计算机组成与嵌入式系统

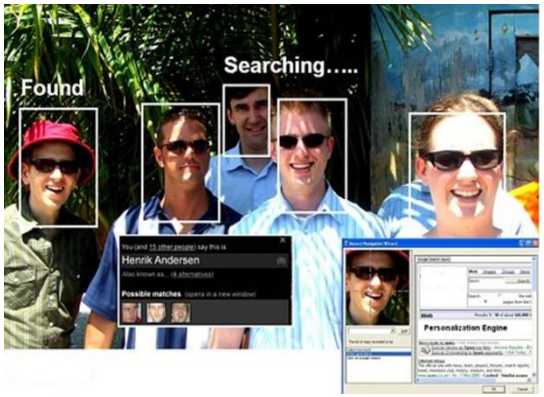
第十章 AI处理器技术

钟胜 颜露新
人工智能与自动化学院 飞行器导航制导系
2022年春季

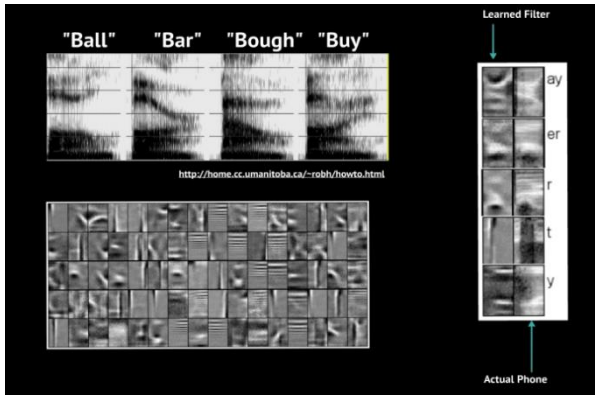
本章目标

- 了解CNN网络结构
- 熟悉CNN网络计算特性
- 熟悉CNN并行加速计算技术
- 熟悉典型AI处理器

10.1 人工智能不断飞速发展



lfw人脸测试准确度99%
(成人仅97%)



语音速记战胜
人类专业速记员

AlphaGo战胜李世石

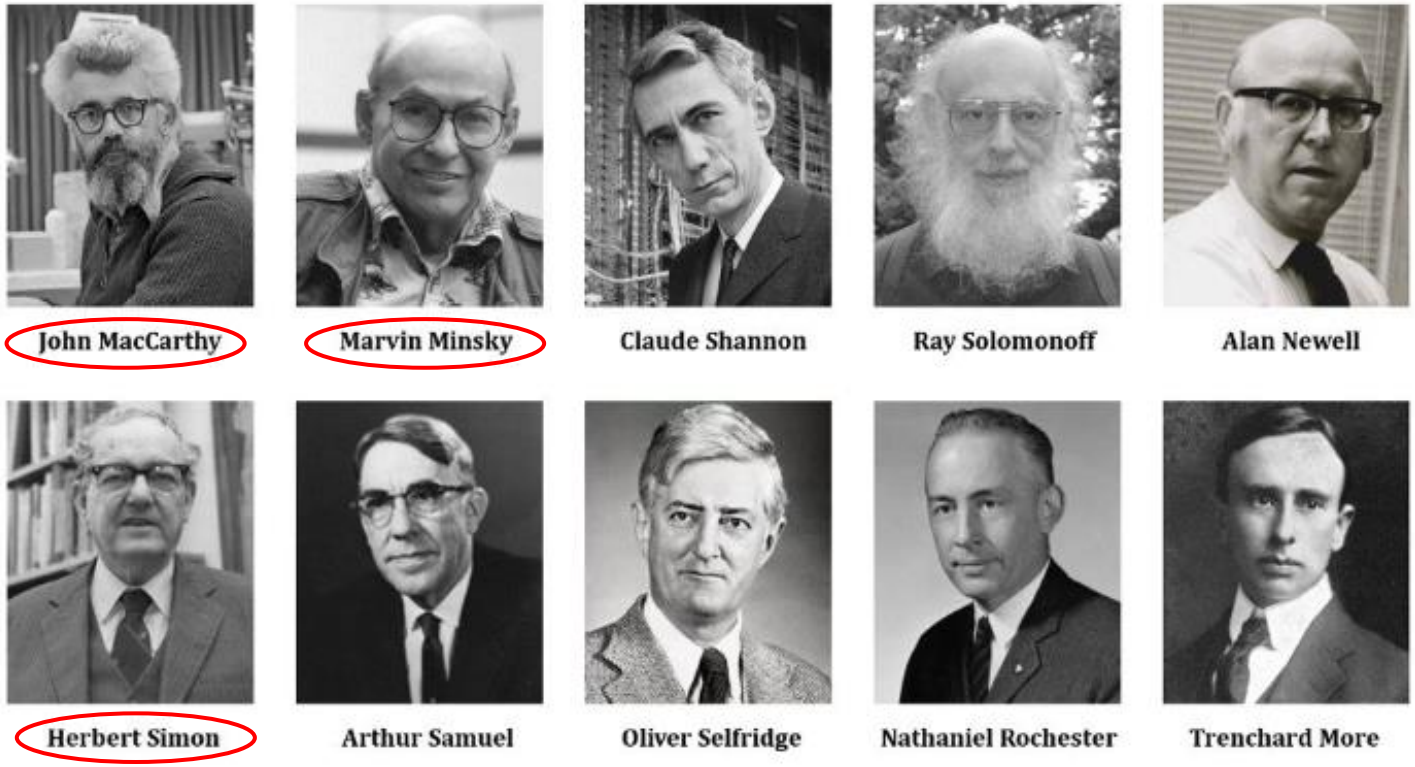
人工智能算法在多种应用上接近或超过了人类水平

10.1 人工智能诞生的前夜

- 1930s-1950s，理论的发展为AI破土而出奠定基础
  - Norbert Wiener 的控制论
  - Claude Shannon的信息论
  - Alan Mathison Turing的计算理论
  - 神经科学的发展
- 1950 Turing提出著名的图灵测试：

  如果机器与人类进行对话而人无法辨别机器身份，则该机器具有智能
- 1951 Marvin Minsky和Dean Edmonds创建神经网络机器SNARC
- 1955 Allen Newell和Herbert Simon开发“逻辑理论家”证明《数学原理》52个定理中的38个

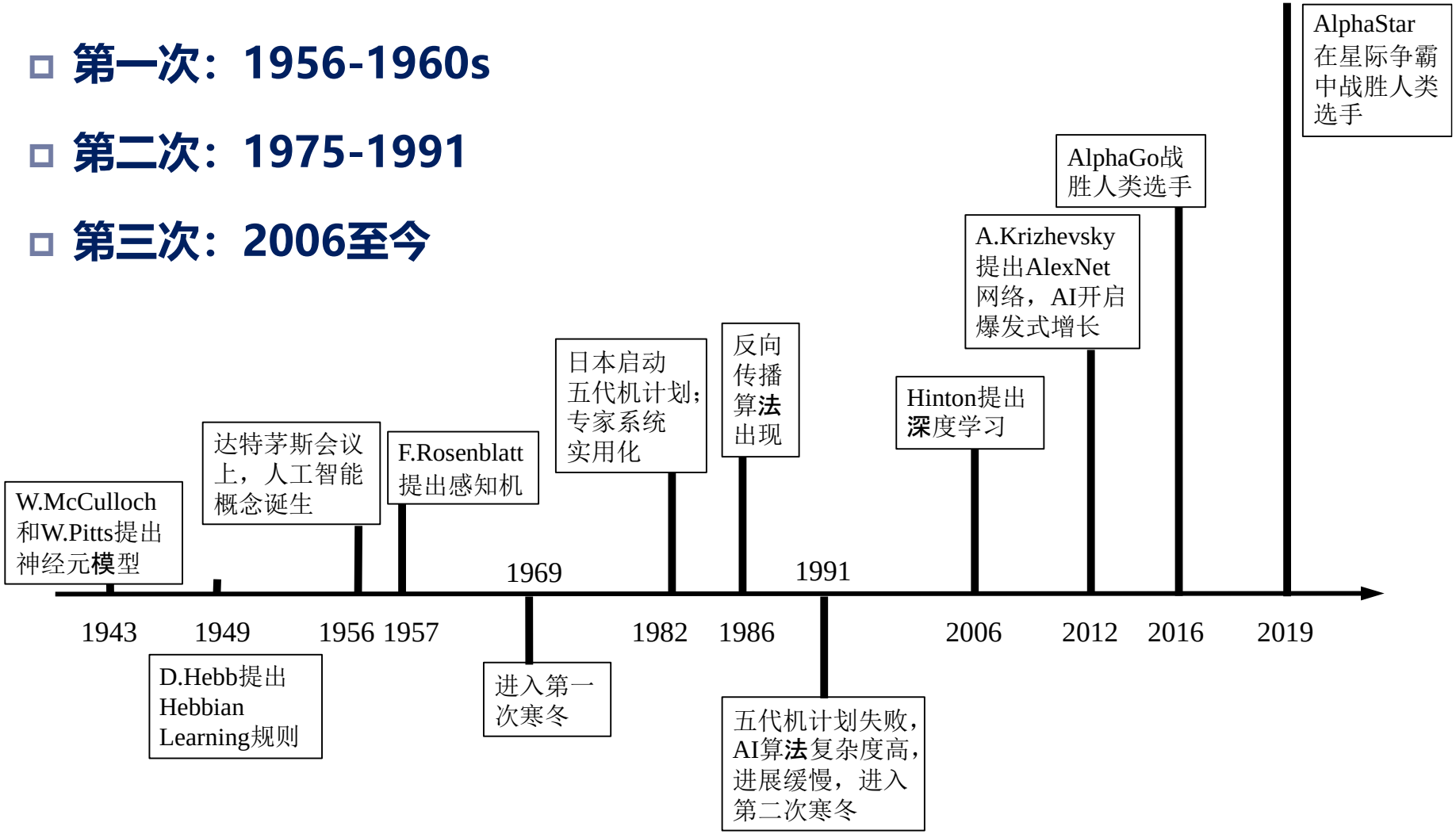
10.1 AI诞生—1956年达特茅斯人工智能研讨会



如果学习或者智能的其他特征可以被精确描述，就可以用一台机器来模拟智能，并尝试让机器使用语言、形成抽象概念、解决人类才能解决的各种问题，甚至自我完善

10.1 人工智能的三次热潮

- 第一次：1956-1960s
- 第二次：1975-1991
- 第三次：2006至今



10.1 人工智能的三个流派

- **行为主义(actionism)：又称为进化主义或控制论学派，基于控制论，构建感知-动作型控制系统**
- **符号主义(symbolicism)**：又称为逻辑主义、心理学派或计算机学派，**基于符号逻辑的方法，用逻辑表示知识和求解问题**
- **连接主义(connectionism)：又称为仿生学派或生理学派，基于大脑中神经元细胞连接的计算模型，用人工神经网络来拟合智能行为**

以深度神经网络为代表的连接主义是当前业界关注的焦点

10.2 卷积神经网络(CNN)结构简介

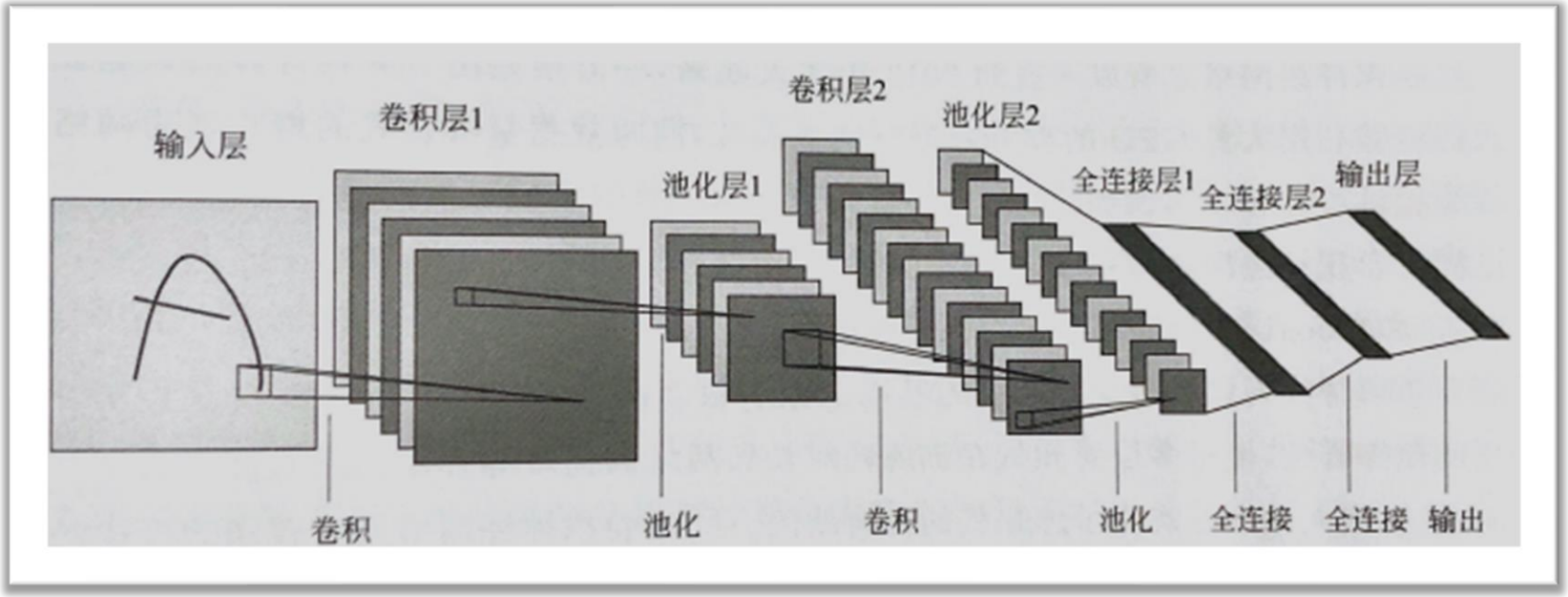
卷积神经网络CNN是多层神经网络的代表

- CNN本质上是M-P神经元模型的递归调用
- 结构定义、网络模型训练和推理计算三个过程
- 层次结构定义：输入/卷积/池化/全连接/输出层，灵活排列或重复利用、再进行叠加，产生繁多的CNN
- 网络模型训练：大数据训练，获得网络最优权重及其它参数
- 推理计算：对新的数据完成测试或预测

CNN为具体智能应用提供端到端的学习模型

10.2 LetNet5(1998)网络结构

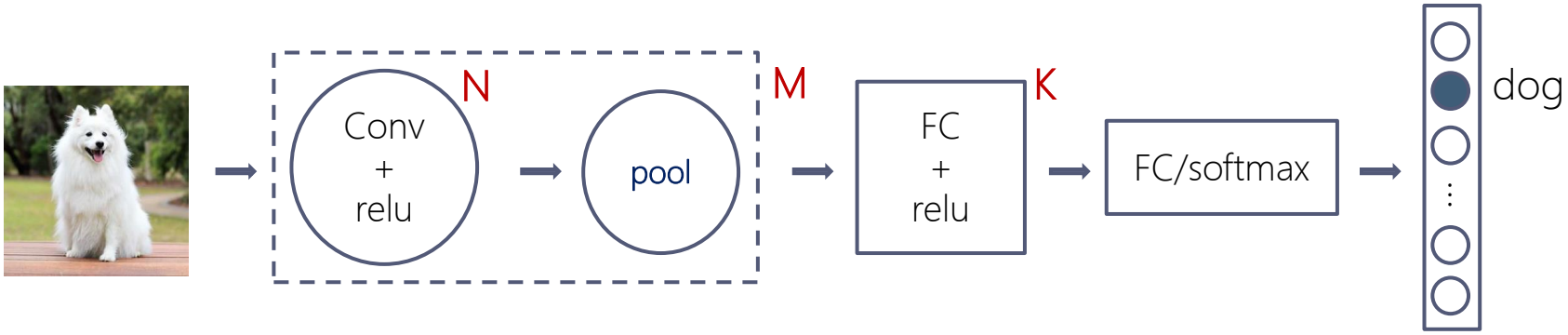
最早的CNN网络是Yann LeCun 1998年提出的LetNet5



- 功能层：卷积层、池化层
- 每个功能层都有：输入特征图 IFM、输出特征图OFM
- 同功能层：所有IFM大小相同，所有OFM大小相同，IFM和OFM是否相同取决于功能层的计算特性
- 输入数据预处理

10.2 卷积神经网络结构

层排列规律



- **常见卷积神经网络由卷积层（激活）、池化层和全连接层构成；**
- **各层的常见排列方式如图所示，其中N、M、K为重复次数；**
- **例如：N=3，M=1，K=2 情况下的网络结构为：**

$$\text{input} \rightarrow \text{conv(relu)} \rightarrow \text{conv(relu)} \rightarrow \text{conv(relu)} \rightarrow \text{pool} \rightarrow \text{FC(relu)} \rightarrow \text{FC(relu)} \rightarrow \text{FC} \rightarrow \text{output}$$

- **其中卷积和池化部分可包含分支和连接结构**

10.2 卷积神经网络结构

为何选择 “深” 而非 “广” 的网络结构

- 即使只有一层隐层，只要有足够的神经元，神经网络理论上可以拟合任意连续函数。为什么还要使用深层网络结构?

- **深度网络可从局部到整体“理解图像”**

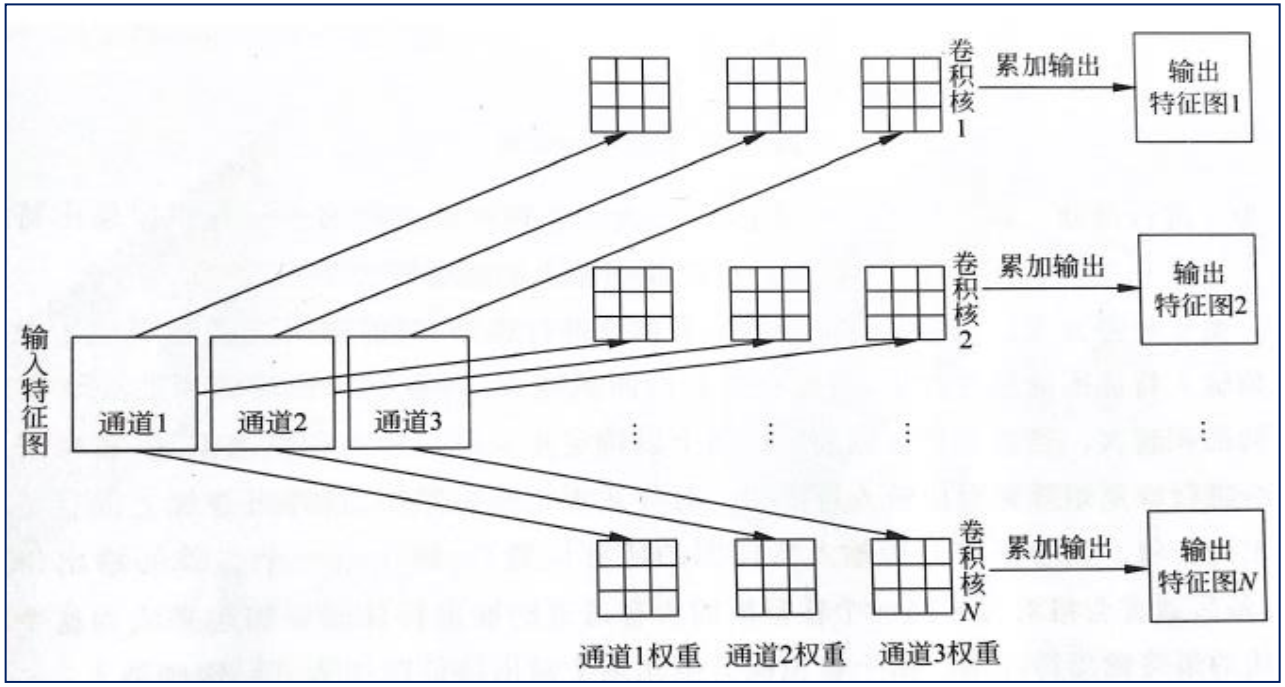
  学习复杂特征时（例如人脸识别），浅层的卷积层感受野小，学习到局部特征，深层的卷积层感受野大，学习到整体特征。

- **深度网络可减少权重数量**

  以宽度换深度，用多个小卷积替代一个大卷积，在获得更多样特征的同时所需权重数量也更少。

10.2 卷积层及计算特性

- CNN网络运算的核心层，增强特征信息、过滤无用信息
- 通过多次卷积计算提取IFM关键特征，为下一层生成OFM
- 不同卷积核(Filter/Kernel)提取不同特征，二维权重矩阵
- 步长：卷积核在IFM上一次滑动距离



卷积层计算示意图

10.2 卷积层

- 卷积计算

w0	w1	w2
w3	w4	w5
w6	w7	w8

filter/kernel

将权重作为参数，在训练中学习

数学：

$$y(n) = \sum_{i=-\infty}^{\infty} x(i)h(n-i) = x(n) * h(n)$$

*** 表示卷积**

神经网络：实际为计算矩阵内积(相关系数)

2 (1)	3 (0)	1 (1)	5	2	3
7 (4)	4 (-3)	5 (2)	2	3	1
3 (3)	9 (0)	6 (-1)	0	4	2
0	6	4	7	1	2
4	1	0	8	0	6
7	0	2	1	6	3

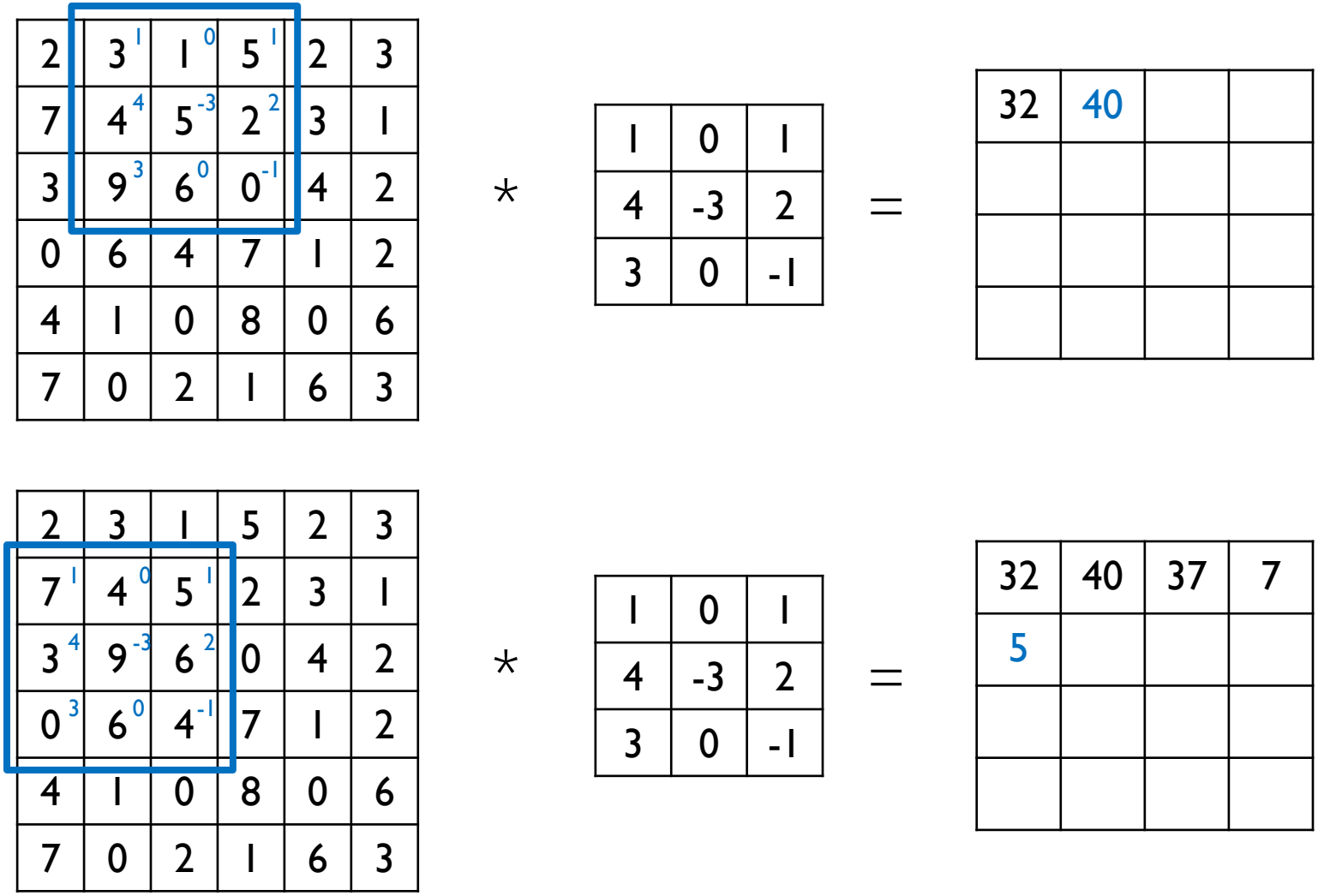
$*$

1	0	1
4	-3	2
3	0	-1

$=$

32			

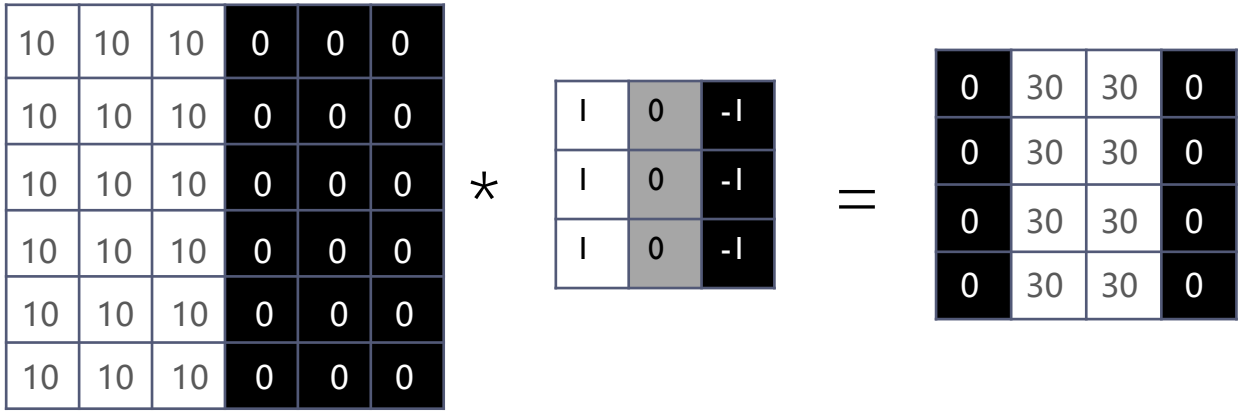
10.2 卷积层


2 3 1 0 5 1 2 3
7 4 4 5 -3 2 2 3 1
3 9 3 6 0 0 -1 4 2
0 6 4 7 1 2
4 1 0 8 0 6
7 0 2 1 6 3
*
1 0 1
4 -3 2
3 0 -1
=
32 40
2 3 1 5 2 3
7 1 4 0 5 1 2 3 1
3 4 9 -3 6 2 0 4 2
0 3 6 0 4 -1 7 1 2
4 1 0 8 0 6
7 0 2 1 6 3
*
1 0 1
4 -3 2
3 0 -1
=
32 40 37 7
5

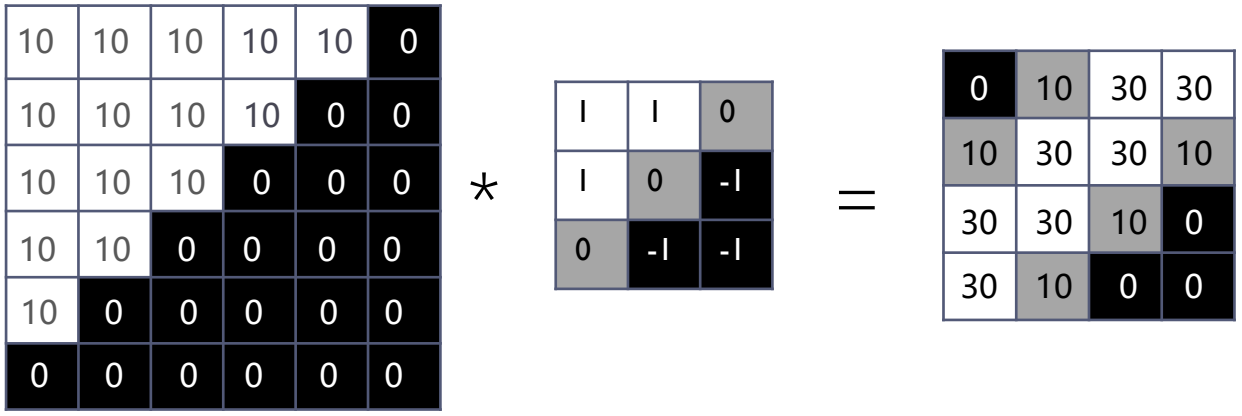
10.2 卷积层

卷积层如何检测特征

◆ **检测垂直边缘**

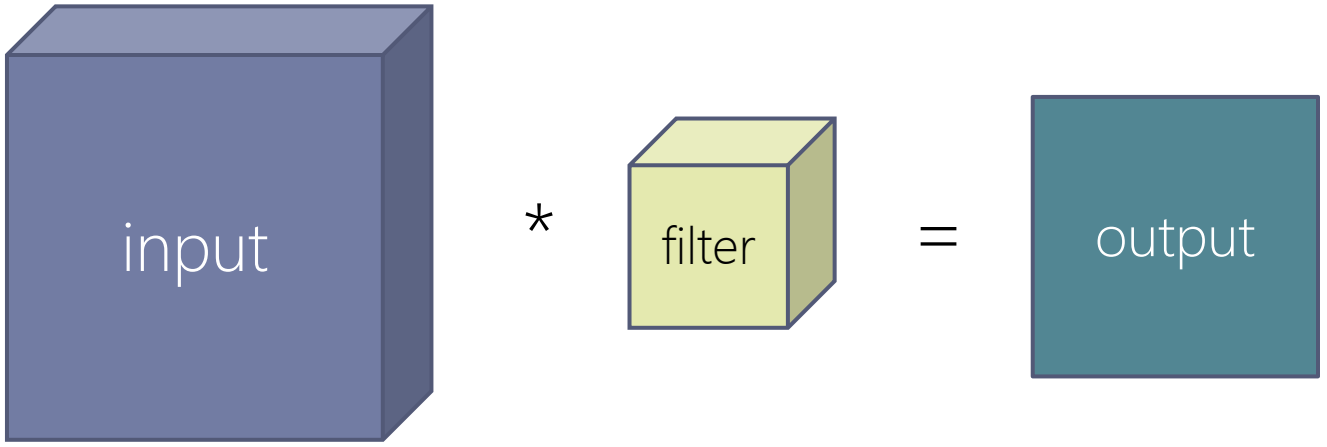


◆ **检测对角线边缘**

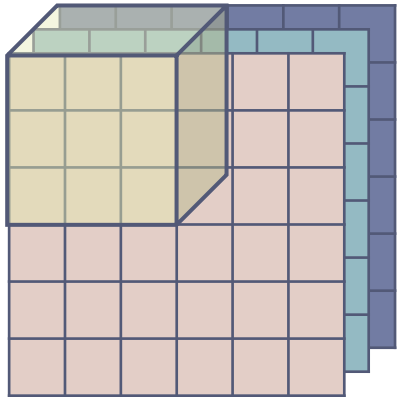


10.2 卷积层

- 多输入特征图—单输出特征图 卷积运算

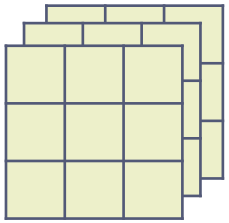


10.2 卷积层



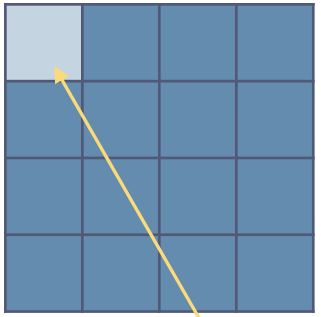
6x6x3

*



3x3x3

=



4x4

C = 0

0	0	0
0	2	2
0	1	2

-1	1	1
-1	1	-1
1	-1	1

C = 1

0	0	0
0	0	2
0	1	2

*

1	-1	-1
-1	0	-1
-1	0	1

=

2-2-1+2
+
0-2+0+2
+
(-1)+0+0+2

=

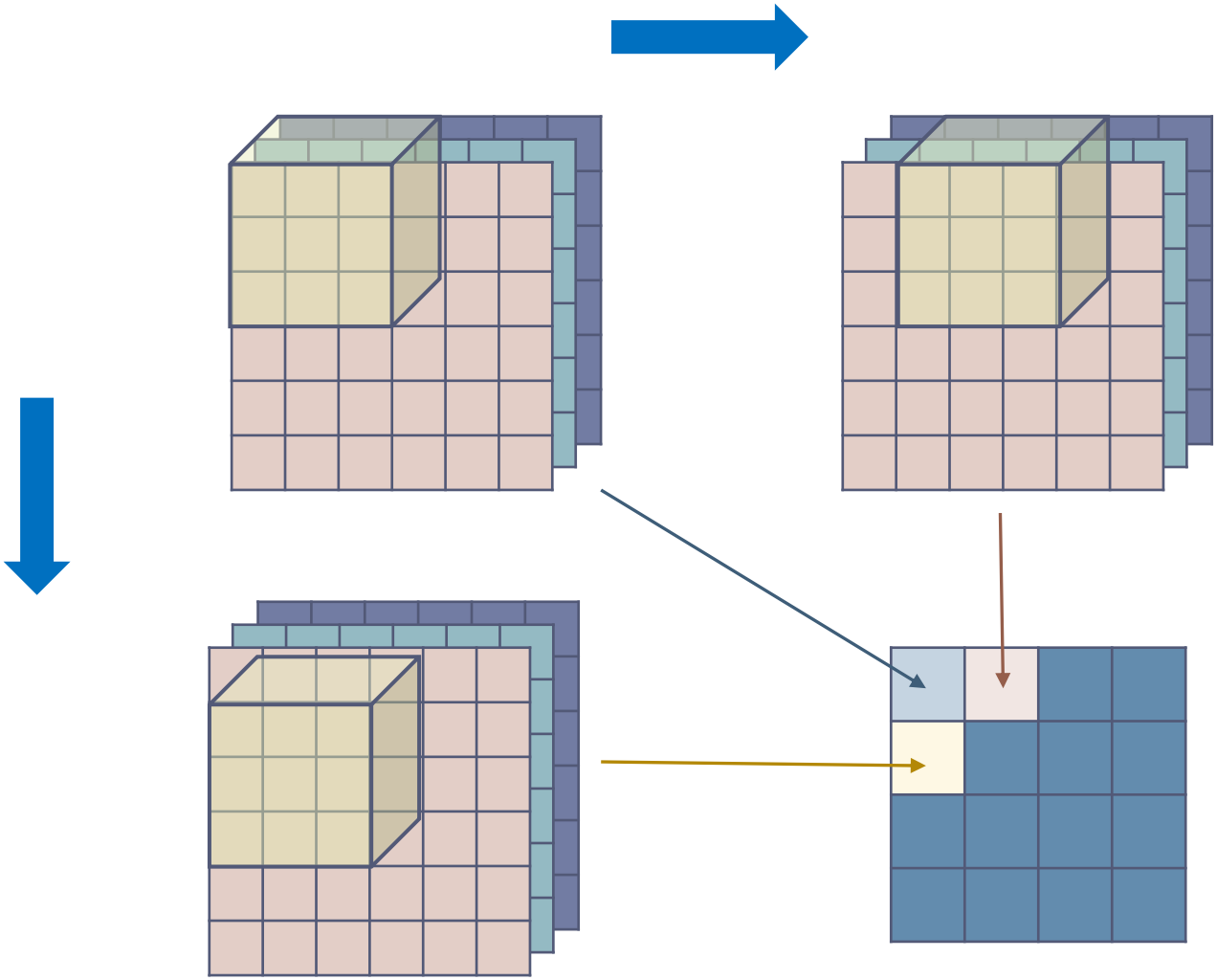
2

C = 2

0	0	0
0	1	1
0	0	2

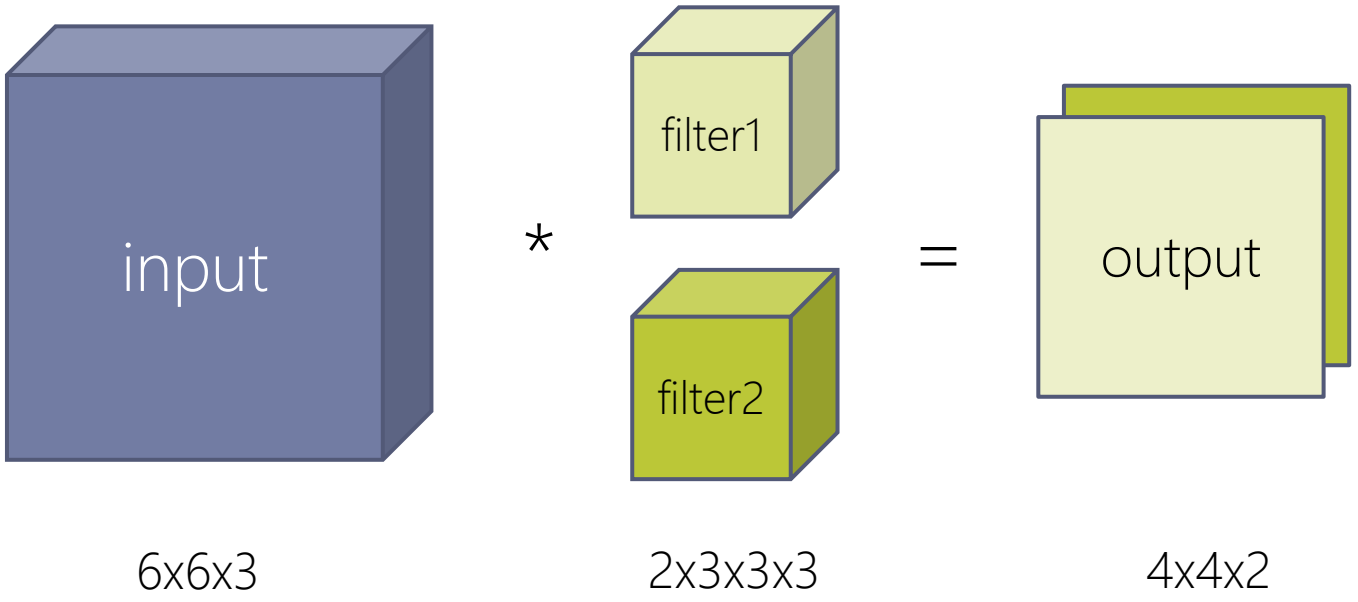
1	-1	-1
-1	-1	0
-1	1	1

10.2 卷积层



10.2 卷积层

- 多输入特征图—多输出特征图 卷积运算

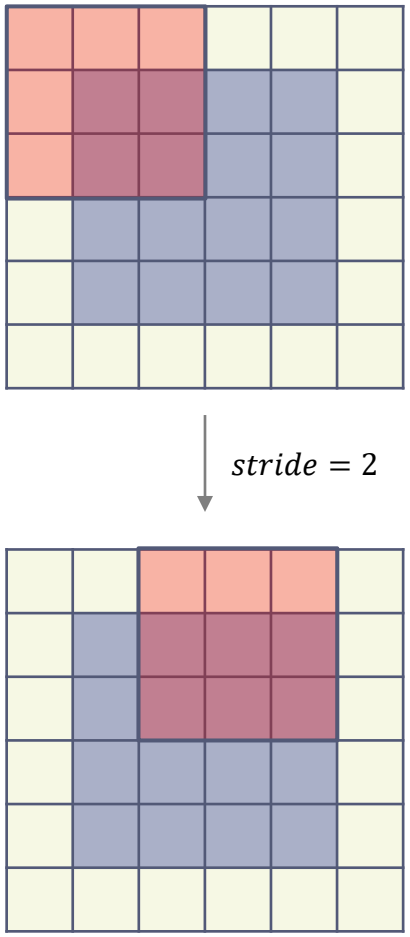


不同的过滤器可检测不同特征

10.2 卷积层

- 边界扩充(padding)
  - ◆ **扩大输入图像/特征图的尺寸并填充像素**
  - ◆ **防止深度网络中图像被动持续减小**
  - ◆ **强化图像边缘信息**
- **卷积步长(stride)**
  - ◆ **滑动滤波器时每次移动的像素点个数**
  - ◆ **与pad共同确定输出图像尺寸**

$input = 4 \times 4$，$filter = 3 \times 3$，$pad = 1$

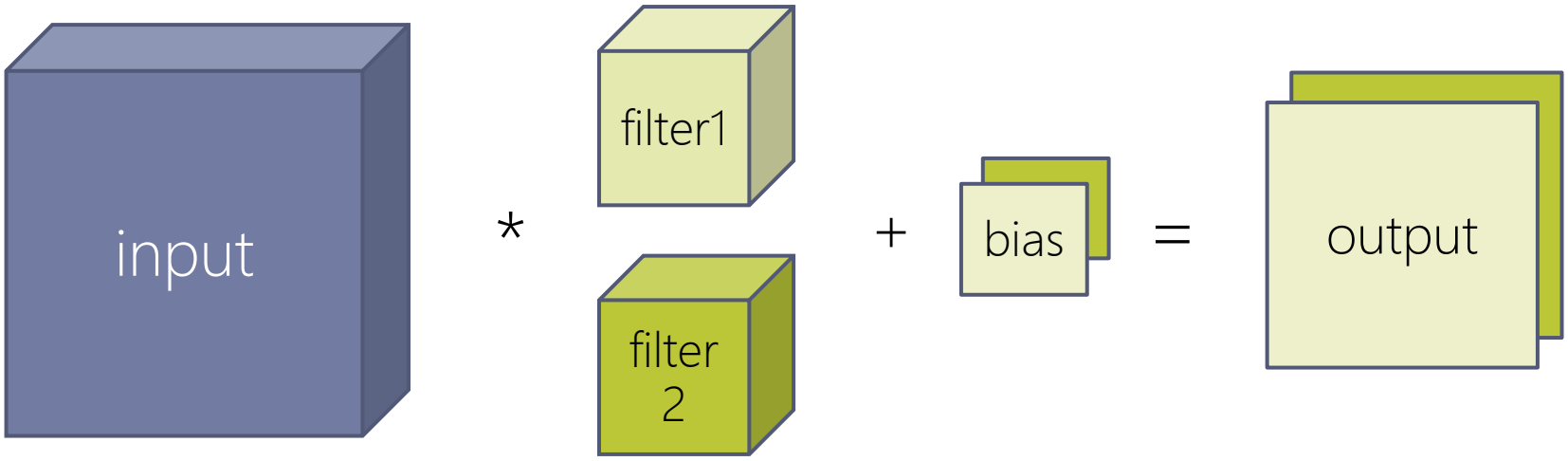


$$input = w_i \times h_i,\ filter = f \times f,\ pad = p,\ stride = s$$

$$output = \left\lfloor \frac{w_i + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{h_i + 2p - f}{s} + 1 \right\rfloor$$

10.2 卷积层

- **总结：卷积层参数**



$w_i \times h_i \times c_i$　　$c_o \times f \times f \times c_i$　　$1 \times 1 \times c_o$　　$w_o \times h_o \times c_o$

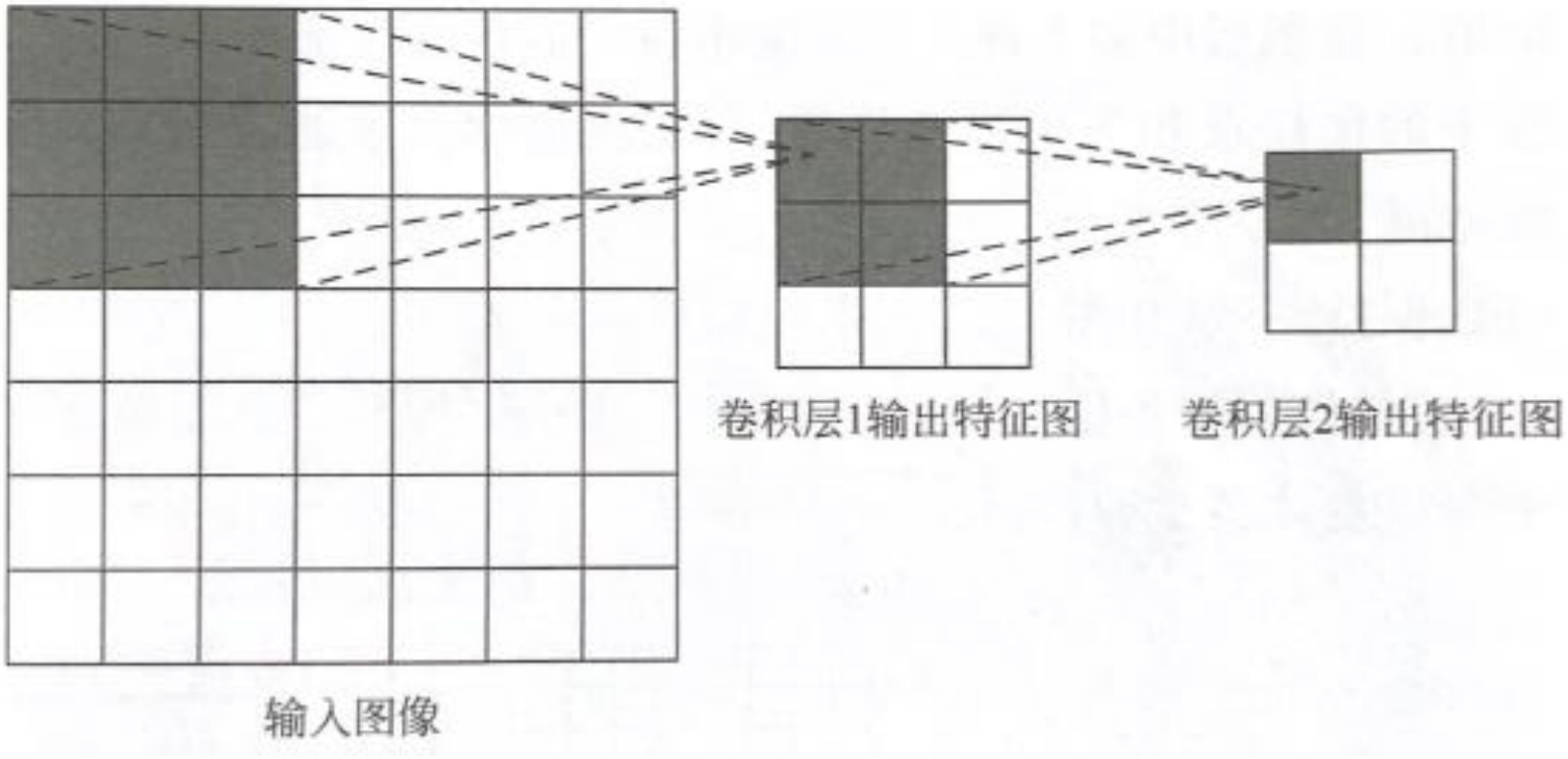
- stride
- pad
- $\text{output} = \left[\frac{w_i+2p-f}{s}+1\right] \times \left[\frac{h_i+2p-f}{s}+1\right]$

◆ **filter：可训练**

◆ **bias：可训练，使分类器偏离激活函数原点，更灵活；**

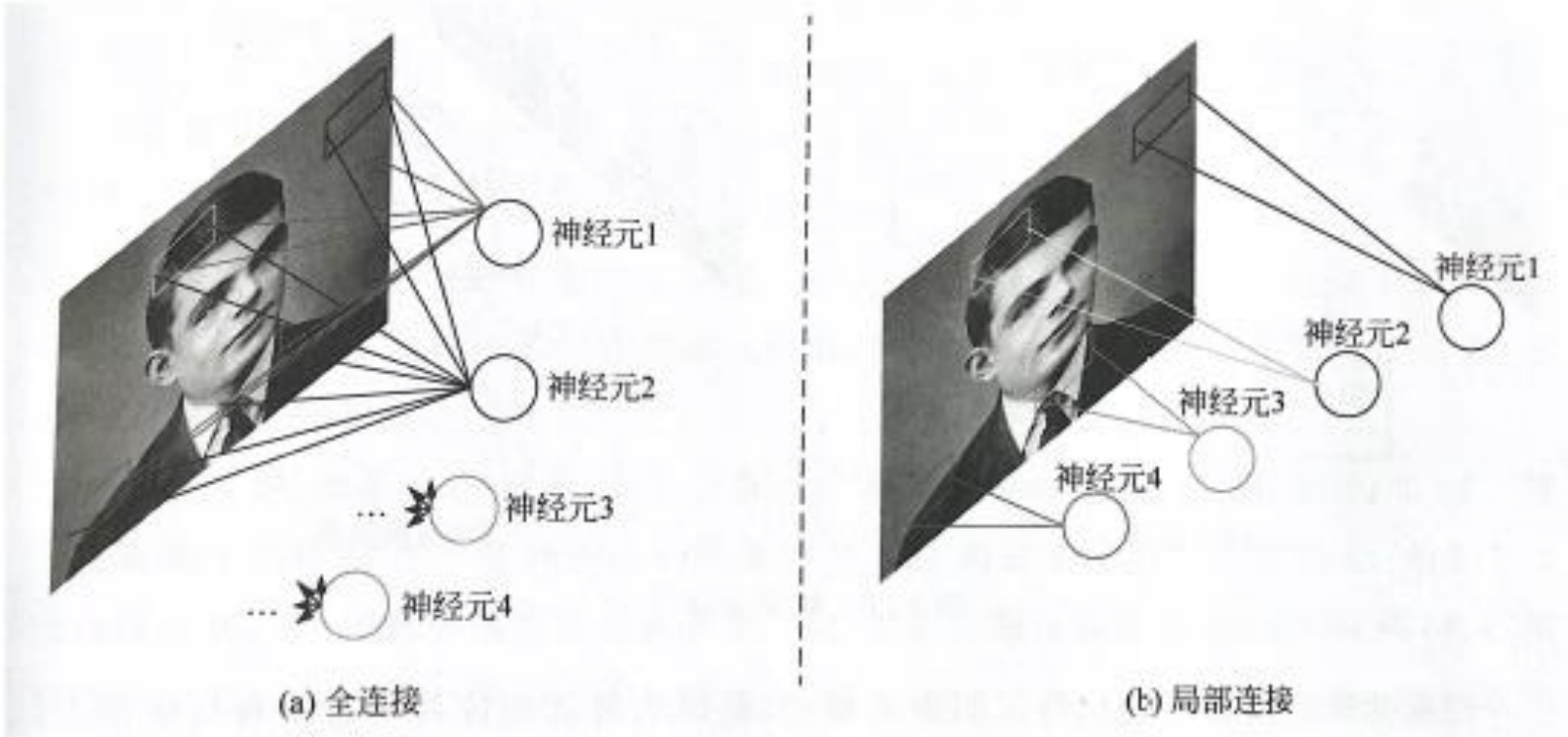
◆ **activation**

10.2 感受野—卷积窗口尺寸



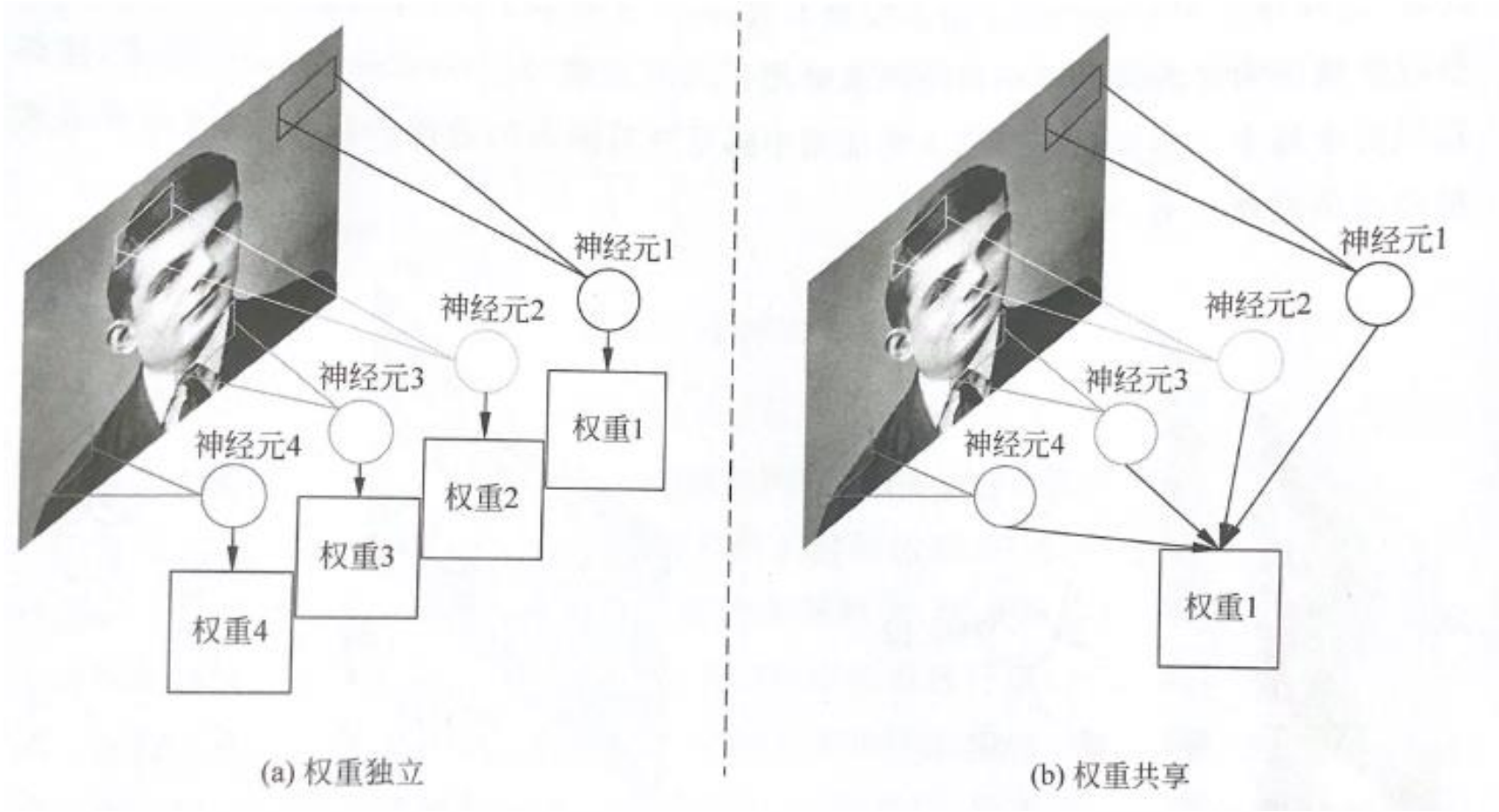
感受野示意图

10.2 卷积层特性—局部连接



视觉具有强局部性，距离越远的像素点相关性越弱，局部连接具有合理性

10.2 卷积层特性—权重共享



图像具有固有特性信息，在不同部分均有体现，权重共享具有合理性

10.3 卷积层计算方法

X

1	1	1	1	2
1	1	1	2	1
0	0	2	1	2
0	0	0	1	2
1	2	1	1	1

W_1

1	0	1
0	0	−1
0	0	−1

b_1 0

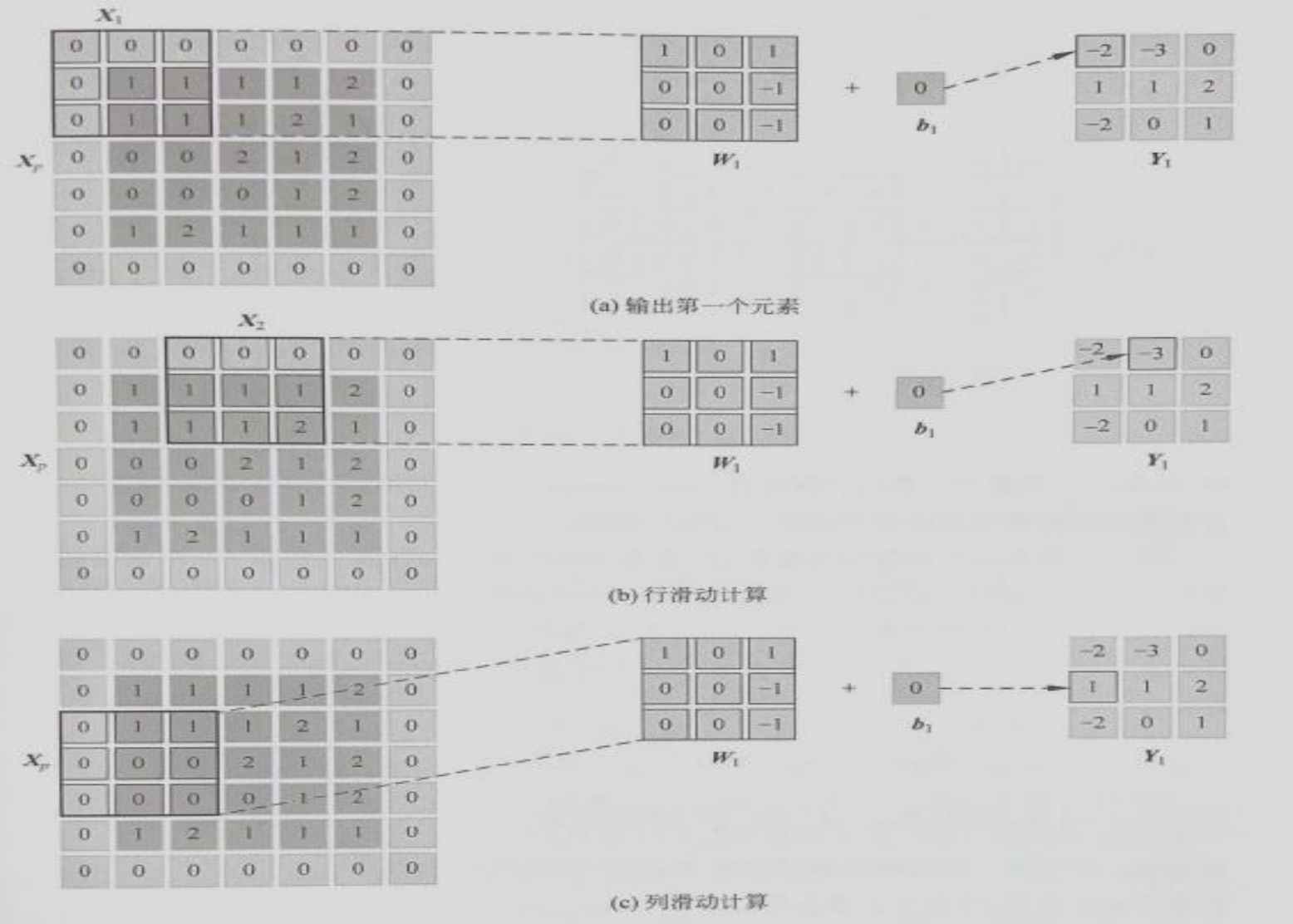
W_2

−1	0	0
−1	0	0
−1	1	−1

b_2 1

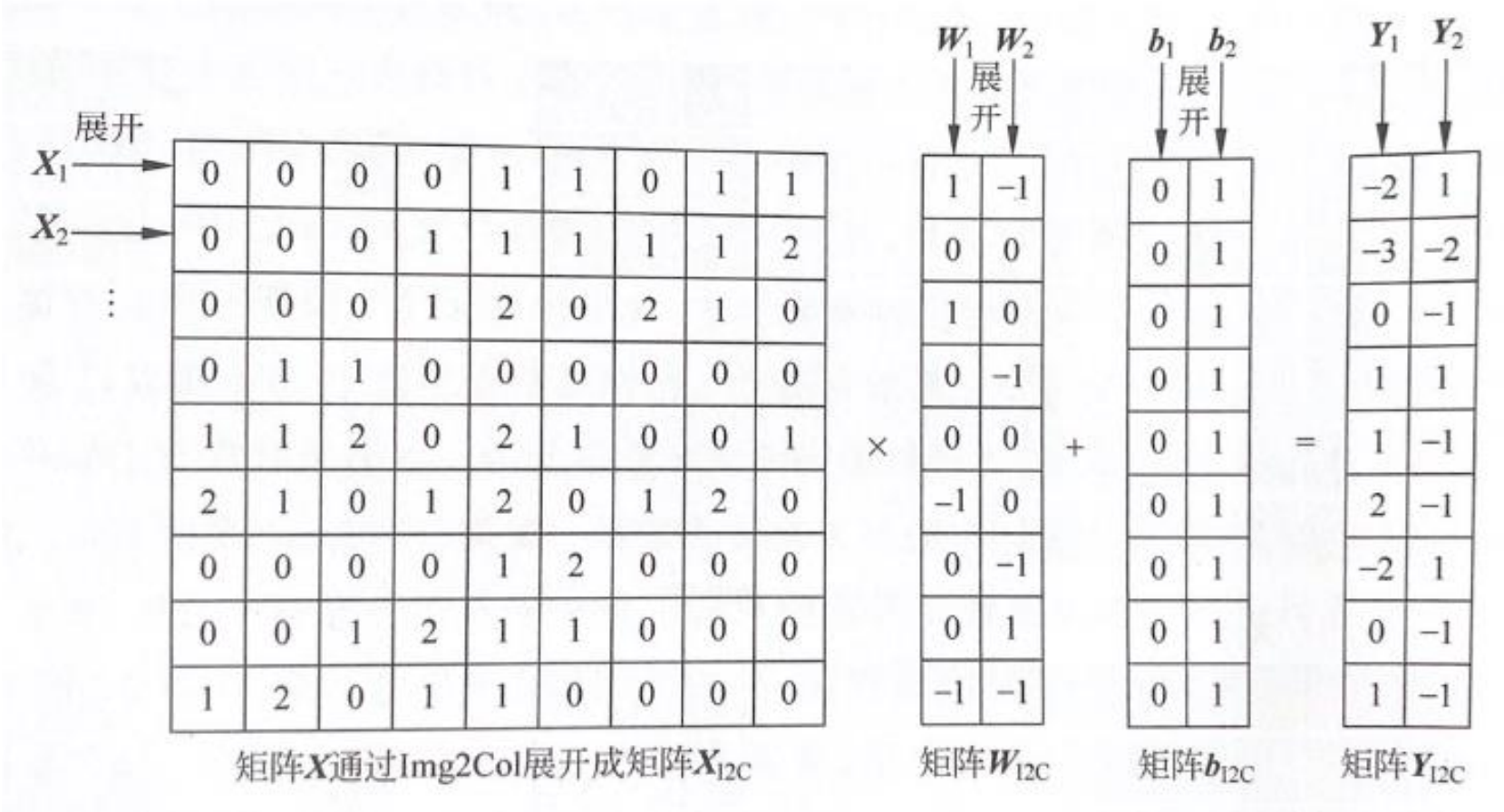
卷积输入数据

10.3 直接卷积计算



谷歌的TPU采用直接卷积脉动阵列计算方式

10.3 矩阵相乘计算卷积

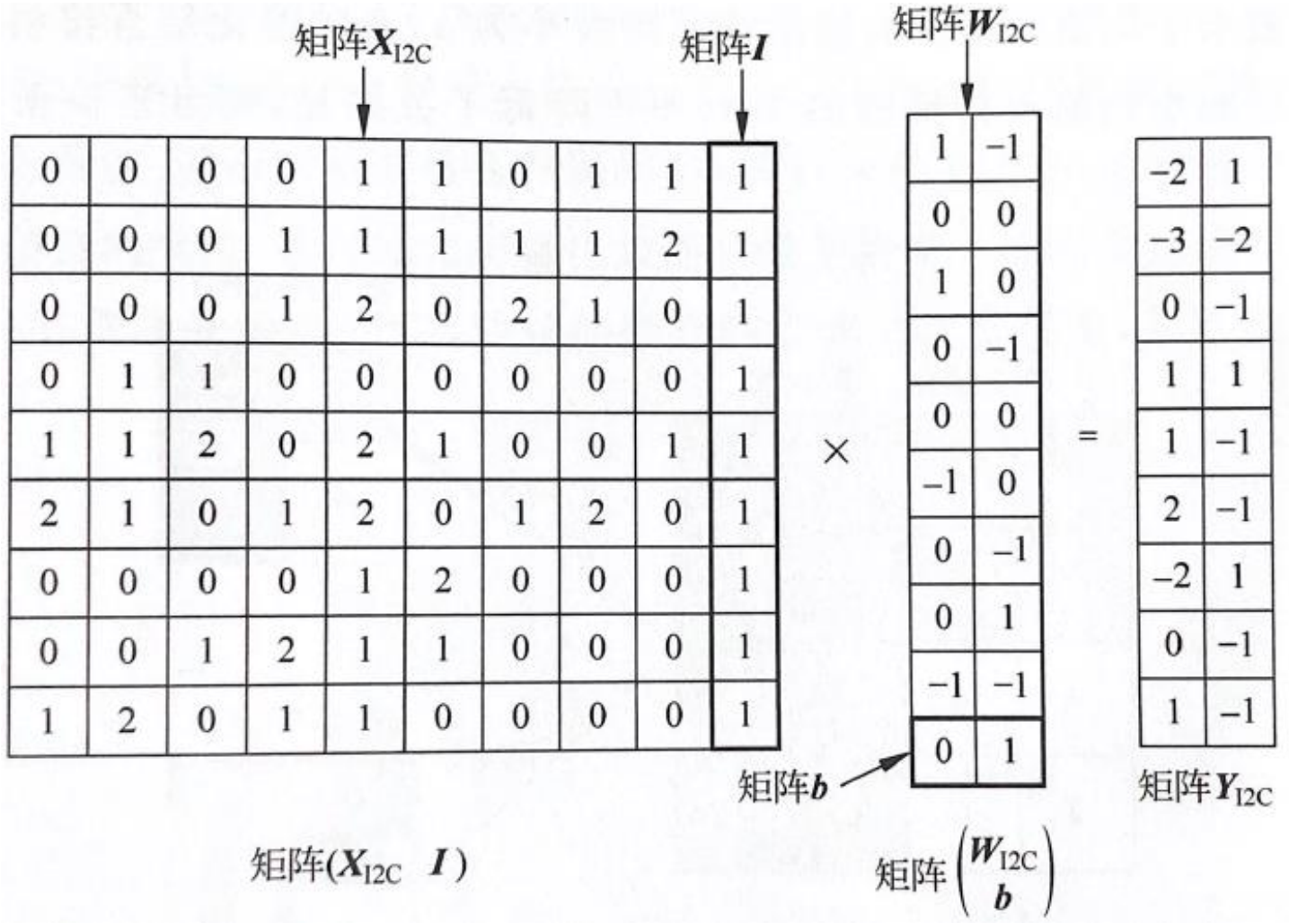


◆Img2Col：每一个输入特征子矩阵展开为一行（或一列）

◆CPU、GPU等通用编程性的处理器上，包括华为昇腾处理器

◆ 提供专门基本线性代数程序库BLAS高效实现向量和矩阵运算

10.3 卷积+偏置优化



0	0	0	0	1	1	0	1	1	1
0	0	0	1	1	1	1	1	2	1
0	0	0	1	2	0	2	1	0	1
0	1	1	0	0	0	0	0	0	1
1	1	2	0	2	1	0	0	1	1
2	1	0	1	2	0	1	2	0	1
0	0	0	0	1	2	0	0	0	1
0	0	1	2	1	1	0	0	0	1
1	2	0	1	1	0	0	0	0	1

1	−1
0	0
1	0
0	−1
0	0
−1	0
0	−1
0	1
−1	−1
0	1

−2	1
−3	−2
0	−1
1	1
1	−1
2	−1
−2	1
0	−1
1	−1

累加偏置优化

10.3 矩阵相乘计算卷积

矩阵相乘计算卷积

◆需要计算的特征子矩阵存放在连续内存

◆有利于一次将所需的数据直接按照需要的格式取出计算

◆减少了访存次数，从而减少整体计算时间

直接计算卷积

◆输入特征子矩阵存放在内存中，地址有重叠但不连续

◆计算时需要多次访存

◆增加了数据传输时间、影响整体计算速度

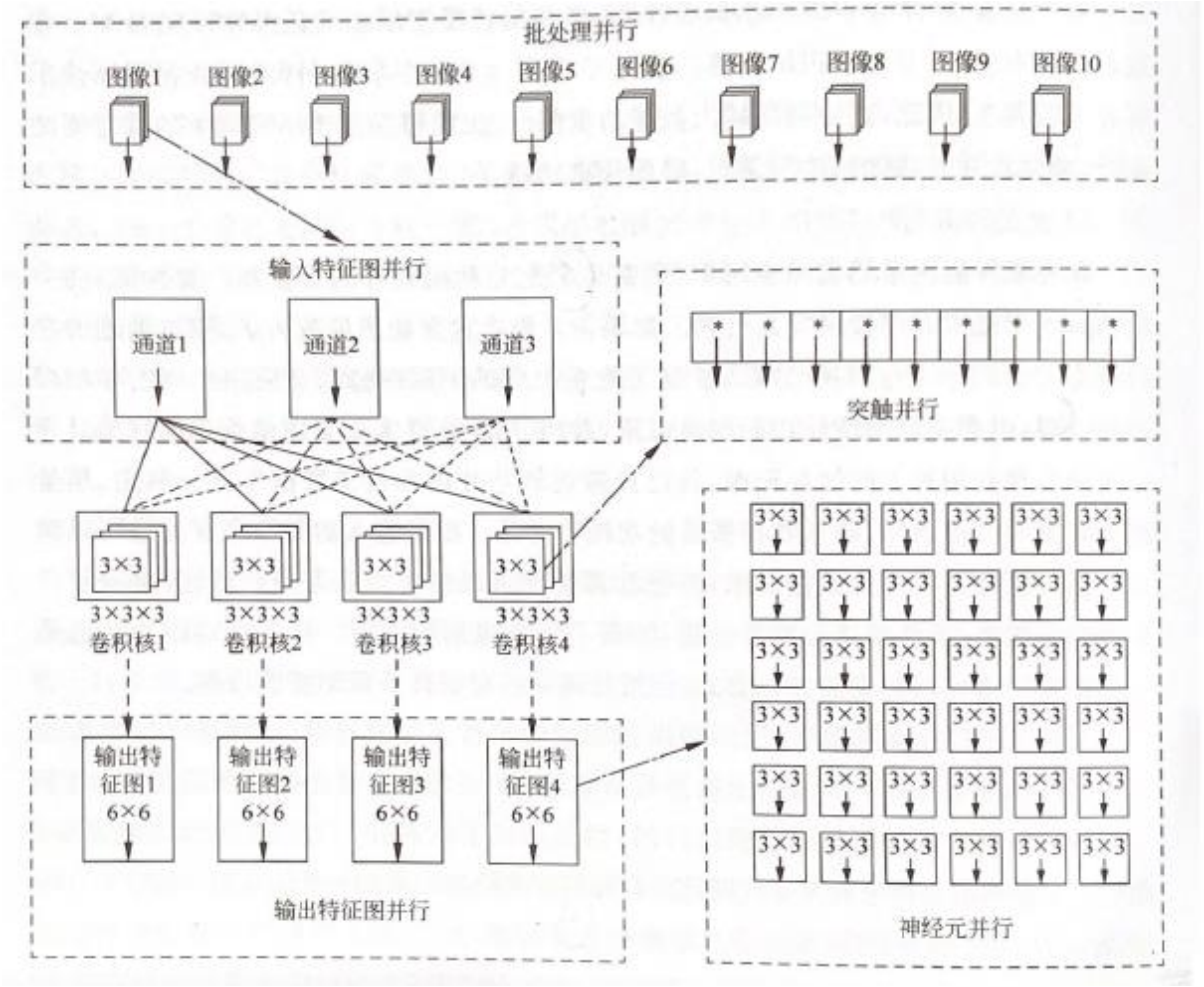
10.3 卷积层5种层次并行优化

- 突触并行
- 神经元并行
- IFM并行
- OFM并行
- 批处理并行

层次由低到高

计算数据量由小到大

资源需求越来越高

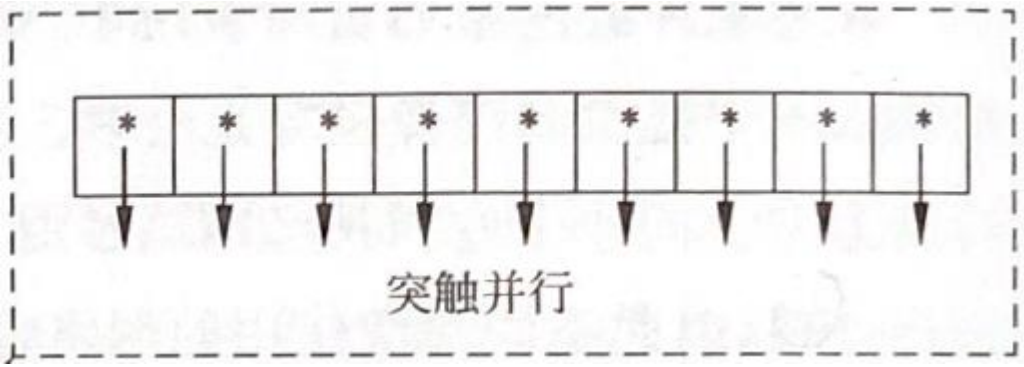


N通道、单个核K×K、核数量M、OFM大小$S_o \times S_o$

10.3 突触并行

单次卷积窗内的计算并行

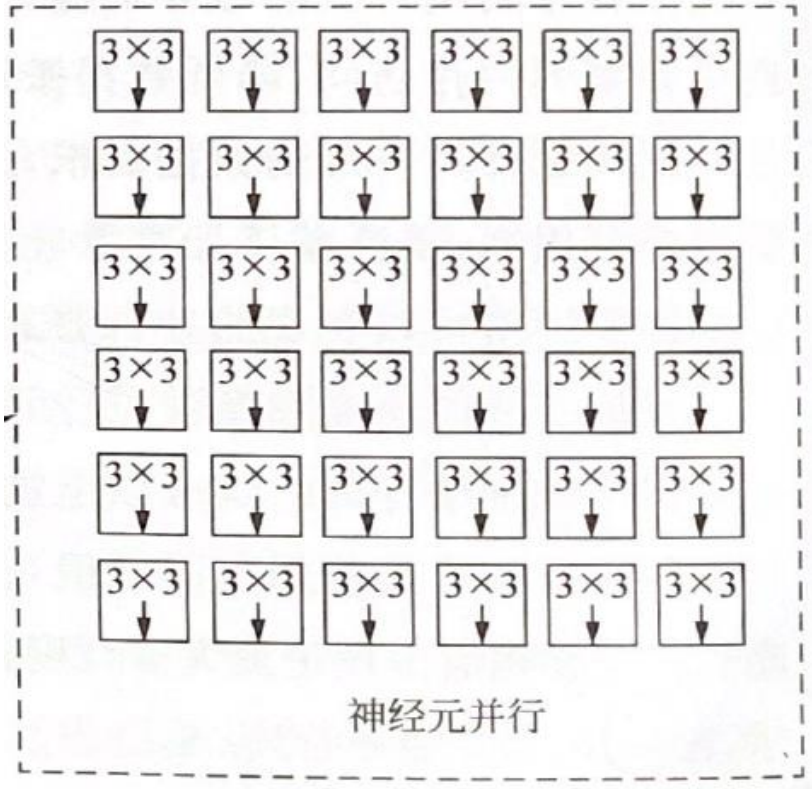
- K×K个乘法操作无数据依赖，可以并行执行
- 一次乘法操作类似神经元的一个突触信号作用在树突上
- 最大突触并行度为K×K



10.3 神经元并行

多次卷积窗间的计算并行

- 每个卷积窗口计算的过程无数据依赖，可以并行执行
- 每个卷积窗口内计算类似一个神经元的计算，称为神经元并行
- 最大神经元并行度为$S_o \times S_o$

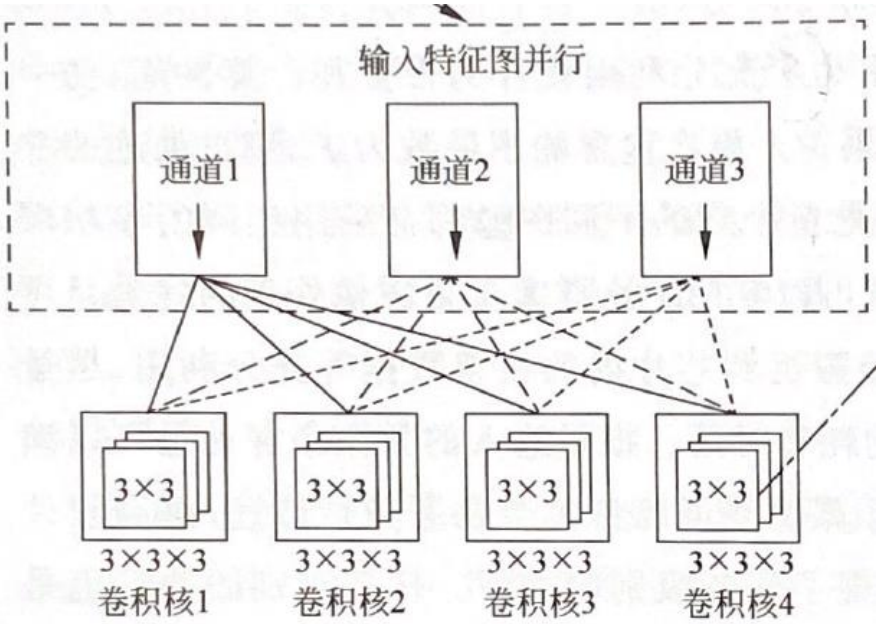


注意：多个卷积窗口输入特征值会有重叠，重叠部分的数据具有重用性，充分利用可以减少片上缓存需求和访存次数，从而降低功耗同时提高性能

10.3 输入特征图并行

多个通道间的计算并行

- IFM的多个通道的卷积运算过程相互独立，可以并行执行
- N通道IFM最大并行度为N

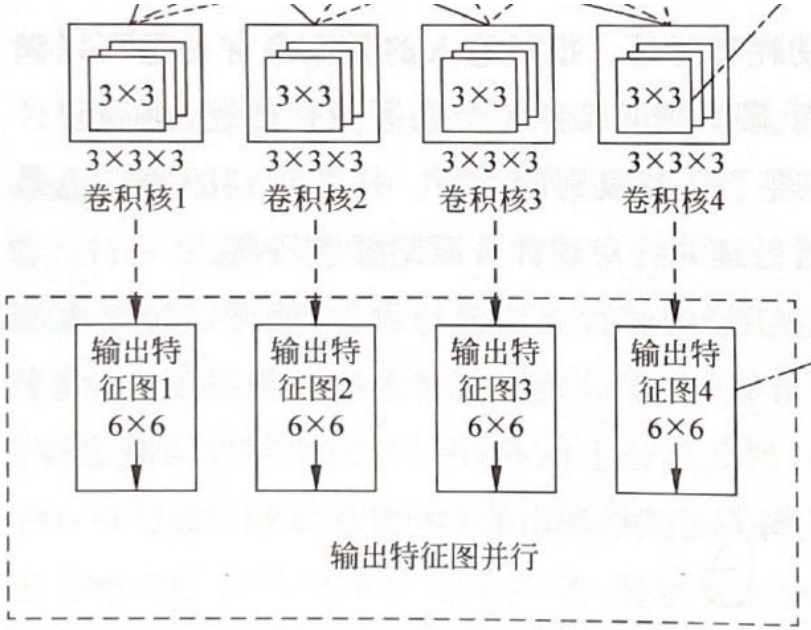


注意：多个通道的卷积运算之间的数据没有重叠，硬件提供足够的带宽和算力，就能进行IFM的并行计算

10.3 输出特征图并行

多个卷积核卷积过程的计算并行

- 同一IFM，多个卷积核产生多个OFM，多个卷积核计算数据独立无相互依赖，可以并行执行
- M个卷积核在输出特征图层级的最大并行度为M

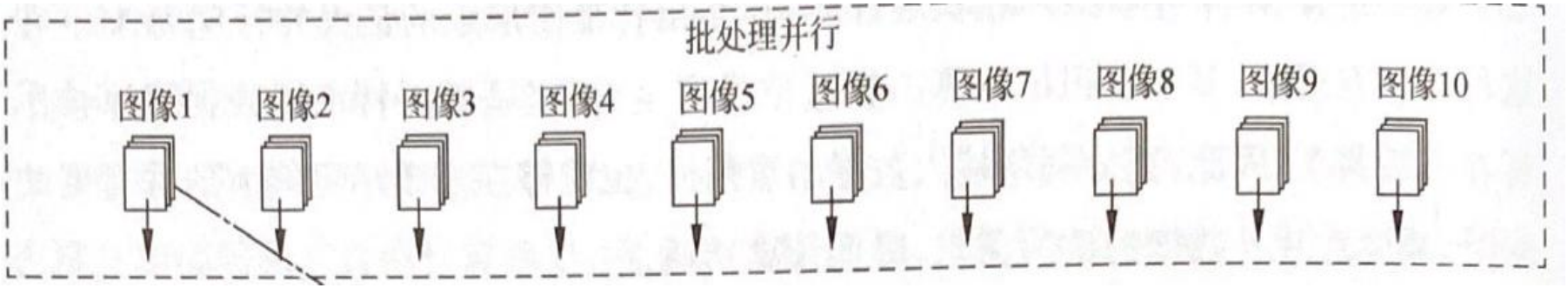


注意：多个卷积核运算的权重值没有重合，但IFM数据可以被所有卷积核共享，输入数据有重用性，可减少片外数据访存，降低计算功耗

10.3 批处理并行

输入一个批次的图像进行批处理，任务级别的并行

- **一个批次包含的图像为P，批处理的最大并行度为P**
- **不同图像使用的网络权重可以复用，充分利用已经搬运到片上的权重数据，避免频繁访存，降低功耗和延时**
- **满足片上海量计算资源的数据需求（避免等数据）**



10.3 5种层次的并行度

理论上，5种层次的并行方式均可以同时实施

- 最大并行度为= $K \times K \times S_o \times S_o \times N \times M \times P$

实现最大并行度等同于所有任务的卷积层中的所有乘法同时计算

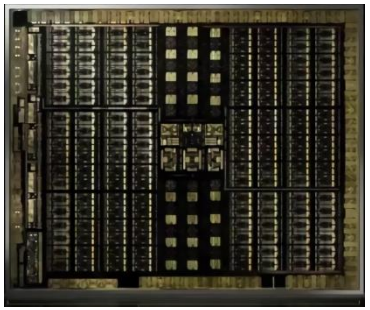
- 最大并行度为= $3 \times 3 \times 6 \times 6 \times 3 \times 4 \times 10 = 38880$

可同时进行38880个乘法计算！

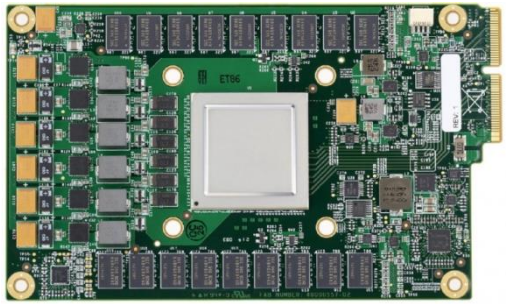
实际情况，图像尺寸大、卷积核众多、图像数也很多，存在数据传输通信、访存开销等，阿姆达定律的限制，无法达到理论并行加速比

10.4 AI处理器发展历史

- 人工智能的四大要素：数据、算法、场景、**算力**
- AI处理器：也被称为AI加速器，即专门用于处理人工智能应用中的大量计算任务的功能模块
- AI处理器的发展经历了从CPU到GPU，再到FPGA和ASIC芯片的发展历程
- 计算机科学导向、神经科学导向（类脑芯片）

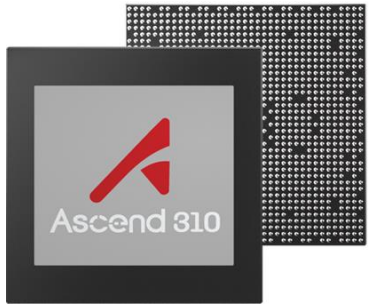


NIVIDA的TU102



Google的TPU

Xilinx 的Versal



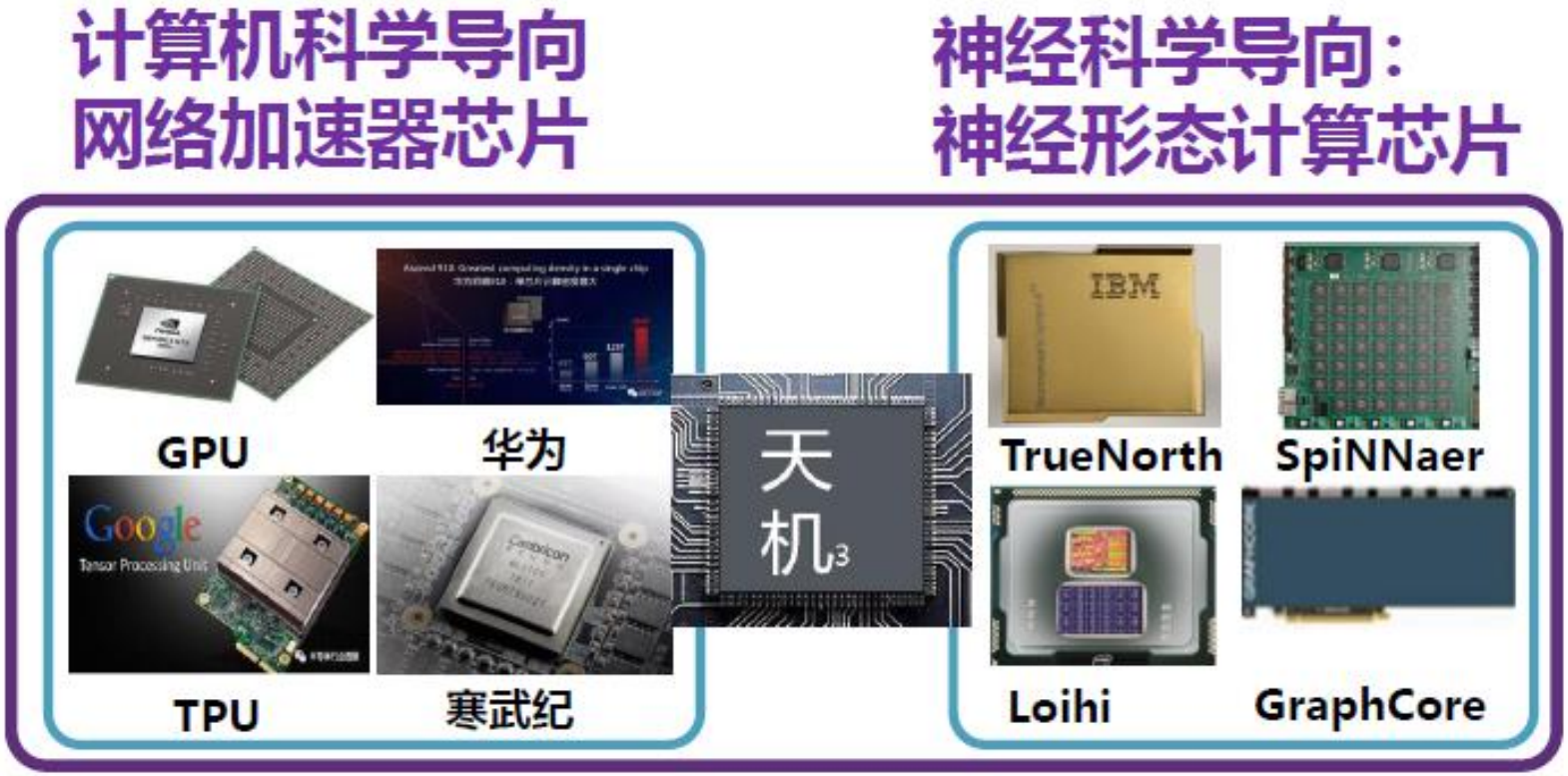
华为昇腾310

10.4 AI处理器应用场景

- Training（训练）和Inference（推理）两个阶段：
  - Training环节涉及海量的训练数据和复杂的深度神经网络结构，运算量巨大，需要庞大的计算规模，对于处理器的计算能力、精度、可扩展性等性能要求很高，常用NVIDIA的GPU集群、Google的TPU等
  - Inference环节指利用训练好的模型，使用新的数据去“推理”出各种结论，Inference的计算量相比Training少很多，仍然涉及大量的矩阵运算，常用GPU、FPGA和ASIC

10.4 AI处理器

从计算原理来看，大致分为两种类型：



10.4 AI处理器—计算机科学导向

从技术架构来看，大致分为四个类型：

- CPU（Central Processing Unit，中央处理器）：是超大规模集成电路，是计算机的运算和控制核心，侧重指令执行的逻辑控制
- GPU（Graphics Processing Unit，图形处理器）：采用比较简单的存储模型和数据执行流程，侧重大规模密集型数据并行计算
- ASIC/SoC（Application Specific Integrated Circuit，专用集成电路；System on Chip，系统芯片）：为特定计算任务定制的集成电路，量体裁衣，能效折中
- FPGA（Field Programmable Gate Array，现场可编程门阵列）：丰富的硬件资源和可编程性，硬件结构可根据需要实时配置灵活改变

10.4 AI处理器 — CPU

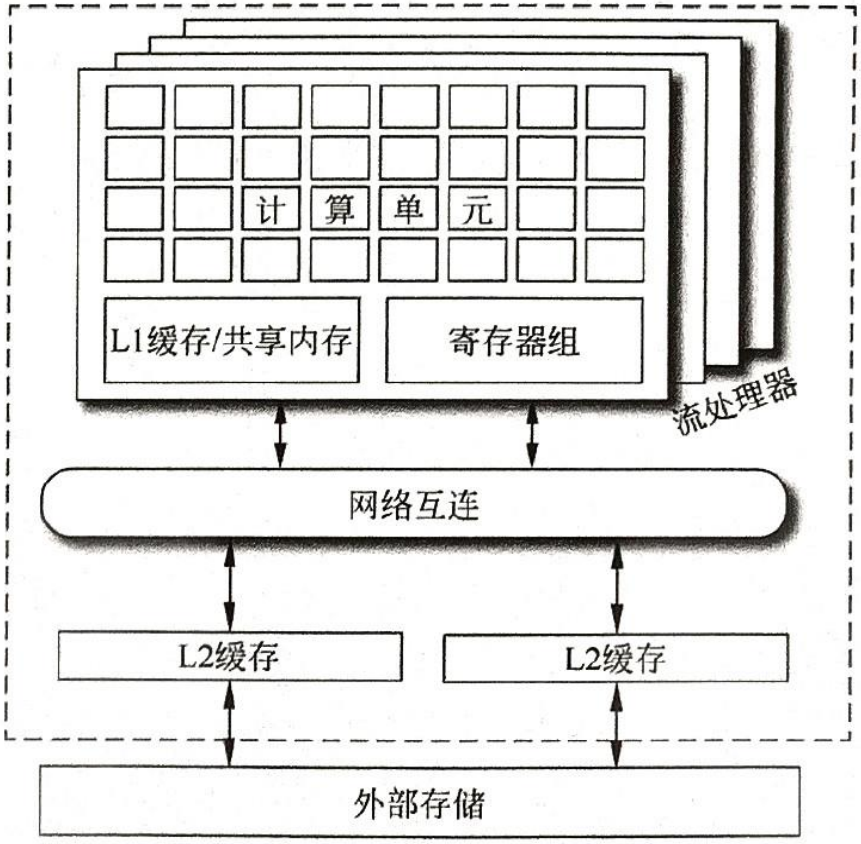
- CPU(Central Processing Unit)
  - 早期计算机性能随着摩尔定律逐年稳步提升
  - 增加核数提升性能，多核处理器更好的满足了软件对硬件的速度需求，但带来功耗和成本增加。
  - 增加指令（修改架构）的方式提升AI性能
    - Intel（CISC架构）加入AVX512等指令，在ALU计算模块加入矢量运算模块（FMA）
    - ARM（RISC架构）加入Cortex A等指令集并计划持续升级
  - 提高频率提升性能，但提升空间有限，同时高主频会导致芯片出现功耗过大和过热问题。

10.4 AI处理器 — GPU

- GPU(Graph Processing Unit)
  - GPU(Graph Processing Unit)在矩阵计算和并行计算上具有突出的性能，异构计算的主力，最早作为深度学习的加速芯片被引入AI领域，且生态成熟
  - Nvidia沿用GPU架构，对深度学习主要向两个方向发力：
    - 丰富生态：推出cuDNN针对神经网络的优化库，提升易用性并优化GPU底层架构
    - 提升定制性：增加多数据类型支持（不再只支持Float32，增加Int8等）；添加深度学习专用模块（如引入并配备张量核的改进型架构，V100的TensorCore）
  - 当前主要问题在于：成本高，能耗比低，延迟高

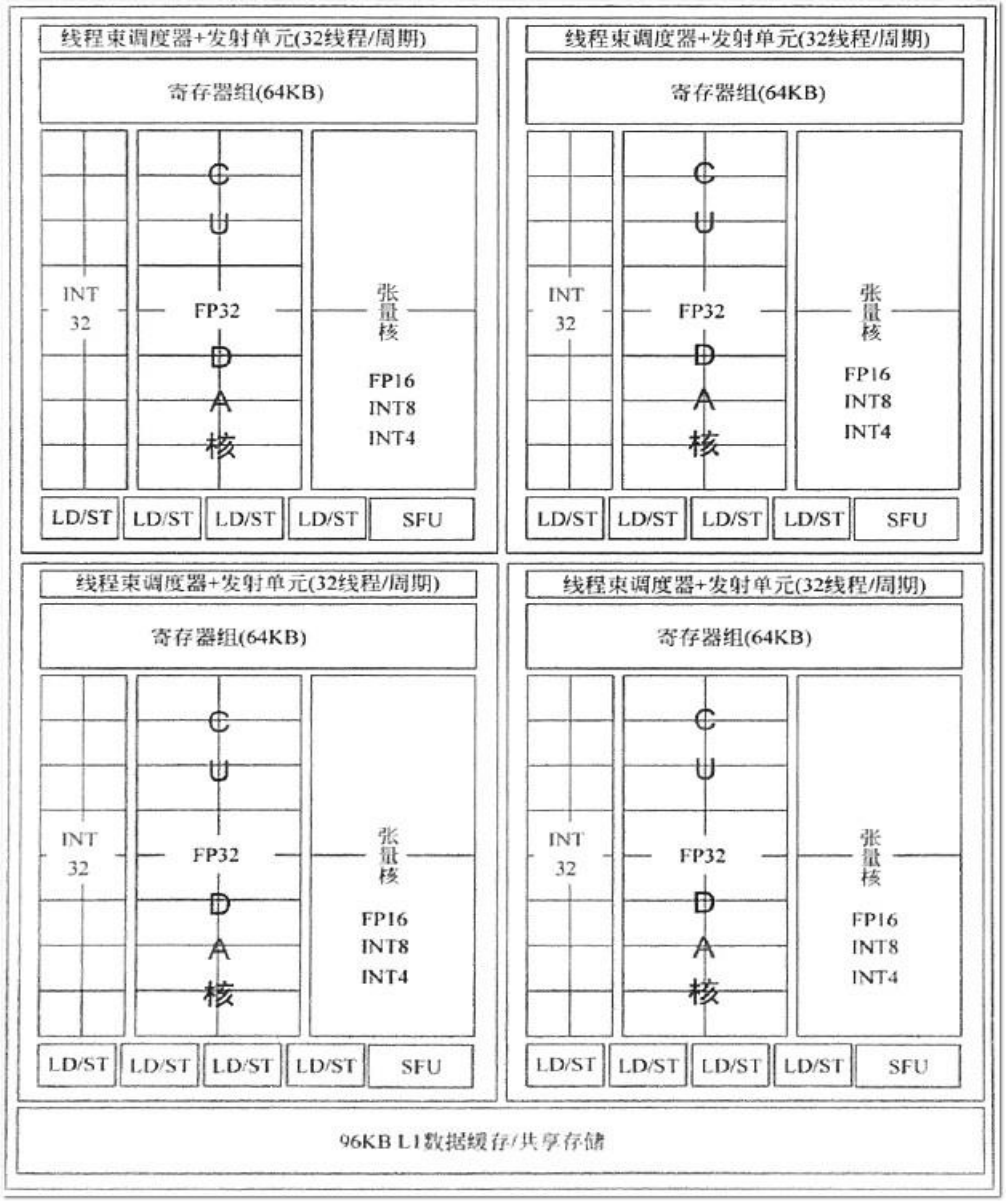
10.4 现代GPU架构

- 流处理器
- 多层级片上存储器
- 网络互连结构
- Turing架构TU102
  - 72个流处理器
  - L1缓存+共享内存+寄存器组+L2缓存+外部存储



10.4 Turing流处理器

- 64个CUDA核，负责FP32计算
- SIMT方式在CUDA框架下调用
- 8个张量核，支持FP16、INT8和INT4等多精度计算
- 通过WMMA指令在CUDA框架下调用张量核
- 256KB寄存器组、96KB的L1缓存存放数据
  - 每个线程分配一定数量的寄存器存放程序变量
  - L1缓存可配置为共享内存
  - L1缓存执行机制由内部控制



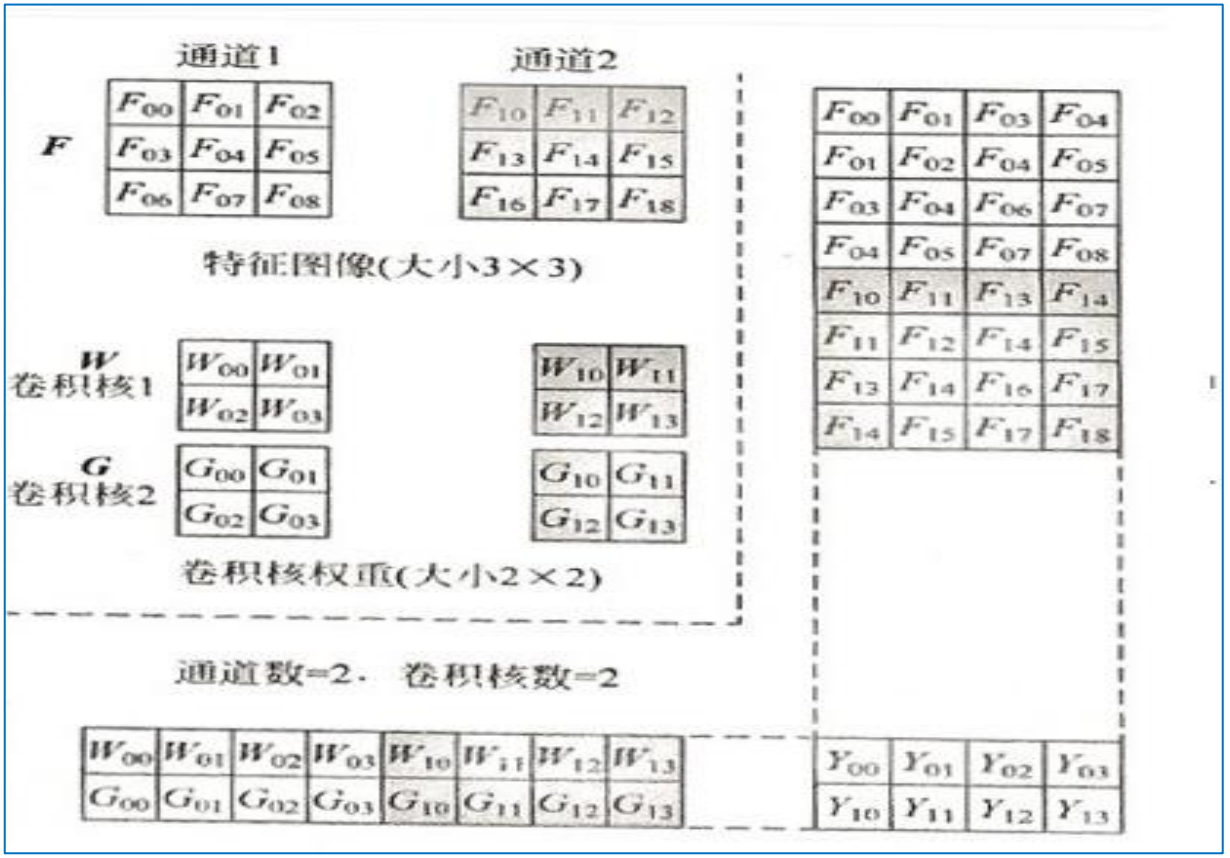
10.4 张量核Tensor Core

- 每个TC单周期内可执行64个FP16融合乘加运算(FMA)
- 每个流处理器单周期可实现512个FP16融合乘加运算(FMA)
- 每个流处理器单周期可实现2048个INT8融合乘加运算(FMA)
- WMMA指令编程TC，进入TC后分解为细粒度的HMMA指令，控制流处理器中的线程
- 每个线程在一个时钟周期内完成1次4个数的点积运算(FEDPs)
- 每个TC在一个时钟周期内完成2个4×4矩阵相乘并累加一个4×4矩阵

$$\begin{pmatrix} D_{00} & D_{01} & D_{02} & D_{03} \\ D_{10} & D_{11} & D_{12} & D_{13} \\ D_{20} & D_{21} & D_{22} & D_{23} \\ D_{30} & D_{31} & D_{32} & D_{33} \end{pmatrix} = \begin{pmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & A_{22} & A_{23} \\ A_{30} & A_{31} & A_{32} & A_{33} \end{pmatrix} \times \begin{pmatrix} B_{00} & B_{01} & B_{02} & B_{03} \\ B_{10} & B_{11} & B_{12} & B_{13} \\ B_{20} & B_{21} & B_{22} & B_{23} \\ B_{30} & B_{31} & B_{32} & B_{33} \end{pmatrix} + \begin{pmatrix} C_{00} & C_{01} & C_{02} & C_{03} \\ C_{10} & C_{11} & C_{12} & C_{13} \\ C_{20} & C_{21} & C_{22} & C_{23} \\ C_{30} & C_{31} & C_{32} & C_{33} \end{pmatrix}$$

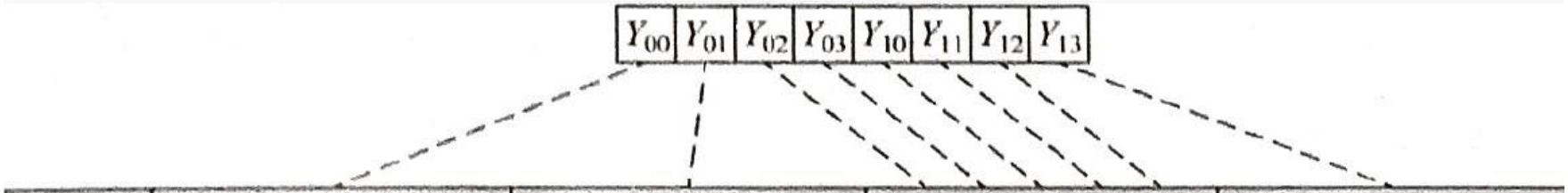
10.4 GPU加速技术

- 利用神经网络计算的数据独立性和可并行性
- GPU加速关键方式：并行化、矢量化，通用矩阵相乘GEMM
- 矩阵相乘，采用单指令多线程(SIMT)处理模式



10.4 GPU加速技术

- 每个线程在每个周期内执行1次乘加MAC运算，多个线程独立并行执行
- 每个周期：计算本次相应的两个输入点的乘法，并累加上次乘法的部分和
- 8个周期后，8个线程同时输出2×4矩阵乘法结果



周期	线程1	线程2	…	线程8
1	$PS_1=W_{00}\cdot F_{00}+0$	$PS_2=W_{00}\cdot F_{01}+0$	…	$PS_8=G_{00}\cdot F_{04}+0$
2	$PS_1=W_{01}\cdot F_{01}+PS_1$	$PS_2=W_{01}\cdot F_{02}+PS_2$	…	$PS_8=G_{01}\cdot F_{05}+PS_8$
3	$PS_1=W_{02}\cdot F_{03}+PS_1$	$PS_2=W_{02}\cdot F_{04}+PS_2$	…	$PS_8=G_{02}\cdot F_{07}+PS_8$
4	$PS_1=W_{03}\cdot F_{04}+PS_1$	$PS_2=W_{03}\cdot F_{05}+PS_2$	…	$PS_8=G_{03}\cdot F_{08}+PS_8$
5	$PS_1=W_{10}\cdot F_{10}+PS_1$	$PS_2=W_{10}\cdot F_{11}+PS_2$	…	$PS_8=G_{10}\cdot F_{14}+PS_8$
6	$PS_1=W_{11}\cdot F_{11}+PS_1$	$PS_2=W_{11}\cdot F_{12}+PS_2$	…	$PS_8=G_{11}\cdot F_{15}+PS_8$
7	$PS_1=W_{12}\cdot F_{13}+PS_1$	$PS_2=W_{12}\cdot F_{14}+PS_2$	…	$PS_8=G_{12}\cdot F_{17}+PS_8$
8	$PS_1=W_{13}\cdot F_{14}+PS_1$	$PS_2=W_{13}\cdot F_{15}+PS_2$	…	$PS_8=G_{13}\cdot F_{18}+PS_8$

10.4 GPU、CPU对比

- GPU主要面对类型高度统一、相互无依赖的大规模数据和不需打断的纯净计算环境
  - 拥有若干由数以千计的更小的核心（专为同时处理多重任务而设计）组成的大规模并行计算架构
  - 基于大吞吐量设计
    - 有很多ALU和很少cache（和CPU目的不同，为thread提高服务），缓存合并访问DRAM，带来时延问题。
    - 控制单元合并访问。
    - 大量ALU实现大量thread并行掩盖时延问题
  - 擅长计算密集和易于并行的程序

10.4 GPU、CPU对比

- CPU需要很强通用性处理不同数据类型，同时需要逻辑判断，还会引入大量分支跳转和中断处理
  - 由专为串行处理而优化的几个核心组成
  - 基于低延时设计
    - 强大的ALU单元，可在很短时钟周期完成计算
    - 大量缓存降低延时
    - 高时钟频率
    - 复杂逻辑控制单元，多分支程序可通过分支预测能力降低时延
    - 对于依赖之前指令结果的部分指令，逻辑单元决定指令在pipeline中的位置实现数据快速转发
  - 擅长逻辑控制、串行运算

10.4 CPU+GPU异构协同

CPU

◆侧重指令执行逻辑控制

◆复杂逻辑运算、多种数据类型的混合计算

GPU

◆数据并行计算

◆大规模、密集型、规整性数据计算

CPU+ GPU异构协同

CUDA(Compute Unified Device Architecture)

◆对CPU/GPU通用统一的基础软件框架

◆专用指令集架构和GPU内部的并行计算引擎

◆提供GPU硬件直接访问接口

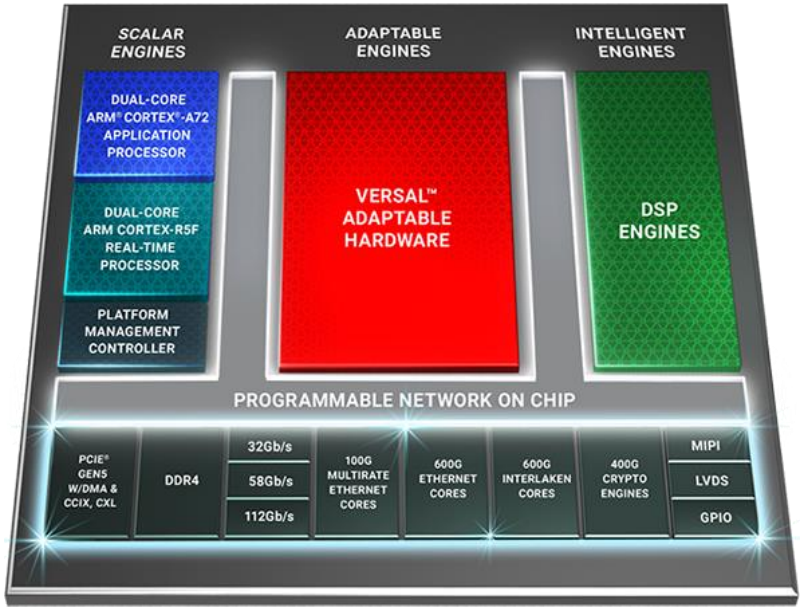
◆类C语言编程

10.4 AI处理器— FPGA

- FPGA
  - 采用HDL可编程方式，灵活性高，可重构（烧），可深度定制
  - 可通过多片FPGA联合将DNN模型加载到片上进行低延迟计算，计算性能优于GPU，但由于需考虑不断擦写，性能达不到最优（冗余晶体管和连线，相同功能逻辑电路占芯片面积更大）
  - 由于可重构，供货风险和研发风险较低，成本取决于购买数量，相对自由
  - 设计、流片过程解耦，开发周期较长（通常半年），门槛高

10.4 AI处理器现状 — FPGA

XILINX AI FPGA方案

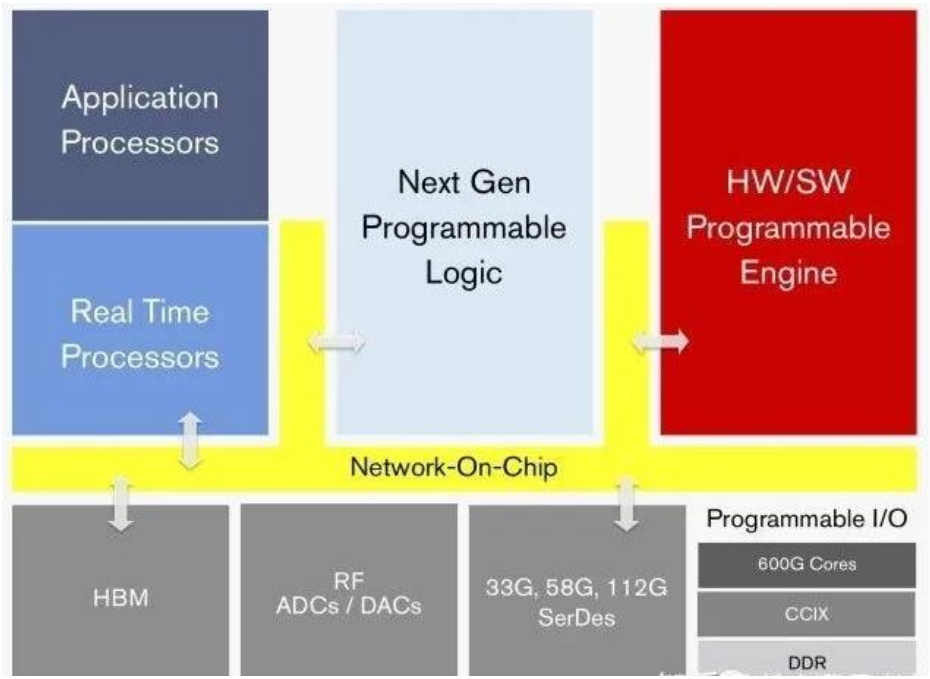


Intel(Altera) AI FPGA方案

要高效的执行AI算法，让底层的运算器件更为符合算法特征只是做了一半，甚至只是一小半工作。 “外行看计算，内行看存储”。对于算法加速，很多时候取决于存储的数据能否及时地被“送”到运算单元中，从而能保证运算单元可以满负荷运转。Xilinx和Intel也采取了不同的策略，Intel“求诸于外”而Xilinx“反求诸于内”。

10.4 自适应计算加速平台ACAP

- XILINX的ACAP(Adaptive Computation Acceleration Platform)
- FPGA向可编程逻辑门阵列转变到**动态可配置的异构加速新型器件**
- Versal： AI Core 、 Prime 、Premium 三个系列

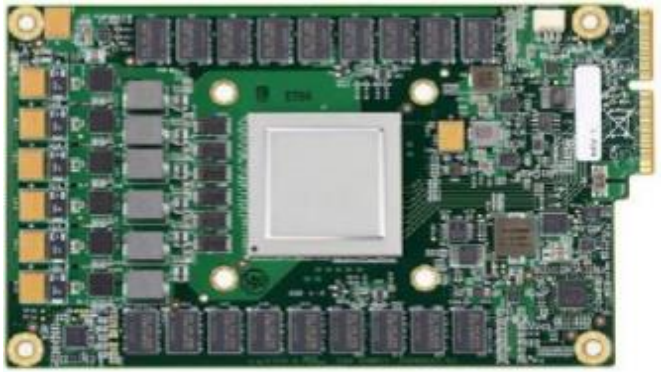


Premium框图

Xilinx 的Versal

10.4 AI处理器 — TPU

- TPU(Tensor Processing Unit)
  - 谷歌从2006年起致力于将专用集成电路ASIC的设计理念应用到神经网络领域，发布了支撑深度学习开源框架TensorFlow的人工智能定制芯片TPU(Tensor Processing Unit)
  - 利用大规模脉动阵列结合大容量片上存储来高效加速深度神经网络中最为常见的卷积运算：
    - 脉动阵列可用来优化矩阵乘法和卷积运算，以达到提供更高算力和更低能耗的作用
  - 第一代TPU 28nm，40W，700M主频
  - 设计为SATA硬盘插槽板卡



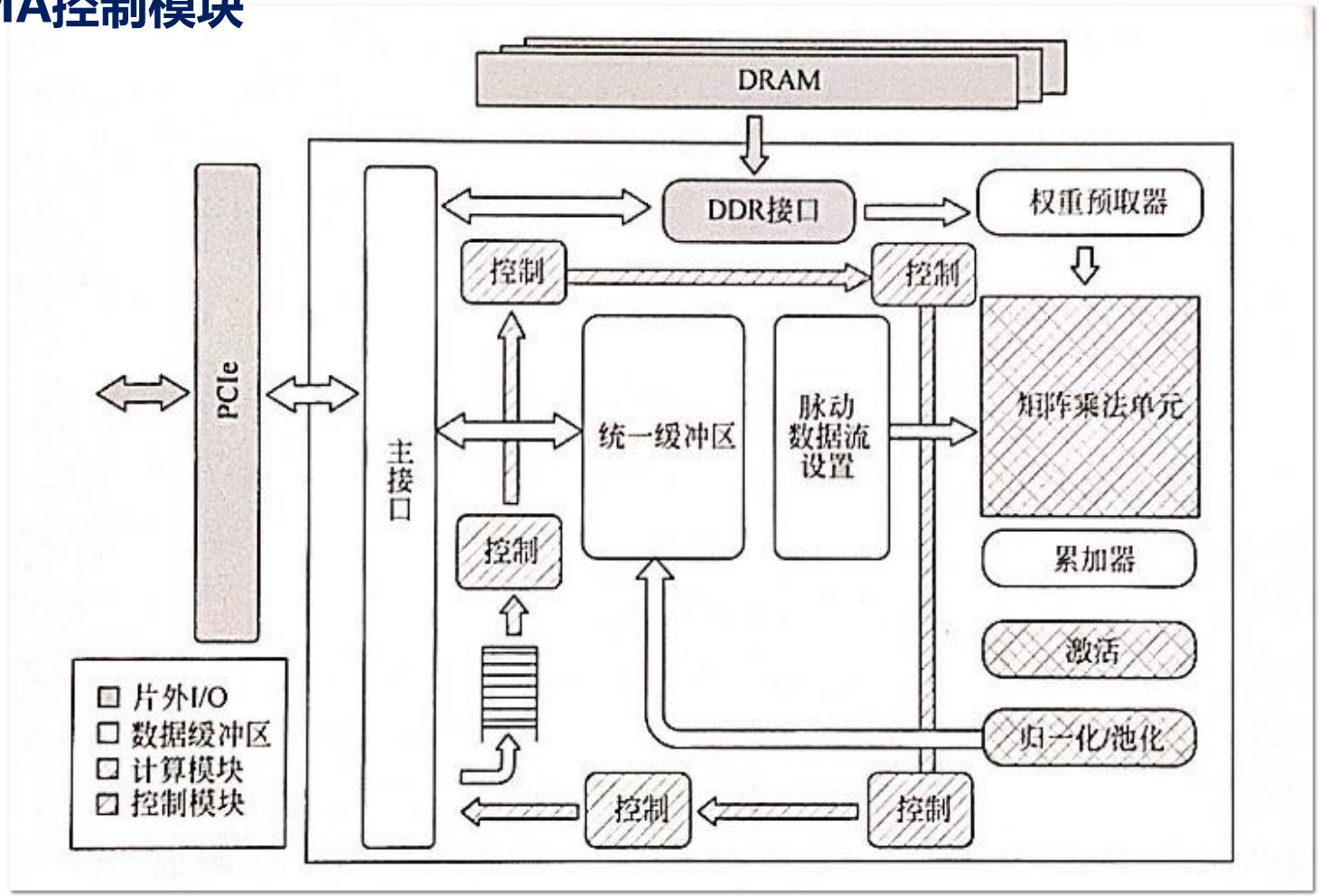
10.4 脉动阵列技术—因TPU新生

- 脉动阵列 systolic array - 因Google TPU获得新生
- Why systolic architectures?
  - Simple and regular design
  - Concurrency and communication
  - Balancing computation with I/O
- 脉动阵列的特征
  - 由多个同构的PE构成，可以是一维或二维或树的结构
  - PE功能相对简单，通过大量PE并行来提高运算效率
  - PE只能向相邻的PE发送数据，数据向“下游”流动，直到流出最末PE
- 脉动架构灵活性、扩展性较差，被发明之后并没有得到广泛应用
- DNN大量使用卷积运算和矩阵运算，这正好也是是脉动架构的优势

H. T. Kung, Why systolic architectures, IEEE Computer, 1982,15(1):37-46

10.4 谷歌TPU架构

- 包括：脉动阵列、矢量计算单元、主接口模块、队列模块、统一缓冲区、DMA控制模块



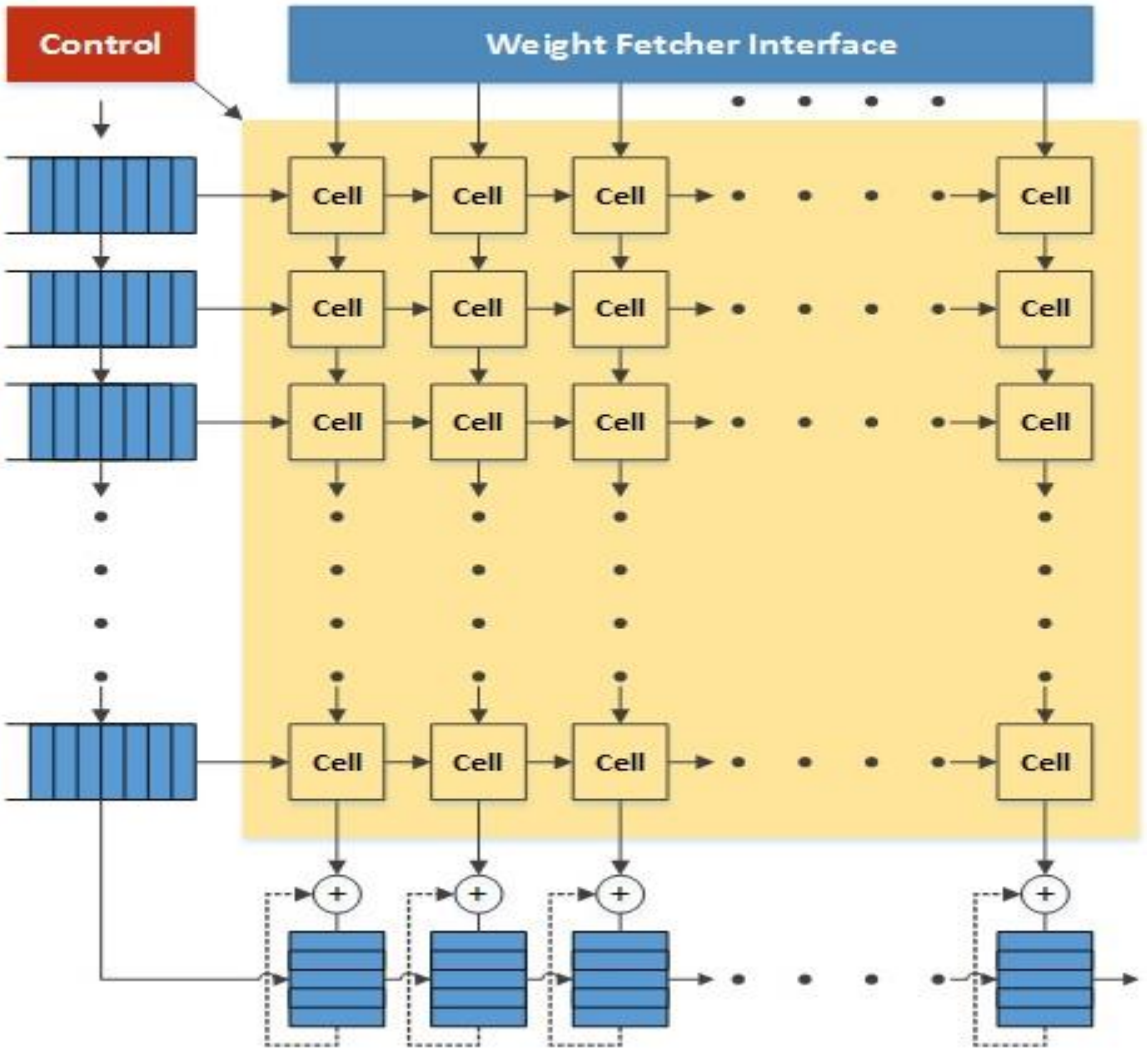
ISCA2017 In-Datacenter Performance Analysis of a Tensor Processor Unit

10.4 TPU神经网络计算指令

- TPU指令属于CISC指令集，执行的指令通过PCIe总线进入
- 多数是宏指令，实质是硬件控制状态机，大幅降低指令译码和存储的开销
- 神经计算的指令有5条
  - 数据读指令 Read_Host_Memory
  - 权重读指令 Read_Weight
  - 矩阵运算指令 MatrixMultiply/Convolve, Activate
  - 数据写回指令 Write_Host_Memory
- 指令格式占12位：统一缓冲区地址3位、累加器缓冲区地址2位，操作数长度4位，操作码和标志位3位

10.4 谷歌TPU脉动阵列架构

- 256×256 阵列高效完成卷积或者矩阵计算



10.4 AI处理器—华为昇腾（Ascend）

- 华为自研的硬件架构“达芬奇架构”，专门针对DNN计算特性量身定做，高性能3D Cube矩阵计算单元为基础，实现算力和能效比的大幅提升
- 达芬奇架构指令采用CISC，包含高效的神经网络计算特殊专用指令

Ascend 310 —面向推理

- Ascend 310
- 架构: 达芬奇
- 半精度 (FP16): 8 Tera-FLOPS
- 整数精度 (INT8) : 16 Tera-OPS
- 16 通道 全高清 视频解码器 – H.264/265;
- 1 通道 全高清 视频编码器 – H.264/265;
- 最大功耗: 8W
- 12nm

Ascend 910 —面向训练

- Ascend 910
- 架构: 达芬奇
- 半精度 (FP16): 256 Tera-FLOPS
- 整数精度 (INT8) : 512 Tera-OPS
- 128 通道 全高清 视频解码器 – H.264/265;
- 最大功耗: 350W
- 7nm

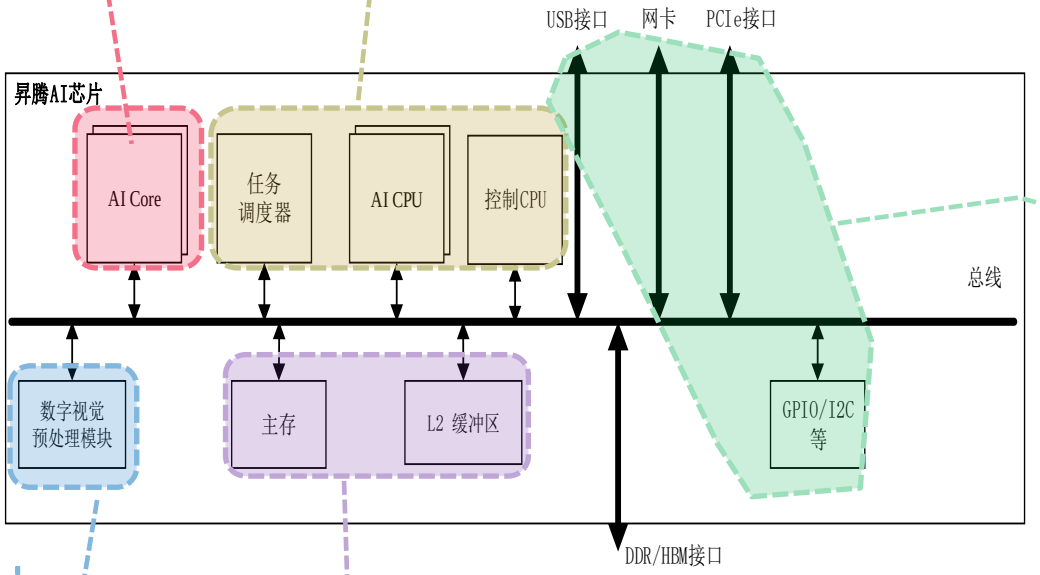
10.4 Ascend 310处理器整体架构

AI Core

昇腾AI芯片的计算核心，负责执行矩阵、向量、标量计算密集的算子任务，采用达芬奇架构。 Ascend 310集成了2个AI Core

ARM CPU核心

集成了8个ARM A55，一部分部署为AI CPU，执行不适合跑在AI Core上的算子（承担非矩阵类复杂计算）；一部分部署为专用于控制芯片整体运行的控制CPU。两类任务占用的CPU核数可由软件根据系统实际运行情况动态分配。还部署了一个专用CPU作为任务调度器（TS），实现计算任务在AI Core上的高效分配和调度；该CPU专门服务于AI Core和AI CPU，不承担任何其他的事务和工作。



对外接口

支持PCIE3.0、RGMII、USB3.0等高速接口、以及GPIO、UART、I2C、SPI等低速接口

DVPP

数字视觉预处理子系统，完成图像视频的编解码。用于将从网络或终端设备获得的视觉数据，进行预处理以实现格式和精度转换等要求，之后提供给AI计算引擎

Cache & Buffer

SoC片内有层次化的memory结构，AI core内部有两级memory buffer，SOC片上还有8MB L2 buffer，专用于AI Core、AI CPU，提供高带宽、低延迟的memory访问，集成了LPDDR4x控制器，提供更大容量的DDR内存

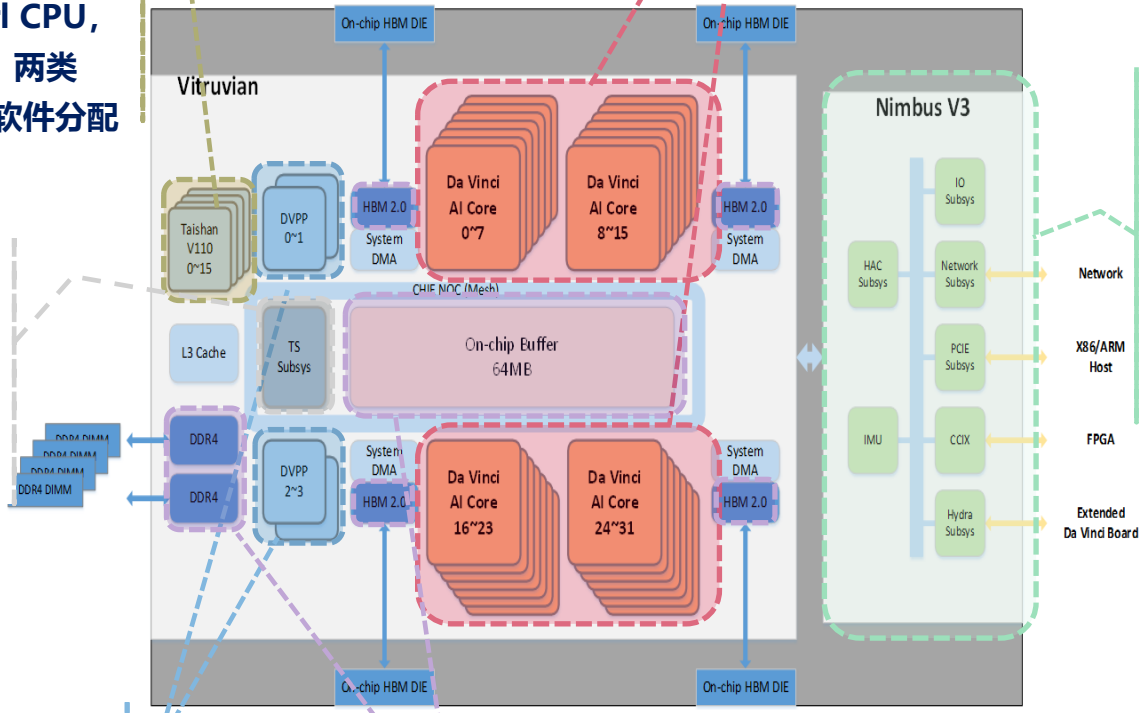
10.4 Ascend 910处理器整体架构

CPU子系统

集成16个TaishanV110 Core（4个构成一个Cluster）。部分部署为AI CPU，承担部分AI计算功能；部分部署为Ctrl CPU，负责整SoC的控制功能。两类CPU占用的CPU核数由软件分配

AI Core

昇腾AI芯片的计算核心，主要负责执行矩阵、向量计算密集的算子任务，采用达芬奇架构。Ascend 910集成了32个AI Core。



Nimbus

提供x16 PCIe 4.0接口，和Host CPU对接，100G NIC（支持ROCE V2协议）用于跨服务器传递数据；集成1个A53 CPU核，执行启动、功耗控制等管理任务

TS CPU

独立的4核A55 Cluster负责任务调度，算子任务切分之后，通过硬件调度器（HWTS），分发给AI Core或AI CPU

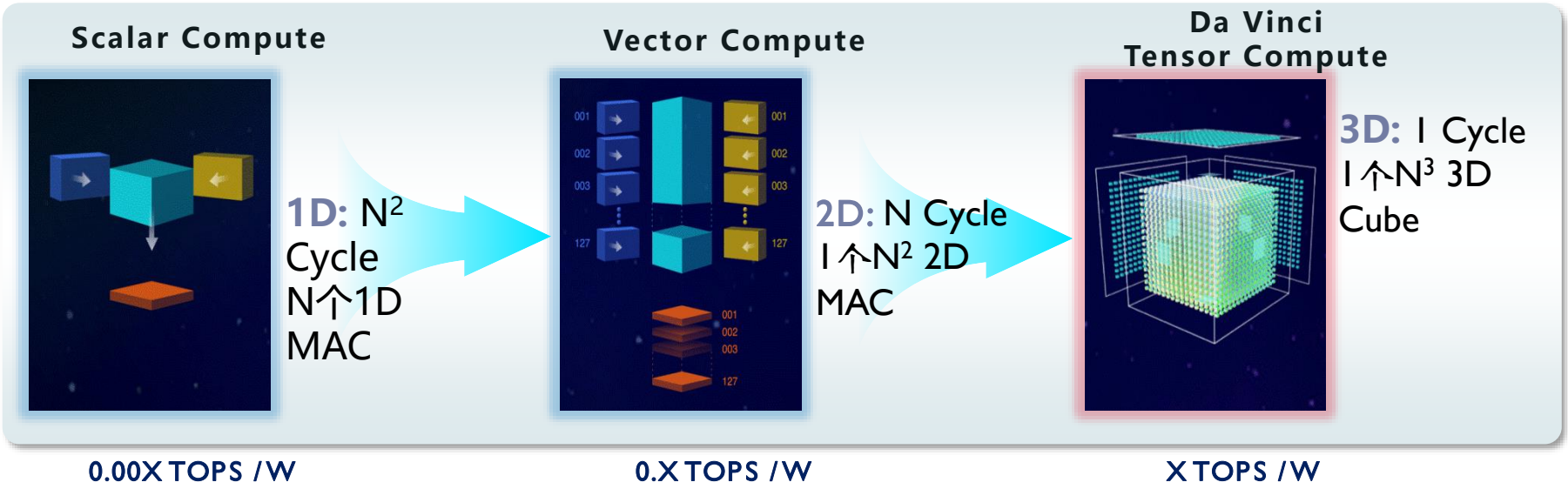
DVPP

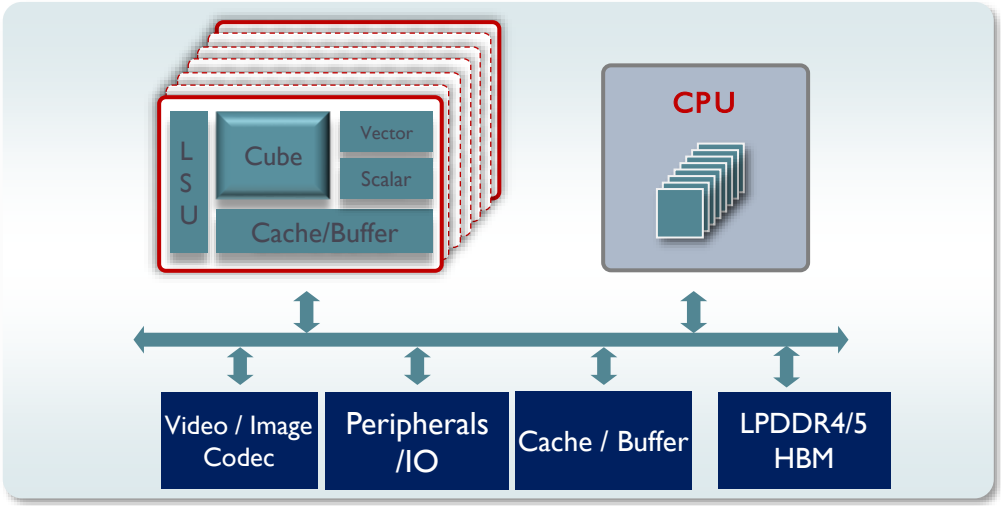
数字视觉预处理子系统，图像视频编解码等预处理操作

Cache & Buffer

片内有层次化的memory结构，AI Core内部有两级memory buffer，还有64MB L2 buffer，专用于AI Core、AI CPU，提供高带宽、低延迟的访存。Virtruvian连接4个HBM 2.0颗粒，总计32GB，还集成DDR 4.0控制器，提供DDR内存

10.4 昇腾Da Vinci AI技术架构

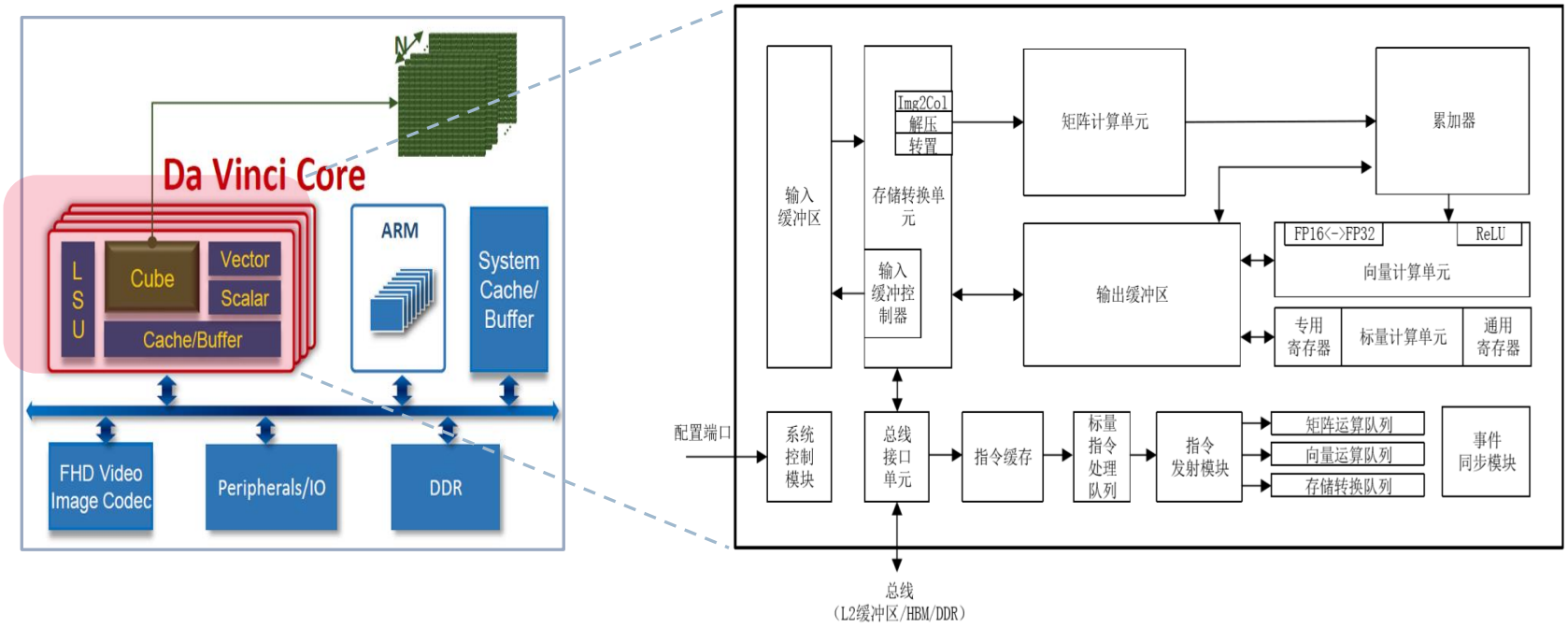




3D Cube:16^3三维弹性立方体

- **高算力：可在一个时钟周期内完成4096个FP16 MAC 运算**
- **高效：支持几十毫瓦IP到几百瓦芯片，适应端、边和云的平滑架构扩展**

10.4 AI Core

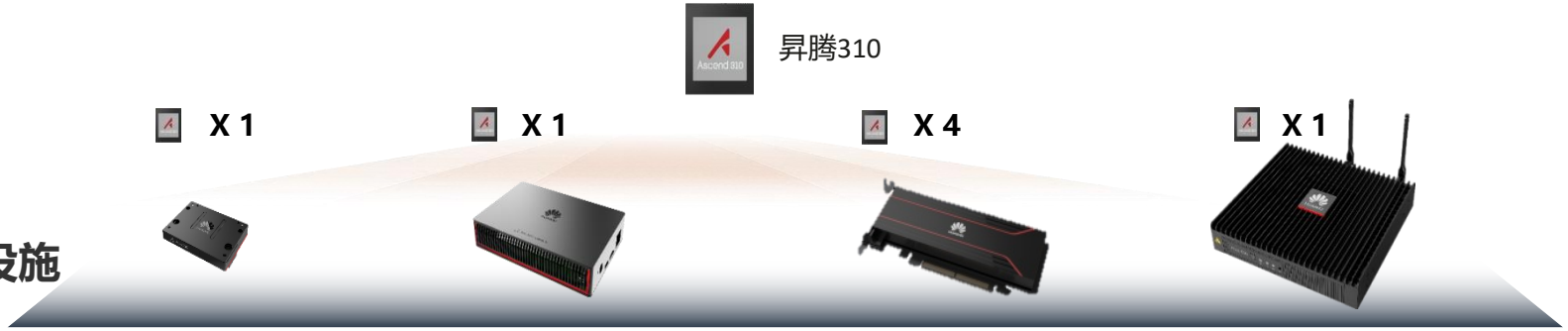


- 计算单元：矩阵、向量、标量
- 存储系统： AI Core片上存储和相应的数据通路构成了存储系统
- 控制单元：整个计算过程提供了指令控制，负责整个AI Core的运行

- AI Core是昇腾AI处理器的计算核心，采用华为自研的达芬奇架构
- Ascend310/910，AI Core里的计算、存储和带宽资源有不同的规格

10.4 基于昇腾AI处理器的产品形态

AI基础设施



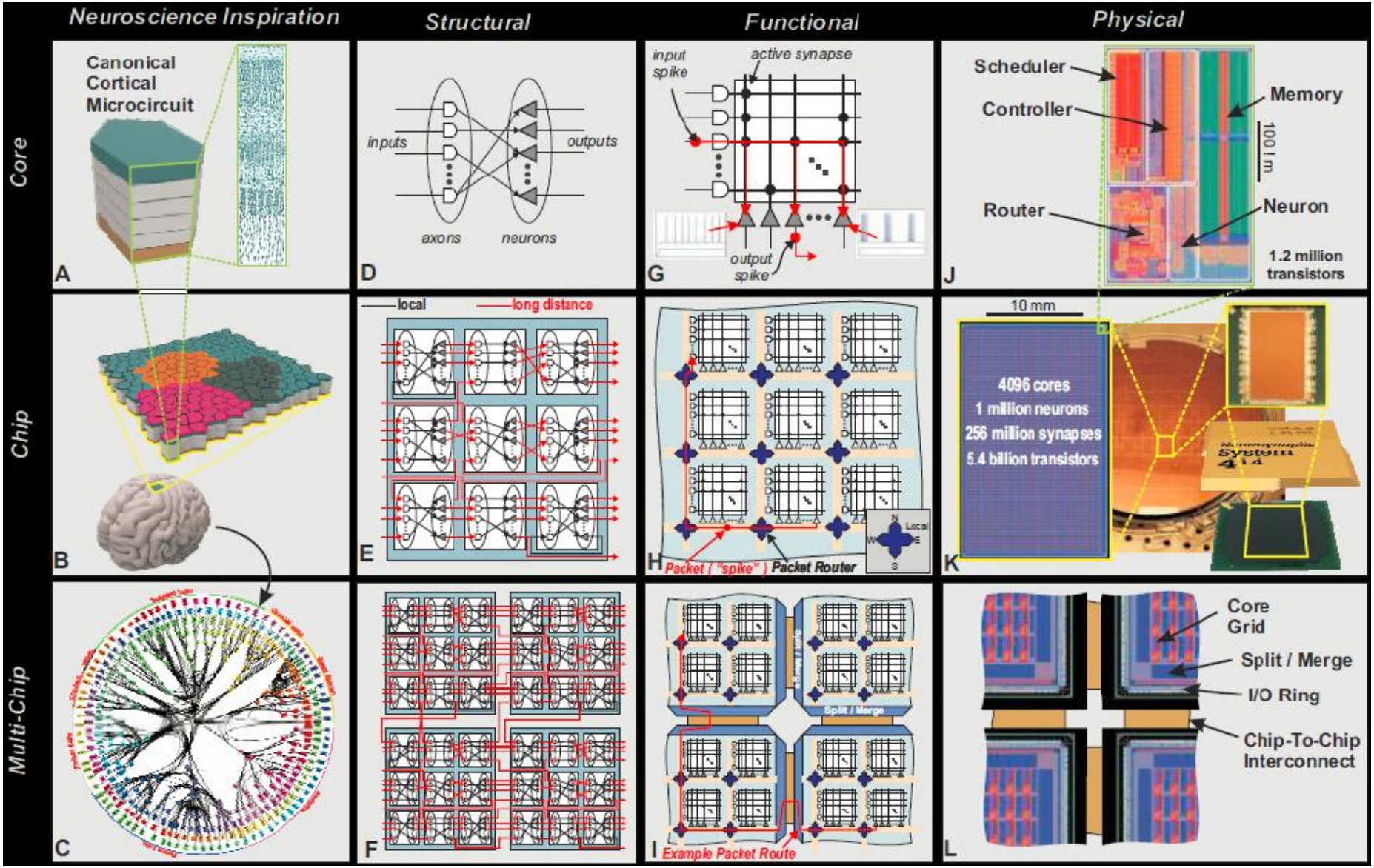
Atlas 200(加速模块)	Atlas 200DK(开发套件)	Atlas 300(PCIE加速卡)	Atlas 500(智能小站)
• 16TOPS INT8 @ 9.5W • 16路高清视频实时分析 • 4GB/8GB内存 | PCIe 3.0 x 4接口 • 工作温度：-25℃~+80℃ • 尺寸：52 x 38 x 10.2 mm	• 16TOPS INT8 @ 24W • 1*USB type-C | 2* CCM接口 | 1*GE网口 | 1* SD卡插槽 • 8GB内存 • 工作温度：0℃~45℃ • 尺寸：125 × 80 × 24 mm	• 64TOPS INT8 @ 67W • 64路高清视频实时分析 • 32GB内存 | 204.8GB/s内存带宽 • PCIe 3.0 x16，半高半长卡	• 16 TOPS INT8 @ 25-40W • 支持WiFi & LTE • 16路高清视频实时分析 • 无风扇设计 | –40°C 至 +70°C | TEC散热技术

10.4 基于昇腾AI处理器的产品形态

10.4 神经科学AI处理器—TrueNorth

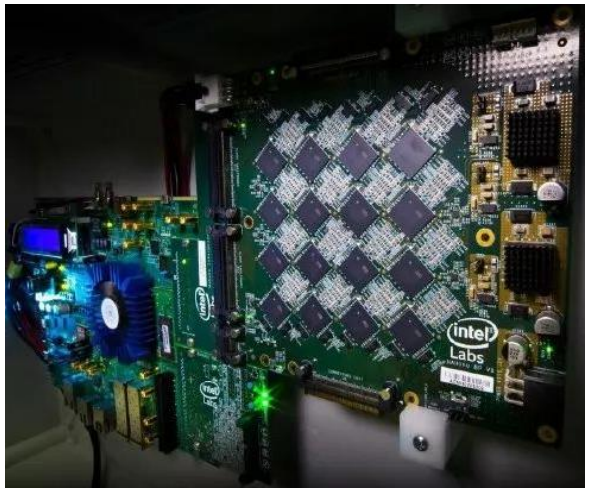
- IBM 公司 2014 年公布类脑芯片 TrueNorth， 是 IBM 参与 DARPA 的研究项目 SyNapse 的成果
- SyNapse （Systems of Neuromorphic Adaptive Plastic Scalable Electronics）（自适应可塑可伸缩电子神经系统，SyNapse也正好是突触），目标是开发出打破冯•诺依曼体系
- 数字处理器当作神经元，内存作为突触，内存、CPU和通信部件完全集成在一起， 神经元之间可以快捷相互通信，只要接收到其他神经元发送的脉冲（动作电位），这些神经元就会同时做动作
- TrueNorth 尺寸与邮票相当，
  集成54亿个硅晶体管、内置4096个内核、100万个“神经元”
  2.56亿个“突触” 功耗65mW

10.4 神经科学AI处理器—TrueNorth



10.4 神经科学AI处理器—Loihi

- Intel公司2017 年发布类脑芯片 Loihi
- 采用异构设计， 14nm工艺，128个神经形态核心+3个X86核心，集成13万个神经元、1.3亿个触突
- 神经拟态计算方式，异步电路，不需要全局时钟信号，使用异步脉冲神经网络(SNN)
- Pohoiki Beach神经拟态系统，包含64颗Loihi芯片，集成1320亿晶体管，总面积3840平方毫米，800万个神经元、80亿个突触



10.4 天机类脑芯片（清华）

《自然》总编斯基珀博士2010接受新华社专访指出：

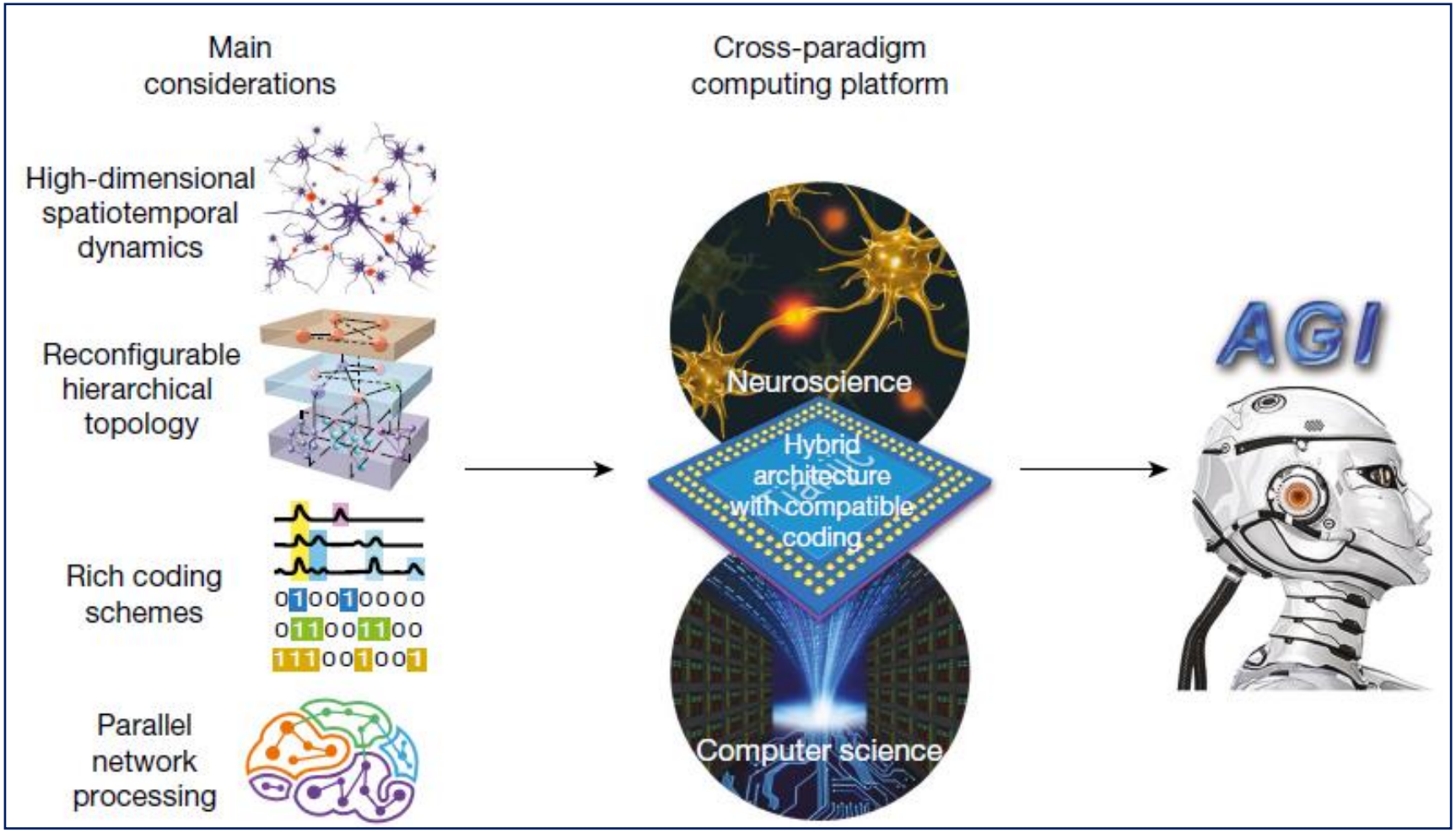
“清华天机芯片的论文将人工智能中的计算机科学研究与神经科学研究结合起来，是人工智能领域的重要里程碑。”

清华大学类脑中心施路平教授团队

- 2015，第一代“天机芯”110nm, DEMO
- 2017，第二代“天机芯”28nm
- 2019，第三代“天机芯”登上《自然》封面
- 支持ANN和SNN的类脑芯片

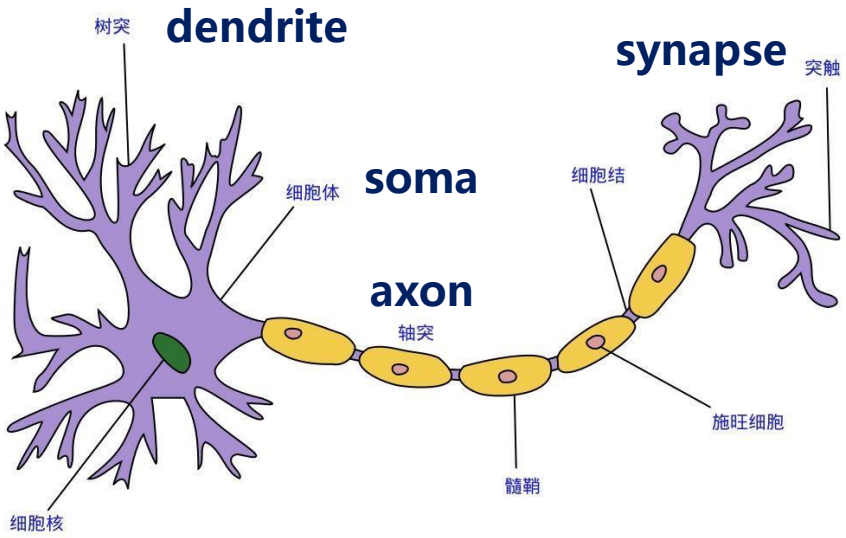
J. Pei, et al. Towards artificial general intelligence with hybrid Tianjic chip architecture. Nature, vol. 572: 106-111, 2019

10.4 天机类脑芯片（清华）

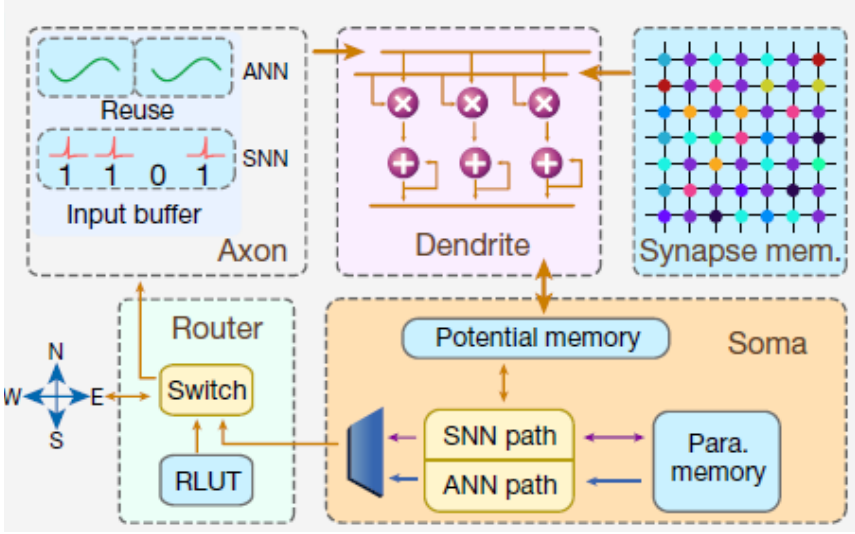


AGI计算的神经科学-计算机科学交叉范式

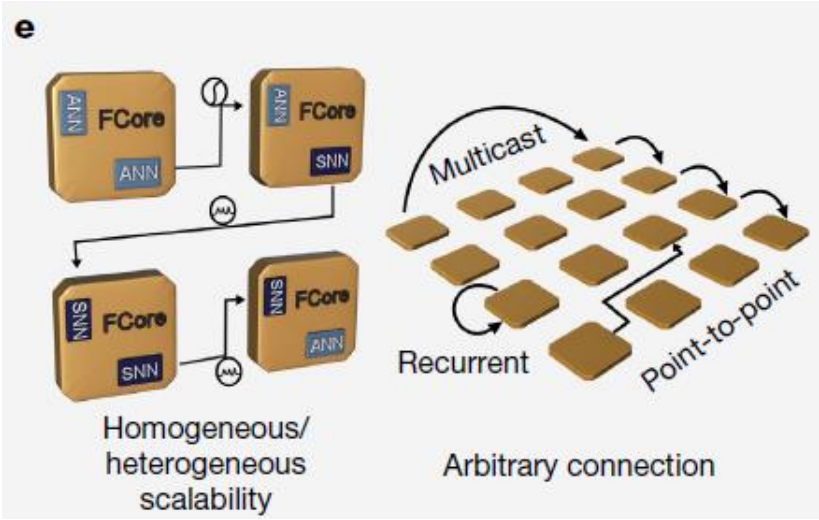
10.4 天机类脑芯片（清华）



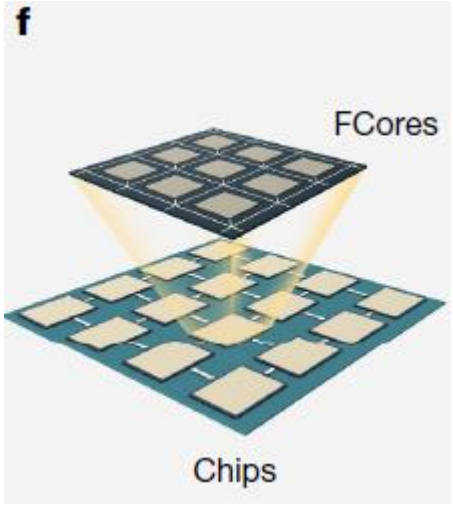
生物神经元结构

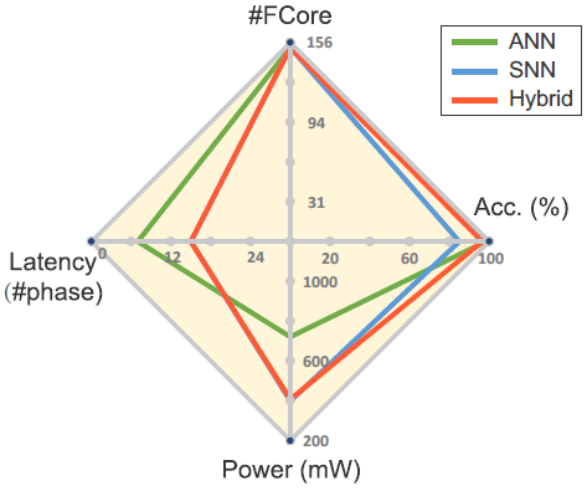


统一功能核FCore



FCore灵活配置和互联拓扑





芯片级2D网格

10.4 天机类脑芯片（清华）