AI 芯片 – AI 计算体系

深度学习计算模式

ZOMI

Talk Overview

1. AI 计算体系

- 深度学习计算模式
- 计算体系与矩阵运算

2. AI 芯片基础

- 通用处理器 CPU
- 从数据看 CPU 计算
- 通用图形处理器 GPU
- AI专用处理器 NPU/TPU
- 计算体系架构的黄金10年

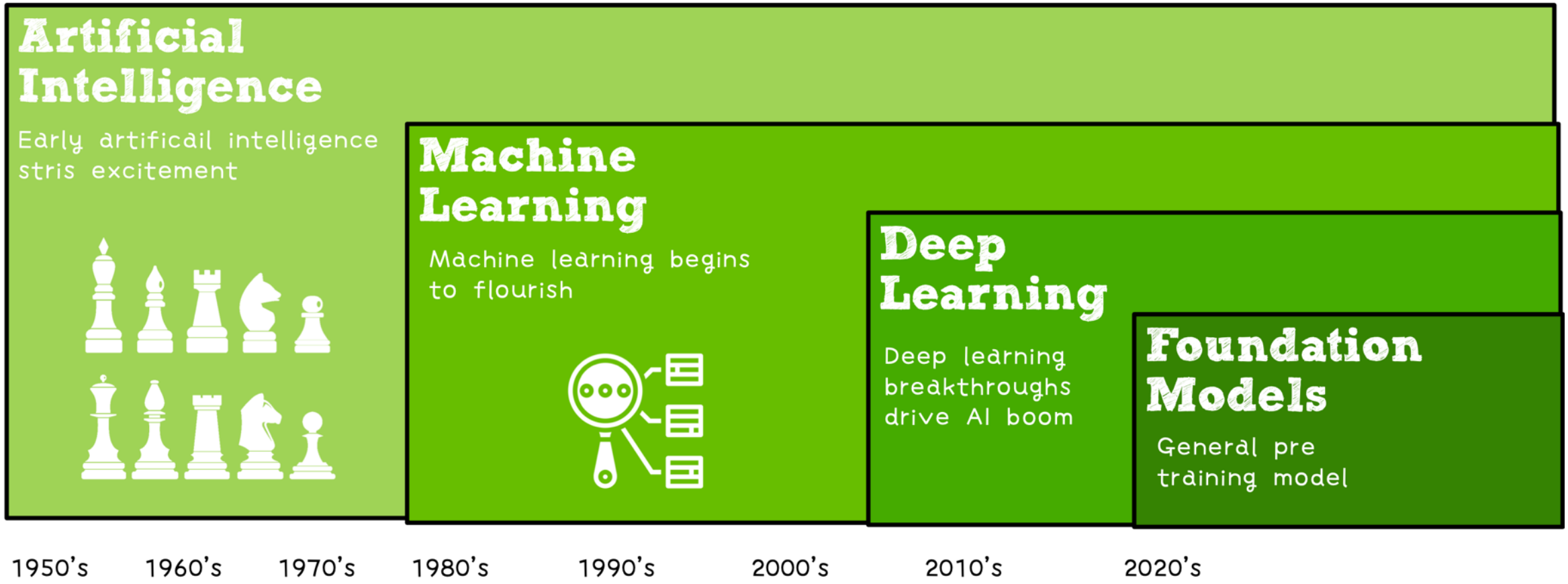
Talk Overview

I. 深度学习计算模式

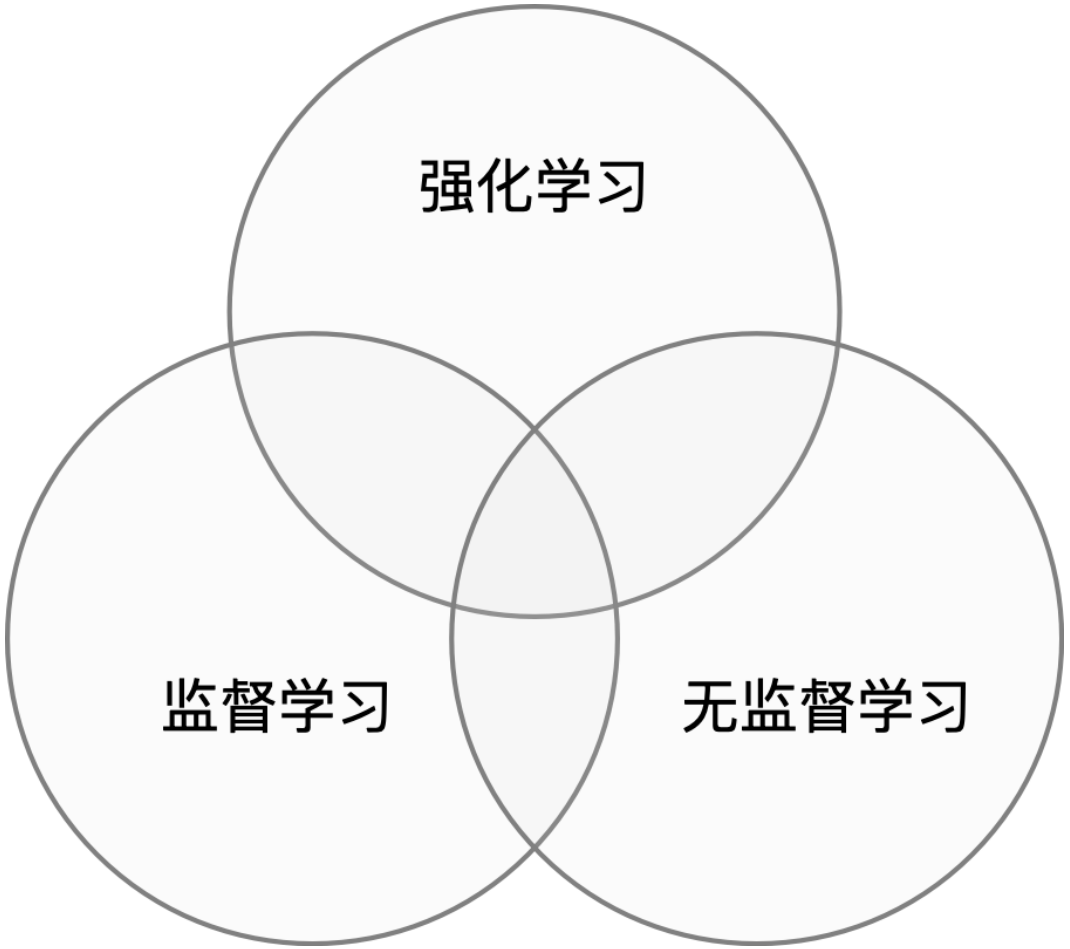
- The History – AI 的发展和范式
- Models Architecture – 经典模型结构
- Quantization and Pruning – 模型量化与剪枝
- Efficient Models – 轻量化网络模型
- Models Parallel – 大模型分布式并行

AI的发展和范式

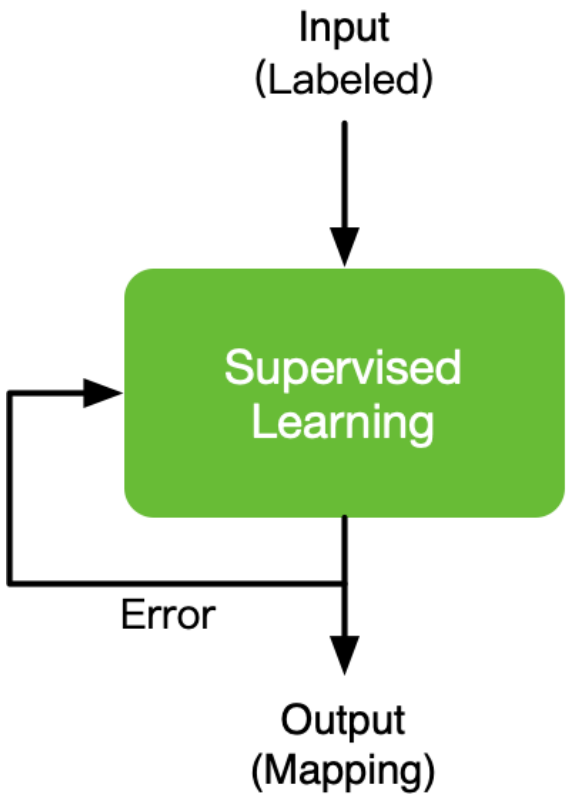
AI 总体发展路线

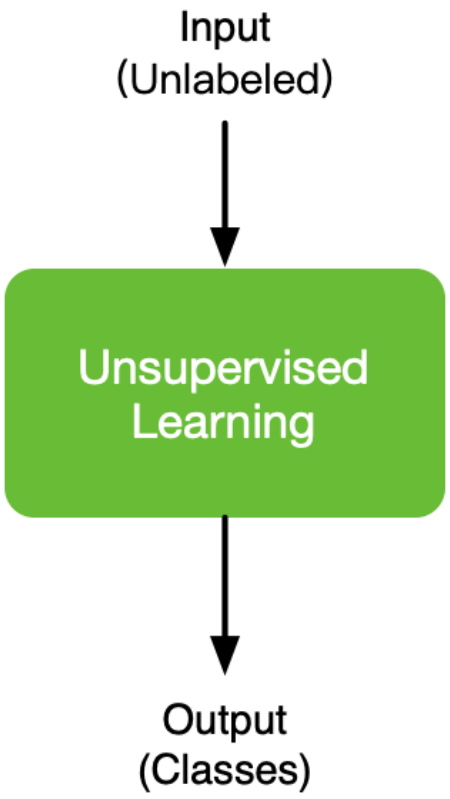


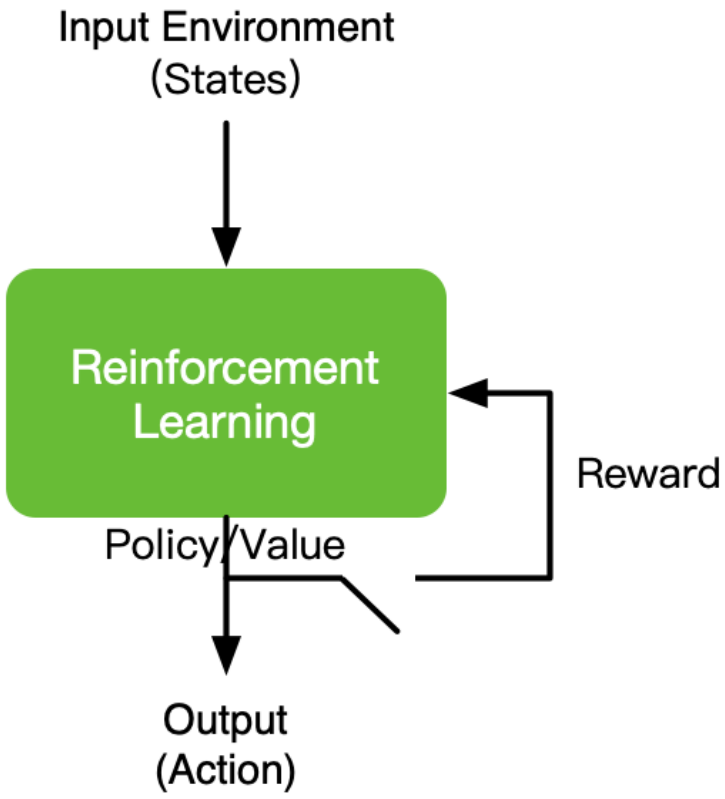
AI 三大范式



AI 三大范式流程

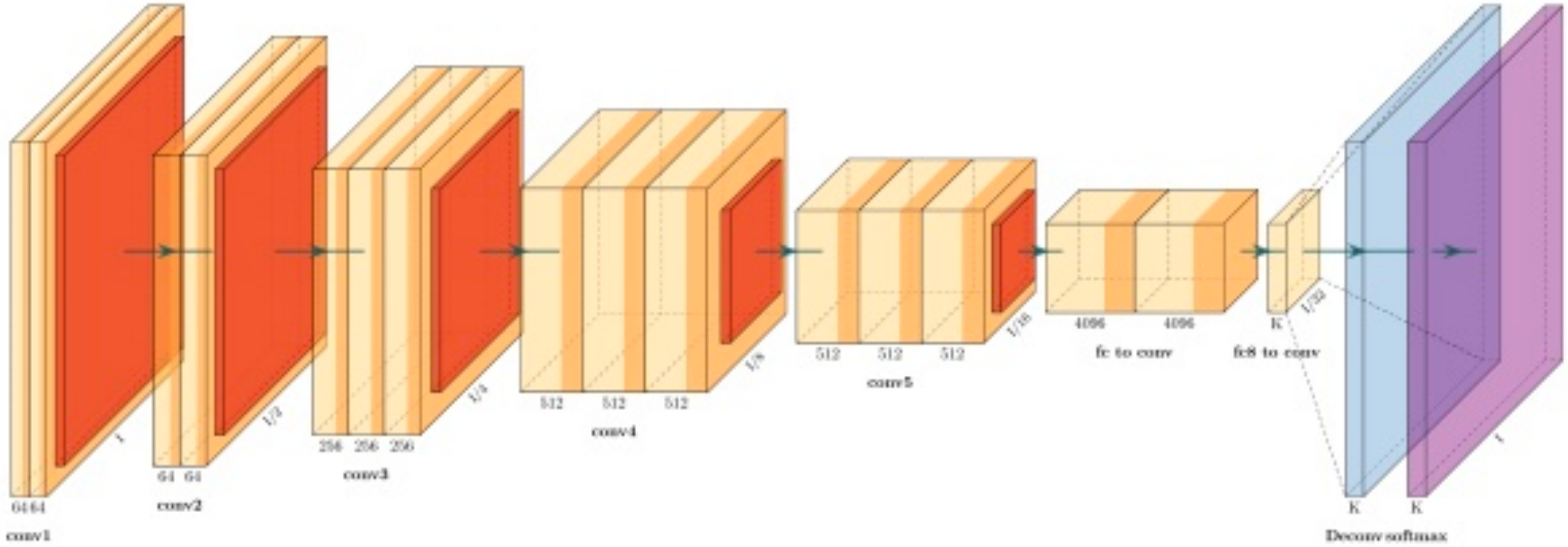






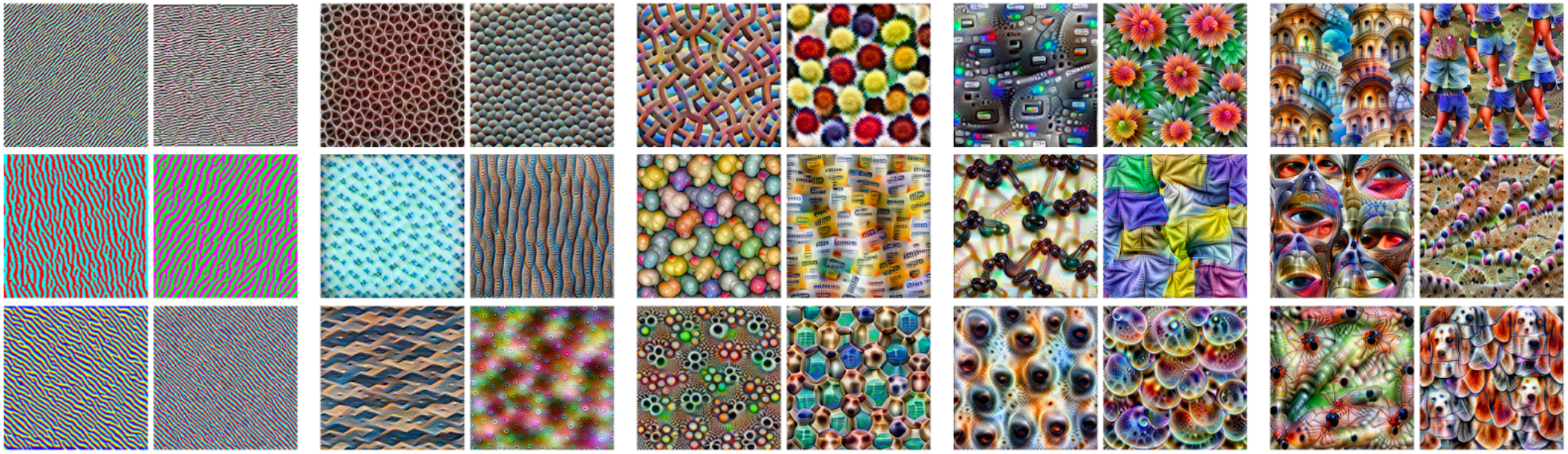
网络模型结构设计&演进

什么是神经网络？



https://ai.googleblog.com/2017/11/feature-visualization.html

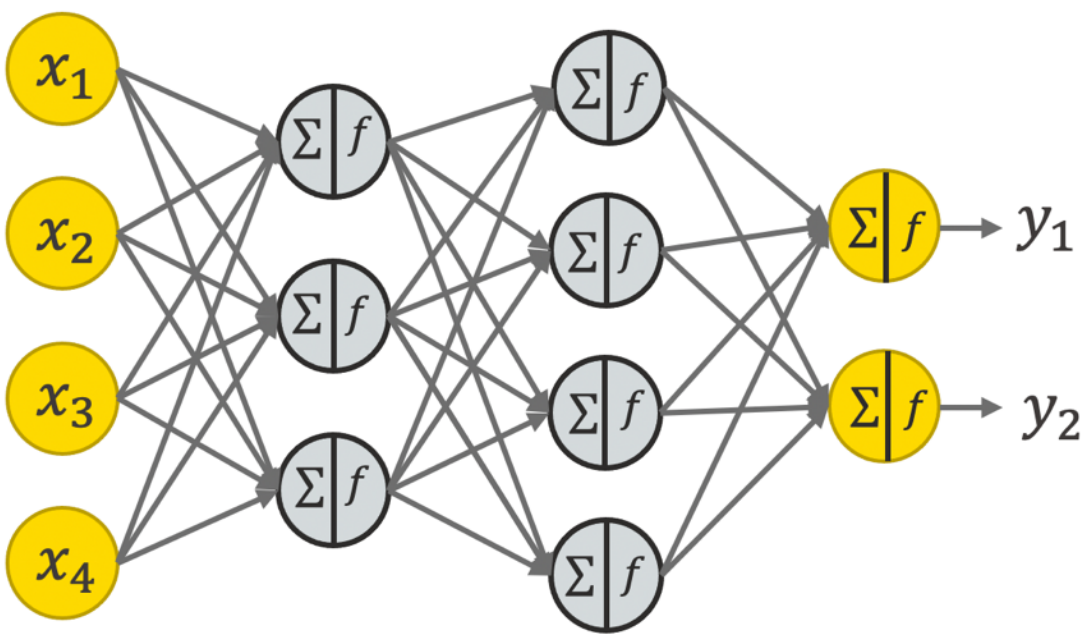
神经网络可视化

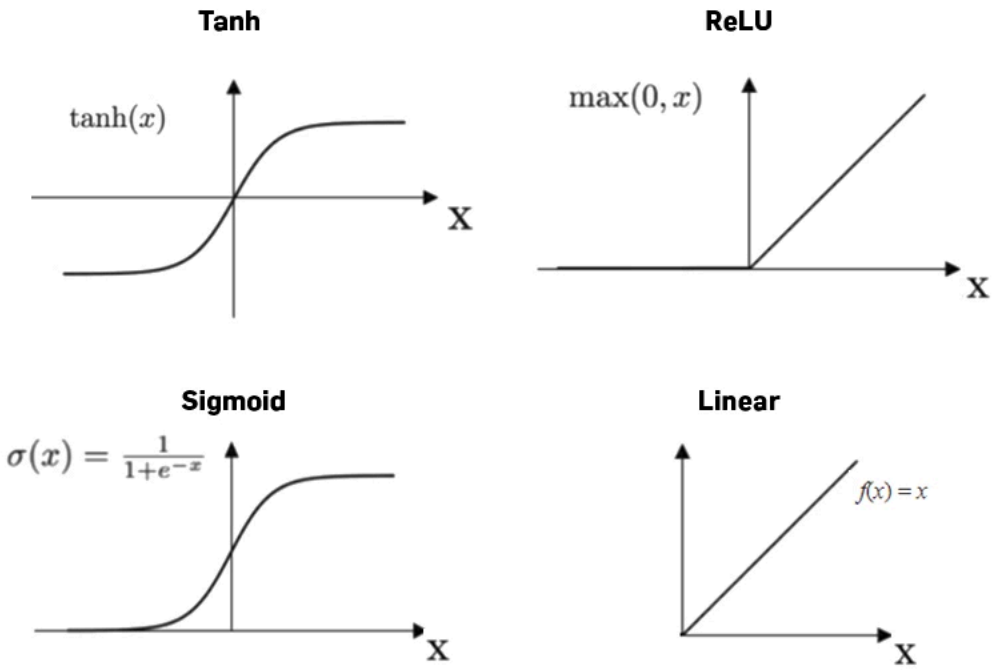


https://ai.googleblog.com/2017/11/feature-visualization.html

神经网络主要计算：权重求和

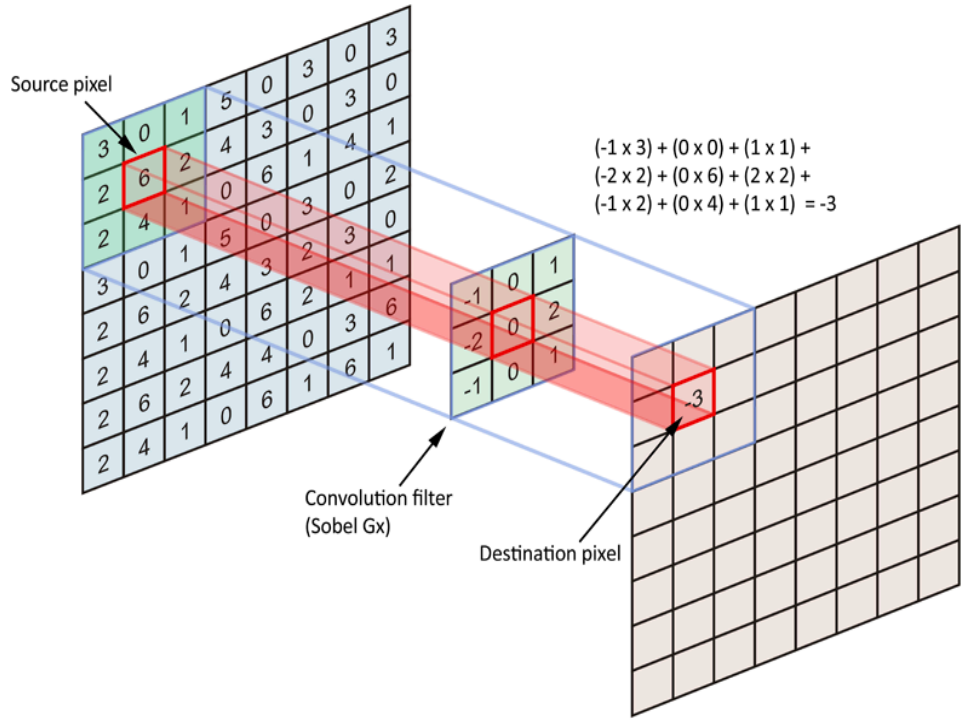
- 主要的计算模式：**multiply and accumulate (MAC)** > 90% computation





主流的网络模型结构 (I)

- **全连接 Fully Connected Layer**
  - Feed forward, fully connected
  - Multilayer Perceptron(MLP)
- **卷积层 Convolutional Layer**
  - Feed forward, sparsely-connected, weight shading
  - Convolutional Neural Network(CNN)
  - Typically used for images



主流的网络模型结构 (II)

- **循环网络 Recurrent Layer**
  - Feedback
  - Recurrent Neural Network(RNN/LSTM)
  - Typically used for sequential data(e.g., speech, language)
- **注意力机制 Attention Layer**
  - Attention(matrix multiply) + Feed forward, fully connected
  - Foundation Models
  - Transformer

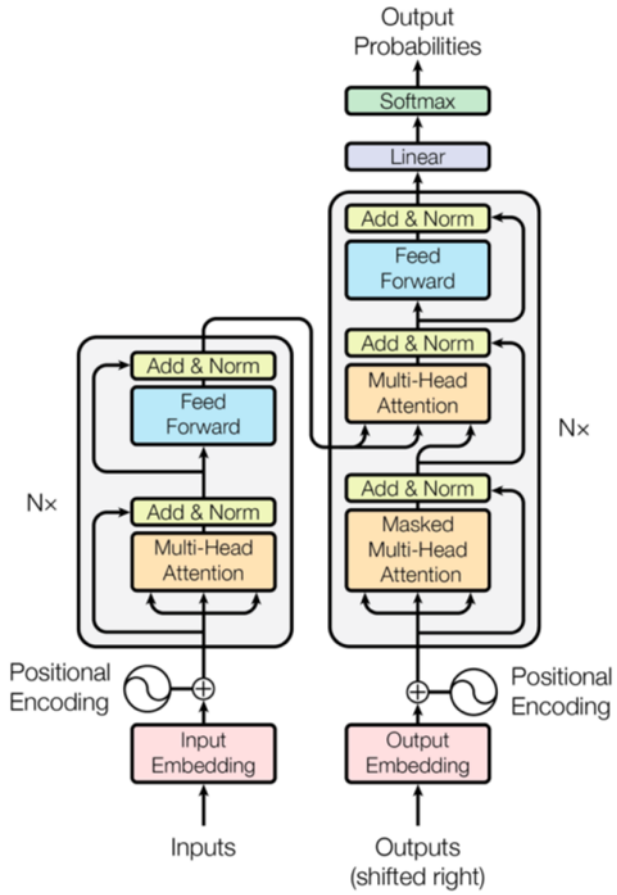
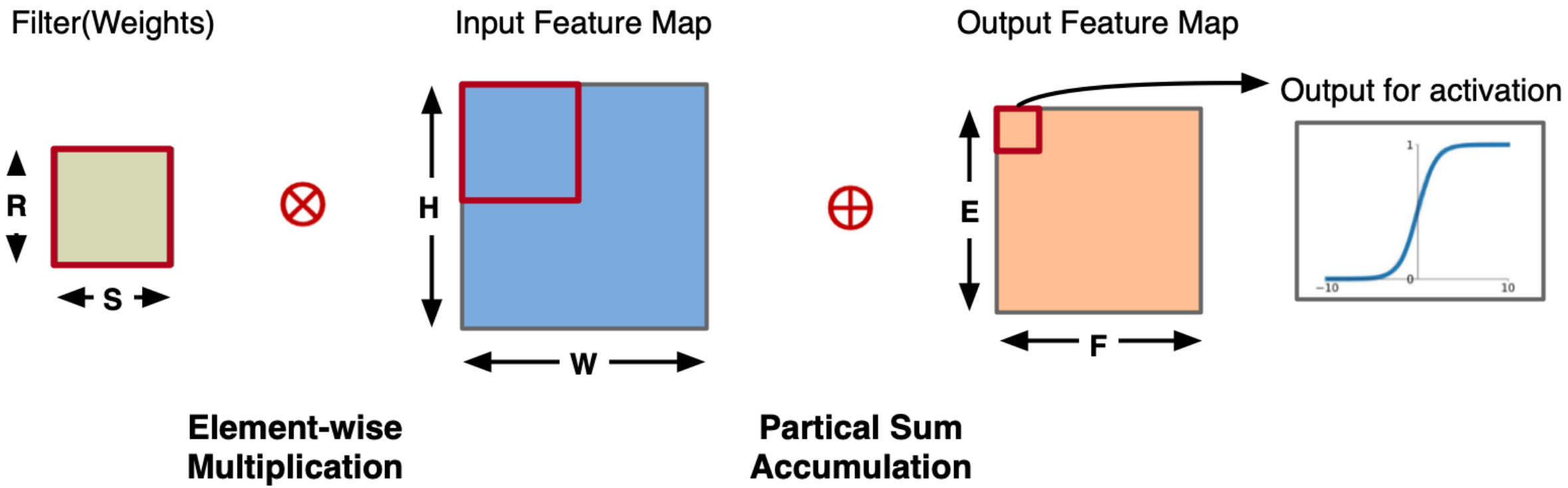


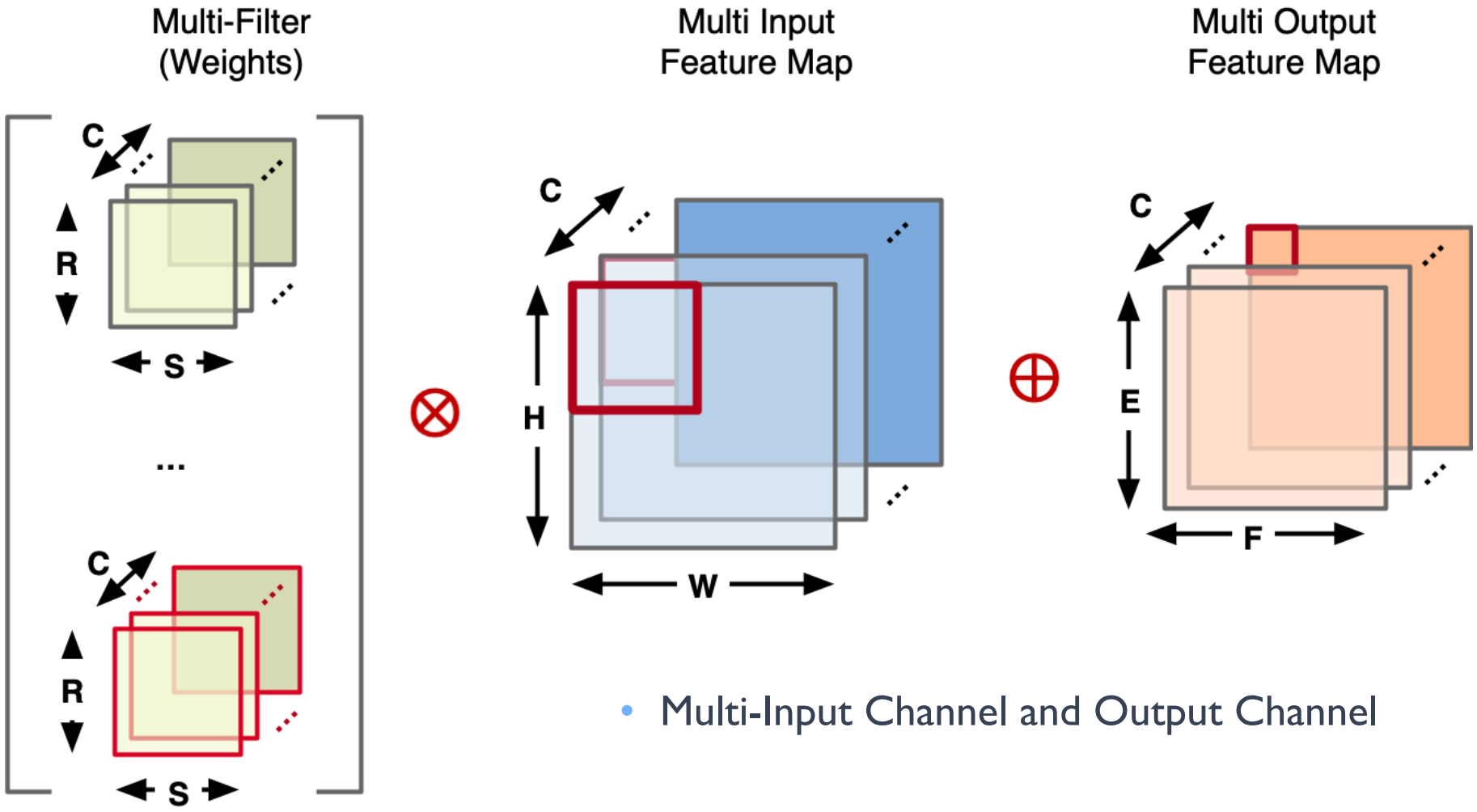
Figure 1: The Transformer - model architecture.

卷积计算 Convolution in CNN

- Key Operations is **multiply and accumulate (MAC)** > 90% computation

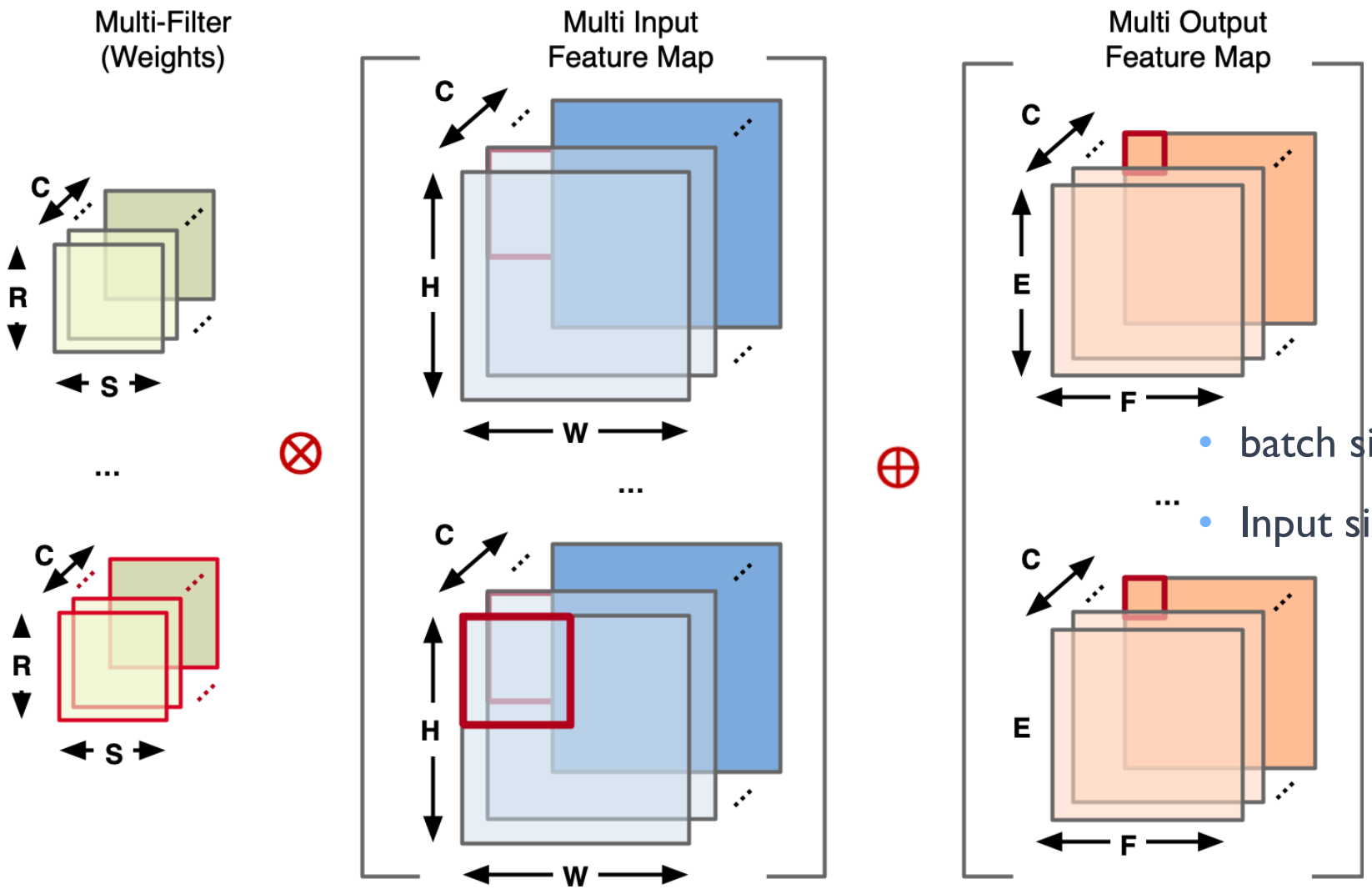


卷积计算 Convolution in CNN



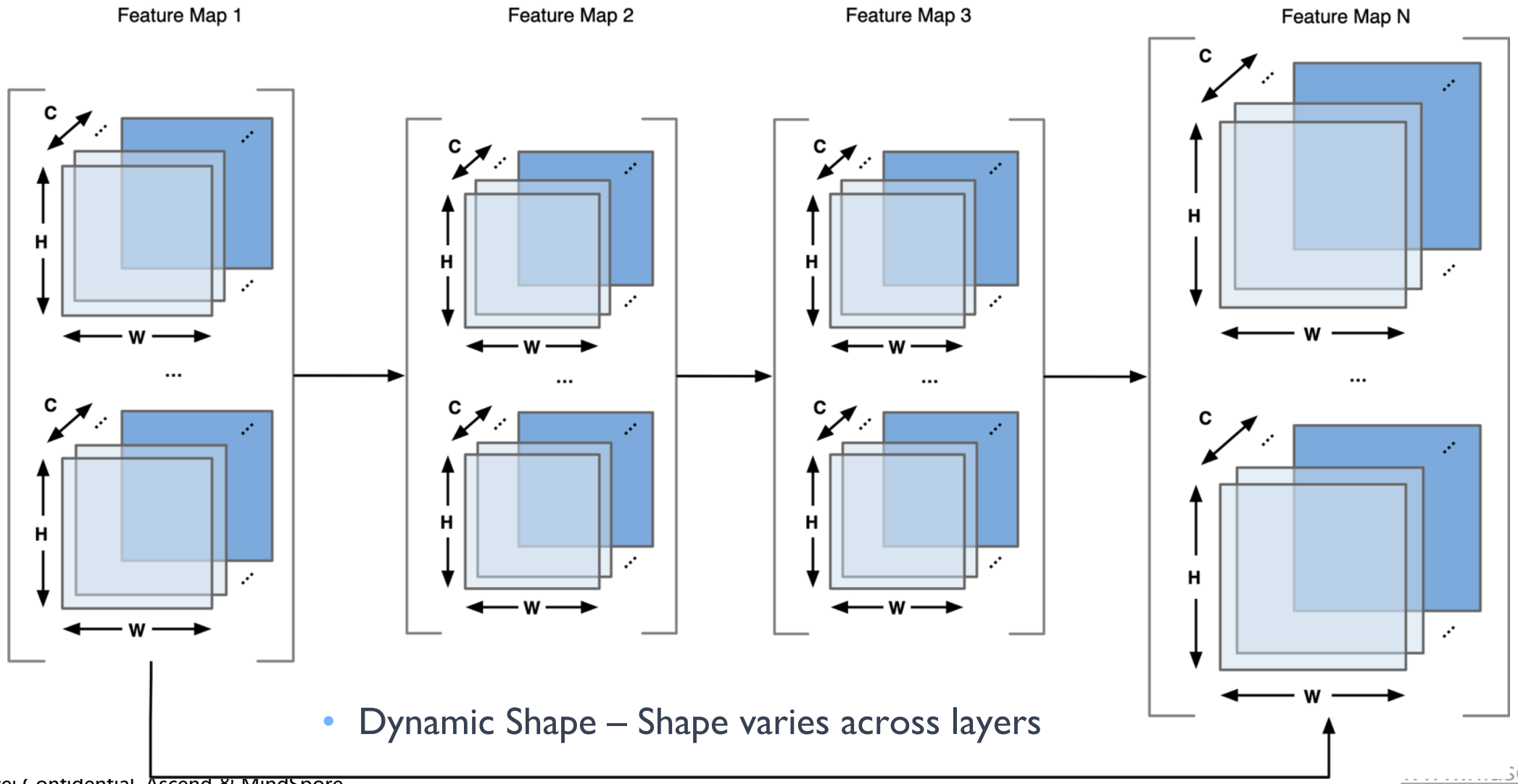
- Multi-Input Channel and Output Channel

卷积计算 Convolution in CNN



- batch size: 1-256(N)
- Input size: 15600x23600

卷积计算 Convolution in CNN



- Dynamic Shape – Shape varies across layers

经典网络模型

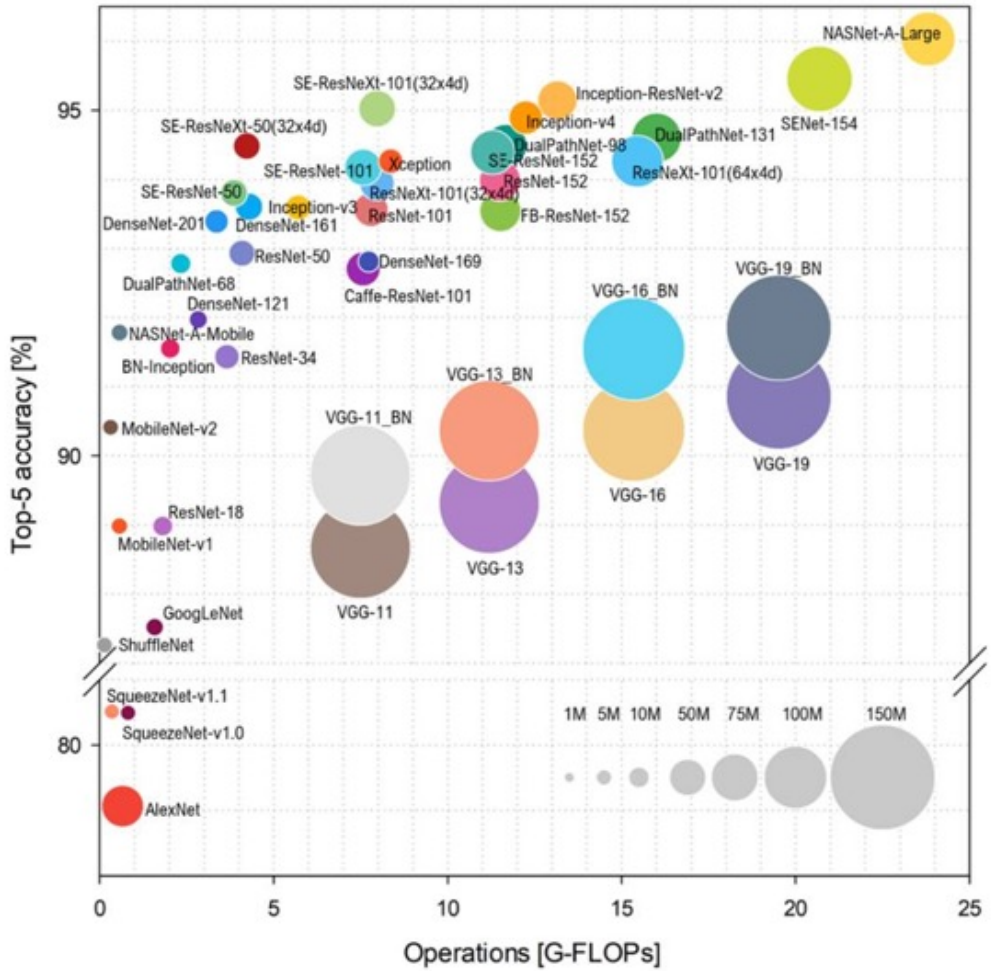
Models getting larger and deeper

Metrics	LeNet-5	AlexNet	VGG16	GoogleNet	ResNet50	EfficientNet-B4
Top-5 error(ImageNet)	n/a	16.4	7.4	6.7	5.3	3.7
Input Size	28x28	227x227	224x224	224x224	224x224	380x380
# Conv Layer	**2**	**5**	**16**	**21**	**49**	**96**
# of Weights	2.6k	2.3M	14.7M	6.0M	23.5M	14M
# of MACs	283k	666M	15.3G	1.43G	3.86G	4.4G
# FC Layers	**2**	**3**	**3**	**1**	**1**	**65**
# of Weights	58k	58.6M	124M	1M	2M	4.9M
# of MACs	58k	58.6M	124M	1M	2M	4.8M
Total Weights	**60k**	**61M**	**138M**	**7M**	**25.5M**	**19M**
Total MACs	**341k**	**724M**	**15.5G**	**1.43G**	**3.9G**	**4.4G**
Reference	Lecun,1998	Krizhevsky,2012	Simonyan,2015	Szegedy,2015	He,2016	Tan,2019

经典网络模型

Models getting larger and deeper

Metrics	LeNet-5	AlexNet	VGG16	GoogleNet	ResNet50	EfficientNet-B4
Top-5 error(Image Net)	n/a	16.4	7.4	6.7	5.3	3.7
Input Size	28x28	227x227	224x224	224x224	224x224	380x380
# Conv Layer	**2**	**5**	**16**	**21**	**49**	**96**
# of Weights	2.6k	2.3M	14.7M	6.0M	23.5M	14M
# of MACs	283k	666M	15.3G	1.43G	3.86G	4.4G
# FC Layers	**2**	**3**	**3**	**1**	**1**	**65**
# of Weights	58k	58.6M	124M	1M	2M	4.9M
# of MACs	58k	58.6M	124M	1M	2M	4.8M
Total Weights	**60k**	**61M**	**138M**	**7M**	**25.5M**	**19M**
Total MACs	**341k**	**724M**	**15.5G**	**1.43G**	**3.9G**	**4.4G**
Reference	Lecun,1998	Krizhevsky,2012	Simonyan,2015	Szegedy,2015	He,2016	Tan,2019



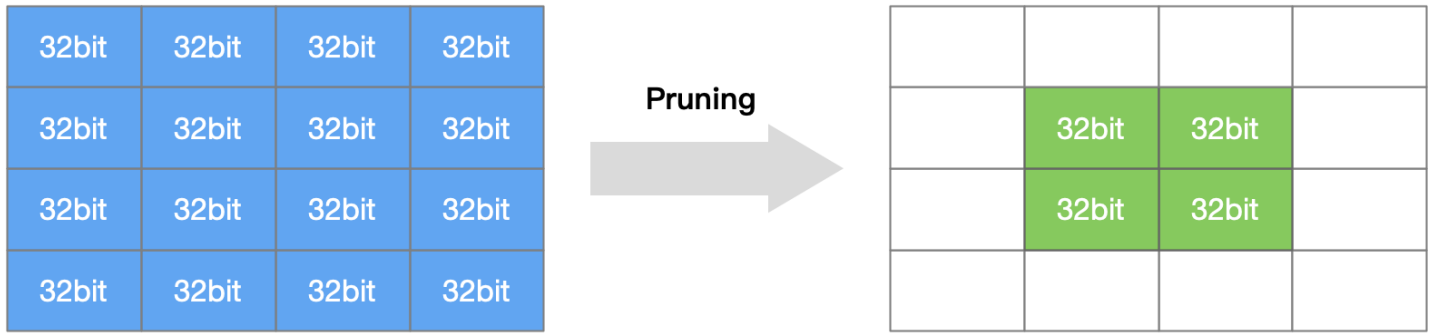
AI 计算模式思考 (I)

1. **需要支持神经网络模型的计算逻辑**
   - 权重数据共享，便于对神经元的权重值进行求和
   - 除了卷积/全连接计算，需要支持激活等Vector计算
2. **能够支持高维的张量存储与计算**
   - 内存 Mem 地址随机/自动索引
   - 大 Channel 和 大 Feature Map 高效加载
3. **支持常用神经网络模型结构**
   - Conv、MatMul、Transformer等高效矩阵乘
   - 快速应对新的 AI 算法与结构

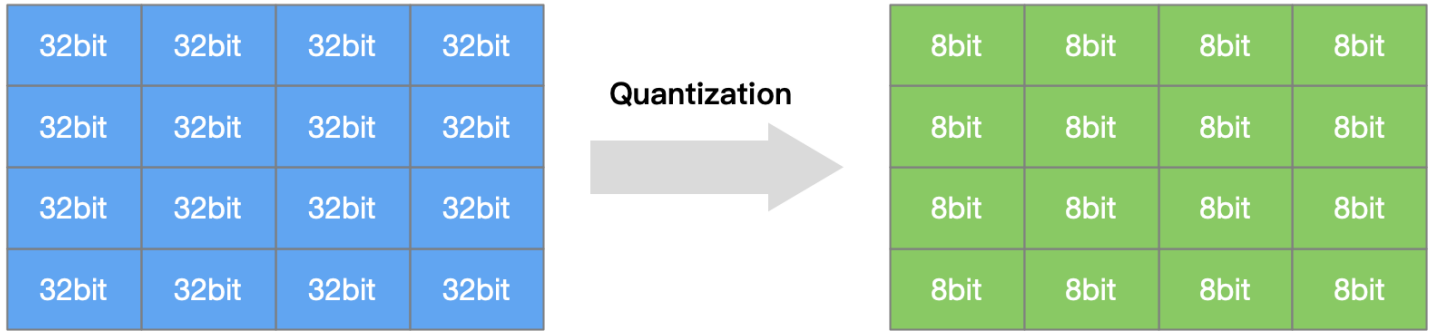
模型量化
网络剪枝

量化压缩 vs 网络剪枝

- 网络剪枝研究模型权重中的冗余， 并尝试删除/修剪冗余和非关键的权重。

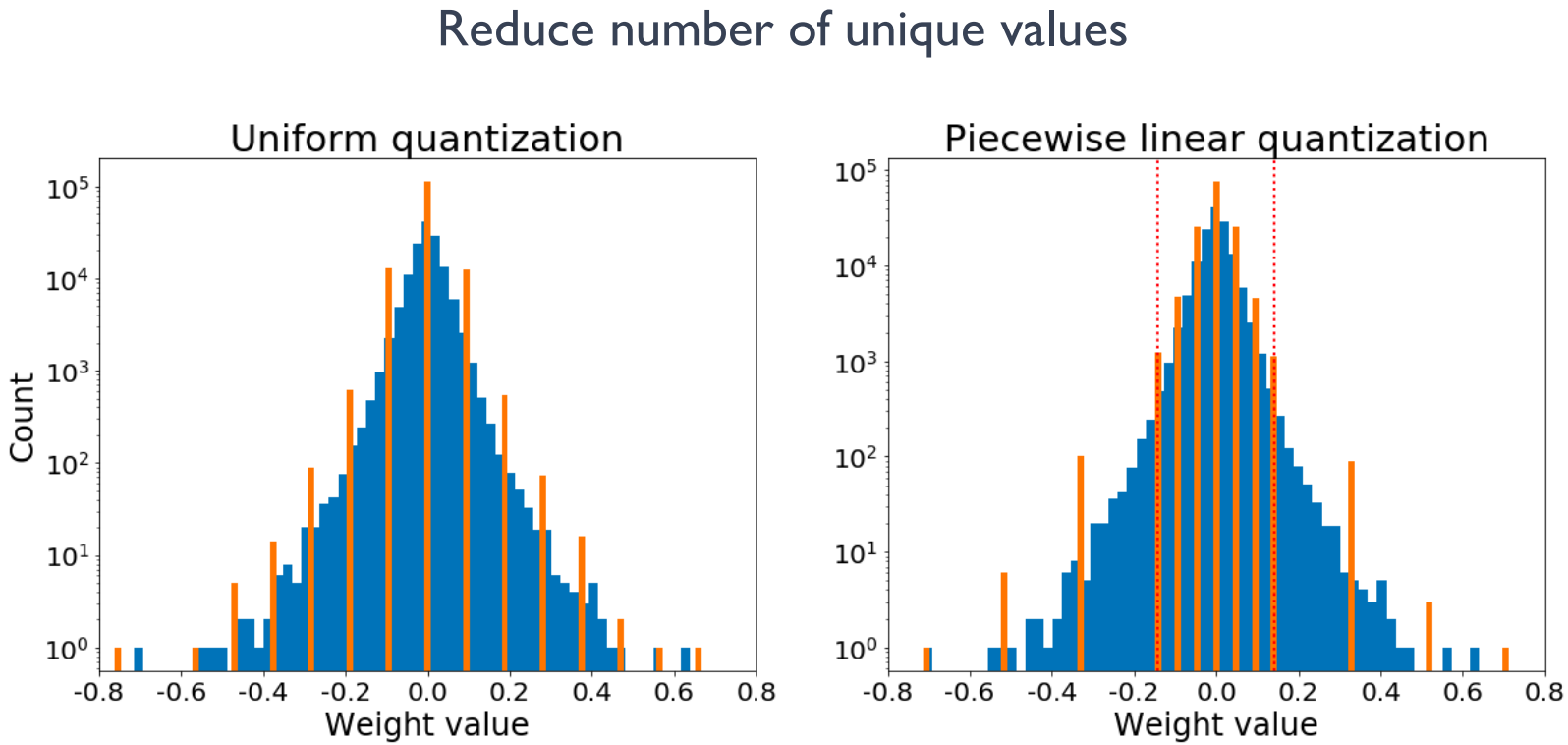


- 模型量化是指通过减少权重表示或激活所需的比特数来压缩模型。



低比特量化特征

1. 参数压缩；
2. 提升速度；
3. 降低内存；
4. 功耗降低；
5. 提升芯片面积；

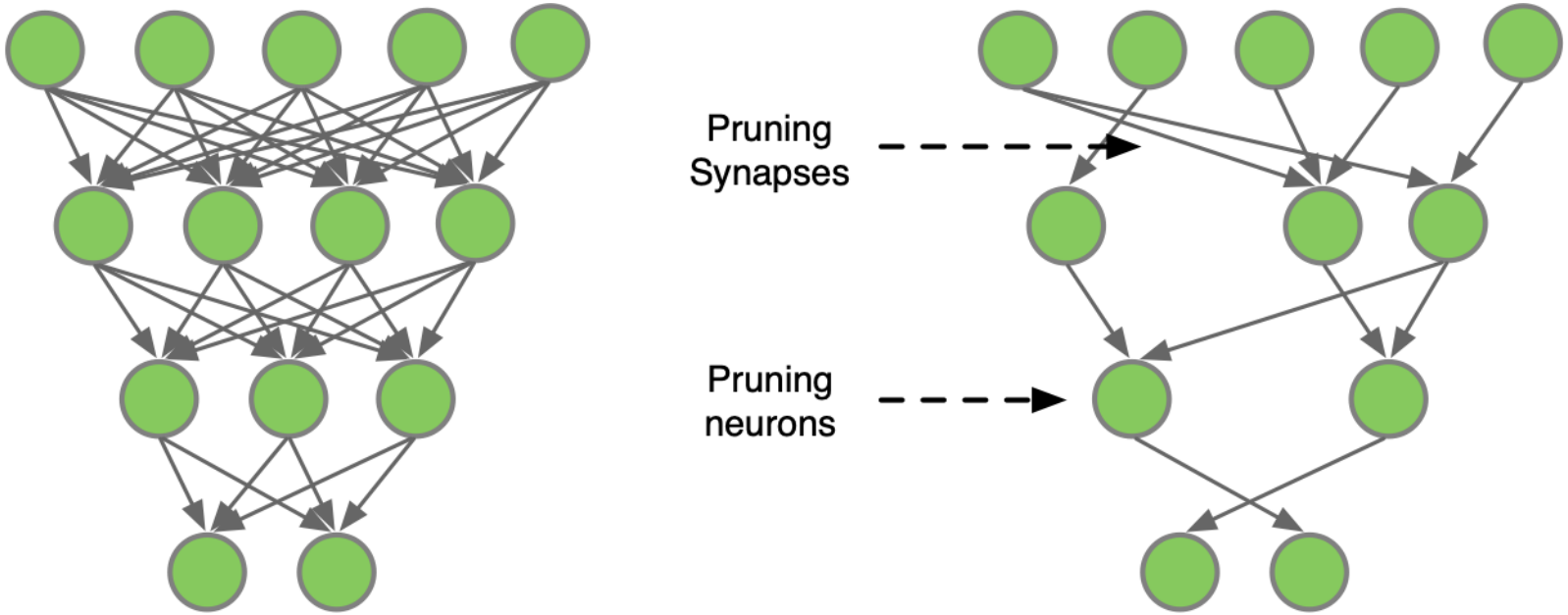


模型量化相关的研究热点

- **感知量化训练 Quantization Training**
  - 8-bit with stochastic rounding 混合bit量化
- **减少计算比特位 Reduce number of bits**
  - Binary Nets 二值化网络模型
- **非线性量化 Non-Linear Quantization**
  - Log-Net
- **减少权重计算 Reduce number of unique weights and activations**
  - ADD Nets 加法网络
  - XNOR-Net 异或网络模型

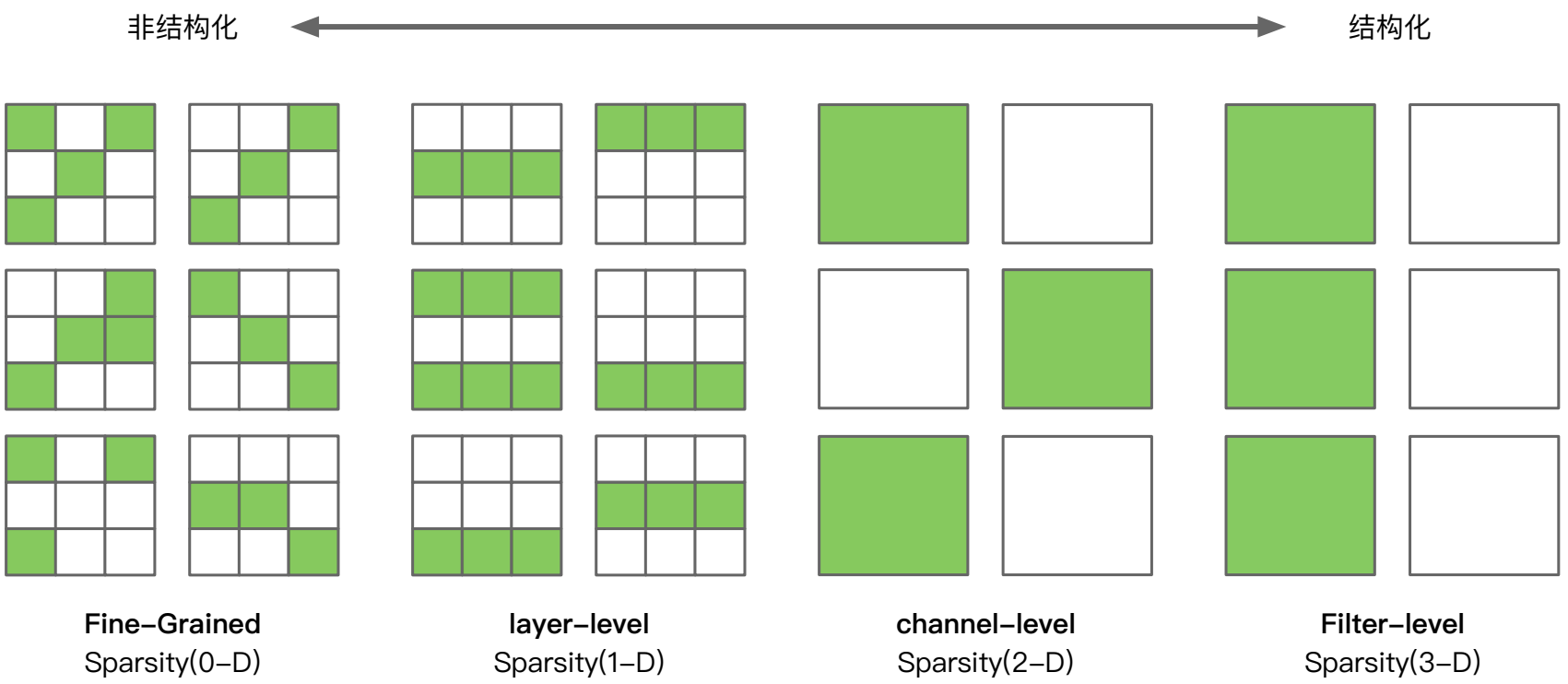
Pruning – Make weights Sparse

- **训练 Training**：训练过参数化模型，得到最佳网络性能，以此为基准；
- **剪枝 Pruning**：根据算法对模型剪枝，调整网络结构中通道或层数，得到剪枝后的网络结构；
- **微调 Finetune**：在原数据集上进行微调，用于重新弥补因为剪枝后的稀疏模型丢失的精度性能。



模型剪枝分类

- **Unstructured Pruning（非结构化剪枝）**
  - 随机对独立的权重或者神经元链接进行剪枝
- **Structured Pruning（结构化剪枝）**
  - 对 filter / channel / layer 进行剪枝



To prune, or not to prune: exploring the efficacy of pruning for model compression

1. 在内存占用相同情况下，大稀疏模型比小密集模型实现了更高的精度。
2. 经过剪枝之后稀疏模型要优于，同体积非稀疏模型。
3. 资源有限的情况下，剪枝是比较有效的模型压缩策略。
4. 优化点还可以往硬件稀疏矩阵储存方向发展。

Table 4: NMT sparse vs dense results

# units	Sparsity	NNZ params	EN-DE BLEU score	DE-EN BLEU score
256	0%	34M	23.52	26.52
512	0%	81M	26.05	28.88
768	0%	140M	26.63	29.41
1024	0%	211M	26.77	29.47
	80%	44M	26.86	29.50
	85%	33M	26.52	29.24
	90%	23M	26.19	28.81

Table 1: Model size and accuracy tradeoff for sparse-InceptionV3

Sparsity	NNZ params	Top-1 acc.	Top-5 acc.
0%	27.1M	78.1%	94.3%
50%	13.6M	78.0%	94.2%
75%	6.8M	76.1%	93.2%
87.5%	3.3M	74.6%	92.5%

AI 计算模式思考 (II)

1. **提供不同的 bit 位数**
   - 对于低比特量化相关的研究落地提供 int8/int4 甚至更低的精度
   - 在 M-bits and E-bits 之间权衡 Tradeoff（如 TF32/BF16）
2. **利用硬件提供稀疏计算**
   - 硬件上减少 0 值的重复计算
   - 减少网络模型对内存的需求，稀疏化网络模型结构

引用

1. https://www.knime.com/blog/a-friendly-introduction-to-deep-neural-networks
2. https://machine-learning.paperspace.com/wiki/activation-function
3. https://developer.nvidia.com/blog/accelerating-ai-training-with-tf32-tensor-cores/