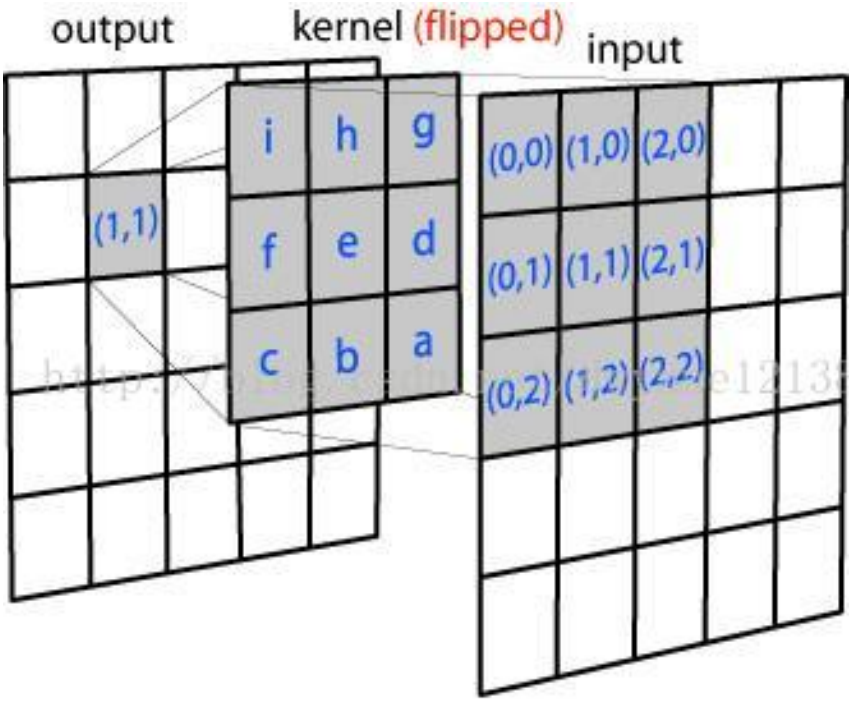
1、矩阵卷积

卷积就是卷积核跟图像矩阵的运算。卷积核是一个小窗口，记录的是权重。卷积核在输入图像上按步长滑动，每次操作卷积核对应区域的输入图像，将卷积核中的权值和对应的输入图像的值相乘再相加，赋给卷积核中心所对应的输出特征图的一个值，如下图所示（这里卷积核要旋转180 °）：

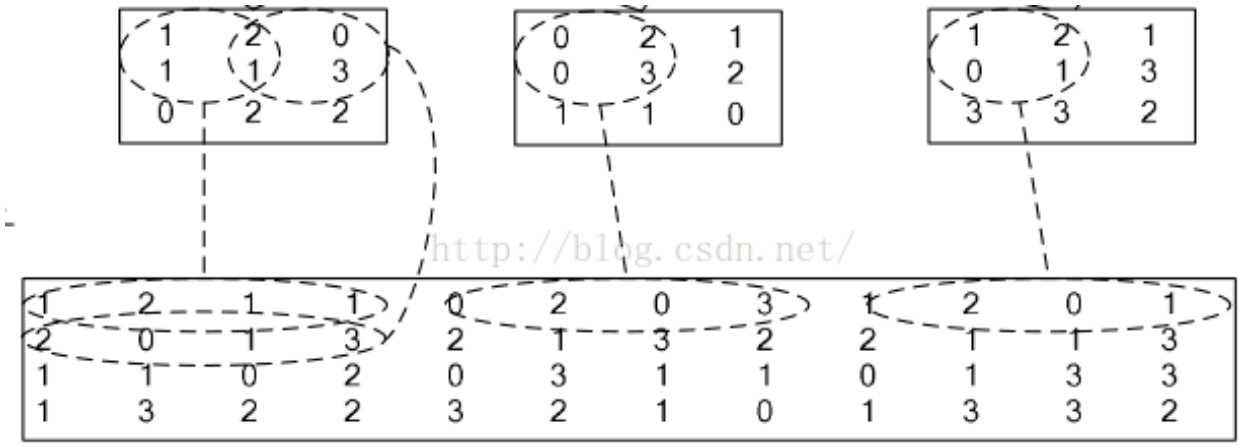


2、im2col的实现

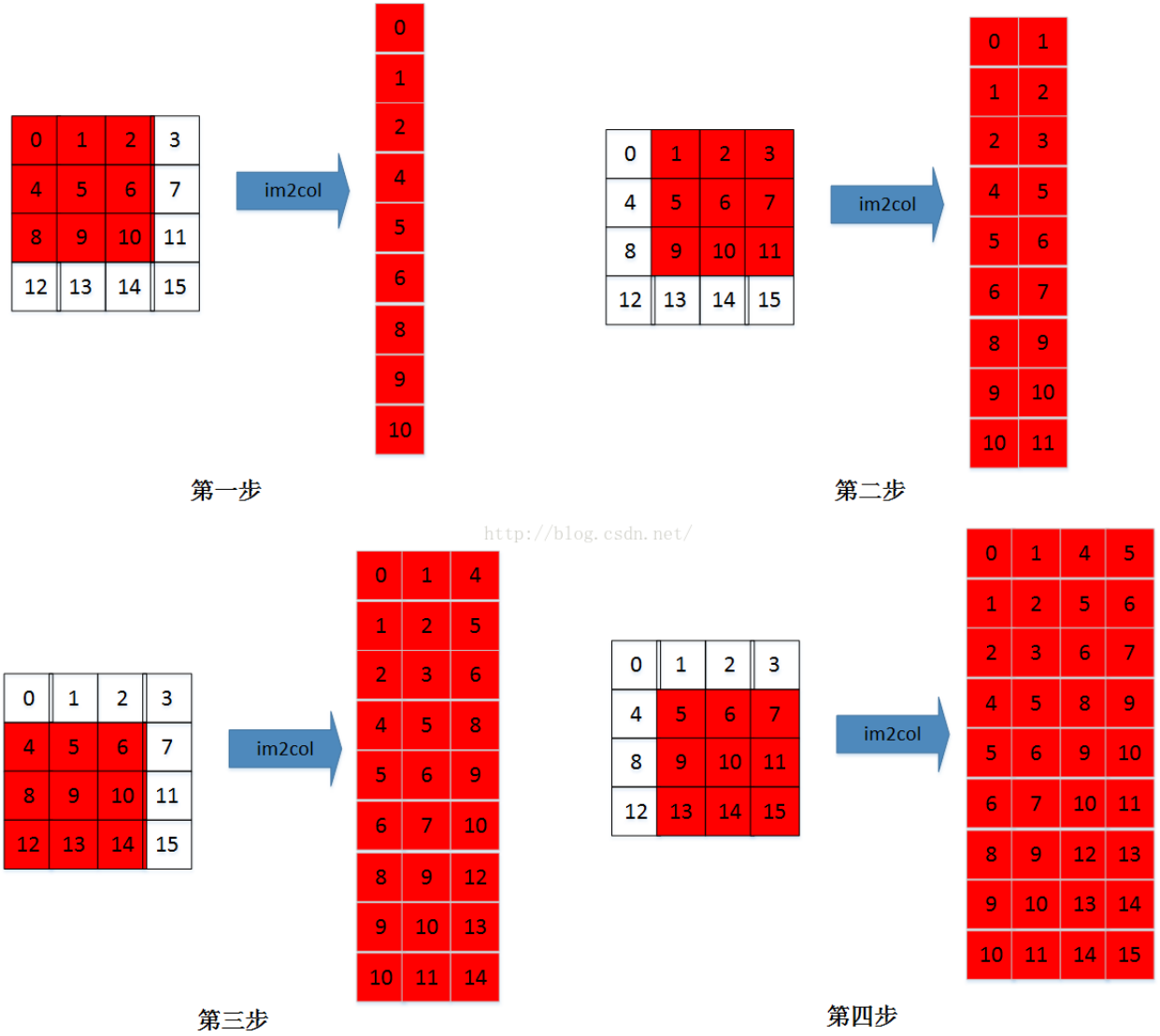
2.1 im2col有什么作用

以上我们已经知道了卷积是如何操作的，im2col的作用就是优化卷积运算，如何优化呢，我们先学习一下这个函数的原理。

我们假设卷积核的尺寸为2*2，输入图像尺寸为3*3.im2col做的事情就是对于卷积核每一次要处理的小窗，将其展开到新矩阵的一行（列），新矩阵的列（行）数，就是对于一副输入图像，卷积运算的次数（卷积核滑动的次数），如下图所示：



以最右侧一列为例，卷积核为2*2，所以新矩阵的列数就为4；步长为一，卷积核共滑动4次，行数就为4. 再放一张图应该看得更清楚。



输入为4*4，卷积核为3*3，则新矩阵为9*4 。看到这里我就产生了一个疑问：我们把一个卷积核对应的值展开，到底应该展开为行还是列呢？卷积核的滑动先行后列还是相反？区别在哪？

这其实主要取决于我们使用的框架访存的方式。计算机一次性读取相近的内存是最快的，尤其是当需要把数据送到GPU去计算的时候，这样可以节省访存的时间，以达到加速的目的。不同框架的访存机制不一样，所以会有行列相反这样的区别。在caffe框架下，im2col是将一个小窗的值展开为一行，而在matlab中则展开为列。所以说，**行列的问题没有本质区别，目的都是为了在计算时读取连续的内存。**

这也解释了我们为什么要通过这个变化来优化卷积。**如果按照数学上的步骤做卷积读取内存是不连续的，这样就会增加时间成本。同时我们注意到做卷积对应元素相乘再相加的做法跟向量内积很相似，所以通过im2col将矩阵卷积转化为矩阵乘法来实现。**

2.2 一个简单实现

基于以上对于im2col的理解，自己写了一个简单实现(c++ opencv3.2)。

```
#include
#include
usingnamespacestd;
usingnamespace cv;
int main()
{
    Mat img =imread("1.jpg");
    Mat kernel=Mat::ones(9,9,CV_8UC1);
  int stride=5;
  int kernum_w,kernum_h;
  if(img.rows%stride==0)
        kernum_w=img.rows/stride;
  else
        kernum_w=img.rows/stride+1;
  if(img.cols%stride==0)
        kernum_h=img.cols/stride;
  else
        kernum_h=img.cols/stride+1;
  int out_w=kernum_w*kernum_h;
  int out_h=kernel.cols*kernel.rows;
  int channel_size=img.cols*img.rows;
  //初始化输入、卷积核并计算新矩阵的尺寸
    Mat C=Mat::zeros(out_w,out_h,CV_8UC3);
  for (int channel=img.channels();channel--;channel>0)
    {
    int i=0;
    for (int im_row=0;im_row)
        {
```

```
        for (int im_col=0;im_col)
            {
            for (int ker_row=0;ker_row)
                    {
                for(int ker_col=0;ker_col)
                        {
                    int input_row=im_row-(kernel.rows-1)/2+ker_row;
                    int input_col=im_col-(kernel.cols-1)/2+ker_col;
                    //索引当前小窗内元素对应到输入的行列值

if(input_row<0||input_col<0||input_row>=img.rows||input_col>=img.cols)
                                        C.at(i,ker_col+kernel.cols*ker_row)[channel]=0;
                    else
                                        C.at(i,ker_col+kernel.cols*ker_row)
[channel]=img.at(input_row,input_col)[channel];
   //超出原图的范围，则新矩阵对应的位置赋零，否则将输入赋给对应的位置
                        }
                    }
                    i++;
            }
            }
       }
       imshow("im2col",C);
       waitKey(0);
   return0;
}
```

对于c++我也只是初学者，关于指针，我还要多加学习，这里对图像的遍历使用了Mat类的成员函数at（），也是一种比较低效率的方法。高效的方法（指针方法）参考这篇 http://blog.csdn.net/daoqinglin/article/details/23628125

参考文献

http://lib.csdn.net/article/aiframework/62849（这里有caffe中的源码解读）

http://blog.csdn.net/mrhiuser/article/details/52672824