MACHINE LEARNING

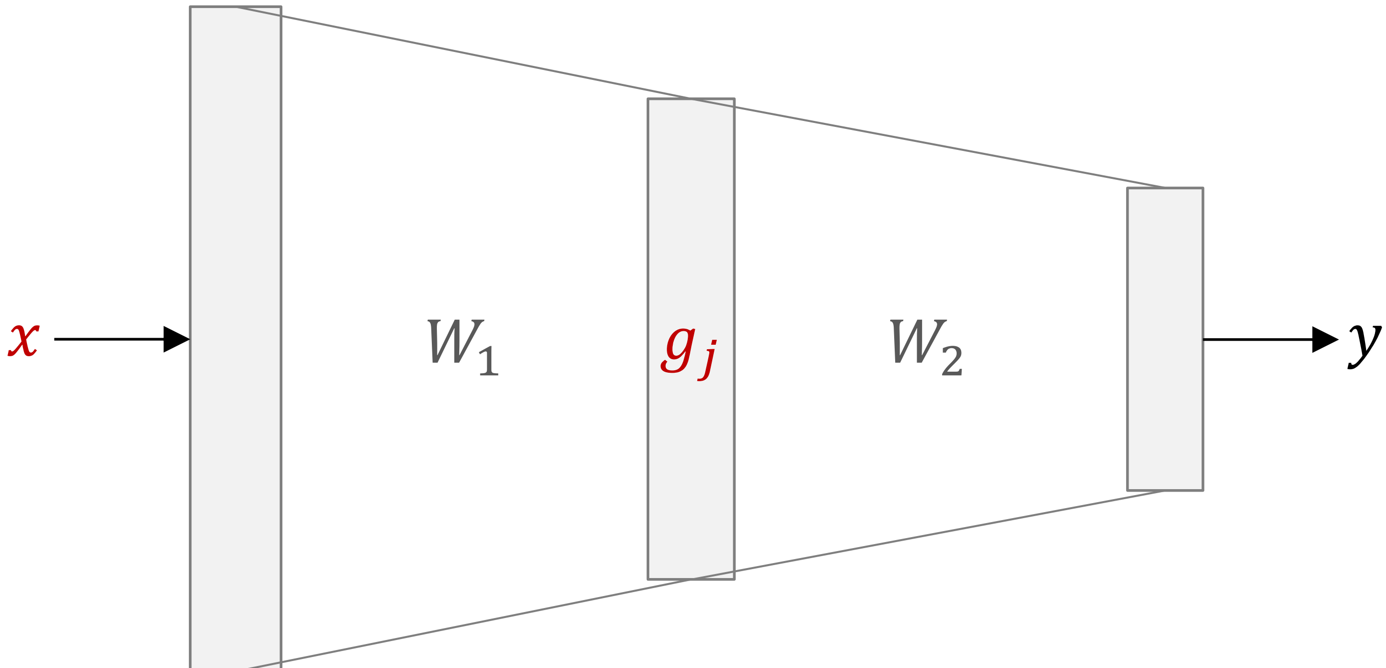
University Master's Degree in Intelligent Systems

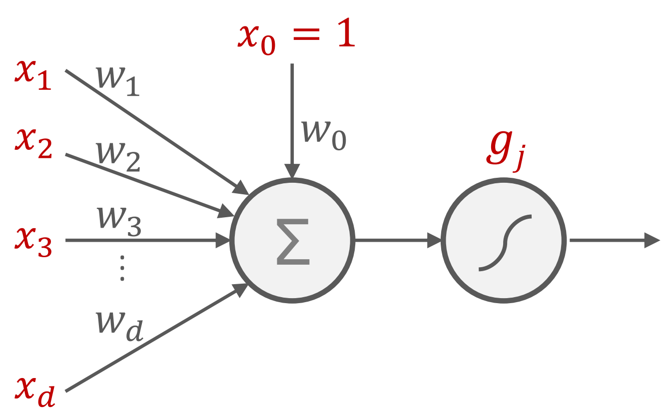
convolutional neural networks

Ramón A. Mollineda Cárdenas

what we already know

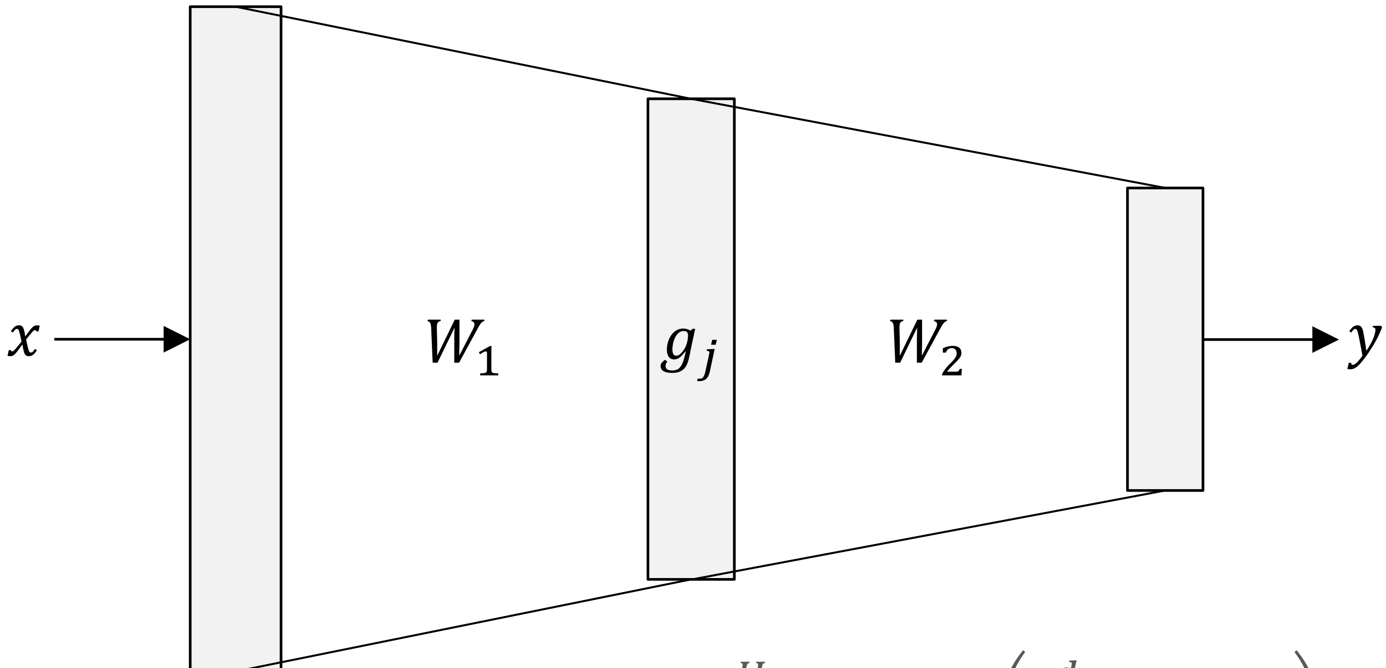
fully-connected neural network





what we already know

fully-connected neural network



$$y_k = \sum_{j=1}^{H} w_2^{jk} \cdot g\left(\sum_{i=1}^{d} w_1^{ij} \cdot x_i\right) = W_2 \cdot g(W_1 \cdot x)$$

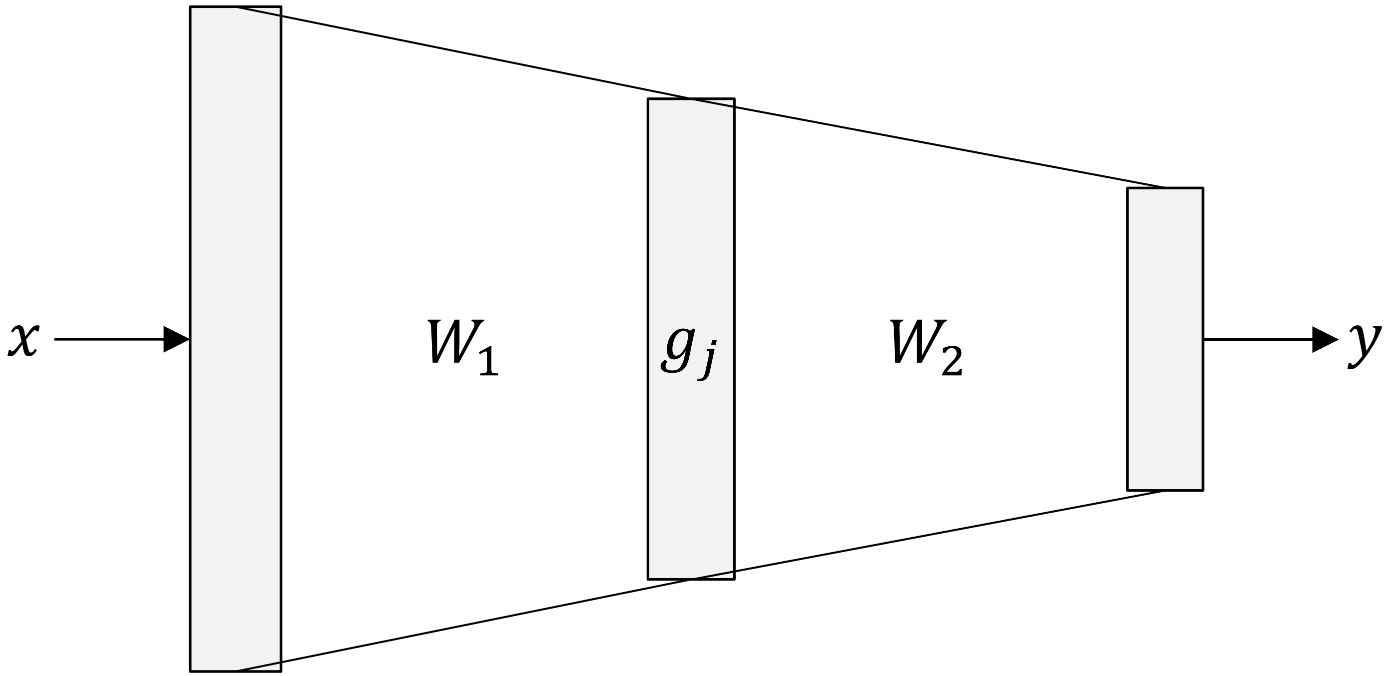
d, input space dimension

H, number of hidden layer units

g, nonlinear activation function

what we already know

fully-connected neural network

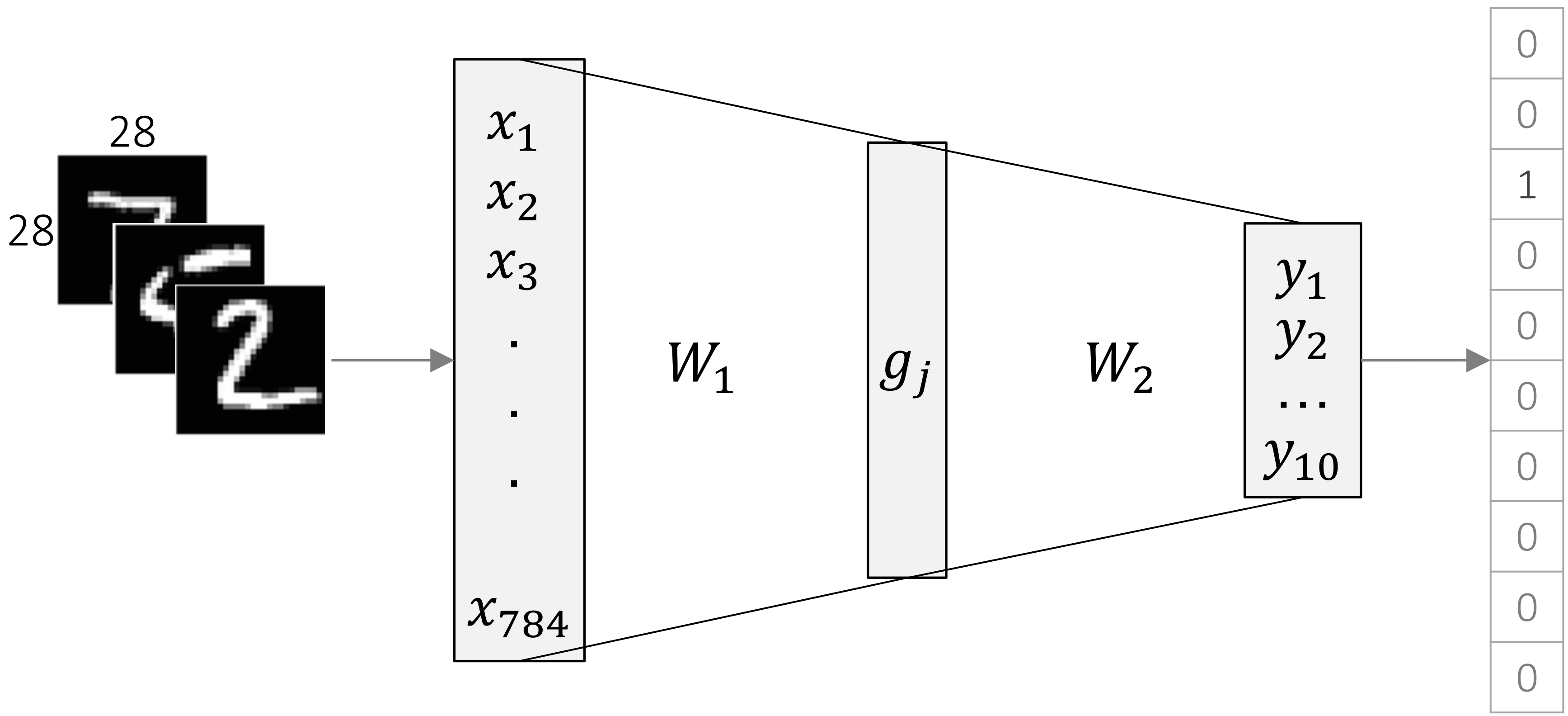


- input: $x \in \mathbb{R}^N$
- two-layer neural network
- g non-linear functions (e.g. sigmoide, relu)
- output: linear combination of g_j

what we already know

fully-connected neural network

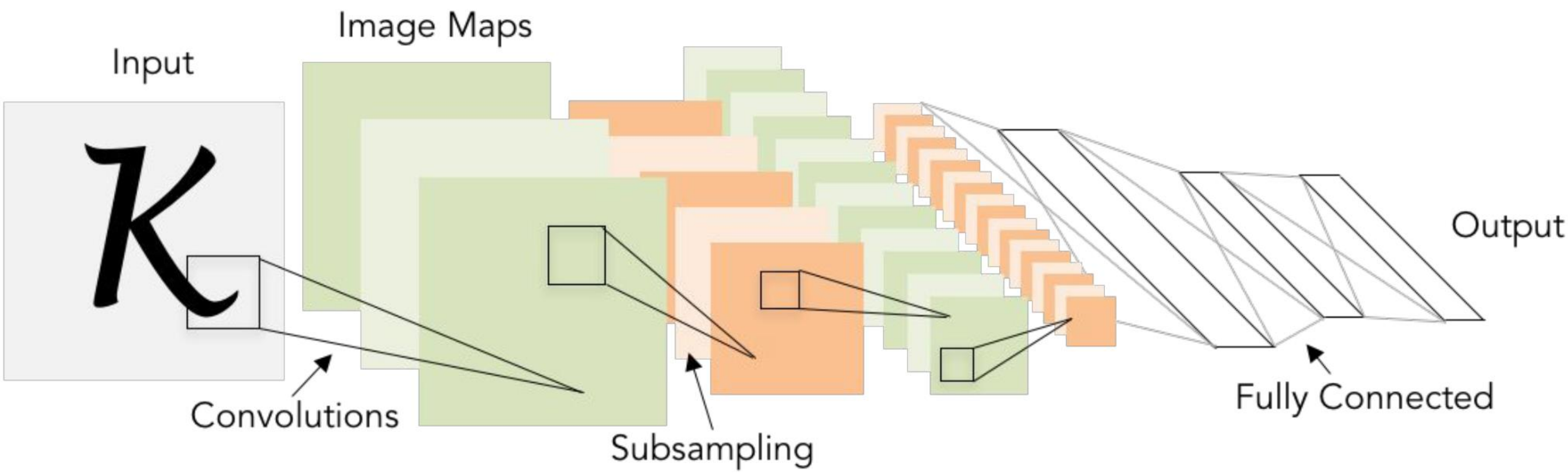
MNIST digits case study



convolutional neural network (CNN)

overview

CNN: end-to-end solutions transforms primary representations into categories

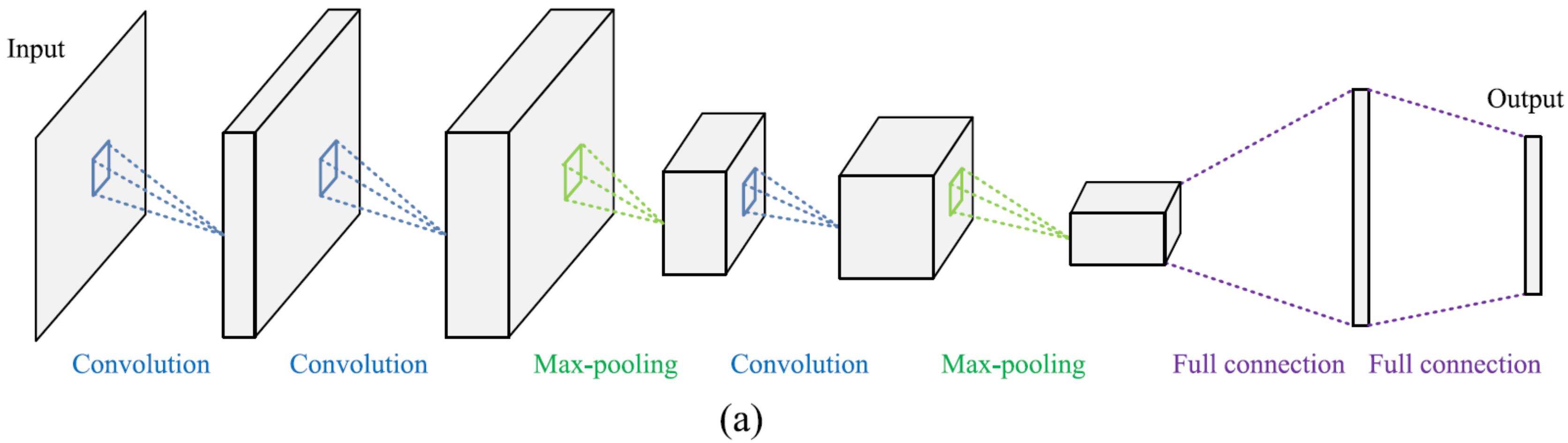


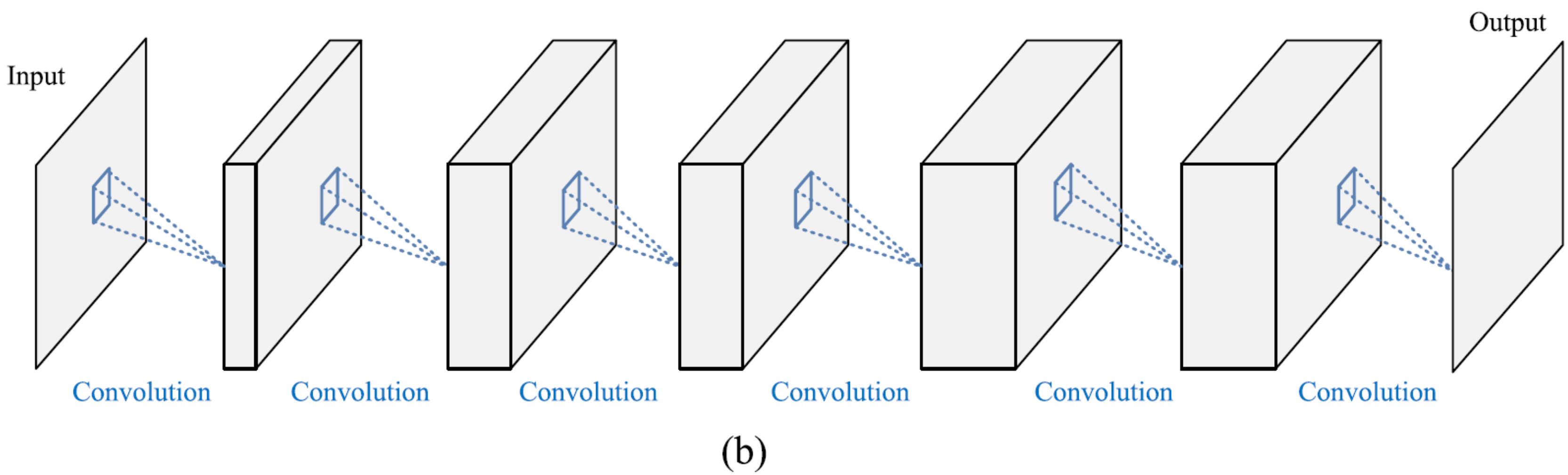
low level features (edges, spots, details) higher spatial resolution, lower depth → high level features (patterns, abstractions) lower spatial resolution, lower depth

Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

convolutional neural network (CNN)

architectural patterns

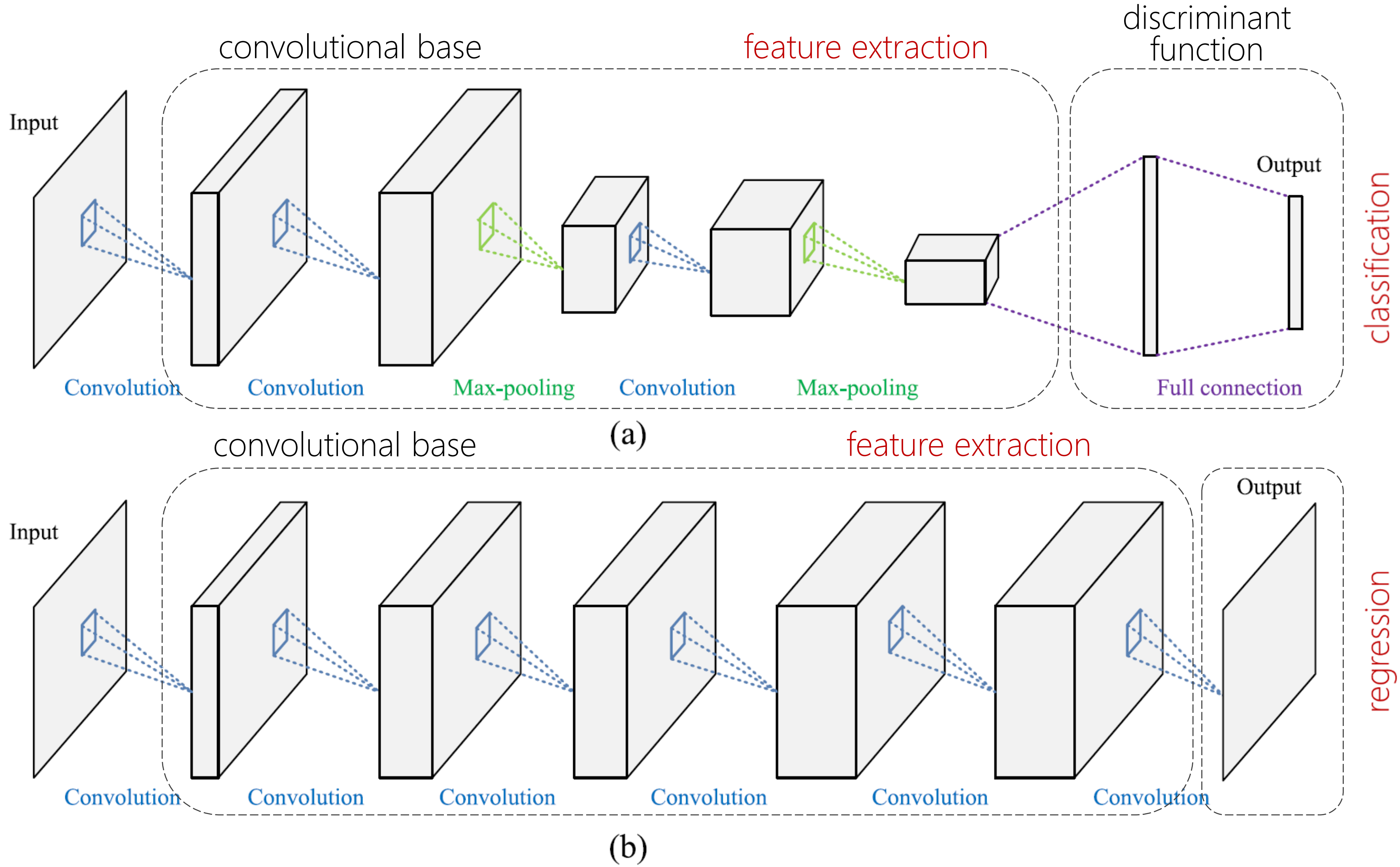




(a) classification (b) regression (image reconstruction)

convolutional neural network (CNN)

architectural patterns

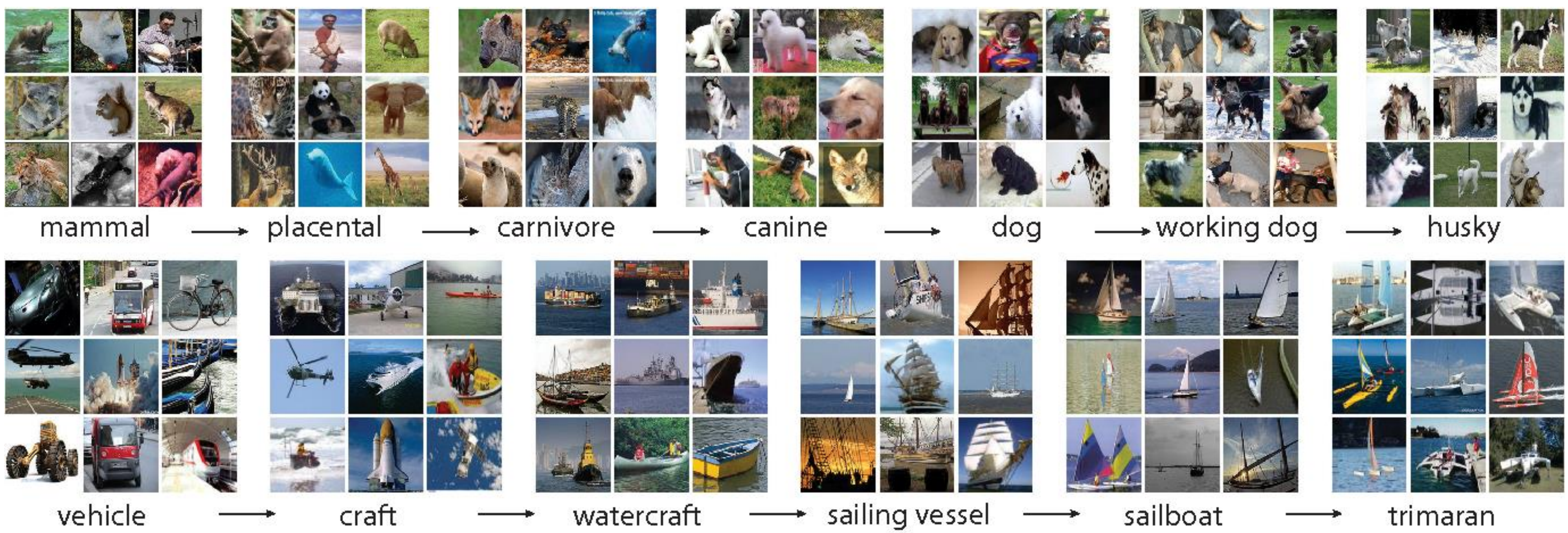


(a) classification (b) regression (image reconstruction)

ImageNet challenge

Large Scale Visual Recognition Challenge (ILSVRC)

IMAGENET (*http://image-net.org*)



Deng, Jia et al. "ImageNet: A large-scale hierarchical image database." 2009 IEEE Conference on Computer Vision and Pattern Recognition (2009): 248-255.

ImageNet challenge

Large Scale Visual Recognition Challenge (ILSVRC)

task: given an image, identify the main object in the image

training data:

- 1,200,000 labeled images
- 1,000 final classes/categories (ground truth)
- one class label per image (identifies the main object)

validation and test data:

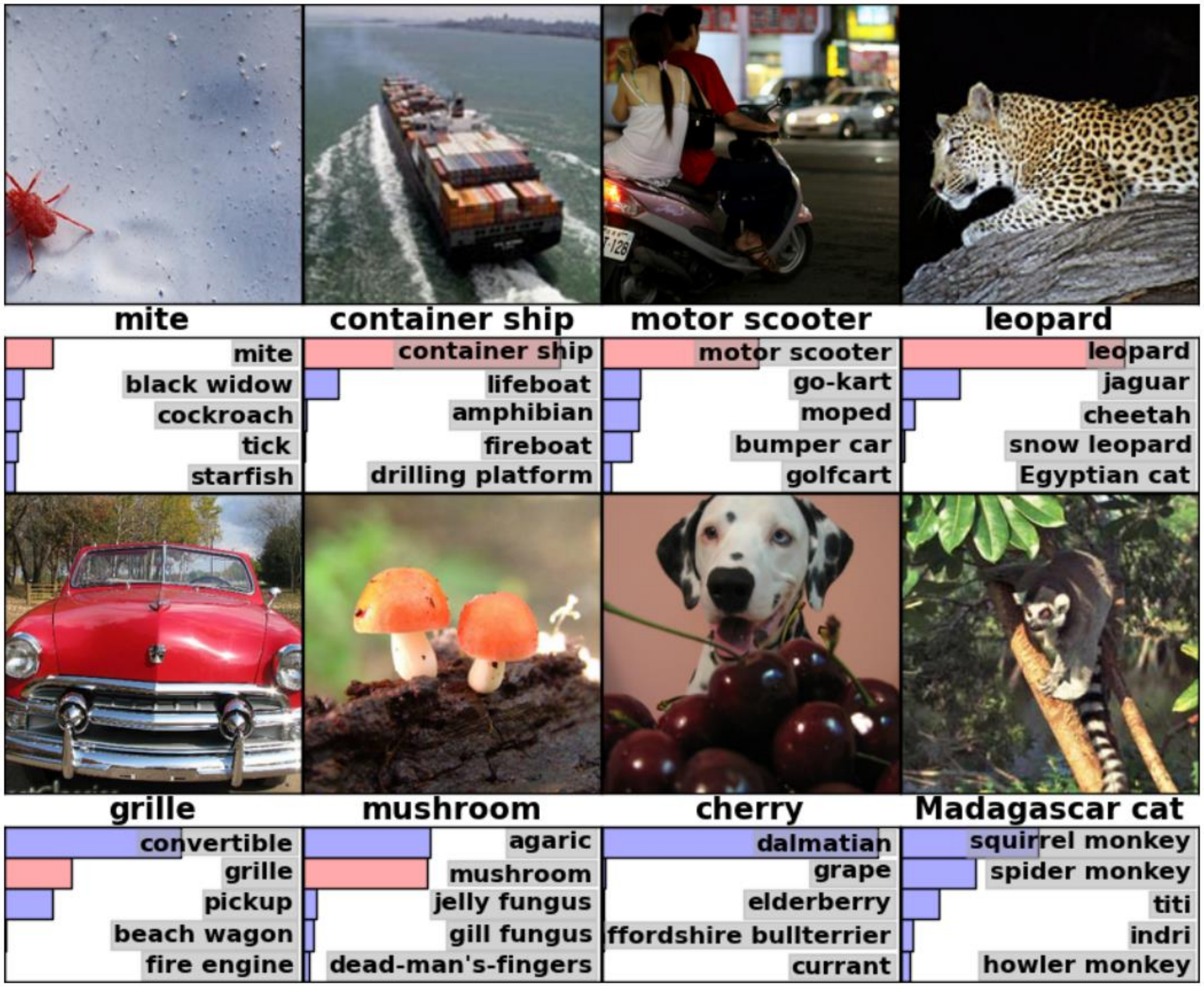
- 150,000 pictures (from Flickr and other search engines)
- 50,000 labeled validation images
- 100,000 unlabeled test data

success/hit: the correct class (ground truth) is one of the 5 most probable classes found by the model (top-5 error)

ImageNet challenge

Large Scale Visual Recognition Challenge (ILSVRC)

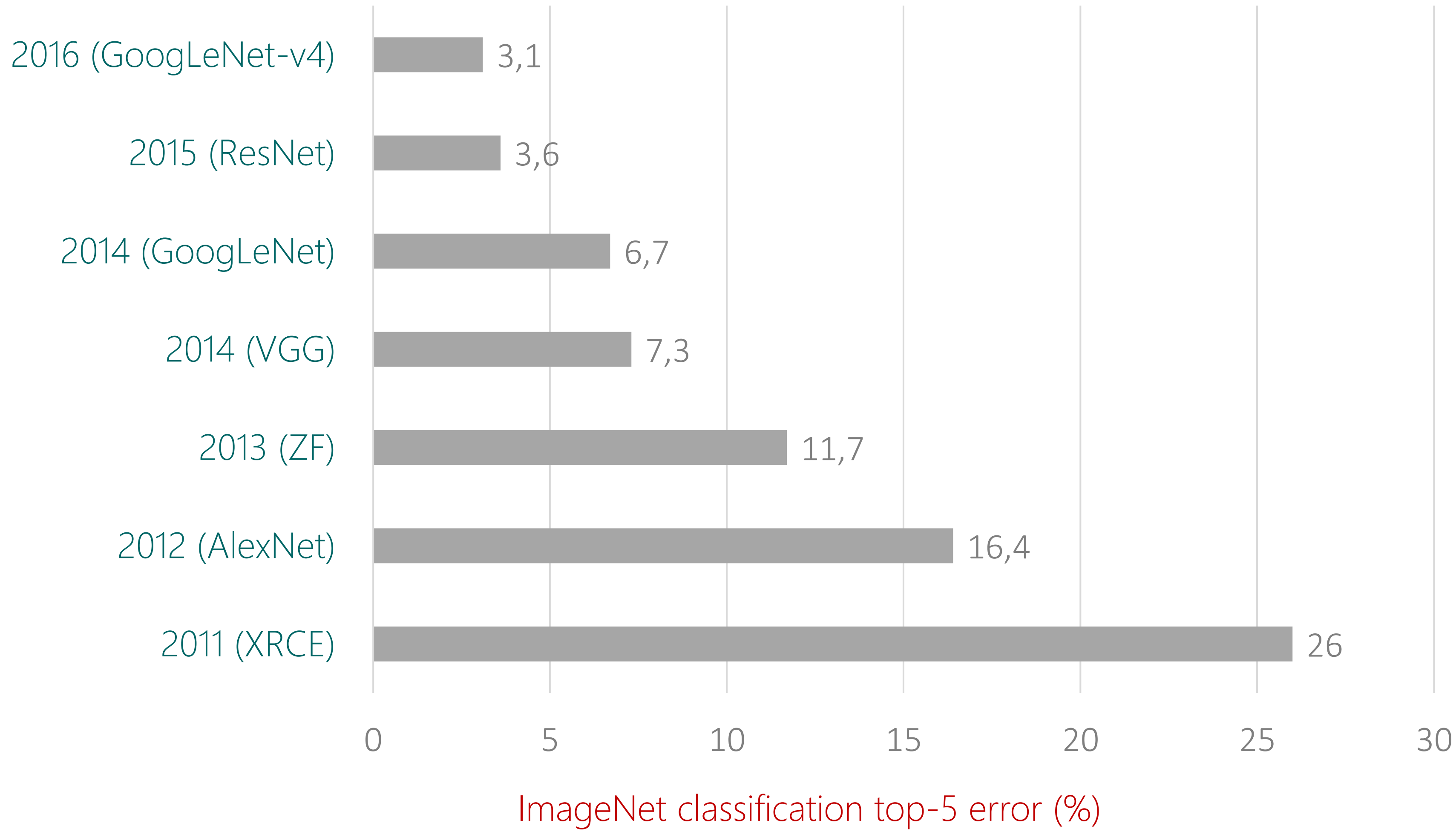
classification (top-5 error)



Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

ImageNet challenge

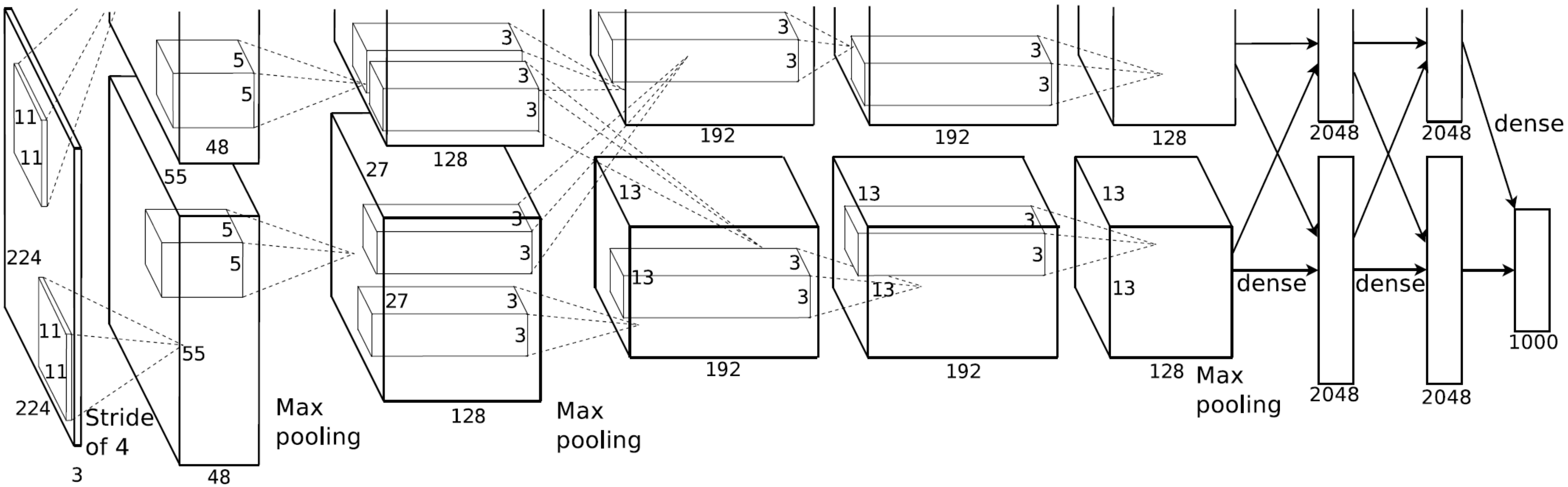
winning models



ImageNet challenge

Large Scale Visual Recognition Challenge (ILSVRC)

AlexNet (winning model in 2012)



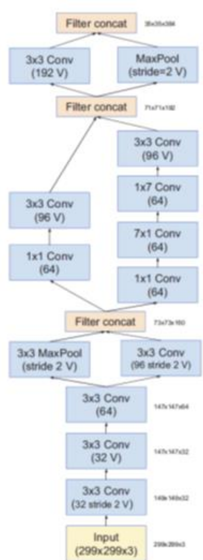
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012 (ver).

top-1 error: 38,1%
top-5 error: 16,4%
layers: 8
¡60 millones de parámetros!

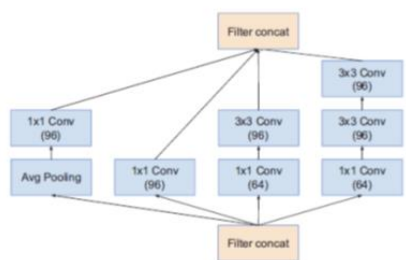
ImageNet challenge

Large Scale Visual Recognition Challenge (ILSVRC)

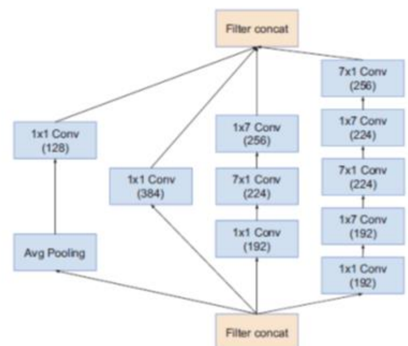
Inception v4 (winning model in 2016)



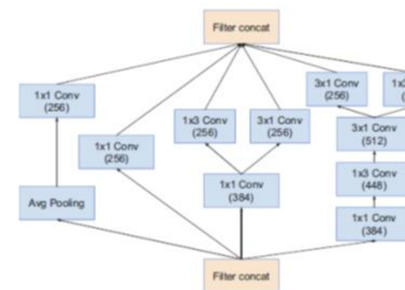
stem



inception A



inception B



inception C

top-1 error: 16,4%
top-5 error: 3,1%
layers ≈ 170 (trainable)
¡43 million parameters!

Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." AAAI. Vol. 4. 2017 ([ver](#)).

convolution

Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

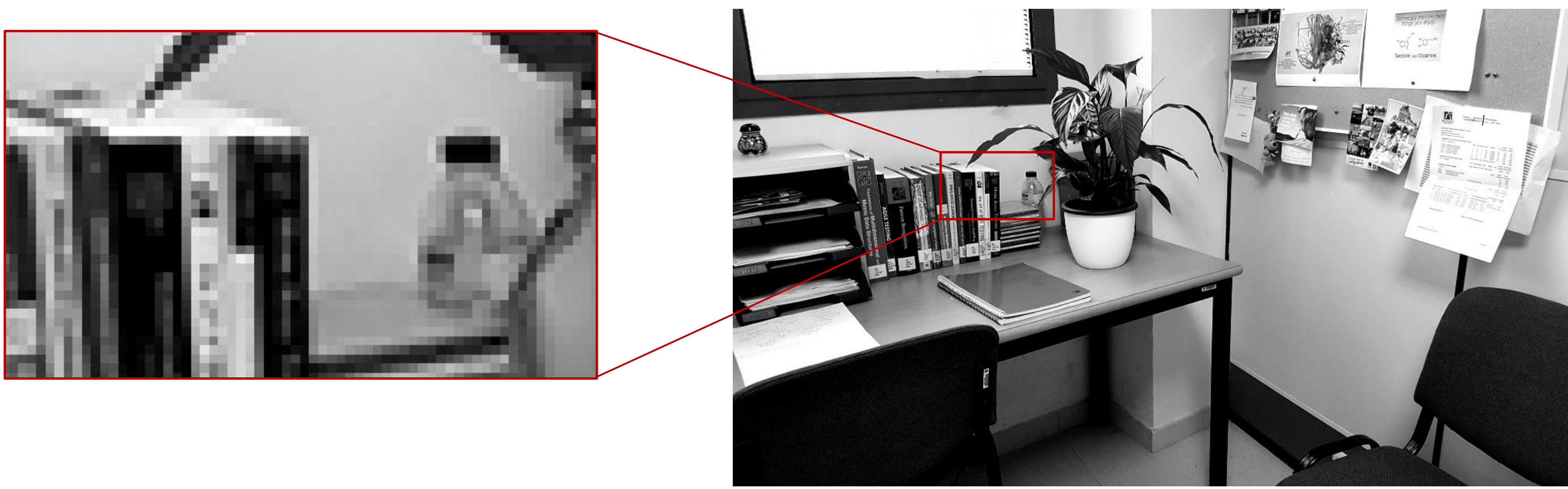
Ian Goodfellow, Yoshua Bengio, Aaron Courville. ***Deep Learning****. MIT Press (2016).*
Disponible en http://www.deeplearningbook.org/.

convolution

digital representation of images

grayscale images

- array of elements called image pixels
- matrix dimensions are called the image resolution
- each cell/element/pixel stores a value: its intensity or gray level
- intensity/gray levels take integer values between 0 and 255
- dark values are close to 0 (black); light greys, close to 255 (white)

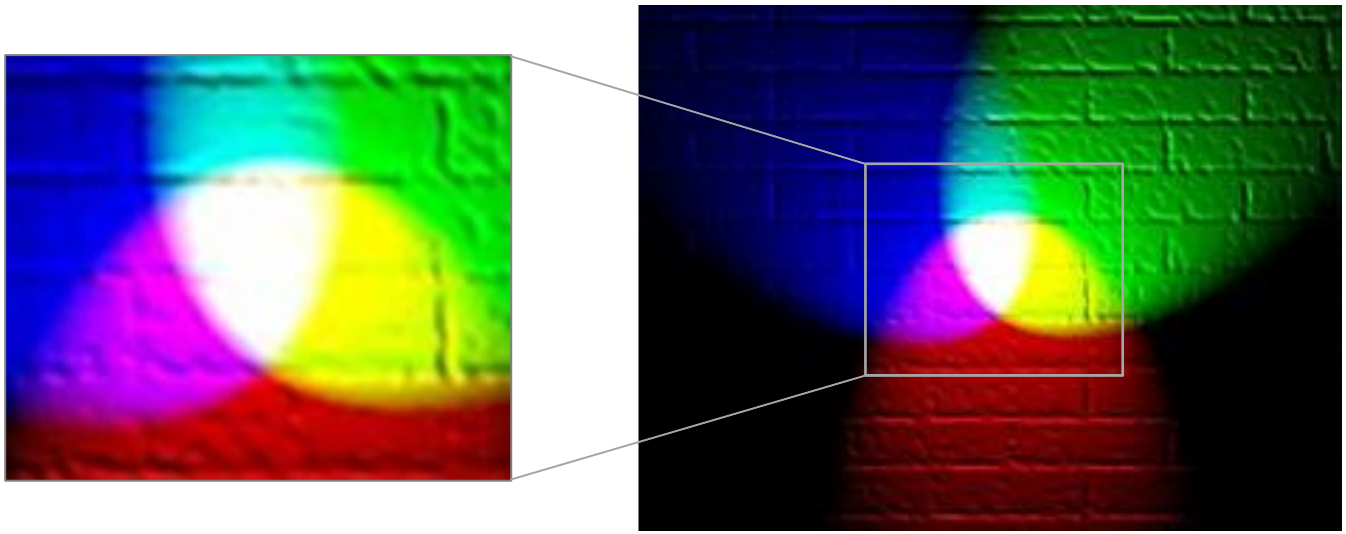


convolution

digital representation of images

color images

- array of elements called image pixels
- matrix dimensions are called the image resolution
- each pixel stores its intensity in 3 channels/values: Red, Green, Blue
- each R, G or B intensity takes value between 0 and 255
- each final color is the result of combining the R, G or B values



Source: Wikipedia (link)

convolution

digital representation of images

let…

- $p_{i,j,k}$ be the intensity value of channel $k \in \{R, G, B\}$ for pixel (i, j)
- for grayscale images, $p_{i,j,k}$ reduces to $p_{i,j}$

normalization [0, 1]

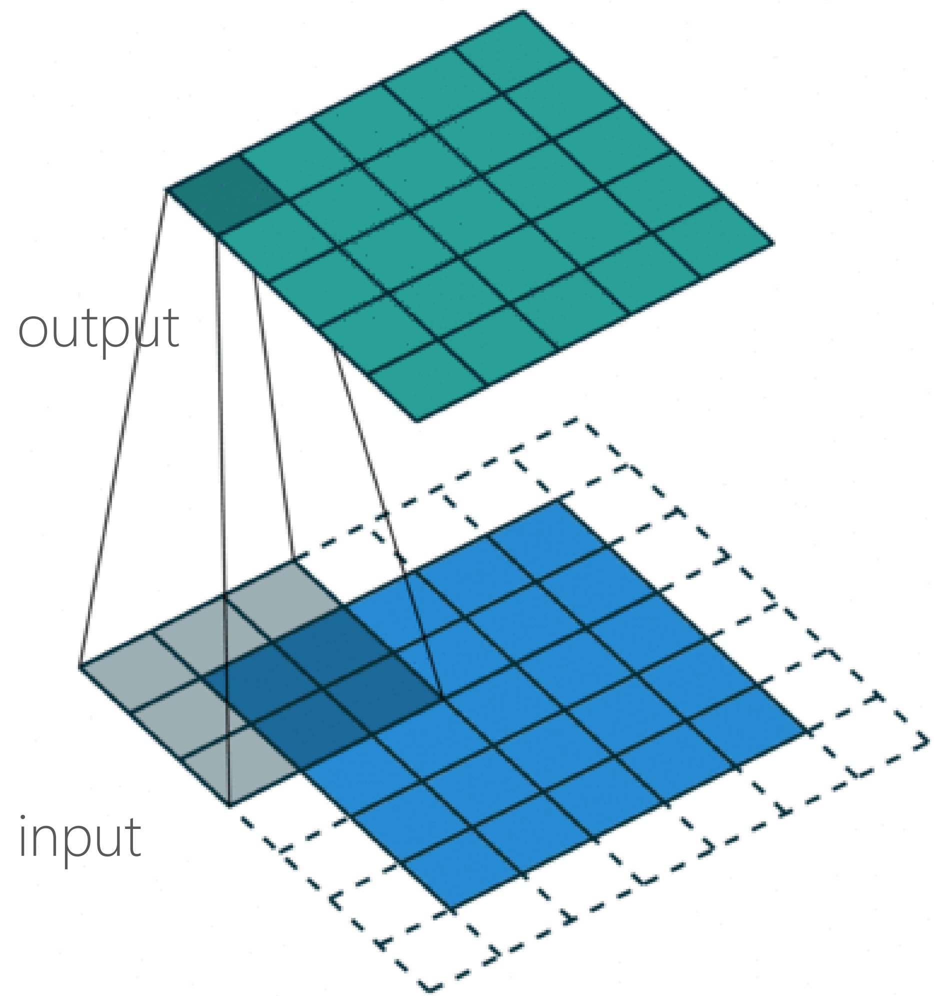
$$\tilde{p}_{i,j,k} = \frac{p_{i,j,k}}{255}$$

normalization [-0.5, 0.5]

$$\tilde{p}_{i,j,k} = \frac{p_{i,j,k}}{255} - 0.5$$

convolution

2D convolution



stride = 1
padding = 1

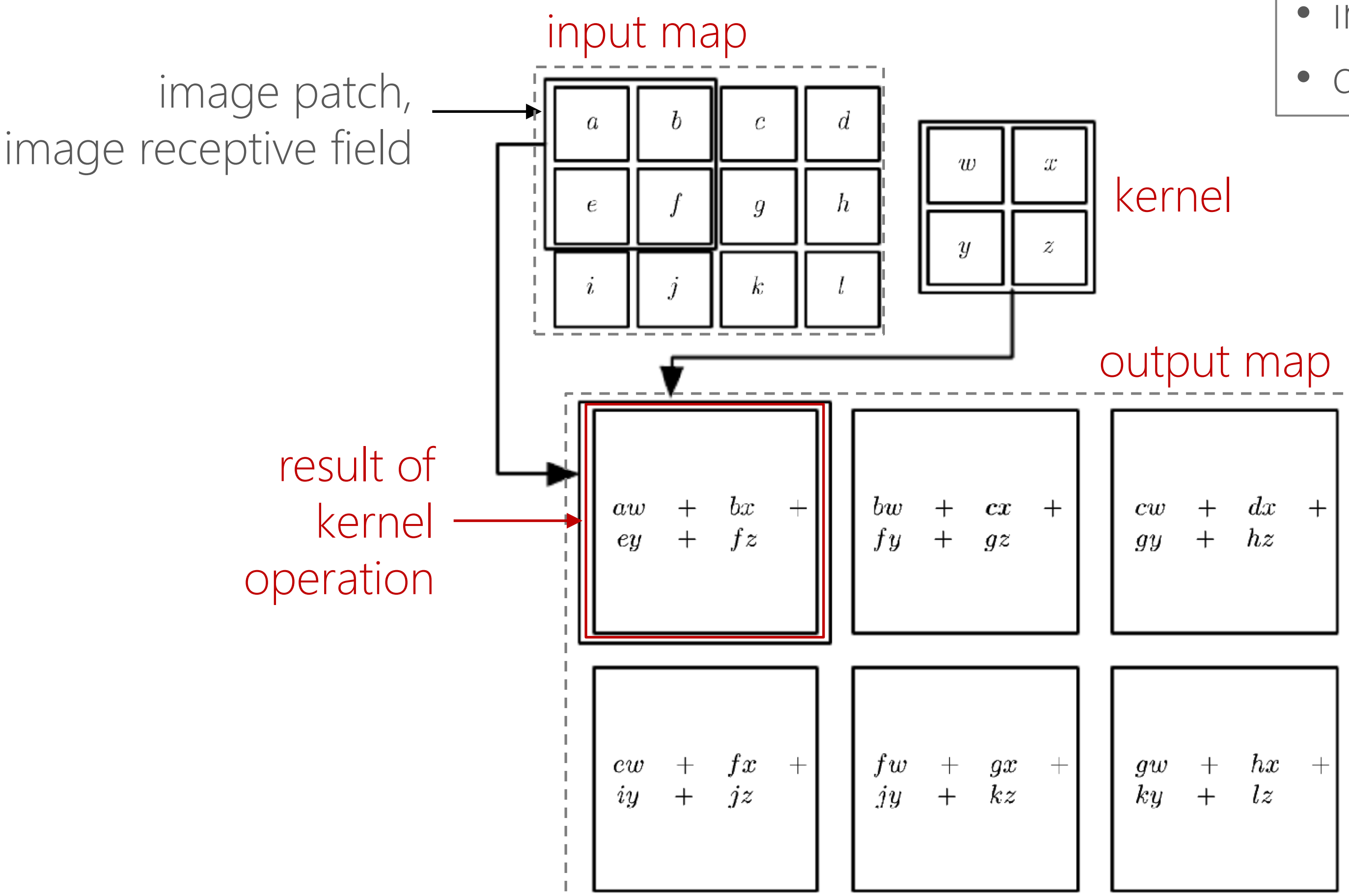
Source: Paul-Louis Pröve, An Introduction to different Types of Convolutions in Deep Learning. Medium.com (link)

convolution

2D convolution

convolution operation

- scalar product
- inner product
- dot product



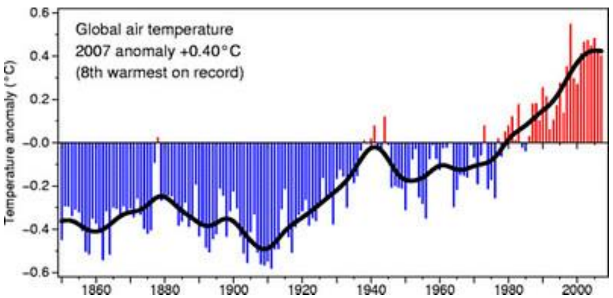
Ian Goodfellow, Yoshua Bengio, Aaron Courville. ***Deep Learning****. MIT Press (2016). Disponible en http://www.deeplearningbook.org/.*

convolution

2D convolution

a convolution operates on "grid-shaped" data:

- a time series, interpretable as a 1D grid



Source: flickr.com

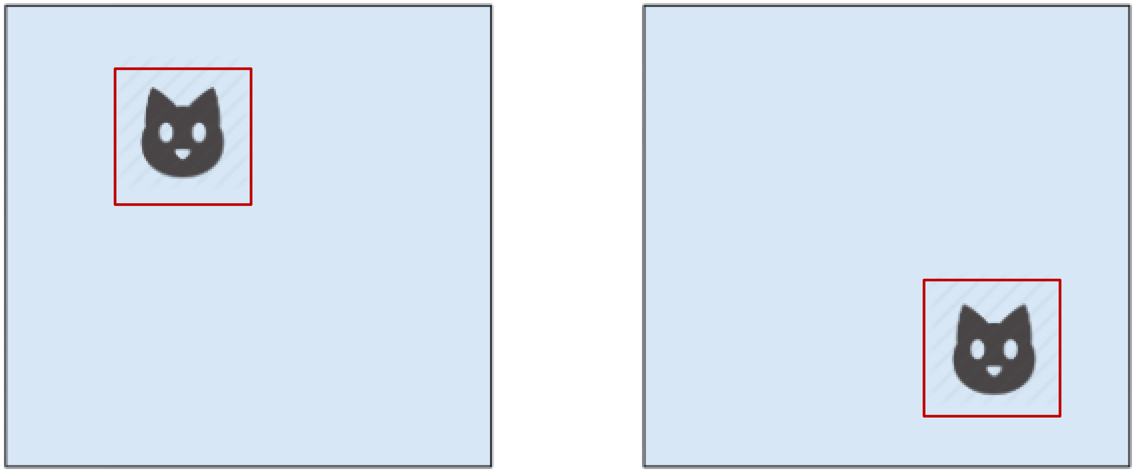
- an image, interpretable as a 2D/3D grid

convolution

2D convolution

convolution operation properties

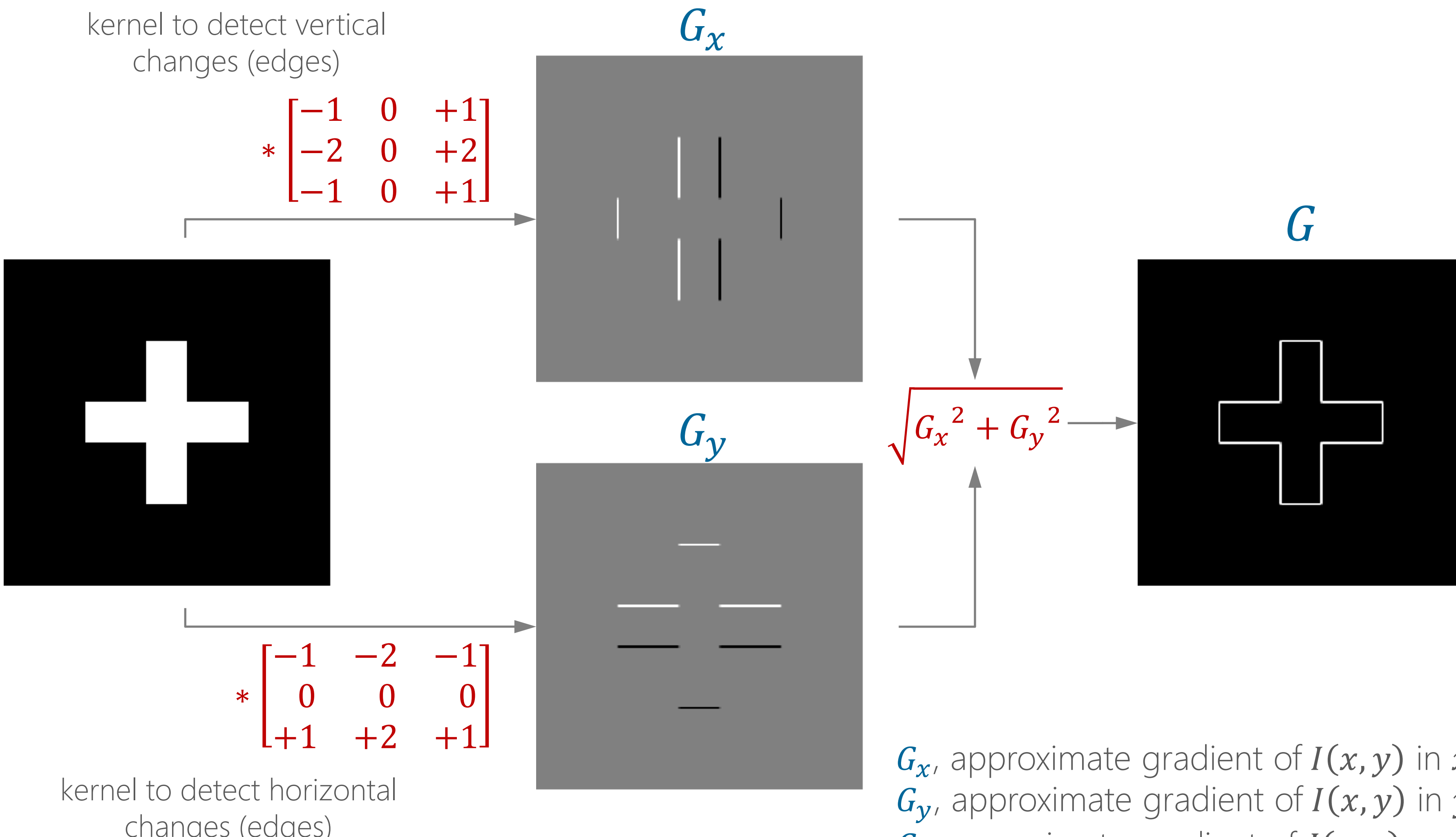
- measures spatial correlations in the image receptive field
- translation invariance: detects pattern at any position in the image



Source: Translational variance in convolutional neural networks. StackExchanged (link).

convolution

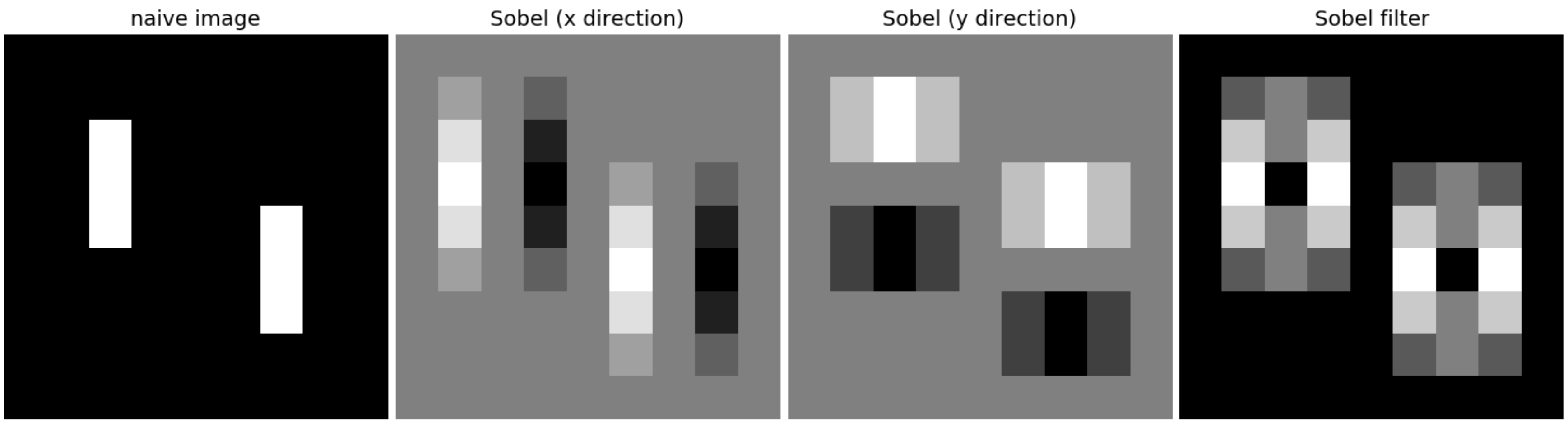
2D convolution: Sobel-Feldman (edge detection)



G_x, approximate gradient of $I(x, y)$ in x
G_y, approximate gradient of $I(x, y)$ in y
G, approximate gradient of $I(x, y)$

convolution

2D convolution: Sobel-Feldman (edge detection)



G_x

```
[[ 0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [ 0.   1.   0.  -1.   0.   0.   0.   0.   0.]
 [ 0.   3.   0.  -3.   0.   0.   0.   0.   0.]
 [ 0.   4.   0.  -4.   0.   1.   0.  -1.   0.]
 [ 0.   3.   0.  -3.   0.   3.   0.  -3.   0.]
 [ 0.   1.   0.  -1.   0.   4.   0.  -4.   0.]
 [ 0.   0.   0.   0.   0.   3.   0.  -3.   0.]
 [ 0.   0.   0.   0.   0.   1.   0.  -1.   0.]
 [ 0.   0.   0.   0.   0.   0.   0.   0.   0.]]
```

G_y

```
[[ 0.   0.   0.   0.   0.   0.   0.   0.   0.]
 [ 0.   1.   2.   1.   0.   0.   0.   0.   0.]
 [ 0.   1.   2.   1.   0.   0.   0.   0.   0.]
 [ 0.   0.   0.   0.   0.   1.   2.   1.   0.]
 [ 0.  -1.  -2.  -1.   0.   1.   2.   1.   0.]
 [ 0.  -1.  -2.  -1.   0.   0.   0.   0.   0.]
 [ 0.   0.   0.   0.   0.  -1.  -2.  -1.   0.]
 [ 0.   0.   0.   0.   0.  -1.  -2.  -1.   0.]
 [ 0.   0.   0.   0.   0.   0.   0.   0.   0.]]
```

image

```
[[0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

numeric output

Do you notice the invariance to the position of the pattern?

G

```
[[0.  0.   0.  0.   0.  0.   0.  0.   0. ]
 [0.  1.4 2.  1.4 0.  0.   0.  0.   0. ]
 [0.  3.2 2.  3.2 0.  0.   0.  0.   0. ]
 [0.  4.  0.  4.  0.  1.4 2.  1.4 0. ]
 [0.  3.2 2.  3.2 0.  3.2 2.  3.2 0. ]
 [0.  1.4 2.  1.4 0.  4.  0.  4.  0. ]
 [0.  0.  0.  0.  0.  3.2 2.  3.2 0. ]
 [0.  0.  0.  0.  0.  1.4 2.  1.4 0. ]
 [0.  0.  0.  0.  0.  0.  0.  0.  0. ]]
```

convolution

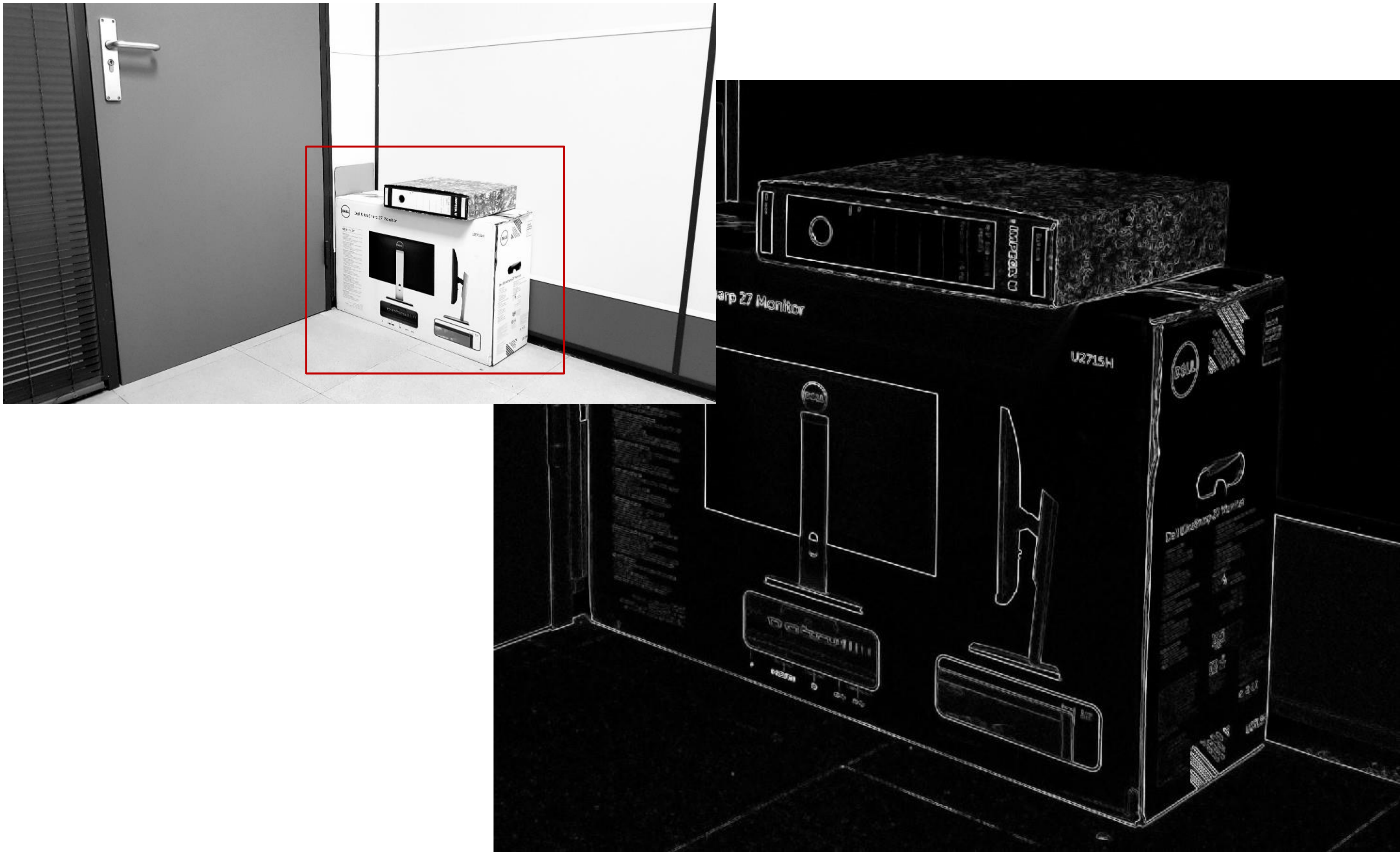
2D convolution: Sobel-Feldman (edge detection)

easy task

corner of my office with mostly straight edges and a few textures

convolution

2D convolution: Sobel-Feldman (edge detection)

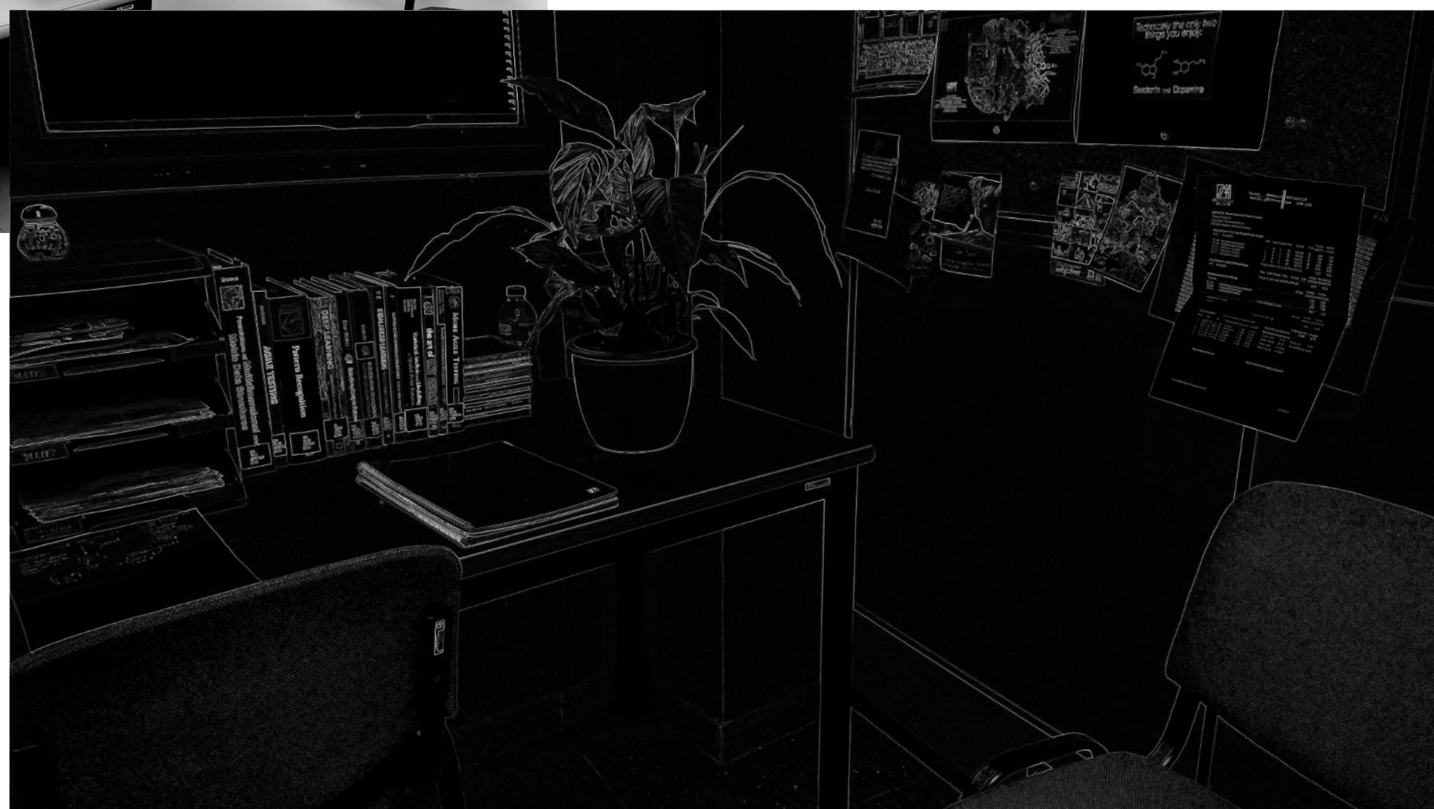


convolution

2D convolution: Sobel-Feldman (edge detection)

difficult task

corner of my office with edges of diverse geometry and rich textures

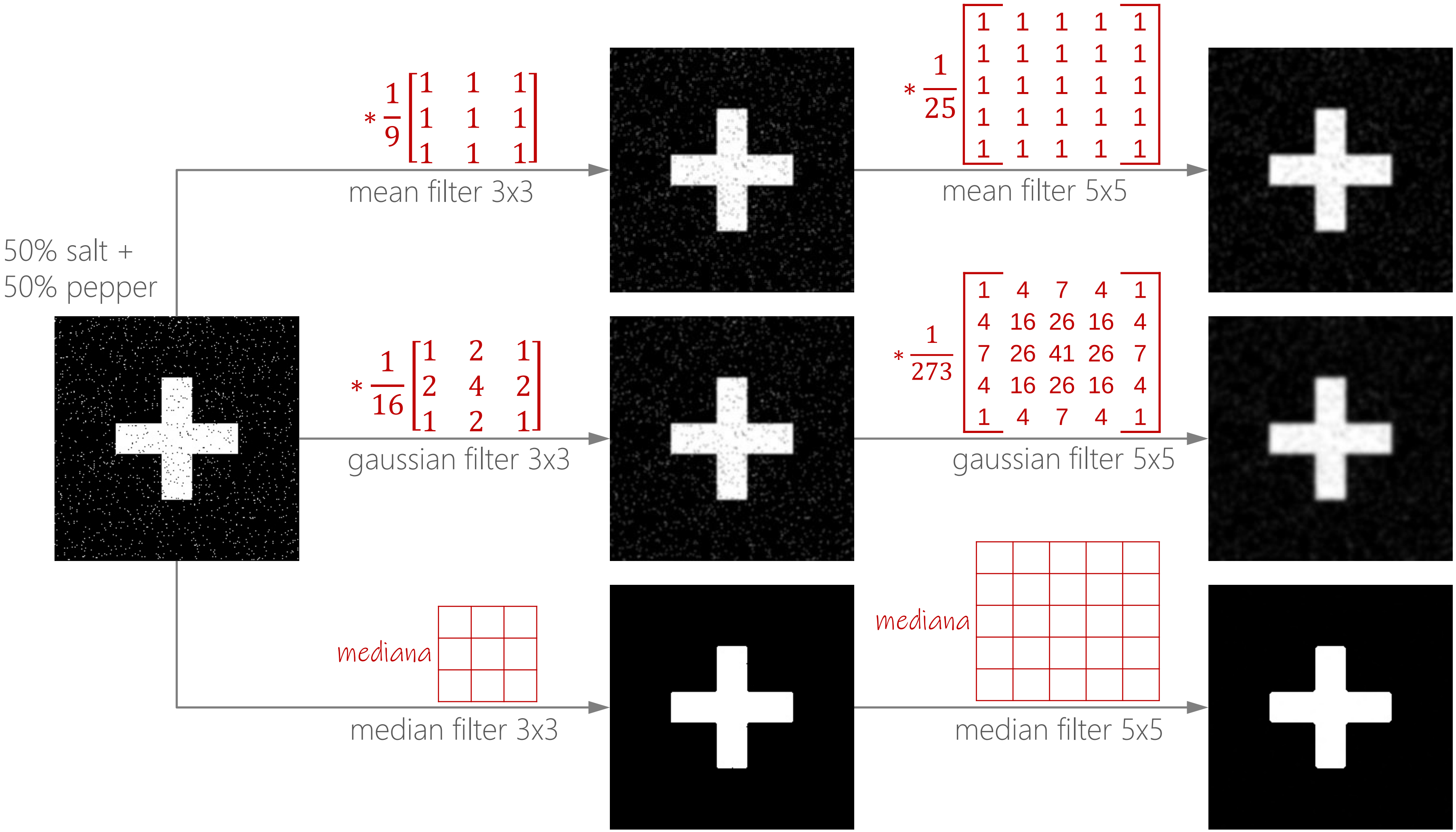


convolution

2D convolution: Sobel-Feldman (edge detection)

convolution

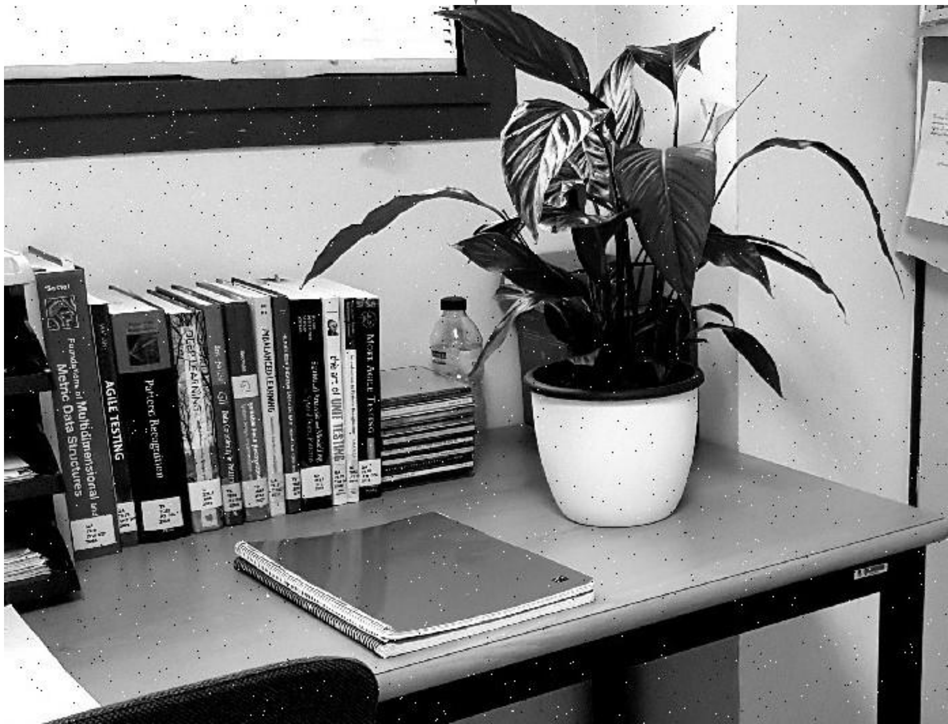
2D convolution: smoothing images



convolution

2D convolution: smoothing images

50% salt +
50% pepper

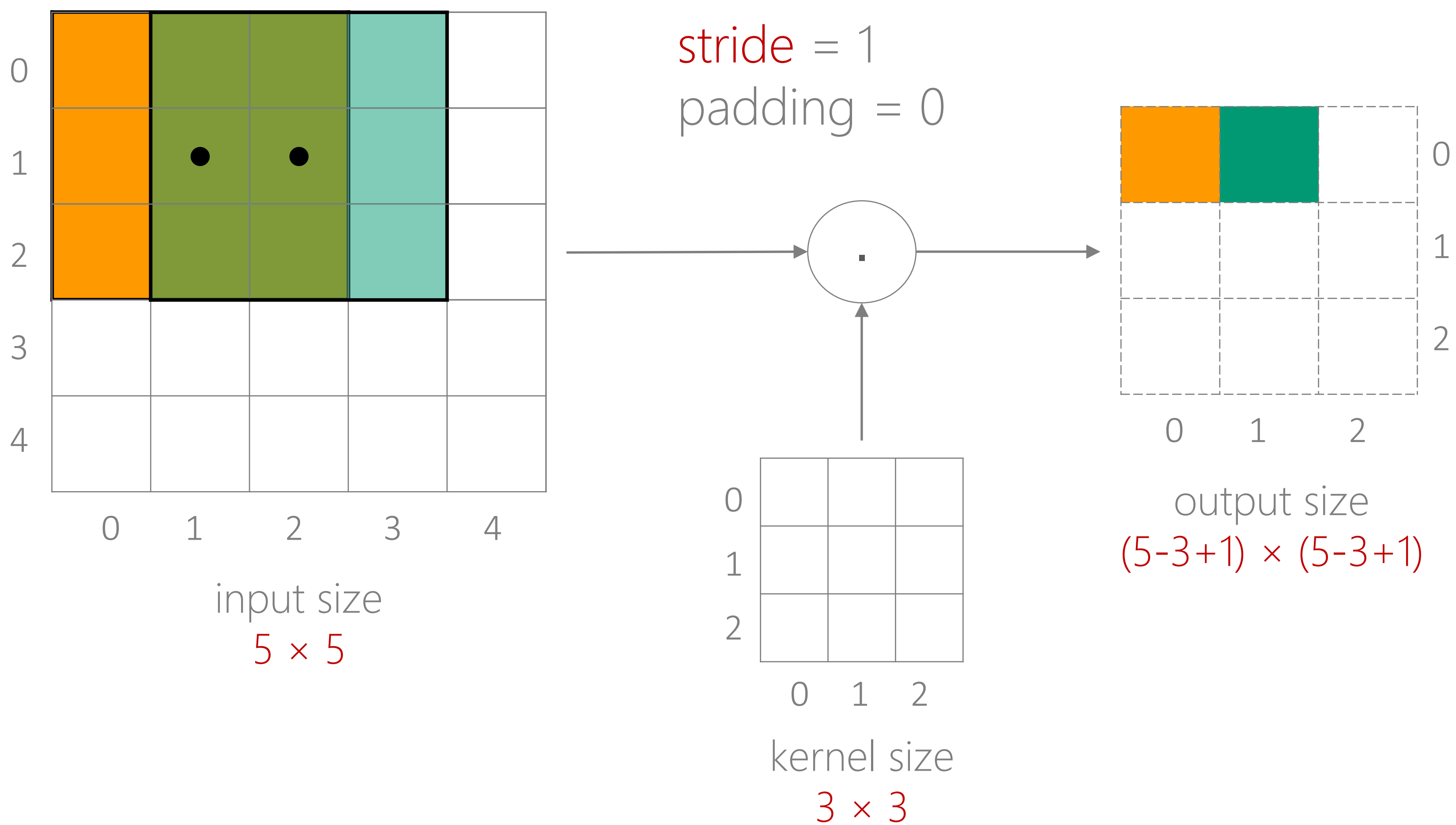


convolution

2D convolution: smoothing images

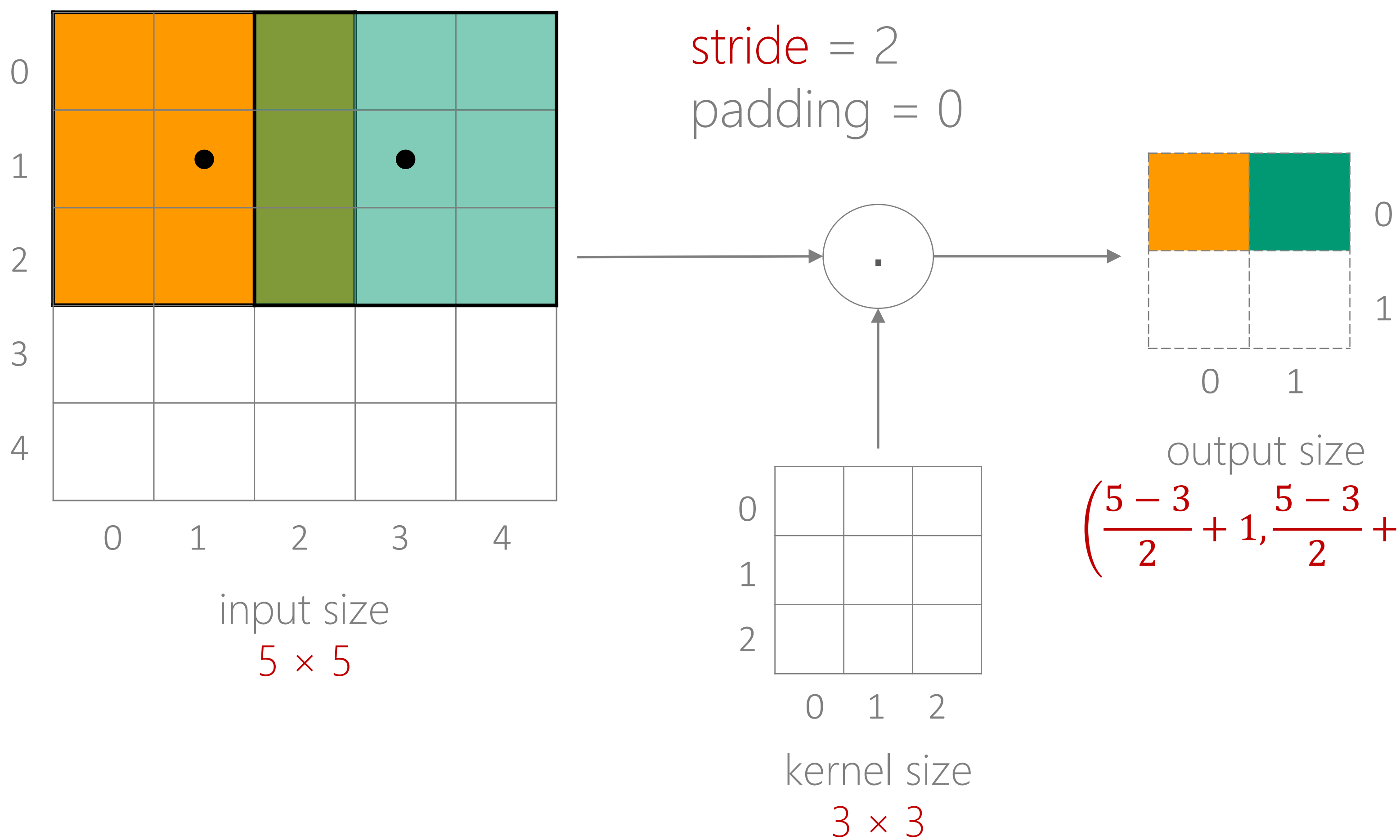
convolution

2D convolution: stride, padding



convolution

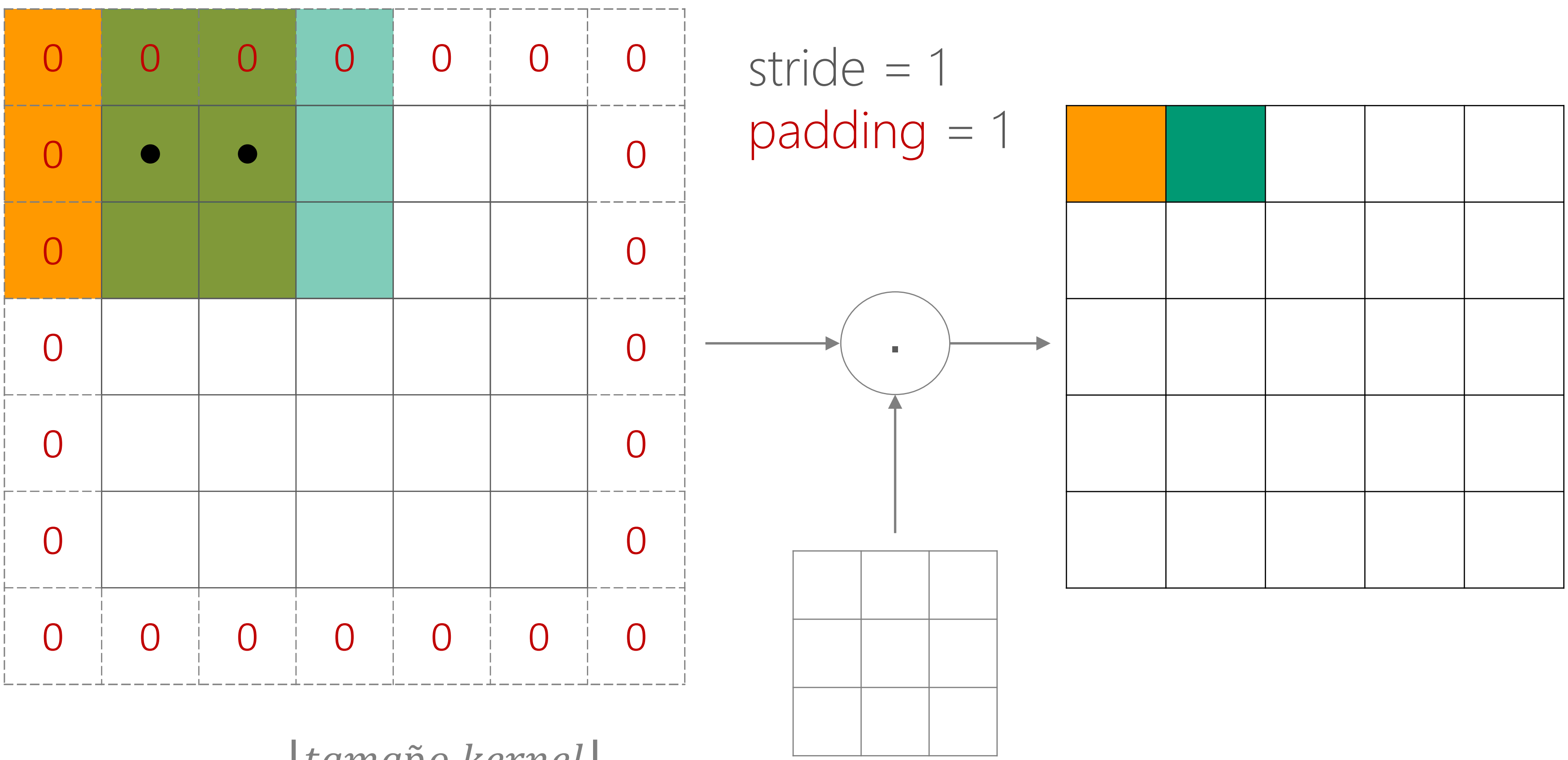
2D convolution: stride, padding



$$\left(\frac{5-3}{2}+1, \frac{5-3}{2}+1\right)$$

convolution

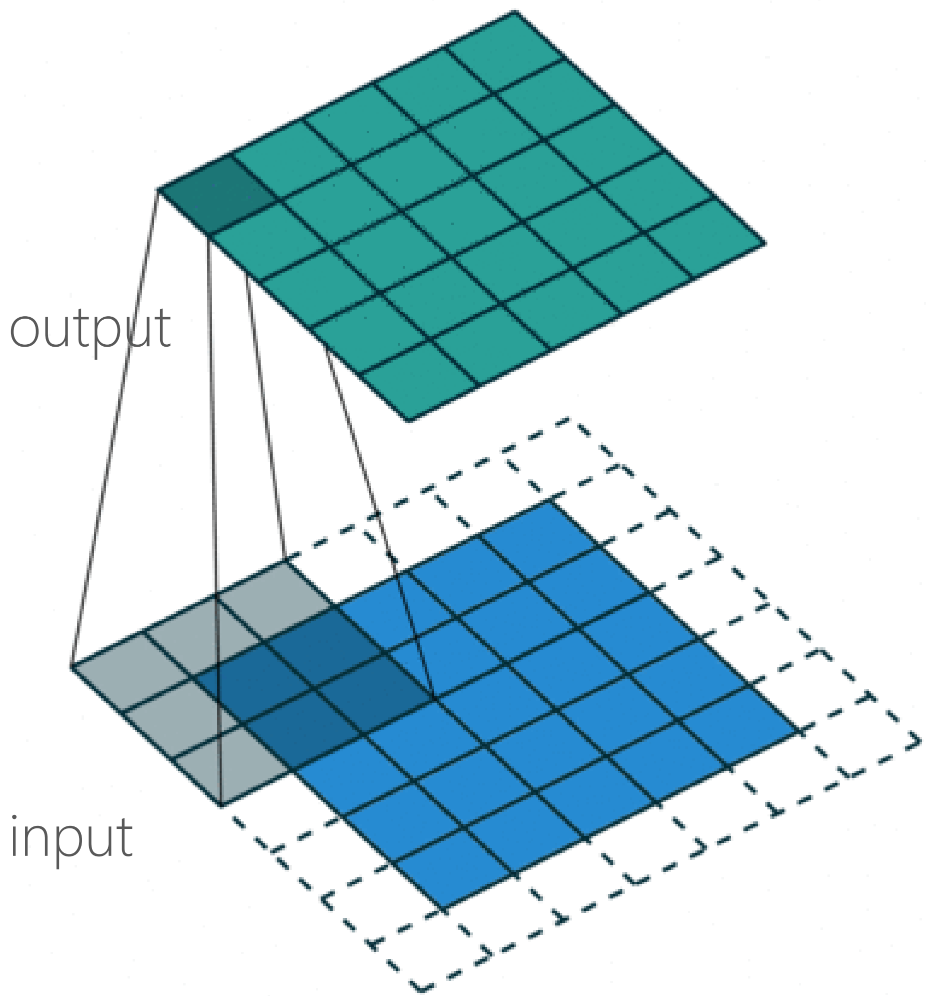
2D convolution: stride, padding



$$padding \leq \left\lfloor \frac{tamaño\ kernel}{2} \right\rfloor$$

convolution

2D convolution: stride, padding



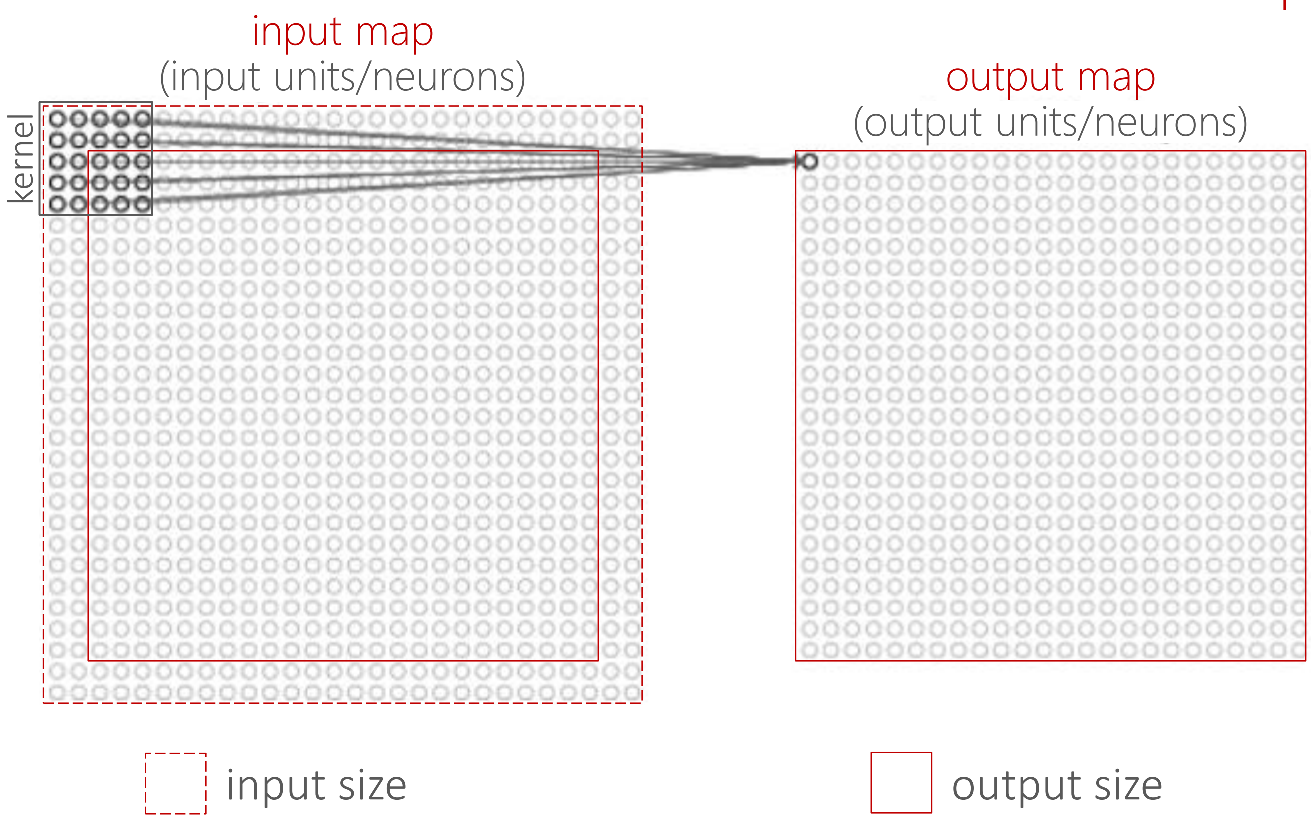
stride = 1
padding = 1

Source: Paul-Louis Pröve, An Introduction to different Types of Convolutions in Deep Learning. Medium.com (link)

convolution

2D convolution

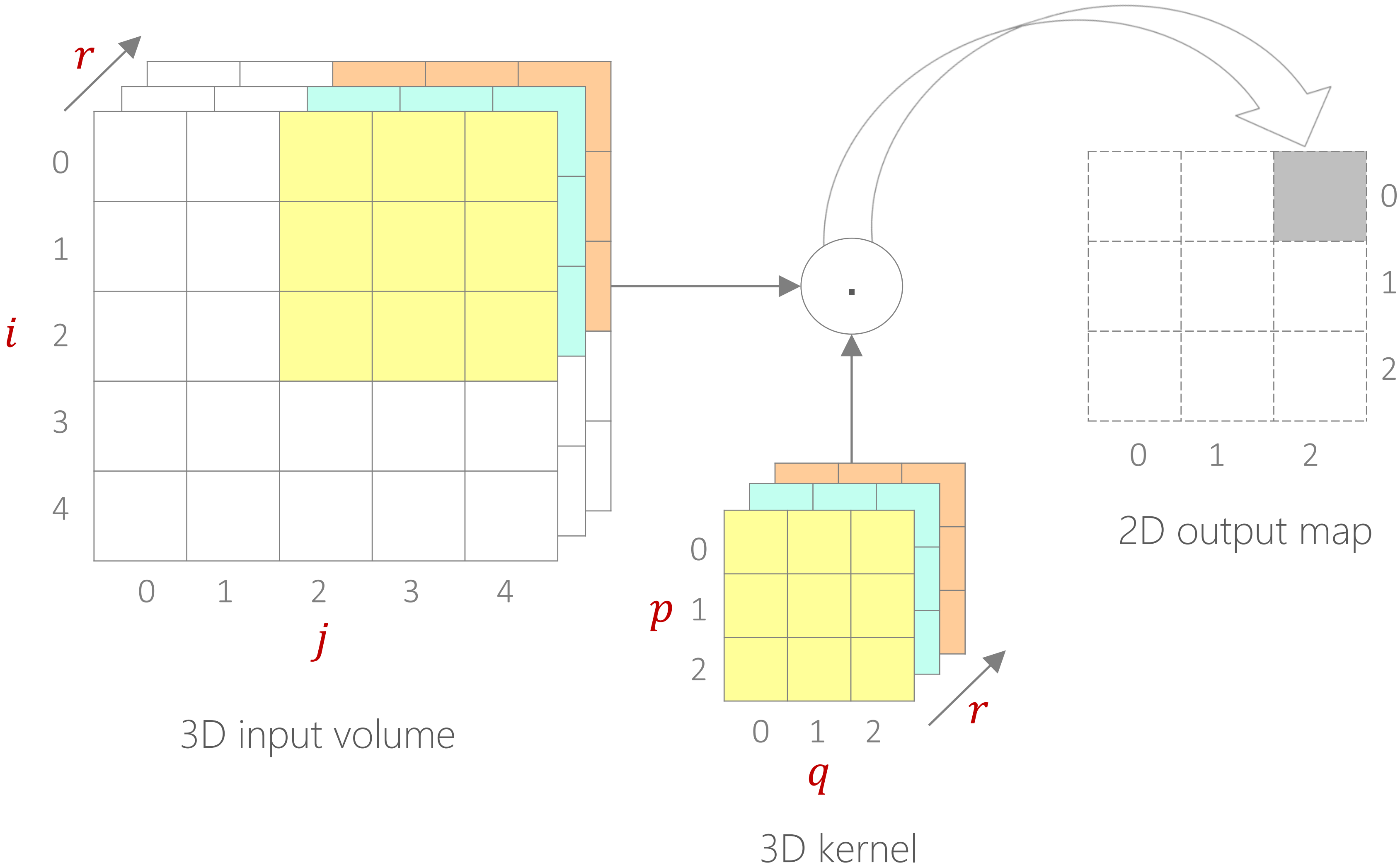
stride = 1
padding = 2



Based on: Michael Nielsen (2019). Neural Networks and Deep Learning, online book (+).

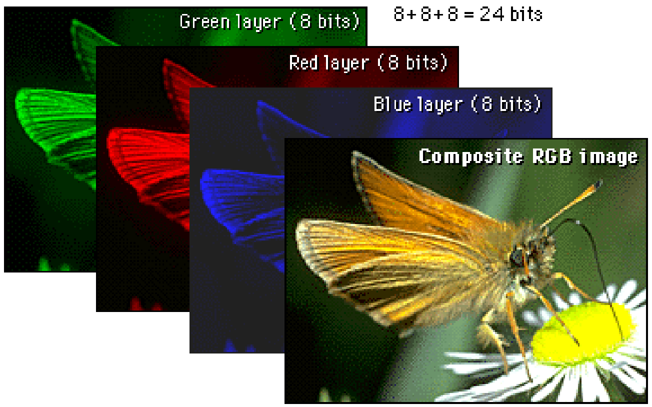
convolution

2D convolution (on 3D maps)



convolution

2D convolution (on multi-channel images)

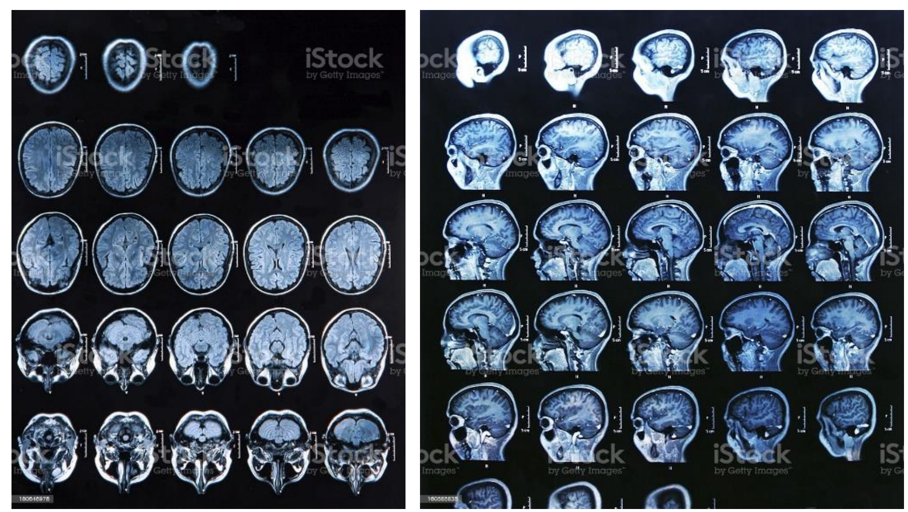


Source: https://webstyleguide.com/wsg1/graphics/display_primer.html

multi-channel images

multi-spectral image

Source: wikipedia.org



Magnetic Resonance Image (MRI)

Source: istockphoto.com

convolution

2D convolution

Let…

I, be a 3D input volume

W, be a 3D kernel (filter, operator)

O, be the 2D output map

$$O(i,j) = (W \cdot I)(i,j) = \sum_p \sum_q \sum_r I_{i+p,j+q,r} \cdot W_{p,q,r} + b$$

convolution

2D convolution

Let...

I, be a 3D input volume

W^c, the c-th 3D kernel of a set of C kernels

O, be the 3D output volume

$$O(i, j, c) = (W^c \cdot I)(i, j, c) = \sum_p \sum_q \sum_r I_{i+p, j+q, r} \cdot W^c_{p,q,r} + b^c$$

convolution

2D convolution: practical exercise

<u>example above</u>: with a 5x5x3 input map, 3x3x3 kernel, padding 0, and stride 1, the output map size was 3x3x1.

what would have been the dimension of the output map with a kernel...

- 2x2x3 -> ?
- 4x4x3 -> ?
- 5x5x3 -> ?

<u>generalization</u>: given an *HxWxC* input, and *KxKxC* kernel, padding *P*, and stride *S*, what would be the size of the output map?

- *KxKxC* -> ?

convolution

2D convolution: practical exercise

example above: with a 5x5x3 input map, 3x3x3 kernel, padding 0, and stride 1, the output map size was 3x3x1.

what would have been the dimension of the output map with a kernel...

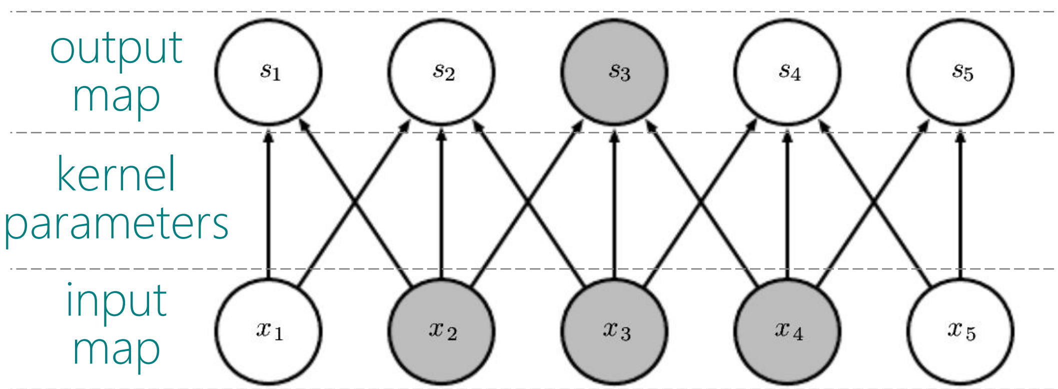
- 2x2x3 -> 4x4x1
- 4x4x3 -> 2x2x1
- 5x5x3 -> 1x1x1

generalization: given an *HxWxC* input, and *KxKxC* kernel, padding *P*, and stride *S*, what would be the size of the output map?

- *KxKxC* -> $\left(\frac{H-K+2P}{S}+1, \frac{W-K+2P}{S}+1\right)$

convolution

convolutional connection pattern vs dense pattern

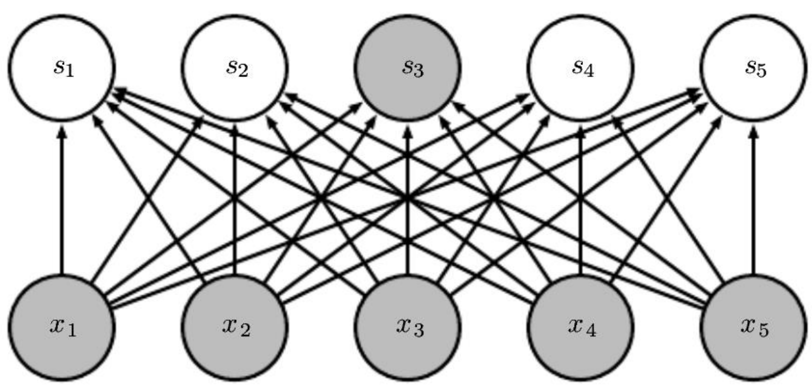


CNN

Convolutional
Neural
Network

sparse connectivity

(when kernel is smaller than input)



FCN

Fully
Connected
Network

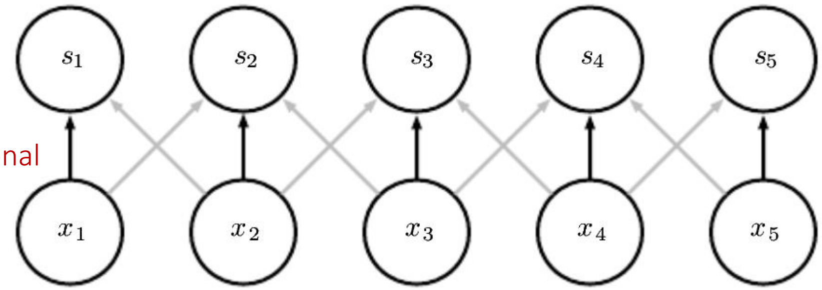
dense connectivity

Ian Goodfellow, Yoshua Bengio, Aaron Courville. ***Deep Learning****. MIT Press (2016).*
Disponible en http://www.deeplearningbook.org/.

convolution

convolutional connection pattern vs dense pattern

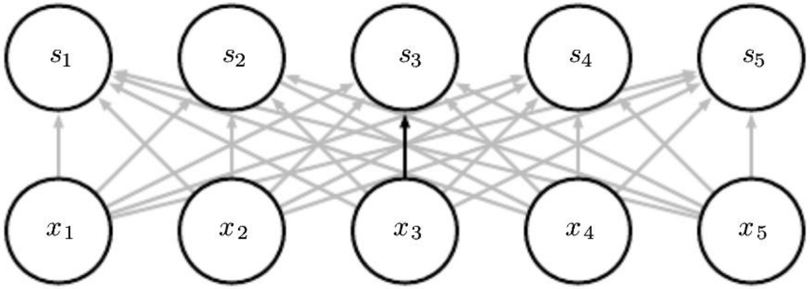
CNN
Convolutional Neural Network



shared parameters

- the no. param. depends on the kernel
- fewer parameters are stored
- fewer calculations are made

FCN
Fully Connected Network



one parameter per connection

- the number of parameters depends on the units of the connected layers

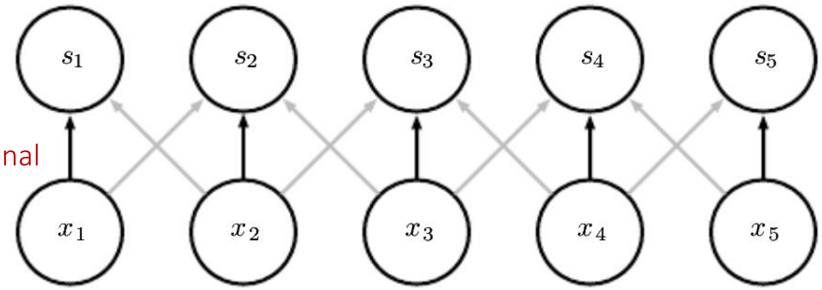
Ian Goodfellow, Yoshua Bengio, Aaron Courville. ***Deep Learning****. MIT Press (2016). Disponible en http://www.deeplearningbook.org/.*

convolution

convolutional connection pattern vs dense pattern

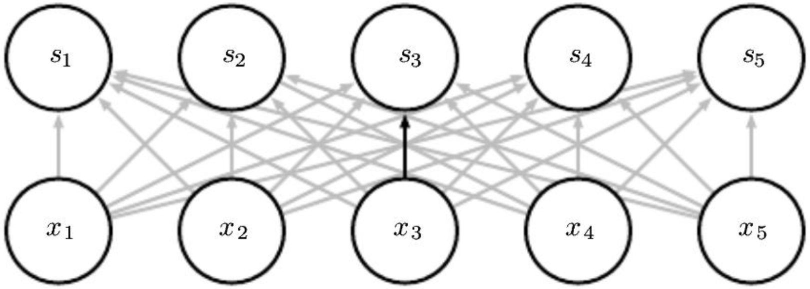
What if you change the size of the network input?

CNN
Convolutional Neural Network



it keeps working
connections/parameters do not depend on input size

FCN
Fully Connected Network



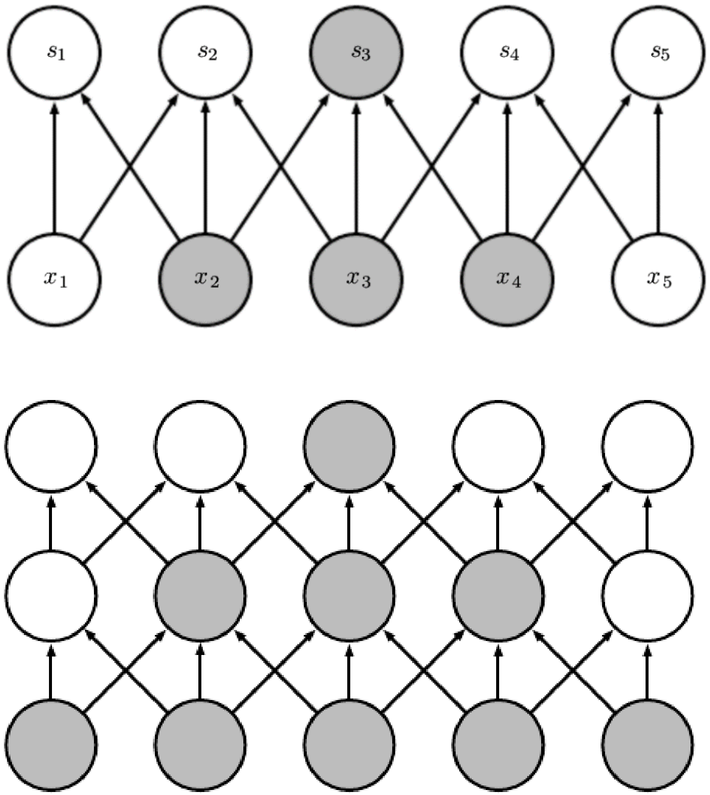
it stops working
connections/parameters DO depend on input size

Ian Goodfellow, Yoshua Bengio, Aaron Courville. ***Deep Learning***. *MIT Press (2016). Disponible en http://www.deeplearningbook.org/.*

convolution

convolutional connection pattern vs dense pattern

CNN
Convolutional
Neural
Network

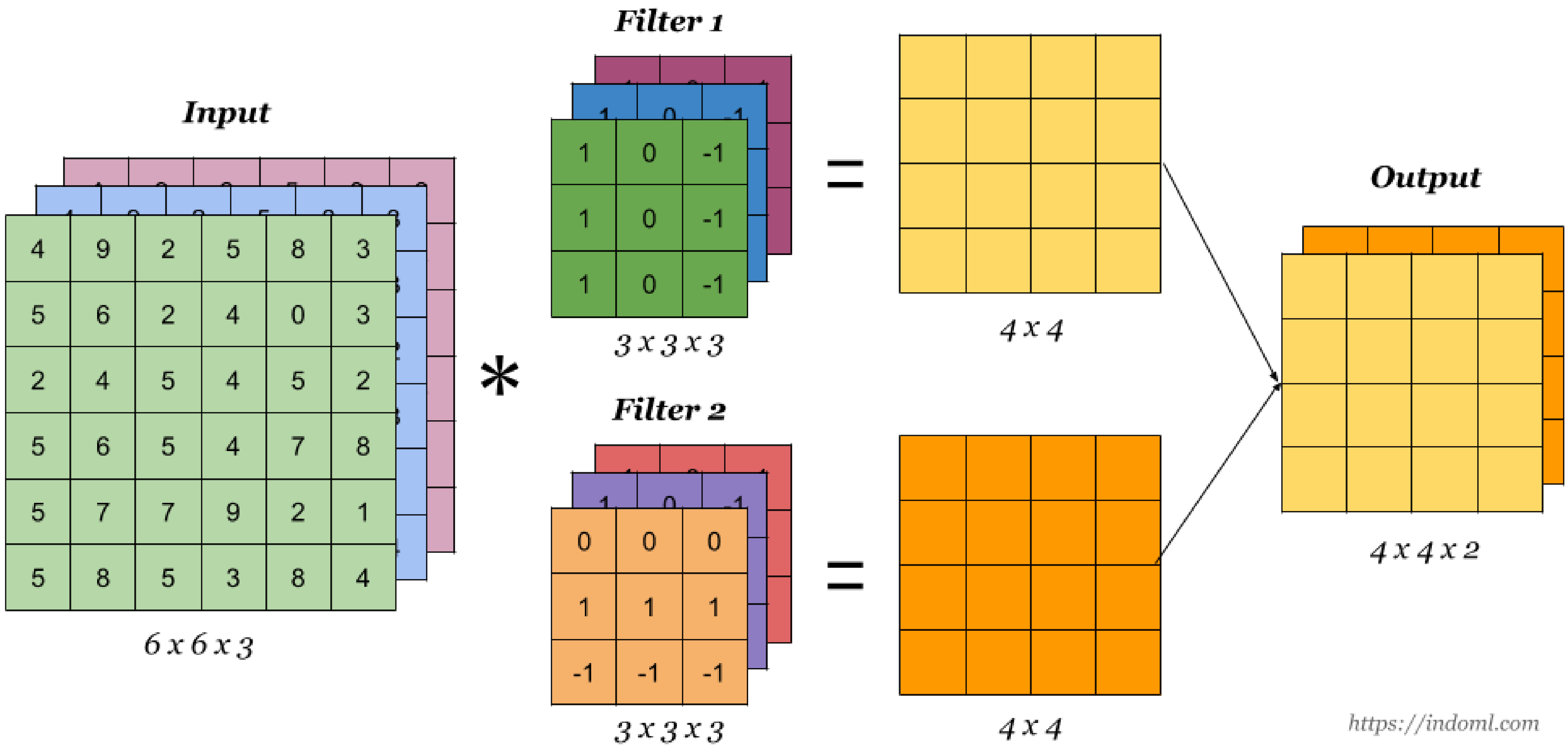


- x_2, x_3, x_4 (input units) are known as the receptive field of s_3
- deeper layer units are related to larger receptive fields
- units in very deep layers may be indirectly connected to all or most of the input image

Ian Goodfellow, Yoshua Bengio, Aaron Courville. ***Deep Learning****. MIT Press (2016).*
Disponible en http://www.deeplearningbook.org/.

convolutional layer

2D convolutions

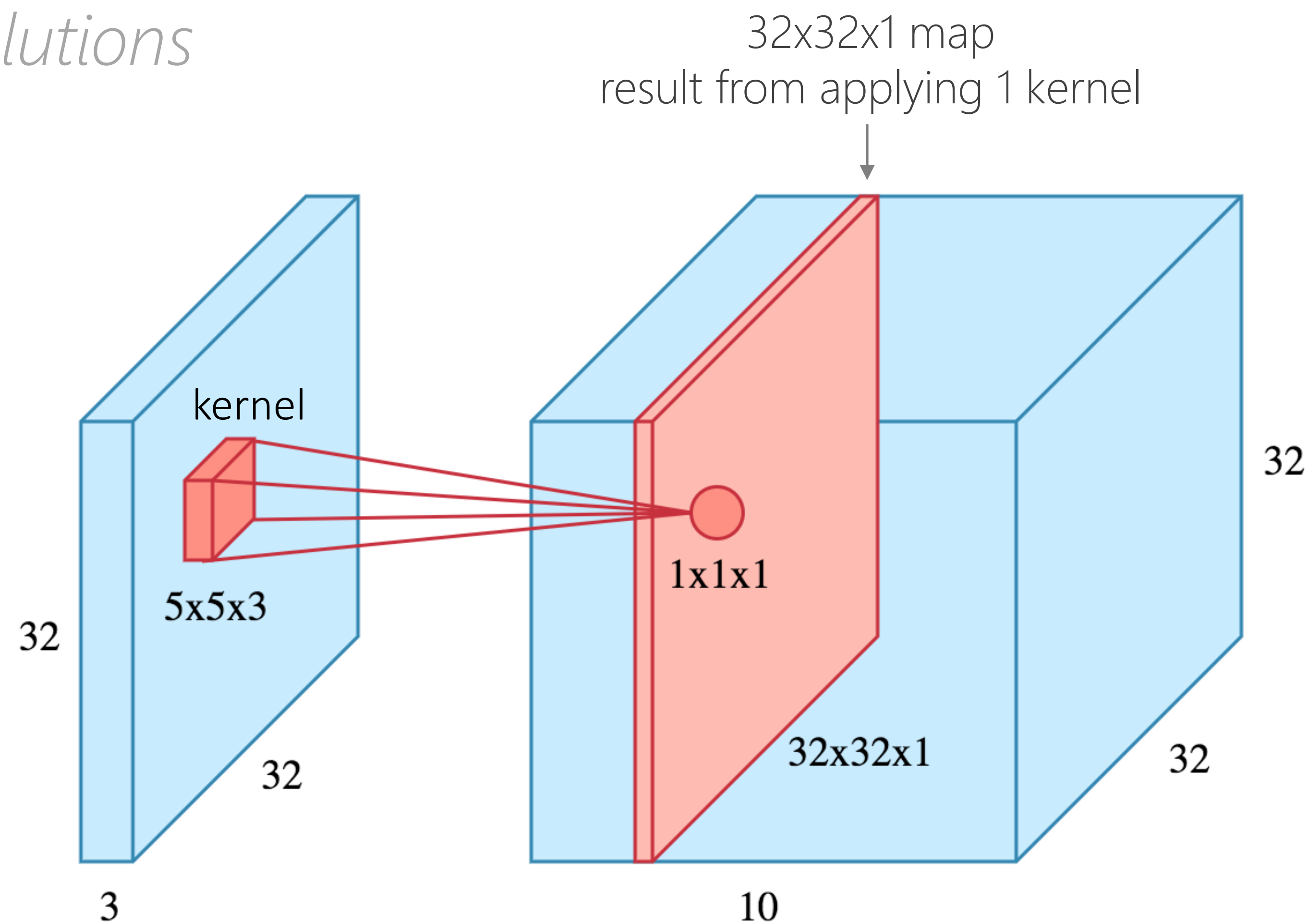


Source: https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

convolutional layer

2D convolutions

2D kernel
spatially small operator; depth equal to that of the entrance



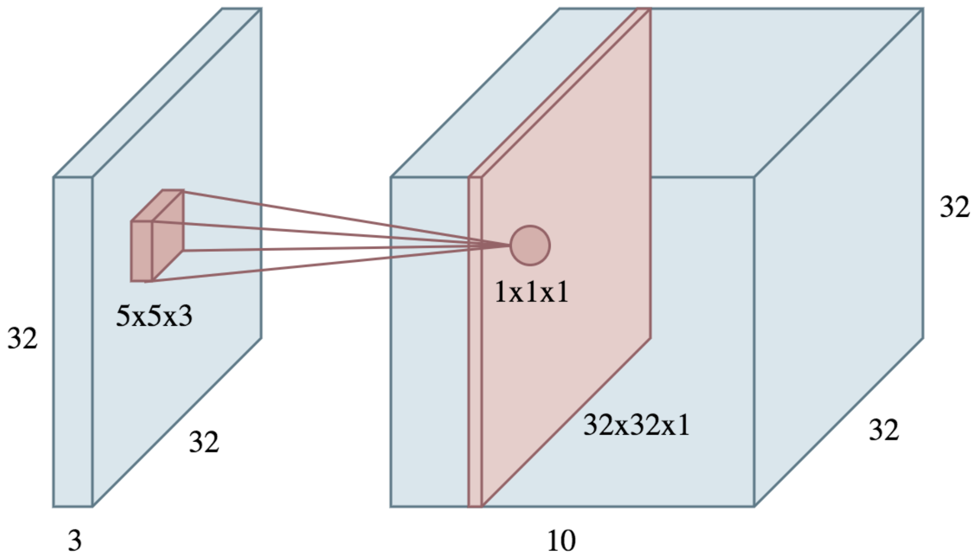
input data volume

32x32x10 output data volume (result of applying 10 kernels over input volume)

Source: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

convolutional layer

2D convolutions



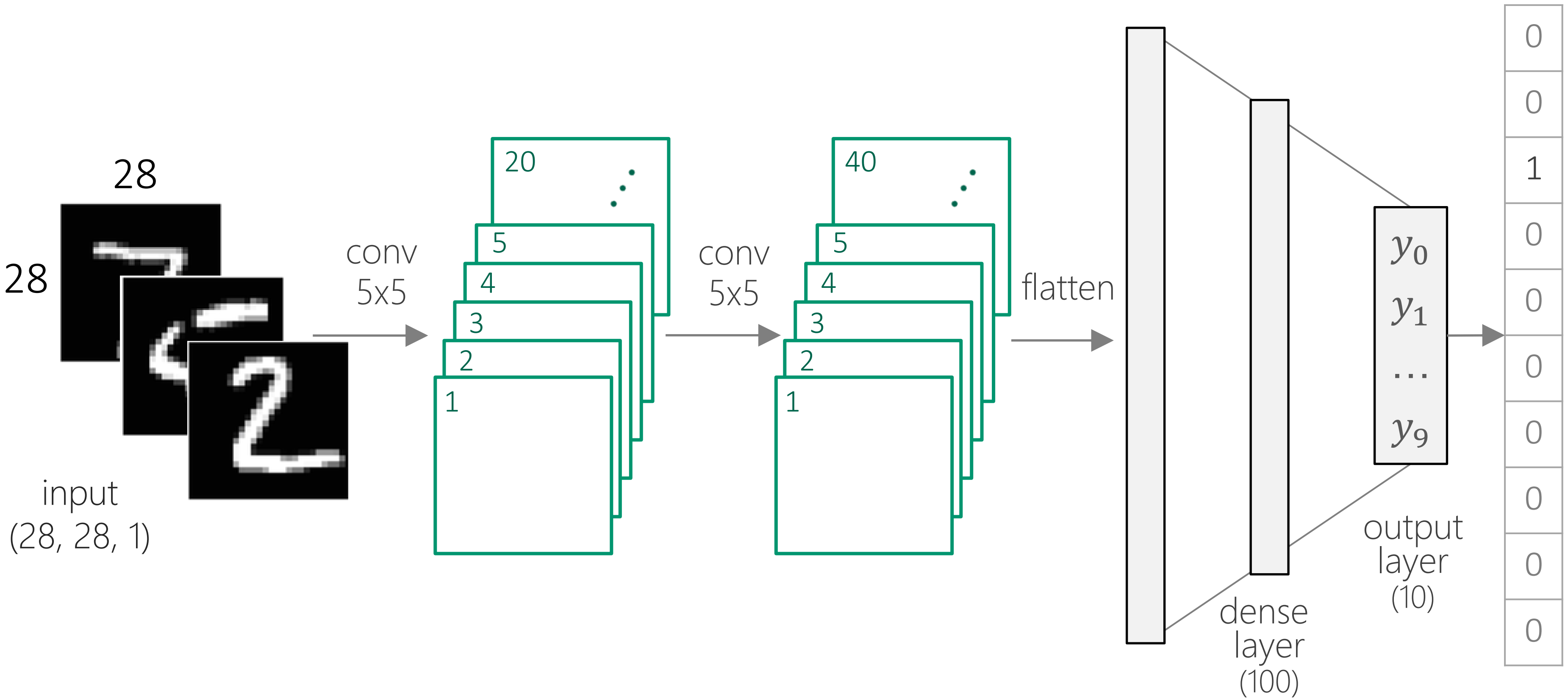
the kernel moves over the input volume and convolves with each position

- operator: 5x5x3 kernel
- input (to the convolution operation): 5x5x3 block of the input volume
- operation: scalar product
- result: scalar (1x1x1)
- kernel trainable params = 75 + 1
- kernel depth = input volumen depth
- output volumen depth = number of kernels

Source: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

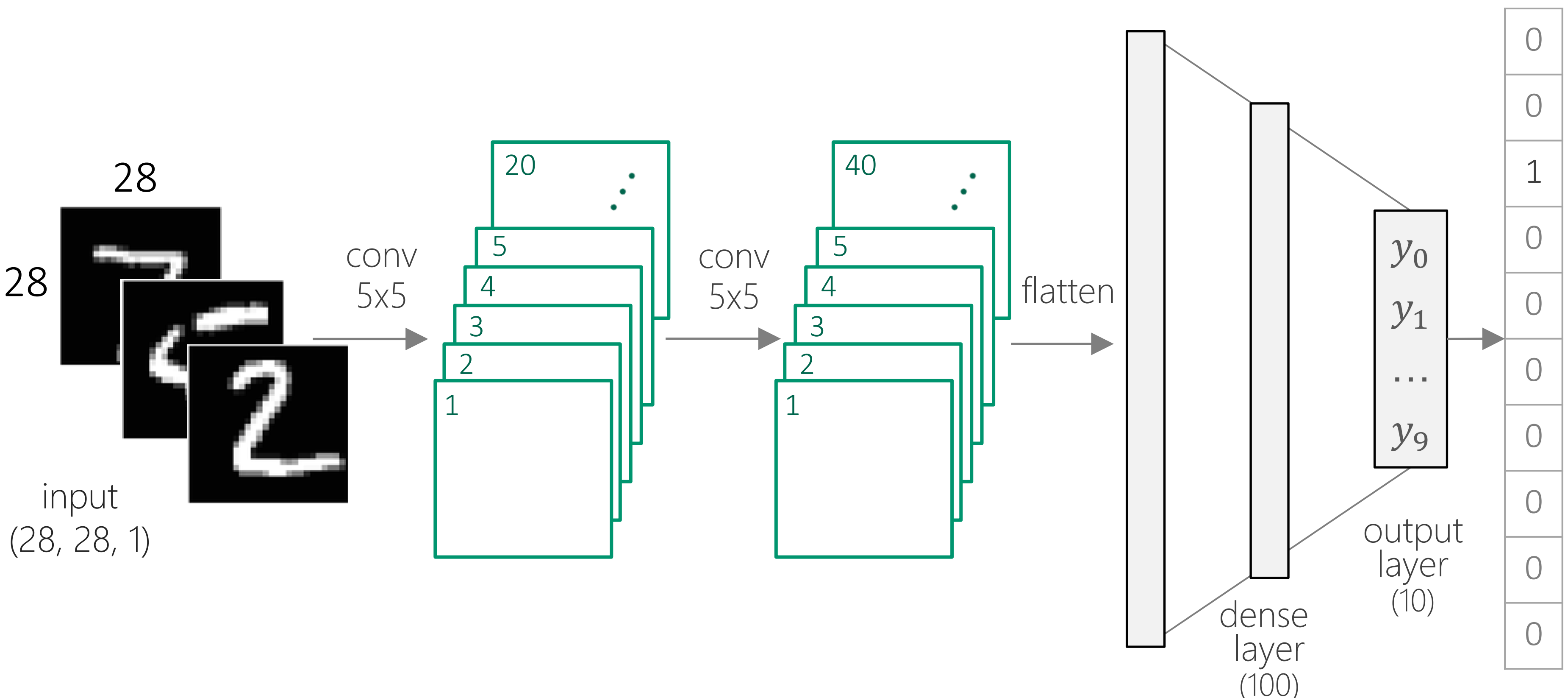
case study

convolutional neural network for the MNIST task



case study

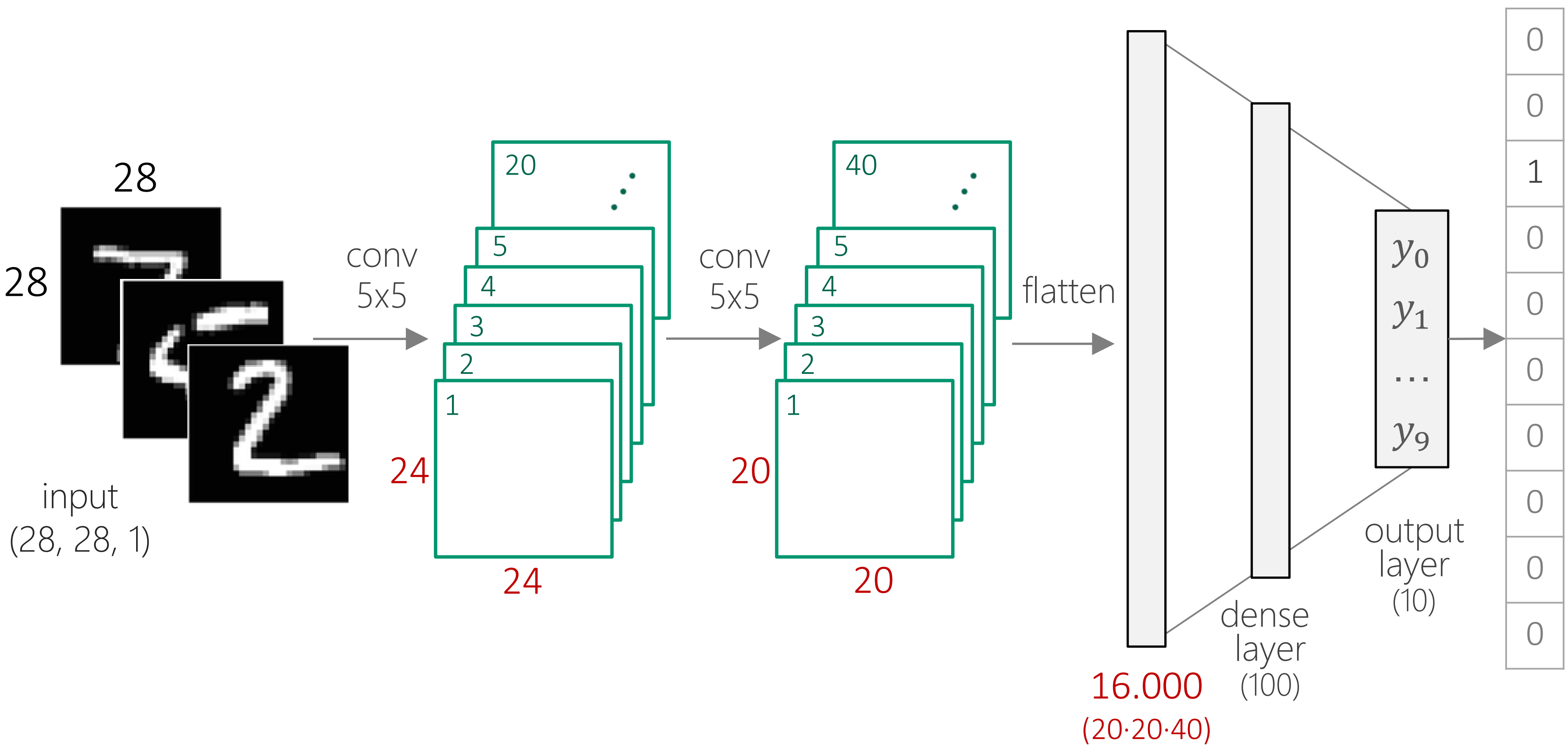
convolutional neural network for the MNIST task



What size is the output of each layer?

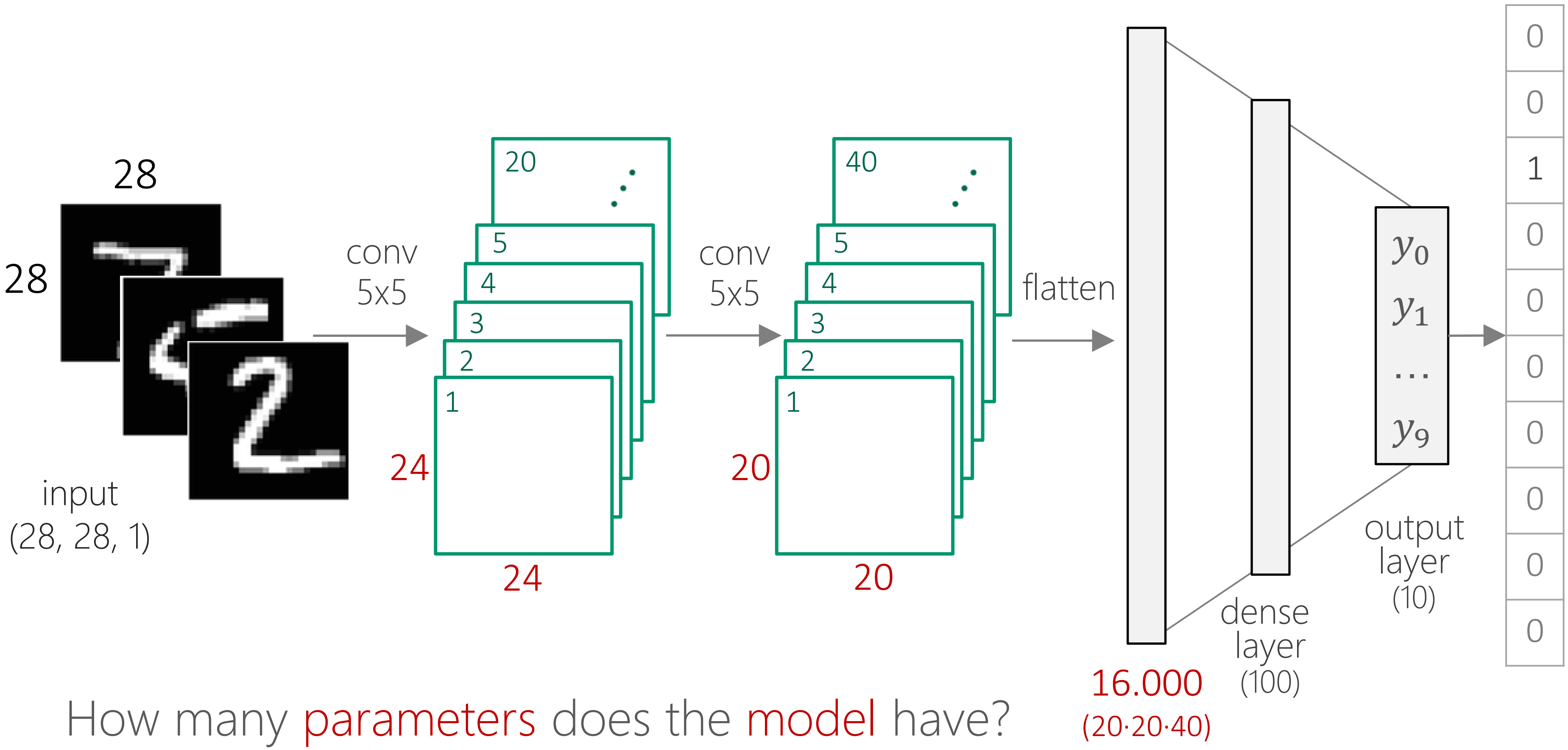
case study

convolutional neural network for the MNIST task



case study

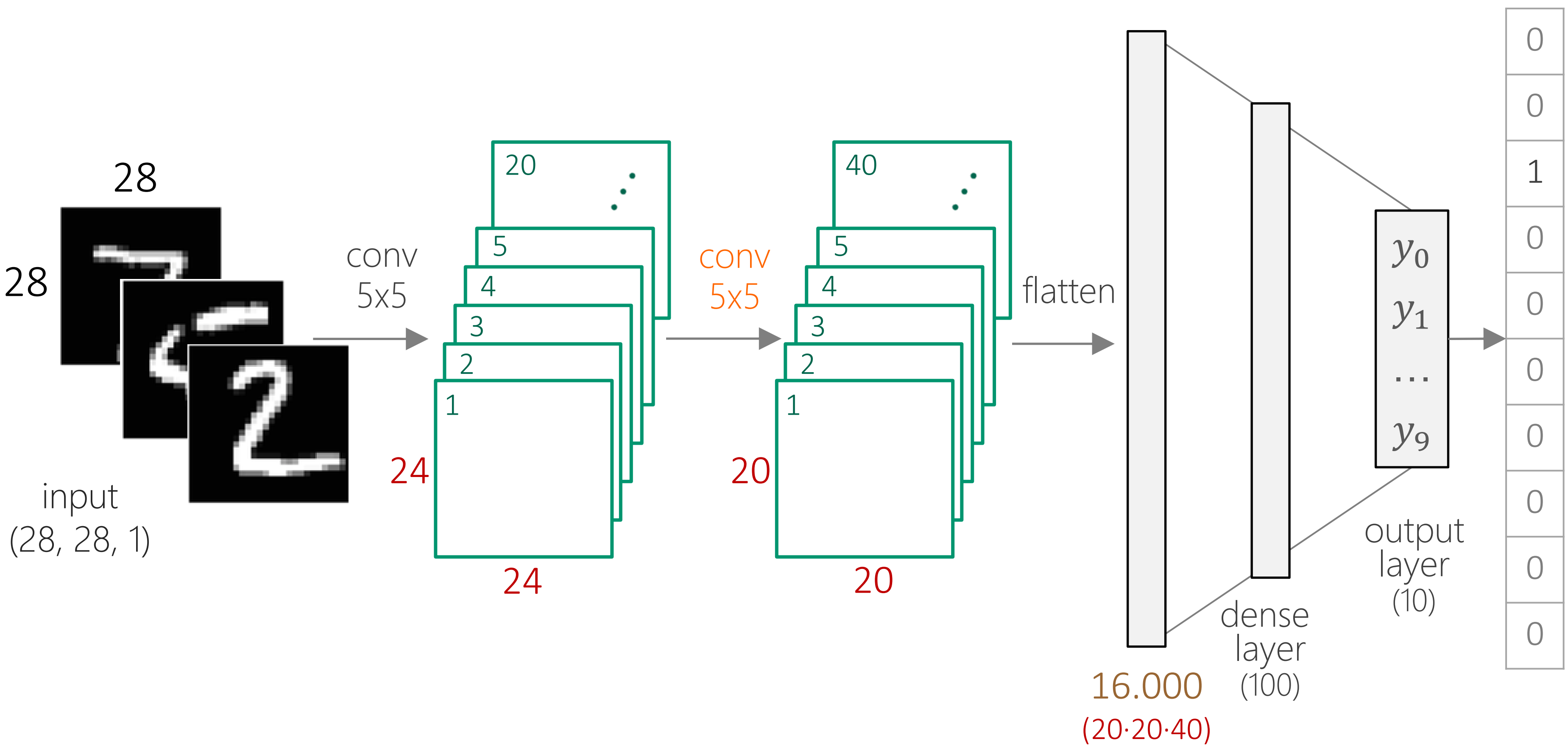
convolutional neural network for the MNIST task



How many parameters does the model have?

case study

convolutional neural network for the MNIST task



20 x 5 x 5 + 20	+	40 x 5 x 5 x 20 + 40	+	16.000 x 100 + 100	+	100 x 10 + 10	=
520	+	20.040	+	1.600.100	+	1.010	=
1.621.670 trainable parameters							

case study

convolutional neural network for the MNIST task

implementation in Keras…

- Keras is a Python library for creating neural networks (deep learning framework)
- interface to high-level modules based on Tensorflow, Theano, or the Microsoft Cognitive Toolkit
- it allows you to combine predefined pieces common in neural networks such as layers, objective functions, activation functions, optimizers, etc.

case study

convolutional neural network for the MNIST task

simplified script

```
(X_train,y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')/255
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')/255
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

model = Sequential()
model.add(Conv2D(20, kernel_size=5, activation='relu', input_shape=(28, 28, 1)))
model.add(Conv2D(40, kernel_size=5, activation='relu'))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=20)
```

case study

convolutional neural network for the MNIST task

hyperparameters

```
(X_train,y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 28, 28, 1).astype('float32')/255
X_test = X_test.reshape(10000, 28, 28, 1).astype('float32')/255
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

model = Sequential()
model.add(Conv2D(20, kernel_size=5, activation='relu', input_shape=(28, 28, 1)))
model.add(Conv2D(40, kernel_size=5, activation='relu'))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=20)
```

case study

convolutional neural network for the MNIST task

activation	'relu'	• Rectified Linear Unit (ReLU) • activation function of hidden layers
activation	'softmax'	• output layer activation function • probability distribution over K outputs
loss	'categorical_crossentropy'	• one-hot vector + softmax + cross entropy • loss function • measures discrepancy between two distributions
optimizer	'adam'	• adam = adaptive moment estimation • parameter optimization method

optimizer

adam: Adaptive Moment Estimation

gradient descent

$$\theta_{t+1} = \theta_t - \gamma \frac{\partial L}{\partial \theta}(\theta_t)$$

adam:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$g_t = \frac{\partial L}{\partial \theta}(\theta_t)$$

typical values

$\beta_1 = 0.9$

$\beta_2 = 0.999$

$\epsilon = 10^{-8}$

Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. Int. Conf. on Learning Representations, p. 1–13, 2015.

optimizer

adam: considerations

- Adam adjusts learning rates individually for each parameter
- it is an evolution of Stochastic Gradient Descent with momentum
- m_t is a first-order moment (moving average of the gradient)
- m_t is a mean estimate of the last $1/(1-\beta_1)$ gradients
- v_t is a second-order moment (it characterizes the variance of g)
- v_t scales the learning rates adaptively for the parameter θ
- the adapted coefficient is inversely proportional to the variance of the gradient
- The epsilon (ε) value is a small constant added for numerical stability.
- $m_0 = 0$ y $v_0 = 0$; $\widehat{m}_t$ y $\hat{v}_t$ are unbiased estimators of the mean and variance of g
- Adam has become a go-to choice for many machine learning practitioners

Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. Int. Conf. on Learning Representations, p. 1–13, 2015.

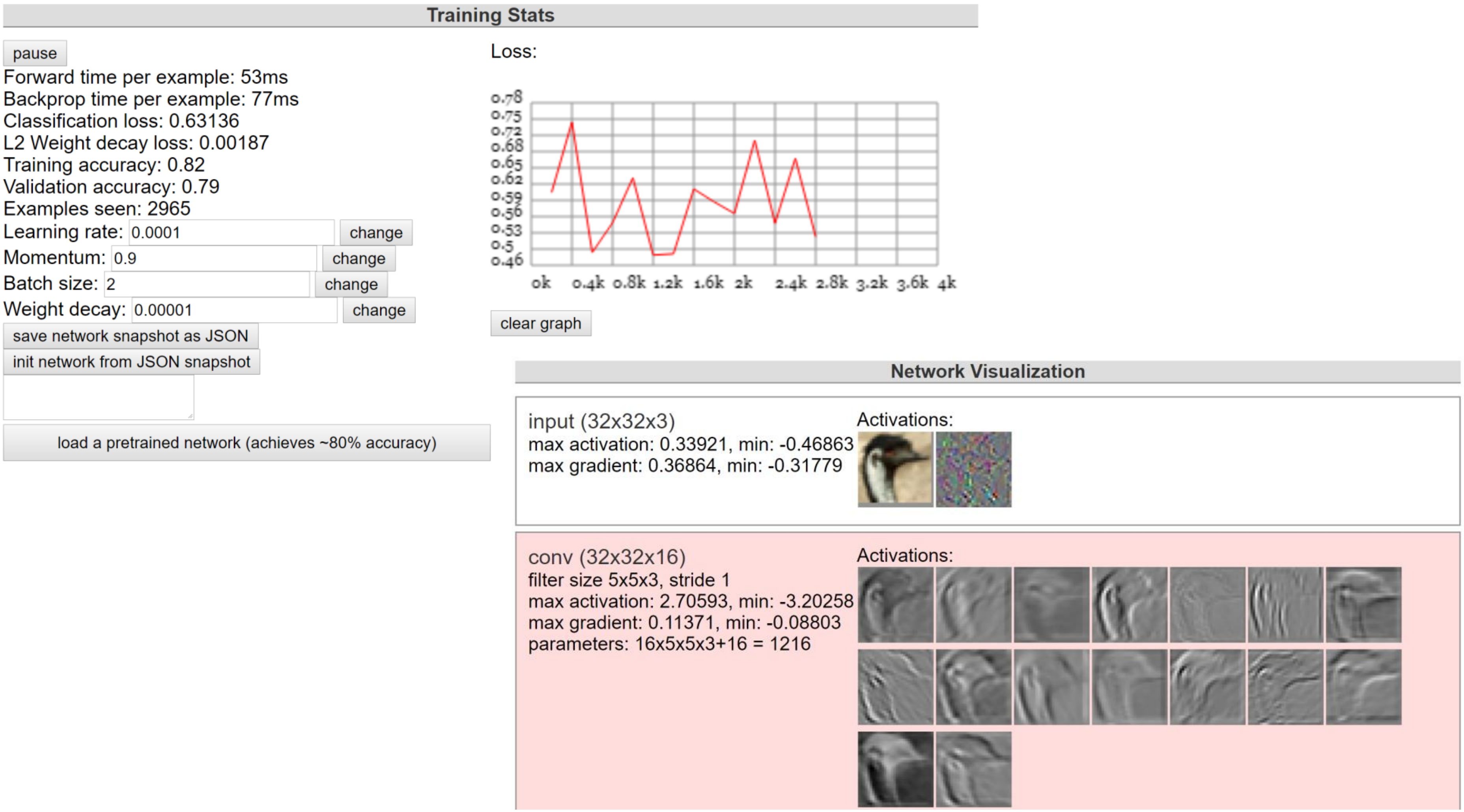
hyperparameters

convolutional neural network for the MNIST task

CNN hyperparameters (MNIST case study)

- 4 trainable layers: 2 convolutional + 2 dense
- layer 1: 20 x 5x5 filters, ReLU activation
- layer 2: 40 x 5x5 filters, ReLU activation
- layer 3: dense with 100 units, ReLU activation
- layer 4: output layer, softmax activation
- padding: 0
- stride: 1

ConvNetJS CIFAR-10 demo



https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

CNN

application scope

application scope

- CNN is able to detect and characterize local spatial patterns
- typical applications on CNN-based images:
    - image classification
    - detection (and location) of objects in images
    - image reconstruction, noise removal
    - image fusion
    - super-resolution

    …

limitations

- CNN might not be useful if data has no spatial or temporal order