

推理引擎-模型转换与优化

计算图优化



ZOMI

Talk Overview

1. 推理系统介绍

- 推理系统架构
- 推理引擎叫故

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

- 架构与流程
- 模型转换技术
- 计算图优化

5. Runtime与在线优化

- 动态batch
- bin Packing
- 多副本并行

Talk Overview

I. 计算图优化

- Challenges and Architecture - 挑战与架构
- graph Optimization - 计算图优化
- Example - ONNX Runtime 图优化
- Optimize Details - 计算图优化详解

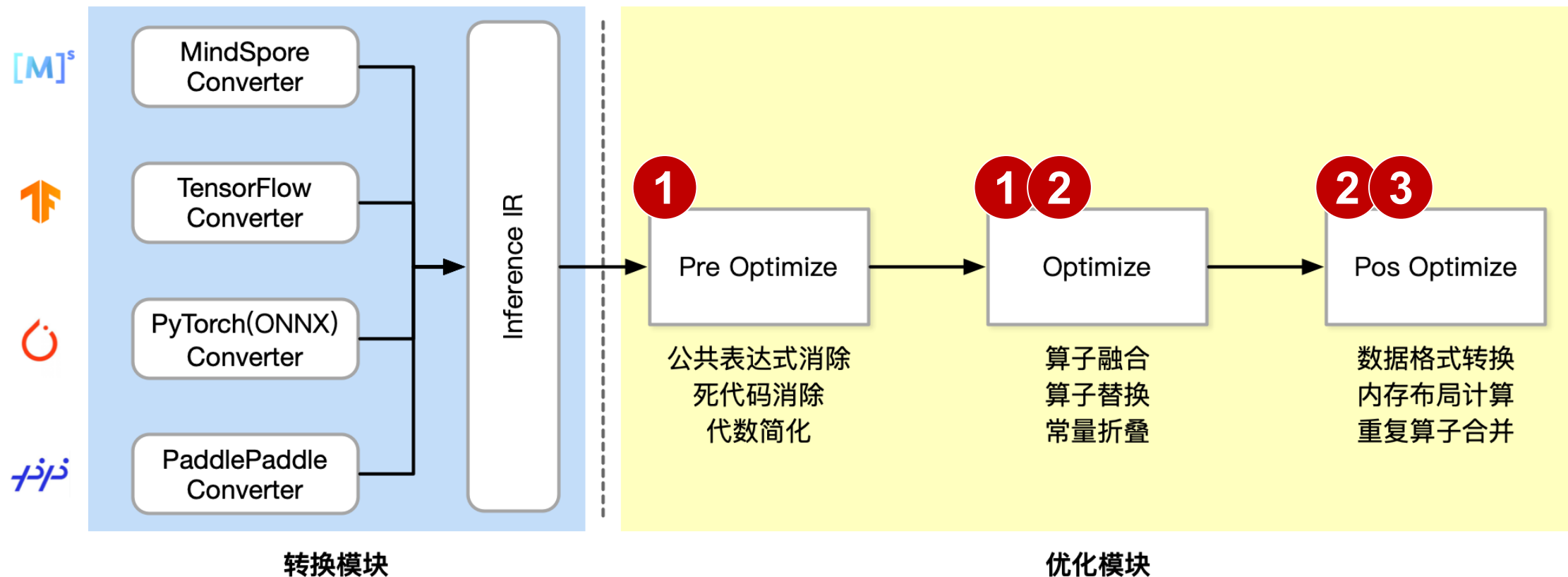


- 代数简化
- 算子优化

图优化方式

- 1. Basic:** 基础优化涵盖了所有保留计算图语义的修改，如：O1常量折叠、O2冗余节点消除和O3有限数量的算子融合。
- 2. Extended:** 扩展优化仅在运行特定后端，如 CPU、CUDA、NPU 后端执行提供程序时适用。其针对硬件进行特殊且复杂的 Kernel 融合策略和方法。
- 3. Layout & Memory:** 布局转换优化，主要是不同 AI 框架，在不同的硬件后端训练又在不同的硬件后端执行，数据的存储和排布格式不同。

工作流程



AI编译器之前端优化

▶ 播放全部

!结构在快速演化，底层计算硬件技术更是层出不穷，对于广大开发者来说不仅要考虑如何在复杂多，还要应对计算框架的持续迭代。AI编译器就成了应对以上问题广受关注的技术方向，让用户仅...

默认排序

升序排序

编辑

1 AI编译器的
01 前端优化
AI编译器 系列之图优化
03:46

AI编译器前端"图层优化"内容概
览!! 【AI编译器】系列之前端优
▶ 600 2022-12-13

2 GraphIR有什么用
02 AI编译器如何接收图层IR?
计算图内容回顾
AI编译器 系列之图优化
11:11

图层IR(Graph IR)是什么? AI编译器
如何接收图层IR进行优化呢? 【AI编
▶ 519 2022-12-13

3 03 AI编译器
--图优化
算子融合原理
17:48

算子融合了解下! AI编译器如何实
现算子融合的? 【AI编译器】系列
▶ 915 2022-12-16

4 数据布局转换
04(上) Data Layout Transform?
为嘛要改变数据的布局?
AI编译器 系列之图优化
16:29

编译器为什么要对数据布局转换呢
Layout Transformations? 【AI编
▶ 404 2022-12-17

5 数据布局转换
04(下) 加速芯片张量存储格式?
AI编译器如何进行数据排布?
AI编译器 系列之图优化
13:54

详解AI编译器数据布局转换方法!
华为昇腾处理器的数据布局格式!
▶ 225 2022-12-17

6 内存分配算法
05 动态内存和静态内存啥关系?
有什么AI内存分配算法?!
AI编译器 图优化
17:09

AI编译器内存优化算法! 动态内存
和静态内存区别! 【AI编译器】前
▶ 328 1-16

7 常量折叠原理
06 跟传统编译器常量折叠啥关系?
Constant Folding!!! 折叠一切!!!
AI编译器 系列之图优化
11:06

常量折叠原理! AI编译器常量折叠
跟传统编译器什么关系? 计算图也
▶ 356 2022-12-18

8 公共子表达式
消除 07 CSE, Common
Subexpression
AI编译器 系列之图优化
08:30

编译器公共子表达式消除的方法!
AI编译器消除公共子表达式 【AI编
▶ 270 2022-12-20

9 死代码消除
08 Dead Code Elimination
消除计算图中没用的节点
AI编译器 系列之图优化
06:24

编译器死代码消除的原理! AI编
器死代码消除 【AI编译器】系列之
▶ 375 2022-12-20

10 代数化简原理
09 Algebraic Reduced
数学来了!初中学回来了!
AI编译器 系列之图优化
09:30

编译器的代数化简原理! AI编译器
的代数化简来啦! 【AI编译器】系
▶ 315 2022-12-21

计算图优化

详解 (II)

② Extended Graph Optimizations 其他图优化

- These optimizations include complex node fusions. They are run after graph partitioning and are only applied to the nodes assigned to the CPU or CUDA or ROCm execution provider.
- 有些 Op 在一些框架上可能没有直接的实现，而是通过一些 Op 的组合，如果推理引擎实现了该 Op，就可以把这些组合转成这个 Op，能够使得网络图更加简明清晰。

② Extended Graph Optimizations 其他图优化

Fuse Layer Norm

- 组合实现的 Norm Op 直接转换成一个Op

Fuse PReLU

- 组合实现的 PReLU Op 直接转换成一个Op

Fuse Matmul Transpose

- 有些框架的矩阵乘法Matmul层自身是不带转置操作的，当需要转置的矩阵乘法时需要前面加一个transpose层。如 Onnx 的 Matmul 自身有是否转置的参数，因此可以将前面的transpose层转换为参数即可

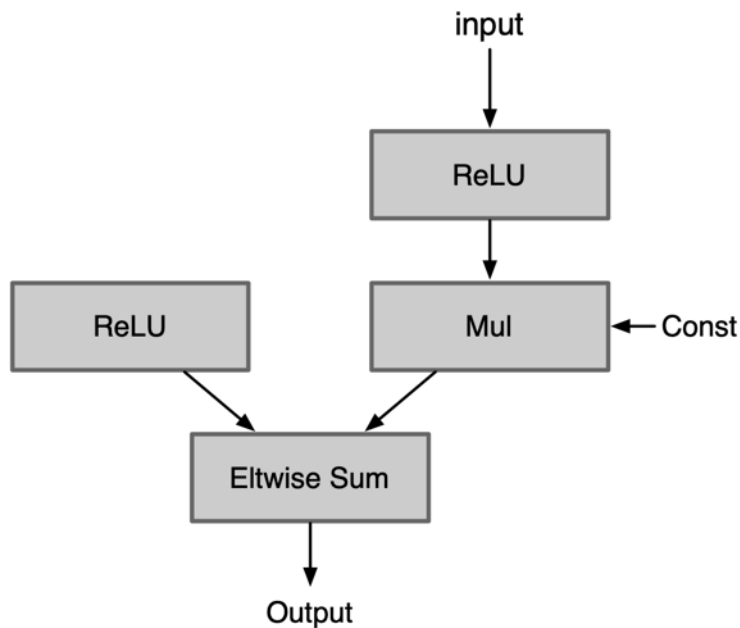
Fuse Binary Eltwise

- $x3 = x1 * b1 + x2 * b2$ ，把 BinaryOp Add 转换成 Eltwise Sum，而 Eltwise Sum是有参数 coeffs，可以完成上述乘法的效果，因此把两个 BinaryOp Mul 的系数融合到 Eltwise Sum 的参数 coeffs

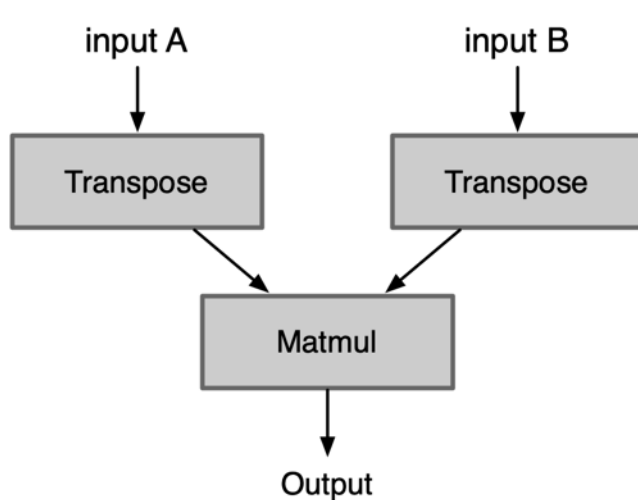
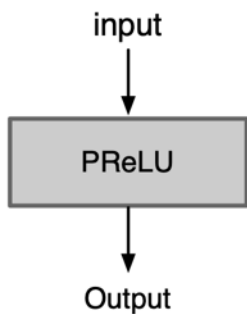
Fuse Reduction with Global Pooling

- 对一个三维 tensor 先后两次分别进行w维度的 reduction mean 和h维度的 reduction mean，最终只剩下c这个维度，就等于进行了一次global_mean_pooling

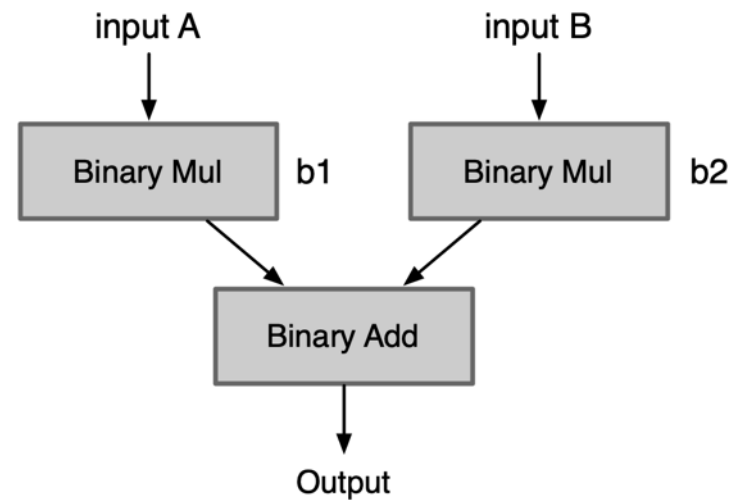
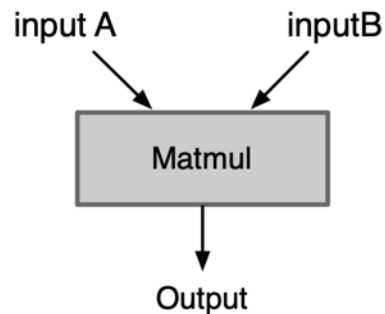
2 Extended Graph Optimizations 其他图优化



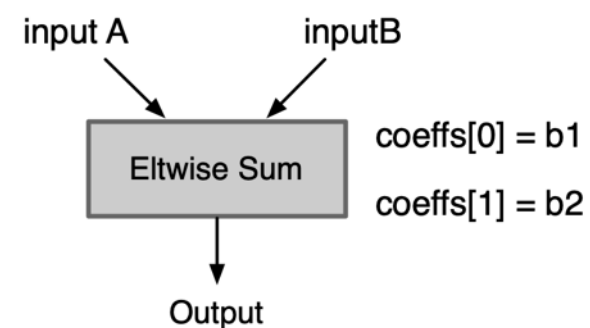
- Fuse PReLU



- Fuse Matmul Transpose



- Fuse Binary Eltwise



② Extended Graph Optimizations 其他图优化

FLASHATTENTION: Fast and Memory-Efficient Exact Attention
with IO-Awareness

Tri Dao[†], Daniel Y. Fu[†], Stefano Ermon[†], Atri Rudra[‡], and Christopher Ré[†]

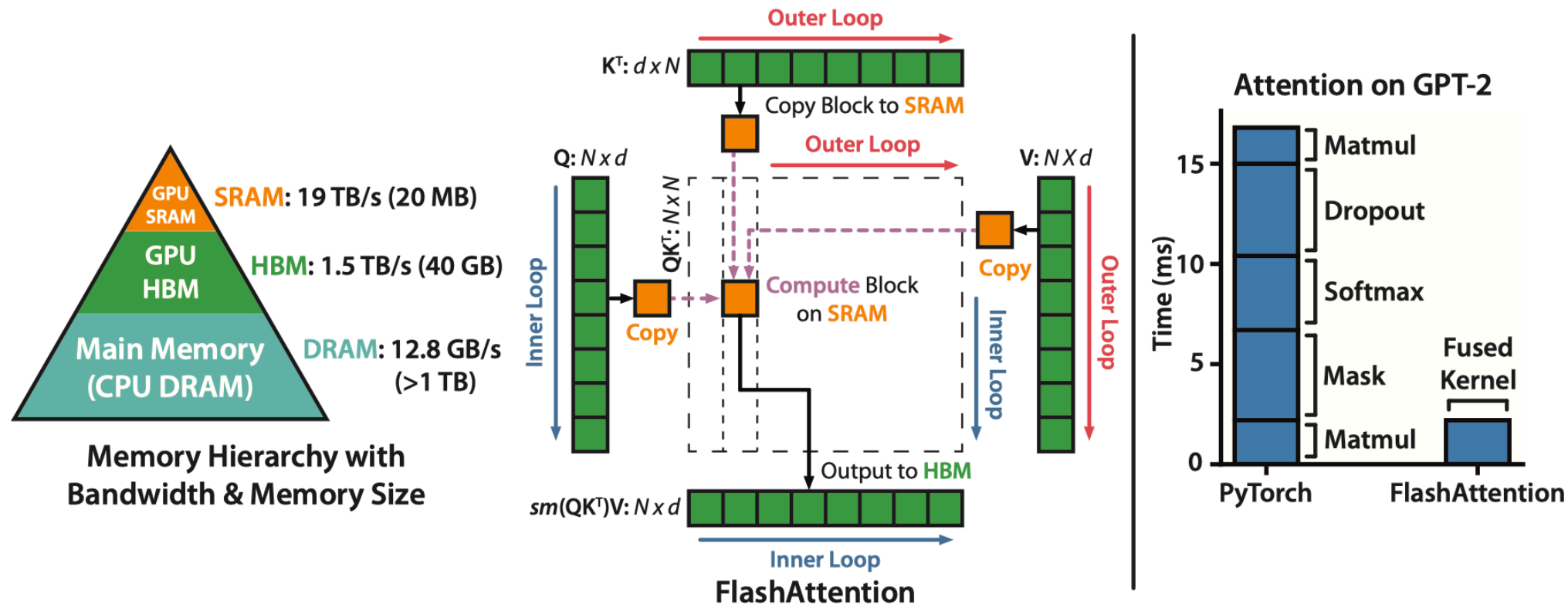


Figure 1: **Left:** FLASHATTENTION uses tiling to prevent materialization of the large $N \times N$ attention matrix (dotted box) on (relatively) slow GPU HBM. In the outer loop (red arrows), FLASHATTENTION loops through blocks of the \mathbf{K} and \mathbf{V} matrices and loads them to fast on-chip SRAM. In each block, FLASHATTENTION loops over blocks of \mathbf{Q} matrix (blue arrows), loading them to SRAM, and writing the output of the attention computation back to HBM. **Right:** Speedup over the PyTorch implementation of attention on GPT-2. FLASHATTENTION does not read and write the large $N \times N$ attention matrix to HBM, resulting in an $7.6\times$ speedup on the attention computation.

计算图优化

详解 (III)

AI编译器之前端优化

▶ 播放全部

!结构在快速演化，底层计算硬件技术更是层出不穷，对于广大开发者来说不仅要考虑如何在复杂多，还要应对计算框架的持续迭代。AI编译器就成了应对以上问题广受关注的技术方向，让用户仅...

默认排序

升序排序

编辑

1 AI编译器的
01 前端优化
AI编译器 系列之图优化
03:46

AI编译器前端"图层优化"内容概
览!! 【AI编译器】系列之前端优
▶ 600 2022-12-13

2 GraphIR有什么用
02 AI编译器如何接收图IR?
计算图内容回顾
AI编译器 系列之图优化
11:11

图层IR(Graph IR)是什么? AI编译器
如何接收图层IR进行优化呢? 【AI编
▶ 519 2022-12-13

3 03 AI编译器
--图优化
算子融合原理
17:48

算子融合了解下! AI编译器如何实
现算子融合的? 【AI编译器】系列
▶ 915 2022-12-16

4 数据布局转换
04(上) Data Layout Transform?
为啥要改变数据的布局?
AI编译器 系列之图优化
16:29

编译器为什么要对数据布局转换呢
Layout Transformations? 【AI编
▶ 404 2022-12-17

5 数据布局转换
04(下) 加速芯片张量存储格式?
AI编译器如何进行数据排布?
AI编译器 系列之图优化
13:54

详解AI编译器数据布局转换方法!
华为昇腾处理器的数据布局格式!
▶ 225 2022-12-17

6 内存分配算法
05 动态内存和静态内存啥关系?
有什么AI内存分配算法?!
AI编译器 图优化
17:09

AI编译器内存优化算法! 动态内存
和静态内存区别! 【AI编译器】前
▶ 328 1-16

7 常量折叠原理
06 跟传统编译器常量折叠啥关系?
Constant Folding!!! 折叠一切!!!
AI编译器 系列之图优化
11:06

常量折叠原理! AI编译器常量折叠
跟传统编译器什么关系? 计算图也
▶ 356 2022-12-18

8 公共子表达式
消除 07 CSE, Common
Subexpression
AI编译器 系列之图优化
08:30

编译器公共子表达式消除的方法!
AI编译器消除公共子表达式 【AI编译
▶ 270 2022-12-20

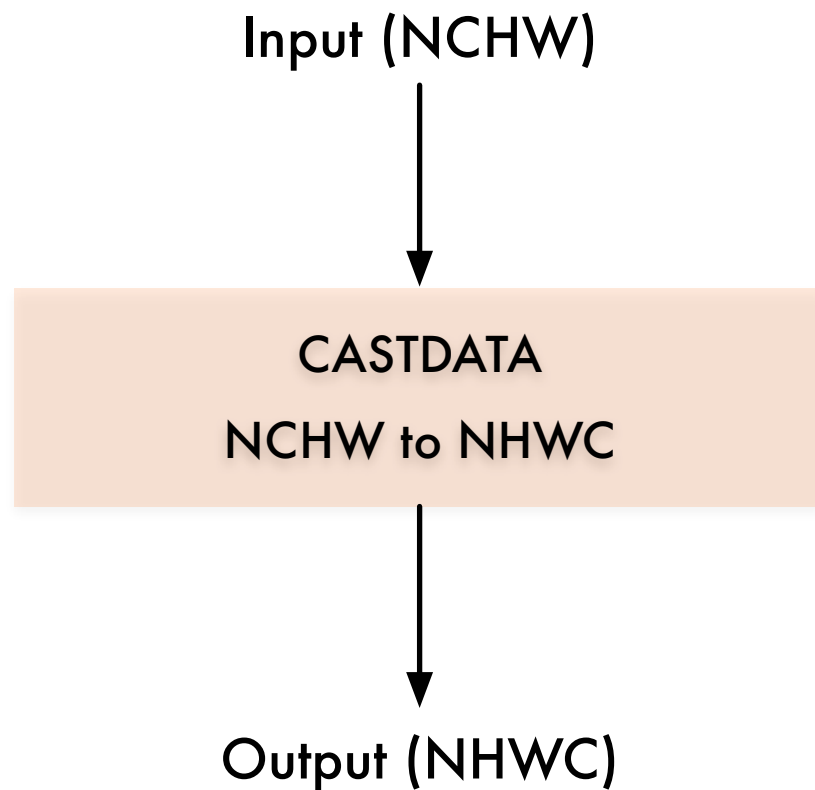
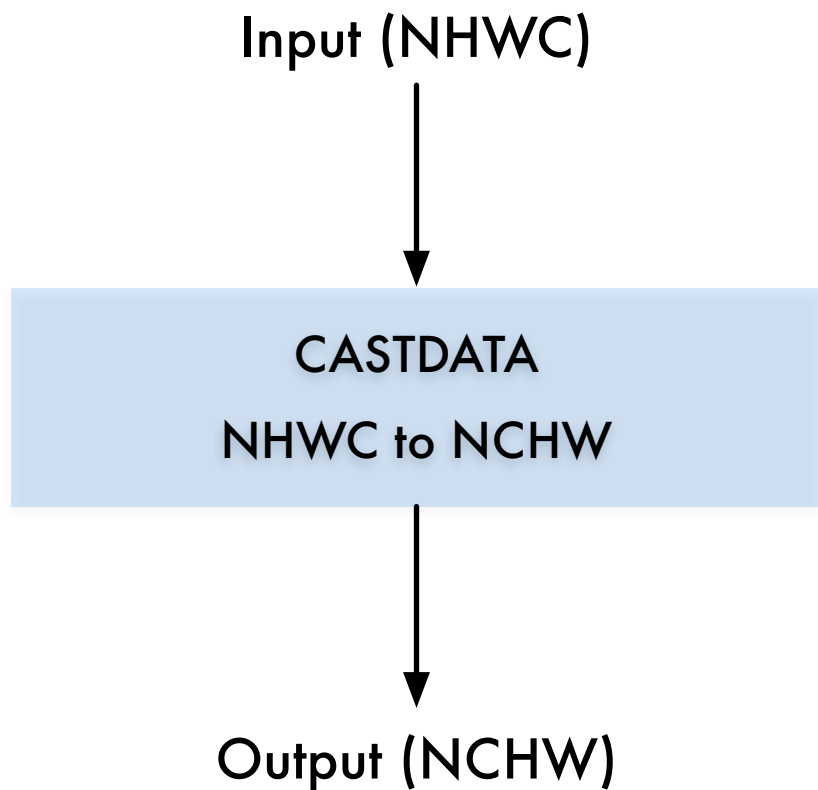
9 死代码消除
08 Dead Code Elimination
消除计算图中没用的节点
AI编译器 系列之图优化
06:24

编译器死代码消除的原理! AI编译
器死代码消除 【AI编译器】系列之
▶ 375 2022-12-20

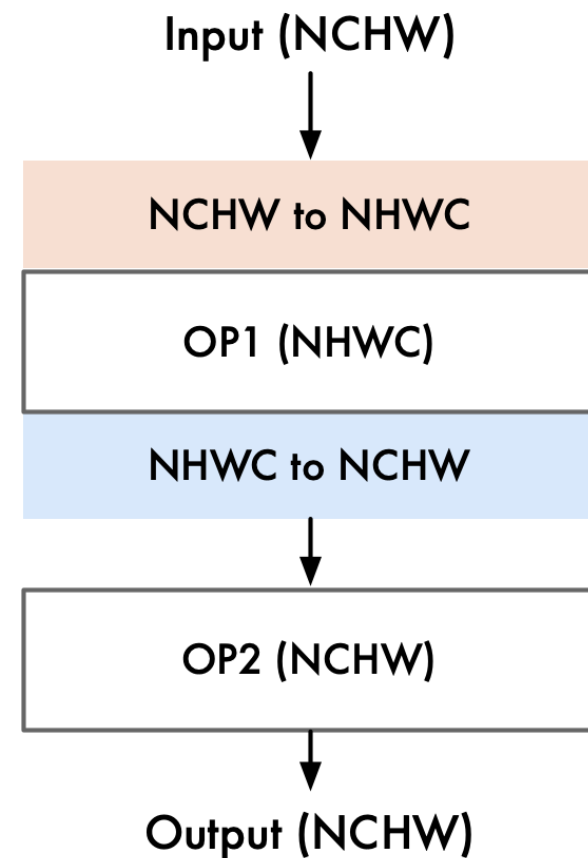
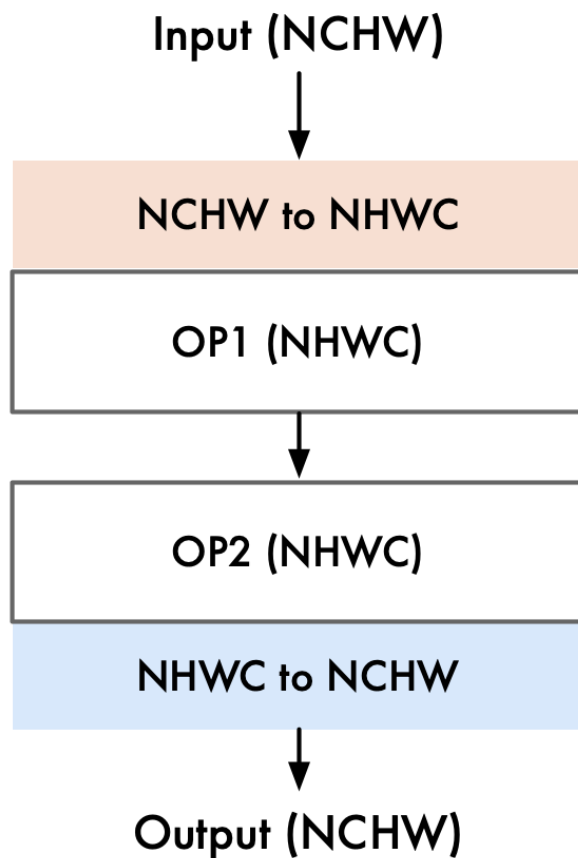
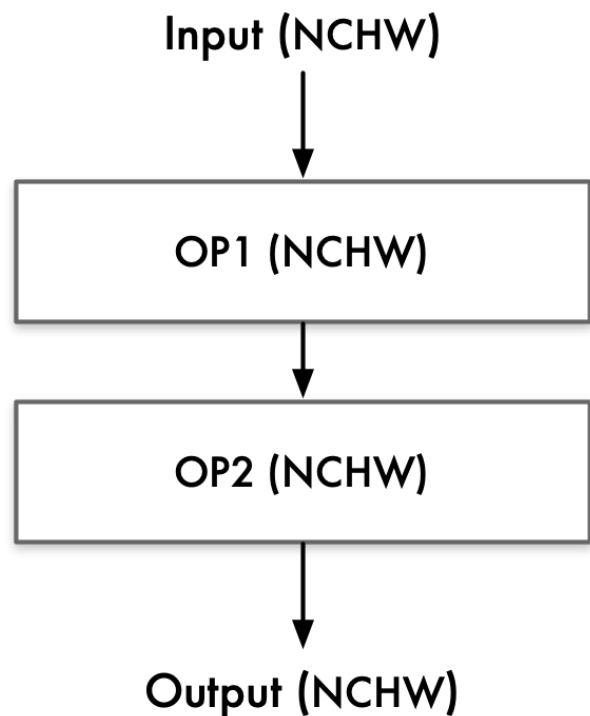
10 代数化简原理
09 Algebraic Reduced
数学来了!初中学回来了!
AI编译器 系列之图优化
09:30

编译器的代数化简原理! AI编译器
的代数化简来啦! 【AI编译器】系
▶ 315 2022-12-21

数据转换节点



常见数据转换节点



AI编译器之前端优化

▶ 播放全部

!结构在快速演化，底层计算硬件技术更是层出不穷，对于广大开发者来说不仅要考虑如何在复杂多，还要应对计算框架的持续迭代。AI编译器就成了应对以上问题广受关注的技术方向，让用户仅...

默认排序

升序排序

编辑



AI编译器前端"图层优化"内容概览!! 【AI编译器】系列之前端优
▶ 600 2022-12-13



图层IR(Graph IR)是什么? AI编译器如何接收图层IR进行优化呢? 【AI编
▶ 519 2022-12-13



算子融合了解下! AI编译器如何实现算子融合的? 【AI编译器】系列
▶ 915 2022-12-16



编译器为什么要对数据布局转换呢 Layout Transformations? 【AI编
▶ 404 2022-12-17



详解AI编译器数据布局转换方法! 华为昇腾处理器的数据布局格式!
▶ 225 2022-12-17



AI编译器内存优化算法! 动态内存和静态内存区别! 【AI编译器】前
▶ 328 1-16



常量折叠原理! AI编译器常量折叠跟传统编译器什么关系? 计算图也
▶ 356 2022-12-18



编译器公共子表达式消除的方法! AI编译器消除公共子表达式 【AI编译
▶ 270 2022-12-20



编译器死代码消除的原理! AI编译器死代码消除 【AI编译器】系列之
▶ 375 2022-12-20

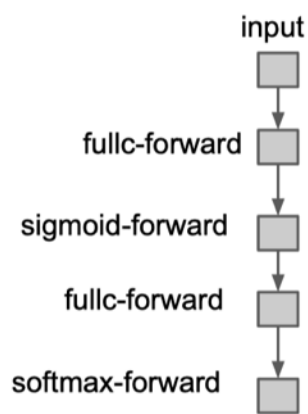


编译器的代数化简原理! AI编译器的代数化简来啦! 【AI编译器】系
▶ 315 2022-12-21

内存优化方法

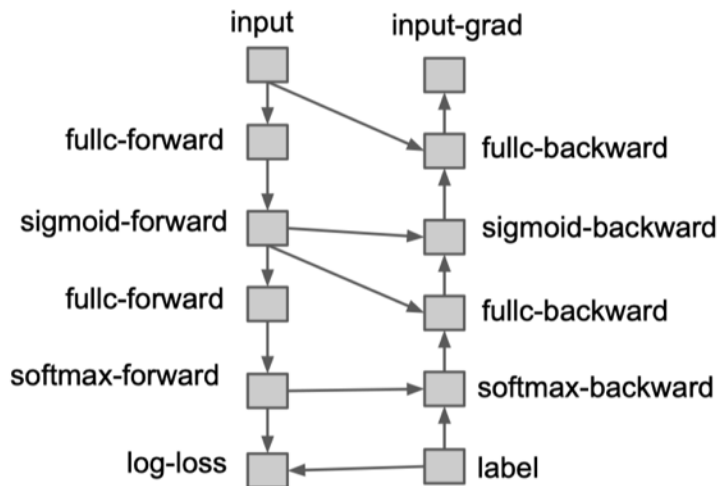
- Inplace operation : 如果一块内存不再需要, 且下一个操作是element-wise, 可以原地覆盖内存
- Memory sharing : 两个数据使用内存大小相同, 且有一个数据参与计算后不再需要, 后一个数据可以覆盖前一个数据

Network Configuration



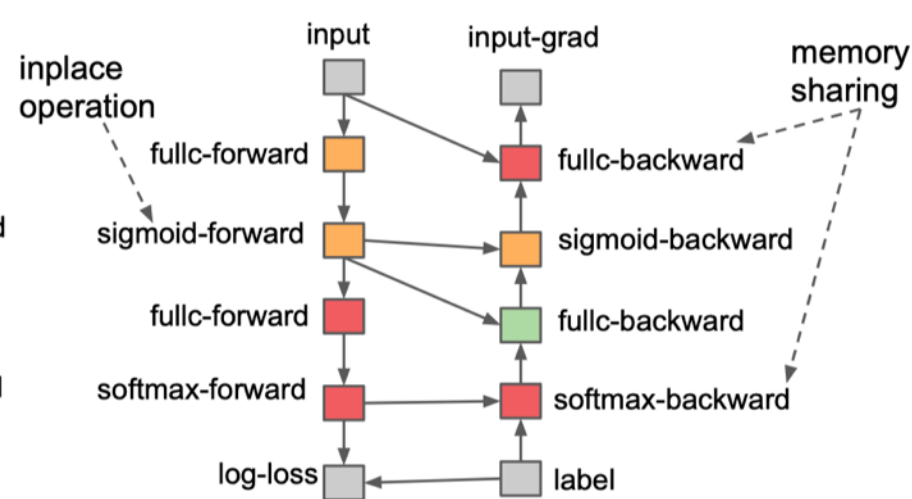
→ data dependency

Gradient Calculation Graph



□ Memory allocation for each output of op, same color indicates shared memory.

A Possible Allocation Plan





BUILDING A BETTER CONNECTED WORLD

THANK YOU

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.