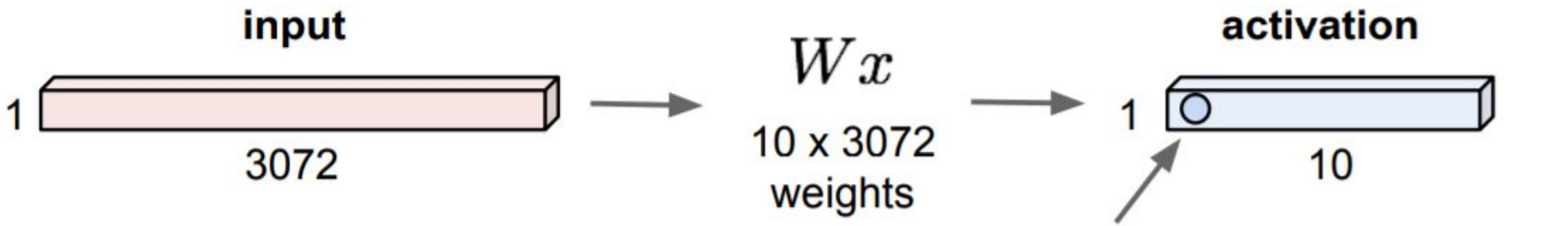
Deep Learning for Computer Vision aka Convolutional Neural Nets

Girish Gore
www.linkedin.com/in/datascientistgirishgore

Images as Plain Neural Network

32x32x3 image -> stretch to 3072 x 1

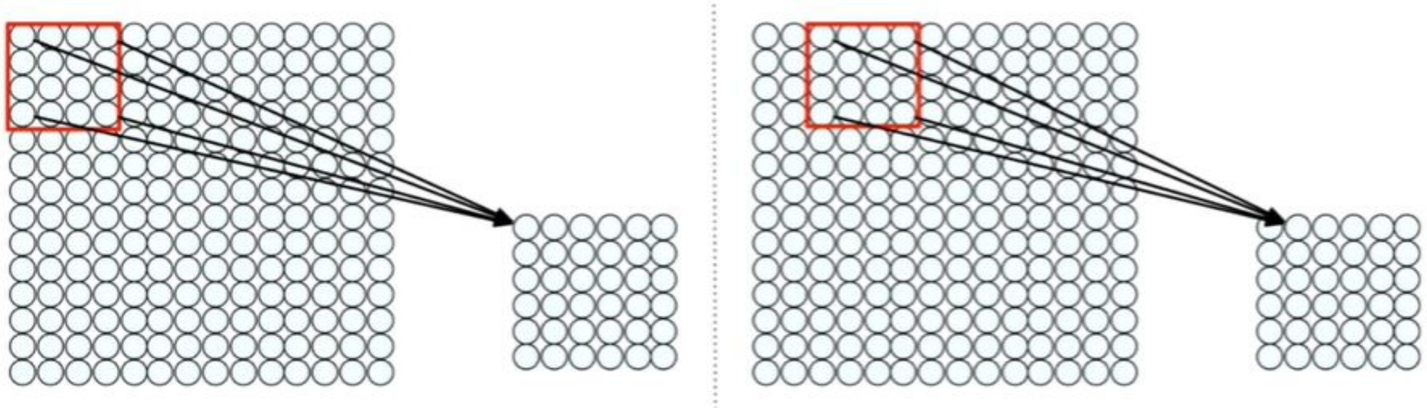


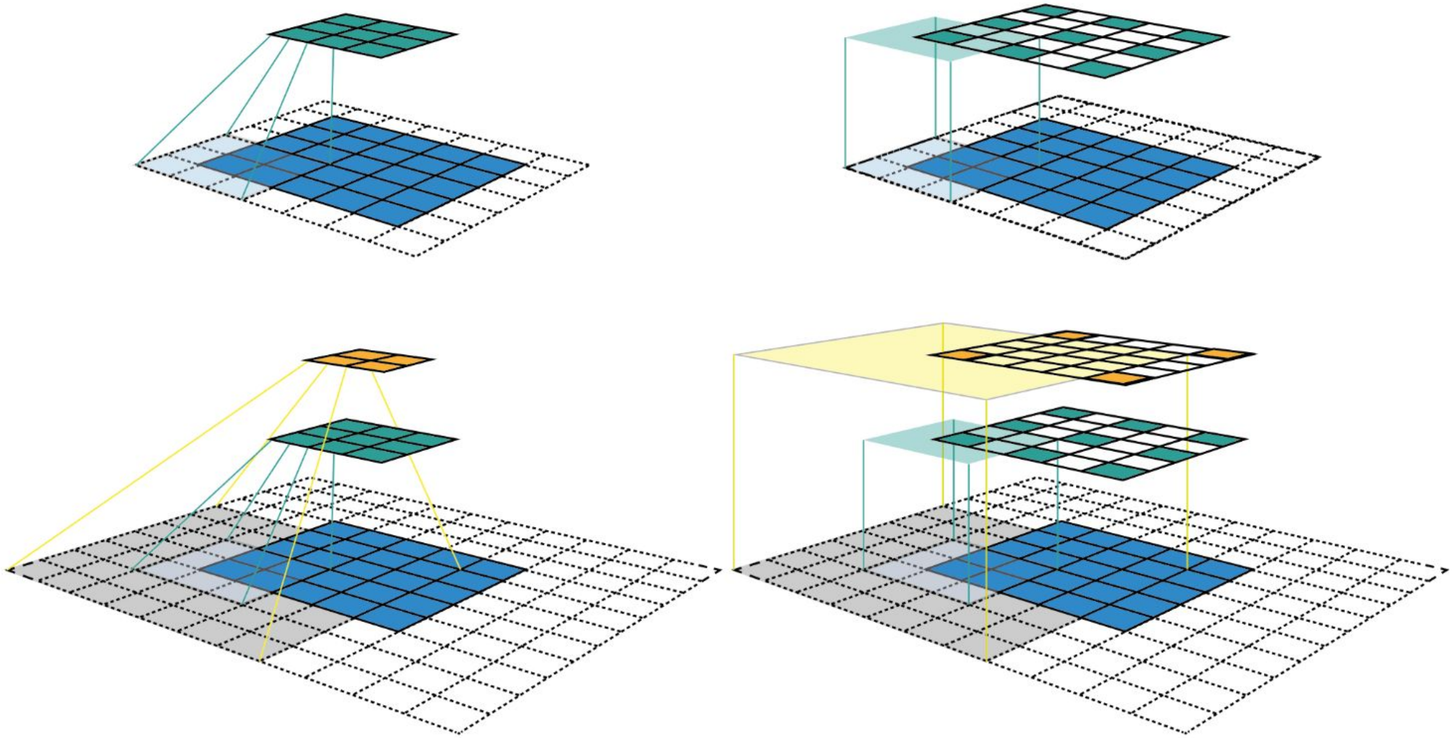
1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

Fully Connected:

- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters!

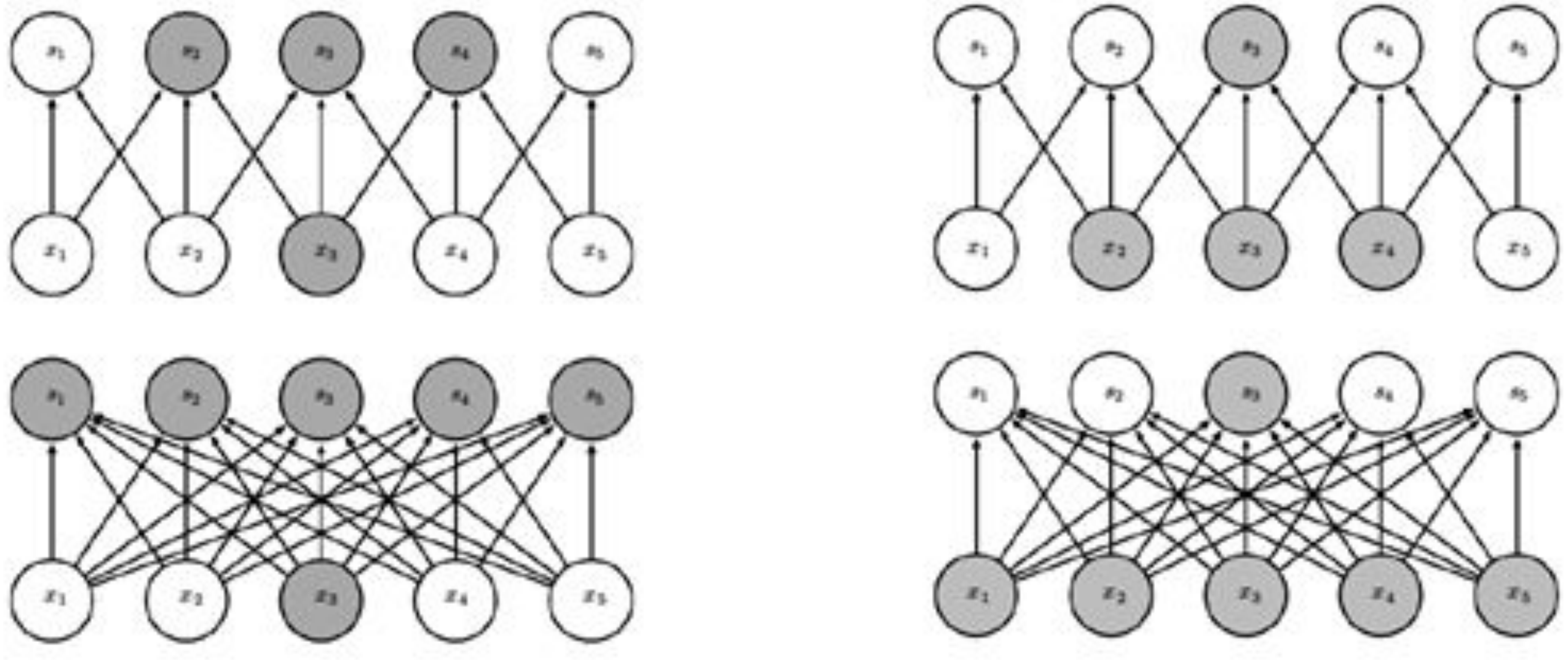
Using Spatial Structure





Receptive field of NN and CNN

Difference in Receptive field between ANN and CNN



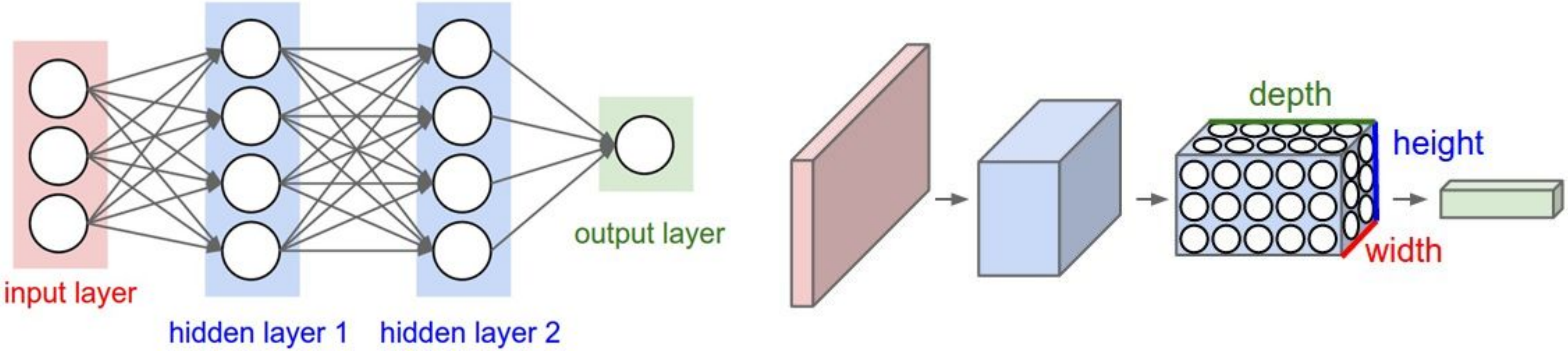
https://www.deeplearningbook.org/

greatlearning
Learning for Life

ANN Vs CNN

1. A powerful way of regularization is *stationarity*, i.e. to treat distant pixels in similar ways while doing any processing, dramatically reduces parameters Also, locality of operations reduces computations

2. Positional information preserved in computed features which utilizes the notion of *spatial neighborhood*



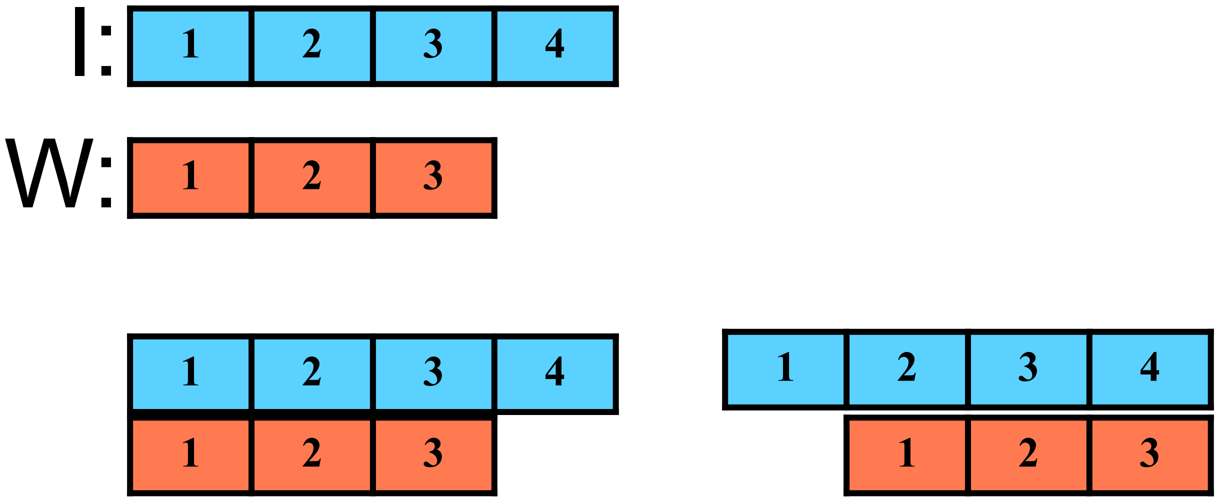
http://cs231n.github.io/

Convolution

Convolution (Discrete 1D)

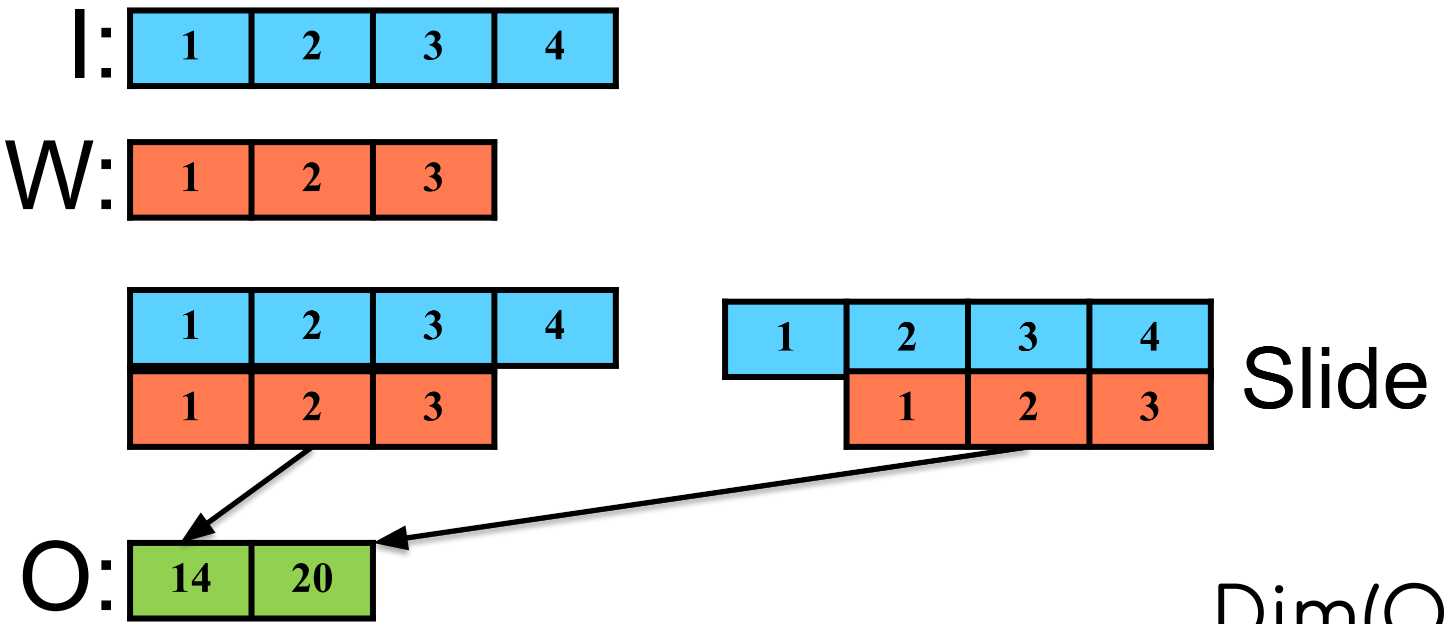
I:	1	2	3	4
W:	1	2	3	

Convolution (Discrete 1D)



Slide

Convolution (Discrete 1D)



Annotation:

I = Input
W = Weights
O = Output

$$\mathrm{Dim}(O) = \mathrm{Dim}(I) - \mathrm{Dim}(W) + 1$$

$$O_1 = I_1 W_1 + I_2 W_2 + I_3 W_3$$

$$O_2 = I_2 W_1 + I_3 W_2 + I_4 W_3$$

Convolution (Discrete 1D) - Padding

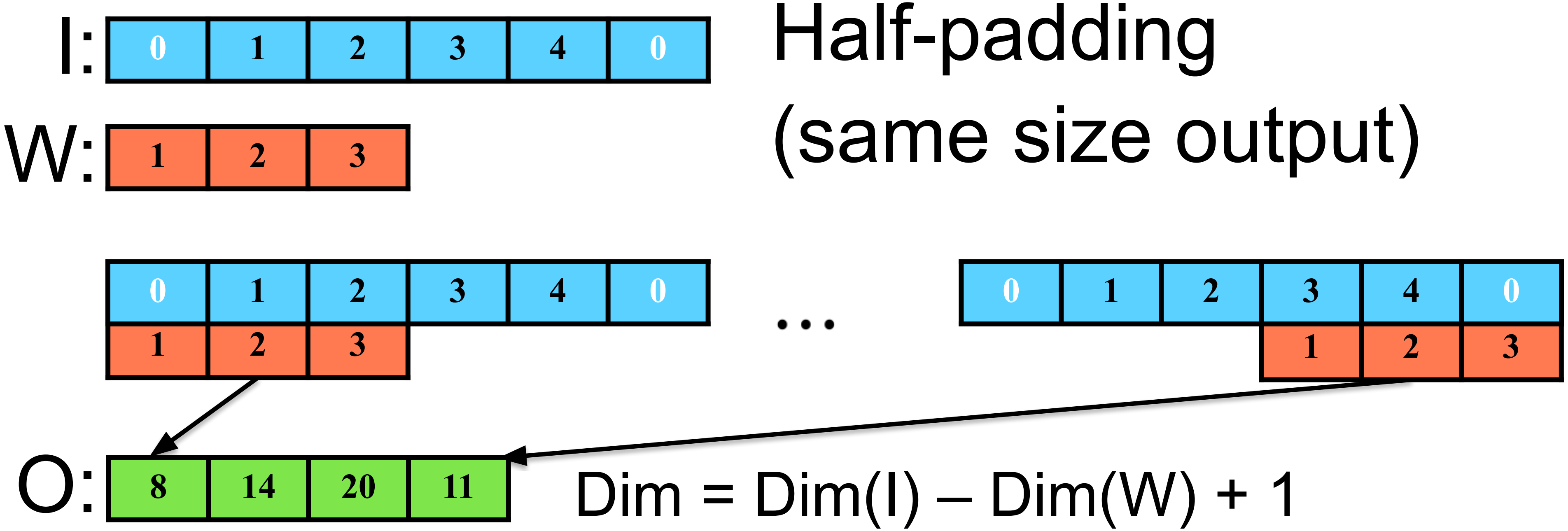
- Sometimes it is convenient to pad the input with zeros on the border of the input volume

- In particular, sometimes it is desirable to exactly preserve the spatial size of the input volume

- The size of this padding is a third hyperparameter. Padding provides control of the output volume spatial size.

Convolution (Discrete 1D) - Padding

Half-padding
(same size output)

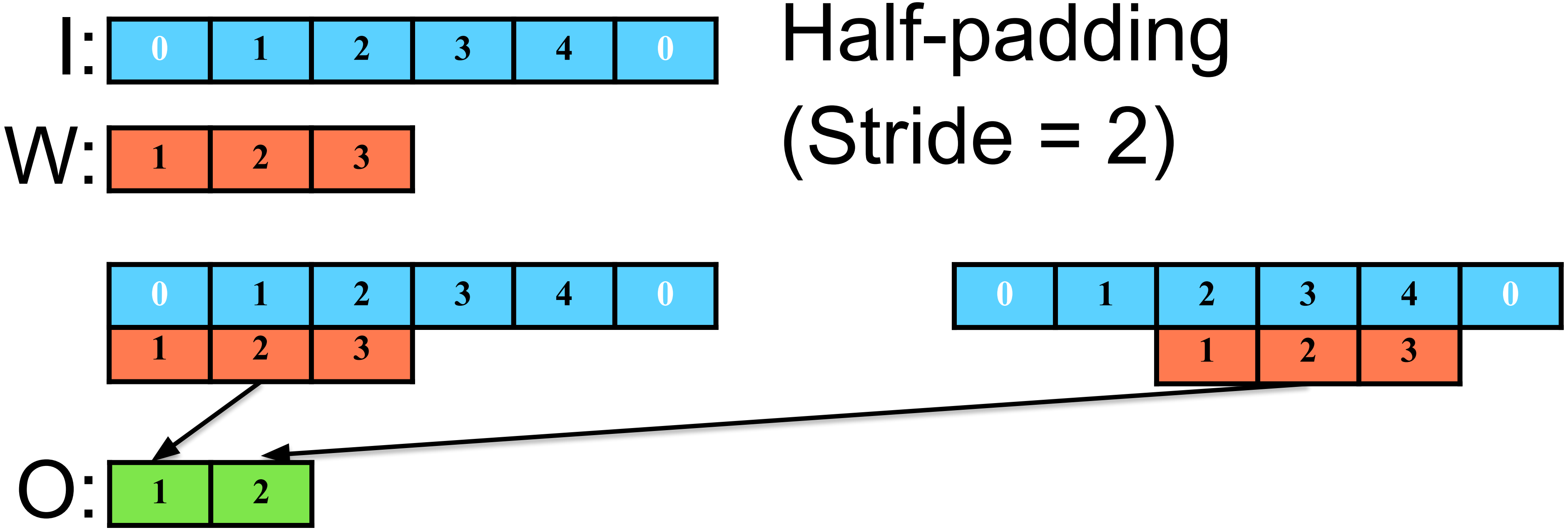


Dim = Dim(I) – Dim(W) + 1

Notice: The dimension of the output matrix remains the same.

Convolution (Discrete 1D) - Stride

Half-padding
(Stride = 2)

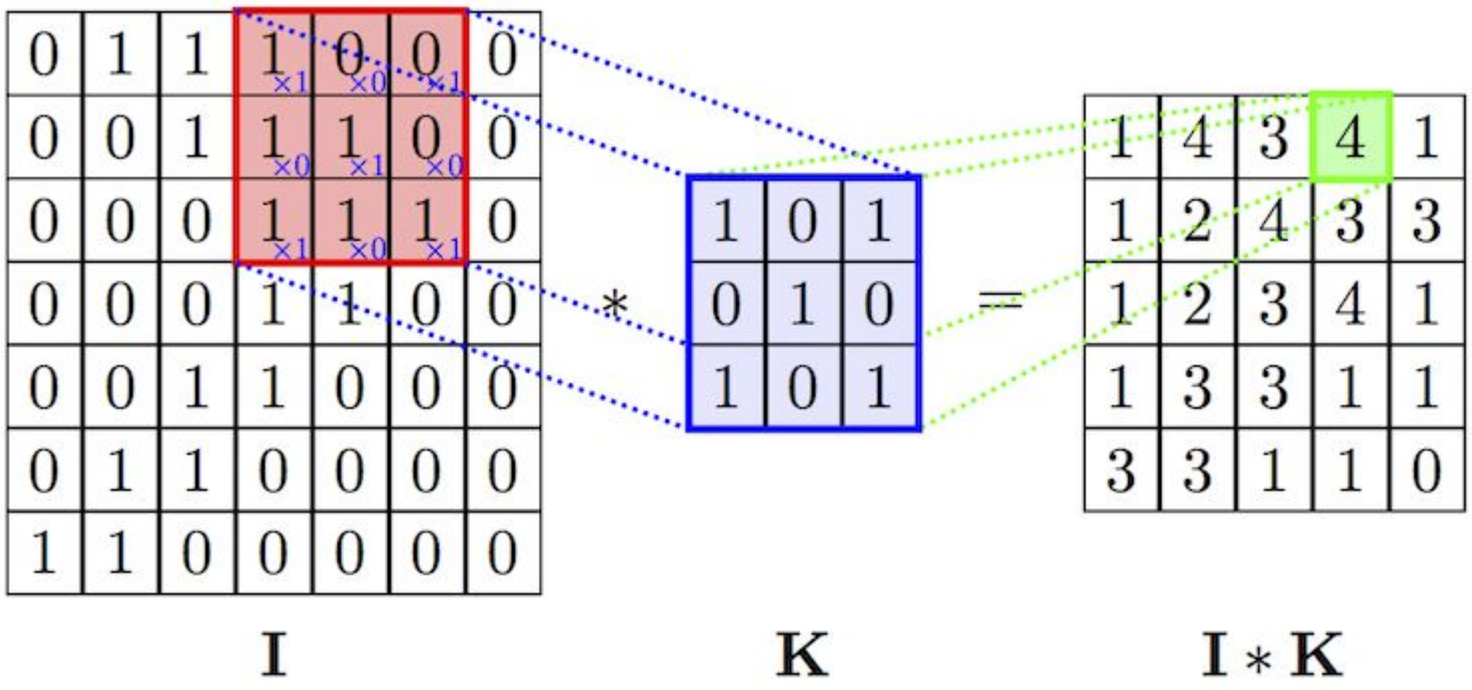


Convolution - 2D

$$
\underbrace{\begin{array}{|c|c|c|c|c|c|c|}
\hline
0 & 1 & 1 & 1 & 0 & 0 & 0 \\ \hline
0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline
0 & 0 & 0 & 1 & 1 & 1 & 0 \\ \hline
0 & 0 & 0 & 1 & 1 & 0 & 0 \\ \hline
0 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hline
0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline
1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline
\end{array}}_{\mathbf{I}}
\ast
\underbrace{\begin{array}{|c|c|c|}
\hline
1 & 0 & 1 \\ \hline
0 & 1 & 0 \\ \hline
1 & 0 & 1 \\ \hline
\end{array}}_{\mathbf{K}}
=
\underbrace{\begin{array}{|c|c|c|c|c|}
\hline
1 & 4 & 3 & 4 & 1 \\ \hline
1 & 2 & 4 & 3 & 3 \\ \hline
1 & 2 & \blacksquare & 4 & 1 \\ \hline
1 & 3 & 3 & 1 & \blacksquare \\ \hline
3 & 3 & 1 & 1 & 0 \\ \hline
\end{array}}_{\mathbf{I} \ast \mathbf{K}}
$$

Can you work out Hidden values ?

Convolution - 2D



Exercise 1

- Let’s take an input image of size 4 * 4 and a kernel of size 2 *2 with no zero padding and stride = 1?
- What is the size of the output image?

Exercise 1 - Solution

- Let's take an input image of size 4 * 4 and a kernel of size 2 *2 with no zero padding and stride = 1?
- What is the size of the output image?

- Ans: 3 * 3

Exercise 2

- Let’s take an input image of size 5 * 5 and a kernel of size 3 * 3 with zero padding p=2 and stride = 2. What is the size of the output?

Exercise 2 - Solution

- Let's take an input image of size 5 * 5 and a kernel of size 3 * 3 with zero padding p=2 and stride = 2. What is the size of the output?

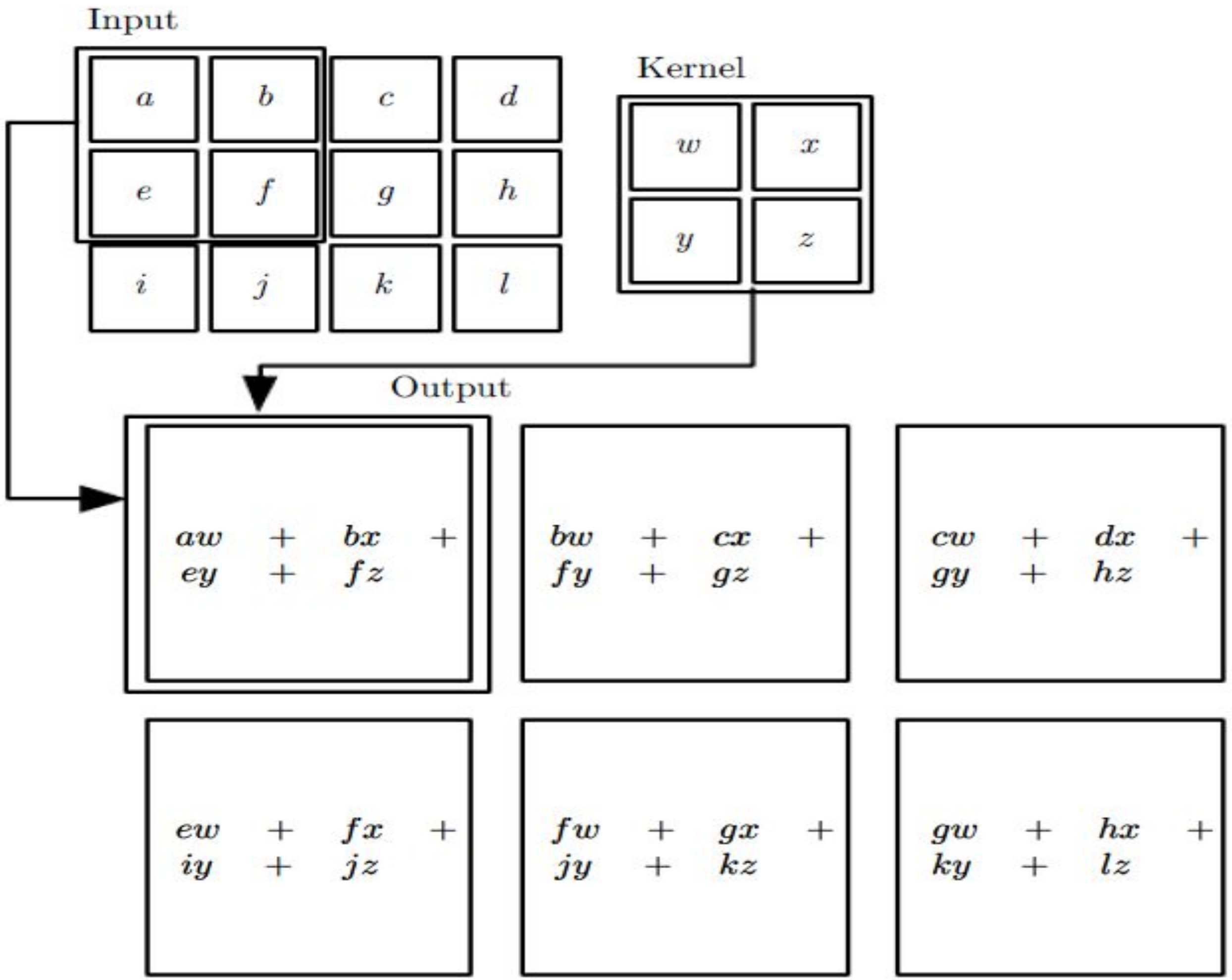
- Ans: 4 * 4

- Formula for Dimension of Output Image:
- $(W - F + 2P)/S + 1$

Where

- W = size of the image
- F = size of filter
- P = padding
- S = stride

Spatial Dimension - Convolution - 2D



Convolution - 2D

$$I \bigotimes W = \sum_{k} \sum_{l} I(k,l) W(i+k, j+l)$$

I = Image

W = Kernel

$$\underbrace{\begin{bmatrix} i_1 & i_2 & i_3 \\ i_4 & i_5 & i_6 \\ i_7 & i_8 & i_9 \end{bmatrix}}_{\boldsymbol{I}} \bigotimes \underbrace{\begin{bmatrix} W_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix}}_{\boldsymbol{W}} \longrightarrow$$

$$I * W = i_1 w_1 + i_2 w_2 + i_3 w_3 + i_4 w_4 + i_5 w_5 + i_6 w_6 + i_7 w_7 + i_8 w_8 + i_9 w_9$$

Spatial Dimension - Convolution - 3D

Image I = 2 x 4 x 4

I[1, :, :]

1	-2	2	2
2	1	3	-2
-2	3	-3	1
-1	2	-4	2

I[2, :, :]

3	0	0	0
-2	-2	1	-1
2	-1	3	1
5	-2	0	1

Weights W = 2 x 2 x 2 x 2
(nOutputPlane x nInputPlane x kH x kW)

W[1, 1, :, :]

1	-2
-2	1

W[2, 1, :, :]

3	1
2	2

W[1, 2, :, :]

1	0
0	1

W[2, 2, :, :]

0	0
0	4

0.1
0.2

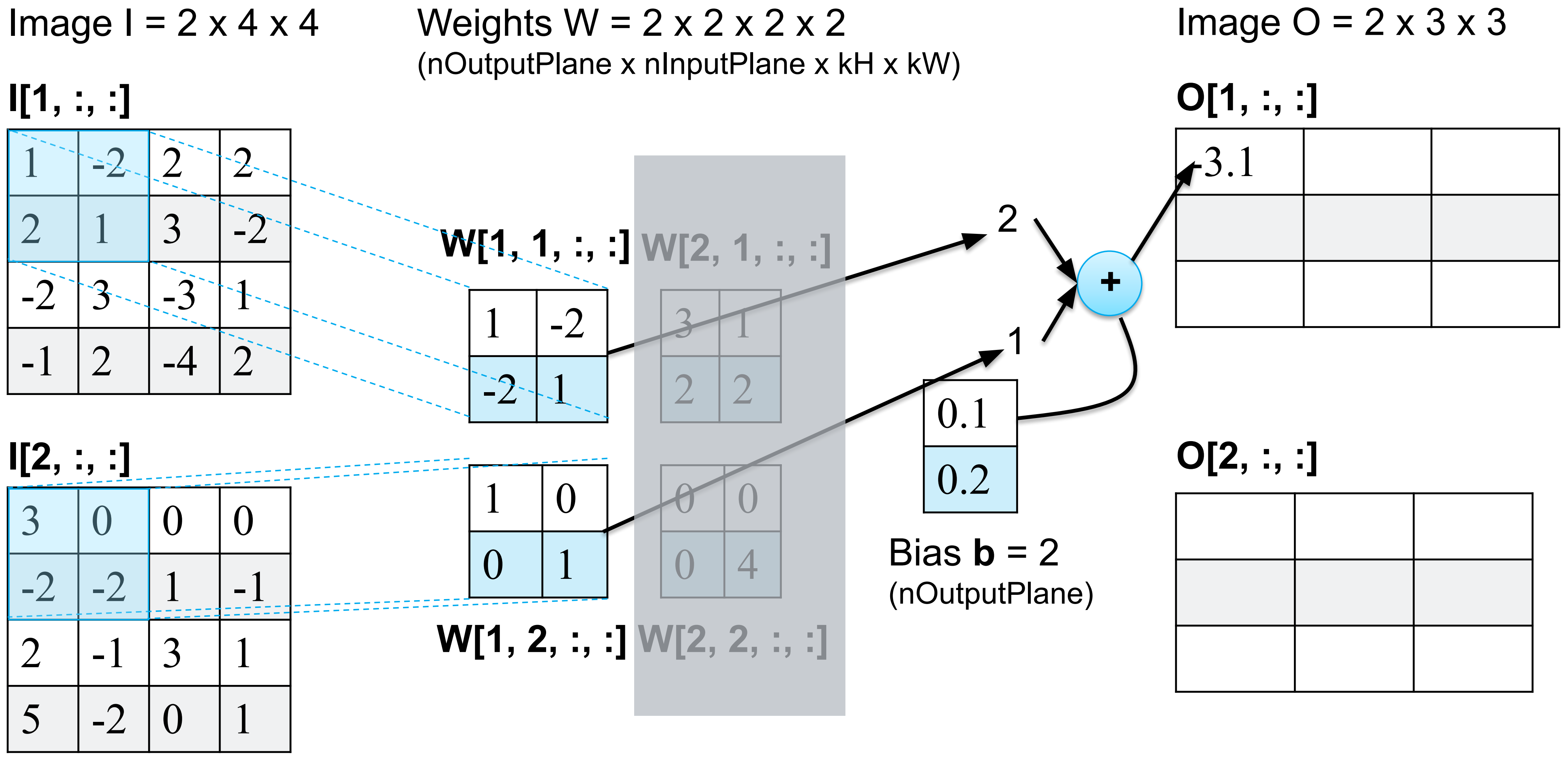
Bias **b** = 2
(nOutputPlane)

Image O = 2 x 3 x 3

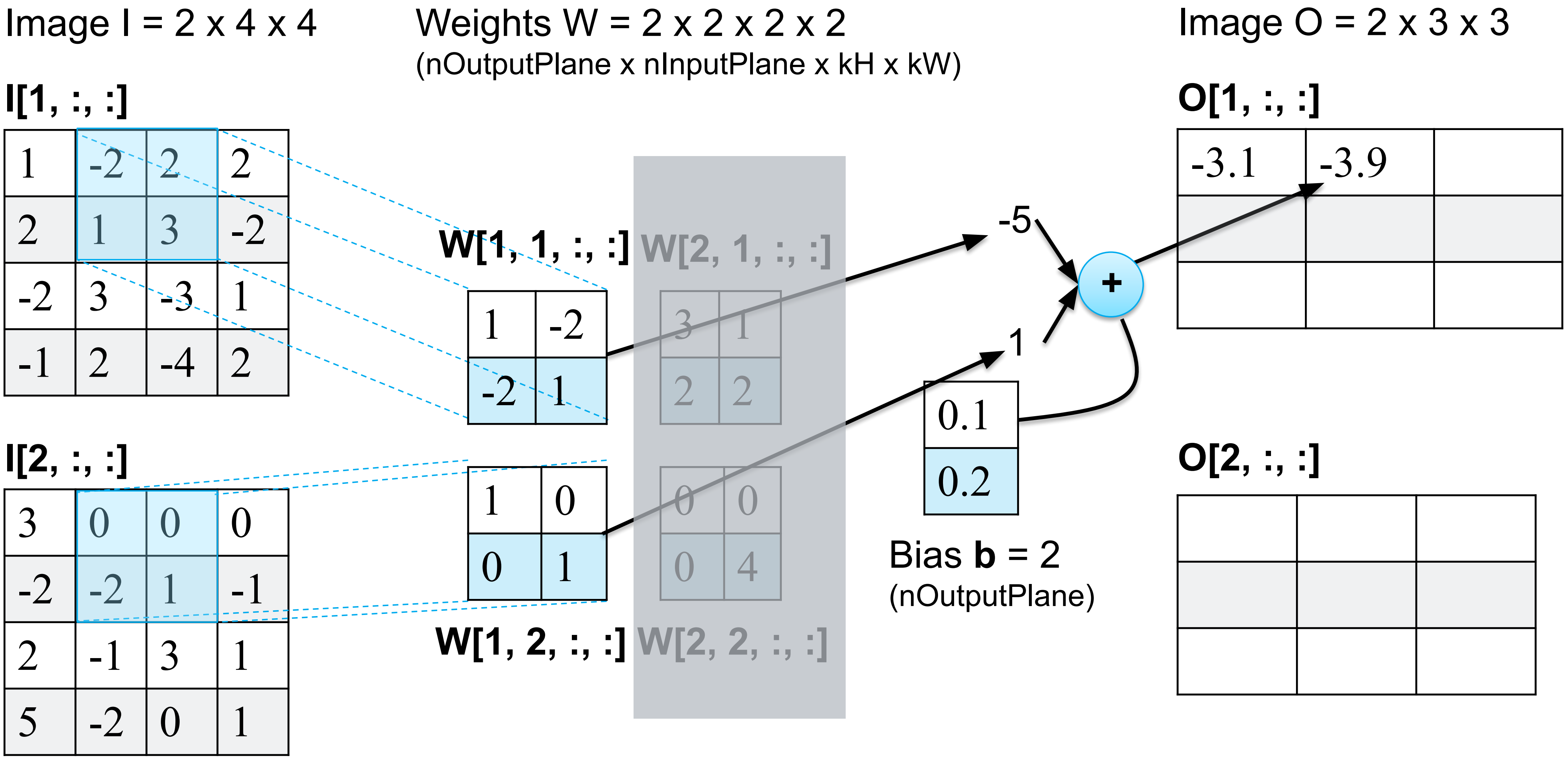
O[1, :, :]

O[2, :, :]

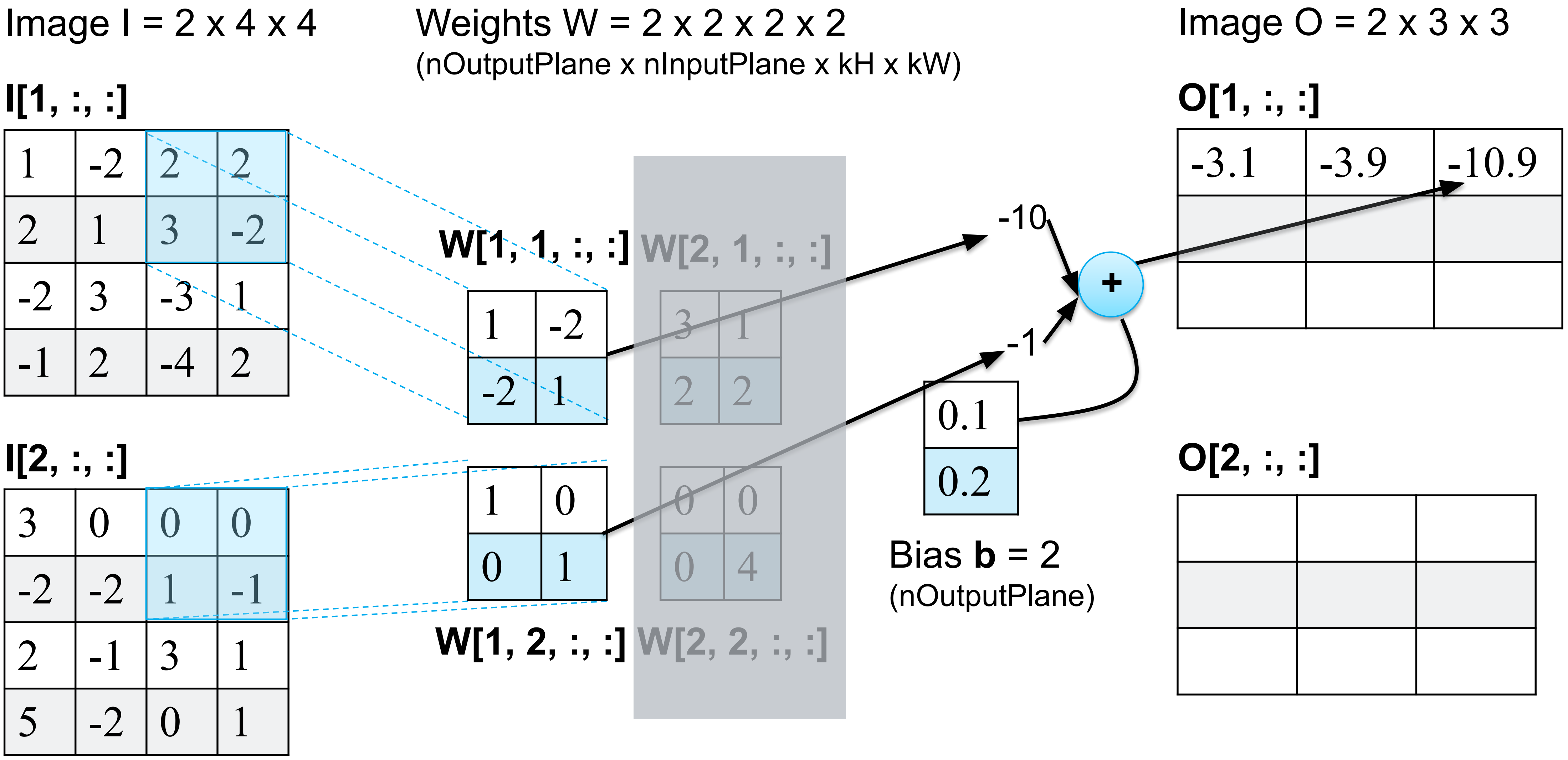
Spatial Dimension - Convolution - 3D



Spatial Dimension - Convolution - 3D



Spatial Dimension - Convolution - 3D



Spatial Dimension - Convolution

greatlearning
Learning for Life

Image I = 2 x 4 x 4

I[1, :, :]

1	-2	2	2
2	1	3	-2
-2	3	-3	1
-1	2	-4	2

I[2, :, :]

3	0	0	0
-2	-2	1	-1
2	-1	3	1
5	-2	0	1

Weights W = 2 x 2 x 2 x 2
(nOutputPlane x nInputPlane x kH x kW)

W[1, 1, :, :]

1	-2
-2	1

W[2, 1, :, :]

3	1
2	2

W[1, 2, :, :]

1	0
0	1

W[2, 2, :, :]

0	0
0	4

-12

4

+

0.1
0.2

Bias **b** = 2
(nOutputPlane)

Image O = 2 x 3 x 3

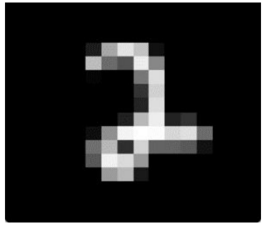
O[1, :, :]

-3.1	-3.9	-10.9
4.1	-12.9	16.1
-3.9	.1	9.1

O[2, :, :]

-0.8	8.2	6.2
5.2	18.2	7.2
-8.8	2.2	-7.8

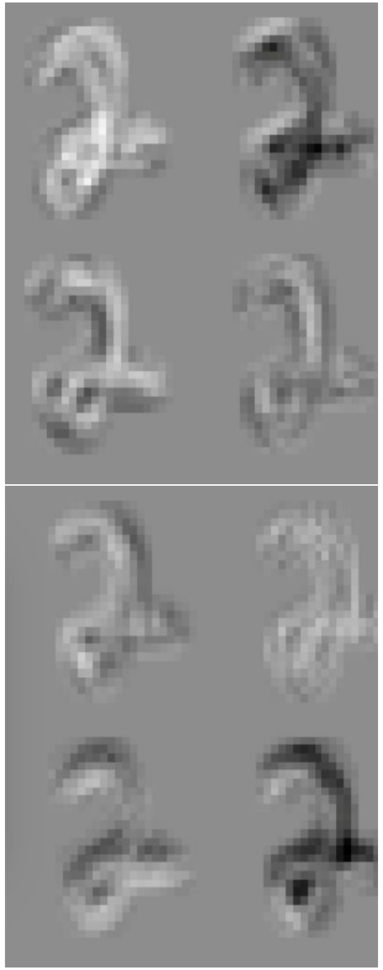
Convolution - one more view point



16 * 16 Image

$n \in \mathbb{R}^{16\times16} \rightarrow$ Convolution $\rightarrow m \in \mathbb{R}^{8\times12\times12}$

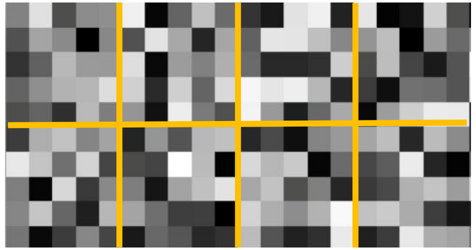
8 Output Feature maps of size 12 * 12



1 represents Input channel - Black & White in this case

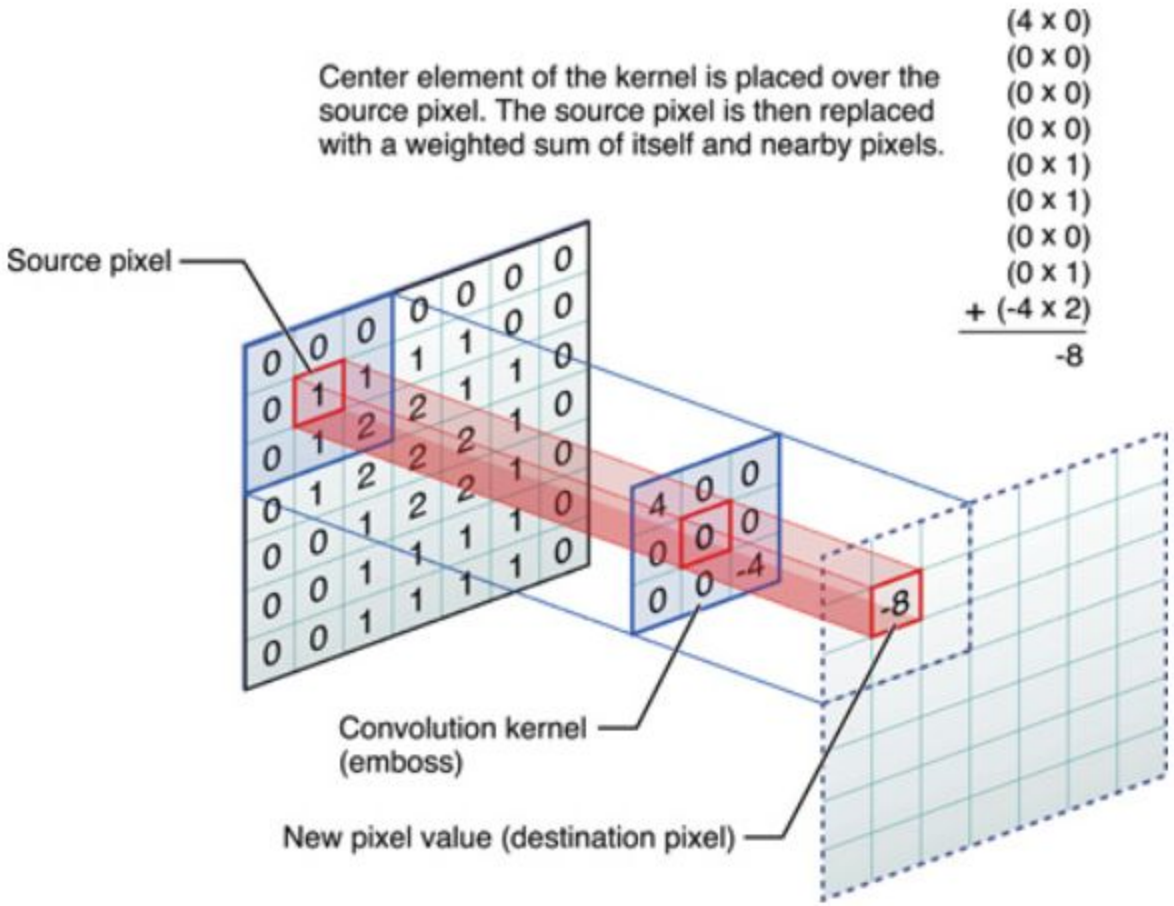
$W \in \mathbb{R}^{1\times8\times5\times5}$

$=$

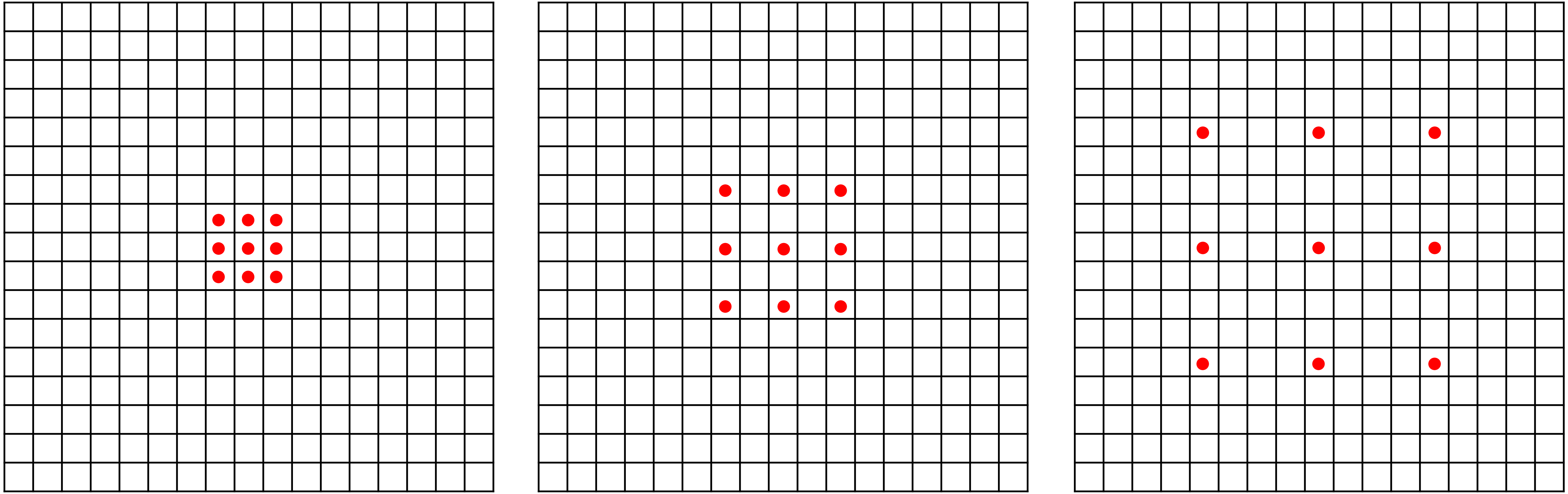


8 Filters of size 5 * 5

Convolution Single Element View



Aside: Dilated Convolution

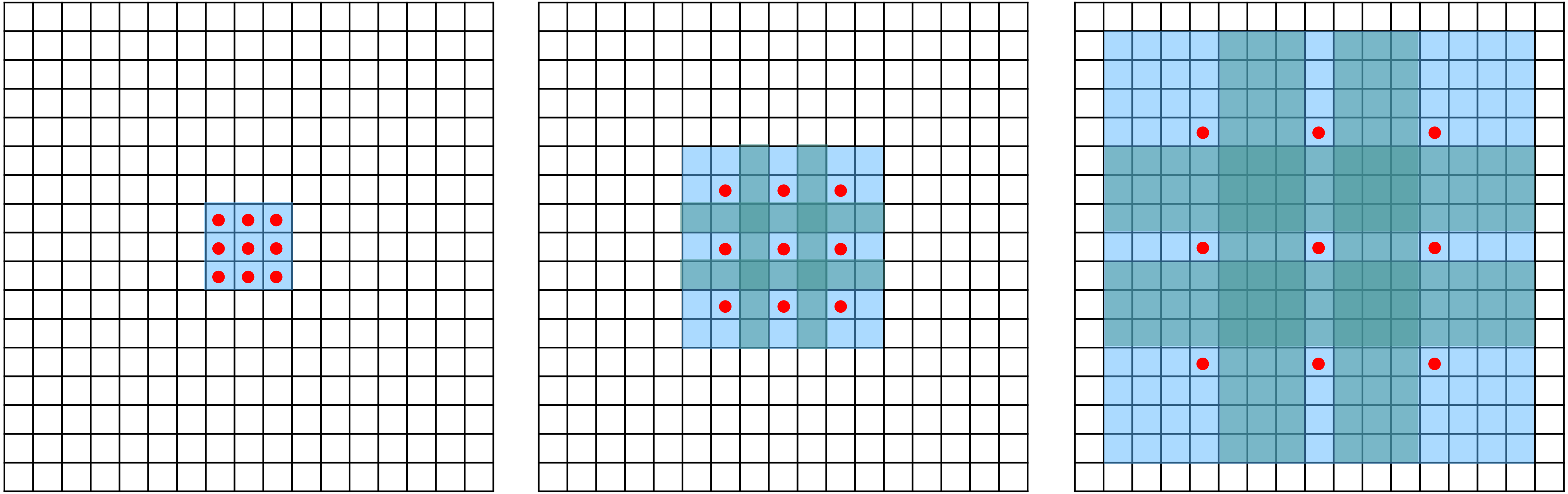


(a) (b) (c)

Aside: Dilated Convolution

greatlearning
Learning for Life

[nn.SpatialDilatedConvolution]



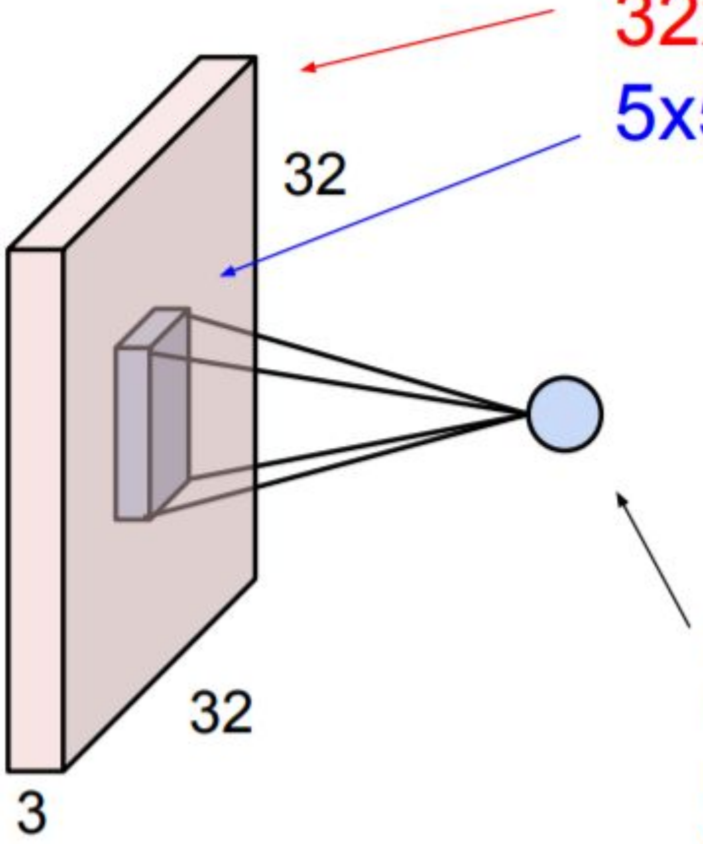
(a) (b) (c)

Source: Multi-Scale Context Aggregation by Dilated Convolutions Fisher Yu, Vladlen Koltun`

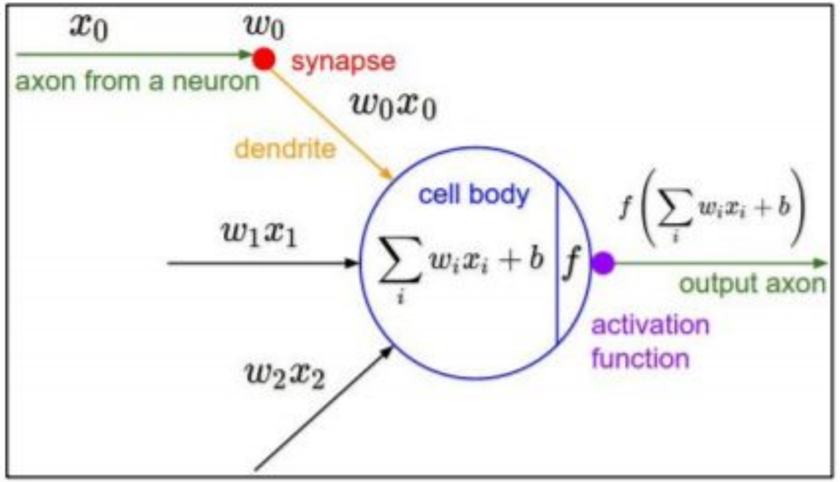
CONV Layer View

32x32x3 image

5x5x3 filter

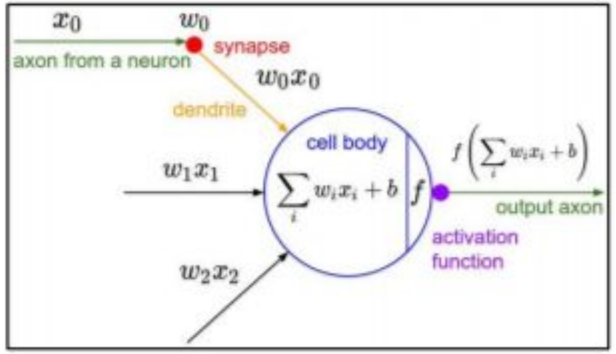


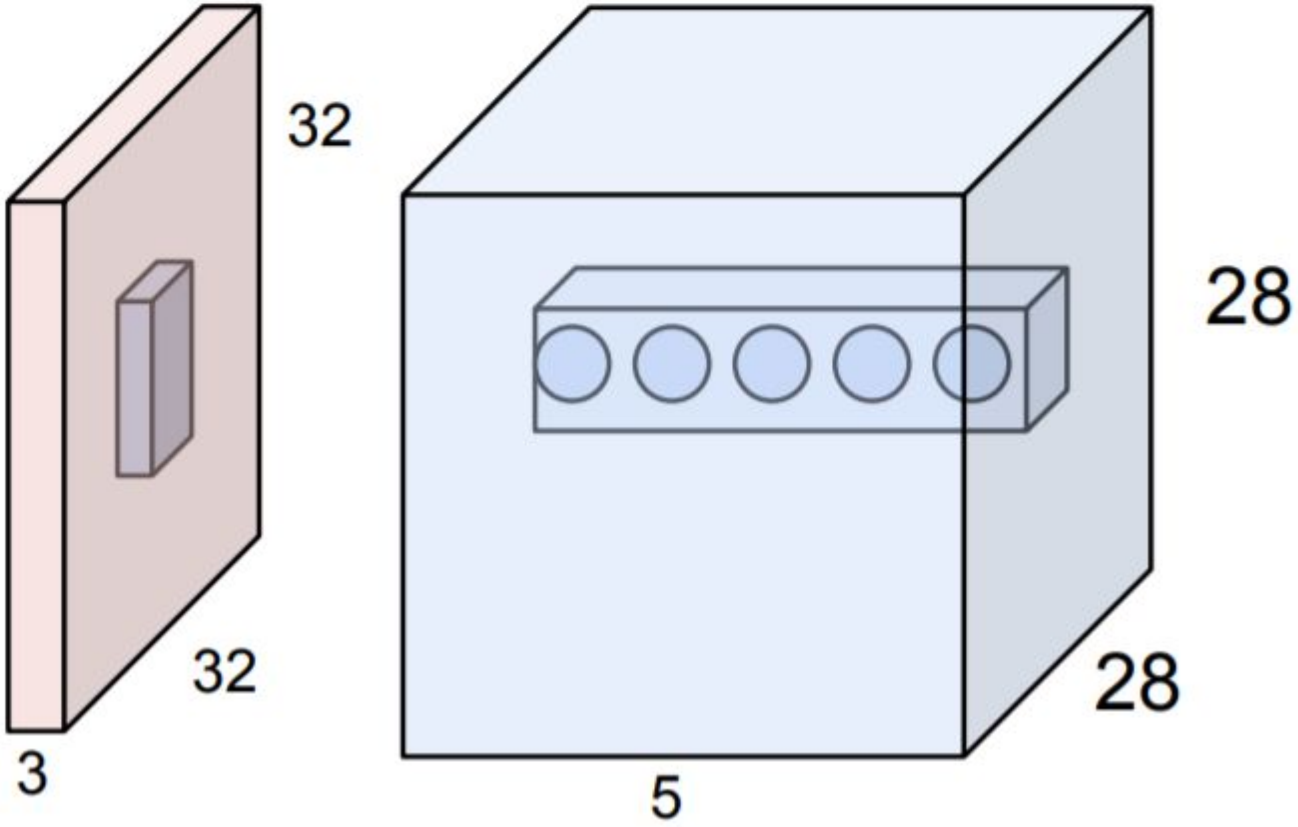
1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)



It's just a neuron with local connectivity...

CONV Layer View





E.g. with 5 filters,
CONV layer consists of
neurons arranged in a 3D grid
(28x28x5)

There will be 5 different
neurons all looking at the same
region in the input volume

Summarizing CONV Operation

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

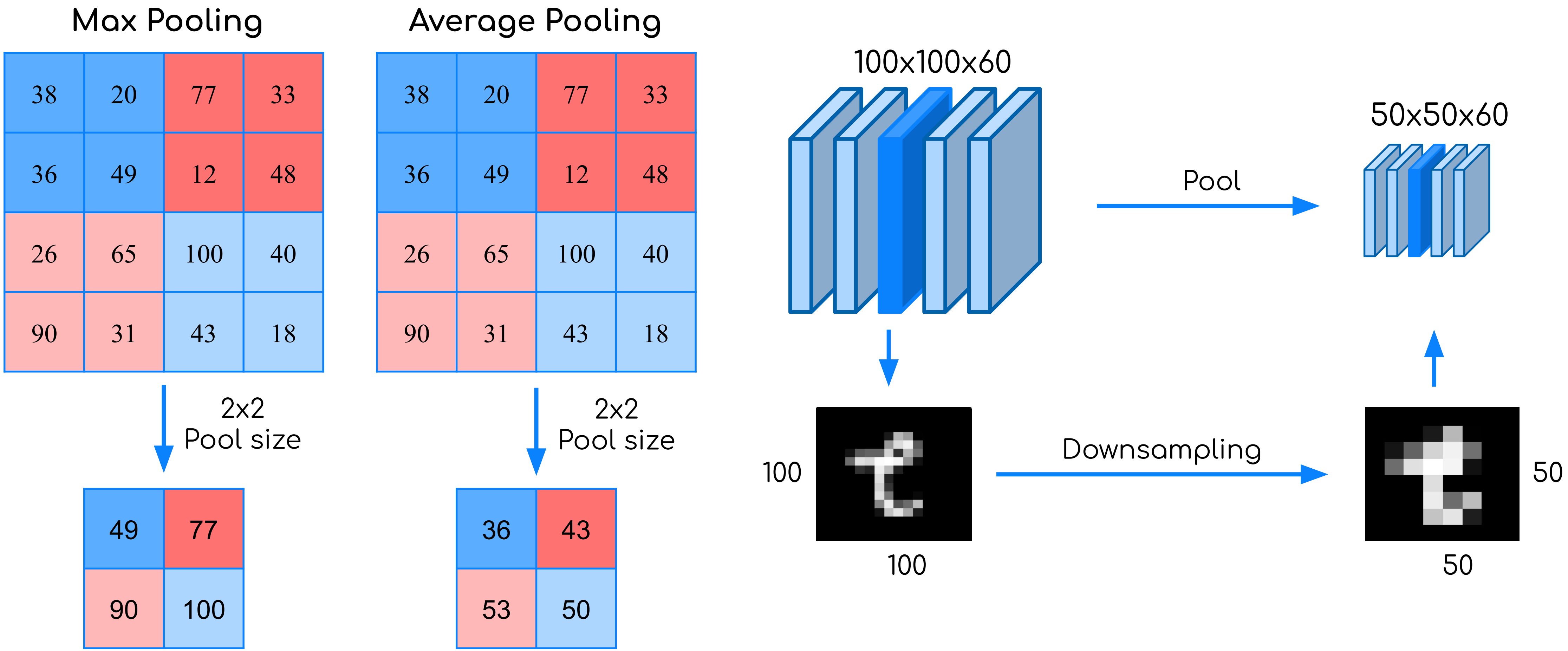
Pooling

Why Pooling?

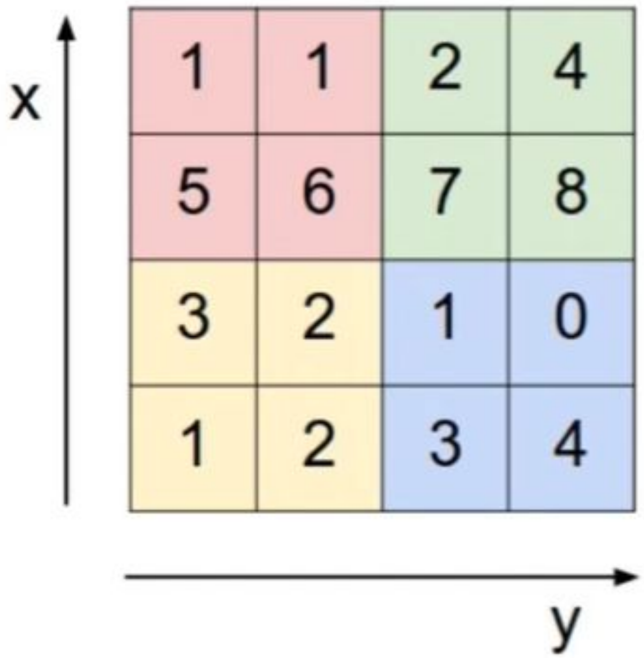
- Also called a sub sampling
- Capture long range interactions (receptive field increases)
- Reduce computational burden for subsequent layers
- Translational Invariance

Pooling in action

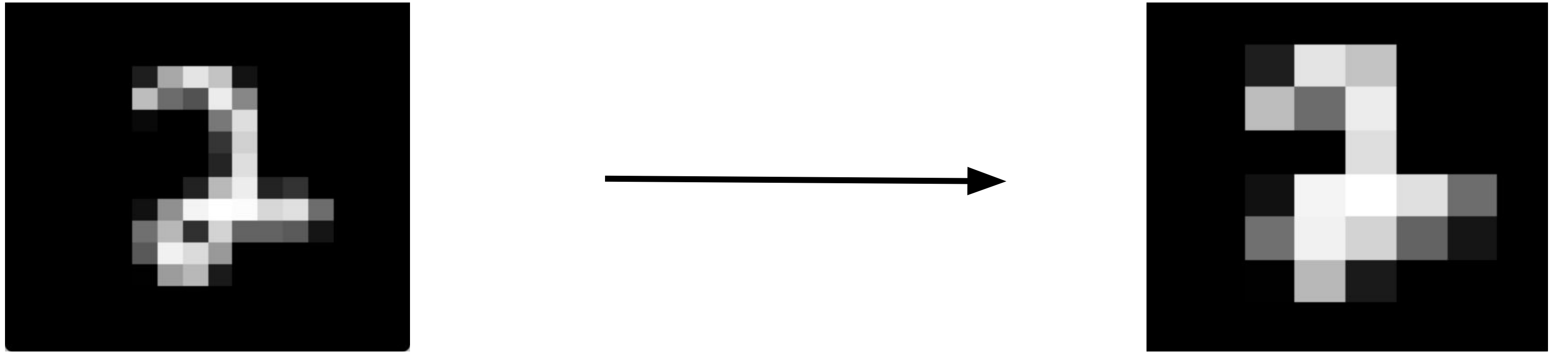
- Pooling doesn't change the depth. It only affects the length and the width of the input.
- It introduces no parameters.



Pooling (Max Pooling)



1) Reduced dimensionality
2) Spatial invariance



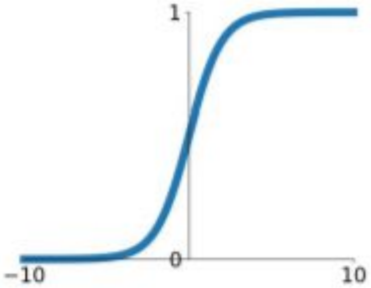
Summarizing POOLING Operation

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Activation Functions

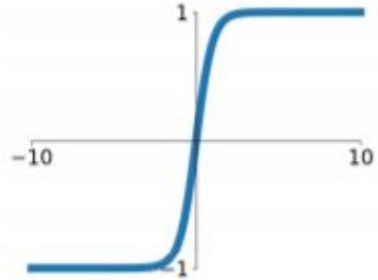
Sigmoid

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



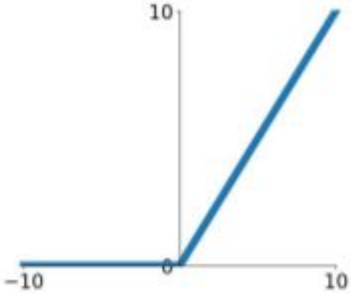
tanh

$\tanh(x)$



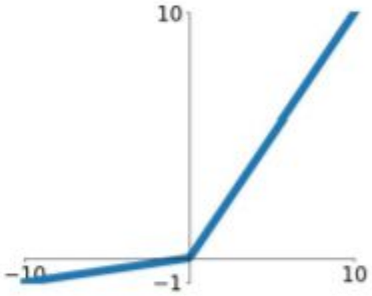
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

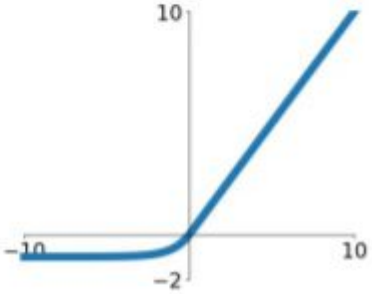


Maxout

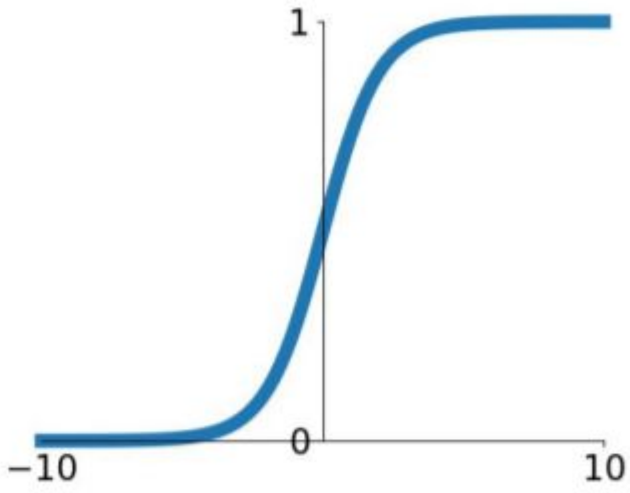
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Sigmoid Activation



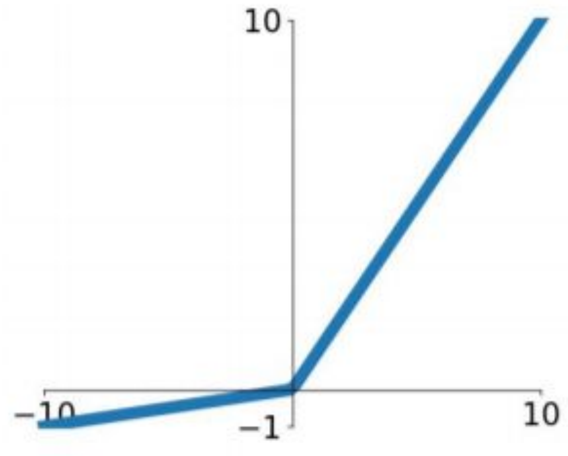
Sigmoid

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

3 problems:

1. Saturated neurons "kill" the gradients
2. Sigmoid outputs are not zero-centered
3. exp() is a bit compute expensive

Relu Activation



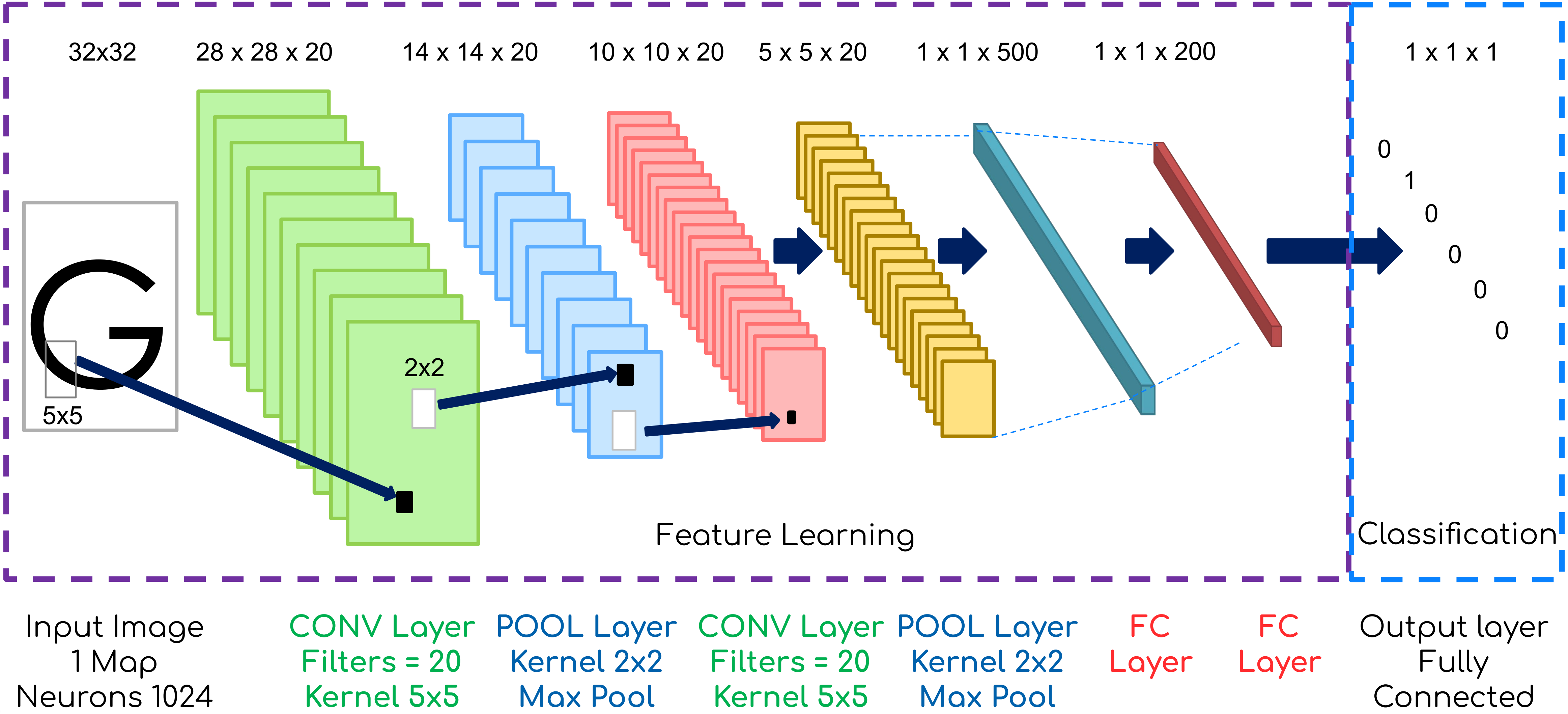
- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not "die".**

Leaky ReLU

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

Lets decode a Fully Convolutional Network

greatlearning
Learning for Life



Translational Invariance

Convolution is not naturally equivariant to some other transformations, such as changes in the scale or rotation of an image. Other mechanisms are necessary for handling these kinds of transformations.

It is the max pooling layer that introduces such invariants

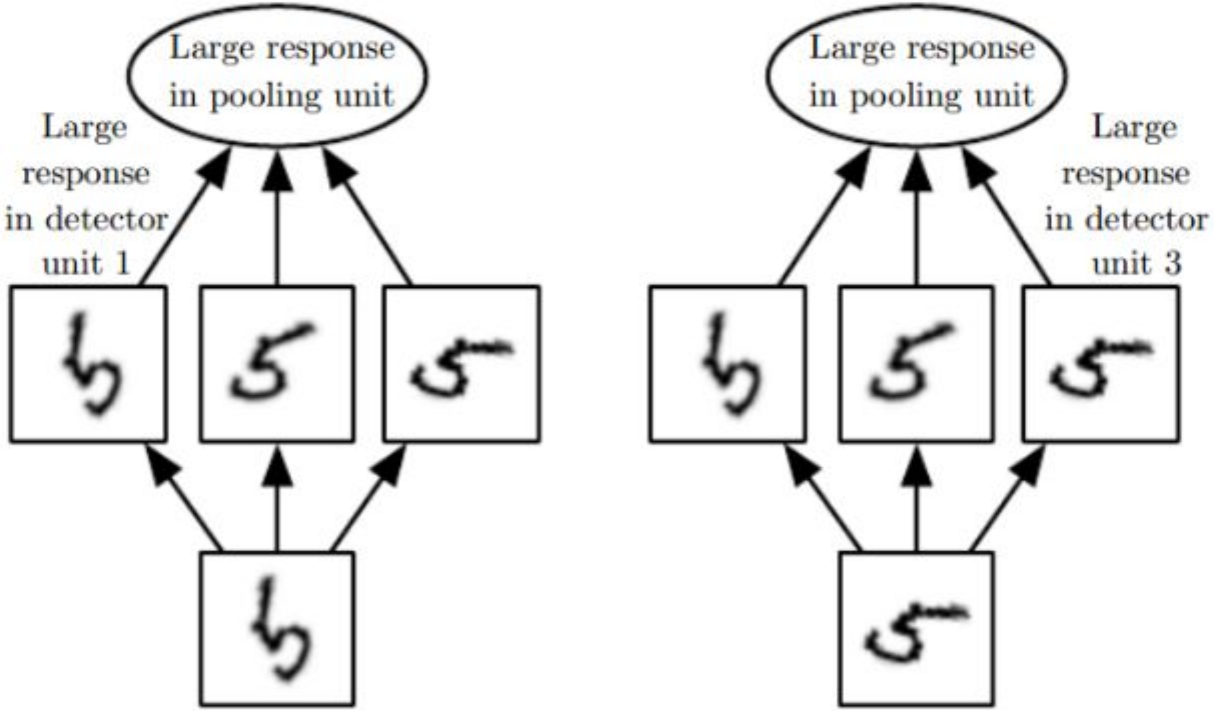
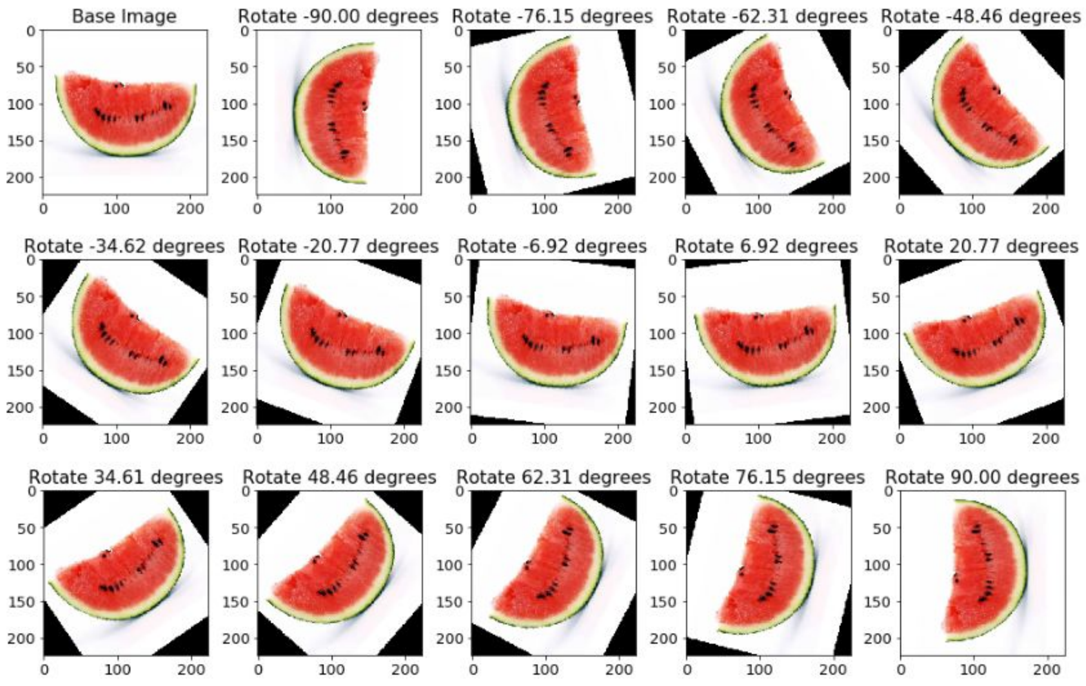


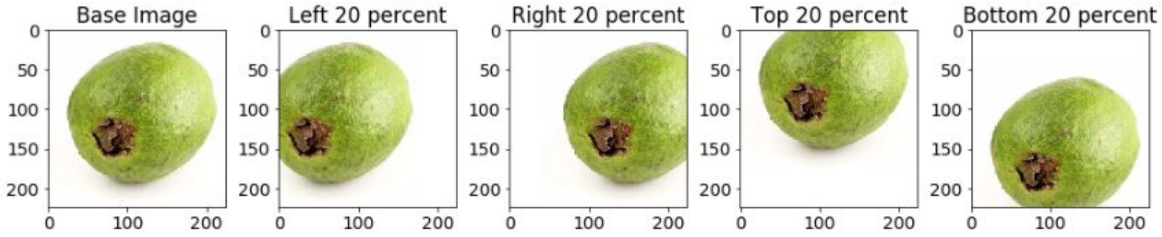
Figure 9.9: *Example of learned invariances:* A pooling unit that pools over multiple features that are learned with separate parameters can learn to be invariant to transformations of the input. Here we show how a set of three learned filters and a max pooling unit can learn to become invariant to rotation. All three filters are intended to detect a hand-written 5. Each filter attempts to match a slightly different orientation of the 5. When a 5 appears in

Data Augmentation

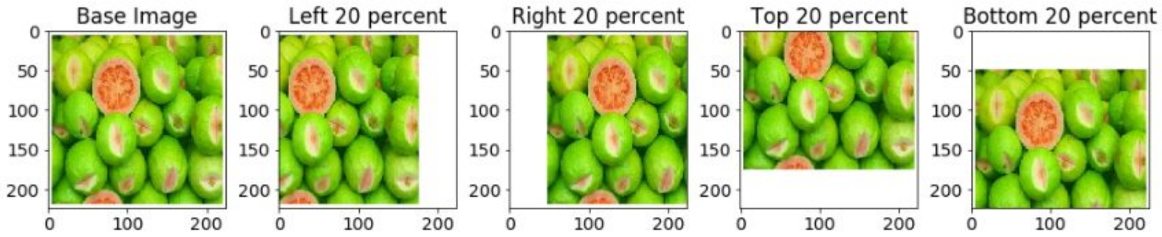


Images can be augmented by

- Translation (shifting)
- Scale
- Rotating
- Mirroring / Flipping
- Colour / Light Variations

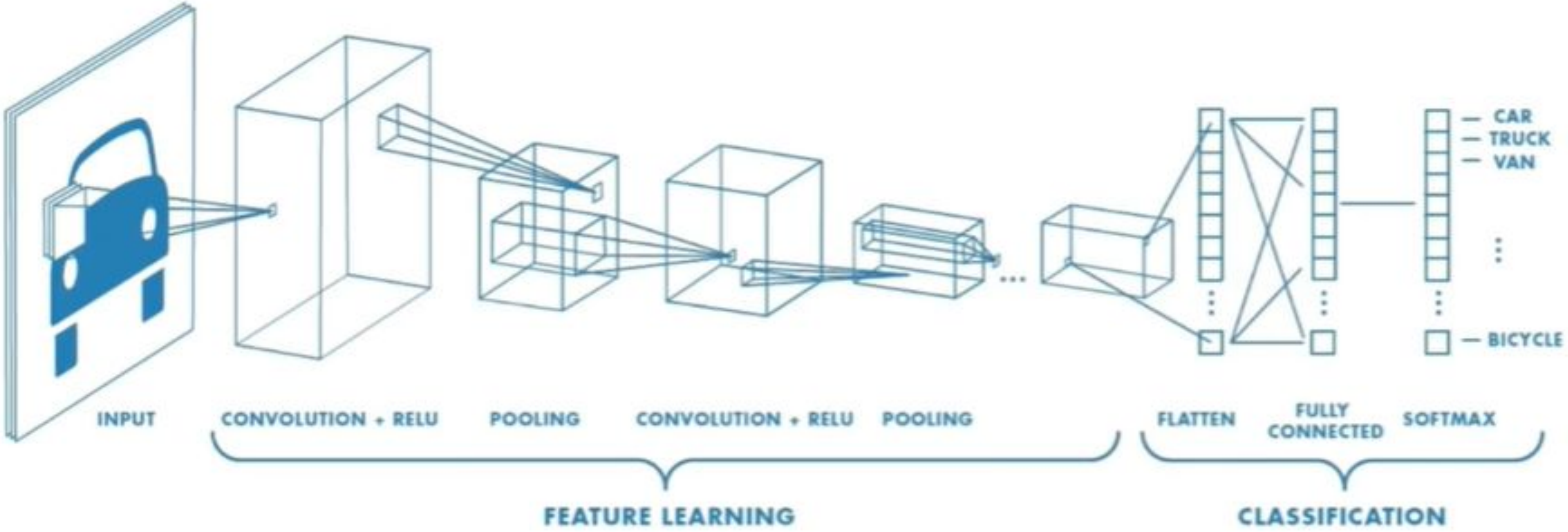


Background color white of image blends with added background color white



Added background color white doesn't blend with image thereby adding noise

Summary : Typical CNN's for Object Classification



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

n_{in}: number of input features

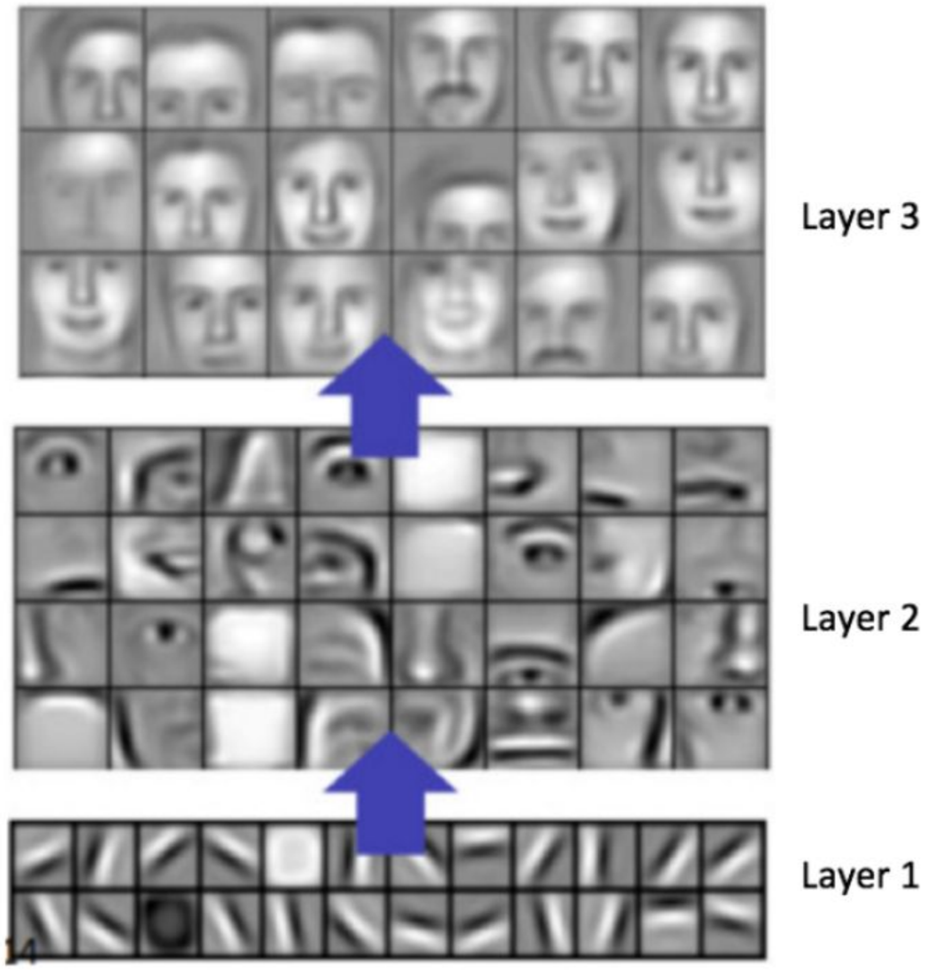
n_{out}: number of output features

k: convolution kernel size

p: convolution padding size

s: convolution stride size

Intuition behind multiple layers



References

https://towardsdatascience.com/what-is-wrong-with-convolutional-neural-networks-75c2ba8fbd6f

He, Kaiming, et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition." arXiv preprint arXiv:1406.4729 (2014). [PDF]

LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324

Fei-Fei, Li, et al. "What do we perceive in a glance of a real-world scene?." Journal of vision 7.1 (2007):

Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection."

Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.•

Felzenszwalb, Pedro, David McAllester, and Deva Ramanan. "A discriminatively trained, multiscale, deformable part model."

Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008

Everingham, Mark, et al. "The pascal visual object classes (VOC) challenge." International Journal of Computer Vision 88.2 (2010): 303-338.

Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.
Russakovsky, Olga, et al. "Imagenet Large Scale Visual Recognition Challenge." arXiv:1409.0575. [PDF]