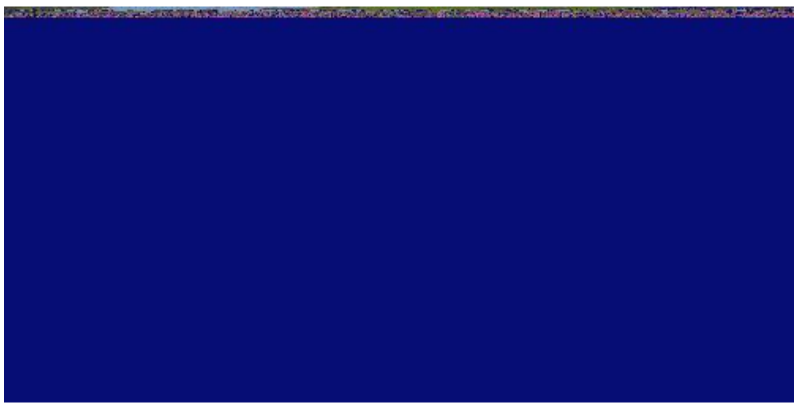


Neural Networks and semantic segmentation

IA&ML Module 4:
Image and Video Processing

Sebastien Drouyer &
Gabriele Facciolo

Lecture 2 - 21-10-2021

Presentation of the course

The objective of this course is to present a panorama of the main modeling aspects and practical insights of neuronal networks (NN) for computer vision applications.

Page: https://gfacciol.github.io/M1_IAML_image/

Lessons:

1. -- Thursday 7/10 (2E34): 14h00-16h30 - Intro NN, backprop and CNN for classification
2. -- Thursday 21/10 (2E34): 13h30-16h00 - Semantic segmentation
3. -- Thursday 18/11 (2E34): 14h00-16h00 - Object detection
4. -- Thursday 25/11 (1B14): 13h30-16h00 - Transfer learning and representation learning

Plan

- Image classification recap
- What is semantic segmentation
- Architectures

Last week recap

Image classification

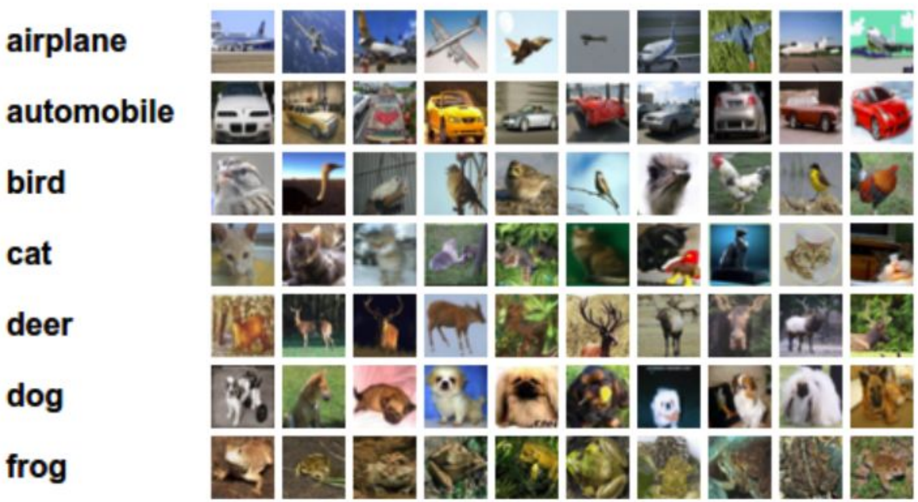
- Image classification is the prototypical computer vision problem
- A nontrivial problem:

$$\mathbf{u}_i \in \mathbb{R}^{H \times W \times 3} \longrightarrow c_i \in \mathcal{G}$$

- Difficult to craft a program to solve it in an unrestricted setting

Data driven approaches

1. Assemble a **dataset** of labeled images
2. **Train a classifier** using the labeled examples
3. **Evaluate** the classifier on new images



A. Krizhevsky, V. Nair, and G. Hinton "CIFAR-10"

O. Russakovsky, et al. "ImageNet Large Scale Visual Recognition Challenge"

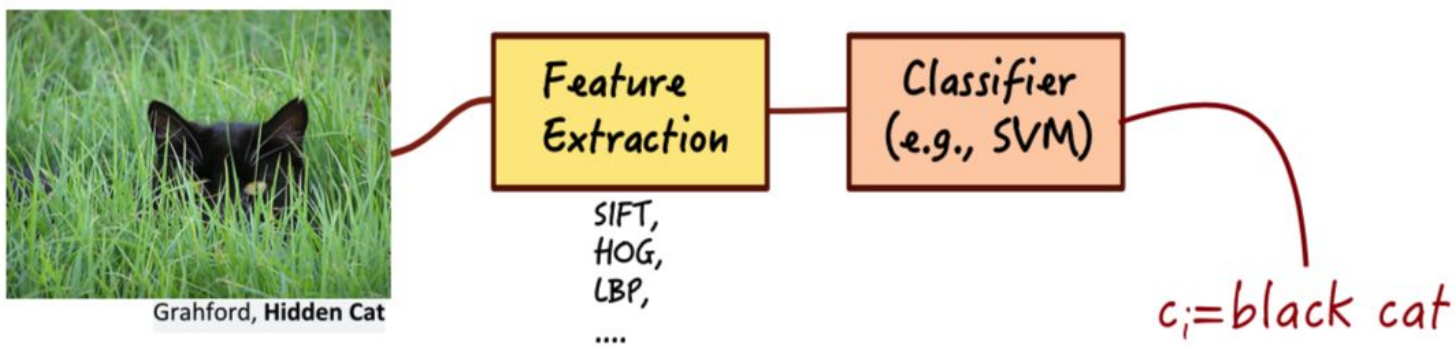
Image classification in detail (side note)

- A classifier is a function f(x) = y
  - x: input (image)
  - y: output (classification)
    - y is not textual, but **a vector of size N** if we need to differentiate between N classes.
    - Each vector's value **represents the likelihood** that the image belongs to the associated class.
    - An image can therefore be classified into one class, or multiple classes depending on the classifier.

- Training is an optimization problem.

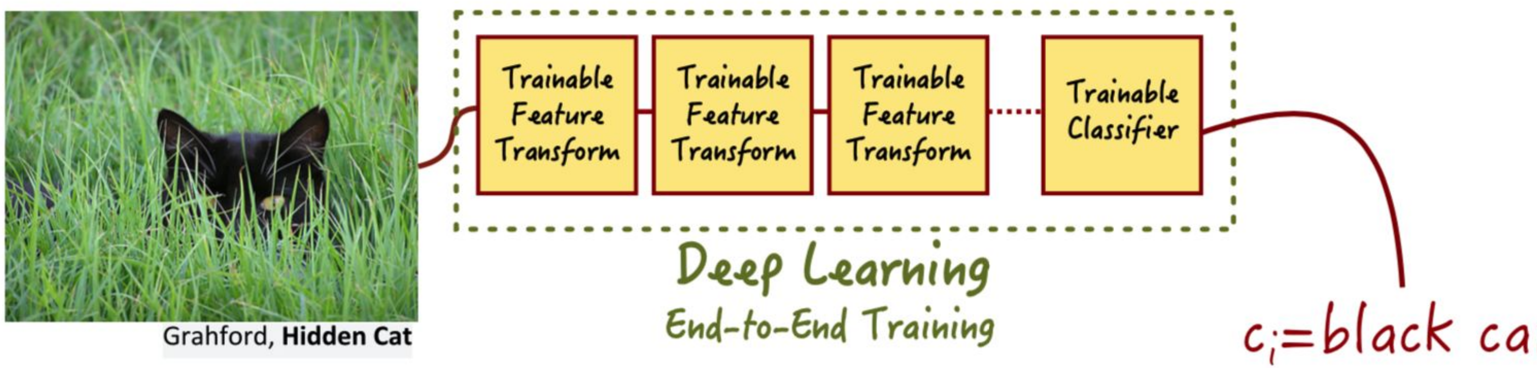
Classic approaches

- First **extract features** (SIFT, HOG...), then feed them to a classifier
- Allows to **reduce the dimension** of the classifier
- **Features are invariant** (to rotation, translation, scale, and illumination changes) and allow to robustly classify



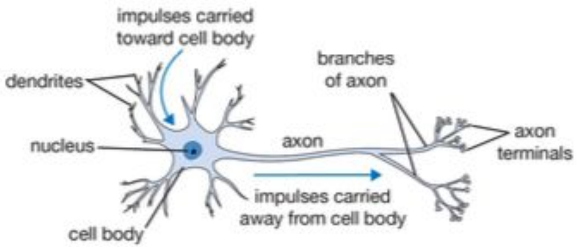
Deep learning approach

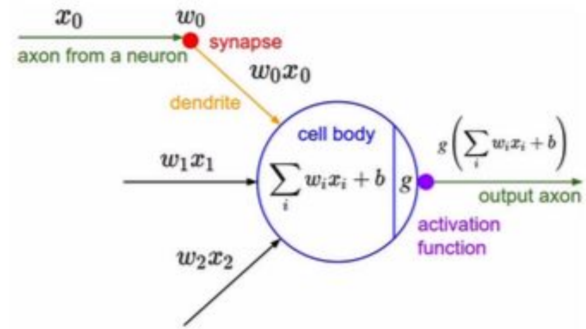
- Learn the features at the same time as the classifier
- Features and classifier are coded in the layers of a DNN
- The network is usually trained in an end-to-end way

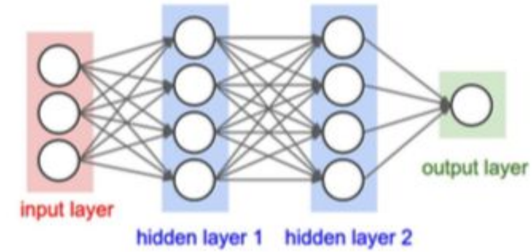


Grahford, **Hidden Cat**

Feedforward Neural Networks







- Neural networks are vaguely inspired on biological neurons
- A neuron/unit is modeled as a composition of an **affine transformation** of its inputs x: $w\,x + b$ and a non-linearity g (**activation function**)

$$f(x) = g(w \cdot x + b)$$

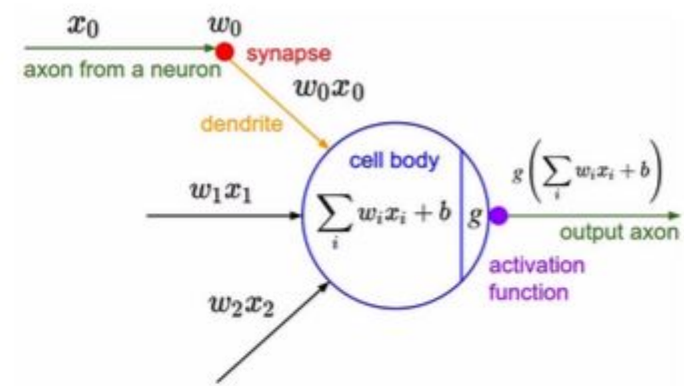
- Often are **grouped in layers**, where each unit is connected to all units from the previous layer

$$y = g_3(b_3 + W_3 \cdot g_2(b_2 + W_2 \cdot g_1(b_1 + W_1 \cdot x)))$$

Perceptron

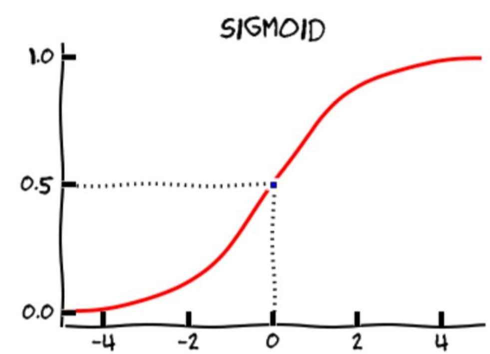
- Binary valued function of its inputs proposed in the 1950's

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0, \\ 0 & otherwise, \end{cases}$$



- The discontinuous Heaviside function makes it hard to train by gradient descent methods
- **Sigmoid** activation is a smooth approximation of Heaviside

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

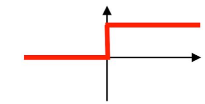
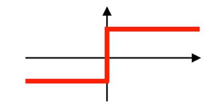
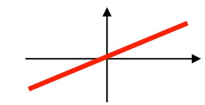
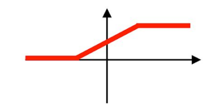
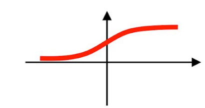
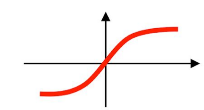
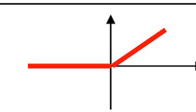
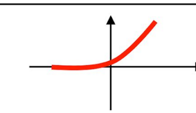


Activation Functions

- **ReLU**: the most frequently used the activation today

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

- Easy to differentiate
- Enable better training of deeper networks

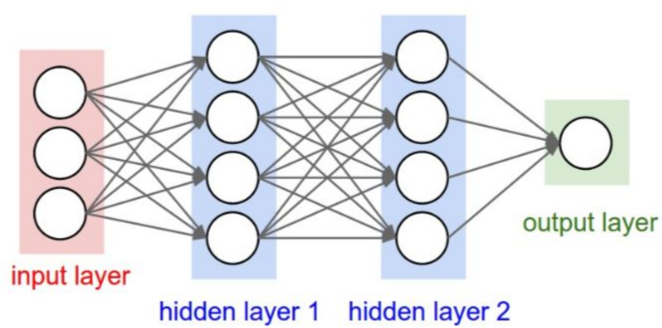
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

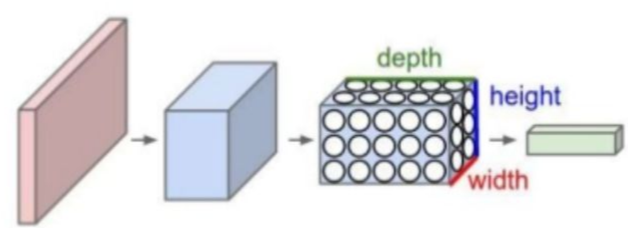
Feedforward Neural Network architecture

- Feedforward networks are often organized in *“layers”*
- The architecture can be specified by an acyclic graph of layers e.g.

$$\mathcal{F}(x) = f_n(f_{n-1}(...(f_2(f_1(x))...))$$

- In image processing and computer vision applications the **input vector has shape *H x W x C*** (height, width, channel)
- ConvNets interpret a layer of neurons as a volume with dimensions (H,W,Depth) , which **preserves the spatial structure of the image**





Activation Functions (side note)

The activation function in the output layer will determine whether the classification is exclusive or not:

- Sigmoid: not exclusive

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

- Softmax: exclusive

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

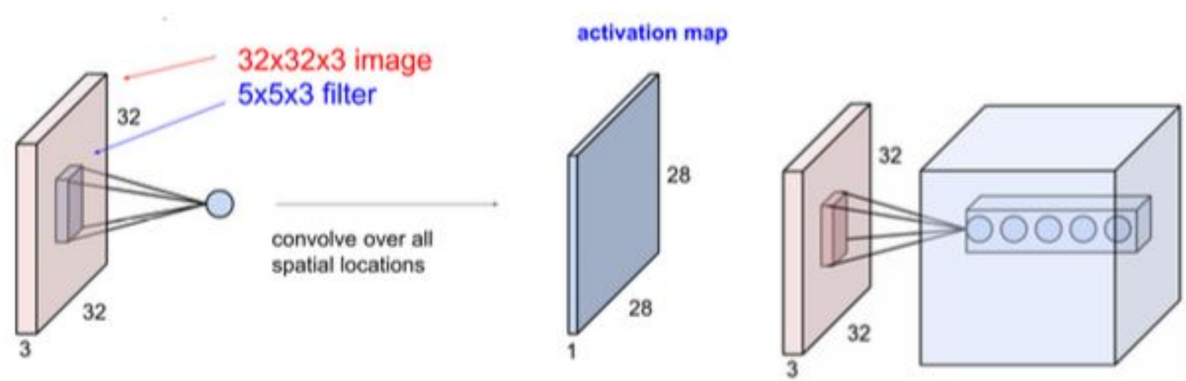
Layers: convolution (Conv)

$$\begin{pmatrix} a & b & 0 & 0 & 0 \\ c & a & b & 0 & 0 \\ 0 & c & a & b & 0 \\ 0 & 0 & c & a & b \\ 0 & 0 & 0 & c & a \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

- A particular case of FC layer

$$y(i,j,l) = b_l + \sum_{(s,t,k) \in \mathrm{supp}(w_l)} x(i+s, j+t, k) w_l(s,t,k)$$

- Each output map is result of convolving the input with a kernel w_l
- Conv layers involve many more connections than unique weights i.e. many connections share the same weight
- Conv layers are translation equivariant

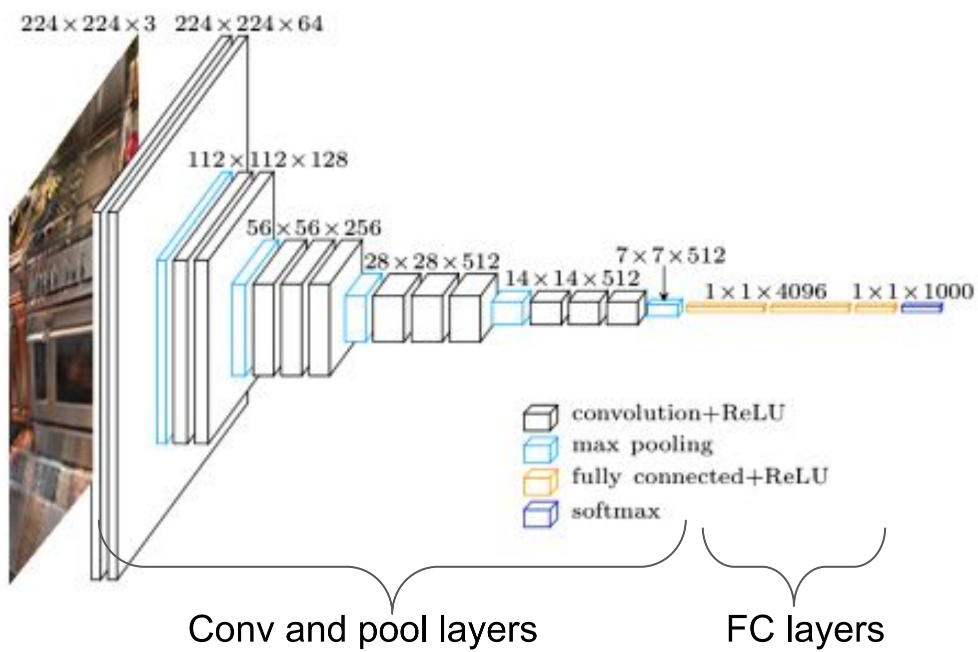


A classification network

VGG [Simonyan, Zisserman. 2014, Very deep convolutional networks for large-scale image recognition.]

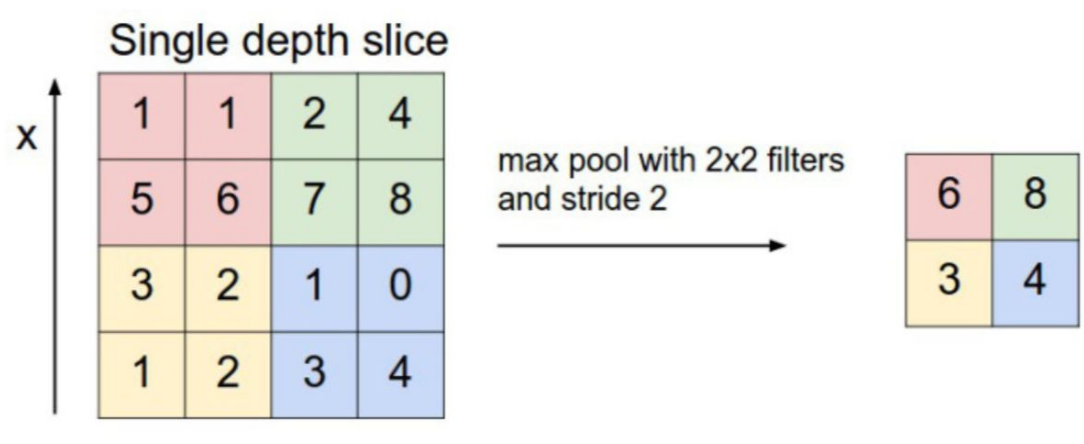
- Encoder type architecture
- Final layers produce a vector of probabilities by applying **softmax**

$$P(Y = j \mid X = \mathbf{x}) = \frac{e^{s_j}}{\sum_k e^{s_k}}$$



Layer: Pooling (POOL)

- Spatial subsampling by **binning of the input features**
- Max Pooling is the most common but average pooling also feasible
- Provides more translation invariance in the feature maps
- The current trend is to use strided convolution instead of pool



Optimization

- Stochastic gradient descent is simple
  - Approximates the gradient of the risk with a small set of training samples (**mini-batch**)
  - Computes the gradient of the mini-batch risk wrt all the parameters and updates them
  - Learning rate τ: controls the step size. It is a very delicate hyperparameter

Algorithm 24: Stochastic gradient descent.

while *stopping criterion not met* **do**
  Sample mini-batch of m samples $x_1, x_2, ..., x_m$ and corresponding targets y_i;
  Compute gradient estimate: $\Delta\theta \longleftarrow \frac{1}{m}\nabla_\theta \sum_i \ell(\mathcal{F}_\theta(x_i), y_i)$
  Update the parameters: $\theta \longleftarrow \theta - \tau \cdot \Delta\theta$

- In practice use adaptive gradient methods with momentum
  - We will use ADAM (Adaptive Moment Optimization) [Kingma, Ba 2014]
- Second order methods also exist ...

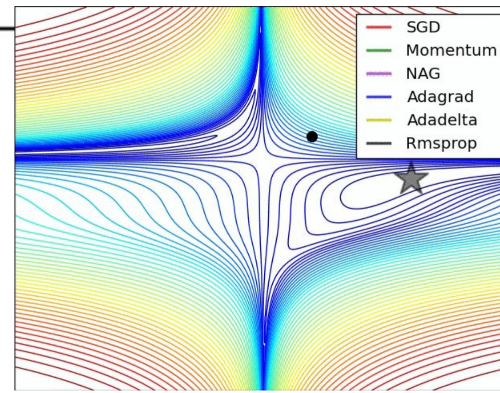
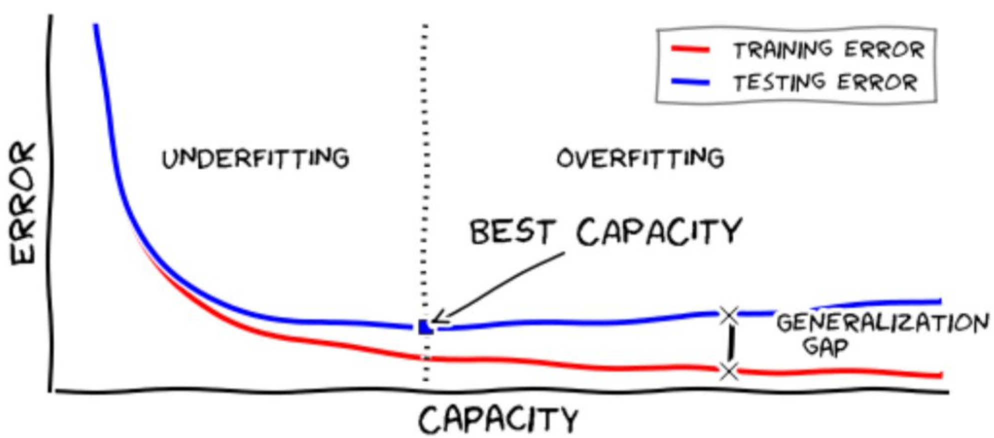


Image credit: Alec Radford

Overfitting and validation

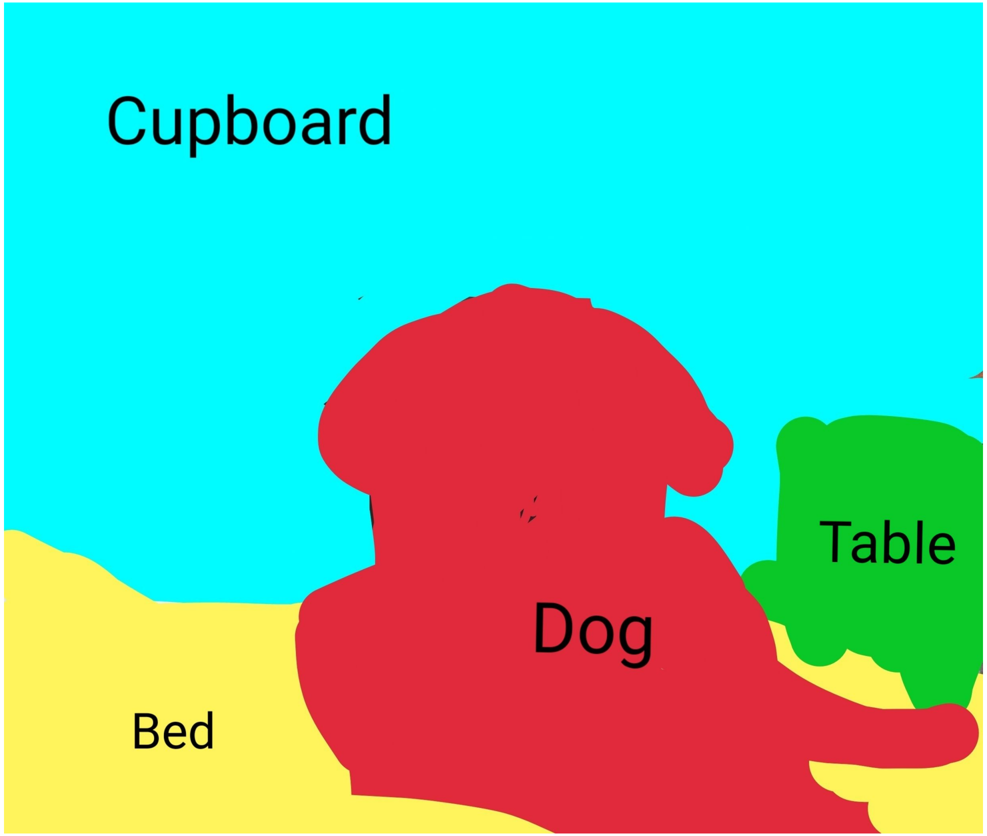
- Defining and estimating the capacity of a NN is still an active research topic. **But we can detect the symptoms of overfitting**.
- The **dataset is split in training, validation, and test sets**
  - Test is used to evaluate the final network. Should only be used once for the final assessment of the performance of the model.
  - Validation is used to monitor the generalization performance during training, allowing to spot overfitting, and tune hyperparameters
  - When train and validation errors diverge too much it is probably due to overfitting



Semantic segmentation

Semantic segmentation

Semantic segmentation: associate label to different areas in an image.



Semantic segmentation

In other words, semantic segmentation consists of classifying each pixel of an image.

→ Classification problem.

Semantic segmentation

A classifier could indeed be trained to classify each pixel independently.

But two neighboring pixels share very similar neighborhood.

→ There is a lot of redundancy when computing convolutions.

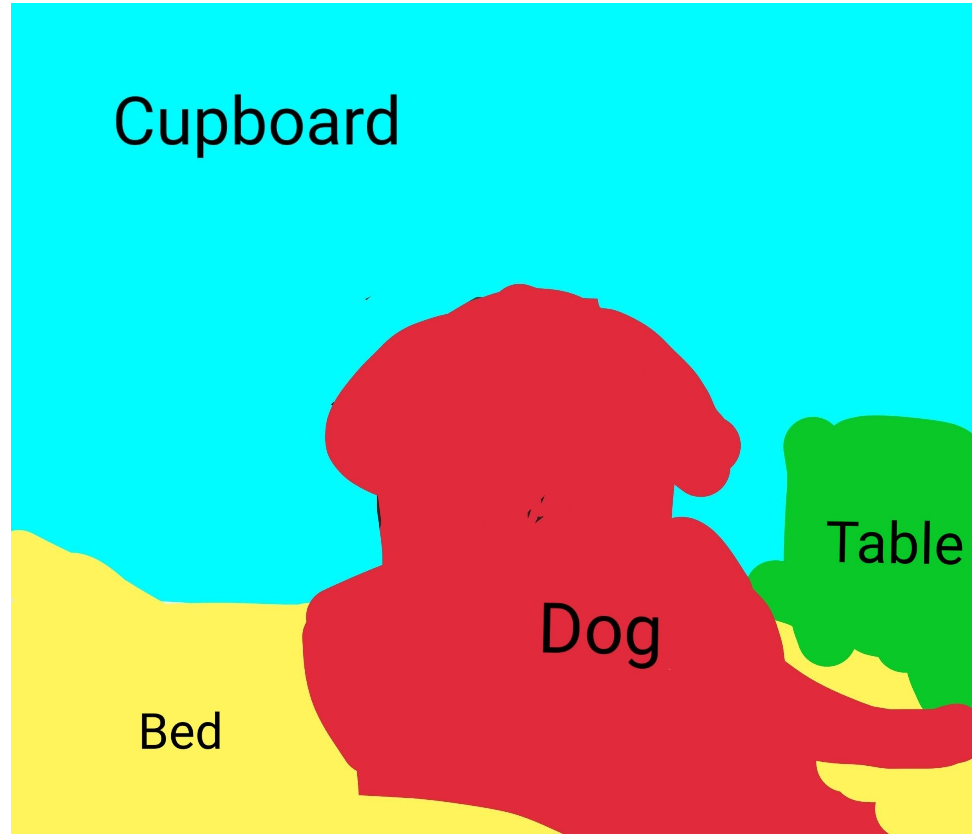
Semantic segmentation

Note

/!\ As for classification, a ground truth is necessary for training the neural network.

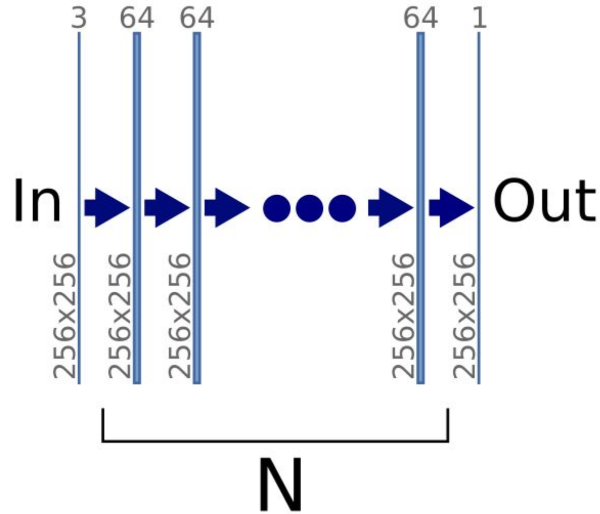
A dataset must be constructed so that each image is associated with a label map. Some annotators are necessary.

Crowdsourcing is sometimes used...



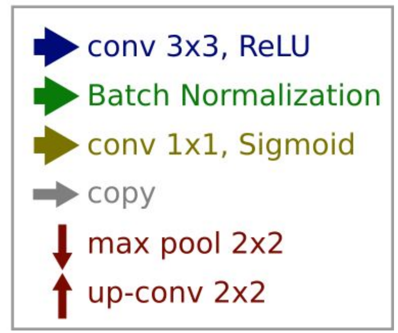
Architectures

Architectures



Solution 1: only apply convolutions.

Problem: need a lot of convolutions to have a good receptive field.



What is the receptive field?

→ How many pixels in the original image have been taken into account to classify one pixel.

If convolutions used are 3x3:

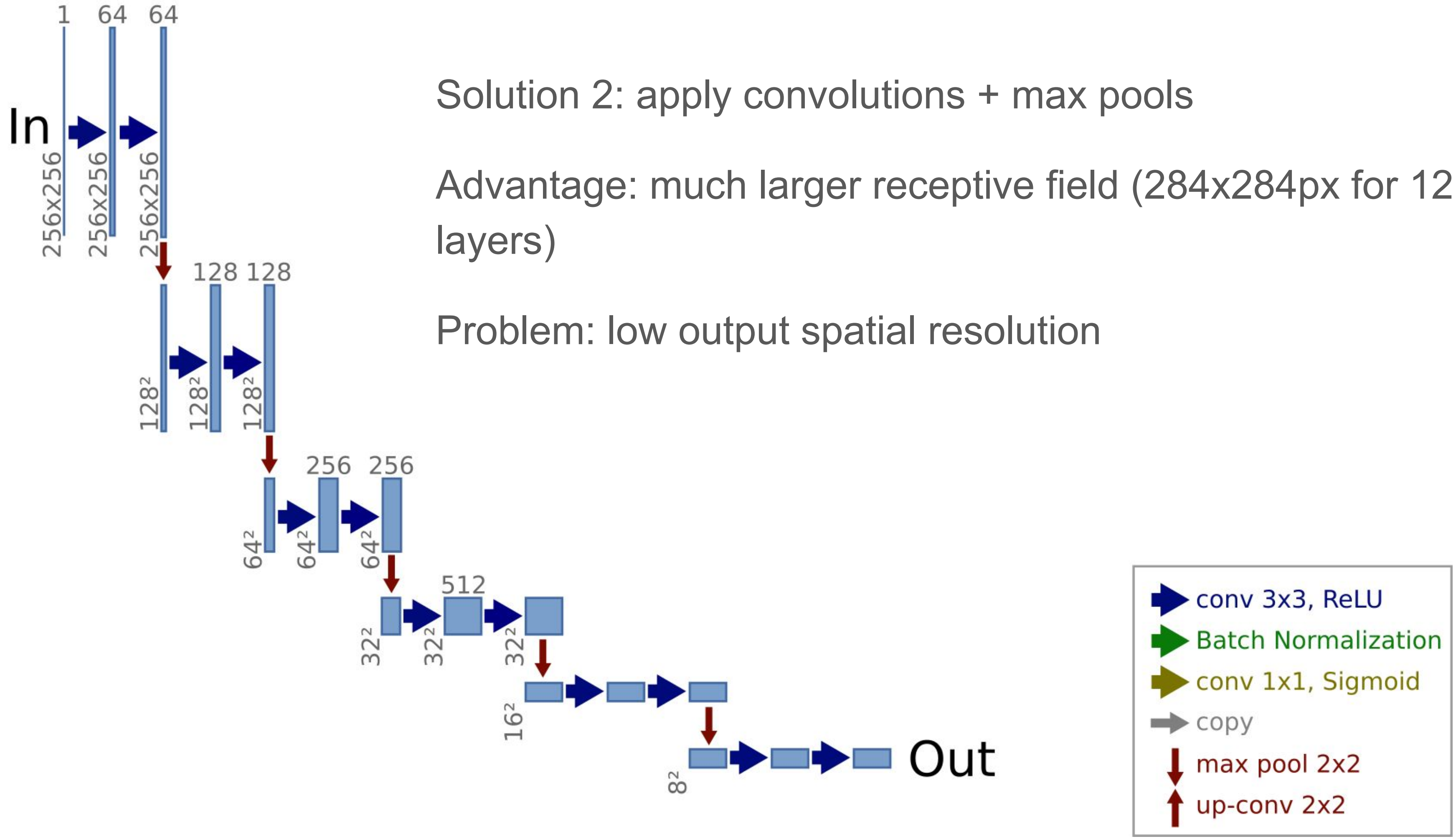
1 layer : 3x3

2 layers: 5x5

N layers: (1 + N x 2) x (1 + N x 2)

We need 49 layers to have a receptive field of 99x99 px. Too much layers, not enough receptive field.

Architectures



Solution 2: apply convolutions + max pools

Advantage: much larger receptive field (284x284px for 12 layers)

Problem: low output spatial resolution

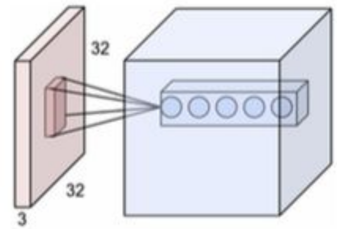
What is the receptive field?

Get back spatial resolution:

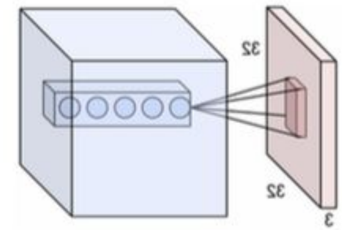
- Interpolation ?

- Transposed convolution

Layers: transposed convolution



Conv



Transposed Conv

- "Splats" the kernel on the output layer (similar to aggregation)
  - **Equivalent to a convolution with the rotated kernel if stride=1**
- It is the transpose of the convolution matrix

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, <u>stride=2</u>, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

When stride>1, convolution transpose is no longer a normal convolution!

Layers: transposed convolution

Transposed convolution of size 2 and stride=2:

1	0
0	0

Image

1	2
3	4

Transposed convolution

1	2	0	0
3	4	0	0
0	0	0	0
0	0	0	0

Output

Layers: transposed convolution

Transposed convolution of size 2 and stride=2:

0	1
0	0

Image

1	2
3	4

Transposed convolution

0	0	1	2
0	0	3	4
0	0	0	0
0	0	0	0

Output

Layers: transposed convolution

Transposed convolution of size 2 and stride=2:

1	1
0	0

Image

1	2
3	4

Transposed convolution

1	2	1	2
3	4	3	4
0	0	0	0
0	0	0	0

Output

Layers: transposed convolution

Transposed convolution of size 3 and stride=2:

1	0
0	0

Image

1	2	3
4	5	6
7	8	9

Transposed convolution

1	2	3	0
4	5	6	0
7	8	9	0
0	0	0	0

Output

Layers: transposed convolution

Transposed convolution of size 3 and stride=2:

0	1
0	0

Image

1	2	3
4	5	6
7	8	9

Transposed convolution

0	1	2	3
0	4	5	6
0	7	8	9
0	0	0	0

Output

Layers: transposed convolution

Transposed convolution of size 3 and stride=2:

1	1
0	0

Image

1	2	3
4	5	6
7	8	9

Transposed convolution

1	3	5	3
4	9	11	6
7	15	17	9
0	0	0	0

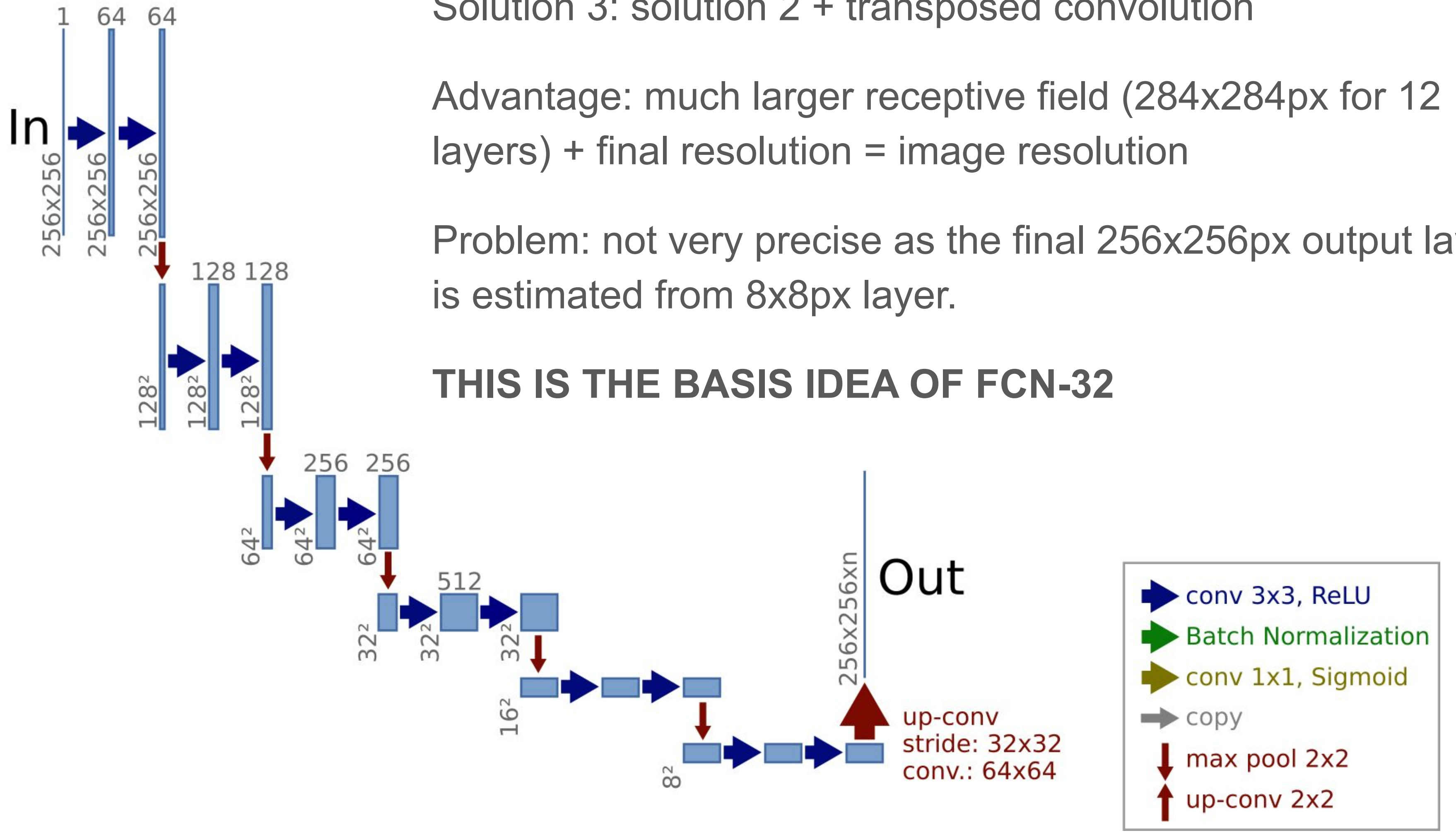
Output

Layers: transposed convolution

In summary:

- If convolution = stride → no overlapping
- If convolution > stride → overlapping. Can be useful for implementing a "smart smoothing".

Architectures



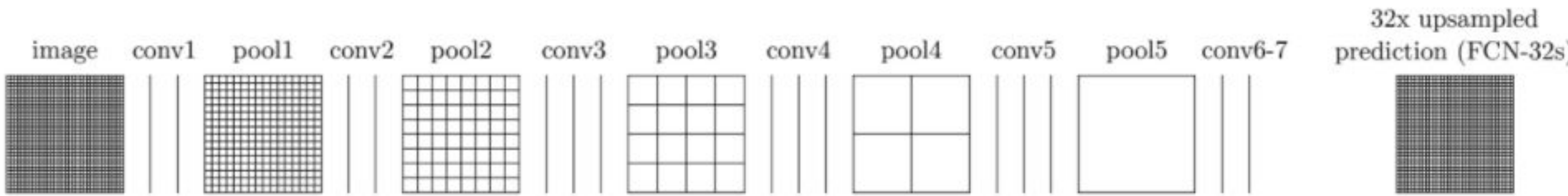
Solution 3: solution 2 + transposed convolution

Advantage: much larger receptive field (284x284px for 12 layers) + final resolution = image resolution

Problem: not very precise as the final 256x256px output layer is estimated from 8x8px layer.

THIS IS THE BASIS IDEA OF FCN-32

Architectures



Extract of Figure 3 of [1]

[1] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

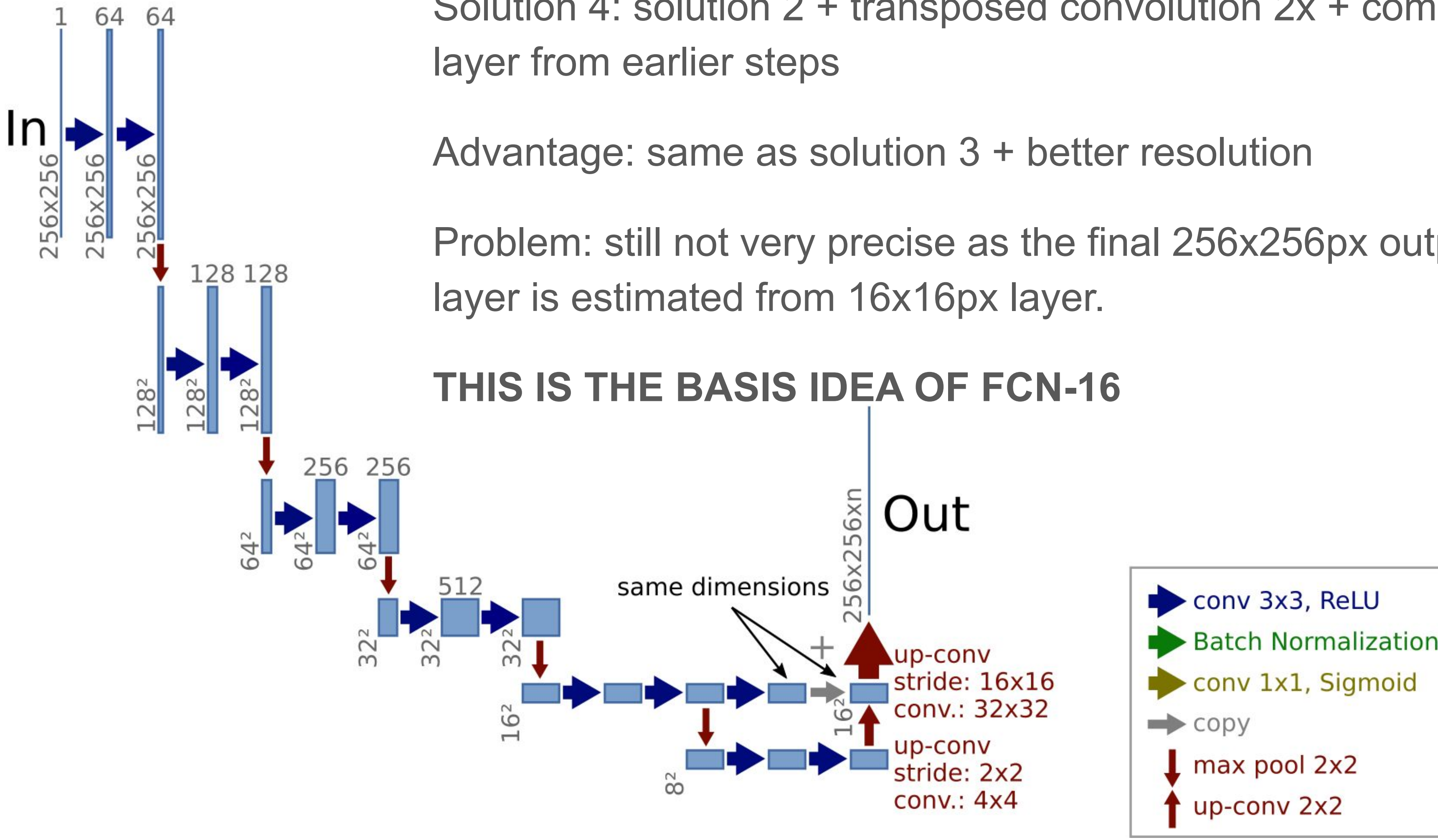
Architectures

Solution 4: solution 2 + transposed convolution 2x + combine layer from earlier steps

Advantage: same as solution 3 + better resolution

Problem: still not very precise as the final 256x256px output layer is estimated from 16x16px layer.

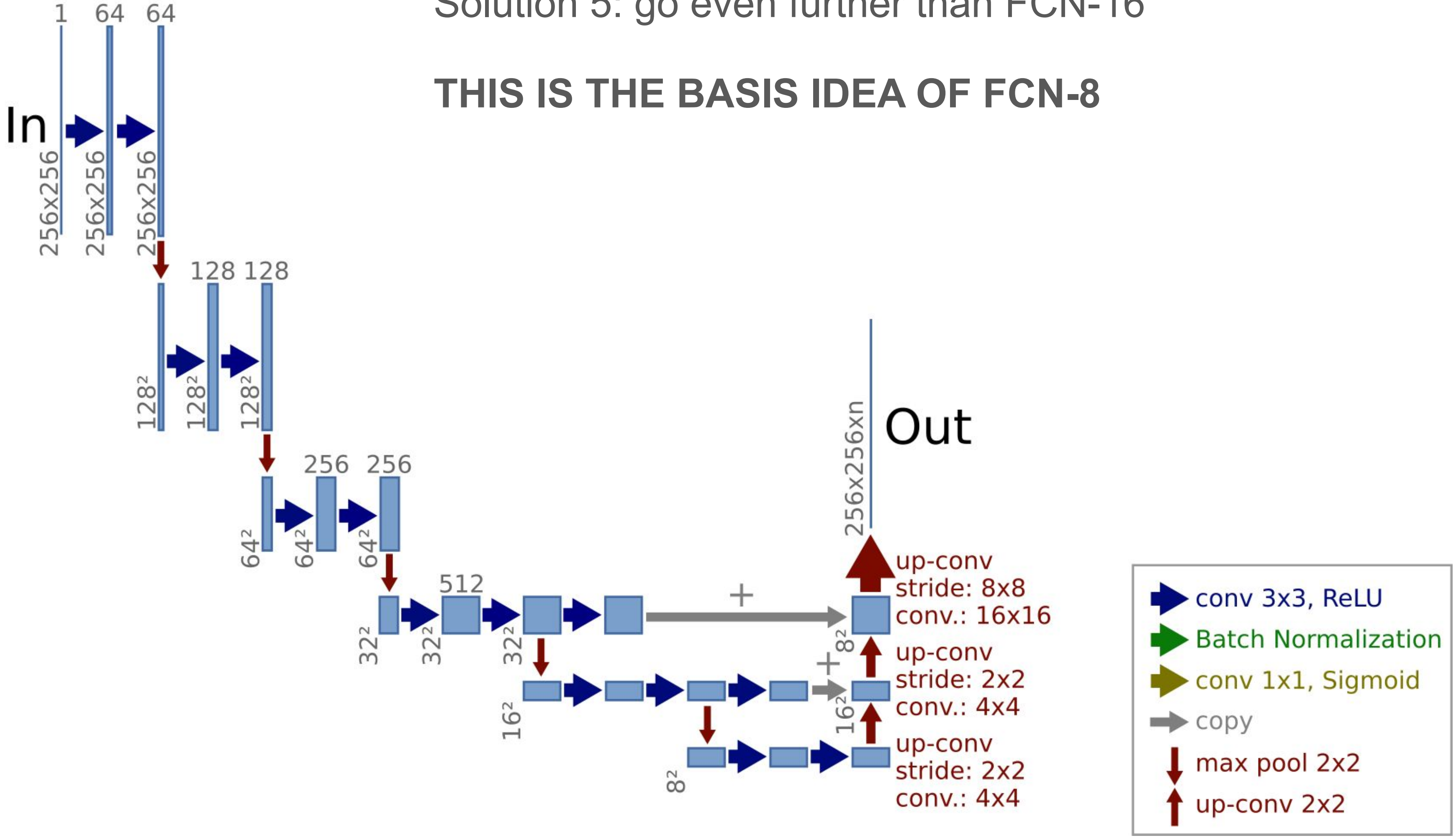
THIS IS THE BASIS IDEA OF FCN-16



Architectures

Solution 5: go even further than FCN-16

THIS IS THE BASIS IDEA OF FCN-8



Architectures

/!\ A lot of variants exist!

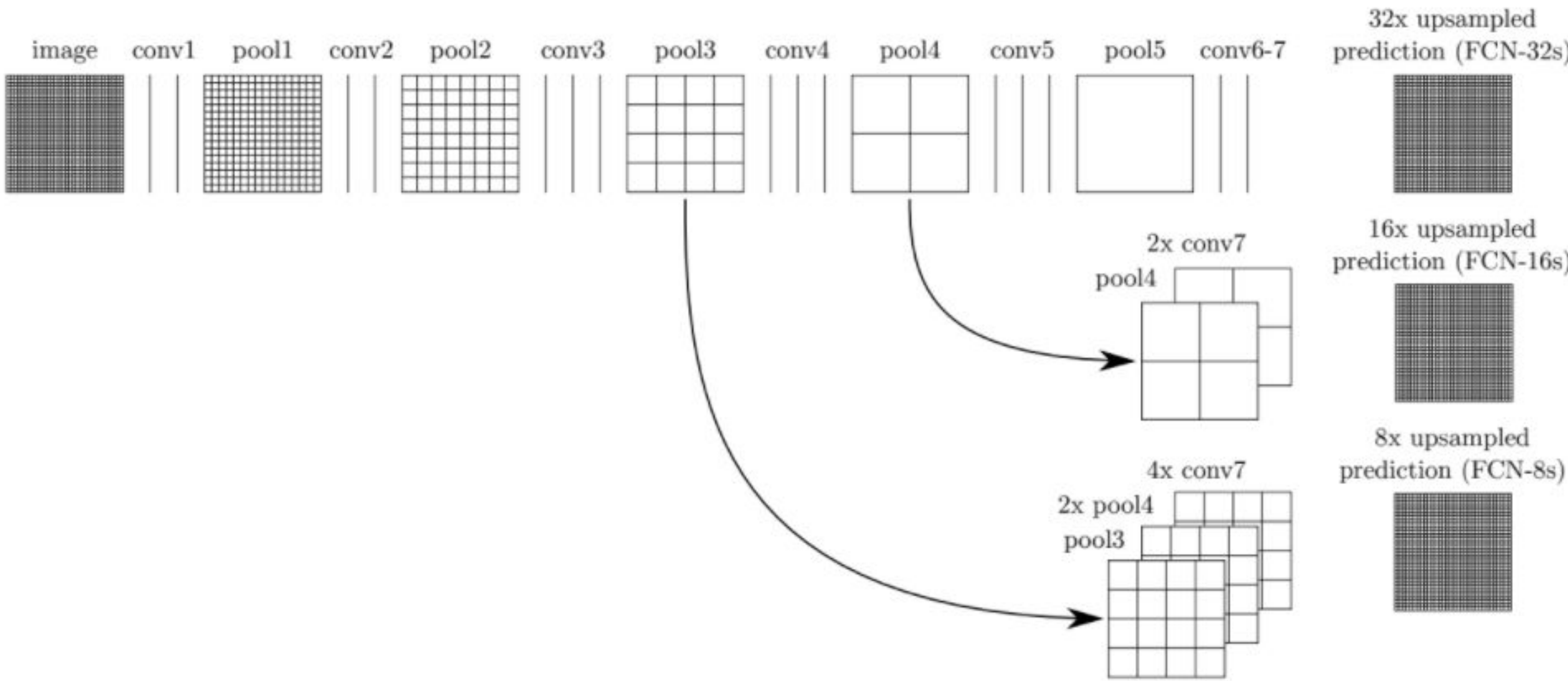


Figure 3 of [1]

[1] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

Architectures

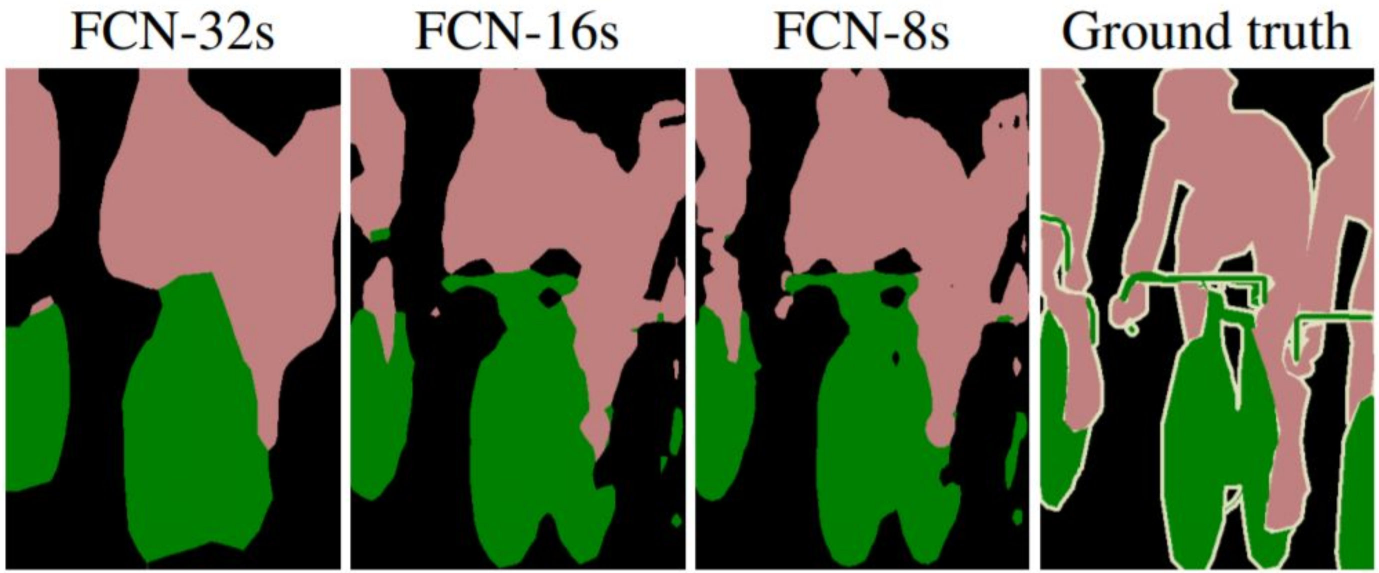


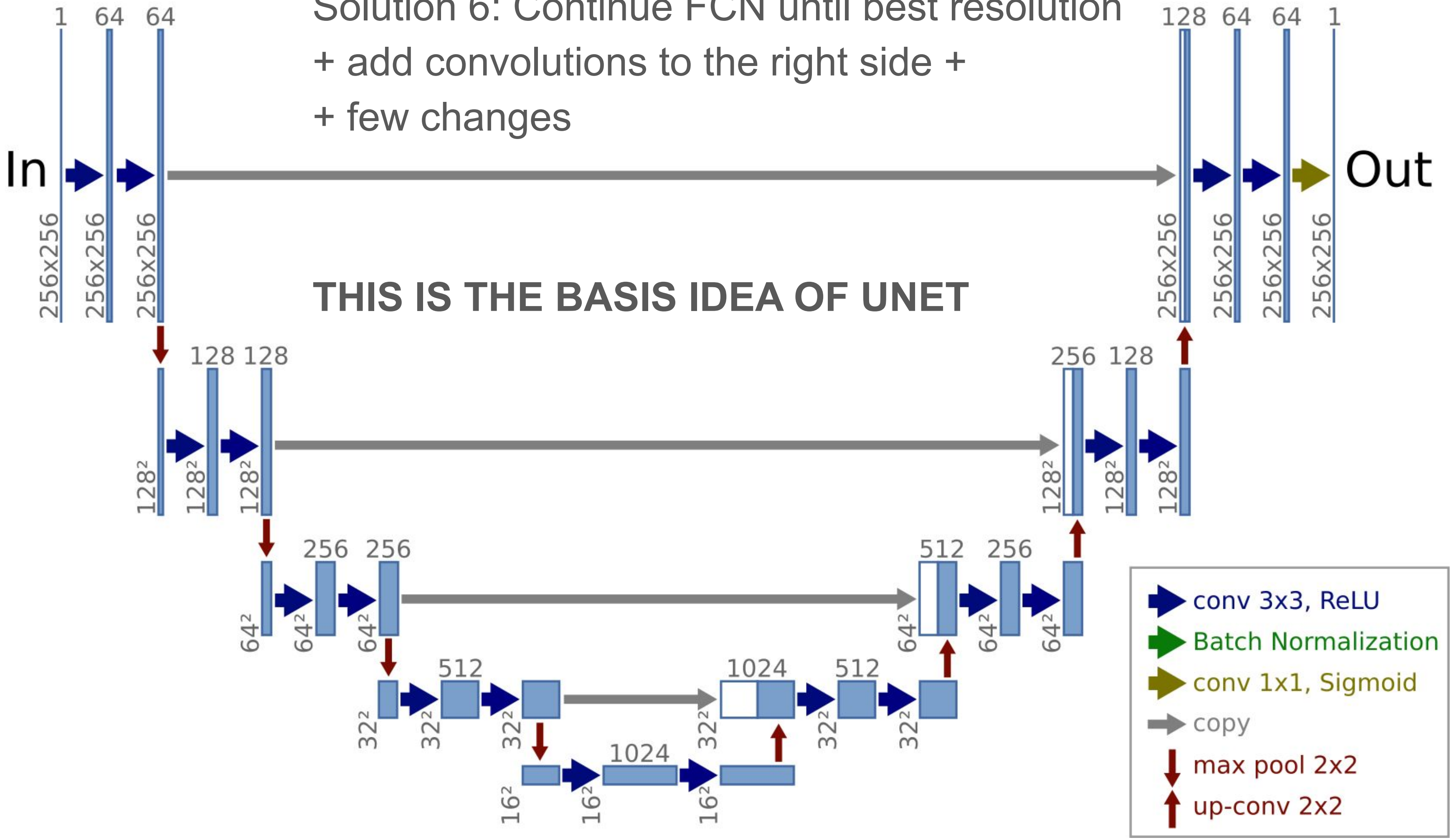
Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

Architectures

Solution 6: Continue FCN until best resolution
+ add convolutions to the right side +
+ few changes

THIS IS THE BASIS IDEA OF UNET

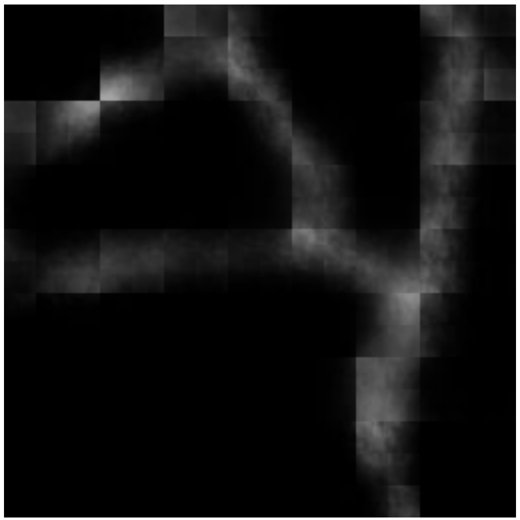


Architectures

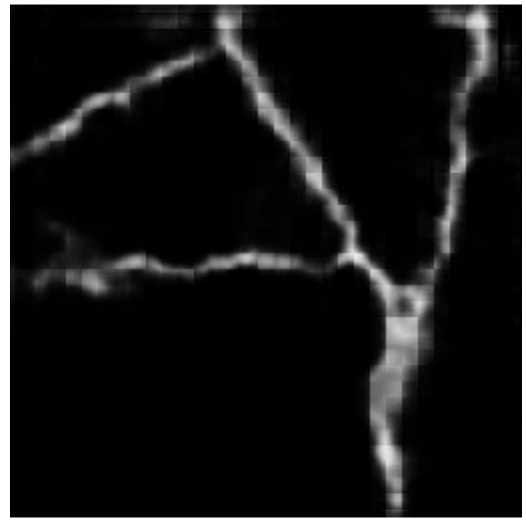
Image

An 'All Terrain' Crack Detector Obtained by Deep Learning on Available Databases, IPOL.

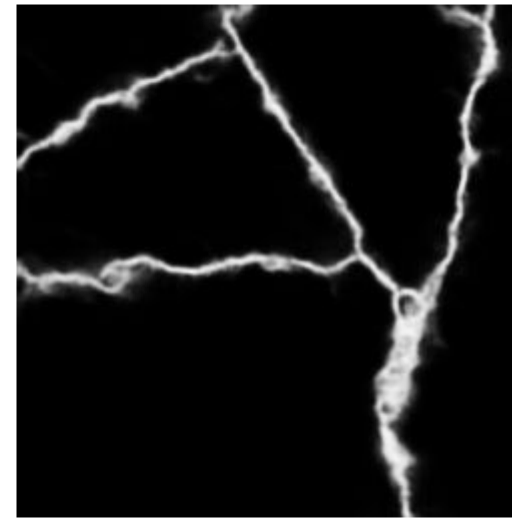
Architectures



FCN-32



FCN-8

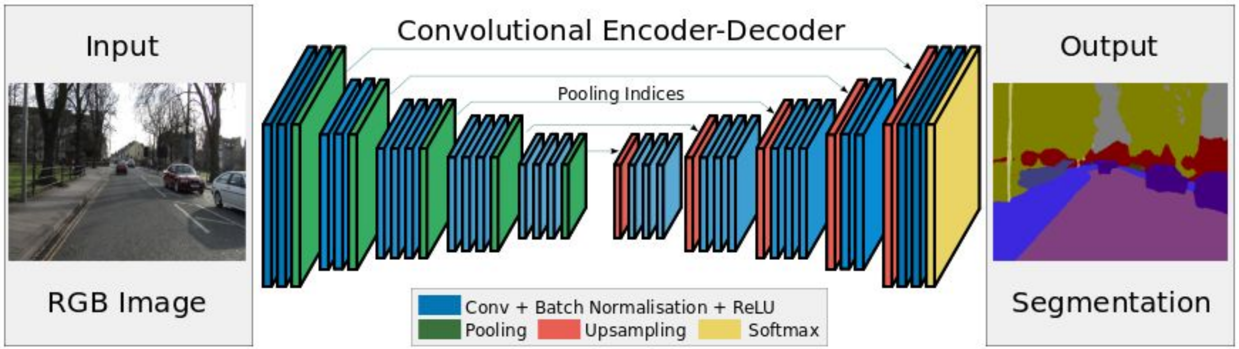


Unet

An 'All Terrain' Crack Detector Obtained by Deep Learning on Available Databases, IPOL.

Architectures

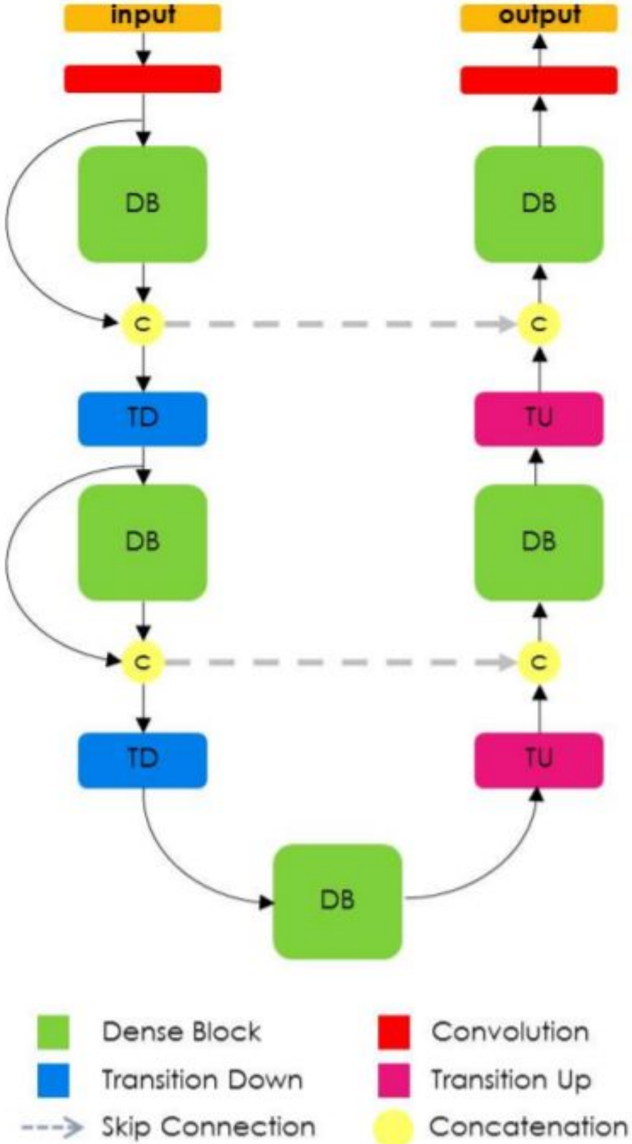
A lot of structures are inspired from U-net



Segnet: A deep convolutional encoder-decoder architecture for image segmentation. Badrinarayanan, Kendall, Cipolla. 2017

Architectures

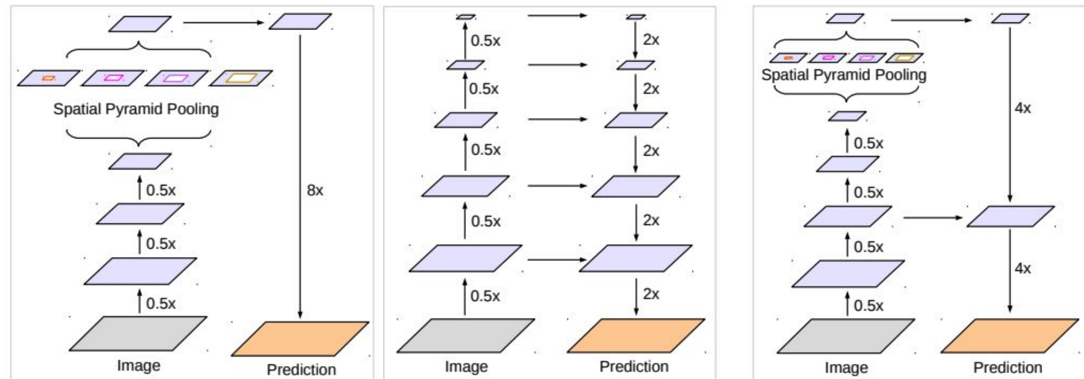
A lot of structures are inspired from U-net



The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation
Jegou, Drozdzal, Vazquez, Romero, Bengio; 2017

Architectures

A lot of structures are inspired from U-net



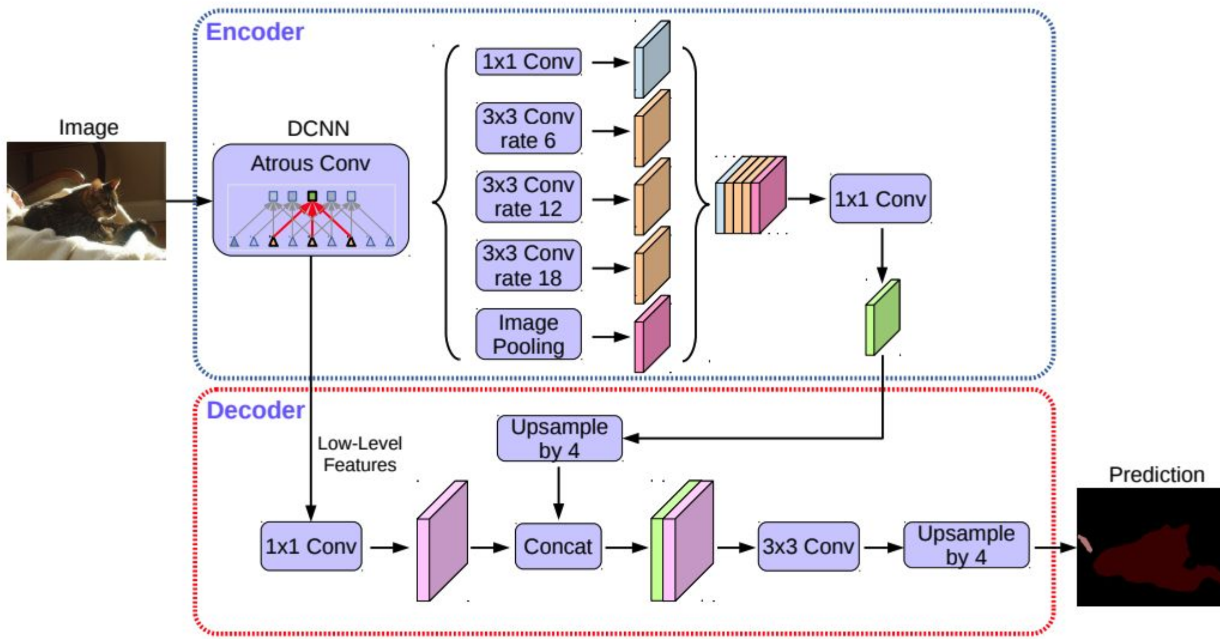
(a) Spatial Pyramid Pooling (b) Encoder-Decoder (c) Encoder-Decoder with Atrous Conv

Fig. 1. We improve DeepLabv3, which employs the spatial pyramid pooling module (a), with the encoder-decoder structure (b). The proposed model, DeepLabv3+, contains rich semantic information from the encoder module, while the detailed object boundaries are recovered by the simple yet effective decoder module. The encoder module allows us to extract features at an arbitrary resolution by applying atrous convolution.

Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. Chen, Zhu, Papandreou, Schroff, Adam. 2018

Architectures

A lot of structures are inspired from U-net



Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. Chen, Zhu, Papandreou, Schroff, Adam. 2018

Homework due by 10/12/2020

The field of computer vision is constantly evolving. So Finding latest works on a subject and determine their usefulness for our needs is an essential skill

- Choose a modern **semantic segmentation network** (with available pretrained weights and example usage code)
    - Attention: the weights will be trained for a specific dataset
- Identify an older pretrained semantic segmentation **network trained on the same dataset.**
- Build a “benchmark” dataset by selecting 4 (le minimum syndical) images from the internet (that are not part of the dataset used for train/validate)
- **Analyze critically** the results of the two networks in a report + notebook.