Convolutional Neural Networks for Semantic Segmentation

Giacomo Boracchi

giacomo.boracchi@polimi.it

Semantic Segmentation Task

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

Zheng et al. *"Conditional Random Fields as Recurrent Neural Networks"*, ICCV 2015

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

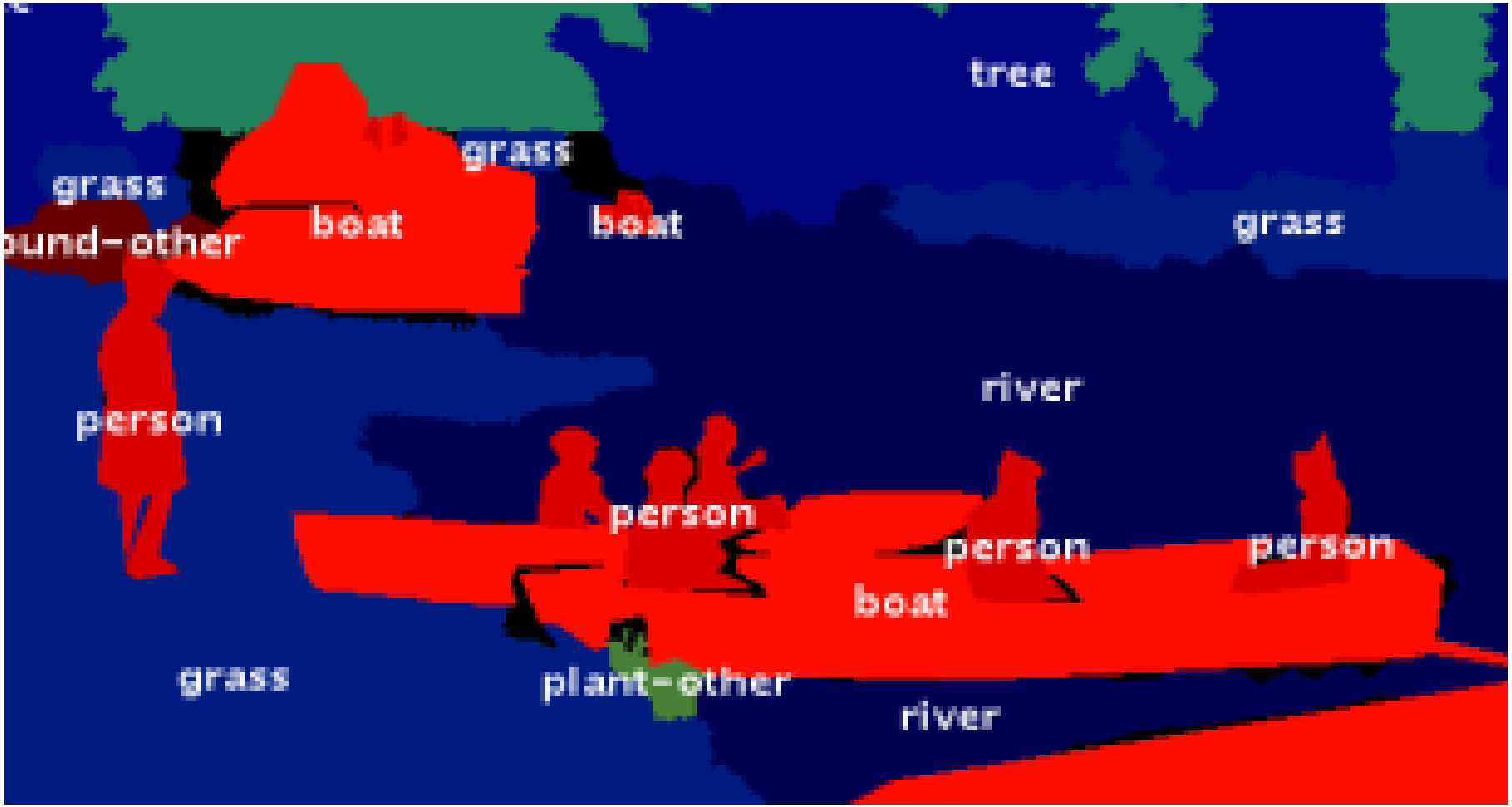
I

GT

http://cocodataset.org/

Training Set

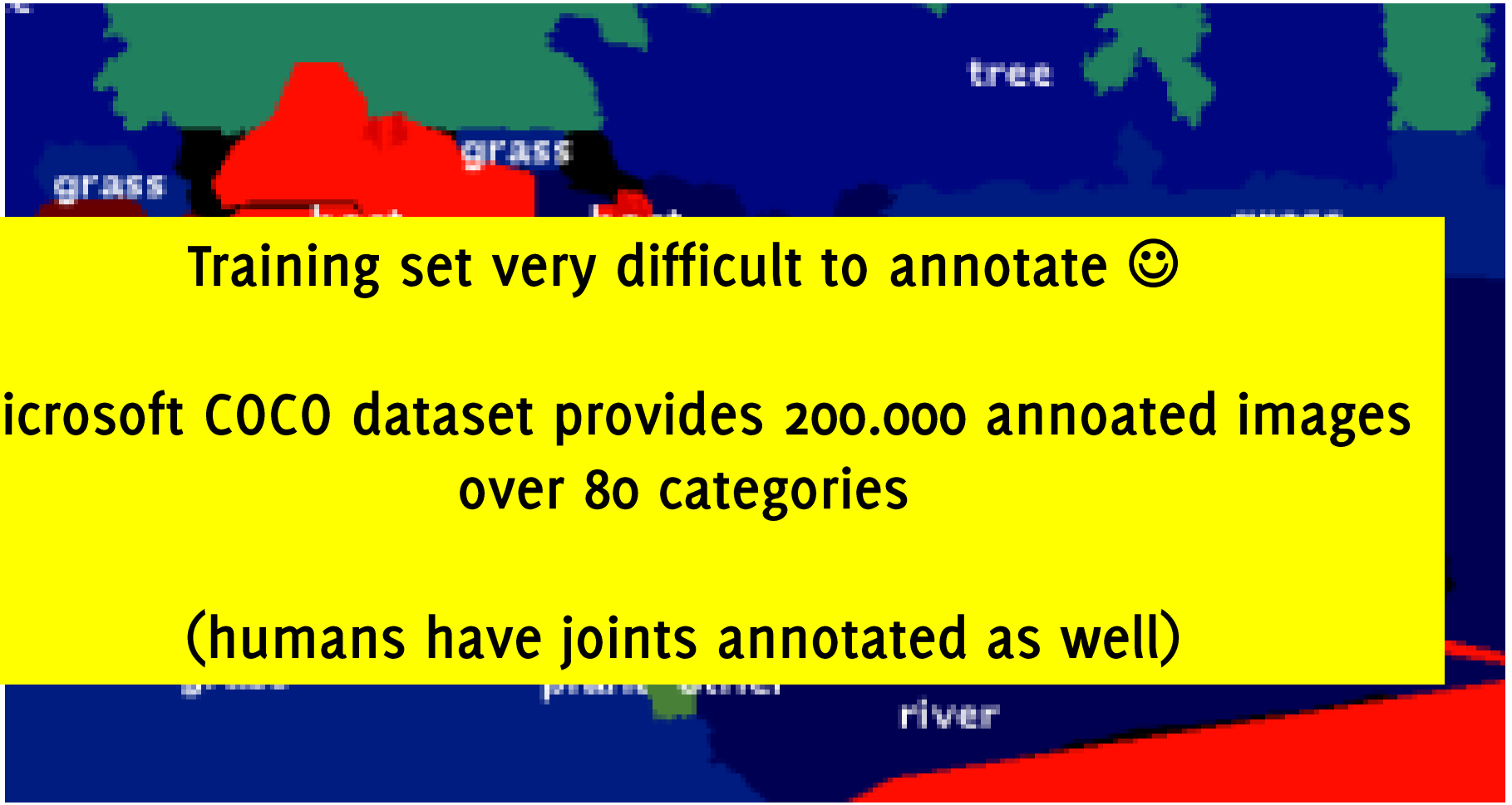
The training set is made of pairs (I, GT), where the GT is a pixel-wise annotated image over the categories in Λ



http://cocodataset.org/

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



Training set very difficult to annotate ☺

Microsoft COCO dataset provides 200.000 annoated images over 80 categories

(humans have joints annotated as well)

http://cocodataset.org/

Semantic Segmentation by Fully Convolutional Neural Networks

Predicting dense outputs for arbitrary-sized inputs

G. Boracchi

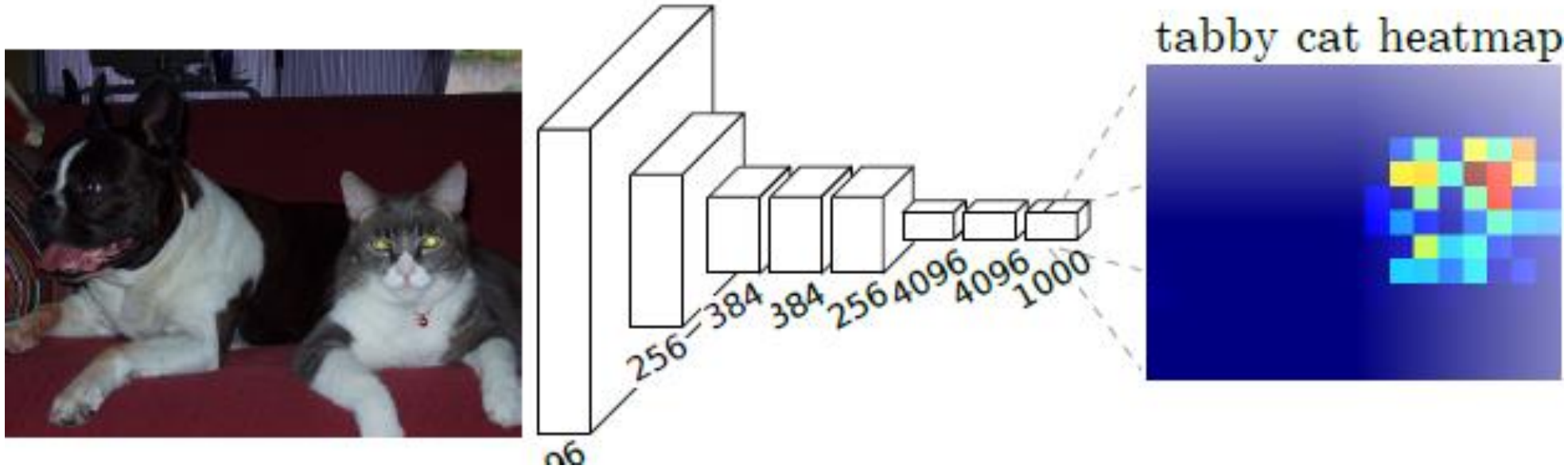
Fully Convolutional Networks for Semantic Segmentation

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

G. Boracchi

Simple Solution (1): Direct Heatmap Predictions

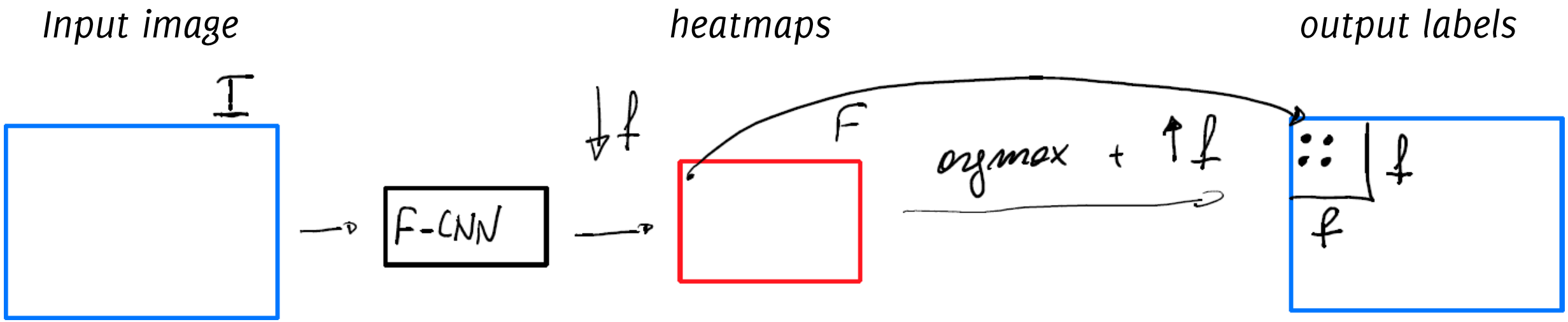
We can assign the predicted label in the heatmap to the whole receptive field, however that would be a **very coarse estimate**



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

Simple Solution (1): Direct Heatmap Predictions

Very coarse estimates



Simple Solution (2): The Shift and Stich

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

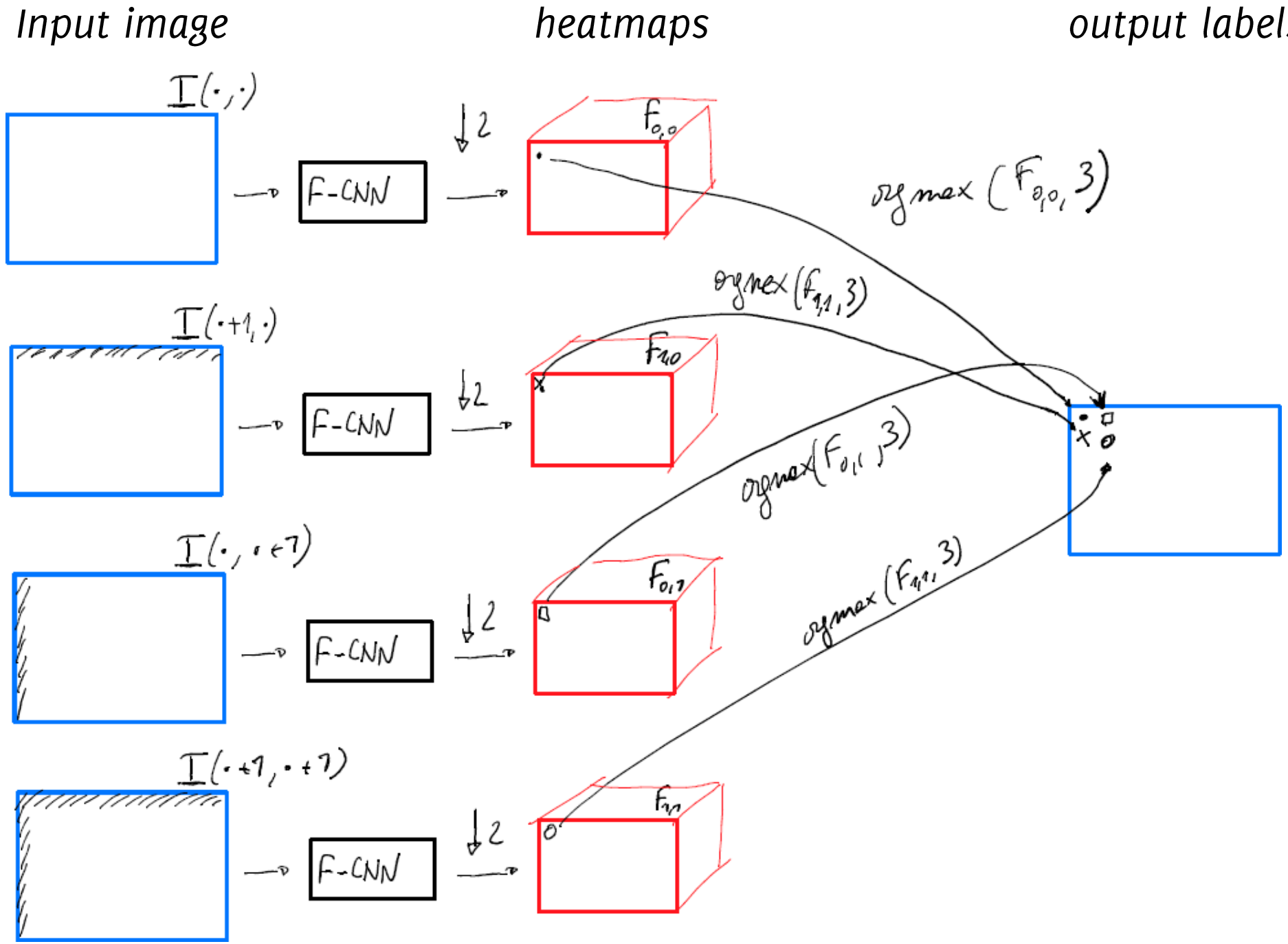
- Compute **heatmaps** for all $\boldsymbol{f^2}$ possible shifts of the input $(0 \leq r, c < f)$
- **Map predictions from the $\boldsymbol{f^2}$** heatmaps to the image: each pixel in the heatmap **provides prediction of the central pixel of the receptive field**
- **Interleave the heatmaps** to form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

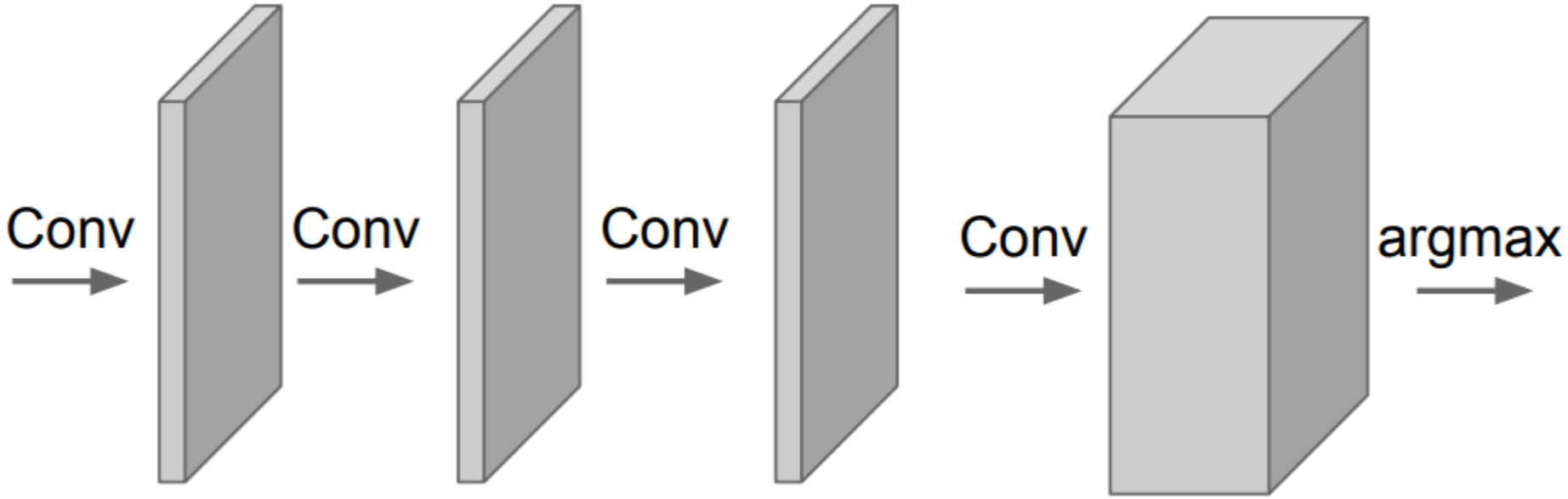

Input image
heatmaps
output labels
I(·,·)
I(·+1,·)
I(·,·+1)
I(·+1,·+1)
F-CNN
F-CNN
F-CNN
F-CNN
↓2
↓2
↓2
↓2
argmax (F0,0,3)
argmax (F1,1,3)
argmax (F0,1,3)
argmax (F1,0,3)

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:

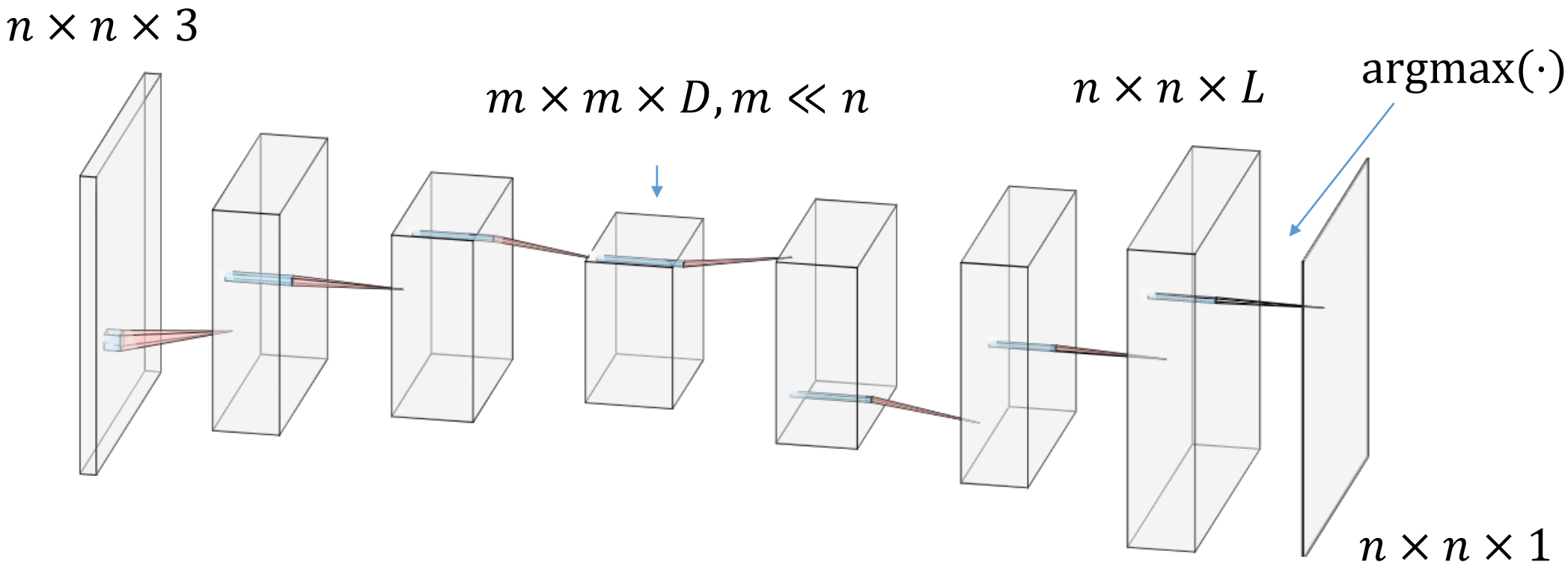
- ***global information resolves what****, while*
- ***local information resolves where***

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

Reduce latent representation dimension

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



Conv + pooling / downsampling

Conv + upsampling

G. Boracchi

Reduce latent representation dimension

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

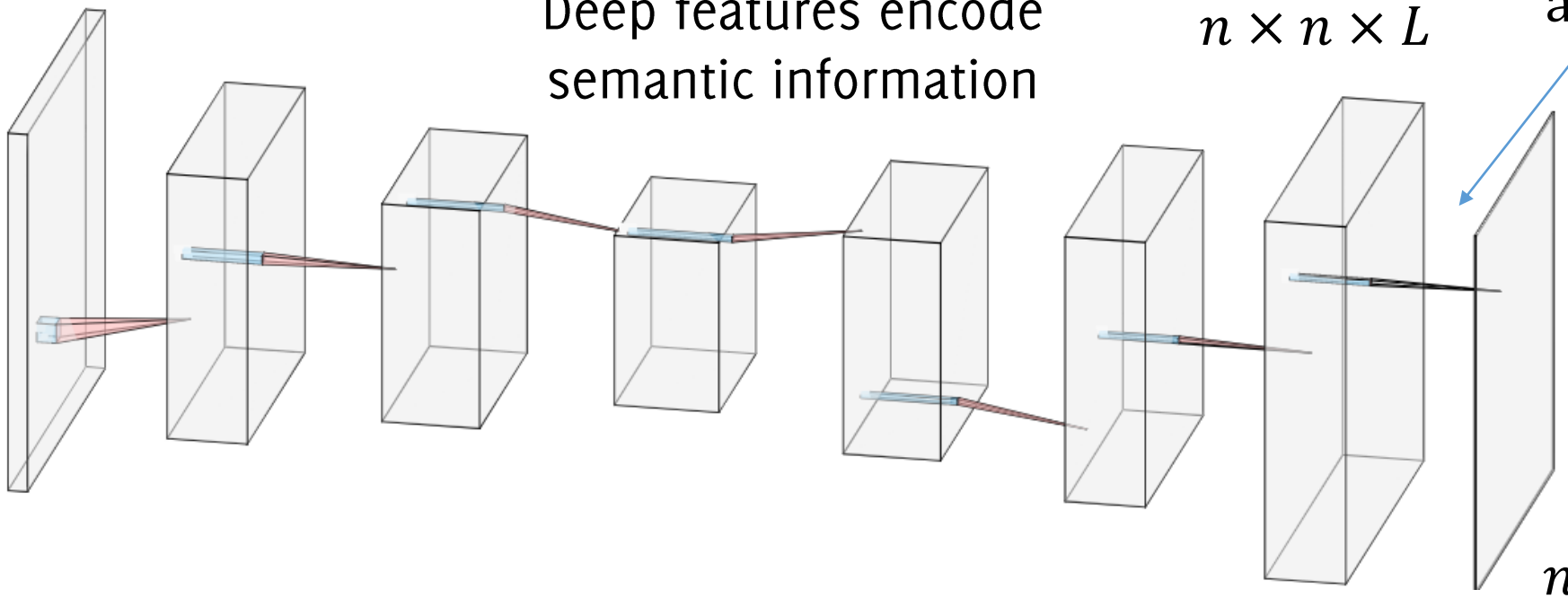
High resolution output to recover local information

$n \times n \times 3$

Deep features encode semantic information

$n \times n \times L$

$\text{argmax}(\cdot)$



$n \times n \times 1$

Conv + pooling / downsampling

Conv + upsampling

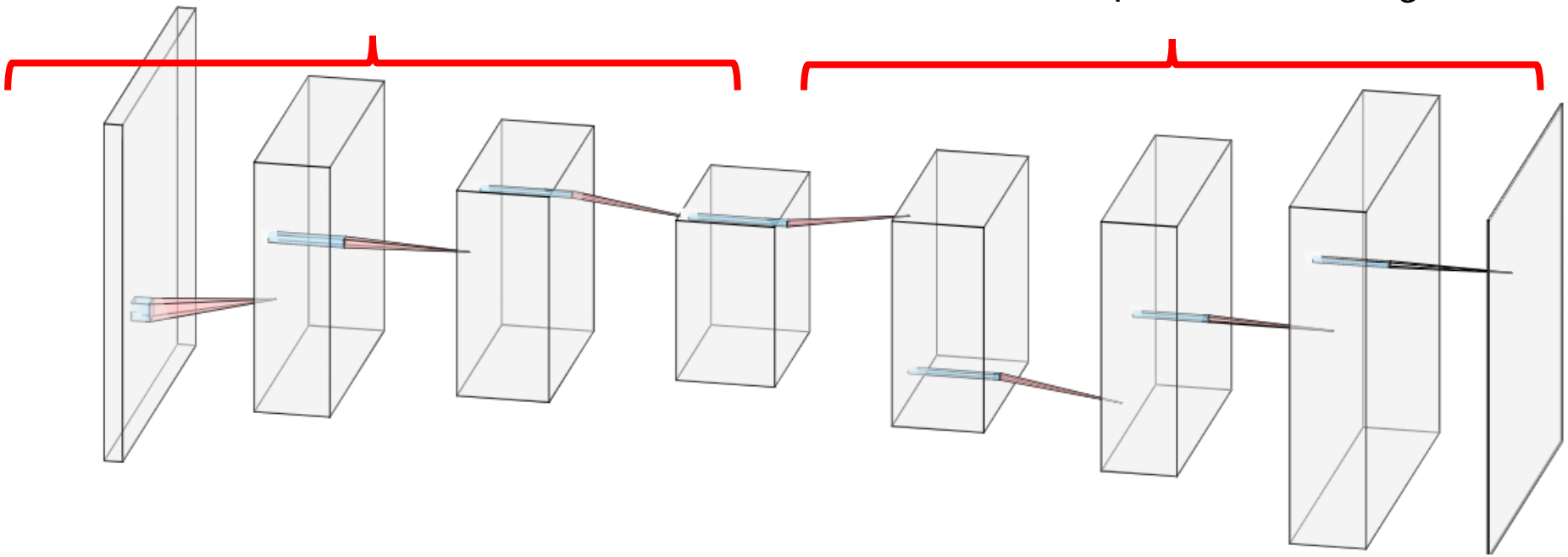
G. Boracchi

Reduce latent representation dimension

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

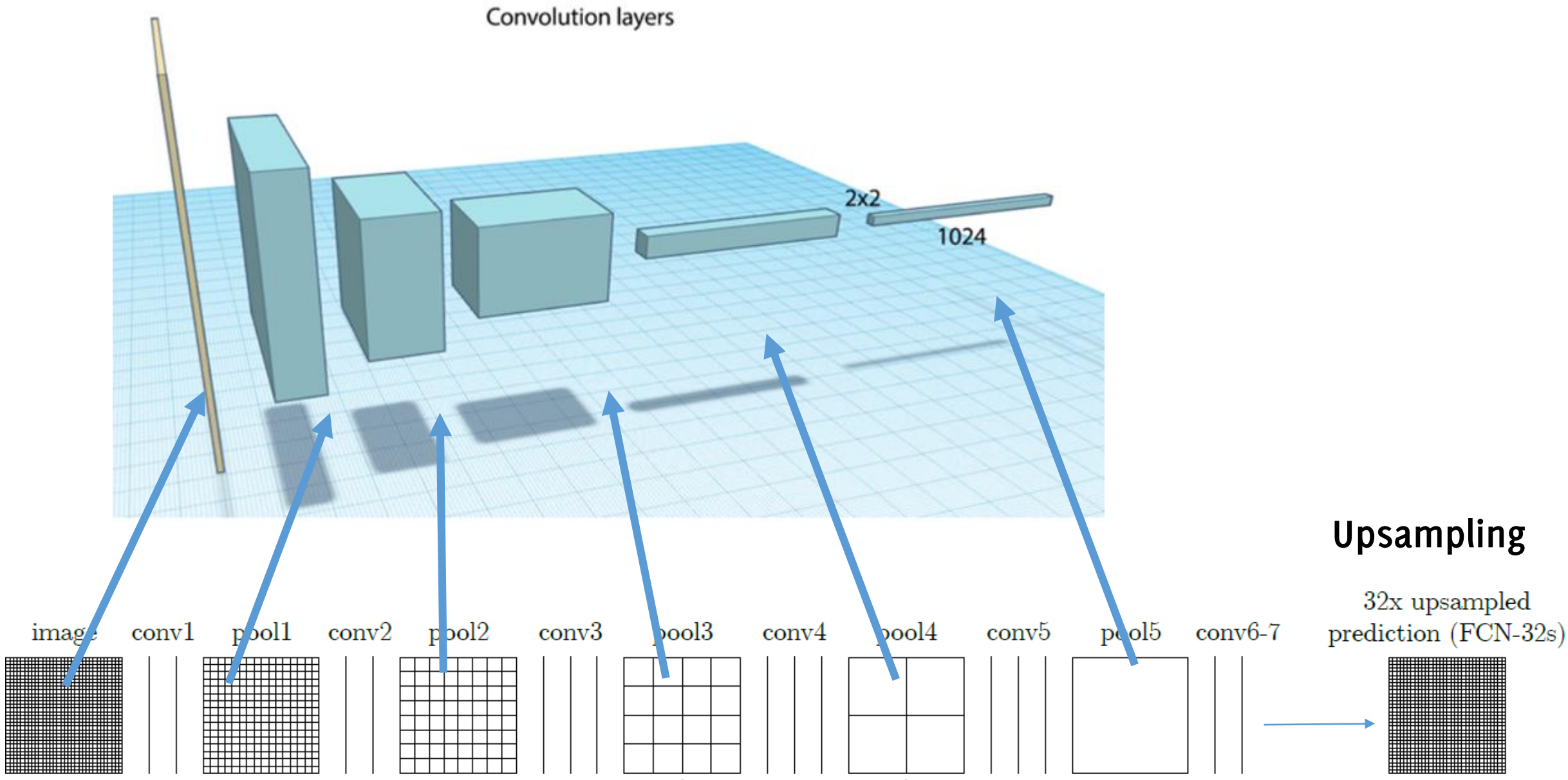


Increasing the image size is necessary to obtain sharp contours and spatially detailed class predictions

Conv + pooling / downsampling

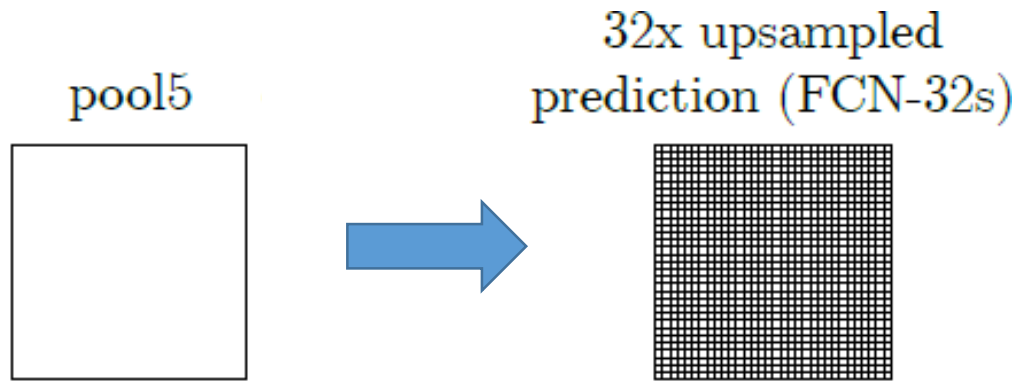
Conv + upsampling

G. Boracchi



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

→

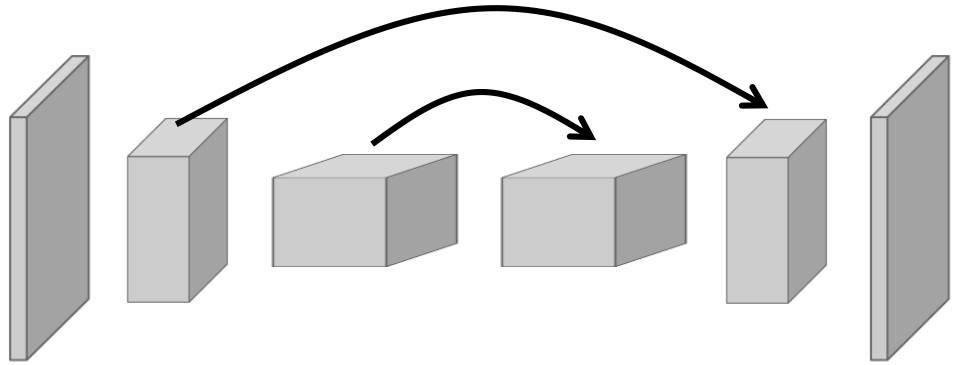
1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

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

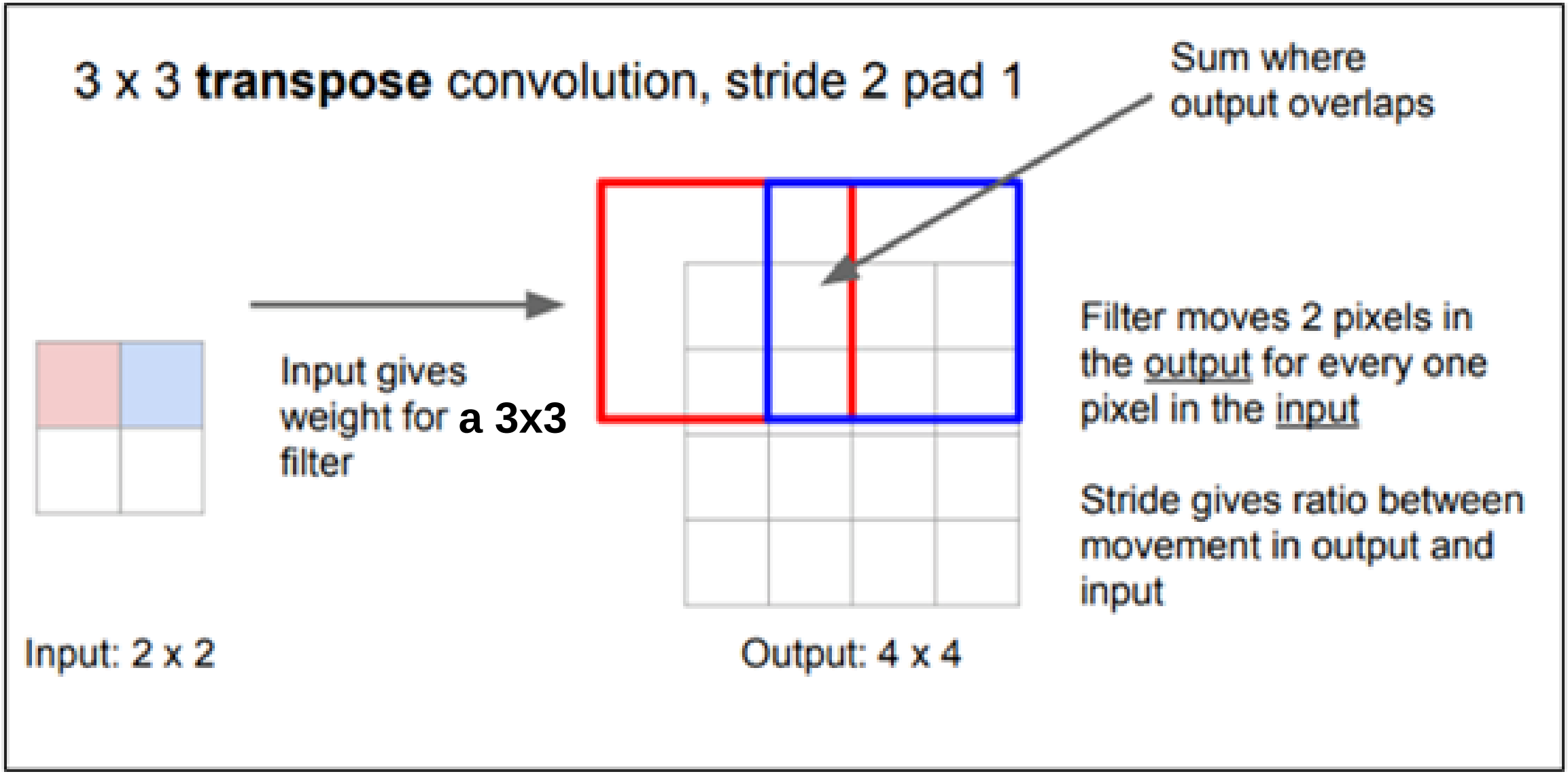
1	2
3	4

Input: 2 x 2

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

Output: 4 x 4

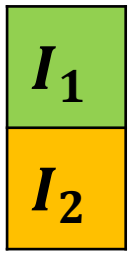
Transpose Convolution



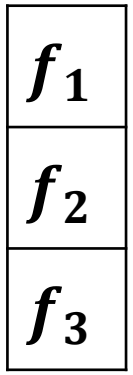
Transpose Convolution

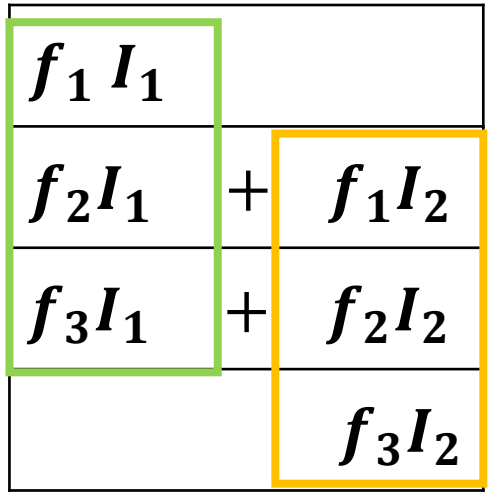
Transpose convolution with stride 1

Input 2 x 1



Filter 3 x 1

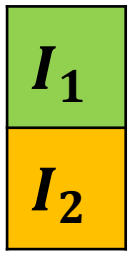




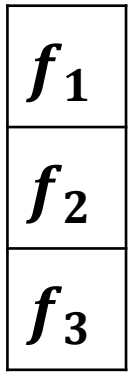
Transpose Convolution

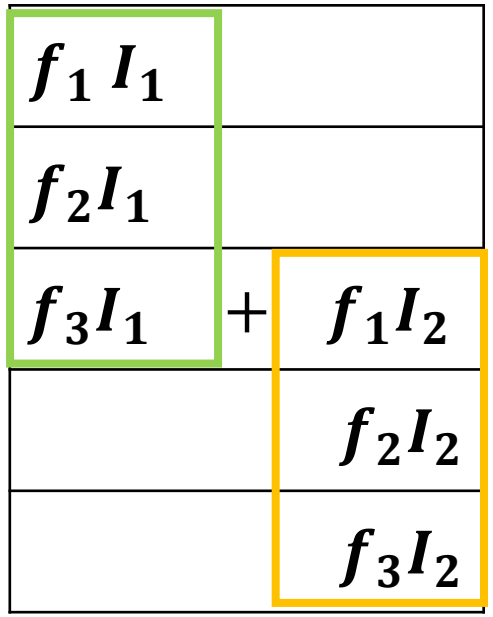
Transpose convolution with stride 2

Input 2 x 1



Filter 3 x 1



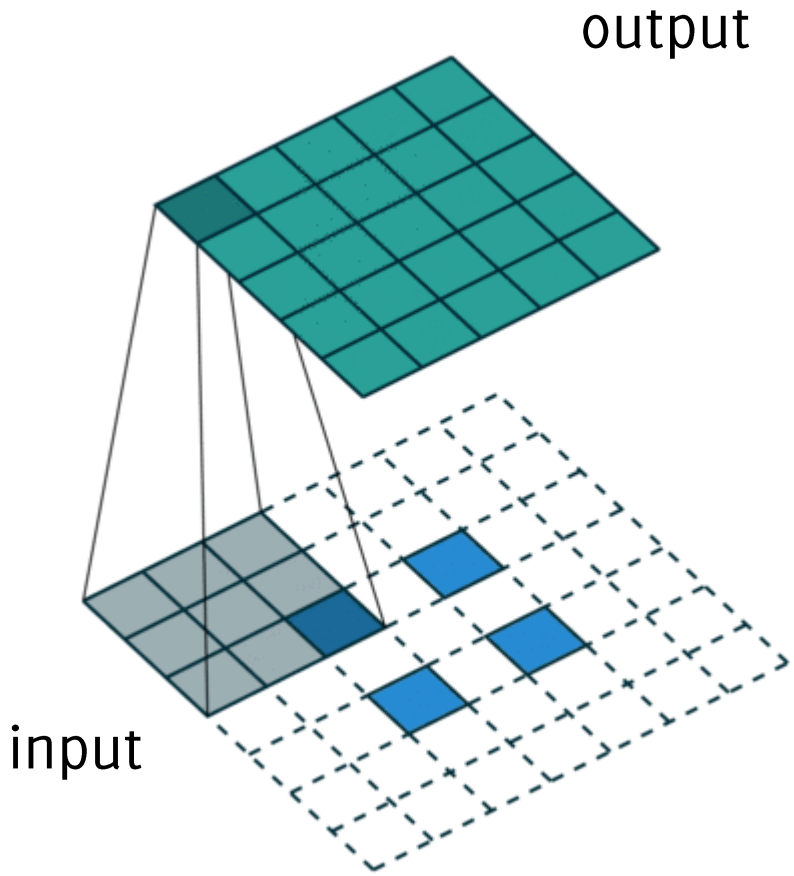


How to perform upsampling?

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).

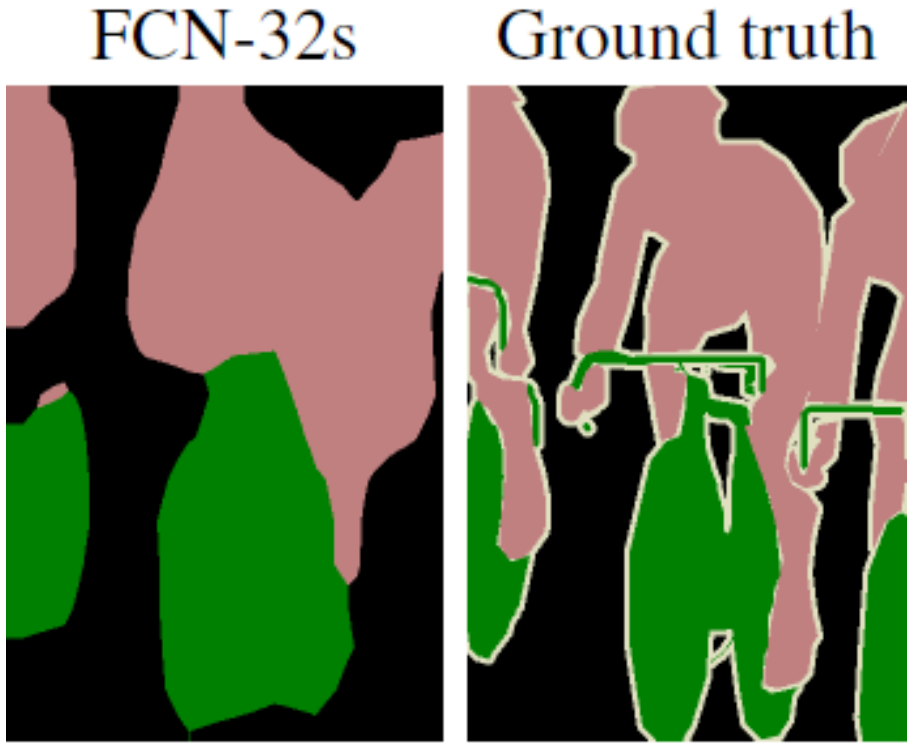
Prediction Upsampling

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

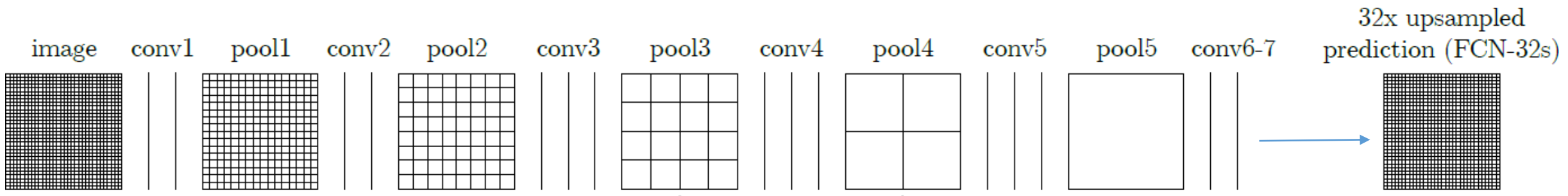
Upsampling filters can thus be learned during network training.

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

Upsampling filters are learned with initialization equal to the bilinear interpolation

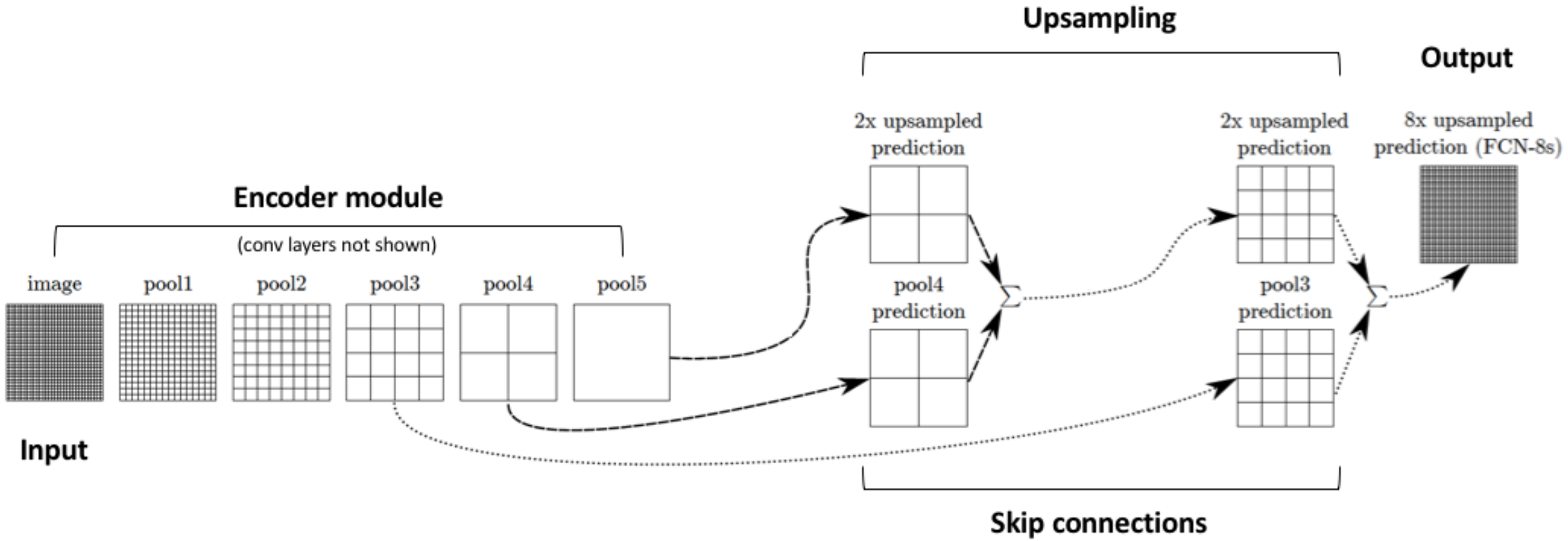


Upsampling filters



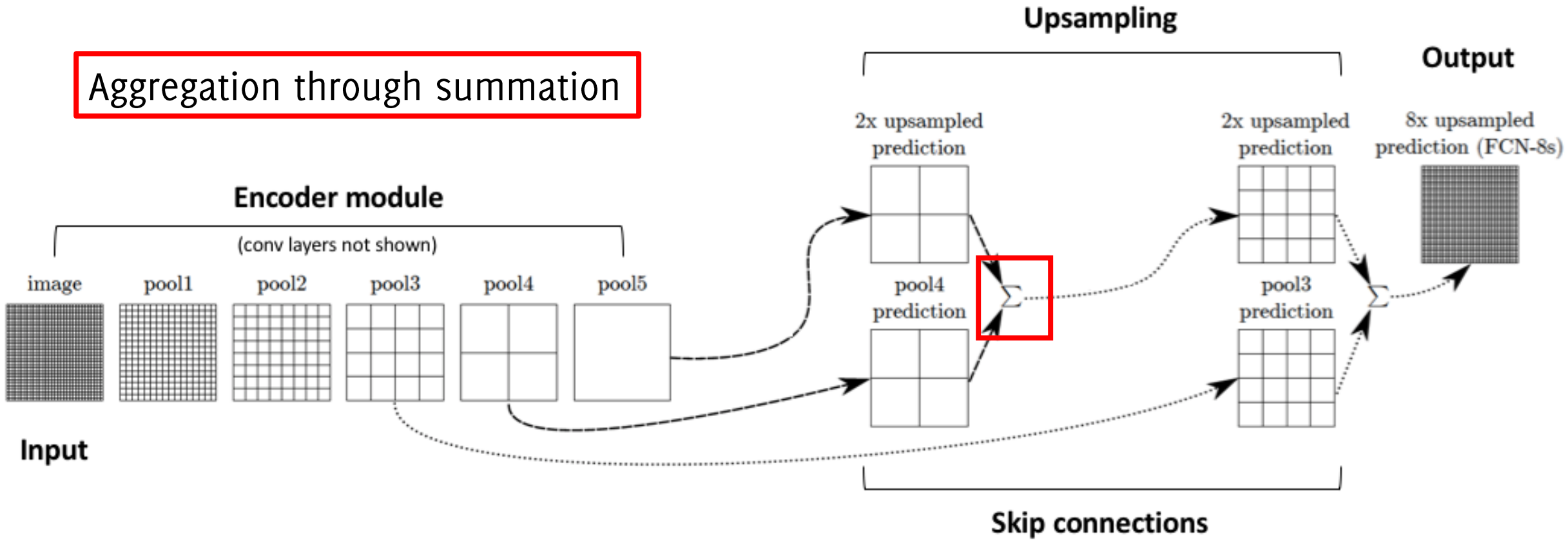
Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

Solution: Skip Connections!



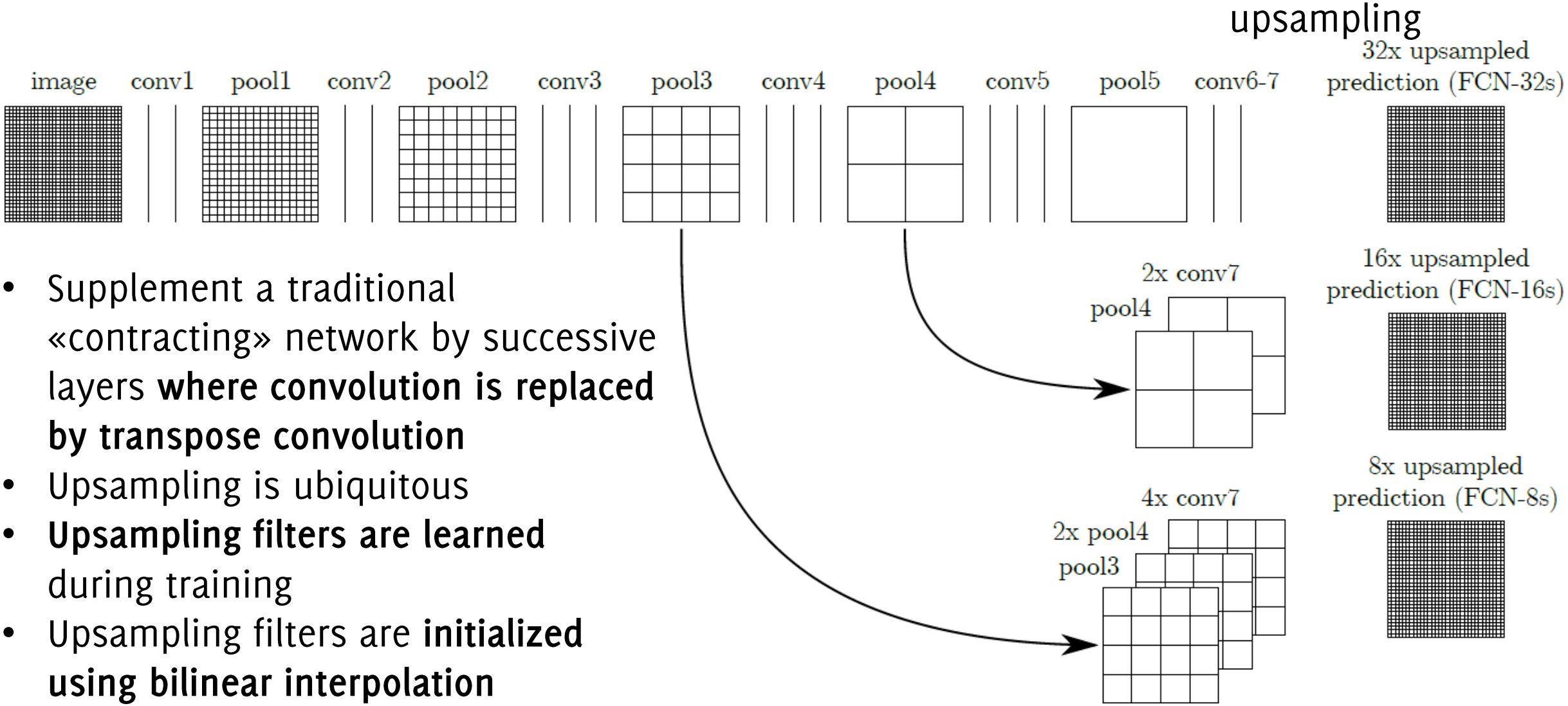
https://www.jeremyjordan.me/semantic-segmentation/

Solution: Skip Connections!



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

Solution: Skip Connections!

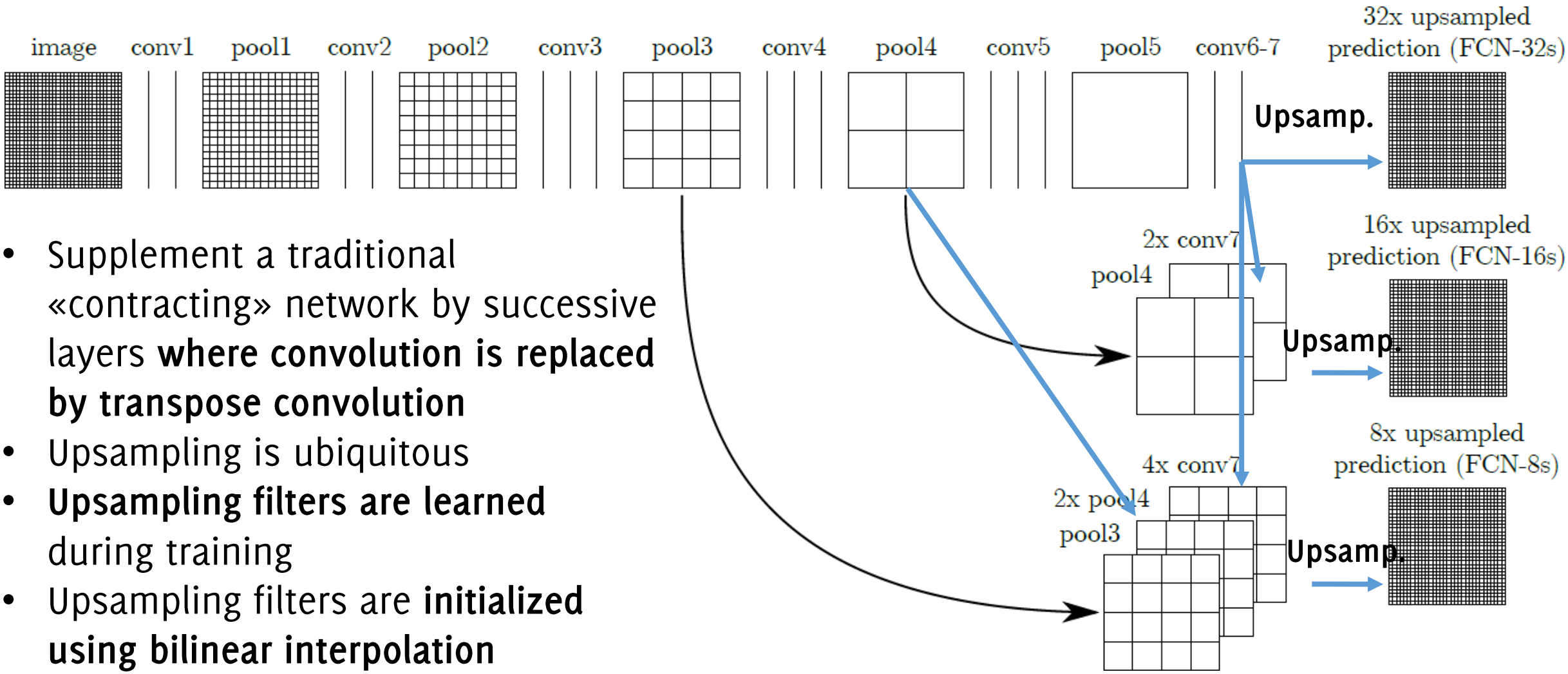


- Supplement a traditional «contracting» network by successive layers **where convolution is replaced by transpose convolution**
- Upsampling is ubiquitous
- **Upsampling filters are learned** during training
- Upsampling filters are **initialized using bilinear interpolation**

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

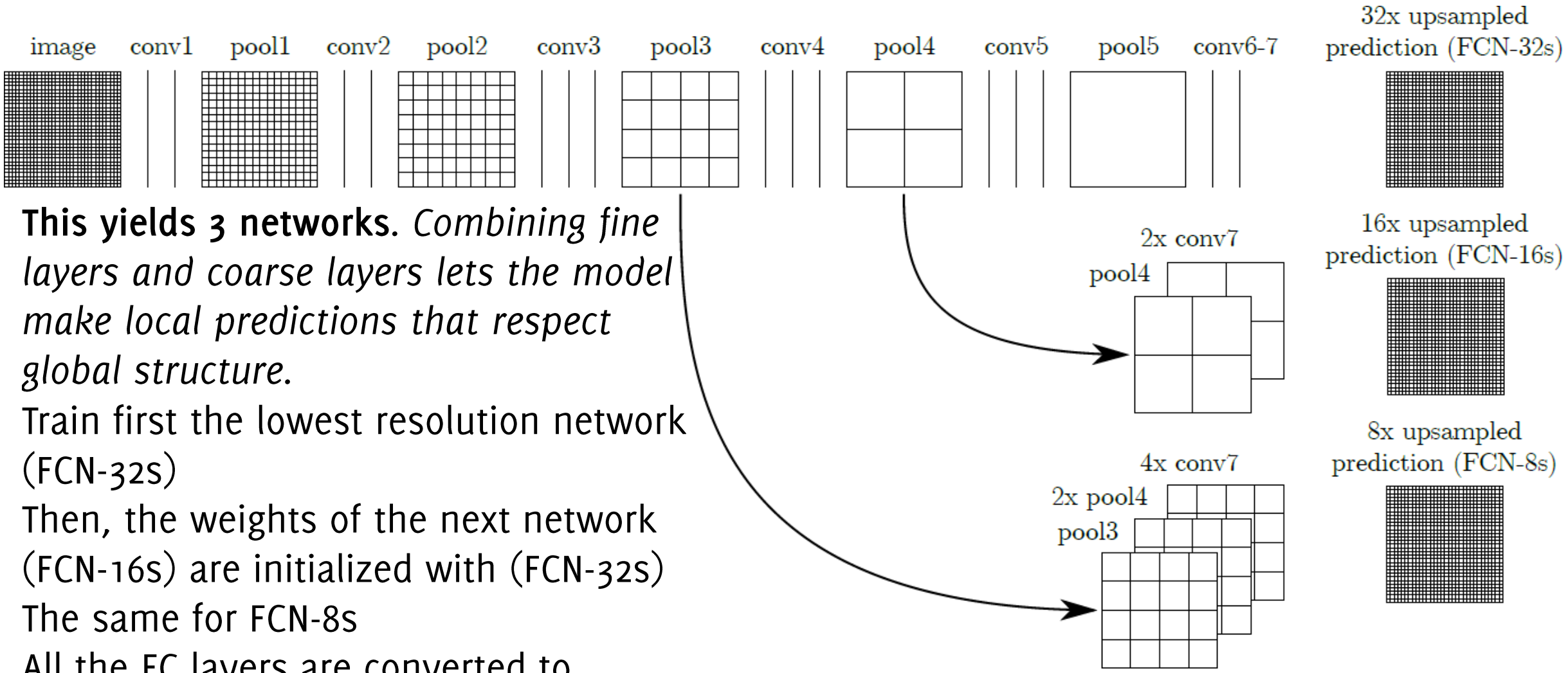
Solution: Skip Connections!

- 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!

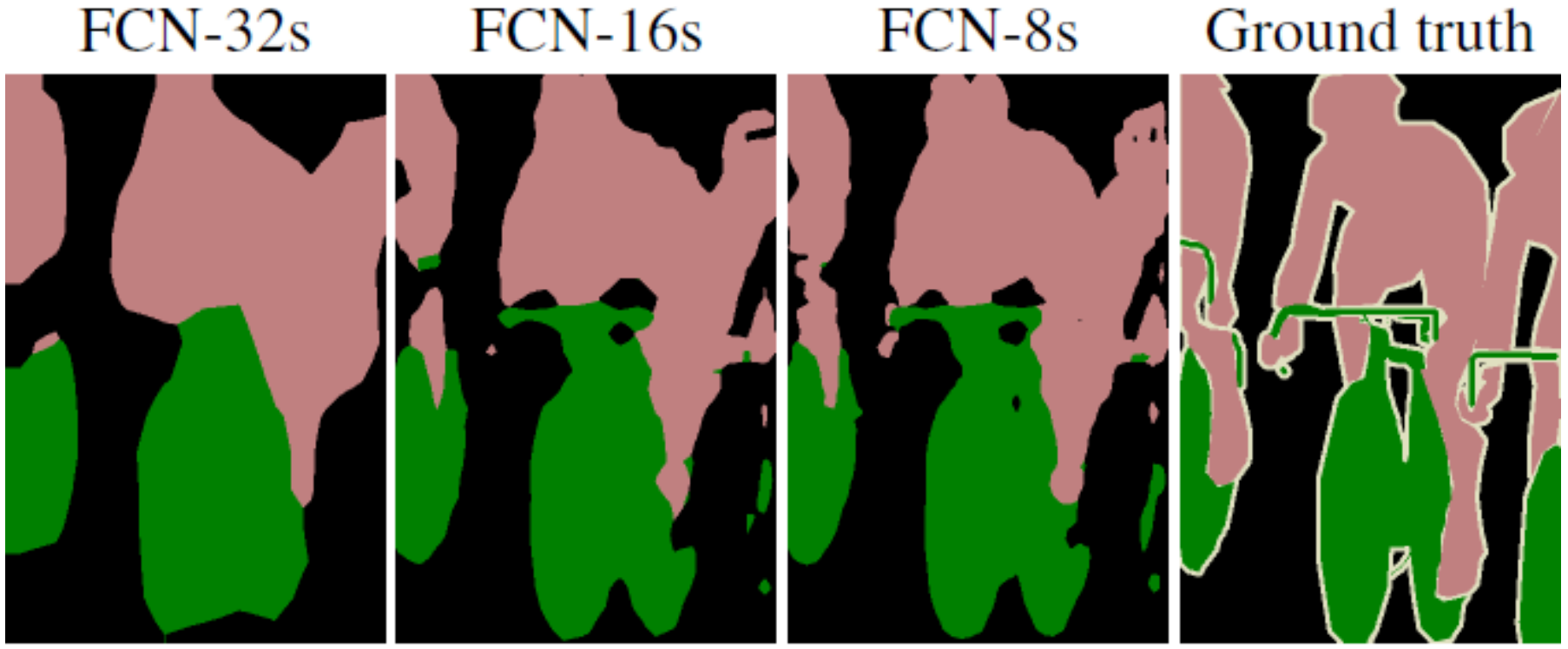


- **This yields 3 networks.** *Combining fine layers and coarse layers lets the model make local predictions that respect global structure.*
- Train first the lowest resolution network (FCN-32s)
- Then, the weights of the next network (FCN-16s) are initialized with (FCN-32s)
- The same for FCN-8s
- All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

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 the upsampling side of the network** and train these filters

FCNN Training Options

The «patch-based» way:

- The classification network is trained to minimize the classification loss ℓ over a mini-batch

$$\hat{\theta} = \min_{\theta} \sum_{\boldsymbol{x_j}} \ell(\boldsymbol{x_j}, \theta)$$

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

- 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

FCNN Training Options: The «full-image» way

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

Learning becomes:

$$\min_{x_j \in I} \sum_{x_j} \ell(x_j, \theta)$$

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

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 FCNN 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

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

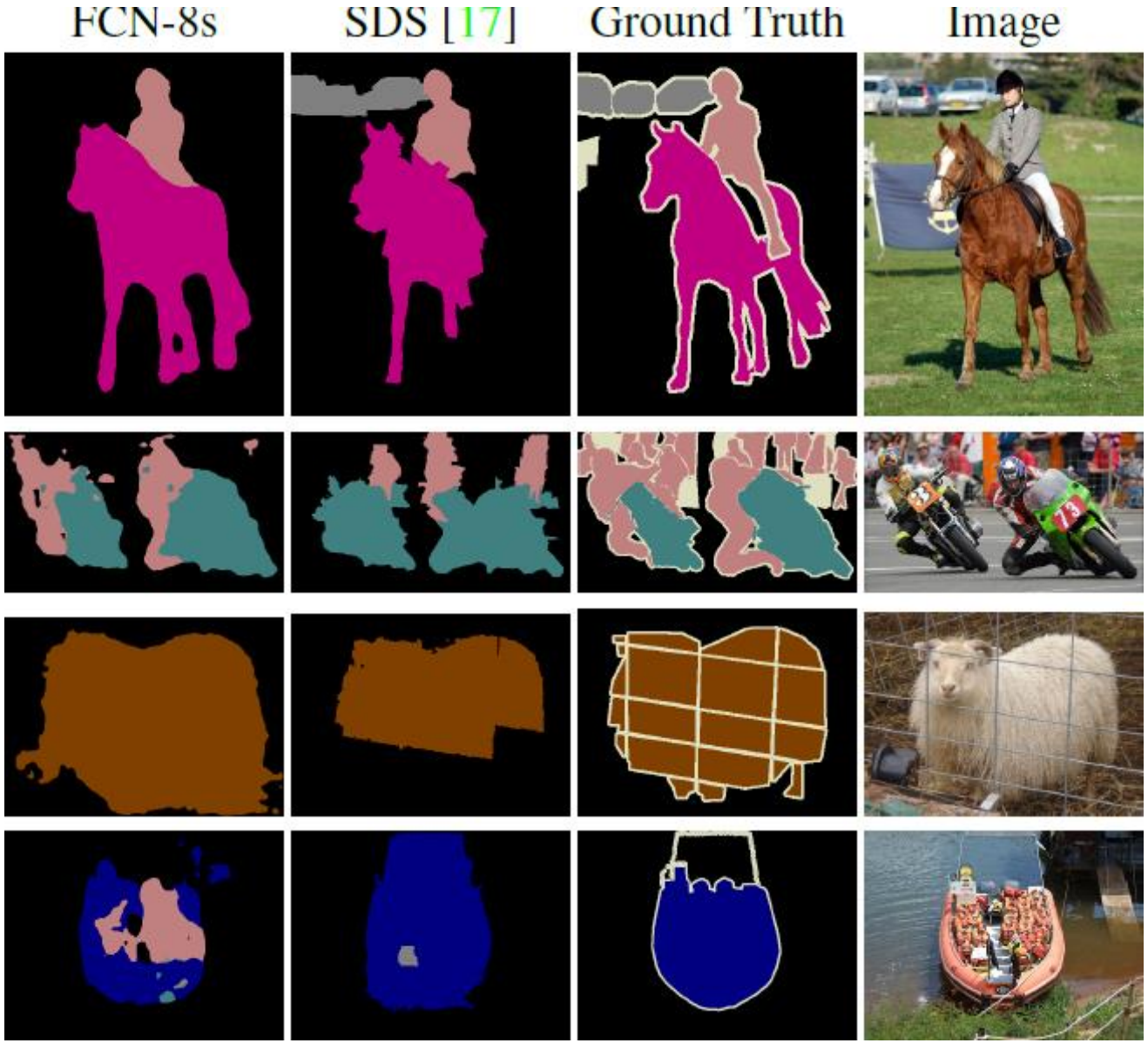
  being $M(\boldsymbol{x_j})$ a binary random variable

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

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

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

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**

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/

G. Boracchi

U-Net

Network formed by:

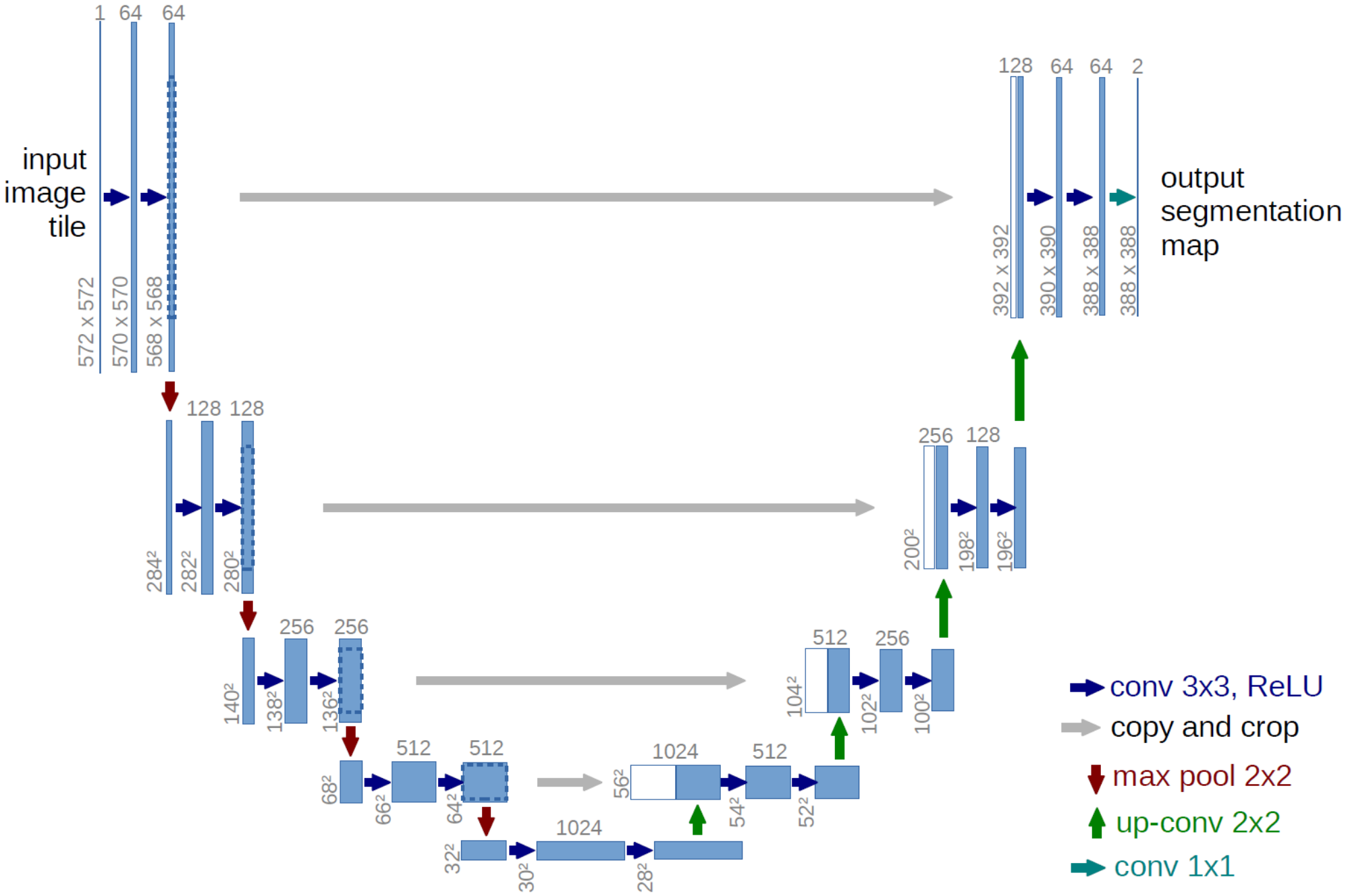
- A contracting path
- An expansive path

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.

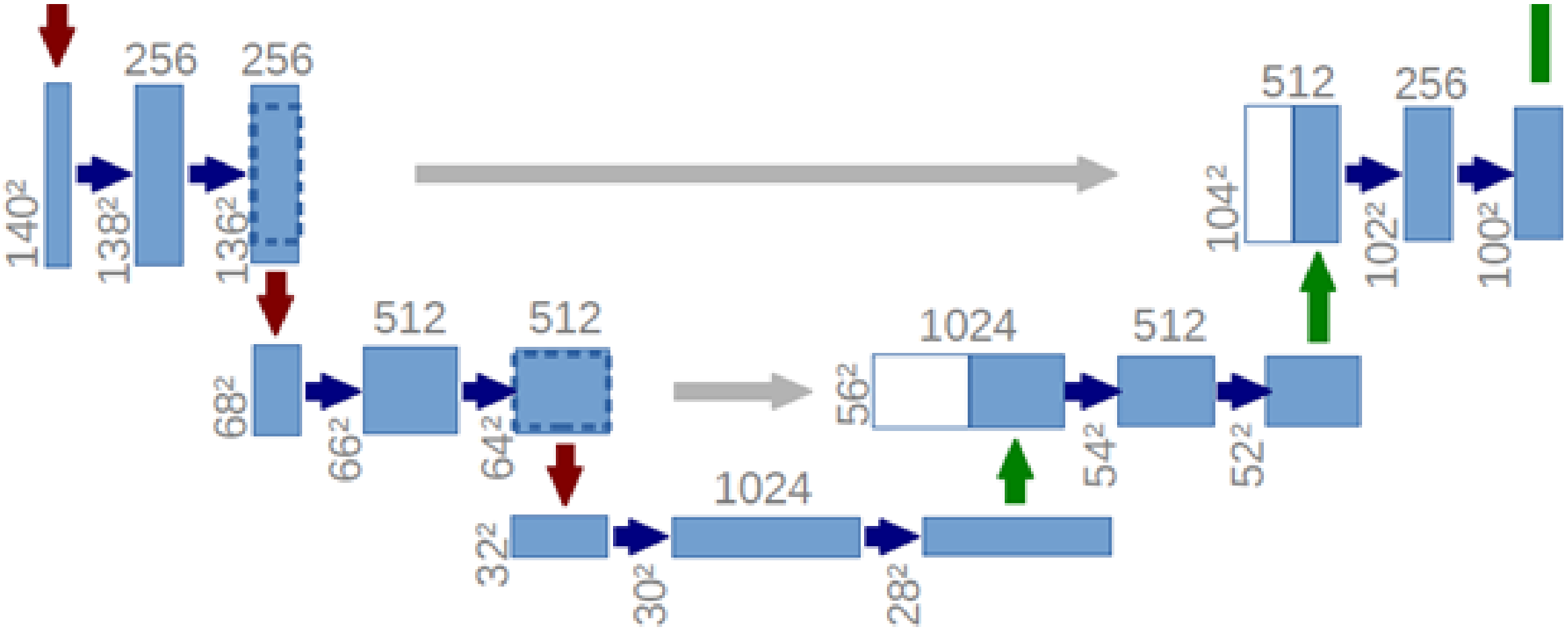
U-Net: Contracting path

Repeats blocks of:

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

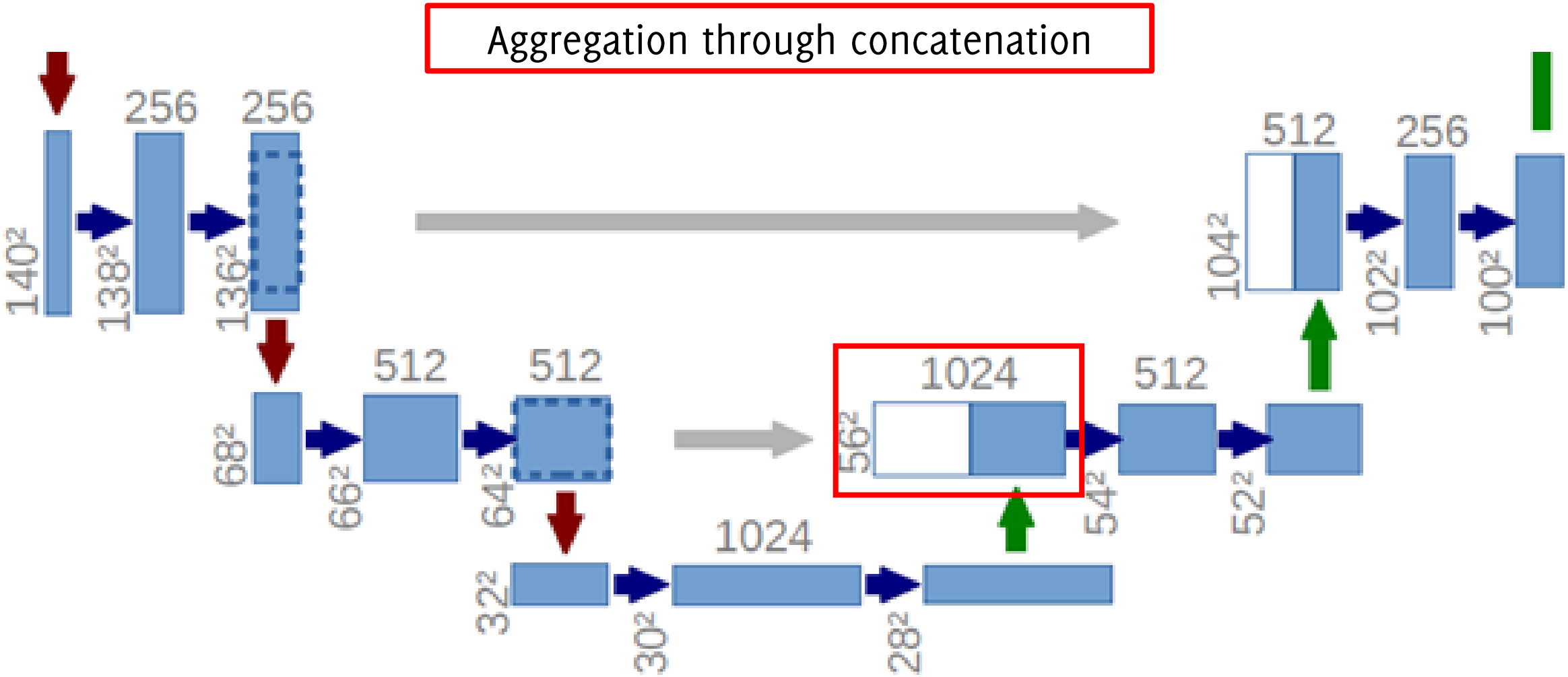
At each downsampling the number of feature maps is doubled

U-Net: Skip connections



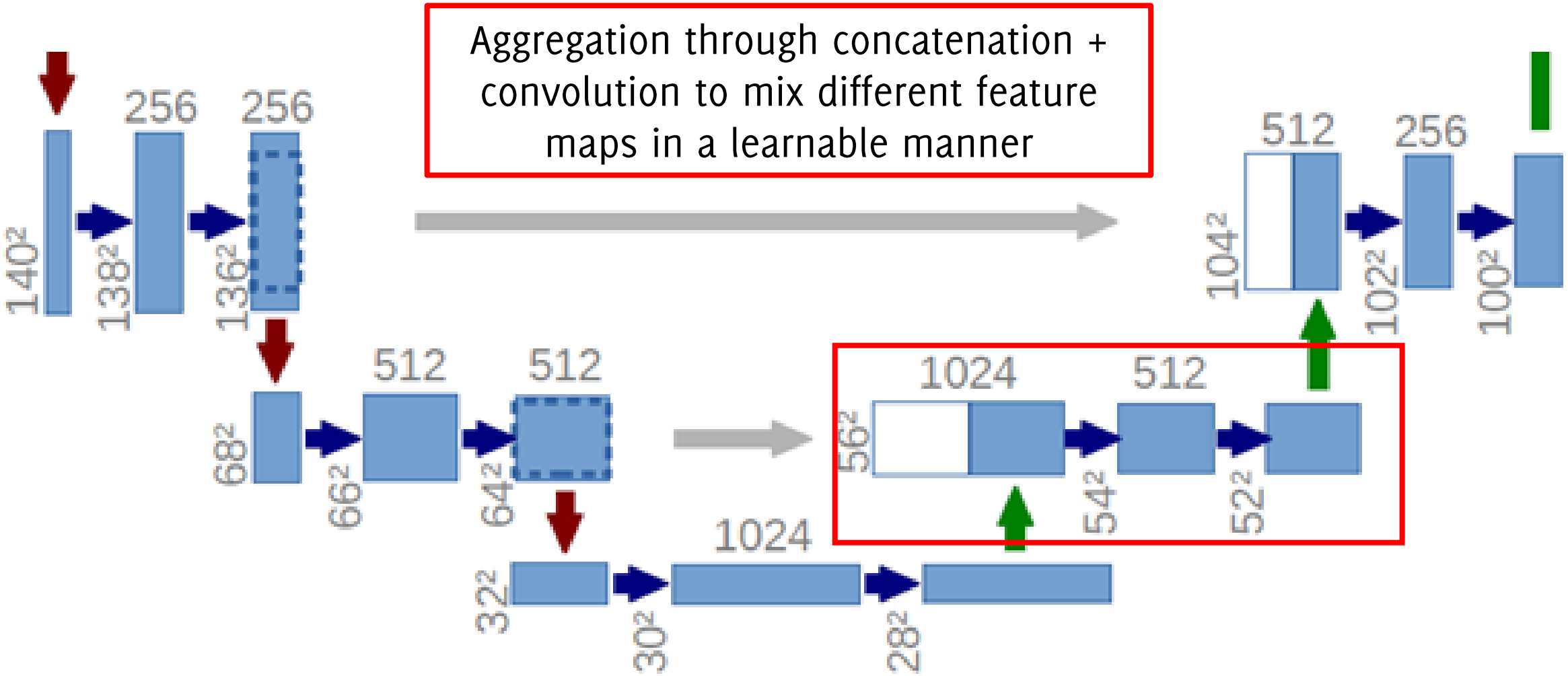
Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *MICCAI*, 2015.

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

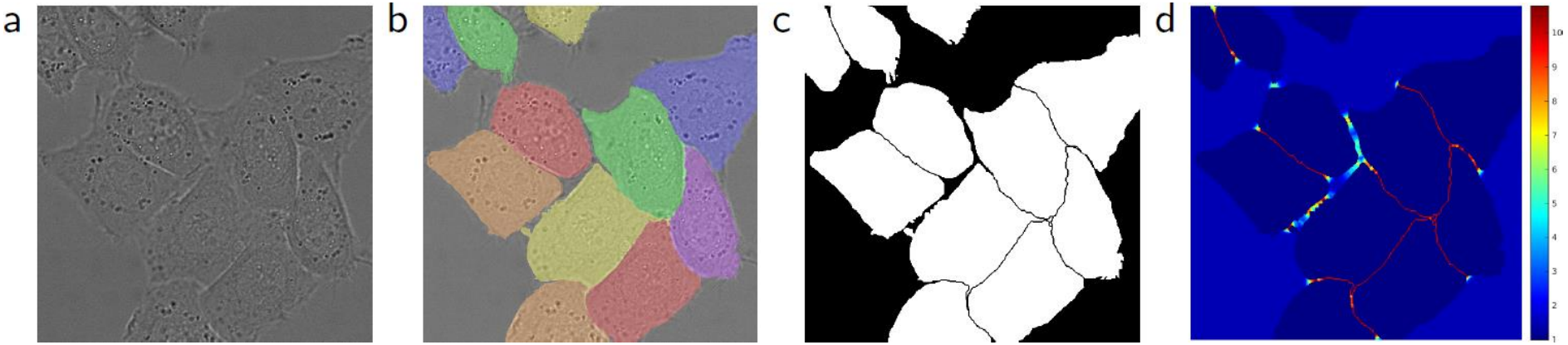
U-net: Training

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.

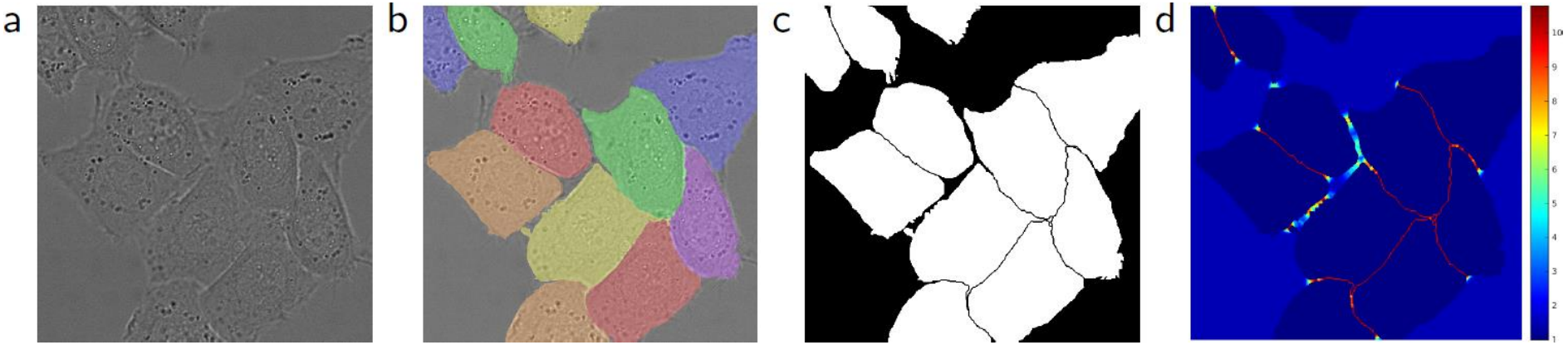
U-net: Training

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



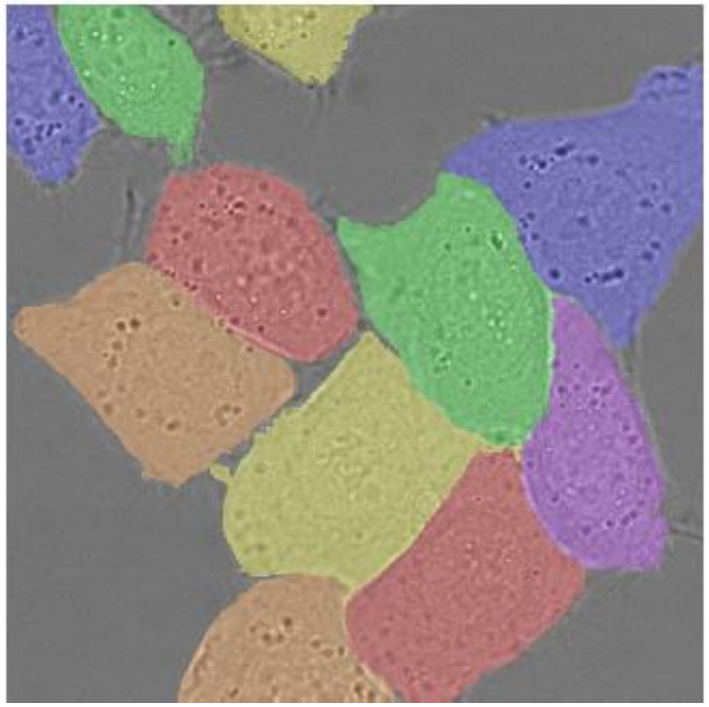
Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *MICCAI*, 2015.

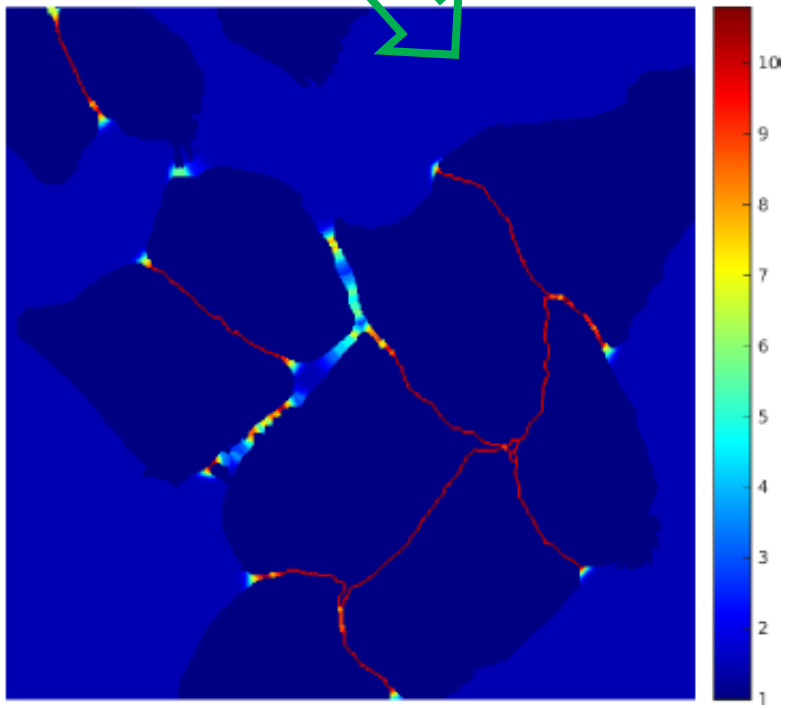
U-net: Training

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.

Global Averaging Pooling

G. Boracchi

Network In Network

Min Lin[1,2], Qiang Chen[2], Shuicheng Yan[2]
[1]Graduate School for Integrative Sciences and Engineering
[2]Department of Electronic & Computer Engineering
National University of Singapore, Singapore
{linmin,chenqiang,eleyans}@nus.edu.sg

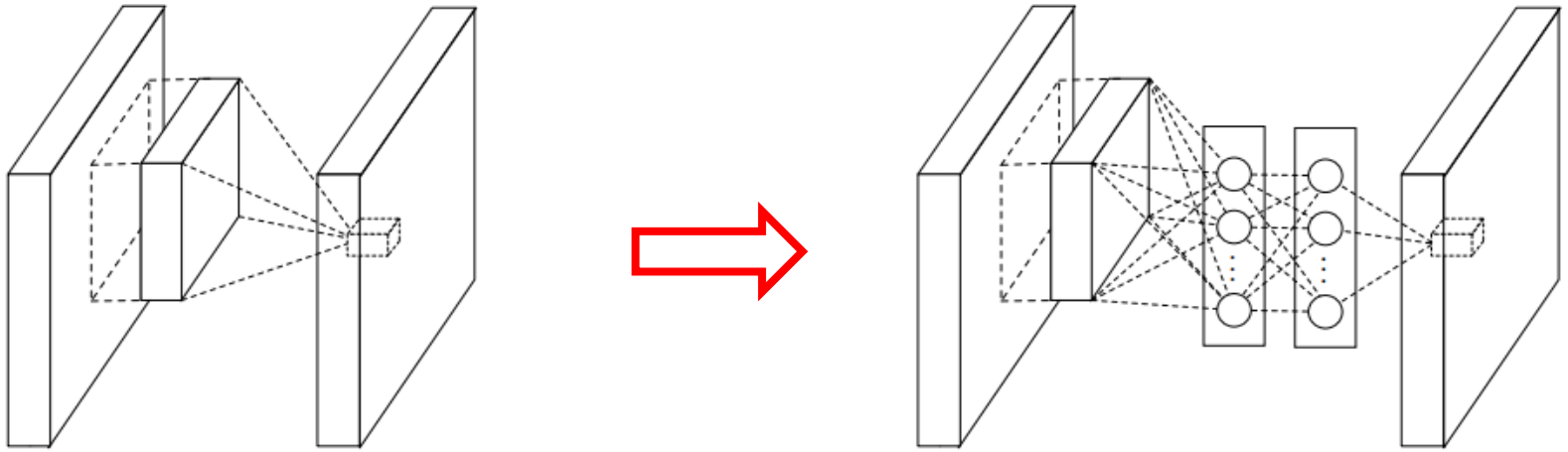
Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

Network in Network

Mlpconv layers: instead of traditional convolutions, **a stack of 1x1 convolutions + RELU**

- 1x1 convolutions used in a stack followed by RELU corresponds to a MLP networks **used in a sliding manner on the whole image**

Each layer features a **more powerful functional approximation** than a convolutional layer which is just linear + RELU

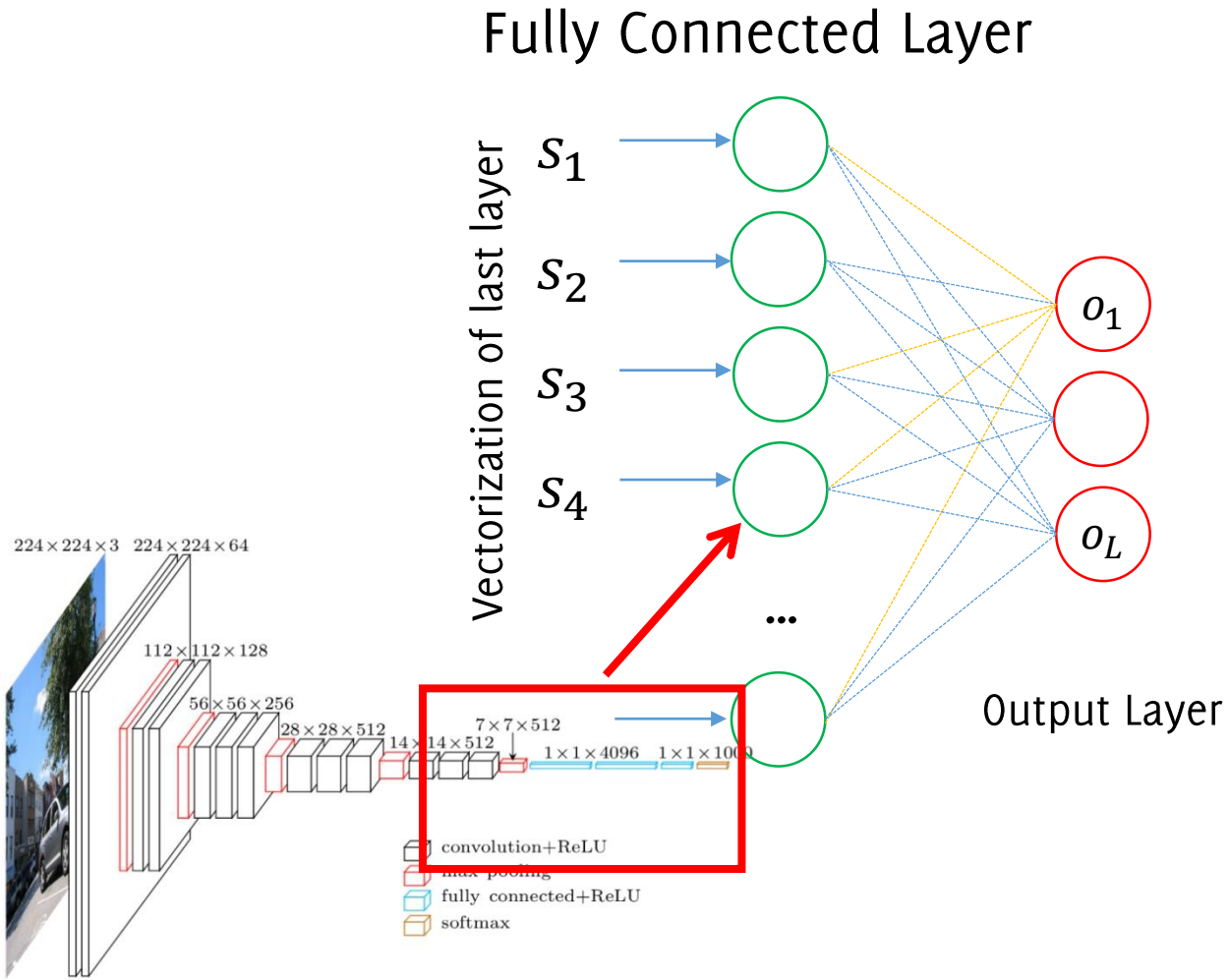


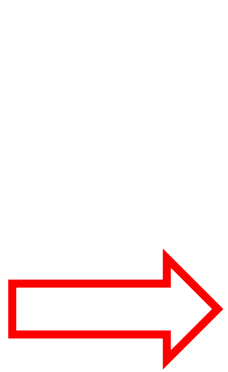
(a) Linear convolution layer (b) Mlpconv layer

Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

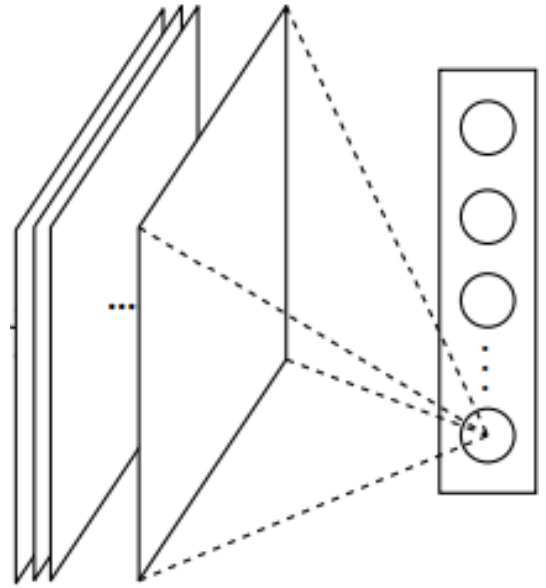
Network in Network

They introduce Global Averaging Pooling Layers





Global Averaging Pooling Layer

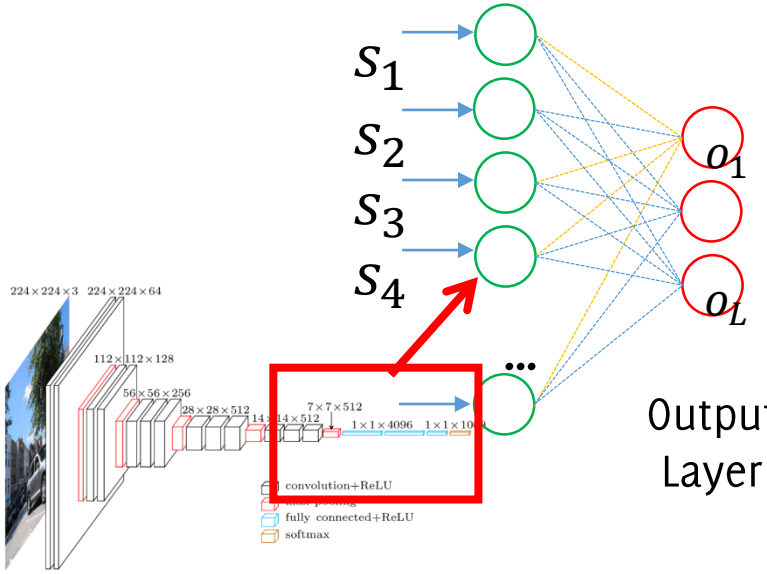


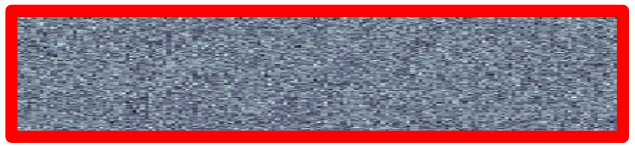
Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

Network in Network: GAP

Global Averaging Pooling Layers: instead of a FC layer at the end of the network, **compute the average of each feature map.**

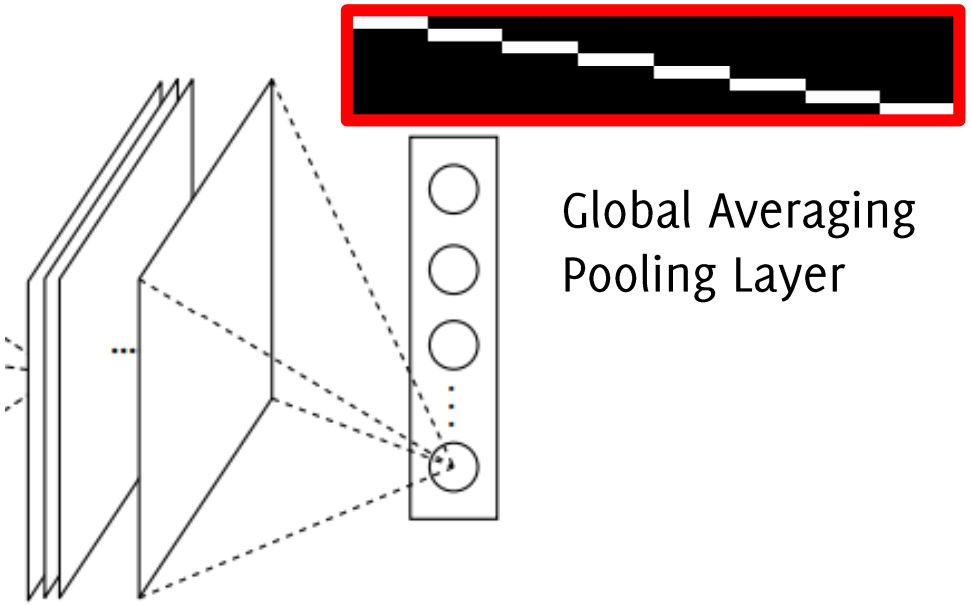
- The transformation corresponding **to GAP is a block diagonal, constant matrix** (consider the input unrolled layer-wise in a vector)
- The transformation of each layer in **MLP corresponds to a dense matrix.**





Fully Connected Layer

Output Layer



Global Averaging Pooling Layer

Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

Rationale behind GAP

Fully connected layers are prone to overfitting

- They have many parameters
- Dropout was proposed as a regularized that randomly sets to zero a percentage of activations in the FC layers during training

The GAP strategy is:

- **Remove the fully connected layer** at the end of the network!
- **Predict by a simple soft-max after the GAP**.
- **Watch out: the number of feature maps has to be equal to the number of output classes!**

The Advantages of GAP Layers:

- No parameters to optimize, lighter networks less prone to overfitting
- More interpretability, creates a direct connection between layers and classes output (we'll see soon)
- This makes GAP a structural regularizer
- More robustness to spatial transformation of the input images
- **The network can be used to classify images of different sizes**
- Classification is performed by a softMax layer at the end of the GAP

Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." ICLR 2014

Network in Network

The whole NiN stacks

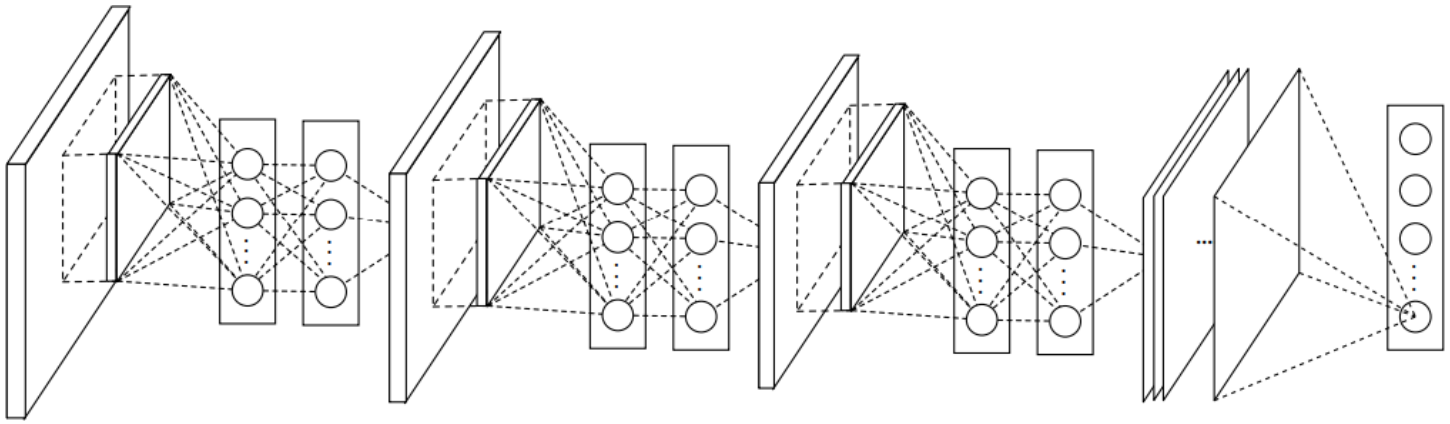
- mlpconv layers (RELU) + dropout
- Maxpooling

} A few layers of these

- Global Averaging Pooling (GAP) layer
- Softmax

} At the end of the network

simple NiNs achieve state-of-the-art performance on «small» datasets (CIFAR10, CIFAR100, SVHN, MNIST) and that **GAP effectively reduces overfitting** w.r.t. FC

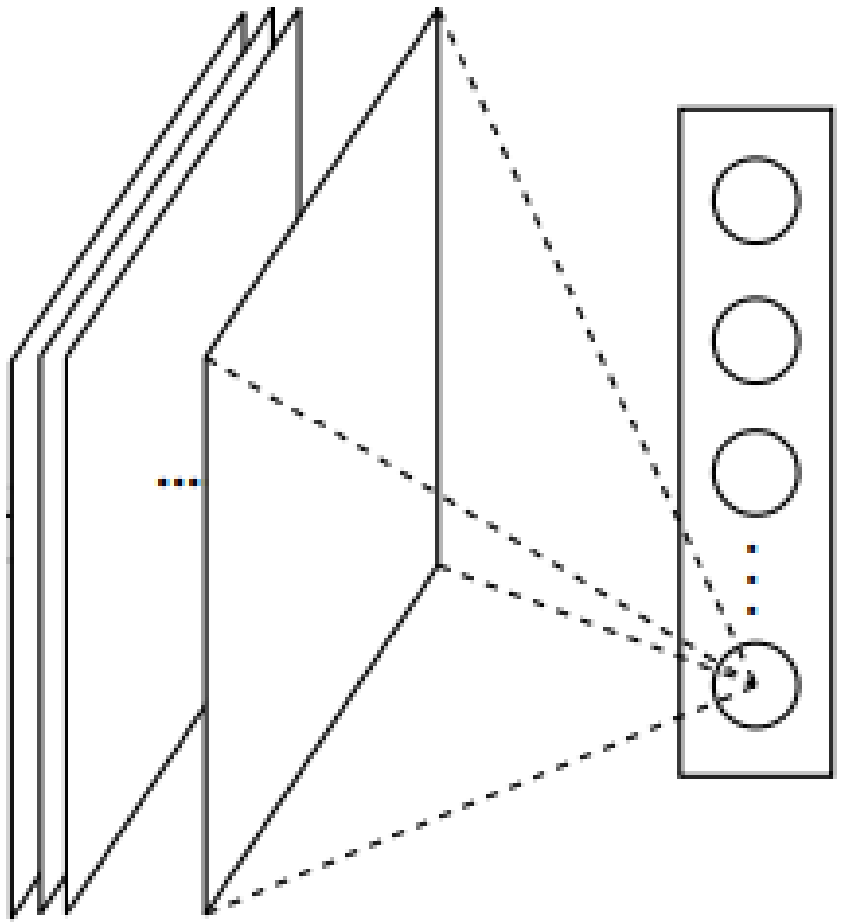


Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

The Global Averaging Pooling (GAP) Layer

We indeed see that GAP is acting as a (structural) regularizer

Method	Testing Error
mlpconv + Fully Connected	11.59%
mlpconv + Fully Connected + Dropout	10.88%
mlpconv + Global Average Pooling	10.41%



Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *arXiv preprint arXiv:1312.4400v3* (2014).

Weakly-Supervised Localization

... Global Averaging Pooling Revisited

G. Boracchi

Weakly supervised localization

Perform localization over an image without images with annotated bounding box

- Training set provided as for classification with image-label pairs (I, ℓ) where no localization information is provided

Learning Deep Features for Discriminative Localization

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba
Computer Science and Artificial Intelligence Laboratory, MIT
{bzhou,khosla,agata,oliva,torralba}@csail.mit.edu

G. Boracchi

The GAP revisited

The advantages of GAP layer **extend beyond** simply acting as a structural **regularizer** that prevents overfitting

In fact, the **network can retain a remarkable localization ability** until the final layer. By a simple tweak it is possible to easily identify the discriminative image regions leading to a prediction.

*A **CNN trained on object categorization** is successfully able to **localize the discriminative regions for action classification** as the objects that the humans are interacting with rather than the humans themselves*

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

Class Activation Mapping

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

Class Activation Mapping (CAM)

Identifying exactly **which regions** of an image **are being used for discrimination**.

CAM are very easy to compute. It just requires:

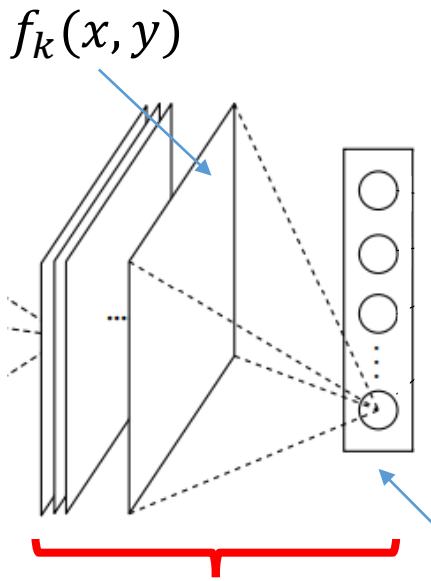
- FC layer after the GAP
- a minor tweak

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

The Global Averaging Pooling (GAP) Layer

A very simple architecture made only of convolutions and activation functions leads to a final layer having:

- n feature maps $f_k(\cdot,\cdot)$ having resolution "similar" to the input image
- a vector after GAP made of n averages F_k



$$F_k = \sum_{(x,y)} f_k(x,y)$$

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

The Global Averaging Pooling (GAP) Layer

Add (and train) a **single FC layer** after the GAP.

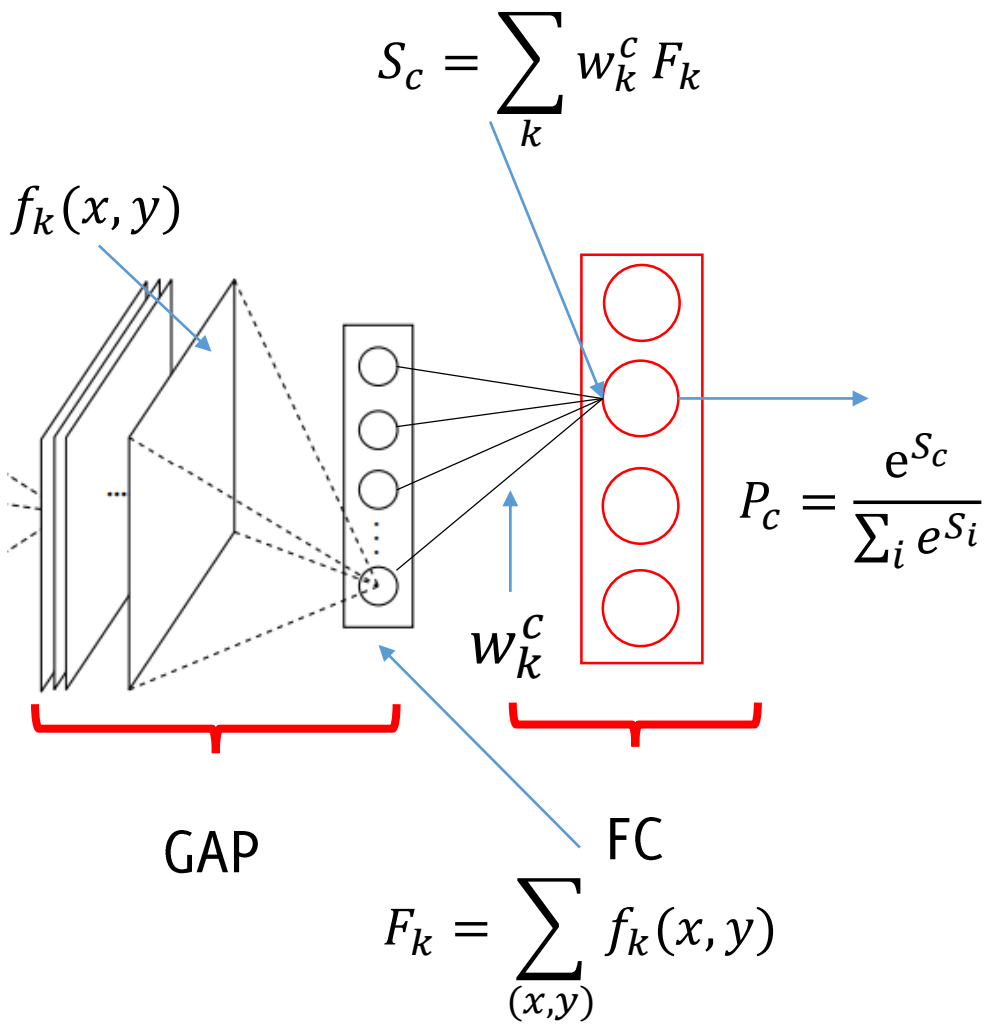
The FC computes S_c for each class c as the weighted sum of $\{F_k\}$, where weights are defined during training

Then, the class probability P_c via soft-max (class c)

Remark: when computing

$$S_c = \sum_k w_k^c F_k$$

w_k^c encodes the importance of F_k for the class c.



Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

The Global Averaging Pooling (GAP) Layer

However

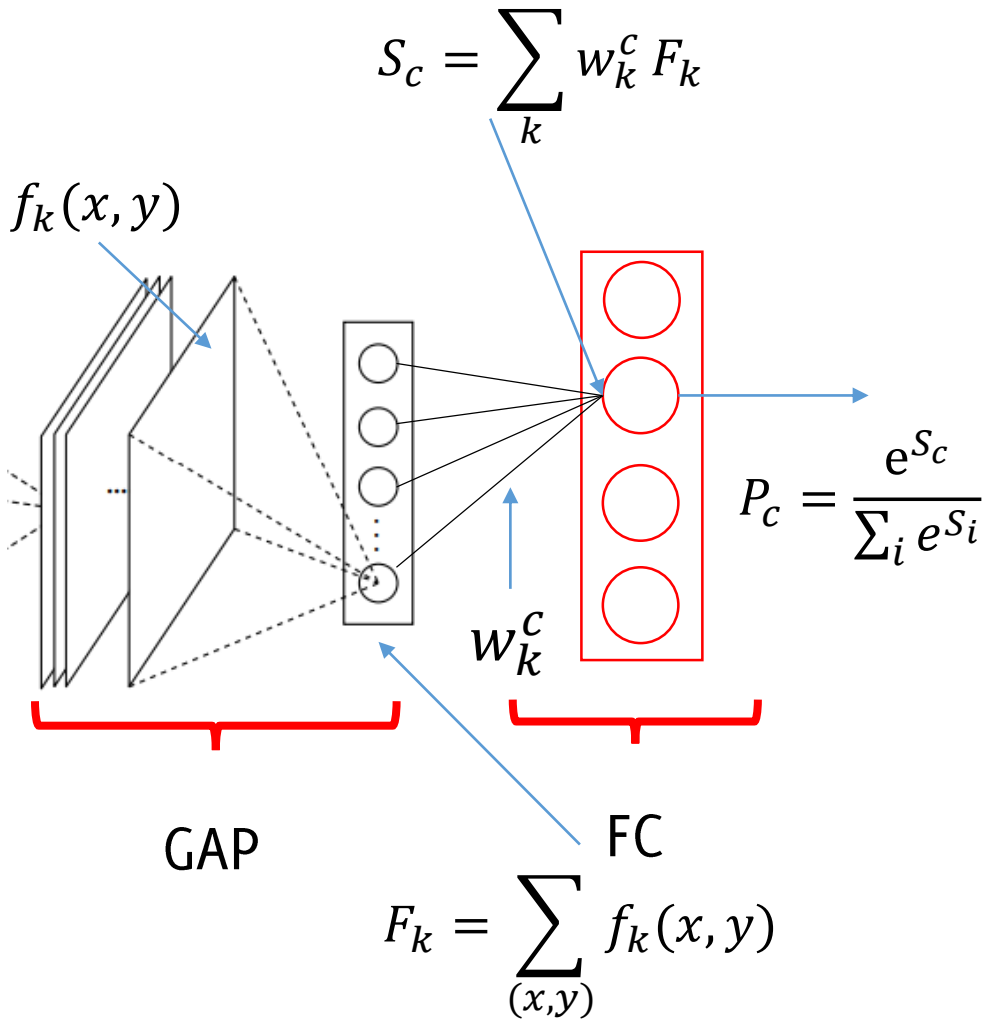
$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c \, f_k(x,y)$$

And CAM is defined as

$$M_c(x,y) = \sum_k w_k^c \, f_k(x,y)$$

where $M_c(x,y)$ directly indicates the importance of the activations at (x,y) for predicting the class c

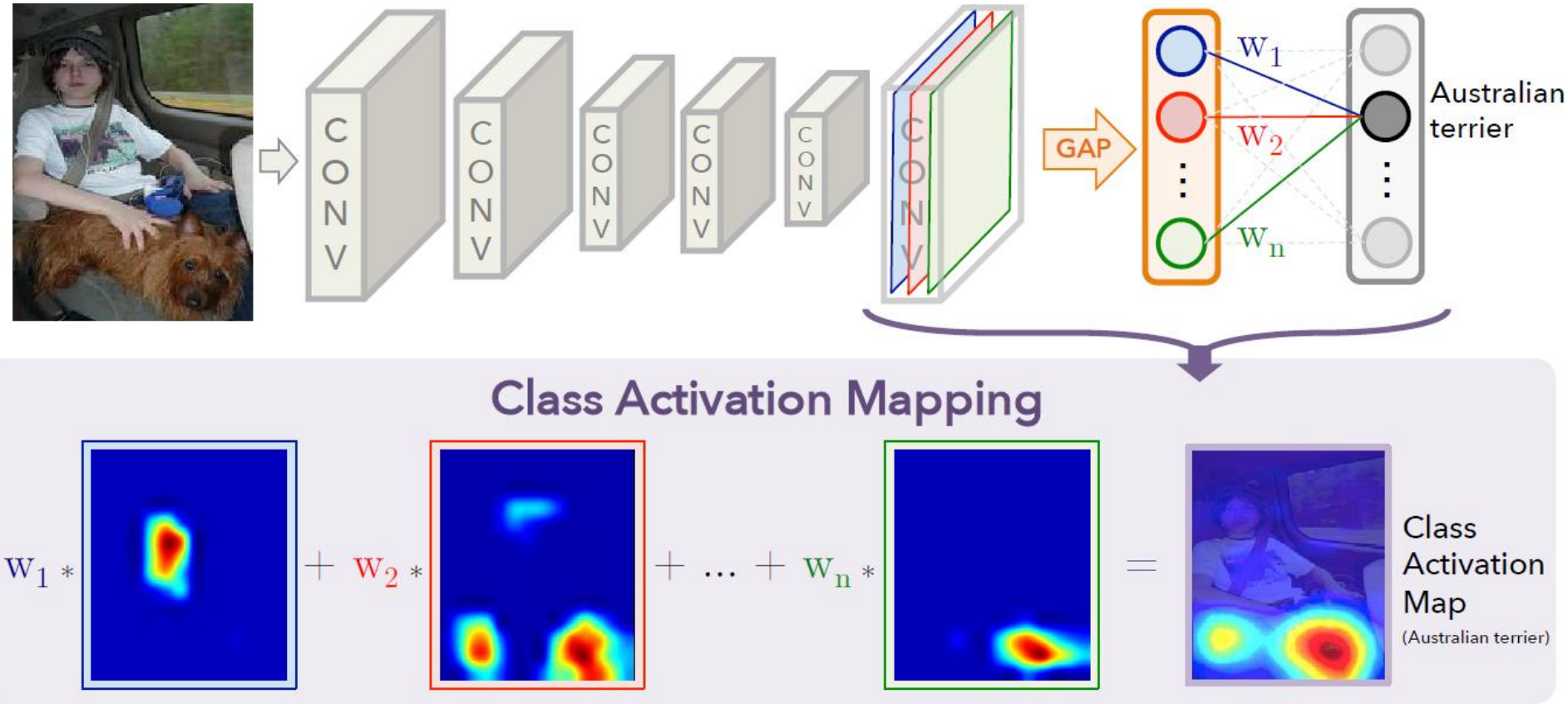
Thanks to the softmax, the depth of the last convolutional volume can be different from the number of classes



Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

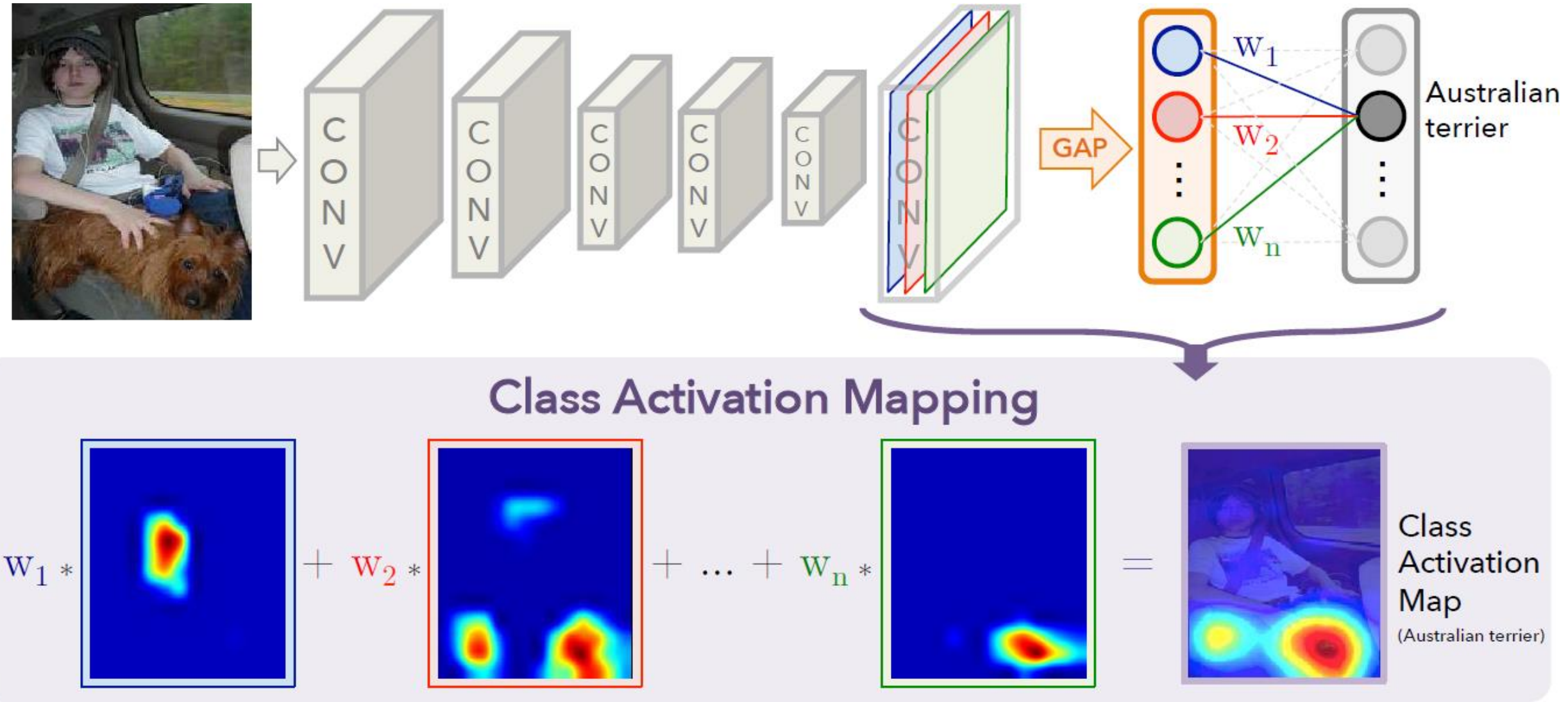
Class Activation Mapping

Now, the weights represents the importance of each feature map to yield the final prediction. Upsampling might be necessary to match the input image



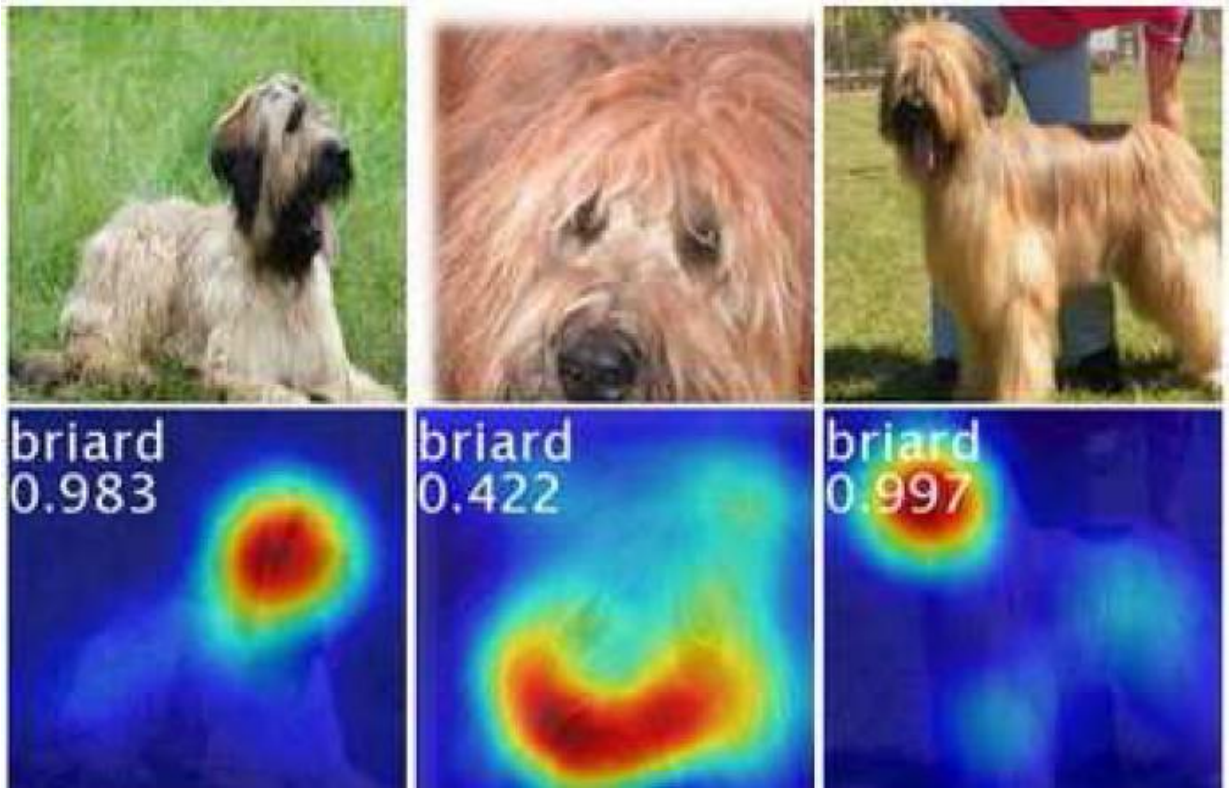
Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

Class Activation Mapping



Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

Class Activation Mapping



Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

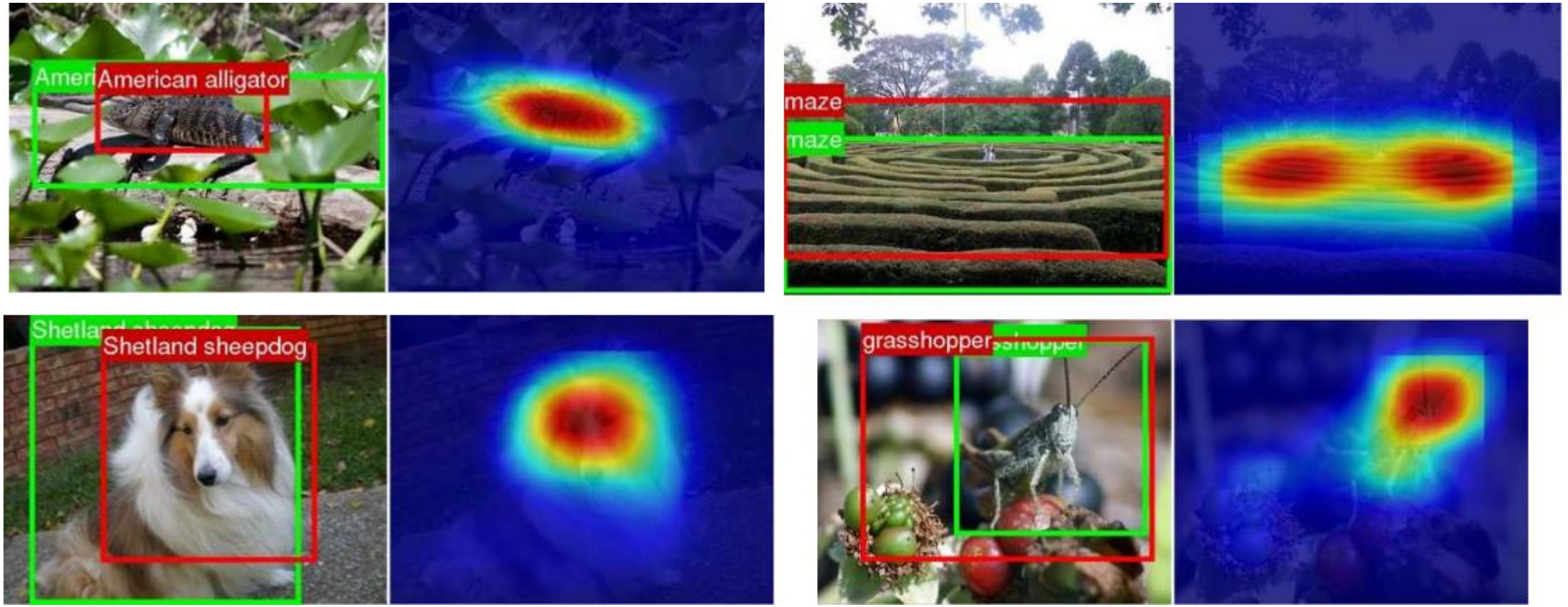
Remarks

- CAM can be **included in any pre-trained network**, as long as all the FC layers at the end are removed
- The **FC used for CAM is simple**, few neurons and no hidden layers
- **Classification performance might drop** (in VGG removing FC means loosing 90% of parameters)
- **CAM resolution** (localization accuracy) can improve by «**anticipating**» GAP to larger convolutional feature maps (but this reduces the semantic information within these layers)
- **GAP**: encourages the **identification of the whole object**, as all the parts of the values in the activation map concurs to the classification
- **GMP (Global Max Pooling)**: it is enough to have a high maximum, thus **promotes specific discriminative features**

Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

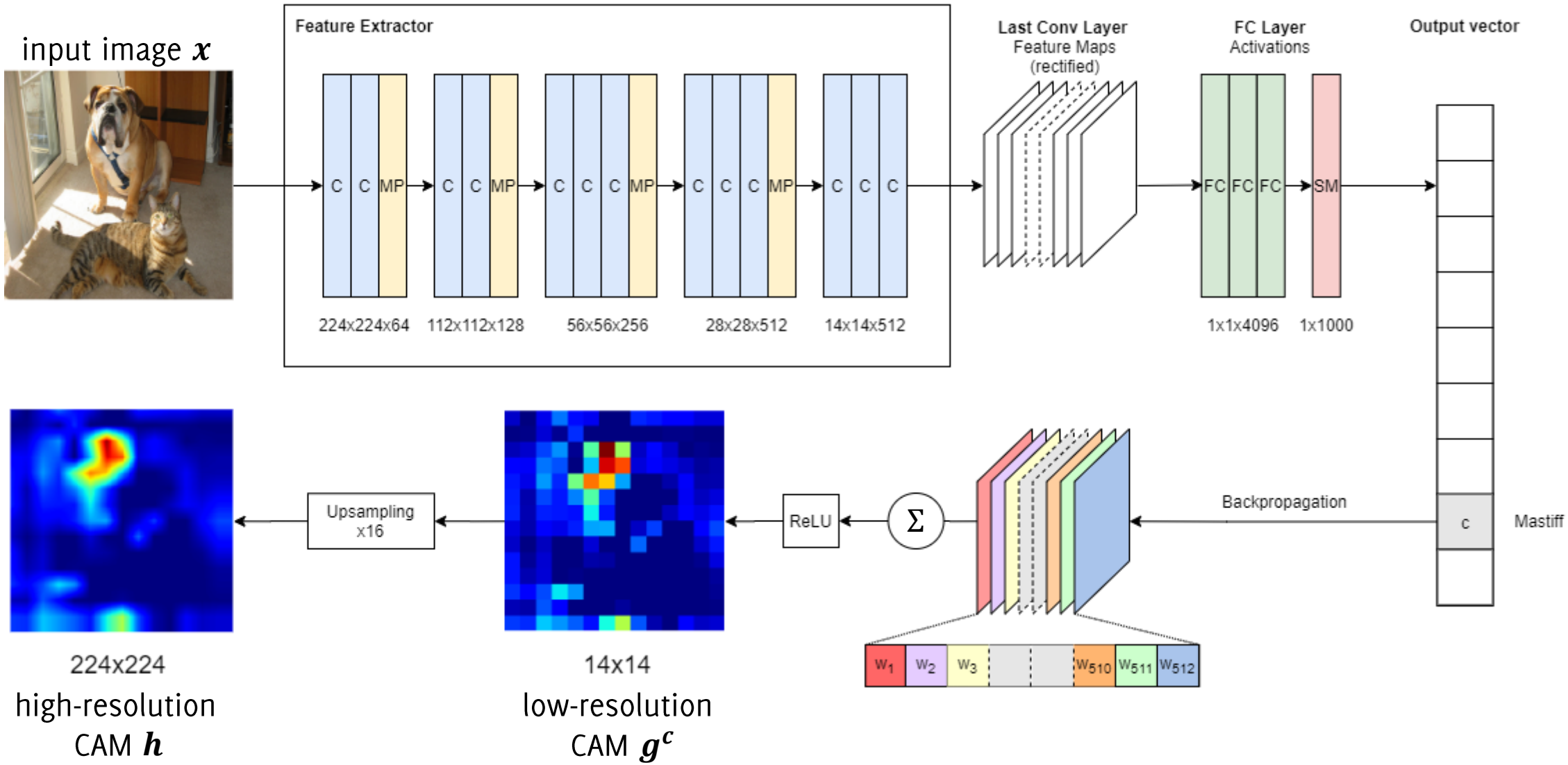
Weakly Supervised Localization

Use thresholding CAM values: > 20% max(CAM), then take the largest componet of the thresholded map (green GT, red estimated location)



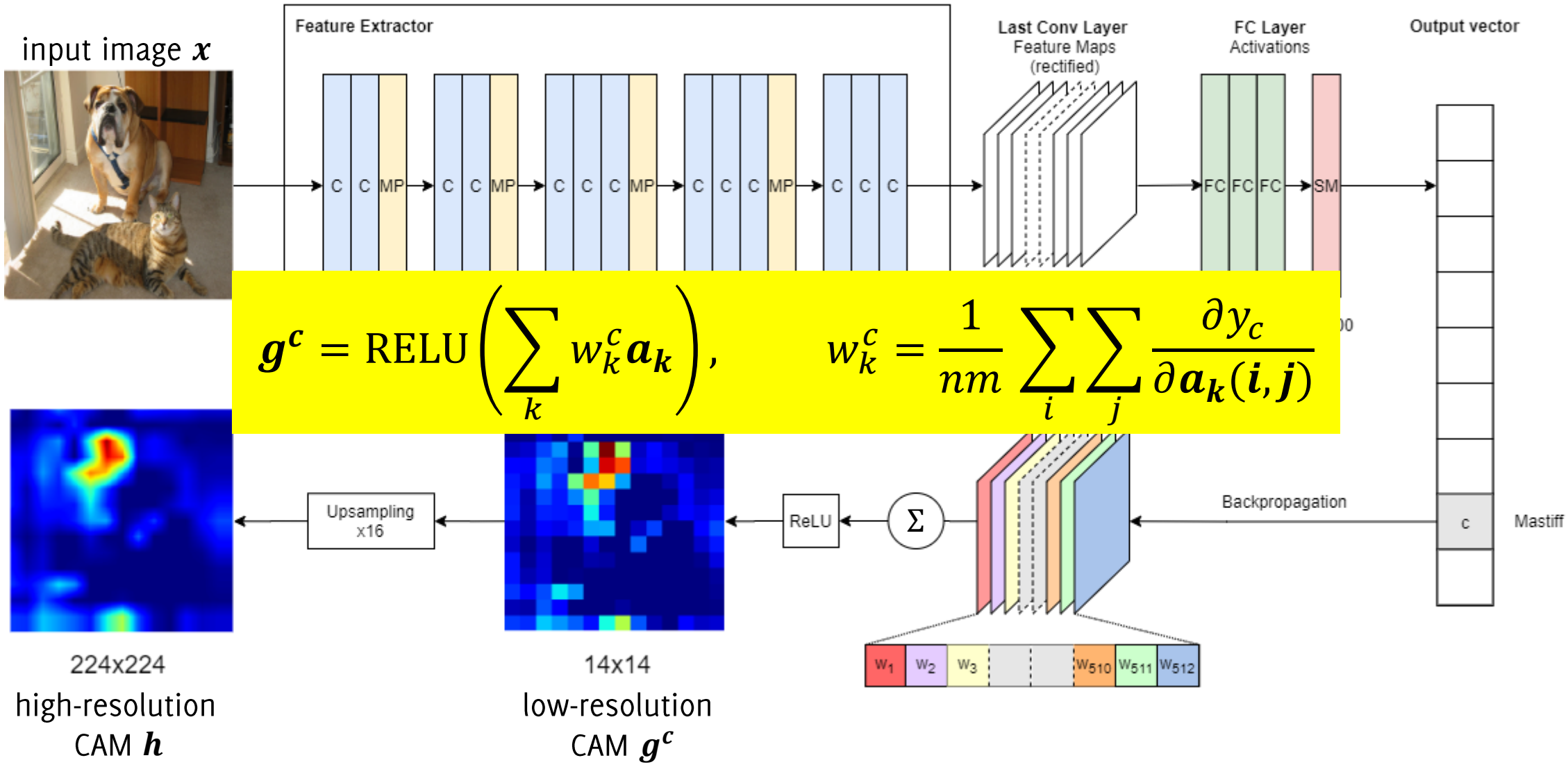
Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016

Grad-CAM and CAM-based techniques



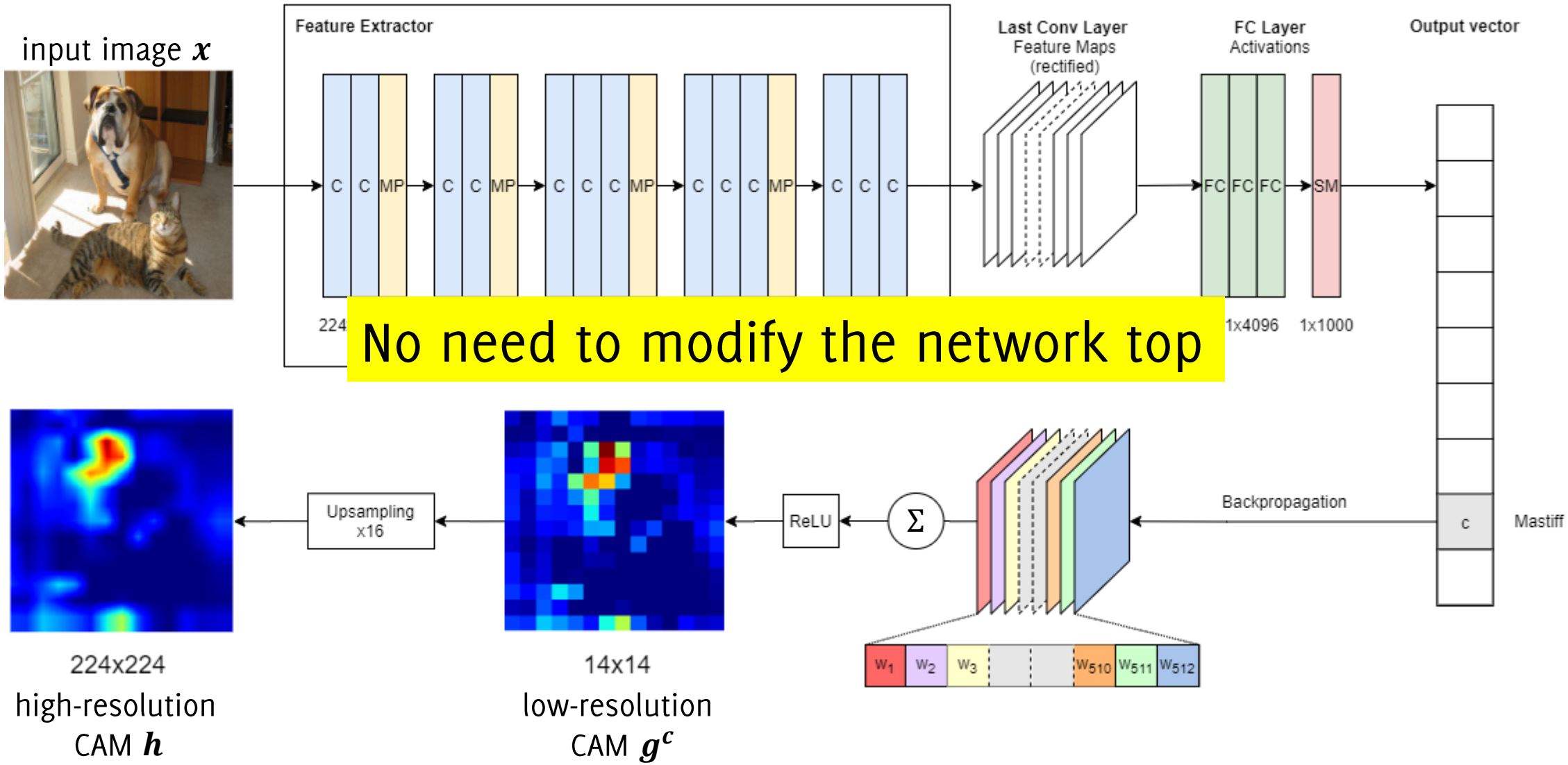
Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV 2017*
Type equation here.

Grad-CAM and CAM-based techniques



Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *ICCV2017*

Grad-CAM and CAM-based techniques



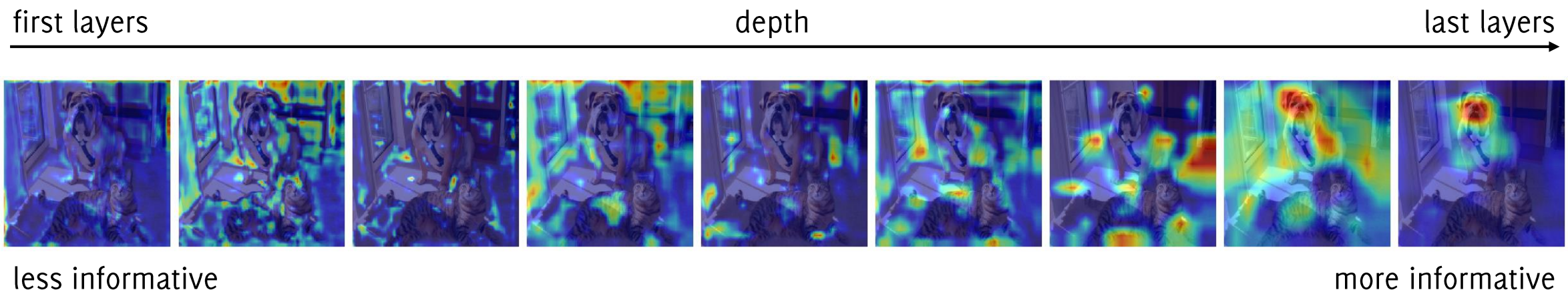
Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *ICCV2017*

Heatmaps Desiderata

Should be **class discriminative**

Should **capture fine-grained details** (high-resolution)

- This is critical in many applications (*e.g.* medical imaging, industrial processes)

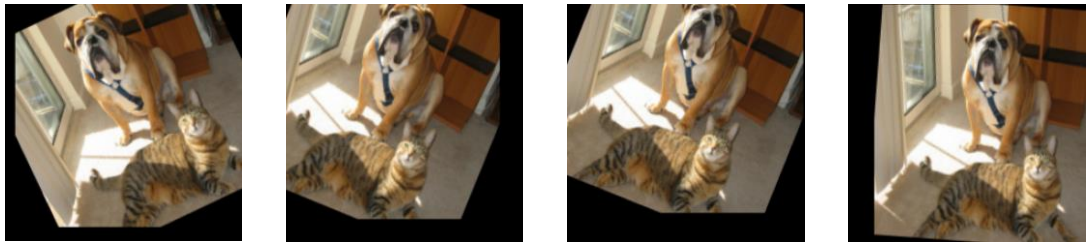


Augmented Grad-CAM

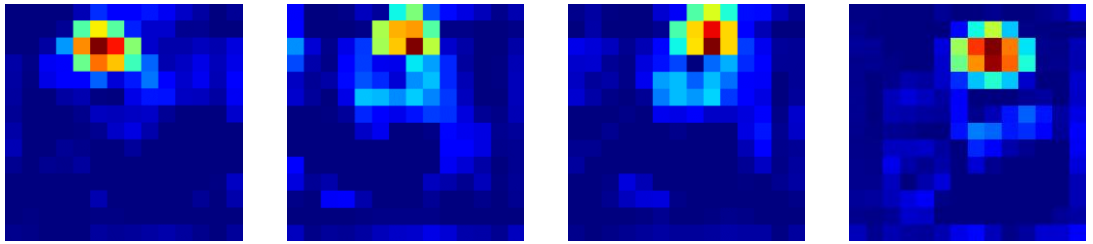
We consider the augmentation operator $\mathcal{A}_l: \mathbb{R}^{N \times M} \rightarrow \mathbb{R}^{N \times M}$, including random rotations and translations of the input image $\boldsymbol{x}$

Augmented Grad-CAM: increase heat-maps resolution through image augmentation

All the responses that the CNN generates to the **multiple augmented versions of the same input image** are very informative for reconstructing the high-resolution heat-map $\boldsymbol{h}$



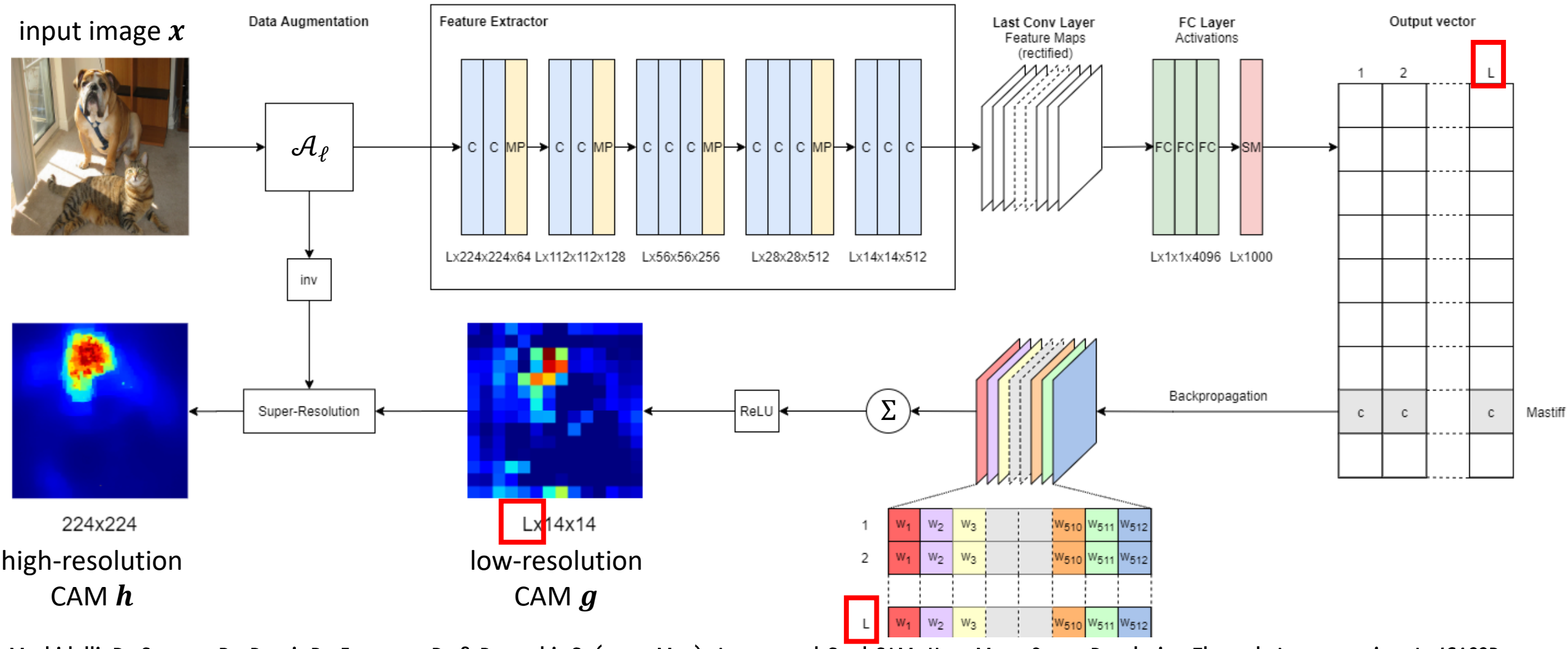
$x_1 = \mathcal{A}_1(x)$ $x_2 = \mathcal{A}_2(x)$ $x_3 = \mathcal{A}_3(x)$ $x_4 = \mathcal{A}_4(x)$



g_1 g_2 g_3 g_4

Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2020, May). Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation. In *ICASSP 2020*

Augmented Grad-CAM

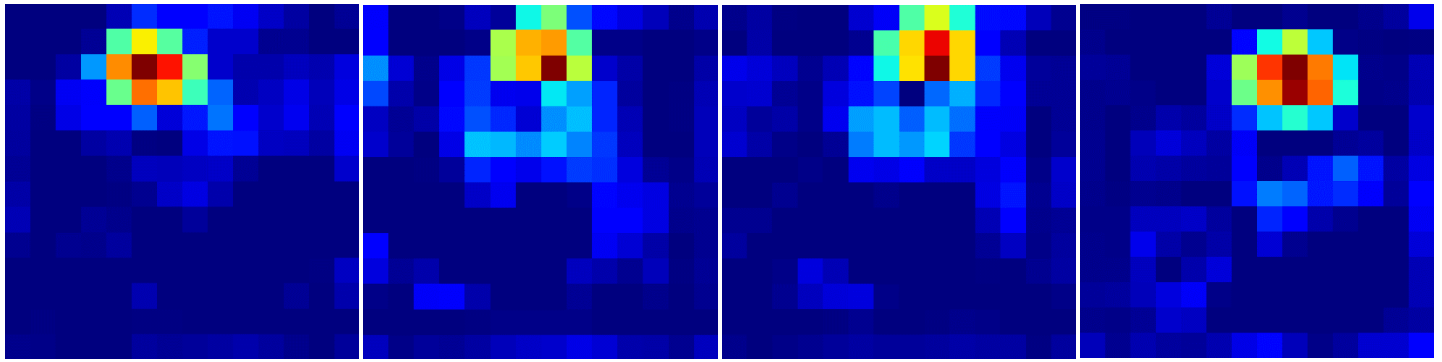


Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2020, May). Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation. In *ICASSP 2020*

The Super-Resolution Approach

We perform heat-map Super-Resolution (**SR**) by taking advantage of the information shared in multiple low-resolution heat-maps computed from the **same input under different – but known – transformations**

CNNs are in general invariant to roto-translations, in terms of predictions, but each $\boldsymbol{g}_\ell$ actually contains different information



g_1 g_2 g_3 g_4

General approach, our SR framework can be combined with any visualization tool (not only Grad-CAM)

The Super-Resolution Formulation

We model heat-maps computed by Grad-CAM as the result of an unknown downsampling operator $\boldsymbol{\mathcal{D}} : \mathbb{R}^{N \times M} \rightarrow \mathbb{R}^{n \times m}$

The high-resolution heat-map $\boldsymbol{h}$ is recovered by solving an inverse problem

$$\underset{h}{\operatorname{argmin}} \frac{1}{2} \sum_{l=1}^{L} \|\mathcal{D}\mathcal{A}_\ell h - g_\ell\|_2^2 + \lambda TV_{\ell_1}(h) + \frac{\mu}{2}\|h\|_2^2 \quad (1)$$

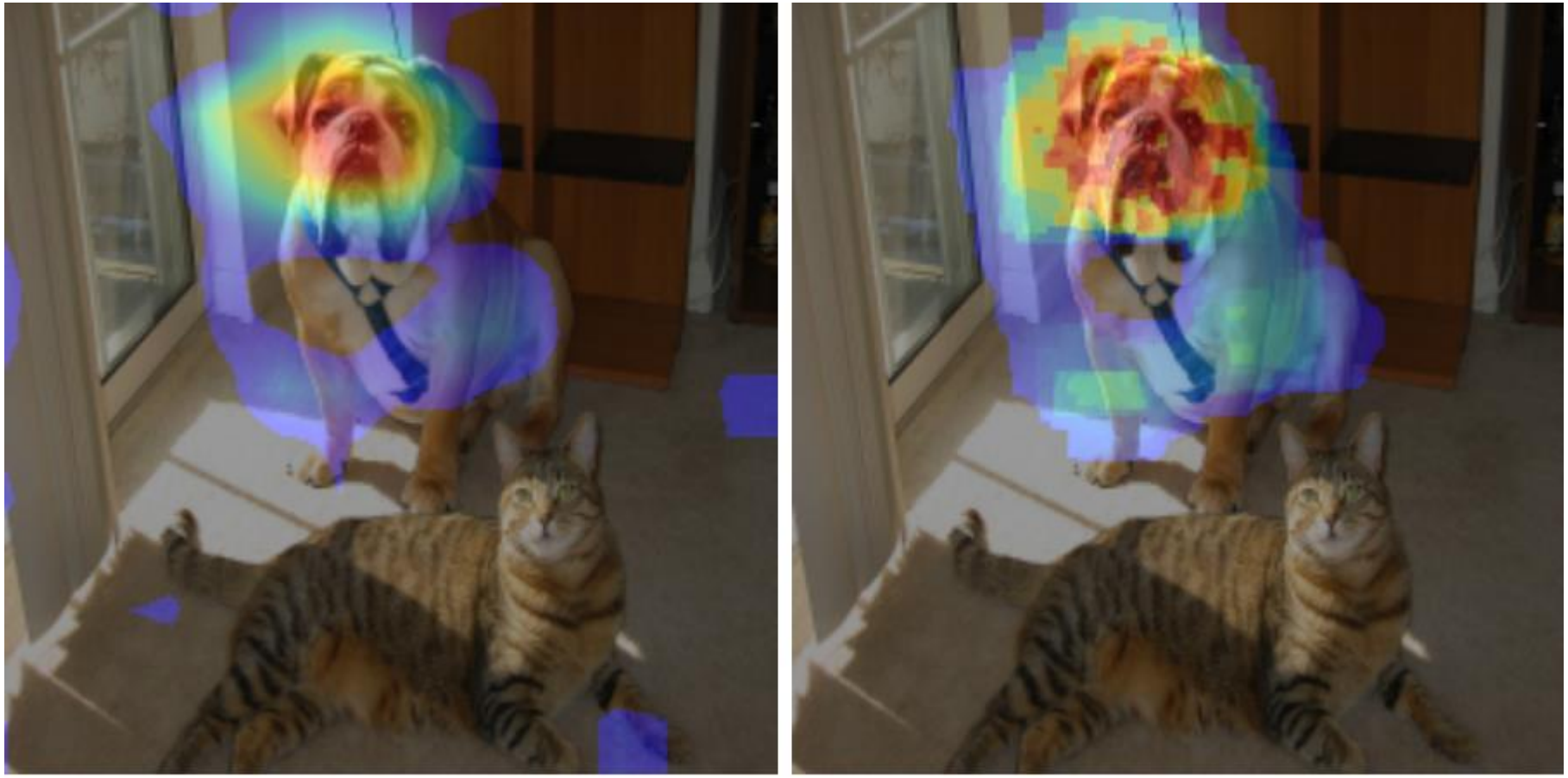
TV_{ℓ_1}: Anistropic Total Variation regularization is used to preserve the edges in the target heat-map (high-resolution)

$$TV_{\ell_1}(\boldsymbol{h}) = \Sigma_{i,j} \|\partial_x \boldsymbol{h}(i,j)\| + \|\partial_y \boldsymbol{h}(i,j)\| \quad (2)$$

This is solved through Subgradient Descent since the function is convex and non-smooth

Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G “Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation”. ICASSP 2020

Augmented Grad-CAM



(a) Grad-CAM. (b) Augmented Grad-CAM.

Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2020, May). Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation. In *ICASSP 2020*

Other popular architectures

Outline

Lecture inspired to:

Notes accompanying the Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

… and papers cited in here!

Going Deeper with Convolutions

Christian Szegedy[1], Wei Liu[2], Yangqing Jia[1], Pierre Sermanet[1], Scott Reed[3],
Dragomir Anguelov[1], Dumitru Erhan[1], Vincent Vanhoucke[1], Andrew Rabinovich[4]
[1]Google Inc. [2]University of North Carolina, Chapel Hill
[3]University of Michigan, Ann Arbor [4]Magic Leap Inc.

[1]{szegedy,jiayq,sermanet,dragomir,dumitru,vanhoucke}@google.com

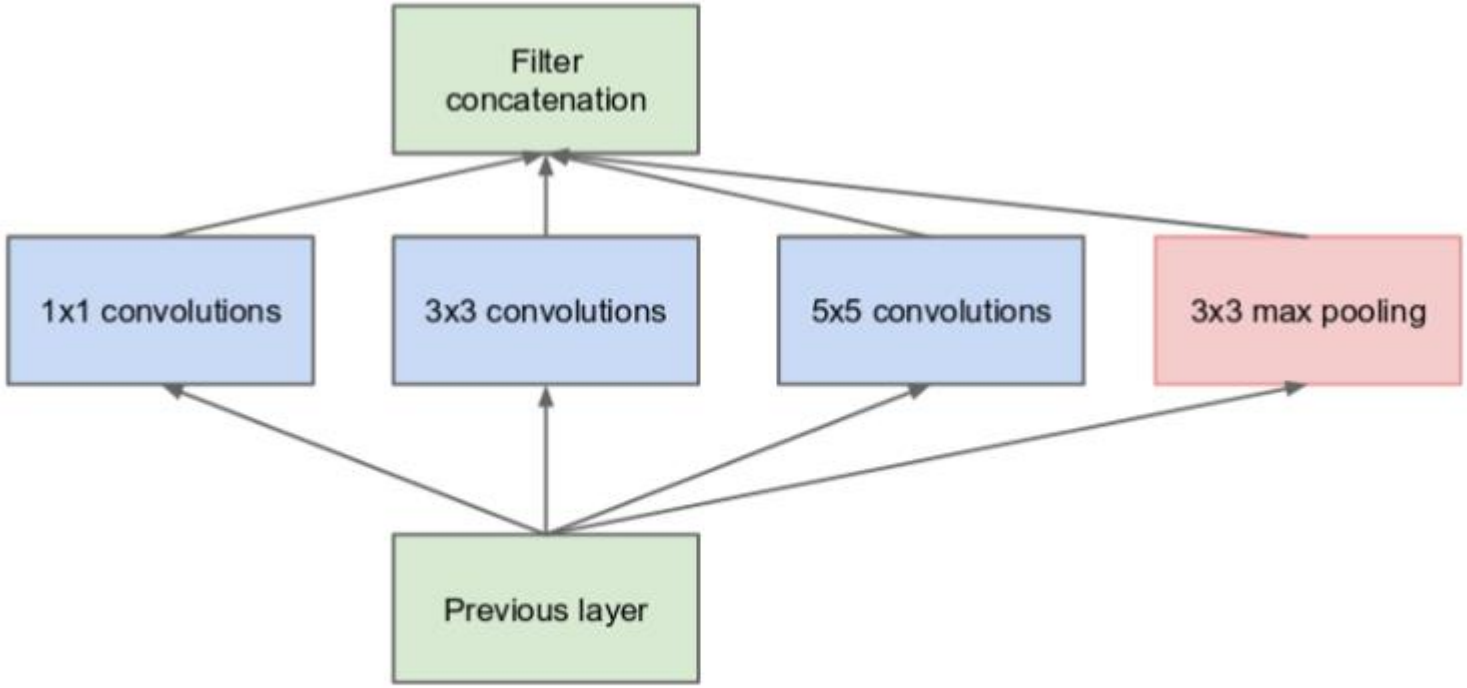
[2]wliu@cs.unc.edu, [3]reedscott@umich.edu, [4]arabinovich@magicleap.com

GoogLeNet and Inception v1 (2014)

Deep network, with **high computational efficiency**

Only **5 million parameters**, 22 layers of Inception modules

Won 2014 ILSVR-classification challenge (6,7% top 5 classification error)

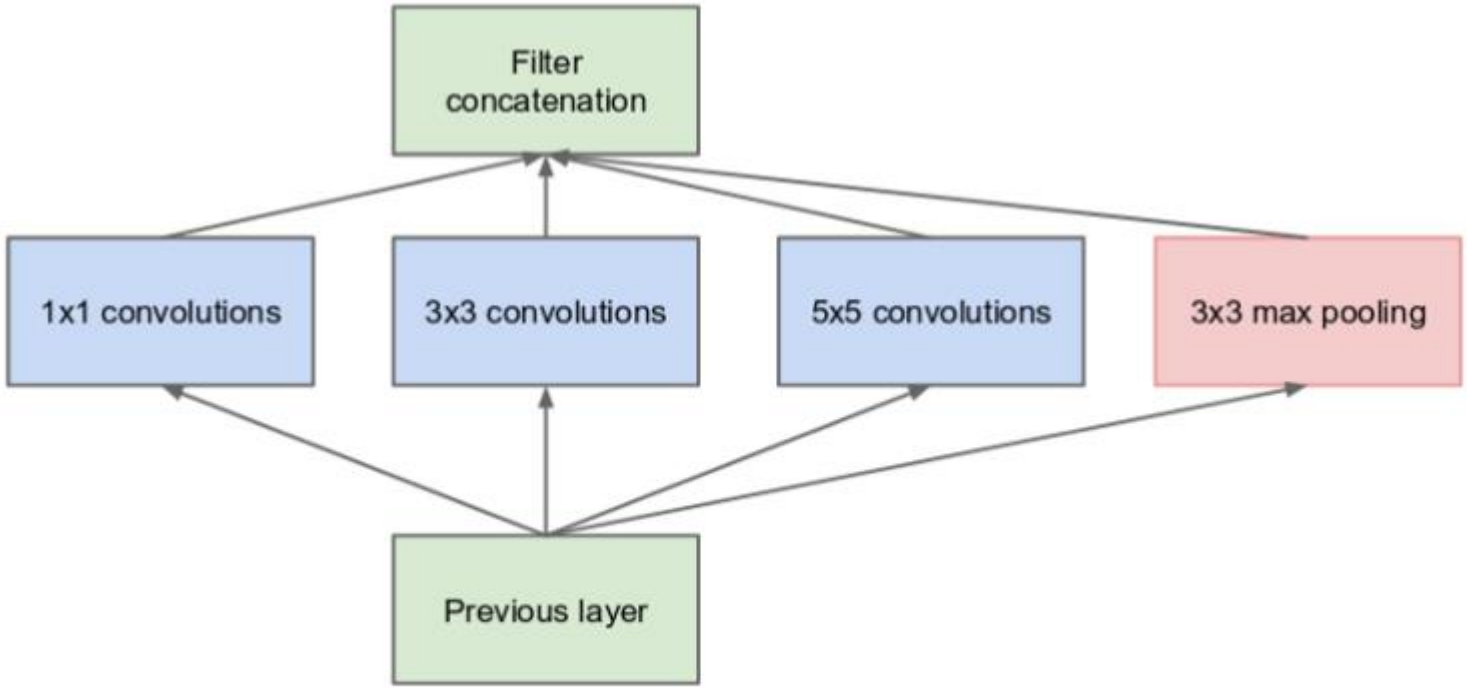


(a) Inception module, naïve version

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

GoogLeNet and Inception v1 (2014)

It is based on inception modules, which are sort of «networks inside the network» or «local modules»

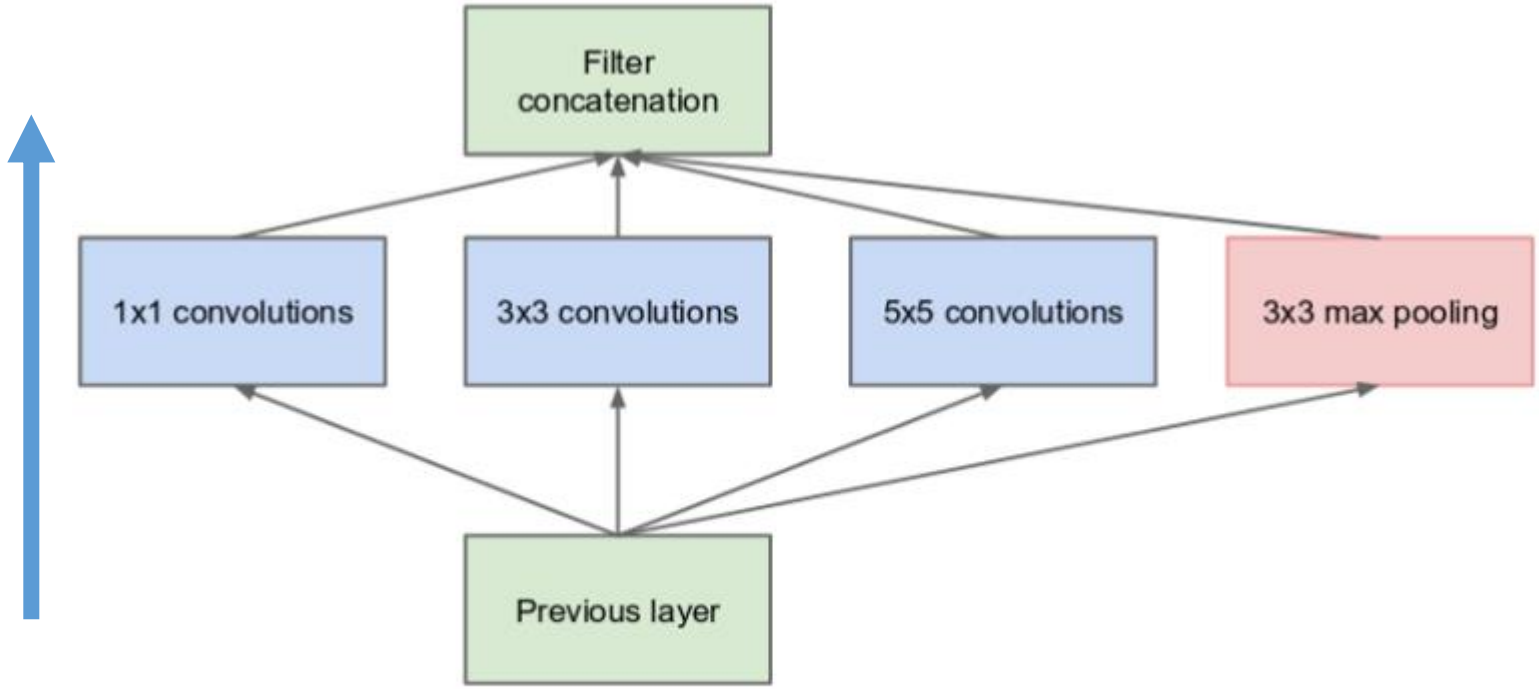


(a) Inception module, naïve version

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

Inception Module (2014)

- Choosing the **right kernel size** for the convolution operations is difficult, as image might show relevant features at different scale
- Deep networks **tend to over fit** and to become very computationally expensive

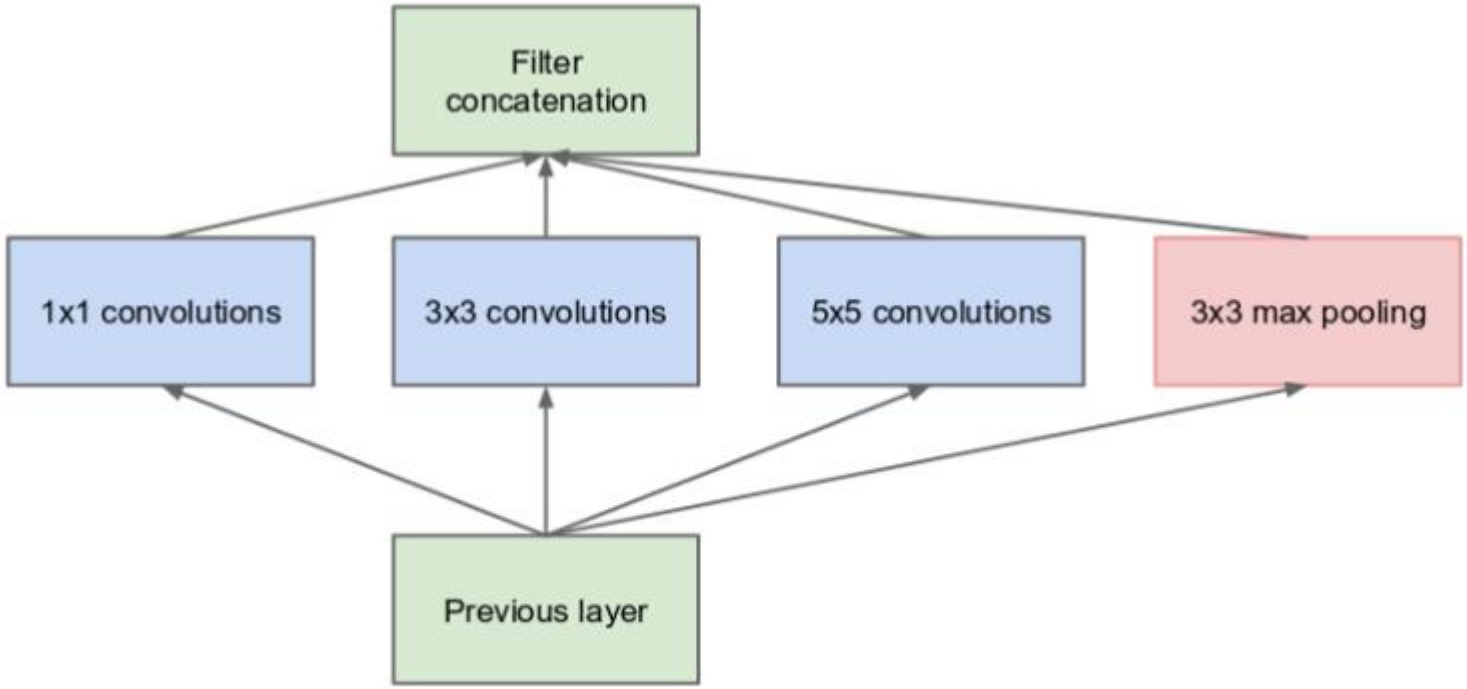


(a) Inception module, naïve version

Szegedy et al. "Going deeper with convolutions." *CVPR 2015.*

Inception Module (2014)

The solution is to **exploit multiple filter size** at the same level (1x1, 3x3, 5x5) and then **merge by concatenation** the output activation maps together



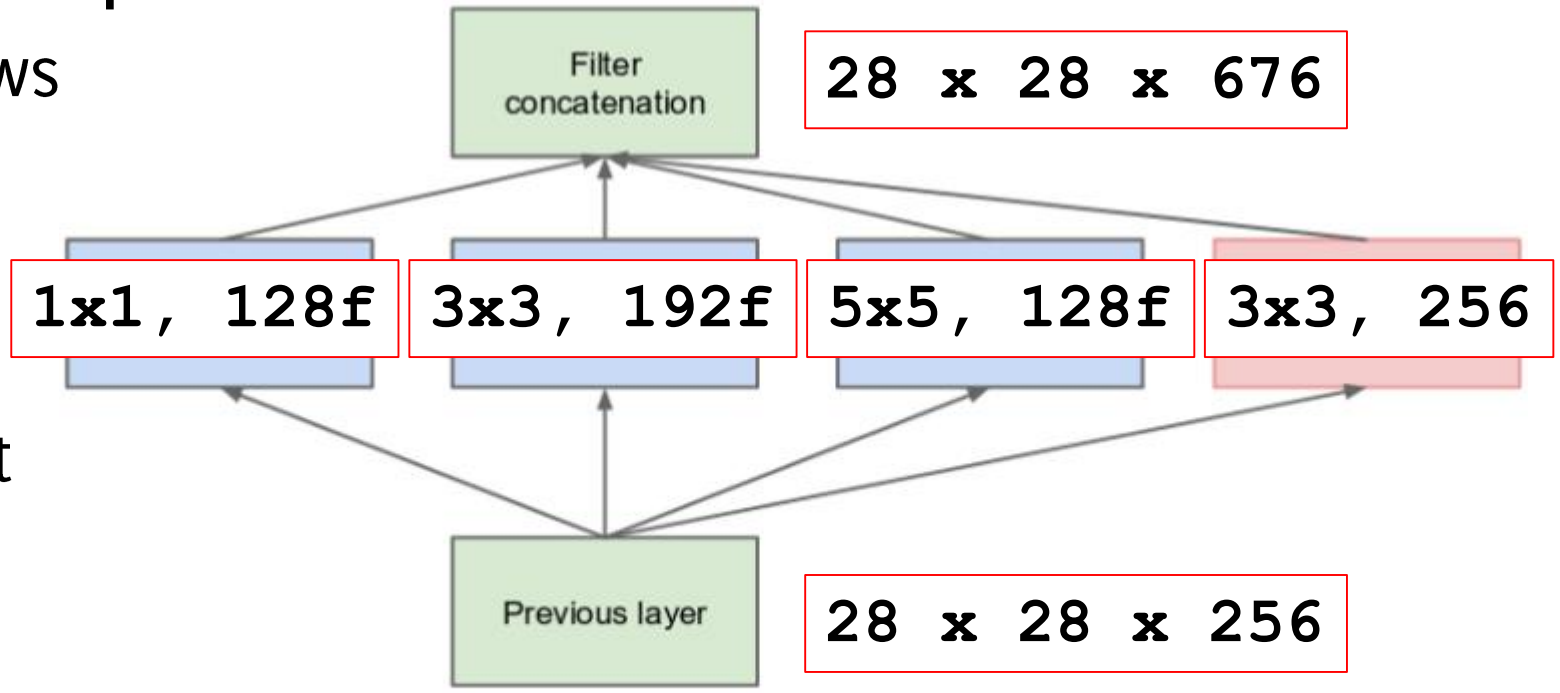
(a) Inception module, naïve version

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

Inception Module (2014)

The solution is to exploit multiple filter size at the same level (1x1, 3x3, 5x5) and then **merge by concatenation** the output activation maps together

- **Zero padding to preserve spatial size**
- The activation map grows much in depth
- A large number of operations to be performed due to the large depth of the input of each convolutional block: **854M operations** in this example



(a) Inception module, naïve version

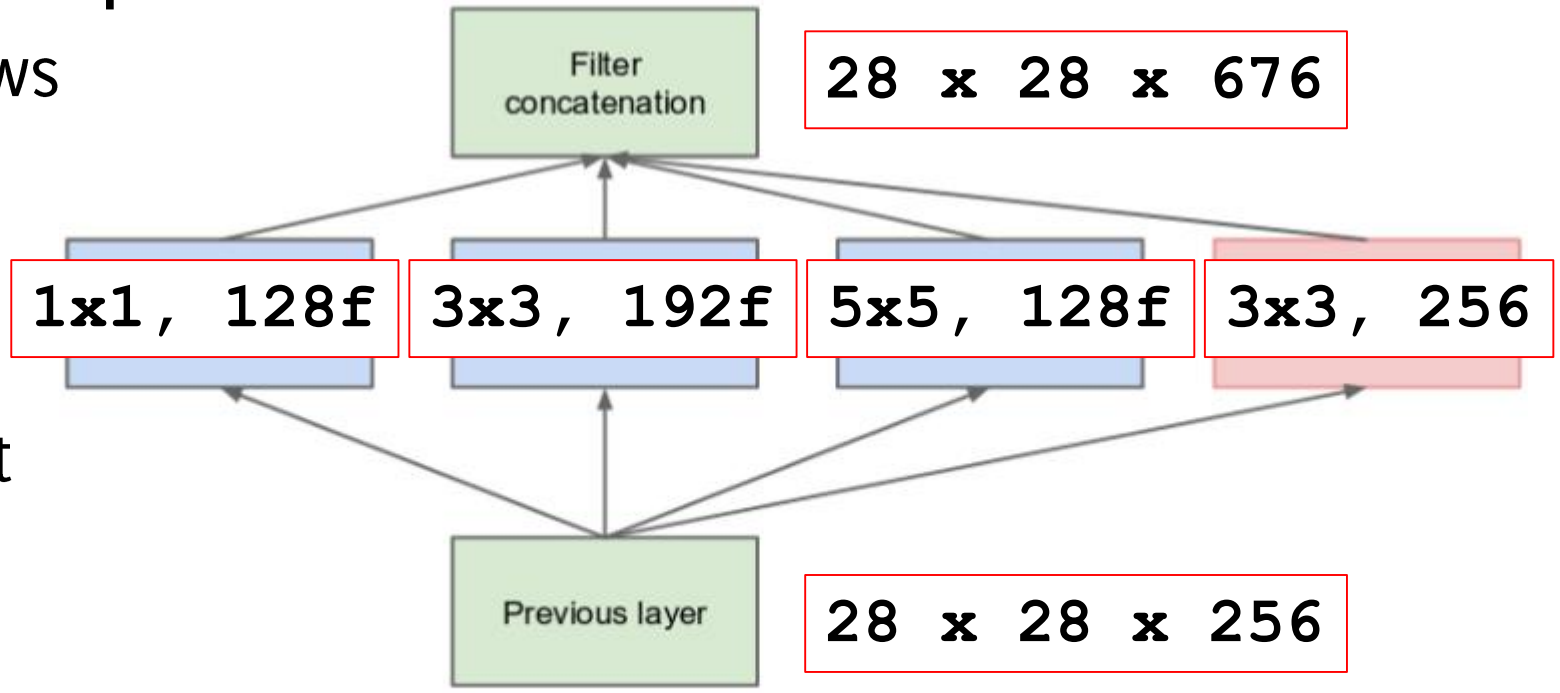
Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

Inception Module (2014)

The sol… 3x3, 5x5) an… togethe…

The spatial extent is preserved, but the depth of the activation map is much expanded. This is very expensive to compute

- **Zero padding to preserve spatial size**
- The activation map grows much in depth
- A large number of operations to be performed due to the large depth of the input of each convolutional block: **854M operations** in this example



(a) Inception module, naïve version

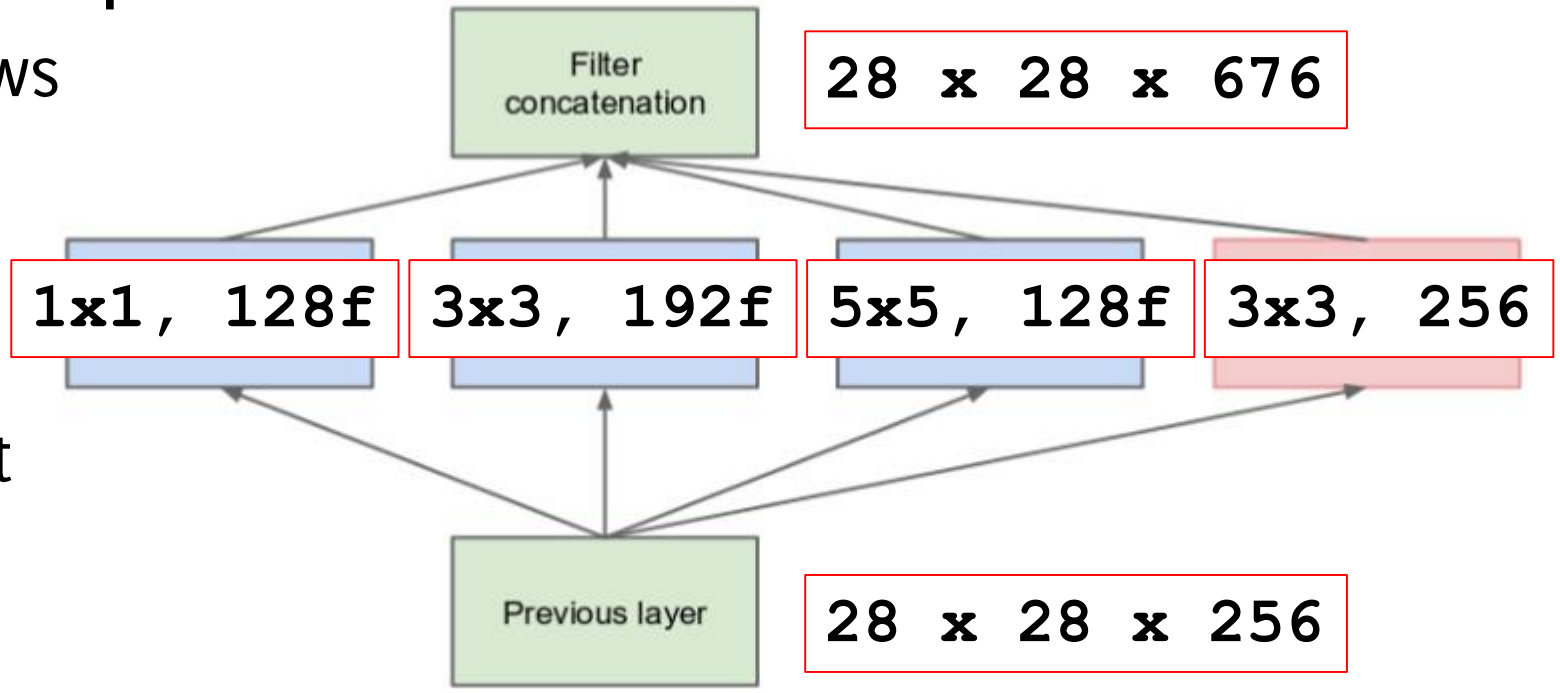
Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

Inception Module (2014)

The sol… 3x3, 5x5) an… togethe…

Computational problems will get significantly worst when stacking multiple layers…

- **Zero padding to preserve spatial size**
- The activation map grows much in depth
- A large number of operations to be performed due to the large depth of the input of each convolutional block: **854M operations** in this example



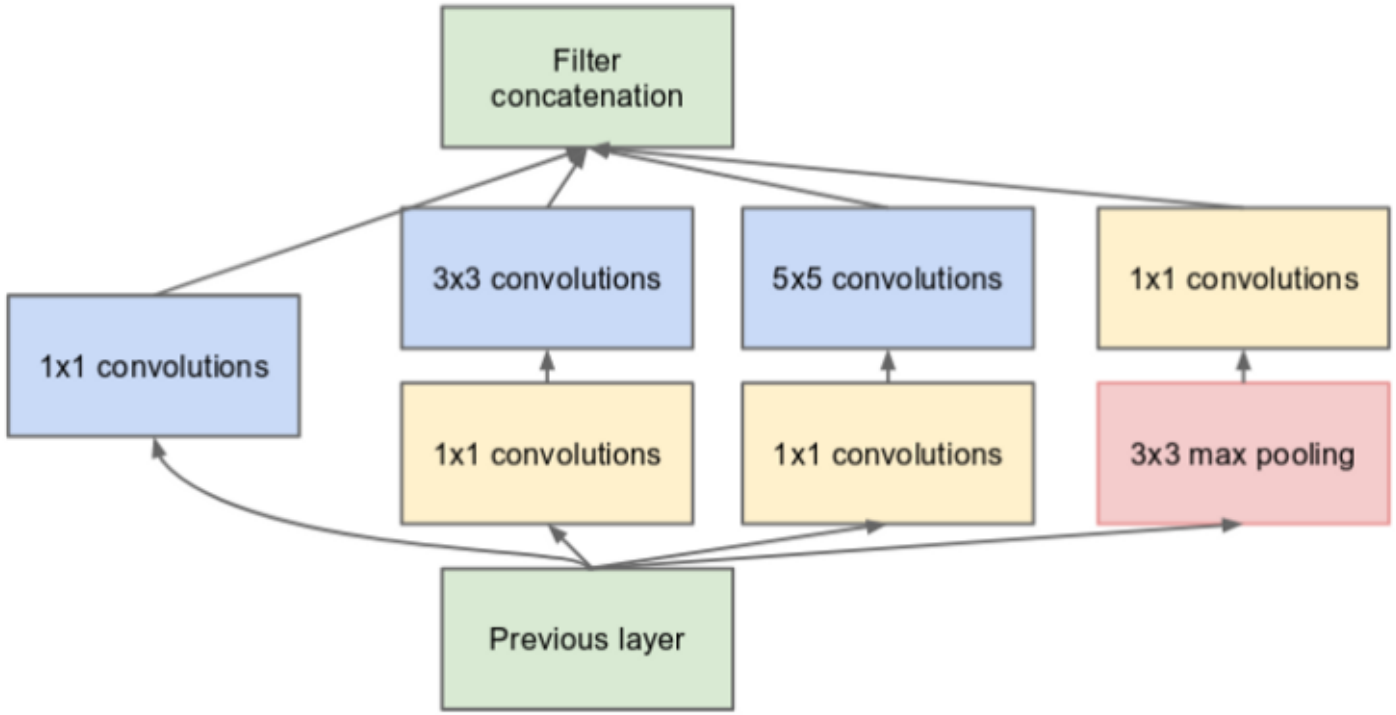
(a) Inception module, naïve version

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

Inception Module (2014)

To reduce the computational load of the network, the number of **input channels** is reduced by adding an **1x1 convolution** layers before the 3x3 and 5x5 convolutions

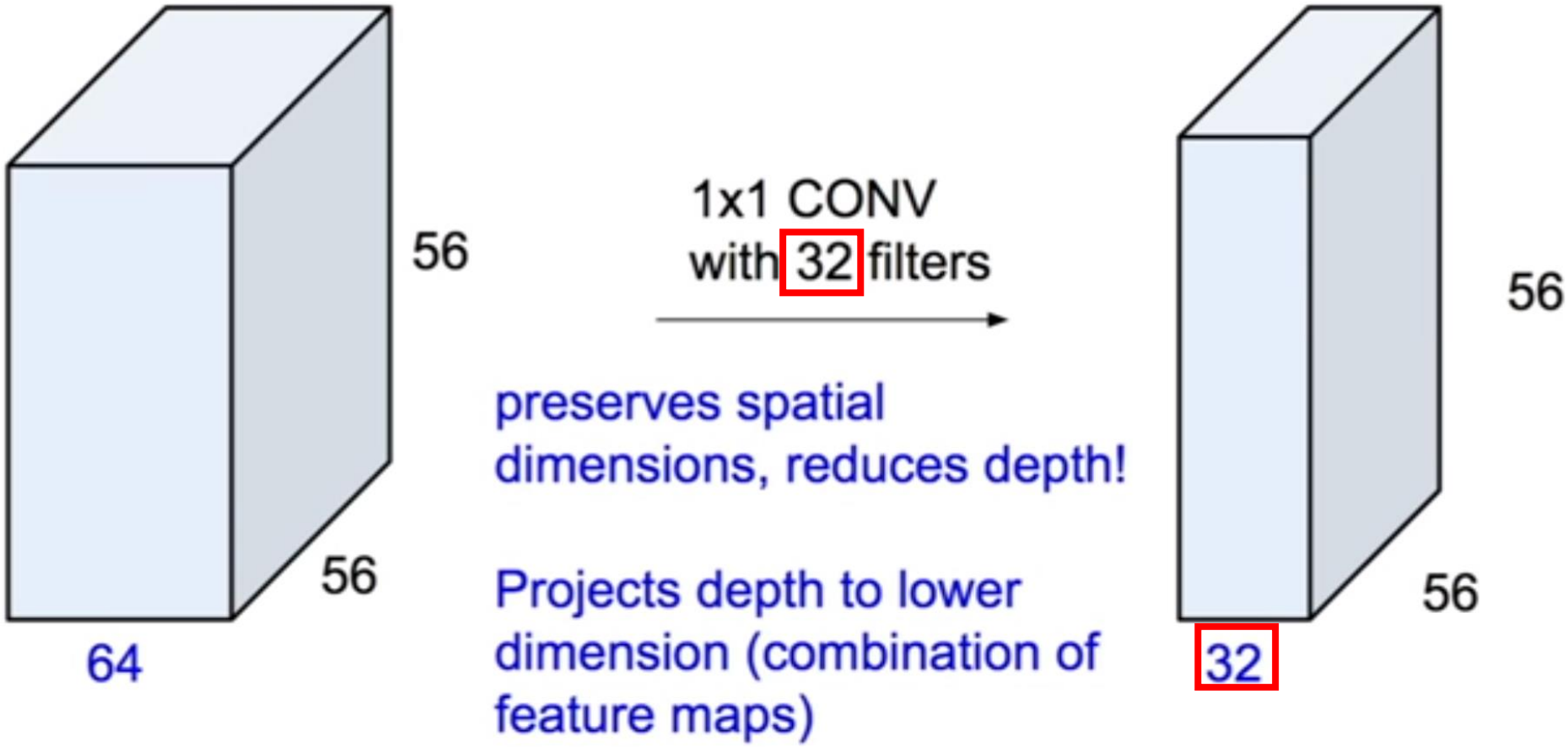
Using these 1x1 conv is referred to as **"bottleneck" layer**



(b) Inception module with dimension reductions

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

1x1 convolution layers as bottleneck

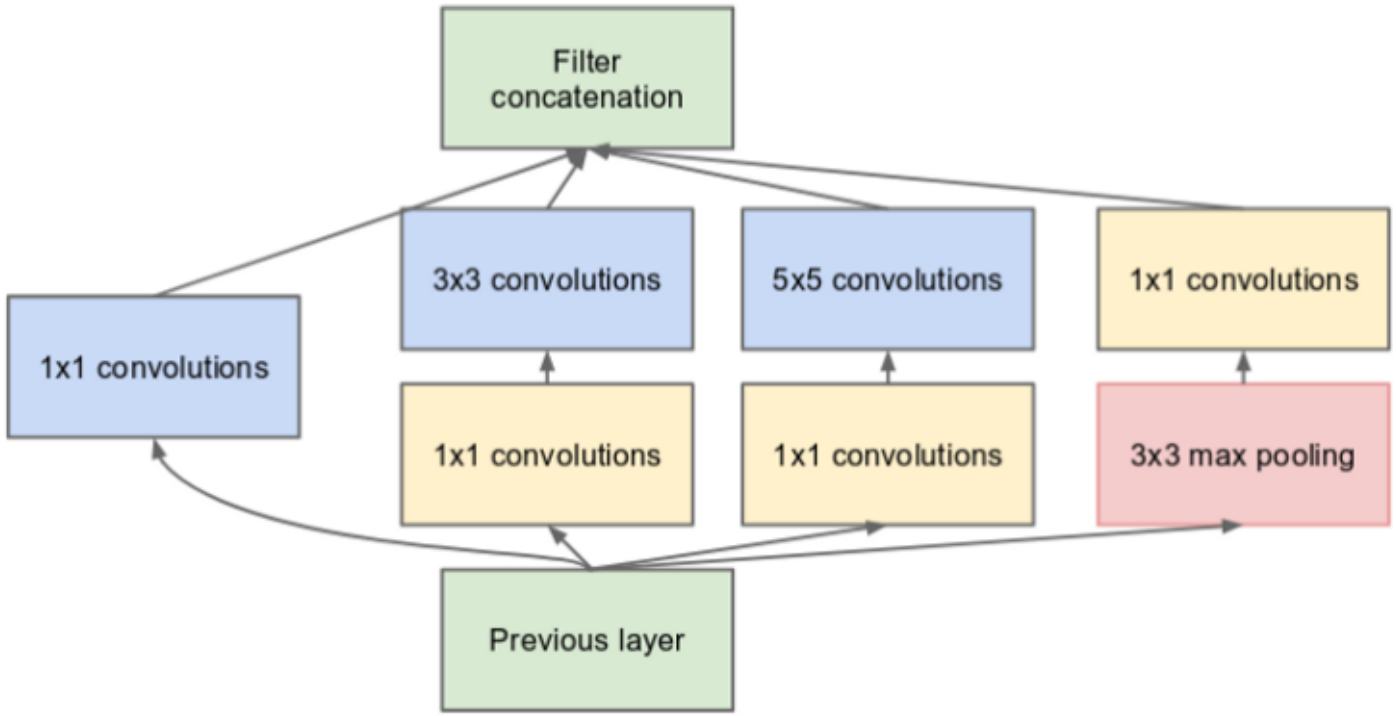


Inception Module (2014)

To reduce the computational load of the network, the number of **input channels** is reduced by adding an **1x1 convolution** layers before the 3x3 and 5x5 convolutions

The output volume has similar size, but the number of operation required is significantly reduced due to the 1x1 conv:
358M operations now

Adding 1x1 convolution layers increases the number of nonlinearities



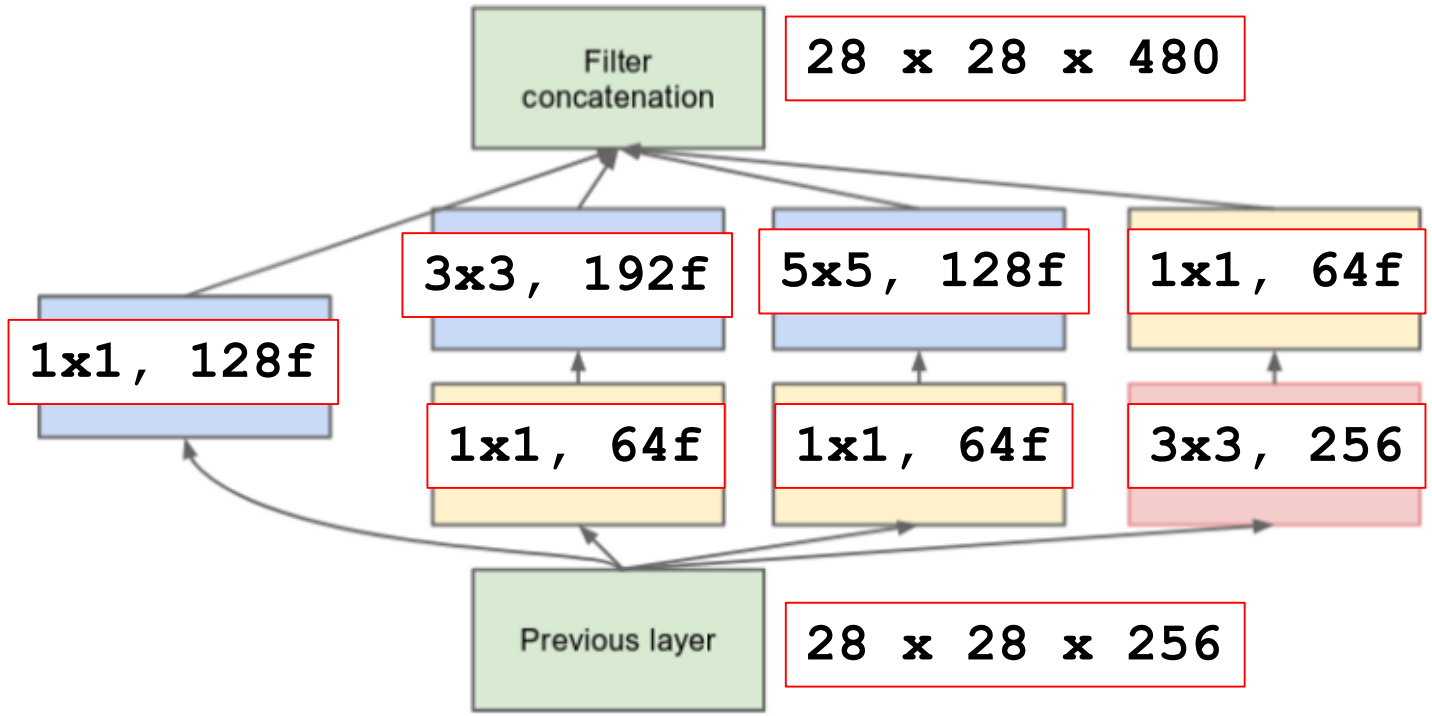
(b) Inception module with dimension reductions

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

Inception Module (2014)

To reduce the computational load of the network, the number of **input channels** is reduced by adding an **1x1 convolution** layers before the 3x3 and 5x5 convolutions

The output volume has similar size, but the number of operation required is significantly reduced due to the 1x1 conv:
358M operations now



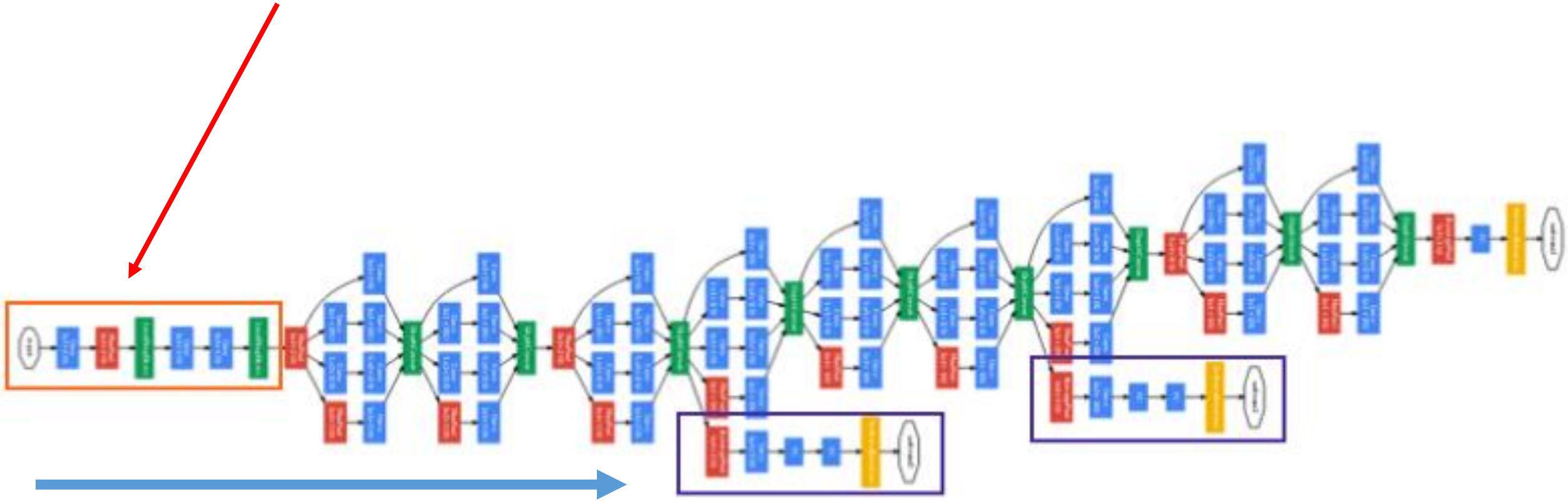
(b) Inception module with dimension reductions

Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

GoogLeNet (2014)

GoogleNet stacks many inception modules: 27 layers considering pooling ones.

At the beginning there are two blocks of conv + pool layers

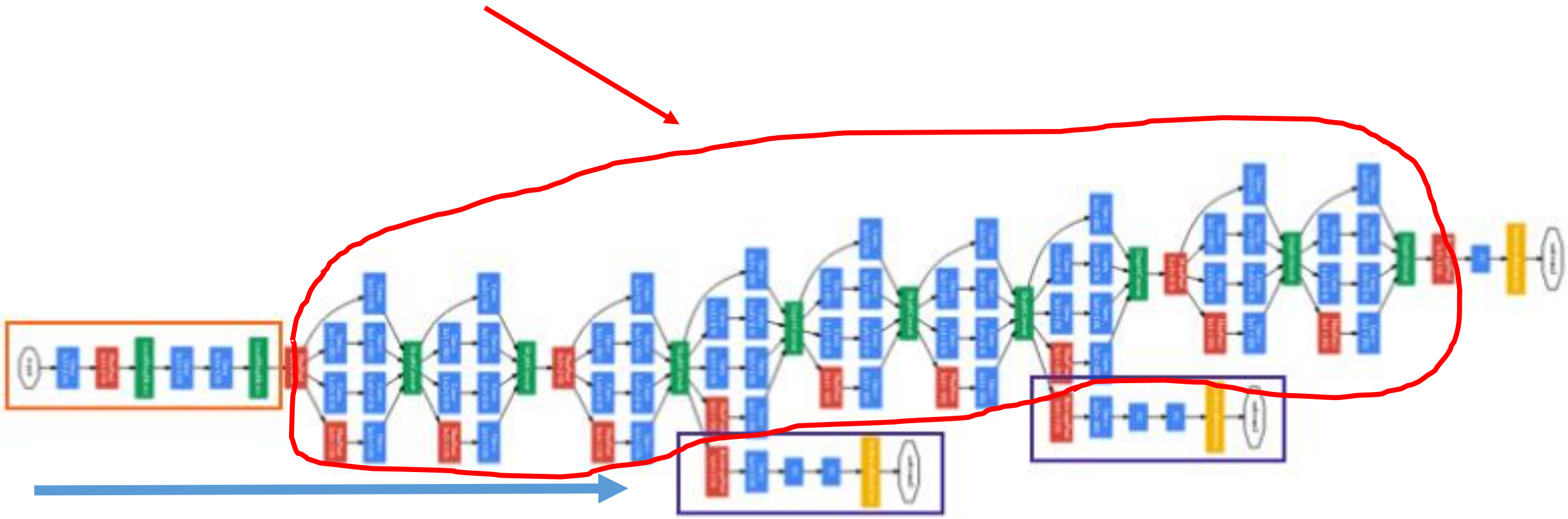


Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

GoogLeNet (2014)

GoogleNet stacks many inception modules: 27 layers considering pooling ones.

Then, there are a stack of 9 of inception modules

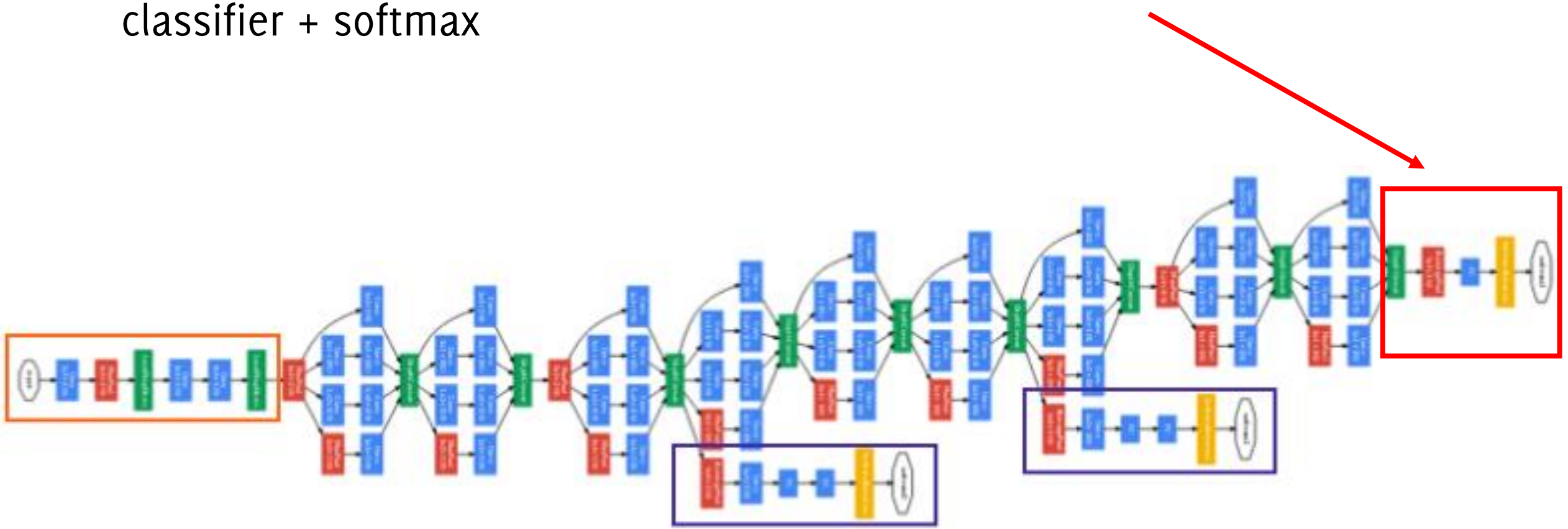


Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

GoogLeNet (2014)

GoogleNet stacks many inception modules: 22 layers considering pooling ones.

No Fully connected layer at the end, simple averaging pooling, linear classifier + softmax



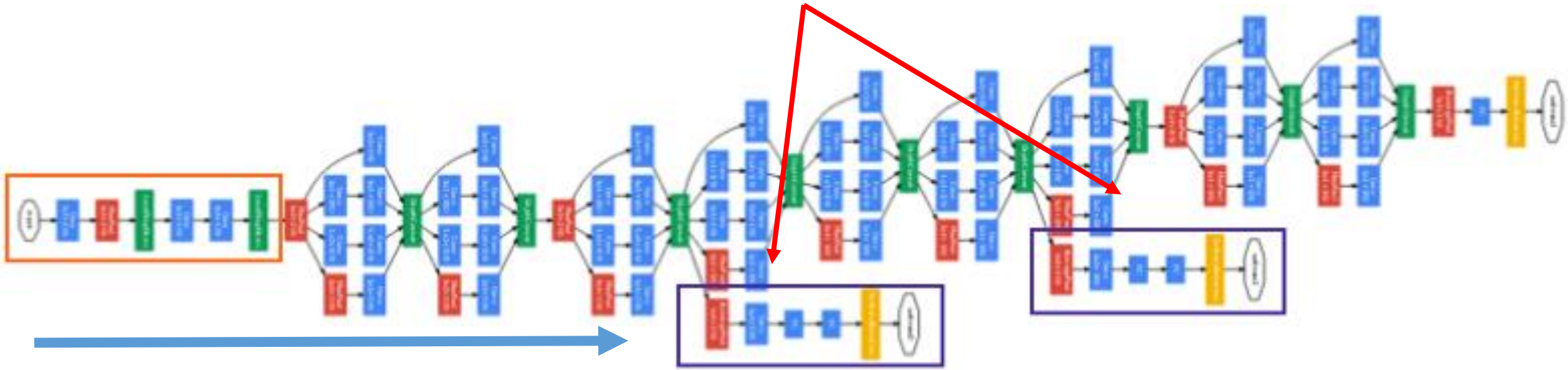
Szegedy et al. "Going deeper with convolutions." *CVPR 2015*

GoogLeNet (2014)

GoogleNet stacks many inception modules: 27 layers considering pooling ones.

It also suffers of the **dying neuron problem**, therefore the authors **add two extra classifiers** on the intermediate representation **to compute an intermediate loss that is used during training.**

You expect intermediate layers to be able to classify as well



Szegedy et al. "Going deeper with convolutions." *CVPR 2015*