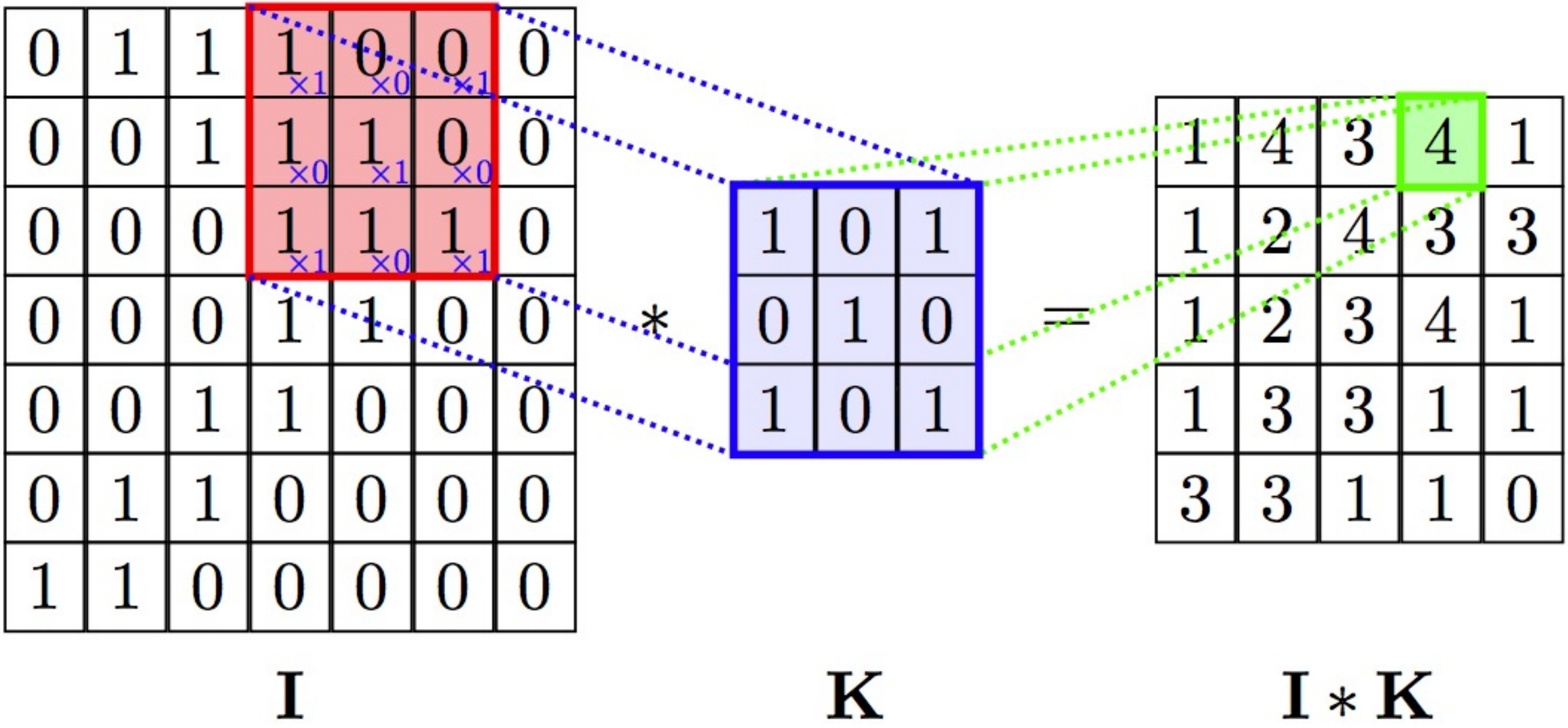

0 1 1 1 0 0 0
0 0 1 1 1 0 0
0 0 0 1 1 1 0
0 0 0 1 1 0 0
0 0 1 1 0 0 0
0 1 1 0 0 0 0
1 1 0 0 0 0 0
×1 ×0 ×1
×0 ×1 ×0
×1 ×0 ×1
*
1 0 1
0 1 0
1 0 1
=
1 4 3 4 1
1 2 4 3 3
1 2 3 4 1
1 3 3 1 1
3 3 1 1 0
I
K
I ∗ K

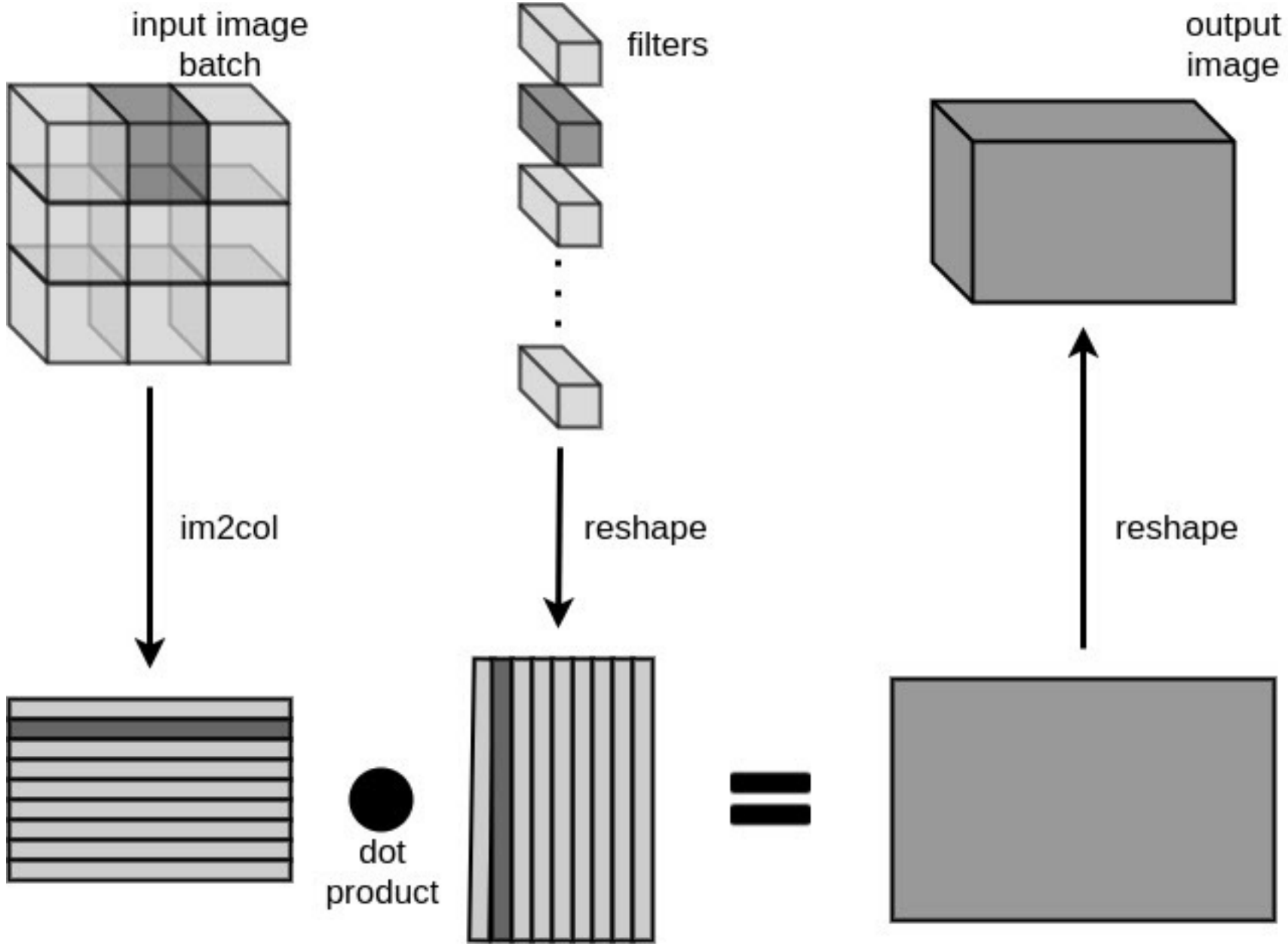
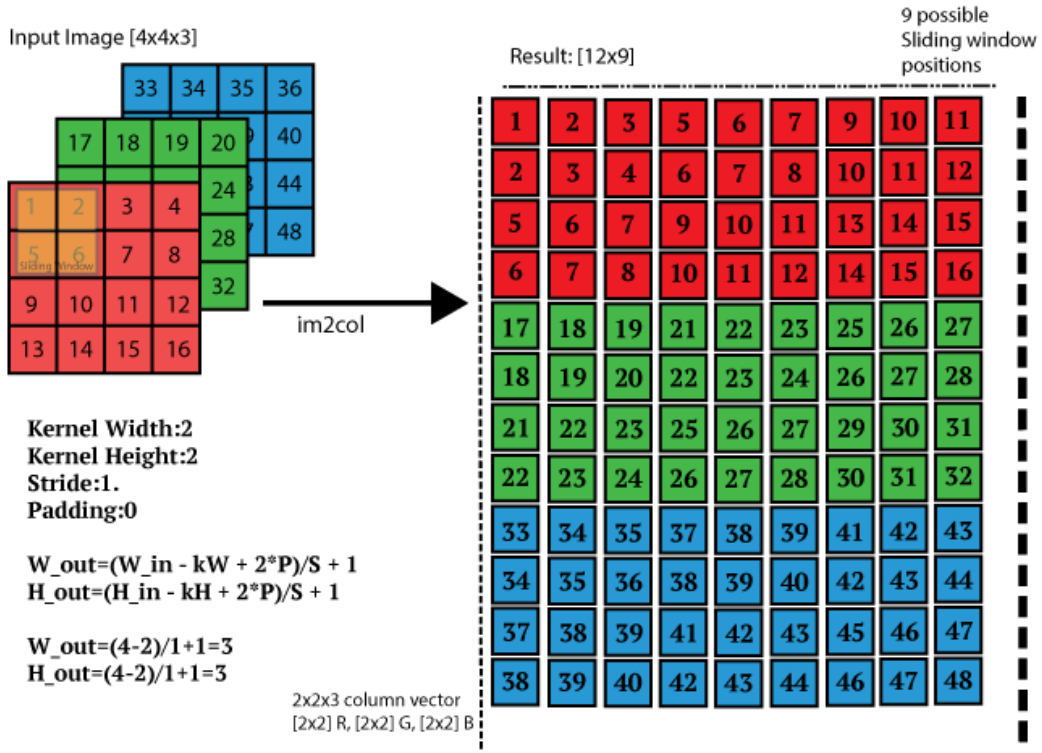
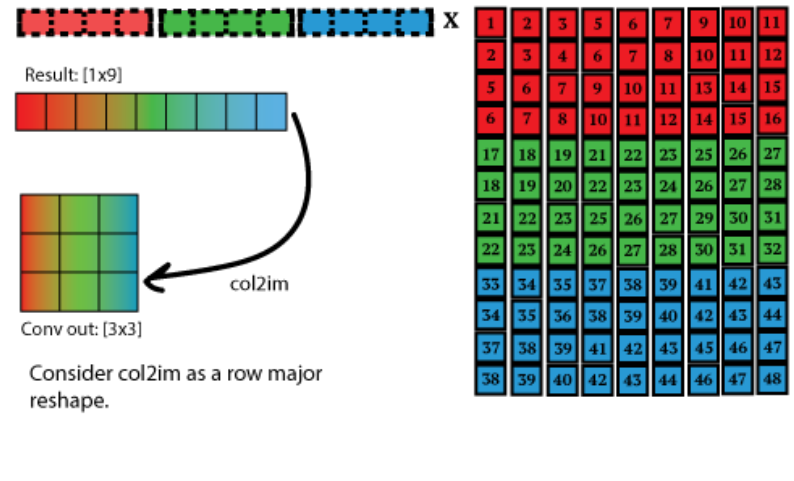

input image
batch
filters
output
image
im2col
reshape
reshape
dot
product

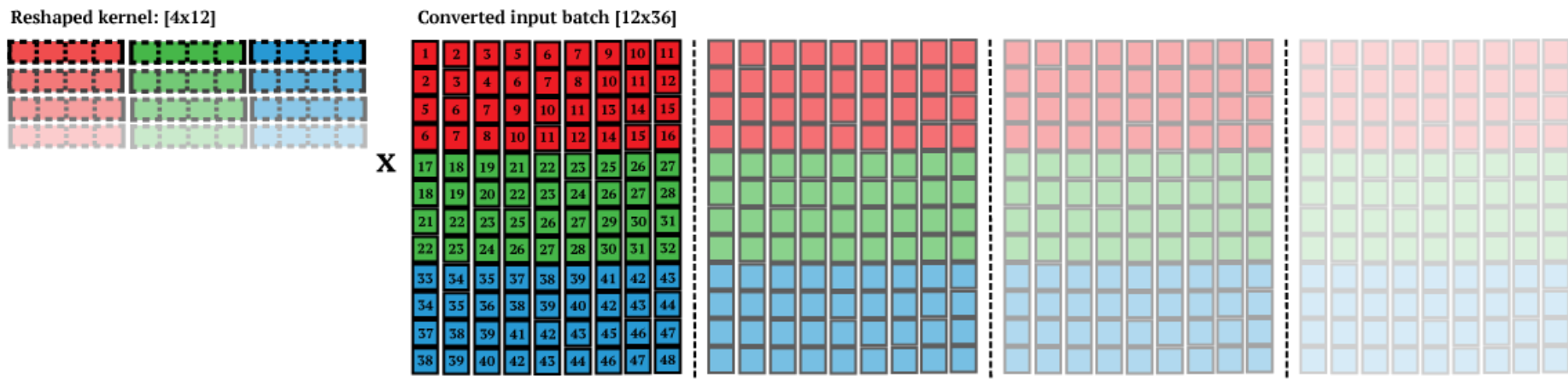
Image to column operation (im2col)
Slide the input image like a convolution but each patch become a column vector.



We can multiply this result matrix [12x9] with a kernel [1x12].
result = kernel x matrix
The result would be a row vector [1x9].
We need another operation that will convert this row vector into a image [3x3].



We get true performance gain
when the kernel has a large number of filters, ie: F=4
and/or you have a batch of images (N=4). Example for the input batch [4x4x3x4], convolved with 4 filters [2x2x3x2] .
The only problem with this approach is the amount of memory



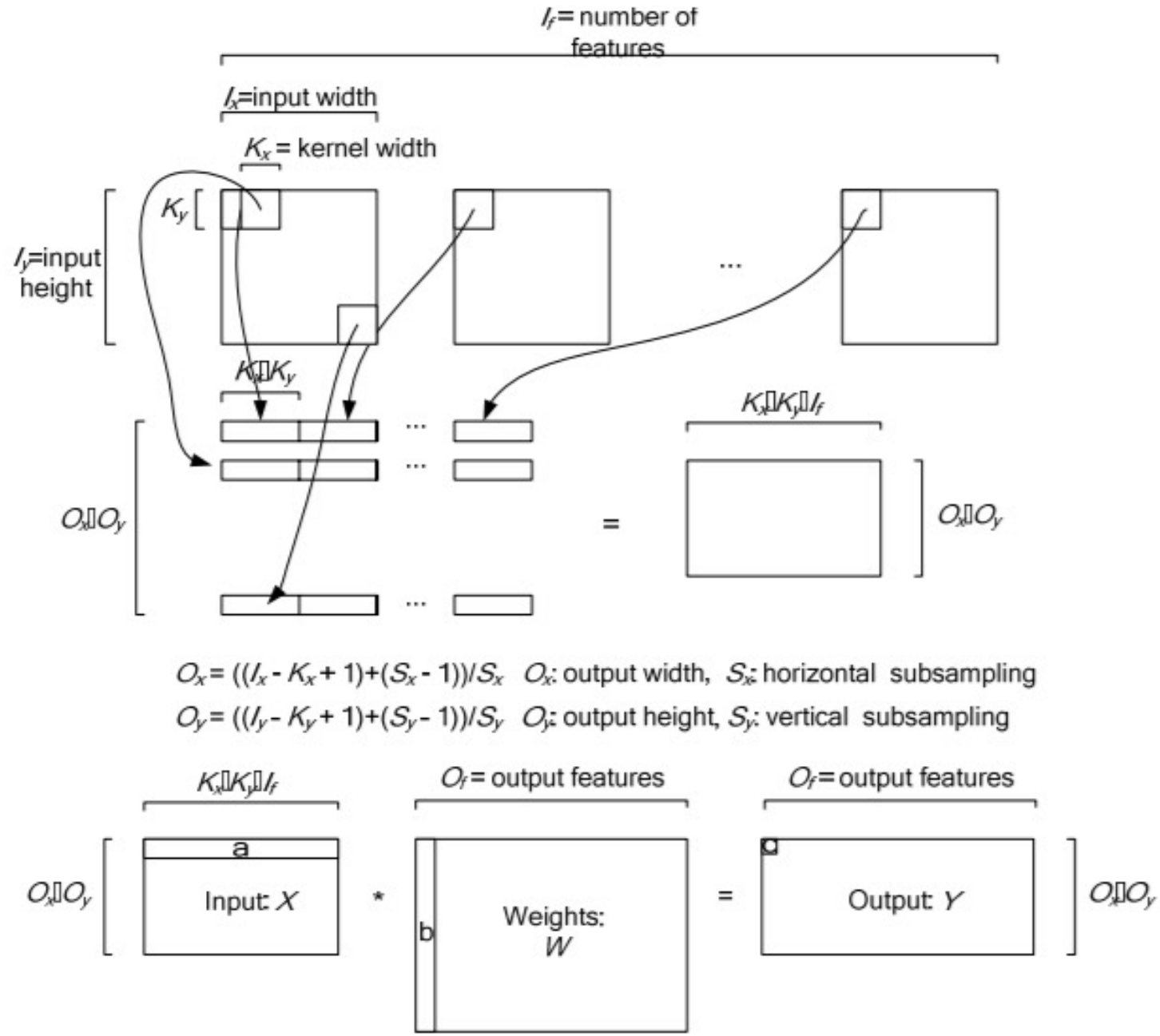
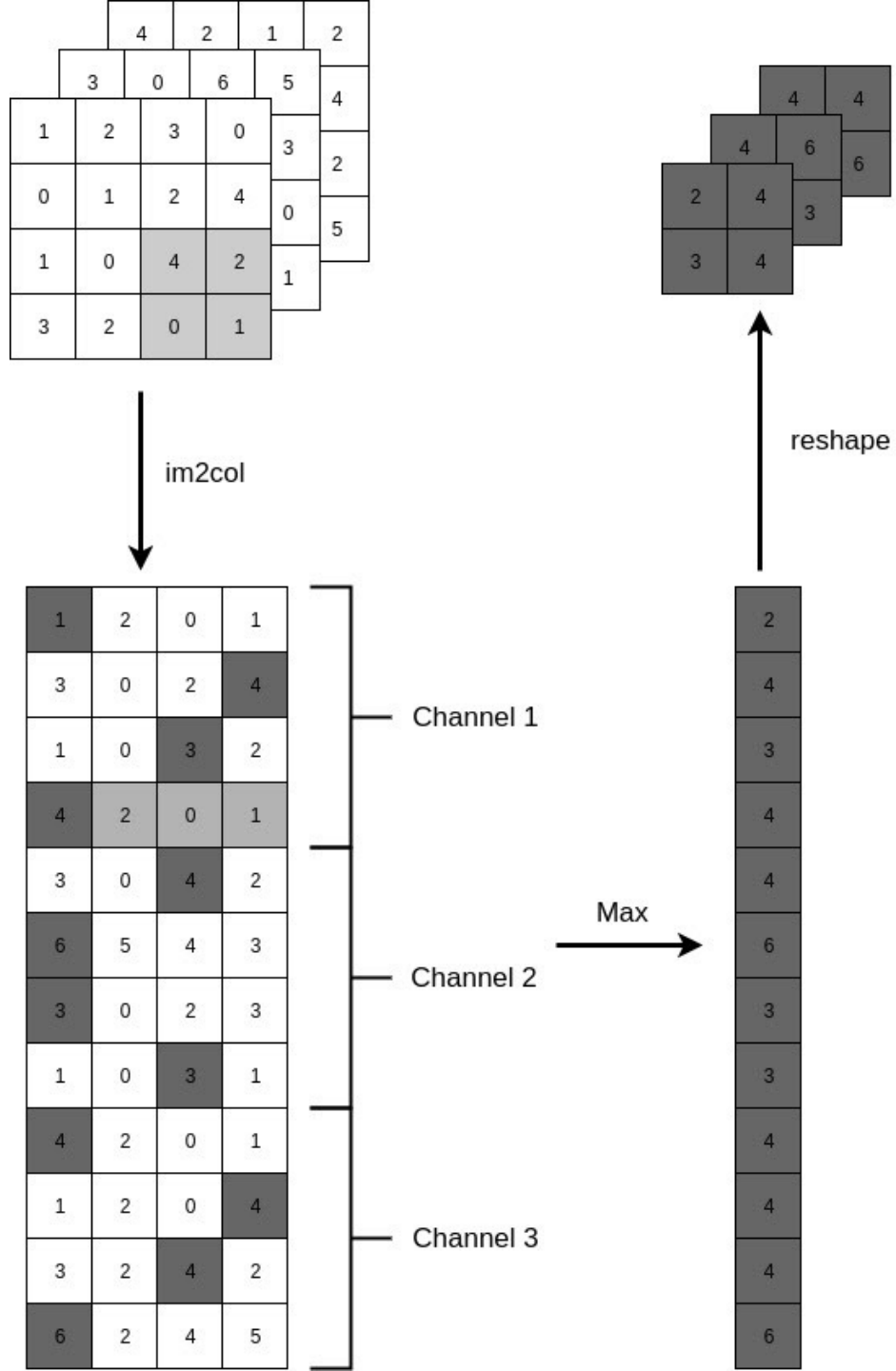


Figure 3. Unrolling the convolution operations in a convolutional layer (biases, sub-sampling, and non-linearity omitted), to produce a matrix-matrix product version.


4 2 1 2
3 0 6 5 4
1 2 3 0 3 2
0 1 2 4 0 5
1 0 4 2 1
3 2 0 1
im2col
1 2 0 1
3 0 2 4
1 0 3 2
4 2 0 1
3 0 4 2
6 5 4 3
3 0 2 3
1 0 3 1
4 2 0 1
1 2 0 4
3 2 4 2
6 2 4 5
Channel 1
Channel 2
Channel 3
Max
2
4
3
4
4
6
3
3
4
4
4
6
reshape
4 4
4 6 6
2 4 3
3 4

Unsupervised Feature Learning

- **Can apply any feature extraction method where labels are not used**
  - "Learn features" from patches