Convolutional Neural Net.

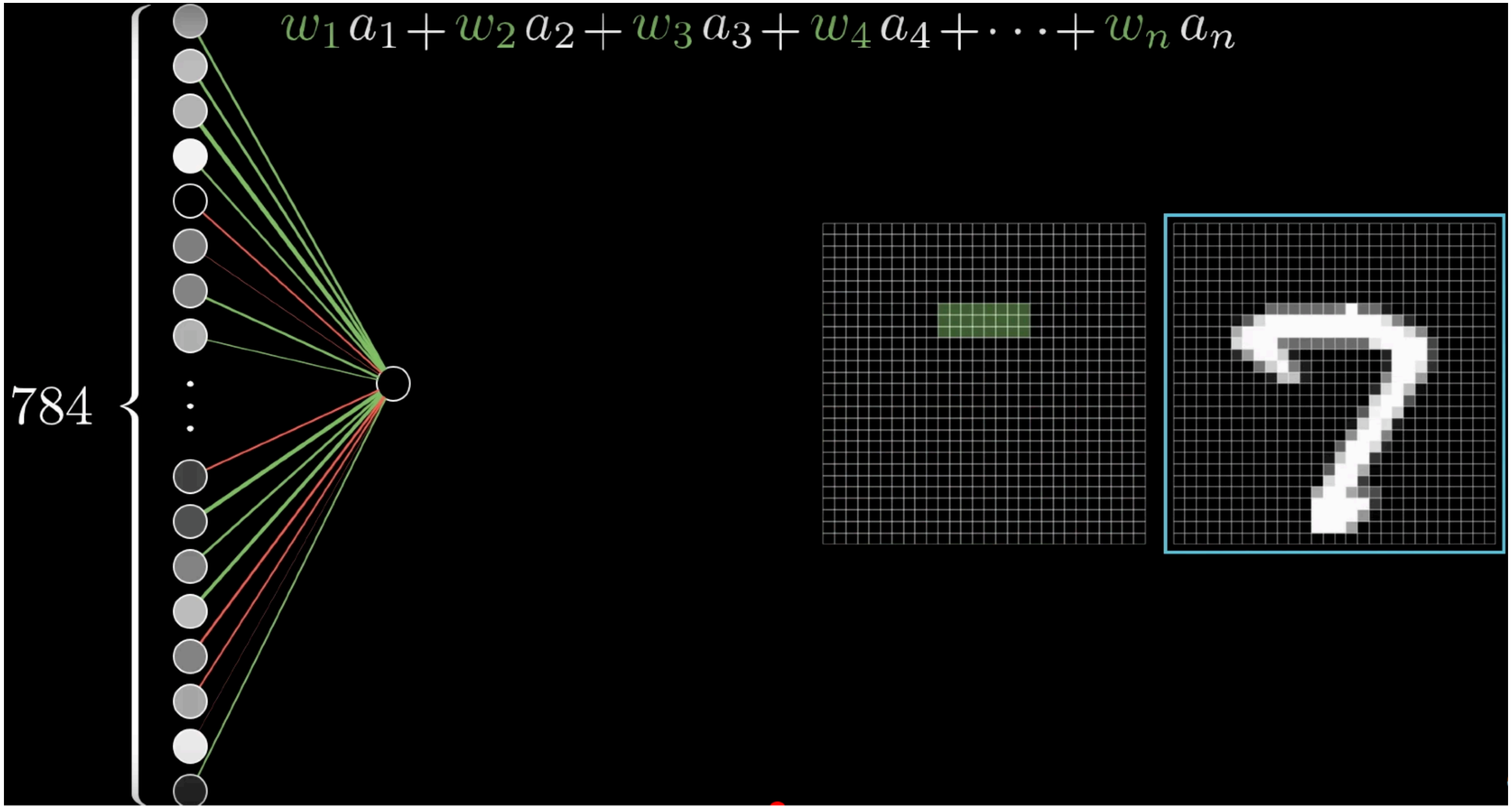
Seyoung Yun

- http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

- http://www.di.ens.fr/~lelarge/dldiy/slides/lecture_6/index.html#80

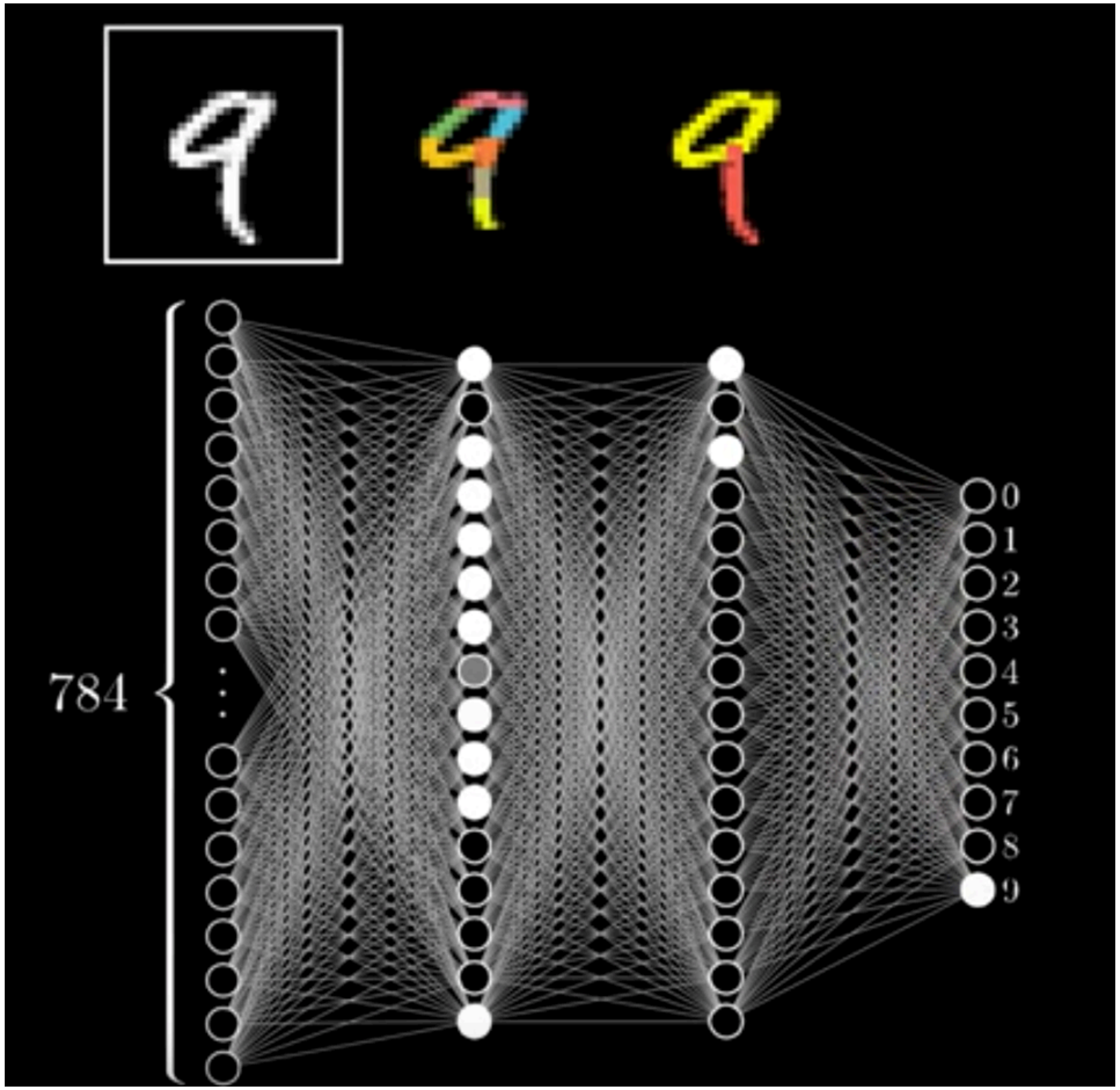
Why we need CNN?

- With a single neuron?



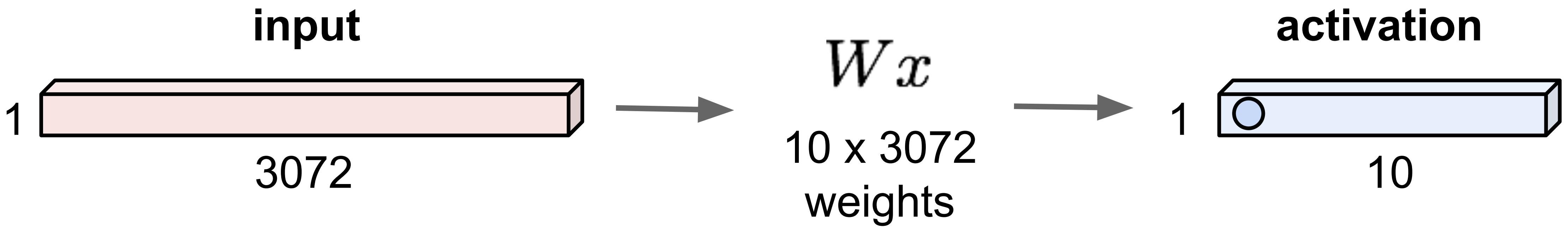
Why we need CNN?

- With FNN?



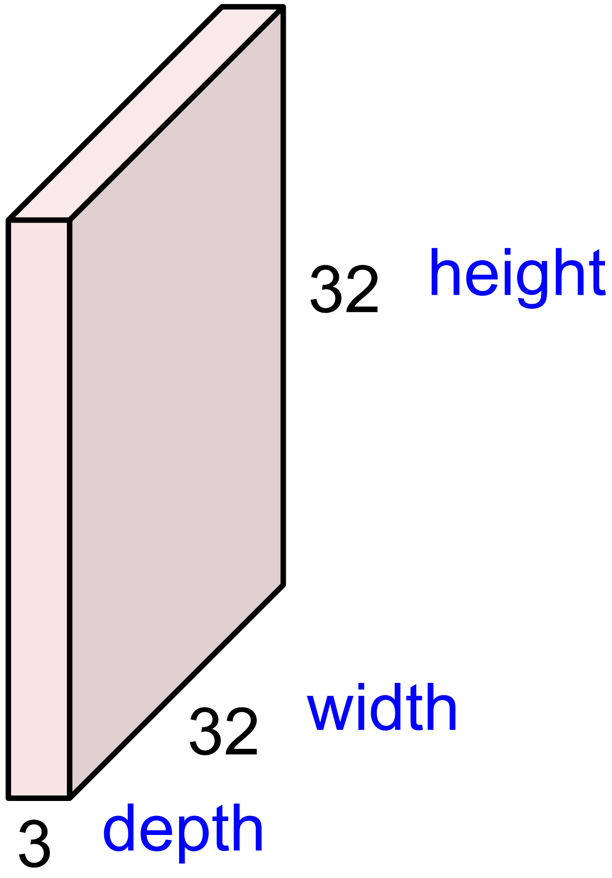
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

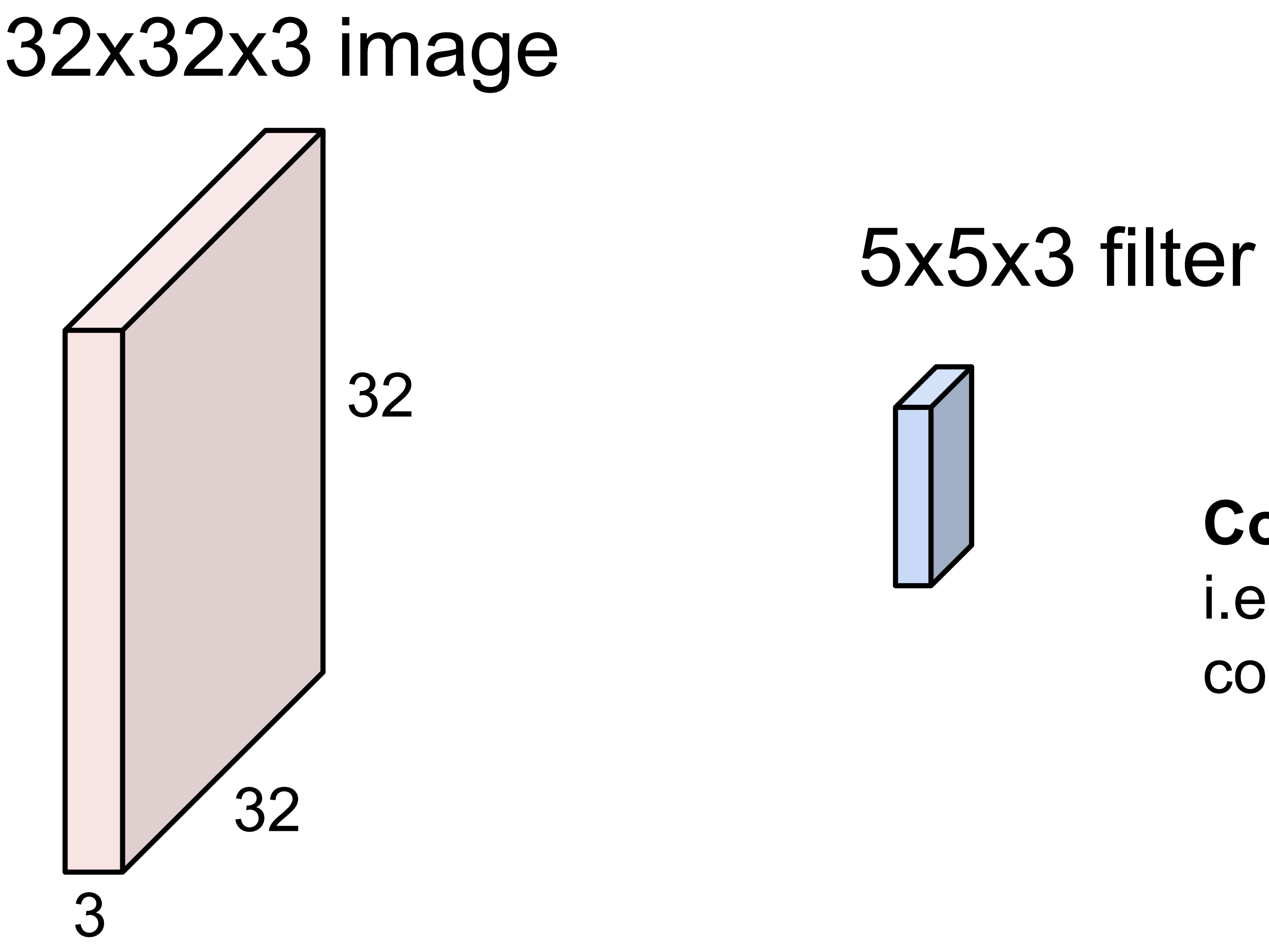


Convolution Layer

32x32x3 image -> preserve spatial structure



Convolution Layer

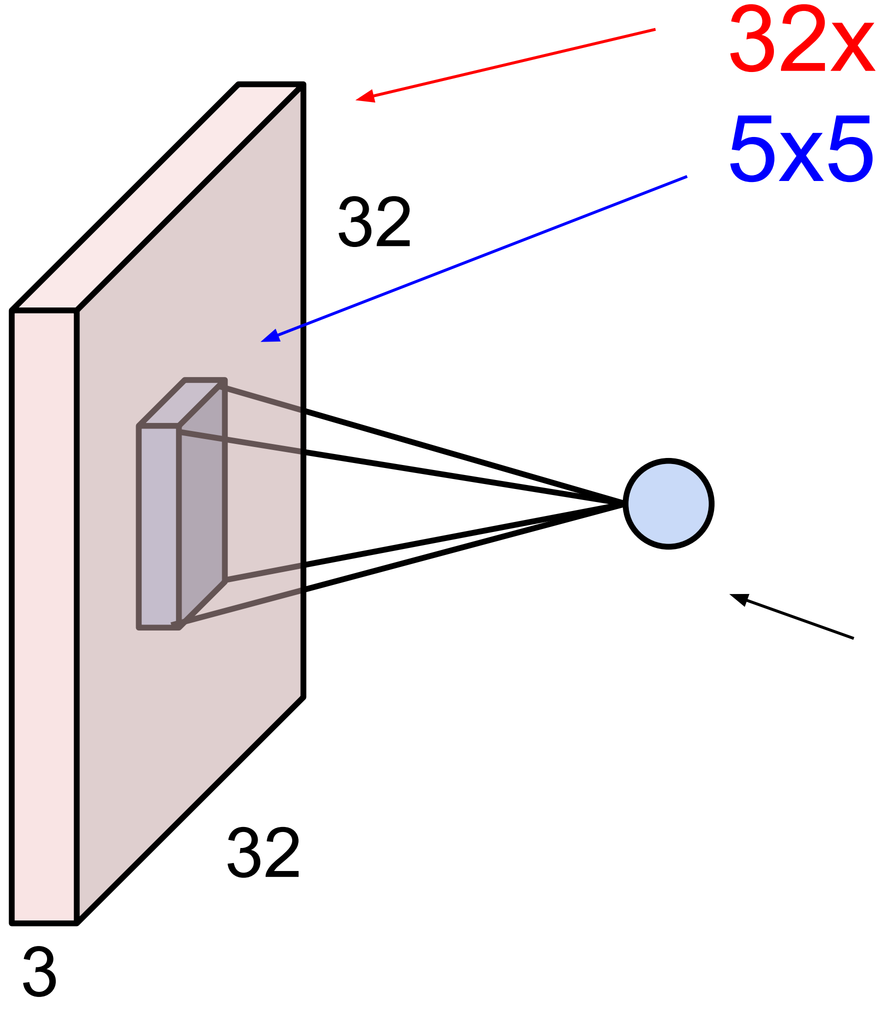


32x32x3 image

5x5x3 filter

Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolution Layer



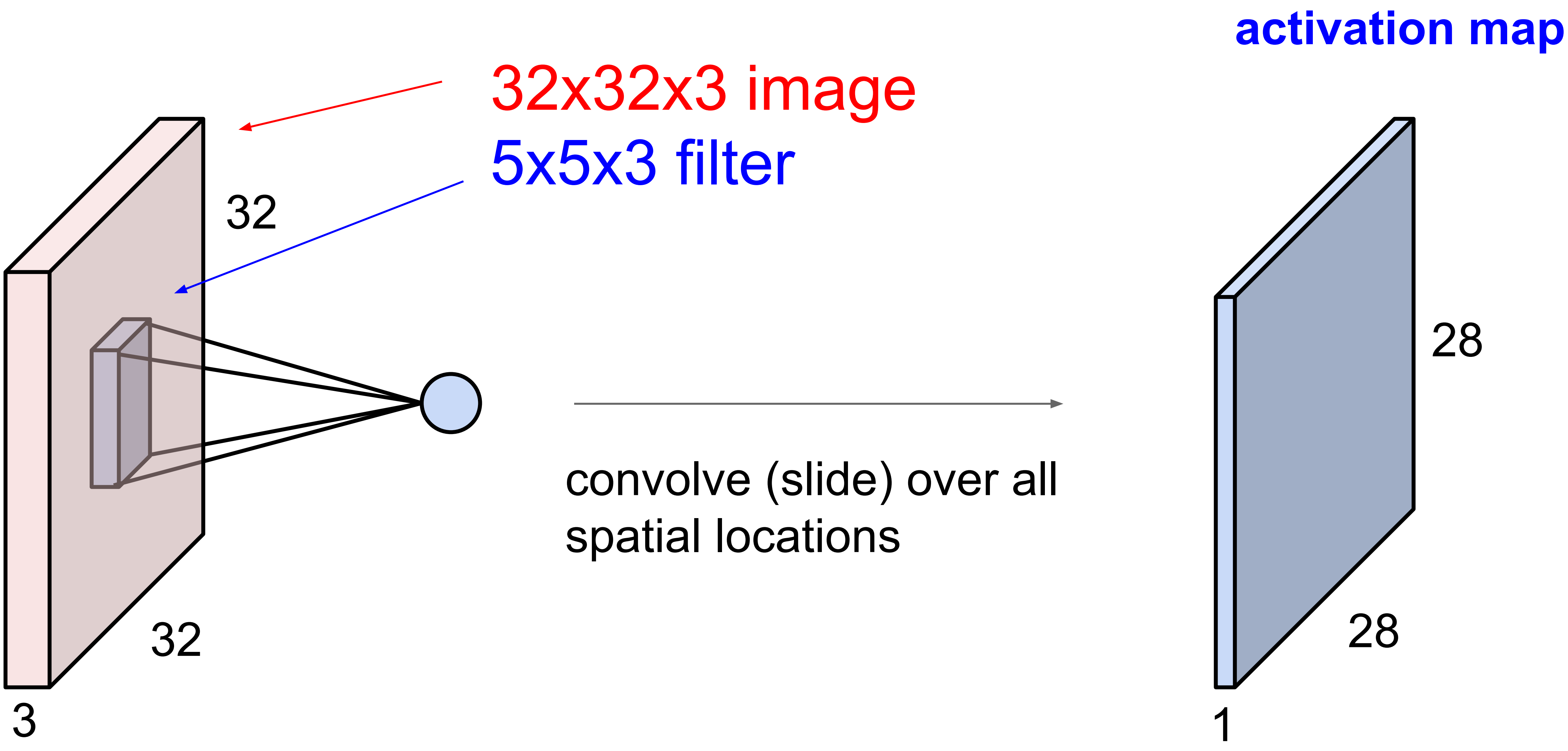
32x32x3 image

5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolution Layer



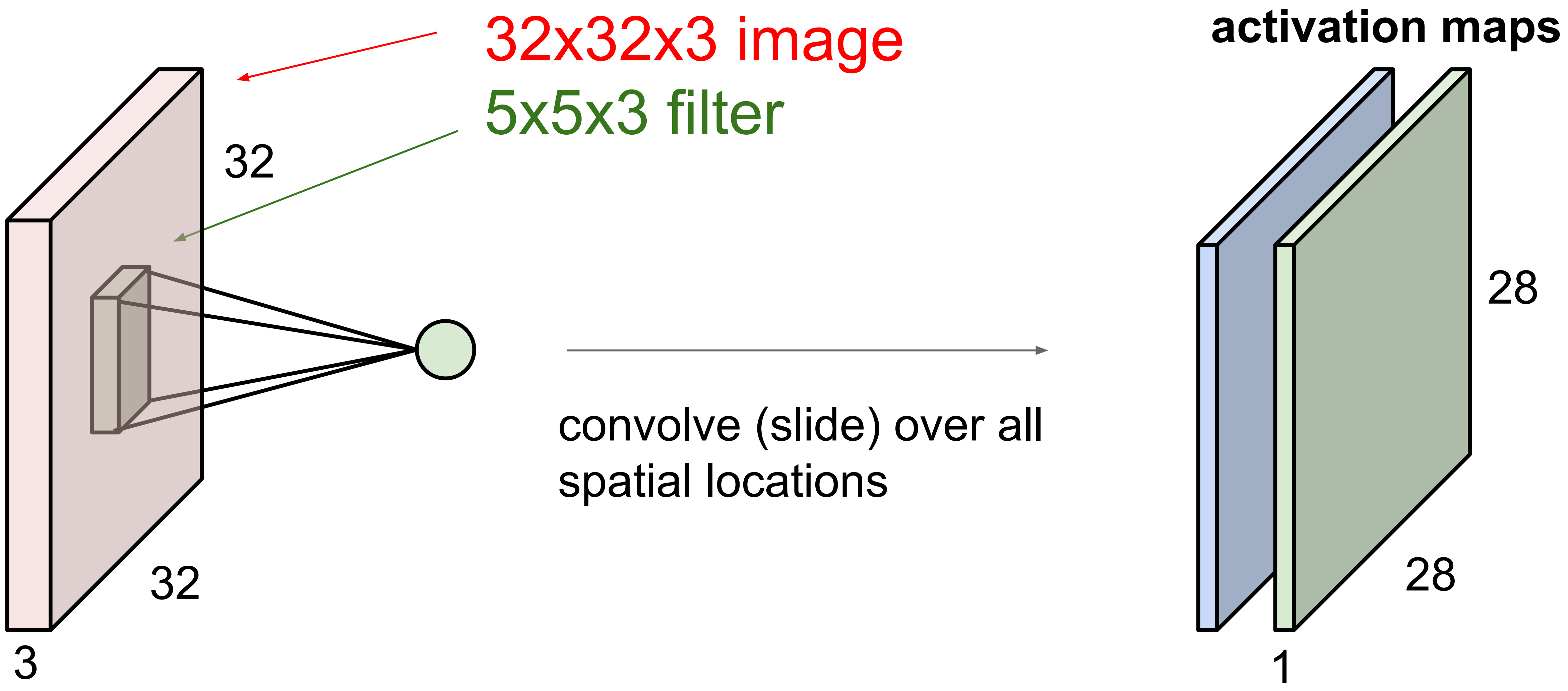
activation map

32x32x3 image

5x5x3 filter

convolve (slide) over all spatial locations

Convolution Layer



32x32x3 image

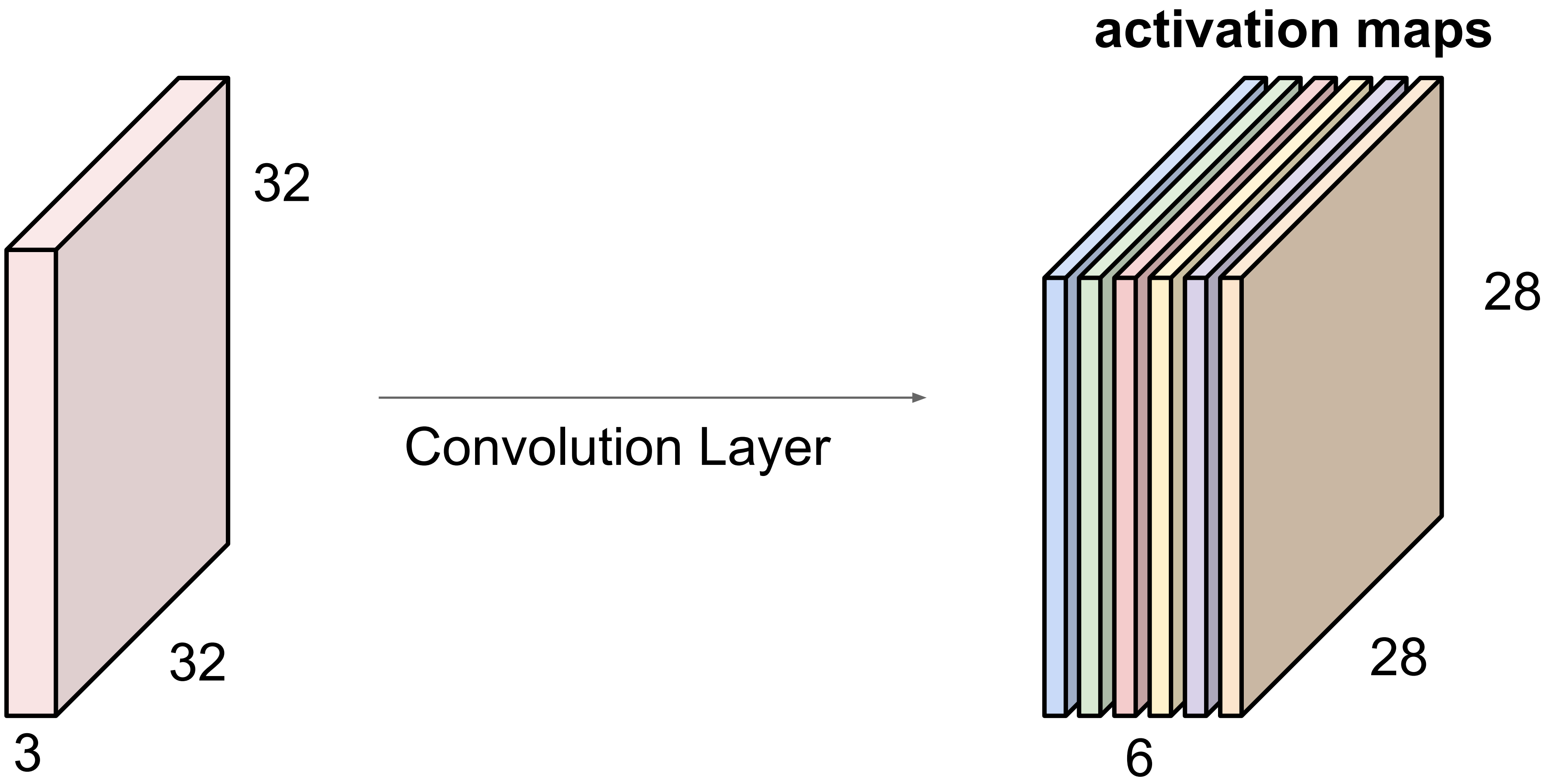
5x5x3 filter

convolve (slide) over all spatial locations

activation maps

Convolution Layer

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

What is Convolution?

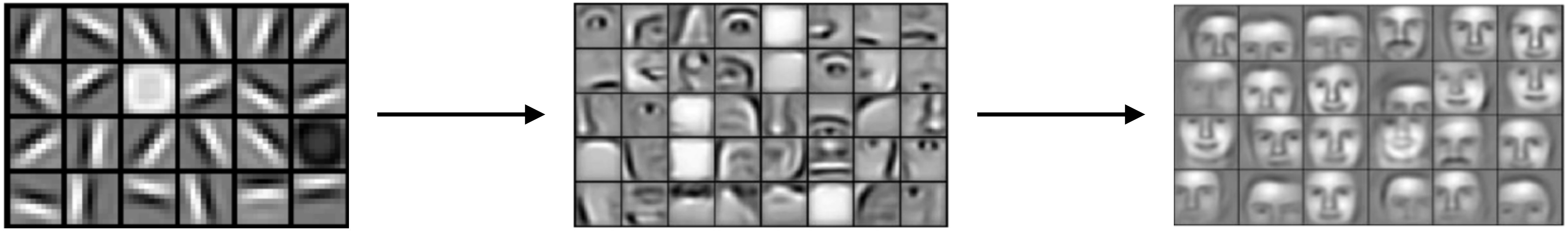
- Convolution

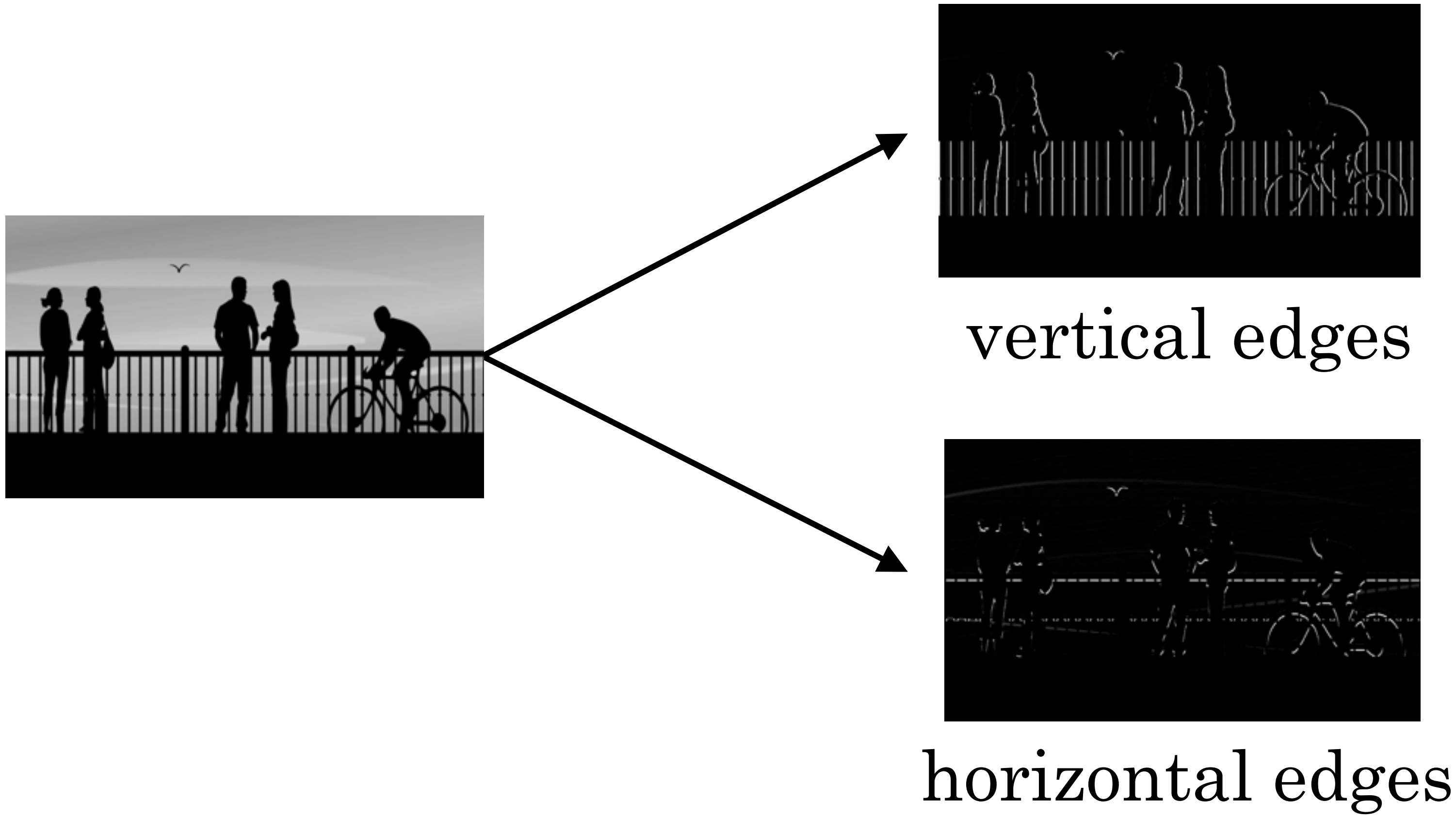
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n).$$

- Cross-correlation

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n).$$

Edge Detection



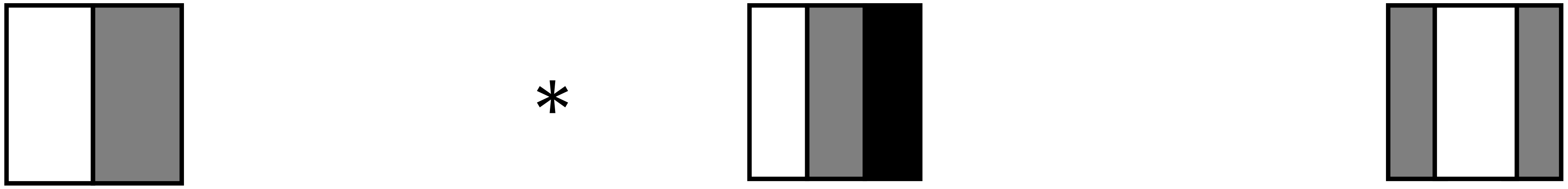


vertical edges

horizontal edges

Edge Detection

$$
\begin{array}{|c|c|c|c|c|c|}
\hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
10 & 10 & 10 & 0 & 0 & 0 \\ \hline
\end{array}
\quad * \quad
\begin{array}{|c|c|c|}
\hline
1 & 0 & -1 \\ \hline
1 & 0 & -1 \\ \hline
1 & 0 & -1 \\ \hline
\end{array}
\quad = \quad
\begin{array}{|c|c|c|c|}
\hline
0 & 30 & 30 & 0 \\ \hline
0 & 30 & 30 & 0 \\ \hline
0 & 30 & 30 & 0 \\ \hline
0 & 30 & 30 & 0 \\ \hline
\end{array}
$$



Edge Detection

1	0	-1
1	0	-1
1	0	-1

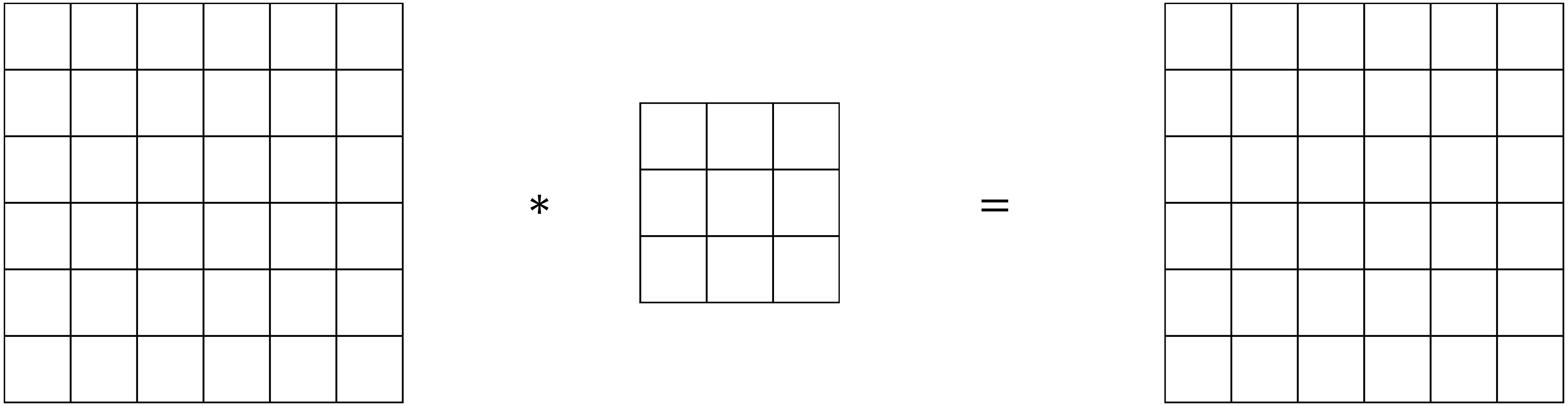
Vertical

1	1	1
0	0	0
-1	-1	-1

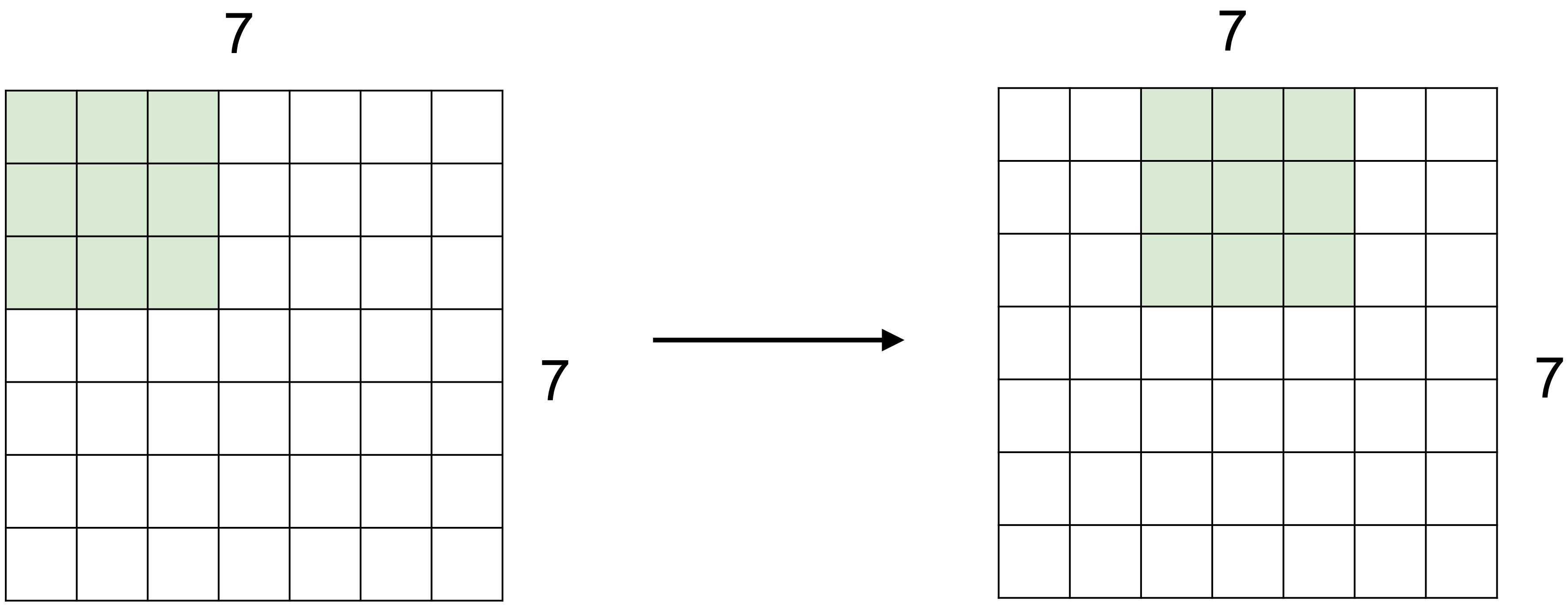
Horizontal

$$\begin{array}{|c|c|c|c|c|c|}\hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ \hline 30 & 10 & -10 & -30 \\ \hline 30 & 10 & -10 & -30 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array}$$

Padding



Stride

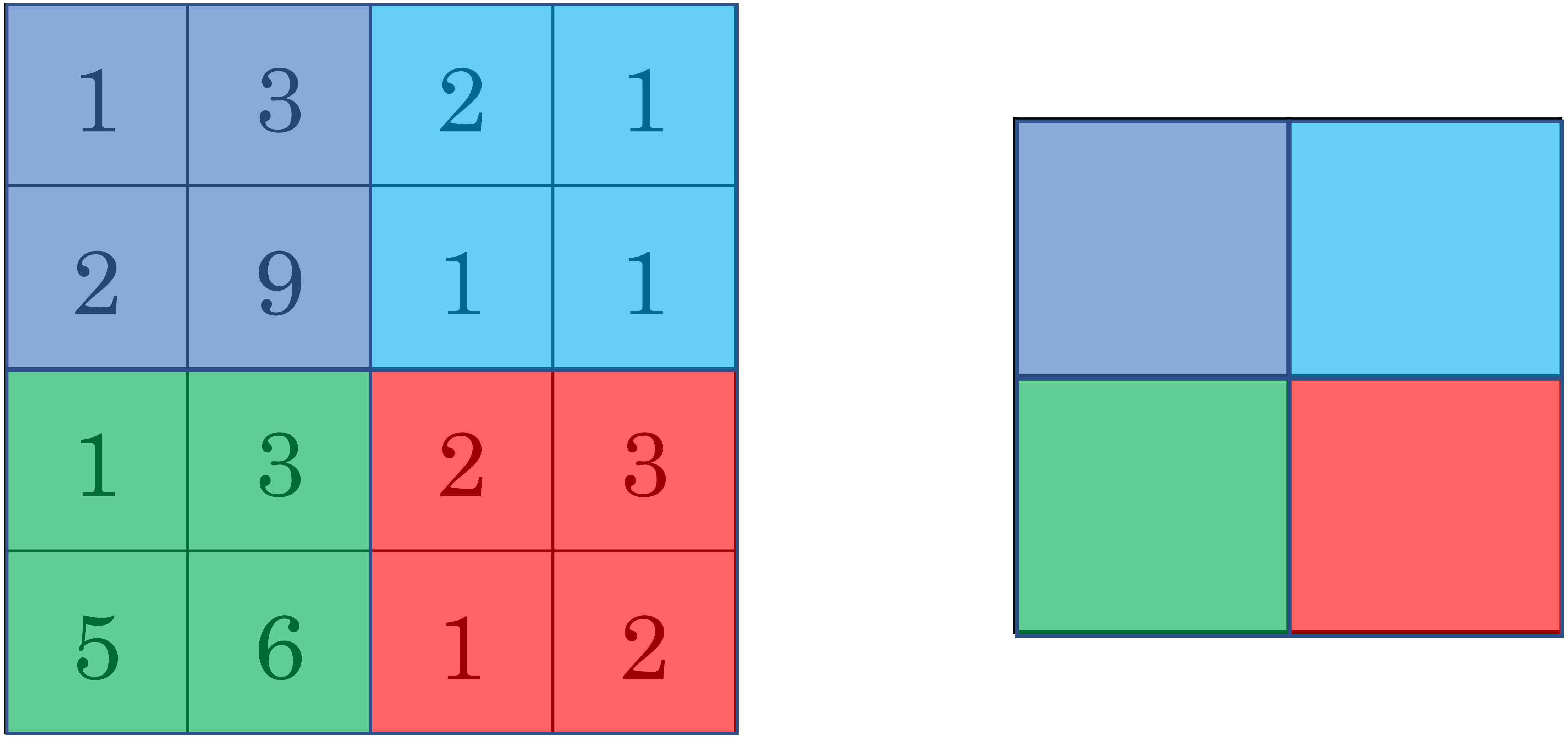


Example

Types of layer in a convolutional network

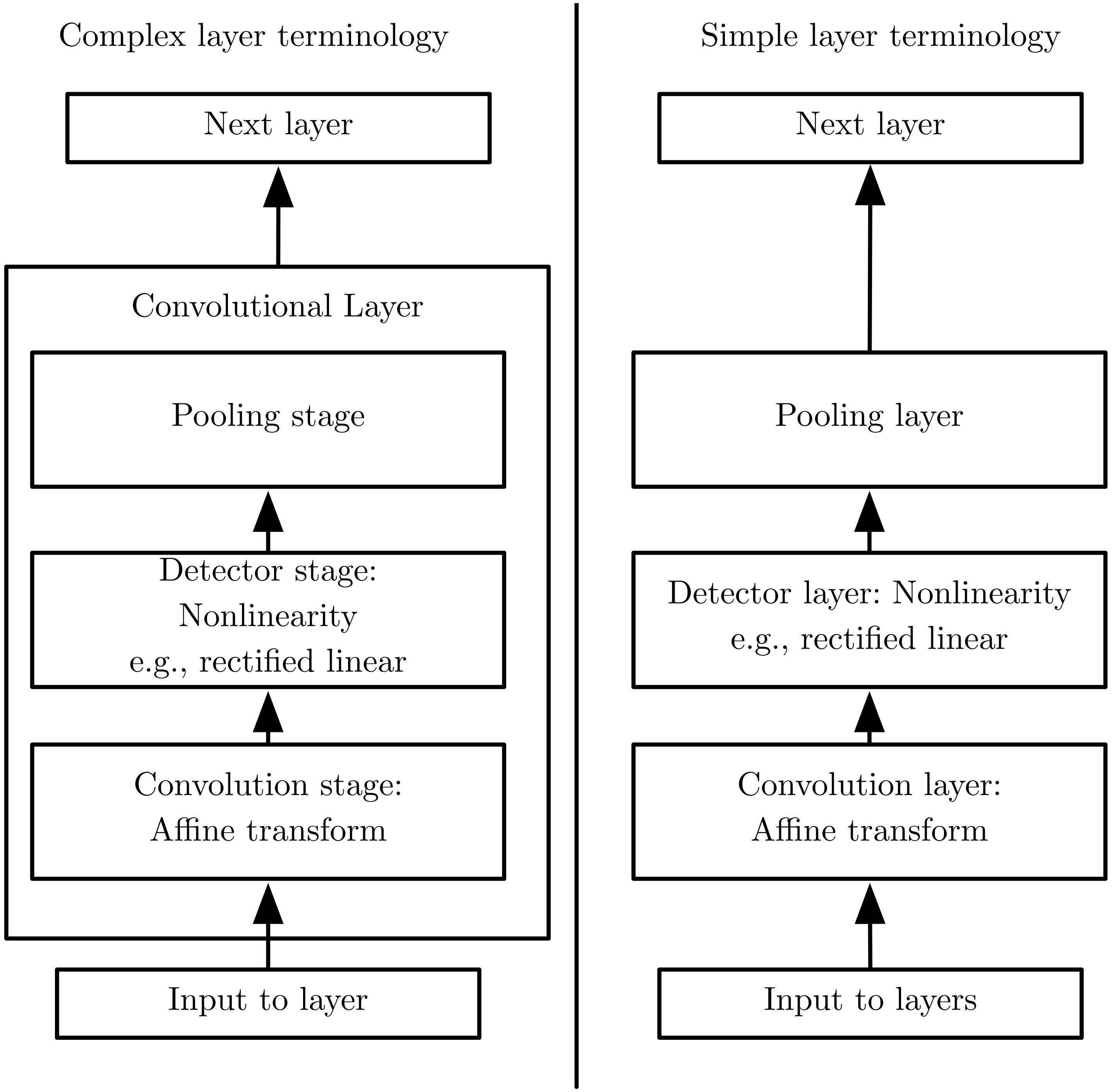
- Convolution
- Pooling
- Fully connected

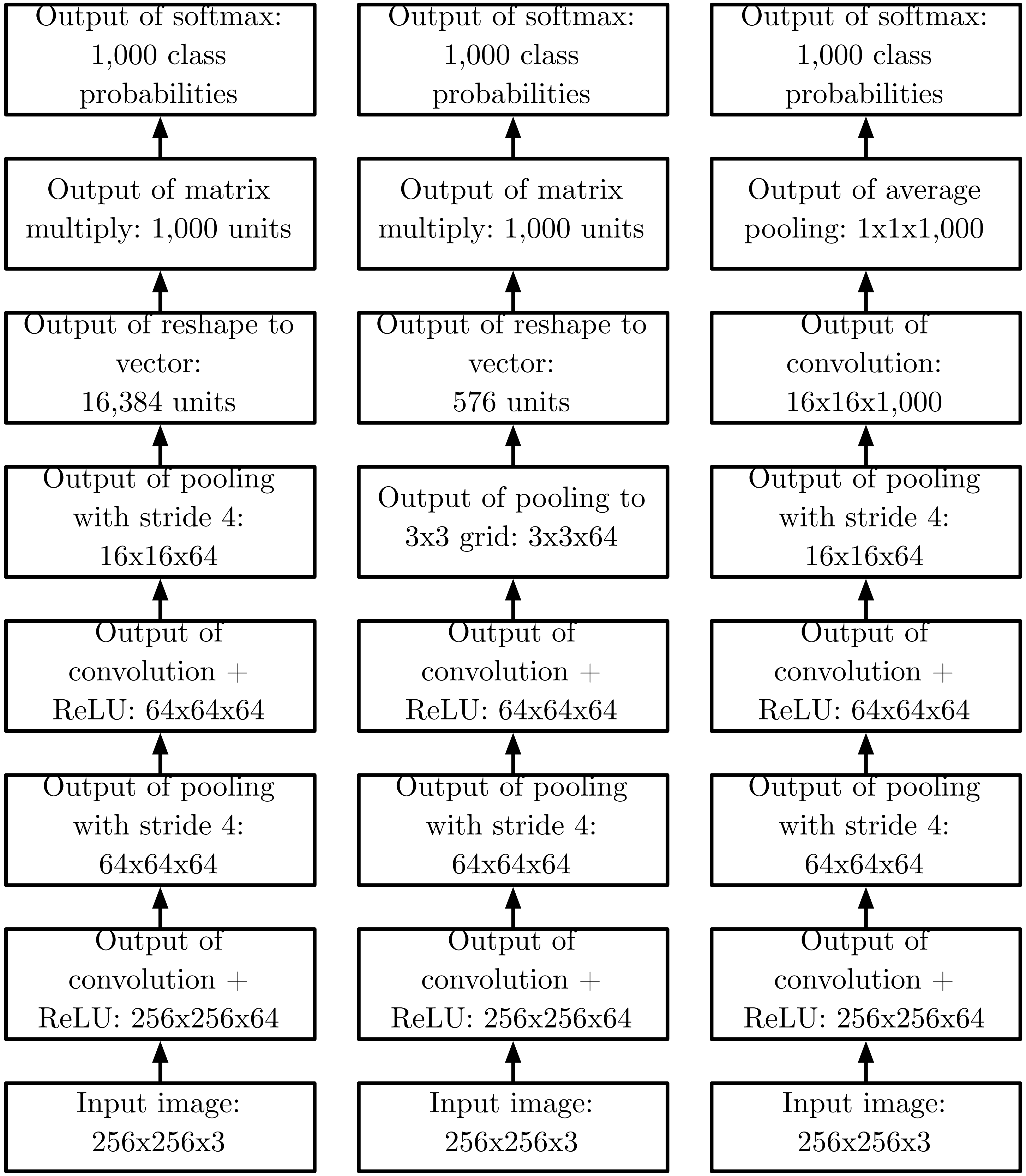
Max pooling



1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

CNN


Complex layer terminology
Next layer
Convolutional Layer
Pooling stage
Detector stage:
Nonlinearity
e.g., rectified linear
Convolution stage:
Affine transform
Input to layer
Simple layer terminology
Next layer
Pooling layer
Detector layer: Nonlinearity
e.g., rectified linear
Convolution layer:
Affine transform
Input to layers



Output of softmax: 1,000 class probabilities

Output of matrix multiply: 1,000 units

Output of reshape to vector: 16,384 units

Output of pooling with stride 4: 16x16x64

Output of convolution + ReLU: 64x64x64

Output of pooling with stride 4: 64x64x64

Output of convolution + ReLU: 256x256x64

Input image: 256x256x3

Output of softmax: 1,000 class probabilities

Output of matrix multiply: 1,000 units

Output of reshape to vector: 576 units

Output of pooling to 3x3 grid: 3x3x64

Output of convolution + ReLU: 64x64x64

Output of pooling with stride 4: 64x64x64

Output of convolution + ReLU: 256x256x64

Input image: 256x256x3

Output of softmax: 1,000 class probabilities

Output of average pooling: 1x1x1,000

Output of convolution: 16x16x1,000

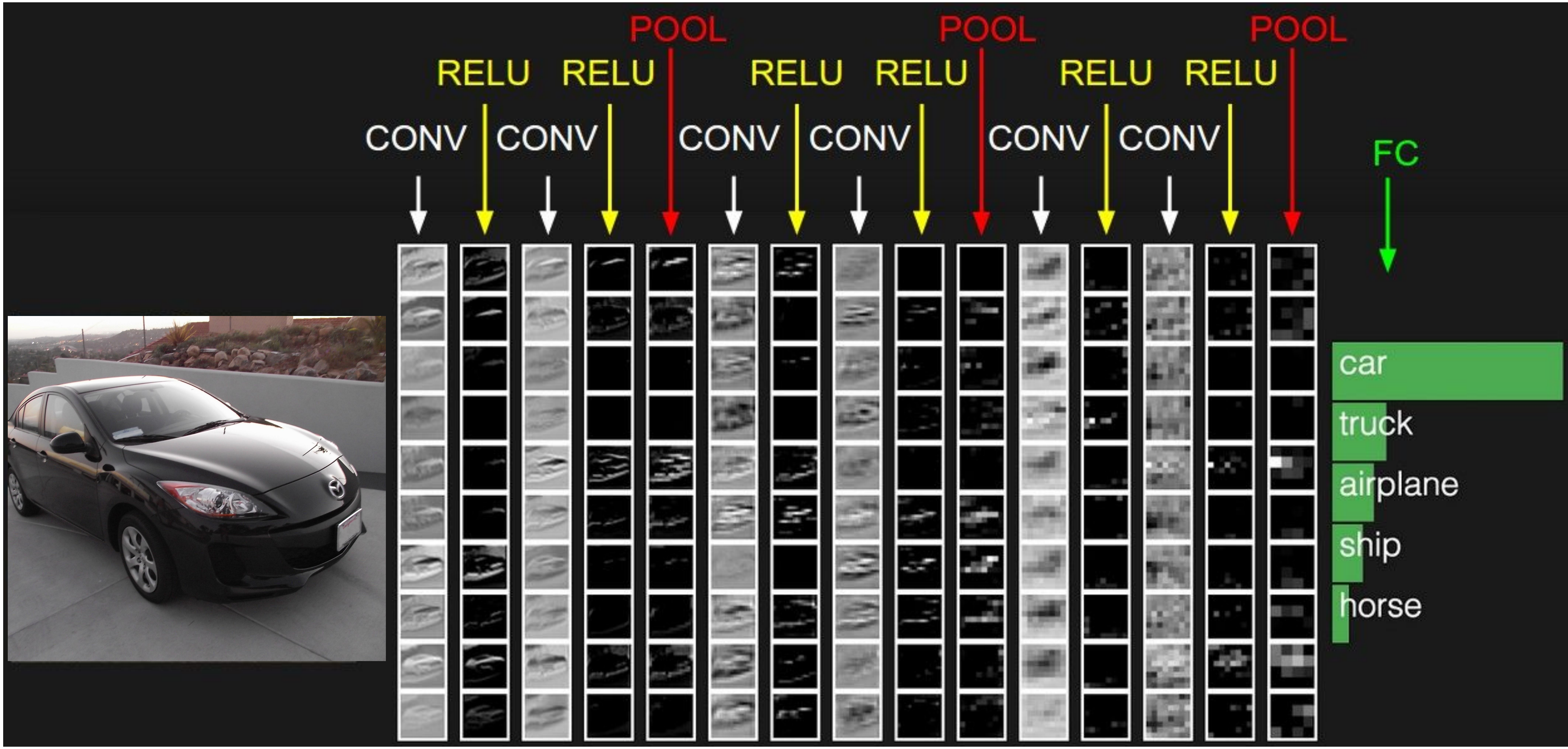
Output of pooling with stride 4: 16x16x64

Output of convolution + ReLU: 64x64x64

Output of pooling with stride 4: 64x64x64

Output of convolution + ReLU: 256x256x64

Input image: 256x256x3


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse