Convolutional Neural Networks for Semantic Segmentation

Giacomo Boracchi

giacomo.boracchi@polimi.it

Data Pre-processing and Batch Normalization

G. Boracchi

Preprocessing

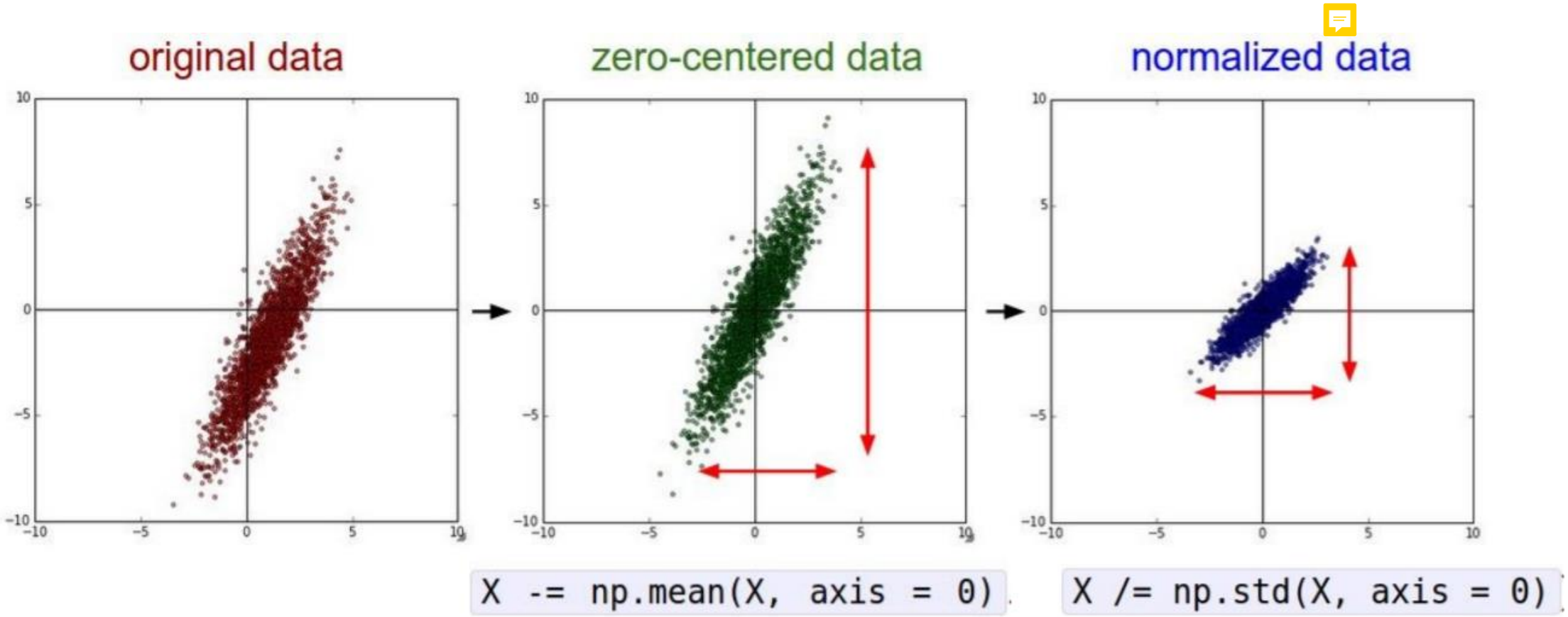
In general, normalization is useful in gradient-based optimizers.

Normalization is meant to bring training data “around the origin” and possibly further rescale the data

In practice, optimization is made easier and results are less sensitive to perturbations in the parameters

There are several options

There are different form of preprocessing



CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Preprocessing for CNN

PCA/Whitening preprocessing are not commonly used with Convolutional Networks.

The most frequent option is to zero-center the data, and it is common to normalize every pixel as well.

Preprocessing and Training:

- Any preprocessing statistics (e.g. the data mean) must be computed on training data, and applied to the validation / test data.
- Do not normalize first and then split in training, validation, test
- Normalization statistics are parameters of your ML model

Preprocessing for CNNs: mean subtraction

e.g. consider CIFAR-10 example with [32,32,3] images

Subtract the mean image (e.g. AlexNet)
(*mean image = [32,32,3] array*)

Subtract per-channel mean (e.g. VGGNet)
(*mean along each channel = 3 numbers*)

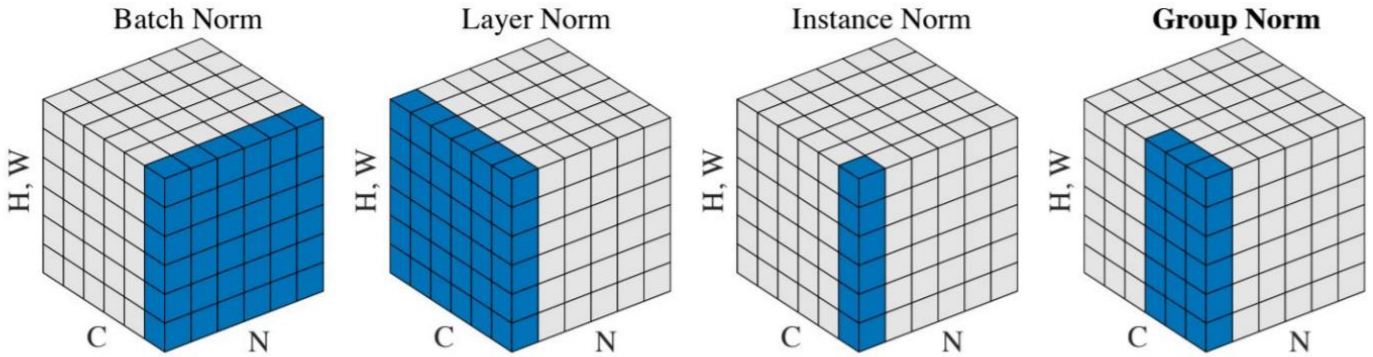
Subtract per-channel mean and Divide by per-channel std (e.g. ResNet)
(*mean and std along each channel = 3 + 3 numbers*)

Batch Normalization

Consider a batch of activations $\{x_i\}$, the following transformation bring these to unit variance and zero mean

$$x_i' = \frac{x_i - E[x_i]}{\sqrt{\text{var}[x_i]}}$$

Where $E[x_i]$ and $\sqrt{\text{var}[x_i]}$ are computed from each batch and separately for each channel!



Wu and He, "Group Normalization", ECCV 2018

Can we get more flexibility than zero-mean, unit variance?

Ioffe, S. and Szegedy, C., 2015, June. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR.

Batch Normalization

Batch normalization adds after standard normalization

$$x_i' = \frac{x_i - E[x_i]}{\sqrt{\text{var}[x_i]}}$$

a further a parametric transformation

$$y_{i,j} = \gamma_j x_i' + \beta_j$$

Where parameters γ and β are learnable scale and shift parameters.

Rmk: estimates $E[x_i]$ and $\sqrt{\text{var}[x_i]}$ are computed on each minibatch, need to be fixed after training. **After training, these are replaced by (running) averages of values seen during training.**

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Ioffe, S. and Szegedy, C., 2015, June. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR.

Batch Normalization

During testing batch normalization becomes a linear operator! Can be fused with the previous fully-connected or conv layer

In practice networks that use Batch Normalization are significantly more robust to bad initialization

Typically Batch Normalization is used in between FC layers of deep CNN, but sometimes also between Conv Layers

Ioffe, S. and Szegedy, C., 2015, June. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR.

Batch Normalization

Pros:

- Makes deep networks much easier to train!
- Improves gradient flow
- Allows higher learning rates, faster convergence
- Networks become more robust to initialization
- Acts as regularization during training
- Zero overhead at test-time: can be fused with conv!

Watch out:

- Behaves differently during training and testing: this is a very common source of bugs!

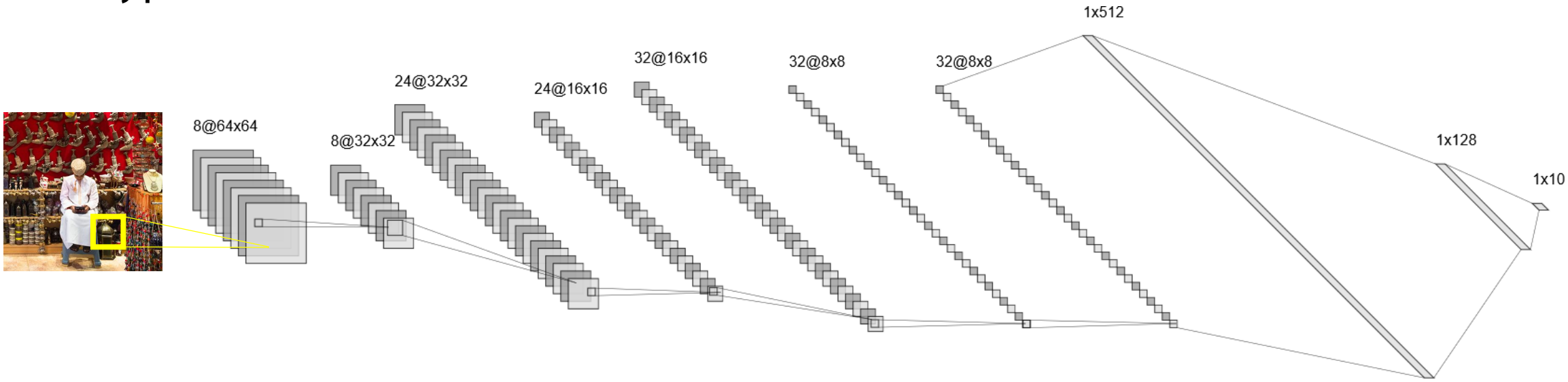
Fully Convolutional Networks

What happens when feeding the network with an image having different size?

G. Boracchi

Convolutional Neural Networks (CNN)

The typical architecture of a convolutional neural network



CNNs are meant to process input of a fixed size (e.g. 200 x 200).

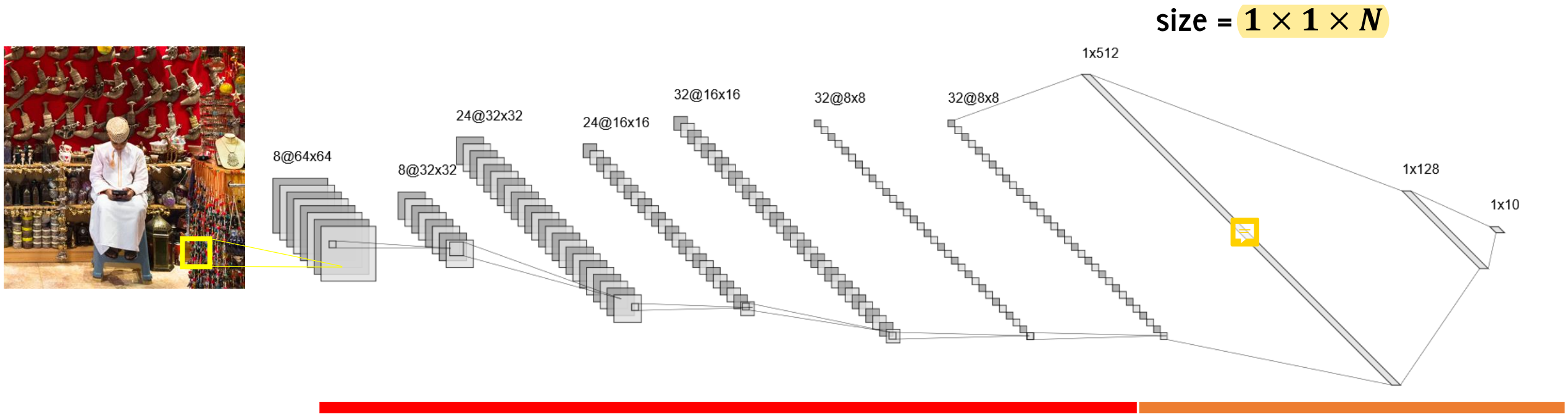
The **convolutional and subsampling layers** operate in a sliding manner over image having arbitrary size

The **fully connected** layer constrains the input to a fixed size.

G. Boracchi

Convolutional Neural Networks (CNN)

The typical architecture of a convolutional neural network



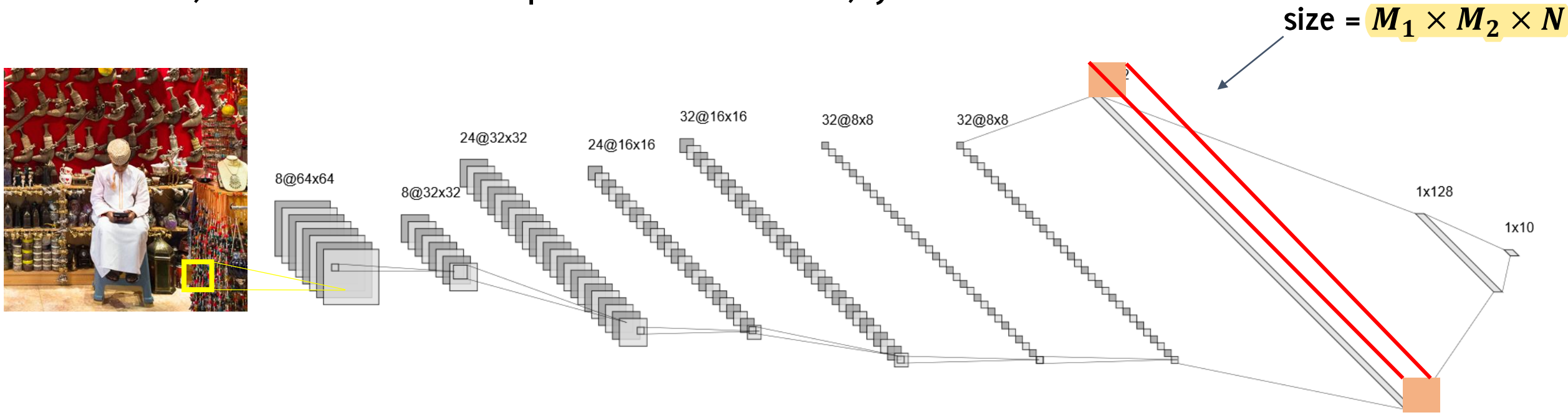
What happens when we feed a larger image to the network?

Convolutional Neural Networks (CNN)

Convolutional filters can be applied to volumes of any size, yielding larger volumes in the network until the FC layer.

The **FC network however does require a fixed input size**

Thus, CNN cannot compute class scores, yet can extract features!

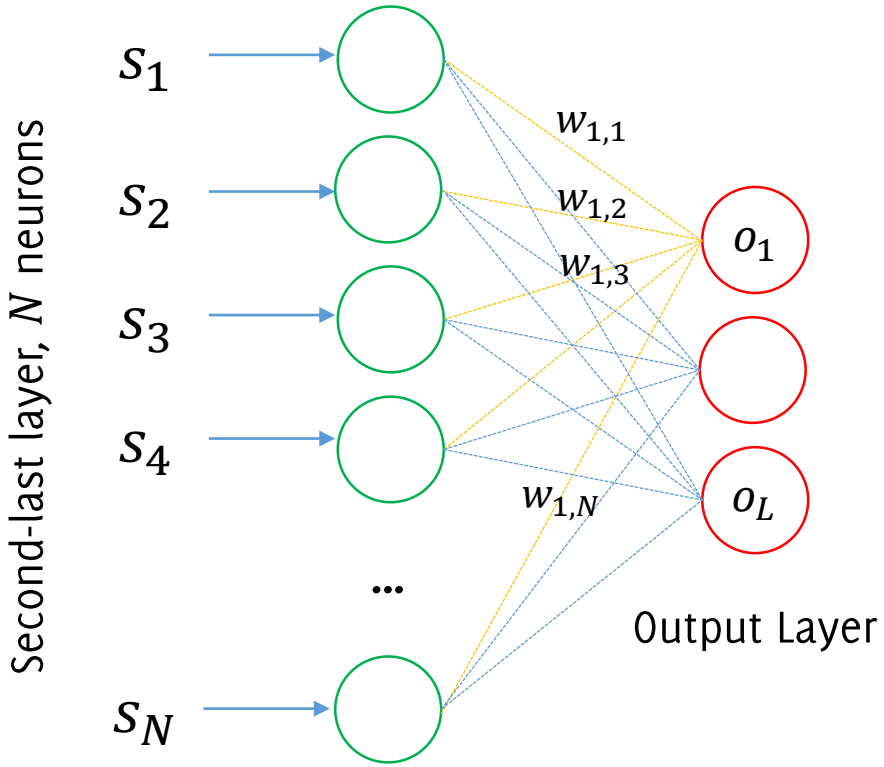


G. Boracchi

Fully Convolutional Neural Networks (FC-CNN)

However, **since the FC is linear, it can be represented as convolution!**

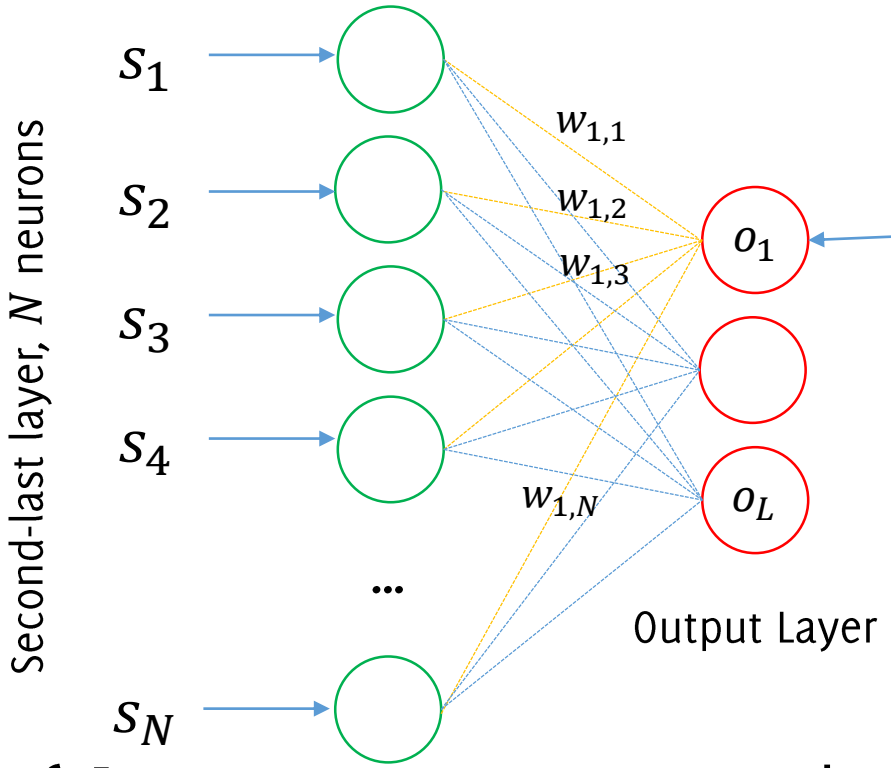
Weights associated to output neuron i : $\boldsymbol{w_i} = \left[w_{i,j}\right]_{j=1:N}$



Fully Convolutional Neural Networks (FC-CNN)

However, **since the FC is linear, it can be represented as convolution!**

Weights associated to output neuron i : $\boldsymbol{w_i} = \left[w_{i,j}\right]_{j=1:N}$



$$o_i = \sum_{j=1:N} w_{i,j} s_j + b_i$$

$$o_i = (\boldsymbol{w_i})' \boldsymbol{s} + b_i$$

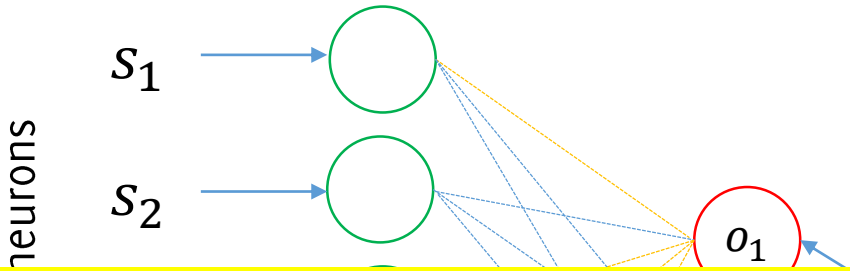
$$o_i = (\boldsymbol{w_i} \circledast \boldsymbol{s})(0,0) + b_i$$

A FC layer of L outputs NN corresponds to a 2DConv layer having **L filters with size $1 \times 1 \times N$**

Fully Convolutional Neural Networks (FC-CNN)

However, **since the FC is linear, it can be represented as convolution!**

Weights associated to output neuron i : $\boldsymbol{w_i} = \left[w_{i,j}\right]_{j=1:N}$



$$o_i = \sum w_{i,j} s_j + b_i$$

This transformation can be applied to each hidden layer of a FC network placed at the CNN top

This transformation consists in reading weights from the neuron in the dense layer end recycling together with bias in the new convolutional layer.

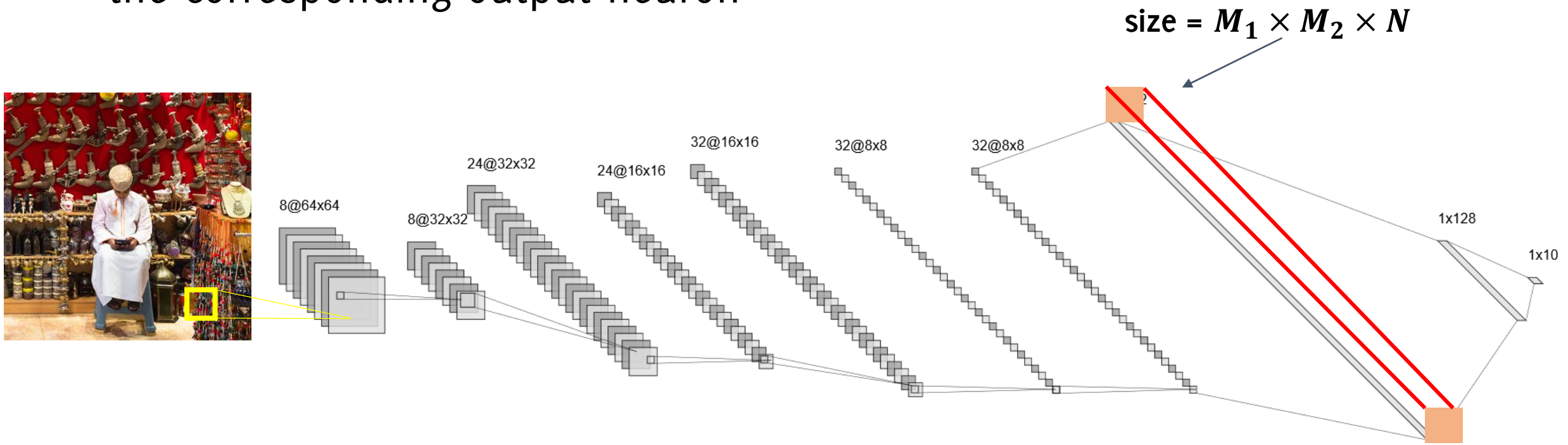
A FC layer

L filters with size $1 \times 1 \times N$

Fully Convolutional Neural Networks (FC-CNN)

However, since the FC is linear, it can be represented as convolution against L filters of size $1 \times 1 \times N$

Each of **these convolutional filters contains the weights of the FC** for the corresponding output neuron

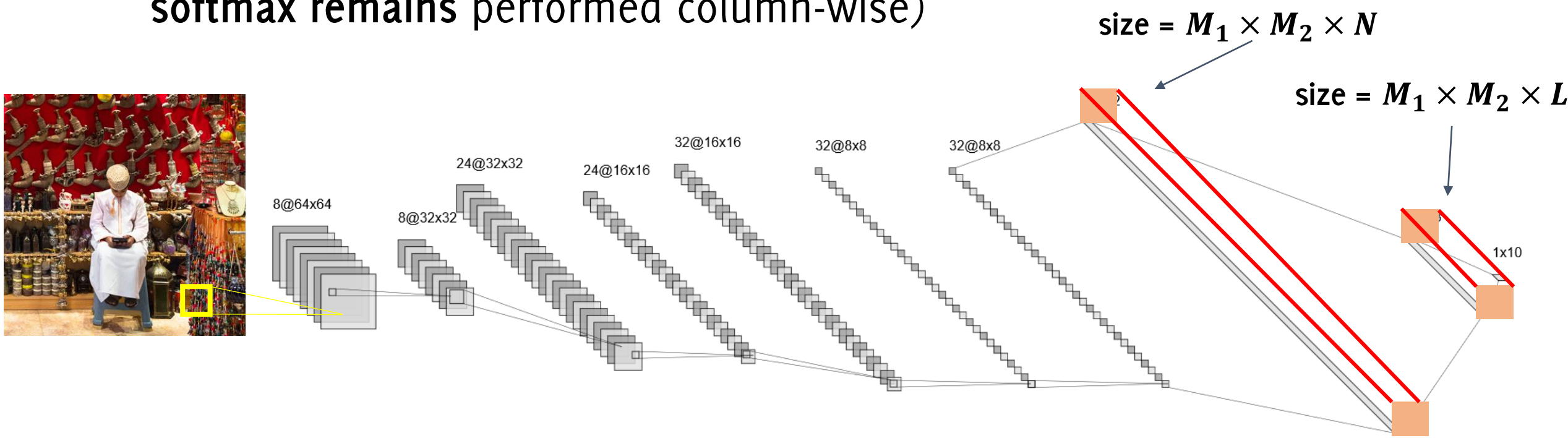


Long, J., Shelhamer, E., Darrell, T. "Fully convolutional networks for semantic segmentation". CVPR 2015

Fully Convolutional Neural Networks (FC-CNN)

For each output class we obtain an image, having:

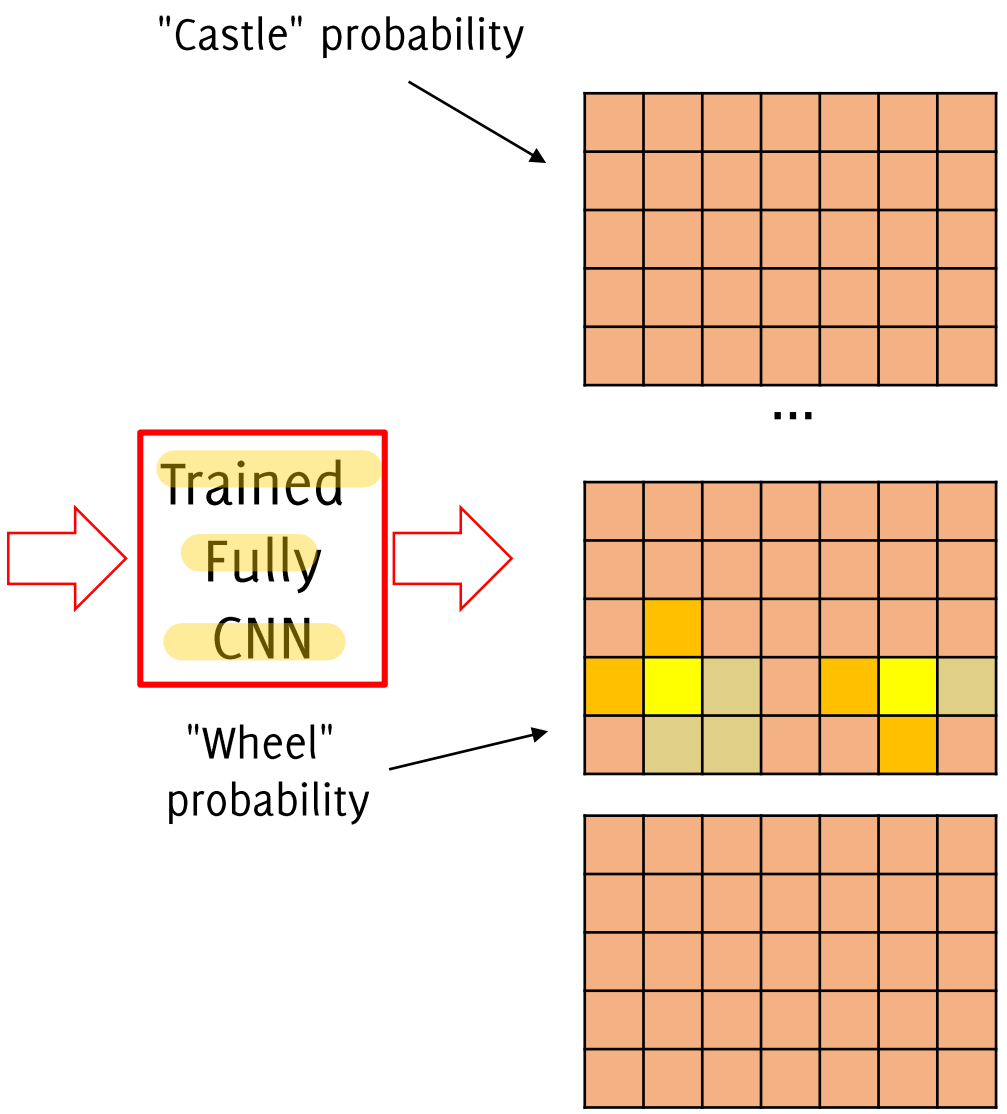
- **Lower resolution** than the input image
- **Class probabilities** for the receptive field of each pixel (assuming **softmax remains** performed column-wise)



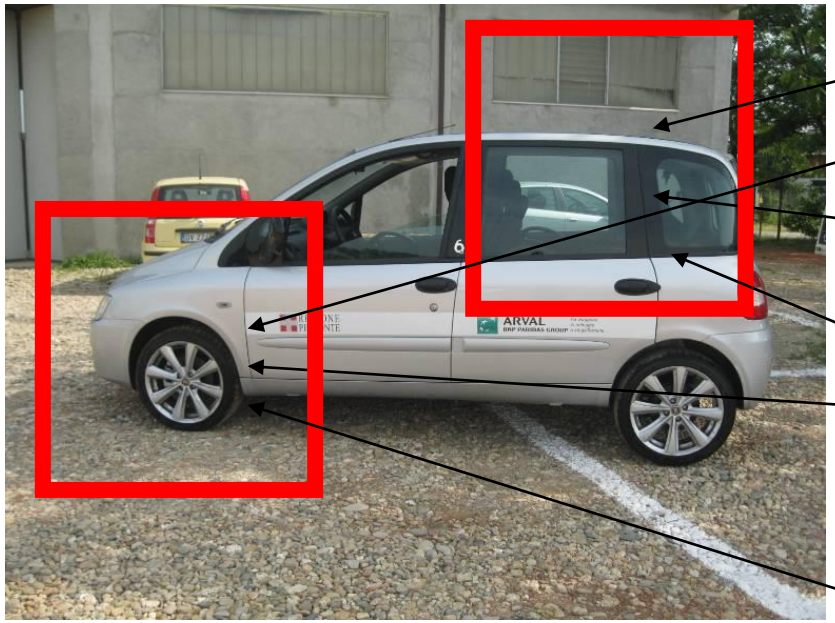
Long, J., Shelhamer, E., Darrell, T. "Fully convolutional networks for semantic segmentation". CVPR 2015

Output of a FC-CNN as heatmaps

A larger image than those used for training the network



Output of a FCNN as heatmaps



Output heatmaps computed after softmax

"Castle" probability

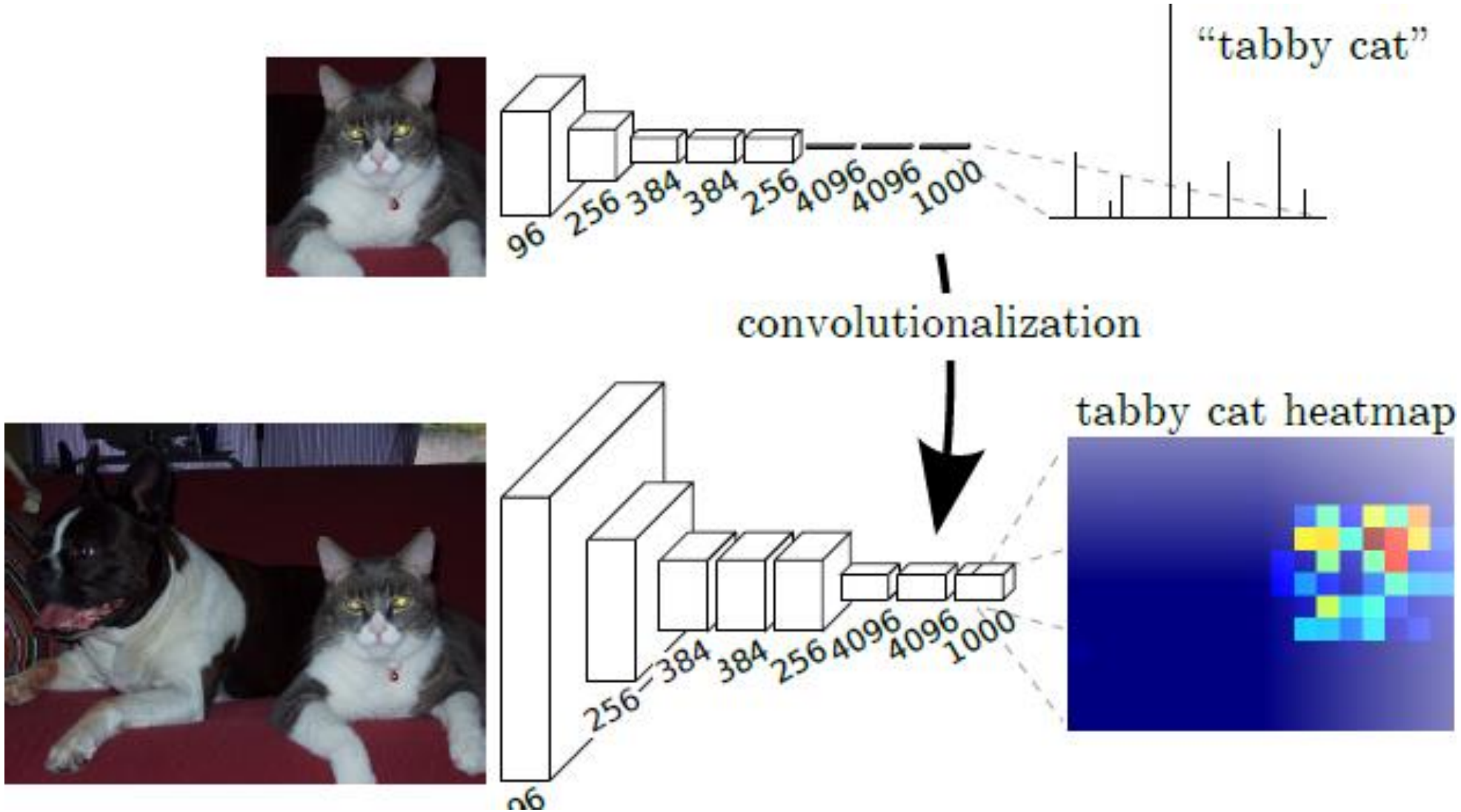
...

"Wheel" probability

Each pixel in the heatmap corresponds to a "*receptive field*" in the input image

G. Boracchi

Migration to FCNN of a pretrained model

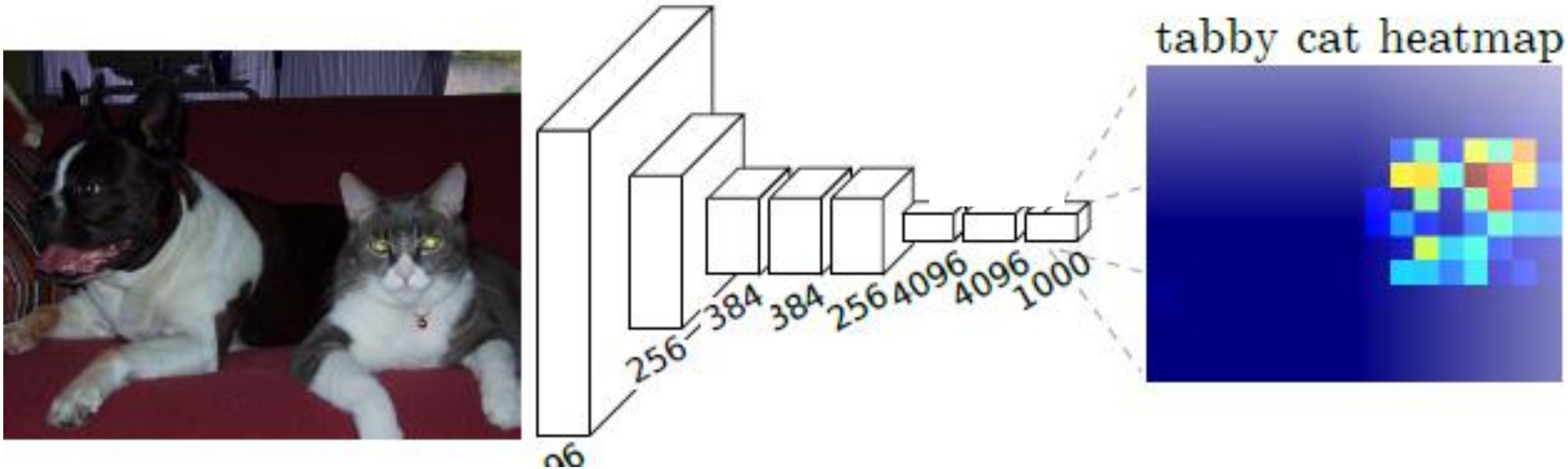


Long, Jonathan, *Evan* Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

Migration to FCNN of a pretrained model

This stack of convolutions operates on the whole image as a filter.

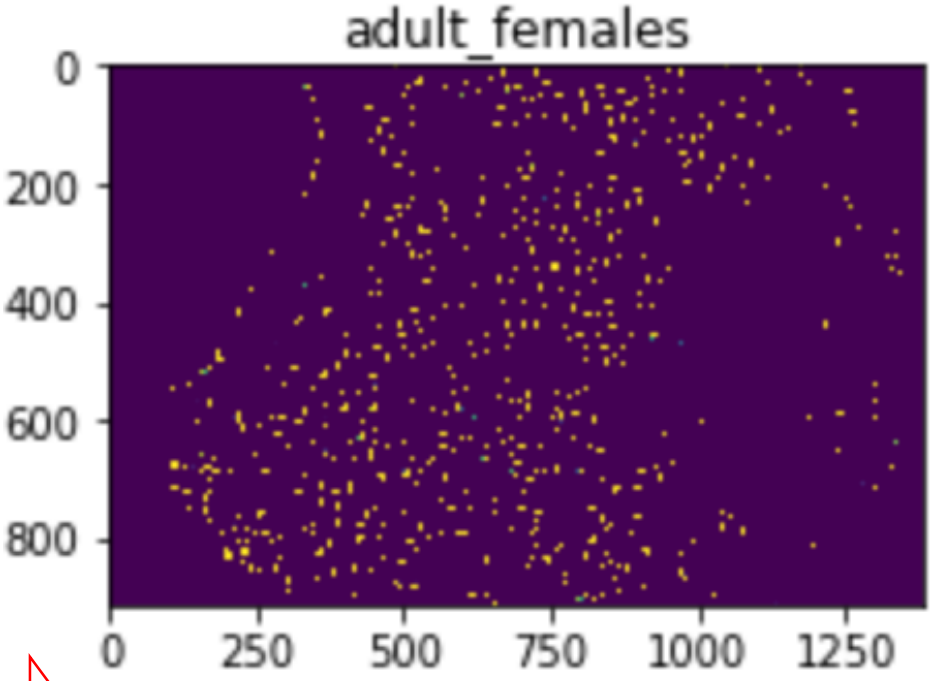
Significantly more efficient than patch extraction and classification: this avoids multiple repeated computations within overlapping patches

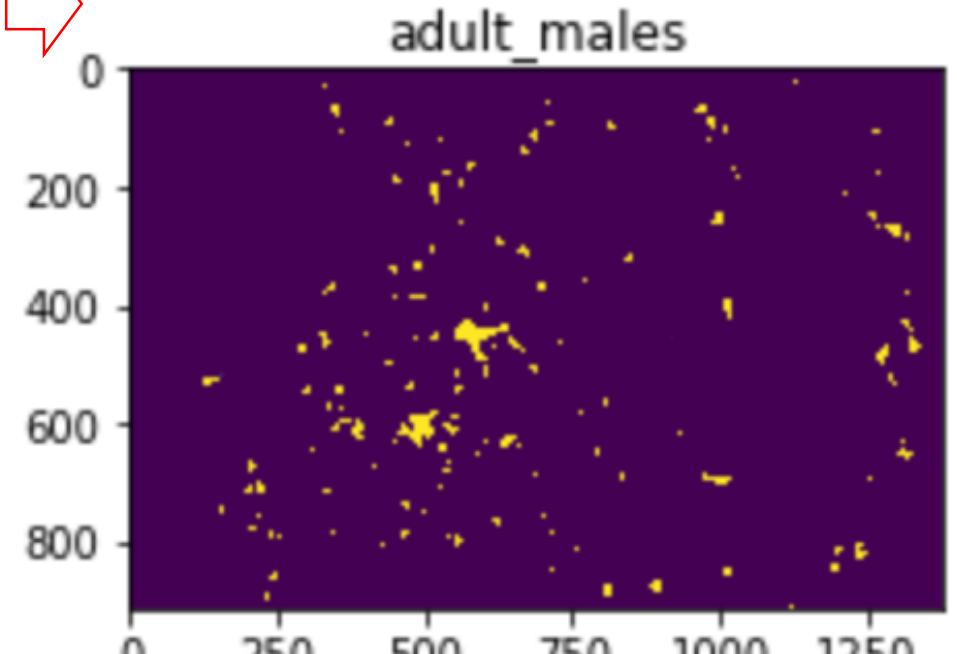


Long, Jonathan, *Evan* Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

Sealion HeatMaps

FCNN

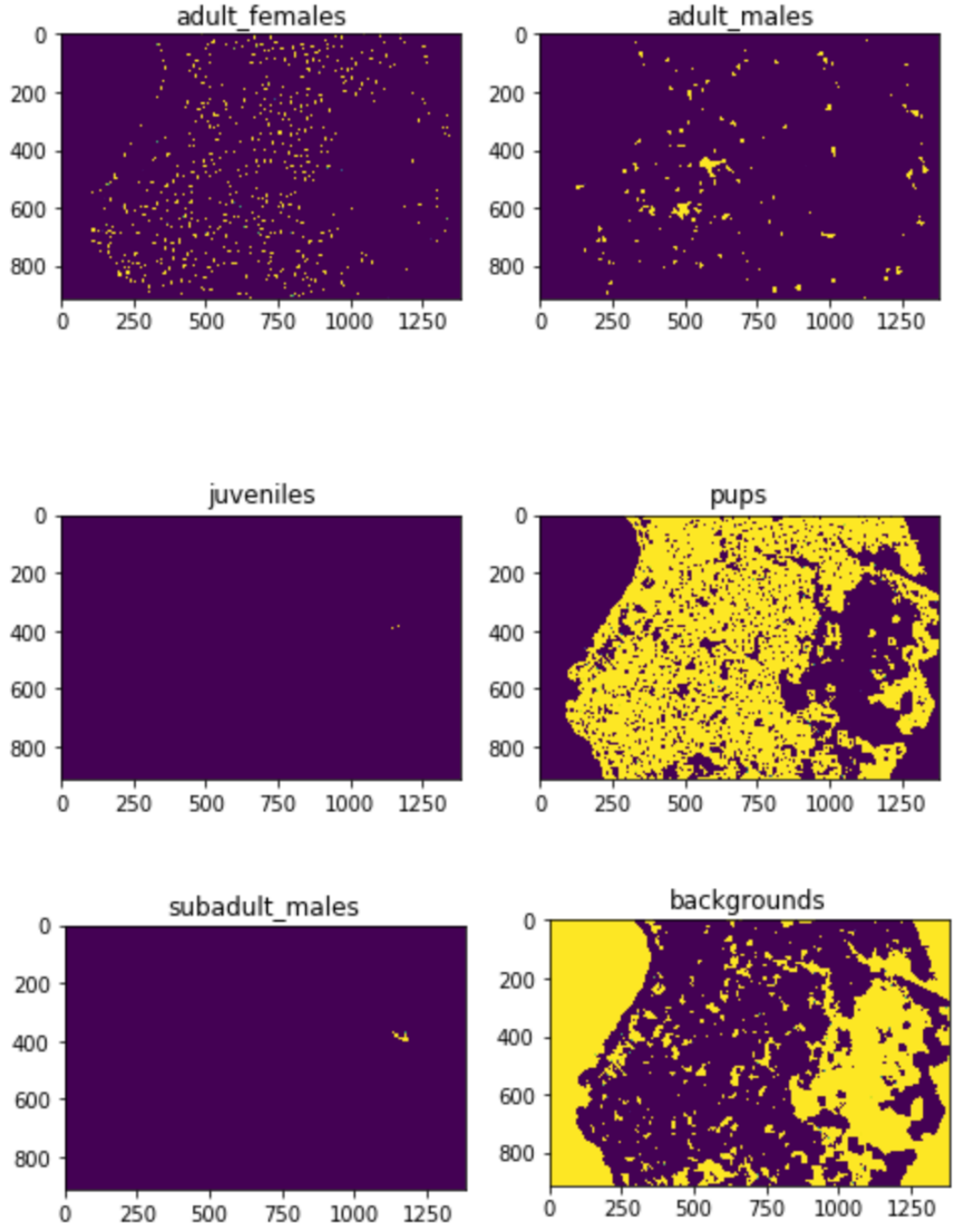




Credits Yinan Zhou
https://github.com/marioZYN/FC-CNN-Demo

Sealion HeatMaps

FCNN



Credits Yinan Zhou
https://github.com/marioZYN/FC-CNN-Demo

Fully convolutional networks in keras

It is necessary to get and set the weights of networks by means of the methods **`get_weights`** and **`set_weights`**

- get the weights of the trained CNN

```
w7, b7 = model.layers[7].get_weights()
```

- reshape these weights to become ten 1x1 convolutional filters having depth equal to 256 (these sizes are dictated by the sizes of the dense layer in the "traditional" CNN)

```
w7.reshape(1, 1, 256, 10)
```

- assign these weights to the FCNN architecture

```
model2.layers[7].set_weights(w7, b7)
```

Visual Recognition Problems

Setting up the stage..

G. Boracchi

Image Classification

$\Lambda = \{\text{"wheel"}, \text{"cars"}, \ldots, \text{"castle"}, \text{"baboon"}\}$

I

⇨ “wheel”

I

⇨ “castle”

Image Classification

$\Lambda = \{\text{"wheel"}, \text{"cars"}, ..., \text{"castle"}, \text{"baboon"}\}$

I

⇨ “wheel” 65%, “tyre” 30%..

I

⇨ “castle” 55%, “tower” 43%..

G. Boracchi

Image Classification, the problem

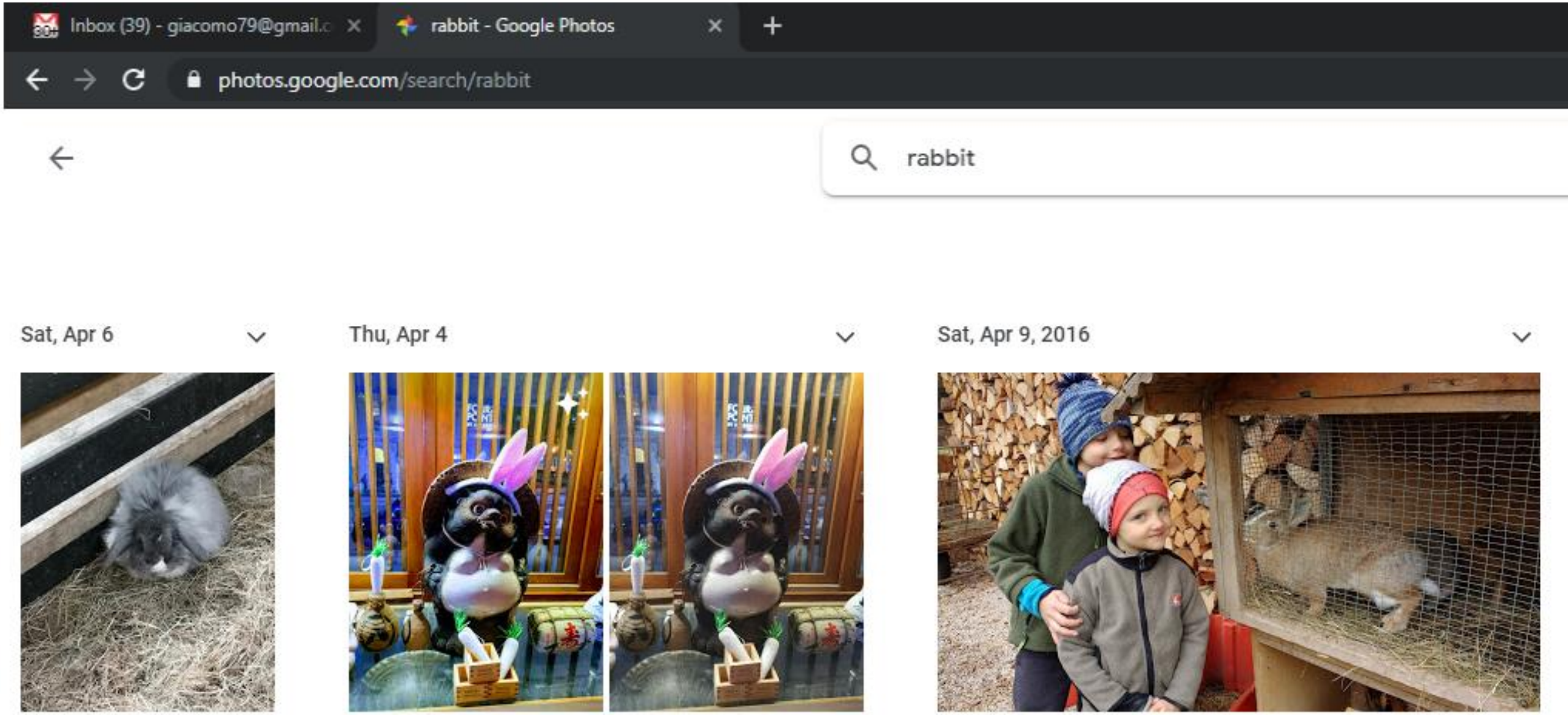
Assign to an input image $I \in \mathbb{R}^{R \times C \times 3}$:

- a label l from a fixed set of categories

$$\Lambda = \{\text{"wheel"}, \text{"cars"}, \ldots, \text{"castle"}, \text{"baboon"}\}$$

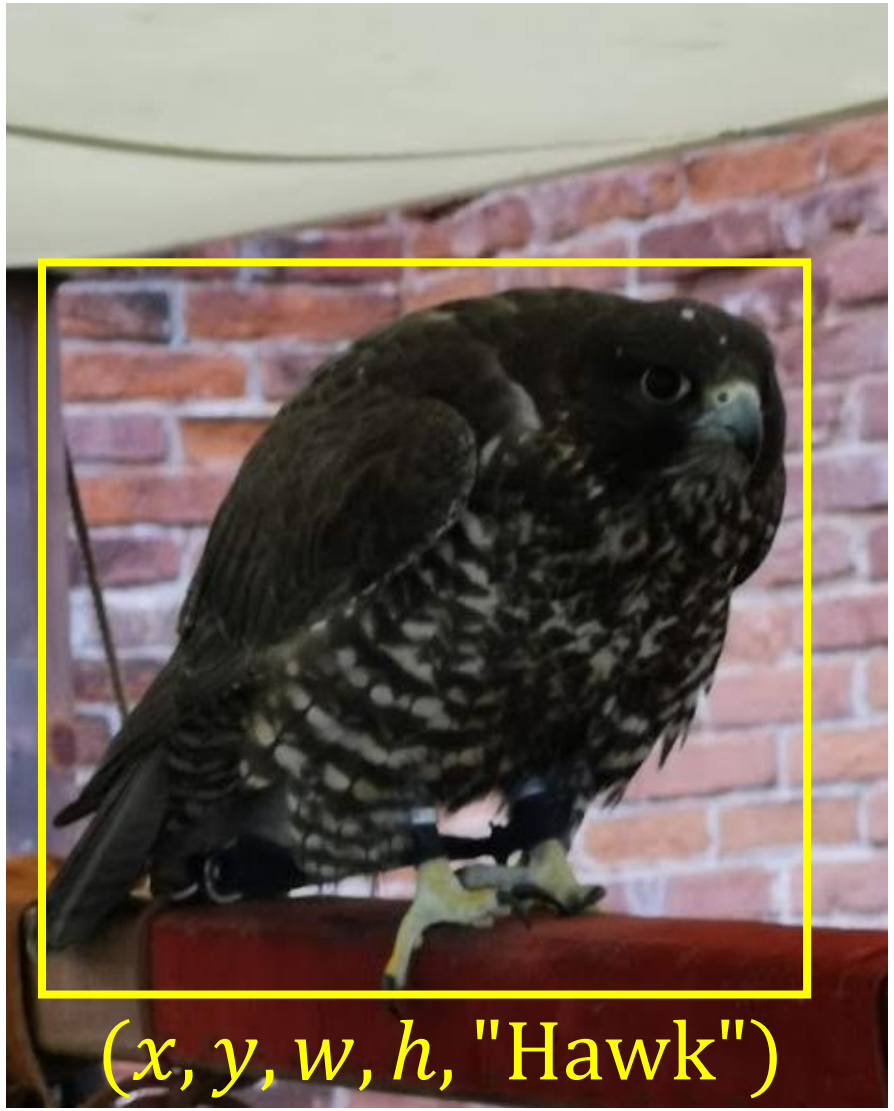
$$I \rightarrow l \in \Lambda$$

Image Classification Example



G. Boracchi

Localization



Localization, the problem

Assign to an input image $I \in \mathbb{R}^{R \times C \times 3}$:

- a label l from a fixed set of categories
$$\Lambda = \{\text{"wheel"}, \text{"cars"}, \ldots, \text{"castle"}, \text{"baboon"}\}$$
- the coordinates (x, y, h, w) of the bounding box enclosing that object

$$I \rightarrow (x, y, h, w, l)$$

Object Detection

G. Boracchi

Object Detection, the problem

Assign to an input image $I \in \mathbb{R}^{R \times C \times 3}$:

- **multiple** labels $\{l_i\}$ from a fixed set of categories $\Lambda = \{$"wheel", "cars", ..., "castle", "baboon"$\}$, each corresponding **to an instance of that object**
- the coordinates $\{(x, y, h, w)_i\}$ of the bounding box enclosing **each** object

$$I \rightarrow \{(x, y, h, w, l)_1, \dots, (x, y, h, w, l)_N\}$$

Semantic Segmentation

Zheng et al. “Conditional Random Fields as Recurrent Neural Networks”, ICCV 2015

Semantic Segmentation, the problem

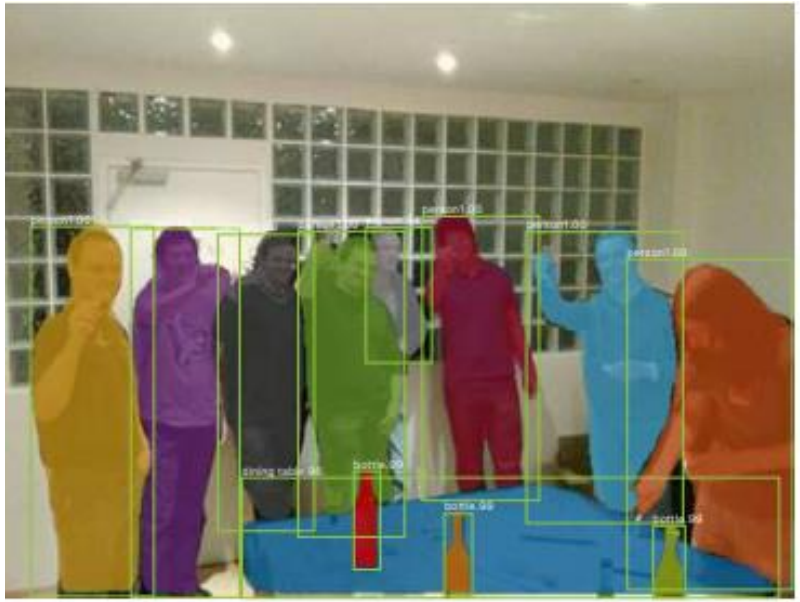
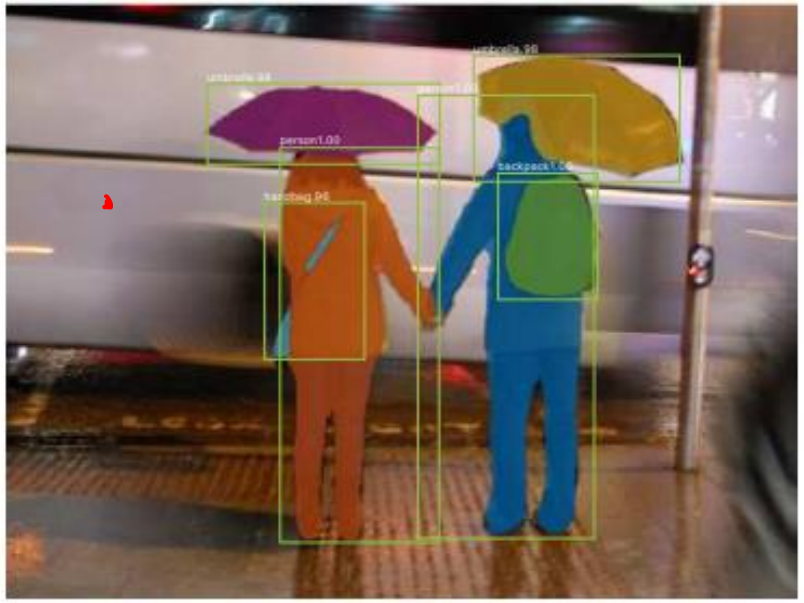
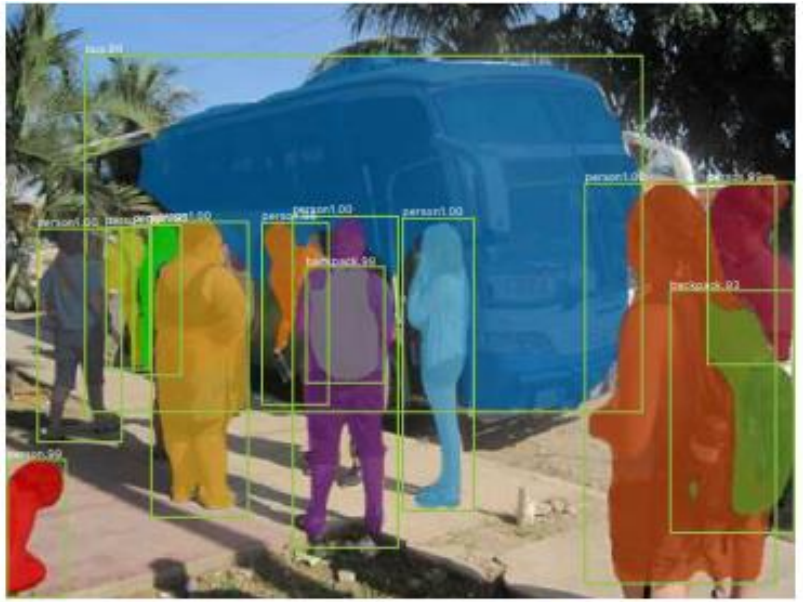
Assign **to each pixel** of an image $I \in \mathbb{R}^{R \times C \times 3}$:

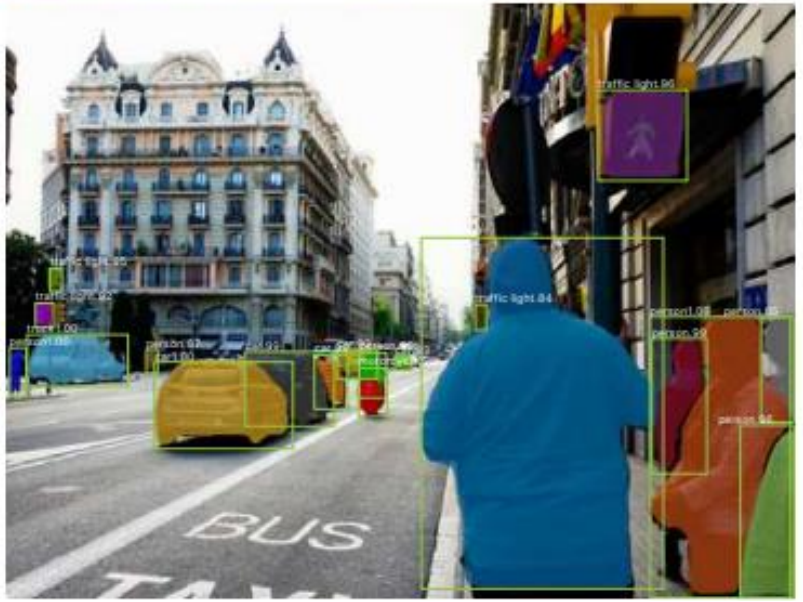
- a label $\{l_i\}$ from a fixed set of categories
  $\Lambda = \{\text{"wheel"}, \text{"cars"}, \ldots, \text{"castle"}, \text{"baboon"}\}$,

$$I \rightarrow S \in \Lambda^{R \times C}$$

where $S(x, y) \in \Lambda$ denotes the class associated to the pixel (x, y)

Instance Segmentation: Mask R-CNN

He, Kaiming, et al. "Mask r-cnn." *CVPR 2017*

Instance Segmentation, the problem

Assign to an input image I:

- **multiple** labels $\{l_i\}$ from a fixed set of categories $\Lambda = \{$"wheel", "cars", ..., "castle", "baboon"$\}$, each corresponding to **an instance of that object**
- the coordinates $\{(x, y, h, w)_i\}$ of the **bounding box enclosing each object**
- the **set of pixels** S in each bounding box corresponding to that label

$$I \rightarrow \{(x, y, h, w, l, S)_1, \ldots, (x, y, h, w, l, S)_N\}$$

Instance Segmentation

G. Boracchi

Semantic Segmentation by Fully Convolutional Neural Networks

Predicting dense outputs for arbitrary-sized inputs

G. Boracchi

Semantic Segmentation Task

The goal of semantic segmentation is:

Given an image I, **associate to each pixel** $(\boldsymbol{r}, \boldsymbol{c})$ **a label** from Λ.

The **result of segmentation is a map of labels** containing in each pixel the estimated class.

Remark: In this image there is no distinction among persons.
Segmentation **does not separate different instances belonging to the same class**.
That would be instance segmentation.

http://www.robots.ox.ac.uk/~szheng/crfasrnndemo

Training Set

The training set is made of pairs (I, GT), where the GT is a pixel-wise annotated image over the categories in Λ

I

GT

http://cocodataset.org/

The Training Set

COCO
Common Objects in Context

COCO Explorer

COCO 2017 train/val browser (123,287 images, 886,284 instances). Crowd labels not shown.

Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." ECCV 2014.

The Training Set $TR = \{(I, GT)_i\}$

I is the input image
GT is the annotation
(shown in overlays)

Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." ECCV 2014.

The Training Set $TR = \{(I, GT)_i\}$

I is the input image
GT is the annotation
(shown in overlays)

Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." ECCV 2014.

The Training Set $TR = \{(I, GT)_i\}$

I is the input image
GT is the annotation
(shown in overlays)

Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." ECCV 2014.

The Training Set

COCO
Common Objects in Context

COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- **Object segmentation**
- **Recognition in context**
- **Superpixel stuff segmentation**
- **330K images (>200K labeled)**
- **1.5 million object instances**
- **80 object categories**
- **91 stuff categories**
- **5 captions per image**
- **250,000 people with keypoints**

Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." ECCV 2014.

The Training Set

COCO
Common Objects in Context

COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- Object segmentation
- text
- gmentation
- K labeled)
- nstances
- es
- ge
- h keypoints

Training set very difficult to annotate ☺

Microsoft COCO dataset provides 200.000 annoated images over 80 categories

(humans have joints annotated as well)

Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." ECCV 2014.

Fully Convolutional Networks for Semantic Segmentation

Jonathan Long* Evan Shelhamer* Trevor Darrell
UC Berkeley
{jonlong,shelhamer,trevor}@cs.berkeley.edu

G. Boracchi

From Classification to Segmentation

Setup:

- You are given a pre-trained CNN for classification
- You possibly performed transfer learning to the problem at hand
- You have «convolutionalized» it to extract heatmaps

Limitation:

- Heatmaps are very low-resolution

Goal:

- Obtain a Semantic Segmentation network for images of arbitrary size

Simple Solution (1): Direct Heatmap Predictions

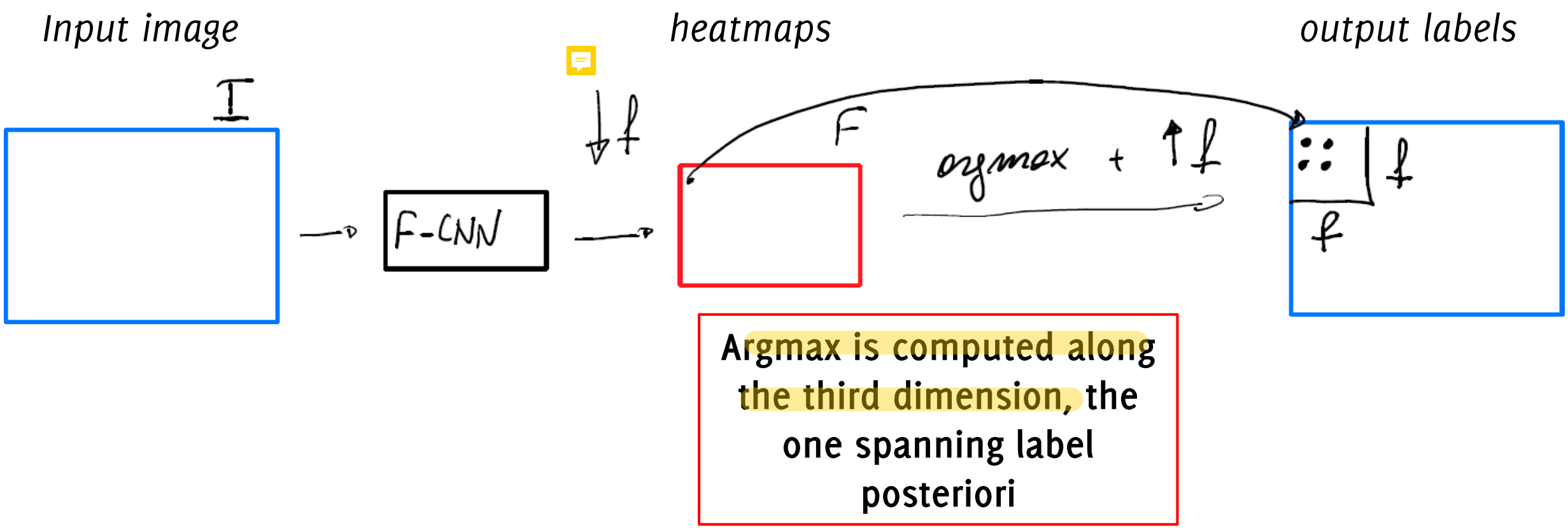
We can assign the predicted label in the heatmap to the whole receptive field, however that would be a **very coarse estimate**

We need to upsample the estimated map!

Long, Jonathan, *Evan* Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

Simple Solution (1): Direct Heatmap Upsampling

Very coarse estimates



Argmax is computed along the third dimension, the one spanning label posteriori

Simple Solution (2): The Shift and Stich

Shift and Stich: Assume there is a downsampling ratio f between the size of input and of the output heatmap

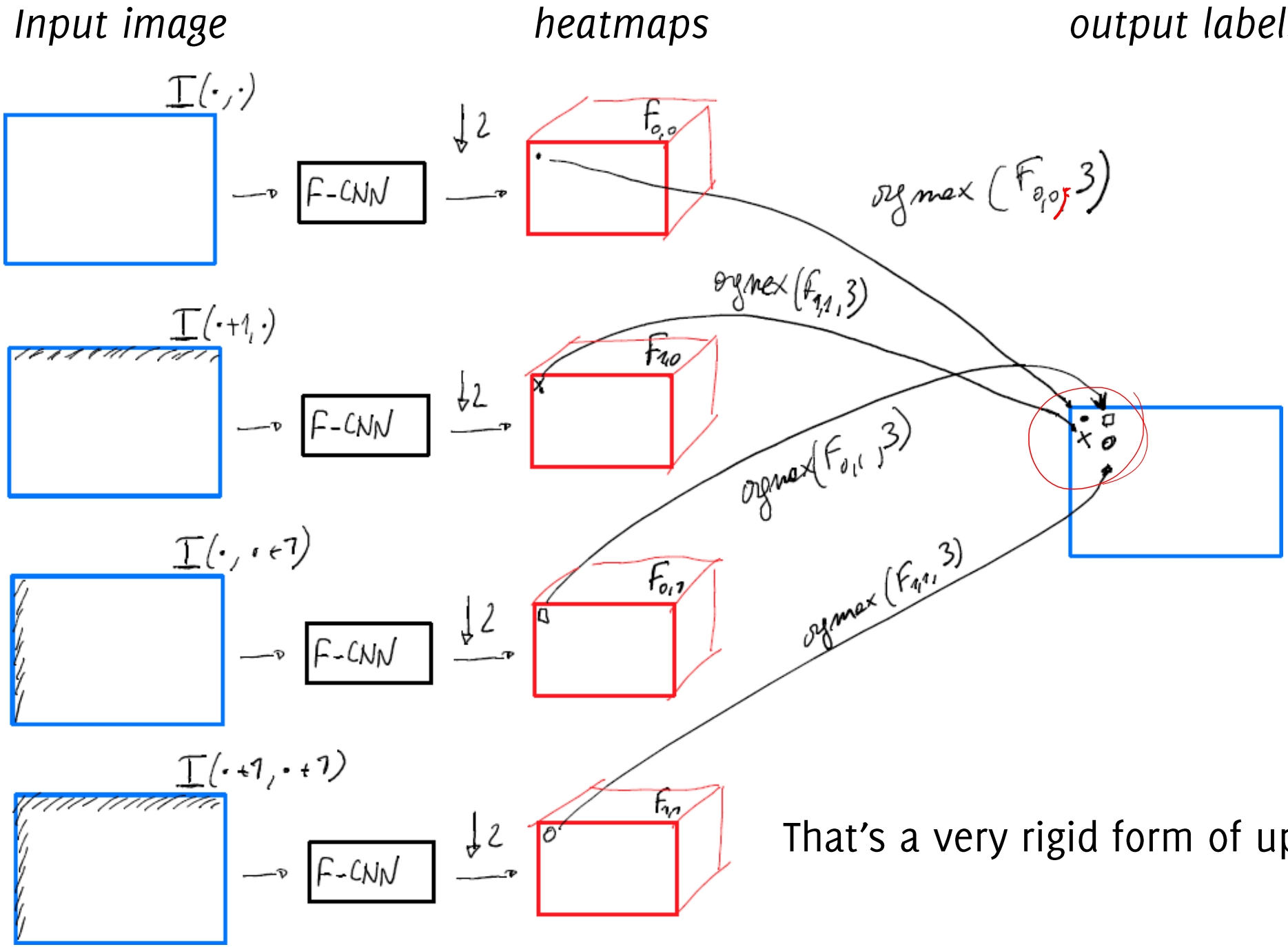
- Compute **heatmaps** for all $\boldsymbol{f^2}$ possible shifts of the input $(0 \leq r, c < f)$
- **Map predictions from the $\boldsymbol{f^2}$** heatmaps to the image: each pixel in the heatmap **provides prediction of the central pixel of the receptive field**
- **Interleave the heatmaps** to form an image as large as the input

This **exploits the whole depth of the network**

Efficient implementation through the à trous algorithm in wavelet

However, **the upsampling method is very rigid**

Long, Jonathan, *Evan* Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015



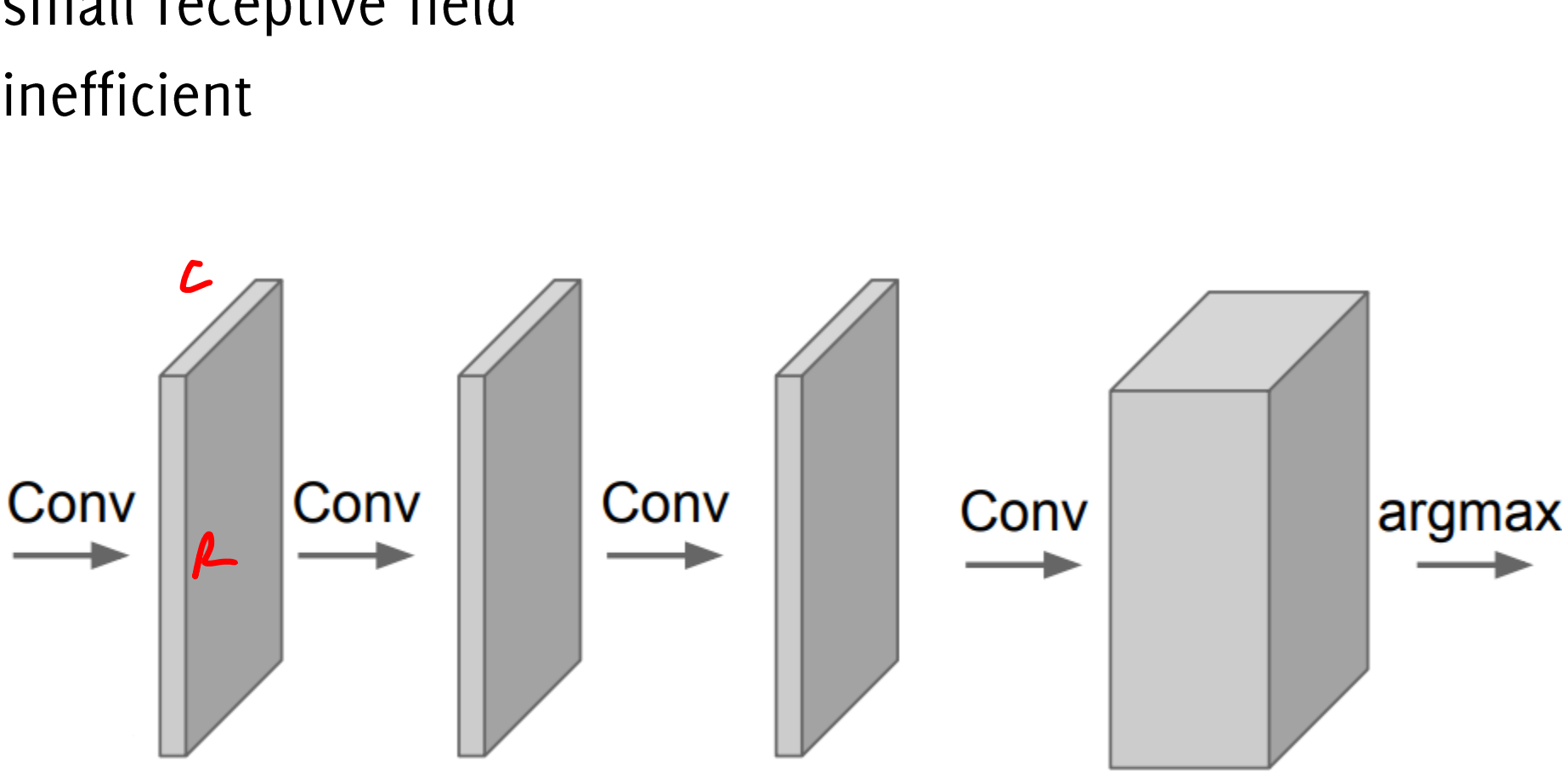
That's a very rigid form of upsamling!

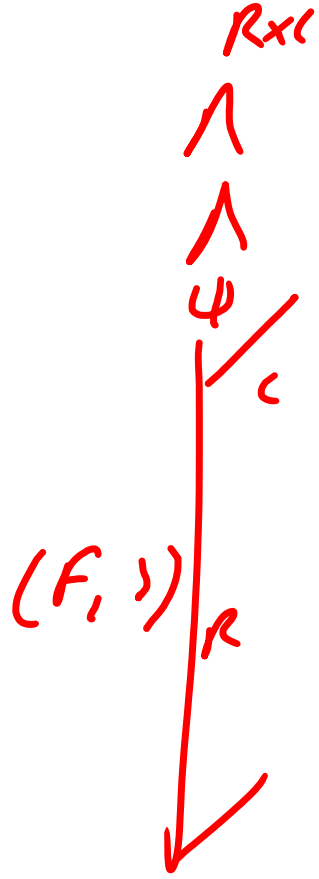
G. Boracchi

Simple Solution (3): Only Convolutions

What if we avoid any pooling (just conv2d and activation layers)?

- Very small receptive field
- Very inefficient





Drawbacks of convolutions only

On the one hand **we need to "go deep" to extract high level information** on the image

On the other hand **we want to stay local not to loose spatial resolution** in the predictions

Semantic segmentation faces an inherent tension between semantics and location:

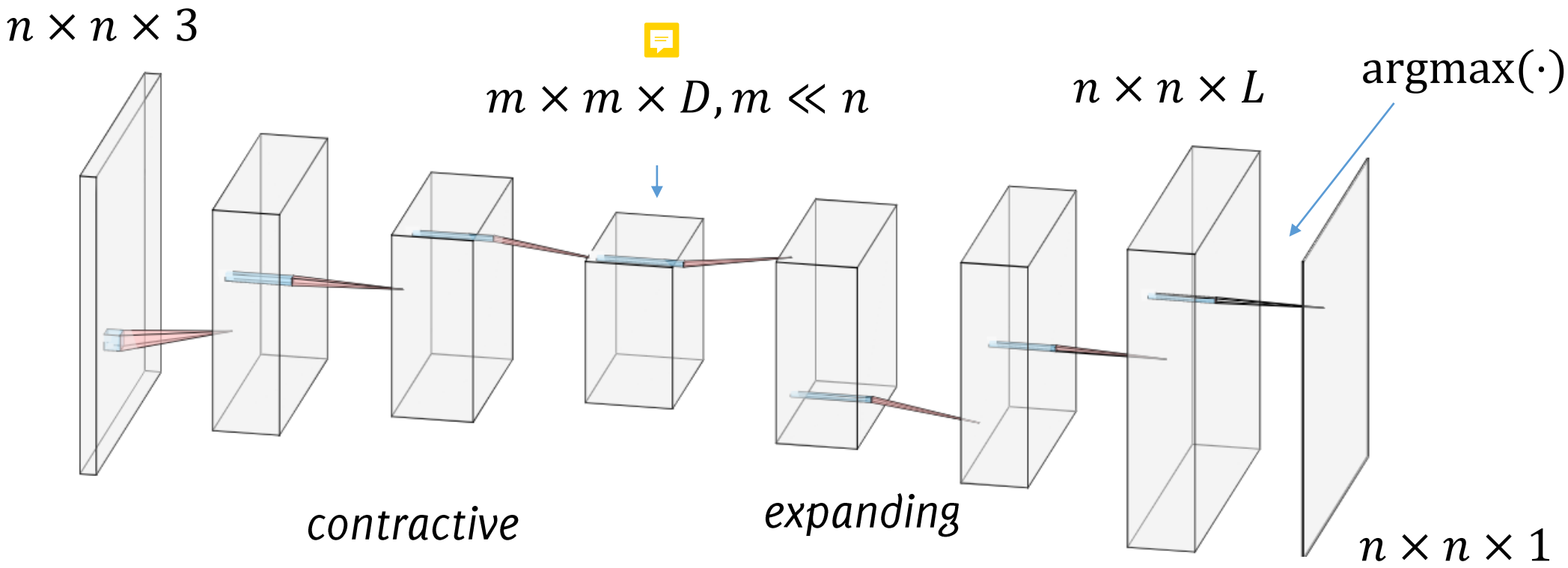
- ***global information resolves what**, while*
- ***local information resolves where***

***Combining fine layers and coarse layers** lets the model make local predictions that respect global structure.*

Long, Jonathan, *Evan* Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

Low-dimensional representation and upsampling

An architecture like the following would probably be more suitable for semantic segmentation

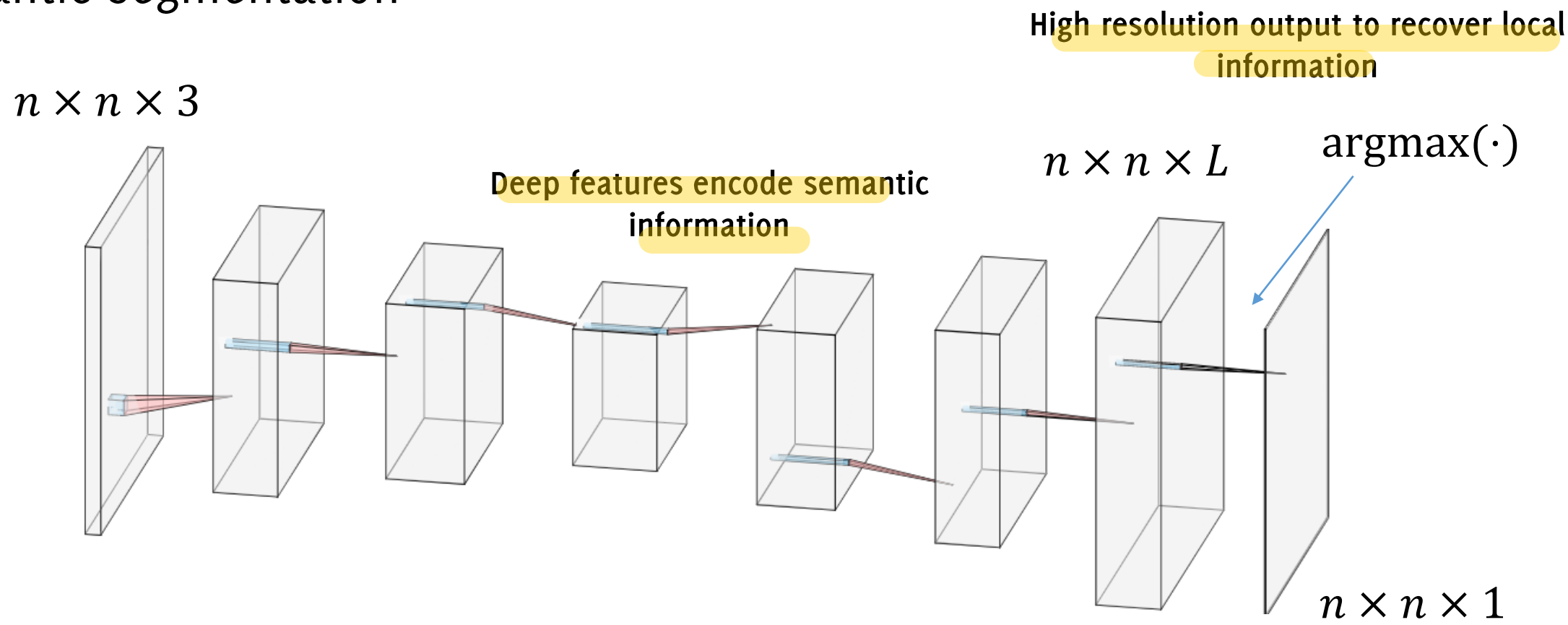


Conv + pooling / downsampling

Conv + upsampling

Low-dimensional representation and upsampling

An architecture like the following would probably be more suitable for semantic segmentation



Conv + pooling / downsampling

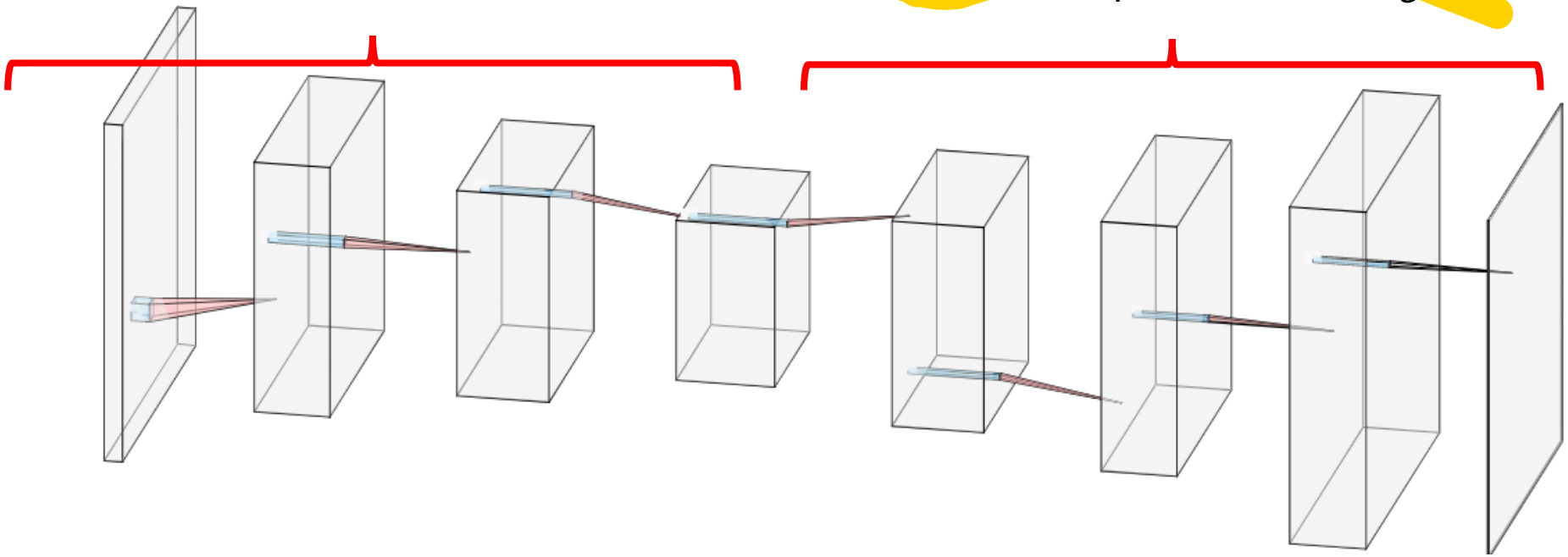
Conv + upsampling

Low-dimensional representation and upsampling

An architecture like the following would probably be more suitable for semantic segmentation

The first half is the same of a classification network

The second half is meant to upsample the predictions to cover each pixel in the image



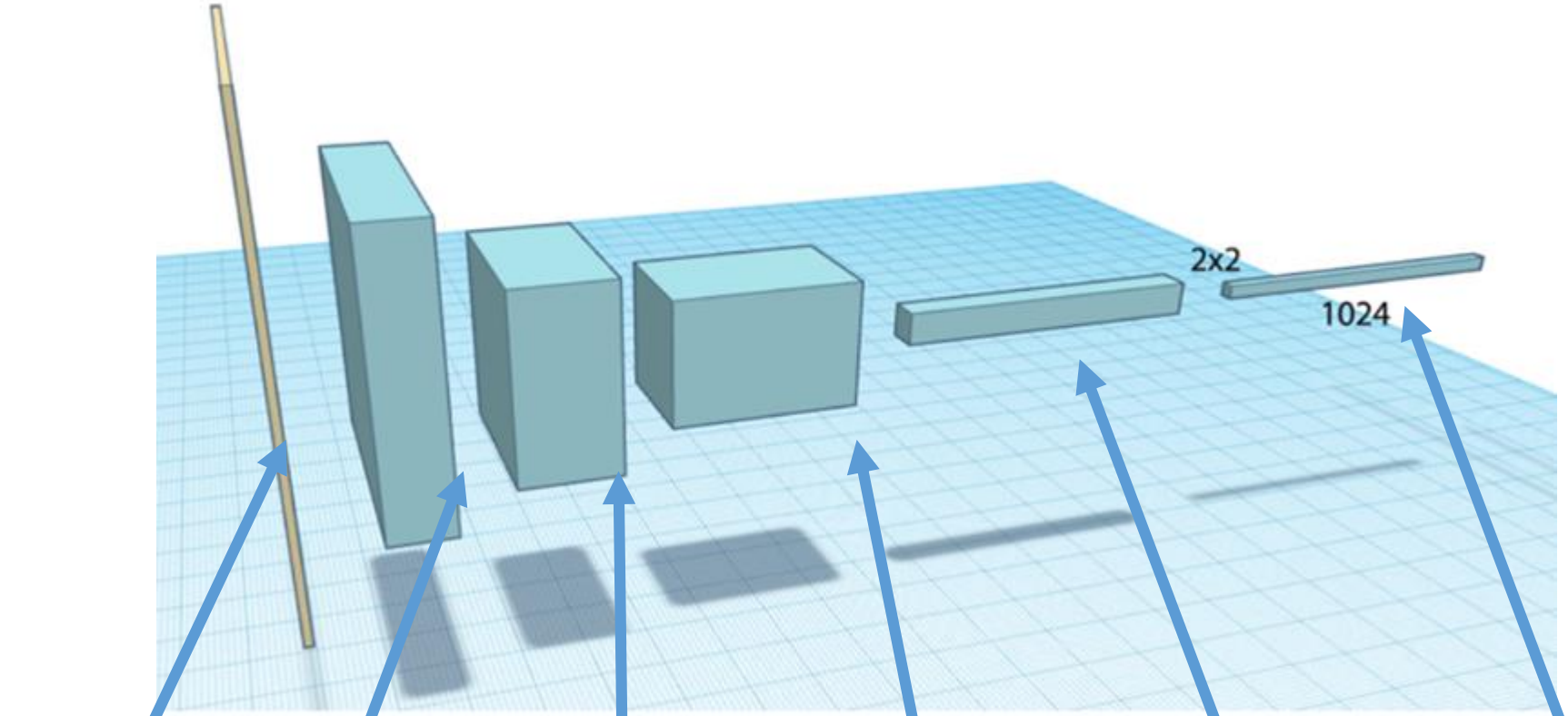
Increasing the image size is necessary to obtain sharp contours and spatially detailed class predictions

Conv + pooling / downsampling

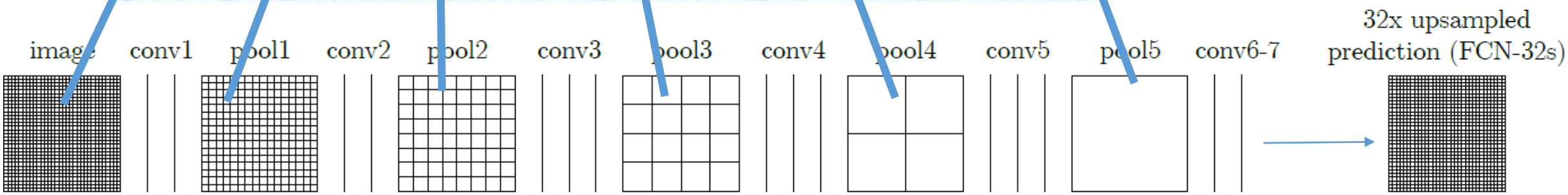
Conv + upsampling

G. Boracchi

Low-dimensional representation and upsampling

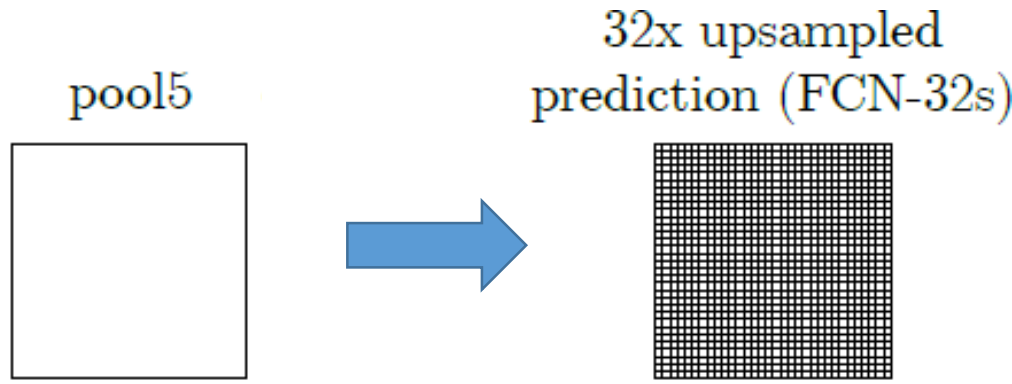


Upsampling



Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

How to perform upsampling?



Nearest Neighbor

1	2
3	4

→

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

Input: 2 x 2 Output: 4 x 4

"Bed of Nails"

1	2
3	4

→

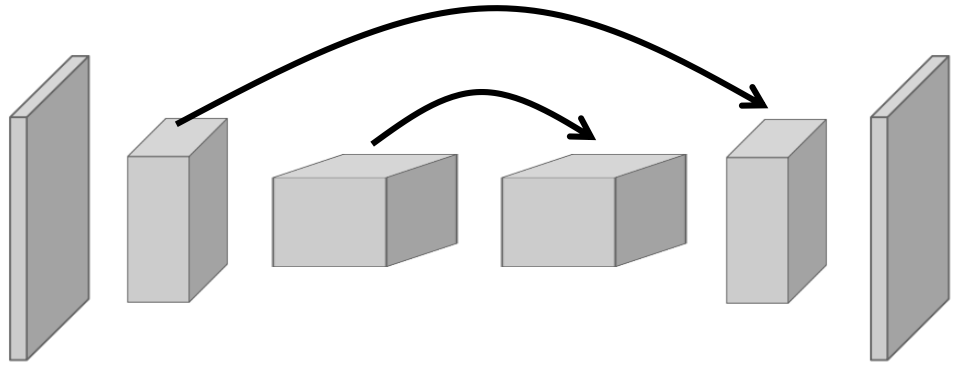
1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Input: 2 x 2 Output: 4 x 4

CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Max Unpooling

You have to keep track of the locations of the max during maxpooling



Max Pooling
Remember which element was max!

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

Input: 4 x 4

5	6
7	8

Output: 2 x 2

Rest of the network

Max Unpooling
Use positions from pooling layer

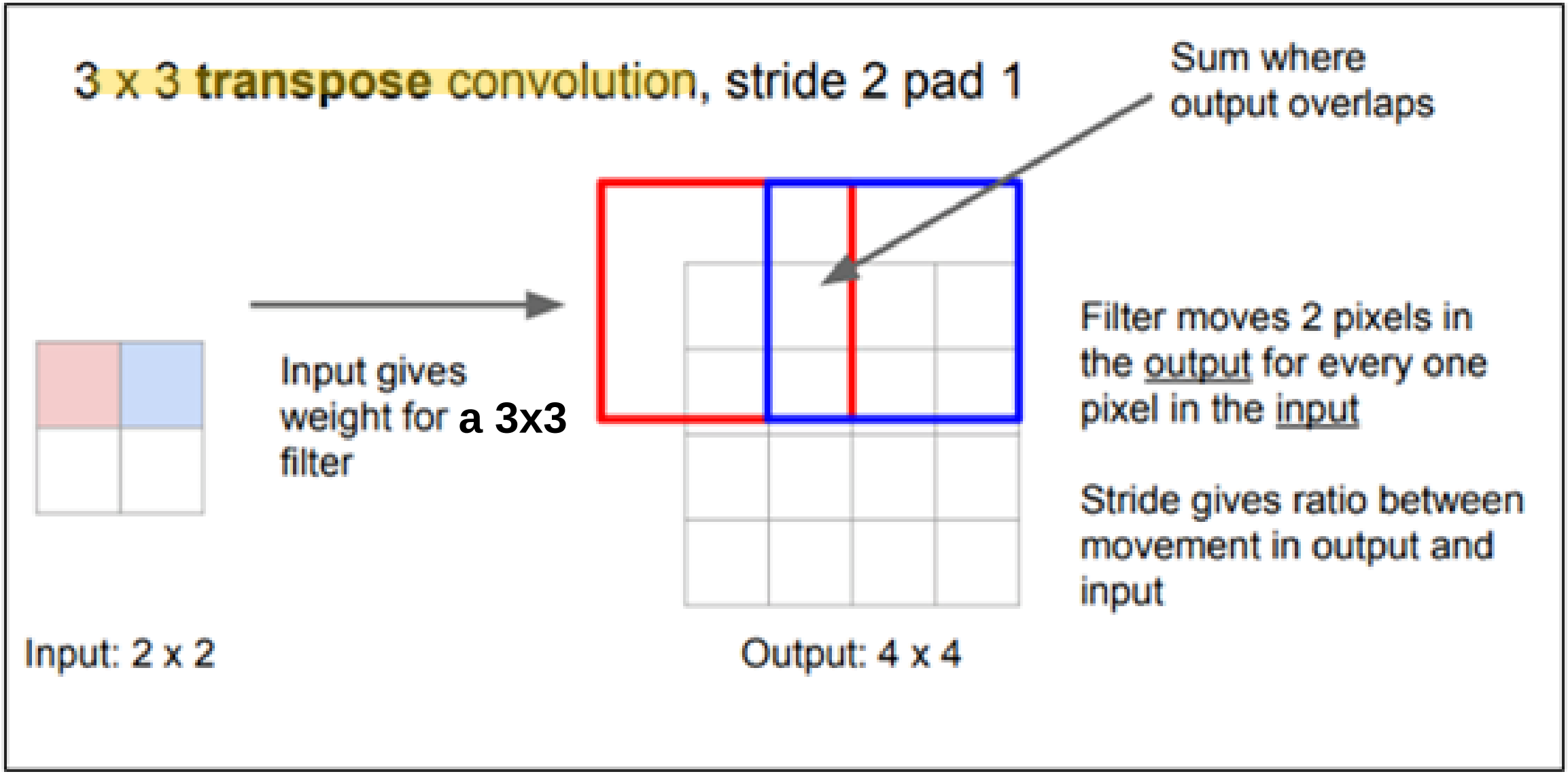
1	2
3	4

Input: 2 x 2

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

Output: 4 x 4

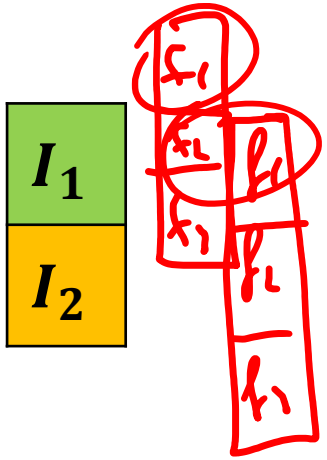
Transpose Convolution



Transpose Convolution

Transpose convolution with stride 1

Input 2 x 1



Filter 3 x 1

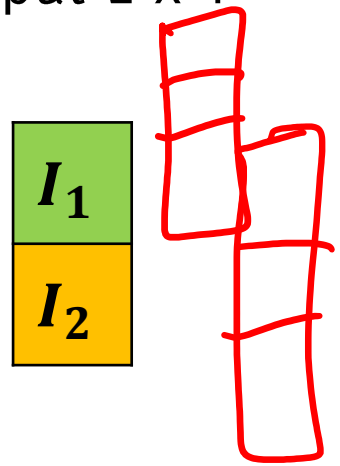
f_1
f_2
f_3

$f_1 I_1$		
$f_2 I_1$	+	$f_1 I_2$
$f_3 I_1$	+	$f_2 I_2$
		$f_3 I_2$

Transpose Convolution

Transpose convolution with stride 2

Input 2 x 1



Filter 3 x 1

f_1
f_2
f_3

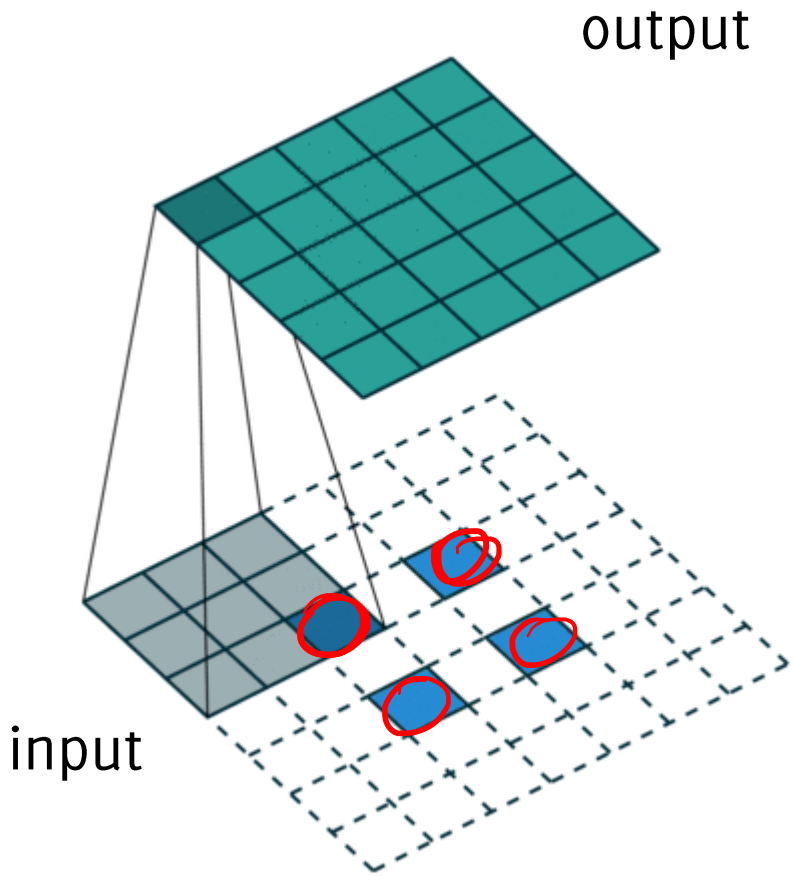
$f_1 I_1$		
$f_2 I_1$		
$f_3 I_1$	+	$f_1 I_2$
		$f_2 I_2$
		$f_3 I_2$

How to perform upsampling?

Transpose Convolution can be seen as a traditional convolution after having upsampled the input image

Many names for transpose convolution: fractional strided convolution, backward strided convolution, deconvolution (very misleading!!!)

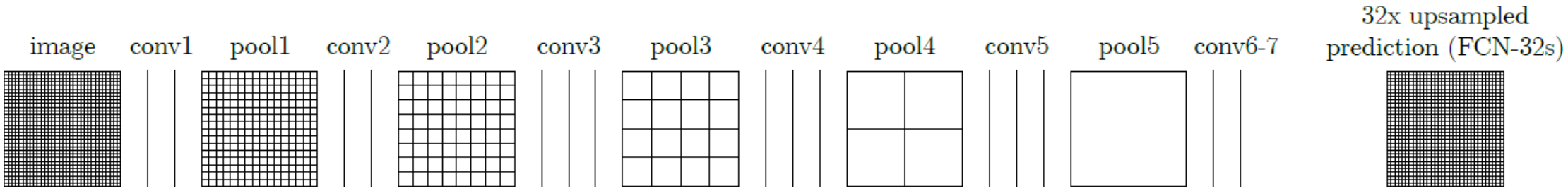
Upsamping based on convolution gives more degrees of freedom, since the **filters can be learned**!



https://github.com/vdumoulin/conv_arithmetic

Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

FC-CNN Upsampling



deconvolution layer to bilinearly upsample the coarse outputs to pixel-dense outputs

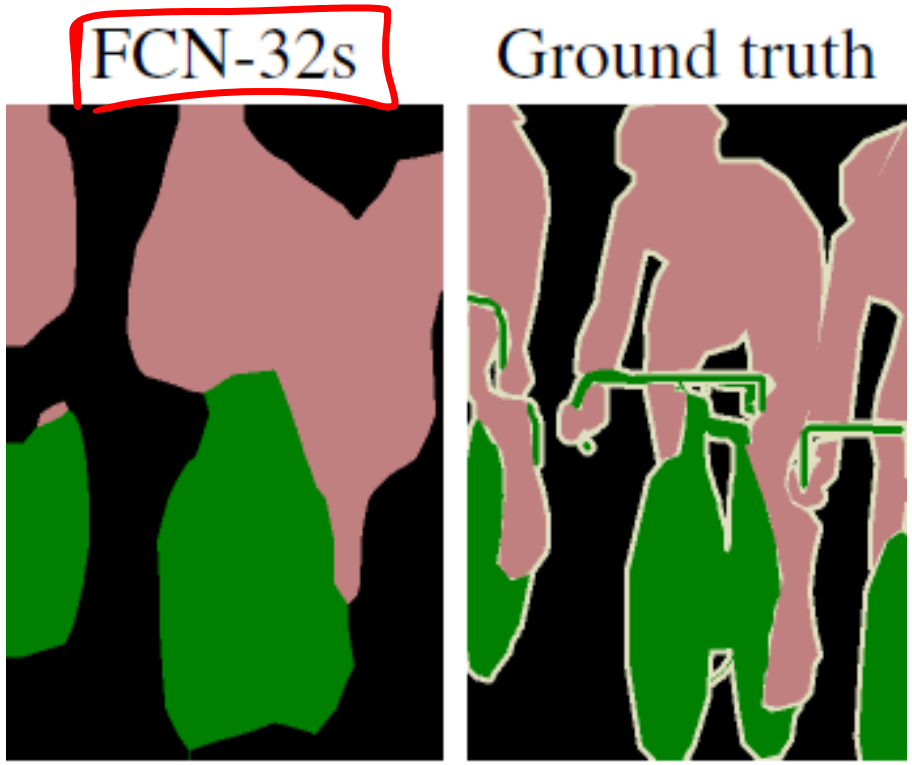
Prediction Upsampling

Linear upsampling of a factor f can be implemented as a convolution against a filter with a fractional stride $1/f$.

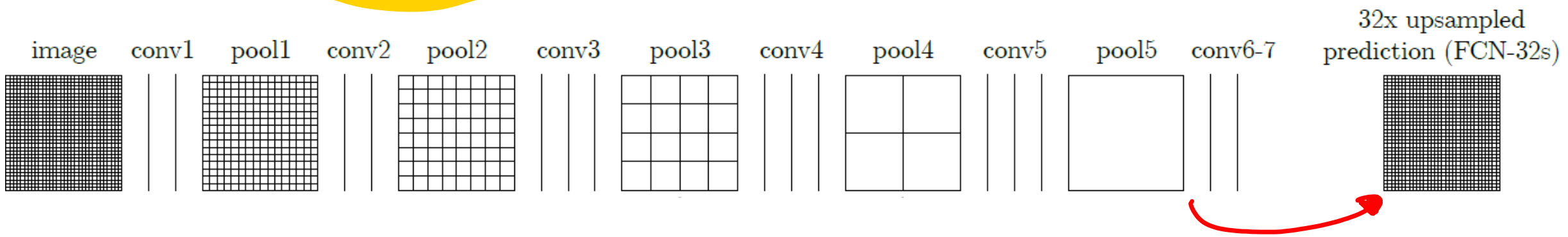
Upsampling filters can thus be learned during network training.

These **predictions** however are **very coarse**

Upsampling filters are learned with initialization equal to the bilinear interpolation

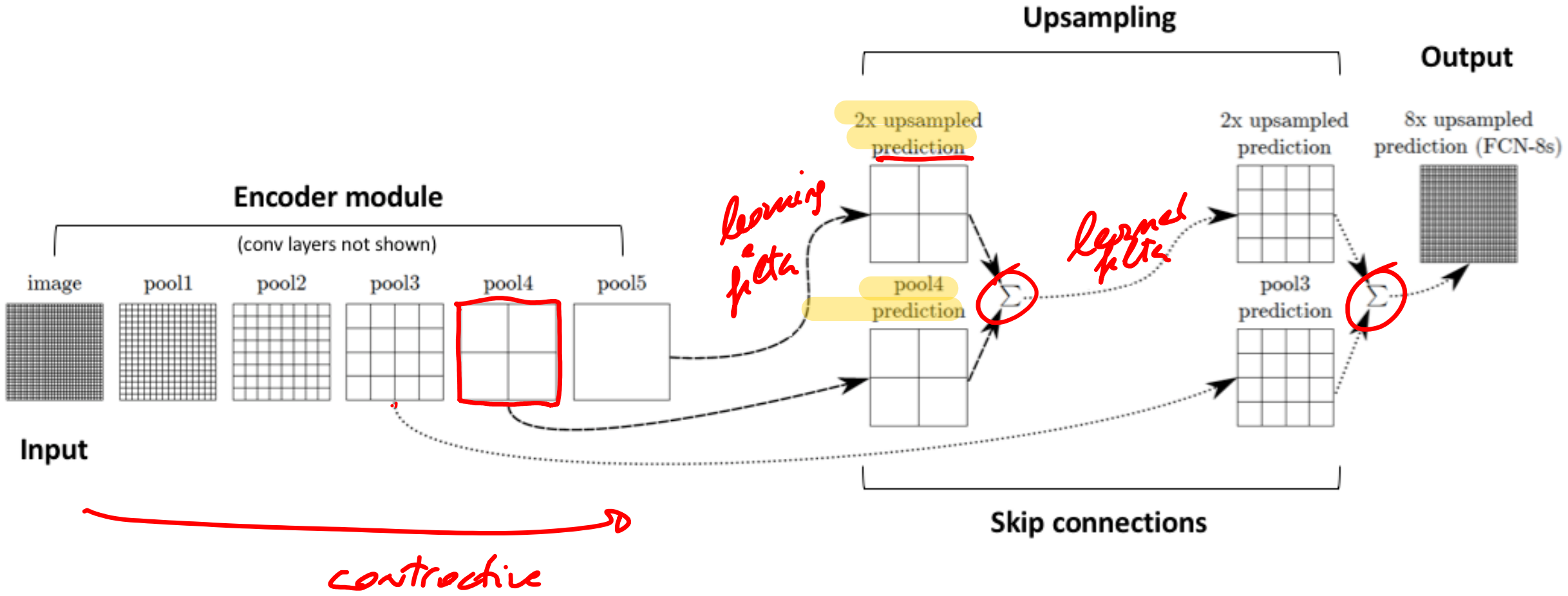


Upsampling filters



Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

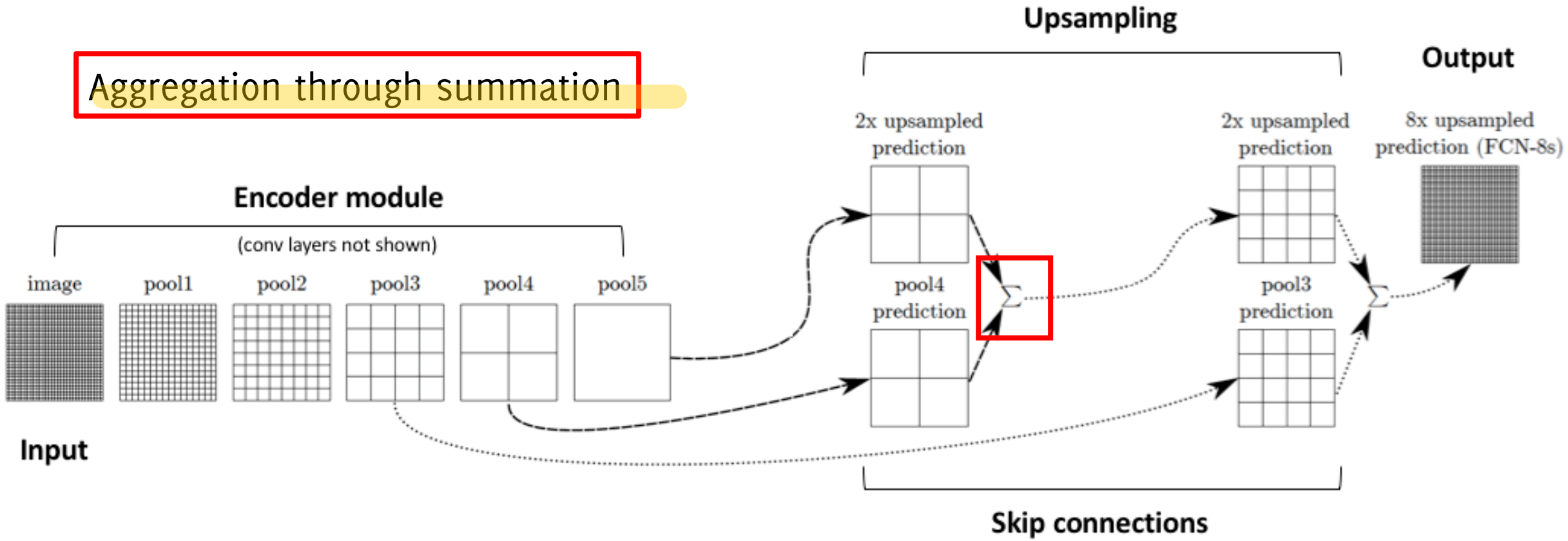
Improve segmentation: Skip Connections!



https://www.jeremyjordan.me/semantic-segmentation/

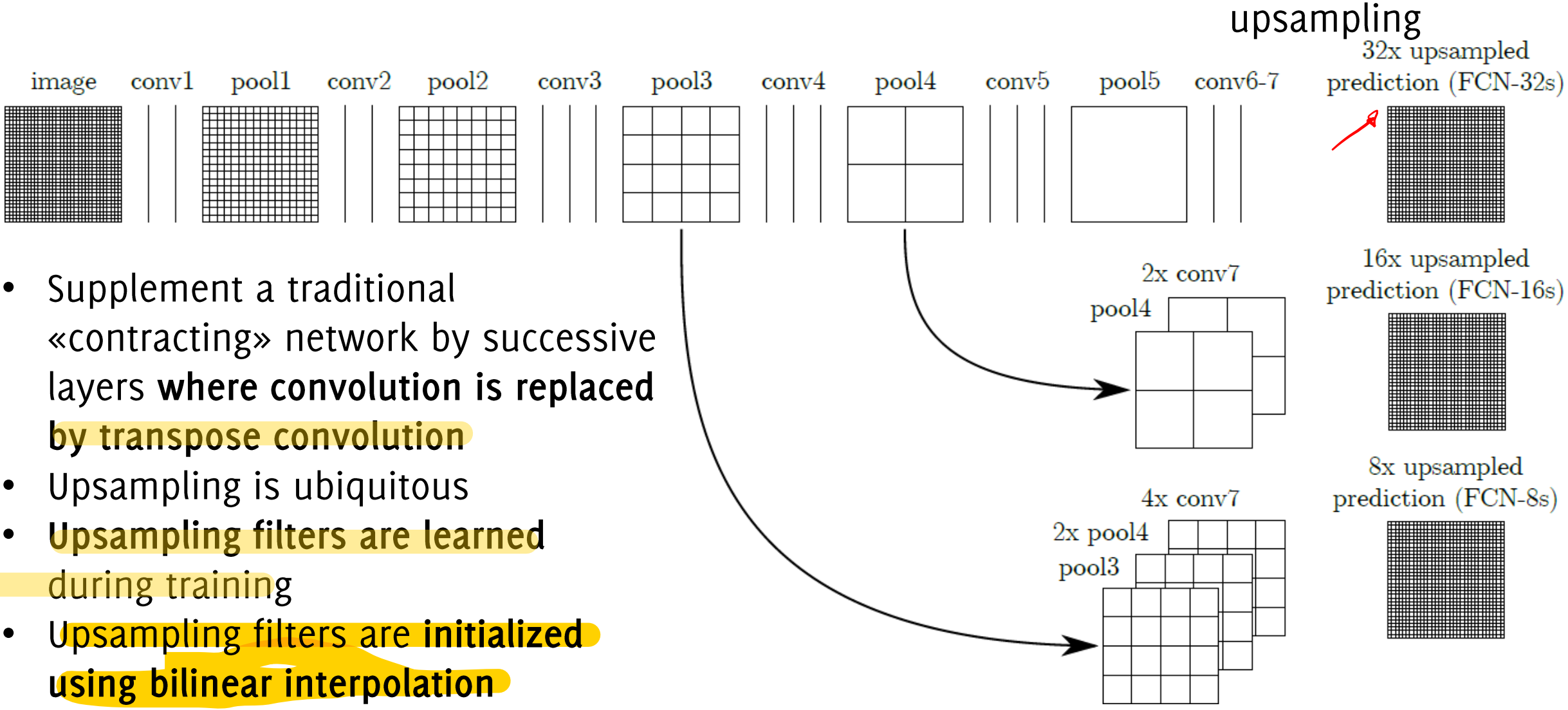
Solution: Skip Connections!

Aggregation through summation



https://www.jeremyjordan.me/semantic-segmentation/

Solution: Skip Connections!

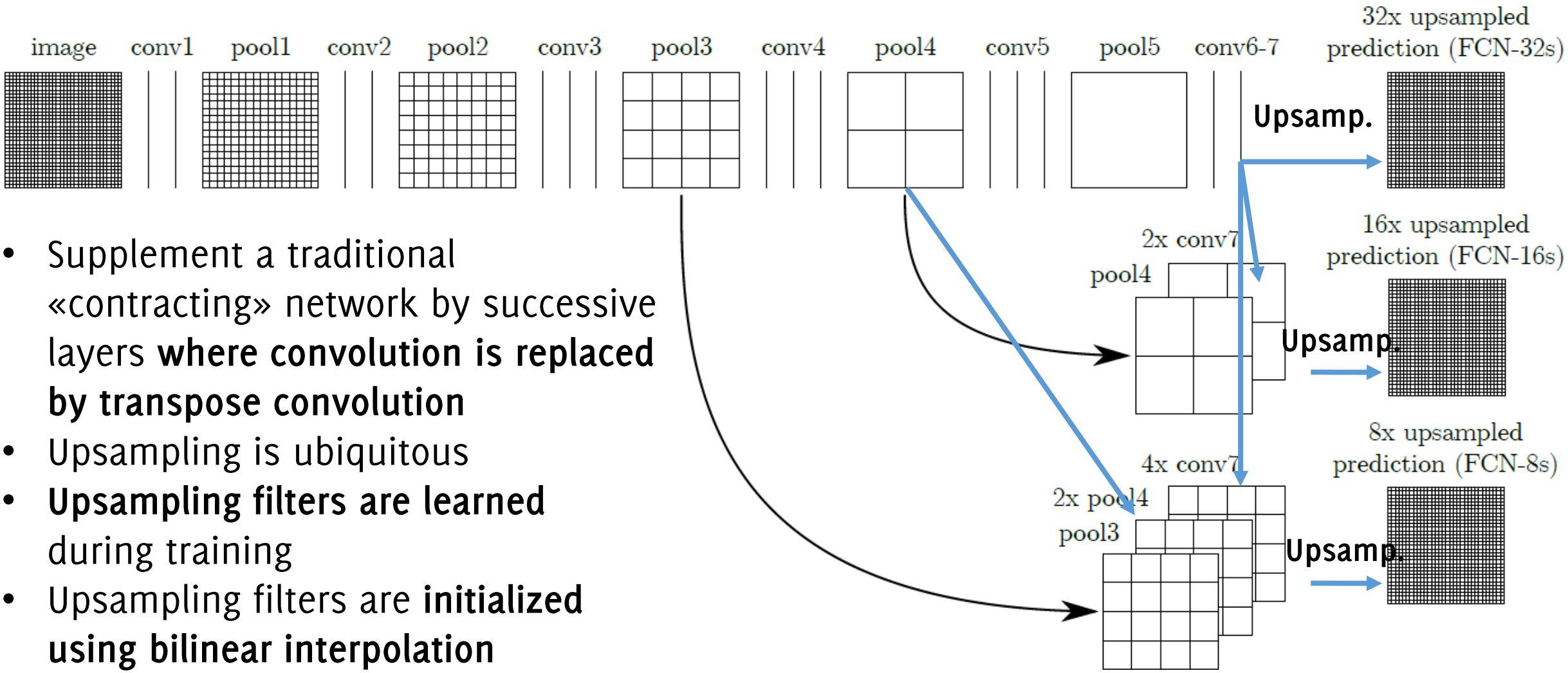


- Supplement a traditional «contracting» network by successive layers **where convolution is replaced by transpose convolution**
- Upsampling is ubiquitous
- **Upsampling filters are learned** during training
- Upsampling filters are **initialized using bilinear interpolation**

Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

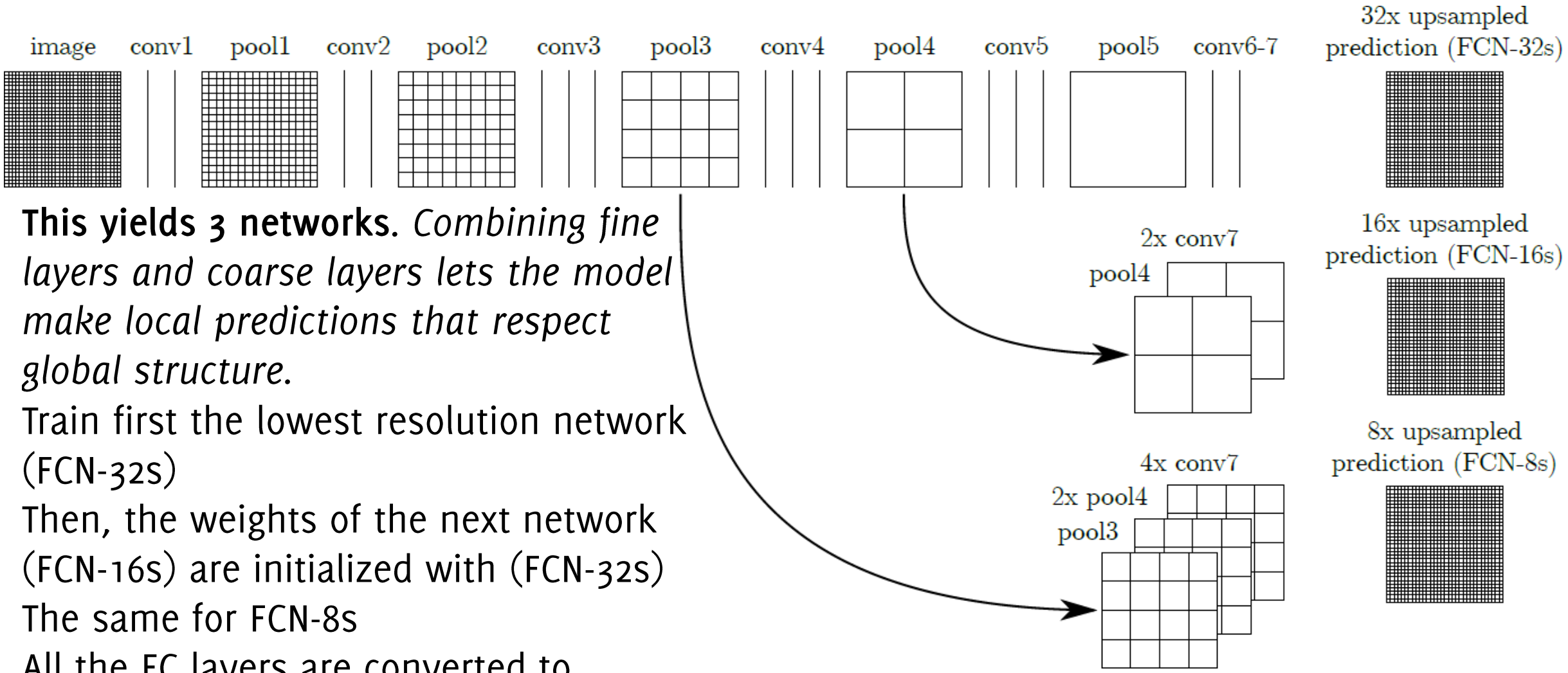
Solution: Skip Connections!

- Supplement a traditional «contracting» network by successive layers **where convolution is replaced by transpose convolution**
- Upsampling is ubiquitous
- **Upsampling filters are learned** during training
- Upsampling filters are **initialized using bilinear interpolation**



Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

Solution: Skip Connections!

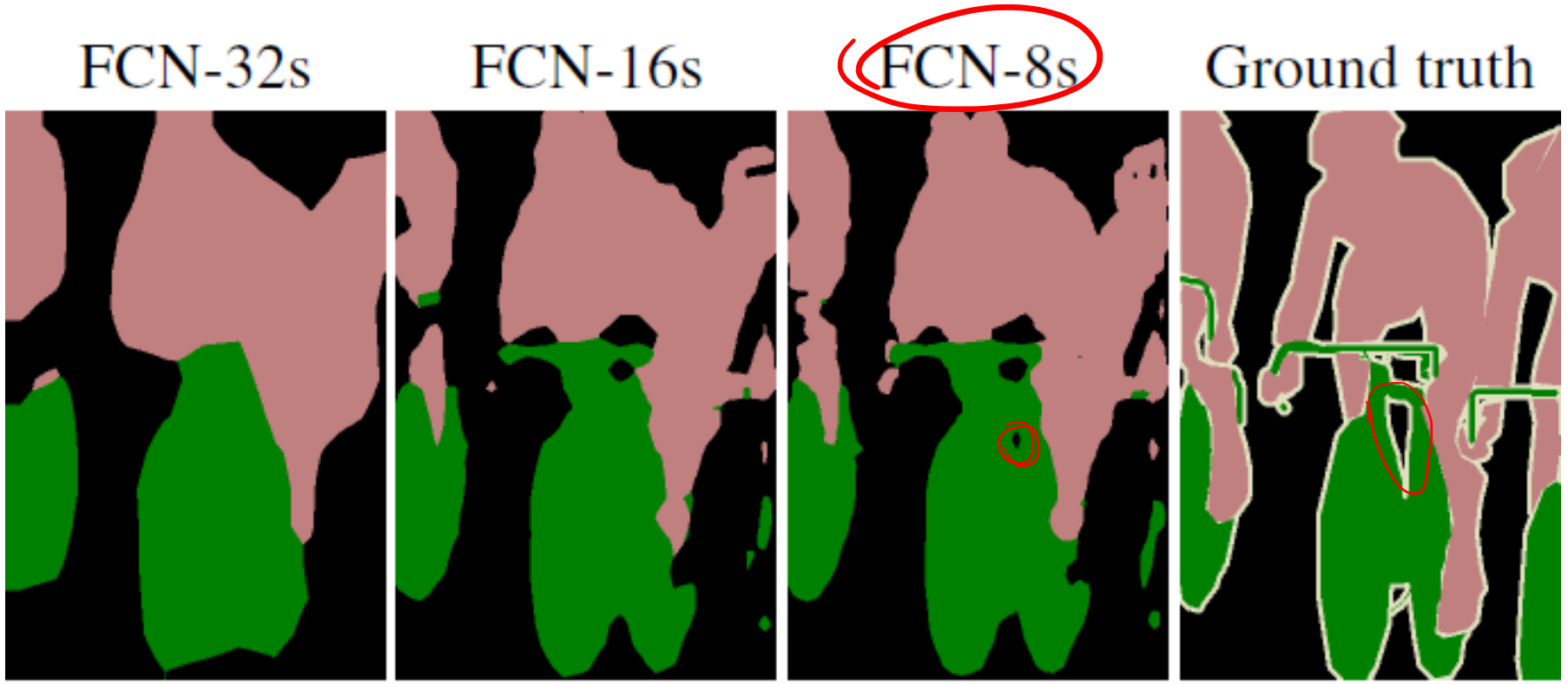


- **This yields 3 networks.** *Combining fine layers and coarse layers lets the model make local predictions that respect global structure.*
- Train first the lowest resolution network (FCN-32s)
- Then, the weights of the next network (FCN-16s) are initialized with (FCN-32s)
- The same for FCN-8s
- All the FC layers are converted to convolutional layers 1x1

Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

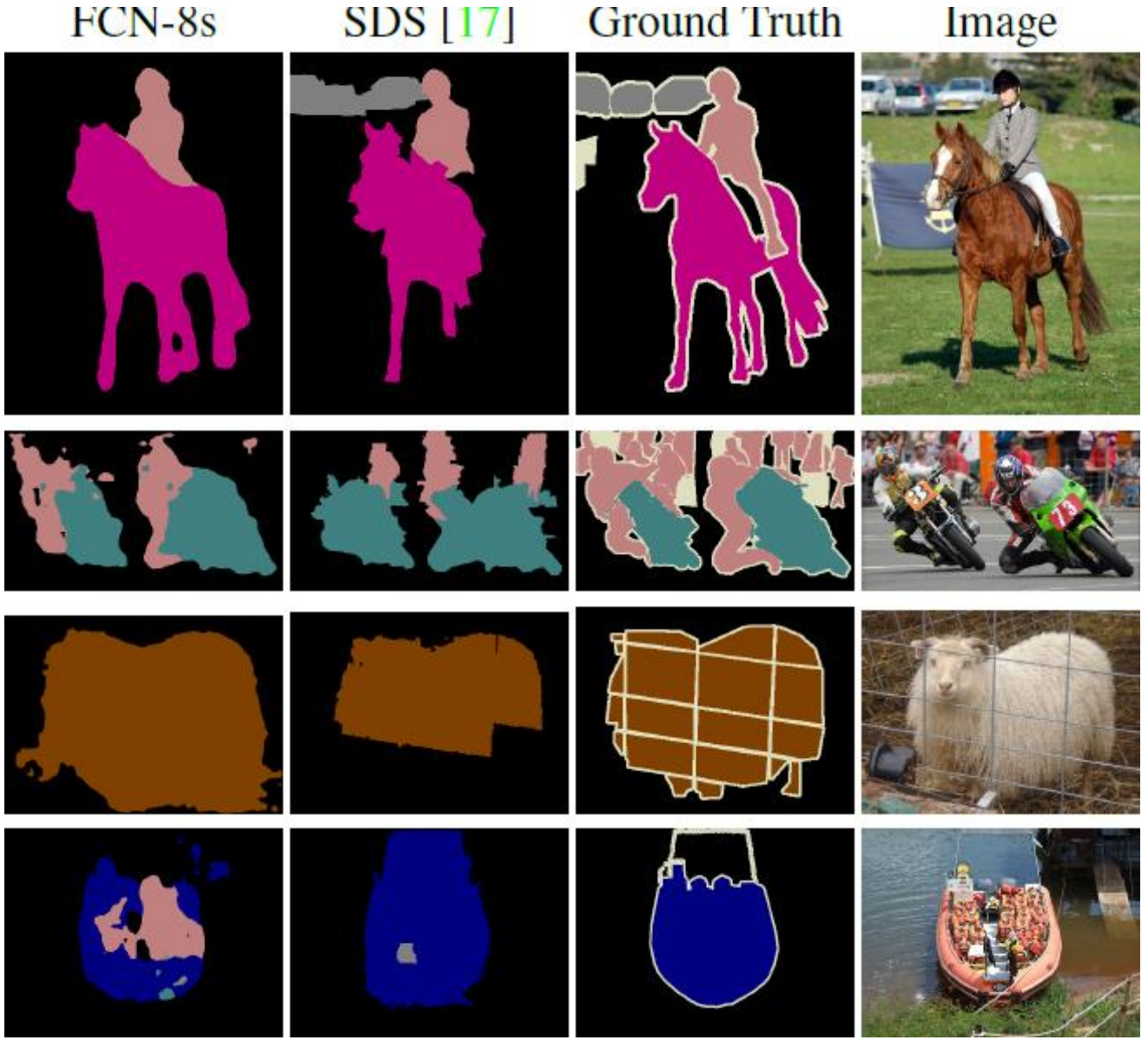
Semantic Segmentation

Retaining intermediate information is beneficial, the deeper layers contribute to provide a better refined estimate of segments



Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

Semantic Segmentation Results



Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

Comments

- Both learning and inference can be performed on the whole-image-at-a-time
- Both in full-image or batch-training it is possible to **perform transfer learning/fine tuning** of **pre-trained classification models** (segmentation typically requires fewer labels than classification)
- **Accurate pixel-wise prediction** is achieved by upsampling layers
- **End-to-end training is more efficient** than patch-wise training
- Outperforms state-of the art in 2015
- Being fully convolutional, this network **handles arbitrarily sized input**

Training a Fully-Convolutional CNN

G. Boracchi

FCNN Training Options: the «patch-based» way

- Prepare a training set for a classification network
- Crop as many patches $\boldsymbol{x_i}$ from annotated images and assign to each patch, the label corresponding to the patch center

FCNN Training Options

The «patch-based» way:

- Prepare a training set for a classification network
- Crop as many patches $\boldsymbol{x_i}$ from annotated images and assign to each patch, **the label corresponding to the patch center**
- **Train a CNN for classification** from scratches, or
  fine tune a pre-trained model over the segmentation classes
- Convolutionalization: once trained the network, **move the FC layers to 1x1 convolutions**
- **Design and train the upsampling side of the network**

FC-CNN Training Options

The «patch-based» way:

- The classification network is trained to minimize the classification loss ℓ over a mini-batch $\boldsymbol{B}$

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{\boldsymbol{x_j} \in \boldsymbol{B}} \ell(\boldsymbol{x_j}, \theta)$$

  where $\boldsymbol{x_j}$ belongs to a mini-batch $\boldsymbol{B}$

- Batches of patches are **randomly assembled during training**
- **It is possible to resample patches for solving** class imbalance
- It is **very inefficient**, since convolutions on overlapping patches are repeated multiple times

FC-CNN Training Options: The «full-image» way

Since the network provides dense predictions, it is possible to directly train a FCNN that includes upsampling layers as well

Learning becomes:

$$\hat{\theta} = \underset{x_j \in I}{\operatorname{argmin}} \sum_{x_j \in R} \ell(x_j, \theta)$$

Where x_j **are all the pixels in a region** R **of the input image** and the loss is evaluated over the corresponding labels in the annotation for semantic segmentation. Therefore, **each region provides already a mini-batch estimate for computing gradient.**

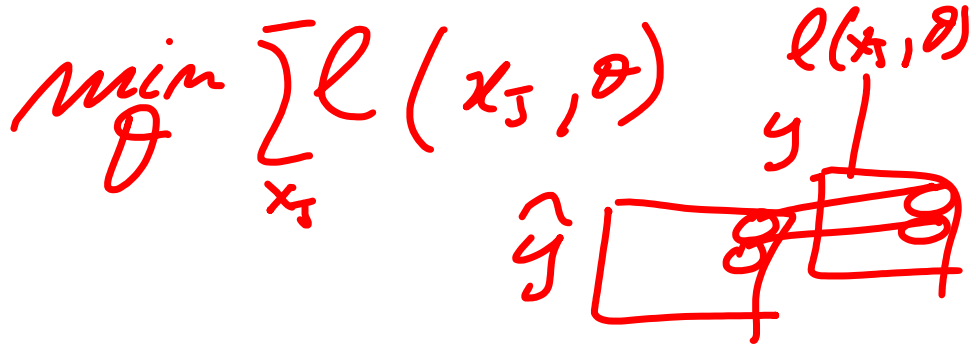
Whole image fully convolutional training is identical to patchwise training where each batch consists of all the receptive fields of the units below the loss for an image.

FCNN Training Options

The «full-image» way:

- **FCNN are trained in an end-to-end manner to predict the segmented output** $S(\cdot,\cdot)$
- This loss is the sum of losses over different pixels. Derivatives can be easily computed through the whole network, and this can be trained through backpropagation
- No need to pass through a classification network first
- Takes **advantage of FC-CNN efficiency**, does not have to re-compute convolutional features in overlapping regions

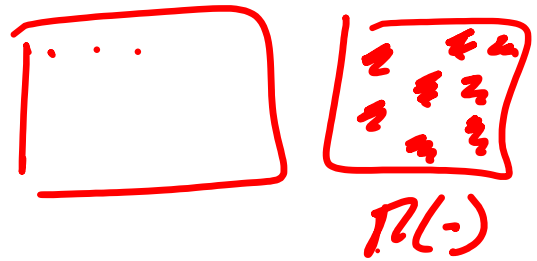
FCNN Training Options



Limitations of full-image training and solutions:

- **Minibatches in patch-wise training are assambled randomly.** Image regions in full-image training are not. To make the estimated loss a bit stochastic, adopt random mask

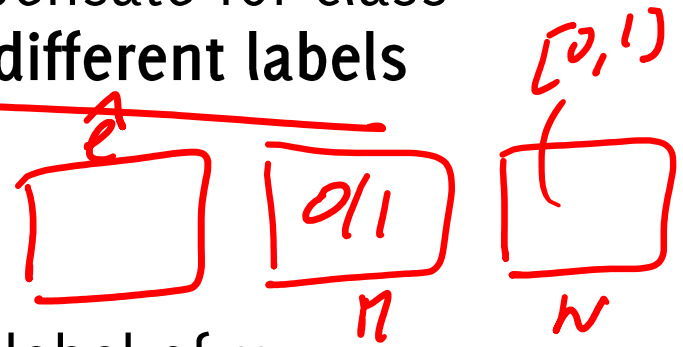
$$\text{minimize} \sum_{x_j} M(\boldsymbol{x_j}) \ell(\boldsymbol{x_j}, \theta)$$



being $M(\boldsymbol{x_j})$ a binary random variable

- **It is not possible to perform patch resampling** to compensate for class imbalance. One should go **for weighting the loss over different labels**

$$\text{minimize} \sum_{x_j} w(\boldsymbol{x_j}) \ell(\boldsymbol{x_j}, \theta)$$



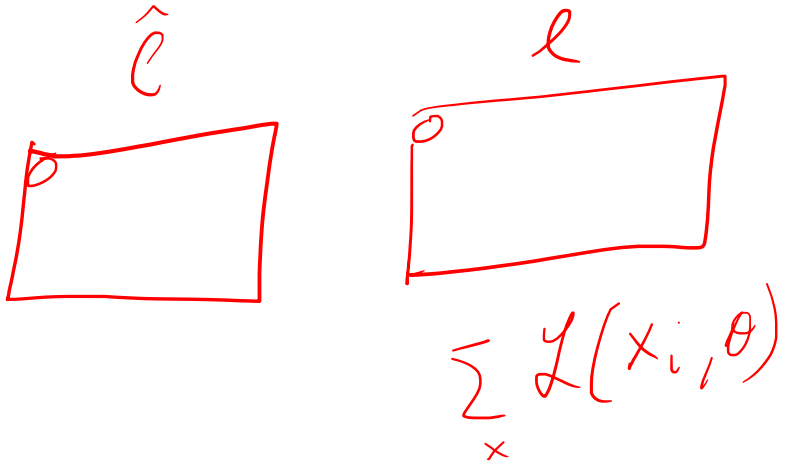
being $w(\boldsymbol{x_j})$ a weight that takes into account the true label of $\boldsymbol{x_j}$

U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies,
University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/

U-Net



Network formed by:

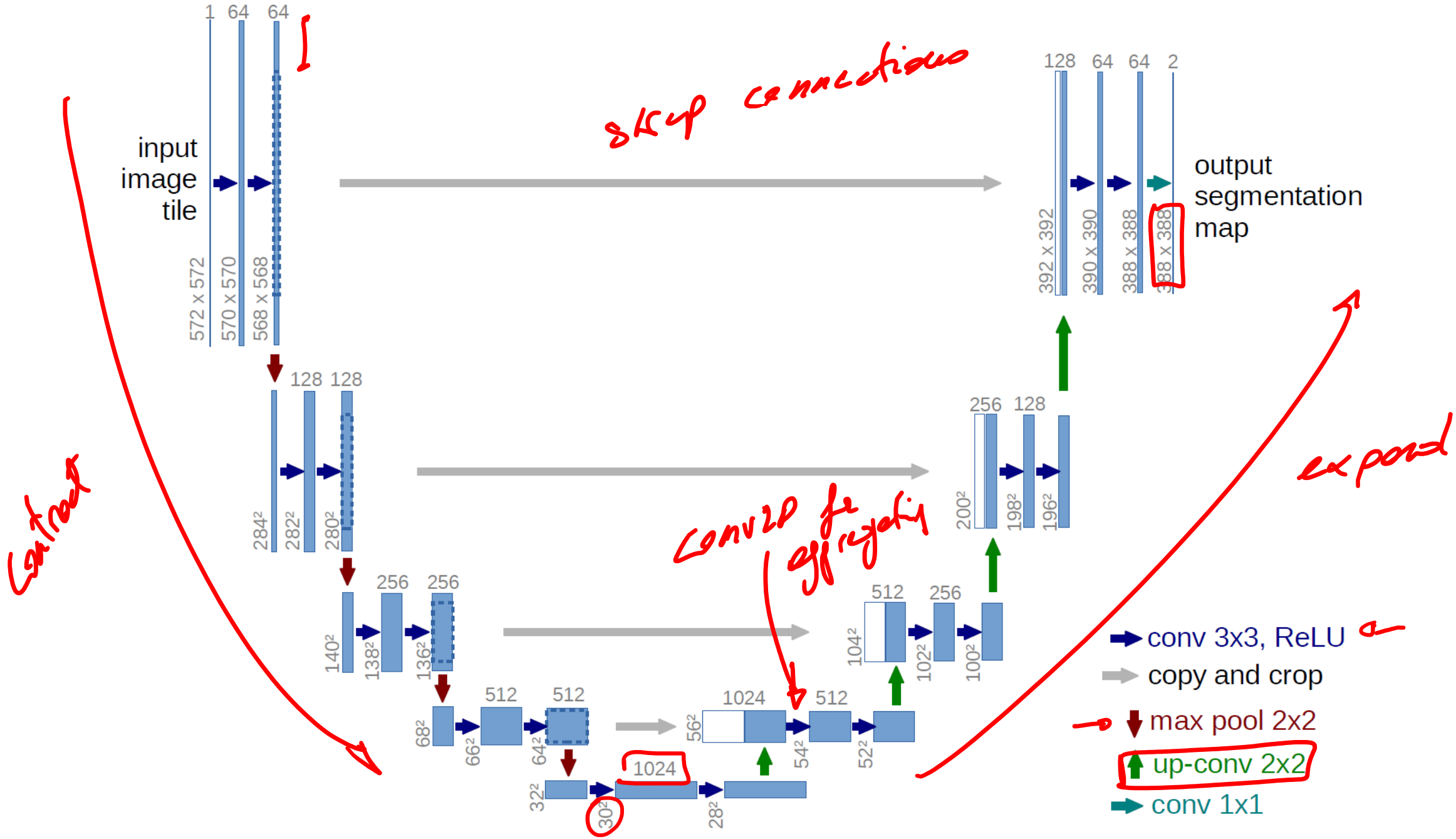
- A contracting path
- An expansive path

No fully connected layers

Major differences w.r.t. (Long et al. 2015):

- use **a large number of feature channels** in the upsampling part, while in (long et al. 2015) there were a few upsampling. The network become symmetric
- **Use excessive data-augmentation** by applying elastic deformations to the training images

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *MICCAI*, 2015.



Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *MICCAI*, 2015.

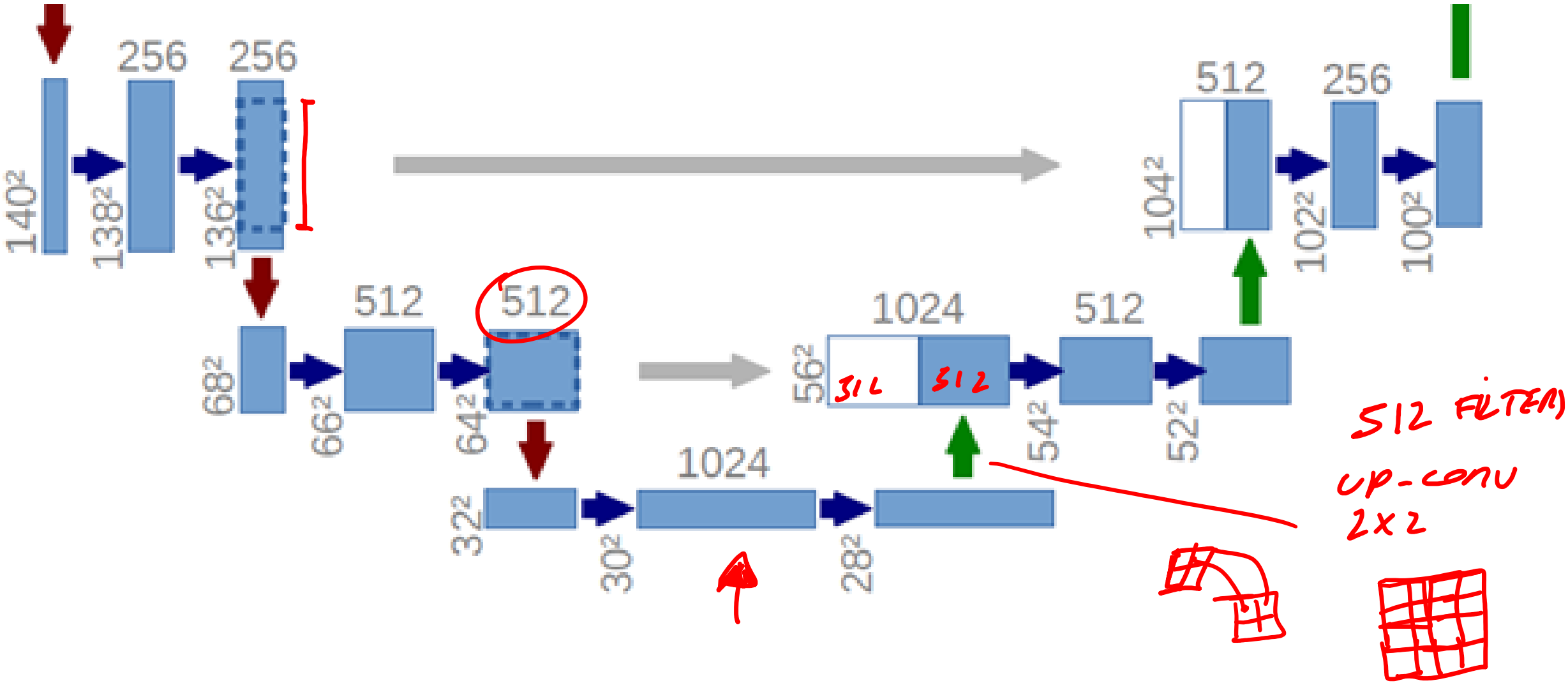
U-Net: Contracting path

Repeats blocks of:

- 3×3 convolution + ReLU ('valid' option, no padding)
- 3×3 convolution + ReLU ('valid' option, no padding)
- Maxpooling 2×2

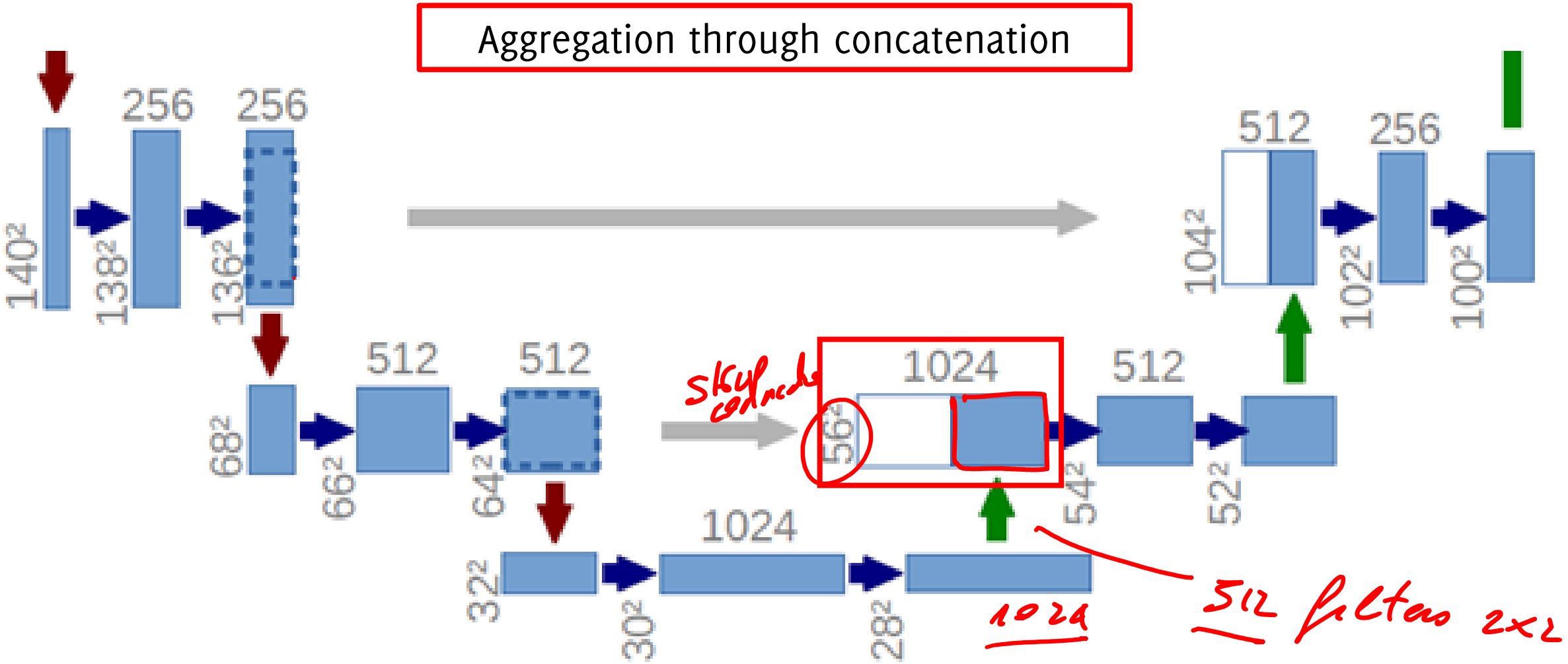
At each downsampling the number of feature maps is doubled

U-Net: Skip connections



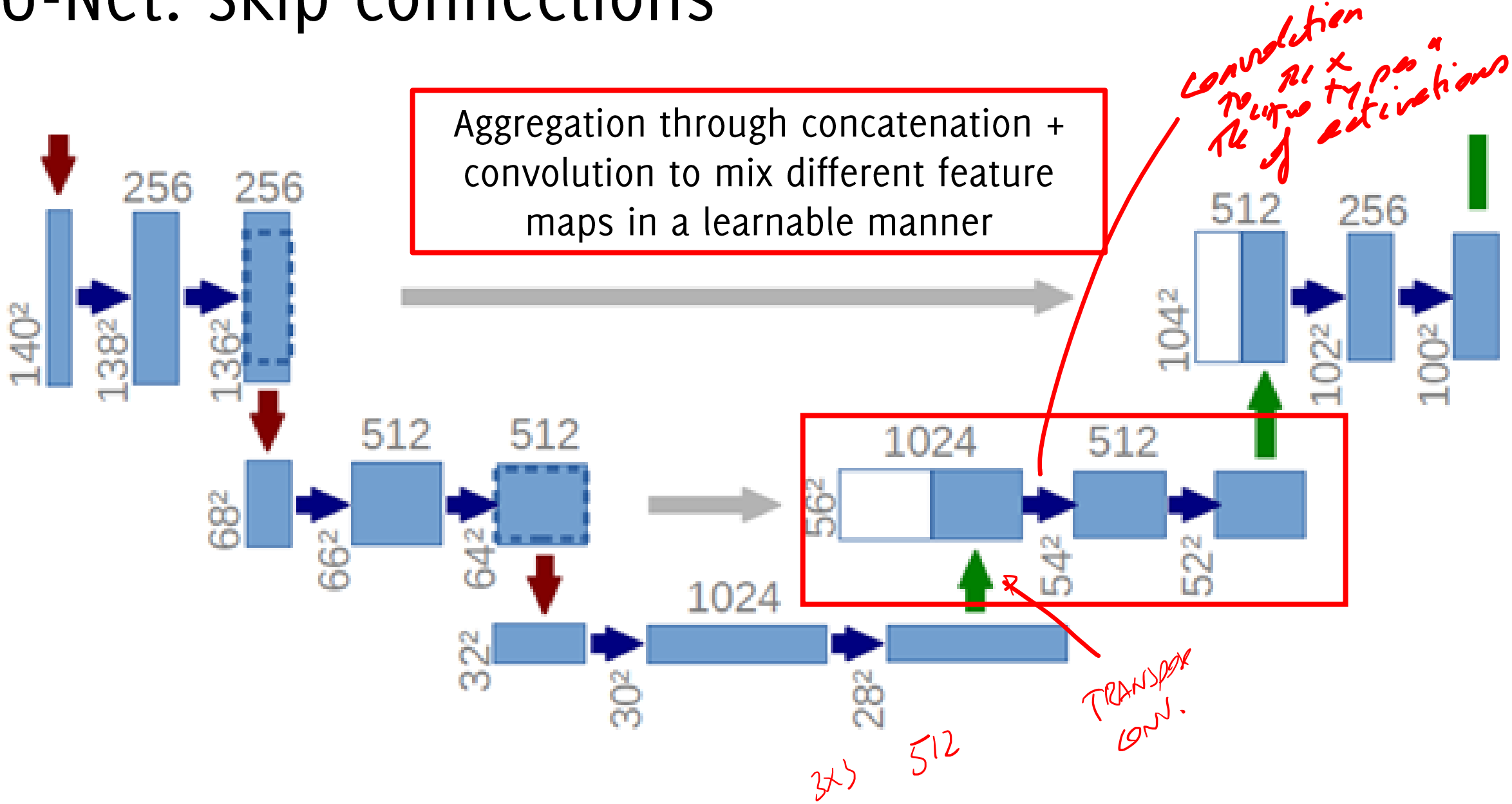
Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *MICCAI*, 2015.

U-Net: Skip connections



Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *MICCAI*, 2015.

U-Net: Skip connections



Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *MICCAI*, 2015.

U-Net: Expanding path

Repeats blocks of:

- 2×2 transpose convolution, halving the number of feature maps (but doubling the spatial resolution)
- Concatenation of corresponding cropped features
- 3×3 convolution + ReLU
- 3×3 convolution + ReLU

Aggregation during upsampling

U-Net: Network Top

No fully connected layers: there are L convolutions against filters $1 \times 1 \times N$, to yield predictions out of the convolutional feature maps

Output image is smaller than the input image by a constant border

U-Net: Training

Full-image training by a weighted loss function

$$\hat{\theta} = \min_{\theta} \sum_{\boldsymbol{x}_j} w(\boldsymbol{x}_j)\ell(\boldsymbol{x}_j, \theta)$$

where the weight

$$w(\boldsymbol{x}) = w_c(\boldsymbol{x}) + w_0\, e^{-\frac{(d_1(\boldsymbol{x})+d_2(\boldsymbol{x}))^2}{2\sigma^2}}$$

- w_c is used to balance class proportions (remember no patch resampling in full-image training)
- d_1 is the distance to the border of the closest cell
- d_2 is the distance to the border of the second closest cell

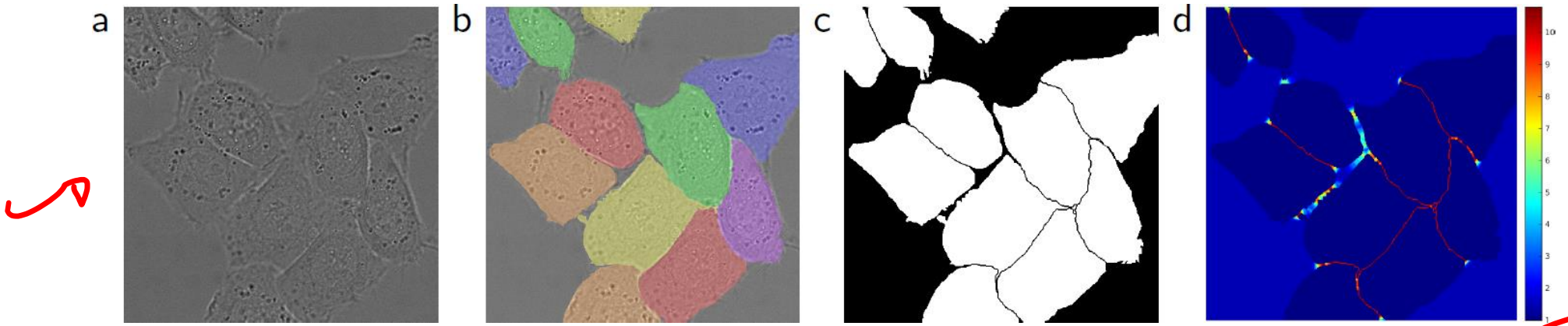
U-net: Training

Full-image training by a weighted loss function

$$w(\boldsymbol{x}) = w_c(\boldsymbol{x}) + w_0\, e^{-\frac{\left(d_1(\boldsymbol{x})+d_2(\boldsymbol{x})\right)^2}{2\sigma^2}}$$

- w_c is used to balance class proportions (remember no patch resampling in full-image training)
- d_1 is the distance to the border of the closest cell
- d_2 is the distance to the border of the second closest cell

Weights are large when the distance to the first two closest cells is small



Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *MICCAI*, 2015.

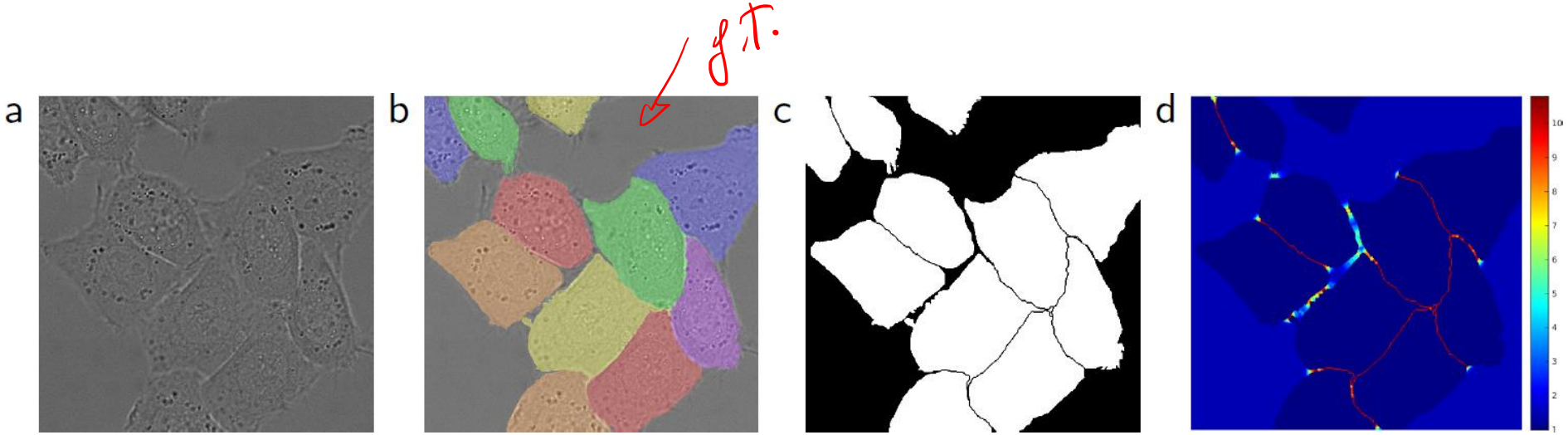
U-net: Training

Full-image training by a weighted loss function

$$w(\boldsymbol{x}) = w_c(\boldsymbol{x}) + w_0\, e^{-\frac{\left(d_1(\boldsymbol{x})+d_2(\boldsymbol{x})\right)^2}{2\sigma^2}}$$

Takes into account class unbalance in the training set

Enhances classification performance at borders of different objects



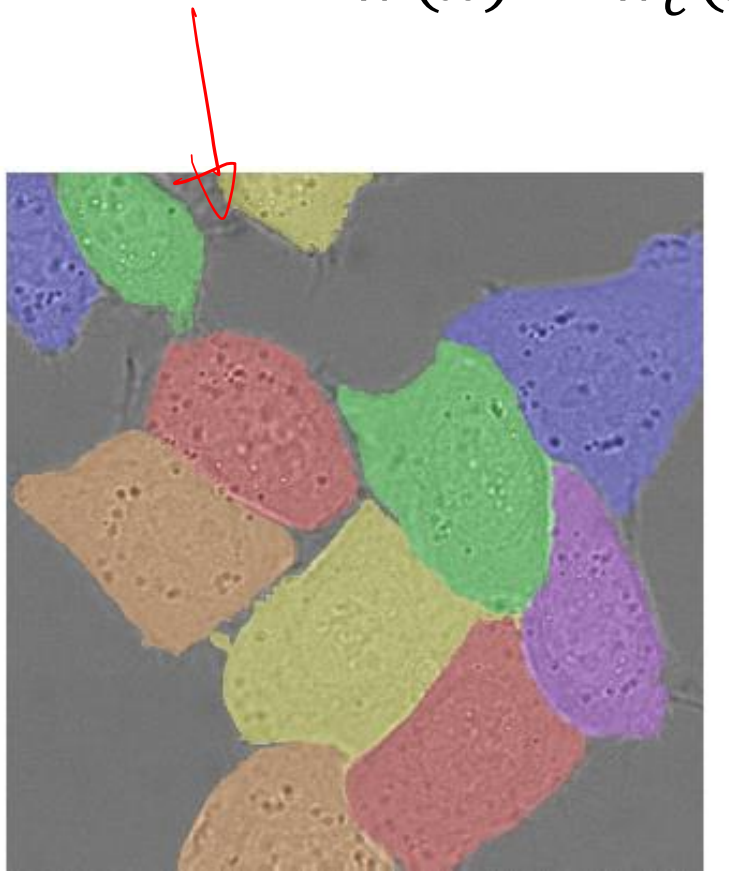
Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *MICCAI*, 2015.

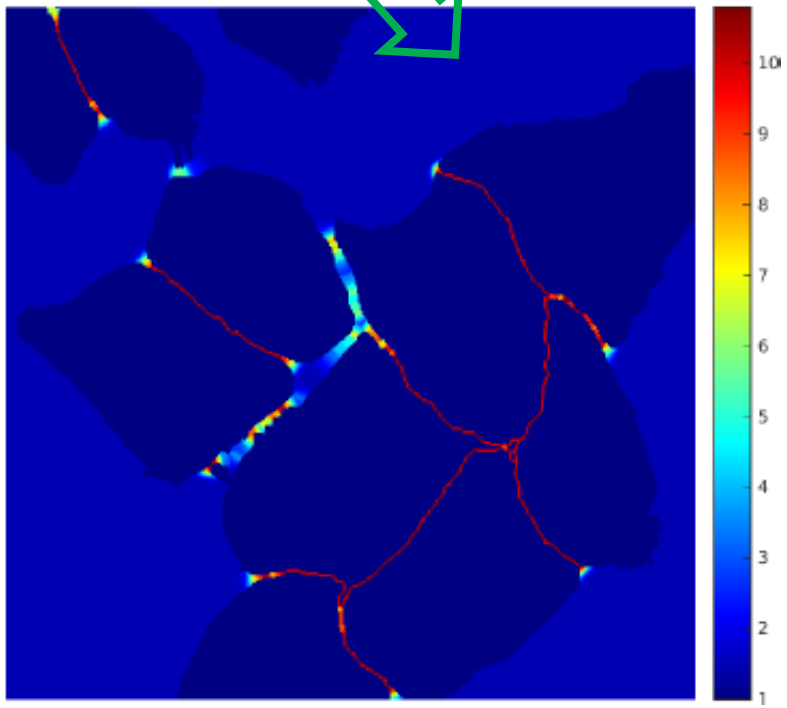
U-net: Training

This term is large at pixels close to borders delimiting objects of different cells

Full-image training by a weighted loss function

$$w(\boldsymbol{x}) = w_c(\boldsymbol{x}) + w_0\, e^{-\frac{\left(d_1(\boldsymbol{x})+d_2(\boldsymbol{x})\right)^2}{2\sigma^2}}$$





Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *MICCAI*, 2015.