week18实验记录

zxp

January 20, 2024

1 environment

cpu:Inter i5-9300H (2.4 GHz)

System:Ubuntu 22.04.1

Compiler:gcc 12.3

环境变化是放假了，电脑放在了实验室，这周用笔记本跑的实验。因为笔记本性能更差，按原来那样测要太久，所以把批次改成了1，并且改成测20次取平均值。

2 code

写了一版不是由初始的input张量转换成im2win张量而是直接用随机数赋值的版本。对im2win进行了优化。尝试改变im2win的数据布局。

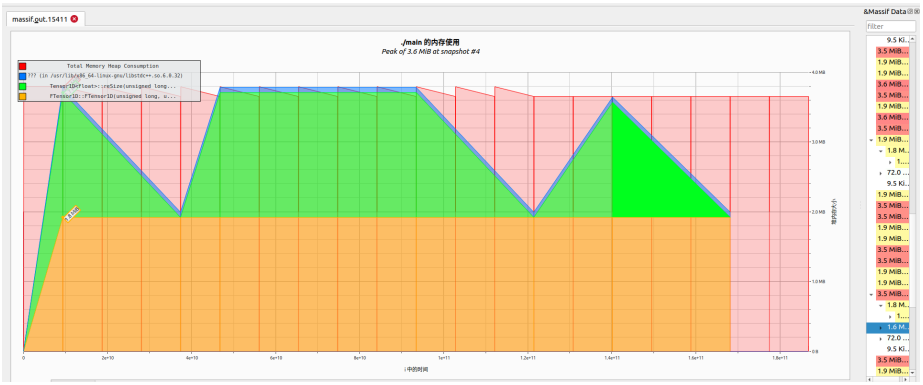

Figure 1: 初始im2win的内存情况

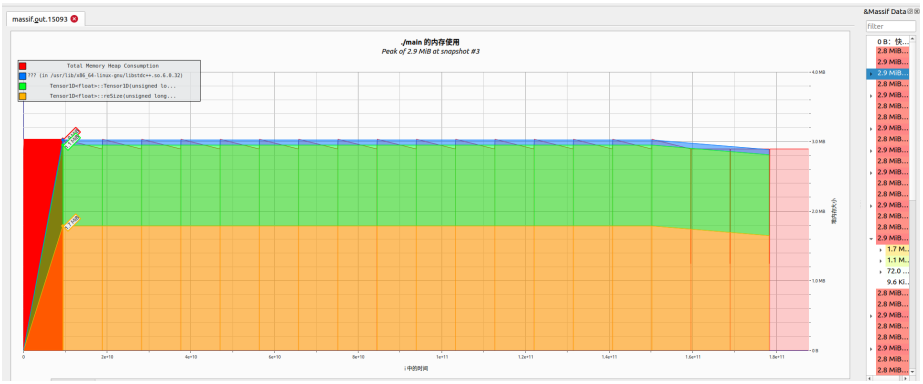

Figure 2: 直接用随机数赋值的内存情况

3 Experiment

测试了不是由初始的input张量转换成im2win张量而是直接用随机数赋值的性能。下面是在O0优化下conv1情况下两种不同优化的内存情况。

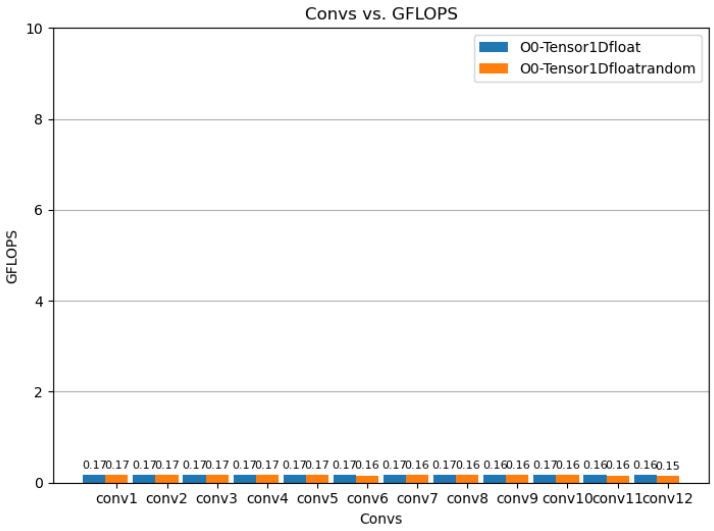

Figure 3: O0

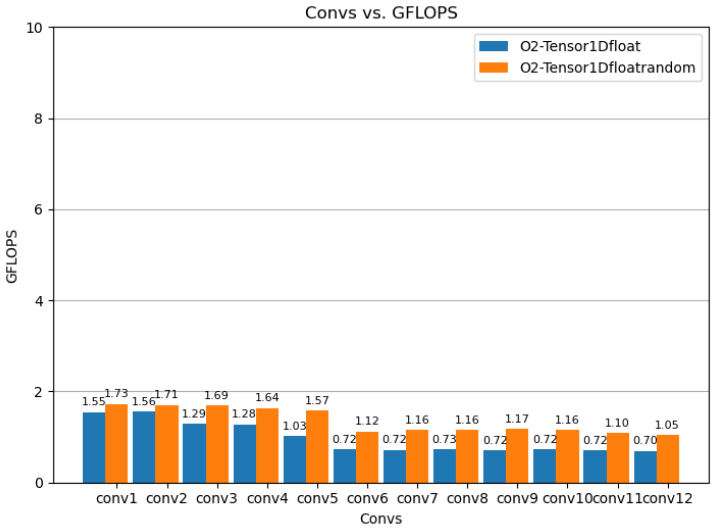

Figure 4: O2

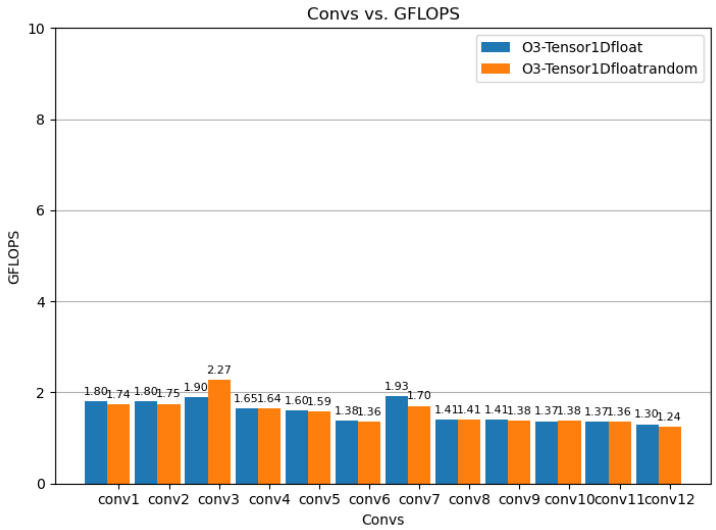

Figure 5: O3

3.1 Analysis

在O2优化下，直接用随机数赋值性能比由初始的input张量转换成im2win张量要好，其他情况十分接近。少了初始的input张量占用内存，直接用随机数赋值的内存占用肯定是比由初始的input张量转换成im2win张量要小的。但是输出张量是下一层的输入张量，im2win卷积得到的输出张量和直接卷积得到的输出张量布局上是一样的，即使在第一层减少内存开销，后面的层还是得花im2win变化的内存开销，而且这是直接随机赋值，读图片像素可能不会像随机赋值这样顺利。

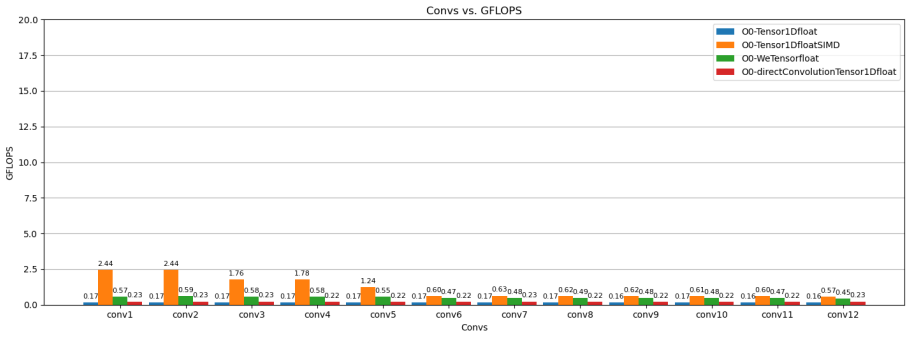

Figure 6: O0

4 Experiment2

对im2win进行了优化，用于对比使用float和double的区别，这周只写了使用float的，因为之前float和double区别不大，应该是计算限制，做的优化主要是使用SIMD和FMA和index-hosting和hosting。使用simd的地方是窗口部分，这是im2win比直接卷积好的地方，直接卷积因为步长的限制和核的大小的限制，步长让input张量参与计算数据在内存中不连续，核很多大小是3x3，不好使用SIMD。在256位寄存器能放8个float数据，但是im2win用来计算的窗口是核展开成1纬的大小，3x3的核在内存中也有9个是连续的，并且im2win变换input张量参与计算数据也好放进simd，然后将这些数据放入256位寄存器中然后使用FMA。hosting优化单纯的hosting了一个输出张量的元素。

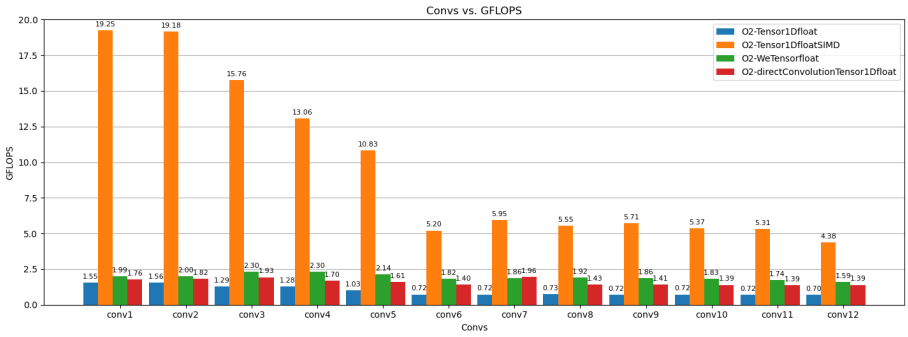

Figure 7: O2

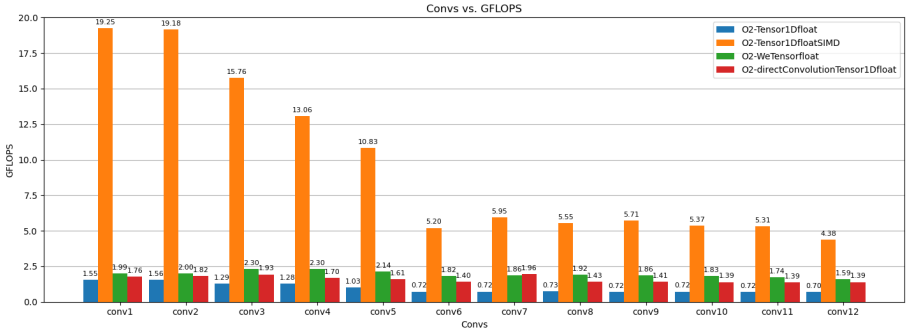

Figure 8: O3

4.1 Analysis

效果十分显著，特别是在conv1-5上面。后面的conv卷积核都是3x3，连续性的部分少，所以优化效果更不明显。

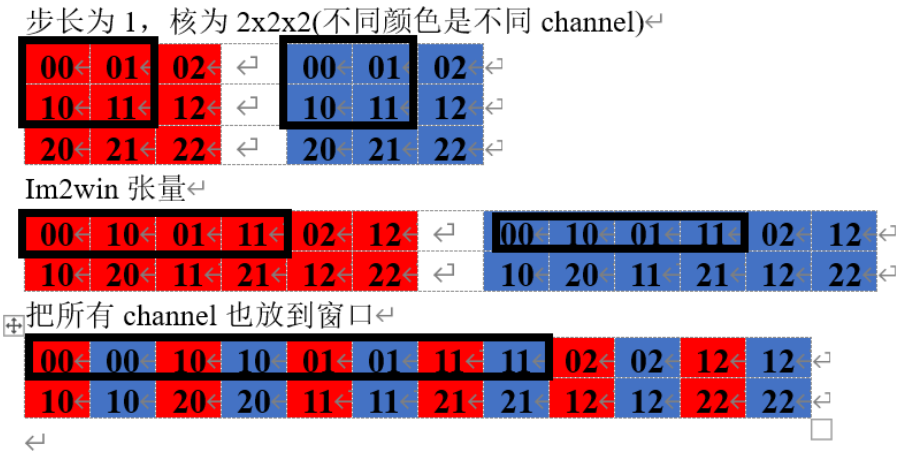

Figure 9: 示意图

5 Experiment3

我觉得可以试试另一种数据布局，之前写的im2win变换在channel上是并排的，但input张量和卷积核的channel是一一对应的，每个channel都是一个核的hight*width大小的窗口在im2win张量对应的一个上channel滑动，如果把所有channel也放到这个窗口，数据连续性的部分会更大，只是改变了数据布局占的内存大小不会增加。
测试时使用的循环顺序不一样，原本的最内三层是(filterchannel,filterwidth,filterhight)，改变数据布局（图中黄色的块）使用的是(filterwidth,filterhight,ilterchannel)，因为这些循环顺序对数据布局的数据连续性比较友好。

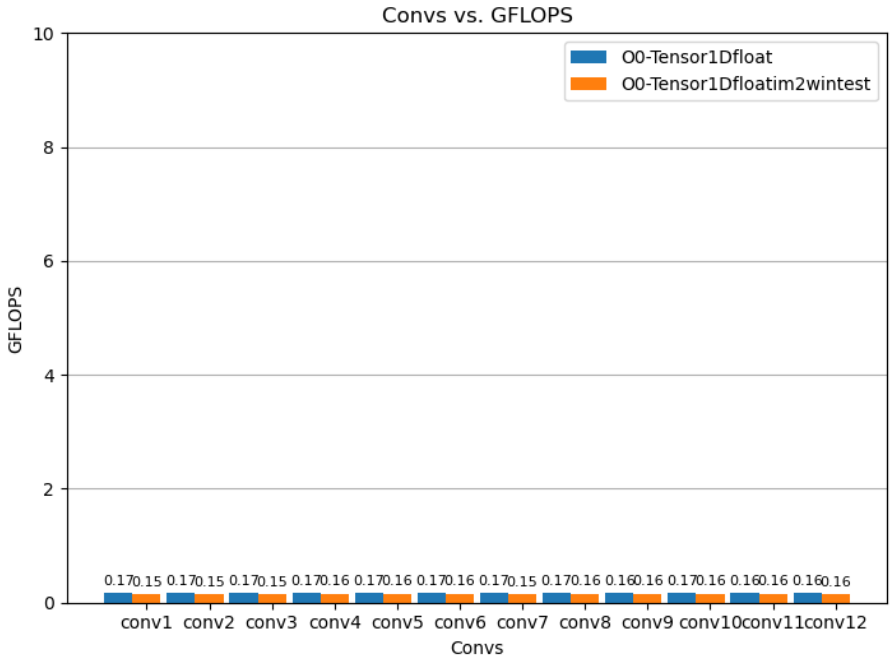
Convs vs. GFLOPS
O0-Tensor1Dfloat
O0-Tensor1Dfloatim2wintest
GFLOPS
Convs
0
2
4
6
8
10
0.17 0.15
0.17 0.15
0.17 0.15
0.17 0.16
0.17 0.16
0.17 0.16
0.17 0.15
0.17 0.16
0.16 0.16
0.17 0.16
0.16 0.16
0.16 0.16
conv1
conv2
conv3
conv4
conv5
conv6
conv7
conv8
conv9
conv10
conv11
conv12

Figure 10: O0

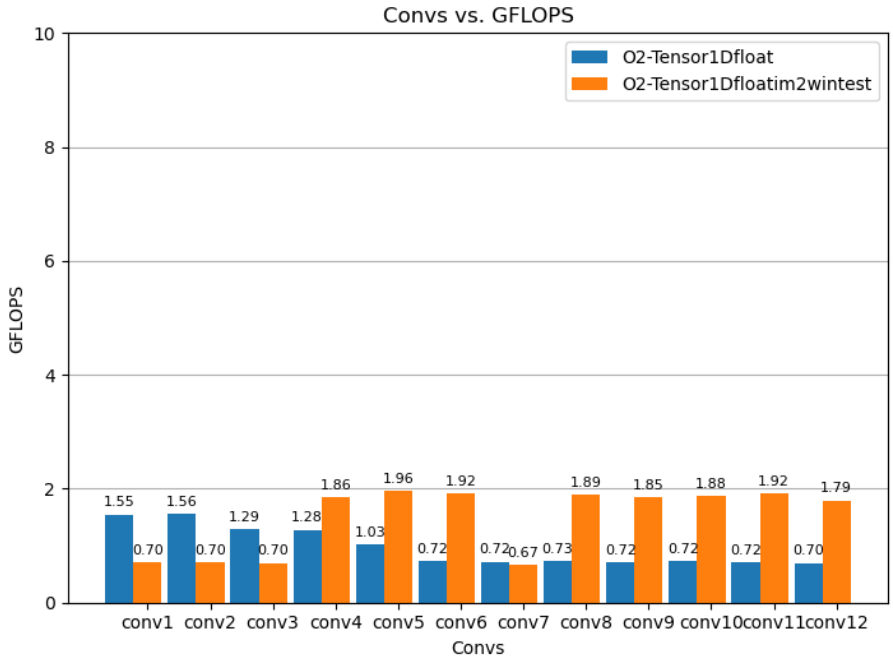
Convs vs. GFLOPS
O2-Tensor1Dfloat
O2-Tensor1Dfloatim2wintest
GFLOPS
Convs
0
2
4
6
8
10
1.55
0.70
1.56
0.70
1.29
0.70
1.28
1.86
1.03
1.96
0.72
1.92
0.72
0.67
0.73
1.89
0.72
1.85
0.72
1.88
0.72
1.92
0.70
1.79
conv1
conv2
conv3
conv4
conv5
conv6
conv7
conv8
conv9
conv10
conv11
conv12

Figure 11: O2

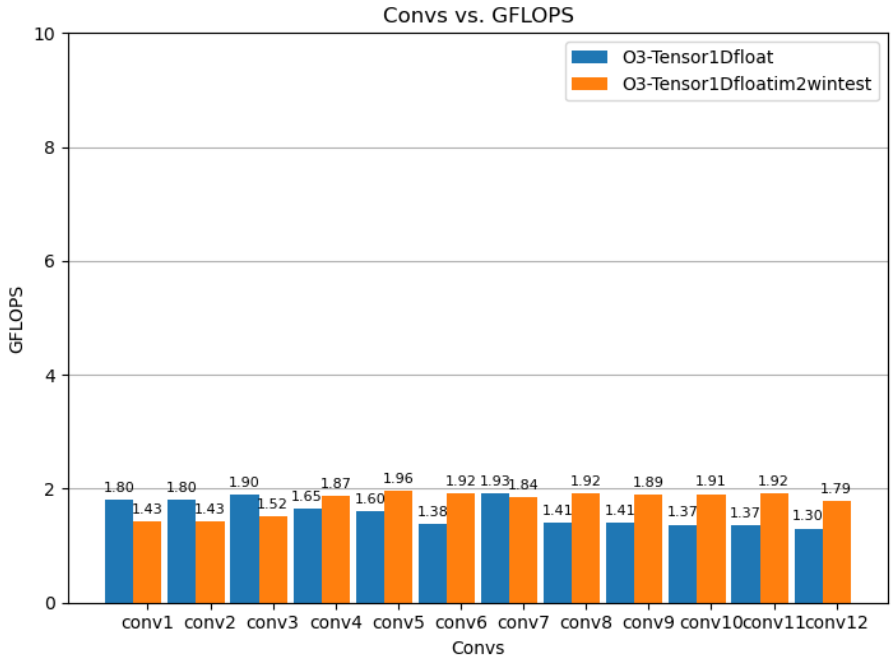

Figure 12: O3

5.1 Analysis

O0情况下性能差距不大，但是O2O3下conv1，2，3，7之前的布局性能要比这种好，其他conv则是这种布局好一点，表现不好可能是核太大缓存放不下，表现好应该是数据连续性好一点。