Convolutions and CNNs

Badri N Subudhi

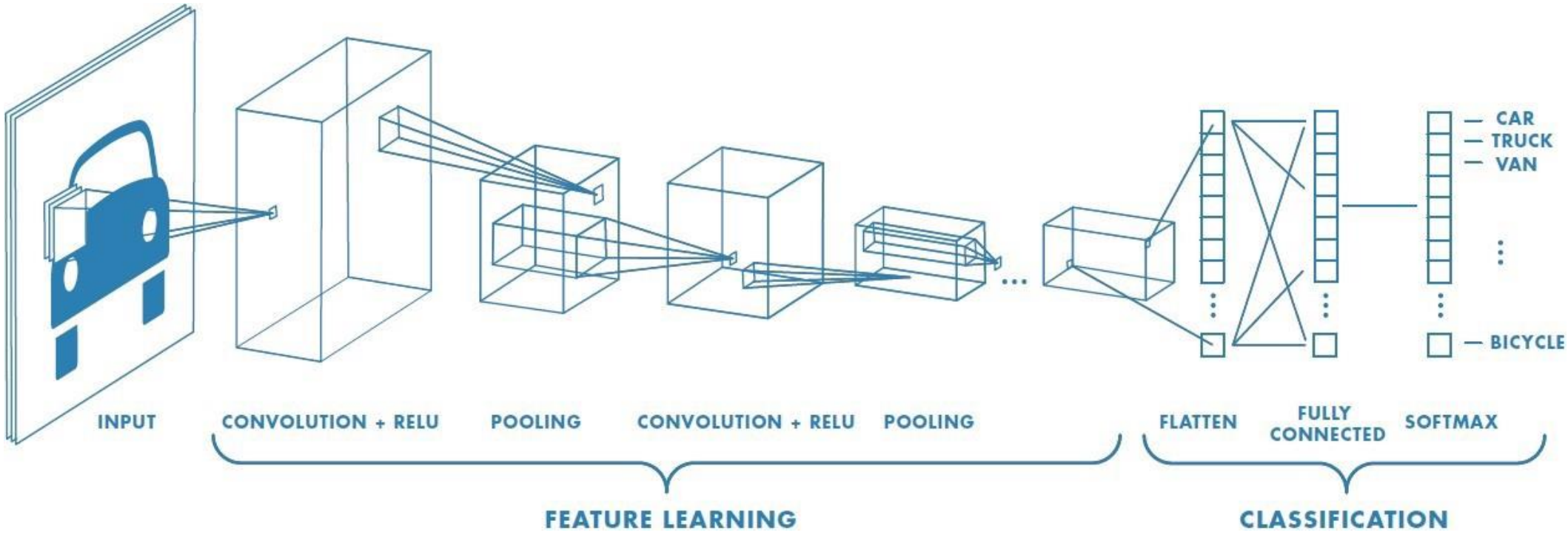
Associate Professor

IIT Jammu

CNNs ?

CNN : Neural Network with a convolution operation instead of matrix multiplication in at least one of the layers

A typical CNN architecture



Convolution and Correlation

Convolution and correlation operations are basically used to extract information from images.

Convolution and correlation are basically linear and shift-invariant operations

The term *linear* indicates that a pixel is replaced by the linear combination of its neighbors.

The term *shift invariant* indicates that the same operation is performed at every point in the image.

Convolution

Convolution is basically a mathematical operation where each value in the output is expressed as the sum of values in the input multiplied by a set of weighting coefficients.

Depending upon the weighting coefficients, convolution operation is used to perform spatial domain low-pass and high-pass filtering of the image.

An image can be either smoothened or sharpened by convolving the image with respect to low-pass and high-pass filter mask respectively.

Convolution has a multitude of applications including image filtering, image enhancement, image restoration, feature extraction and template matching.

$$y[n_1, n_2] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x(k_1, k_2) h(n_1 - k_1, n_2 - k_2)$$

The input matrix x(m, n) and h(m, n). Perform the linear convolution between these two matrices.

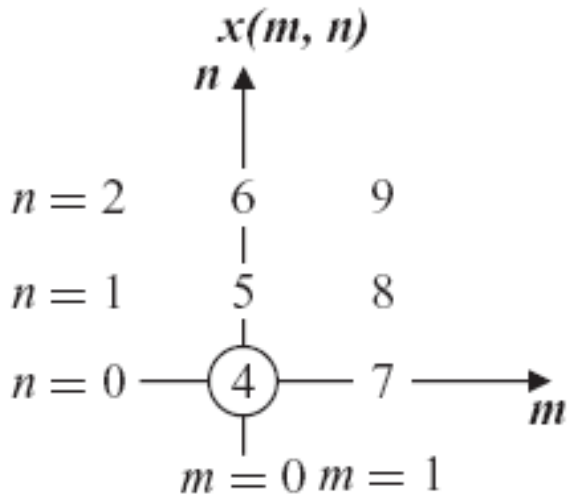
$$x(m,n)=\begin{pmatrix}4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix} \qquad h(m,n)=\begin{pmatrix}1\\ 1\\ 1\end{pmatrix}$$

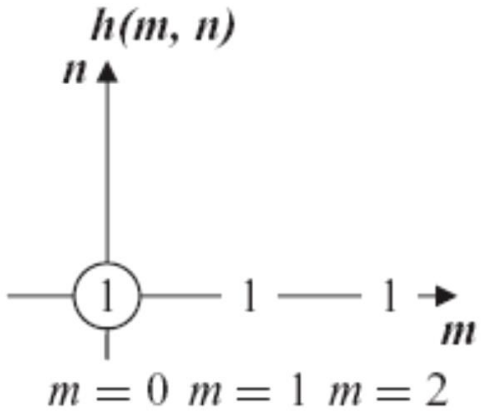
$$x(m,n)=\begin{pmatrix}\underset{4}{(0,0)} & \underset{5}{(0,1)} & \underset{6}{(0,2)}\\ \underset{7}{(1,0)} & \underset{8}{(1,1)} & \underset{9}{(1,2)}\end{pmatrix} \qquad h(m,n)=\begin{pmatrix}\underset{1}{(0,0)}\\ \underset{1}{(1,0)}\\ \underset{1}{(2,0)}\end{pmatrix}$$

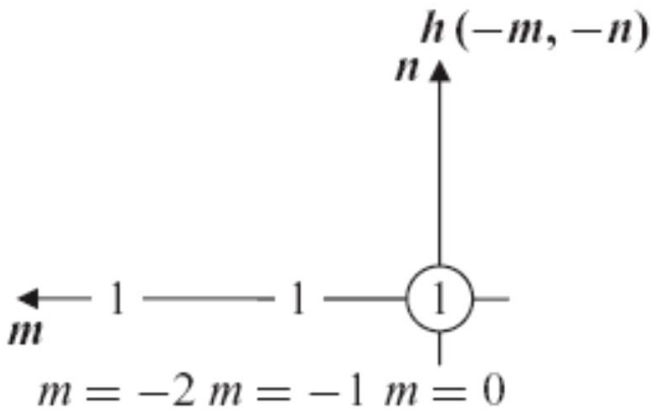
$$\text{Dimension of reslutant matrix}=\begin{cases}(\text{No. of rows of } x(m,n)+\text{No. of rows of } h(m,n)-1)\\ \times\\ (\text{No. of columns of } x(m,n)+\text{No. of columns of } h(m,n)-1)\end{cases}$$

Dimension of resultant matrix = (2 + 3 − 1) × (3 + 1 − 1) = 4 × 3

$$y(m,n) = \begin{pmatrix} y(0,0) & y(0,1) & y(0,2) \\ y(1,0) & y(1,1) & y(1,2) \\ y(2,0) & y(2,1) & y(2,2) \\ y(3,0) & y(3,1) & y(3,2) \end{pmatrix}$$

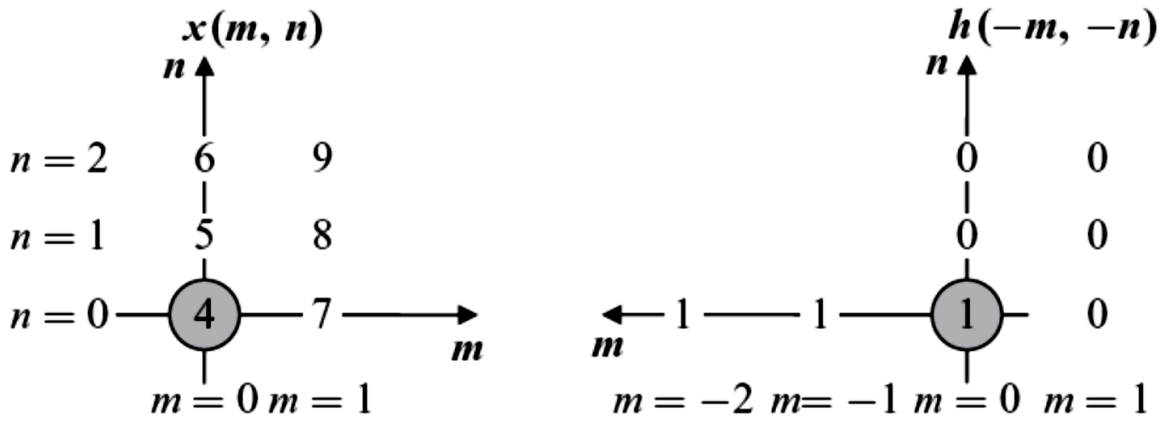






***To determine the value of y*(0, 0)**

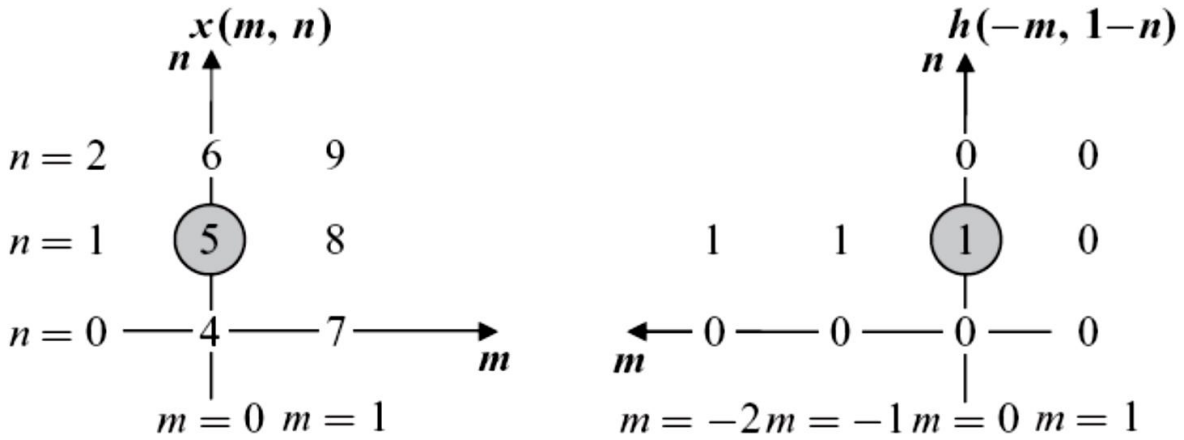
The common values of $x(m, n)$ and $h(-m, -n)$ are multiplied and then added to get the value of y (0, 0). The shaded circle indicates the common area between the two signals.



The value $y(0, 0)$ is obtained as $y(0, 0) = 4 \times 1 = 4$.

***To determine the value of y*(0,1)**

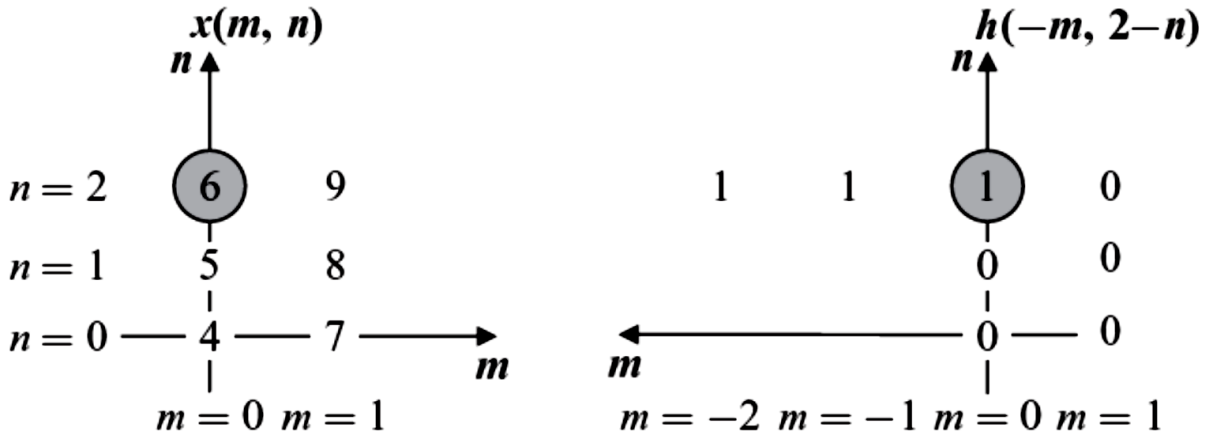
Now the signal $h(-m, -n)$ is shifted along the 'n' axis by one unit to get $h(-m, 1-n)$ and $x(m, n)$ is unaltered. The common value between the two signals is multiplied and then added to get y (0, 1).



The value of $y(0, 1)$ is given as $y(0, 1) = 5 \times 1 = 5$.

***To determine the value of y*(0, 2)**

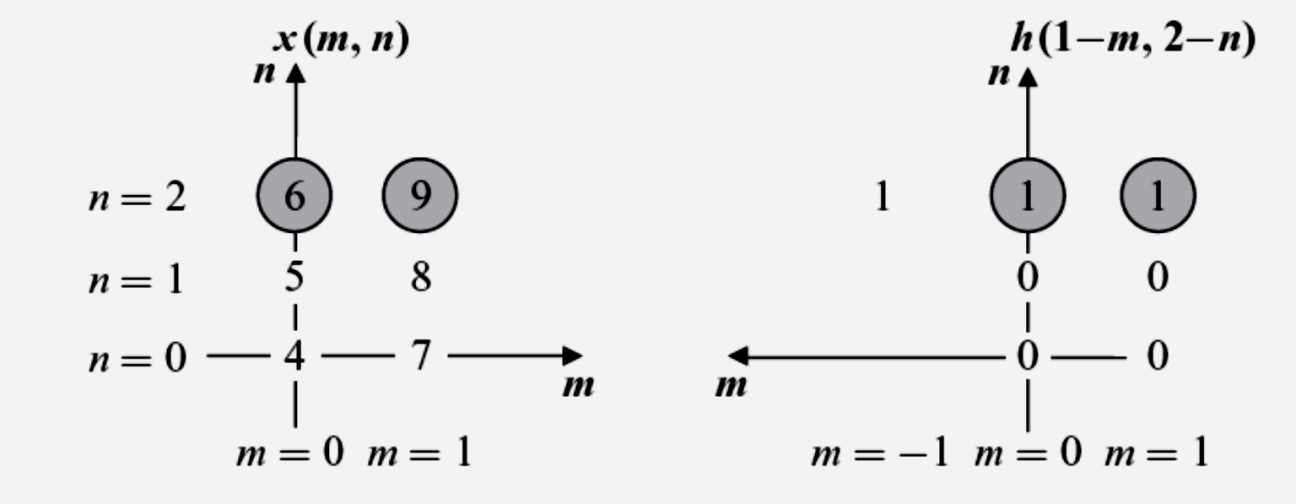
The value of $h(-m, -n)$ is shifted by two units along the 'n' axis to get $h(-m, 2-n)$. Then, the common values between $x(m, n)$ and $h(-m, 2-n)$ are multiplied and then added to get $y(0, 2)$.



The resultant value of $y(0, 2)$ is $y(0, 2) = 6 \times 1 = 6$.

***To determine the value of y*(1, 2)**

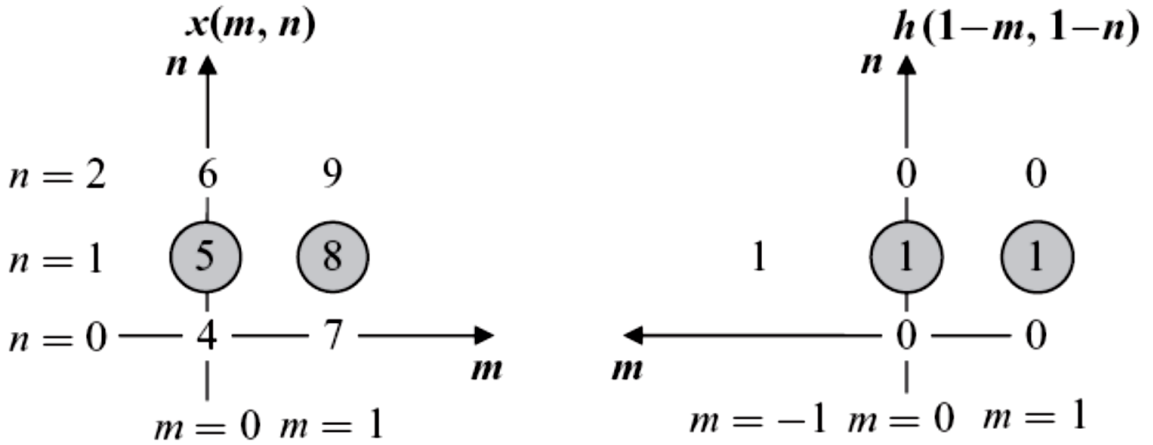
Here, $h(-m, 2-n)$ is shifted along the 'm' axis to one unit towards right to get $h(1-m, 2-n)$. Then the common values between $x(m, n)$ and $h(1-m, 2-n)$ are multiplied and added to get $y(1, 2)$.



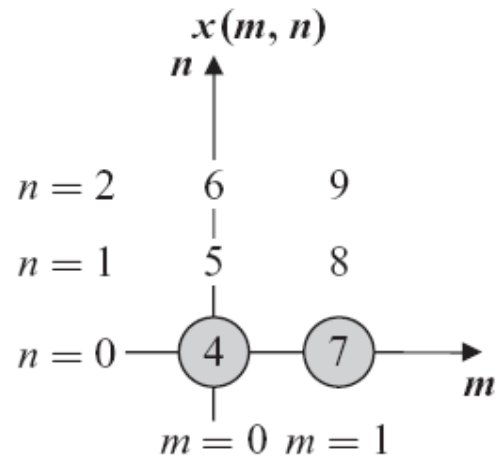
The final value of $y(1, 2) = 6 \times 1 + 9 \times 1 = 15$.

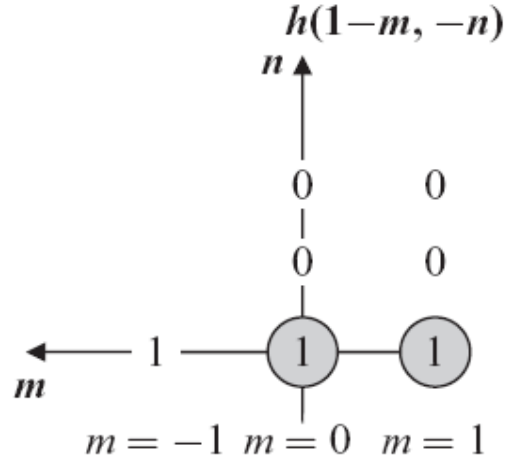
***To determine the value of y*(1, 1)**

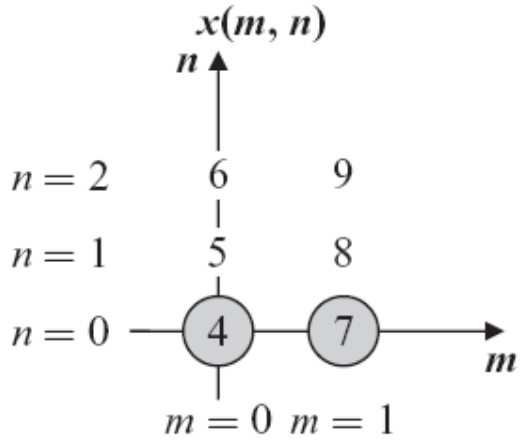
Now the value of $h(1-m, 2-n)$ is shifted down along the 'n' axis to get $h(1-m, 1-n)$. The common values between $x(m, n)$ and $h(1-m, 1-n)$ are multiplied and added to get $y(1, 1)$.

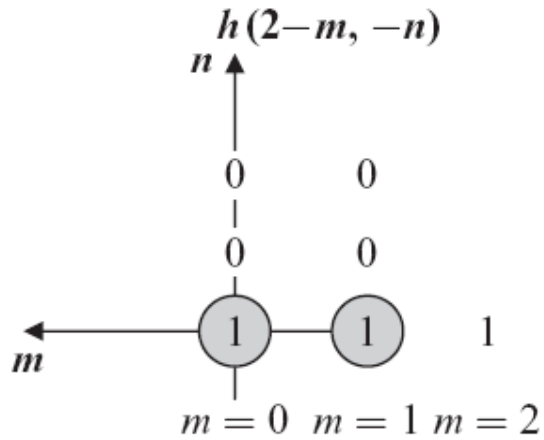


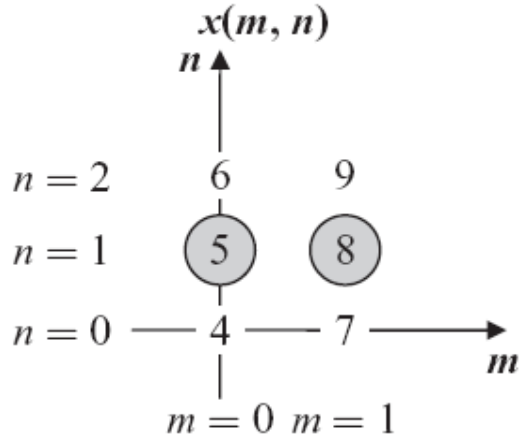
The value of $y(1, 1)$ is obtained as $y(1, 1) = 5 \times 1 + 8 \times 1 = 13$.

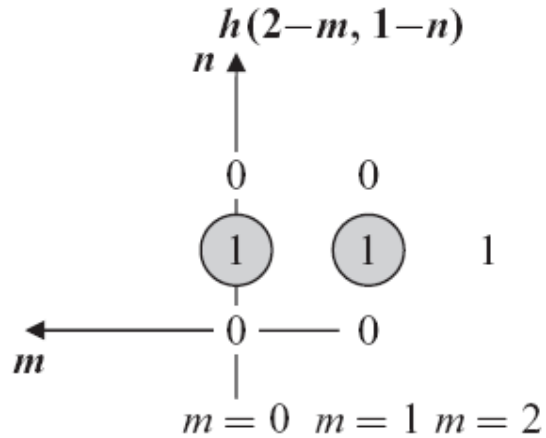

x(m, n)
n
n = 2
n = 1
n = 0
6
9
5
8
4
7
m
m = 0
m = 1

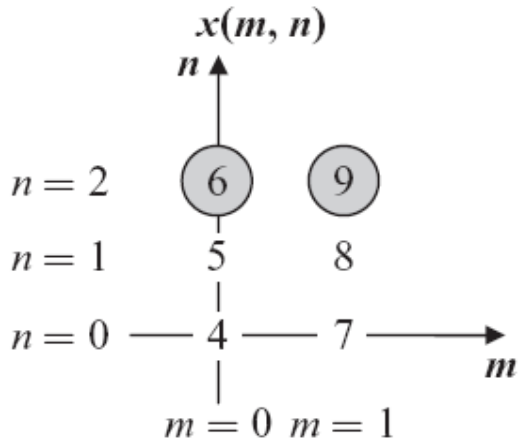

h(1−m, −n)
n
0
0
0
0
1
1
1
m
m = −1
m = 0
m = 1

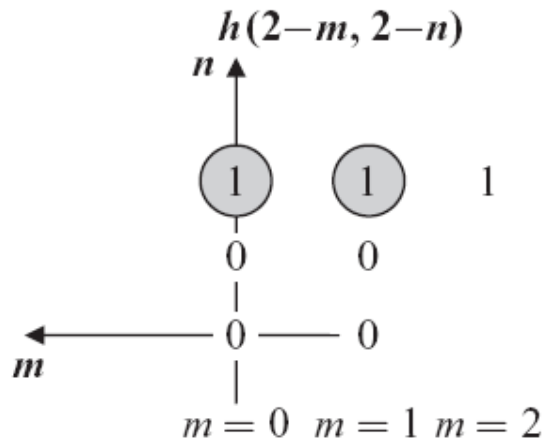

x(m, n)
n
n = 2
n = 1
n = 0
6
9
5
8
4
7
m
m = 0
m = 1

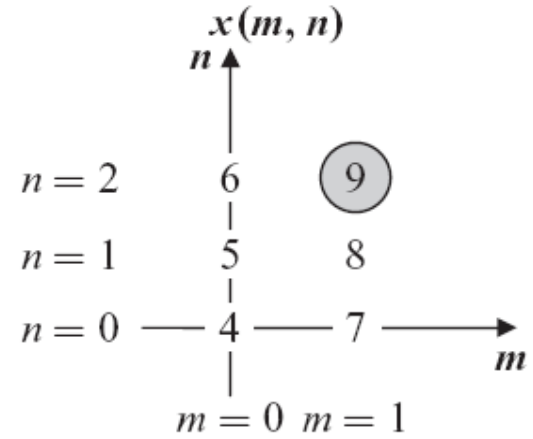

h (2−m, −n)
n
0
0
0
0
1
1
1
m
m = 0
m = 1
m = 2

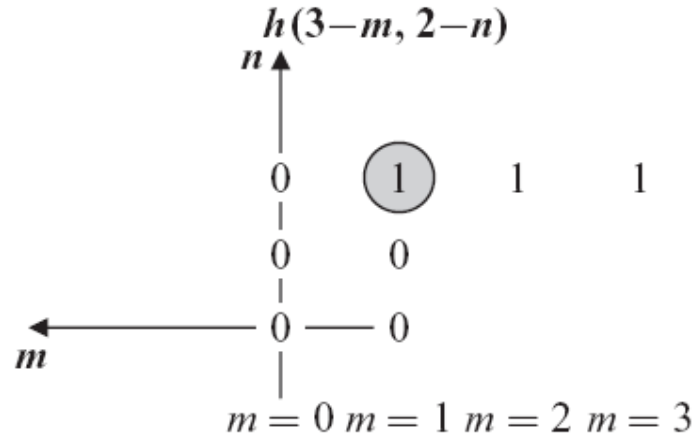

x(m, n)
n
n = 2
n = 1
n = 0
6
9
5
8
4
7
m
m = 0 m = 1

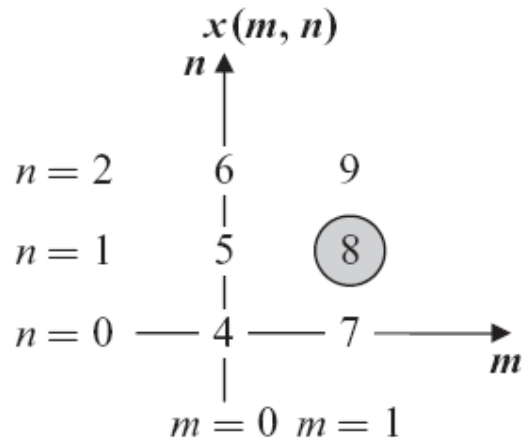

h(2−m, 1−n)
n
0
0
1
1
1
0
0
m
m = 0 m = 1 m = 2

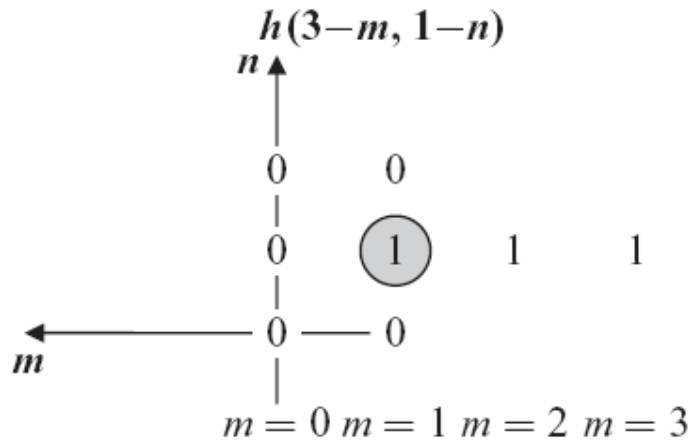

x(m, n)
n
n = 2
n = 1
n = 0
6
9
5
8
4
7
m
m = 0 m = 1

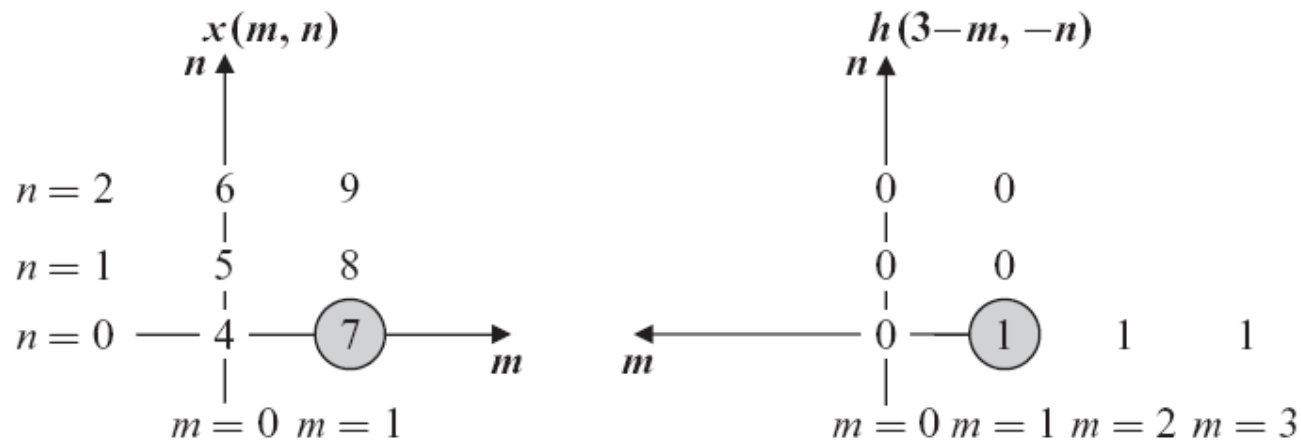

h(2−m, 2−n)
n
1
1
1
0
0
0
0
m
m = 0 m = 1 m = 2


x(m, n)
n
n = 2
n = 1
n = 0
6
9
5
8
4
7
m
m = 0 m = 1


h(3−m, 2−n)
n
0
1
1
1
0
0
0
0
m
m = 0 m = 1 m = 2 m = 3


x(m, n)
n
n = 2
n = 1
n = 0
6
9
5
8
4
7
m
m = 0 m = 1


h(3−m, 1−n)
n
0
0
0
1
1
1
0
0
m
m = 0 m = 1 m = 2 m = 3



$$y(m,n) = \begin{pmatrix} 4 & 5 & 6 \\ 11 & 13 & 15 \\ 11 & 13 & 15 \\ 7 & 8 & 9 \end{pmatrix}$$

Matrix representation

Perform the linear convolution between the two matrices x(m, n) and h(m, n) given below

$$x(m,n)=\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix} \quad h(m,n)=\begin{pmatrix}1 & 1\\ 1 & 1\\ 1 & 1\end{pmatrix}$$

$$y(m,n)=\begin{pmatrix}1 & 3 & 5 & 3\\ 5 & 12 & 16 & 9\\ 12 & 27 & 33 & 18\\ 1 & 24 & 28 & 15\\ 7 & 15 & 17 & 9\end{pmatrix}$$

Matrix representation

$$x(m,n)=\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix} \qquad h(m,n)=\begin{pmatrix}3 & 4 & 5\end{pmatrix}$$

$$y(m,n)=\begin{pmatrix}3 & 10 & 22 & 22 & 15\\ 12 & 31 & 58 & 49 & 30\\ 21 & 52 & 94 & 76 & 45\end{pmatrix}$$

Matrix representation

Correlation

Correlation is a mathematical operation that is similar to convolution. Correlation is basically used to obtain similarity between two signals.

Autocorrelation is basically a signal correlated with itself.

The amplitude of each sample in the cross-correlation is a measure of how much one signal resembles the other.

$$x[n] * x^*[-n] = \sum_{k=-\infty}^{\infty} x[k]x^*[-(n-k)]$$

$$x[m,n] * x^*[-m,-n] = \sum_a \sum_b x(a,b)x^*[-(m-a),-(n-b)]$$

$$x[m,n] * x^*[-m,-n] = \sum_a \sum_b x(a,b)x^*[(a-m),(b-n)] = r_{xx}(m,n)$$

Determine the correlation between two matrices

$$x_1[m, n] = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix} \qquad x_2[m, n] = \begin{bmatrix} 1 & 5 \\ 2 & 3 \end{bmatrix}.$$

Step 1
From the given $x_2[m, n]$ determine $x_2[-m, -n]$.

Step 1a To get the folded version of $x_2[m, n]$, first fold $x_2[m, n]$ column wise to get

$$x_2[m, -n] = \begin{bmatrix} 5 & 1 \\ 3 & 2 \end{bmatrix}$$

Step 1b Then fold $x_2[m, -n]$ along row-wise to get $x_2[-m, -n]$ which is given by

$$x_2[-m,-n]=\begin{bmatrix}3 & 2\\ 5 & 1\end{bmatrix}$$

Step 2

Now we have to perform linear convolution between $x_1[m, n]$ and $x_2[-m, -n]$.

Step 2a Formation of block matrix The number of block matrices depends on the number of columns of $x_1[m, n]$. In this case, the number of columns of $x_2[m, n]$ is two. Hence, two block matrices have to be formed. Let the two block matrices be H_0 and H_1 respectively.

Step 2b Formation of block matrix H_0 The block matrix H_0 formed from the first row of $x_1[m, n]$ is given below.

$$H_0 = \begin{bmatrix} 3 & 0 \\ 1 & 3 \\ 0 & 1 \end{bmatrix}$$

Step 2b Formation of block matrix H_1 The block matrix H_1 formed from the second row of $x_1[m, n]$ is given below.

$$H_1 = \begin{bmatrix} 2 & 0 \\ 4 & 2 \\ 0 & 4 \end{bmatrix}$$

Step 3 Formation of block Toeplitz matrix

Let the block Toeplitz matrix be denoted by A. The number of zeros to be appended in the block matrix A depends on the number of rows of $x_2[m, n]$. In this case, $x_2[m, n]$ has two rows; hence one zero has to be appended. The matrix A is given by

$$A = \begin{bmatrix} H_0 & 0 \\ H_1 & H_0 \\ 0 & H_1 \end{bmatrix}.$$

Then, substituting the values of H_0 and H_1 in the matrix A, we get

$$A = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 3 & 0 \\ 4 & 2 & 1 & 3 \\ 0 & 4 & 0 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

The resultant matrix $y[m, n]$ is obtained by multiplying the matrix A with a matrix whose elements are from $x_2[-m, -n]$.

$$y[m,n]=\begin{bmatrix}3&0&0&0\\1&3&0&0\\0&1&0&0\\2&0&3&0\\4&2&1&3\\0&4&0&1\\0&0&2&0\\0&0&4&2\\0&0&0&4\end{bmatrix}\times\begin{bmatrix}3\\2\\5\\1\end{bmatrix}=\begin{bmatrix}9\\9\\2\\21\\24\\9\\10\\22\\4\end{bmatrix}$$

The result $y[m, n]$ is a 3×3 matrix which is given by

$$y[m,n]=\begin{bmatrix}9&9&2\\21&24&9\\10&22&4\end{bmatrix}$$

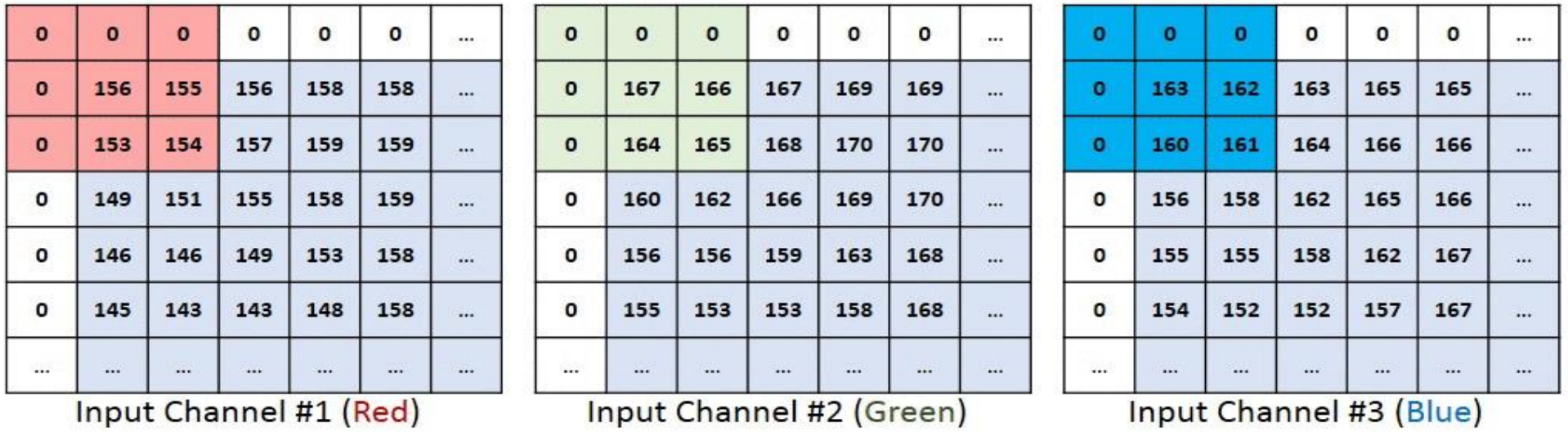
Determine the correlation between two matrices

$$x_1[m, n] = \begin{bmatrix} 1 & 5 \\ 2 & 4 \end{bmatrix} \qquad x_2[m, n] = \begin{bmatrix} 3 & 2 \\ 4 & 1 \end{bmatrix}$$

Determine the correlation between two matrices

$$x_1[m, n] = \begin{bmatrix} 3 & 2 \\ 1 & 5 \end{bmatrix} \qquad x_2[m, n] = \begin{bmatrix} 3 & 2 \\ 1 & 5 \end{bmatrix}$$

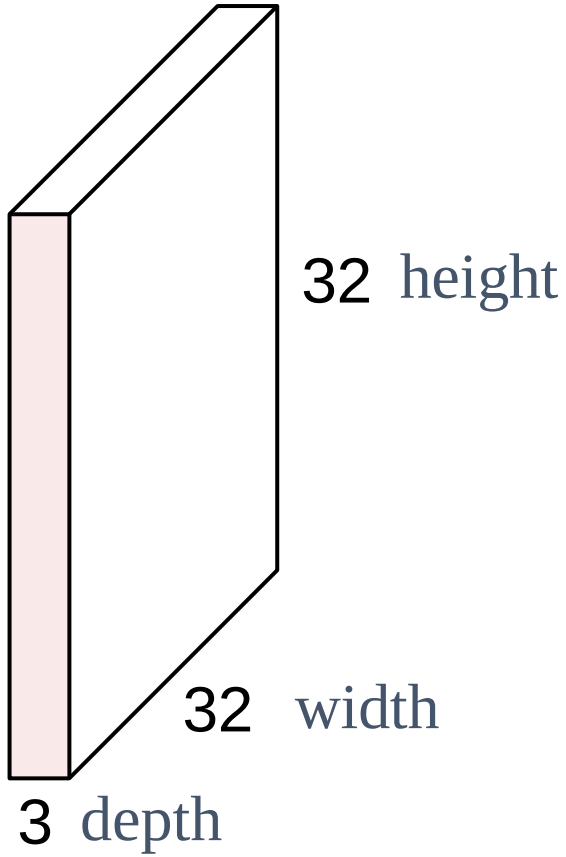
The Convolution operation



Convolution Layers

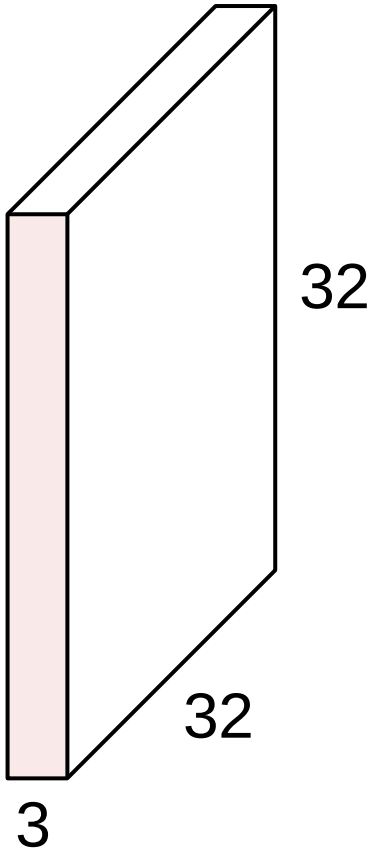
Convolution Layer

32x32x3 image

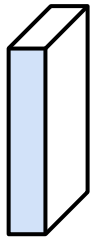


Convolution Layer

32x32x3 image



5x5x3 filter

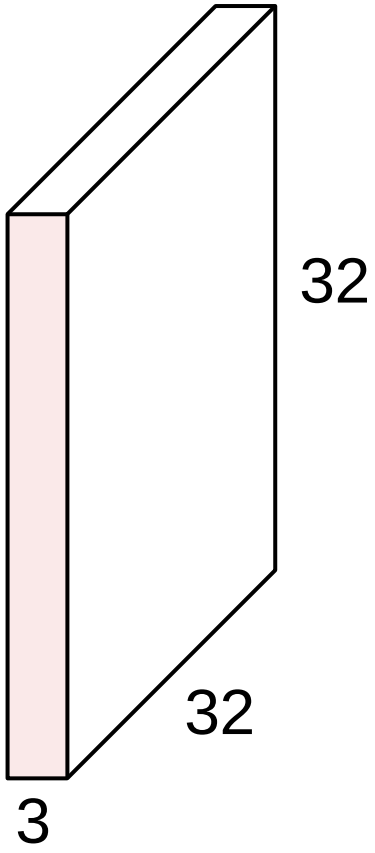


Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

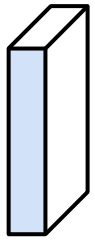
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

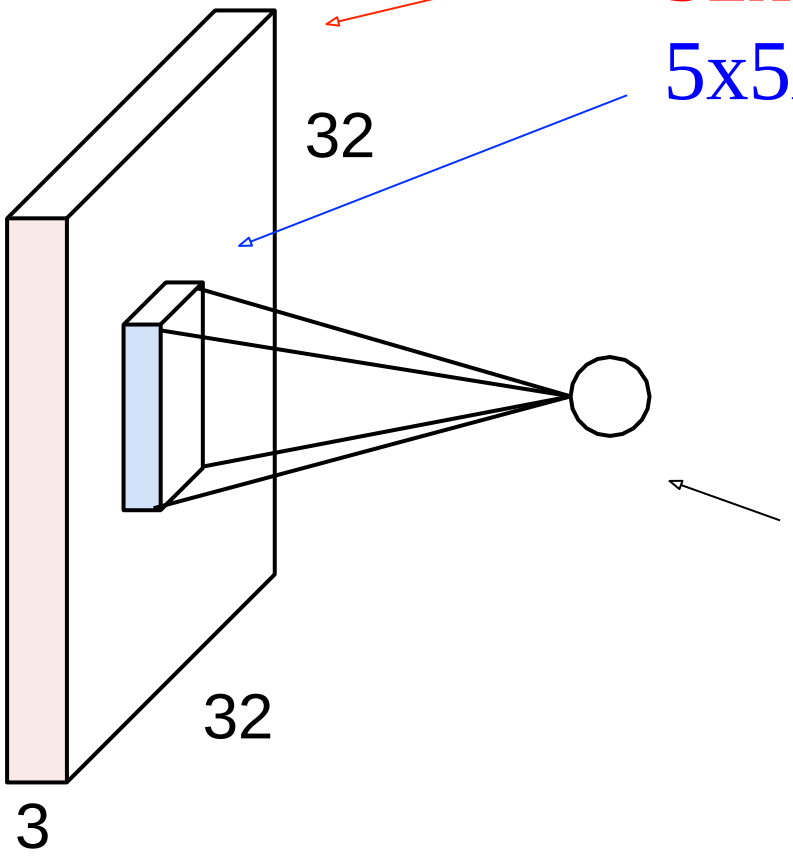


5x5x3 filter



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

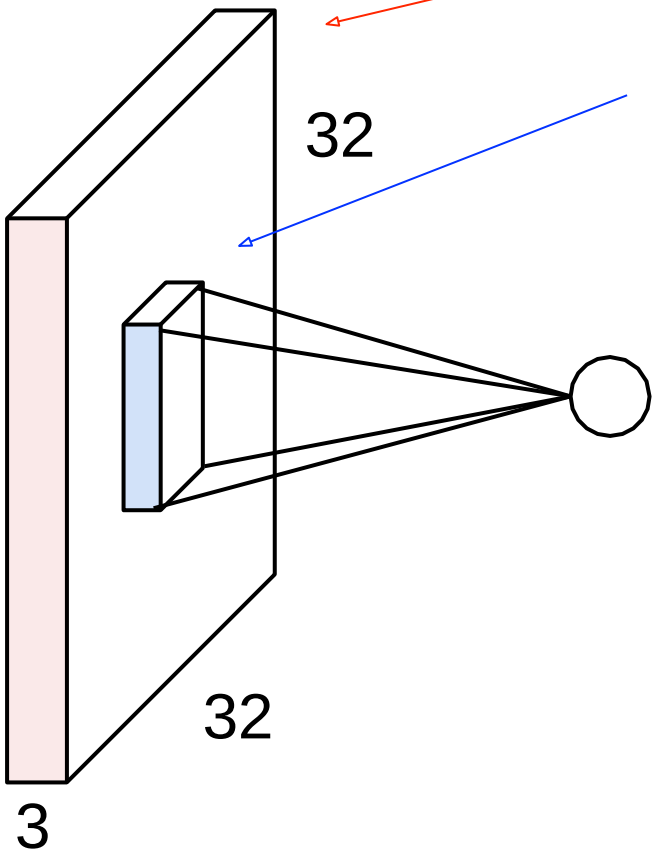
Convolution Layer



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

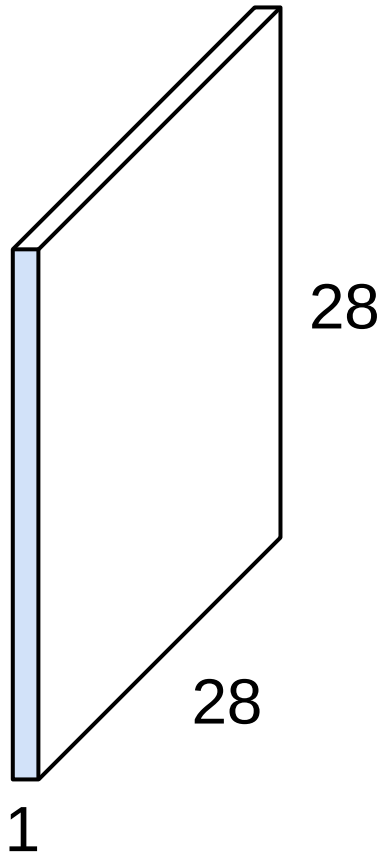


32x32x3 image

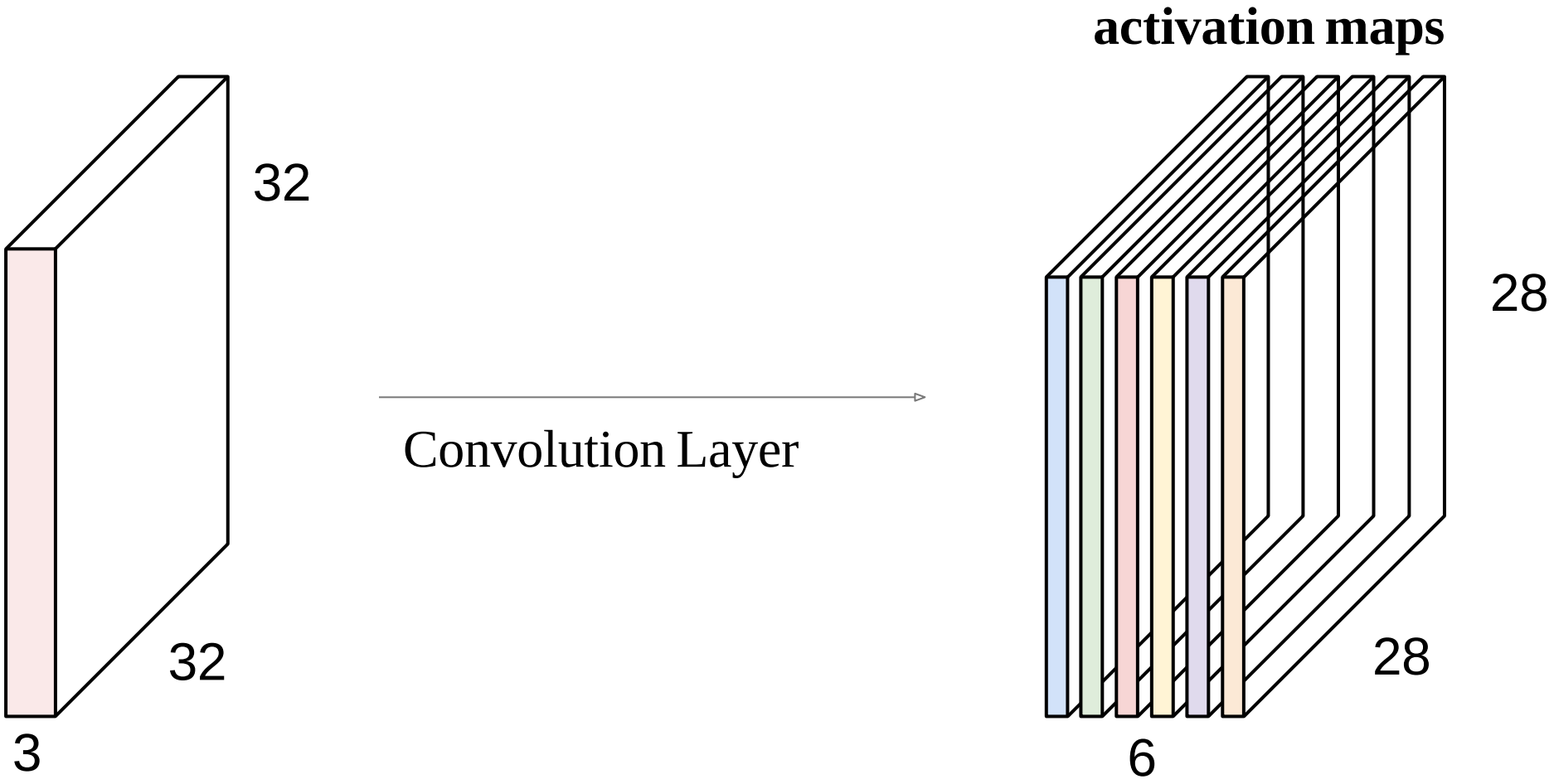
5x5x3 filter

convolve (slide) over all spatial locations

activation map

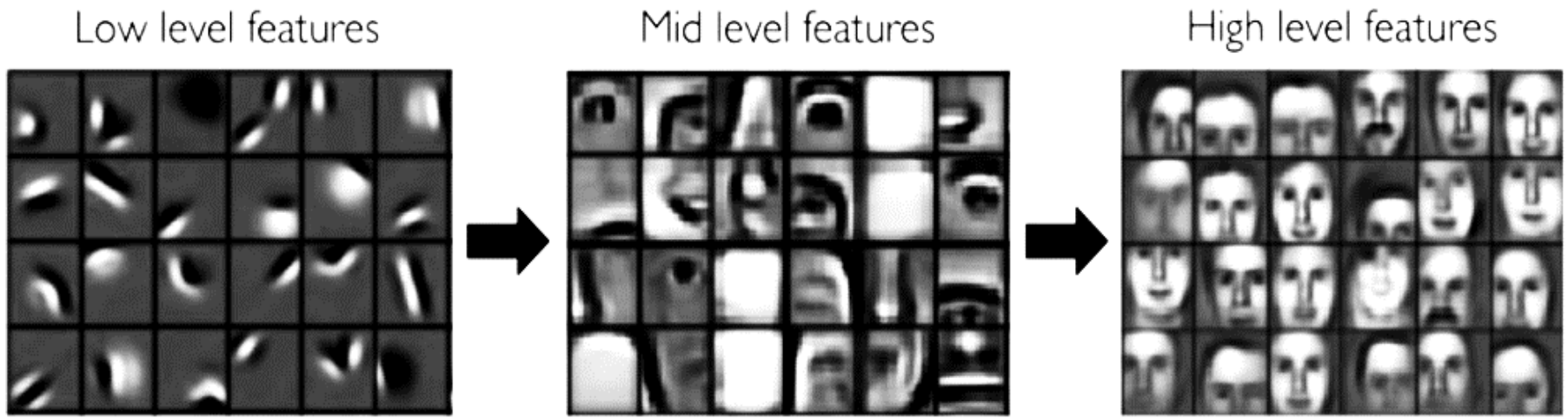


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

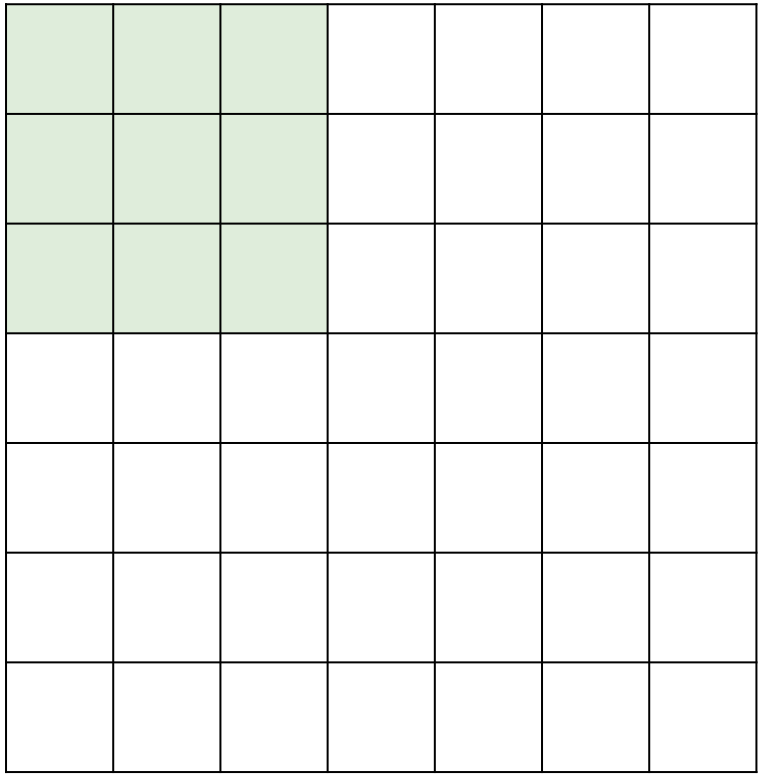
Representation Learning in Deep CNNs



Strides

A closer look at spatial dimensions:

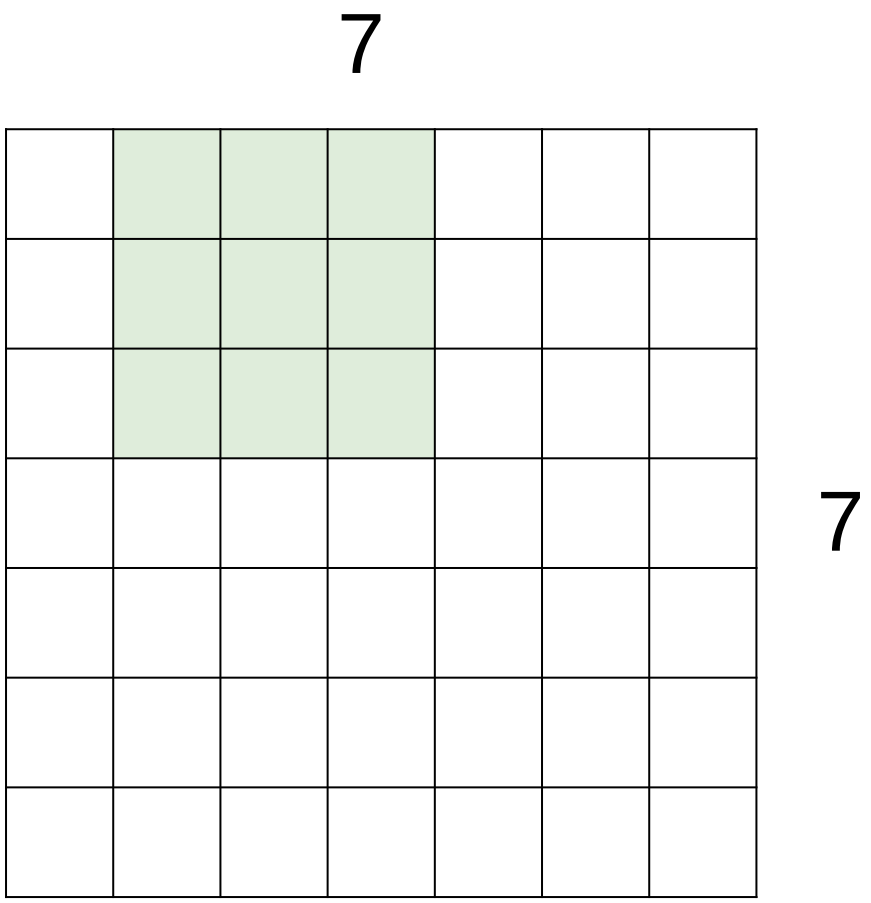
7



7

7x7 input (spatially) assume 3x3 filter

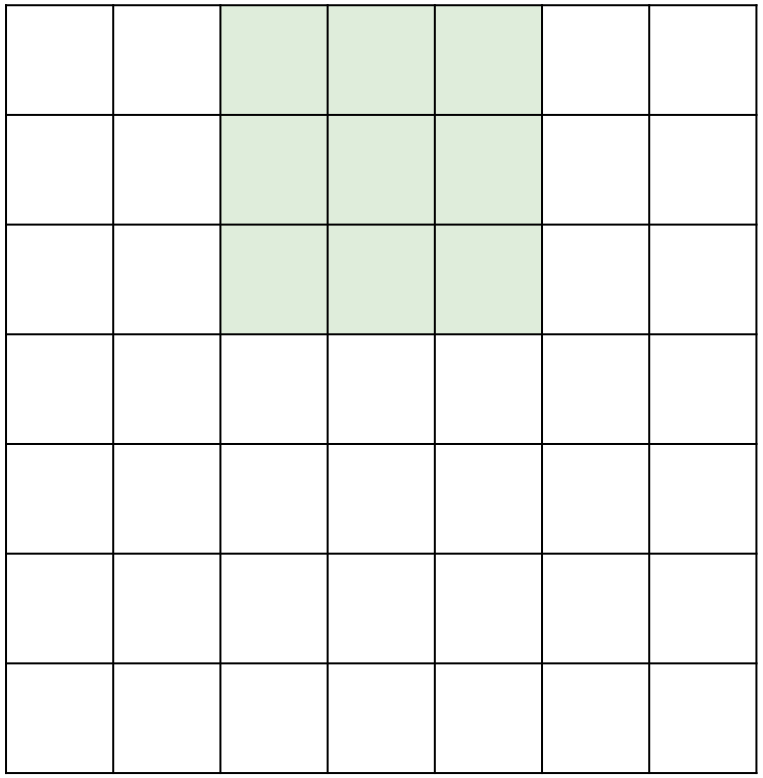
A closer look at spatial dimensions:



7x7 input (spatially) assume
3x3 filter

A closer look at spatial dimensions:

7

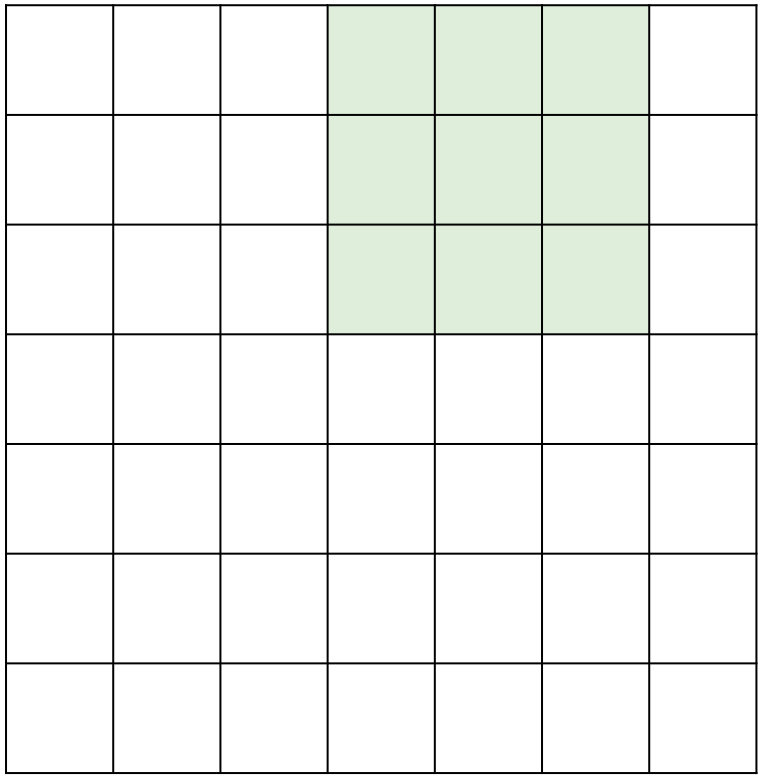


7

7x7 input (spatially) assume
3x3 filter

A closer look at spatial dimensions:

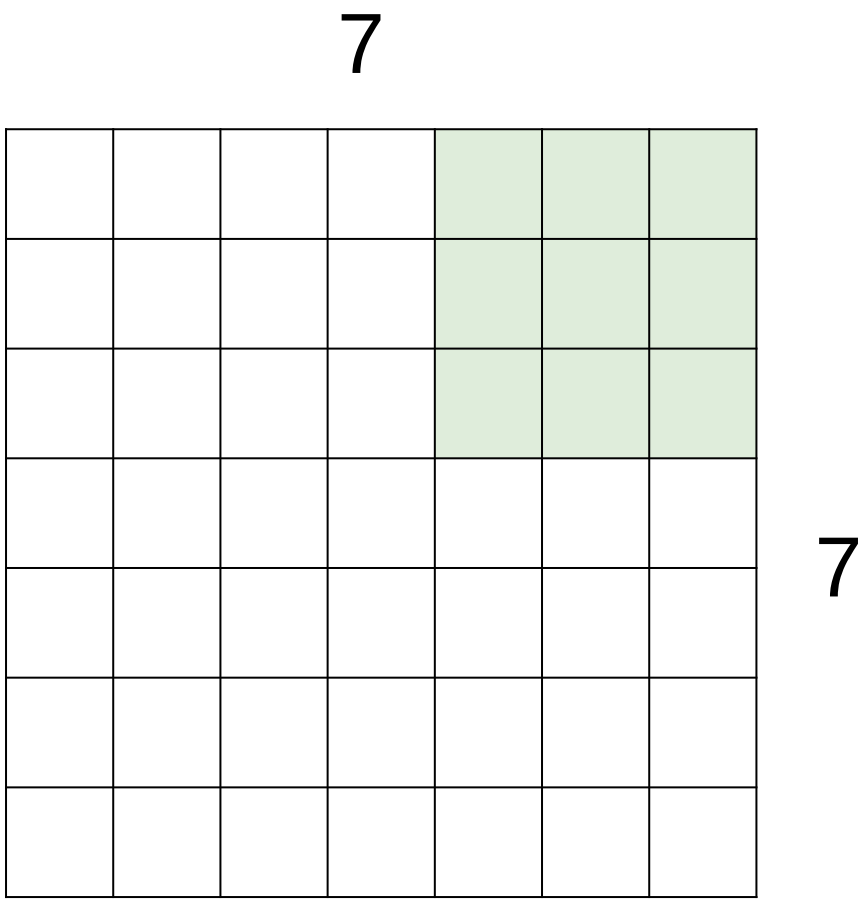
7



7

7x7 input (spatially) assume
3x3 filter

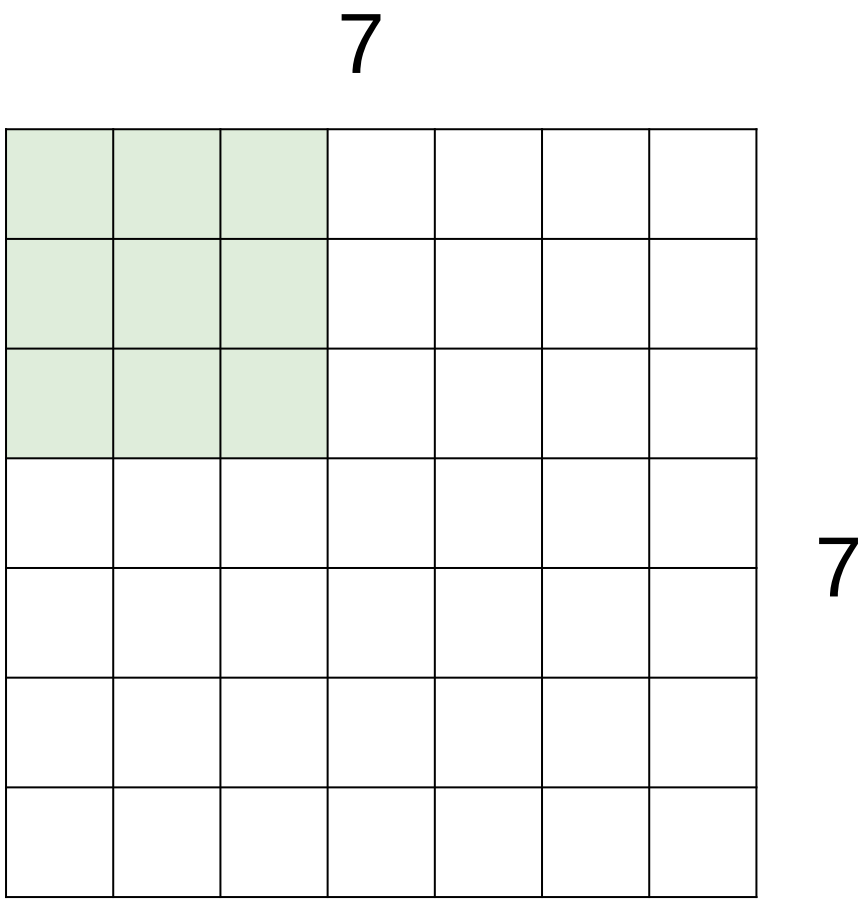
A closer look at spatial dimensions:



7x7 input (spatially) assume
3x3 filter

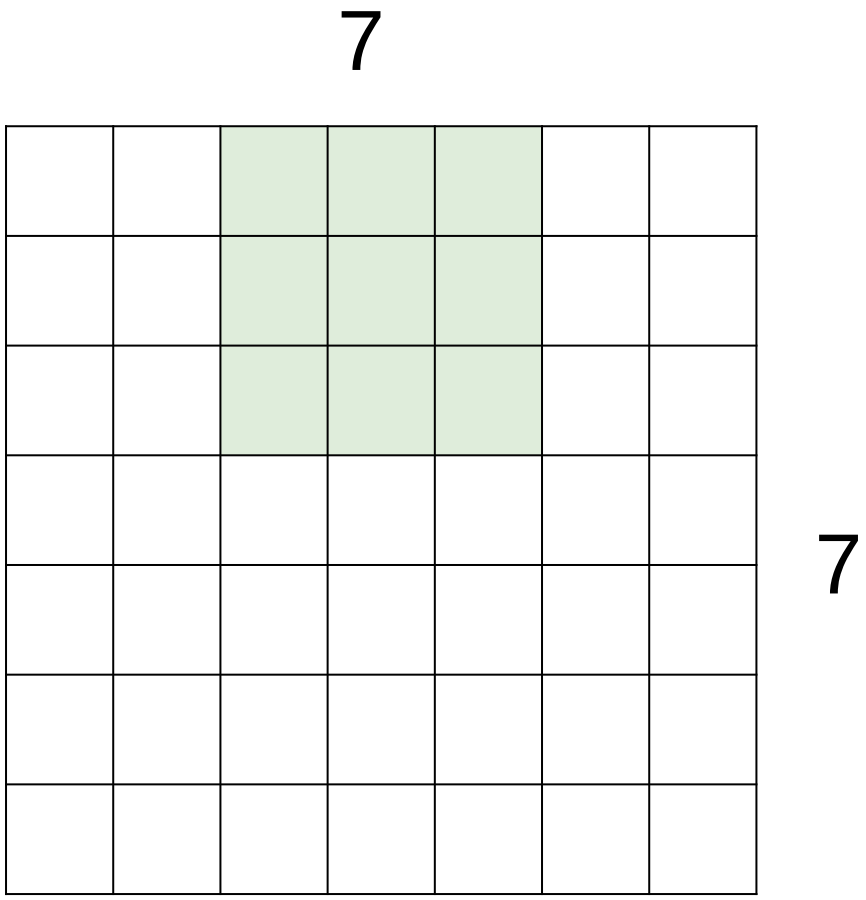
=> 5x5 output

A closer look at spatial dimensions:



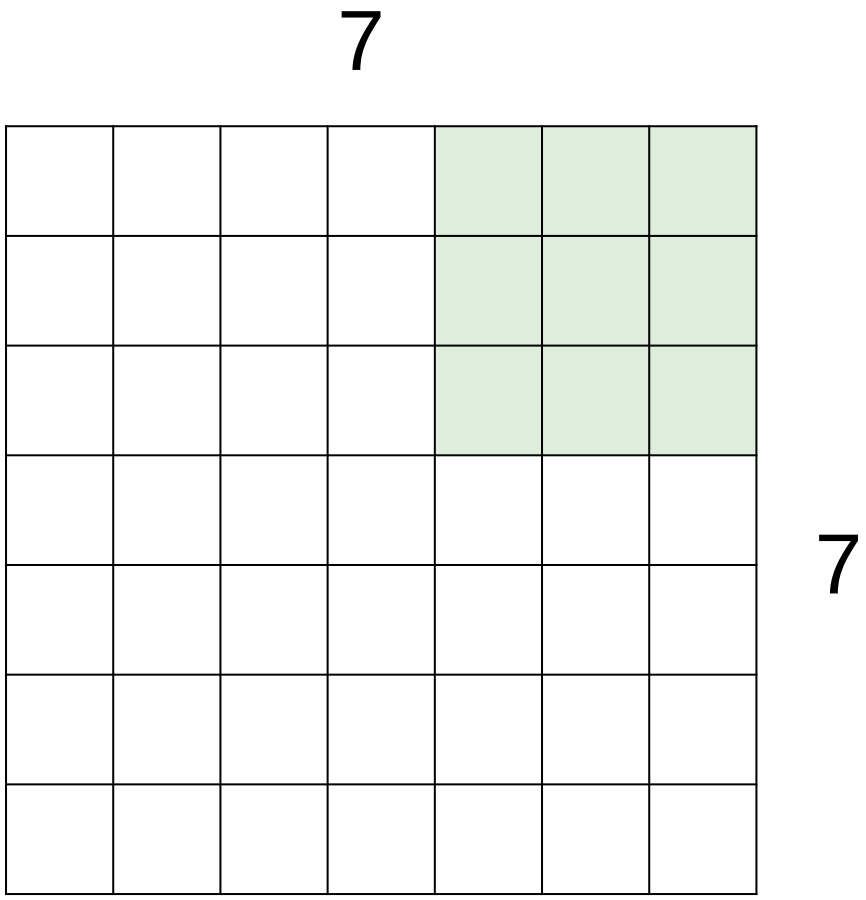
7x7 input (spatially) assume 3x3 filter applied **with stride 2**

A closer look at spatial dimensions:



7x7 input (spatially) assume
3x3 filter applied **with stride 2**

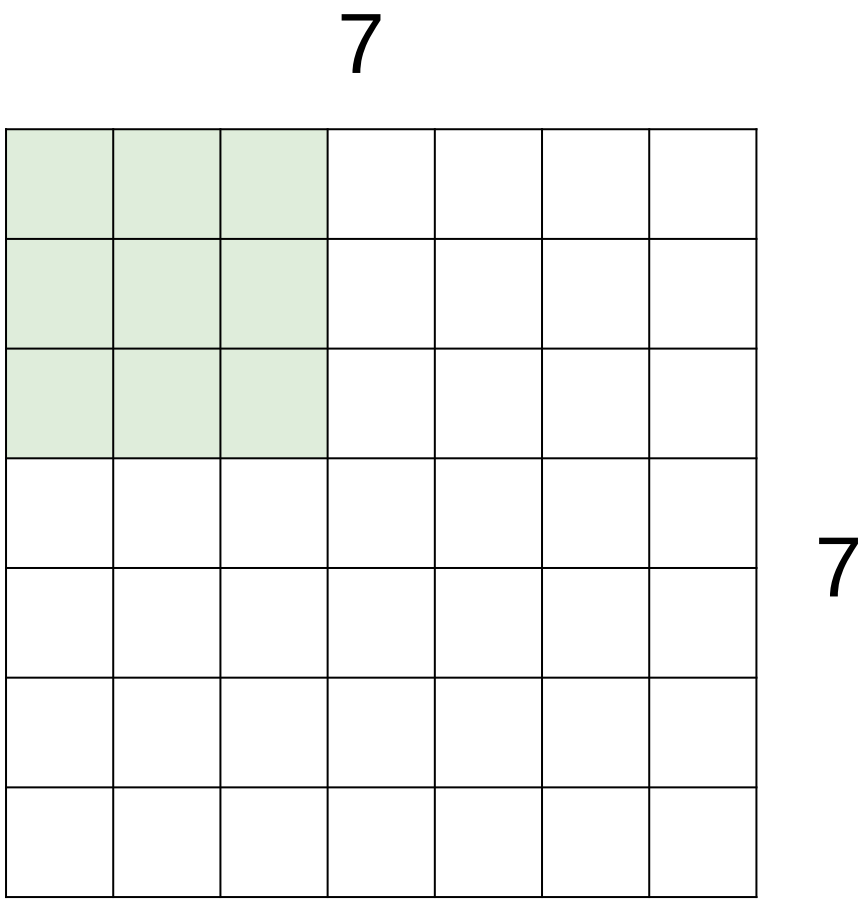
A closer look at spatial dimensions:



7x7 input (spatially) assume
3x3 filter applied **with stride 2**

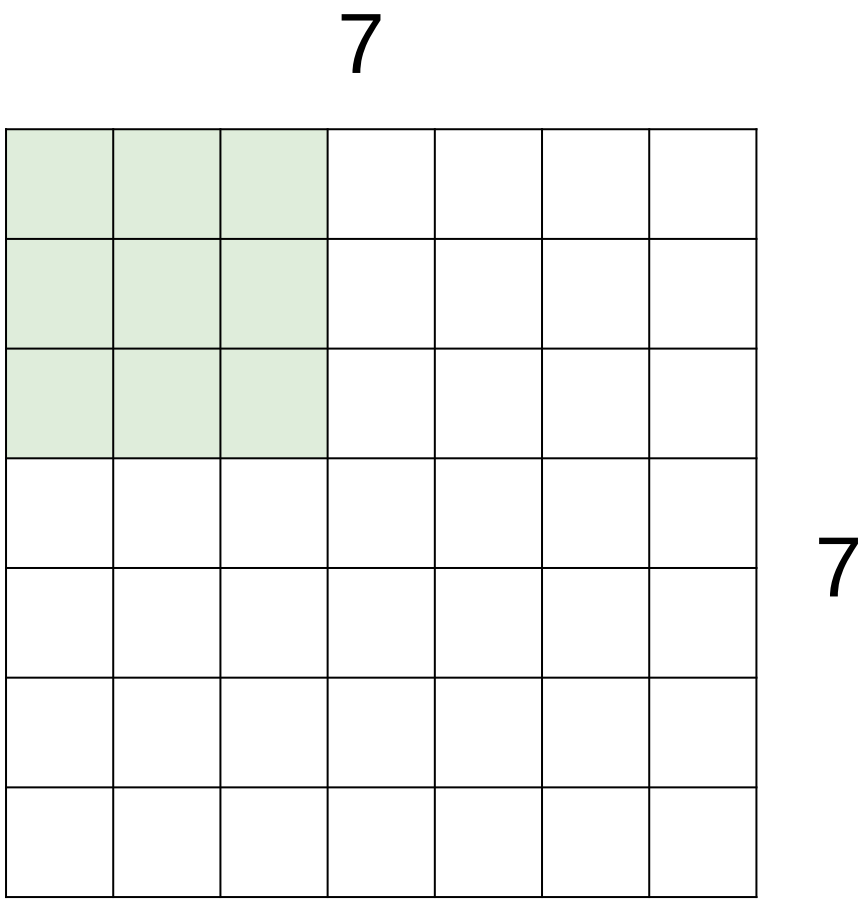
=> 3x3 output!

A closer look at spatial dimensions:



7x7 input (spatially) assume 3x3 filter applied **with stride 3?**

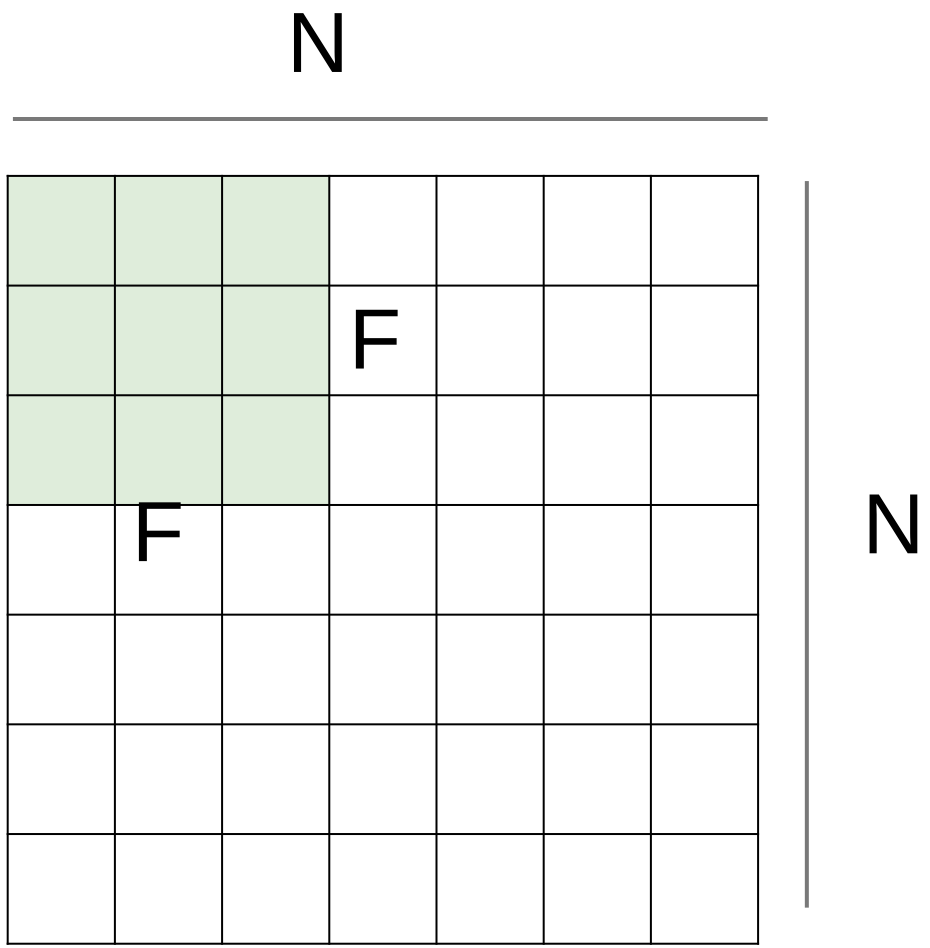
A closer look at spatial dimensions:



7x7 input (spatially) assume 3x3 filter applied **with stride 3?**

doesn't fit!

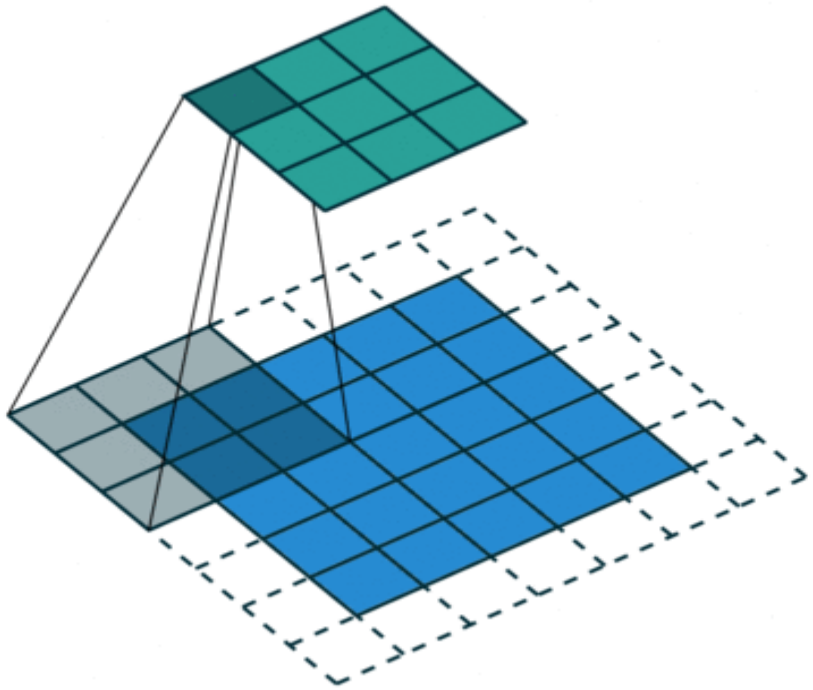
cannot apply 3x3 filter on 7x7 input with stride 3.

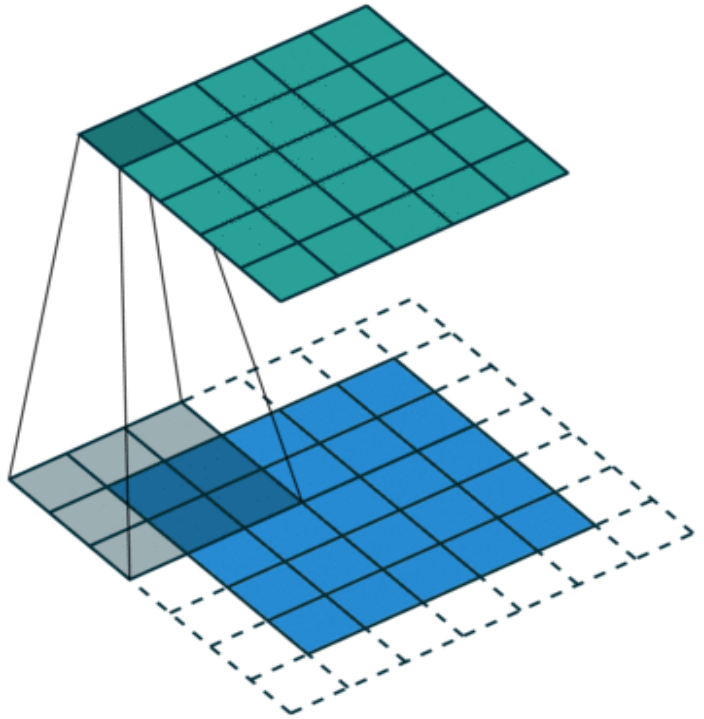


Output size:
(N - F) / stride + 1

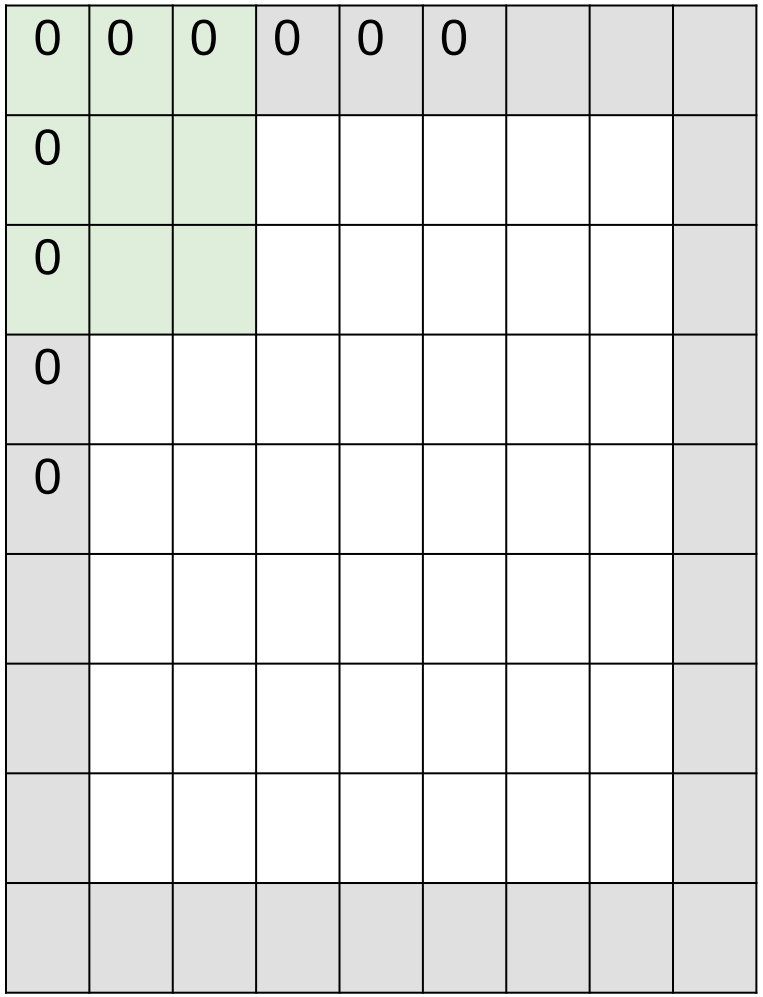
e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33

Zero-Padding





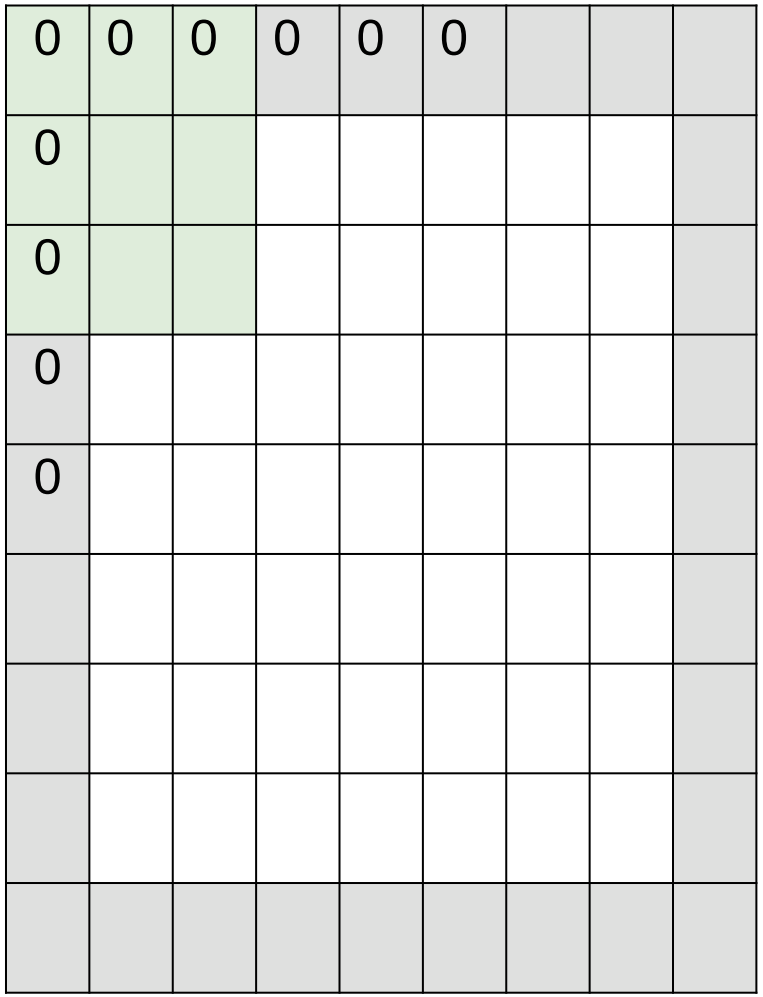
Zero-Padding: common to the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

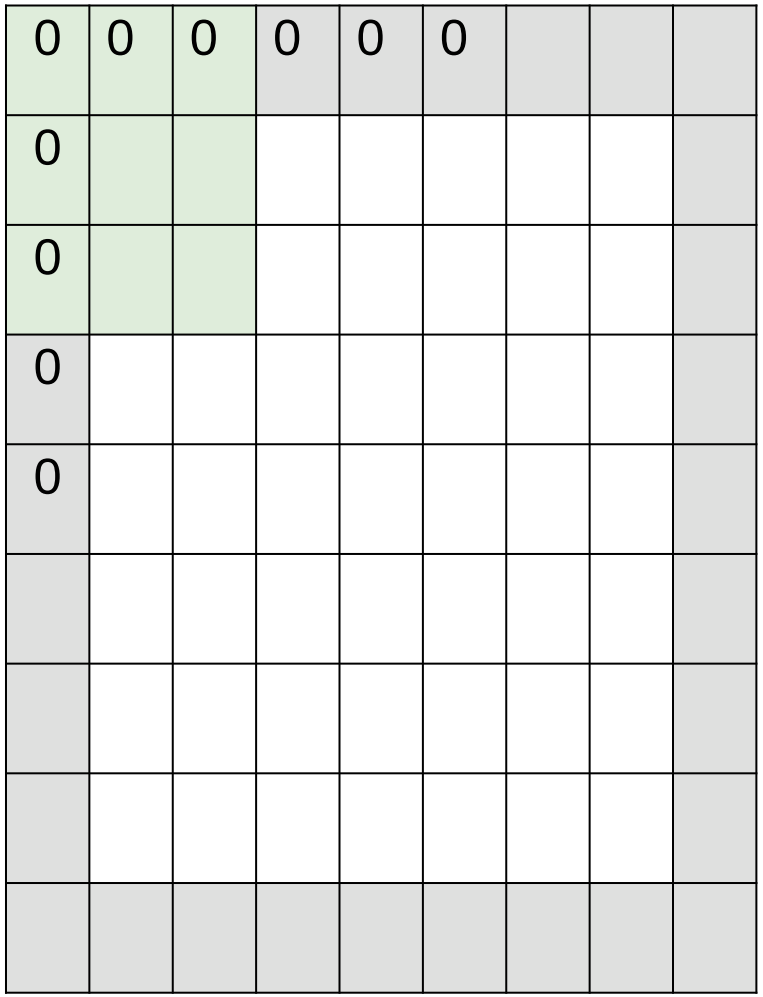
Zero-Padding: common to the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

Zero-Padding: common to the border



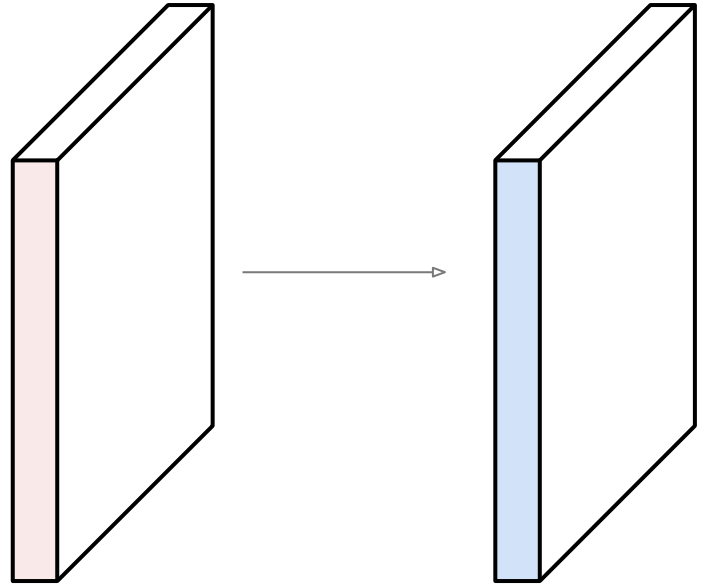
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1 F = 5
=> zero pad with 2 F = 7 =>
zero pad with 3

Examples:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

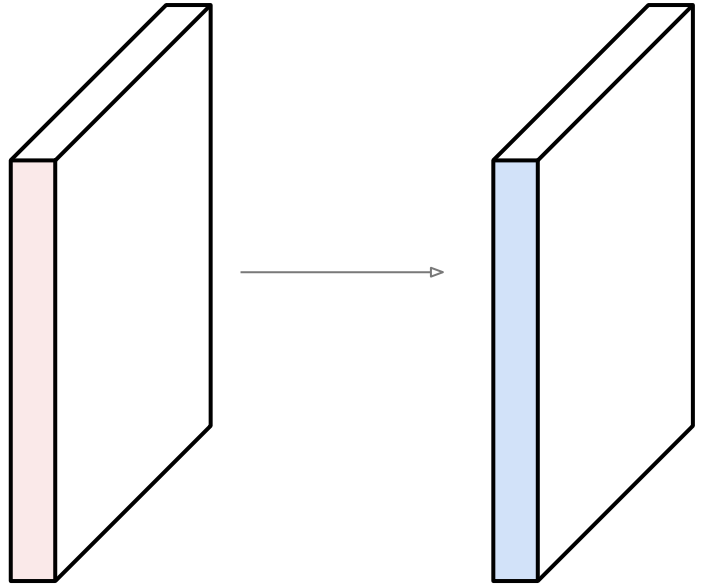
Output volume size: ?



Examples time:

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

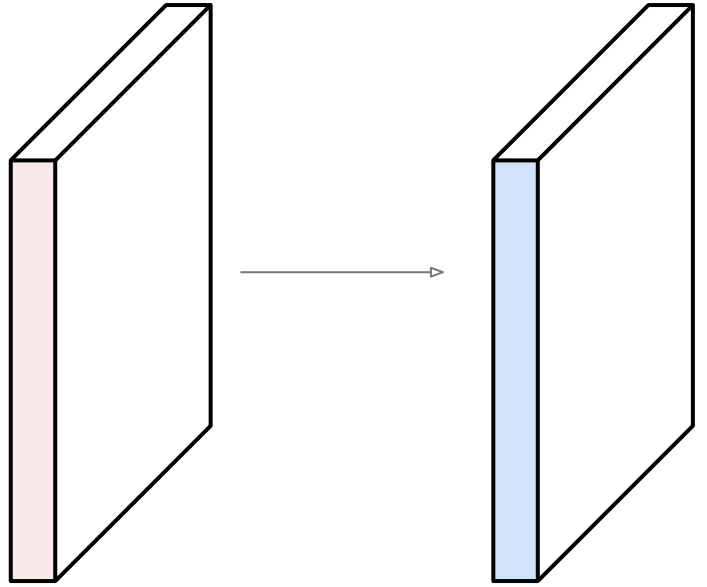
Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10



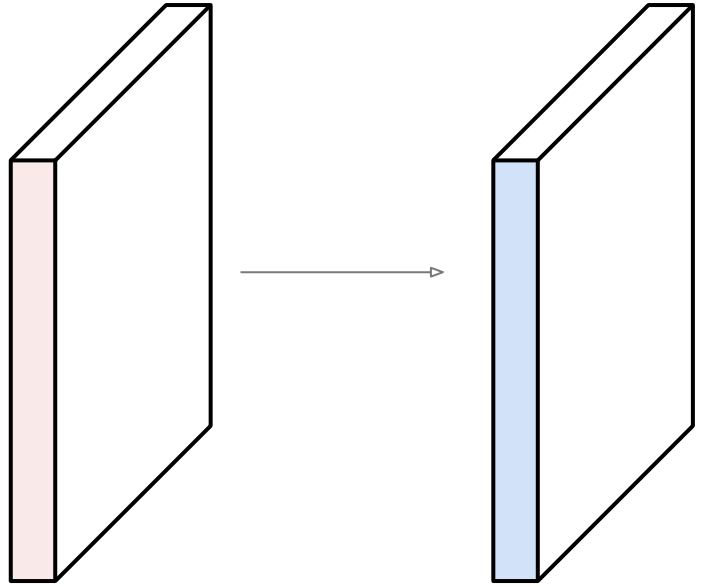
Examples :

Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?



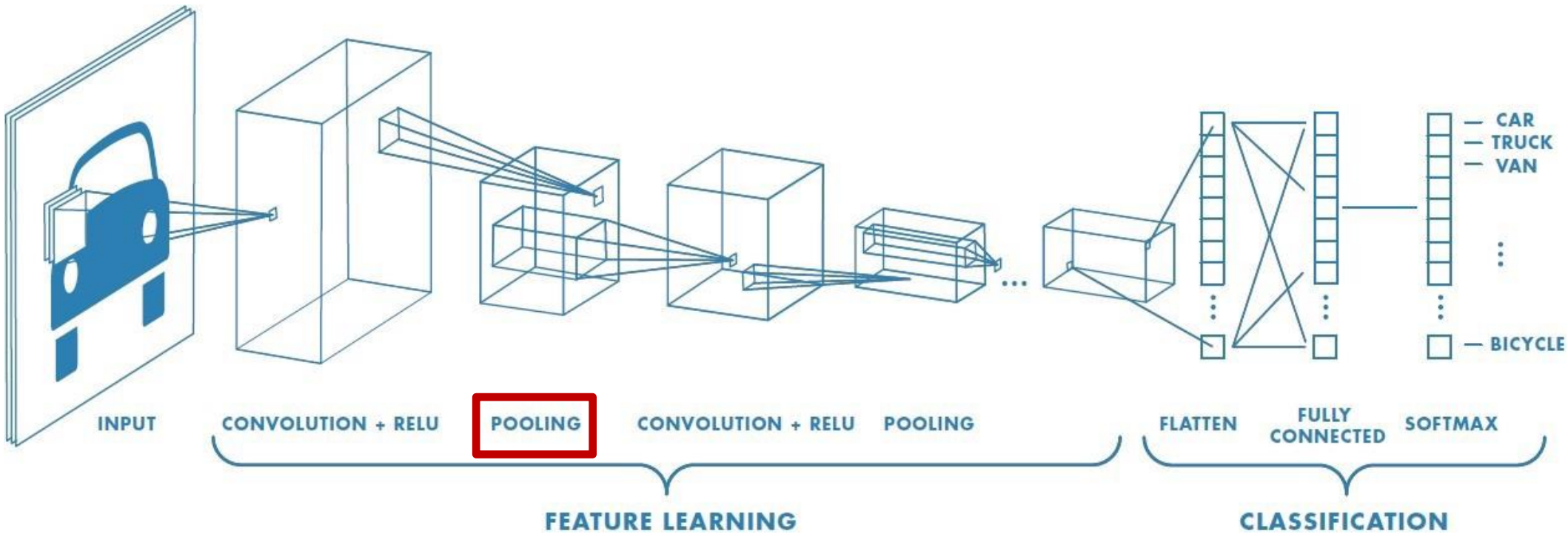
Examples :



Input volume: **32x32x**3
10 5x5 filters with stride 1, pad 2

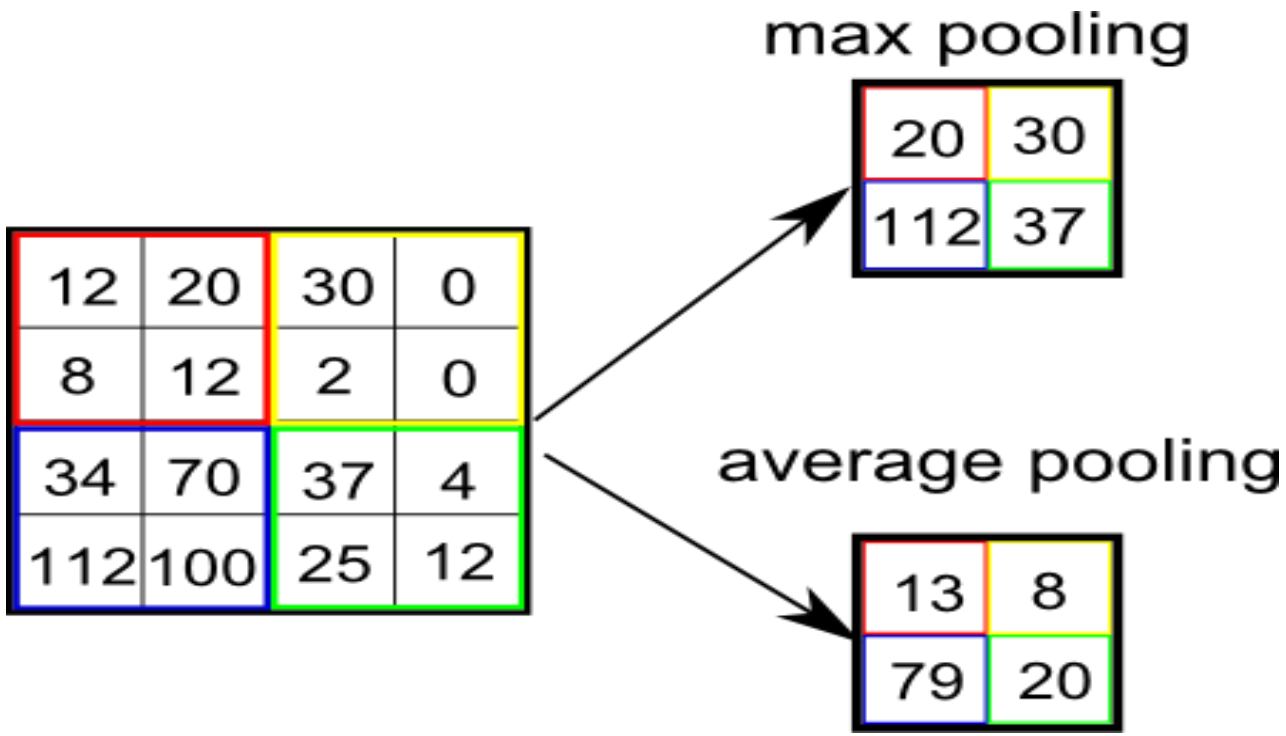
Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params
=> 76*10 = **760**

(+1 for bias)


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

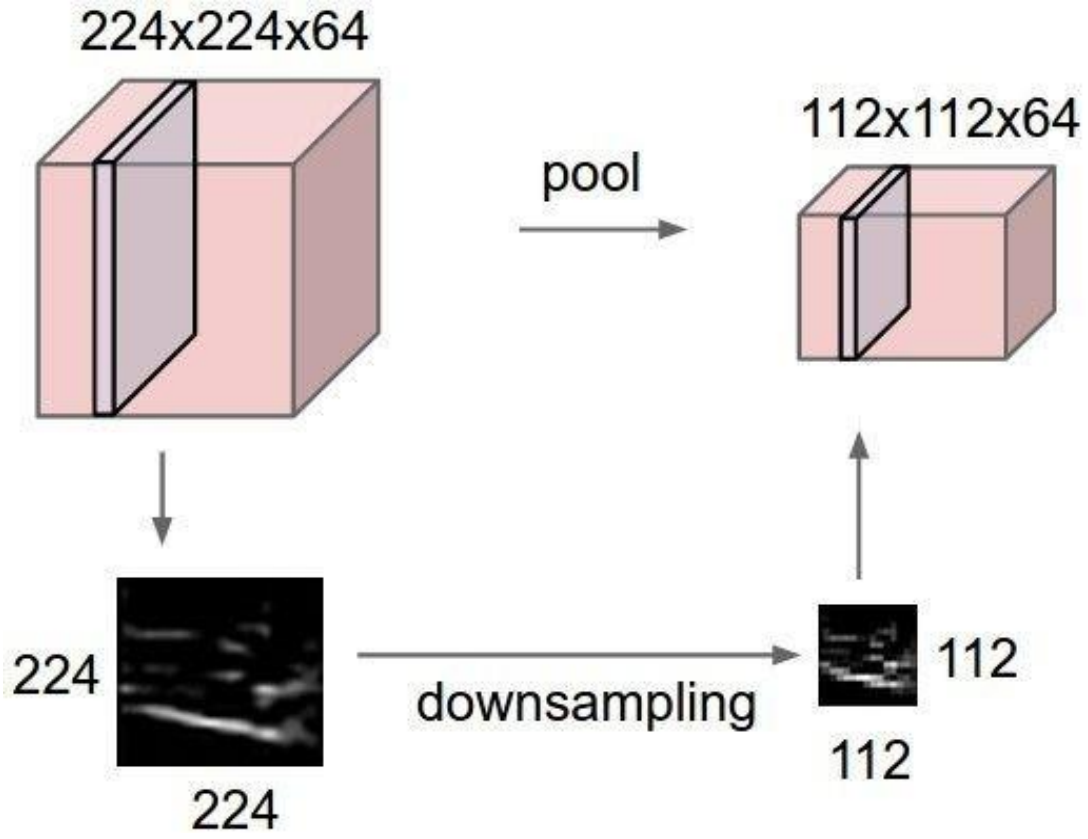
Pooling

Pooling


max pooling
average pooling
12 20 30 0
8 12 2 0
34 70 37 4
112 100 25 12
20 30
112 37
13 8
79 20

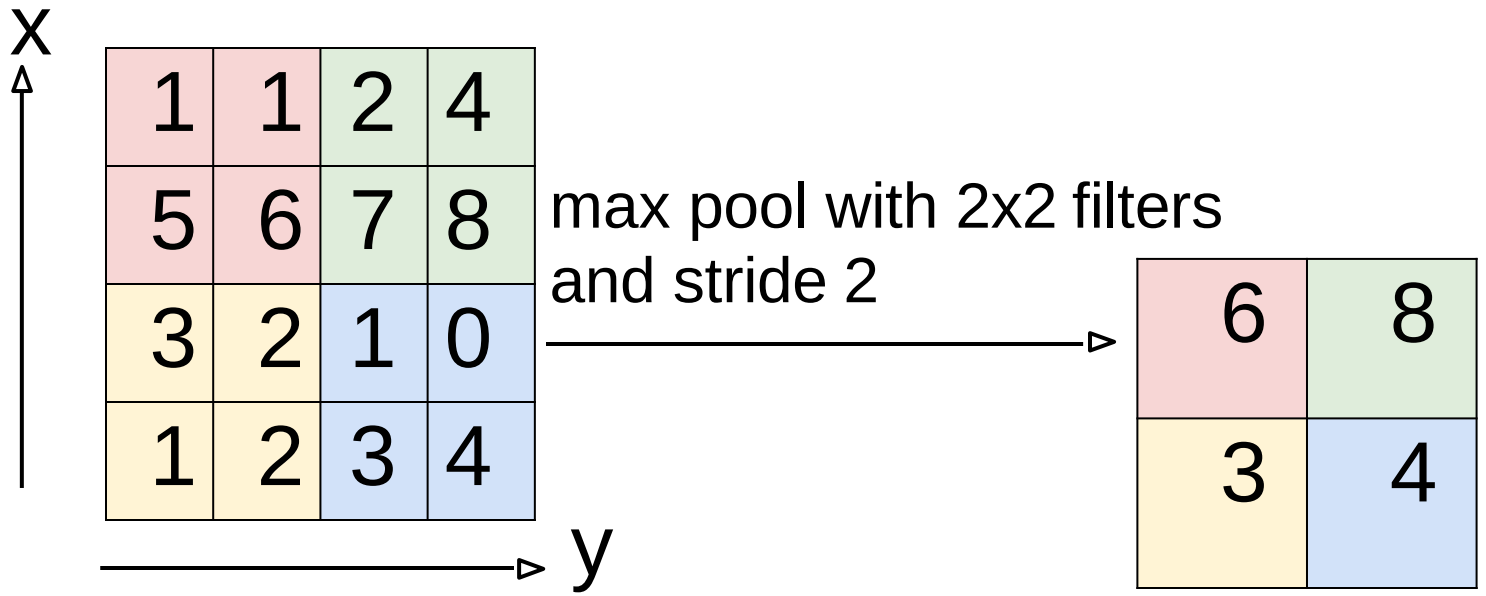
Pooling

- makes the representations smaller and more manageable
- operates over each activation map independently



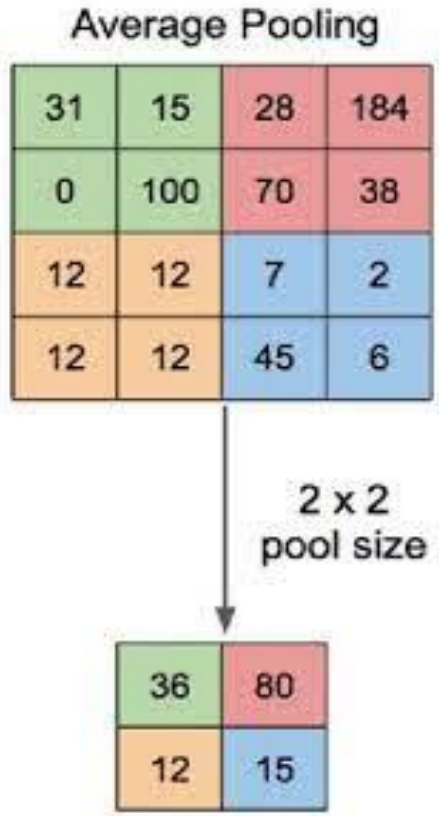
Max Pooling

Single depth slice



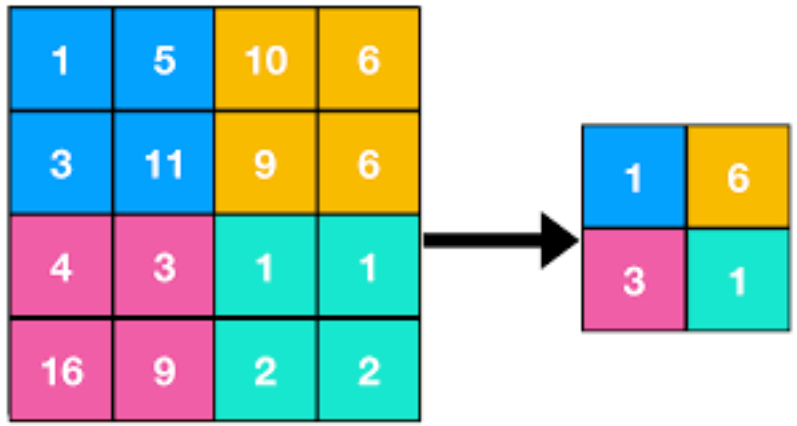
Average Pooling

- Image patches equivalent to kernel size are made
- **Average value** in each patch is the pooled value

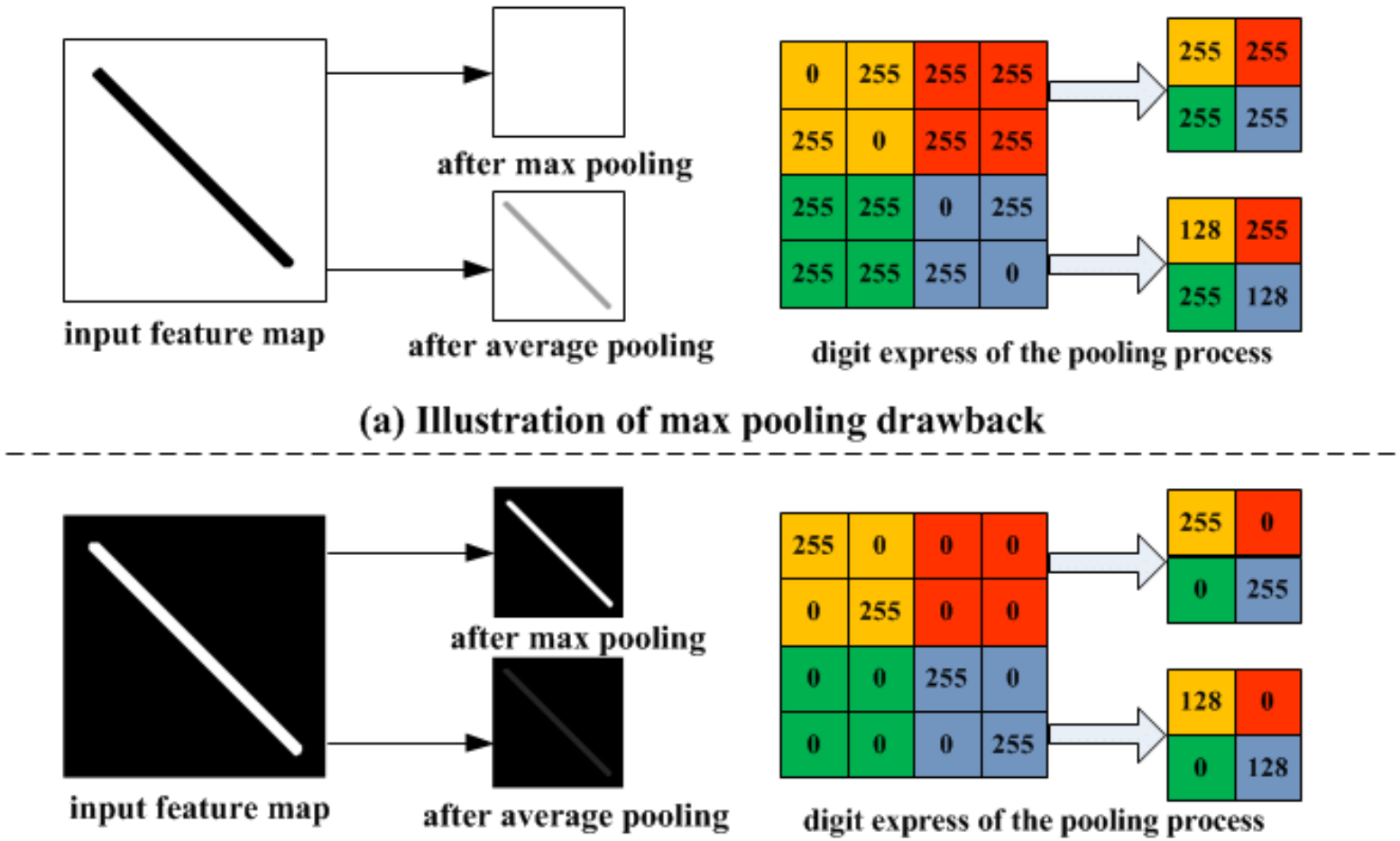


Min Pooling

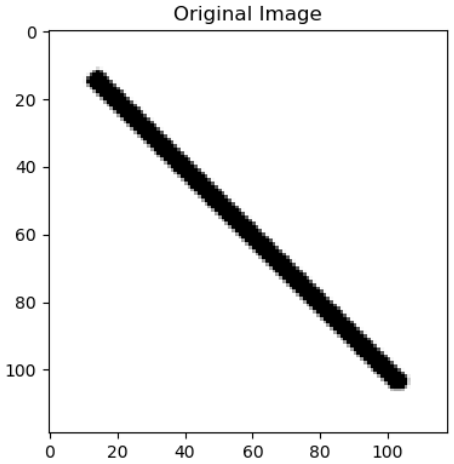
- Image patches equivalent to kernel size are made
- **Min value** in each patch is the pooled value

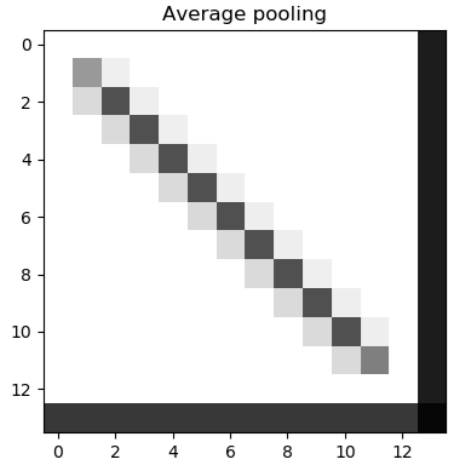


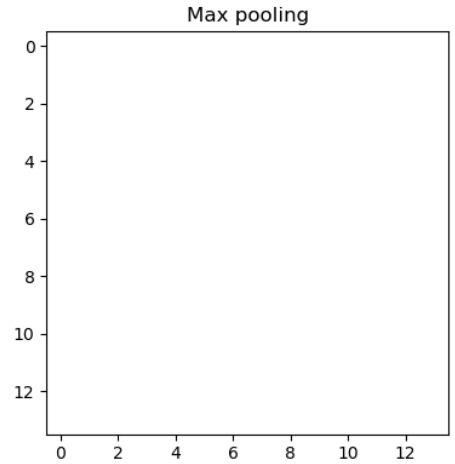
Pooling comparison

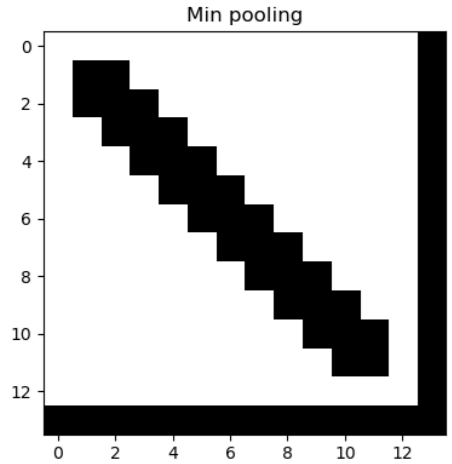


Pooling comparison







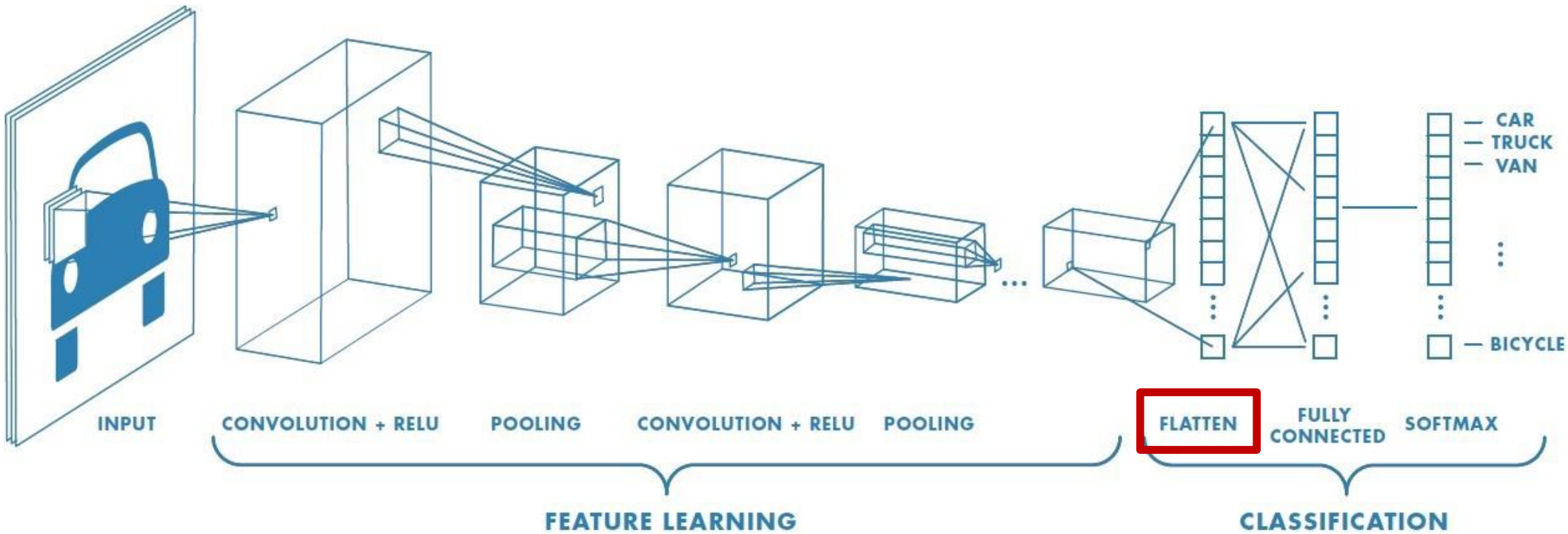


Pros

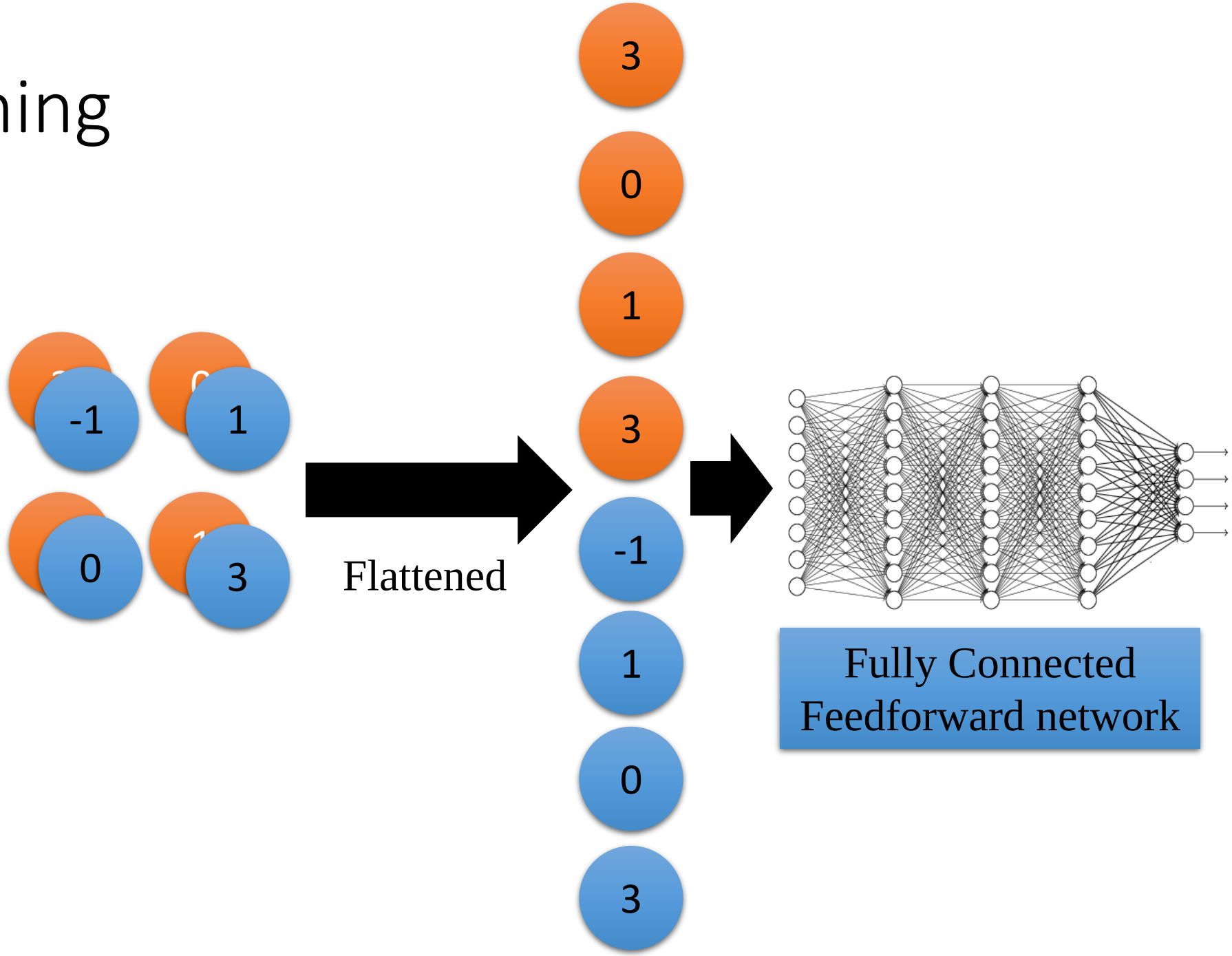
- Computational Efficiency
- Dimensionality Reduction
- Noise Reduction
- Translation invariance

Cons

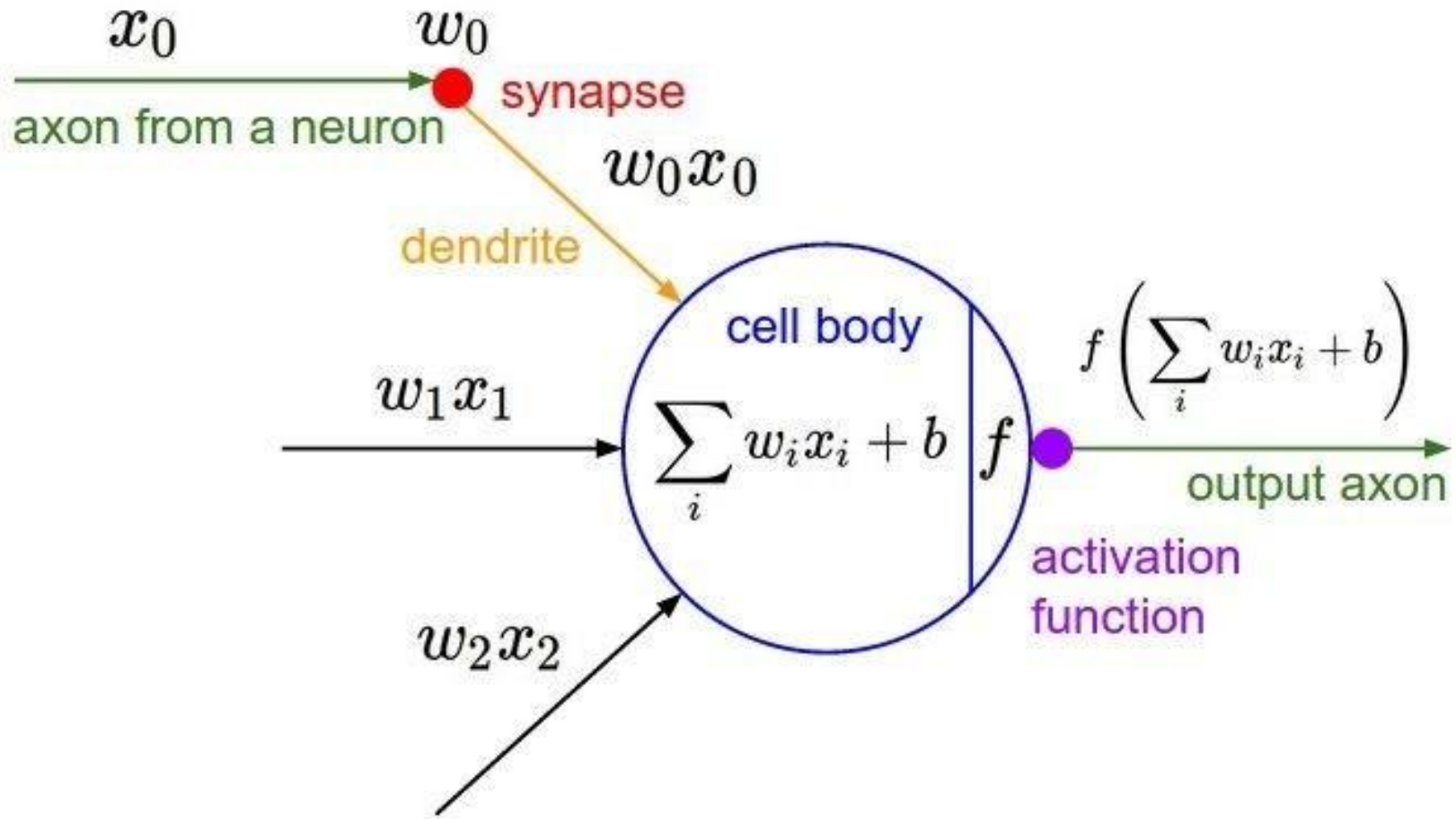
- Information loss
- Sensitivity to Hyperparameters
- Pooling artifacts


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

Flattening


-1
1
0
3
Flattened
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

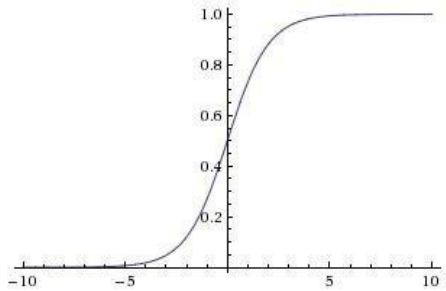
Activation Functions



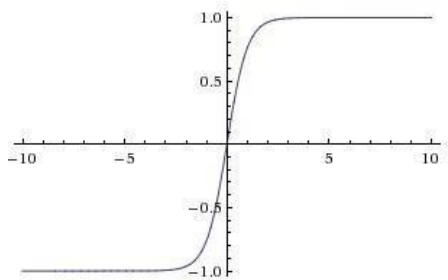
Activation Functions

Sigmoid

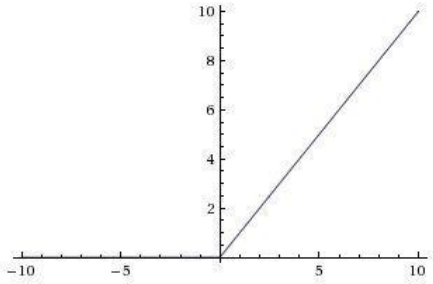
$$\sigma(x) = 1/(1+e^{-x})$$



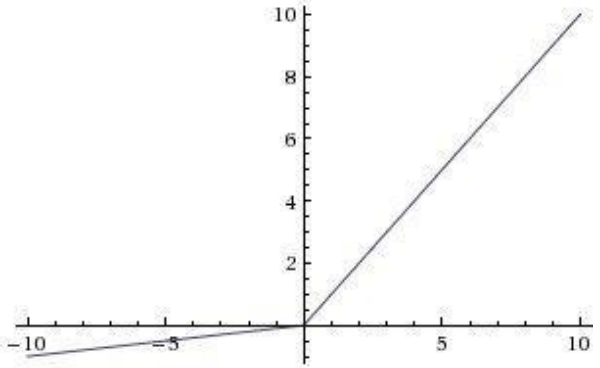
tanh tanh(x)



ReLU max(0,x)

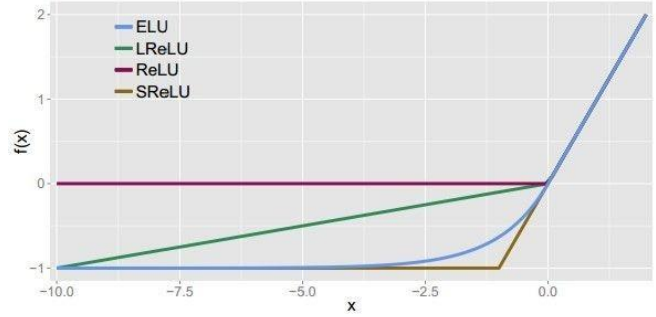


Leaky ReLU

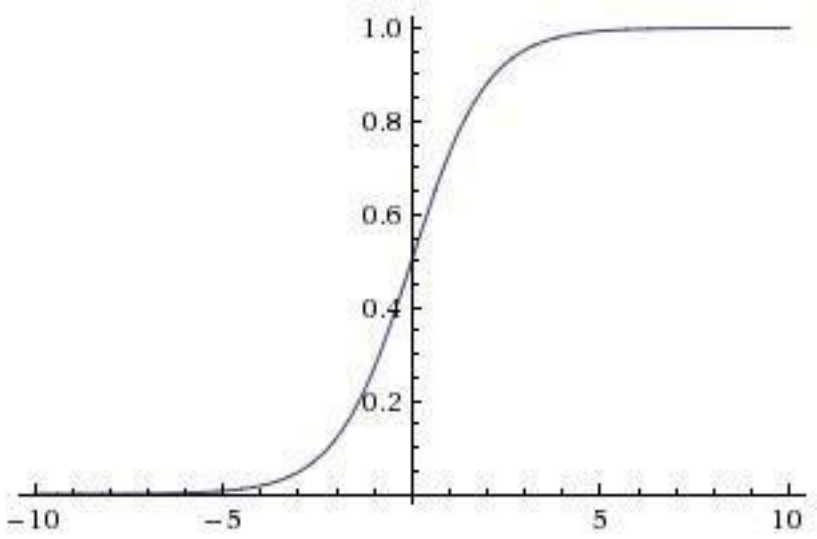


Maxout ELU $\max(w_1^T x + b_1, w_2^T x + b_2)$

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$



Activation Functions

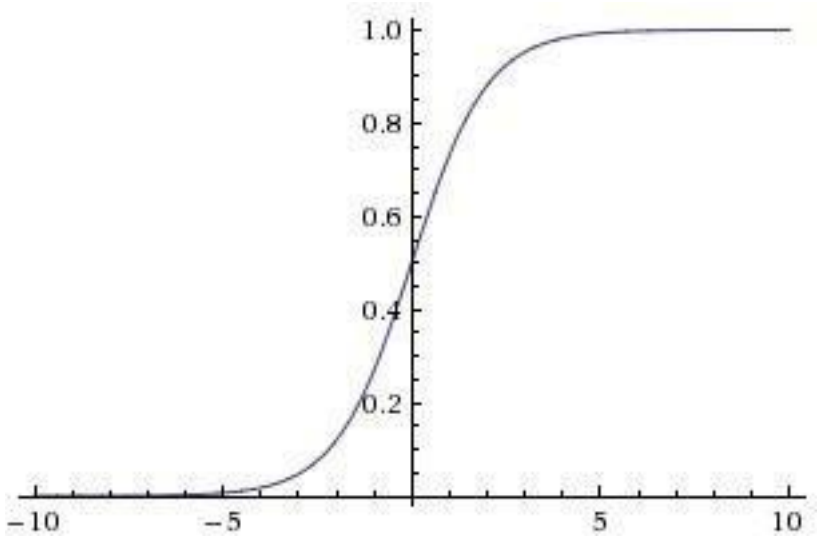


Sigmoid

$$\sigma(x) = 1/(1 + e^{-x})$$

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

Activation Functions



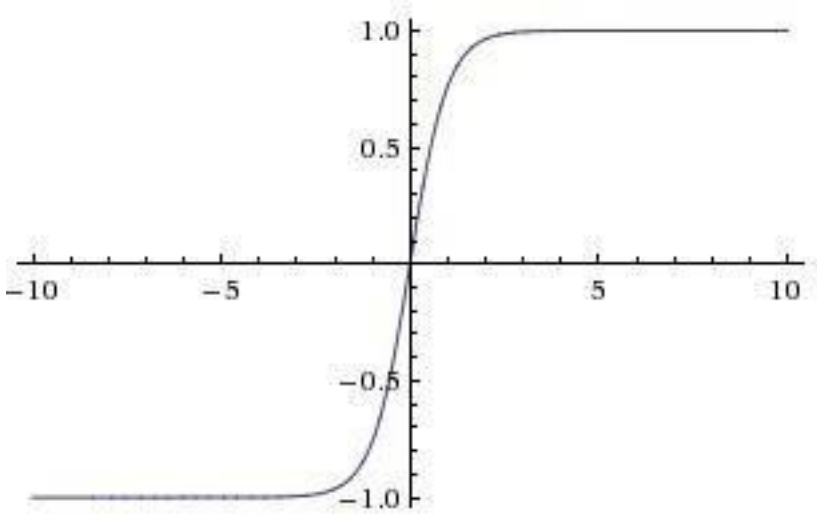
Sigmoid

$$\sigma(x) = 1/(1 + e^{-x})$$

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating “firing rate” of a neuron

1. Sigmoid outputs are not zero-centered
2. exp() is a bit compute expensive

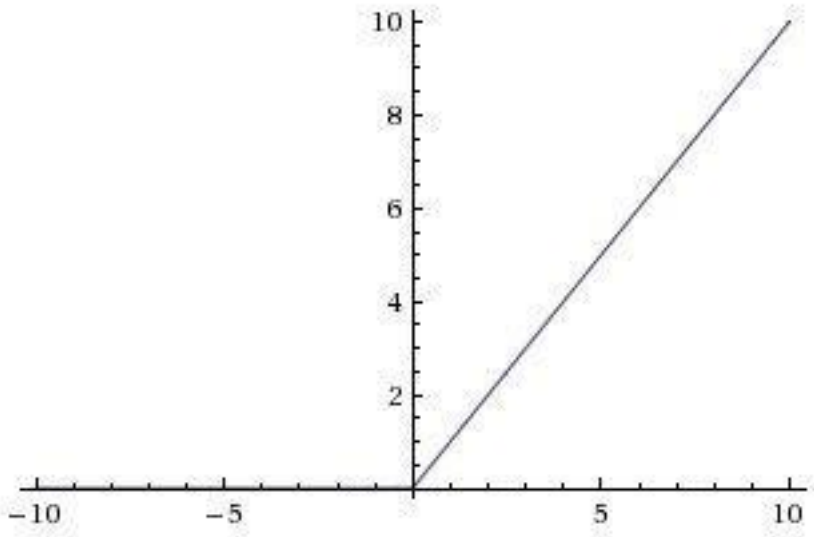
Activation Functions



tanh(x)

- Squashes numbers to range [-1,1]
- zero centered (nice)
- still kills gradients when saturated :(

Activation Functions



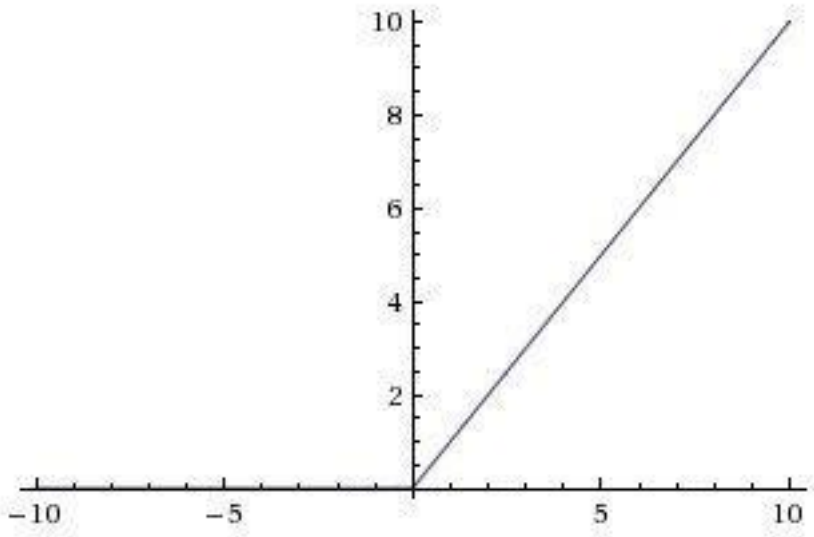
ReLU

(Rectified Linear Unit)

Computes **f(x) = max(0,x)**

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

Activation Functions



ReLU

(Rectified Linear Unit)

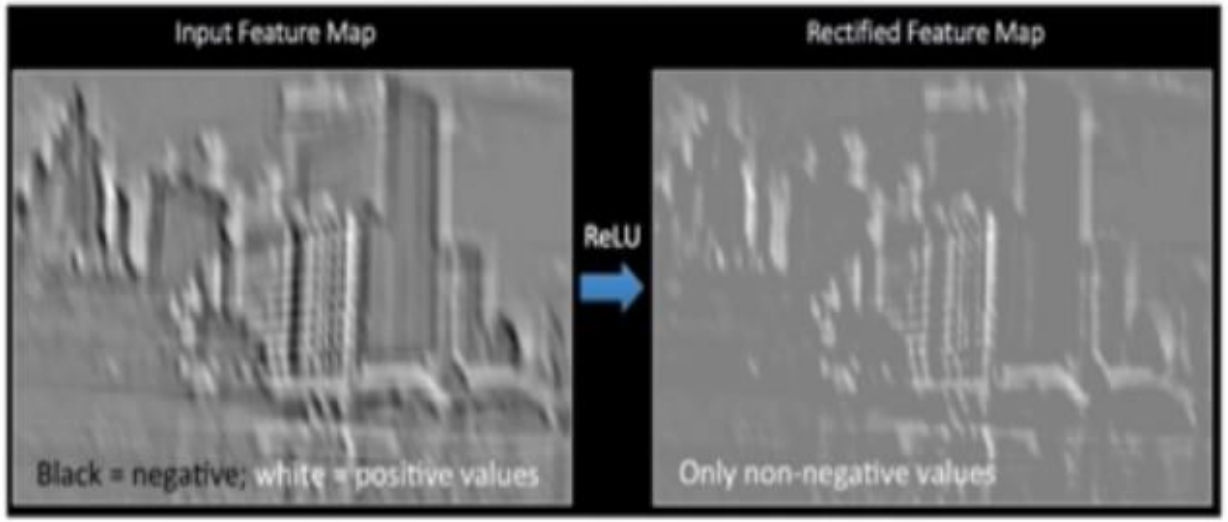
Computes **f(x) = max(0,x)**

- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

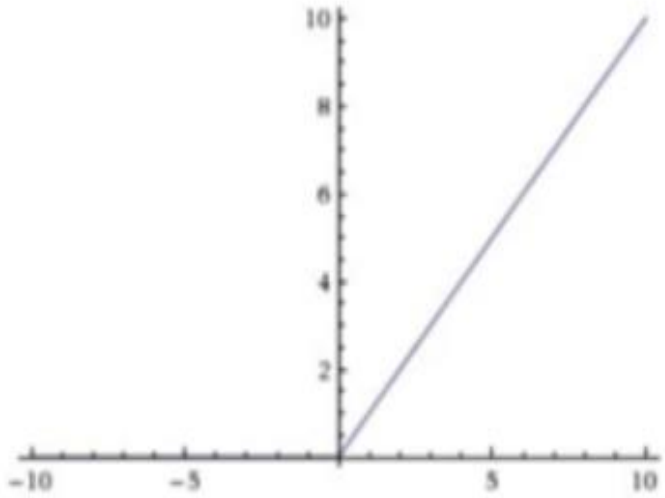
- Not zero-centered output

Introducing Non-Linearity

- Apply after every convolution operation (i.e., after convolutional layers)
- ReLU: pixel-by-pixel operation that replaces all negative values by zero. **Non-linear operation!**

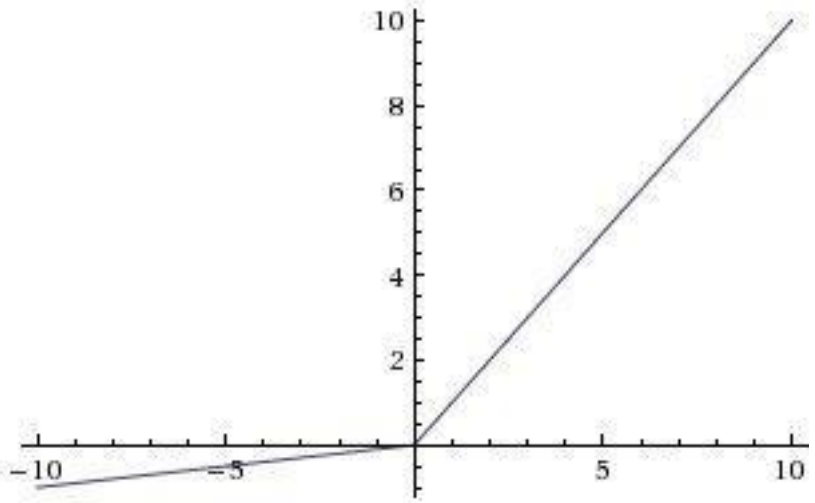


Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

Activation Functions



Leaky ReLU

$$f(x) = \max(0.01x, x)$$

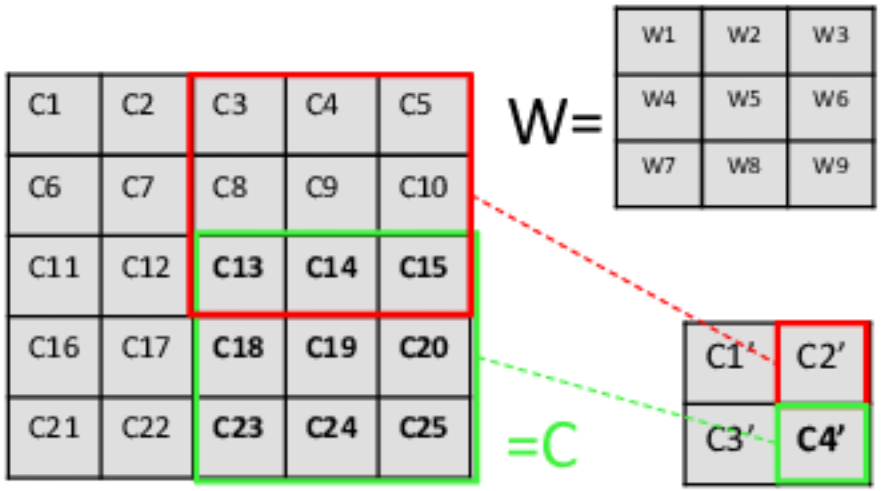
- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not “die”.**

Deconvolution: An Introduction

- Convolution reduces the dimension of data and extracts the features
- The image is translated to feature space
- How to represent features in image domain?

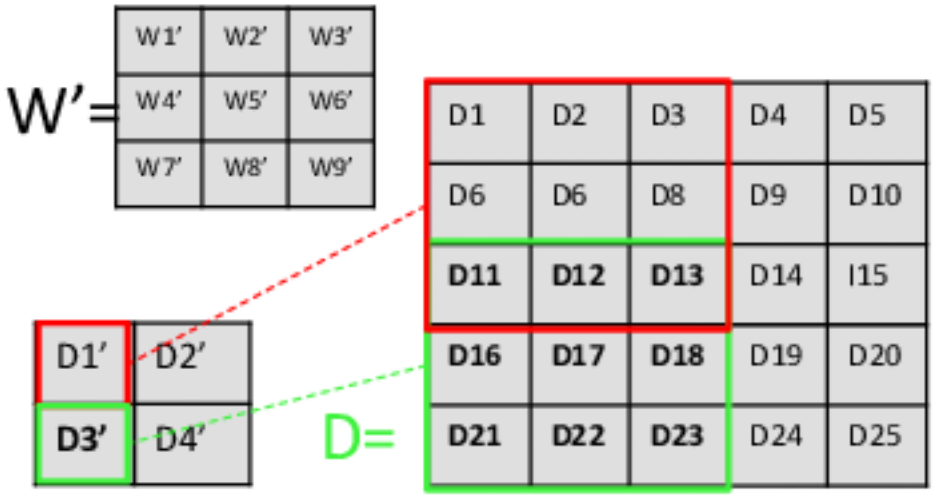
Deconvolution: An Introduction

- Deconvolution helps in achieving the inverse operation



$C4' = W * C$

Convolution (stride=2)

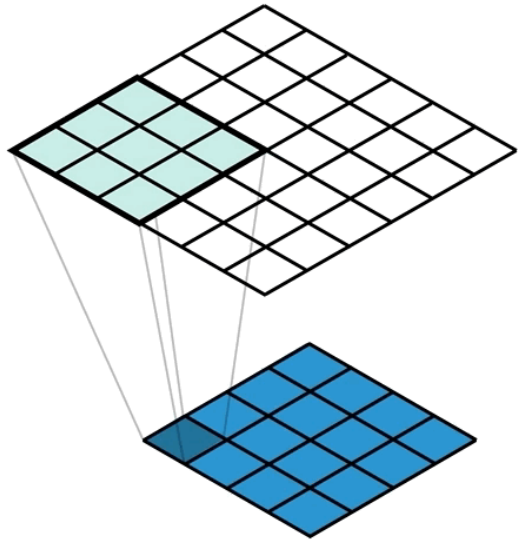


$D = D3' \times W'$

Deconvolution (stride=2)

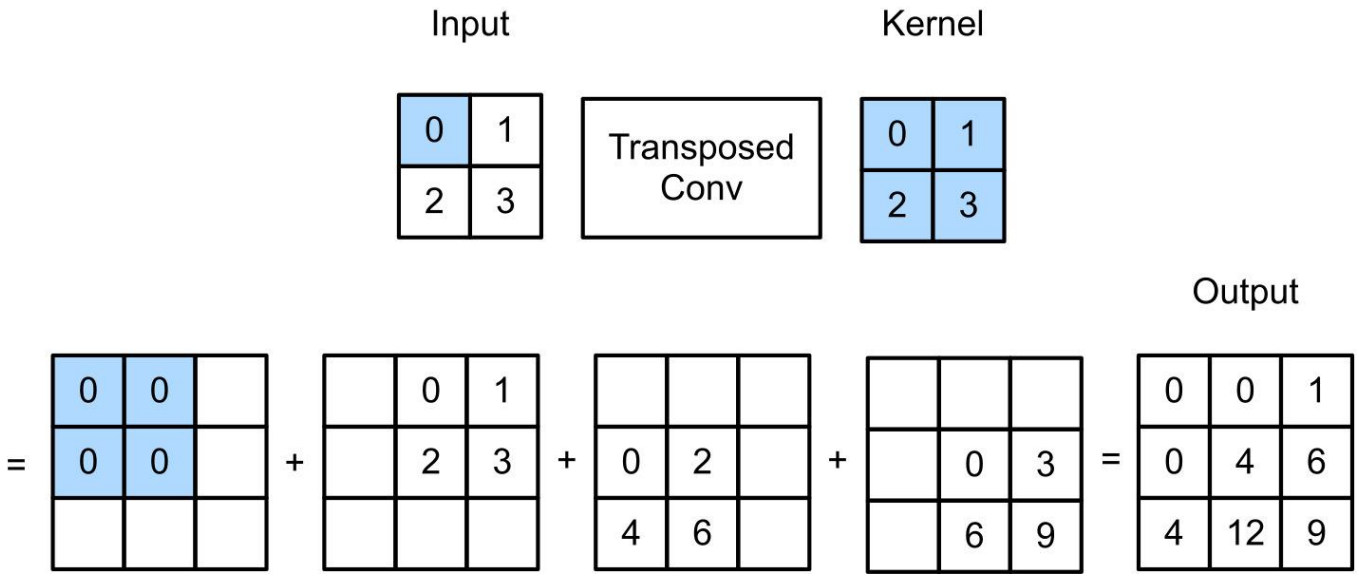
Deconvolution: Brief Concept

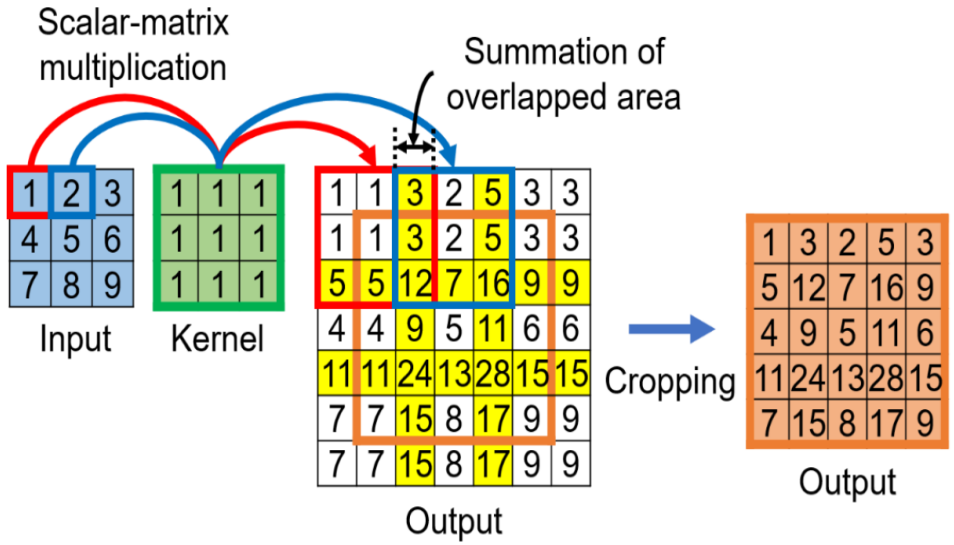
- Unlike convolution layers, which reduces spatial dimension
- Deconvolution Upsample the spatial resolution of input data



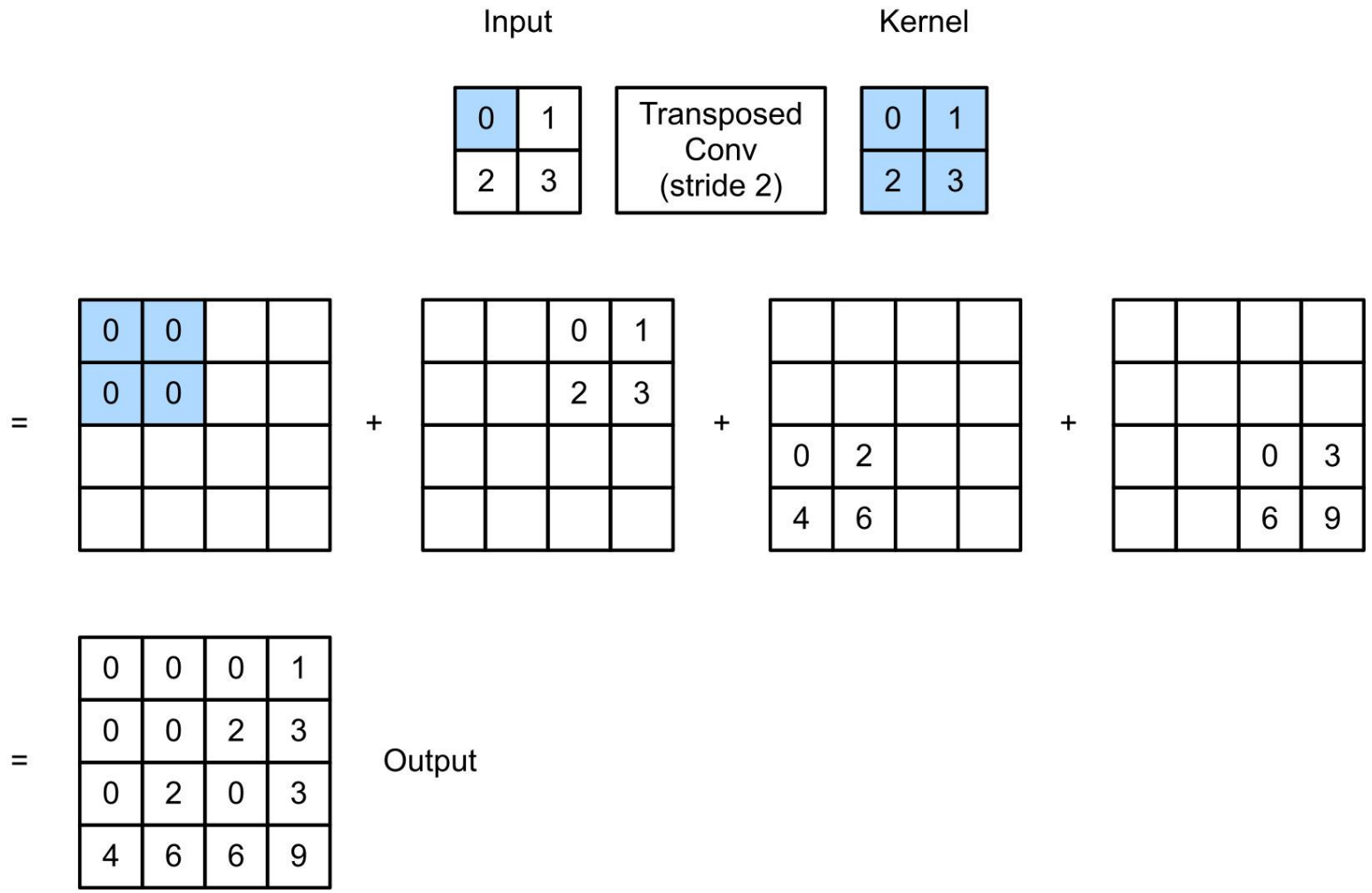
Types of Deconvolution: Padding-free

- No padding is required
- The output is sum of overlapped region
- Output Size= stride(inputsize-1)+filtersize-(2*padding)



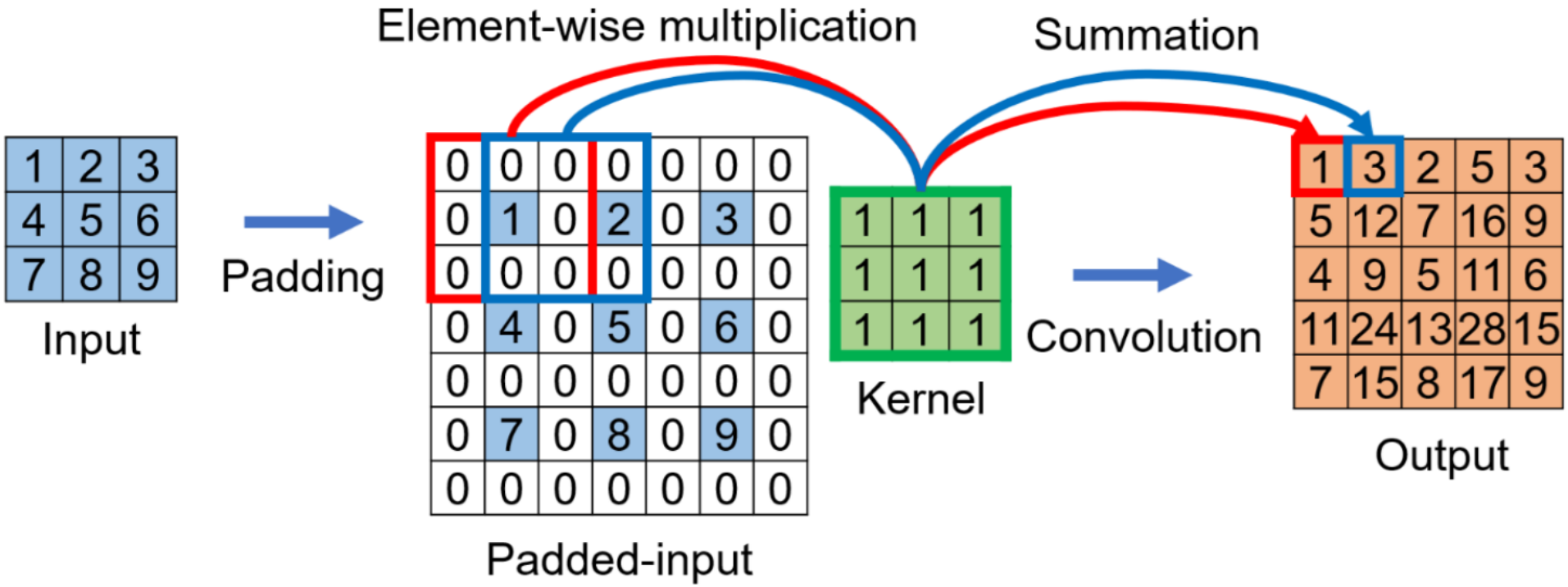


Types of Deconvolution: Padding-free with stride-2



Types of Deconvolution: Zero Padding

- Input image is padded across each element and convolution is applied

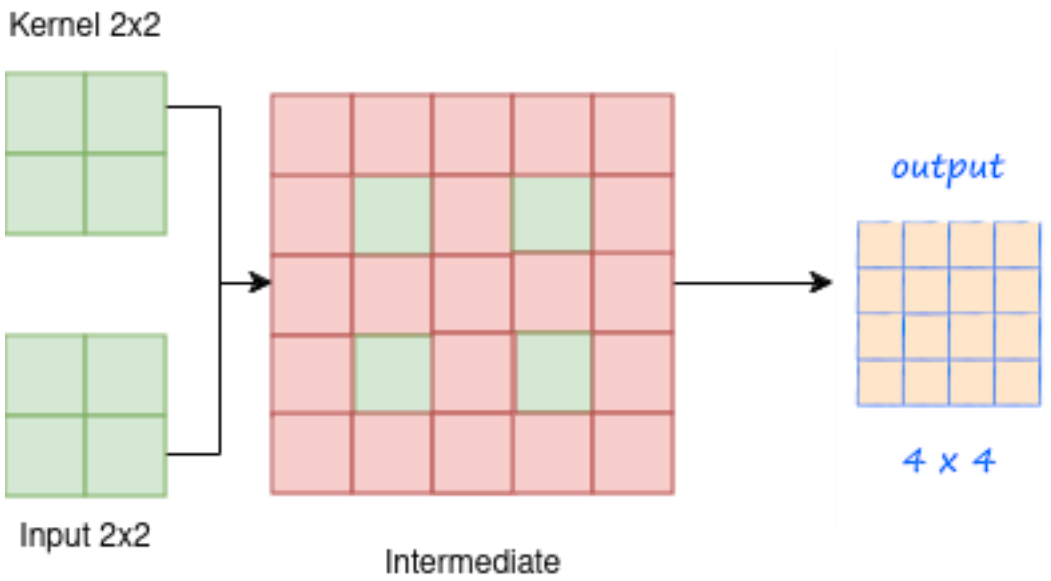


Output size calculation

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

$$Outputsize = \frac{5 - 2 + 0}{1} + 1$$



Thank You