本周主要做的是优化im2win，主要分为了三个方向去优化Im2win。

第一个方向是改变输入张量和卷积核张量按照通道这个维度展开的大小，

第二个方向是改变数据存储方式为数据对齐

第三个方向是对输出张量融合不同的维度组合

因为我的电脑出问题了，开不了机，数据是跑出来了，但是处理数据的脚本是在之前的电脑上面。所以我就看实验结果大概总结一下。、

第一个方向我的做法是在conv1 conv2 conv3 conv4 conv7上面对输入张量的通道不展开，然后其他conv 在原来全部展开的基础上/2，/4也就是原来输入张量维度512，全部展开，现在我只展开256 或者128层。因为之前猜测会不会因为im2win输入张量相邻两个窗口之间隔的元素太多了。在conv1 conv2上面的步长是4，在横向和纵向的步长都是4.，当我们im2win变化后，横向步长变为了4 x 卷积核的高， 纵向步长变为1，经过按照卷积核通道维度展开后，横向步长变为了 4 x 卷积核高 x 卷积核通道，纵向1。 直接卷积上横向步长为4 x 卷积核通道， 纵向不变。也就是说直接卷积的横向步长是im2win 1/卷积核高，纵向步长是im2win的4倍。所以我们可以通过改变按照通道展开的大小来修改im2win的横向步长。

说下结果， 在conv1 conv2 conv3 conv4 conv7层都是正优化，他们都是全部没展开，

而 其他层都是负优化，在conv5上来说，展开 大小/2 大概降了100gflops, 展开/4大概相比原来全展开下降了200gflops.

Conv1 conv2 都快了100，这里面就只有conv7这一层超过了直接卷积，conv7可以到700。

第二个方向上，采用第一种内存对齐的方式有效果，第二种效果并不是很明显。

采用第一种5 6 层都可以到2000，这比之前快了300,其他不好做比较，没有整体的图。

第三个方向上，我尝试了b+c+h+w, b+c+h, c+h+w ,c+h , h+w 这5中融合方式，测出来的结果都不太理想，都没有融合 b+c这种方式好。

我觉得因为在直接卷积上面因为每个conv的最优化的卷积分块大小，unrolling大小都是一个一个试出来的，分块大小就是先从1 慢慢加，中间会有一个分块大小的gflops到达最大值，然后再慢慢下降。按照输出张量的分块大小其实也不用全部一个一个是，因为有些conv层他们卷积核宽高一样，步长一样，那么基本上最优化的分块大小也差不多，例如conv1和conv2，他们对输出张量的宽分块大小相近基本上就能达到最优。