Achievements and challenges of semantic image segmentation with deep learning

E. Decencière

MINES ParisTech
PSL Research University
Center for Mathematical Morphology

Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
5. Image segmentation
6. 3D
7. Conclusion

Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
5. Image segmentation
6. 3D
7. Conclusion

What is deep learning?

Definition

Deep learning is the branch of machine learning that studies artificial neural networks.

$$\text{Deep Learning} \subset \text{Machine Learning} \subset \text{Artificial Intelligence}$$

A revolution in image analysis

- 2010: graph cuts, snakes, mathematical morphology, machine learning, etc.
- Today: deep learning.

The rise of deep learning

Nature, 2016

Differentiable programming

Definition

Differentiable programming is an algorithmic framework where differentiable operators are combined together to build complex systems. The parameters of the system can then be optimized via gradient descent thanks to back-propagation.

The object of study of differential programming are *computational graphs*.

Computational graph

Definition

A computational graph is a direct acyclic graph such that:

- A node is a mathematical operator
- To each edge is associated a value (scalar, vector, matrix, tensor, ...)
- Each node can compute the values of its output edges from the values of its input edges

Computing a *forward pass* through the graph means choosing its input values, and then progressively computing the values of all edges.

Computational graph example

Computational graph of:

$$\sigma(w_1 x + w_2 y + b)$$

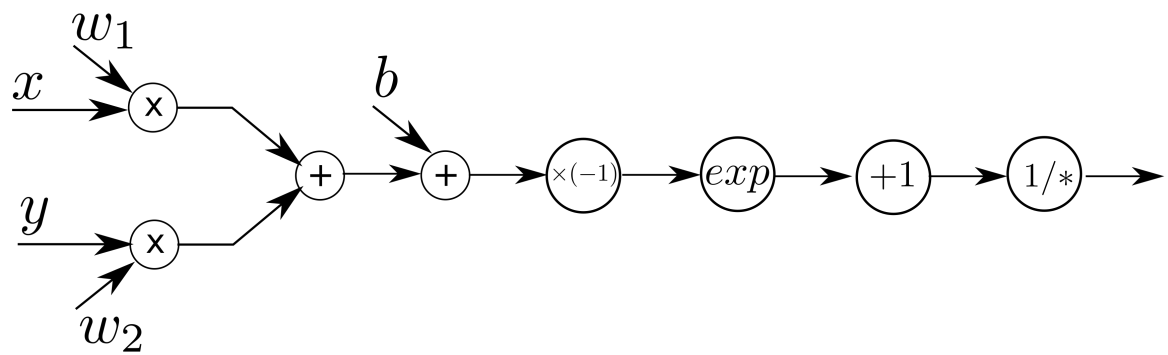
where σ is the sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$

Computational graph example

Computational graph of:

$$\sigma(w_1 x + w_2 y + b)$$

where σ is the sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$

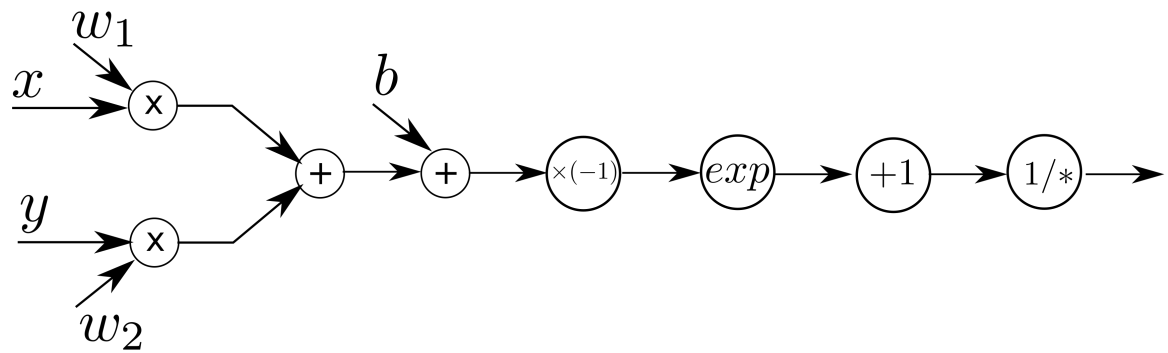


Computational graph example

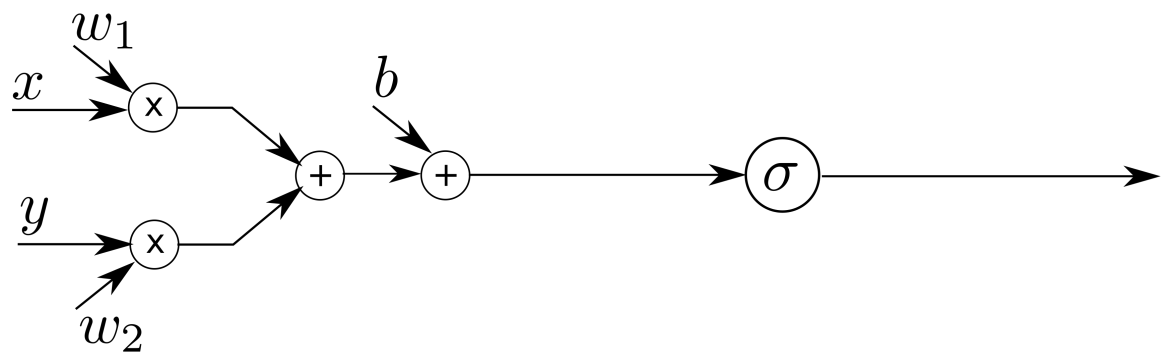
Computational graph of:

$$\sigma(w_1 x + w_2 y + b)$$

where σ is the sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$



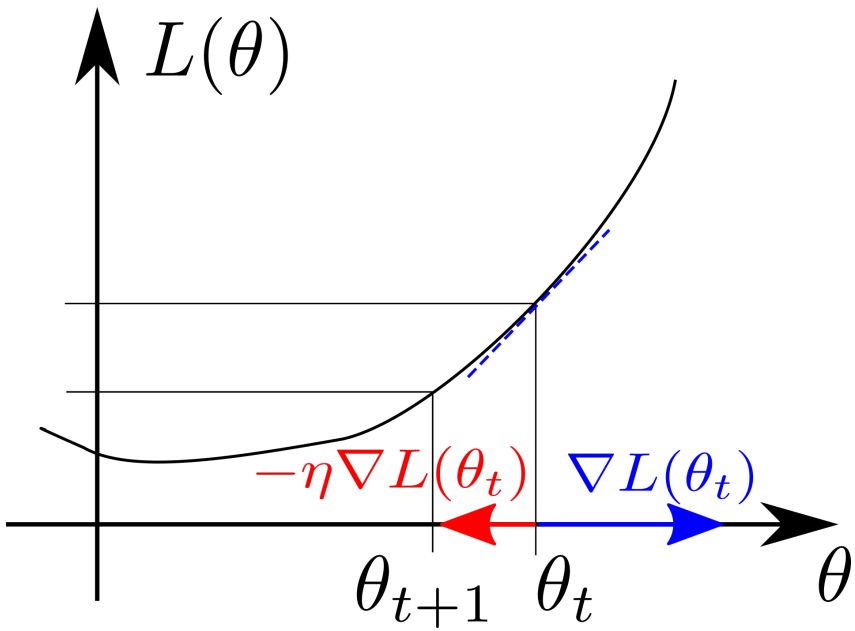
The graph can be represented at different levels of detail:



Setting the parameters of the computational graph

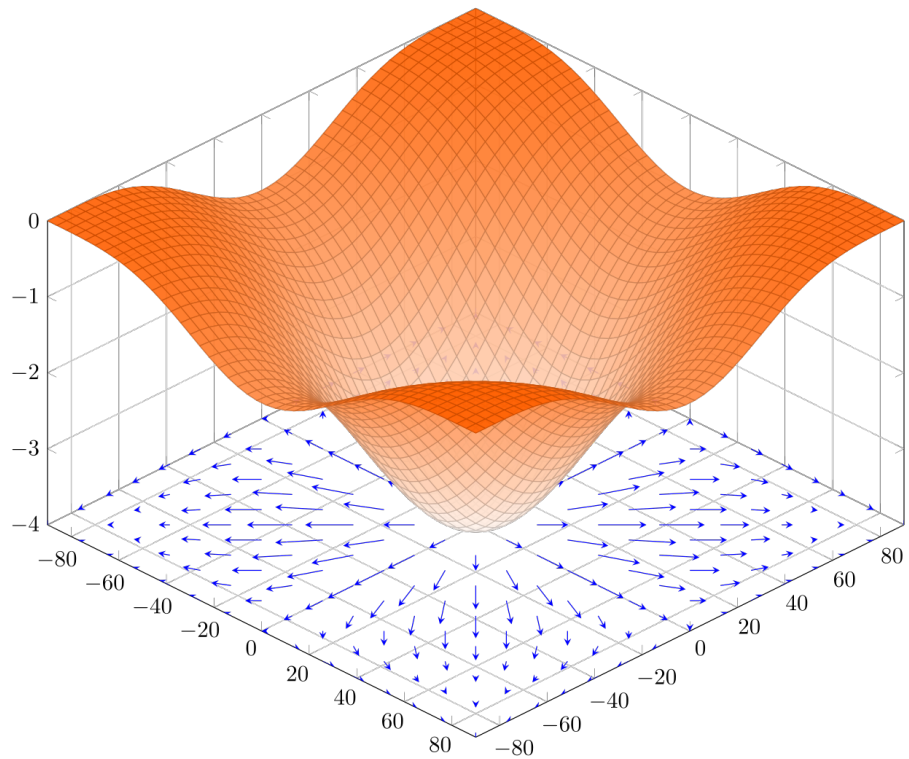
- A computational graph is used to compute a transformation between its inputs and outputs.
- In order to set its parameters, a loss function is chosen. Its minimization via gradient descent allows to set the parameters of the model.

Gradient descent illustration (1D)



$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\nabla L(\boldsymbol{\theta}_t)$$

Gradient descent illustration (2D)



Definition: gradient

Let L be a derivable function from $\mathbb{R}^n$ into $\mathbb{R}$. Its gradient ∇L is:

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial L}{\partial \boldsymbol{\theta}_1}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial L}{\partial \boldsymbol{\theta}_n}(\boldsymbol{\theta}) \end{pmatrix}$$

Credits: By MartinThoma, CC0, https://commons.wikimedia.org/

Gradient descent applied to computational graphs

In the case of computational graphs, the loss L depends on each parameter $\boldsymbol{\theta}_i$ via the composition of several functions. In order to compute the gradient $\nabla_{\boldsymbol{\theta}} L$ we will make extensive use of the chain rule theorem.

Chain rule theorem

Let f_1 and f_2 be two differentiable real functions ($\mathbb{R} \to \mathbb{R}$). Then for all x in $\mathbb{R}$: :

$$(f_2 \circ f_1)'(x) = f_2'(f_1(x)).f_1'(x)$$

Leibniz notation

Let us introduce variables x, y and z:

$$x \xrightarrow{f_1} y \xrightarrow{f_2} z$$

Then:

$$\frac{\mathrm{d}z}{\mathrm{d}x} = \frac{\mathrm{d}z}{\mathrm{d}y} \cdot \frac{\mathrm{d}y}{\mathrm{d}x}$$

The backpropagation algorithm

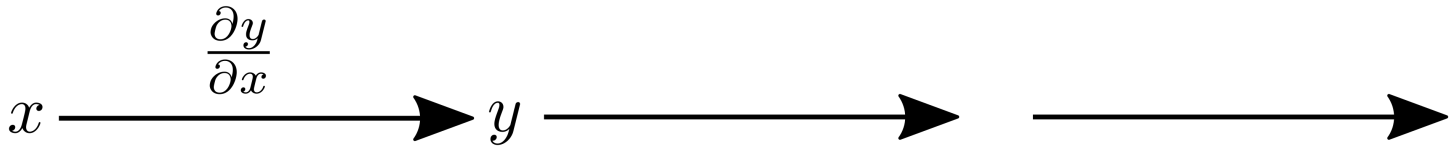
- The backpropagation algorithm is used in a computational graph to efficiently compute the partial derivatives of the loss with respect to each parameter of the network.
- One can trace the origins of the method to the sixties
- It was first applied to NN in the eighties
  [Werbos, 1982, LeCun, 1985]

The backpropagation algorithm: intuition

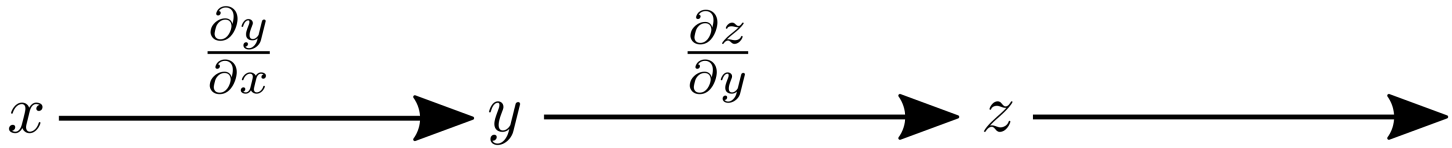
- Given a computational graph, the main idea is to compute the local partial derivatives during a forward pass
- Then, during a backward pass, the partial derivatives of the loss with respect to each parameter is computed

Simple backpropagation example

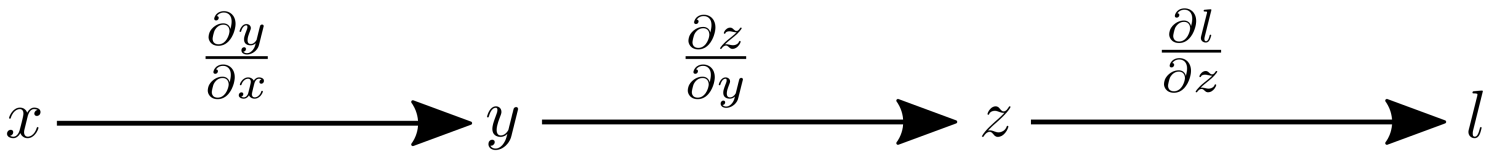
Simple backpropagation example



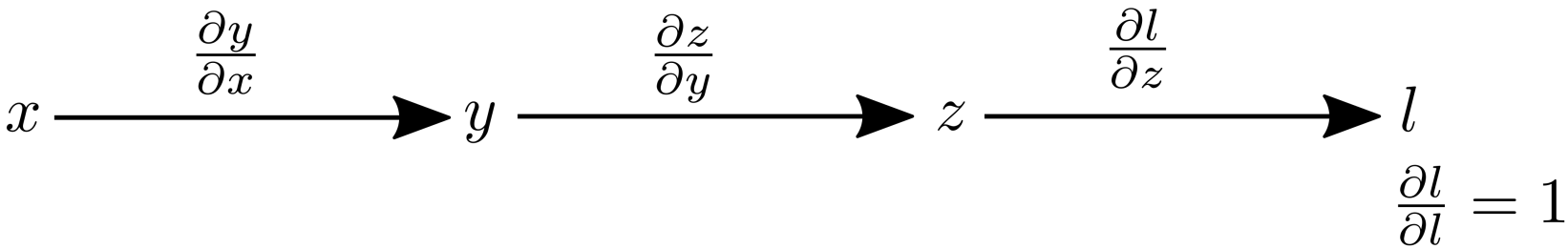
Simple backpropagation example



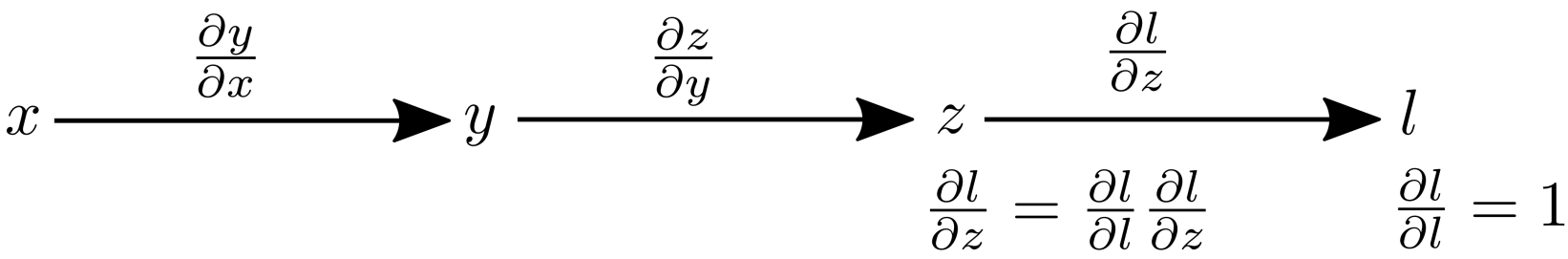
Simple backpropagation example



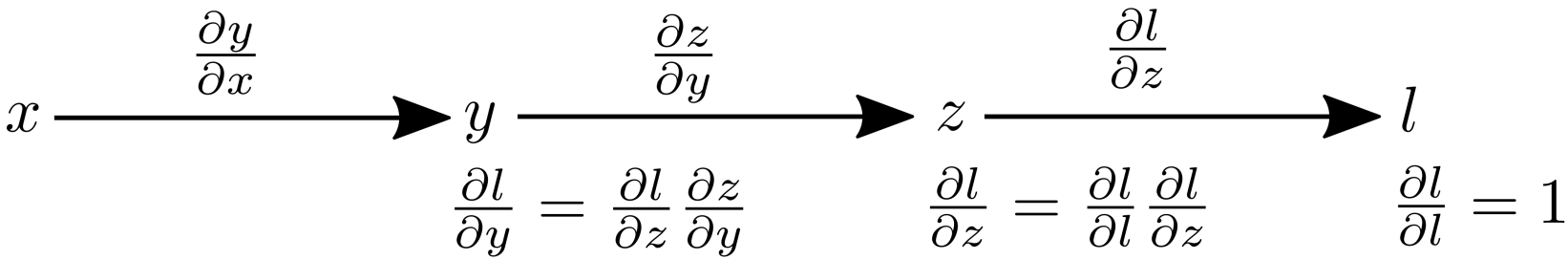
Simple backpropagation example



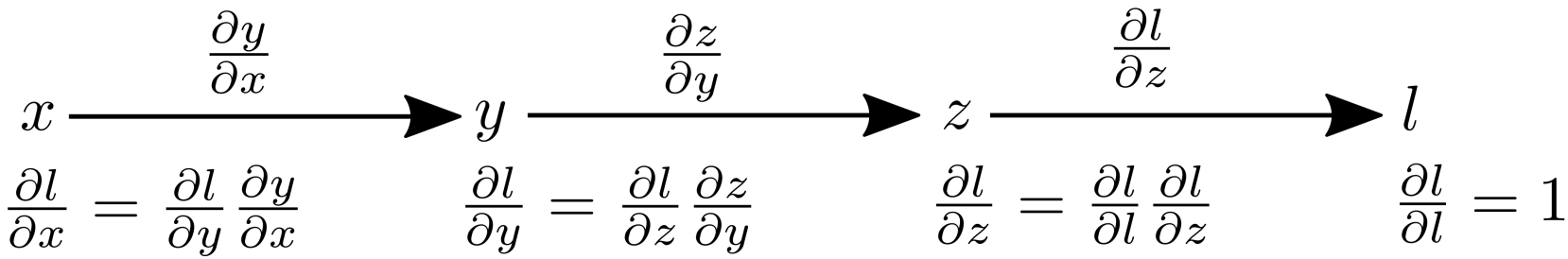
Simple backpropagation example



Simple backpropagation example



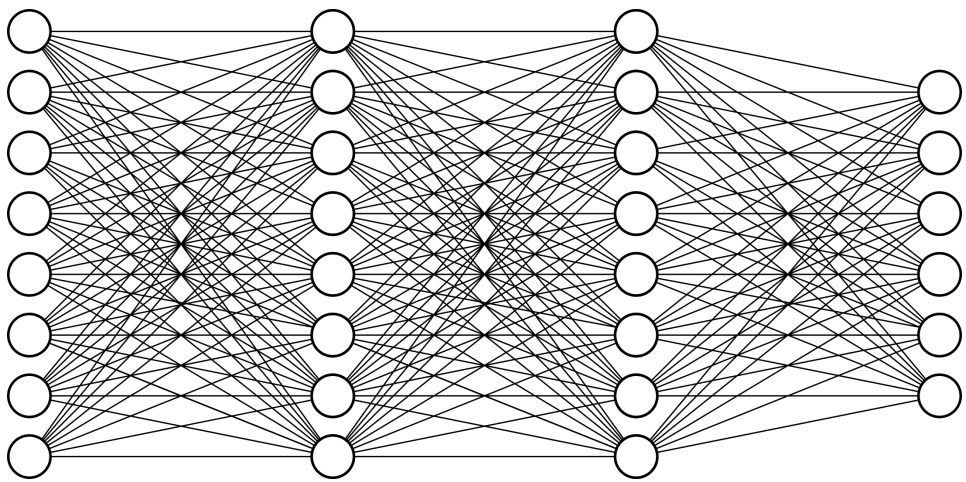
Simple backpropagation example



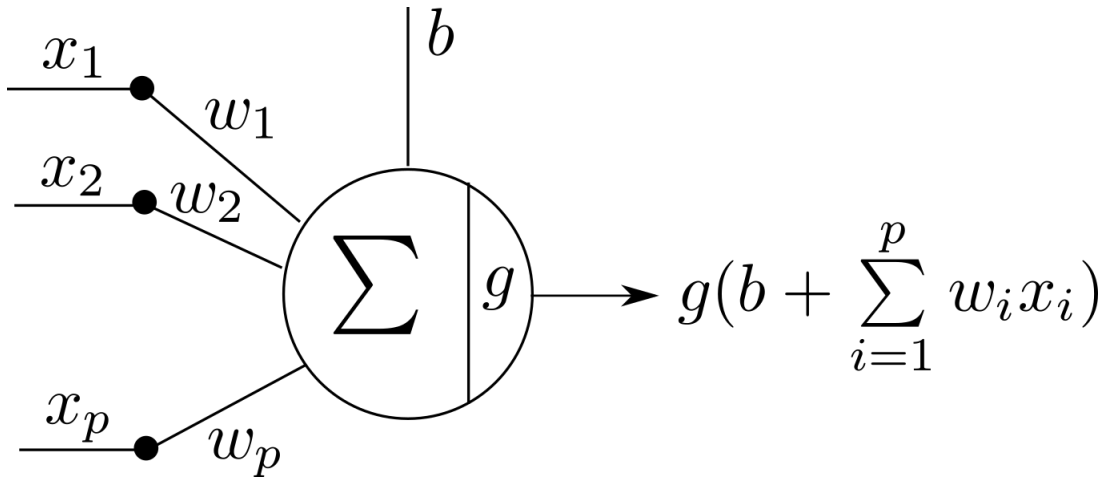
Feed-forward neural networks

Definition

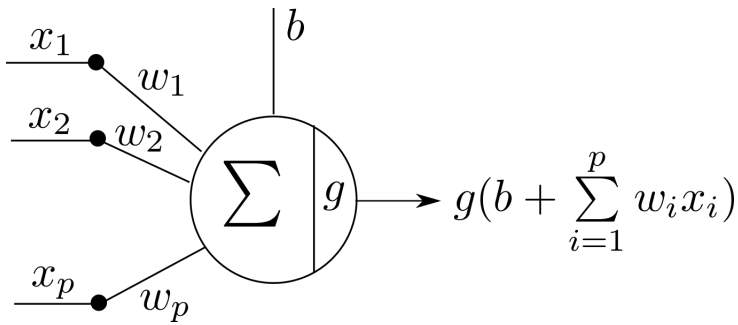
- A feed-forward neural networks is a computational graph
- Its computing units, the neurons, are organized in layers



Artificial neuron



Notations



With

$$\mathbf{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_p \end{pmatrix} = (w_1, \ldots, w_p)^T$$

and

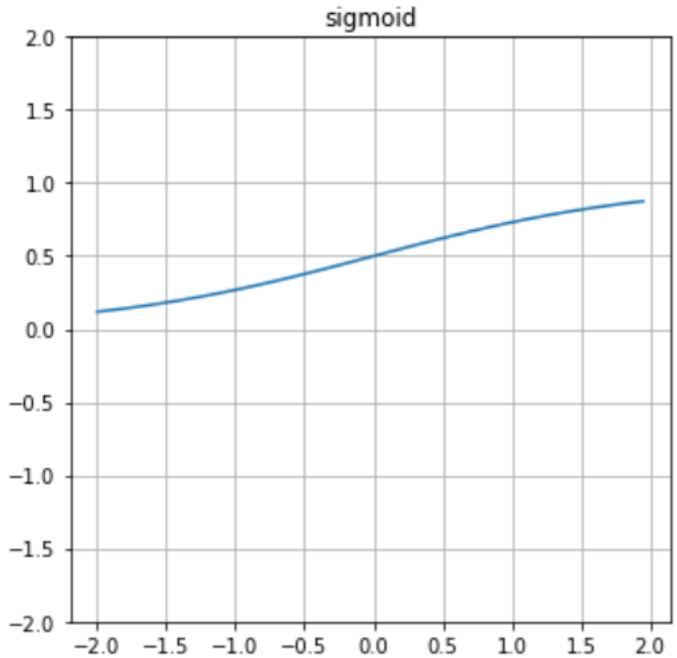
$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix} = (x_1, \ldots, x_p)^T$$

We can simply write:

$$\mathrm{g}(b + \sum_{i=1}^{p} w_i x_i) = \mathrm{g}(b + \mathbf{w}^T \mathbf{x})$$

Activation: sigmoid

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

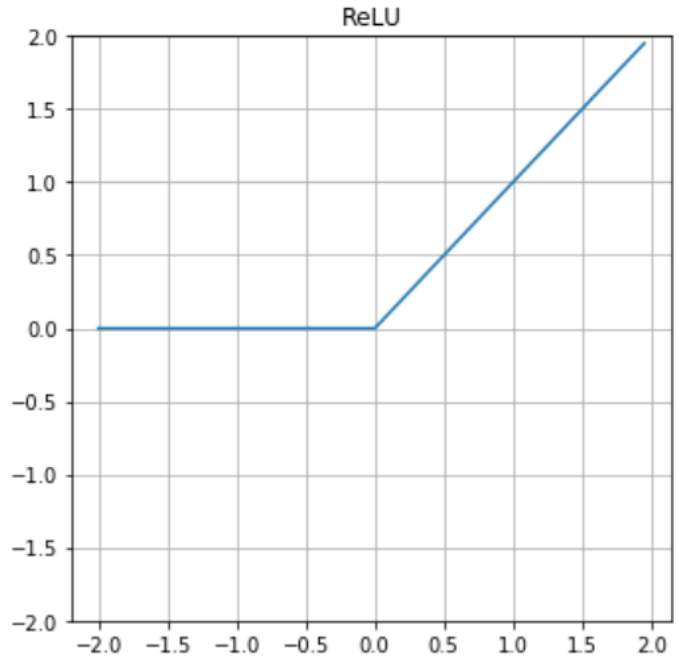


Remarks

+ Similar to binary activation, but with usable gradient
  - However, gradient tends to zero when input is far from zero
  - More computationally intensive

Activation: rectified linear unit (ReLU)

$$\mathrm{g}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

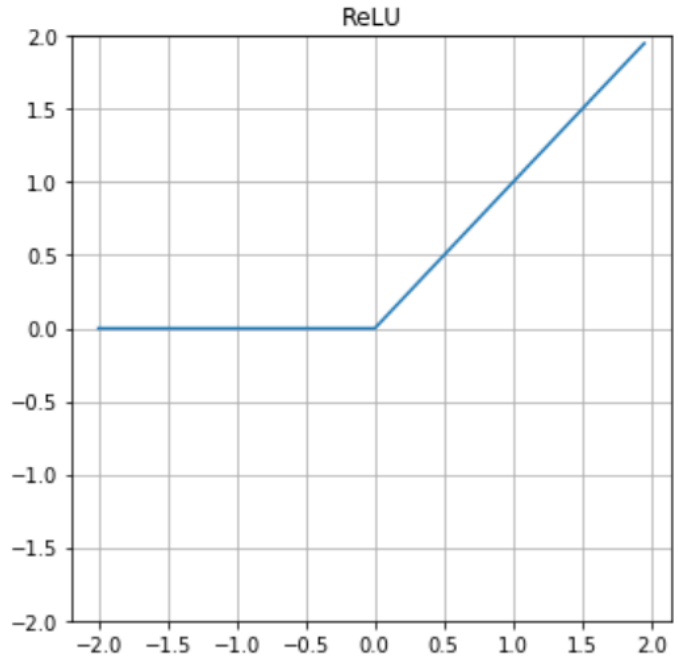


Remarks

+ Usable gradient when activated
+ Fast to compute
+ High abstraction

Activation: rectified linear unit (ReLU)

$$\mathrm{g}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$



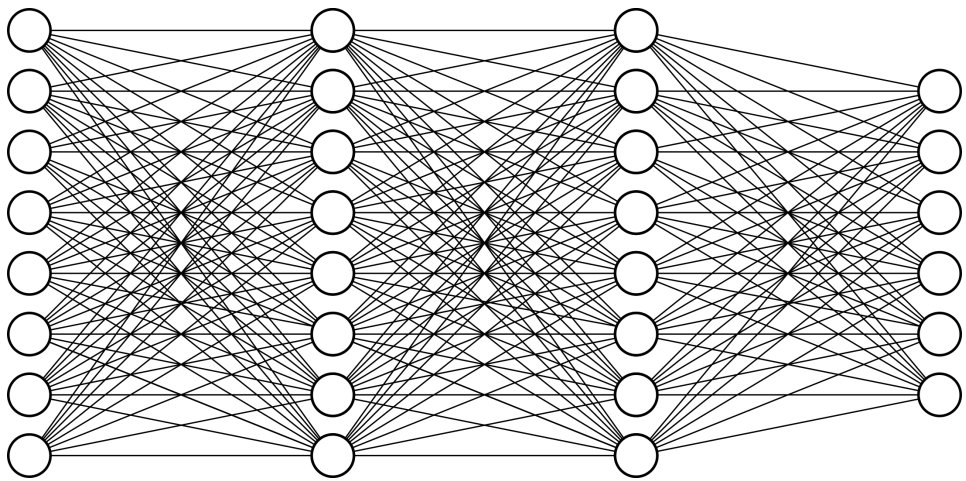
Remarks

+ Usable gradient when activated
+ Fast to compute
+ High abstraction

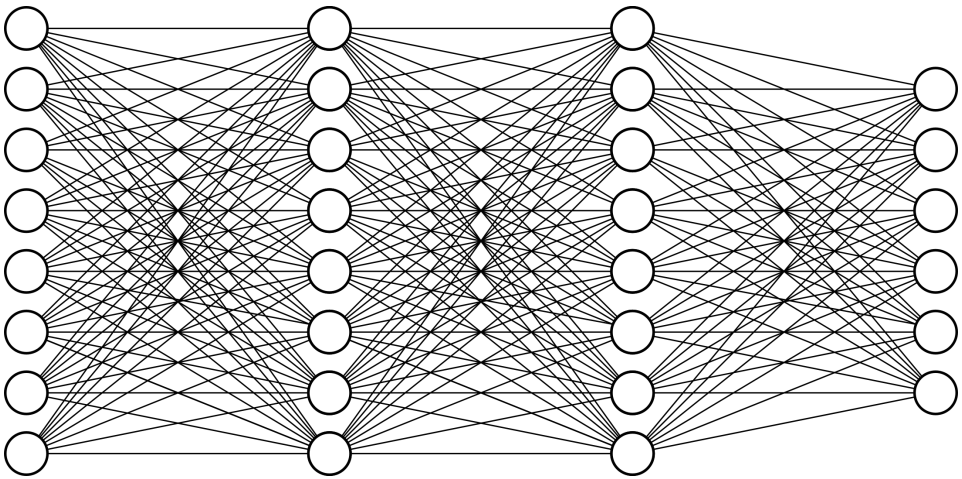
ReLU is the most commonly used activation function.

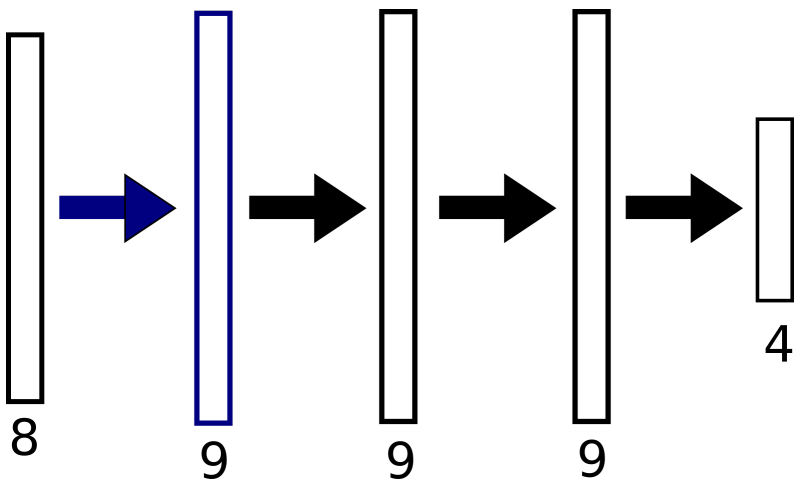
Fully-connected layer

- A layer is said to be fully-connected (FC) if each of its neurons is connected to all the neurons of the previous layer
- If a FC layer contains r neurons, and the previous layer q, then its weights are a 2D dimensional array (a matrix) of size $q \times r$



Graphical representation of NNs





- Data is organized into arrays, linked with operators
- A layer corresponds to an operator between arrays.

Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
5. Image segmentation
6. 3D
7. Conclusion

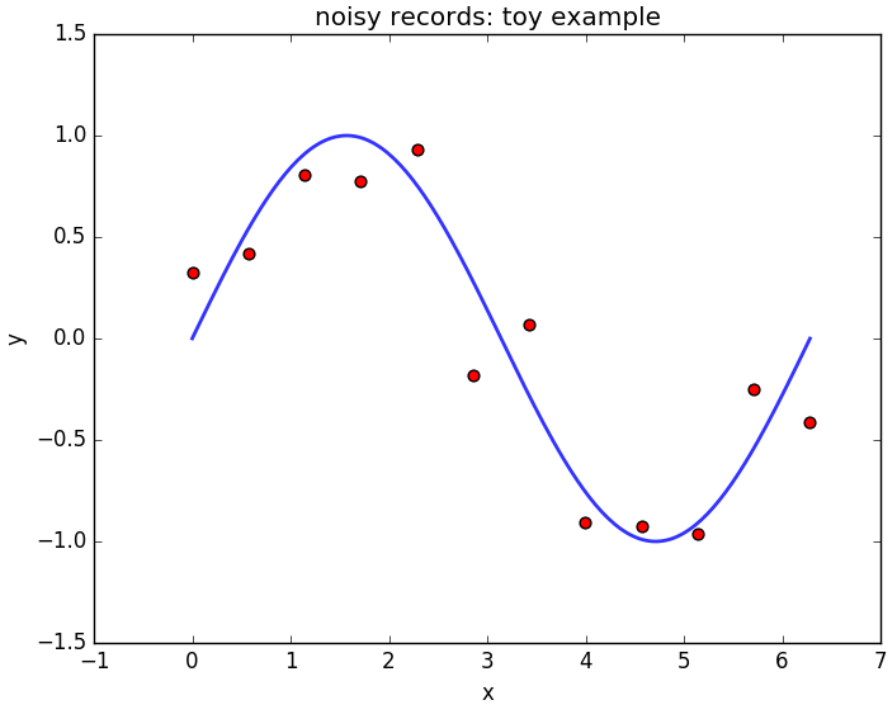
Machine Learning: basic definitions

- Machine Learning aims at predicting some output y from an input (or measurement) x:

$$y = f(x) \tag{1}$$

- In this formulation, Machine Learning aims at finding (learning) f from available data.
- The data that is used to learn f is called **training set**, denoted by $\mathbf{X}$.
- In this general formulation, there is no particular limitation as to the mathematical nature of x and y.

A simple example: polynomial curve fitting[1]



- From a set of measured points (x_i, y_i) (red), we would like to build a model to predict the value y for any given x.
- The true function is $g(x) = \sin(x)$ (displayed in blue).
- The measurements y_i are noisy outputs of that function, i.e.

$$y_i = \sin(x_i) + \epsilon \ , \qquad \epsilon \sim \mathcal{N}(0, 0.2) \tag{2}$$

[1] Example adapted from [Bishop, 2006]

A simple example: polynomial curve fitting

- We use the following polynomial model:

$$\begin{aligned} f(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m \\ &= \boldsymbol{\theta}^T \boldsymbol{\phi}(x) \end{aligned} \tag{3}$$

- Parameter vector: $\boldsymbol{\theta} = (a_0, a_1, \ldots, a_m)^T$
- Here, the initial measurement x is a scalar. In our model, we map x to a higher dimensional space:

$$\begin{aligned} \boldsymbol{\phi} : \mathbb{R}^P &\rightarrow \mathbb{R}^Q \\ x &\rightarrow \boldsymbol{\phi}(x) = (1, x, x^2, \ldots, x^m)^T \end{aligned} \tag{4}$$

- The model is linear in the parameters θ and linear in $\boldsymbol{\phi}$, but for $m > 1$, the model is not linear in x.

A simple example: polynomial curve fitting

- One classical approach is to minimize the least squared error between measured and predicted values:

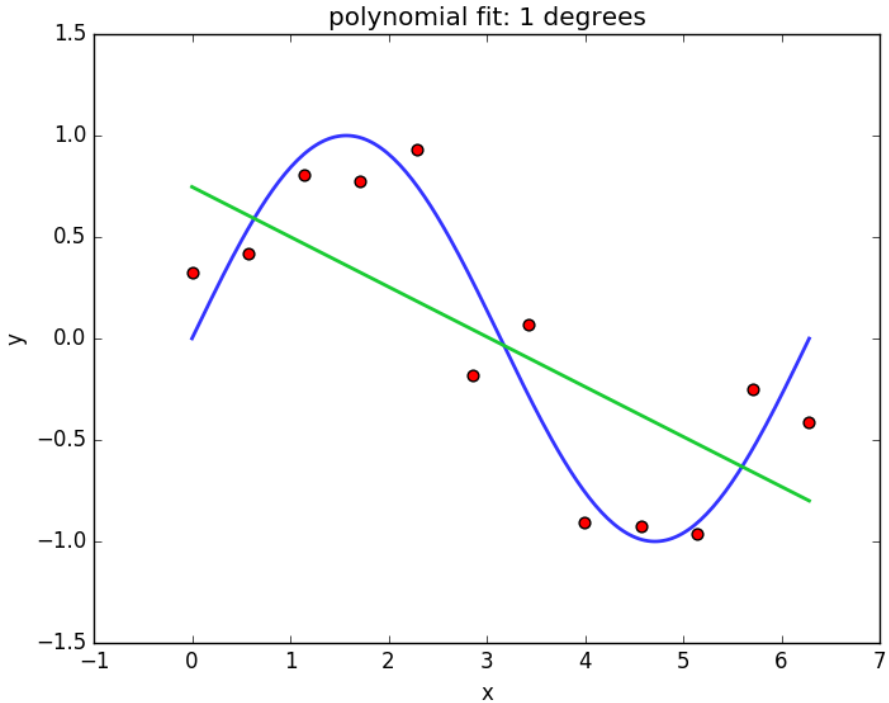
$$\begin{aligned} \min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) &= \min_{\boldsymbol{\theta}} \sum_{i=1}^{N} (y_i - f(x_i))^2 \\ &= \min_{\boldsymbol{\theta}} \sum_{i=1}^{N} (y_i - \boldsymbol{\theta}^T \boldsymbol{\phi}(x_i))^2 \end{aligned} \tag{5}$$

- This can be achieved by setting the gradient with respect to $\boldsymbol{\theta}$ to zero:

$$\nabla_{\boldsymbol{\theta}} L = (\frac{\partial L}{\partial a_0}, \frac{\partial L}{\partial a_1}, \ldots, \frac{\partial L}{\partial a_m})^T = 0 \tag{6}$$

- Unlike most optimization problems in this course, this leads to an analytical solution for $\boldsymbol{\theta}$. This is known as **linear regression**. For more details, we refer to [Hastie et al., 2009].

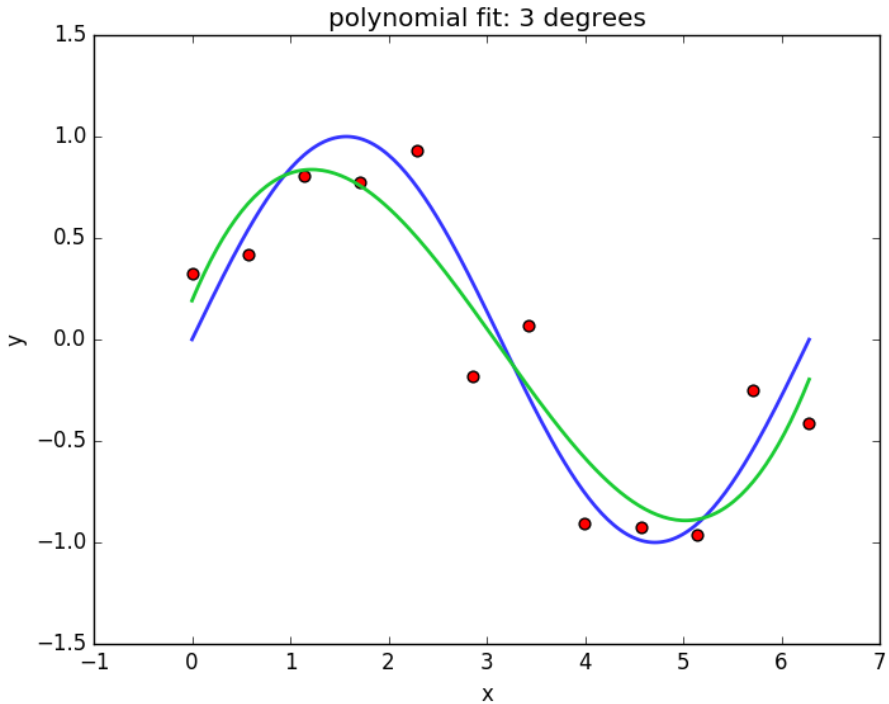
Overfitting and underfitting



$$\|\boldsymbol{\theta}\|^2 = 0.67$$

For $m = 1$, the model is linear in its inputs. The solution is not capable of modeling the measured data points; we get a poor approximation of the original function. The family of functions we have used was not complex enough to model the true data distribution. We also speak of **underfitting**.

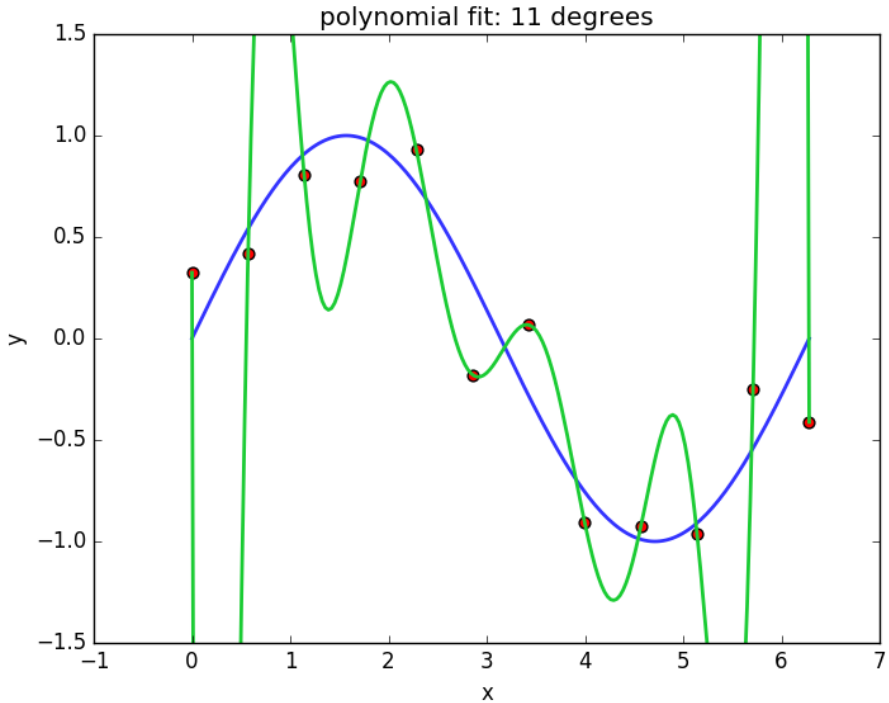
Overfitting and underfitting



$$\|\boldsymbol{\theta}\|^2 = 1.72$$

For $m = 3$, we obtain a solution that seems to be quite right: it is sufficiently complex to model the true data distribution, but not too complex to model the small variations which are due to noise.

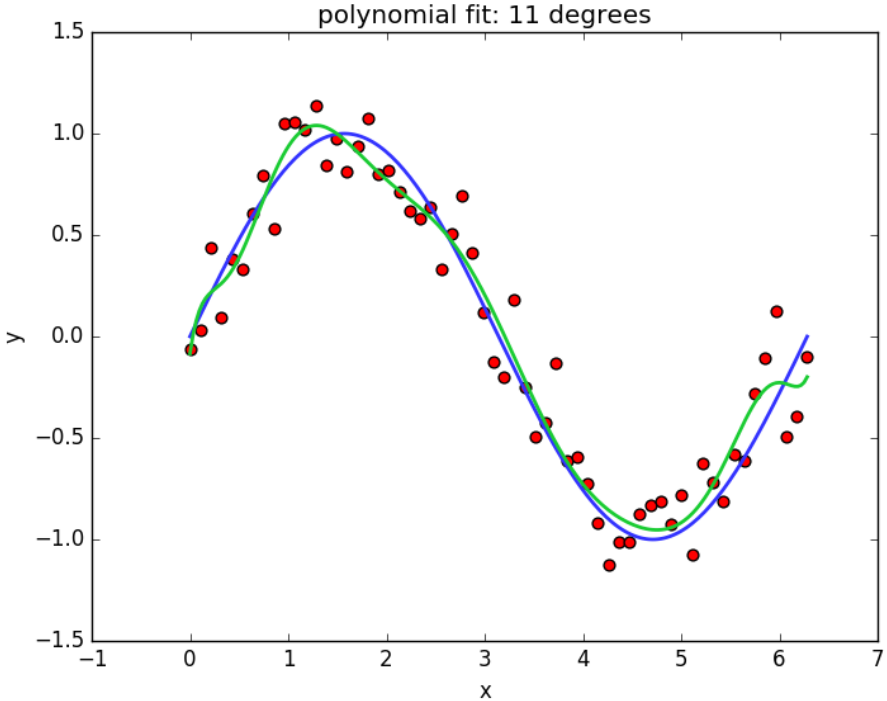
Overfitting and underfitting



$$\|\boldsymbol{\theta}\|^2 \approx 10^7$$

For $m = 11$, we obtain a solution that has zero error (the function passes through every point of the training set). But the coefficients with large absolute values that cancel each other precisely on the training points lead to a highly unstable function. We speak of **overfitting** and **poor generalization**.

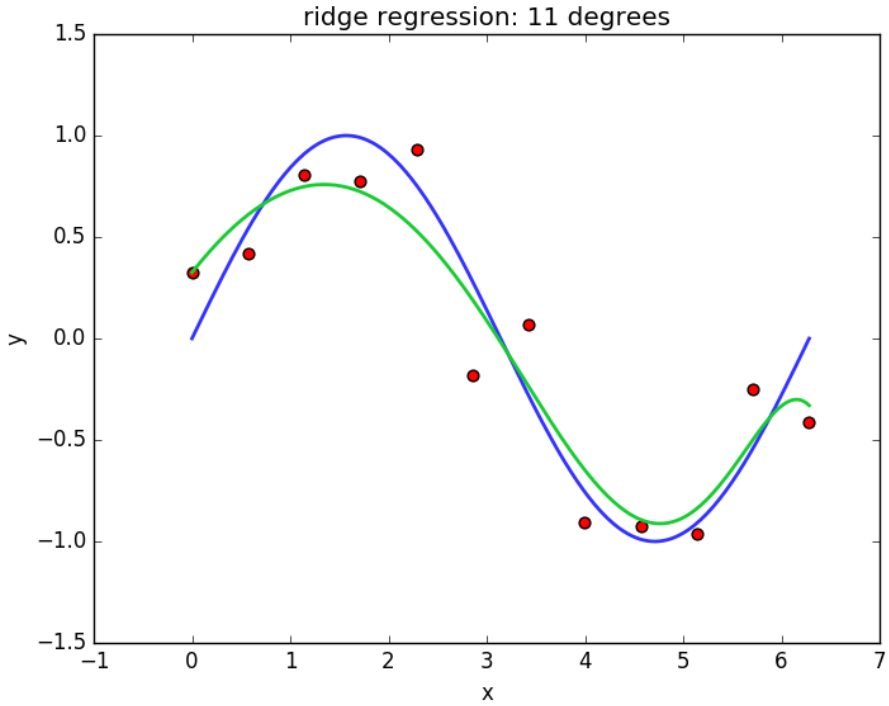
Overfitting and underfitting



$$\|\boldsymbol{\theta}\|^2 = 5647$$

One way of reducing overfitting is to increase the number of samples. Even if the function is complex, it cannot be "too wild", as it has to find a compromise between many training samples. This however implies the annotation (or measurement) of more samples.

Overfitting and underfitting



$$\|\boldsymbol{\theta}\|^2 = 0.41$$

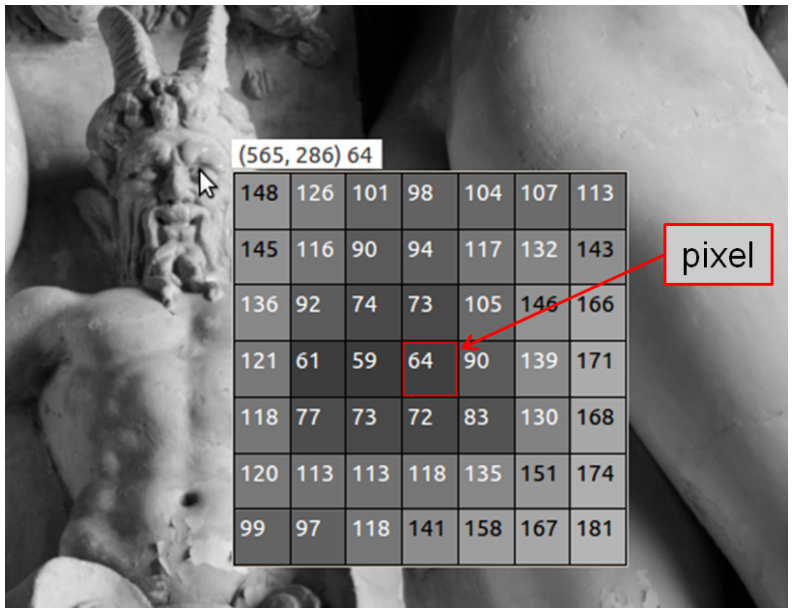
Another way of preventing overfitting without increasing the number of samples, is to add a penalization term in the optimization procedure. This is also known as **regularization**:

$$L = \sum_{i=1}^{N} (y_i - \boldsymbol{\theta}^T \boldsymbol{\phi}(x_i))^2 + \lambda \|\boldsymbol{\theta}\|^2 \tag{7}$$

A picture is worth a thousand words

Definition

- Classically, an image is a matrix of values belonging to $[0, \ldots, 255]$ (grey level images) or to $[0, \ldots, 255]^3$ (color images).
- More generally, an image is a q-dimensional array of values belonging to R^d.



Grey level values around the left eye of the faun

The role of annotated image databases

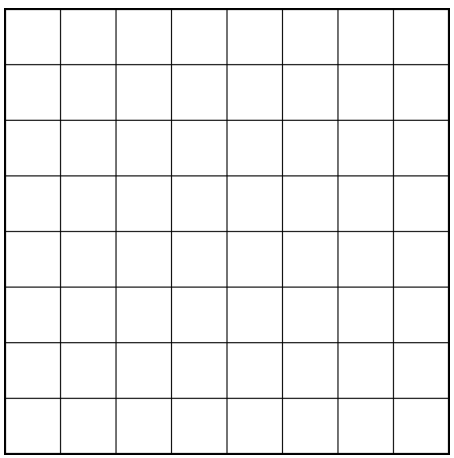
Image databases including *annotations* (typically some kind of high level information) are essential to the development of *supervised* machine learning methods for image analysis.

Annotations

- Image class
- Measure(s) obtained from the image
- Position of objects within the image
- Segmentation

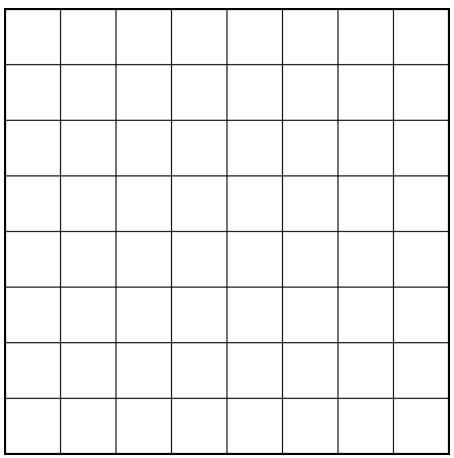
Layers representation

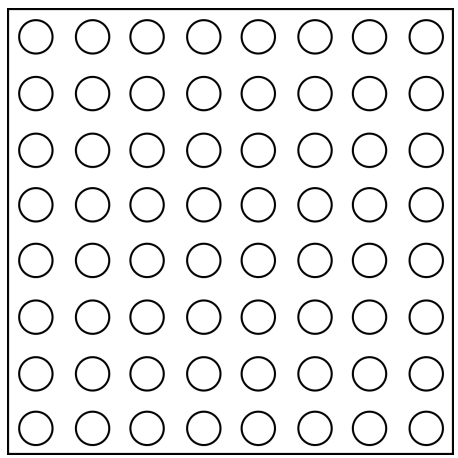
For illustration purposes, in the following slides images and filters will be displayed as rows of neurons – these can be seen as 1D arrays or as sections of 2D arrays.
We represent some connections between neurons. Each such connection is associated to a weight. The bias are not represented, to avoid clutter, but must not be forgotten.



Layers representation

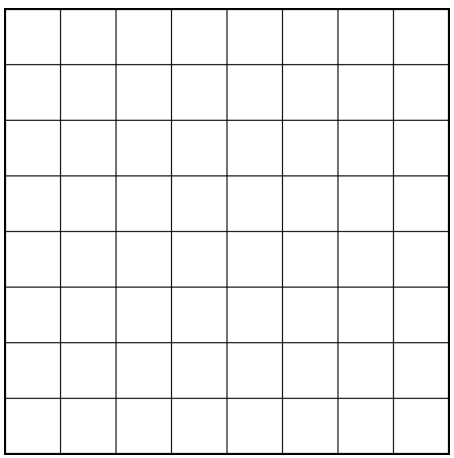
For illustration purposes, in the following slides images and filters will be displayed as rows of neurons – these can be seen as 1D arrays or as sections of 2D arrays.
We represent some connections between neurons. Each such connection is associated to a weight. The bias are not represented, to avoid clutter, but must not be forgotten.

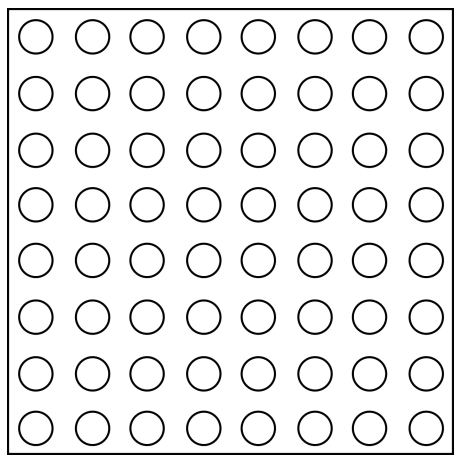


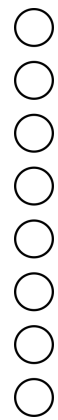


Layers representation

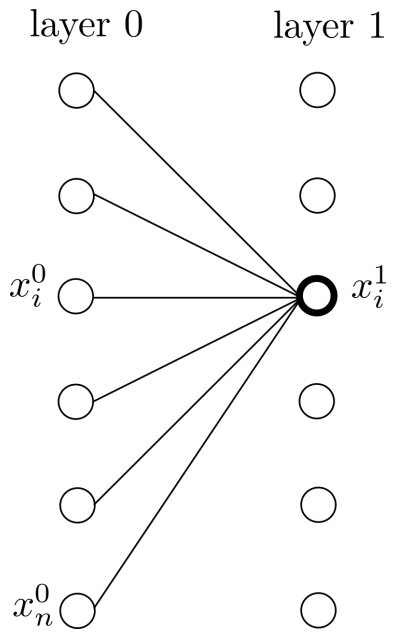
For illustration purposes, in the following slides images and filters will be displayed as rows of neurons – these can be seen as 1D arrays or as sections of 2D arrays.
We represent some connections between neurons. Each such connection is associated to a weight. The bias are not represented, to avoid clutter, but must not be forgotten.





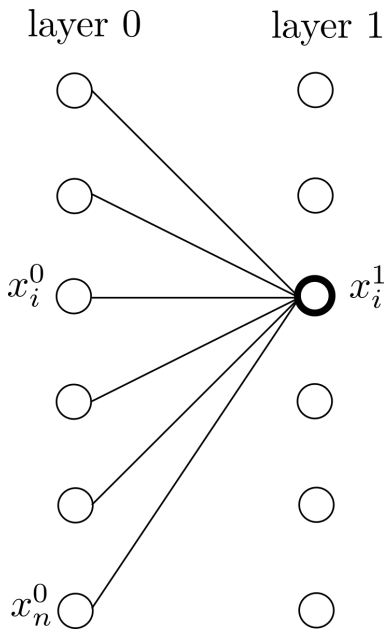


Towards convolutional layers

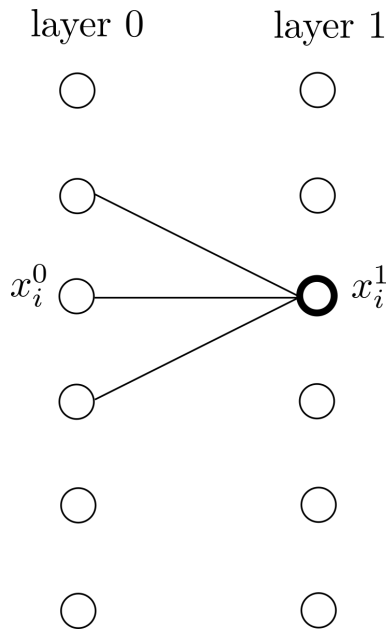


Fully connected layer:
$n(n+1)$ weights

Towards convolutional layers

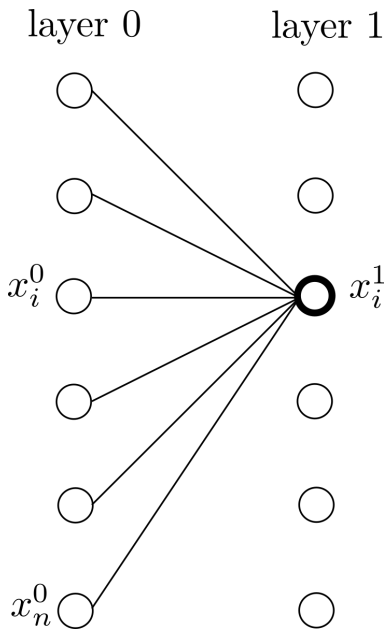


Fully connected layer:
$n(n+1)$ weights

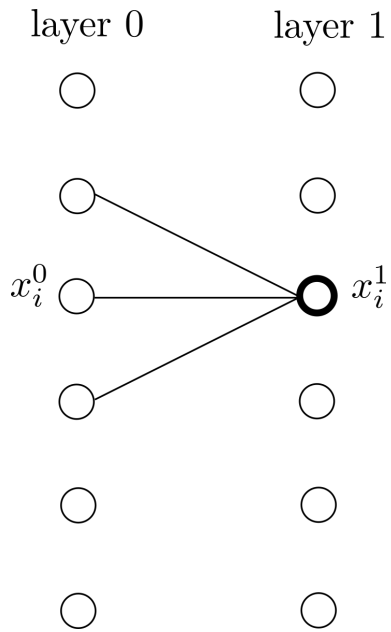


Locally conn. layer:
$n(s+1)$ weights

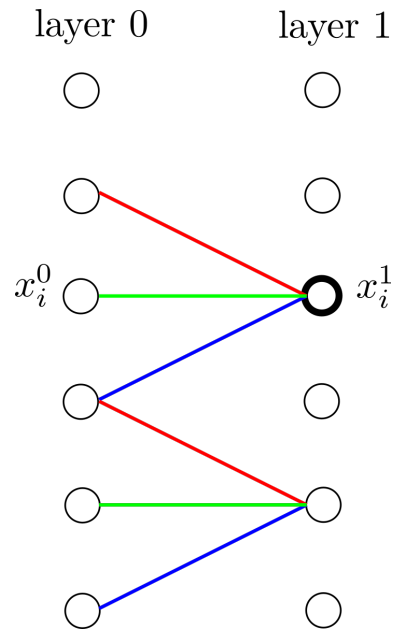
Towards convolutional layers



Fully connected layer:
$n(n+1)$ weights



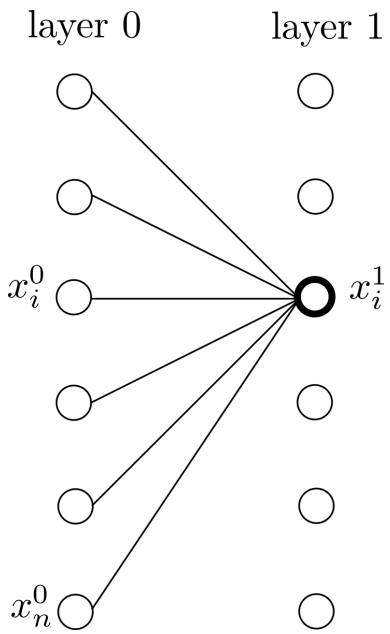
Locally conn. layer:
$n(s+1)$ weights



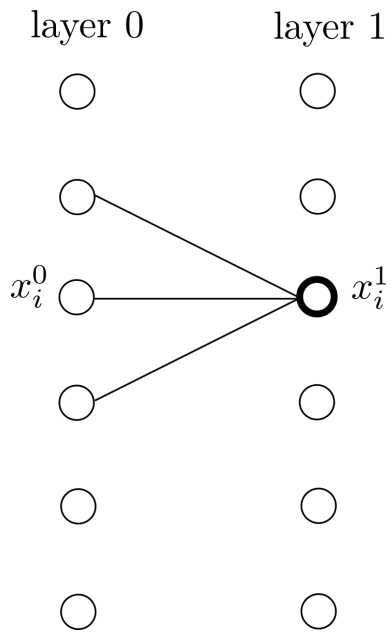
Weight replication: $s+1$ weights.
Convolutional layer.

Towards convolutional layers: some figures

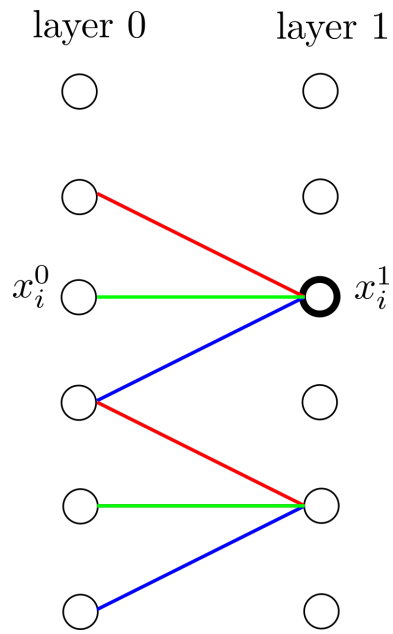
- 3×3 convolutions: $s = 9$
- Toy image: $n = 28 \times 28 = 784$
- Typical image: $n = 1000 \times 1000 = 10^6$



Fully connected layer:
$n(n+1)$ weights
$\approx 6.10^5$
$\approx 10^{12}$



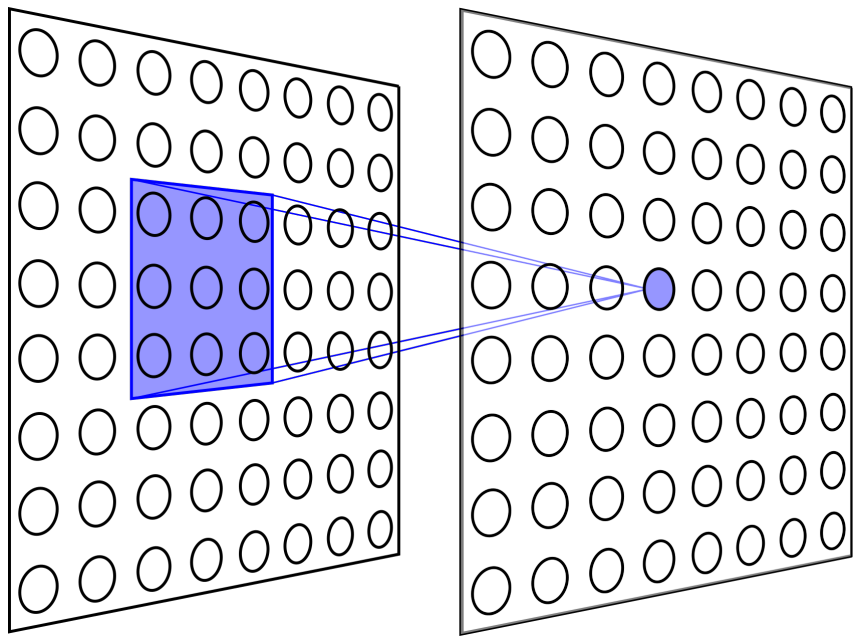
Locally conn. layer:
$n(s+1)$ weights
7840
10^7



Weight replication: $s+1$ weights.
10
10

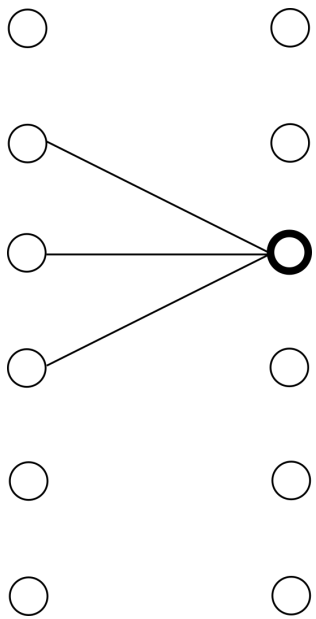
Convolutional layer illustration in 2D

- Illustration of a convolution of size 3×3



Stride

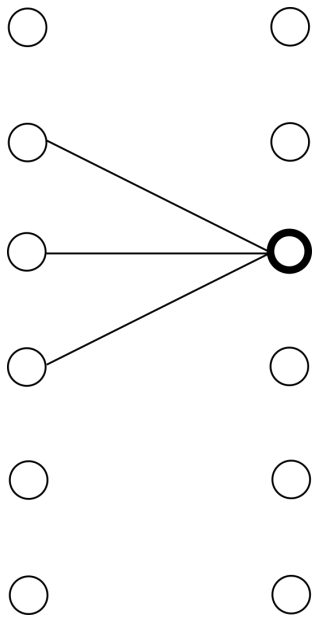
A convolutional layer can at the same time downsample the image by applying a sampling step, or *stride*.



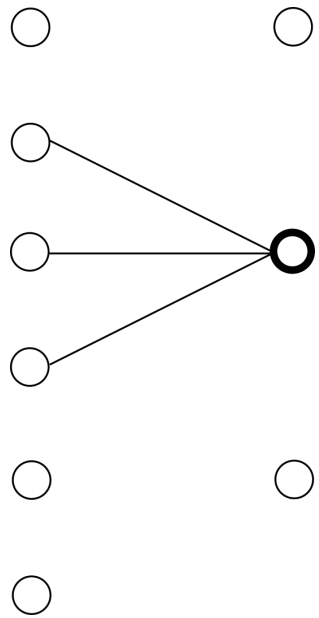
Stride 1

Stride

A convolutional layer can at the same time downsample the image by applying a sampling step, or *stride*.



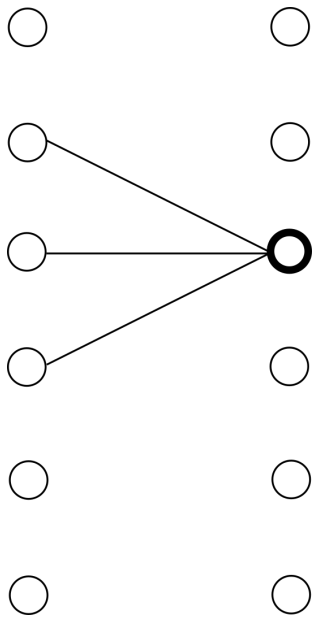
Stride 1



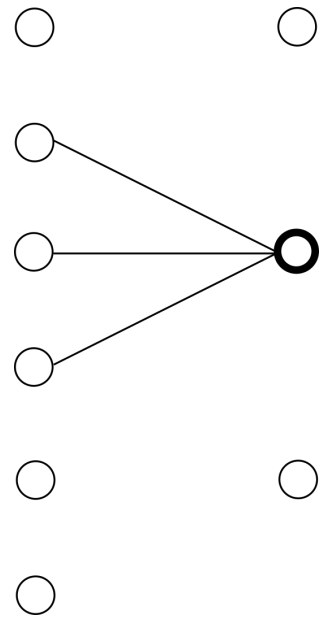
Stride 2

Stride

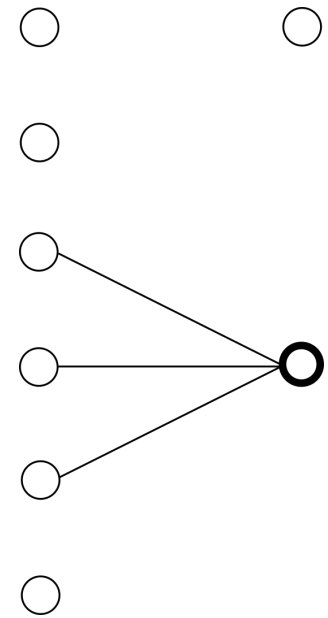
A convolutional layer can at the same time downsample the image by applying a sampling step, or *stride*.



Stride 1

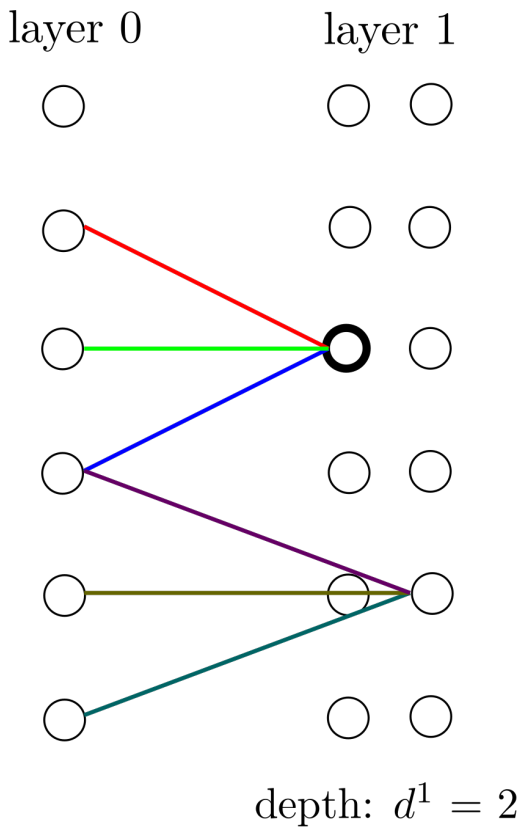


Stride 2



Stride 3

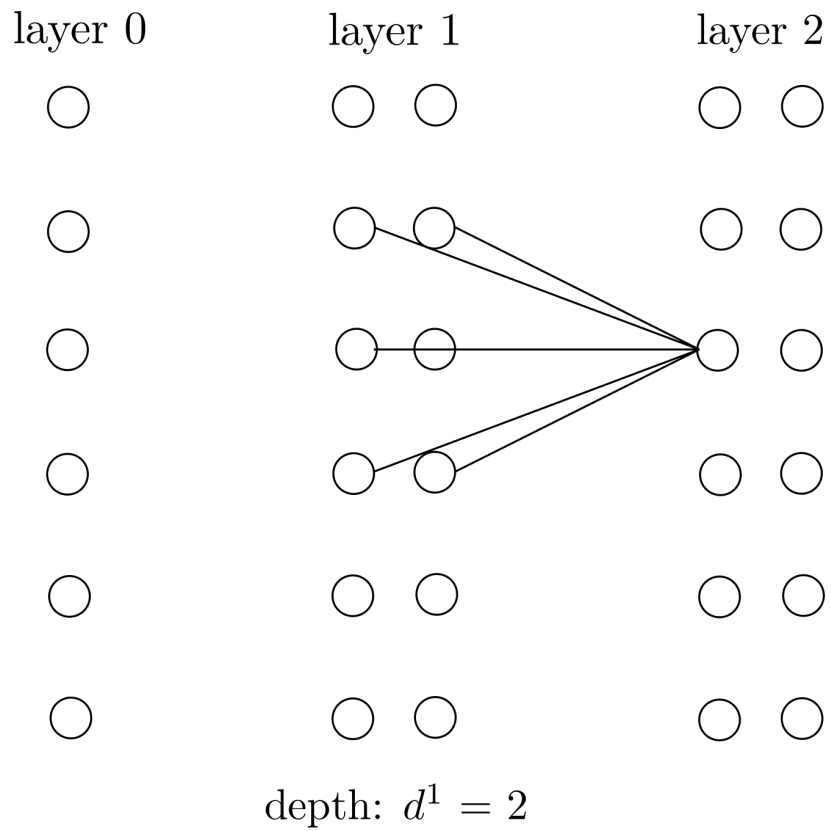
Several filters in the same convolutional layer



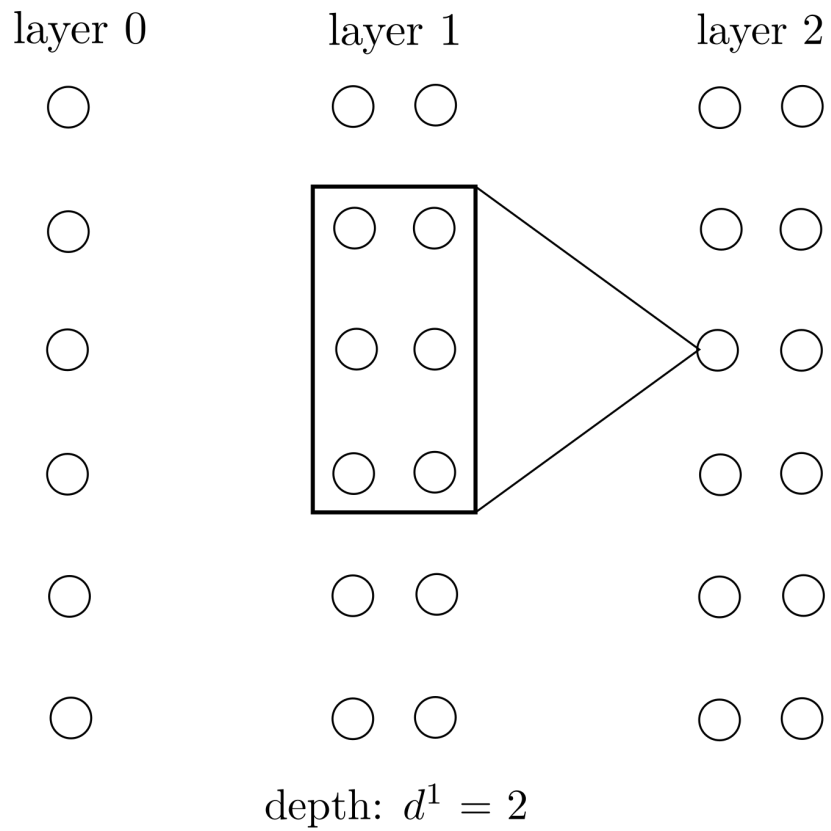
Note on vocabulary

The depth of a layer is often called the number of filters.

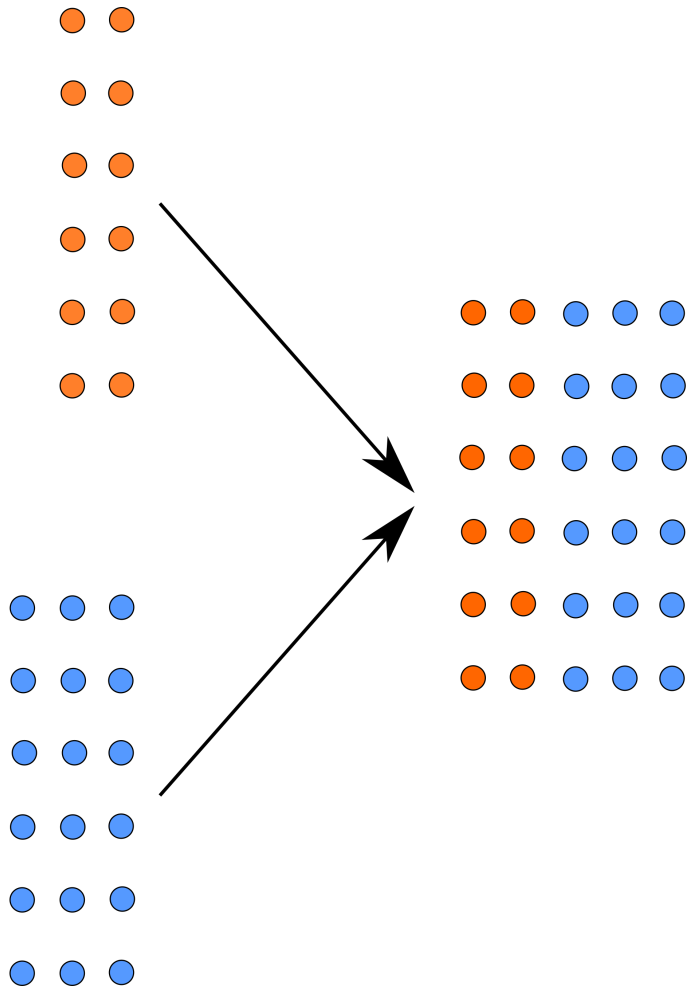
Several filters in the same convolutional layer



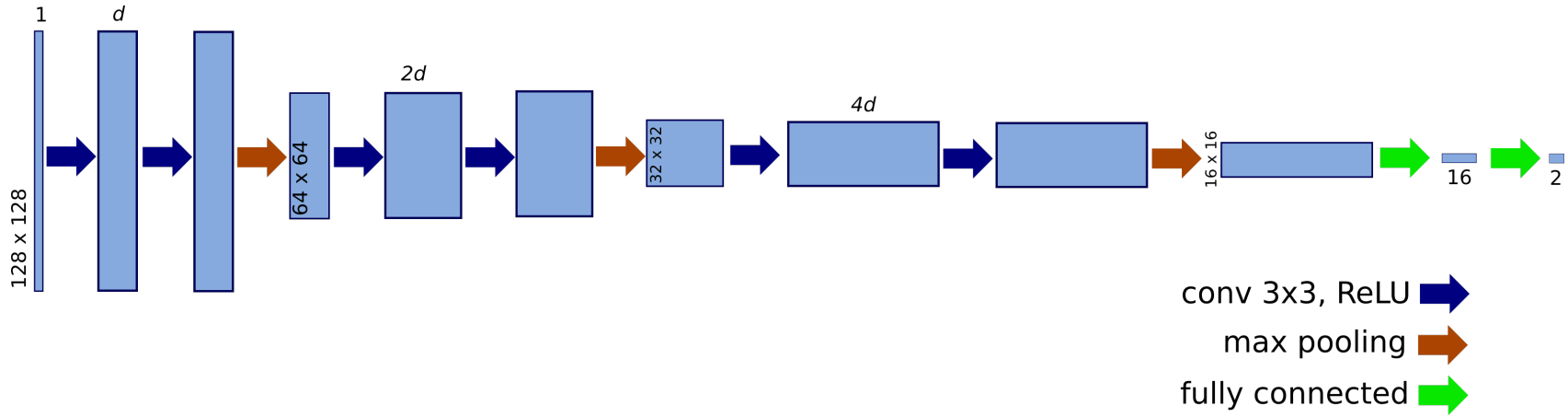
Several filters in the same convolutional layer



Branch merging: concatenation

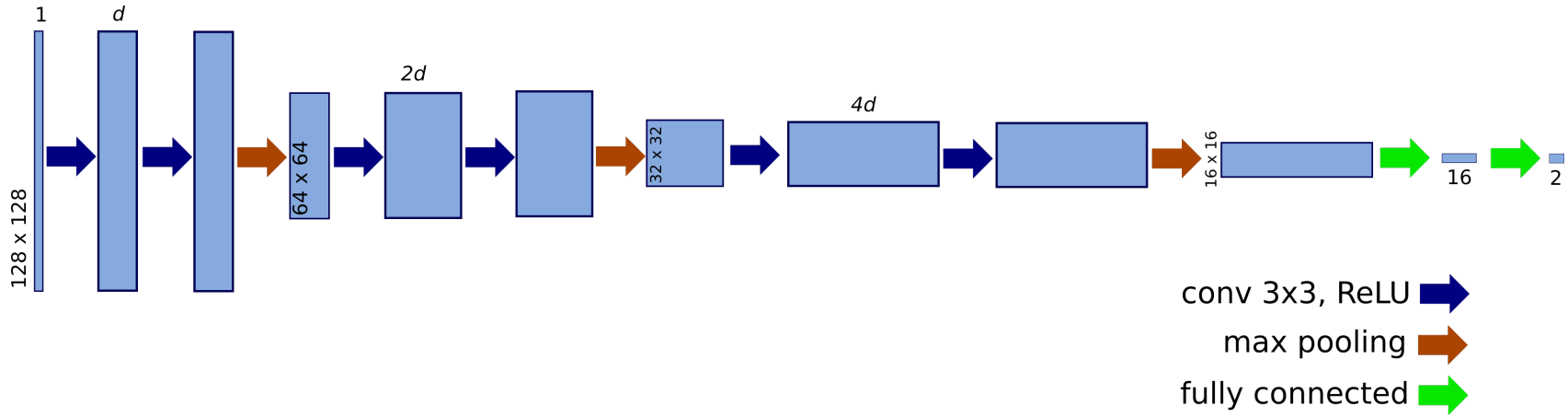


1D representations



Credits: NN is work of Robin Alais et al.
Fundus image by Mikael Häggström, used with permission (CC0).

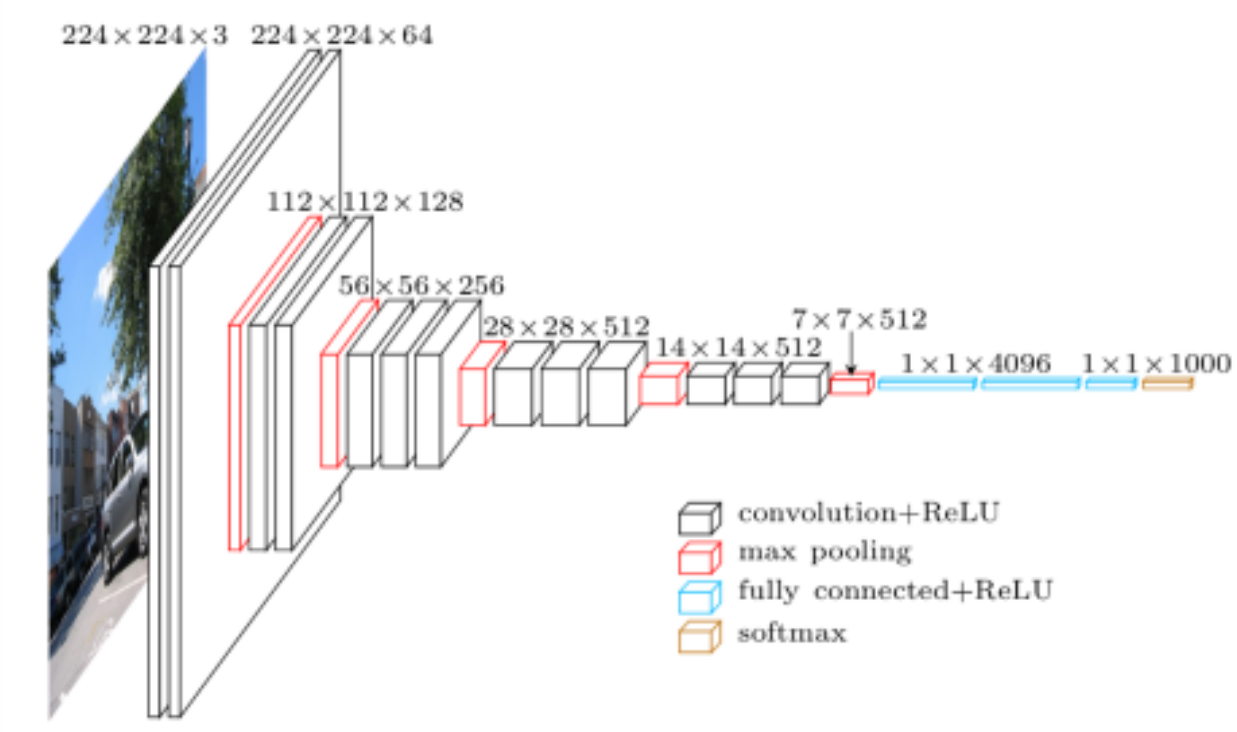
1D representations



This NN was used to estimate the position of the center of the macula on fundus images.

Credits: NN is work of Robin Alais et al. Fundus image by Mikael Häggström, used with permission (CC0).

2D representations

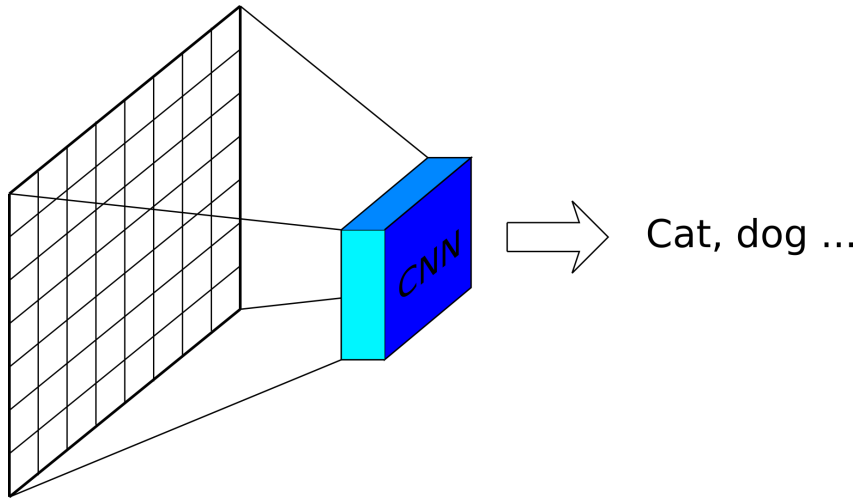


Credits: VGG16 (From https://www.cs.toronto.edu/ frossard/post/

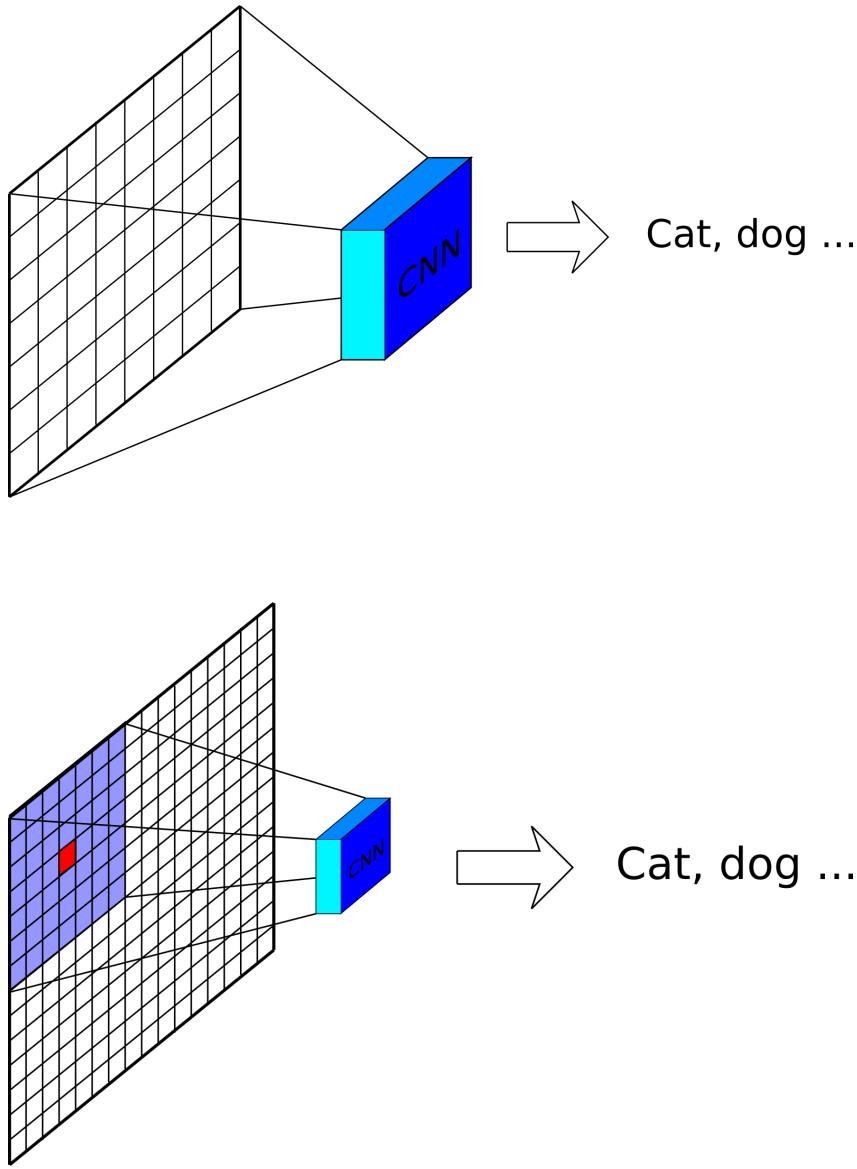
Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
5. Image segmentation
6. 3D
7. Conclusion

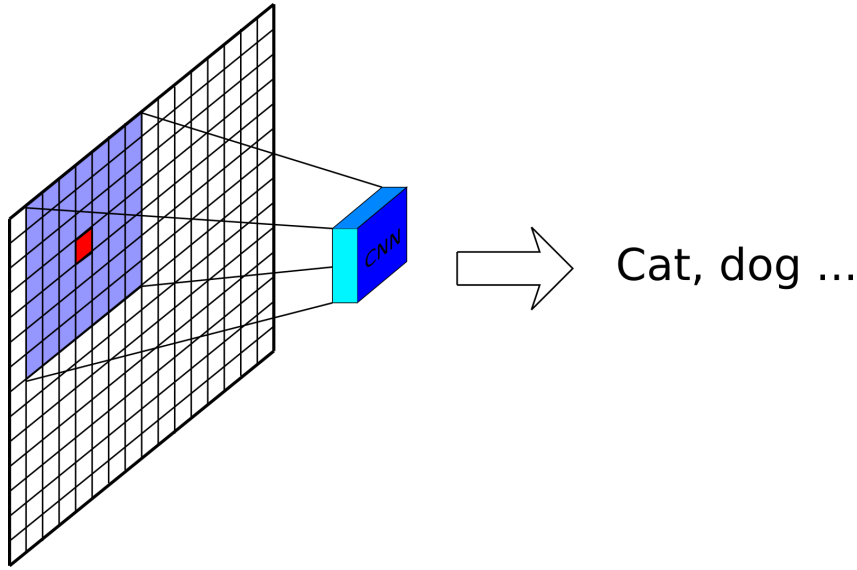
From classification nets to image-to-image nets



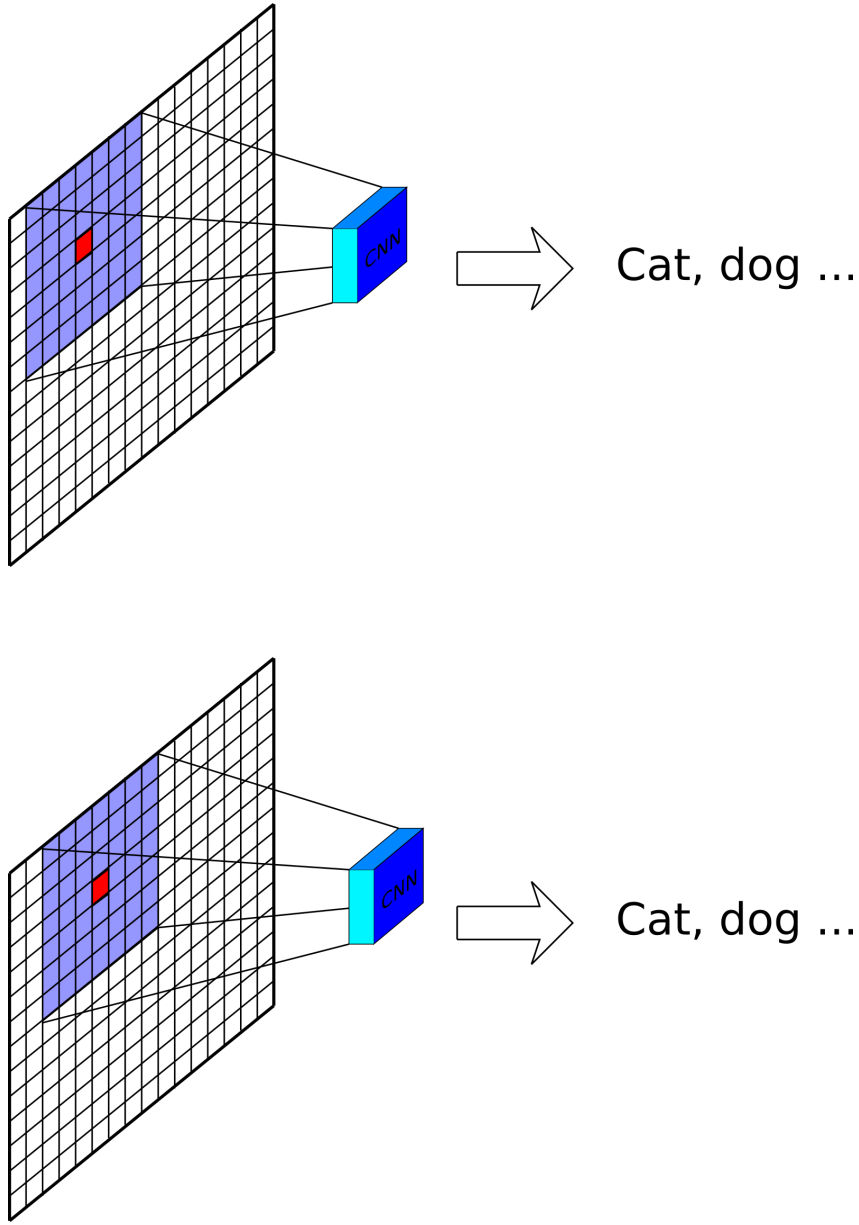
From classification nets to image-to-image nets



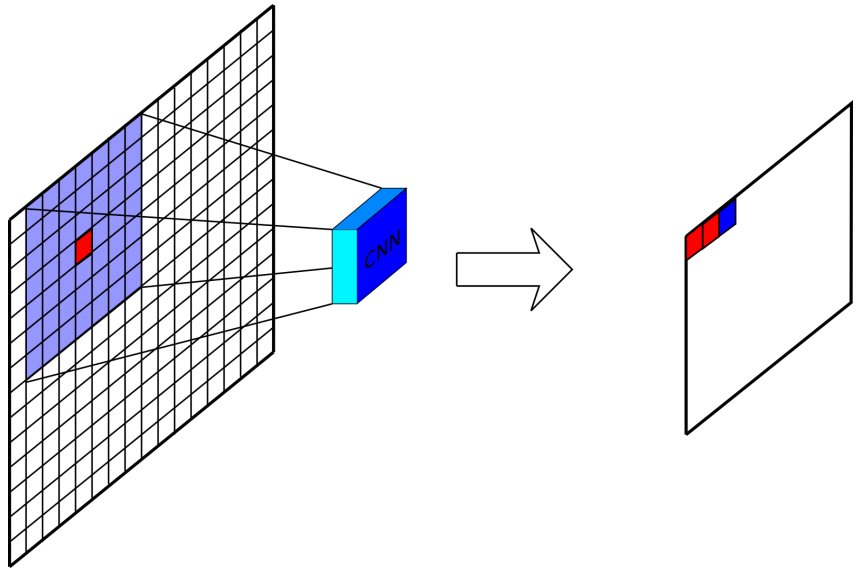
From classification nets to image-to-image nets



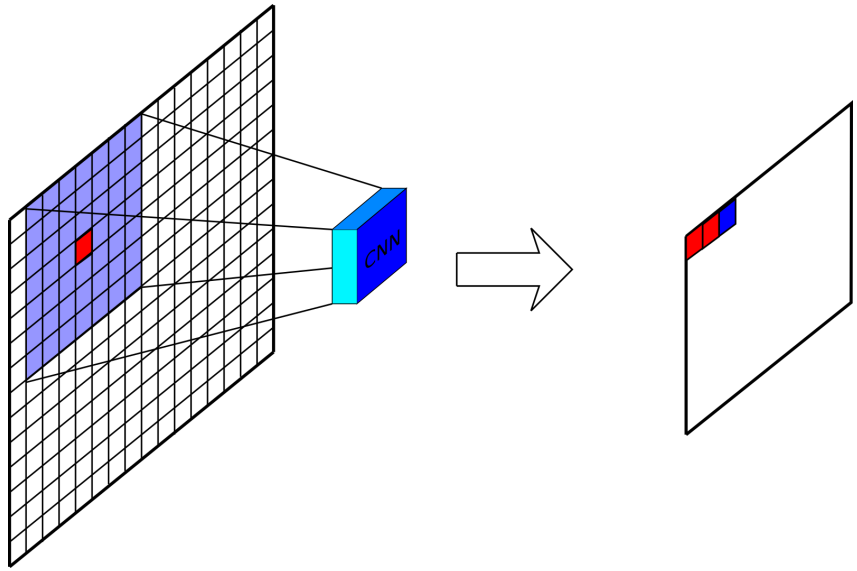
From classification nets to image-to-image nets



From classification nets to image-to-image nets



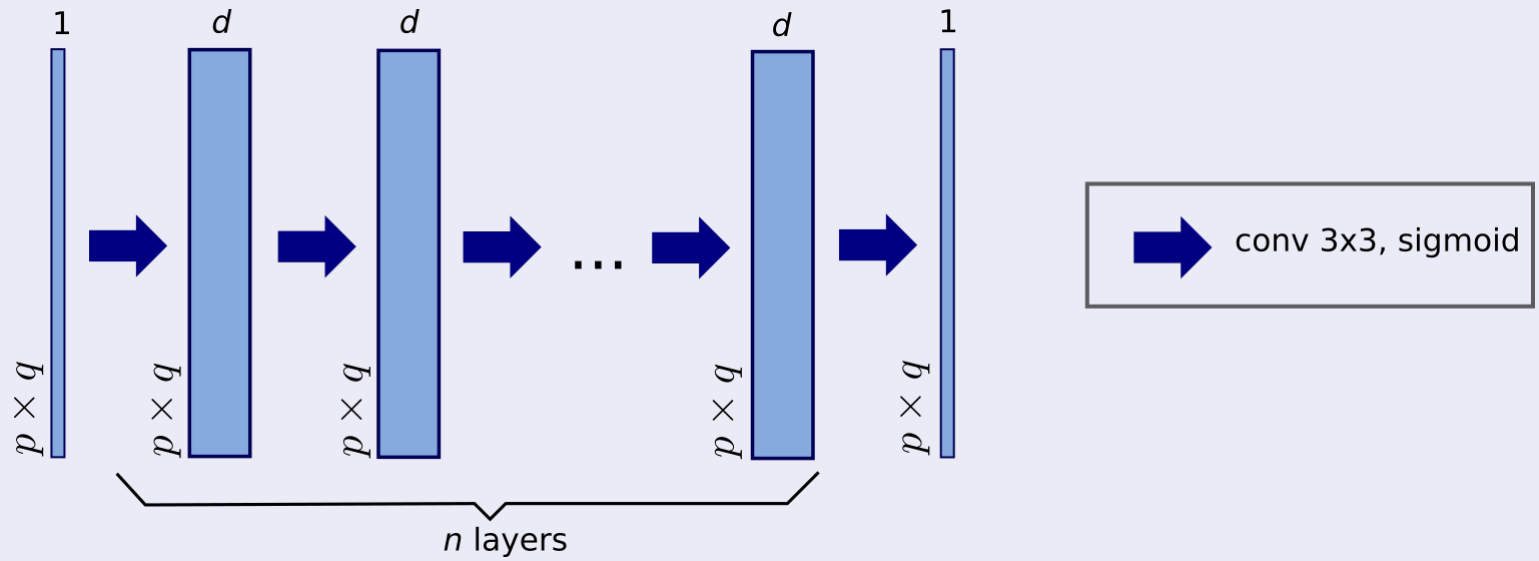
From classification nets to image-to-image nets



- The neuron membrane segmentation challenge winner [Cireşan et al., 2012] used this strategy. It is inefficient.
- The idea to compute the whole output image with a single pass through the network had already been proposed [Feng Ning et al., 2005, Jain et al., 2007] (but with architectures more complex than those used today).

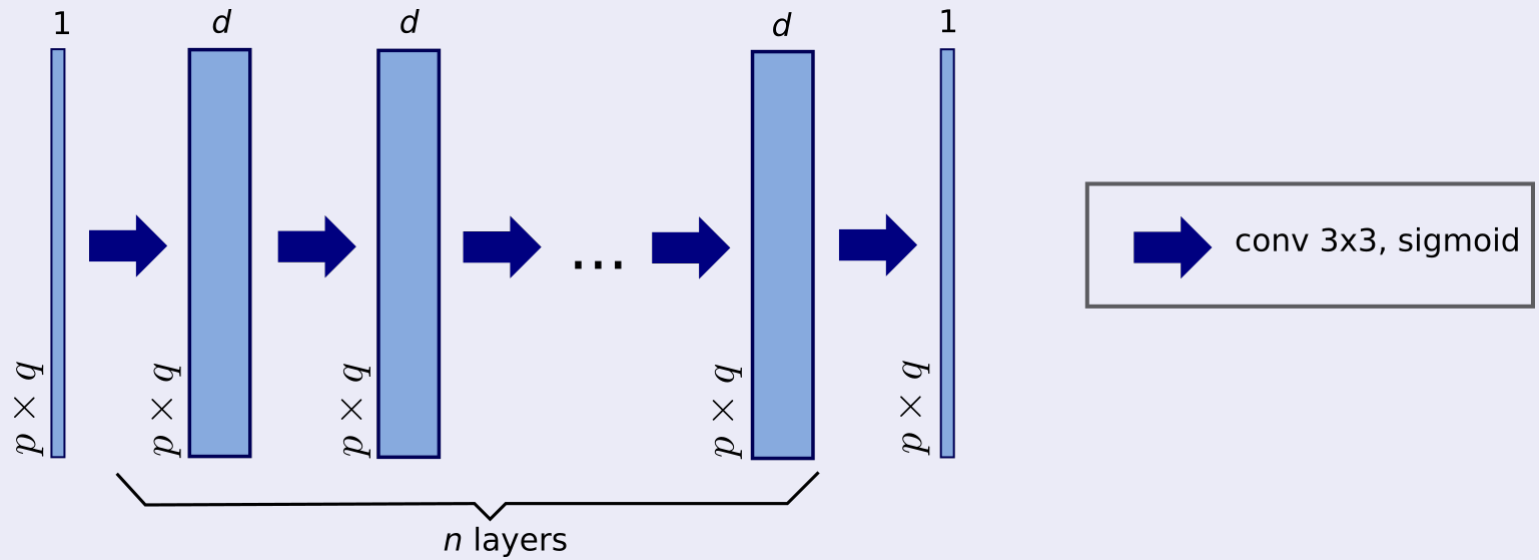
The simplest image-to-image architecture

Example: plain CNN [Pang et al., 2010]



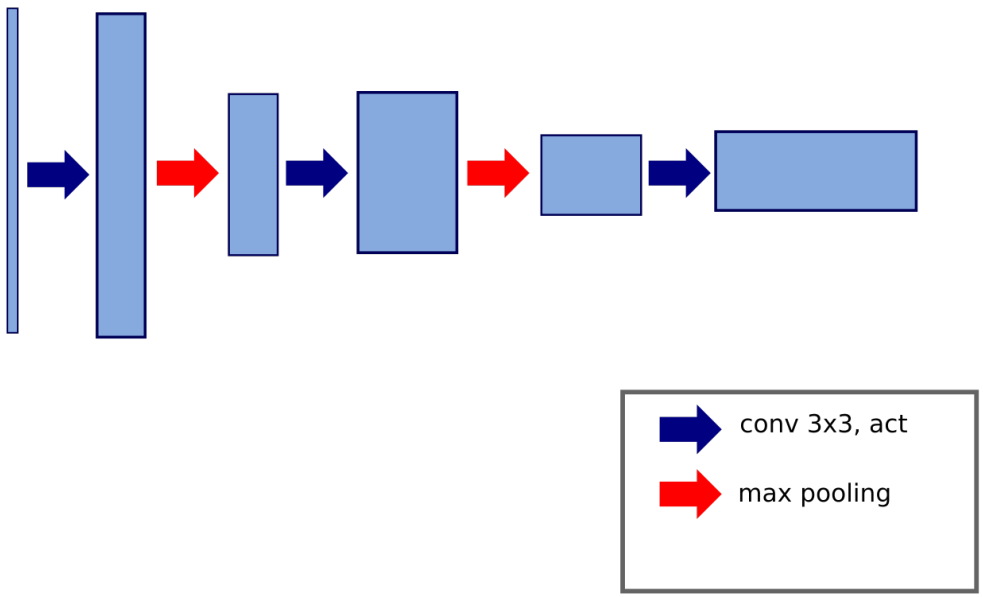
The simplest image-to-image architecture

Example: plain CNN [Pang et al., 2010]

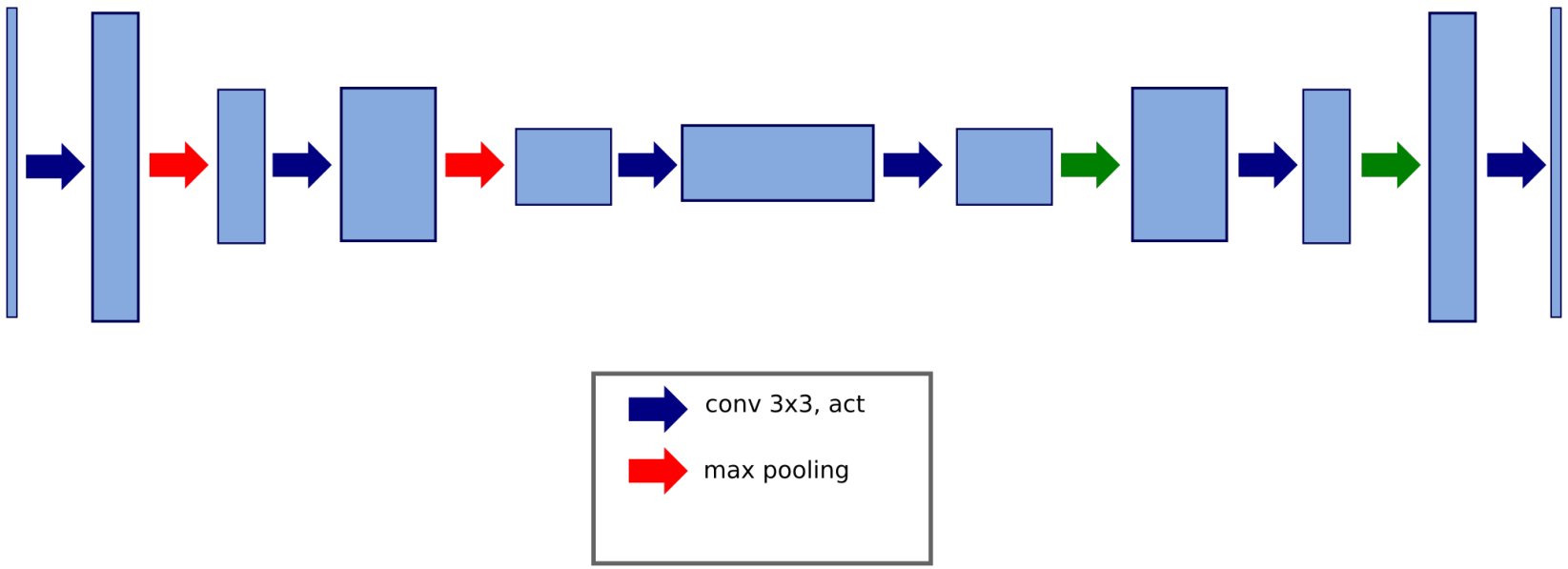


Note that today we would use ReLU activations on all layers except the last.

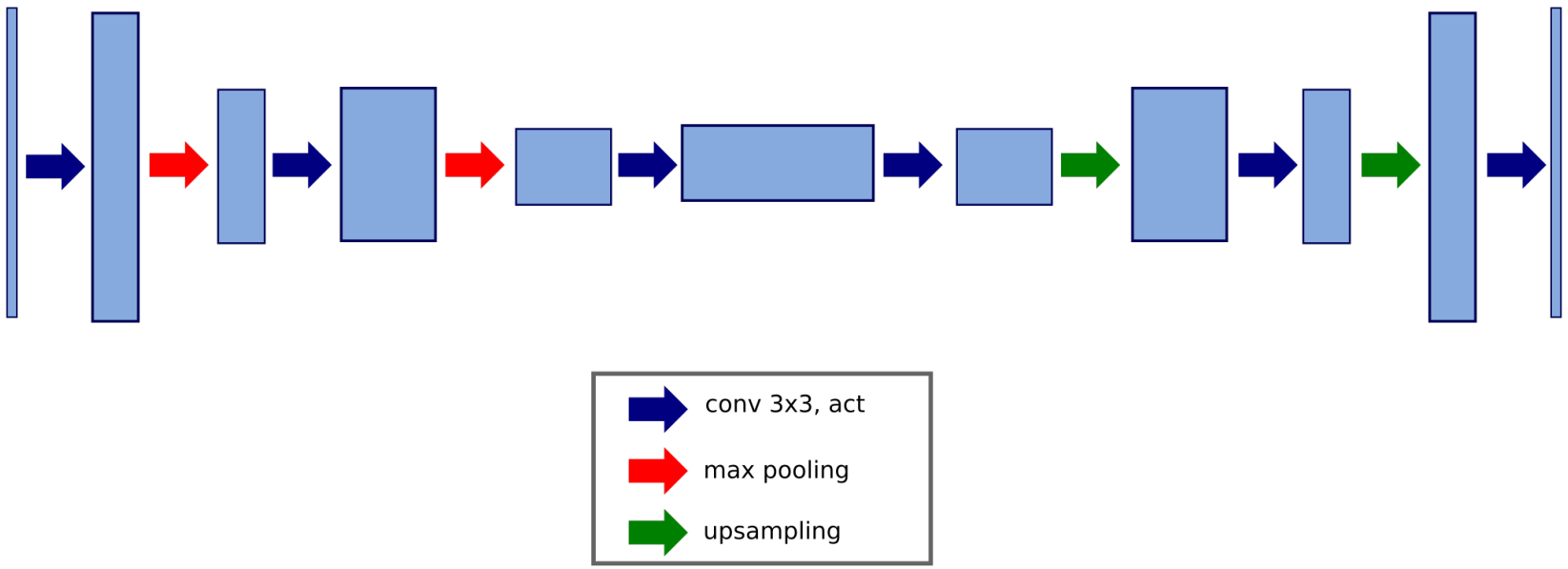
Going back to the original image size



Going back to the original image size



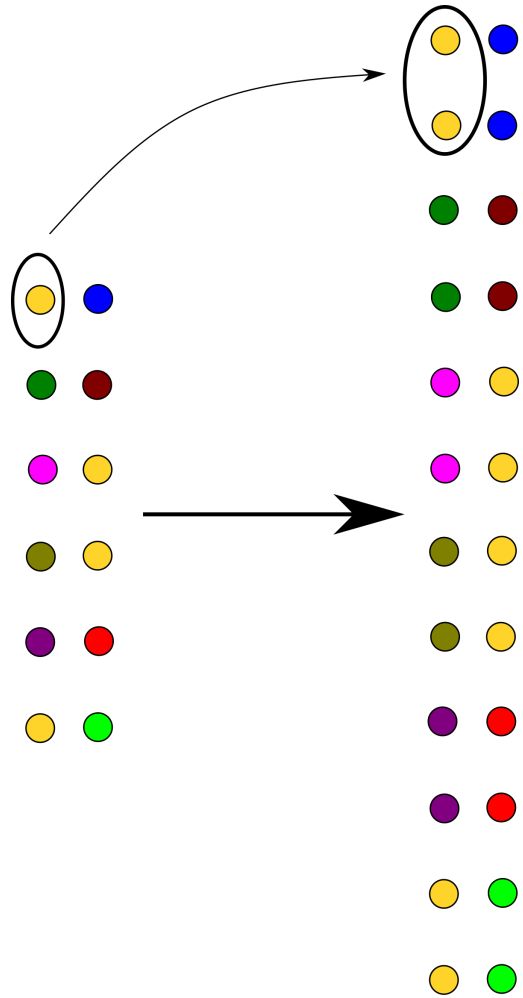
Going back to the original image size



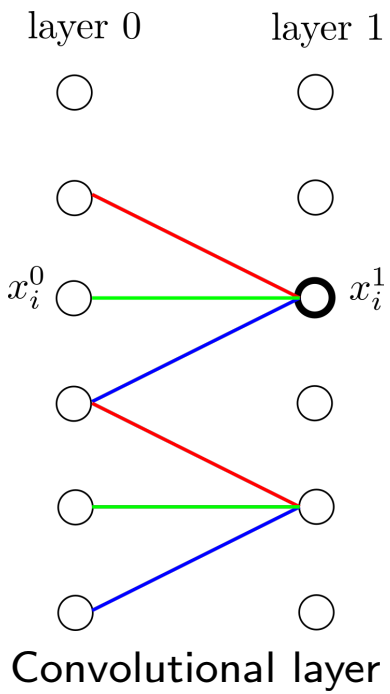
Upsampling techniques

- Replication
- Transposed convolution
- Pooling index memorization

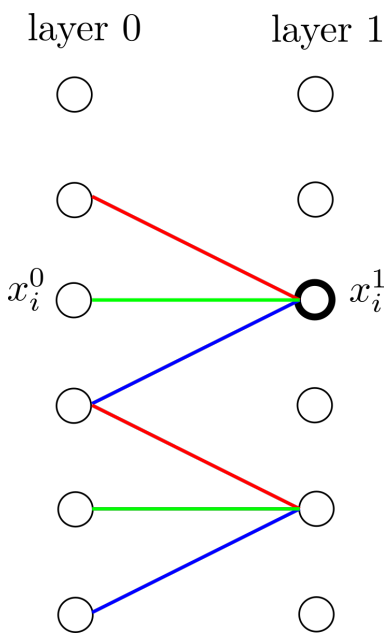
Upsampling through replication



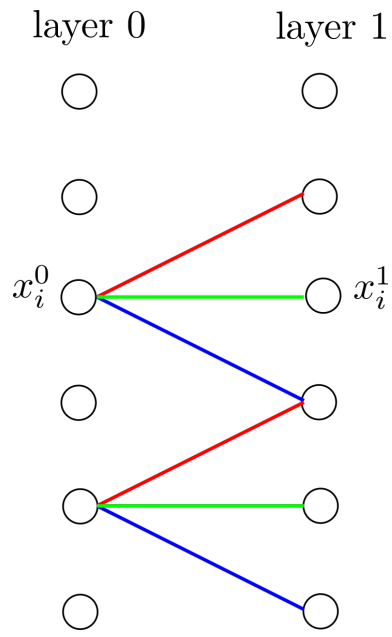
Transposed convolution



Transposed convolution

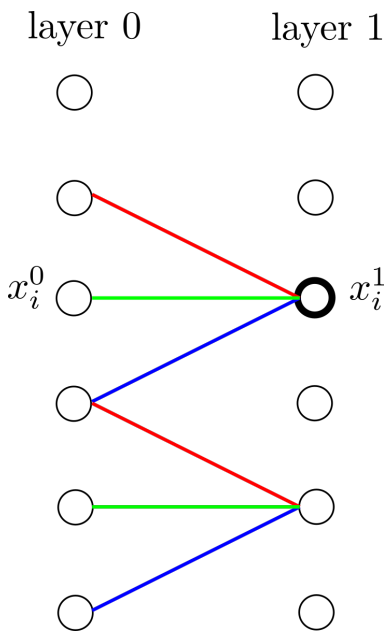


Convolutional layer

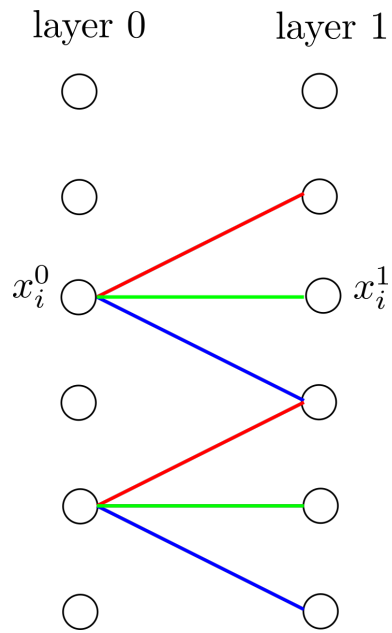


Transposed convolutional layer

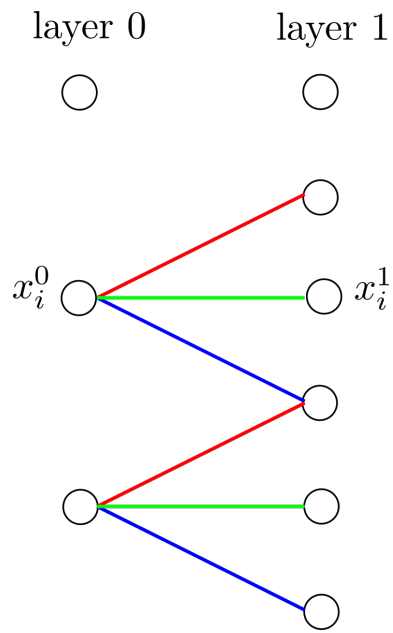
Transposed convolution



Convolutional layer



Transposed convolutional layer



Transposed convolutional layer with stride $= 2$

Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
   - Receptive field
   - Translation equivariance
   - Other properties
5. Image segmentation
6. 3D
7. Conclusion

Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
   - Receptive field
   - Translation equivariance
   - Other properties
5. Image segmentation
6. 3D
7. Conclusion

Receptive field

Definition: links between neurons

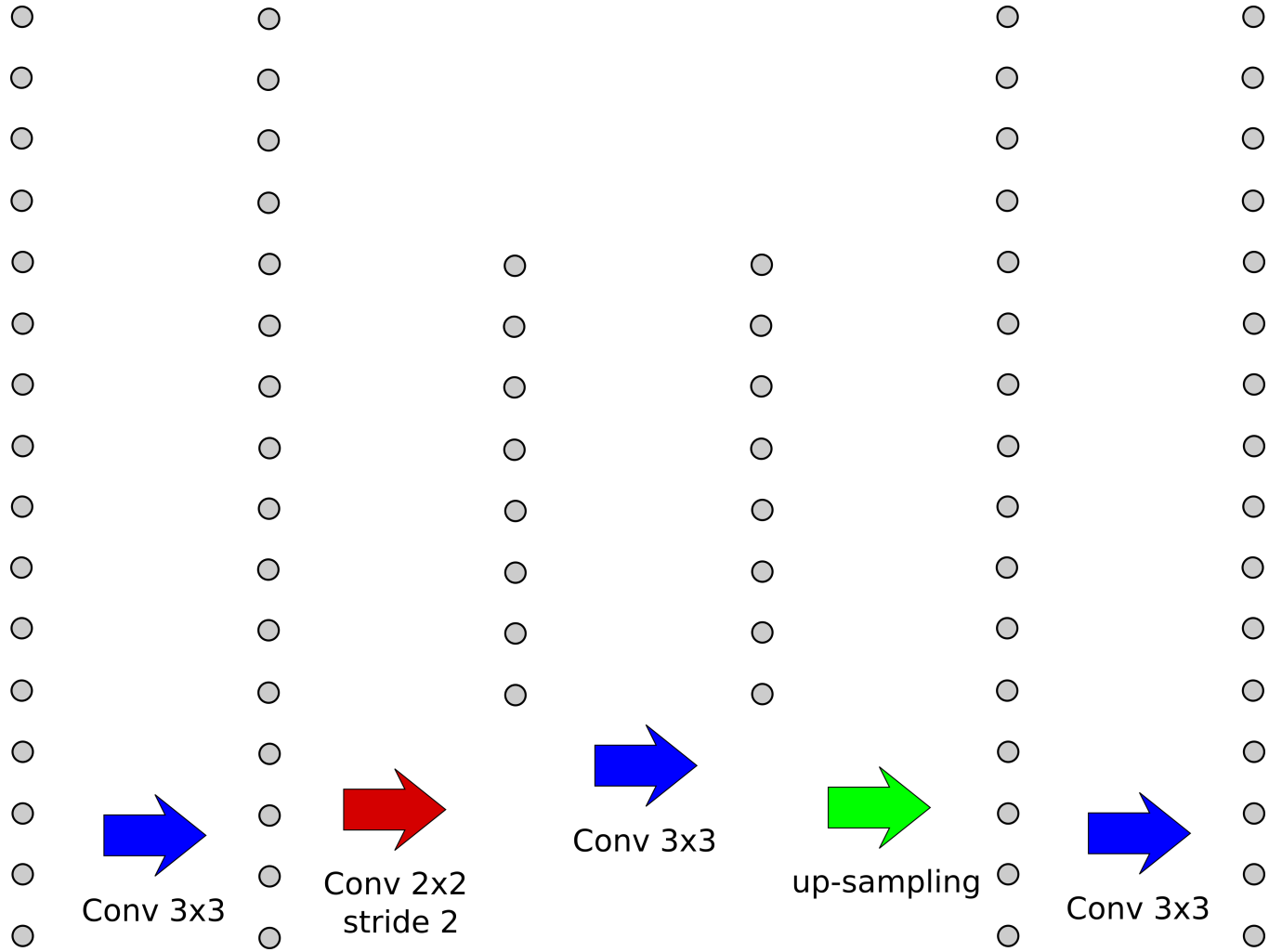
In a NN, we say that neuron a is linked to neuron b if there is an oriented path in the corresponding graph going from a to b.

Definition

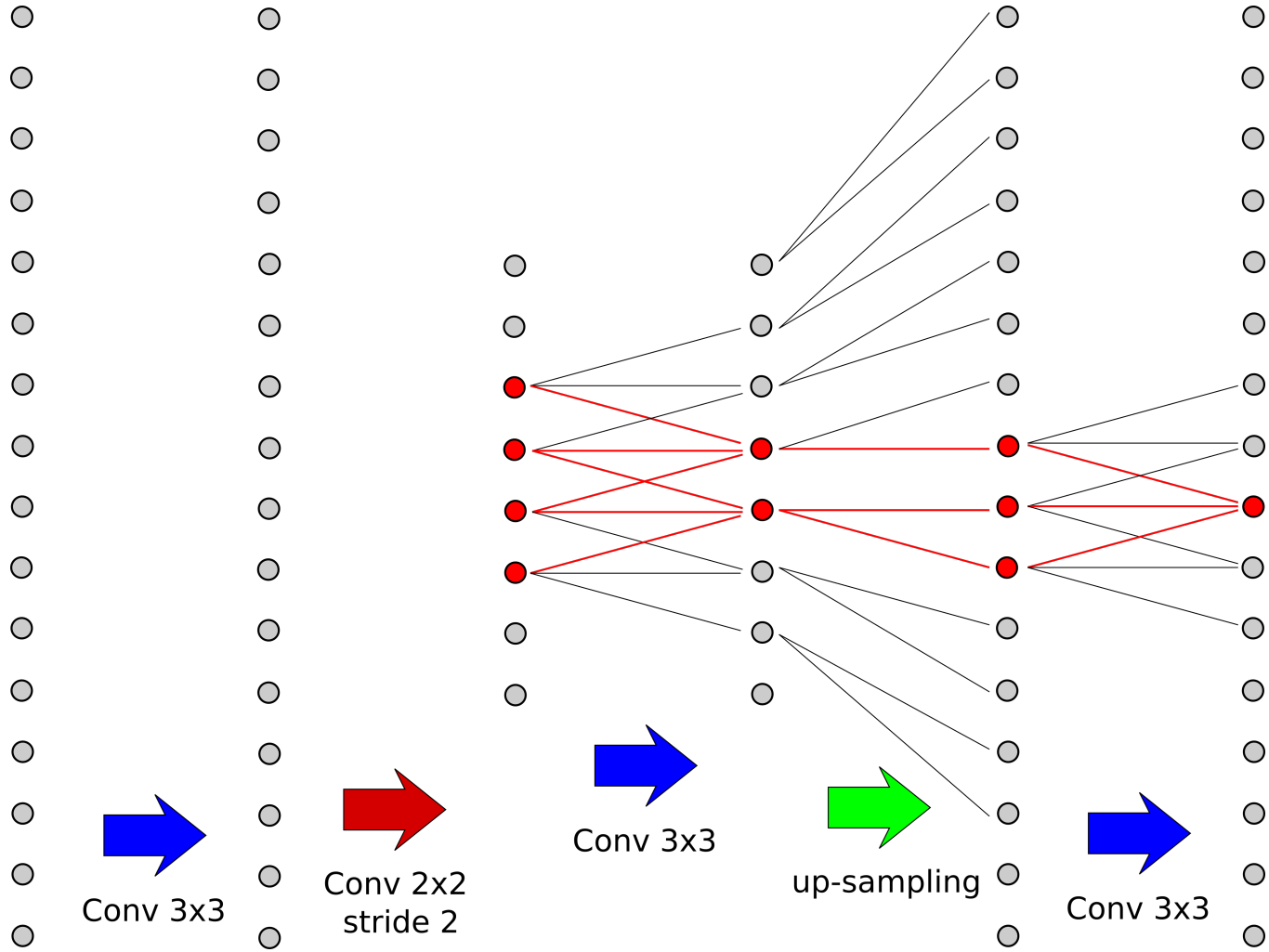
The receptive field of a neuron in a NN is the set of *input neurons* that are linked to that neuron.
The size of the receptive field is an essential property when designing a fully-convolutional NN architecture.

Note that in most cases, receptive fields are square shaped (border effects ignored).

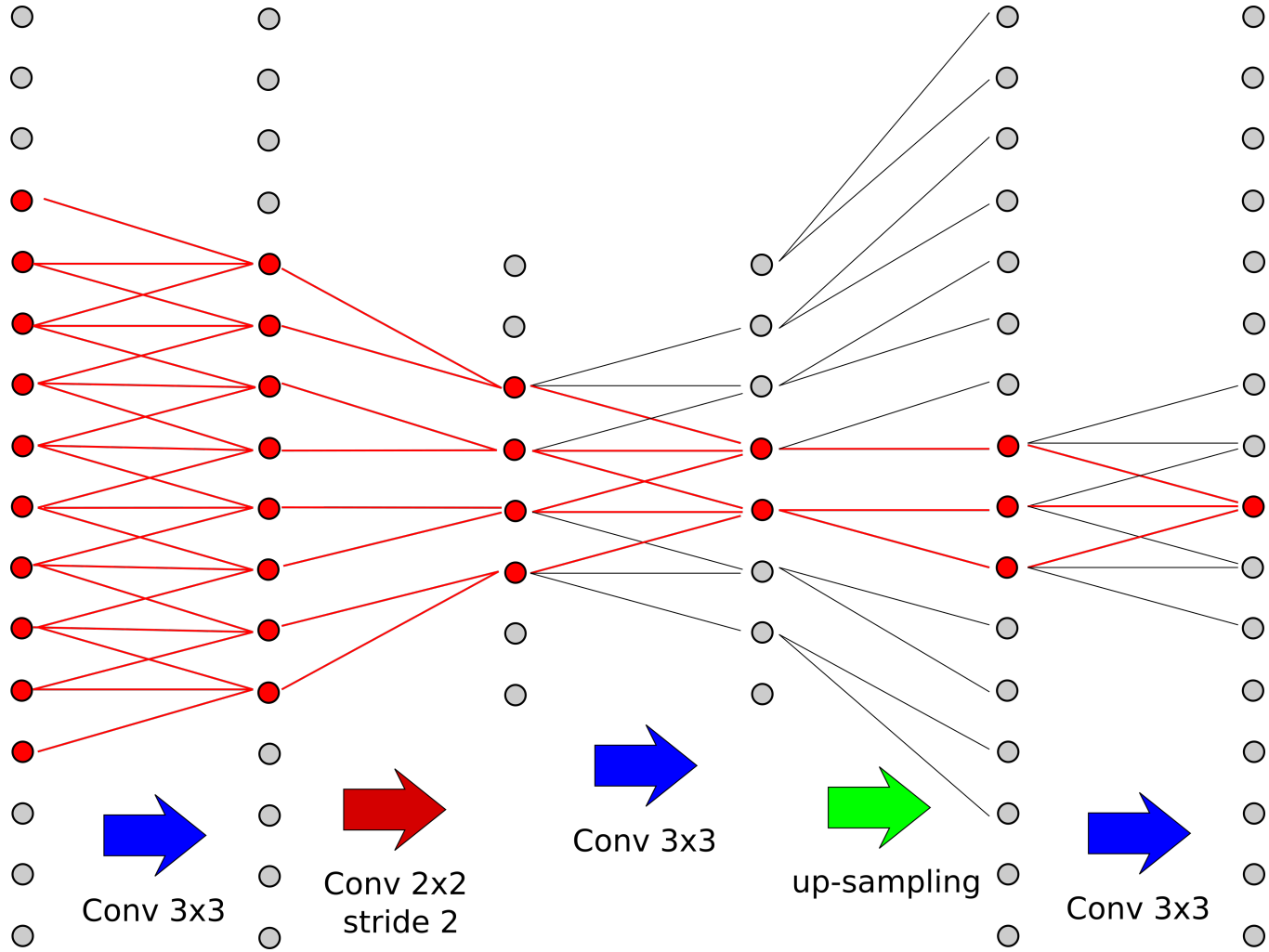
Illustration



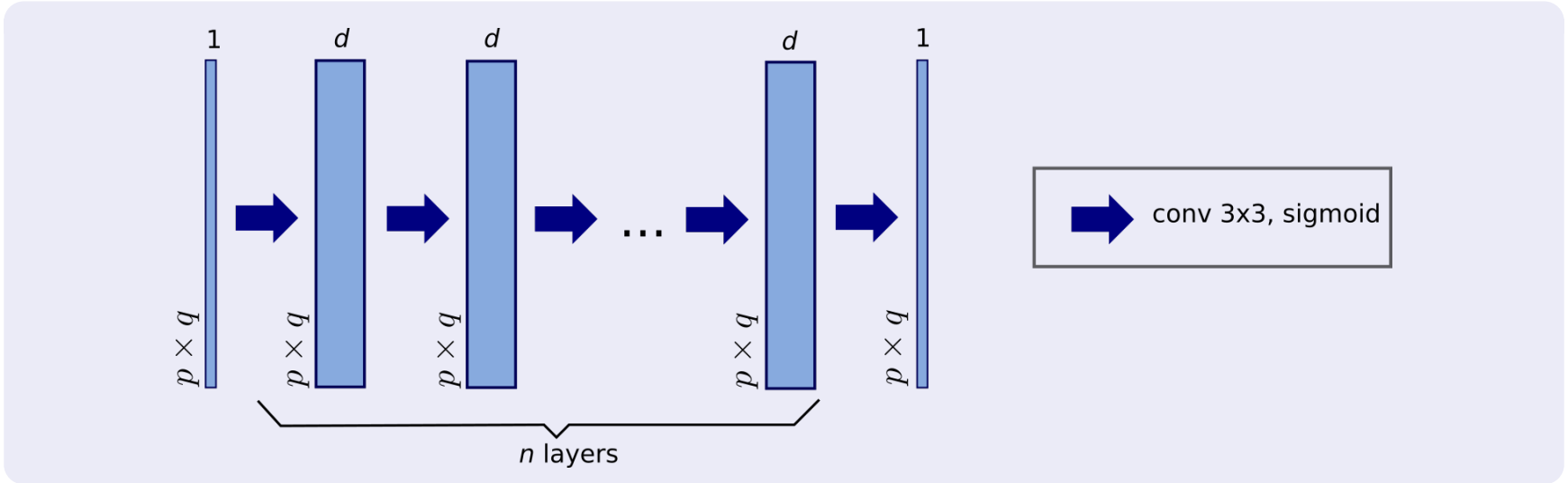
Illustration



Illustration



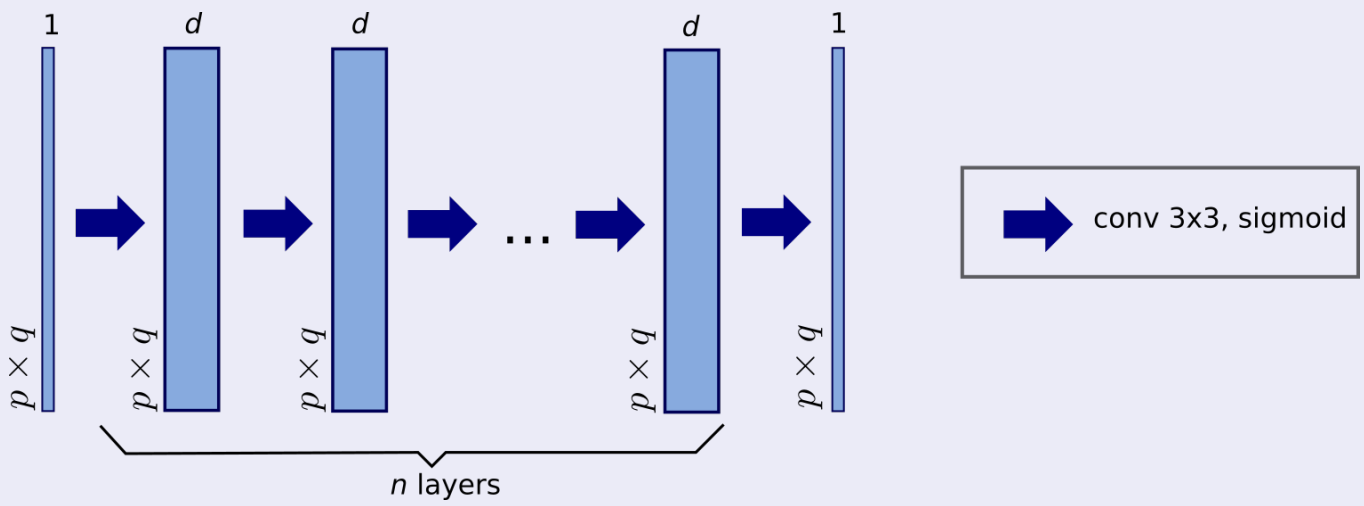
Example



What is the width of the receptive field of the neurons in the last layer?

1. n
2. $1 + 2 \times n$
3. $1 + 2 \times (n + 1)$
4. It depends on the input image

Example



What is the width of the receptive field of the neurons in the last layer?

1. n
2. $1 + 2 \times n$
3. $1 + 2 \times (n + 1)$
4. It depends on the input image

Answer: $1 + 2 \times (n + 1)$

Conclusion on the receptive field

When choosing or designing a NN architecture:

- Establish its minimal required size with respect to the application
- Make it larger than that...

Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
   - Receptive field
   - Translation equivariance
   - Other properties
5. Image segmentation
6. 3D
7. Conclusion

Equivariance

Definition

A function $f : E \longrightarrow F$ is equivariant with respect to the functions $t_E : E \longrightarrow E$ and $t_F : F \longrightarrow F$ iff $\forall x \in E$:

$$f(t_E(x)) = t_F(f(x))$$

Translation equivariance

- When $E = F$ and t_E and t_F are the same, any, translation, then we have *translation equivariance*.
- Translation equivariance is an often sought property for image processing operators.
- Note that we often abusively say *invariant* to translation instead of *equivariant* to translation.
- Given that in all practical cases images are defined on a bounded set, this property is only true "far enough" from the borders

Translation equivariance: comments

- If padding is used in the network, border effects can be important.

Translation equivariance: comments

- If padding is used in the network, border effects can be important.
- Translation equivariance is not always welcome!

Translation equivariance: comments

- If padding is used in the network, border effects can be important.
- Translation equivariance is not always welcome!
- Position information can also be used in the network:

Translation equivariance: comments

- If padding is used in the network, border effects can be important.
- Translation equivariance is not always welcome!
- Position information can also be used in the network:
  - Through masks or segmentations

Translation equivariance: comments

- If padding is used in the network, border effects can be important.
- Translation equivariance is not always welcome!
- Position information can also be used in the network:
  - Through masks or segmentations
  - Through pixel coordinates

Illustration

When aiming to segment the papilla or the macula, translation equivariance is not necessarily welcome.

Credits: OPHDIAT database

Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
   - Receptive field
   - Translation equivariance
   - Other properties
5. Image segmentation
6. 3D
7. Conclusion

Image size flexibility

- A NN containing fully-connected layers can only process images of a given size.

Image size flexibility

- A NN containing fully-connected layers can only process images of a given size.
- A fully convolutional NN can be applied to images of any size, as long as its dimensions are compatible with the subsampling steps of the network.

Image size flexibility

- A NN containing fully-connected layers can only process images of a given size.
- A fully convolutional NN can be applied to images of any size, as long as its dimensions are compatible with the subsampling steps of the network.
- Practical limit: the memory of the system.

Image size flexibility

- A NN containing fully-connected layers can only process images of a given size.
- A fully convolutional NN can be applied to images of any size, as long as its dimensions are compatible with the subsampling steps of the network.
- Practical limit: the memory of the system.
- Note that as the input image gets larger, border effects become proportionally less present.

Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
5. Image segmentation
   - Binary segmentation
   - Semantic segmentation
   - Instance segmentation
   - U-Net
6. 3D

The specific case of image segmentation

Definition: image segmentation

Let I be an image defined on D. A segmentation of I is a partition of D. In practice the regions of the segmentation should correspond to the objects in I, which is application dependant.

- A partition is often represented as a labelled image
- In order to make the segments symmetric, each one is represented by a different channel

Image segmentation example

Credits: Pascal VOC database

Some vocabulary on segmentation

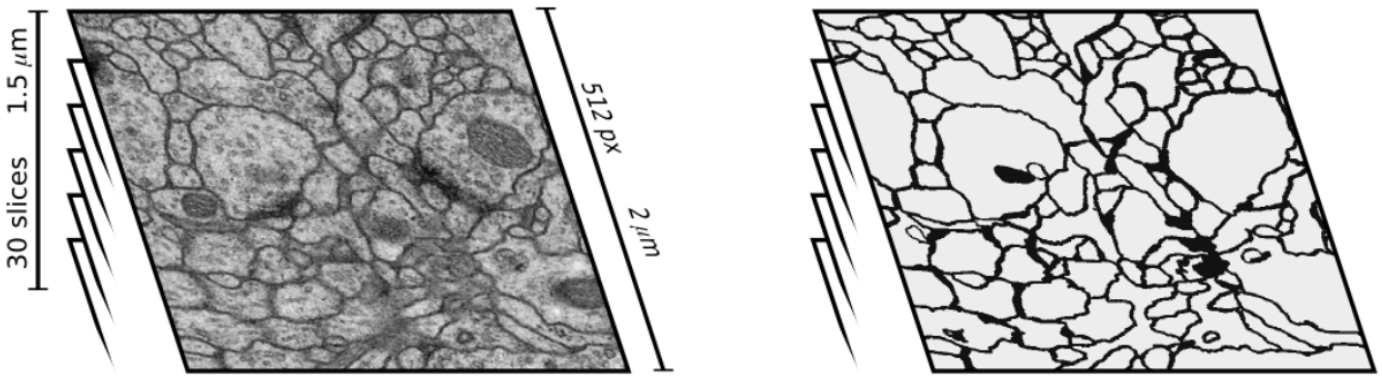
- **Object detection / localization**: bounding box around the object(s).
- **Binary segmentation**: segmentation in 2 classes, background and object.
- **Semantic segmentation**: a label is given to each pixel, according to the object it belongs to.
- **Instance segmentation**: identify each separate object, even if they belong to the same class.

Contents

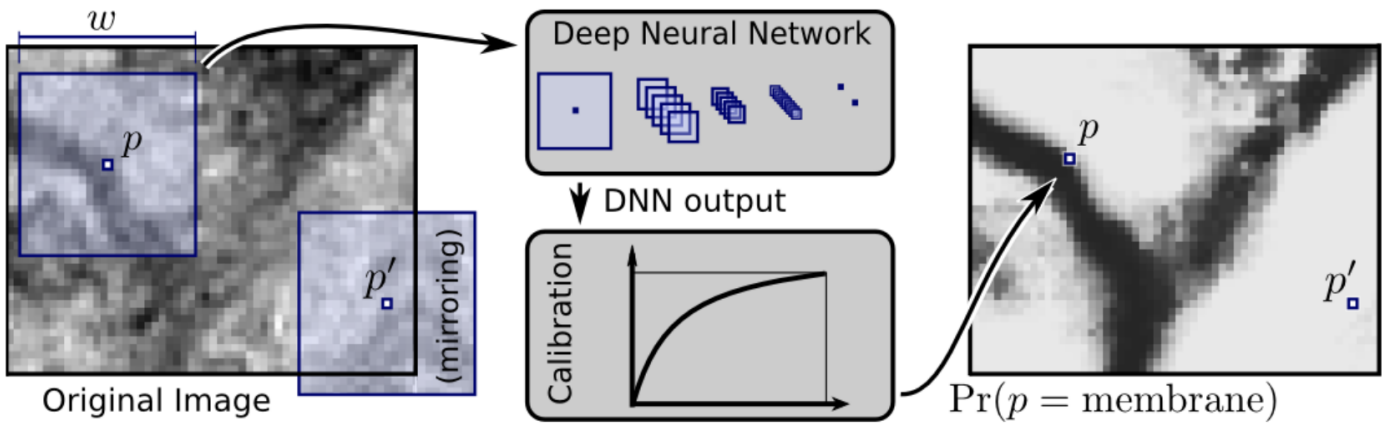
1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
5. Image segmentation
   - Binary segmentation
   - Semantic segmentation
   - Instance segmentation
   - U-Net
6. 3D

Neuron membrane segmentation challenge (ISBI 2012)

- Train: single stack of size $30 \times 512 \times 512$.
- Test: a second stack of same size.



Neuron membrane segmentation challenge winner [Cireşan et al., 2012]

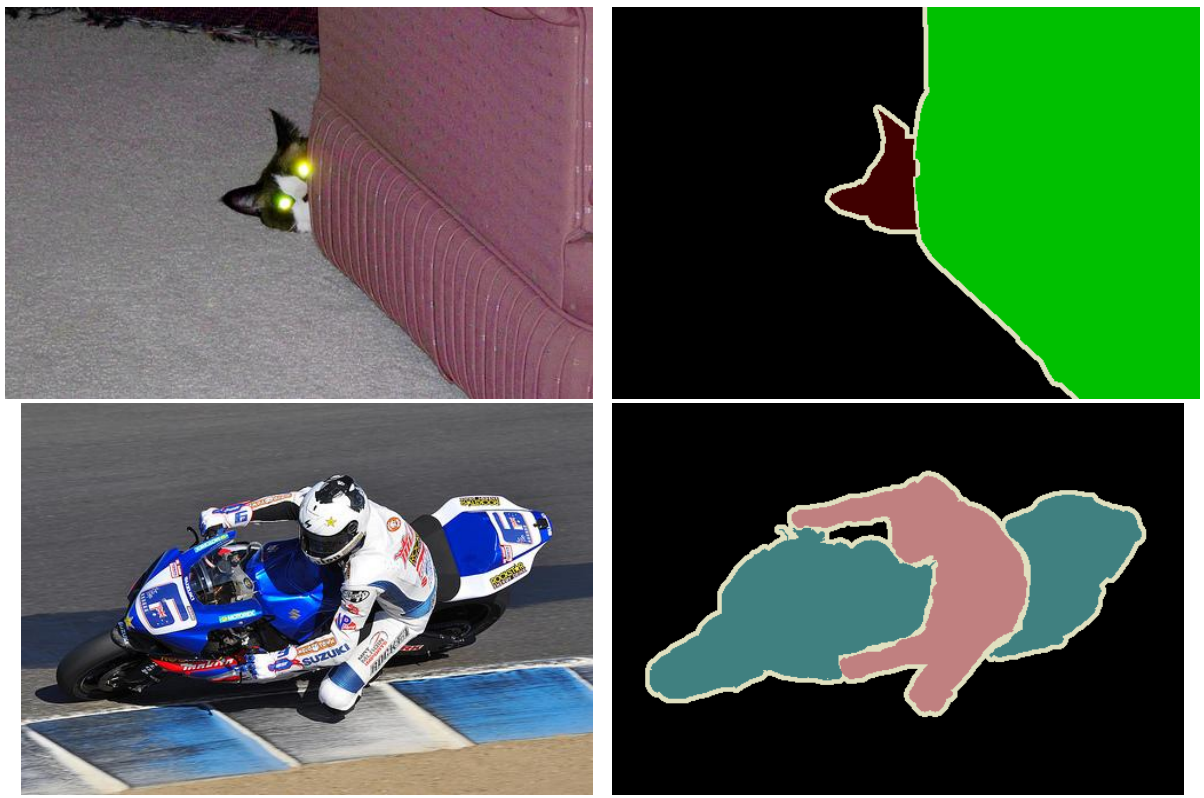


Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
5. Image segmentation
   - Binary segmentation
   - Semantic segmentation
   - Instance segmentation
   - U-Net
6. 3D

Pascal visual object classes segmentation challenge 2012 [Everingham et al., 2014]

- 1464 training and 1449 validation images
- automatic online test, with unknown images
- 20 image categories (cat, sofa, motorbike, person, etc.)



Convolutional nets for semantic image segmentation

Three papers in 2015:

- Fully convolutional networks for semantic segmentation [Long et al., 2015]
- U-Net: convolutional networks for biomedical image segmentation [Ronneberger et al., 2015]
- SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation [Badrinarayanan et al., 2015]

Remarks

- These architectures easily contain a number of parameters of the order of 10^7 (28 million for U-Net)
- Their optimization might be difficult
- But you can reduce the number of filters or the number of layers

Contents

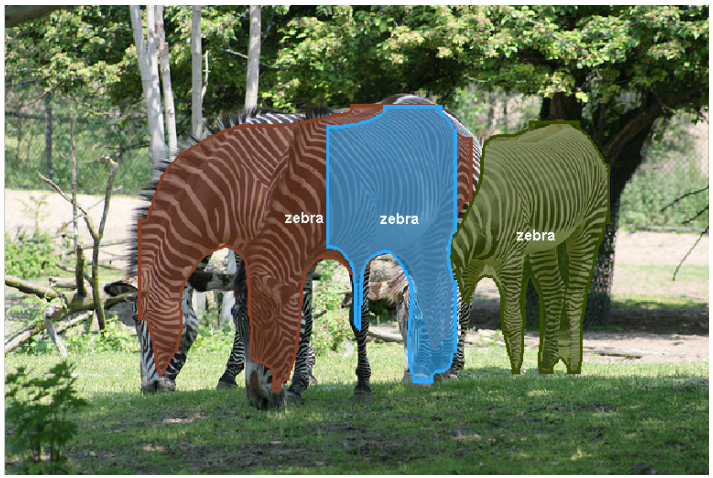
1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
5. Image segmentation
   - Binary segmentation
   - Semantic segmentation
   - Instance segmentation
   - U-Net
6. 3D

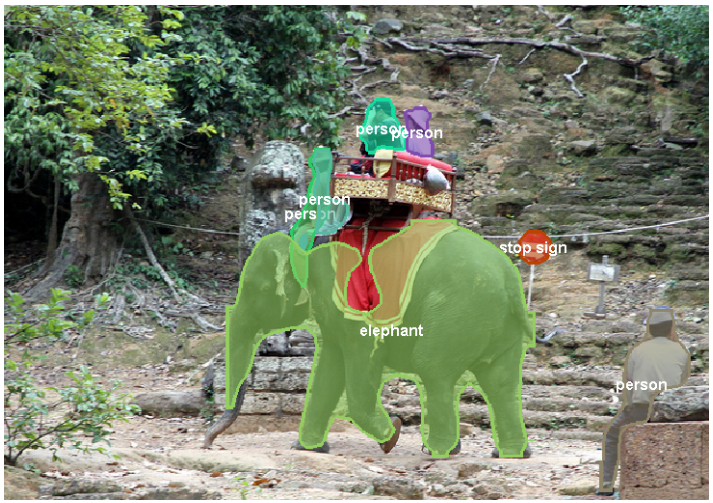
COCO: common objects in context [Lin et al., 2014]

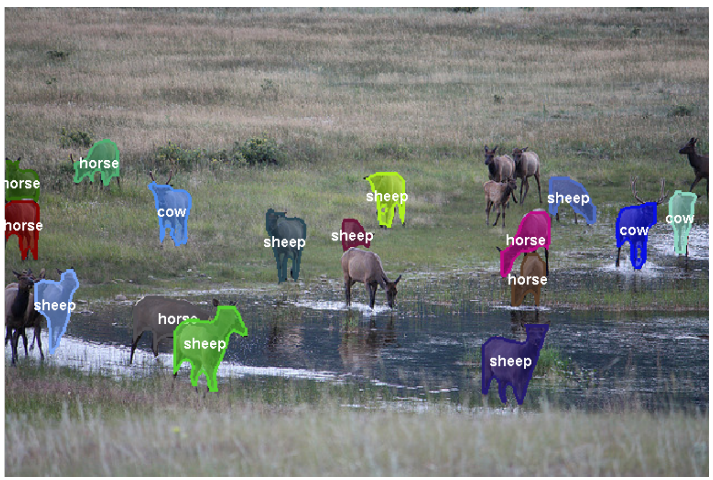
- 2 million objects, from 80 categories, in 300 000 images

Winner 2016: Fully Convolutional Instance-aware Semantic Segmentation (Microsoft) [Li et al., 2016]

COCO instance segmentation challenge: examples of 2016 winner results



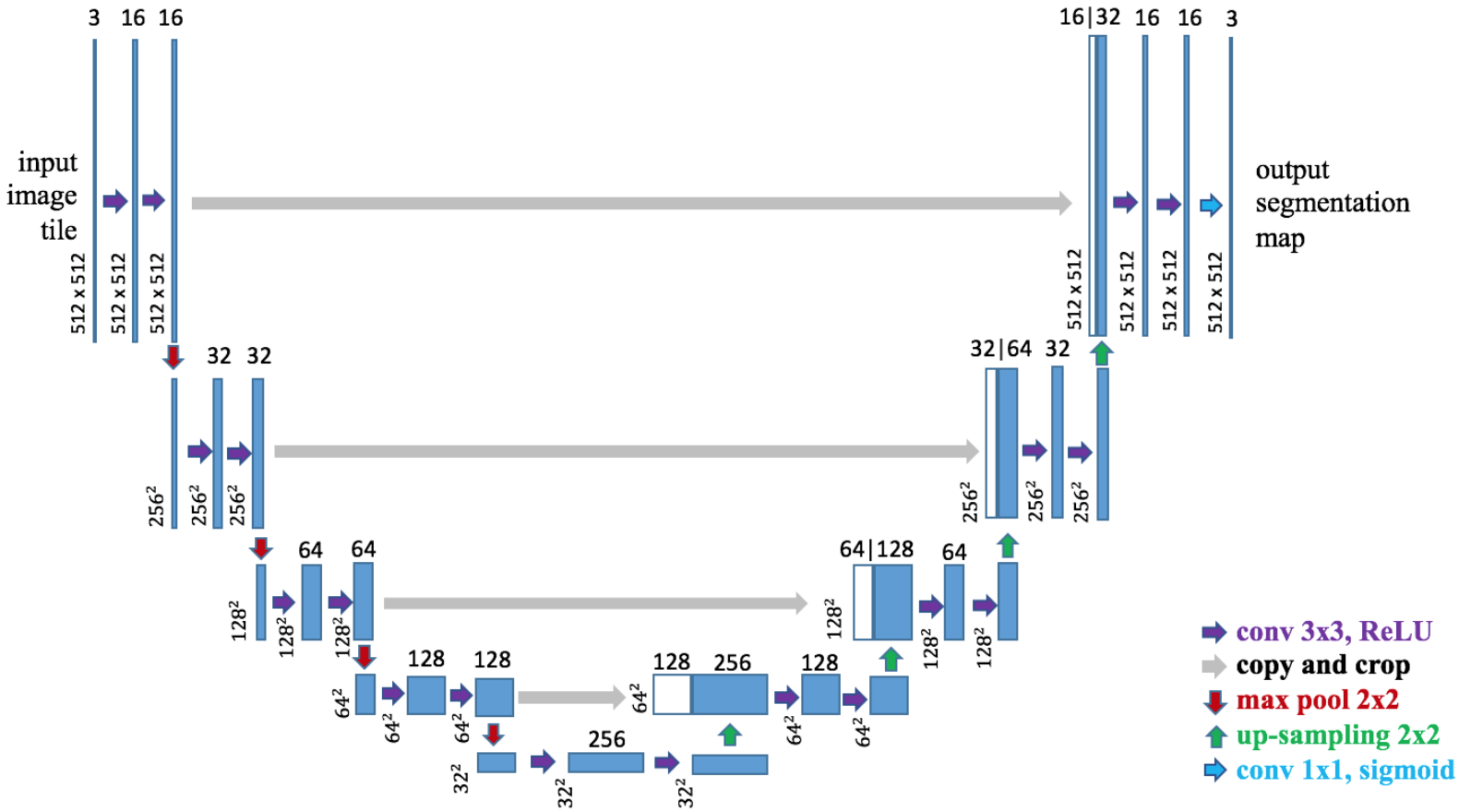




Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
5. Image segmentation
   - Binary segmentation
   - Semantic segmentation
   - Instance segmentation
   - U-Net
6. 3D

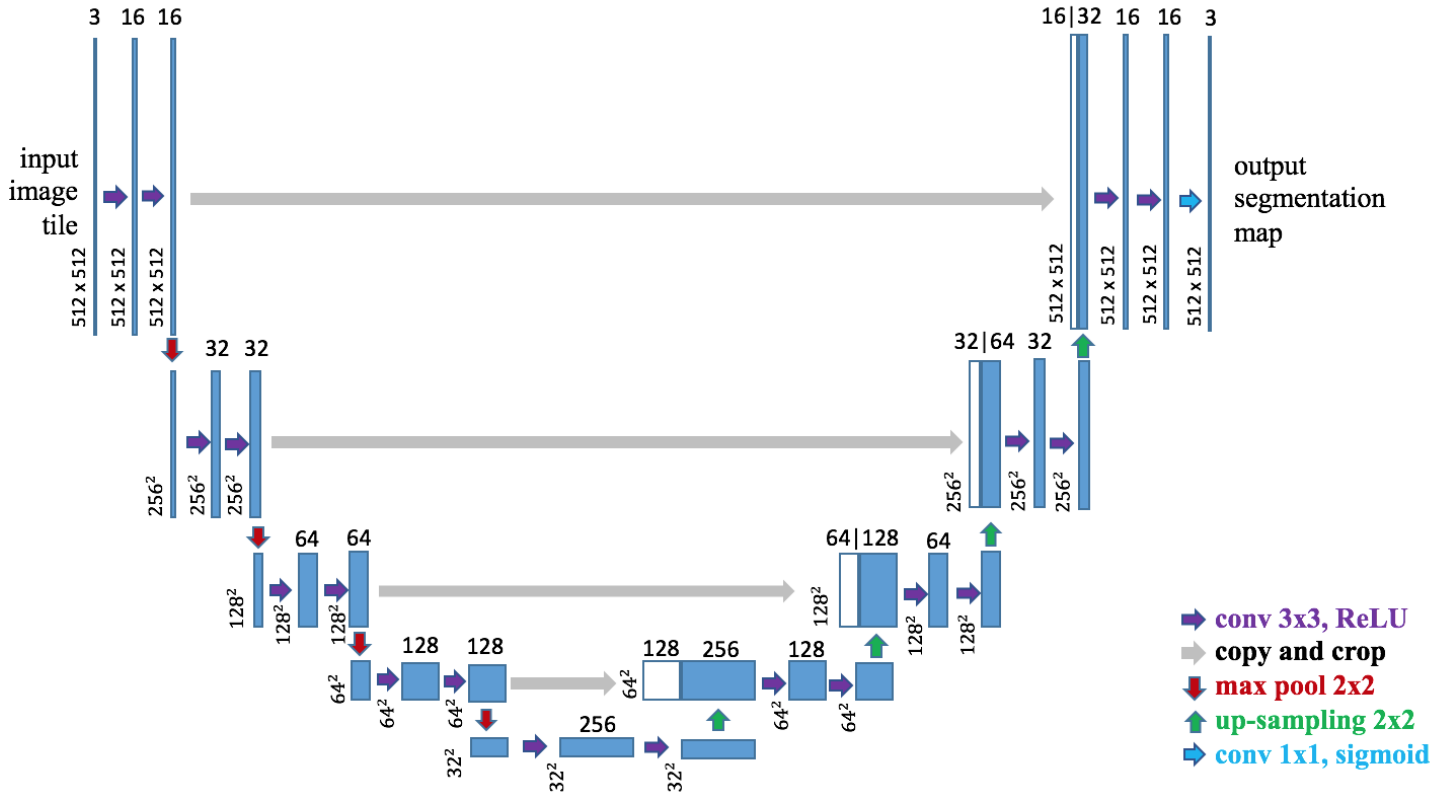
U-Net architecture [Ronneberger et al., 2015]



Quizz

- Size and number of channels of input images?
- Segmentation into how many regions?

U-Net main ideas



- Encoding branch inspired by classification nets
- Decoding branch is symmetrical
- Skip connections

U-Net improvements

- Convolutions with stride 2 instead of max-pooling in the encoder
- Transposed convolutions instead of simple up-sampling in the decoder
- Batch normalization
- Using an already optimized classification network as backbone for the encoder

Dealing with image sizes during training

- In segmentation applications, original images are often of different sizes and possibly very large.

Dealing with image sizes during training

- In segmentation applications, original images are often of different sizes and possibly very large.
- In theory, given the translation equivariance of fully-convolutional NN, we could use them directly as input. In practice, we are limited by memory size.

Dealing with image sizes during training

- In segmentation applications, original images are often of different sizes and possibly very large.
- In theory, given the translation equivariance of fully-convolutional NN, we could use them directly as input. In practice, we are limited by memory size.
- Solution: extract fixed-sized crops from your training set:

Dealing with image sizes during training

- In segmentation applications, original images are often of different sizes and possibly very large.
- In theory, given the translation equivariance of fully-convolutional NN, we could use them directly as input. In practice, we are limited by memory size.
- Solution: extract fixed-sized crops from your training set:
  - make them as large as possible, to reduce border effects

Loss functions for image segmentation

- $\hat{\mathbf{y}} = (\hat{\mathbf{y}}_i)$: network output
- $\mathbf{y} = (\mathbf{y}_i)$: binary expected output
- We suppose that all $\hat{\mathbf{y}}_i$ are in $[0, 1]$
- We want the $\hat{\mathbf{y}}$ to be *as close as possible* to $\mathbf{y}$

Loss functions for image segmentation

A loss function inherited from image classification

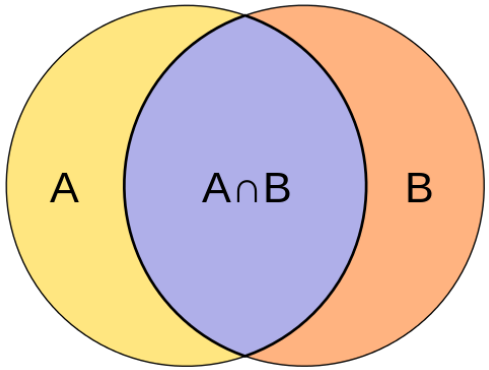
- Cross-entropy: $-\sum_i y_i \log(\hat{y}_i)$

Measures used in image processing

Let A and B be two sets, not simultaneously empty.

Dice coefficient

$$D(A,B) = \frac{2|A \cap B|}{|A| + |B|}$$



Jaccard index

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Properties

- $\forall A, B : 0 \leq J(A,B) \leq D(A,B) \leq 1$
- If $A = B$, then $D(A,B) = J(A,B) = 1$
- If $A \cap B = \emptyset$, then $D(A,B) = J(A,B) = 0$

Credits: Public domain (Wikipedia.org)

Generalization to $[0, 1]$

Jaccard index

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

But $\mathbf{y}$ and $\hat{\mathbf{y}}$ are in $[0, 1]^n \ldots$

Generalization to $[0, 1]$

Jaccard index

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

But $\mathbf{y}$ and $\hat{\mathbf{y}}$ are in $[0, 1]^n \ldots$

Jaccard similarity

$$J(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\sum_i y_i \hat{y}_i}{\sum_i y_i + \sum_i \hat{y}_i - \sum_i y_i \hat{y}_i}$$

($\mathbf{y}$ and $\hat{\mathbf{y}}$ are not simultaneously equal to 0)

Corresponding loss function

Jaccard loss

$$j(\mathbf{y}, \hat{\mathbf{y}}) = 1 - \frac{\sum_i y_i \hat{y}_i}{\sum_i y_i + \sum_i \hat{y}_i - \sum_i y_i \hat{y}_i + \epsilon}$$

Constant ϵ, which is typically "small", keeps the denominator "far enough" from zero.

Corresponding loss function - variant

Jaccard loss

$$j_2(\mathbf{y}, \hat{\mathbf{y}}) = 1 - \frac{\sum_i y_i \hat{y}_i}{\sum_i y_i^2 + \sum_i \hat{y}_i^2 - \sum_i y_i \hat{y}_i + \epsilon}$$

This version works slightly better than the first [Duque-Arias et al., 2021].

Conclusion on loss functions

- Use the Jaccard loss as base line for segmentation problems.
- Note that these losses compute their values pixel-wise: they do not take into account any structure (for example, continuity).
- Working on specific losses enforcing structure = interesting research path.

Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
5. Image segmentation
6. 3D
7. Conclusion

Processing 3D images

- 2D: Slice by slice
- 2,5D image: to treat slice s, take into account slices $s-1$ and $s+1$
- 3D

Advantage and drawbacks of 3D approaches

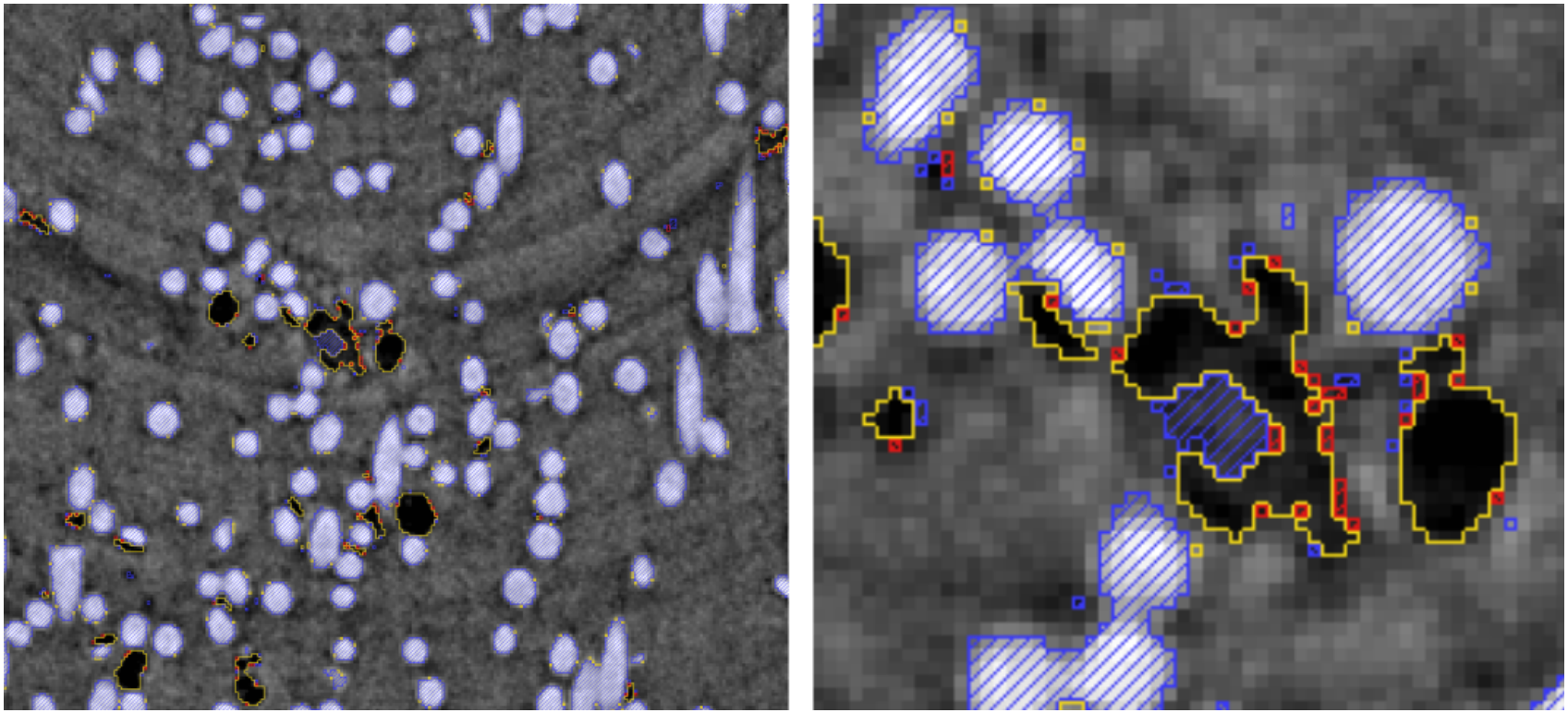
Advantage

- 3D structure

Drawbacks

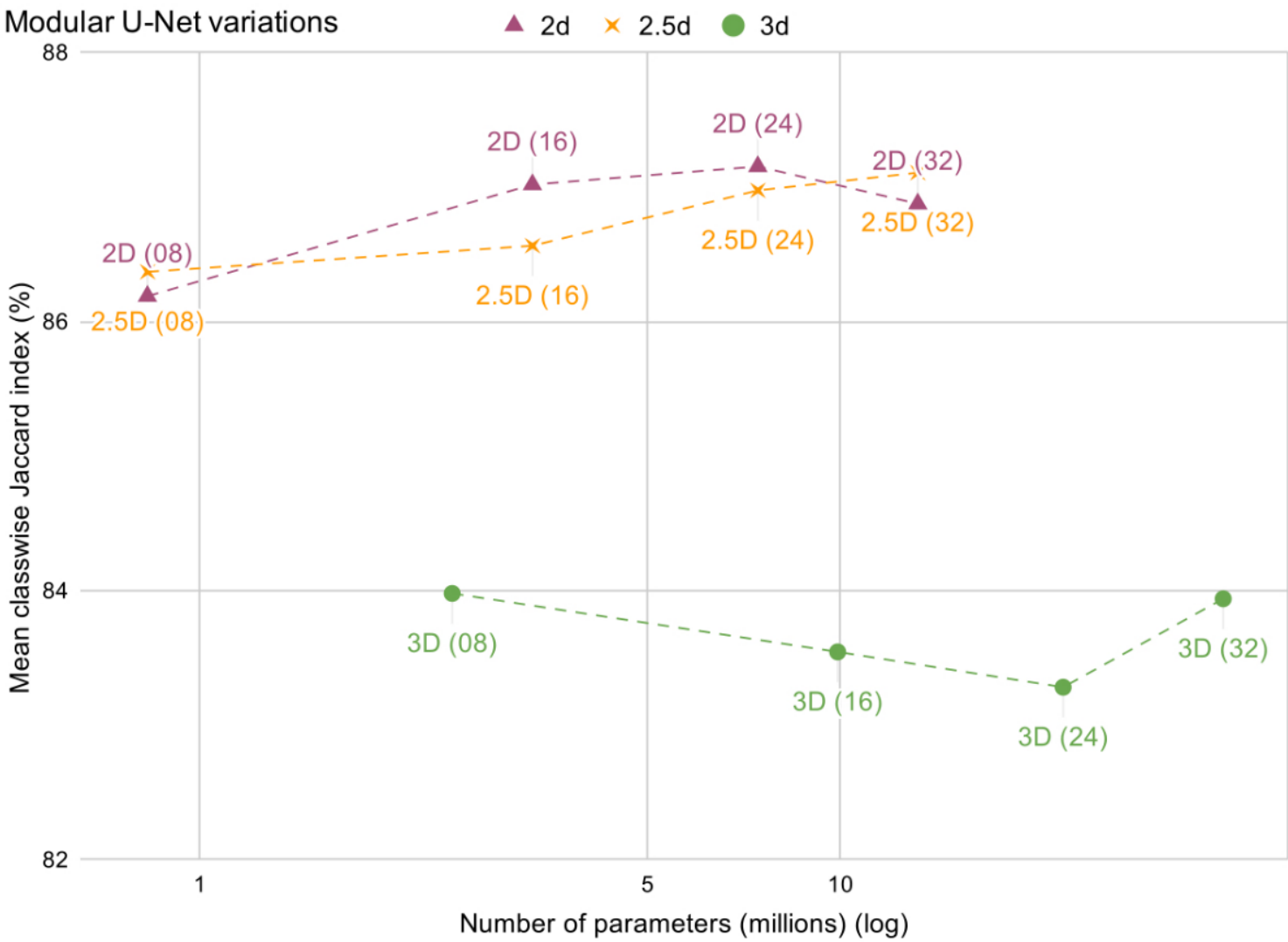
- Memory usage, smaller 2D receptive field
- Annotation

Example [Bertoldo et al., 2021]

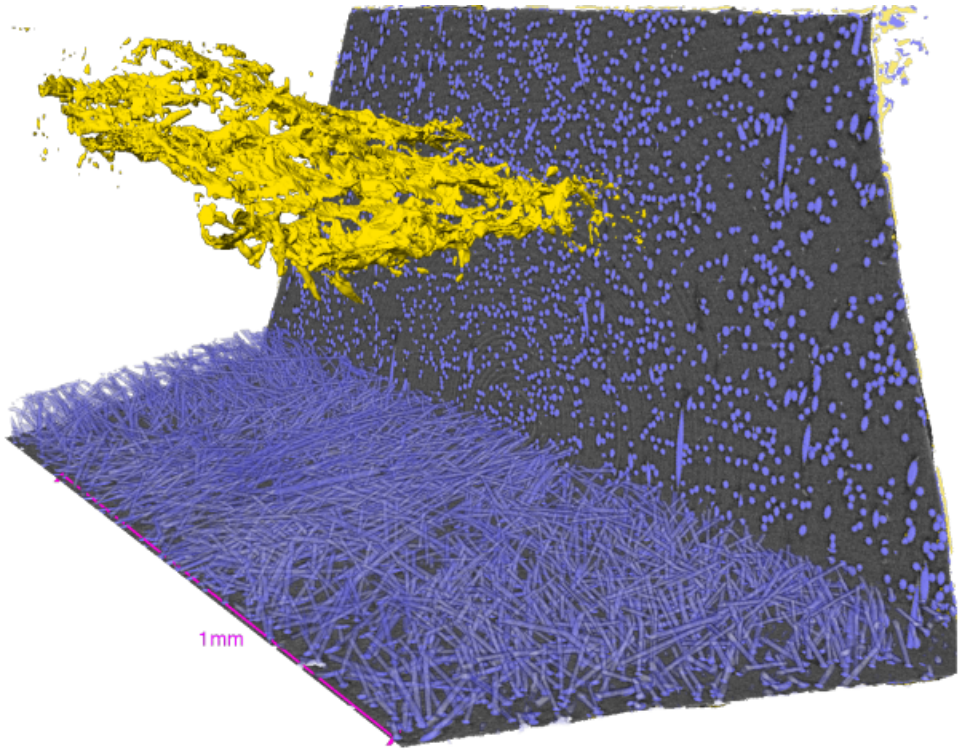


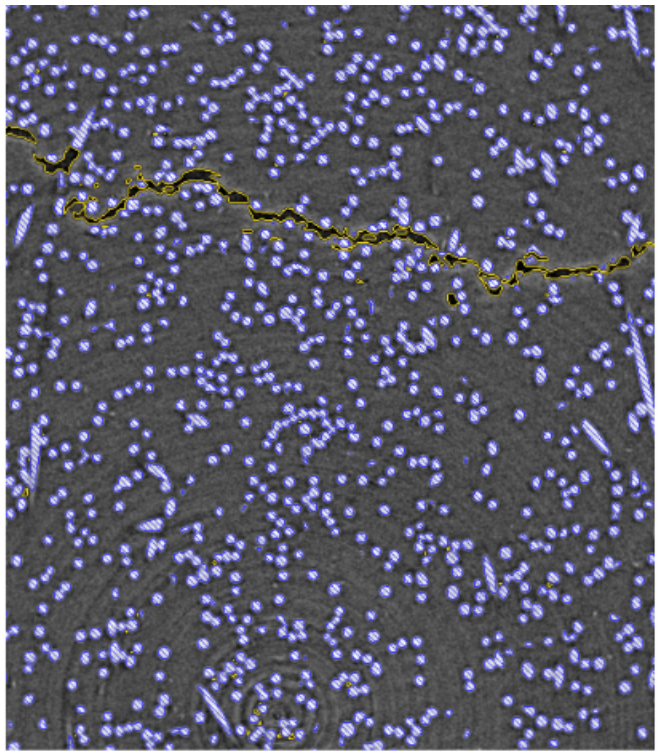
Annotated X-CT of polyamide 66 reinforced by glass fibers

Quantitative results



Test





Contents

1. Introduction
2. Convolutional neural networks
3. From classification to image-to-image translation
4. Properties of fully-convolutional neural networks
5. Image segmentation
6. 3D
7. Conclusion

Image segmentation: a solved problem?

- Progress in image segmentation since 2012 has been enormous
- Several complex problems have now satisfactory solutions
- Training can be a problem (large annotated databases, difficult optimization)
- There are still challenges ahead...

Some research subjects

- Optimization - a very general, and essential, subject
- Making training databases as small as possible
- Specific losses
- Taking *a priori* structural information into account
- Transformers for image segmentation

IntACT segmentation course material

https://colibris.link/RasJ1

References I

[Badrinarayanan et al., 2015] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *arXiv:1511.00561 [cs]*. arXiv: 1511.00561.

[Bertoldo et al., 2021] Bertoldo, J. P. C., Decencière, E., Ryckelynck, D., and Proudhon, H. (2021). A Modular U-Net for Automated Segmentation of X-Ray Tomography Images in Composite Materials. *Frontiers in Materials*, 8.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[Cireșan et al., 2012] Cireșan, D., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2012). Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 2843–2851. Curran Associates, Inc.

[Duque-Arias et al., 2021] Duque-Arias, D., Velasco-Forero, S., Deschaud, J.-E., Goulette, F., Serna, A., Decencière, E., and Marcotegui, B. (2021). On power Jaccard losses for semantic segmentation. In *VISAPP 2021 : 16th International Conference on Computer Vision Theory and Applications*, Vienne (on line), Austria.

References II

[Everingham et al., 2014] Everingham, M., Eslami, S. M. A., Gool, L. V., Williams, C. K. I., Winn, J., and Zisserman, A. (2014). The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1):98–136.

[Feng Ning et al., 2005] Feng Ning, Delhomme, D., LeCun, Y., Piano, F., Bottou, L., and Barbano, P. E. (2005). Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, 14(9):1360–1371. Conference Name: IEEE Transactions on Image Processing.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

[Jain et al., 2007] Jain, V., Murray, J. F., Roth, F., Turaga, S., Zhigulin, V., Briggman, K. L., Helmstaedter, M. N., Denk, W., and Seung, H. S. (2007). Supervised Learning of Image Restoration with Convolutional Networks. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. ISSN: 2380-7504.

[LeCun, 1985] LeCun, Y. (1985). Une procedure d'apprentissage pour reseau a seuil asymmetrique (A learning scheme for asymmetric threshold networks). In *proceedings of Cognitiva 85*.

[Li et al., 2016] Li, Y., Qi, H., Dai, J., Ji, X., and Wei, Y. (2016). Fully Convolutional Instance-aware Semantic Segmentation. *arXiv:1611.07709 [cs]*. arXiv: 1611.07709.

References III

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2014). Microsoft COCO: Common Objects in Context. *arXiv:1405.0312 [cs]*. arXiv: 1405.0312.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440.

[Pang et al., 2010] Pang, B., Zhang, Y., Chen, Q., Gao, Z., Peng, Q., and You, X. (2010). Cell Nucleus Segmentation in Color Histopathological Imagery Using Convolutional Networks. In *2010 Chinese Conference on Pattern Recognition (CCPR)*, pages 1–5.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, number 9351 in Lecture Notes in Computer Science, pages 234–241. Springer International Publishing.

[Werbos, 1982] Werbos, P. J. (1982). Applications of advances in nonlinear sensitivity analysis. In Drenick, R. F. and Kozin, F., editors, *System Modeling and Optimization*, Lecture Notes in Control and Information Sciences, pages 762–770, Berlin, Heidelberg. Springer.