推理引擎-Kernel优化

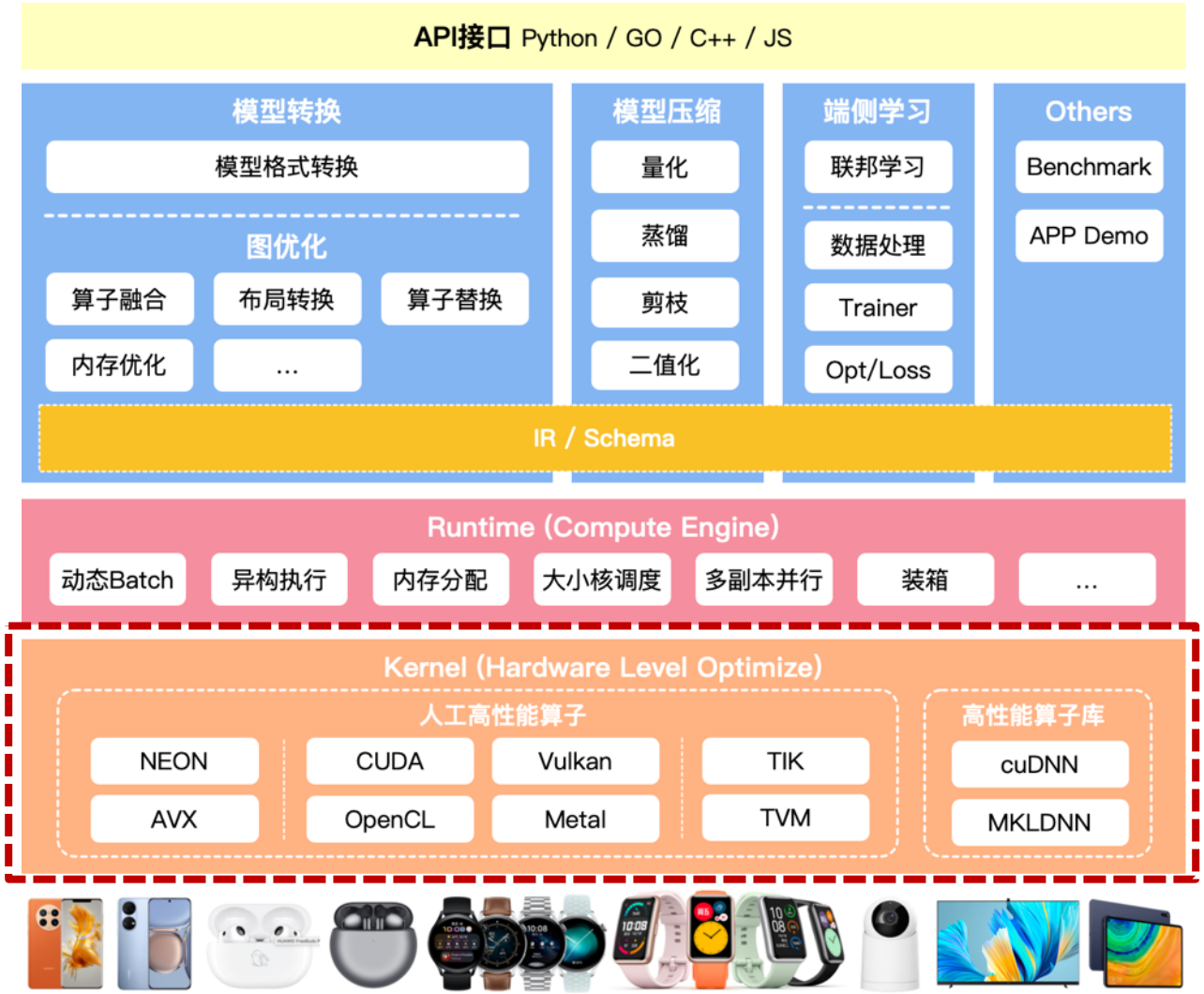
卷积优化原理

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

卷积基础概念

Convolution 卷积

- 卷积（Convolution, aka. Conv）：是神经网络的核心计算之一，它在 CV 领域方面的突破性进展引领了深度学习的热潮。卷积的变种丰富，计算复杂，神经网络运行时大部分时间都耗费在计算卷积，网络模型的发展在不断增加网络的深度，因此优化卷积计算就显得尤为重要。
- 随着技术的发展，研究人员提出了多种优化算法，包括 Im2col、Winograd 等等。Kernel 优化系列首先定义卷积 Conv 的概念，继而简要介绍几种常见的优化方法，并讨论作者在该领域的一些经验。

Convolution

- 在泛函分析中，卷积、旋积或褶积 (Convolution) 是**通过两个函数f和g生成第三个函数的一种数学运算**，其本质是一种特殊的积分变换，表征函数 f 与 g 经过翻转和平移的重叠部分函数值乘积对重叠长度的积分。

- 卷积神经网络（ Convolution Neural Networks, CNN ）的概念拓展自信号处理领域的卷积。 信号处理的卷积定义为：

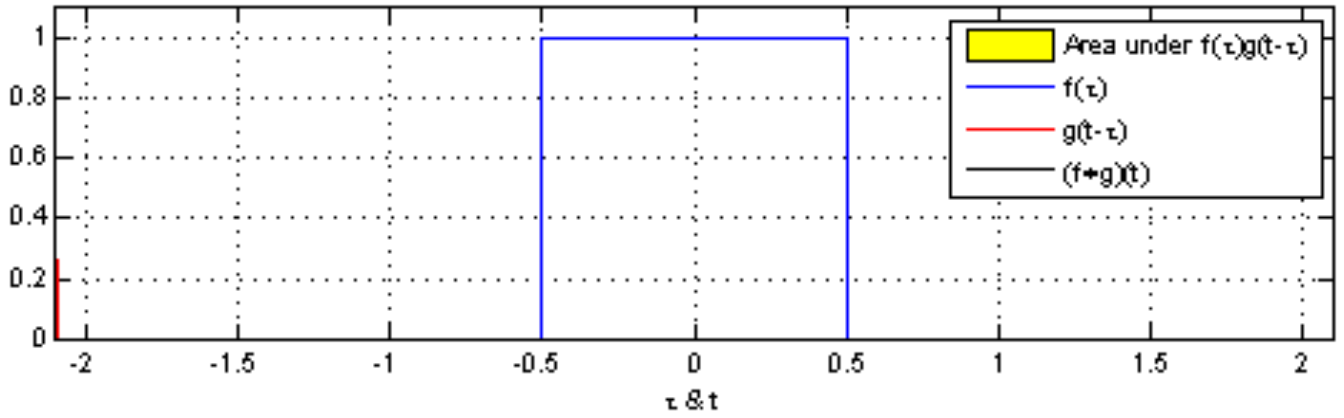
$$(f * g)(t) \triangleq \int_{\mathbb{R}^n} f(\tau)g(t-\tau)d\tau$$

Convolution

- 可以证明，关于几乎所有的实数 x，随着 x 的不同取值，积分定义了一个新函数 $h(x)$，称为函数 f 与 g 的卷积，记为。

$$f(t) = (f * g)(t)$$

- 积计算在直觉上不易理解，其可视化后如*图一*所示。图中红色滑块在移动过程中与蓝色方块的积绘制成的三角图案即为卷积结果 $(f * g)(t)$ 在各点上的取值：



Convolution

$$(f * g)(t) \triangleq \int_{\mathbb{R}^n} f(\tau)g(t-\tau)d\tau$$

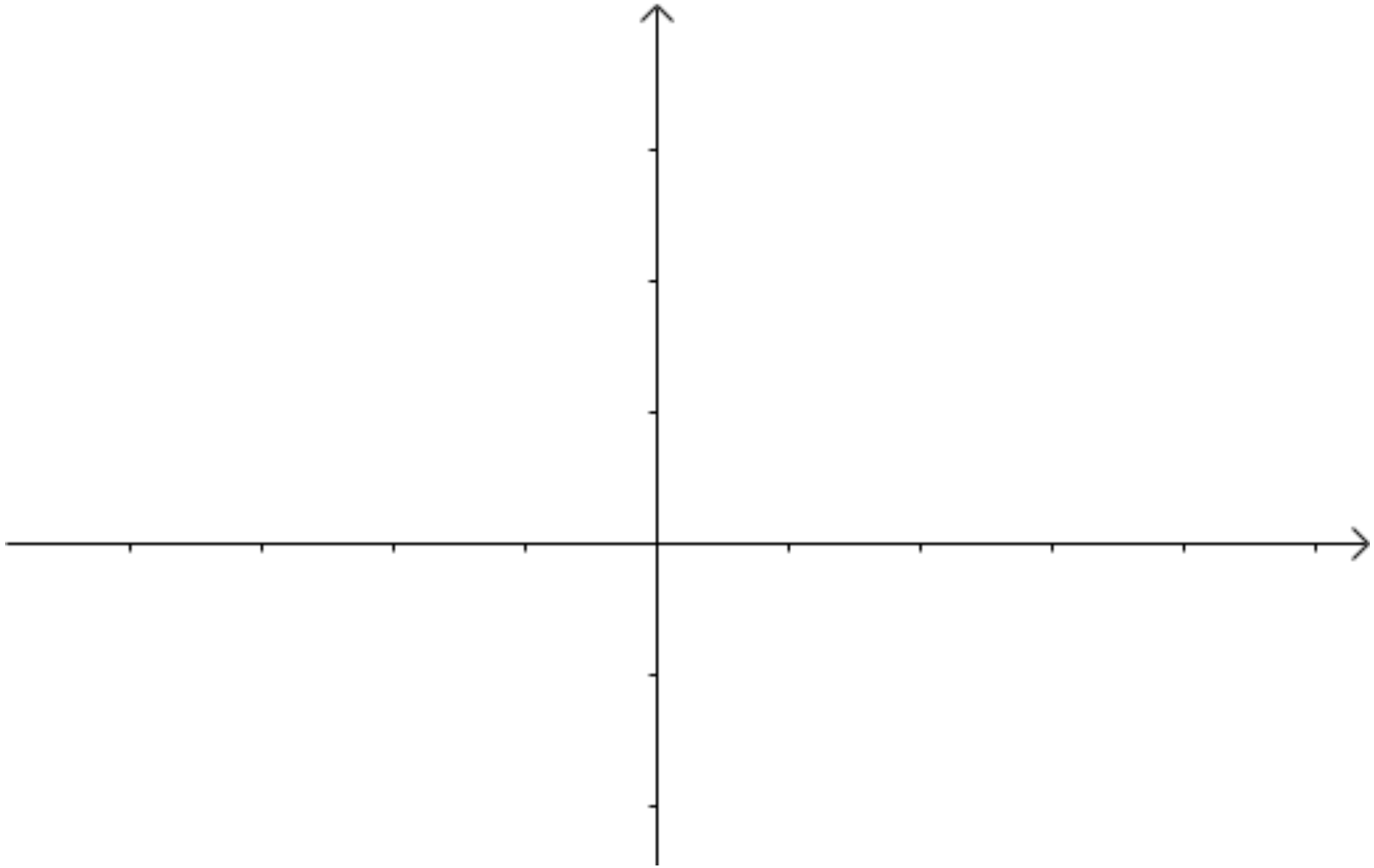
$$t = \tau + (t - \tau)$$

$$x = \tau \qquad\qquad y = +(t - \tau)$$

$$x + y = n$$

Convolution

$$x + y = n$$



Convolution

Convolution

- 对于信号处理的卷积定义为连续的表示，真正计算的过程中会把连续用离散形式进行计算：

$$(f * g)(n) \triangleq \sum_{\mathbb{Z}^n} f(m)g(n-m)\,.$$

- 将该离散卷积公式拓展到二维空间即可得到神经网络中的卷积，可简写为：

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

Convolution

- 将该离散卷积公式拓展到二维空间即可得到神经网络中的卷积，可简写为：

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

- 其中：
  - S 为卷积的输出；
  - I 为卷积输入；
  - K 为卷积核；

Convolution

- 将该离散卷积公式拓展到二维空间即可得到神经网络中的卷积，可简写为：

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

- 可视化：

https://github.com/vdumoulin/conv_arithmetic

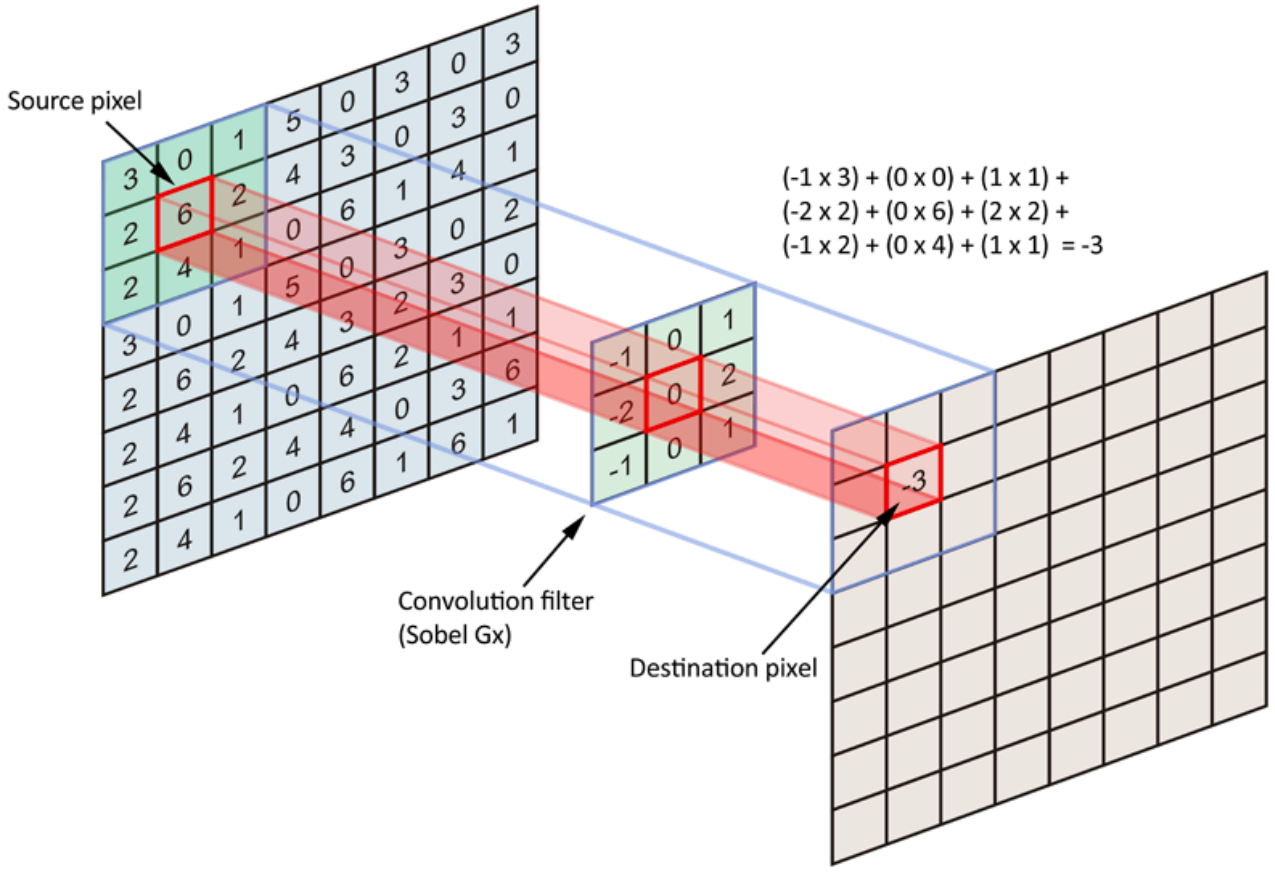
Convolution on Image

- 当应用到计算机视觉中处理图片时，图片的通道（Channel）可以对二维卷积简单堆叠，即：

$$S(i,j) = (I * K)(i,j) = \sum_{c} \sum_{m} \sum_{n} I_c(i-m, j-n) K_c(m,n)$$

- 其中：
  - S 为卷积的输出；
  - I 为卷积输入；
  - K 为卷积核；
  - C 为输入图像通道；

Convolution on Image



Convolution on Tensor

- 当中张量的内存布局为 NHWC 时，卷积计算相应的伪代码如下。其中外三层循环遍历输出 C 的每个数据点，对于每个输出数据都需要经由内三层循环累加求和得到（点积）。

```
for (int oh = 0; oh < OH; oh++) {
    for (int ow = 0; ow < OW; ow++) {
        for (int oc = 0; oc < OC; oc++) {
            C[oh][ow][oc] = 0;
            for (int kh = 0; kh < KH, kh++){
                for (int kw = 0; kw < KW, kw++){
                    for (int ic = 0; ic < IC, ic++){
                        C[oh][ow][oc] += A[oh+kh][ow+kw][ic] * B[kh][kw][ic];
                    }
                }
            }
        }
    }
}
```

Kernel 调度优化方法

- 循环展开（Loop Unrolling）
- 循环分块（loop tiling）
- 循环重排（loop Reorder）
- 循环融合（loop Fusion）
- 循环拆分（loop Split）
- 向量化（Vector）
- 张量化（Tensor）
- 访存延迟（ Latency Hiding ）
- 存储分配（Memory Allocation）

循环优化（Loop Optimization）

指令优化（Instructions Optimization）

存储优化（Memory Optimization）

【AI编译器】后端优化

播放全部

译器跟硬件之间的相连接的模块，更多的是算子或者Kernel进行优化，而优化之前需要把计算图转换
每一个算子/Kernel进行循环优化、指令优化和内存优化等技术。

默认排序　升序排序　编辑

AI编译器后端优化来啦！AI编译器后端架构！【AI编译器】

940　2022-12-22

如何对算子IR表示？算子是如何分开计算和调度两部分？

640　2022-12-23

AI编译器后端算子优化来啦！算子优化手工方式！【AI编译

739　2022-12-25

后端算子循环优化！Loop Optimization常见方法！【AI

514　2022-12-26

算子优化的指令和存储优化！【AI编译器】后端优化05篇

472　2022-12-28

Auto Tuning原理！TVM的AutoTVM实现方式详解！【AI

1027　1–3