推理引擎-Kernel优化

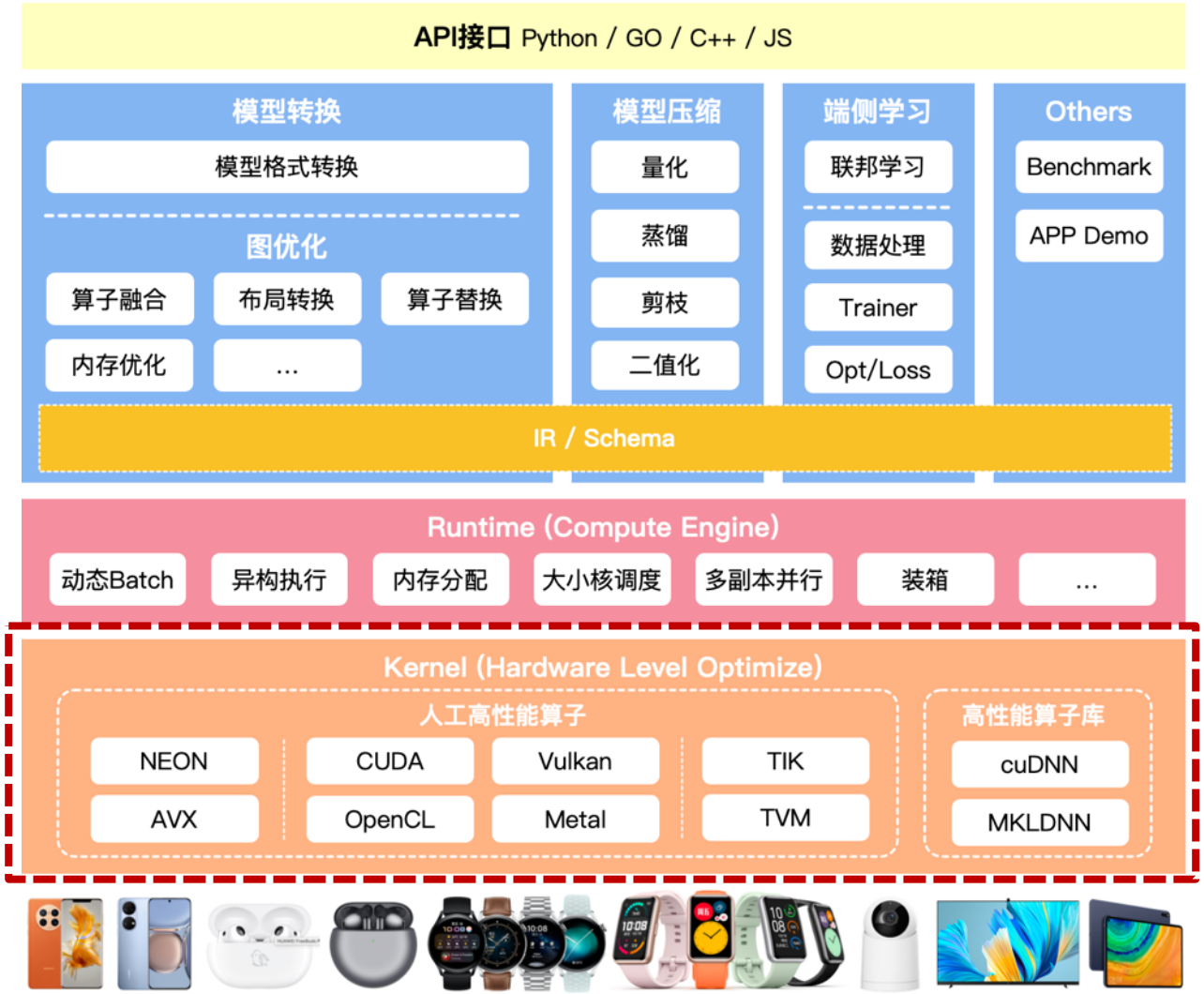
卷积优化Im2Col

ZOMI

Talk Overview

1. **推理系统介绍**：推理系统架构 – 推理引擎架构
2. **模型小型化**：CNN小型化结构 – Transform小型化结构
3. **离线优化压缩**：低比特量化 – 模型剪枝 – 知识蒸馏
4. **模型转换与优化**：模型转换细节 - 计算图优化
5. **Kernel 优化**
   - 算法优化 (Winograd / Strassen)
   - 内存布局 (NC1HWC0 / NCHW4)
   - 汇编优化 (指令与汇编)
   - 调度优化
6. **Runtime 优化**

推理引擎架构



高性能算子层

- 算子优化
- 算子执行
- 算子调度

Talk Overview

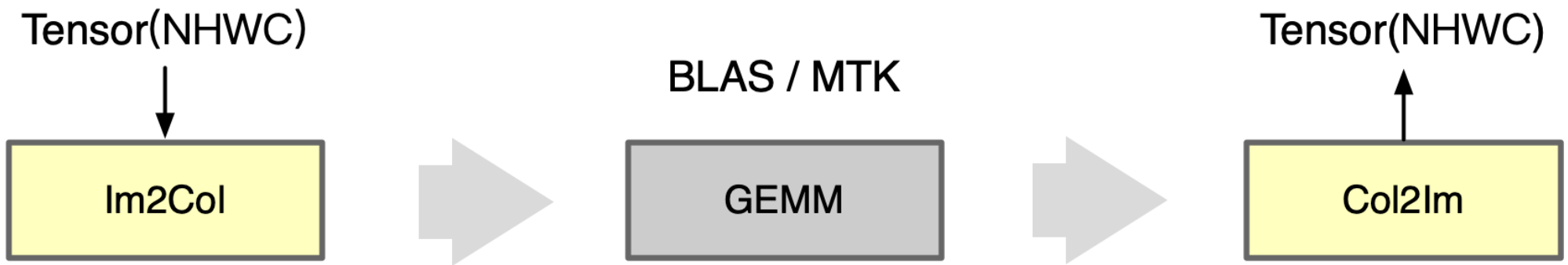
Conv Kernel 优化

- What is Convolution - 卷积的概念
- Im2Col Optimizer - Im2Col 优化算法
- Spatial Pack Optimizer – 空间组合优化
- Winograd Optimizer – Winograd 优化算法
- Indirect Algorithm – QNNPACK 间接卷积优化

Img2col算法

Img2col 介绍

- 作为早期的深度学习框架，Caffe 中卷积的实现采用的是基于 im2col 的方法，至今仍是卷积重要的优化方法之一。
- Im2col 是计算机视觉领域中将图片转换成矩阵的矩阵列（column）的计算过程。由于二维卷积的计算比较复杂不易优化，因此在 AI 框架早期，Caffe 使用 Im2col 方法将三维张量转换为二维矩阵，从而充分利用已经优化好的 GEMM 库来为各个平台加速卷积计算。最后，再将矩阵乘得到的二维矩阵结果使用 Col2Im 将转换为三维矩阵输出。



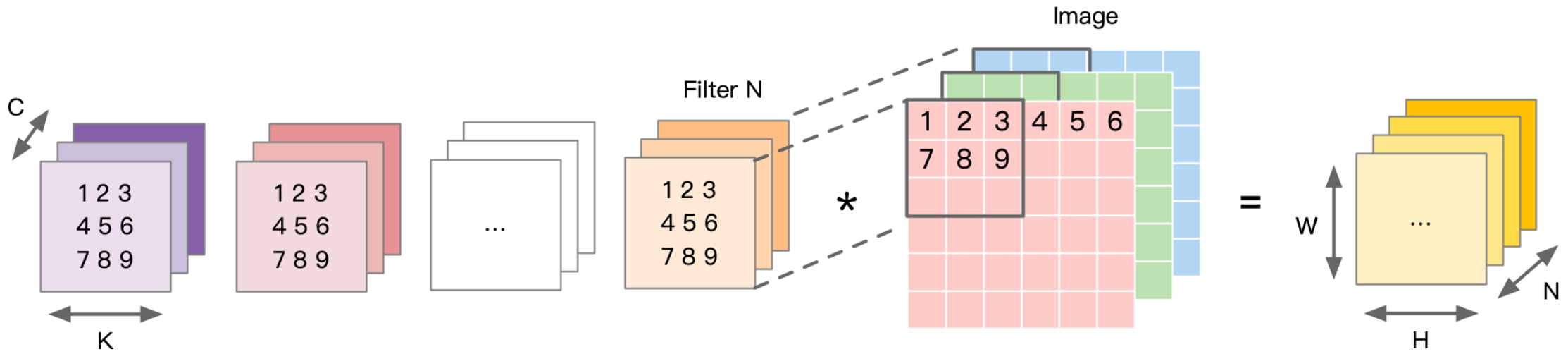
Img2col 算法过程

Im2col+Matmul 方法主要包括两个步骤：

1. 使用 Im2col 将输入矩阵展开一个大矩阵，矩阵每一列表示卷积核需要的一个输入数据。
2. 使用上面转换的矩阵进行 Matmul 运算，得到的数据就是最终卷积计算的结果。

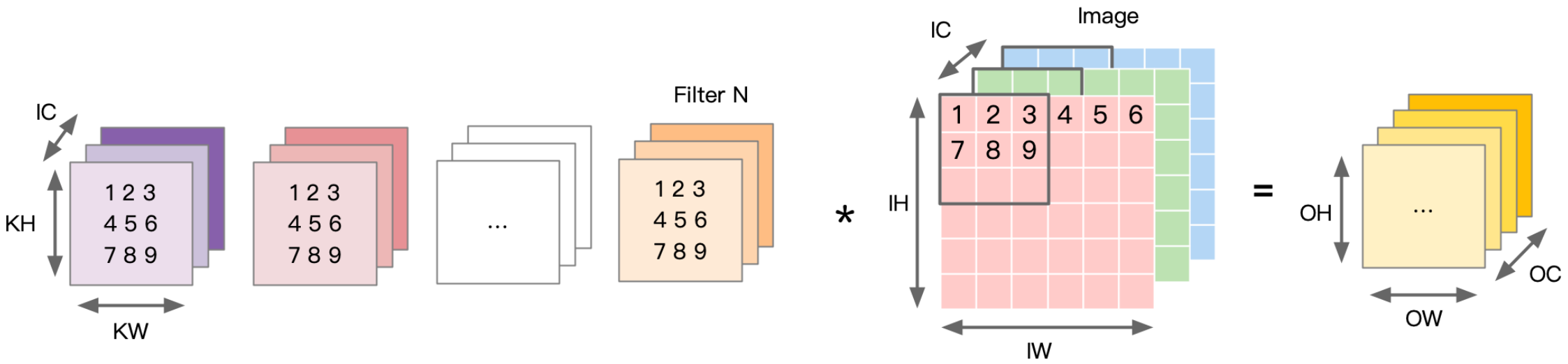
卷积通用过程

- 图像卷积正常的三通道卷积，输入维度为3维（H, W, 3），卷积核维度为（N, C, KH, KW），输出维度为（N, H, W）。卷积的一般计算方式为：



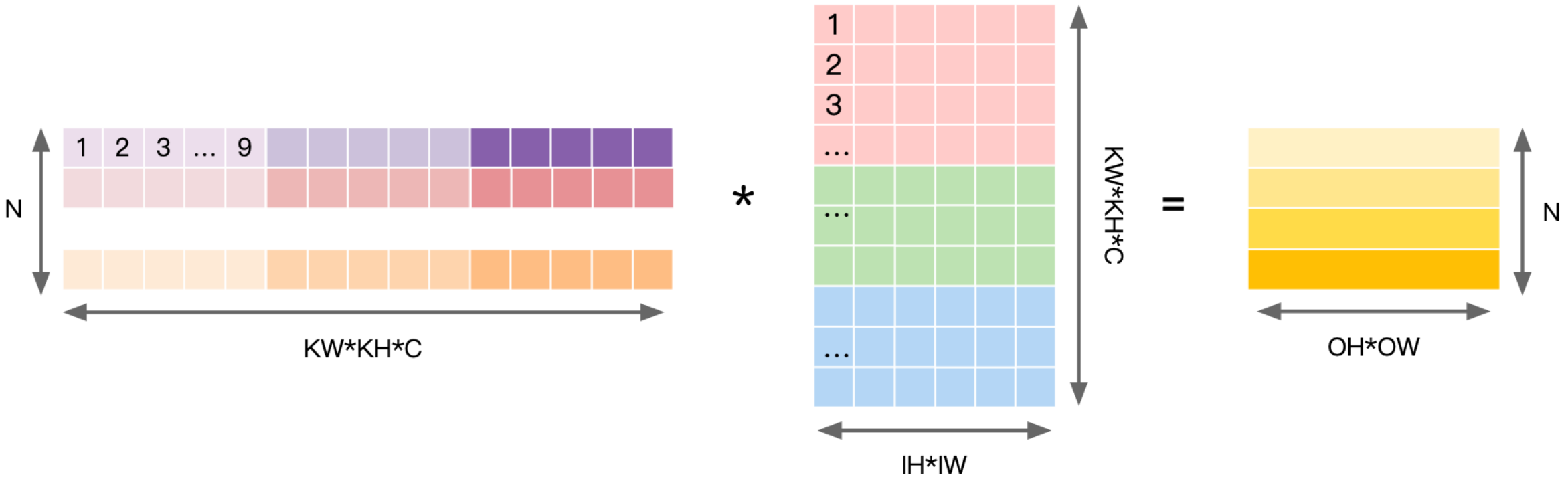
卷积通用过程

- 卷积默认采用数据排布方式为NHWC，输入维度为4维（N,IH,IW,IC），卷积核维度为（OC,KH,KW,IC），输出维度为（N,OH,OW,OC）。卷积的一般计算方式为：



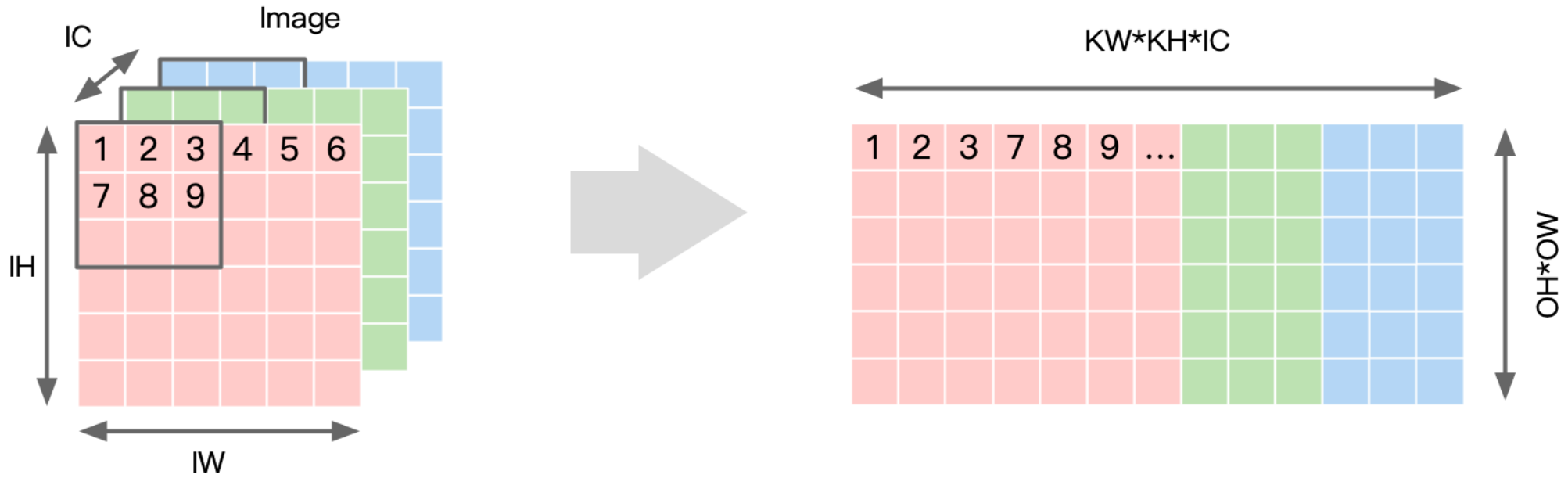
Img2col 算法过程

- 卷积操作转换为矩阵相乘，对 Kernel 和 Input 进行重新排列。将输入数据按照卷积窗进行展开并存储在矩阵的列中，多个输入通道的对应的窗展开之后将拼接成最终输出 Matrix 的一列：



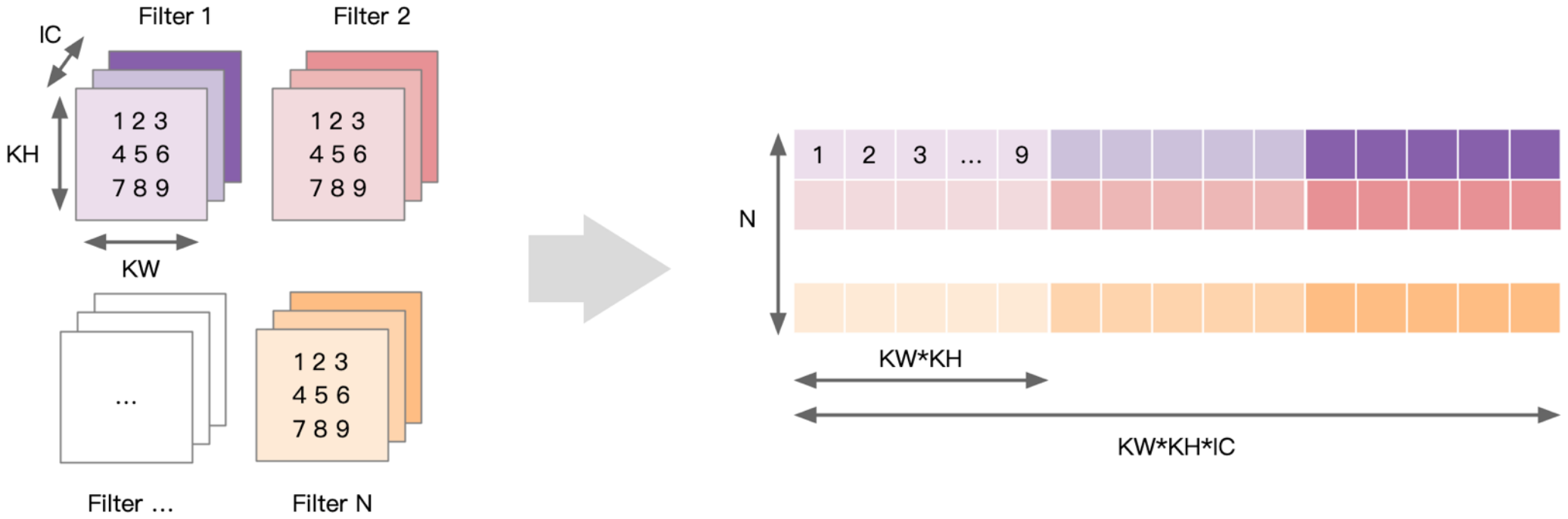
Img2col 算法过程

- 对 Input 进行重排，得到的矩阵见右侧，行数对应输出 OH*OW 个数；每个行向量里，先排列计算一个输出点所需要输入上第一个通道的 KH*KW 个数据，再按次序排列之后通道，直到通道 IC。



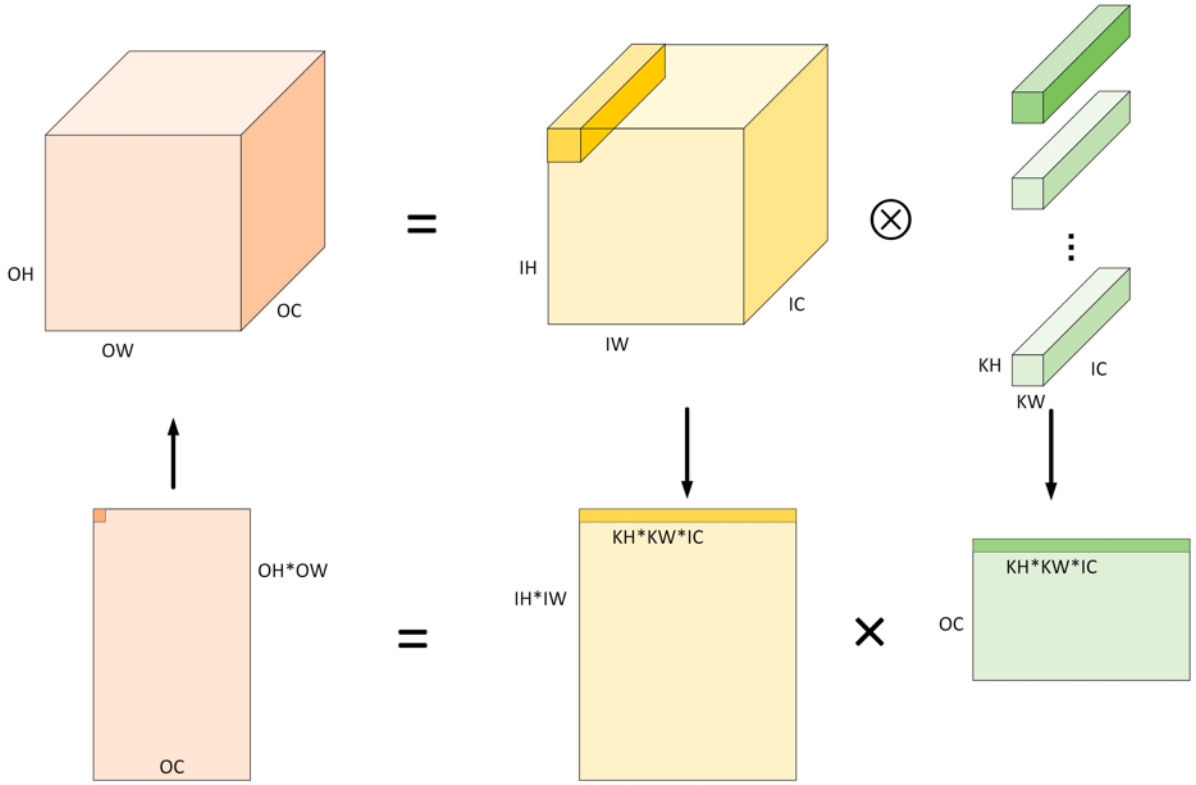
Img2col 算法过程

- 对权重数据进行重排，即以卷积的 stride 为步长展开后续卷积窗并存在 Matrix 下一列。将 N 个卷积核展开为权重矩阵的一行，因此共有 N 行，每个行向量上先排列第一个输入通道上 KH*KW 数据，再依次排列后面的通道直到 IC。



Img2col 算法过程

- 通过数据重排，完成 Im2col 的操作之后会得到一个输入矩阵，卷积的 Weights 也可以转换为一个矩阵，卷积的计算就可以转换为两个矩阵相乘的求解，得到最终的卷积计算结果。



Img2col 算法流程

Im2col 算法计算卷积的过程，具体过程如下（简单起见忽略 Padding 的情况，即认为 OH=IH, OW=IW）：

1. 将输入由 N×IH×IW×IC 根据卷积计算特性展开成 (OH×OW)×(N×KH×KW×IC) 形状二维矩阵。显然，转换后使用的内存空间相比原始输入多约 KH*KW−1 倍；
2. 权重形状一般为 OC×KH×KW×IC 四维张量，可以将其直接作为形状为 (OC)×(KH×KW×IC) 的二维矩阵处理；
3. 对于准备好的两个二维矩阵，将 (KH×KW×IC) 作为累加求和的维度，运行矩阵乘可以得到输出矩阵 (OH×OW)×(OC)；
4. 将输出矩阵 (OH×OW)×(OC) 在内存布局视角即为预期的输出张量 N×OH×OW×OC，或者使用 Col2Im 算法变为下一个算子输入 N×OH×OW×OC

Img2col 总结

- Im2col 计算卷积使用 GEMM 的代价是额外的内存开销。使用 Im2col 将三维张量展开成二维矩阵时，原本可以复用的数据平坦地分布到矩阵中，将输入数据复制了 KH*KW−1 份。
- 转化成矩阵后，可以在连续内存和缓存上操作，而且有很多库提供了高效的实现方法（BLAS、MKL），Numpy 内部基于 MKL 实现运算的加速。

- **在实际实现时，离线转换模块实现的时候可以预先对权重数据执行 Im2col 操作，Input Data Im2col和GEMM的数据重排会同时进行，以节省运行时间。**

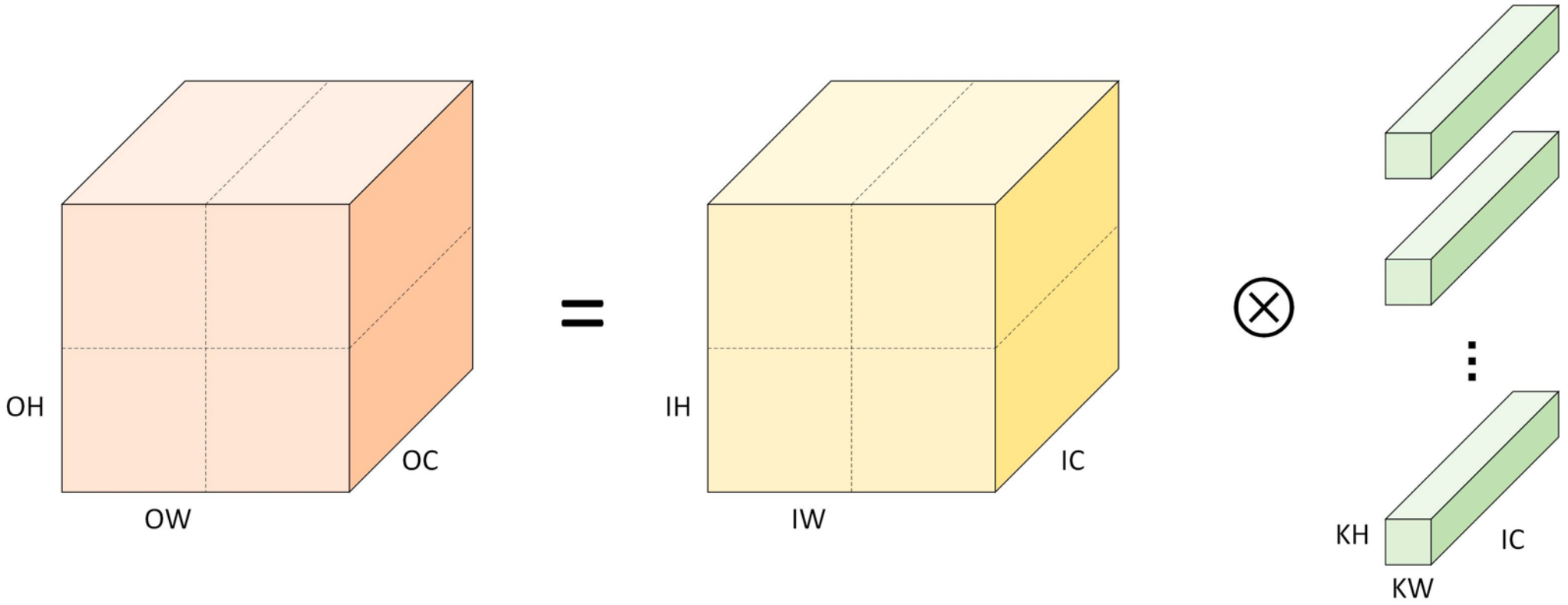
空间组合优化

Img2col 到空间组合优化

- Im2col 是一种比较朴素的卷积优化算法，在没有精心处理的情况下会带来较大的内存开销。空间组合（Spatial pack）是一种类似矩阵乘中重组内存的优化算法。
- **空间组合优化算法：**是一种基于分治法（Divide and Conquer）的方法，基于空间特性，将卷积计算划分为若干份，分别处理。
-

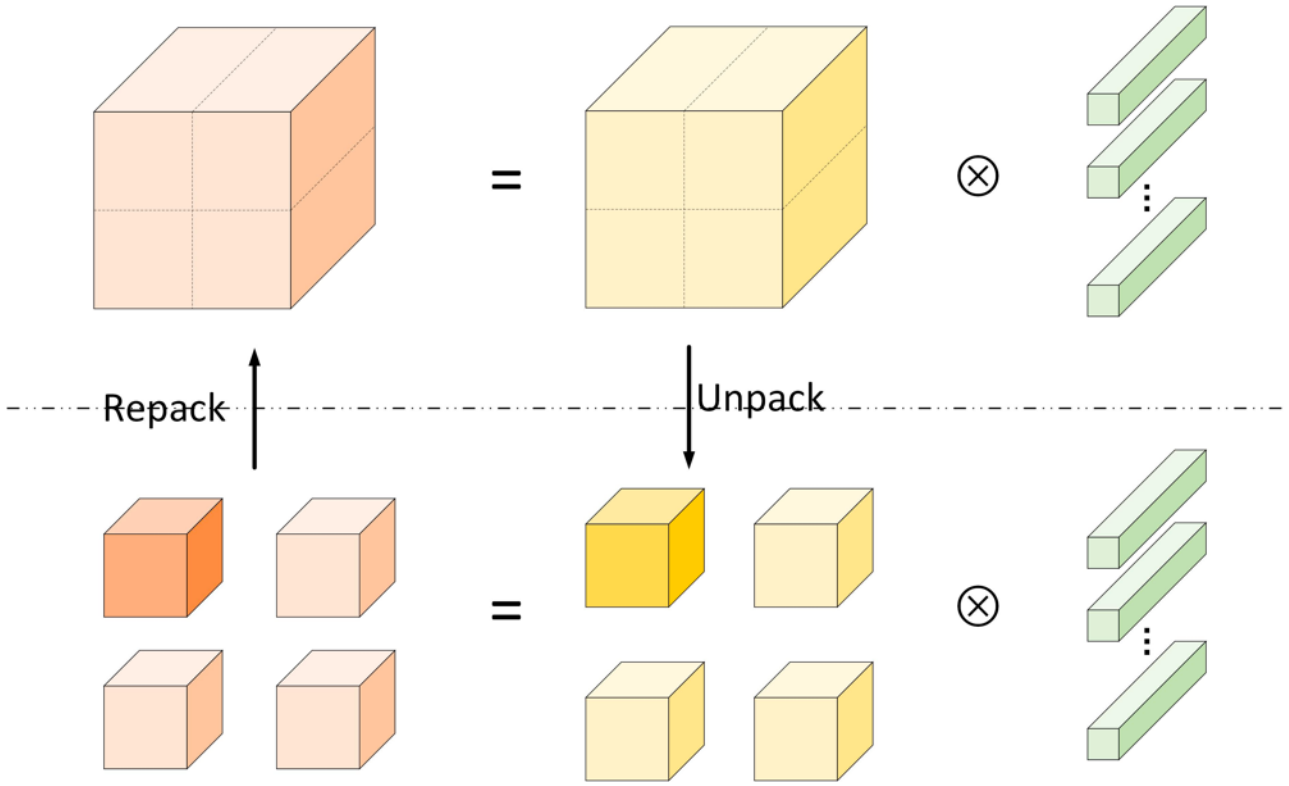
Img2col 到空间组合优化

- 如图所示在空间上将输出、输入划分为四份：



空间组合优化原理

- 划分后，大卷积计算被拆分为若干个小卷积进行计算。划分过程中计算总量不变，但计算小矩阵时访存局部性更好，可以借由计算机存储层次结构获得性能提升：



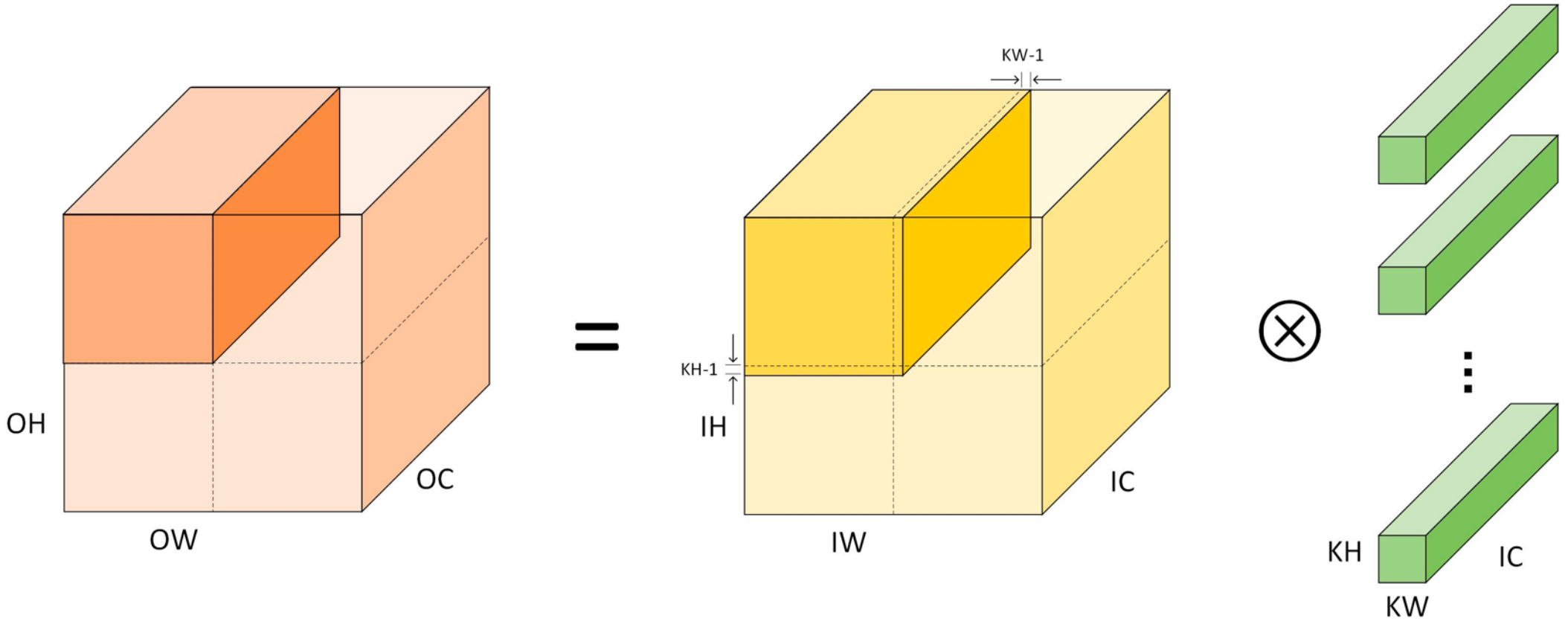
空间组合优化 注意点

- 值得注意的是，上文的描述中忽略了 Padding 的问题。实际将输入张量划分为若干个小张量时，除了将划分的小块中原始数据拷贝外，还需要将相邻的小张量的边界数据拷贝：

$$N \times \left(\frac{H}{h} + 2\left(KH - 1\right) \right) \times \left(\frac{W}{w} + 2\left(KW - 1\right) \right) \times C$$

- 这里的 $2(KH-1)$ 和 $2(KW-1)$ 遵循 Padding 规则。规则为 VALID 时，可以忽略；规则为 SAME 时，位于 Input Tensor 边界一边 Padding 补 0，不在 Input Tensor 边界 Padding 使用邻居张量值。

空间组合优化 注意点



空间组合优化 问题

- 实际应用中可以拆为很多份。例如可以拆成小张量边长为 4 或者 8 ，从而方便编译器向量化计算操作。随着拆分出的张量越小，Padding 引起的额外内存消耗越大，其局部性也越高，负面作用是消耗的额外内存也越多。

空间组合优化 问题

- 对于不同规模的卷积，寻找合适的划分方法不是一件容易的事情。正如计算机领域的许多问题一样，该问题也是可以自动化的，例如通过使用 AI编译器 可以在这种情况下寻找较优的划分方法。

推理引擎架构