【模型推理】一文看懂 winograd 卷积加速算法

欢迎关注我的公众号［极智视界］，获取我的更多笔记分享

O_o >_< o_O O_o ~_~ o_O

本文详细解释了 winograd 算法加速卷积的实现原理。

前面已经写过几篇卷积加速相关的文章，感兴趣的同学可以查阅《【模型推理】一文看懂 Img2Col卷积加速算法》、《【模型推理】一文看懂 Google TPU 脉动阵列加速卷积计算原理》、《【模型推理】谈谈为什么卷积加速更喜欢 NHWC Layout》。

winograd 算法最早是 1980 年由 Shmuel Winograd 提出的《Fast Algorithms for Convolutional Neural Networks》，当时并没有引起太大的轰动。在 CVPR 2016 会议上，Lavin 等人提出了利用 winograd 加速卷积运算，于是 winograd 加速卷积优化在算法圈里火了一把。

winograd 为什么能加速卷积运算呢，简单来说就是用更多的加法计算来减少乘法计算，从而降低计算量，且不像 FFT 那样会引入复数 (关于 FFT 加速卷积后面会再写一篇)，但前提是，处理器中的乘法计算的时钟周期要大于加法计算的时钟周期。好了，下面开始。

1、winograd 加速一维卷积计算

下面是一个比较经典的例子，假设我们的输入信号和卷积核是这样：

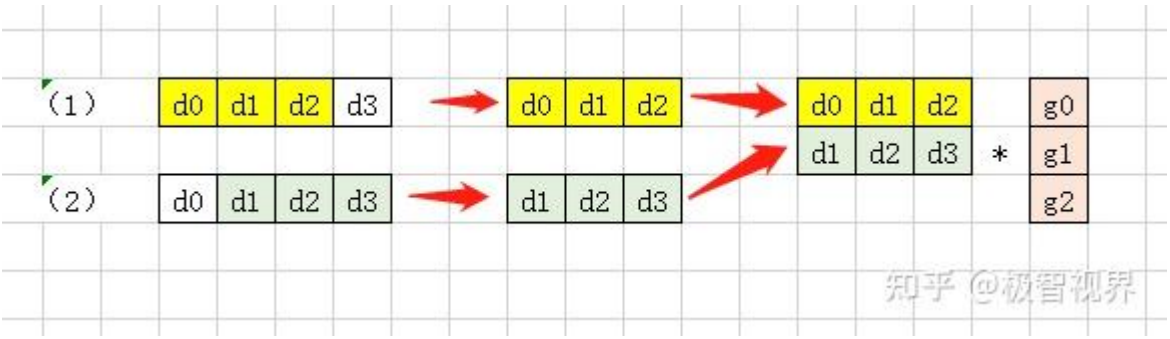
```
# 输入信号
d = [d0, d1, d2, d3]T

# 卷积核
g = [g0, g1, g2]T
```

那么整个的卷积过程可以转换成如下的矩阵乘形式：

$$F(2,3)=\begin{bmatrix} d0 & d1 & d2 \\ d1 & d2 & d3 \end{bmatrix}\begin{bmatrix} g0 \\ g1 \\ g2 \end{bmatrix}=\begin{bmatrix} r0 \\ r1 \end{bmatrix}$$

捋得清楚吗，怎么就转换到如上的矩阵乘了，下面解释一下，应该比较形象了。



这个矩阵乘计算过程是这样的：

```
r0 = d0*g0 + d1*g1 + d2*g2    # 3次乘法+2次加法

r1 = d1*g0 + d2*g1 + d3*g2    # 3次乘法+2次加法
```

以上的计算过程总共需要 6 次乘法和 4 次加法。

观察以上的计算过程，由于在卷积层的设计中，往往 stride < kernel_size 的，所以最后转换的矩阵乘中往往有规律的分布着大量的重复元素，比如这个一维卷积例子中矩阵乘输入矩阵第一行的 d1、d2 和 第二行中的 d1、d2。

而 winograd 的做法是这样的：

$$F(2,3)=\begin{bmatrix} d0 & d1 & d2 \\ d1 & d2 & d3 \end{bmatrix}\begin{bmatrix} g0 \\ g1 \\ g2 \end{bmatrix}=\begin{bmatrix} m1+m2+m3 \\ m2-m3-m4 \end{bmatrix}$$

其中，

$$m_1=(d_0-d_2)g_0 \qquad m_2=(d_1+d_2)\frac{g_0+g_1+g_2}{2}$$

$$m_4=(d_1-d_3)g_2 \qquad m_3=(d_2-d_1)\frac{g_0-g_1+g_2}{2}$$

如上的转换将原本需要 6 次乘法减少到了 4 次，看上去加法运算是增加的，但是 m1、m2、m3、m4 是可以提前计算好的，所以实际加法还是 4 次。以上的转换作者只给出了结论，并没有给出推导，下面我们进行一下反推。

上面已经说过，如果我们直接进行矩阵乘，会得到 r0 = d0 x g0 + d1 x g1 + d2 x g2，r1 = d1 x g0 + d2 x g1 + d3 x g2，所以我们可以令：m1 + m2 + m3 = r0，m2 - m3 - m4 = r1。像这样：

$$m_1+m_2+m_3=d_0*g_0+d_1*g_1+d_2*g_2$$

$$m_2-m_3-m_4=d_1*g_0+d_2*g_1+d_3*g_2$$

先观察，两个等式中 m1、m4 是没有重复出现的，我先令 m1 = d0 x g0，m4 = -d3 x g2，这样可以约掉 m1 和 m4，所以左边只剩两个变量，两个等式两个变量即可求出 m3、m4，所以这个时候的 m1、m2、m3、m4 是这样的：

$$m_1 = d_0 * g_0$$
$$m_2 = \frac{g_1 d_1 + g_2 d_2 + g_0 d_1 + g_1 d_2}{2}$$
$$m_3 = \frac{g_1 d_1 + g_2 d_2 - g_0 d_1 - g_1 d_2}{2}$$
$$m_4 = -d_3 * g_2$$

观察 m2 中包含了 d1、d2、g0、g1、g2，将其转换为两个多项式乘积形式，拆成 d 和 g 分开的形式，如下：

$$m_2 = \frac{(d_1 + d_2)(g_0 + g_1 + g_2)}{2} - \frac{d_2 g_0}{2} - \frac{d_1 g_2}{2}$$

同理，对 m3 也进行如上转换，完了之后现在的 m1、m2、m3、m4 是这样的：

$$m_1 = d_0 * g_0$$
$$m_2 = \frac{(d_1 + d_2)(g_0 + g_1 + g_2)}{2} - \frac{d_2 g_0}{2} - \frac{d_1 g_2}{2}$$
$$m_3 = \frac{(d_2 - d_1)(g_0 - g_1 + g_2)}{2} - \frac{d_2 g_0}{2} + \frac{d_1 g_2}{2}$$
$$m_4 = -d_3 * g_2$$

这个时候让我们回到最开始的等价关系，进行观察，要是我在 m2、m3 上同时加上一个值，对于式 (b) 来说是不变的（所以 m4 不用动），对于式 (a) 来说需要给 m1 减去两倍的这个值。

a $m_1 + m_2 + m_3 = d_0 * g_0 + d_1 * g_1 + d_2 * g_2$

b $m_2 - m_3 - m_4 = d_1 * g_0 + d_2 * g_1 + d_3 * g_2$

观察现在的 m1、m2、m3、m4，当这个值是 (d2g0) / 2 时可以简化表达式，所以这样给上面等式进行等价变换后得到的 m1、m2、m3、m4 如下：

$$m_1 = g_0(d_0 - d_2)$$

$$m_2 = \frac{(d_1 + d_2)(g_0 + g_1 + g_2)}{2} - \frac{d_1 g_2}{2}$$

$$m_3 = \frac{(d_2 - d_1)(g_0 - g_1 + g_2)}{2} + \frac{d_1 g_2}{2}$$

$$m_4 = -d_3 * g_2$$

继续如上操作，如果给 m2 加上一个值，同时给 m3 减去这个值，那么对于式 (a) 来说是不变的 (所以 m1 不用动)，对于式 (b) 来说需要给 m4 减去两倍的这个值才能等价。同样观察现在的 m1、m2、m3、m4，当这个值为 (d1g2) / 2 时可以进一步简化表达式，接着作这样的变换后得到最终的 m1、m2、m3、m4，如下：

$$m_1 = g_0(d_0 - d_2)$$

$$m_2 = \frac{(d_1 + d_2)(g_0 + g_1 + g_2)}{2}$$

$$m_3 = \frac{(d_2 - d_1)(g_0 - g_1 + g_2)}{2}$$

$$m_4 = g_2(d_1 - d_3)$$

开不开心，激不激动，我们经历了上述的推导之后终于得到了作者给你的结果。

将上面的计算过程写成矩阵的形式为：

其中：

- g：表示卷积核；
- d：表示输入信号；
- G：表示卷积核变换矩阵，尺寸为 (m+r-1) x r
- BT：表示输入变换矩阵，尺寸为 (m+r-1) x (m+r-1)
- AT：输出变换矩阵，尺寸为 m x (m+r-1)

2、winograd 加速二维卷积计算

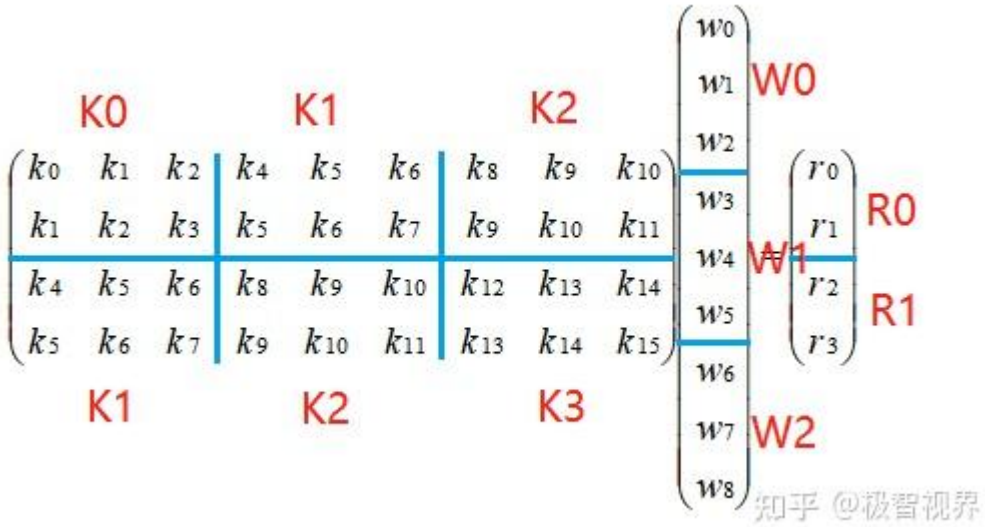
将一维卷积的变换扩展到二维卷积，同样用矩阵形式表示为：

其中，g为 r x r 的卷积核，d 为 (m + r -1) x (m + r -1) 的图像块，此时为 F(2 x 2，3 x 3)。

二维卷积可先参考这个《【模型推理】一文看懂Img2Col卷积加速算法》将卷积过程进行 img2col 展开成矩阵乘的形式，示意图如下：

$$\begin{pmatrix} k_0 & k_1 & k_2 & k_4 & k_5 & k_6 & k_8 & k_9 & k_{10} \\ k_1 & k_2 & k_3 & k_5 & k_6 & k_7 & k_9 & k_{10} & k_{11} \\ k_4 & k_5 & k_6 & k_8 & k_9 & k_{10} & k_{12} & k_{13} & k_{14} \\ k_5 & k_6 & k_7 & k_9 & k_{10} & k_{11} & k_{13} & k_{14} & k_{15} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \end{pmatrix} = \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix}$$

将如上的矩阵乘过程进行分块：



即可以表示成如下形式：

怎么样，熟不熟悉，是不是跟前面一维卷积的形式一毛一样。所以 winograd 对于二维卷积的优化就像套娃一样，你把它一层层拨开就能看到一维卷积的优化姿势了。

3、winograd 优化小结

在 winograd 卷积加速算法中，对于一维卷积，当输出为 m，卷积核长为 r，对应的乘法计算次数为 m+r-1 次；将一维卷积扩展到二维，如果输出维度是 m x n，卷积核维度为 r x s，则需要的乘法次数是 (m+r-1) x (n+s-1)。对一个矩阵大小为 4 x 4 的输入，卷积核大小为 3 x 3，对应的输出为 2 x 2,正常计算的情况下，使用 im2col 加速方法的乘法次数为 2 x 2 x 3 x 3 = 36 次，而当使用 winograd 时，对应的乘法次数为 (2+3−1) x (2+3−1) =16，可以看到乘法次数明显减少，从而加速效果会更加明显。

winograd 算法通过减少乘法次数来实现提速，但是加法的数量会相应增加，同时需要额外的转换计算以及存储转换矩阵，随着卷积核 (kernel) 和 分块 (tile，对于大尺寸 feature map，会将 feature map 切分成一个个等大小有重叠的 tile，在每个 tile 上面进行 winograd 卷积) 的尺寸增大，就需要考虑加法、转换计算 和 存储 的代价，而且 tile 越大，转换矩阵越大，计算精度的损失会进一步增加，所以一般 winograd 只适用于较小的卷积核和 tile（对大尺寸的卷积核，可使用 FFT 进行加速）。

好了收工，不知道你理解了没，欢迎讨论~

【csdn 传送】