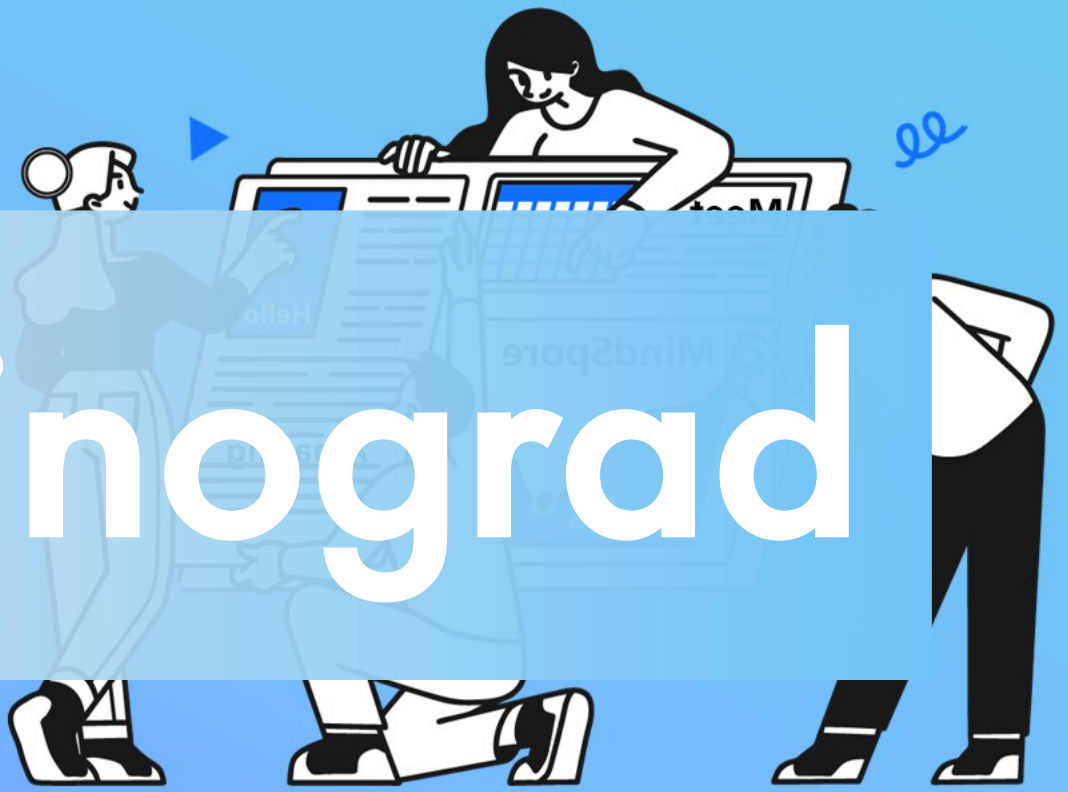
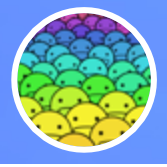
推理引擎-Kernel优化

卷积优化Winograd



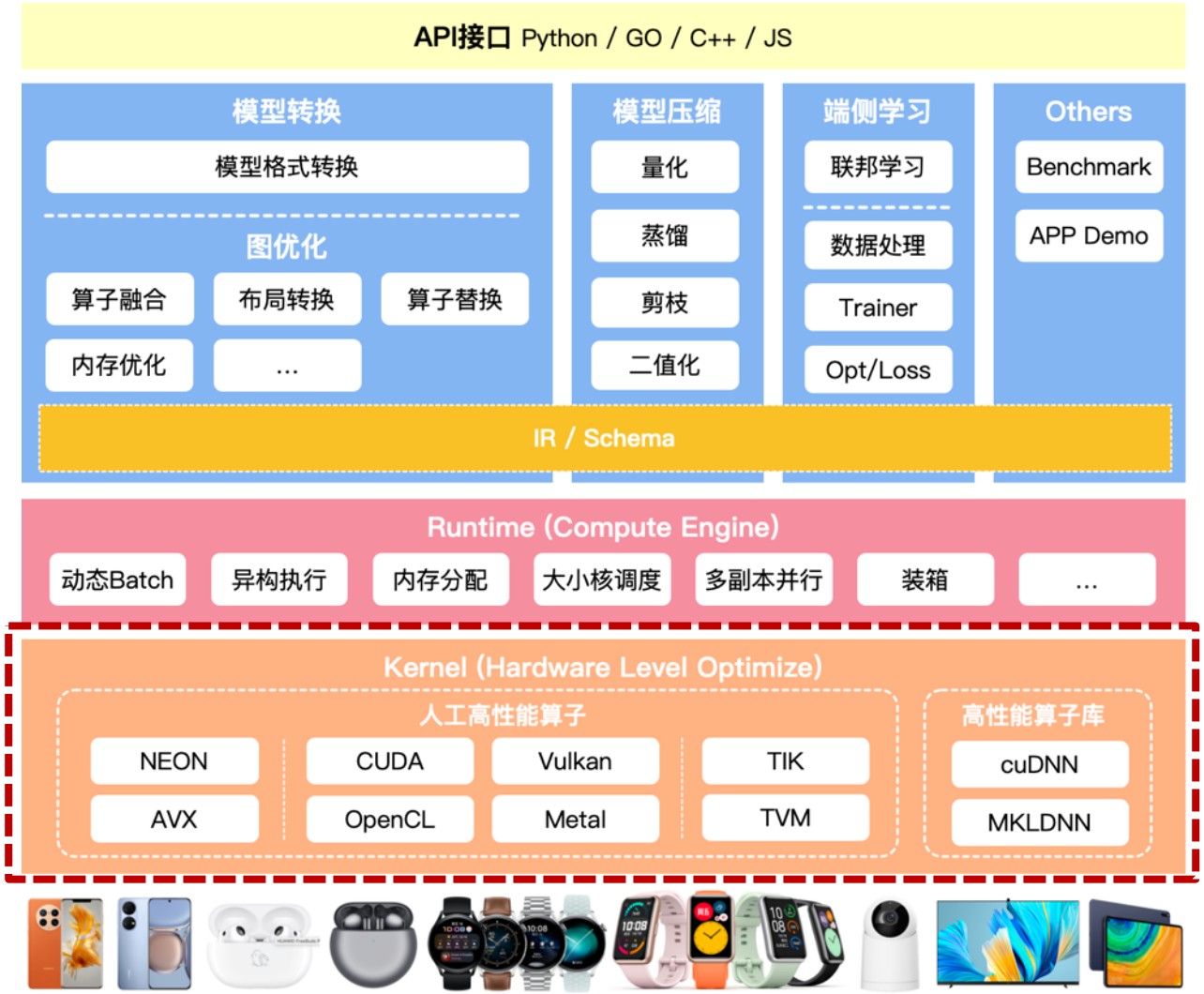


ZOMI

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

www.hiascend.com
www.mindspore.cn

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

Winograd算法

Im2col + Packing 回顾

- 前两节介绍的两种算法，Im2col 在将三维张量组织成矩阵后调用 GEMM 计算库，这些计算库很大程度上使用一些基于访存局部性的优化；空间组合优化则本身就是利用局部性优化的方法。
- Winograd 优化算法则是矩阵乘优化方法中 Coppersmith–Winograd 算法的一种应用，能够优化卷积计算的乘法计算量，是基于算法分析的方法。

www.hiascend.com
www.mindspore.cn

Winograd 算法概述

- Winograd 算法最早是 1980 年由 Shmuel Winograd 在 Arithmetic complexity of computations 上首次提出，当时并没有引起太大的轰动。CVPR 2016 会议上，Andrew Lavin 等人提出了利用 Winograd 提出的算法对卷积运算进行加速，并发表了提出的 Fast Algorithms for Convolutional Neural Networks，于是 Winograd 加速卷积优化在算法圈迅速火起来。
- Winograd 算法已广泛应用于各种推理引擎中，Winograd 算法在卷积优化中的应用的基本方法和矩阵乘中应用类似，通过技巧性的矩阵计算变换，减少计算过程所需的乘法数量。

www.hiascend.com
www.mindspore.cn

Conv 算法原理

- 假设输入信号和卷积核为：

$$f=[d_0,d_1,d_2,d_3]$$

$$g=[g_0,g_1,g_2]^T$$

- 整个的卷积过程可以转换成如下的矩阵乘形式：

$$F(f\cdot g)=\begin{bmatrix}d_0 & d_1 & d_2\\ d_1 & d_2 & d_3\end{bmatrix}\begin{bmatrix}g_0\\ g_1\\ g_2\end{bmatrix}=\begin{bmatrix}r_0\\ r_1\end{bmatrix}$$

www.hiascend.com
www.mindspore.cn

Conv 算法原理

- 整个的卷积过程可以转换成如下的矩阵乘形式：

$$F(f \cdot g) = \begin{bmatrix} d_0 & d_1 & d_2 \\ d_1 & d_2 & d_3 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \end{bmatrix}$$

- 实际的数学操作为：

$$r_0 = d_0 \times g_0 + d_1 \times g_1 + d_2 \times g_2 \qquad \#\ 3\ 次乘法+2\ 次加法$$

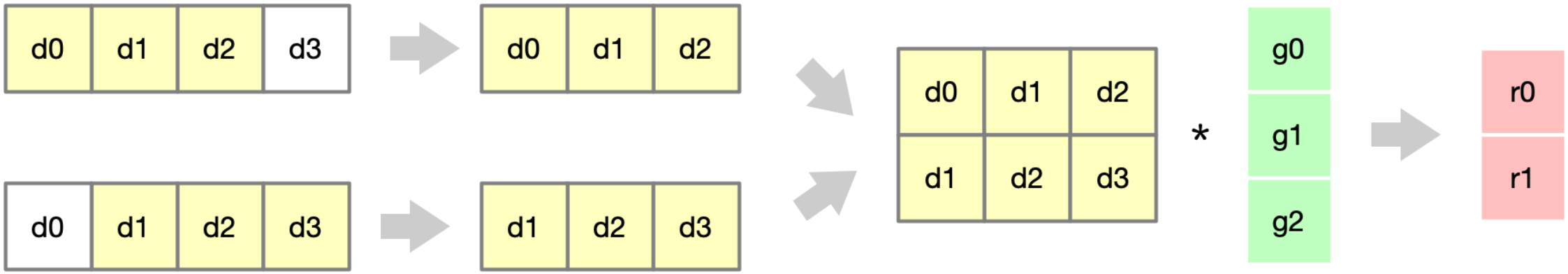
$$r_1 = d_1 \times g_0 + d_2 \times g_1 + d_3 \times g_2 \qquad \#\ 3\ 次乘法+2\ 次加法$$

- 总共需要 6 次乘法和 4 次加法

www.hiascend.com
www.mindspore.cn

Conv 算法原理

- 观察计算过程，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。



www.hiascend.com
www.mindspore.cn

Winograd 一维卷积原理

- Winograd 在在 Arithmetic complexity of computations 证明有：

$$F(f \cdot g) = \begin{bmatrix} d_0 & d_1 & d_2 \\ d_1 & d_2 & d_3 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} m_1 + m_2 + m_3 \\ m_2 - m_3 - m_4 \end{bmatrix}$$

- 其中：

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

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

www.hiascend.com
www.mindspore.cn

Winograd 原理推导

- Winograd 在在 Arithmetic complexity of computations 证明有：

$$F(f \cdot g) = \begin{bmatrix} d_0 & d_1 & d_2 \\ d_1 & d_2 & d_3 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} m_1 + m_2 + m_3 \\ m_2 - m_3 - m_4 \end{bmatrix}$$

- 整个的卷积过程可以转换成如下的矩阵乘形式：

$$F(f \cdot g) = \begin{bmatrix} d_0 & d_1 & d_2 \\ d_1 & d_2 & d_3 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \end{bmatrix}$$

www.hiascend.com
www.mindspore.cn

Winograd 原理推导

- 因此有：

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

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

- 令 $m1 = d0 \times g0$，$m4 = -d3 \times g2$，约掉 m1 和 m4，左边只剩两个变量，两个等式两个变量即可求出 m3、m4，因此 m1、m2、m3、m4 为：

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

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

www.hiascend.com
www.mindspore.cn

Winograd 原理推导

- 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：

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

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

www.hiascend.com
www.mindspore.cn

Winograd 原理推导

- 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：

$$m_1 + m_2 + m_3 = d_0 \times g_0 + d_1 \times g_1 + d_2 \times g_2 \ (a)$$

$$m_2 - m_3 - m_4 = d_1 \times g_0 + d_2 \times g_1 + d_3 \times g_2 \ \ (b)$$

- 当值 $(d2 \times g0) / 2$ 时可以简化表达式，上述等式进行等价变换后得到如下：

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

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

www.hiascend.com
www.mindspore.cn

Winograd 原理推导

- 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：

$$m_1 = g_0 \left(d_0 * g_0 \right) \qquad m_2 = \frac{\left(\mathrm{d}_1 + d_2 \right) \left(g_0 + g_1 + g_2 \right)}{2}$$

$$m_4 = g_2 \left(d_1 - d_3 \right) \qquad m_3 = \frac{\left(\mathrm{d}_2 - d_1 \right) \left(g_0 - g_1 + g_2 \right)}{2}$$

www.hiascend.com
www.mindspore.cn

Winograd 原理推导

- Winograd 已经证明了对于卷积核长度为 r 的一维卷积，计算 m 个输出所需的最少的乘法数量为 $\mu(F(m,r)) = m + r - 1$ 。上面推导结果的计算过程写成矩阵的形式为：

$$Y = A^T[(Gg)\odot(B^T d)]$$

其中：

- g：表示卷积核；
- d：表示输入信号；
- G：卷积核变换矩阵，尺寸为 $(m + r - 1)\times r$
- B^T ：输入变换矩阵，尺寸 $(m + r - 1)\times(m + r - 1)$
- A^T ：输出变换矩阵，尺寸 $m\times(m + r - 1)$

https://github.com/andravin/wincnn

www.hiascend.com
www.mindspore.cn

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：

$$Y = A^T[(GgG^T)\odot(B^TdB)]A$$

其中：

- g：表示 $r\times r$ 卷积核；
- d：表示 $(m+r-1)\times(m+r-1)$ 的输入张量；
- G：卷积核变换矩阵，尺寸为 $(m+r-1)\times r$
- B^T：输入变换矩阵，尺寸 $(m+r-1)\times(m+r-1)$
- A^T：输出变换矩阵，尺寸 $m\times(m+r-1)$

https://github.com/andravin/wincnn

www.hiascend.com
www.mindspore.cn

Winograd 加速二维卷积计算

- 将卷积过程进行 img2col 展开成矩阵乘的形式：

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

www.hiascend.com
www.mindspore.cn

Winograd 加速二维卷积计算

- 将如上的矩阵乘过程进行分块：

$$\begin{array}{ccccccccc} & & d_0 & & & d_1 & & & d_2 \\ \left(\begin{array}{ccc|ccc|ccc} k_0 & k_1 & k_2 & k_4 & k_5 & k_6 & k_8 & k_9 & k_{10} \\ k_1 & k_2 & k_3 & k_5 & k_6 & k_7 & k_9 & k_{10} & k_{11} \\ \hline k_4 & k_5 & k_6 & k_8 & k_9 & k_{10} & k_{12} & k_{13} & k_{14} \\ k_5 & k_6 & k_7 & k_9 & k_{10} & k_{11} & k_{13} & k_{14} & k_{15} \end{array}\right) & & & & & & & & \end{array} \begin{array}{c} \left(\begin{array}{c} w_0 \\ w_1 \\ w_2 \\ \hline w_3 \\ w_4 \\ w_5 \\ \hline w_6 \\ w_7 \\ w_8 \end{array}\right) \\ \begin{array}{l} g_0 \\ g_1 \\ g_2 \end{array} \end{array} = \begin{array}{c} \left(\begin{array}{c} r_0 \\ r_1 \\ \hline r_2 \\ r_3 \end{array}\right) \\ \begin{array}{l} r_0 \\ r_1 \end{array} \end{array}$$

$$\begin{array}{ccccccccc} & & d_1 & & & d_2 & & & d_3 \end{array}$$

- 即可以表示成如下形式：

$$F(f \cdot g) = \begin{bmatrix} d_0 & d_1 & d_2 \\ d_1 & d_2 & d_3 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} m_1 + m_2 + m_3 \\ m_2 - m_3 - m_4 \end{bmatrix}$$

www.hiascend.com
www.mindspore.cn

Winograd

工程实现

实现步骤

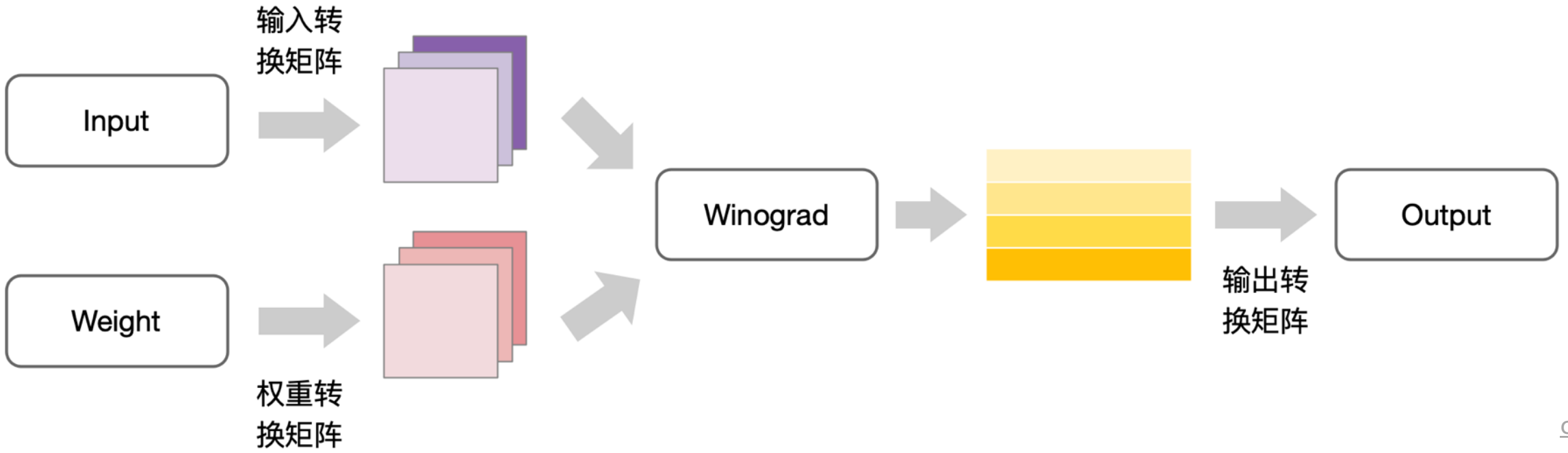
- 在工程实现过程中，主要分 4 步实现 Winograd 算法：

1. 第一步就是对输入卷积核的变换：$U = GgG^T$
2. 第二步就是对输入数据的变换：$V = B^T dB$
3. 第三步就是对M矩阵的计算：$M = \sum U \odot V$
4. 最后一步就是结果的计算：$Y = A^T MA$

www.hiascend.com
www.mindspore.cn

实现步骤

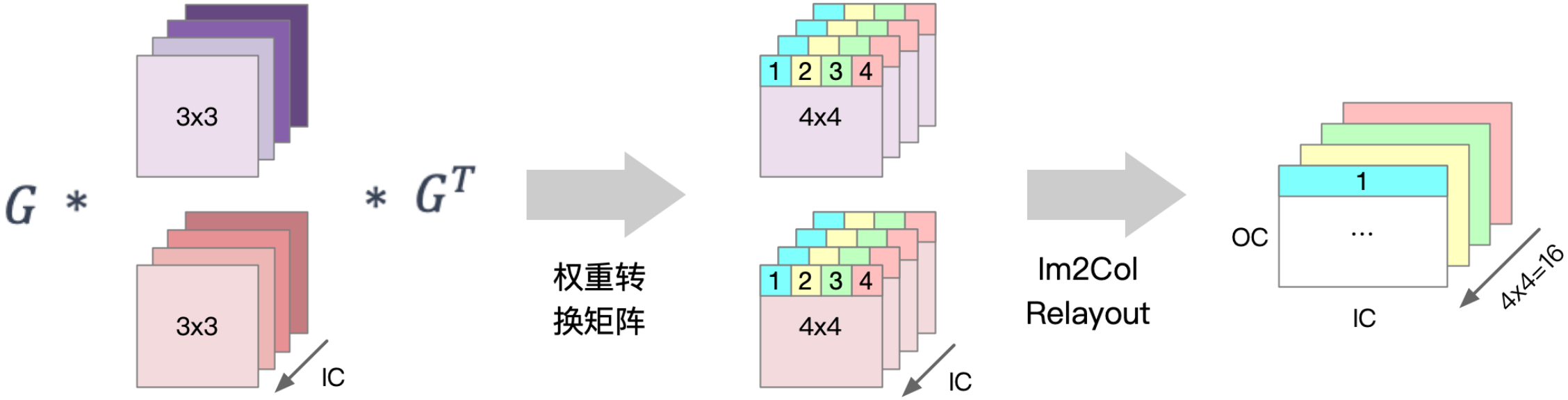
1. 第一步就是对输入卷积核的变换：$U = GgG^T$
2. 第二步就是对输入数据的变换：$V = B^T dB$
3. 第三步就是对M矩阵的计算：$M = \sum U \odot V$
4. 最后一步就是结果的计算：$Y = A^T MA$



om
www.mindspore.cn

实现步骤

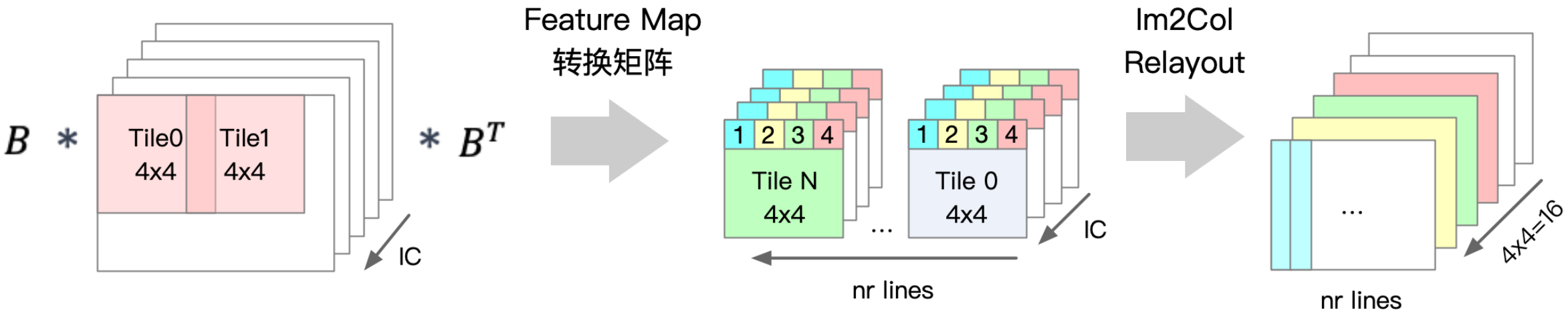
- 对于 weights 的转换，首先通过 Winograd 变换矩阵 G 和 GT 分别将 3x3 的 weight 转换为 4x4 的矩阵，然后将该矩阵中相同位置的点（如图中蓝色为位置 1 的点）转换为一个 IC*OC 的矩阵，最终形成 4x4=16 个转换之后 weights 矩阵。



www.hiascend.com
www.mindspore.cn

实现步骤

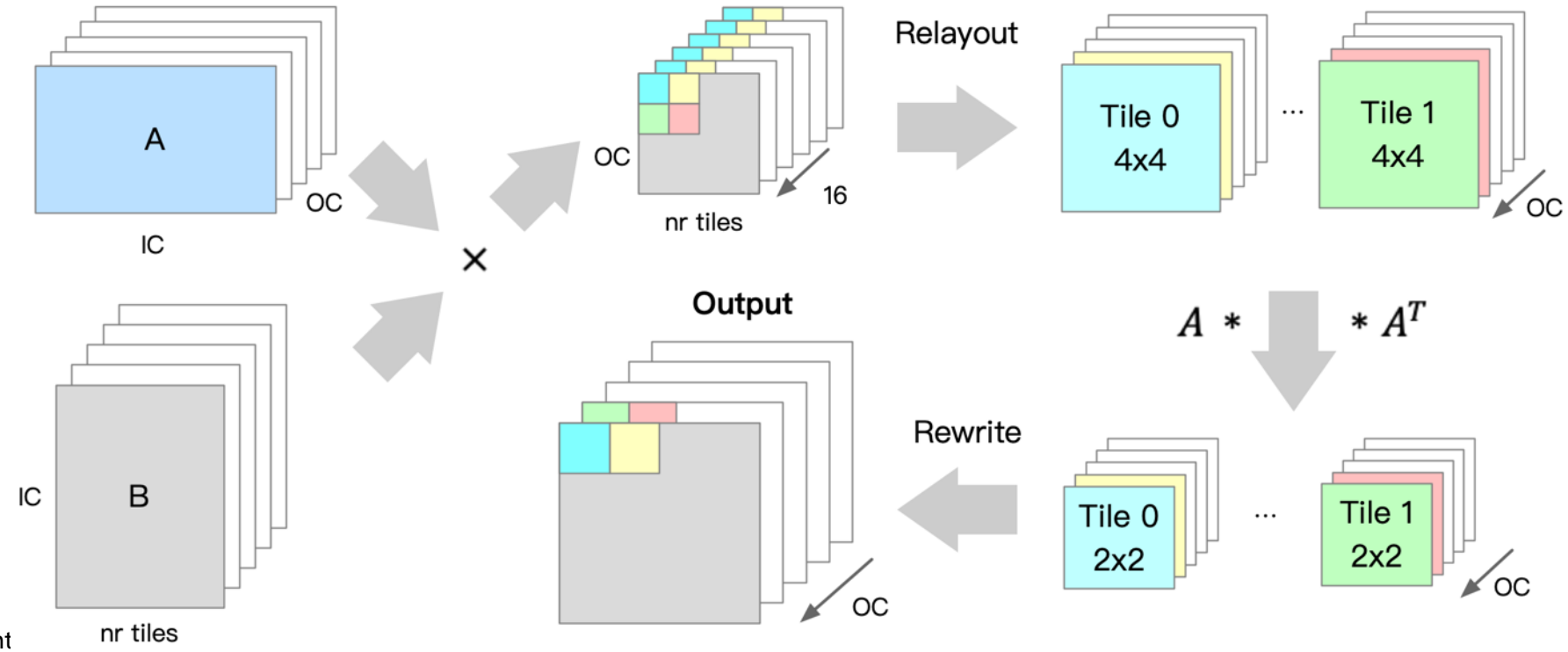
- 对于 FeatureMap 的转换，首先将输入 FeatureMap 按照 4x4 tile 进行切分，然后将每个 tile 通过 B 和 B^T 转换为 4x4 的矩阵，矩阵 B 和 B^T 为 FeatureMap 对应的 Winograd 变换矩阵，然后进行与 weight 处理，转换为 16 个 nr tiles * IC 的 FeatureMap 矩阵。



www.hiascend.com
www.mindspore.cn

实现步骤

- 将上述转换矩阵矩阵乘，得到 16 个 nr tiles * OC 矩阵，然后将相同位置的 16 个点转换为 nr tiles * OC 个 4x4 矩阵，再使用输出的 Winograd 变换矩阵 A 和 A^T 将这些 4x4 的矩阵转换为 2x2 的输出矩阵，最后将这些矩阵写回输出矩阵中就可以得到 Winograd 卷积的最终结果。



www.hiascend.com
www.mindspore.cn

Winograd

回顾与思考

www.hiascend.com
www.mindspore.cn

实现约束与缺点

1. Winograd 计算单个小局部二维卷积的方法，这种算法不能将其直接应用在卷积网络的计算中，否则产生的辅助矩阵规模太大，会影响实际的效果；
2. 不同规模的卷积需要不同规模的辅助矩阵，实时计算出这些辅助矩阵不现实，如果都存储起来会导致规模膨胀。
3. 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，随着卷积核和分块的尺寸增大，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 Winograd 只适用于较小的卷积核和tile。

www.hiascend.com
www.mindspore.cn

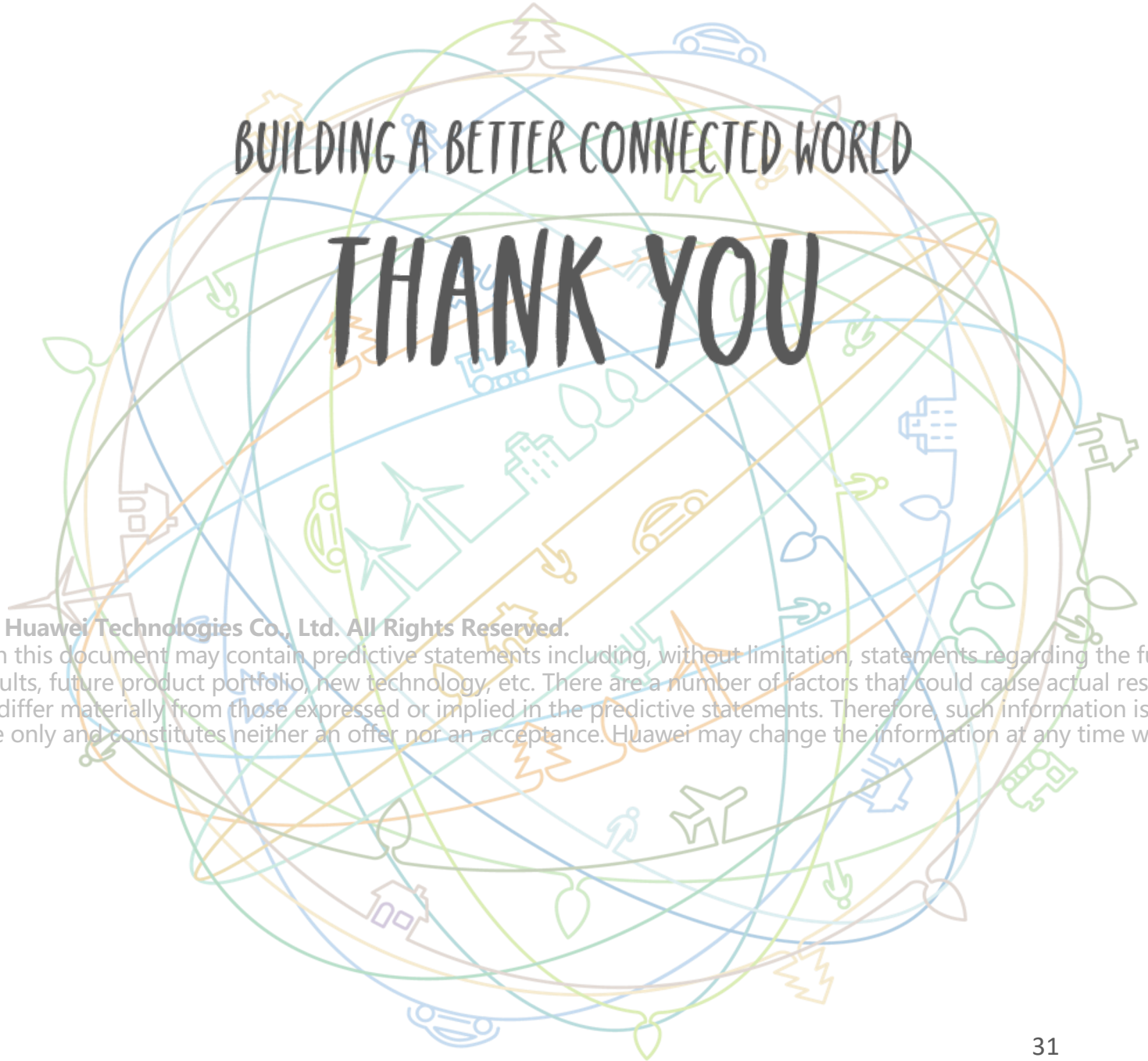
实现约束

1. 在实践中，普遍应用的方法是将一切能固定下来的数据在网络运行前固定。例如，当设计好一个基于 Winograd 算法时，对于特定网络结构的 G 是固定的，对于特定网络 g 也是固定的，那么 $U = GgG^{\mathrm{T}}$ 可以在网络运行前固定。
2. 另一个自然的想法是，Winograd 算法可以和空间组织算法协同工作，即利用局部性也利用算法分析的优化，将卷积计算用空间组合优化算法中的拆分方法，将输入 Input 拆分成若干个小规模卷积。例如拆分成每个小卷积输出 4×4 个数据的卷积。

www.hiascend.com
www.mindspore.cn

引用

1. Winograd, Shmuel. Arithmetic complexity of computations. Vol. 33. Siam, 1980.
2. Lavin, Andrew, and Scott Gray. "Fast algorithms for convolutional neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.



Copyright©2014 Huawei Technologies Co., Ltd. All Rights Reserved.
The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.