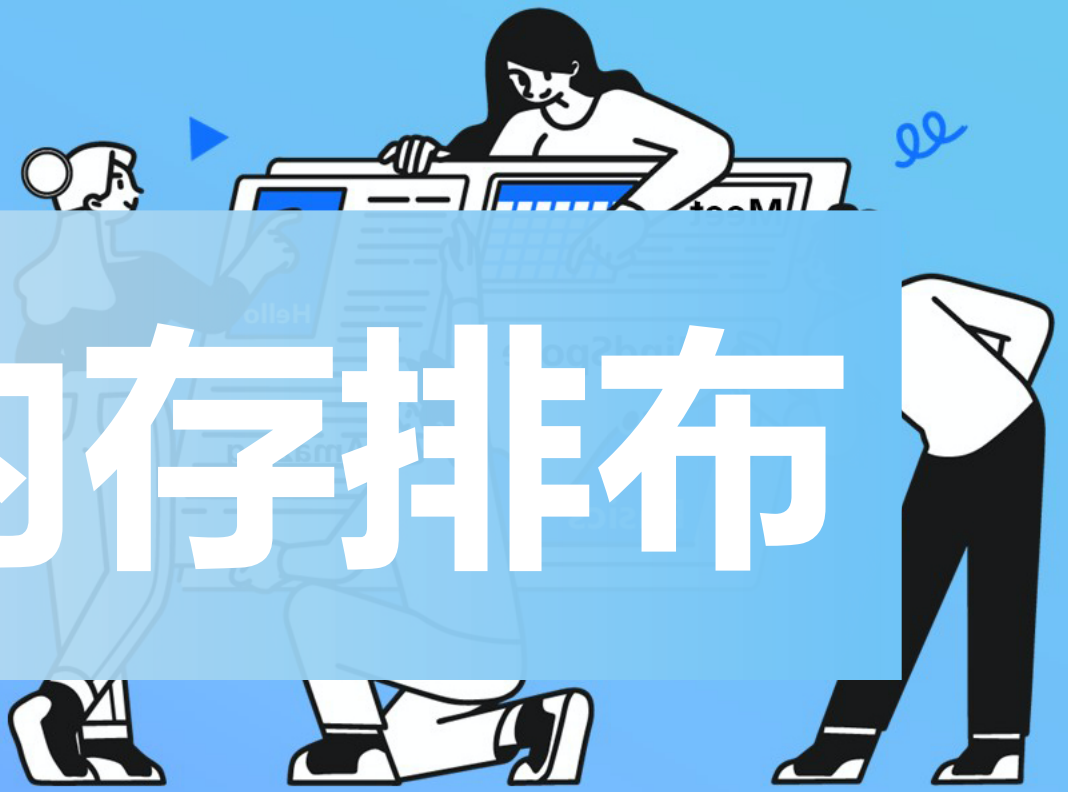
推理引擎-Kernel优化



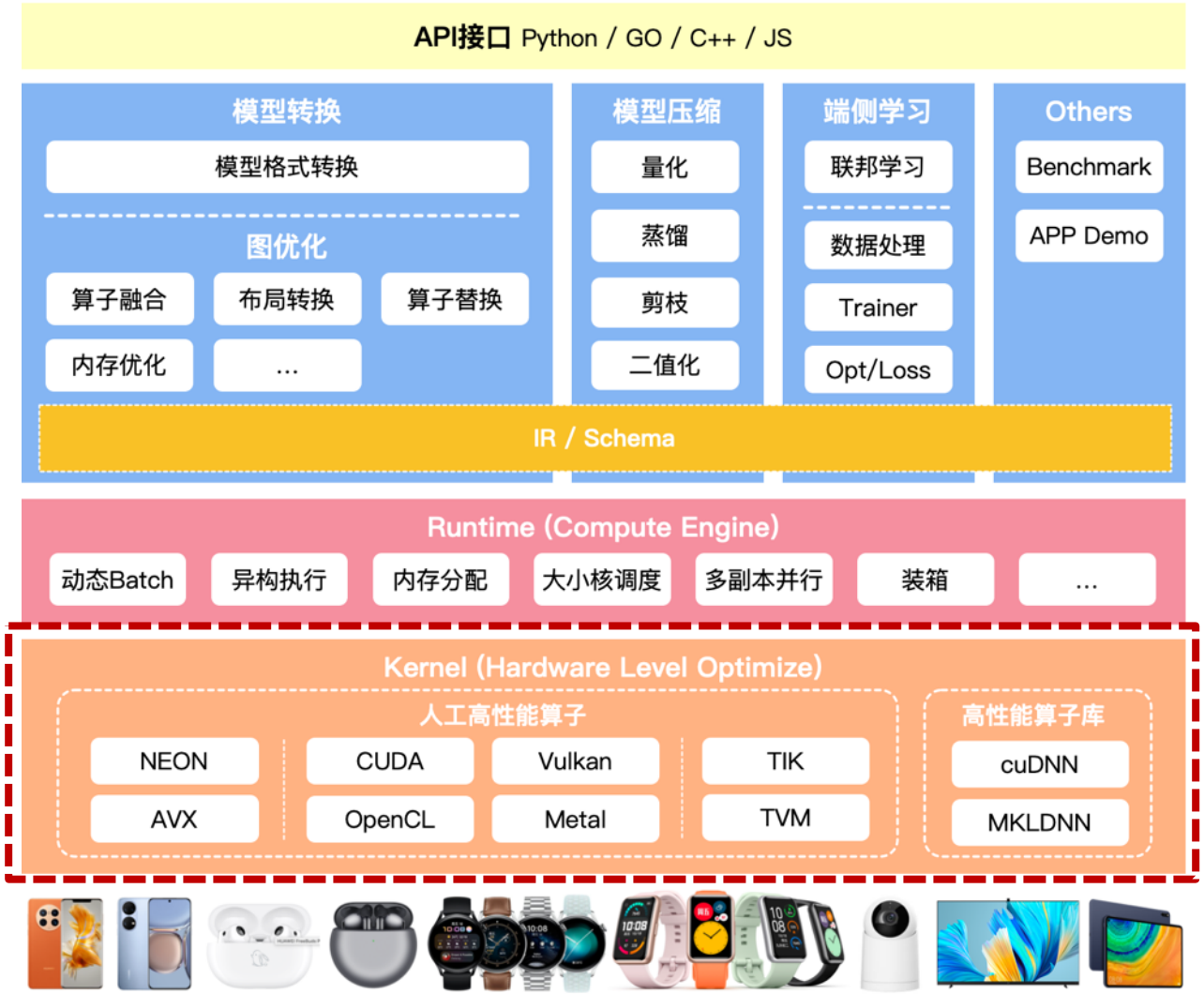
NC4HW4内存排布

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

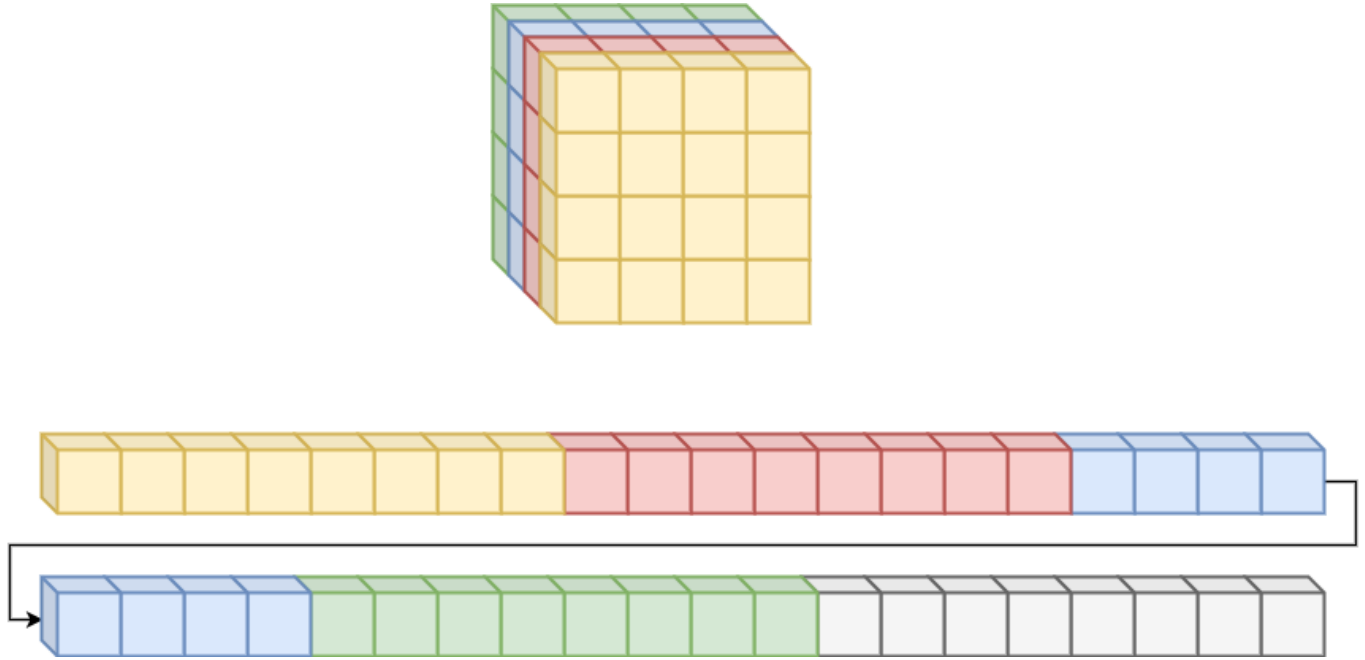
- 算子优化
- 算子执行
- 算子调度

Tensor 内存排布

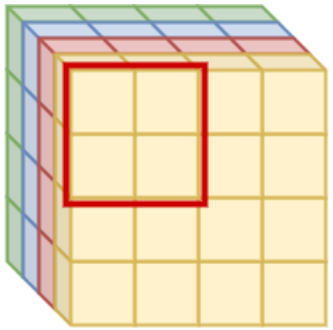
Tensor 常见的内存排布有 NCHW 和 NHWC，除此之外，现在以内存排布方式 NC4HW4为例子，讲讲在 ARM 上如何数据排布？这种排布对性能又有什么提升呢？

张量内存排布 - NCHW

- 当一条指令处理一个数据时，卷积操作需要做循环乘累加。如图所示，与 kernel 对应的 feature map 中的数据不是连续分布的。如果 feature map 空间很大的话，这样不是按照通道顺序取数据，指针会不断来回地在内存空间移动，还会造成 cache miss 严重影响运行性能。

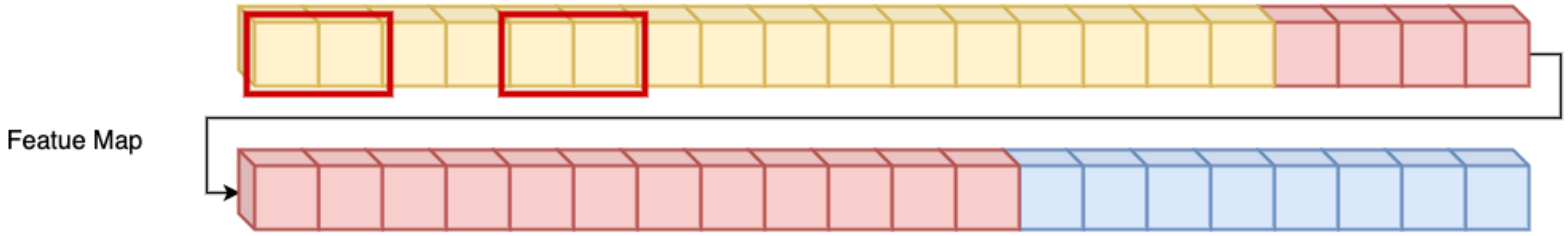


张量内存排布 - 卷积操作

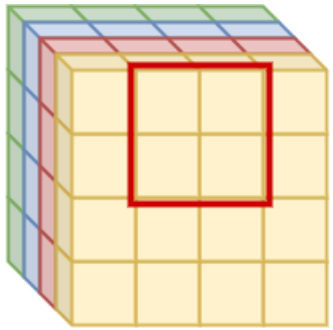


Featue Map

Kernel

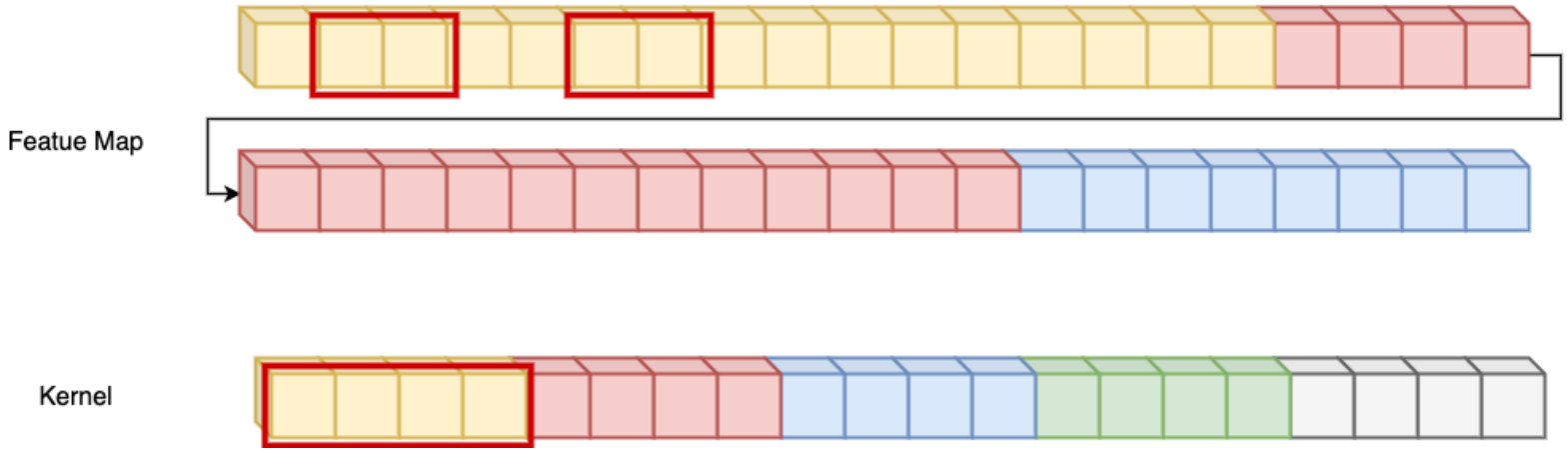


张量内存排布 - 卷积操作



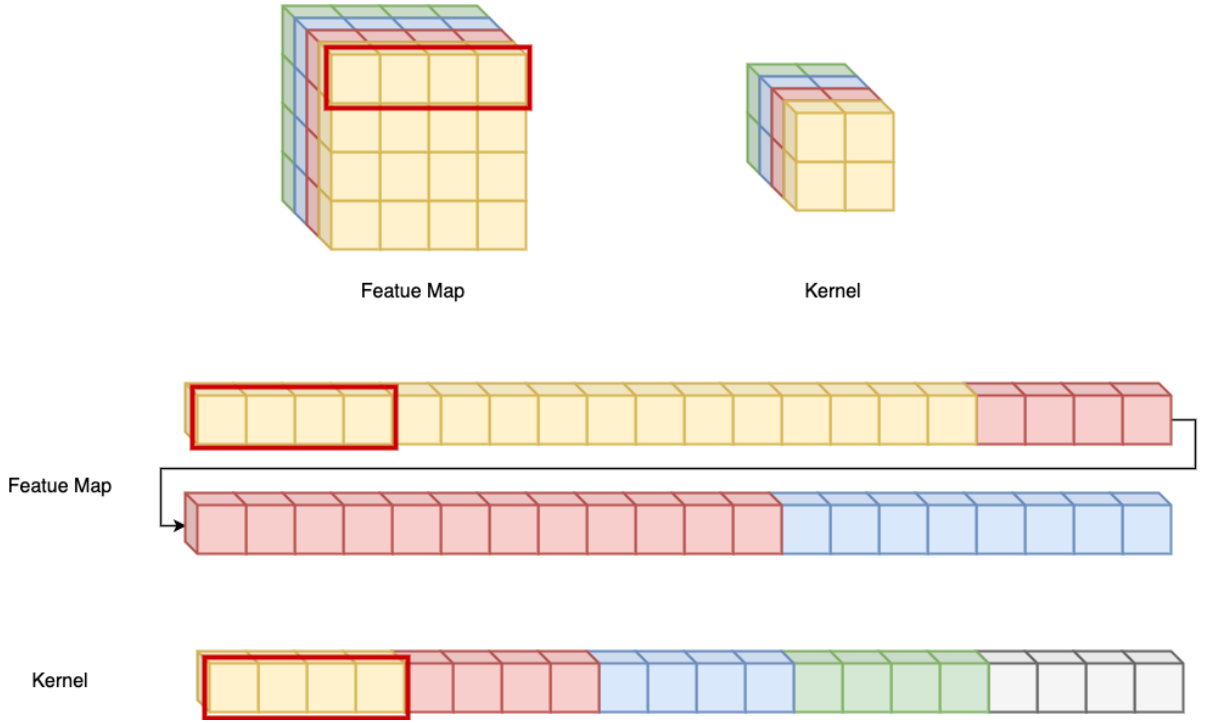
Featue Map

Kernel



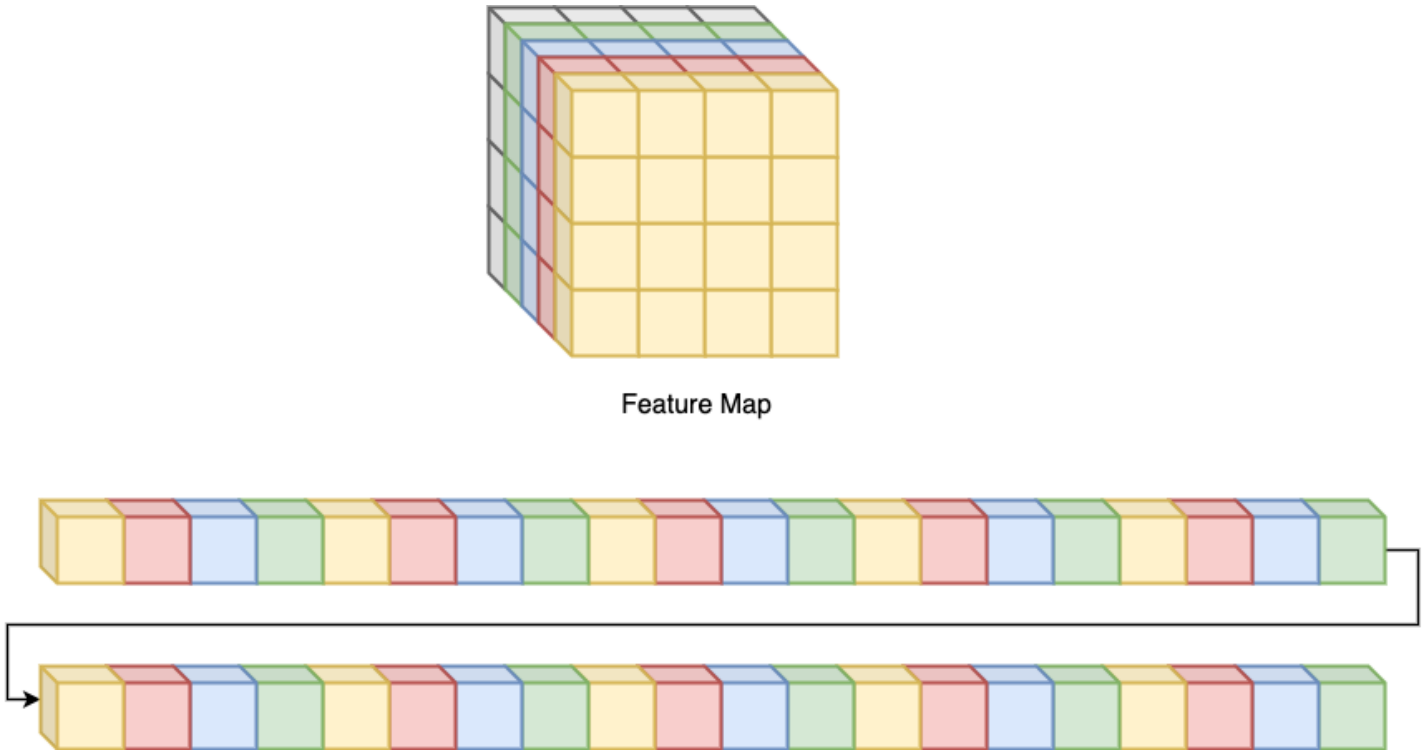
张量内存排布 - HC4HW4

- 当 kernel size 不为 4 的倍数时，想使用诸如 x86 结构 SSE 指令/arm 结构 neon 指令，以及端侧 GPU 的 OpenGL 和 OpenCL 等，可以单指令处理 4 组数据的指令集时，使用NCHW 内存排布同样不方便。



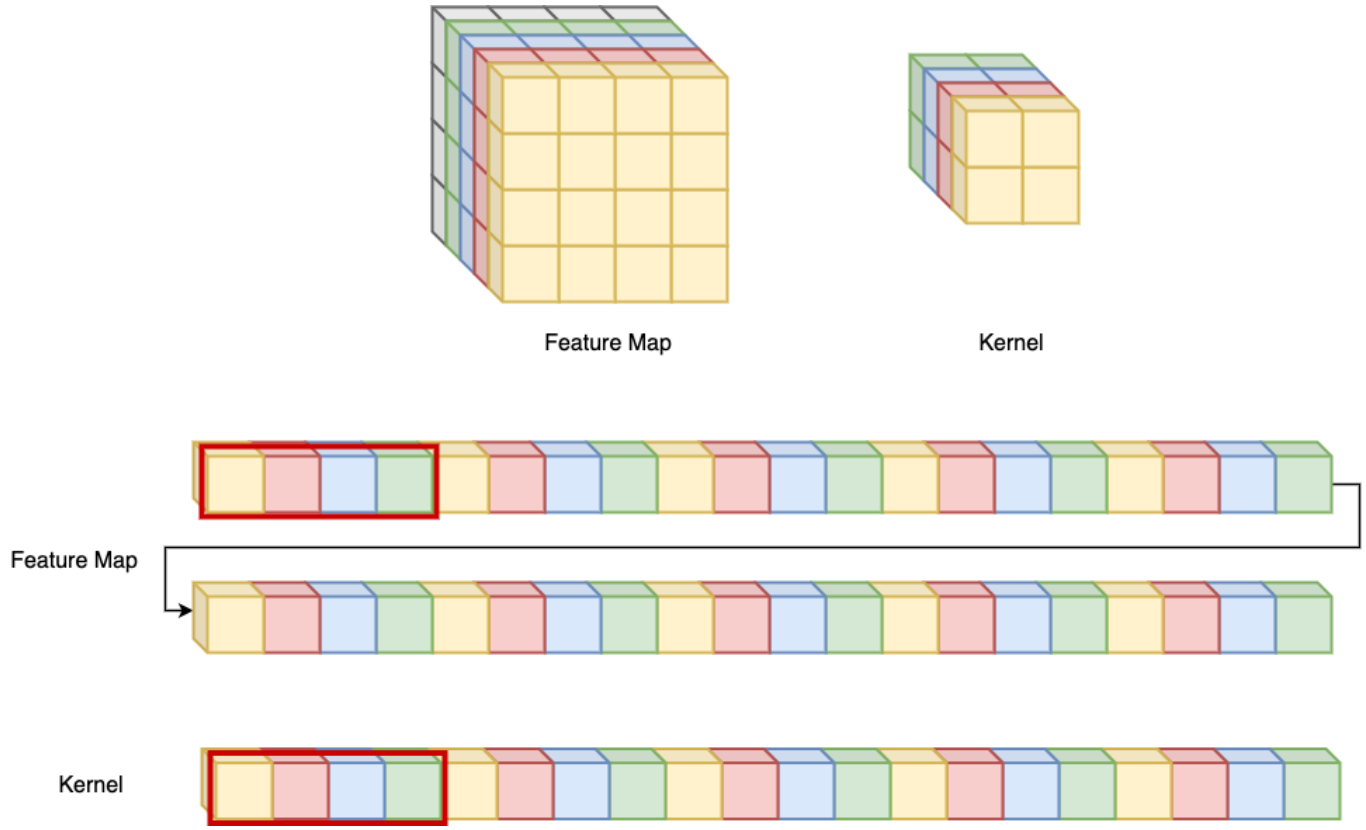
NC4HW4

- HC4HW4 准确的说是：N(C/4)H(W*4)
- 即沿着 Channel 方向取4个数按照 W 方向排列，如果c不是 4 的倍数，则全用 0 补上。



NC4HW4

- 此时进行单指令处理4组数据（SIMD）操作，一次指令处理四个数据，实现卷积操作。



NC4HW4
格式总结

NC4HW4 特点

计算特性

难点：兼顾并行与访问连续

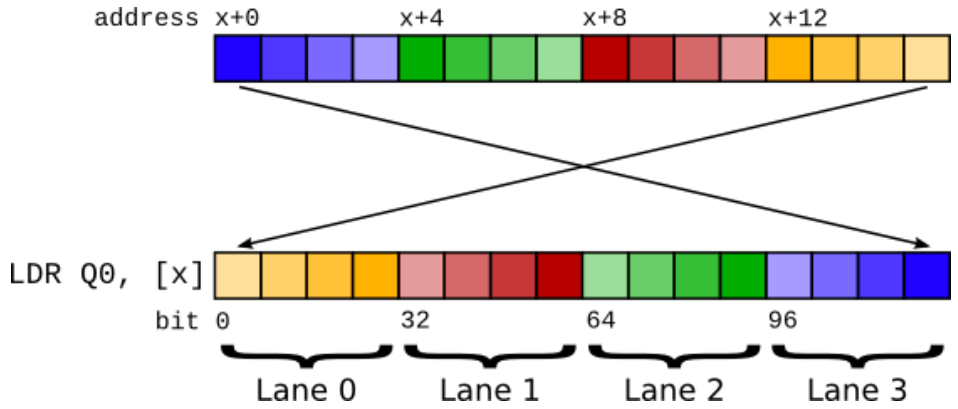
硬件特性

C 方向计算可并行

HW 方向连续访问

SIMD 并行计算

$$AvgPool(i,j,k,l) = \frac{1}{p_w}\frac{1}{p_h}\sum_{p_i=0}^{p_w}\sum_{p_j=0}^{p_h} x(i,j,s_h k+p, s_w l+p_j)$$



nc4hw4 提升卷积推理性能

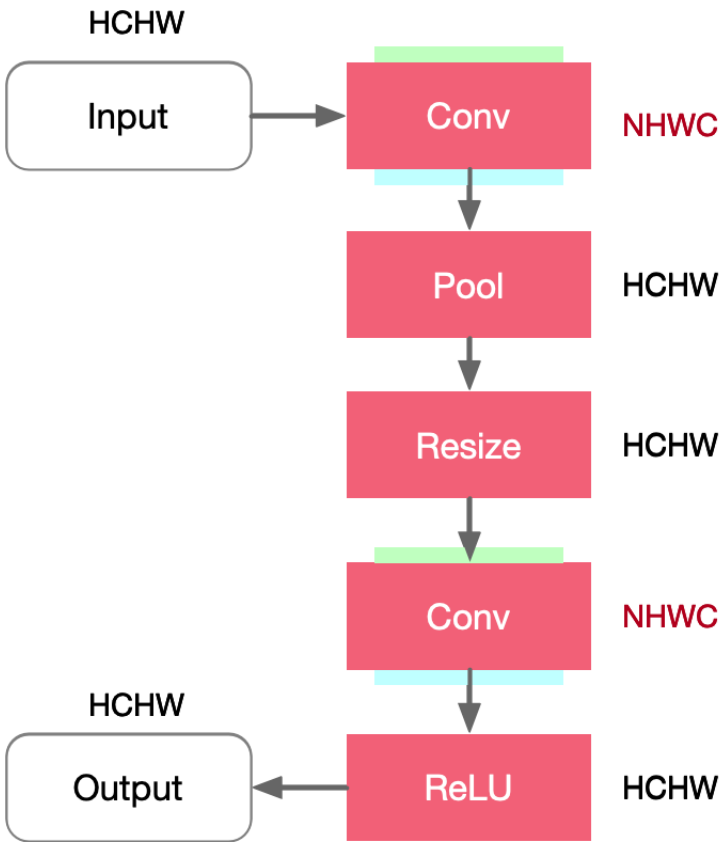
- nc4hw4 是为了配合 Vector 操作，如对 Vector 的读写/乘累加，在arm上可以利用 neon指令。由于在进行 HW 维度上 GEMM 计算时，不同 Channel 维度之间是互不影响，正适合作为Vector 操作的不同元素。nc4hw4 排布配合 neon 指令加速 im2col+gemm 卷积推理的具体过程如下：
- **深度学习的CV算子往往具有如下计算特性，在C方向上计算可并行，但需要读取HW方向相邻数据。为了充分利用 SIMD 加速能力，NC4HW4 布局可以兼顾 SIMD 使用和内存访问连续的需求。**

对计算图的要求

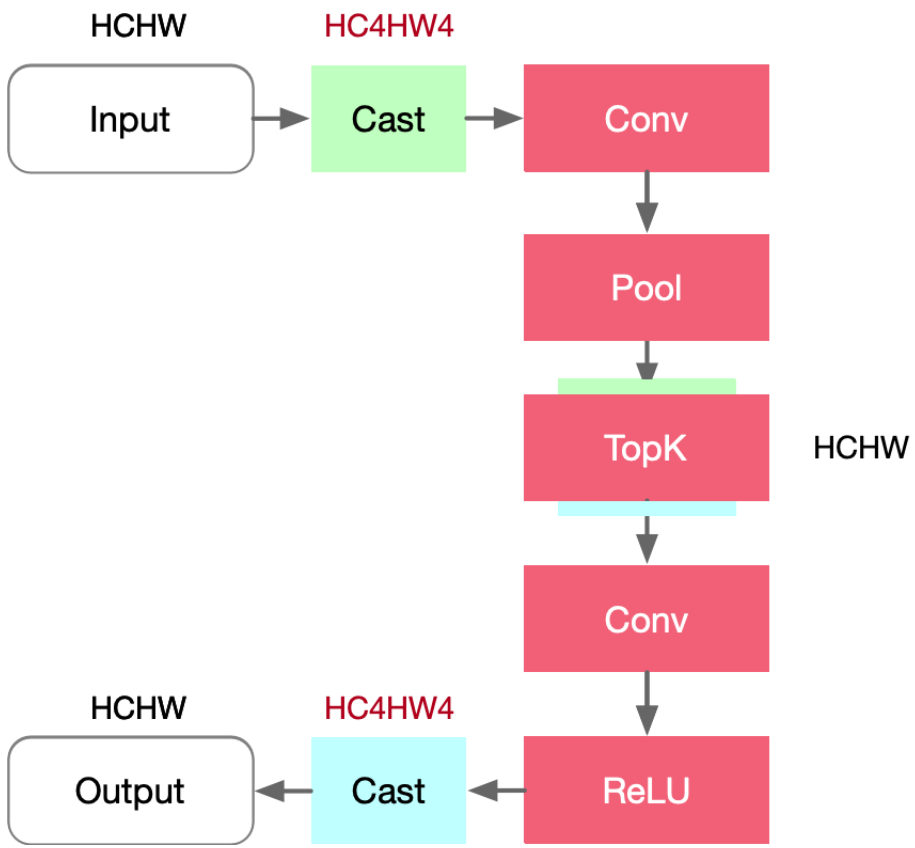
- 不同数据格式（NCHW、NHWC）在不同 Kernel 计算的时候是会带来效率提升，但是也有代价：需要额外进行内存排布的转换，除非整个网络所有 Op 推理实现都是以相同数据排布。

对计算图的要求

正常计算图



目标计算图



优缺点

优点

- 进行 NC4HW4 重排后，可以充分利用 ARM CPU 指令集的特性，实现对卷积等操作进行加速；同时可以较少 cache miss，提高内存命中率。

缺点

- 对于较大的 feature 特征图，如果其 channel 不是 4 的倍数，则会导致补充 0 过多，导致内存占用过高，同时也相应的增加计算量。