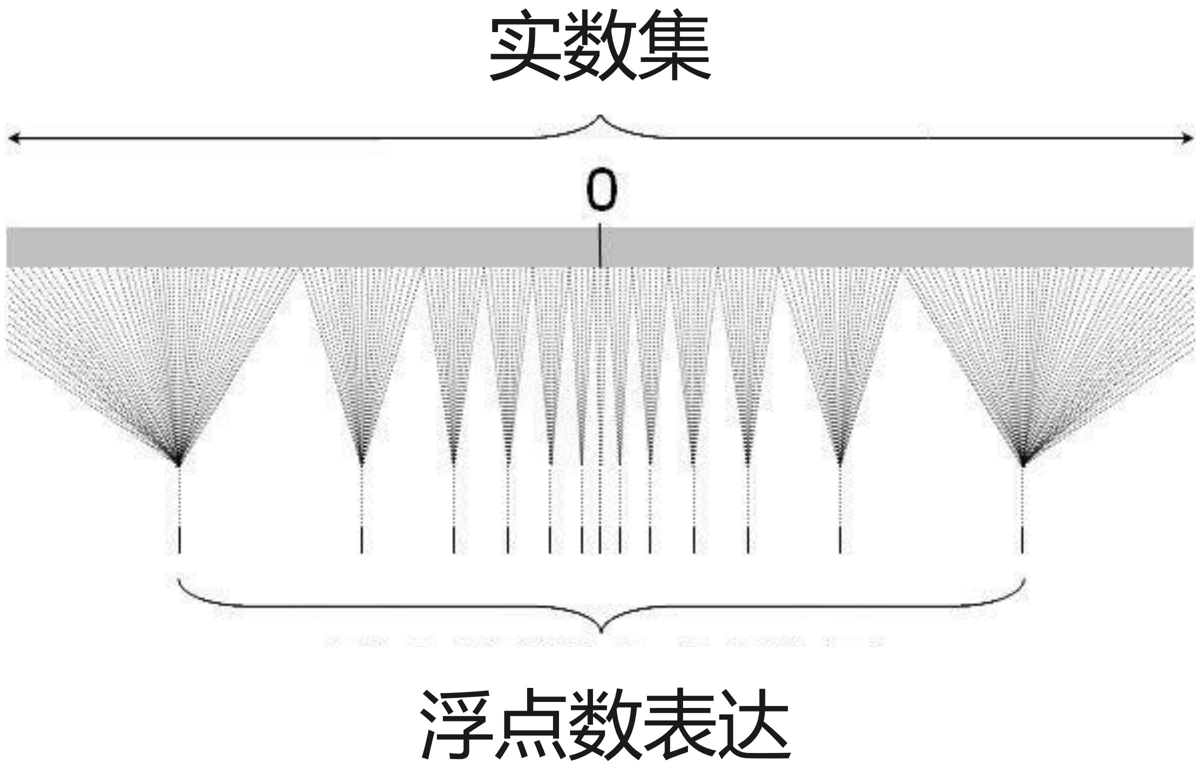
最大值：

$+2^{127} \times 1.11111111111111111111111$

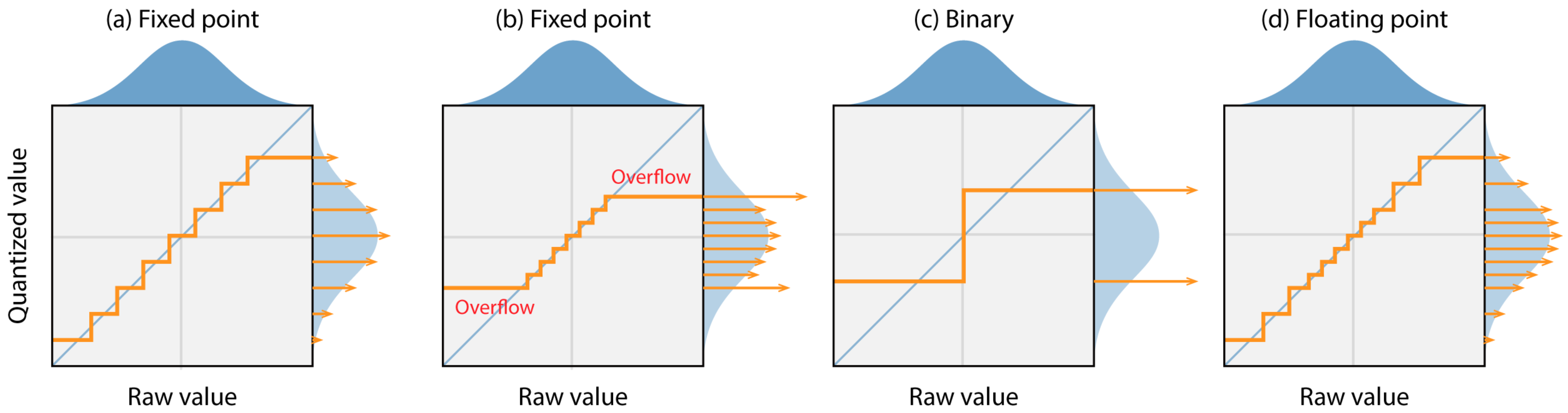
整形：线性采样



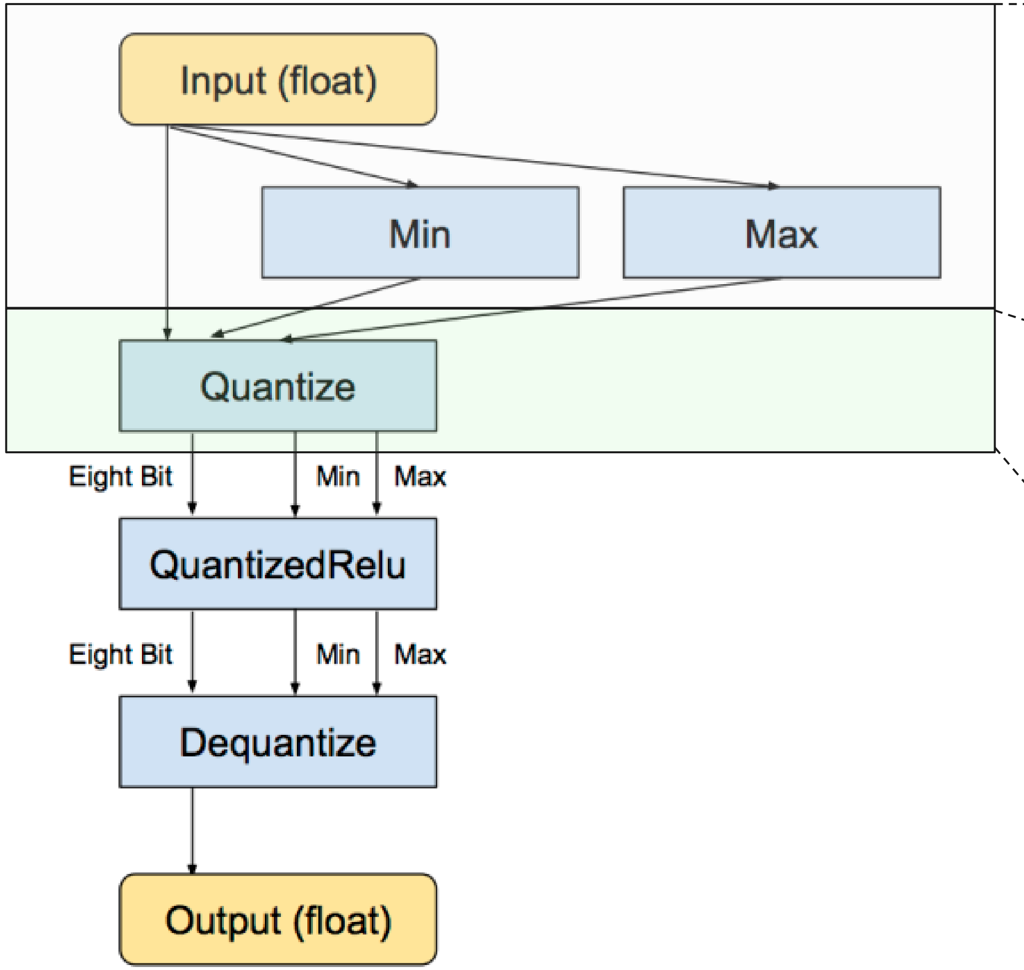
浮点：指数分段采样

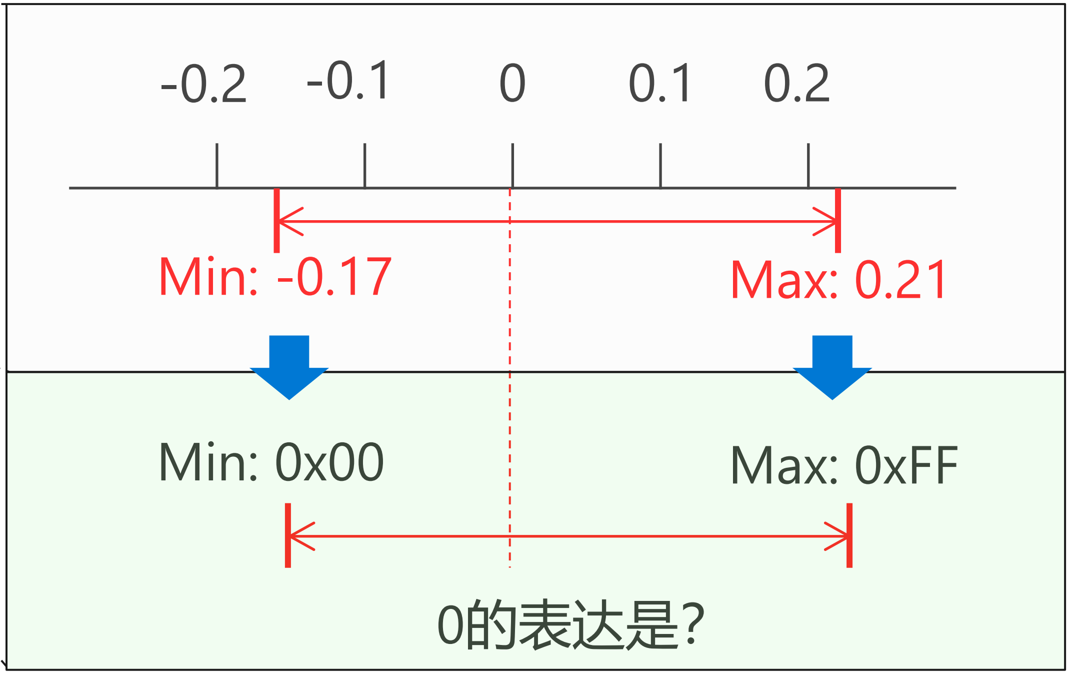


不同数据格式：不同采样精度



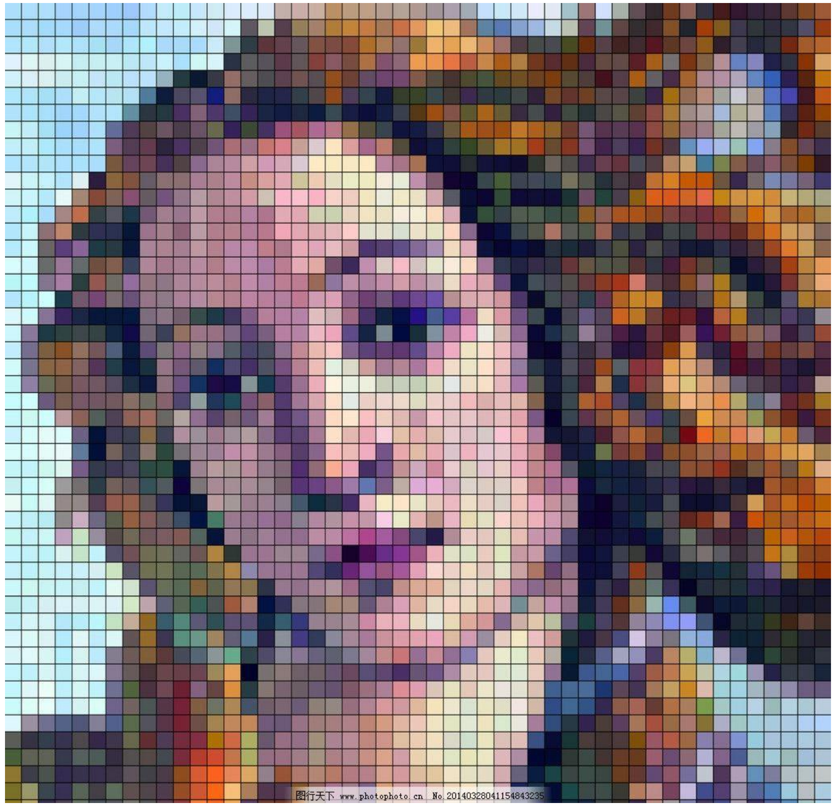
Tensorflow中的定点化方法



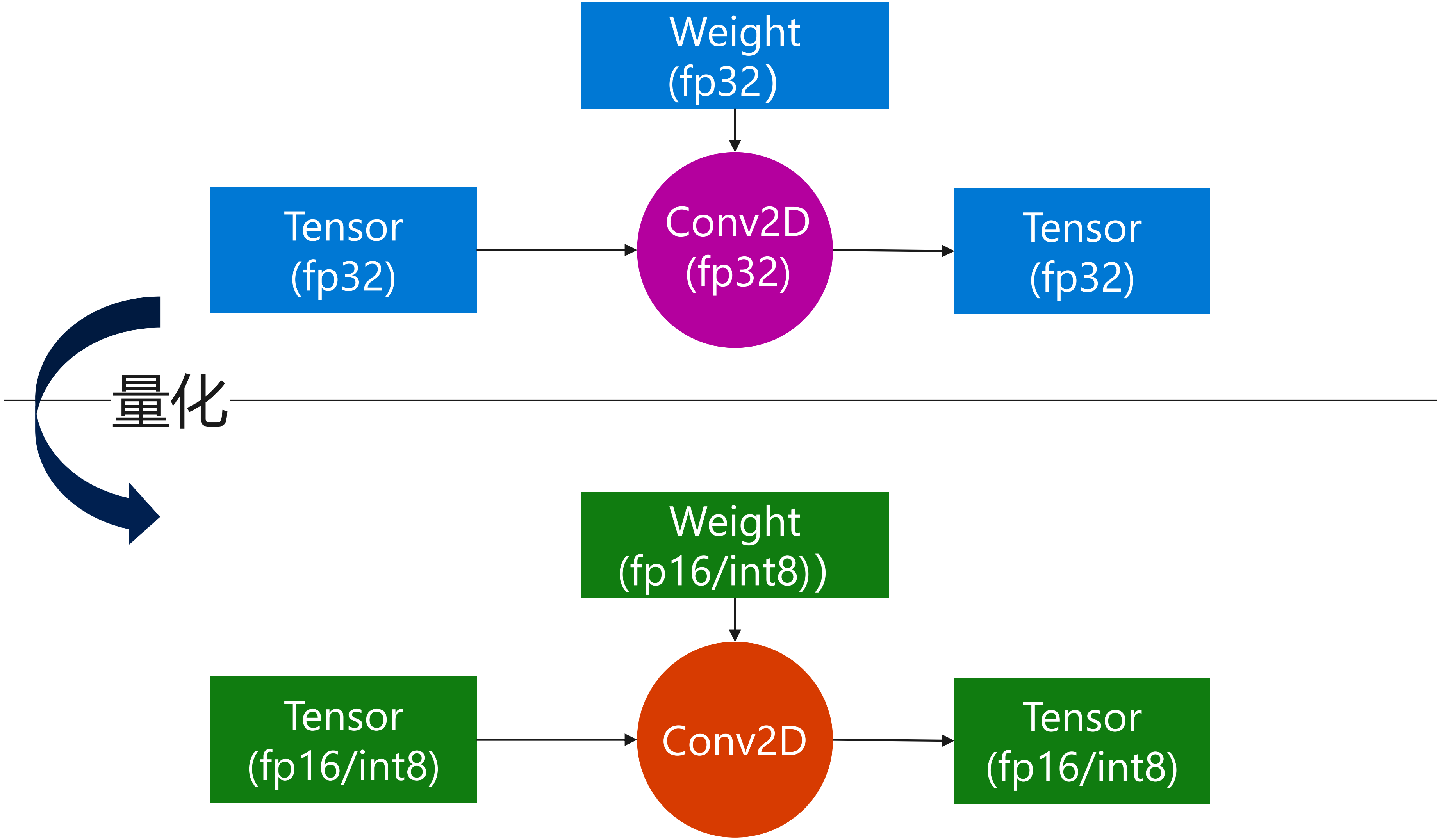


$$x = \frac{0-(-0.17)}{0.21-(-0.17)} \times 256$$
$$=144$$
$$= 0x72$$

量化（quantization）：低精度运算

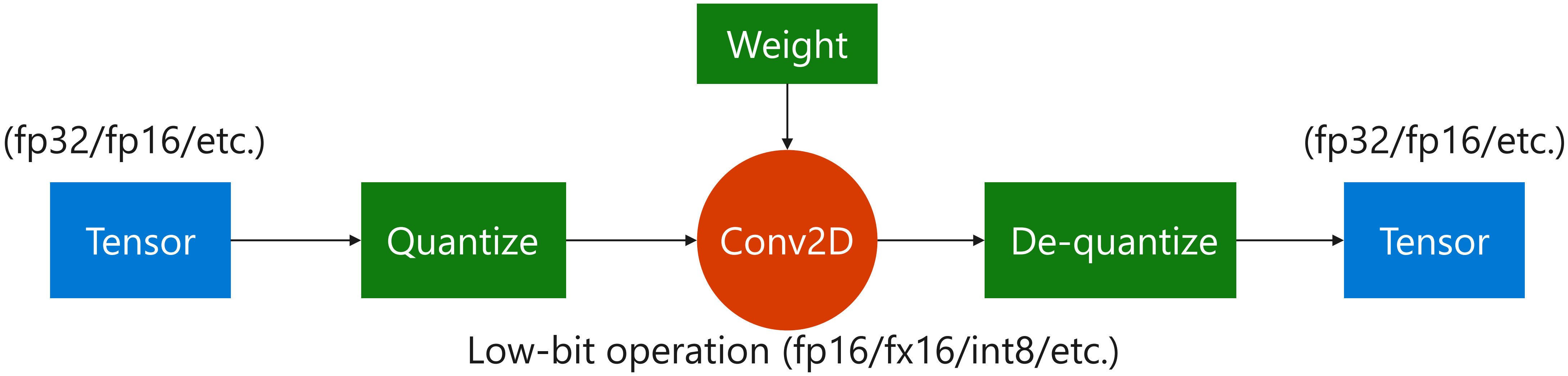


量化（quantization）：低精度运算



Low-bit operation (fp16/fx16/int8/etc.)

激活量化与权重量化


Weight
(fp32/fp16/etc.)
Tensor
Quantize
Conv2D
De-quantize
(fp32/fp16/etc.)
Tensor
Low-bit operation (fp16/fx16/int8/etc.)

研究课题 6：量化位宽与精度格式

三值网络

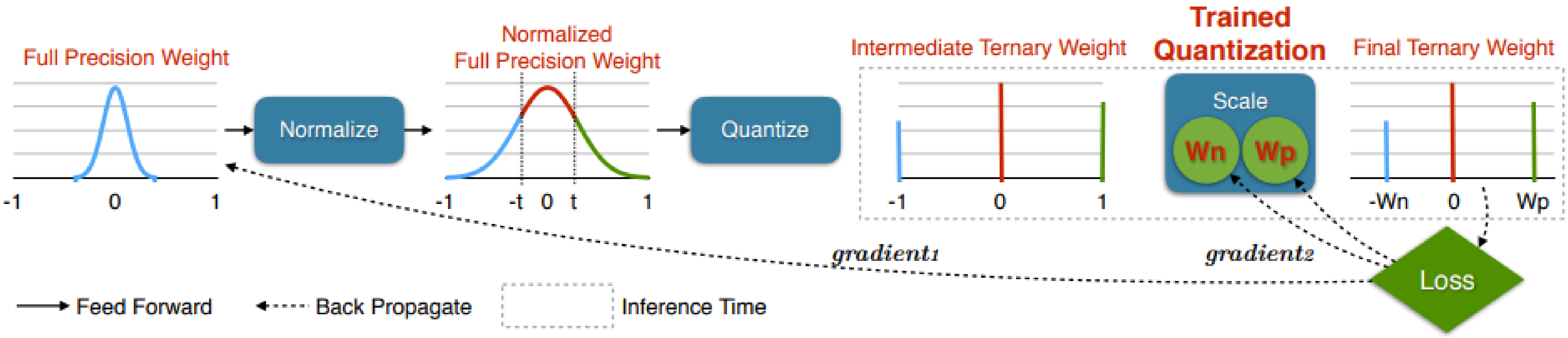
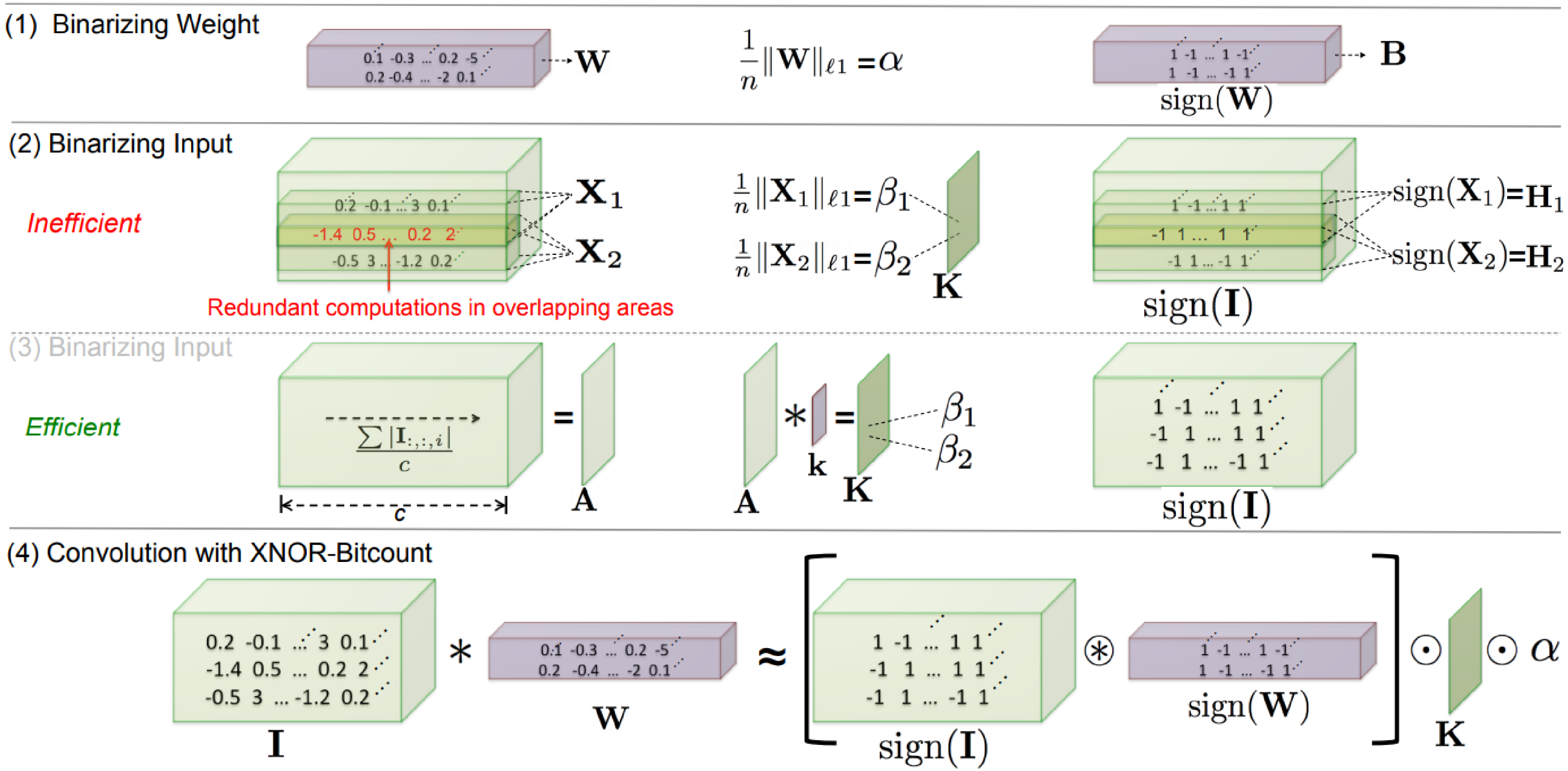


Figure 1: Overview of the trained ternary quantization procedure.

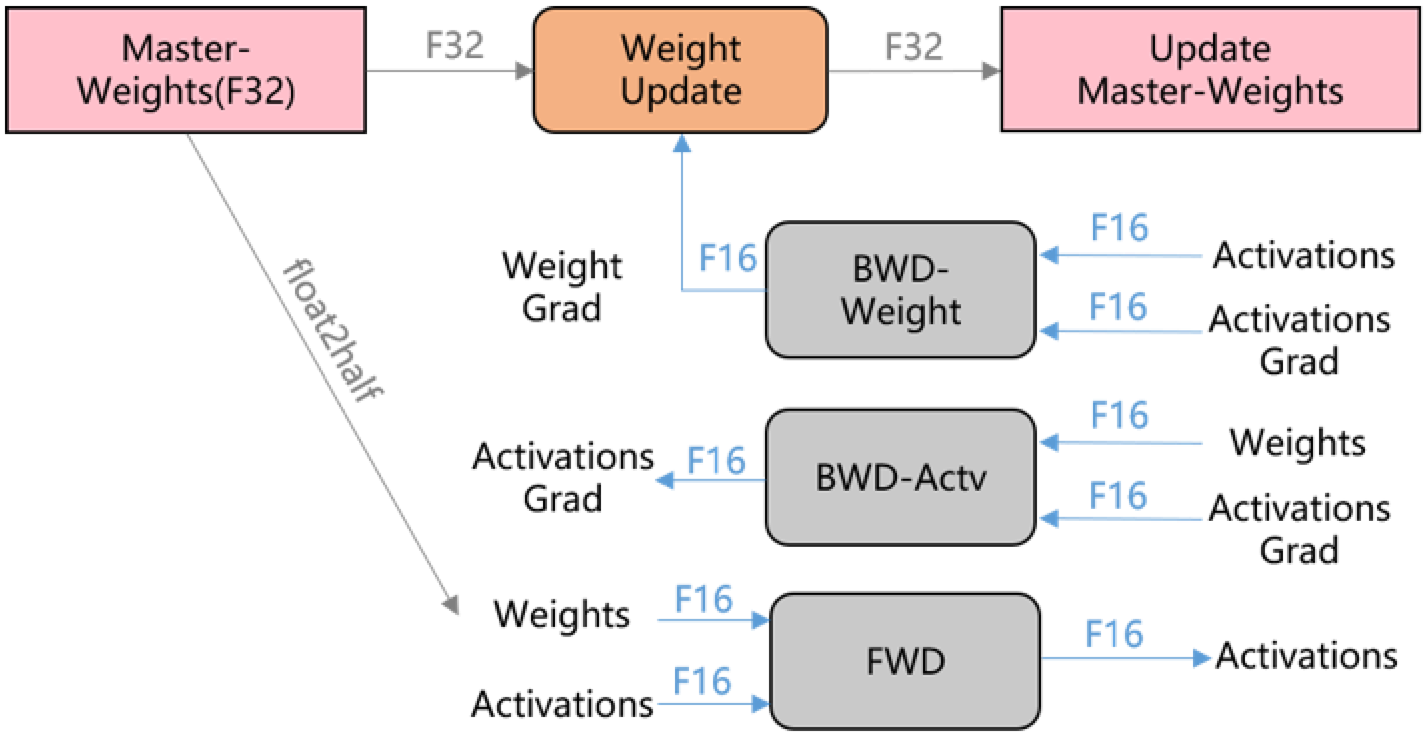
quantized ternary weights: $$w_l^t = \begin{cases} W_l^p : \tilde{w}_l > \Delta_l \\ 0 : |\tilde{w}_l| \leq \Delta_l \\ -W_l^n : \tilde{w}_l < -\Delta_l \end{cases}$$

二值网络



(1) Binarizing Weight
0.1 -0.3 ... 0.2 -5
0.2 -0.4 ... -2 0.1
W
$\frac{1}{n}\|\mathbf{W}\|_{\ell 1} = \alpha$
1 -1 ... 1 -1
1 -1 ... -1 1
B
sign(W)
(2) Binarizing Input
Inefficient
0.2 -0.1 ... 3 0.1
-1.4 0.5 ... 0.2 2
-0.5 3 ... -1.2 0.2
$\mathbf{X}_1$
$\mathbf{X}_2$
$\frac{1}{n}\|\mathbf{X}_1\|_{\ell 1} = \beta_1$
$\frac{1}{n}\|\mathbf{X}_2\|_{\ell 1} = \beta_2$
K
1 -1 ... 1 1
-1 1 ... 1 1
-1 1 ... -1 1
$\text{sign}(\mathbf{X}_1) = \mathbf{H}_1$
$\text{sign}(\mathbf{X}_2) = \mathbf{H}_2$
sign(I)
Redundant computations in overlapping areas
(3) Binarizing Input
Efficient
$\frac{\sum |\mathbf{I}_{:,:,i}|}{c}$
c
=
A
A * k = K
β_1
β_2
1 -1 ... 1 1
-1 1 ... 1 1
-1 1 ... -1 1
sign(I)
(4) Convolution with XNOR-Bitcount
0.2 -0.1 ... 3 0.1
-1.4 0.5 ... 0.2 2
-0.5 3 ... -1.2 0.2
I
*
0.1 -0.3 ... 0.2 -5
0.2 -0.4 ... -2 0.1
W
≈
1 -1 ... 1 1
-1 1 ... 1 1
-1 1 ... -1 1
sign(I)
⊛
1 -1 ... 1 -1
1 -1 ... -1 1
sign(W)
⊙ K ⊙ α

研究课题 7：混合精度训练



思考题：

- Tensorflow中的 8bit 定点化数据如何做乘法运算的呢？

实数运算

向量 A：[-0.1, 0.1, 0.2] ✕ 向量 B：[-2, 1, 3] = 向量 C：[0.2, 0.1, 0.6]

Tensorflow 8-bit 运算

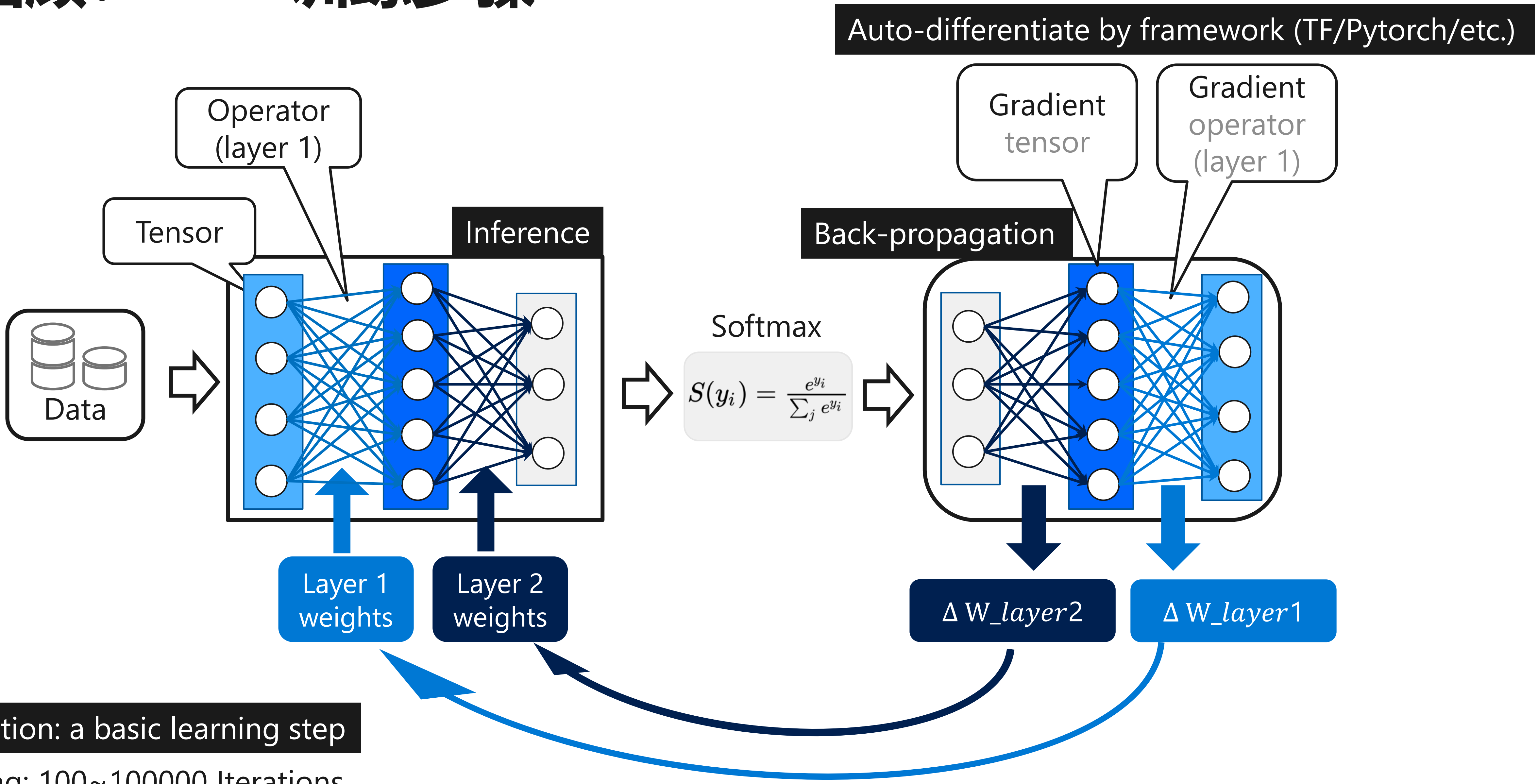
向量 A：[?, ?, ?] ✕ 向量 B：[?, ?, ?] = 向量 C：[?, ?, ?]

课程总结

- 模型压缩和稀疏化是提升DNN模型运行效率的重要手段
- 稀疏化：权重稀疏，激活稀疏，梯度稀疏
- 量化：低精度运算
- 还有许多问题等待进一步的研究
  - 模型压缩和稀疏化的最优解尚待研究
  - 定制化硬件设计以充分发挥稀疏性和量化的优势

Backup

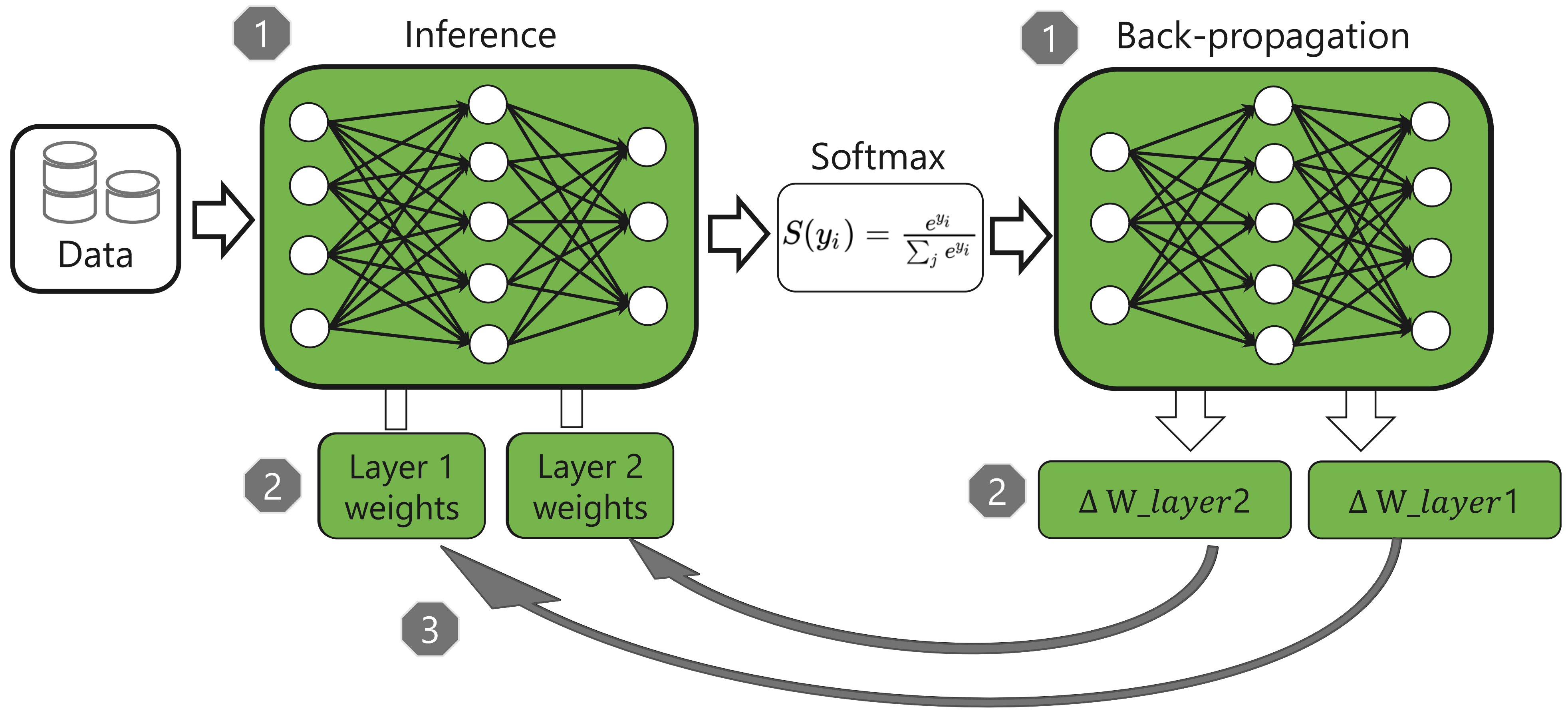
回顾：DNN训练步骤



1 Iteration: a basic learning step

Training: 100~100000 Iterations

稀疏存在于神经网络训练的每一个阶段



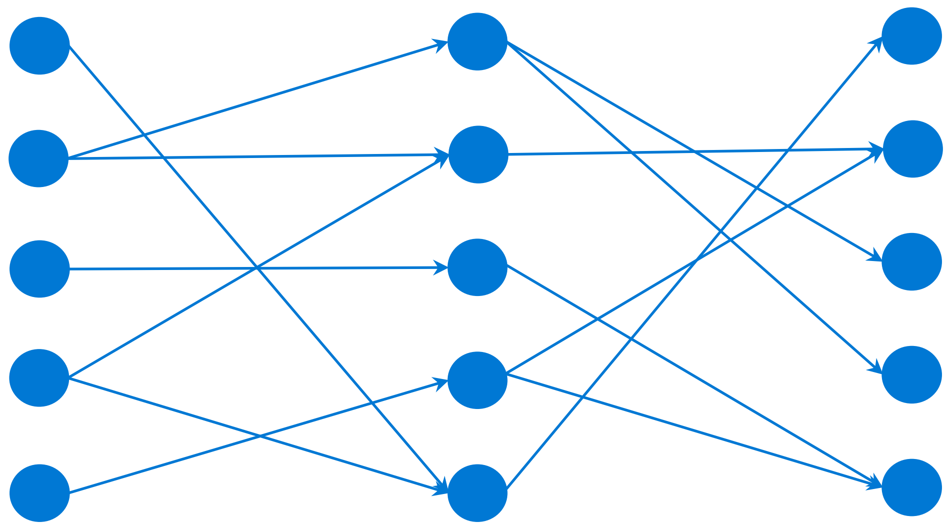
1 Sparse inference/backprop saves computation

2 Sparse matrix encoding saves storage

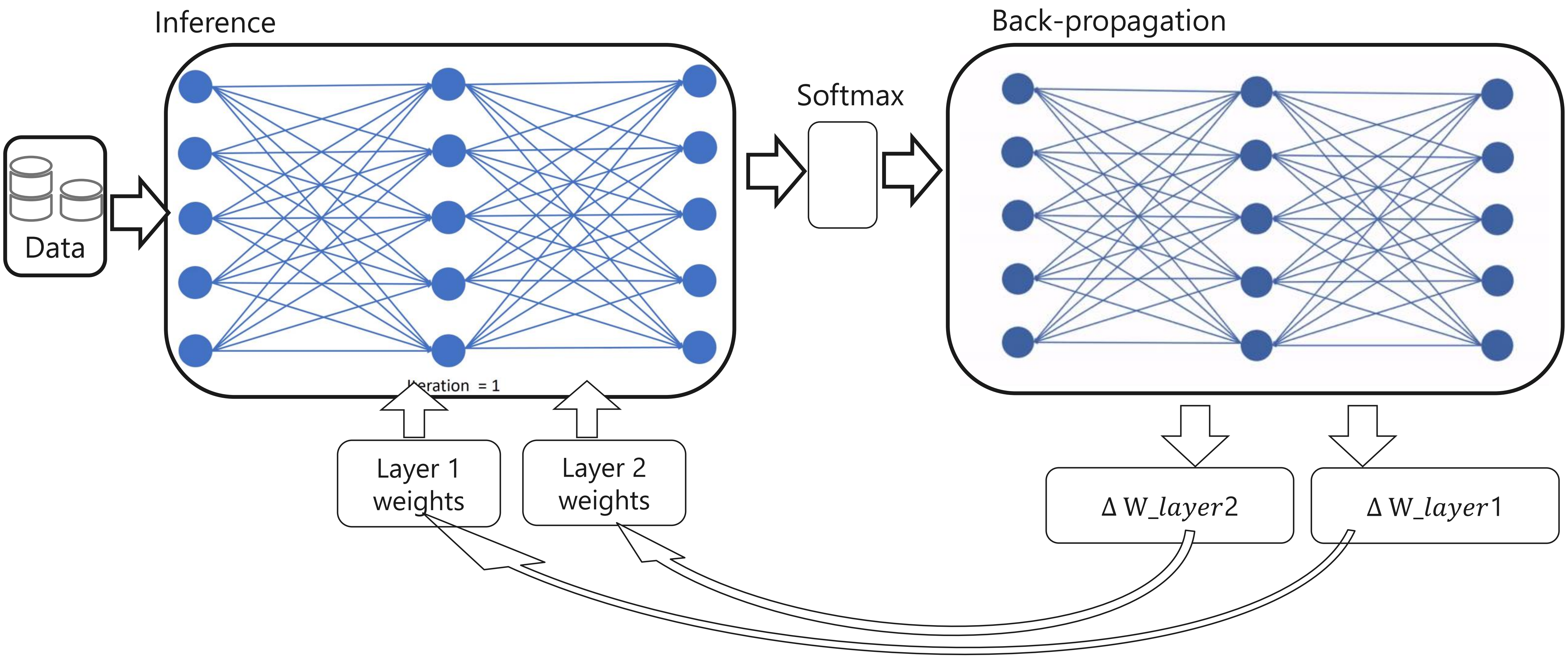
3 Sparse matrix encoding saves communication

前向计算：静态稀疏

neuron pruning is conducted offline



训练：动态稀疏


Inference
Back-propagation
Softmax
Data
Iteration = 1
Layer 1 weights
Layer 2 weights
Δ W_layer2
Δ W_layer1