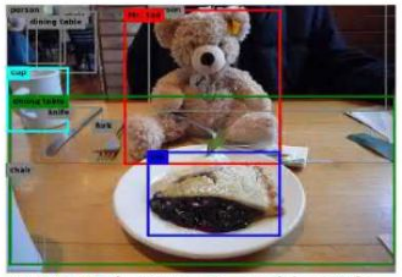
Deep learning for Computer Vision

Outline

- Introduction to Computer Vision
- Problems using Fully Connected Networks on Images
- What are convolutions
- Convolution on Images
- Stride, Padding, Pooling
- Dimension of a Convolution Layer
- Stacking Convolution Layers
- Convolution Neural Networks for images and its applications

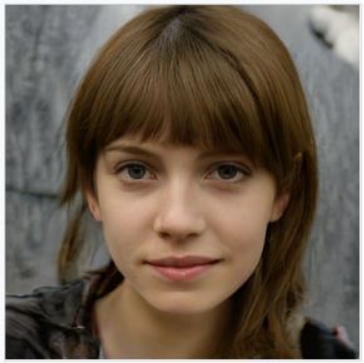
Computer Vision Tasks

segmentation



Object Detection

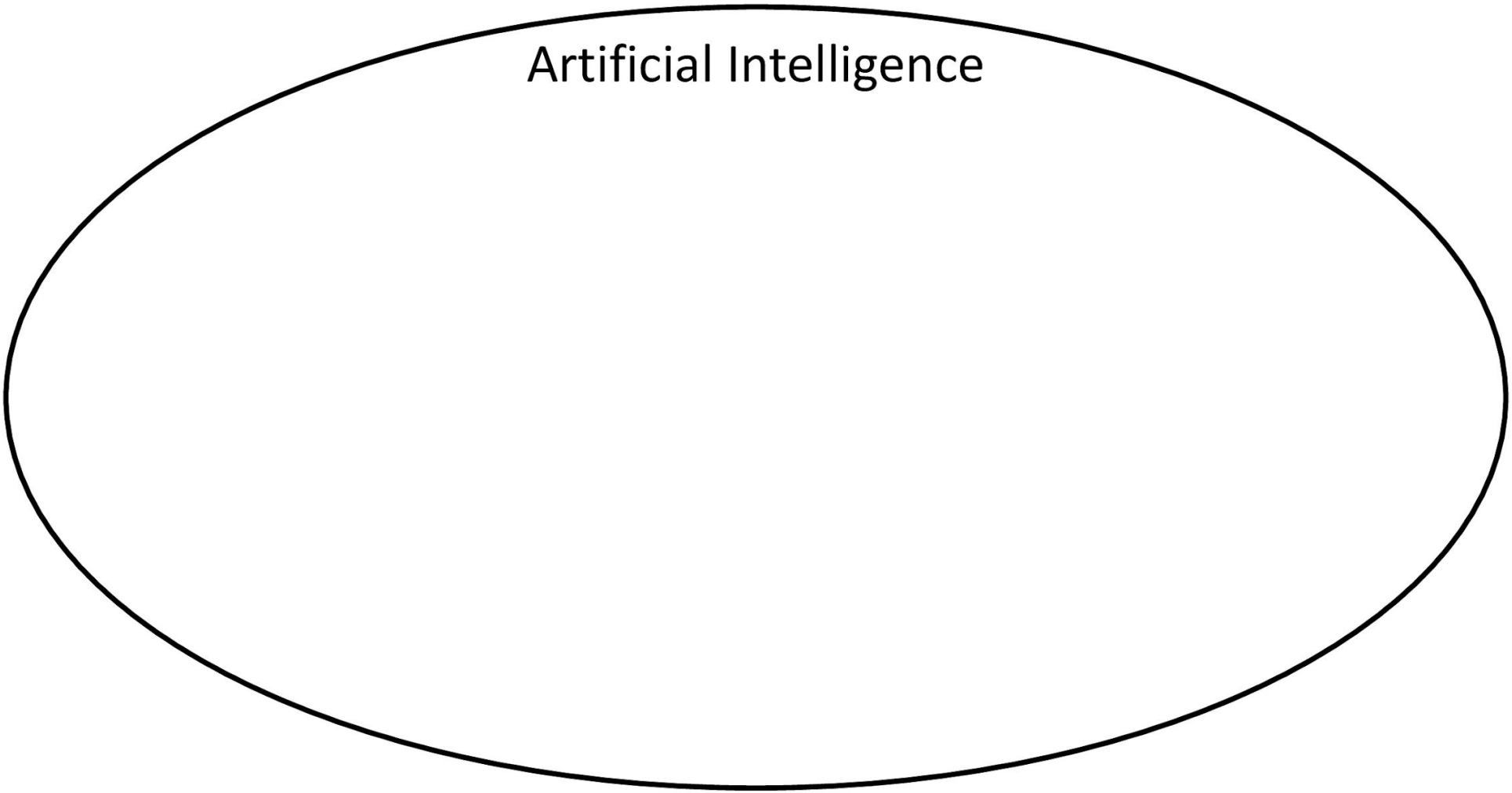
Image Restoration

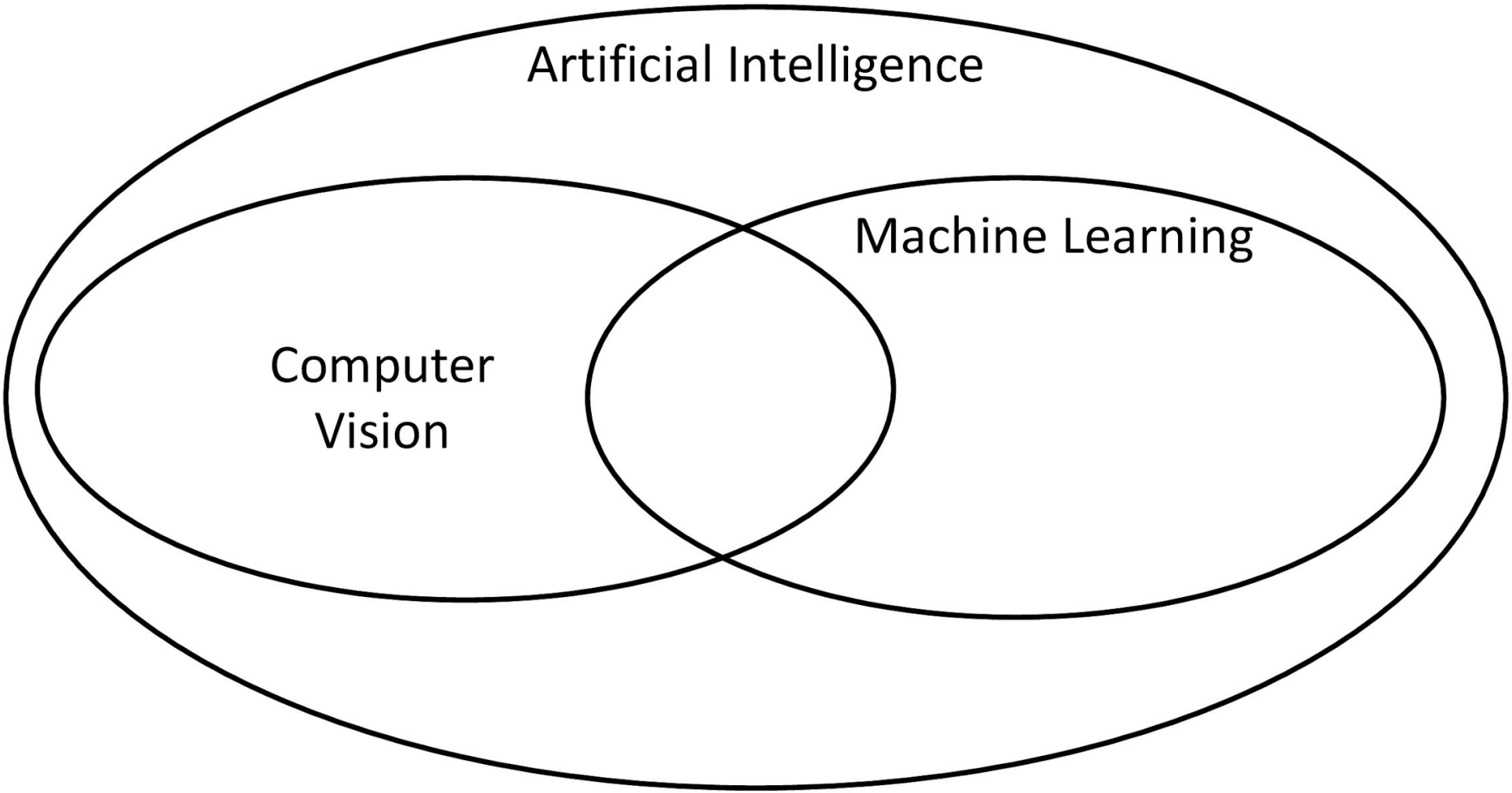


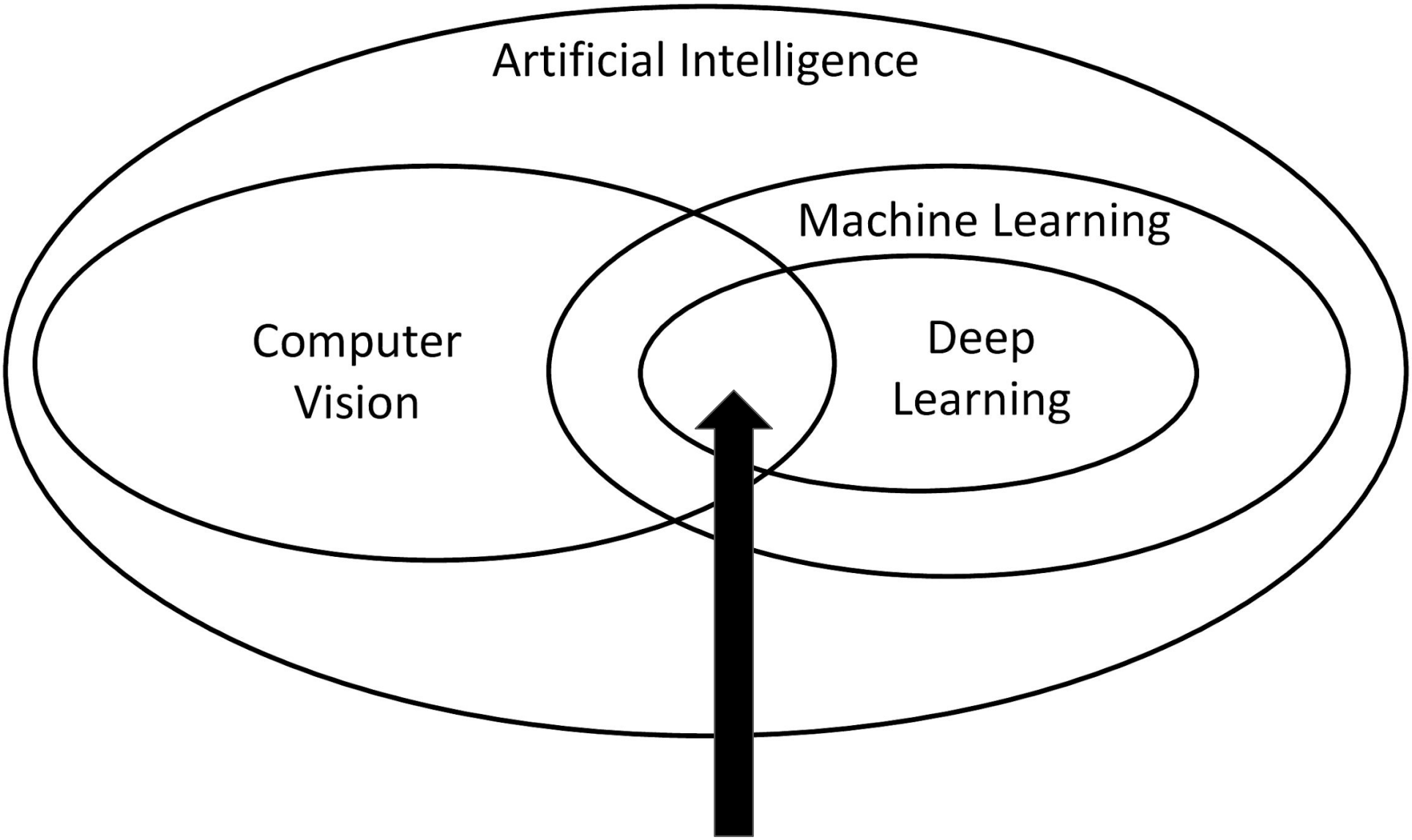
An image generated by a StyleGAN

Image Synthesis

Flash no-flash

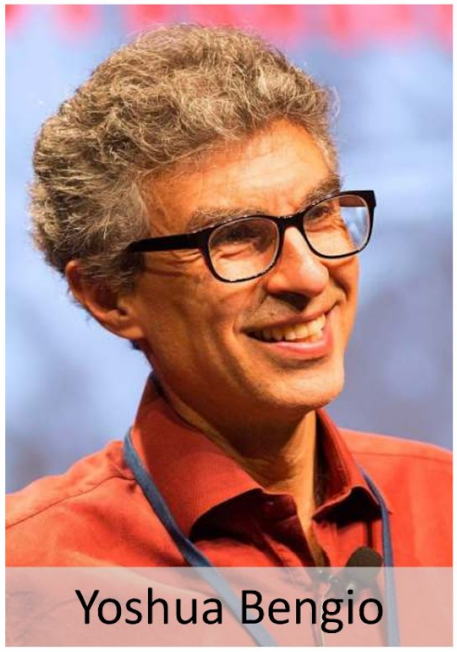

Artificial Intelligence


Artificial Intelligence
Machine Learning
Computer
Vision



Deep learning for Computer Vision

2018 Turing Award



The 2018 Turing Award was awarded jointly to Yoshua Bengio, Geoffrey Hinton, and Yann LeCun for their pioneering work on deep learning

Geoffrey Hinton is known by many to be the godfather of deep learning.

Geoffrey E. Hinton is known by many to be the godfather of deep learning.

Geoffrey Everest Hinton early in his career. His middle name comes from a relative, George Everest, who surveyed **India**

Geoffrey E. Hinton

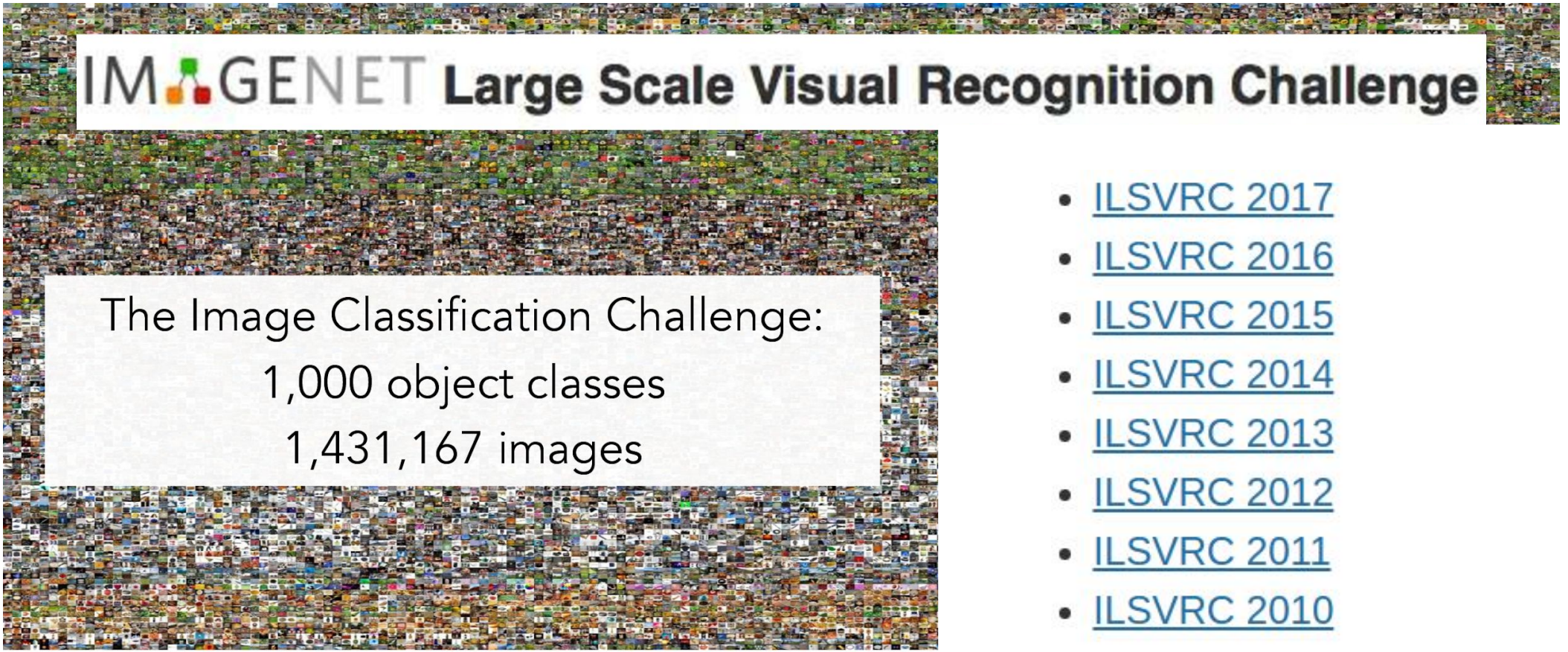
- Geoffrey Hinton's foundational contribution to **backpropagation algorithm** can be traced back to his work on the "Learning Representations by Back-Propagating Errors".
- Hinton's work helped to establish the theoretical foundations for training CNNs and RNNs and showed that it was possible to learn useful representations using Deep Neural Networks.
- For example, Backpropagation through time" (BPTT) algorithm is a variant of the backpropagation algorithm that is specifically designed for training recurrent neural networks.

Yann LeCun

- Yann LeCun is well known for his work on Convolutional Neural Networks, representation learning, geomatic deep learning.
- **LeNet** is a convolutional neural network (CNN) architecture designed by Yann LeCun et. al. in 1998.
- LeNet is used for the recognition of handwritten digits in the MNIST dataset.

Yoshua Bengio

- Yoshua Bengio initial works on "Learning long-term dependencies with gradient descent is difficult", uncover fundamental difficulty of learning in RNNs.
- Bengio's has also done significant contribution in
  - Recurrent neural networks (RNNs)
  - Word embeddings from neural networks and neural language models,
  - Unsupervised deep learning based on auto-encoders,
  - introducing Generative Adversarial Networks (GANs).

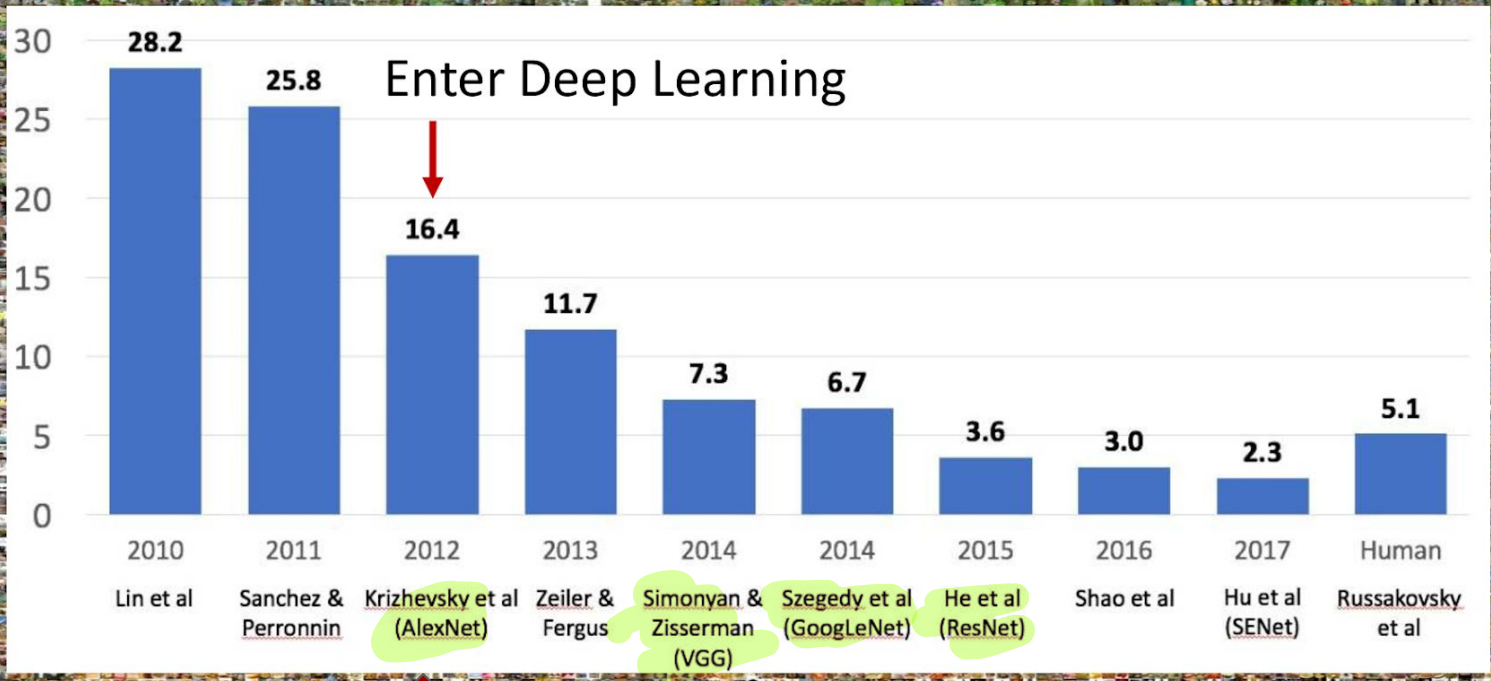


- ILSVRC 2017
- ILSVRC 2016
- ILSVRC 2015
- ILSVRC 2014
- ILSVRC 2013
- ILSVRC 2012
- ILSVRC 2011
- ILSVRC 2010

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

- ILSVRC challenge is designed to evaluate and advance image classification and object detection algorithms.
- ILSVRC is an annual computer vision competition that was first held in 2010.
- The competition uses ImageNet dataset, which contains millions of images organized into thousands of categories.
- The challenge has been a driving force behind many of the recent advances in computer vision, including self-driving cars, robotics, and augmented reality.

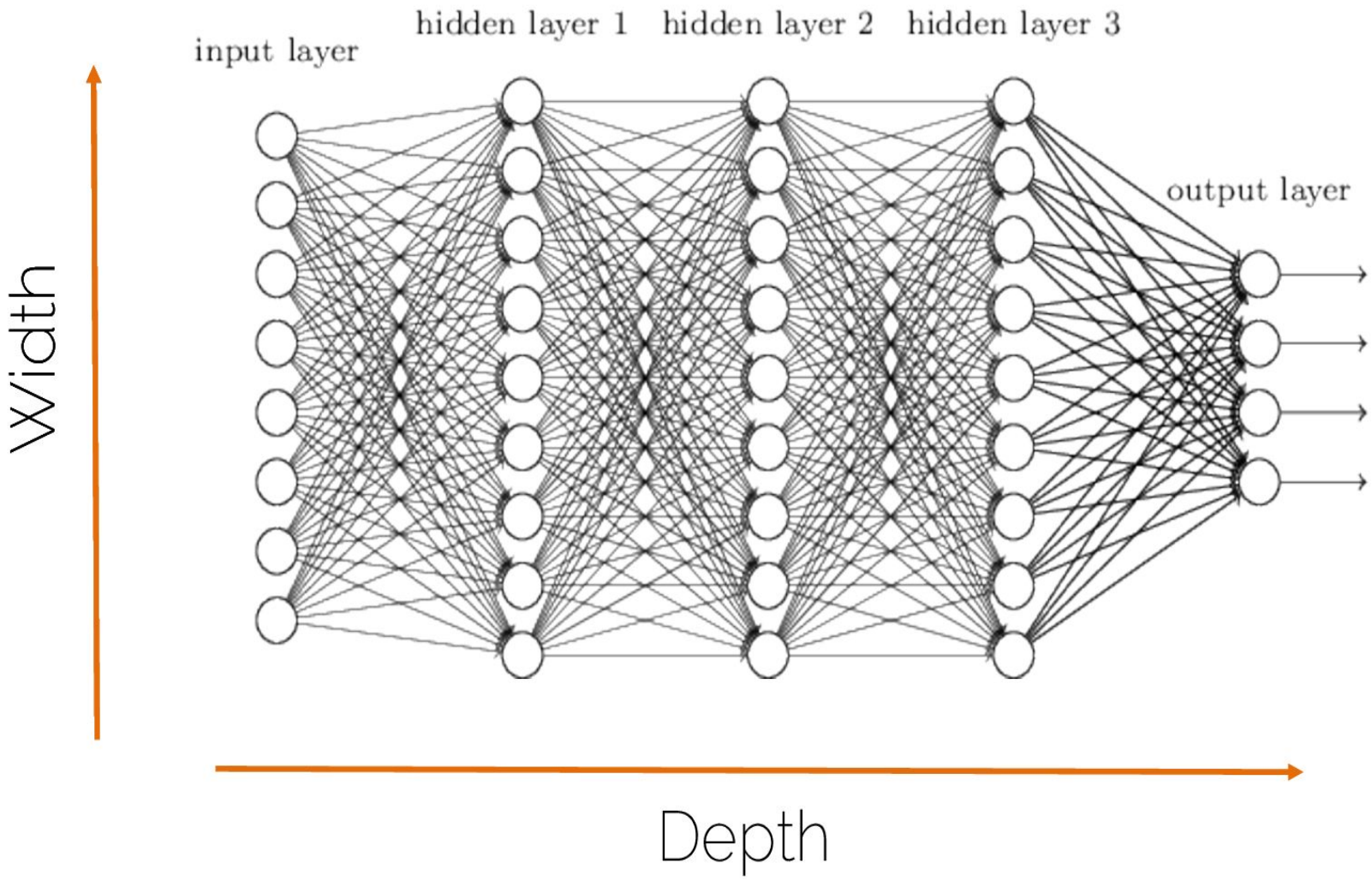
IMAGENET Large Scale Visual Recognition Challenge



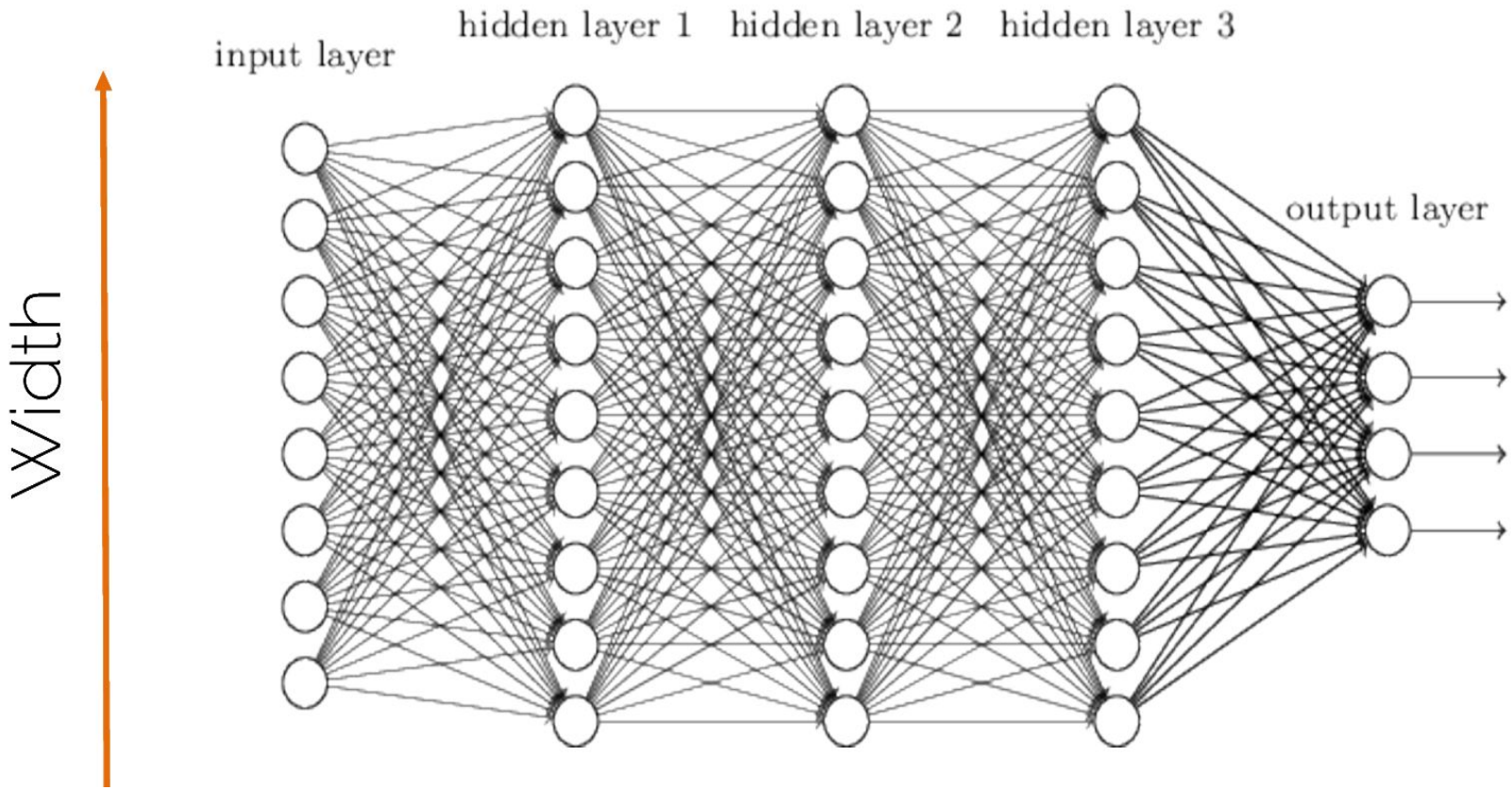
in collaboration with
Geoffrey E. Hinton

Problems using Fully Connected Networks on Images

Fully Connected Neural Network (FC)



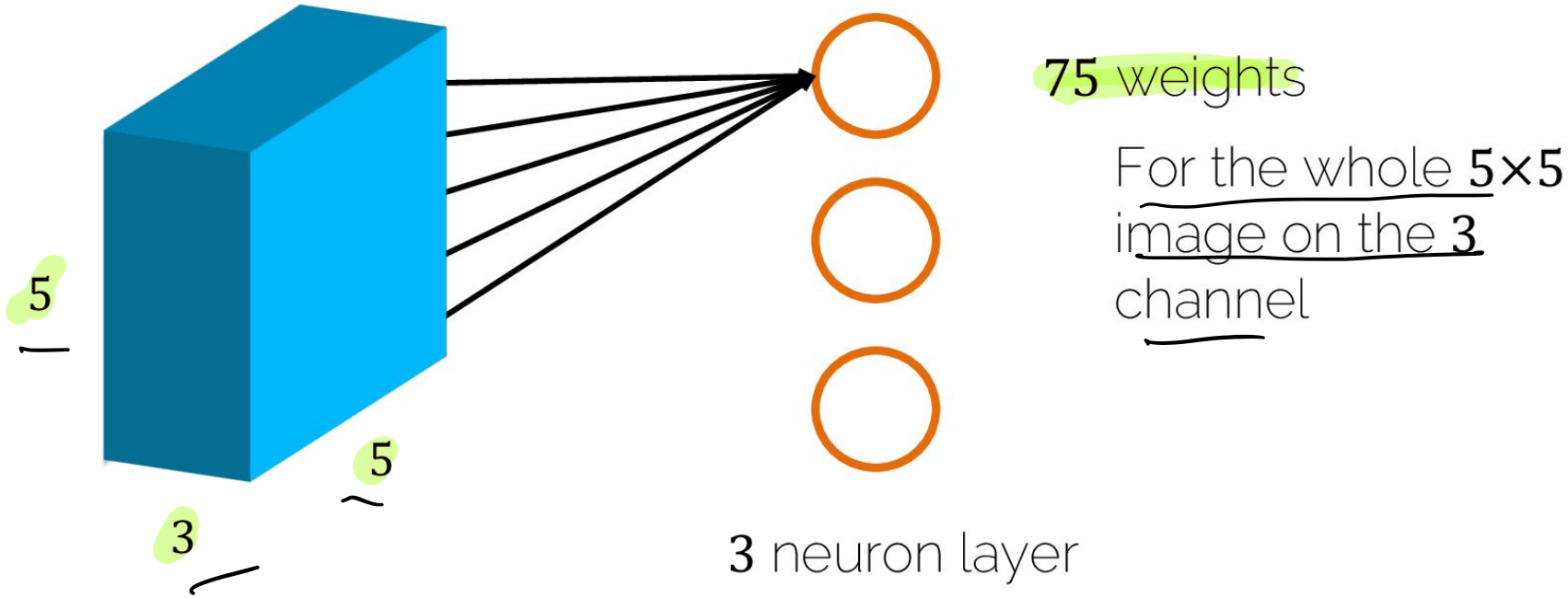
Fully Connected Neural Network (FC)



How to process image data?

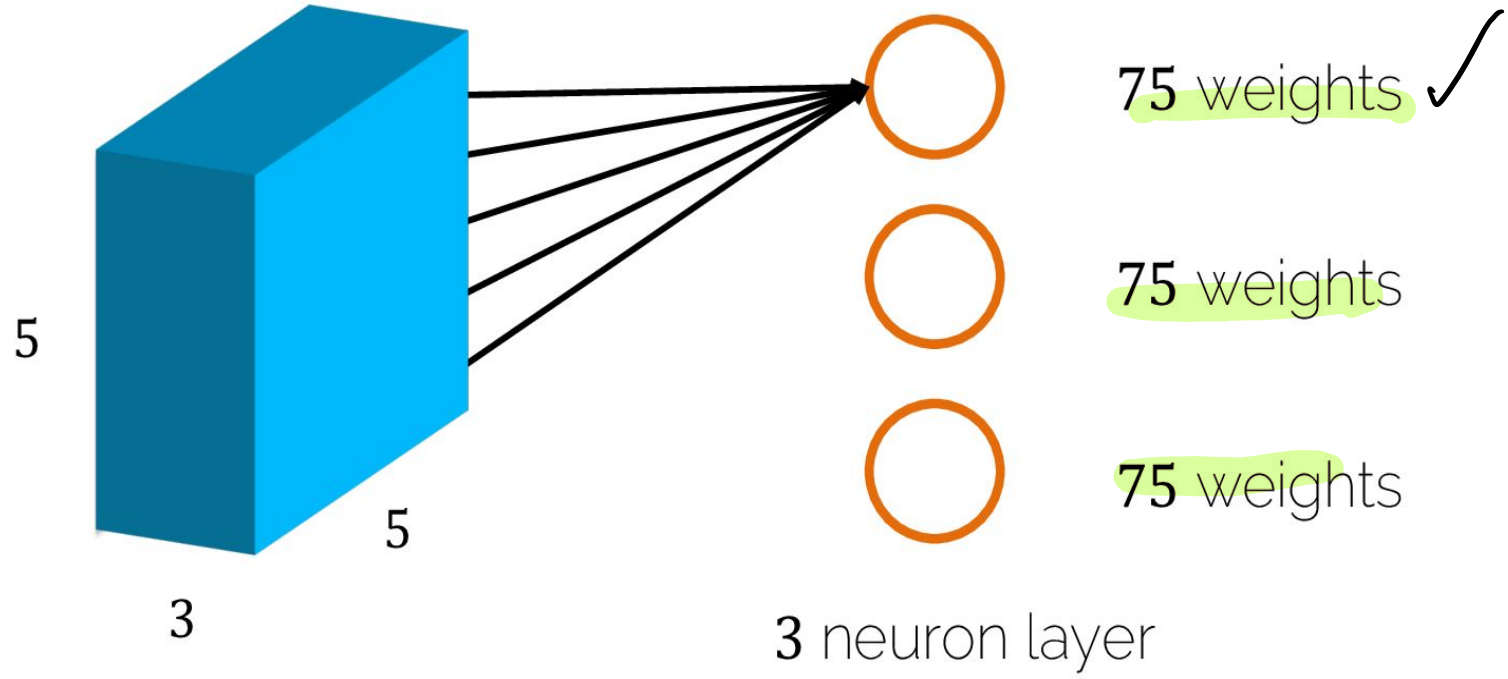
Problems using FC Layers on Images

How to process a tiny image with FC layers



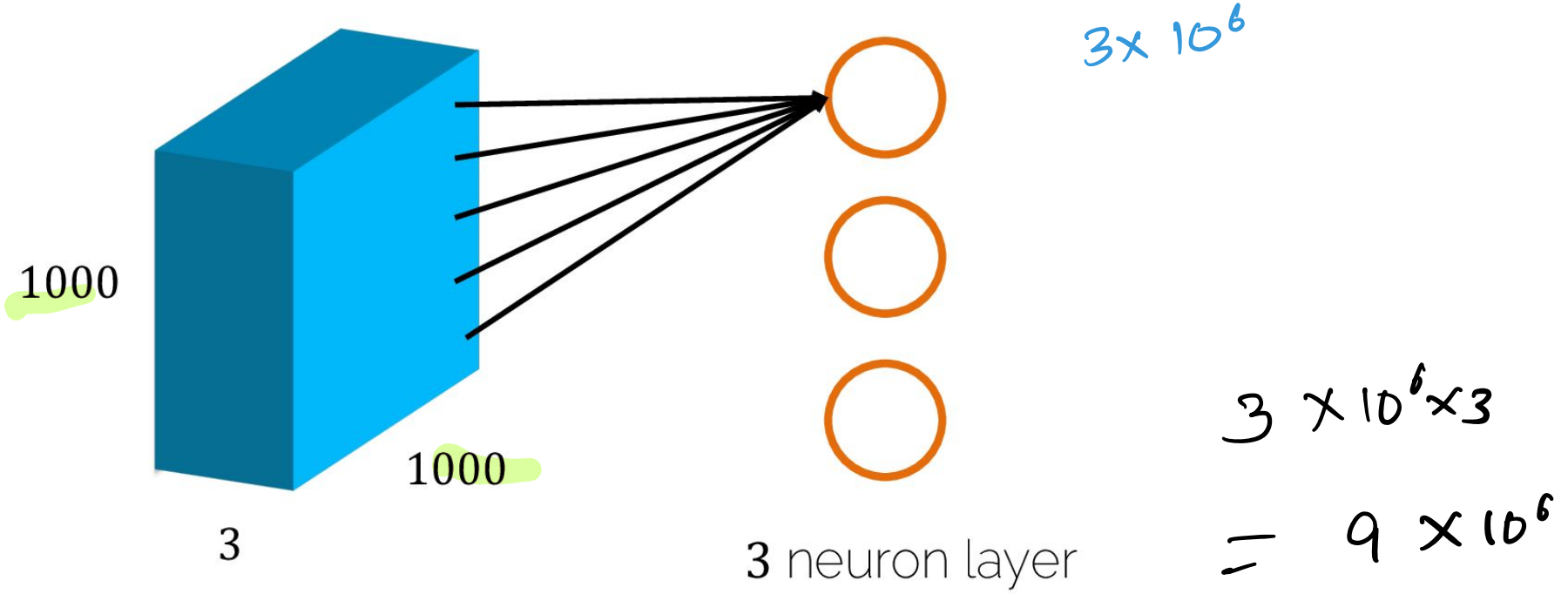
Problems using FC Layers on Images

How to process a tiny image with FC layers



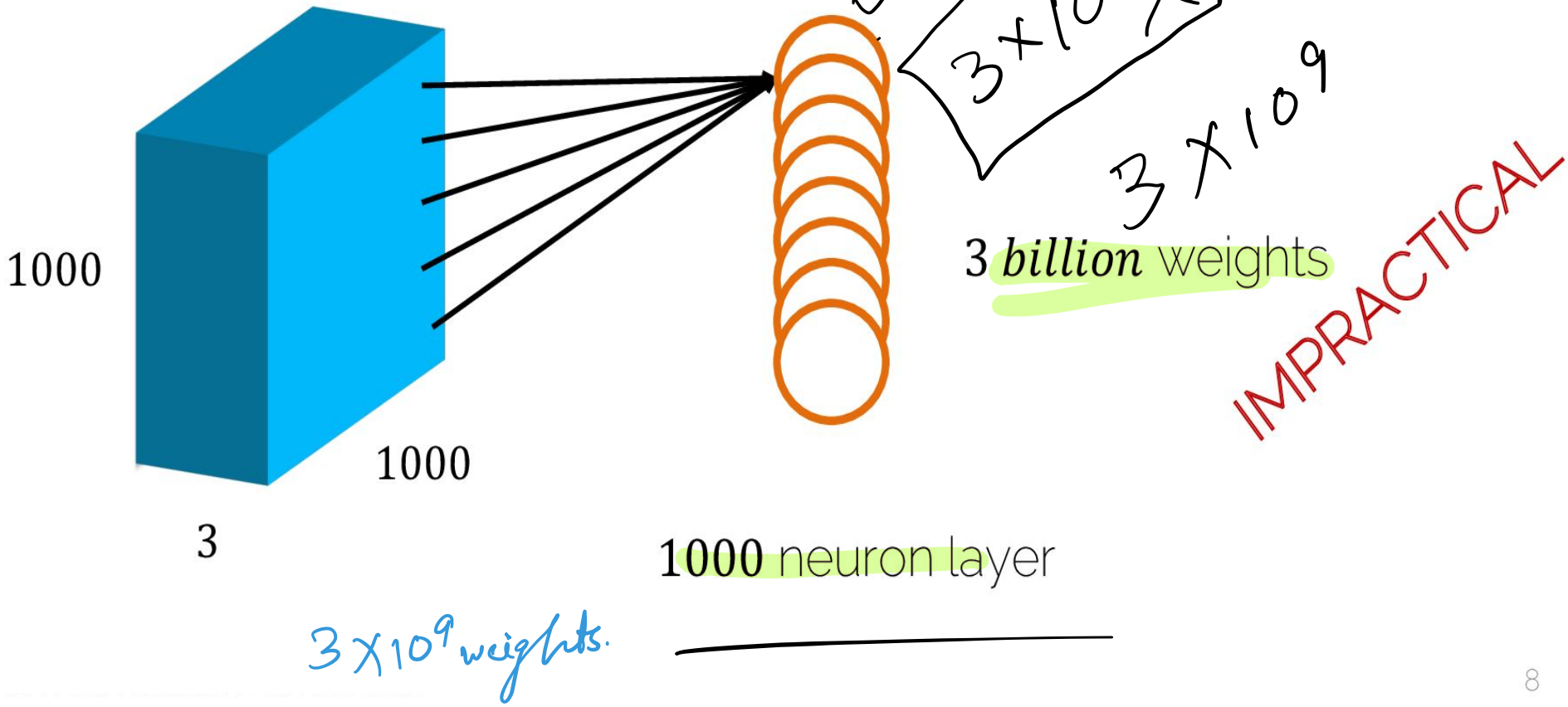
Problems using FC Layers on Images

How to process a tiny image with FC layers



Problems using FC Layers on Images

How to process a tiny image with FC layers



Disadvantages of FC Layers for Image Data

- Dense layers **require a lot of parameters**, which can lead to overfitting when the number of input features is large.
- Dense layers are **not translation invariant**, meaning that small shifts in the input image can result in large changes in the output.
- Dense layers do not take advantage of the spatial structure of images, and can therefore be **inefficient** for processing large images.

Solution: Convolution Neural Networks for images

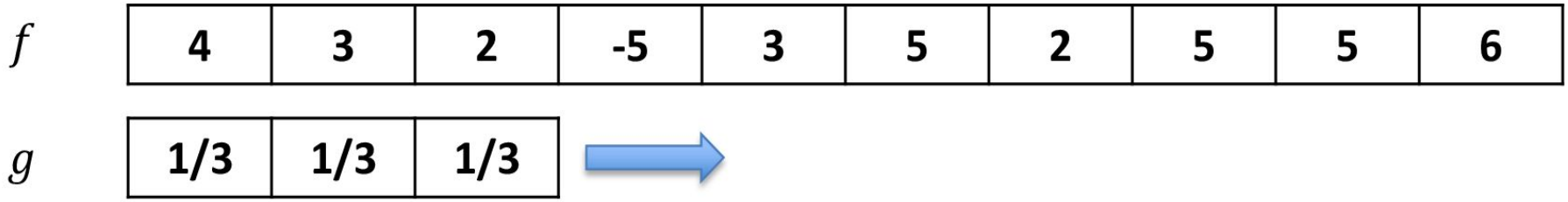
Convolution is performed differently than FC

Convolution Neural Networks for images

What are convolutions?

What are Convolutions?

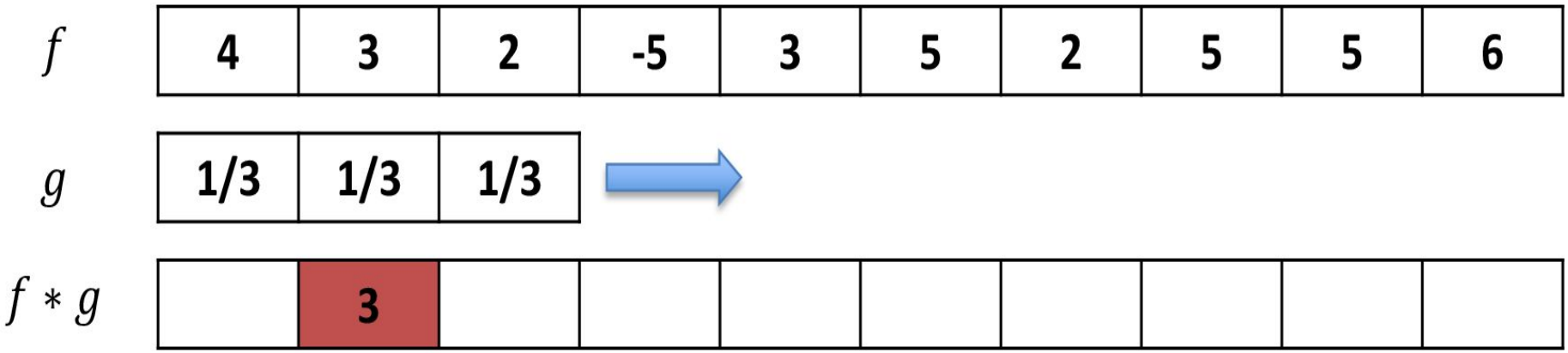
Discrete case: box filter



Slide' **filter kernel** from left to right; at each position, compute a single value in the output data

What are Convolutions?

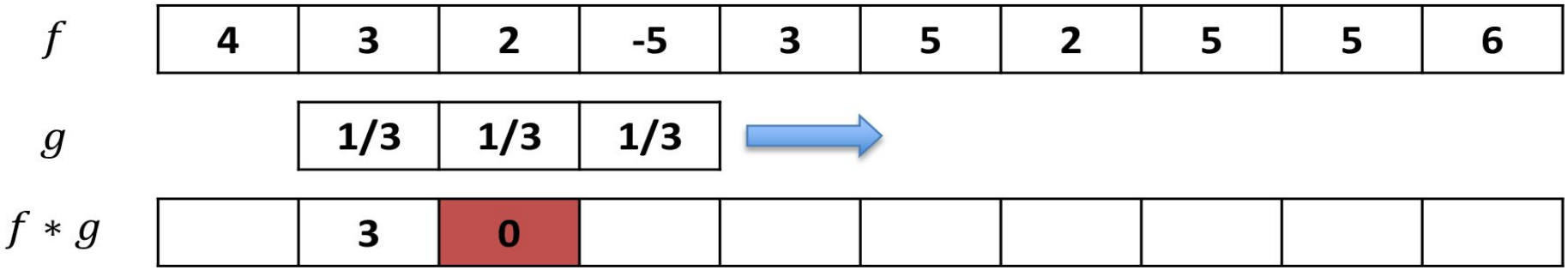
Discrete case: box filter



$$4 \cdot \frac{1}{3} + 3 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} = 3$$

What are Convolutions?

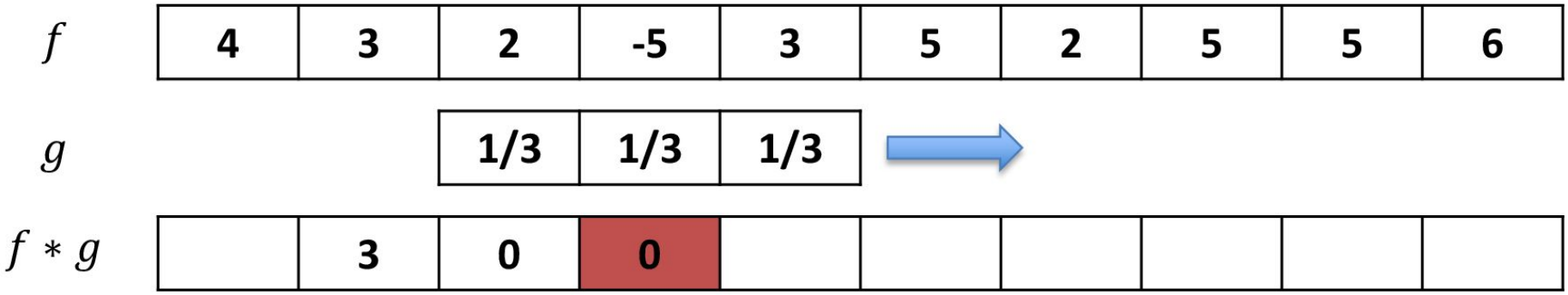
Discrete case: box filter



$$3 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} + (-5) \cdot \frac{1}{3} = 0$$

What are Convolutions?

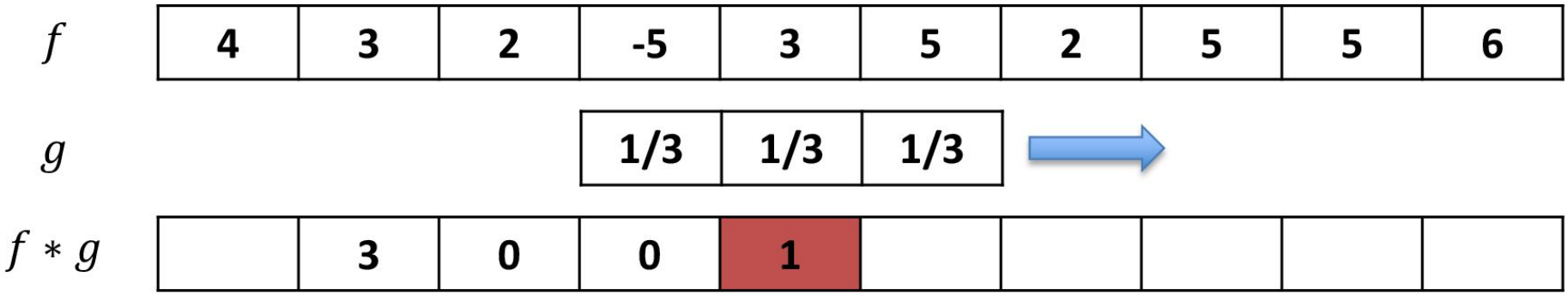
Discrete case: box filter



$$2 \cdot \frac{1}{3} + (-5) \cdot \frac{1}{3} + 3 \cdot \frac{1}{3} = 0$$

What are Convolutions?

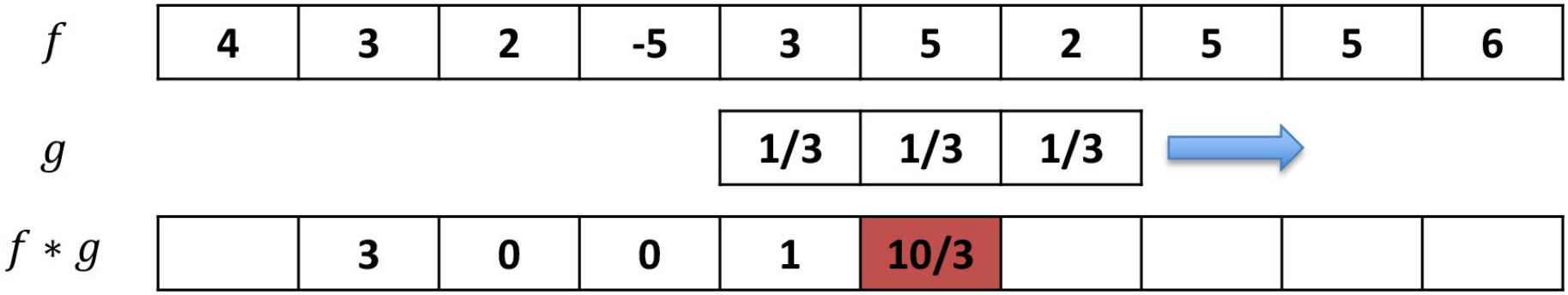
Discrete case: box filter



$$(-5) \cdot \frac{1}{3} + 3 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} = 1$$

What are Convolutions?

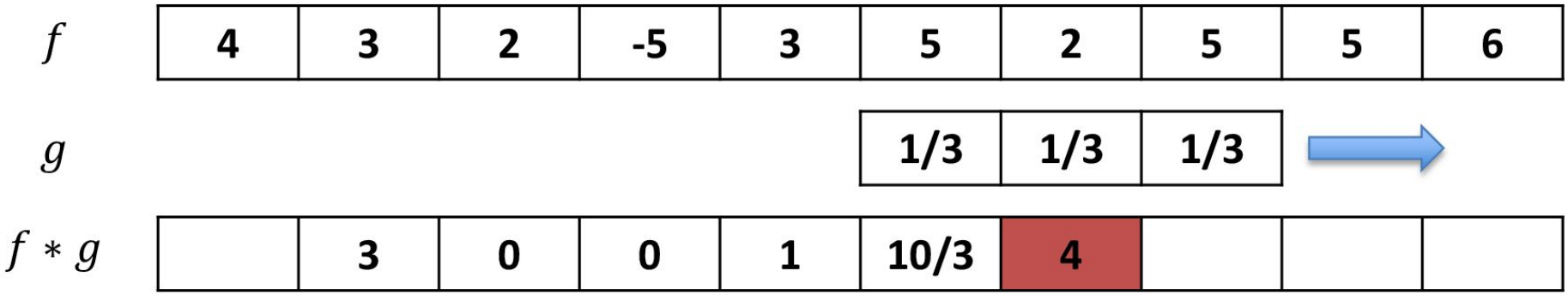
Discrete case: box filter



$$3 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} = \frac{10}{3}$$

What are Convolutions?

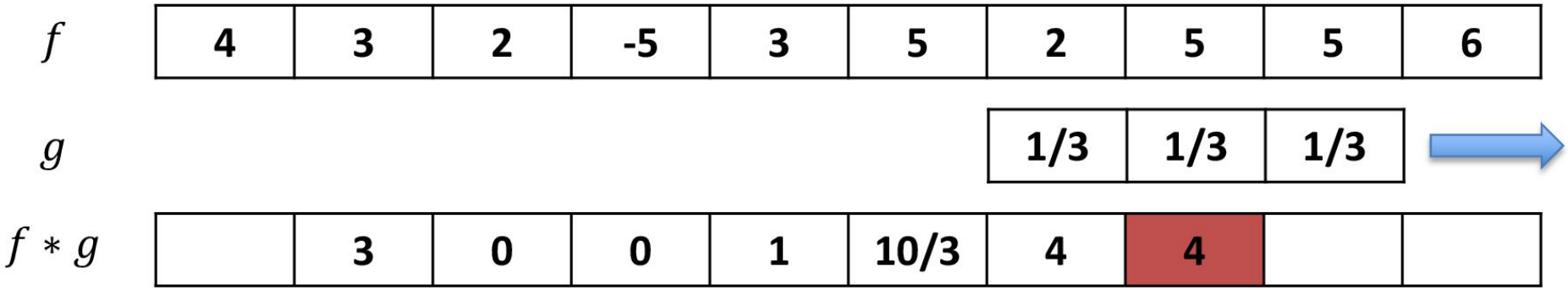
Discrete case: box filter



$$5 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} = 4$$

What are Convolutions?

Discrete case: box filter



$$2 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} = 4$$

What are Convolutions?

Discrete case: box filter

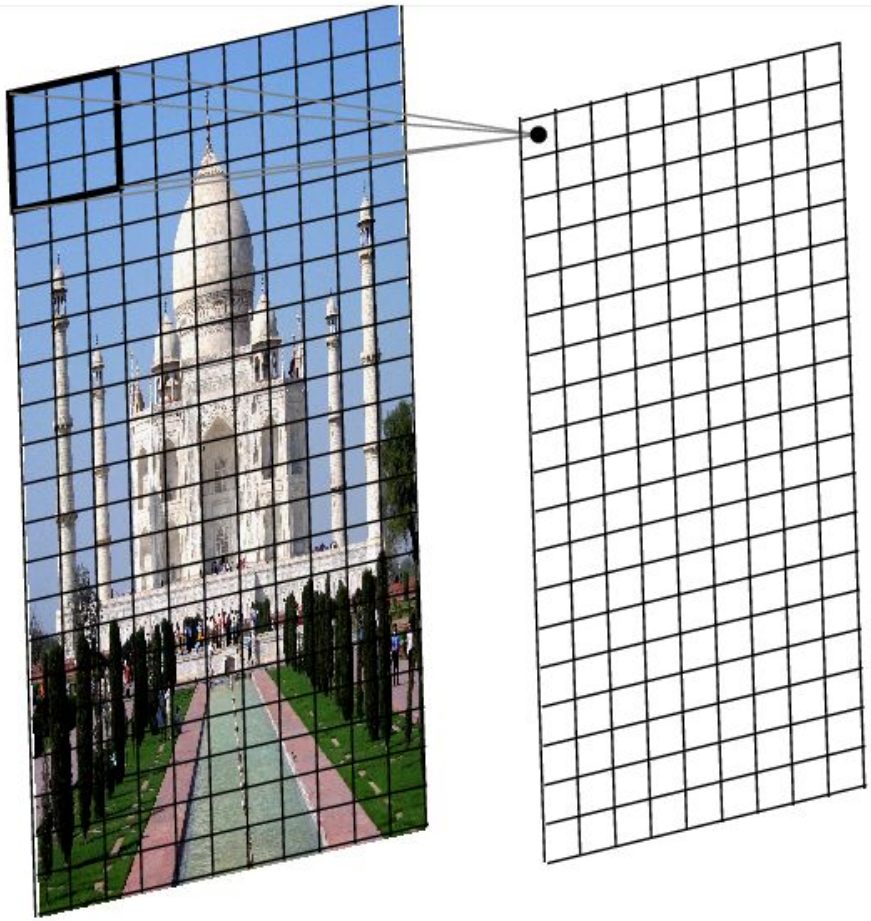
f	4	3	2	-5	3	5	2	5	5	6
g								1/3	1/3	1/3
$f * g$		3	0	0	1	10/3	4	4	16/3	

$$5 \cdot \frac{1}{3} + 5 \cdot \frac{1}{3} + 6 \cdot \frac{1}{3} = \frac{16}{3}$$

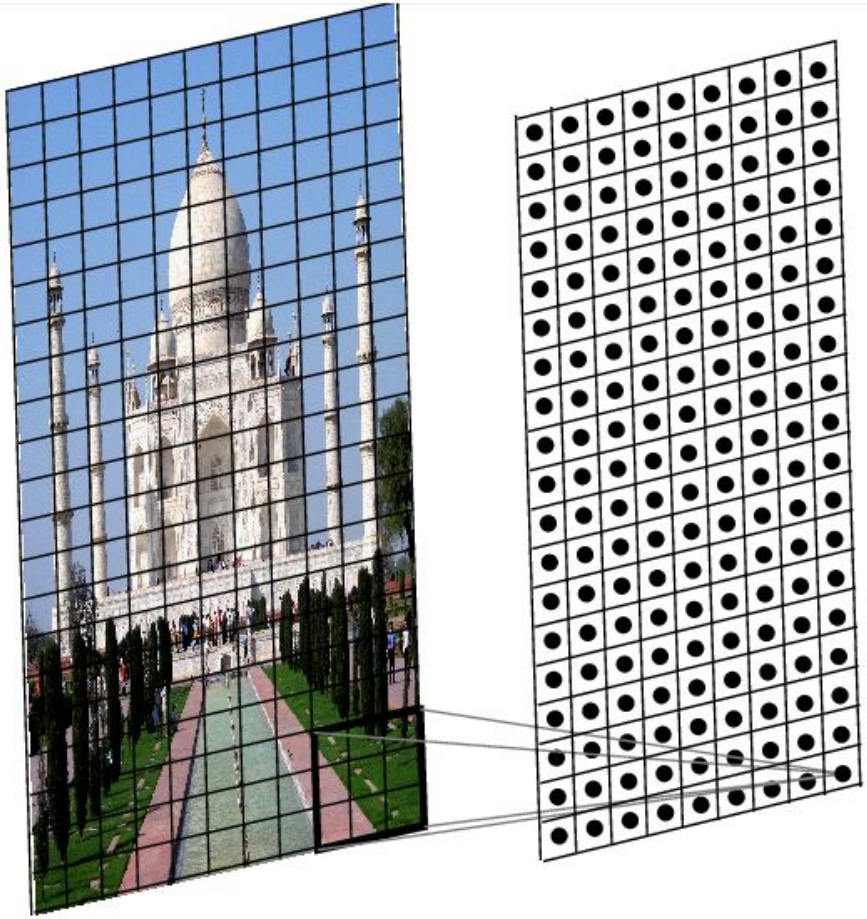
Convolution on Images

Convolutions on images

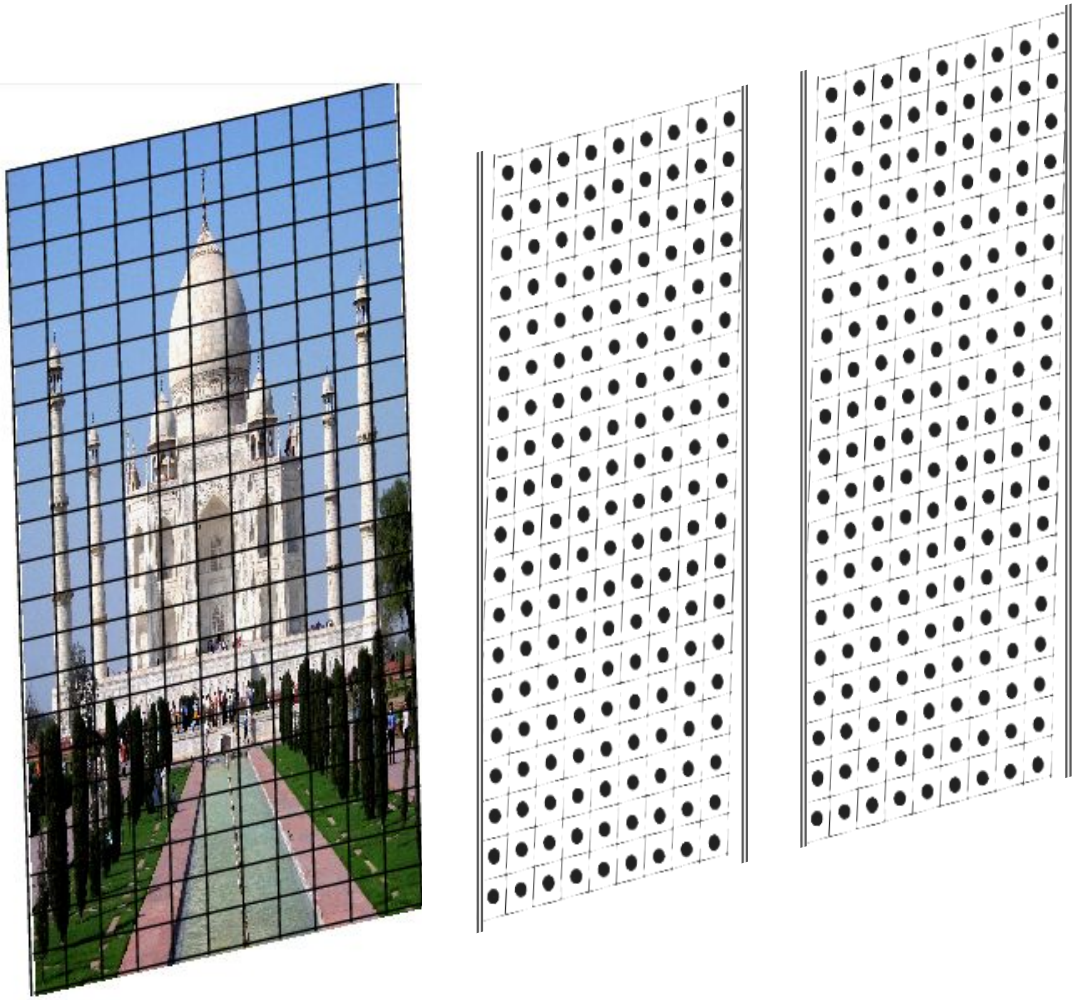
$$\begin{matrix} & 0 & -1 & 0 \\ * & -1 & 5 & -1 \\ & 0 & -1 & 0 \end{matrix} =$$



- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output

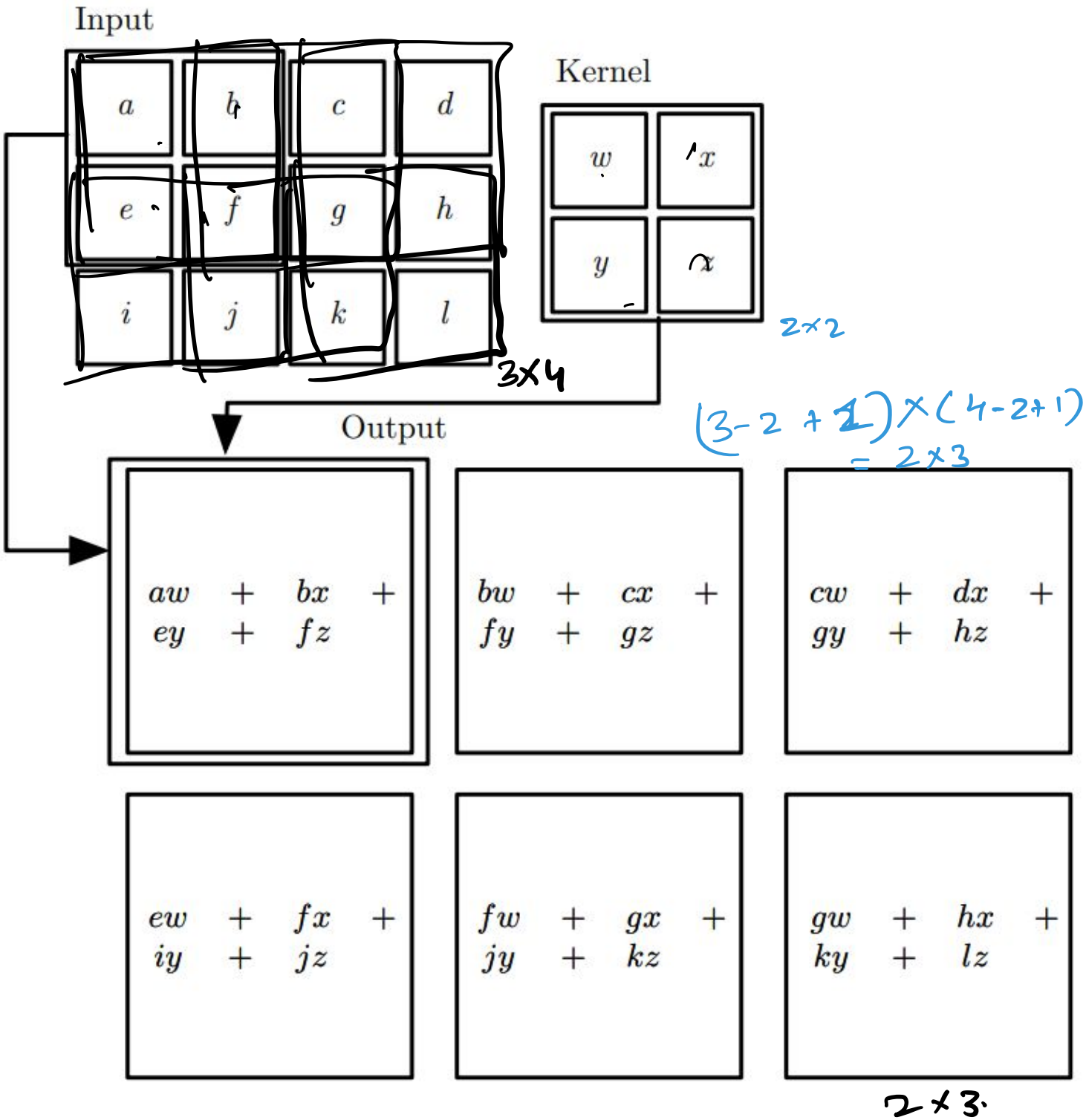


- We just slide the kernel over the input image
- Each time we slide the kernel we get one value in the output
- The resulting output is called a feature map.

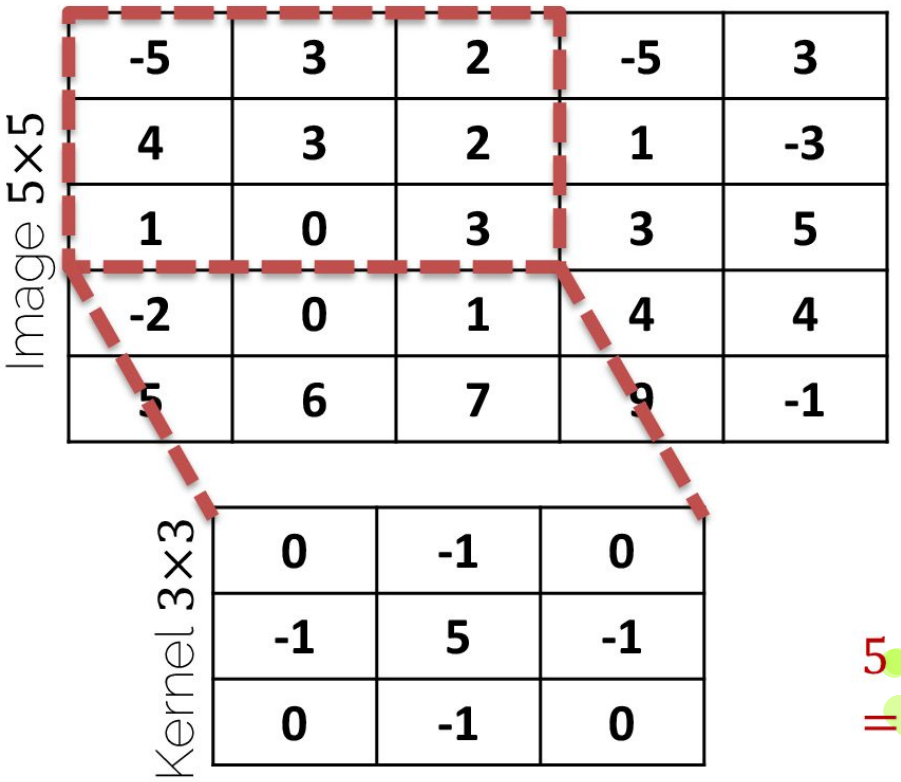


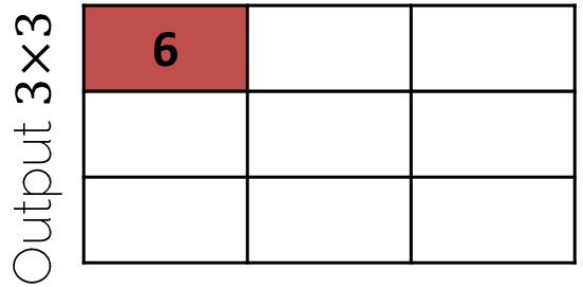
- We can use multiple filters to get multiple feature maps.

Convolution



Convolutions on Images





$5 \cdot 3 + (-1) \cdot 3 + (-1) \cdot 2 + (-1) \cdot 0 + (-1) \cdot 4$
$= 15 - 9 = 6$

5x5x3

Convolutions on Images

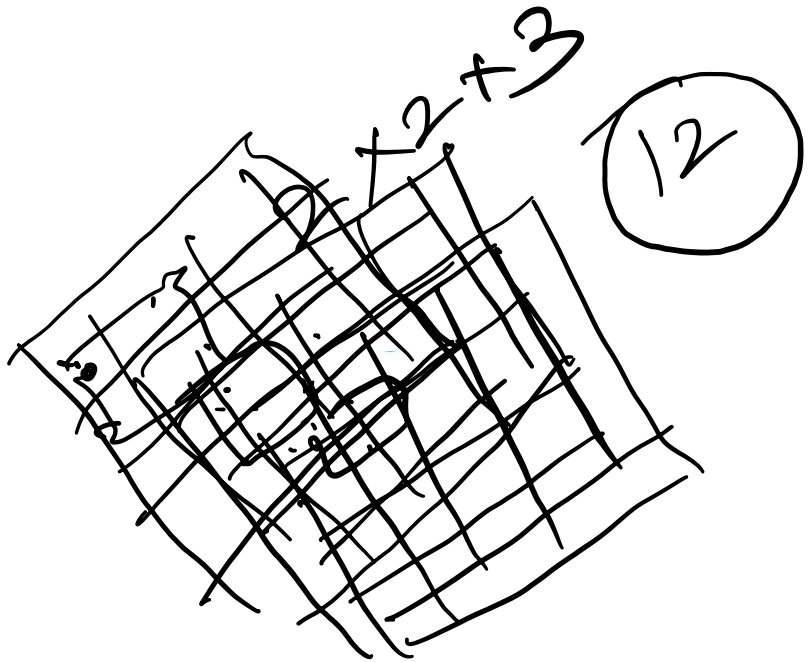


Image 5×5

-5	3	2	-5	3
4	3	2	1	-3
1	0	3	3	5
-2	0	1	4	4
5	6	7	9	-1

Kernel 3×3

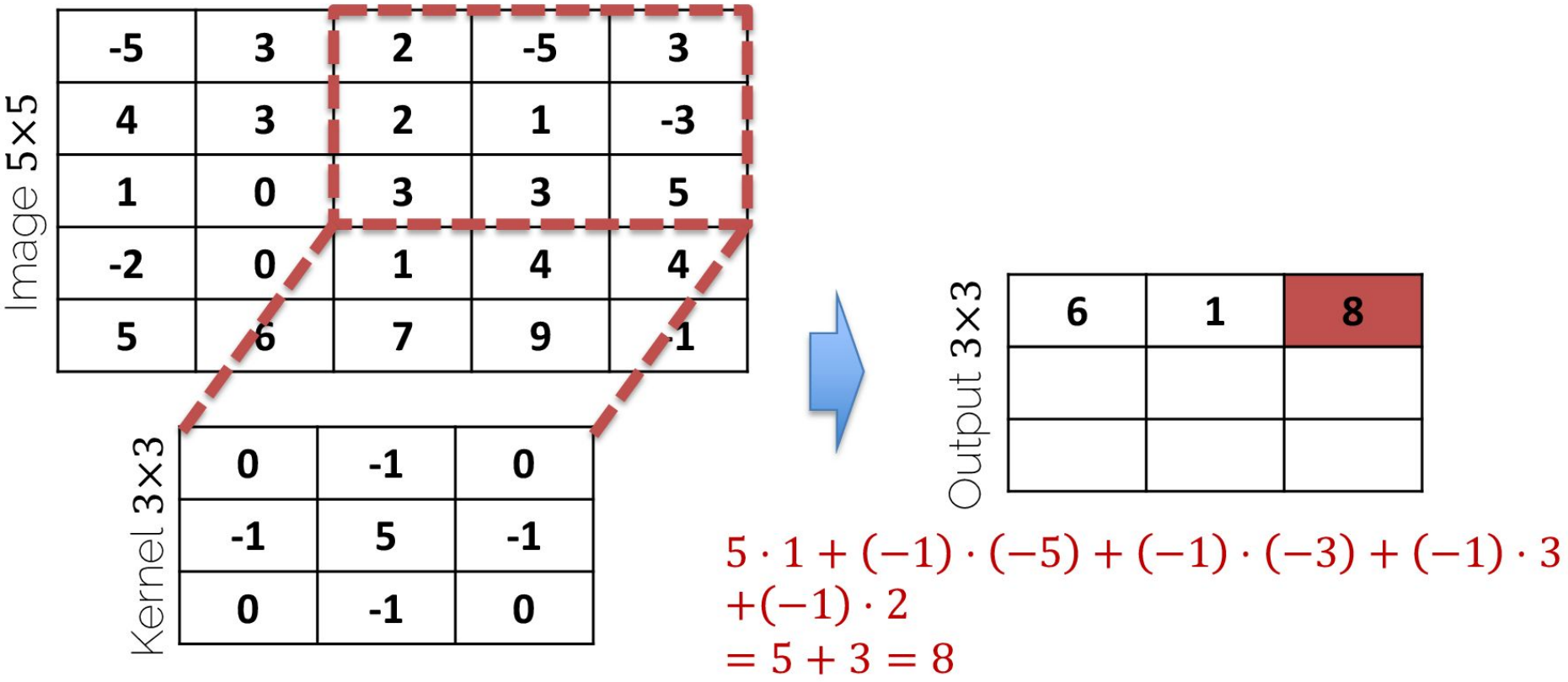
0	-1	0
-1	5	-1
0	-1	0

Output 3×3

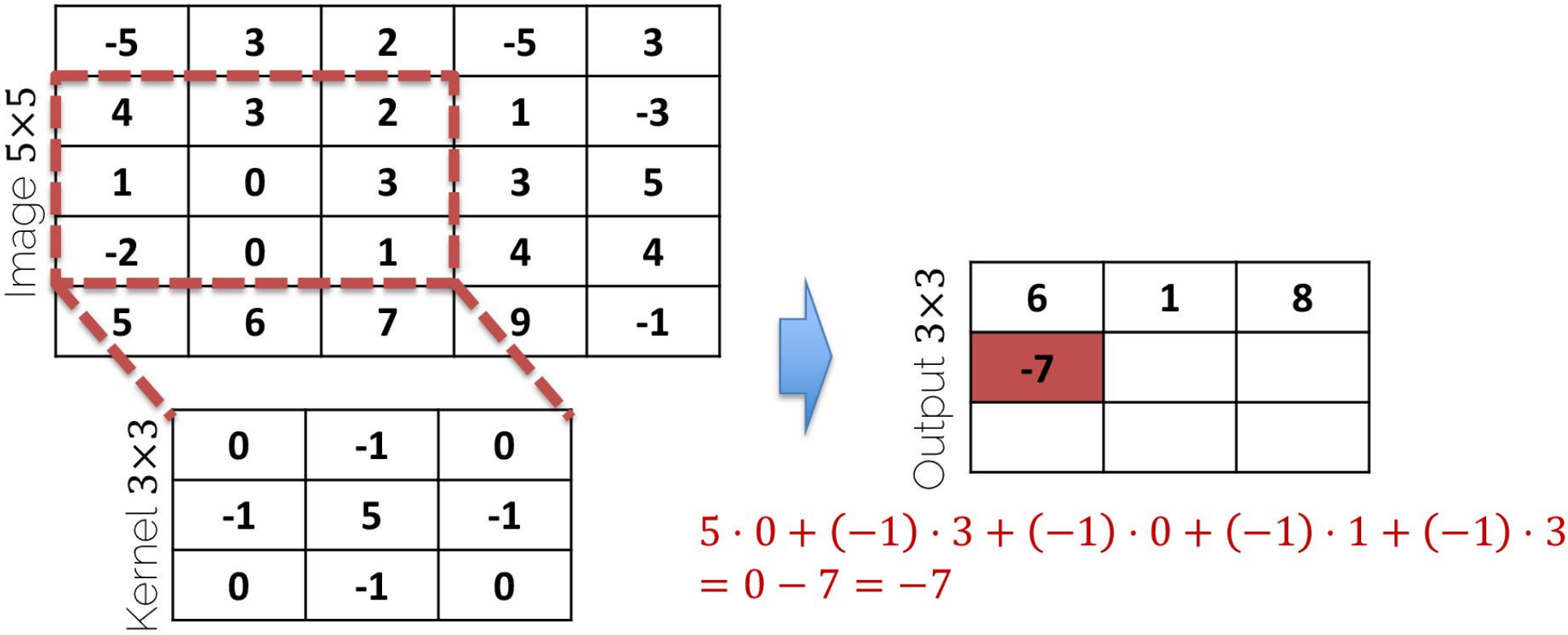
6	1	

$$5 \cdot 2 + (-1) \cdot 2 + (-1) \cdot 1 + (-1) \cdot 3 + (-1) \cdot 3$$
$$= 10 - 9 = 1$$

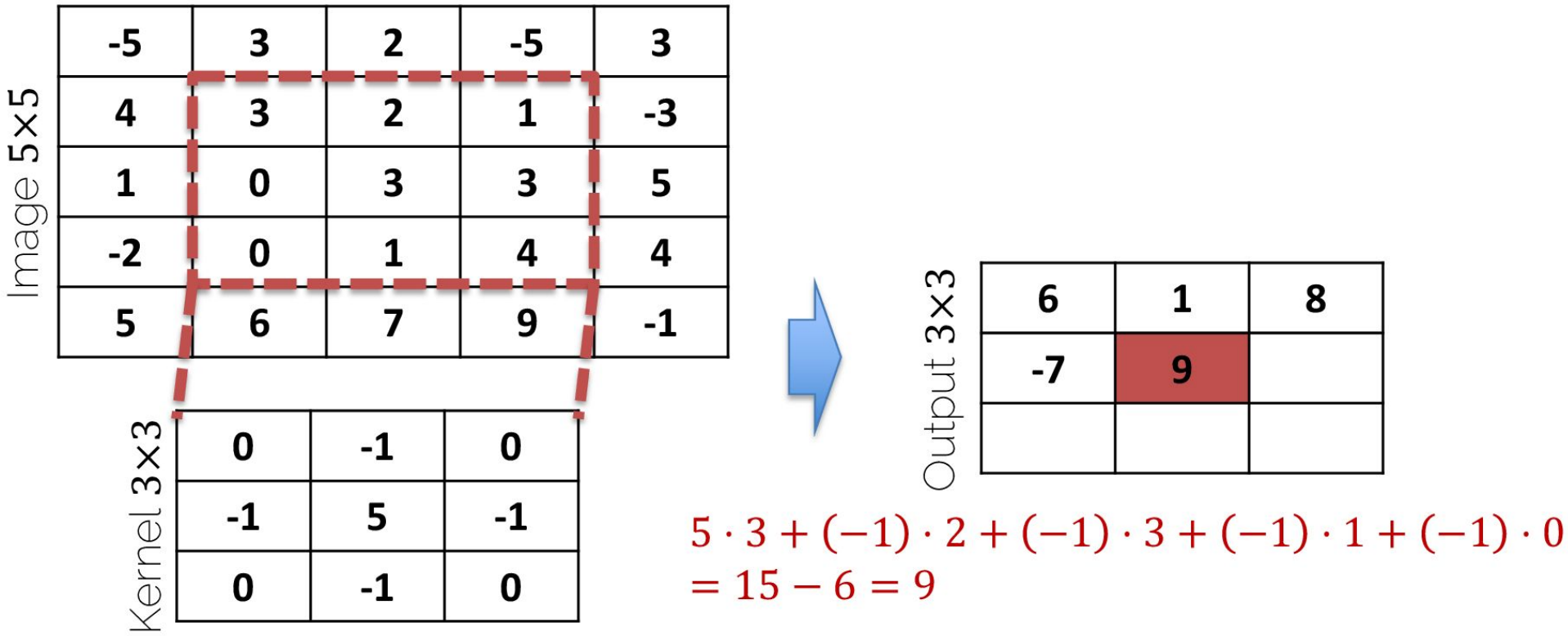
Convolutions on Images



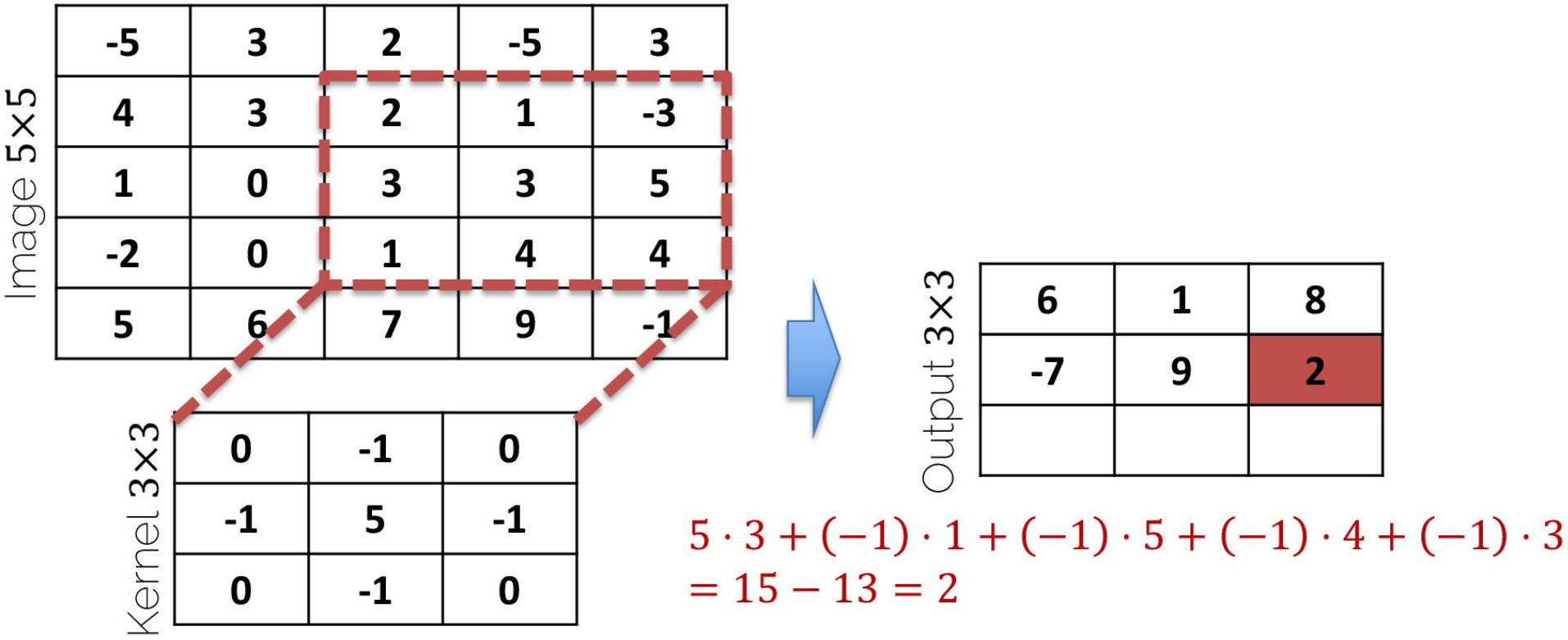
Convolutions on Images



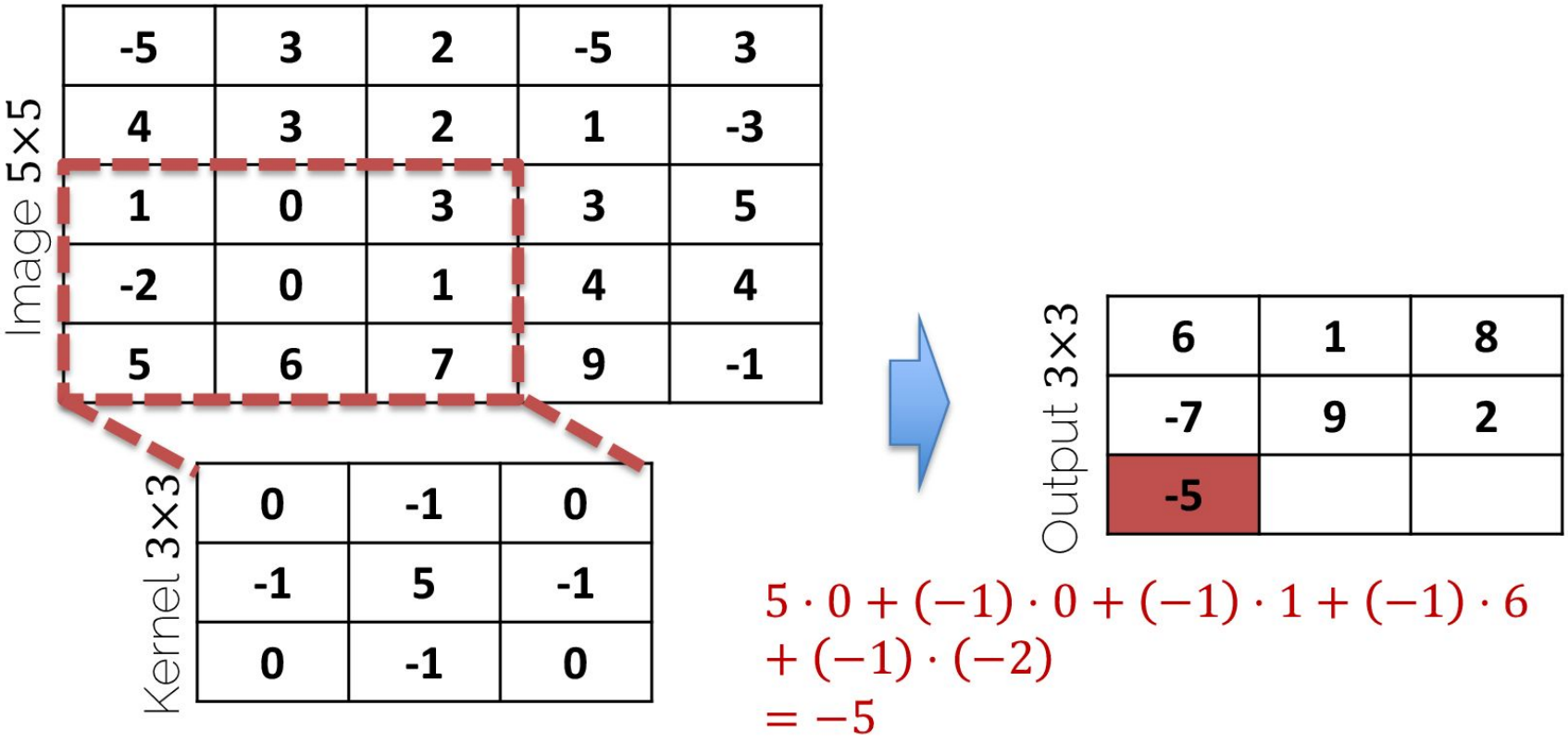
Convolutions on Images



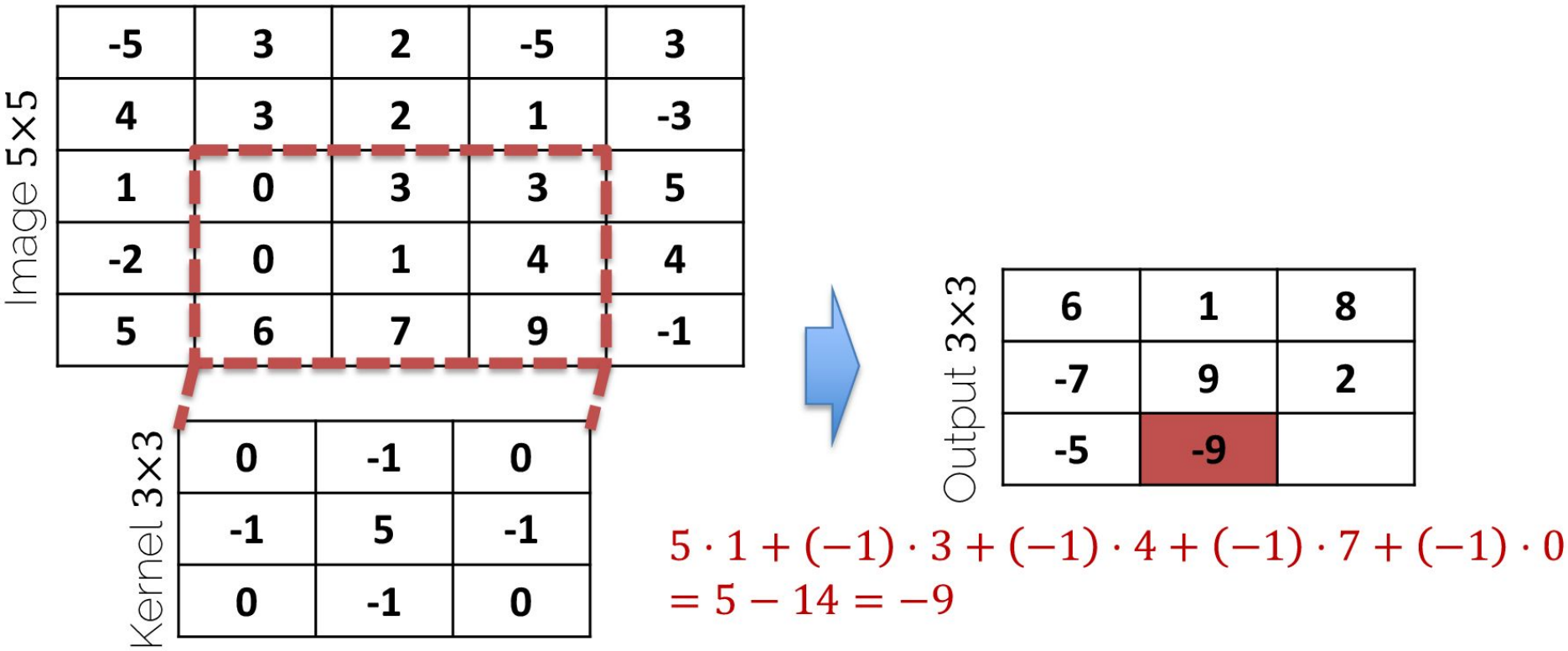
Convolutions on Images



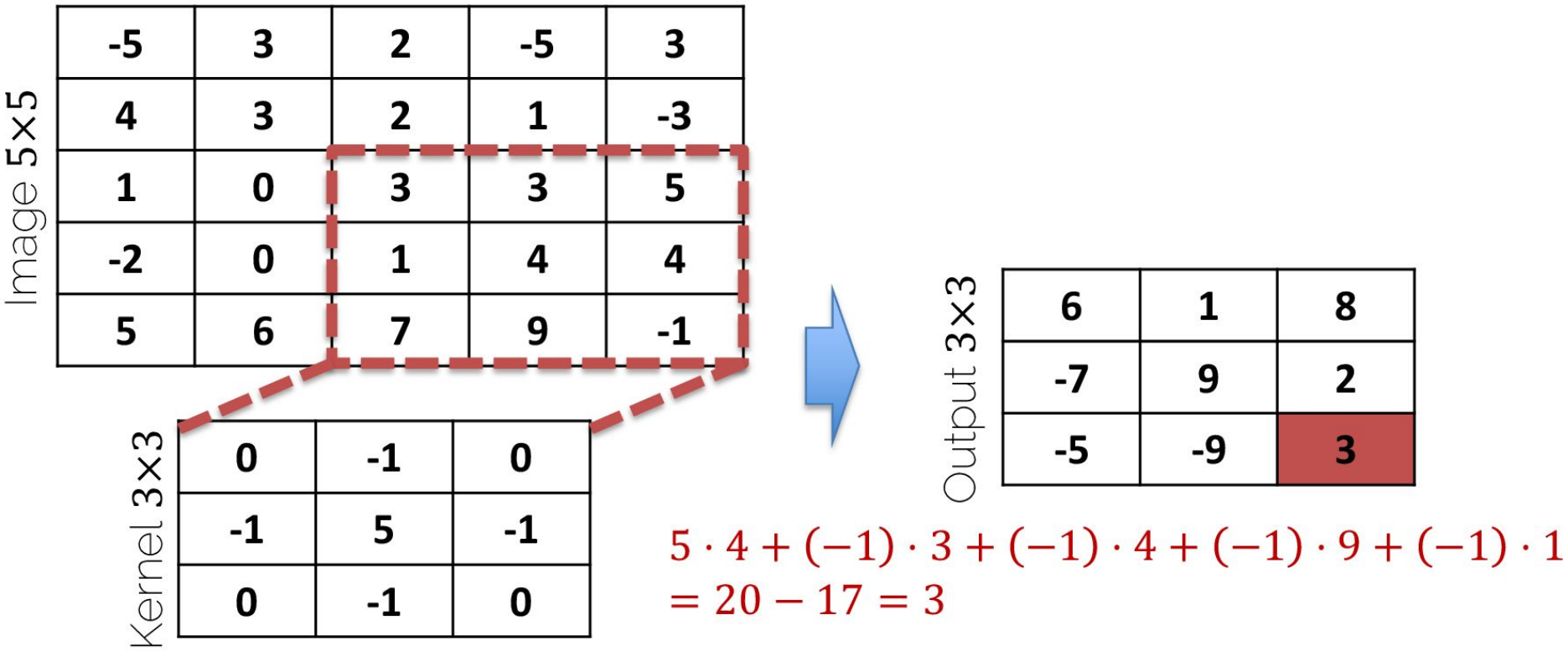
Convolutions on Images



Convolutions on Images



Convolutions on Images



Convolutions on images

$$\begin{matrix} & 0 & -1 & 0 \\ * & -1 & 5 & -1 \\ & 0 & -1 & 0 \end{matrix} =$$

Convolutions on images

$$* \begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix} =$$

sharpens the image

Convolutions on images

$$\begin{matrix} & 1 & 1 & 1 \\ * & 1 & 1 & 1 \\ & 1 & 1 & 1 \end{matrix} =$$

Convolutions on images

$$* \quad \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \quad =$$

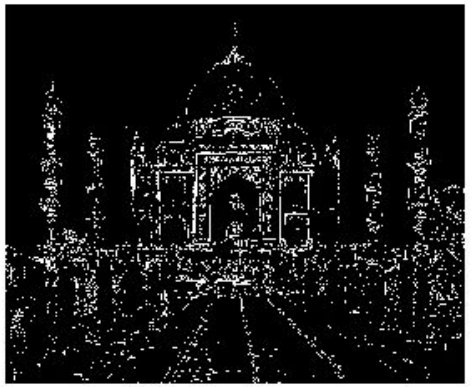
blurs the image

Convolutions on images

$$* \begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix} =$$

Convolutions on images

$$* \begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix} =$$

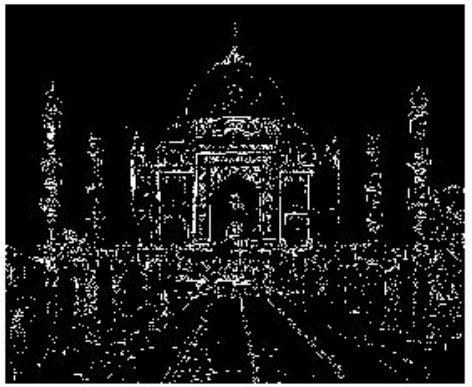


detects the edges

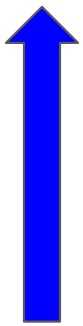
Convolutions on images

$$
\begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix}
$$

$*$ $=$



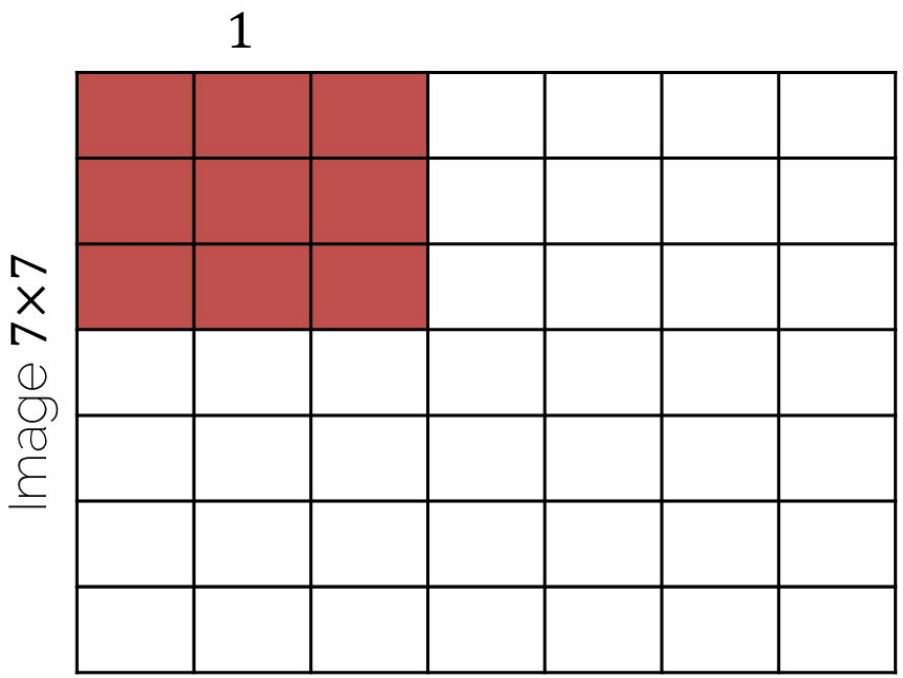
detects the edges



Aim is to learn useful kernel using DL

Feature Map Dimension

Feature Map Dimension



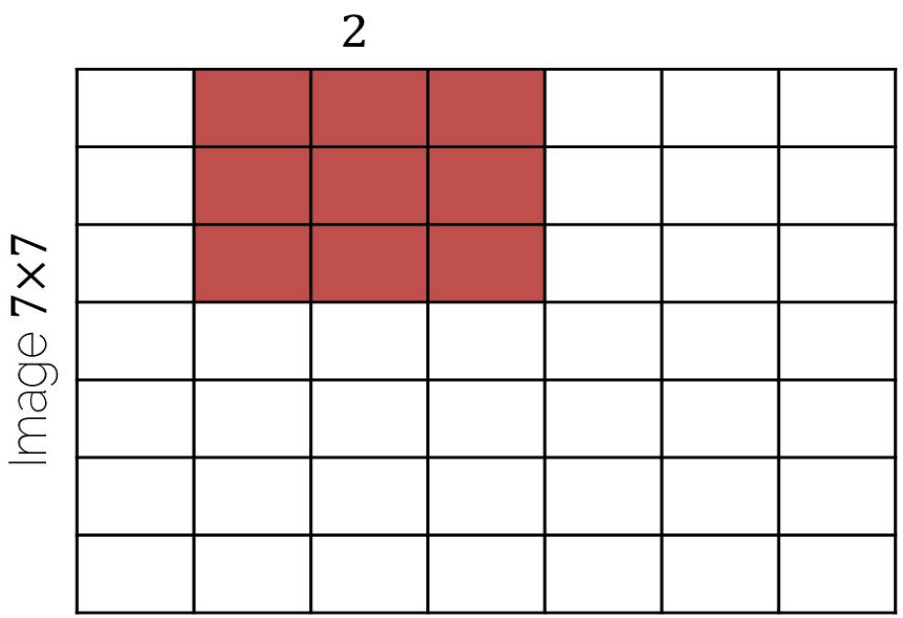
Input 7×7
Filter 3×3

$$\frac{7-3+1}{S}$$

for $S=1$

5 x 5

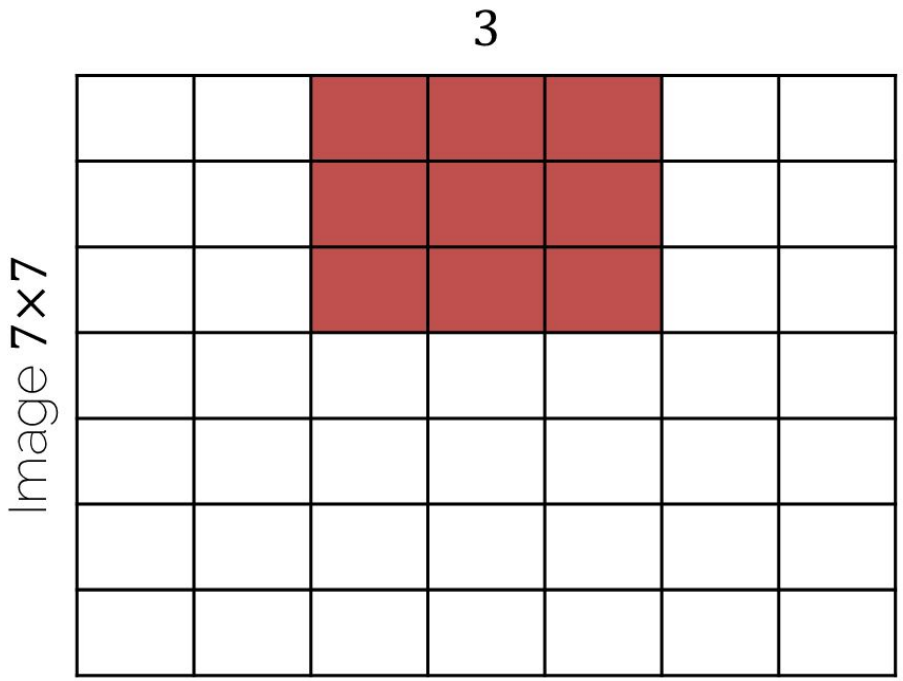
Feature Map Dimension



Input 7×7

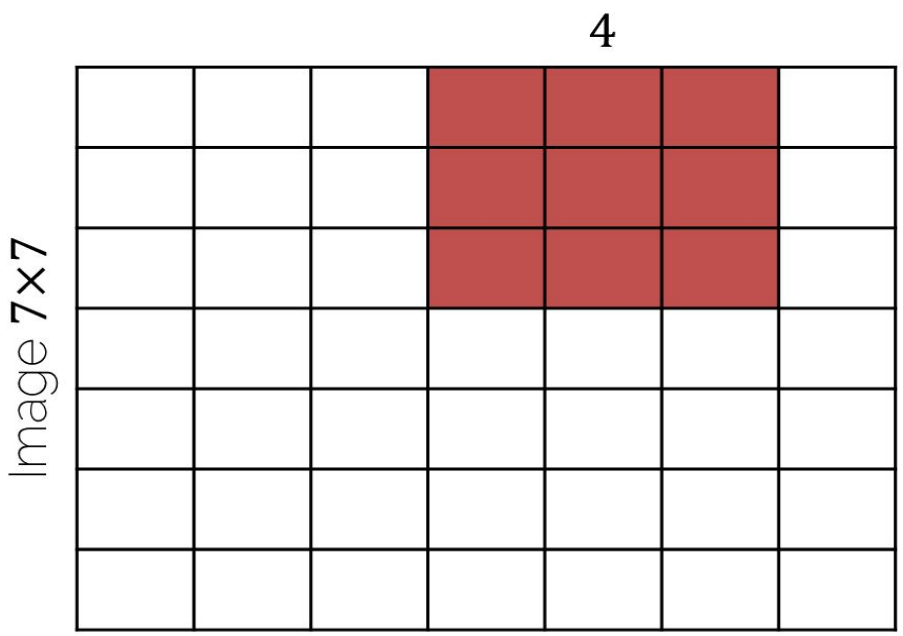
Filter 3×3

Feature Map Dimension



Input 7×7
Filter 3×3

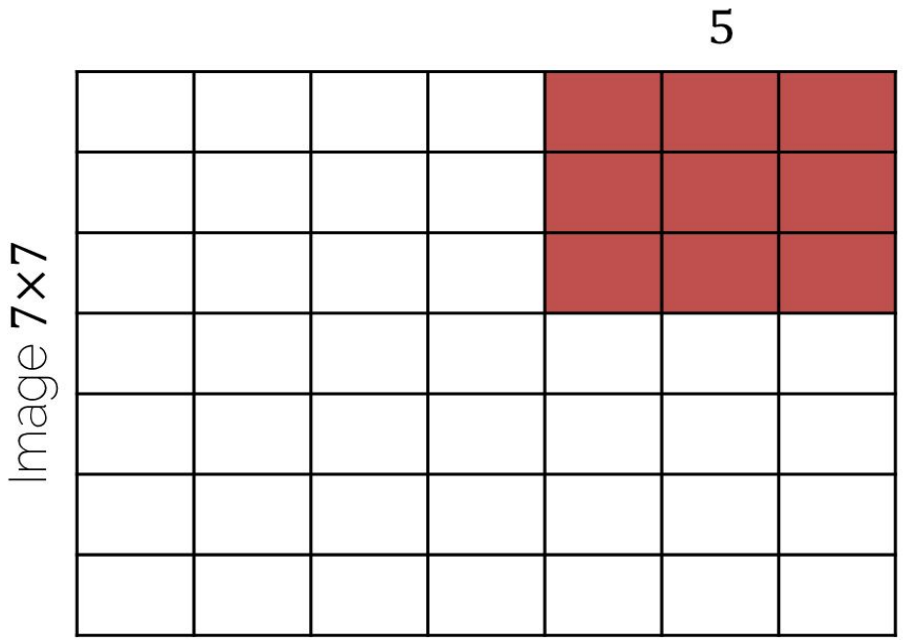
Feature Map Dimension



Input 7×7

Filter 3×3

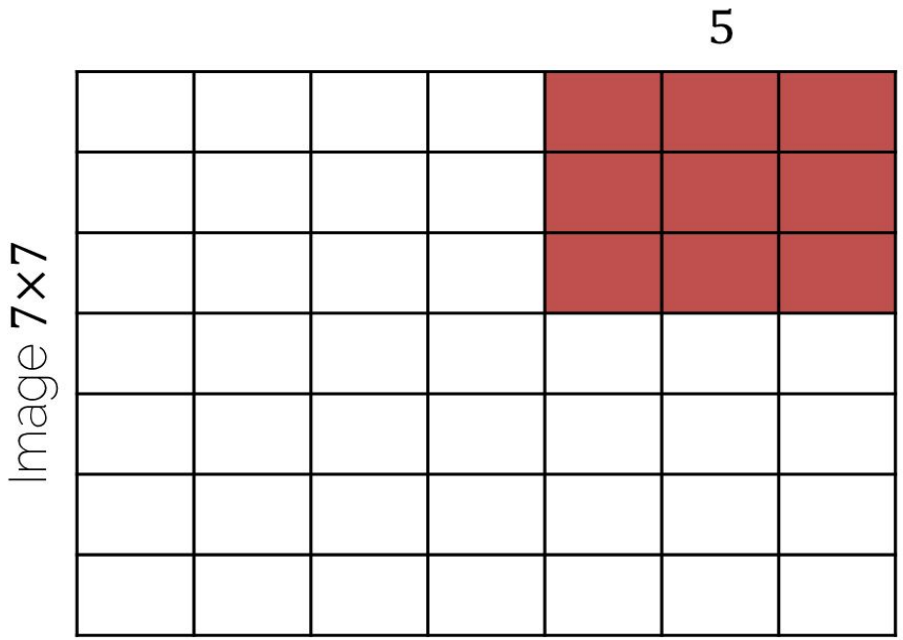
Feature Map Dimension



Input 7×7

Filter 3×3

Feature Map Dimension

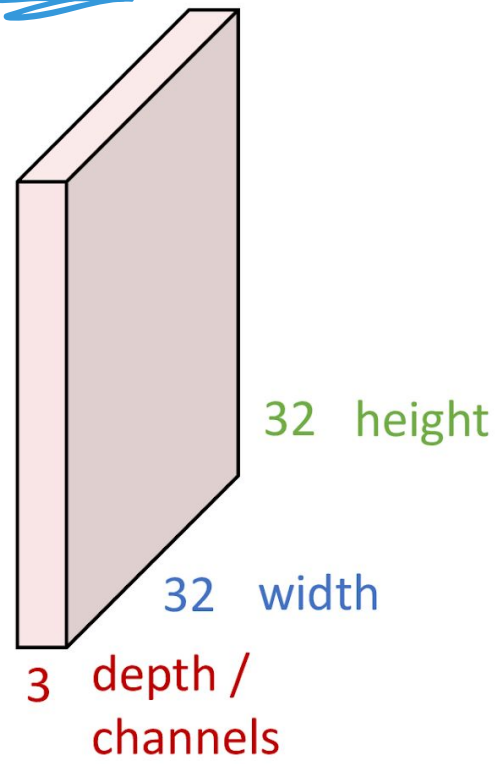


black and white image

Input	7×7
Filter	3×3
Output	5×5

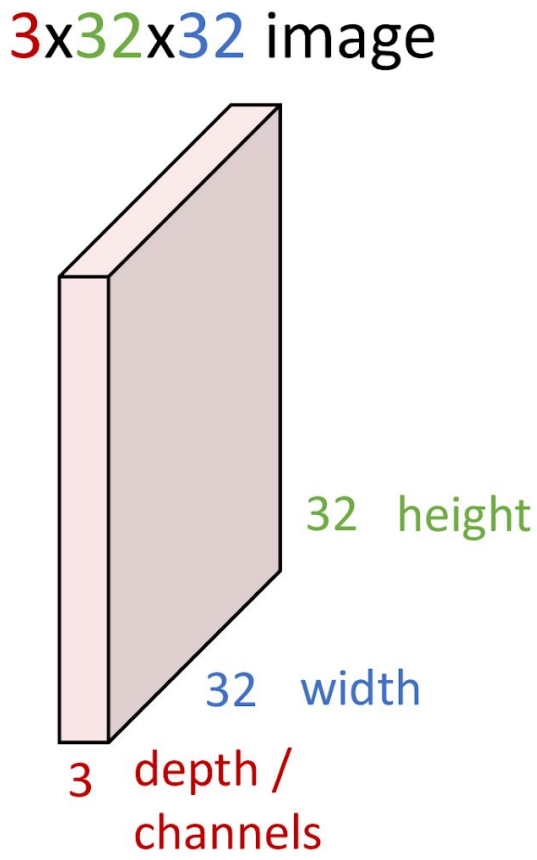
Convolution Layer

3x32x32 image: preserve spatial structure

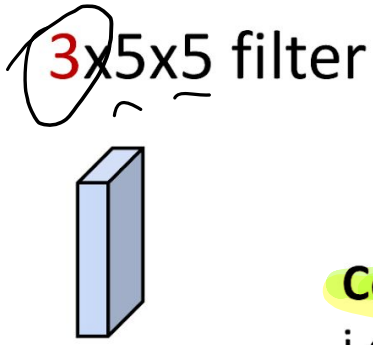


RGB image

Convolution Layer

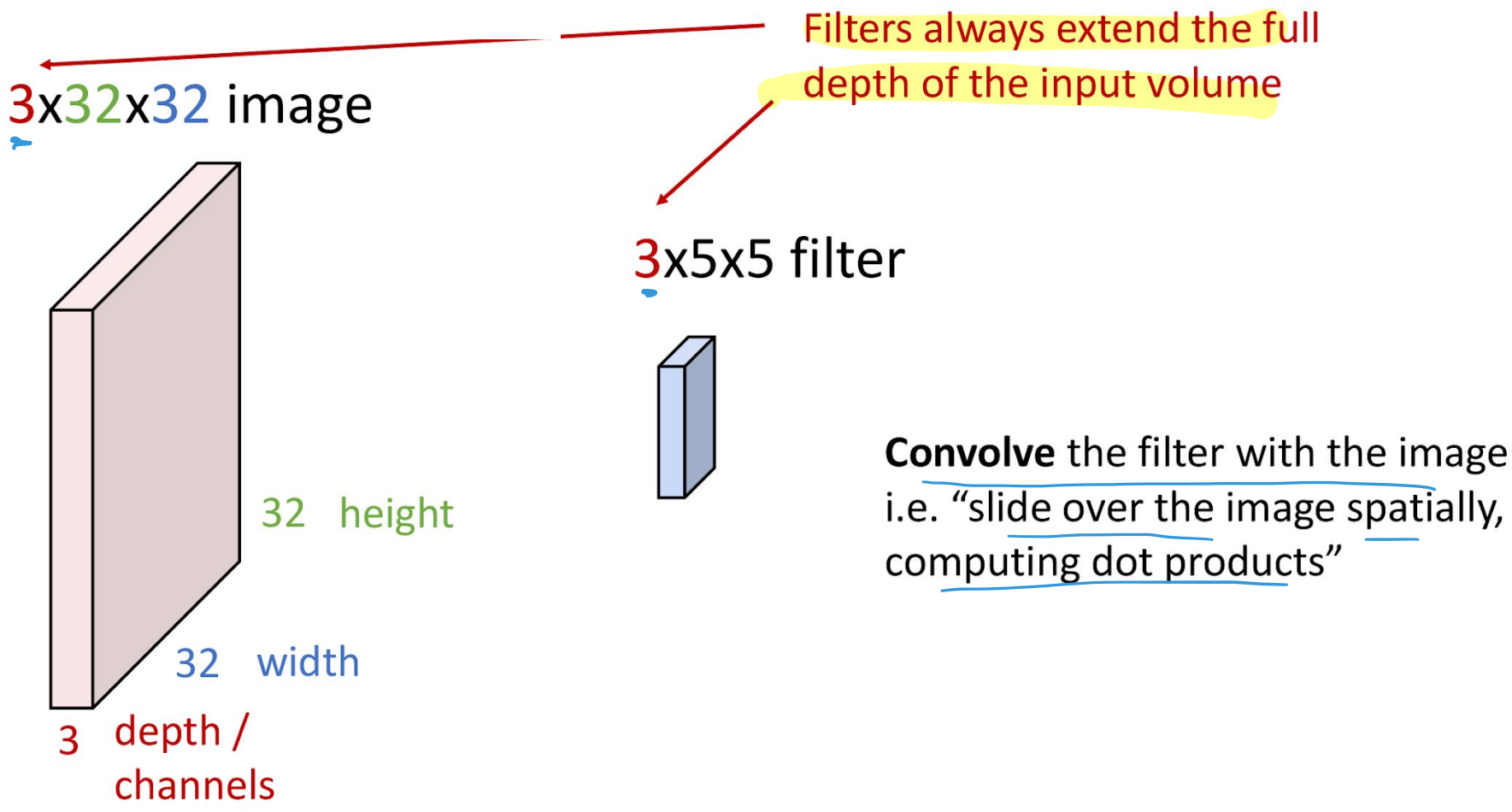


RGB image

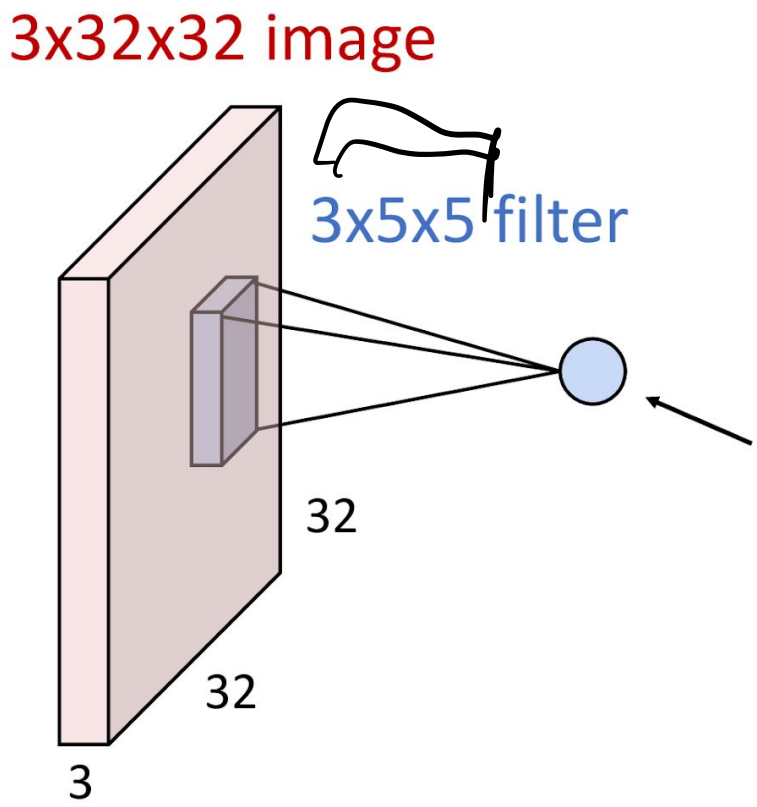


Convolve the filter with the image
i.e. "slide over the image spatially, computing dot products"

Convolution Layer



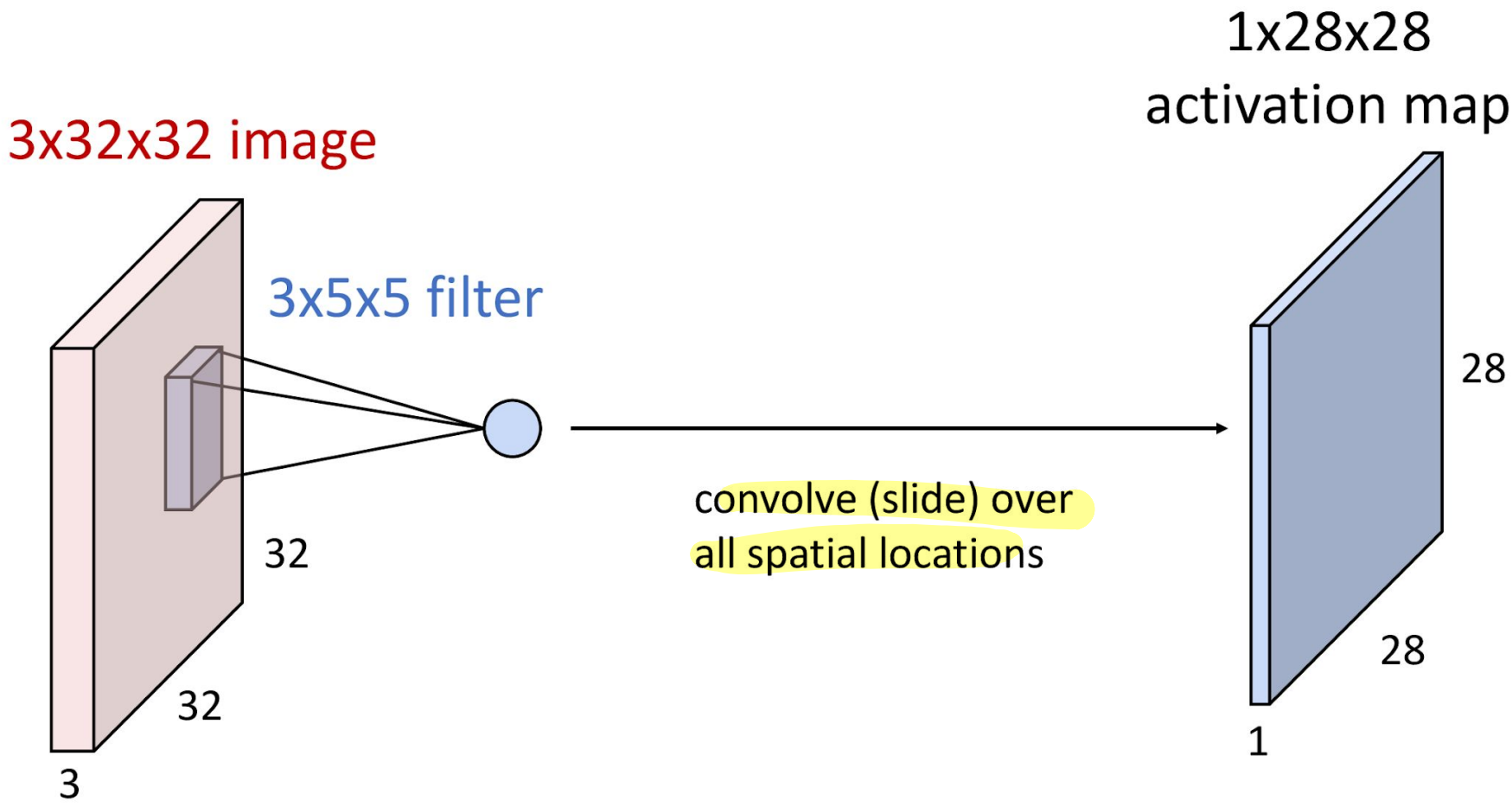
Convolution Layer



1 number:
the result of taking a dot product between the filter and a small 3x5x5 chunk of the image
(i.e. 3*5*5 = 75-dimensional dot product + bias)

RGB image

Convolution Layer

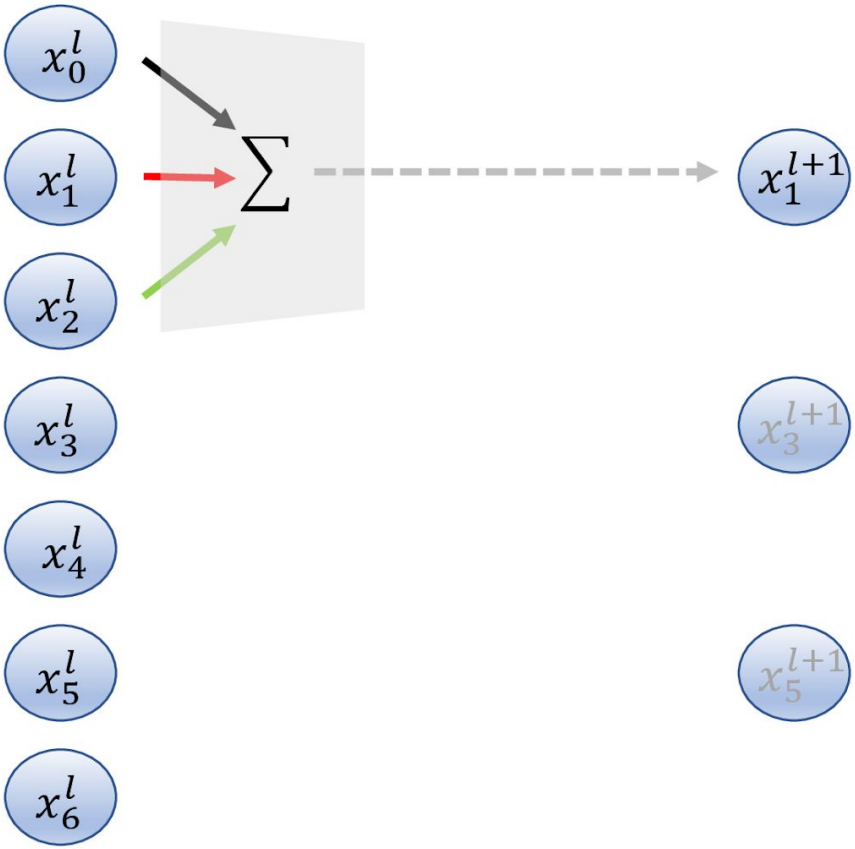


RGB image

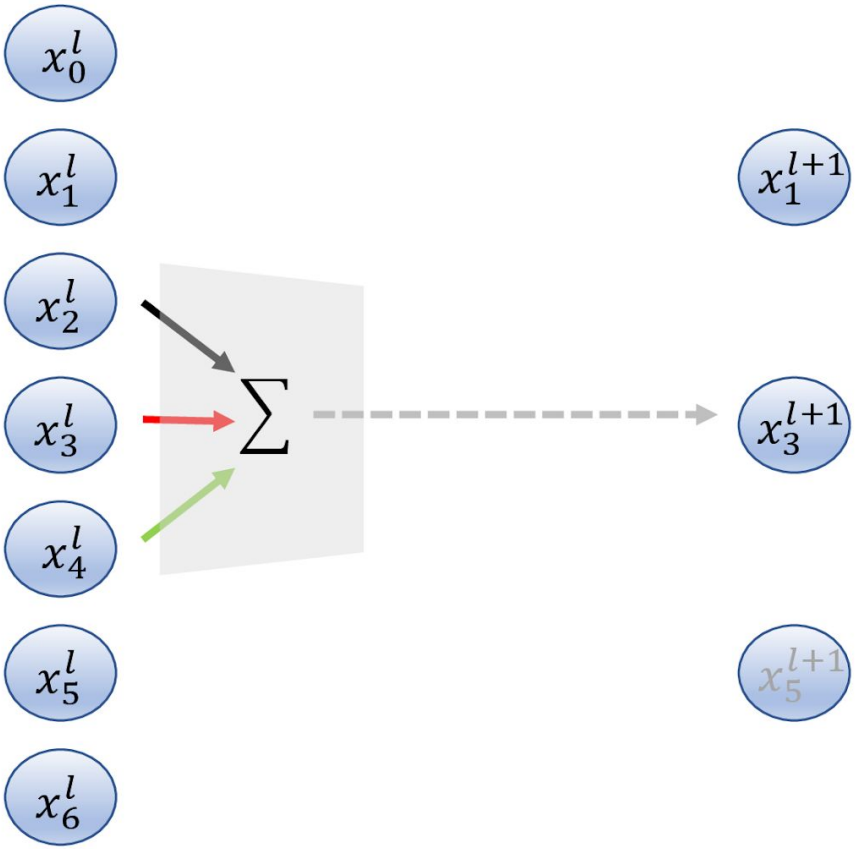
Stride

The stride determines how much the convolutional kernel is moved across the input image at each step.

Stride

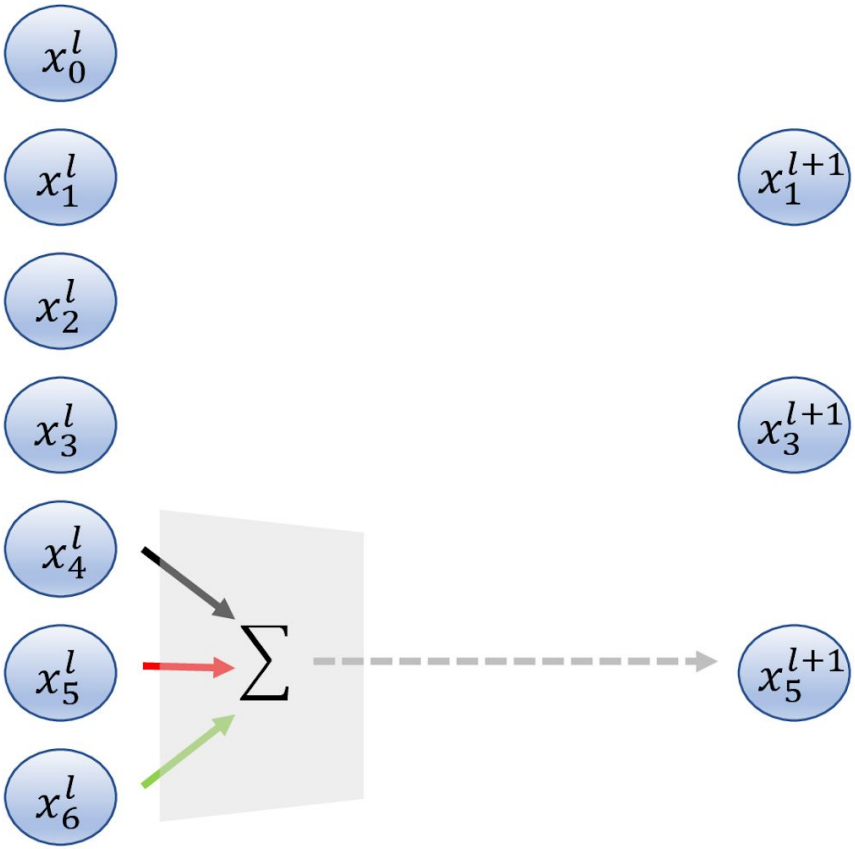


Stride



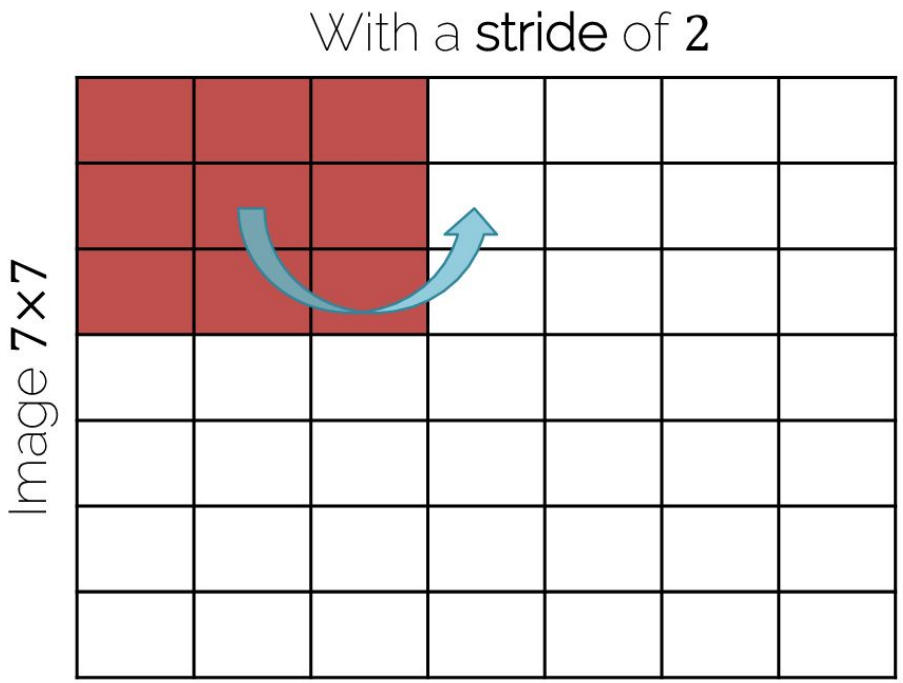
Stride 2

Stride



Stride 2

Convolution on images



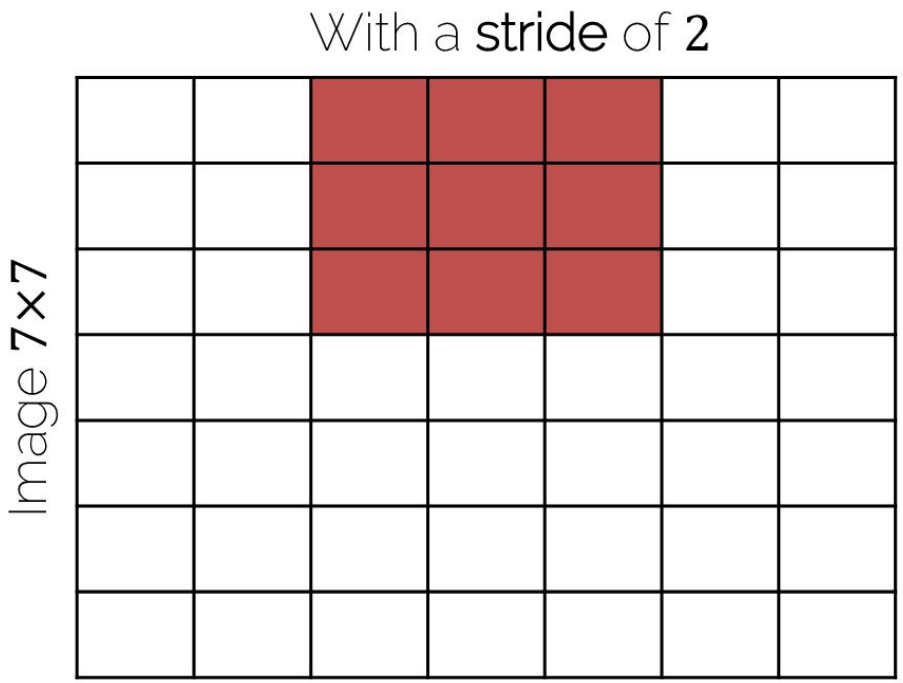
Input 7×7

Filter 3×3

Stride 2

Output

Convolution on images



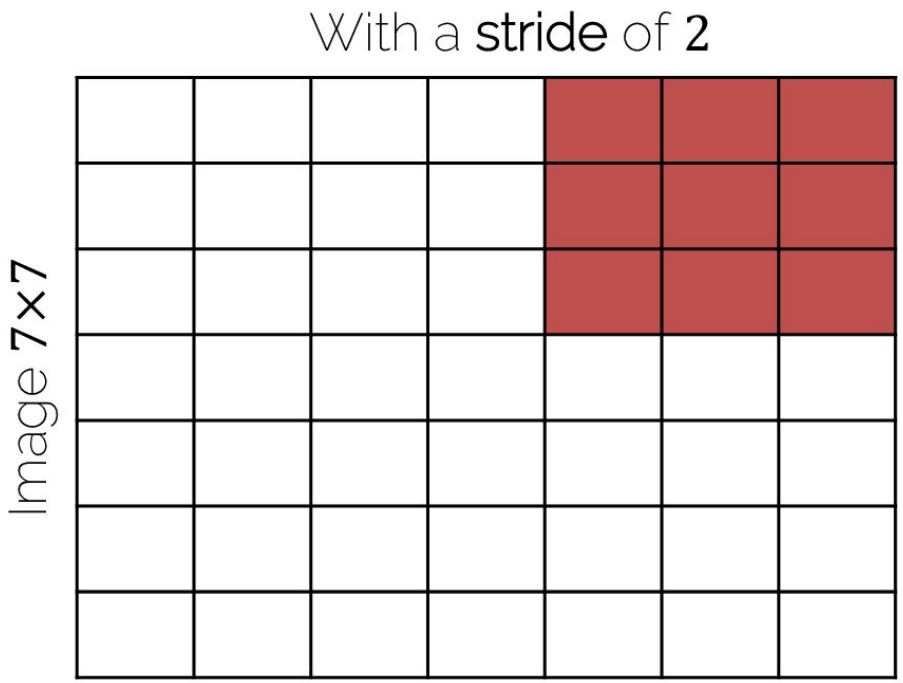
Input 7×7

Filter 3×3

Stride 2

Output

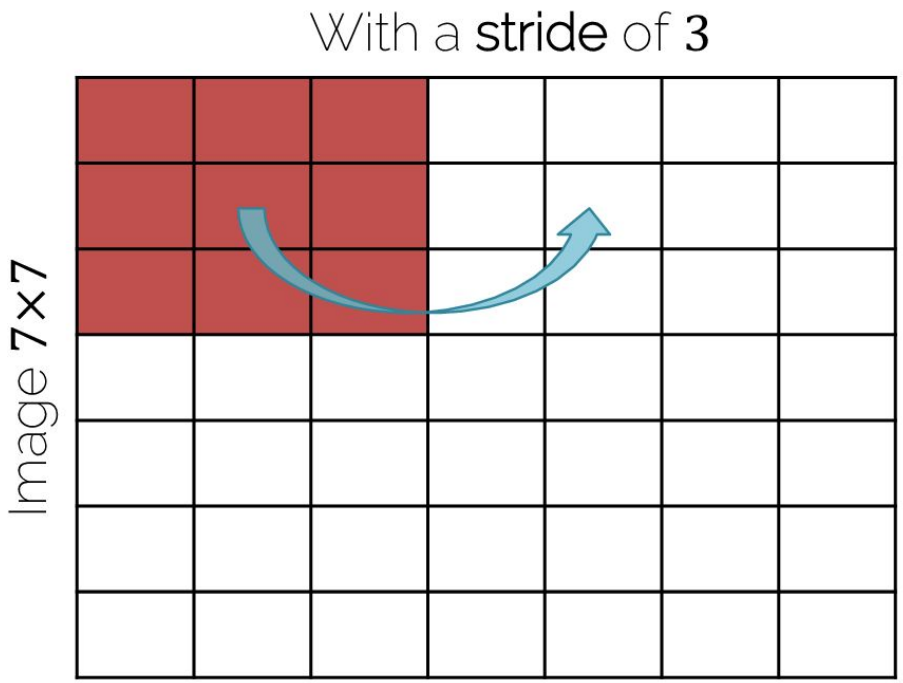
Convolution on images



Input	7×7
Filter	3×3
Stride	2
Output	3×3

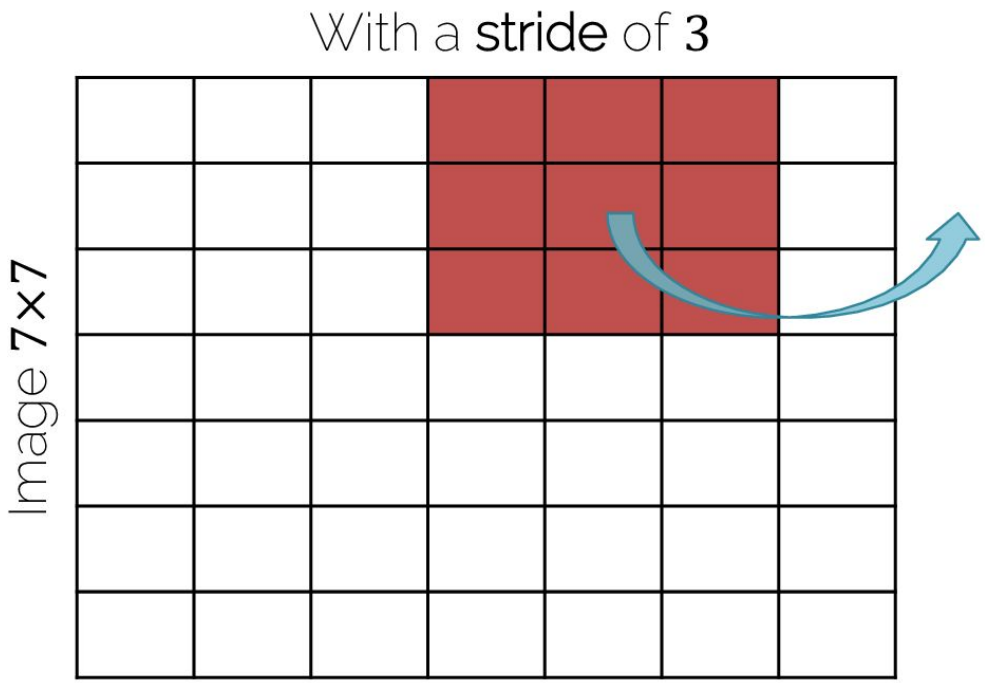
$$\frac{7-3}{2}+1$$

Feature Map Dimension



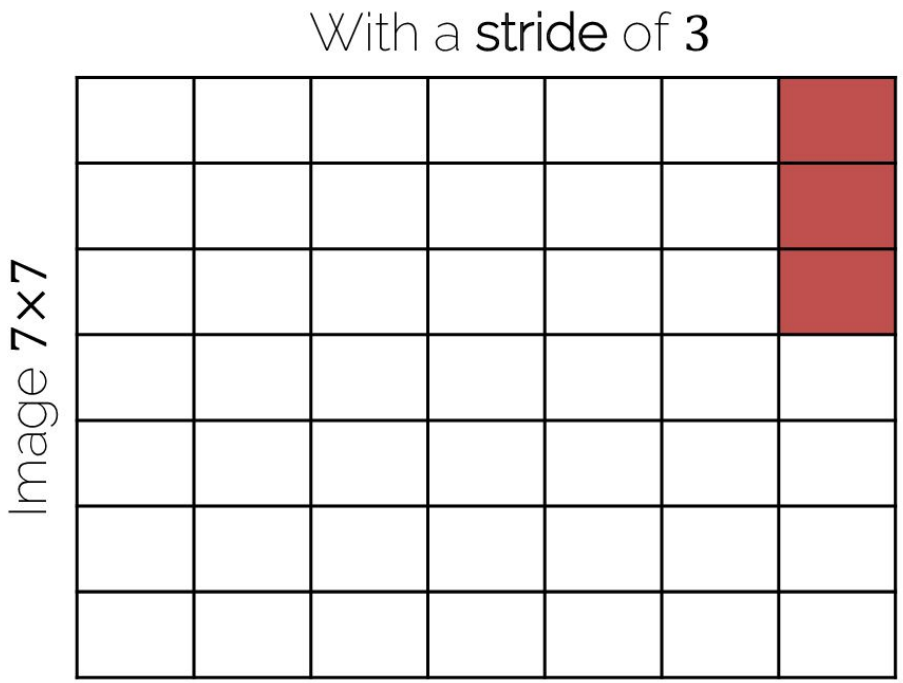
Input	7×7
Filter	3×3
Stride	3
Output	$? \times ?$

Feature Map Dimension



Input	7×7
Filter	3×3
Stride	3
Output	$? \times ?$

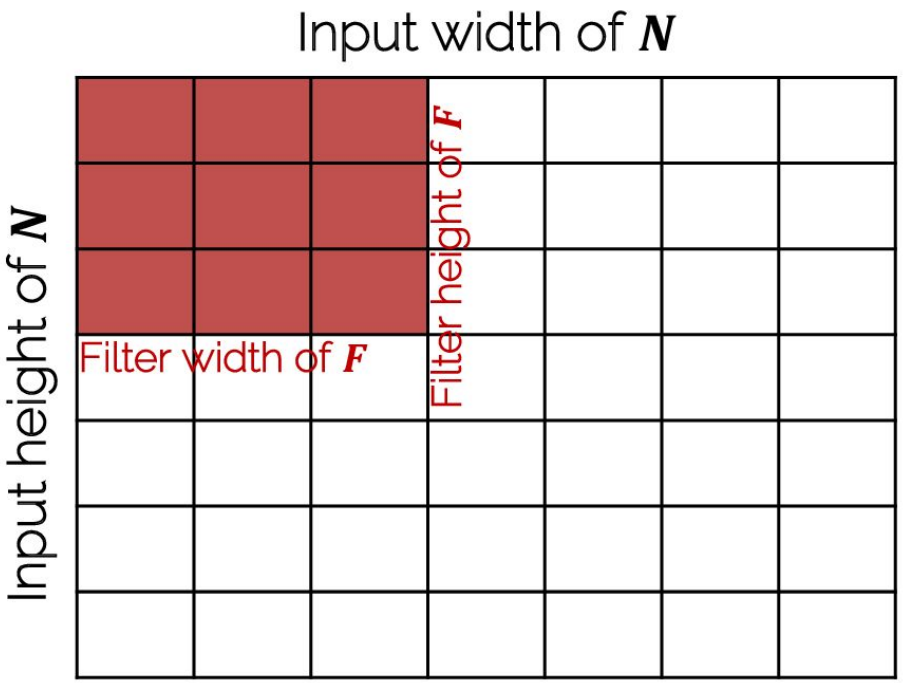
Feature Map Dimension



Input	7×7
Filter	3×3
Stride	3
Output	$? \times ?$

Does not fit and
illegal operation

Feature Map Dimension



Input	$N \times N$
Filter	$F \times F$
Stride	S
Output	$\left(\frac{N-F}{S}+1\right) \times \left(\frac{N-F}{S}+1\right)$

$N = 7, F = 3, S = 1$: $\frac{7-3}{1}+1 = 5$

$N = 7, F = 3, S = 2$: $\frac{7-3}{2}+1 = 3$

$N = 7, F = 3, S = 3$: $\frac{7-3}{3}+1 = 2.\overline{3}$

Fractions are illegal

Stride

The advantages of using stride include:

- **Stride Reduce the size of the output feature map**, which can help to reduce the computational cost and memory requirements of the network.
- Striding can make the network more efficient by **reducing the number of operations** required to process the input data.
- Striding can help to **reduce overfitting** by reducing the number of parameters in the network and forcing the network to learn more abstract features.