Deep Learning For Visual Recognition

Giacomo Boracchi

CVPR USI, May 11 2020

giacomo.boracchi@polimi.it

https://boracchi.faculty.polimi.it/

Outline

- Problem Formulation
- Image Classification Approaches and CNNs as Data-driven feature extractors
- Peculiarities of CNN
- Fully Convolutional CNNs
- Training with Data Scarcity: Trasfer Learning and Data Augmentation
- Deep Networks for Semantic Segmentation

.... object detection

GAN

(....)

Setting up the stage..

Image Classification

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

I

"wheel"

I

"castle"

Image Classification

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

I 

⇨ “wheel” 65%, “tyre” 30%..

I 

⇨ “castle” 55%, “tower” 43%..

Image Classification, the problem

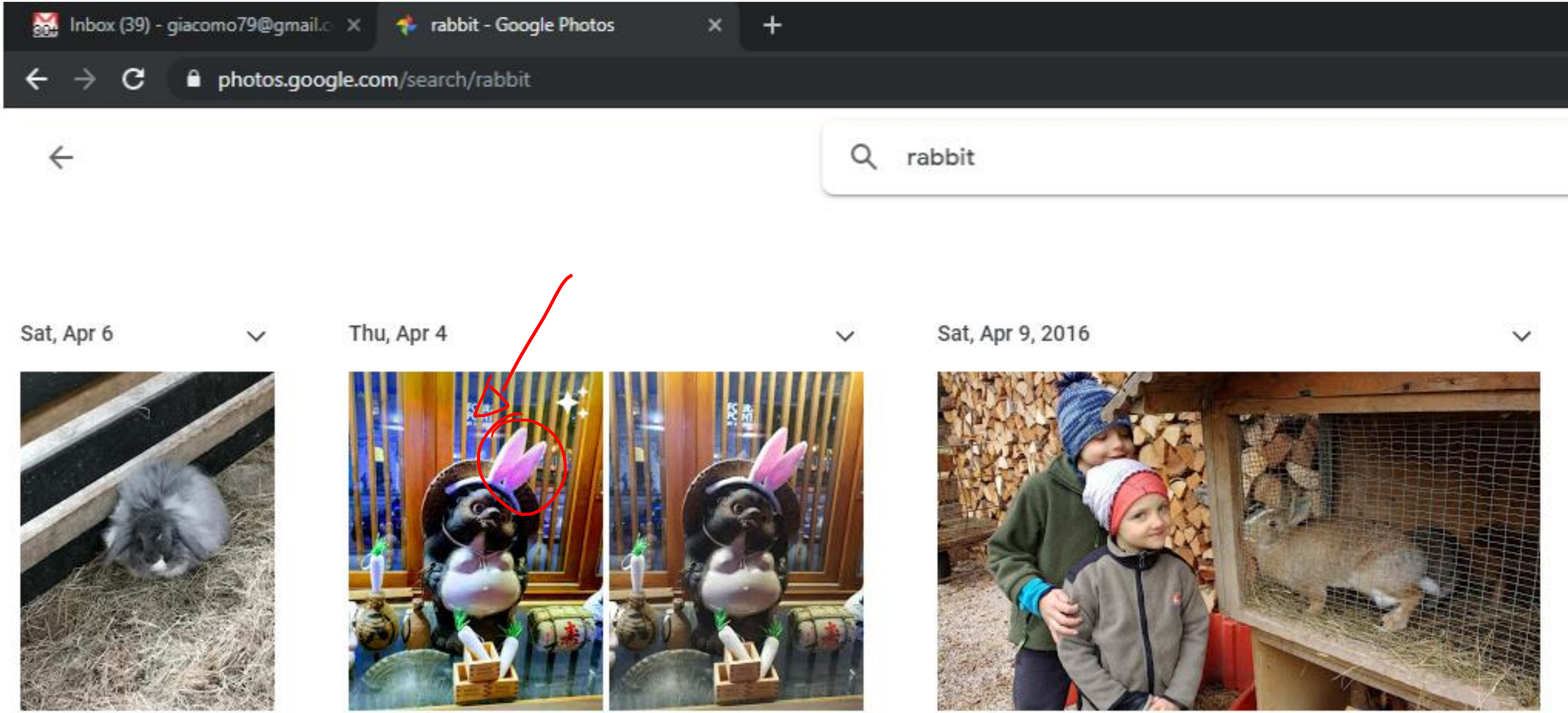
Assign to an input image $I \in \mathbb{R}^{R \times C \times 3}$:

- a label l from a fixed set of categories

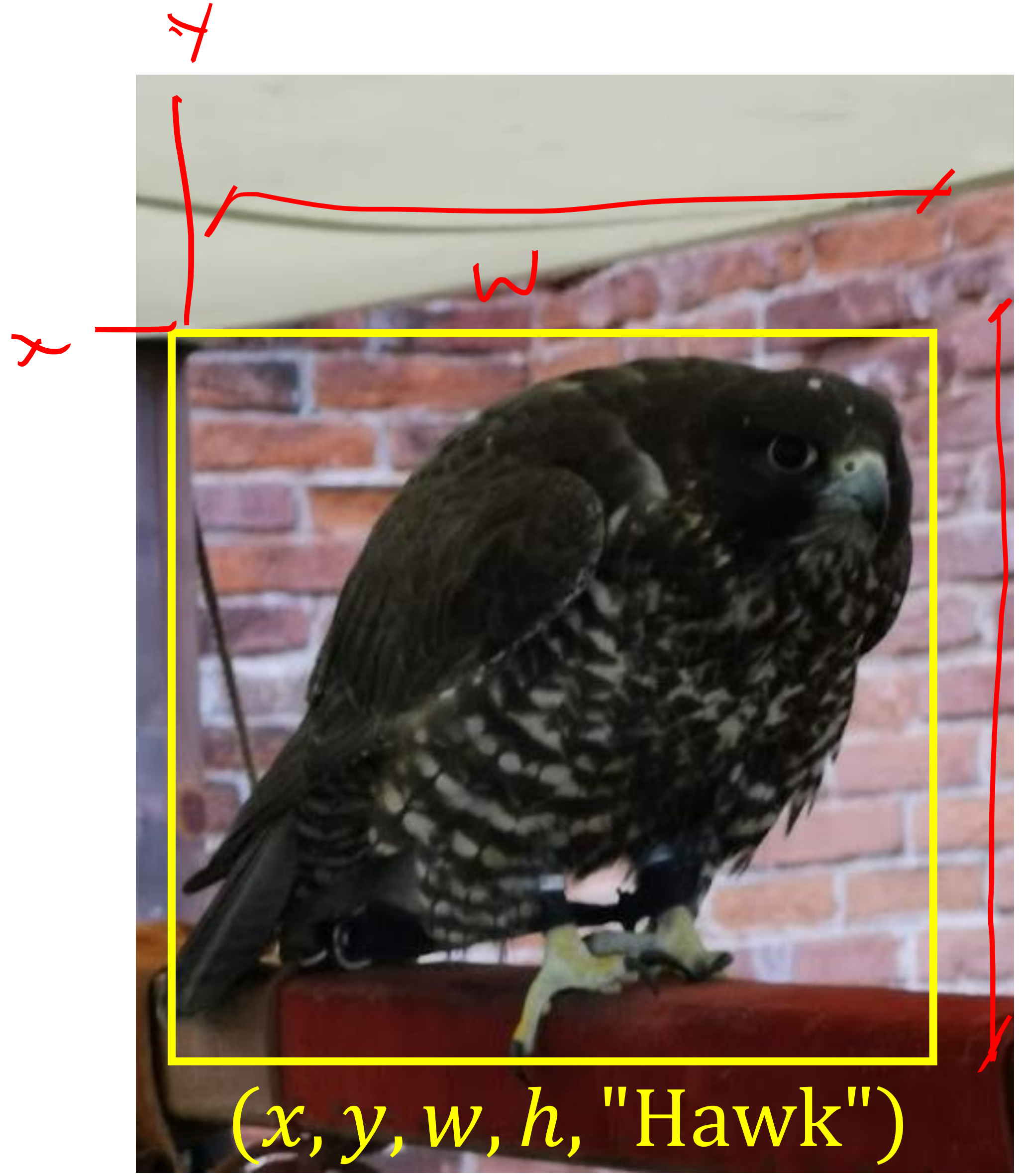
$$\Lambda = \{\text{"wheel"}, \text{"cars"}, \ldots, \text{"castle"}, \text{"baboon"}\}$$

$$\boxed{I \to l \in \Lambda}$$

Image Classification Example



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

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

- the coordinates (x, y, h, w) of the bounding box enclosing that object

real categorical

$$I \to (\boxed{x, y, h, w}, l)$$

5d space

Object Detection

cchi

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 $\Lambda = \{\text{"wheel"}, \text{"cars"}, \dots, \text{"castle"}, \text{"baboon"}\}$, each corresponding **to an instance of that object**
- the coordinates $\{(x, y, h, w)_i\}$ of the bounding box enclosing **each** object

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

Segmentation

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

Image Segmentation, the problem

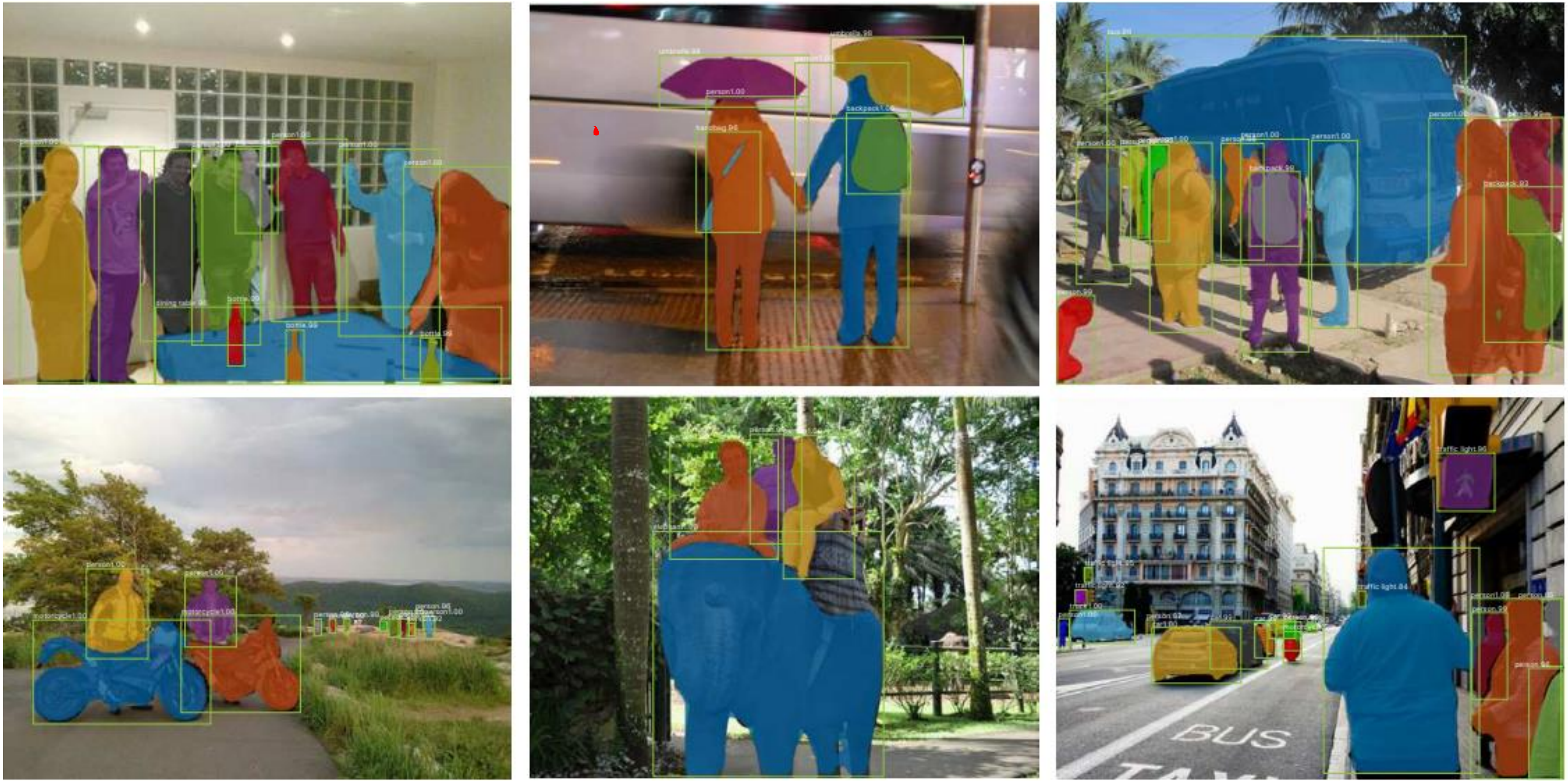
Assign **to each pixel** of an image $I \in \mathbb{R}^{R \times C \times 3}$:

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

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

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:

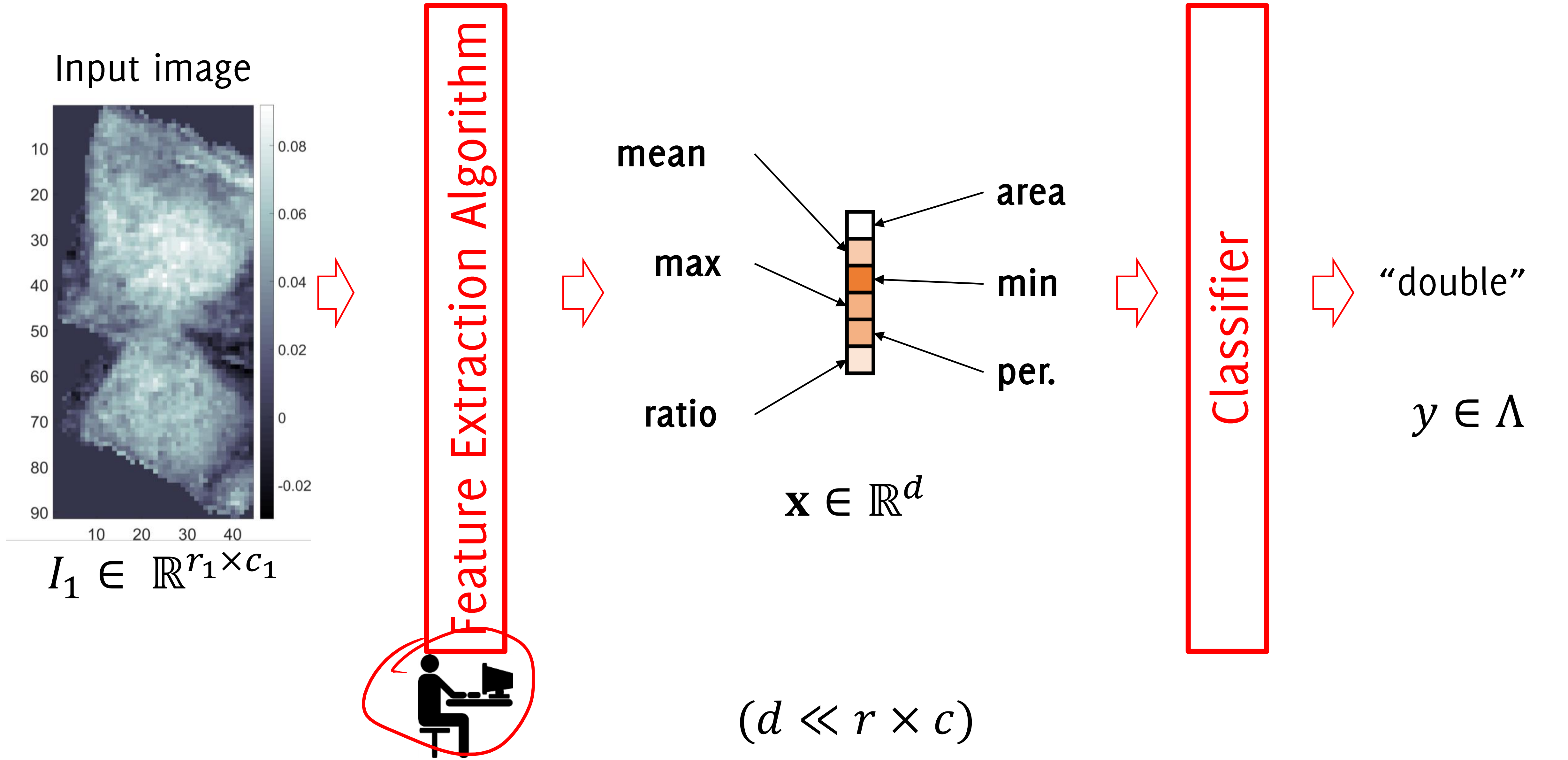
- **multiple** labels $\{l_i\}$ from a fixed set of categories $\Lambda = \{$"wheel", "cars", …, "castle", "baboon"$\}$, each corresponding to **an instance of that object**
- the coordinates $\{(x, y, h, w)_i\}$ of the **bounding box enclosing each object**
- the **set of pixels** S in each bounding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

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

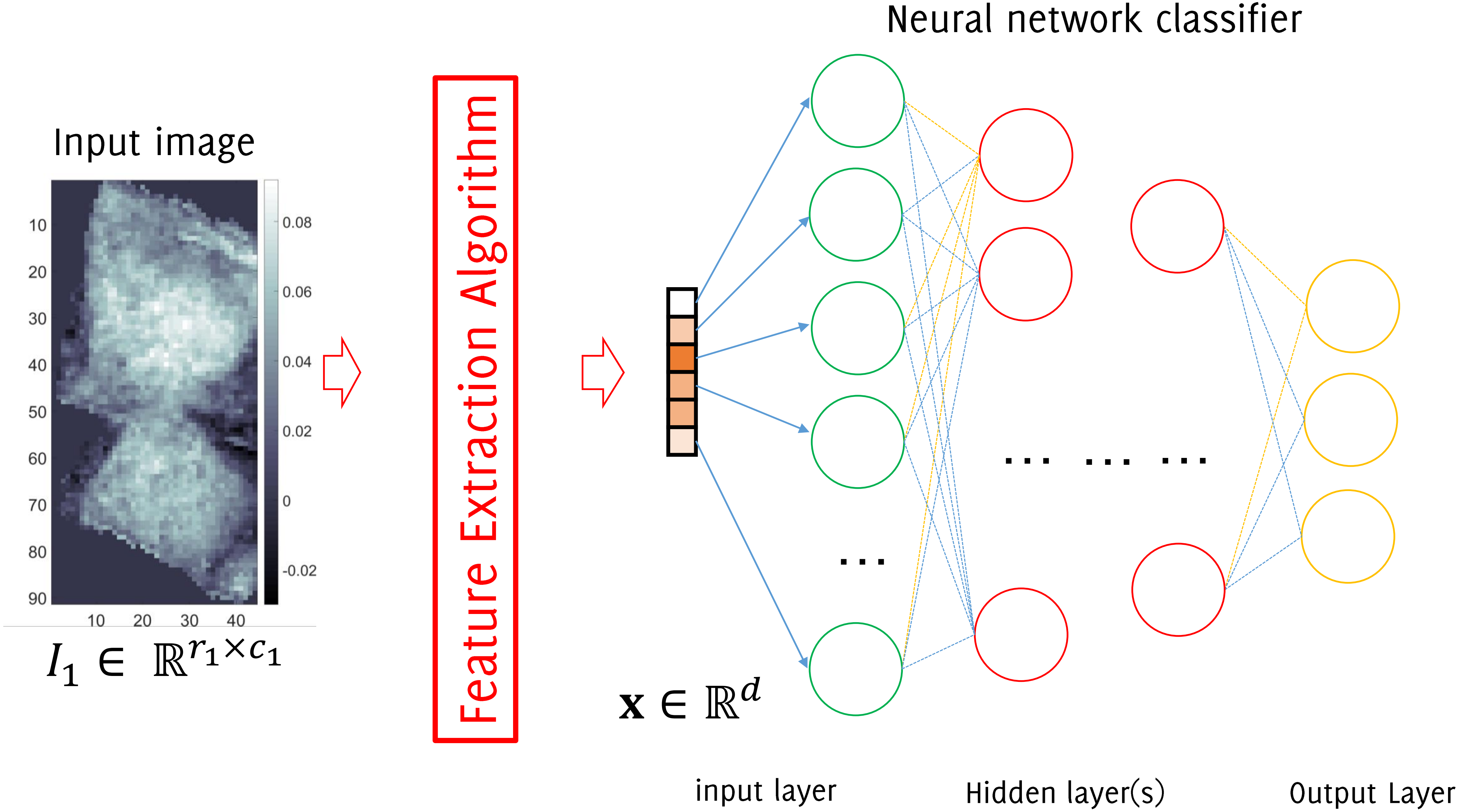
Instance Segmentation

Approaches to Image Classification

Hand Crafted Features

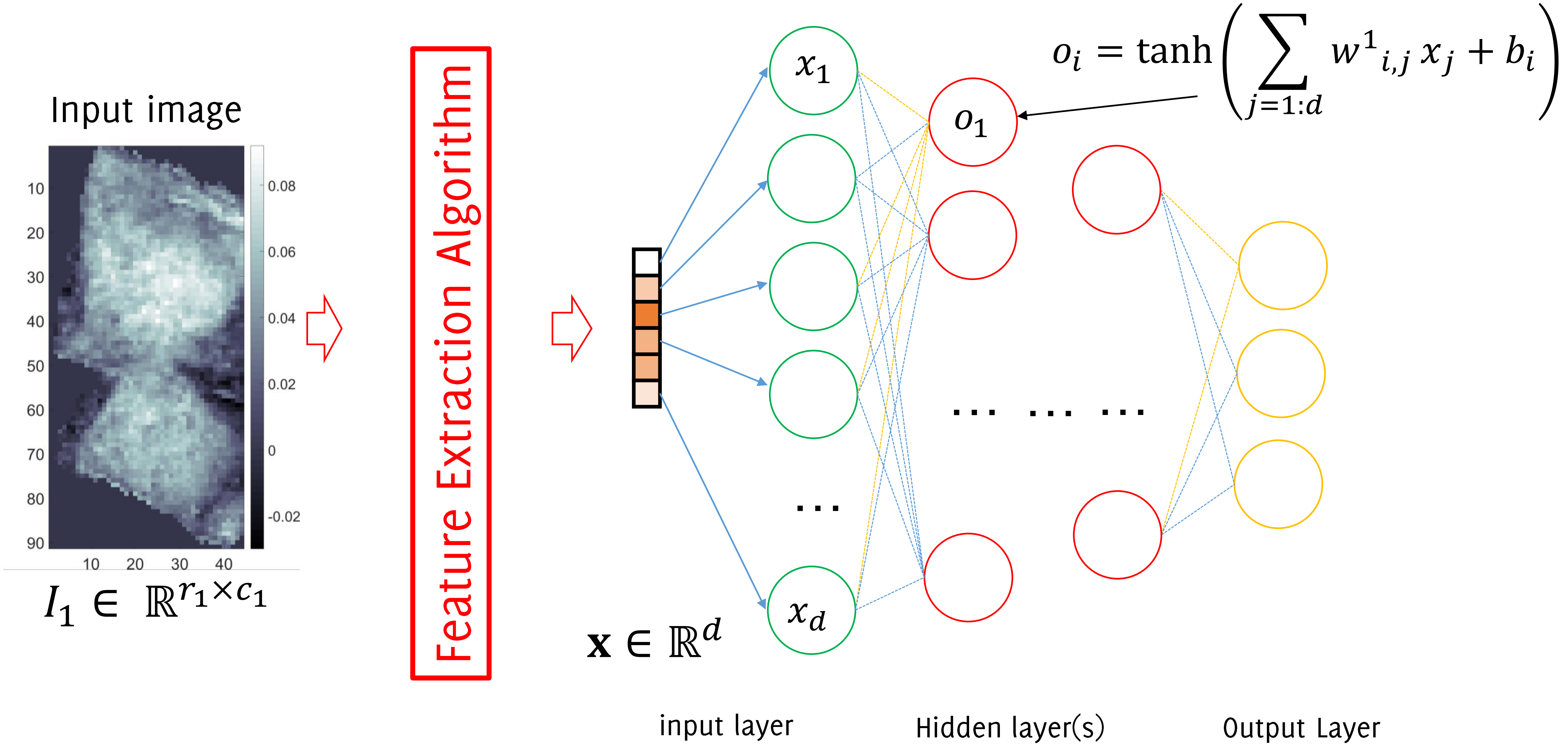


Hand Crafted Features

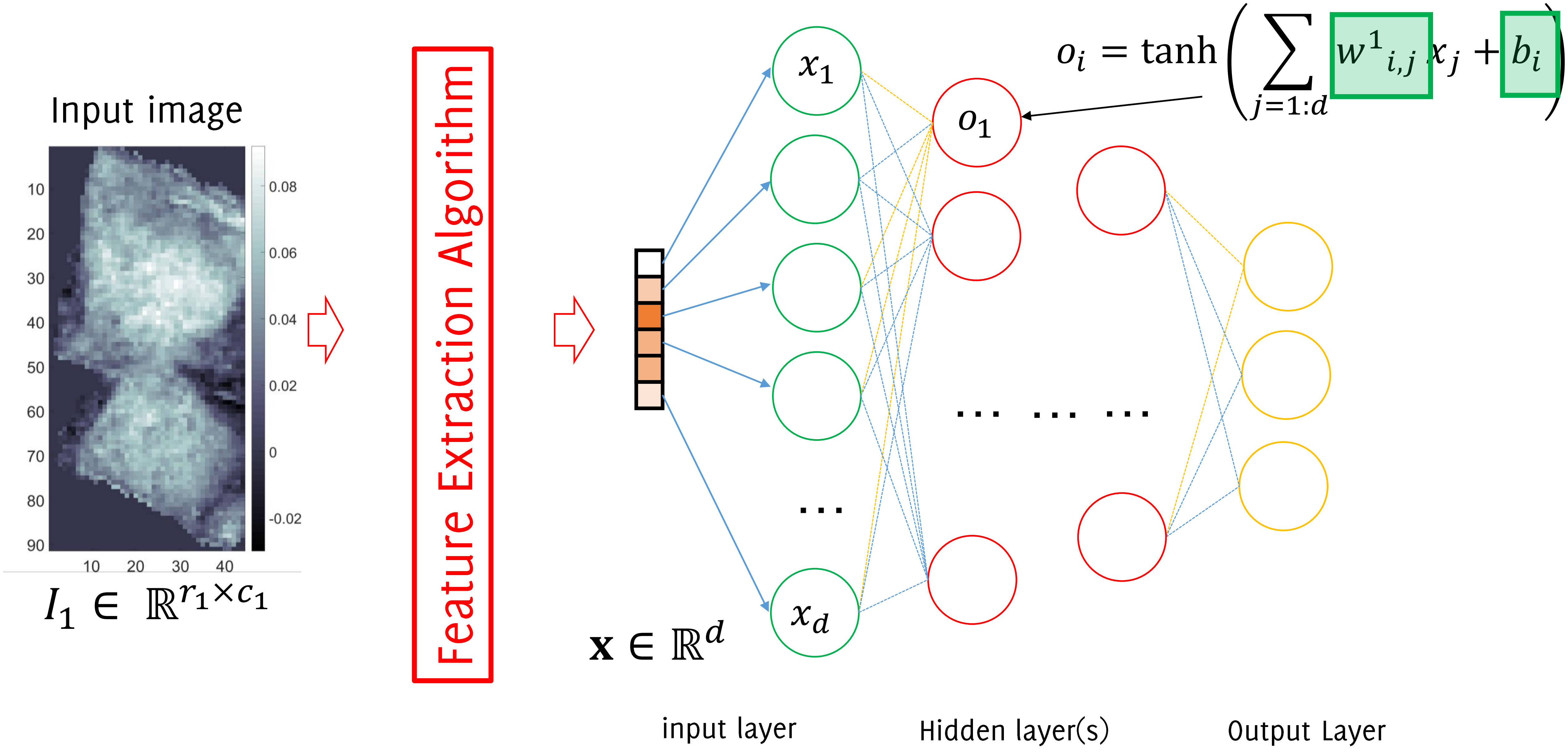


Giacomo Boracchi

Hand Crafted Features

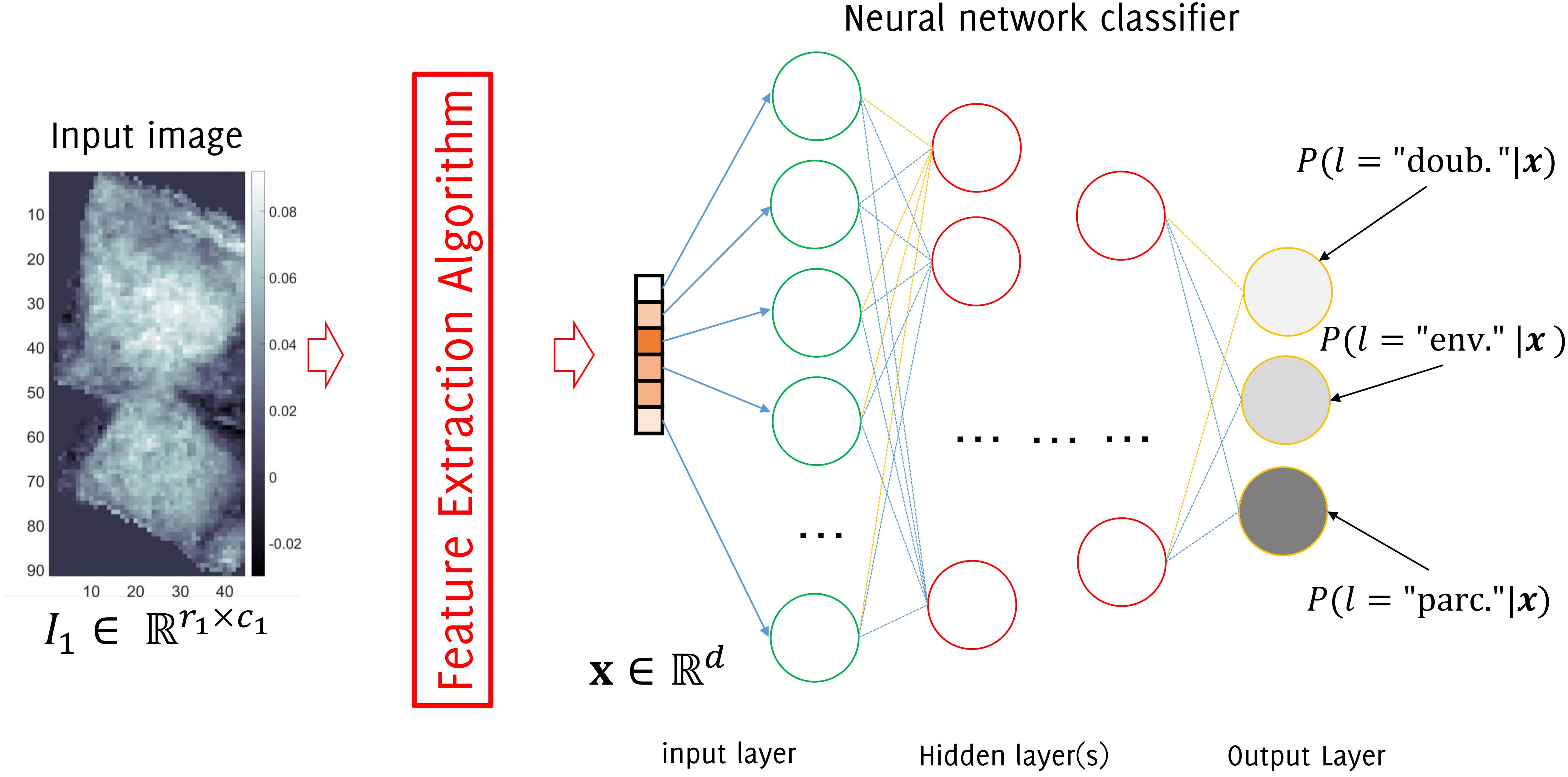


Hand Crafted Features



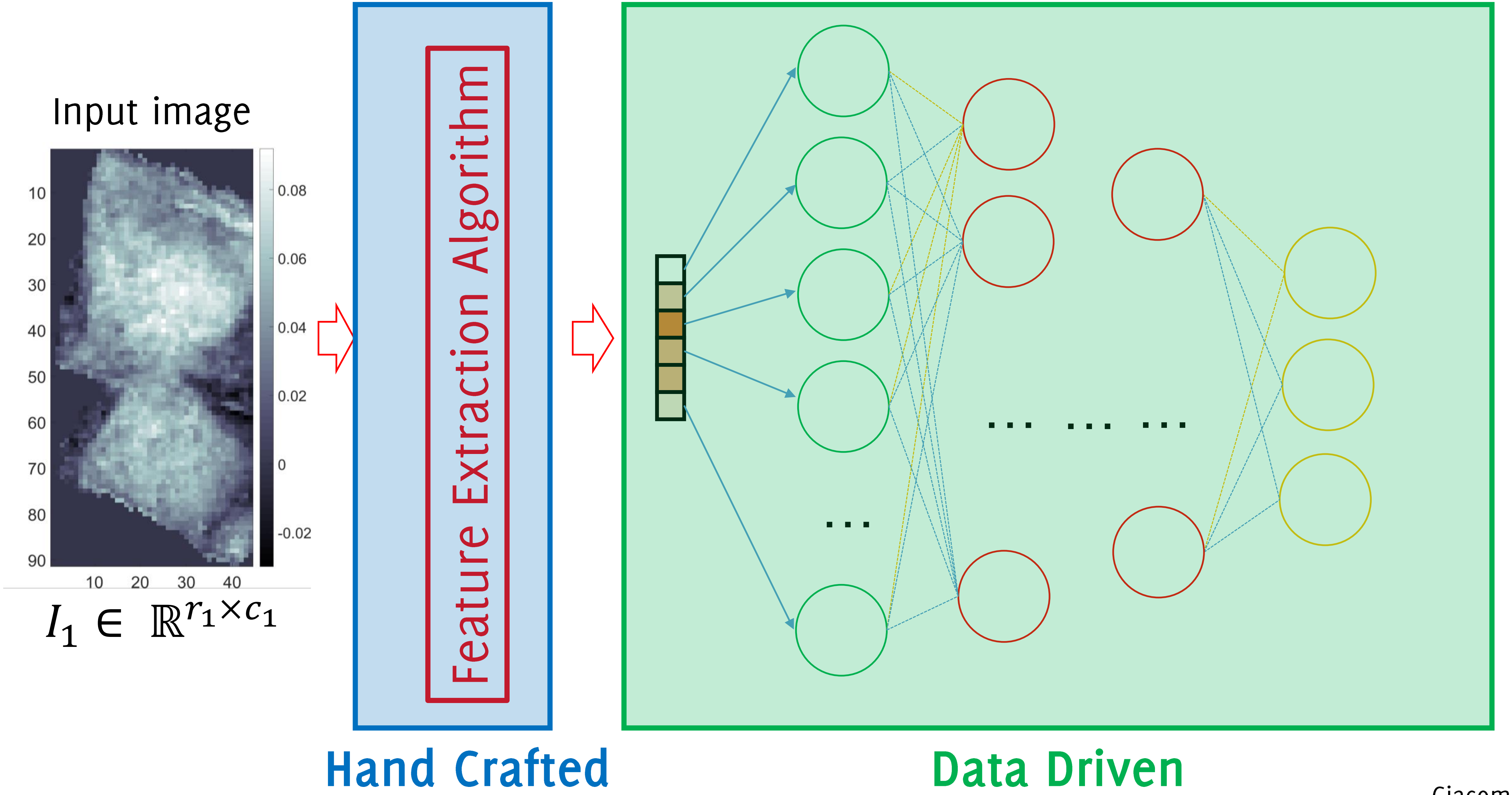
Giacomo Boracchi

Hand Crafted Features



Giacomo Boracchi

Hand Crafted Features



Giacomo Boracchi

Data Driven Models

They are defined from a training set of supervised pairs

$$TR = \{(\boldsymbol{x}, l)_i, i = 1, \dots, N\}$$

The model parameters (e.g. Neural Network weights) are set to minimize a **loss function**

Can definitvely boost the image classification performance

Hand Crafted Featues, pros:

- **Exploit a priori** / expert **information**
- Features are **interpretable** (you might understand why they are not working)
- You can **adjust features** to improve your performance
- **Limited amount of training data** needed
- You can give more relevance to some features

Hand Crafted Featues, cons:

- Requires a lot of **design/programming efforts**
- **Not viable** in many **visual recognition** tasks (e.g. on natural images) which are easily performed by humans
- **Risk of overfitting** the training set used in the design
- **Not very general and "portable"**

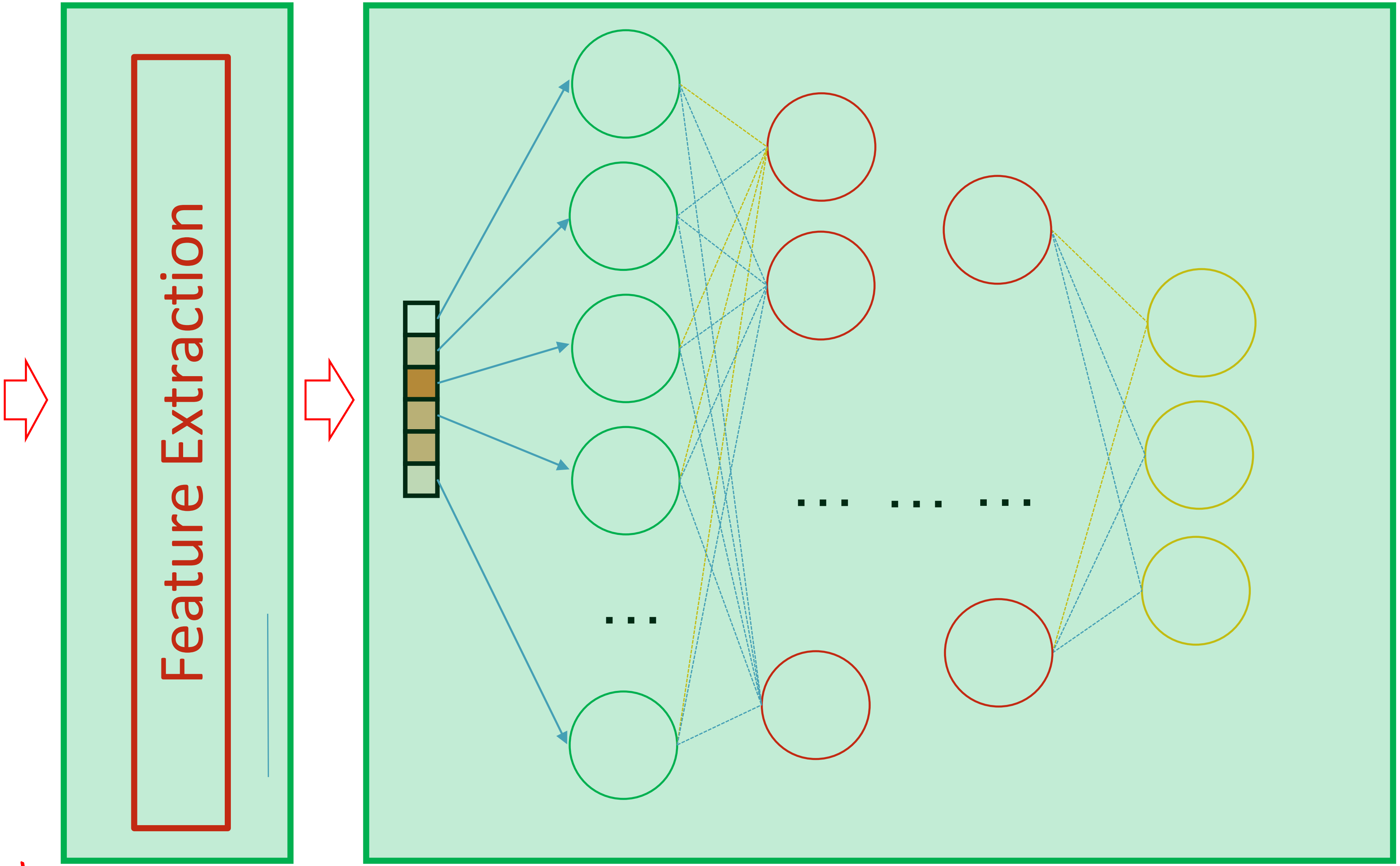
Data-Driven Features

... the advent of deep learning

Data-Driven Features

Input image

$I_1 \in \mathbb{R}^{r_1 \times c_1}$

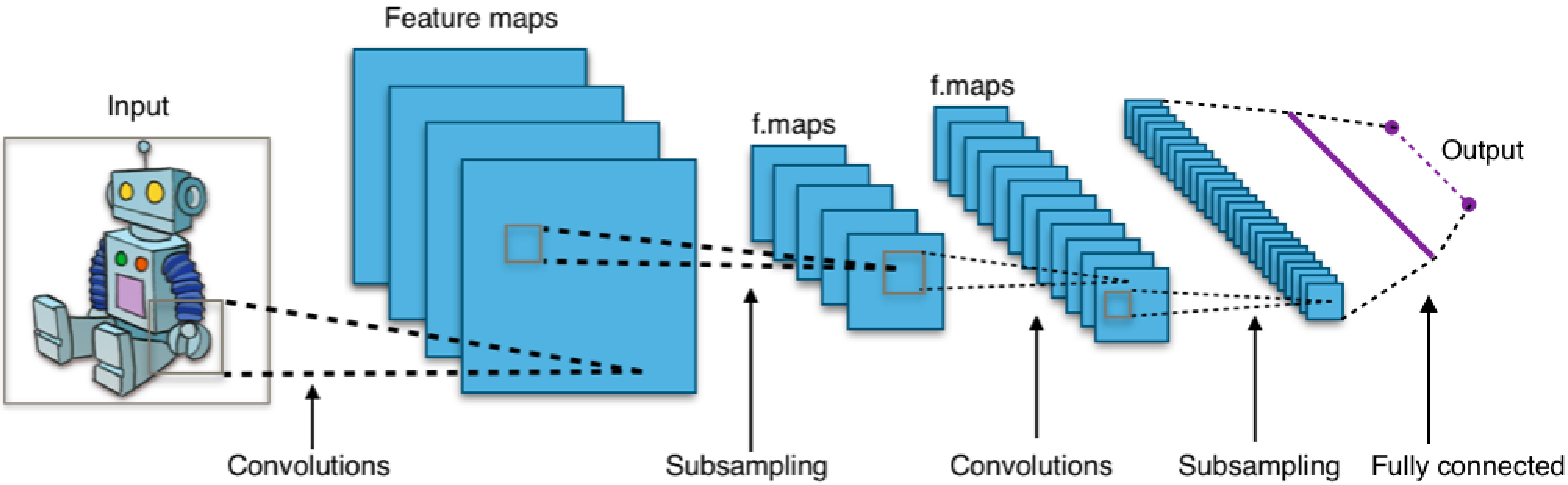


Data Driven

Data Driven

Convolutional Neural Networks (CNN)

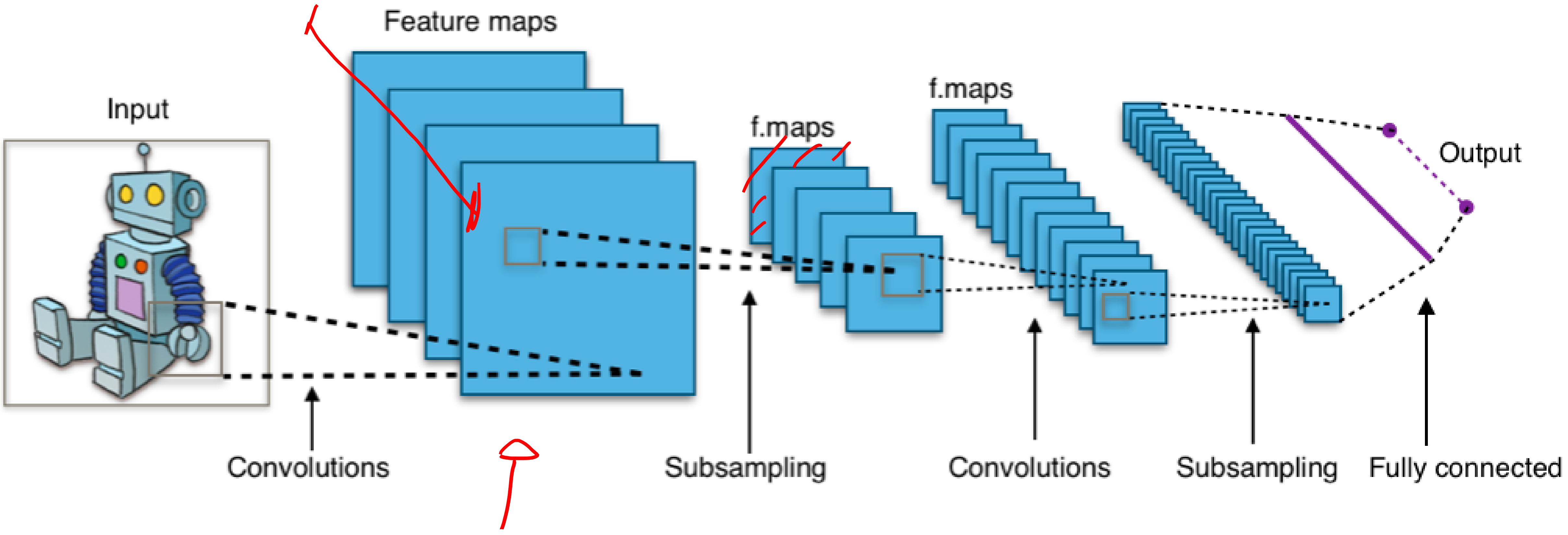
The typical architecture of a convolutional neural network



Convolutional Neural Networks (CNN)

The typical architec

Btw, this figure contains an error.
If you are CNN-Pro, you should spot it!

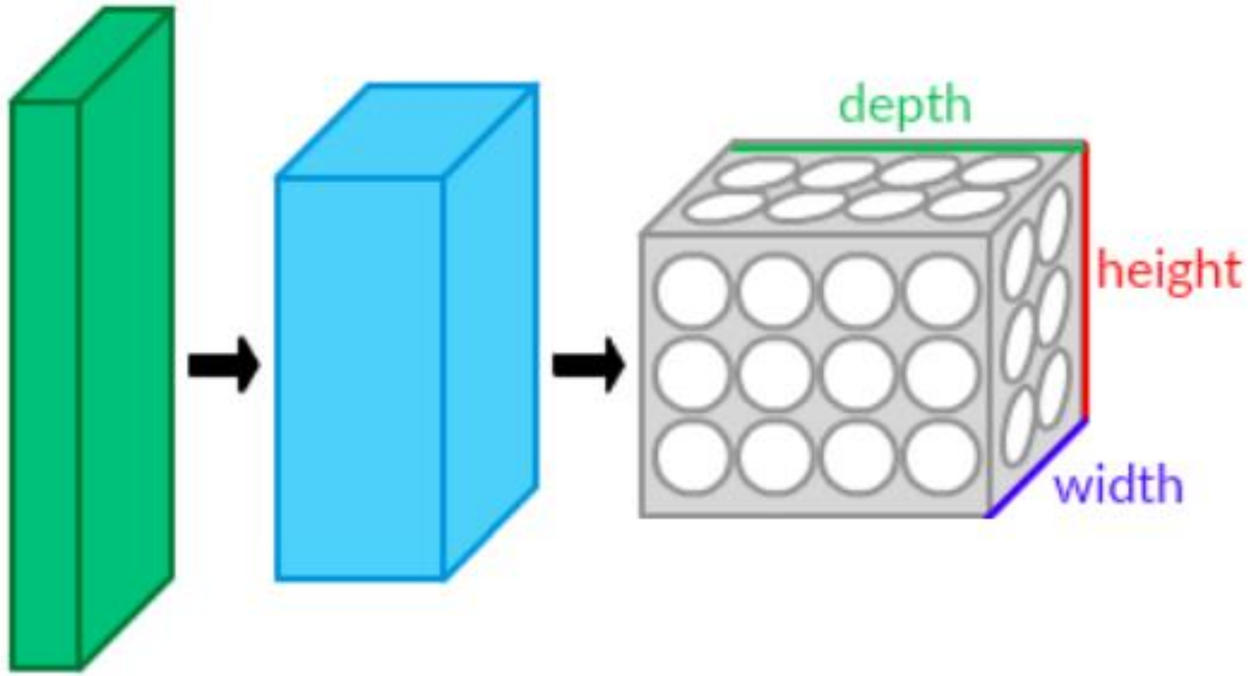


By Aphex34 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45679374

Convolutional Neural Networks (CNN)

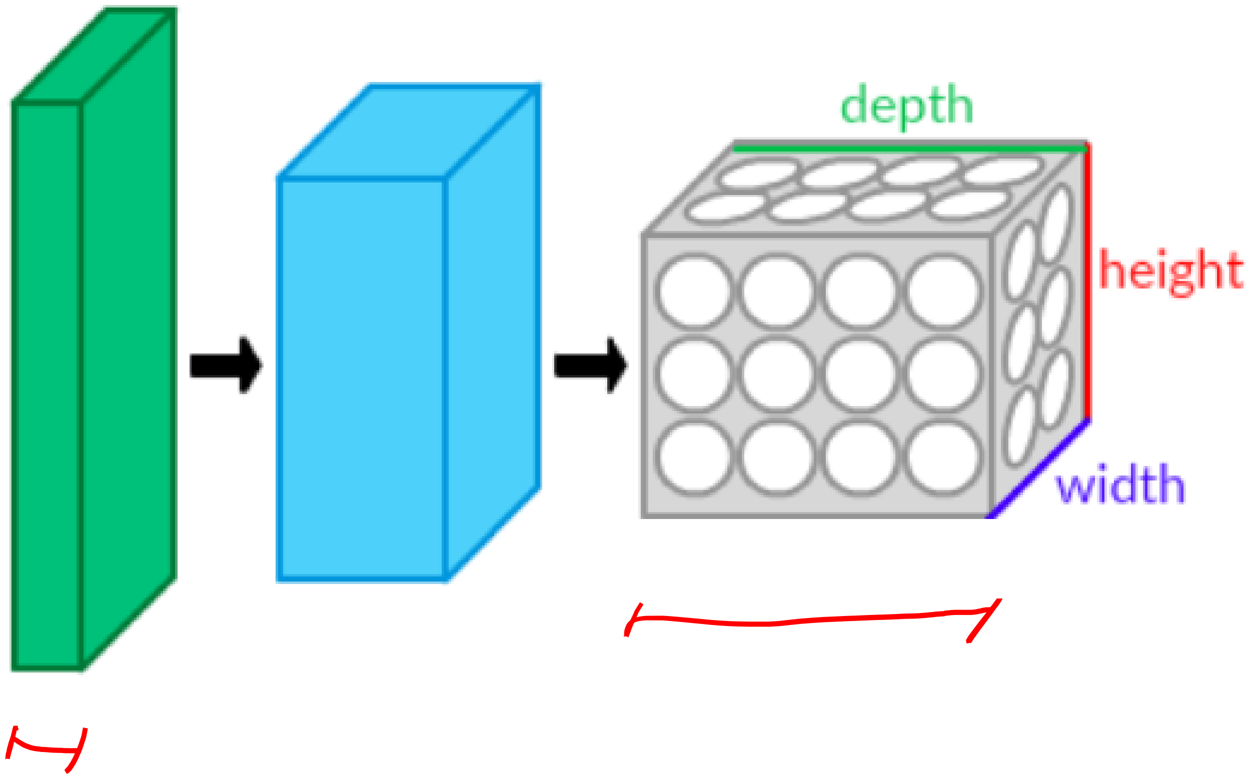
CNN are typically made of blocks that include:

- Convolutional layers
- Nonlinearities (activation functions)
- Maxpooling



Convolutional Neural Networks (CNN)

- An image passing through a CNN is transformed in a sequence of volumes.
- As the depth increases, the height and width of the volume decreases
- Each layer takes as input and returns a volume



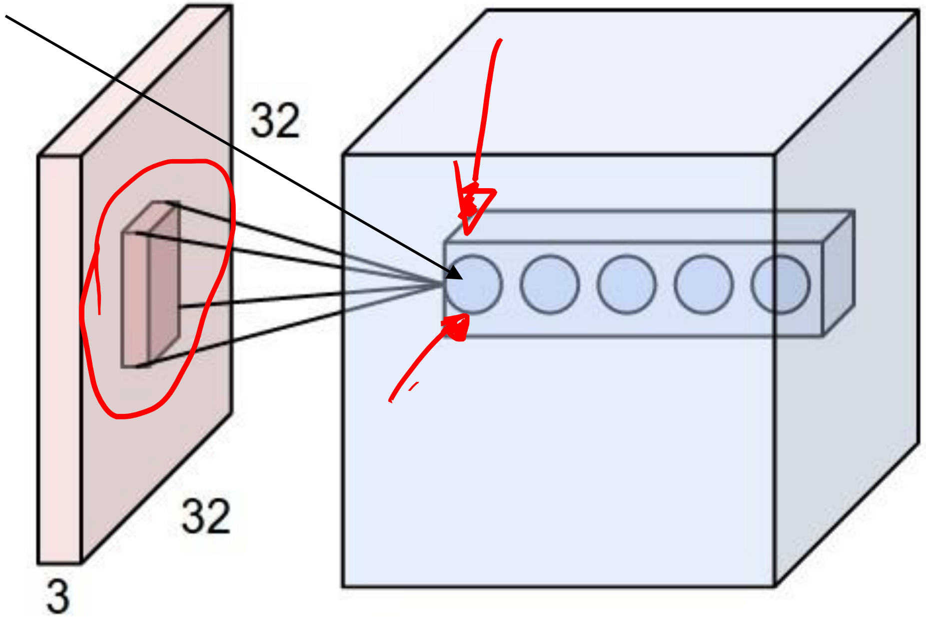
By Aphex34 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45661858

Convolutional Layers

Convolutional layers "mix" all the input components

The output is a linear combination of all the values in a region of the input

$$y = \sum_i w_{i,j}^1 \, x_i + b^1$$



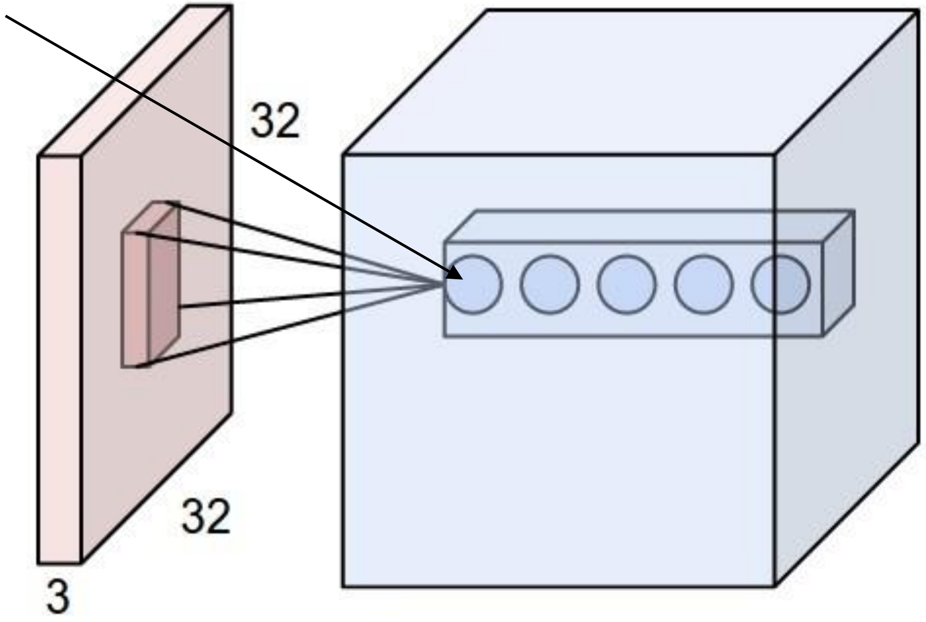
Convolutional Layers

Convolutional layers "mix" all the input components

The output is a linear combination of all the values in a region of the input

$$y = \sum_i \boxed{w_{i,j}^1} x_i + \boxed{b^1}$$

The parameters of this layer are called filters.



Convolutional Layers

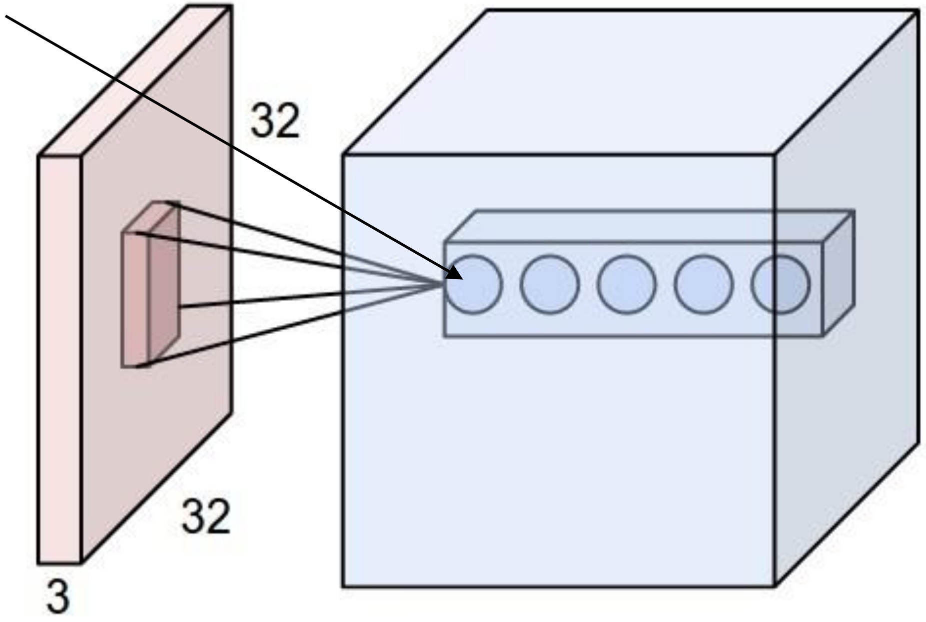
Convolutional layers "mix" all the input components

The output is a linear combination of all the values in a region of the input

$$y = \sum_i \boxed{w_{i,j}^1} x_i + \boxed{b^1}$$

The parameters of this layer are called filters.

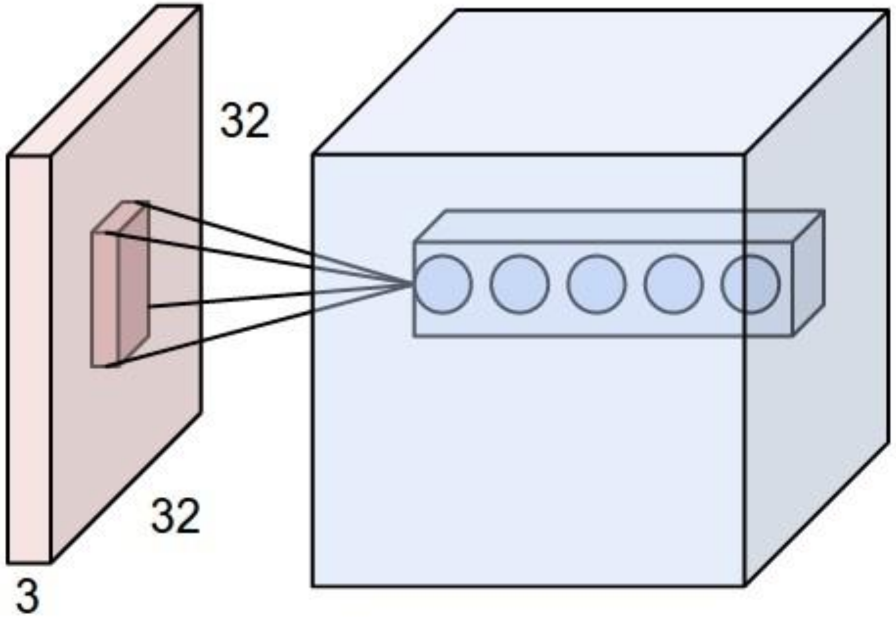
The same filter is used through the whole spatial extent of the input.



Convolutional Layers, Remarks:

- Convolutional Layers are described by a set of filters.
- Filters represent the weights of these linear combination.
- Filters typically have very small spatial extent and large depth extent.

The output of the convolution against a filter becomes a slice in the volume feed to the next layer

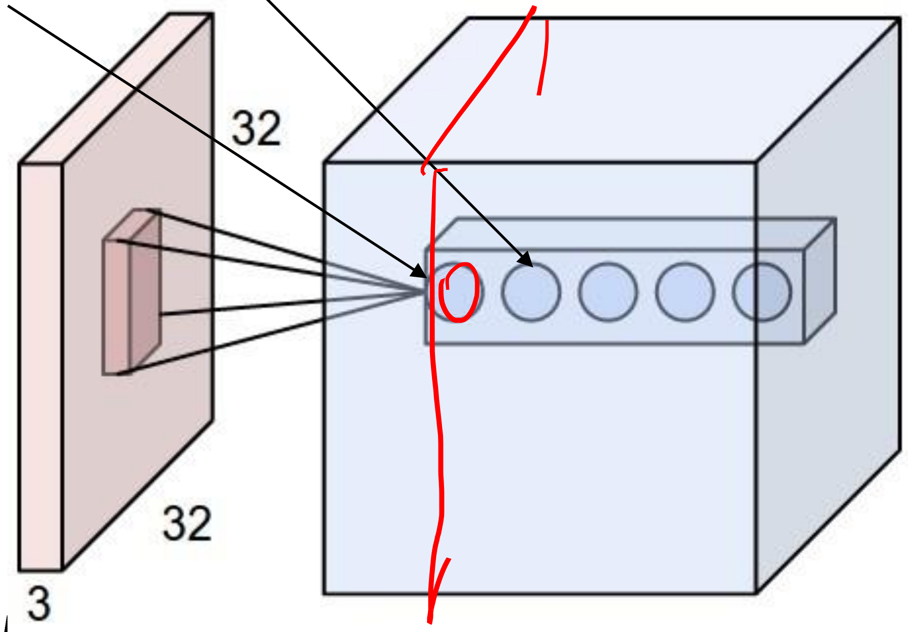


Convolutional Layers, Remarks:

- Each filter corresponds to a different layer in the output

$$\sum_i w_{i,j}^1 \, x_i + b^1$$

$$\sum_i w_{i,j}^2 \, x_i + b^2$$

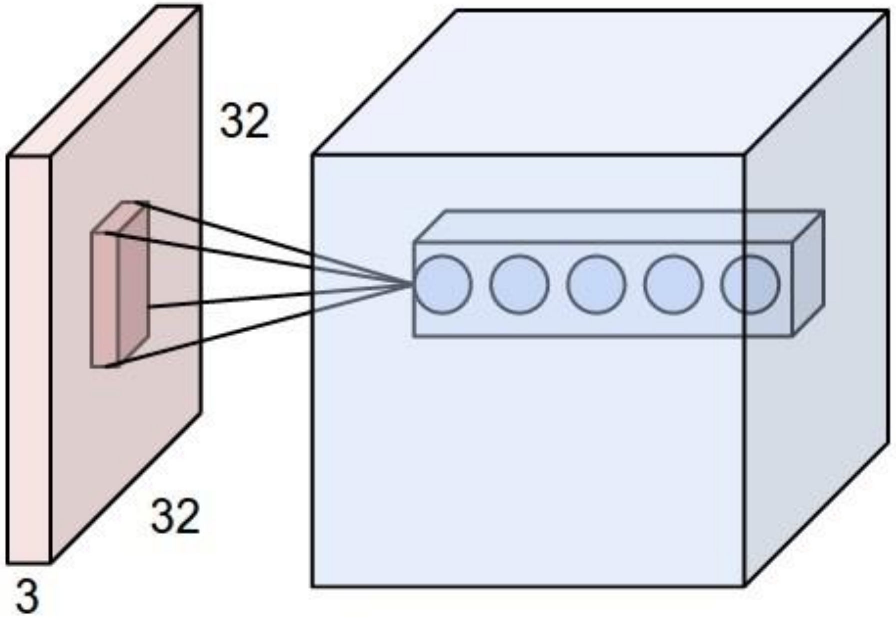


Convolutional Layers, Remarks:

- Each filter corresponds to a different layer in the output
- Each filter has depth equal to the depth of the input volume
- Convolutional layers "mix" all the input components
- The output is also called volume or activation maps

3x3x5

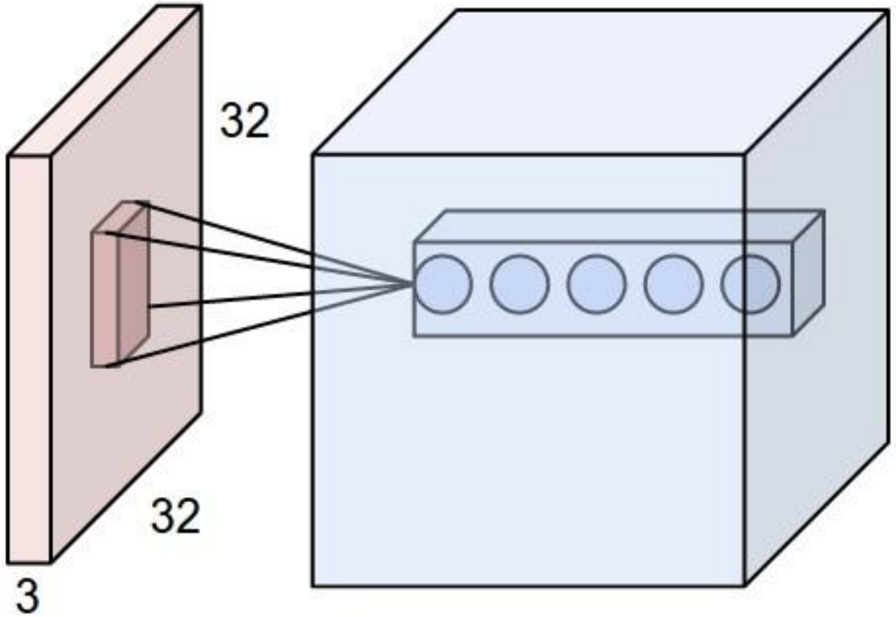
5



By Aphex34 - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=45659236

Convolutional Layers, Remarks:

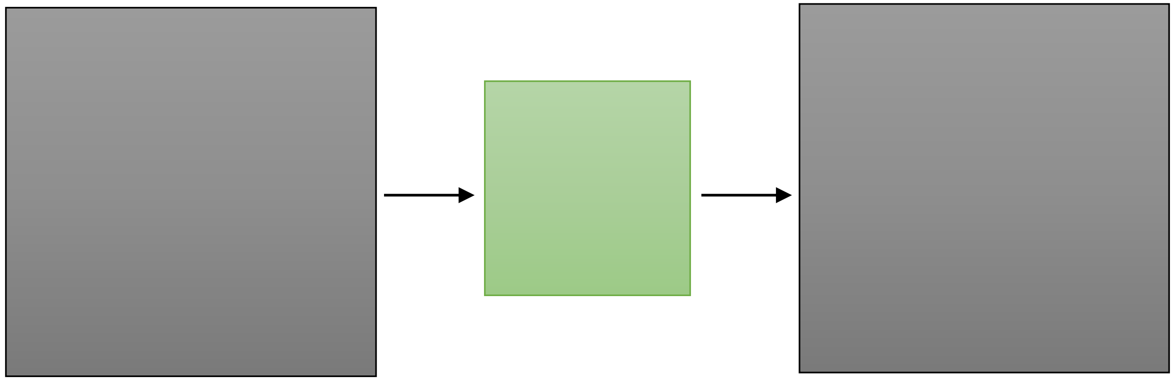
Remarks: The depth of each filter is never mentioned when designing a CNN, because this cannot be adjusted but necessarily correspond to the layers of the input volume



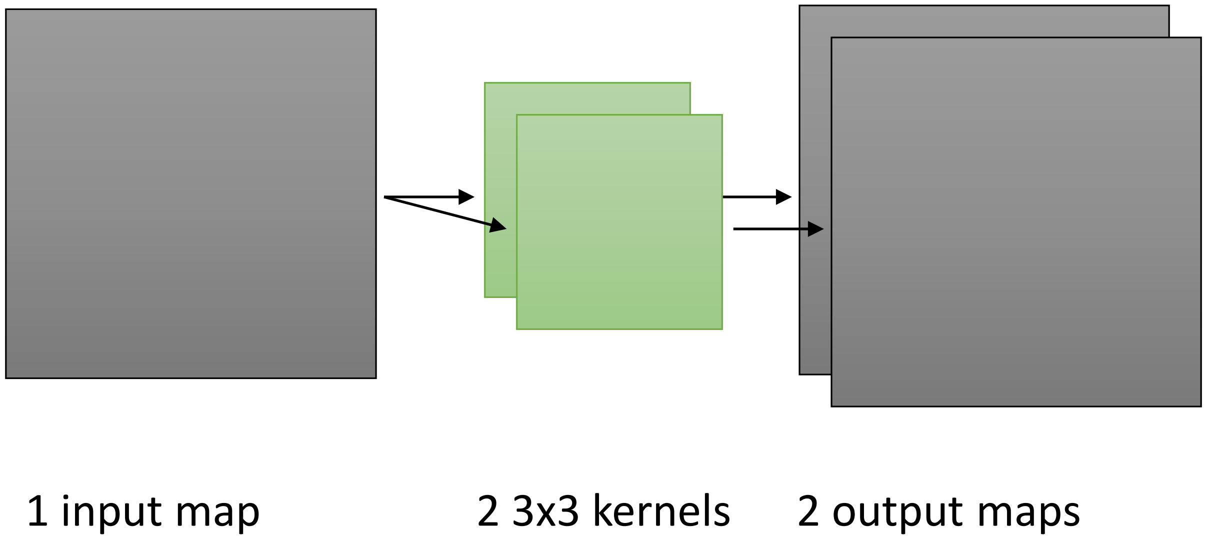
By Aphex34 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45659236

i

CNN Arithmetic

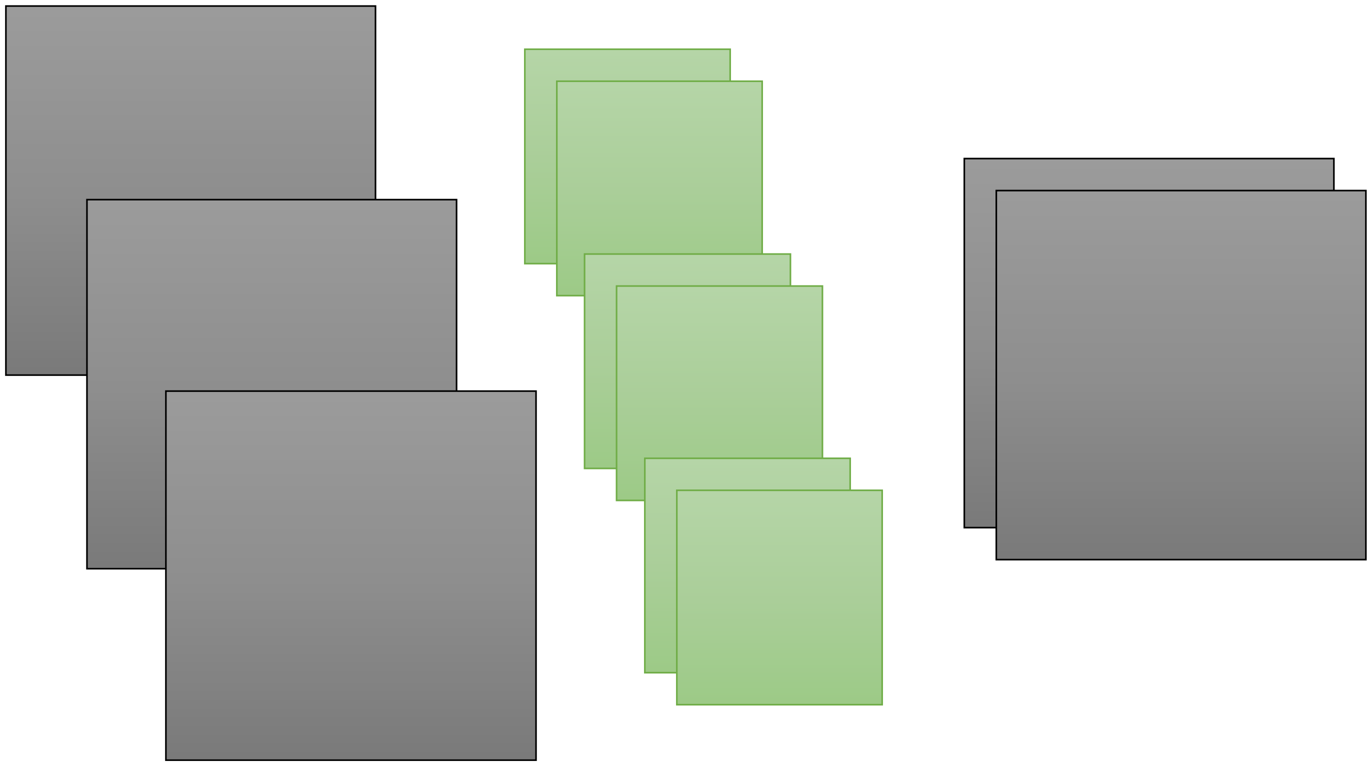


CNN Arithmetic



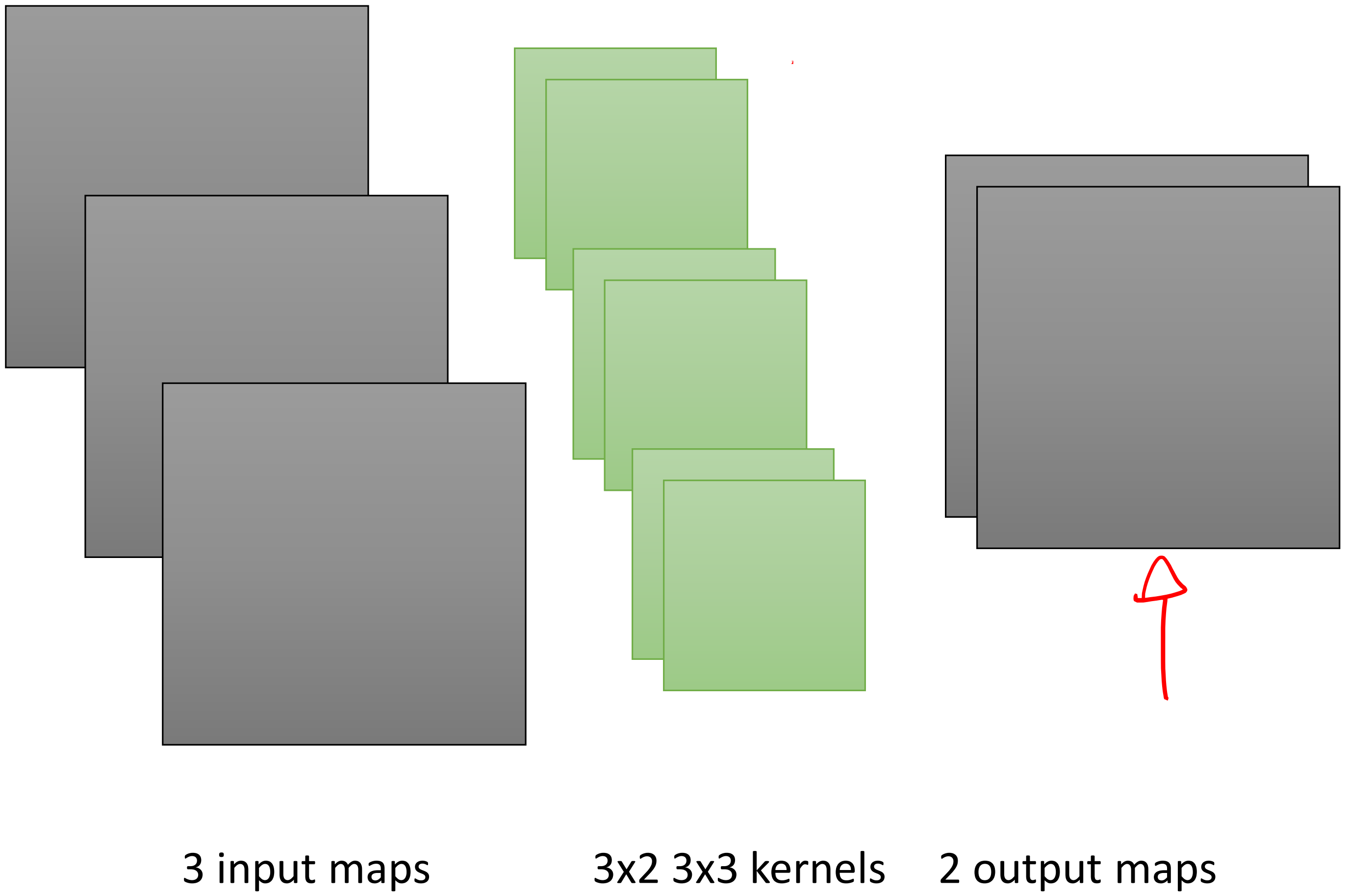
CNN Arithmetic

2 kernels

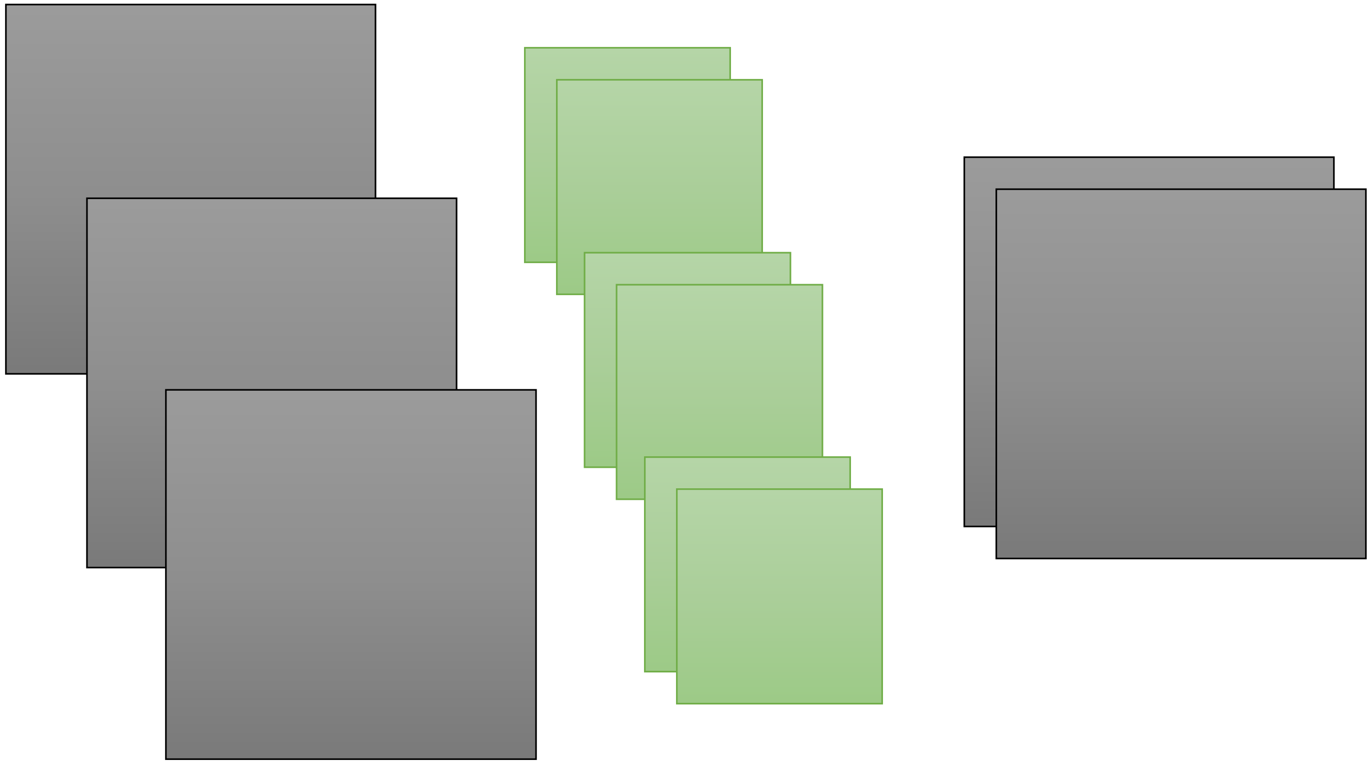


3 input maps   3x2 3x3 kernels   2 output maps

CNN Arithmetic

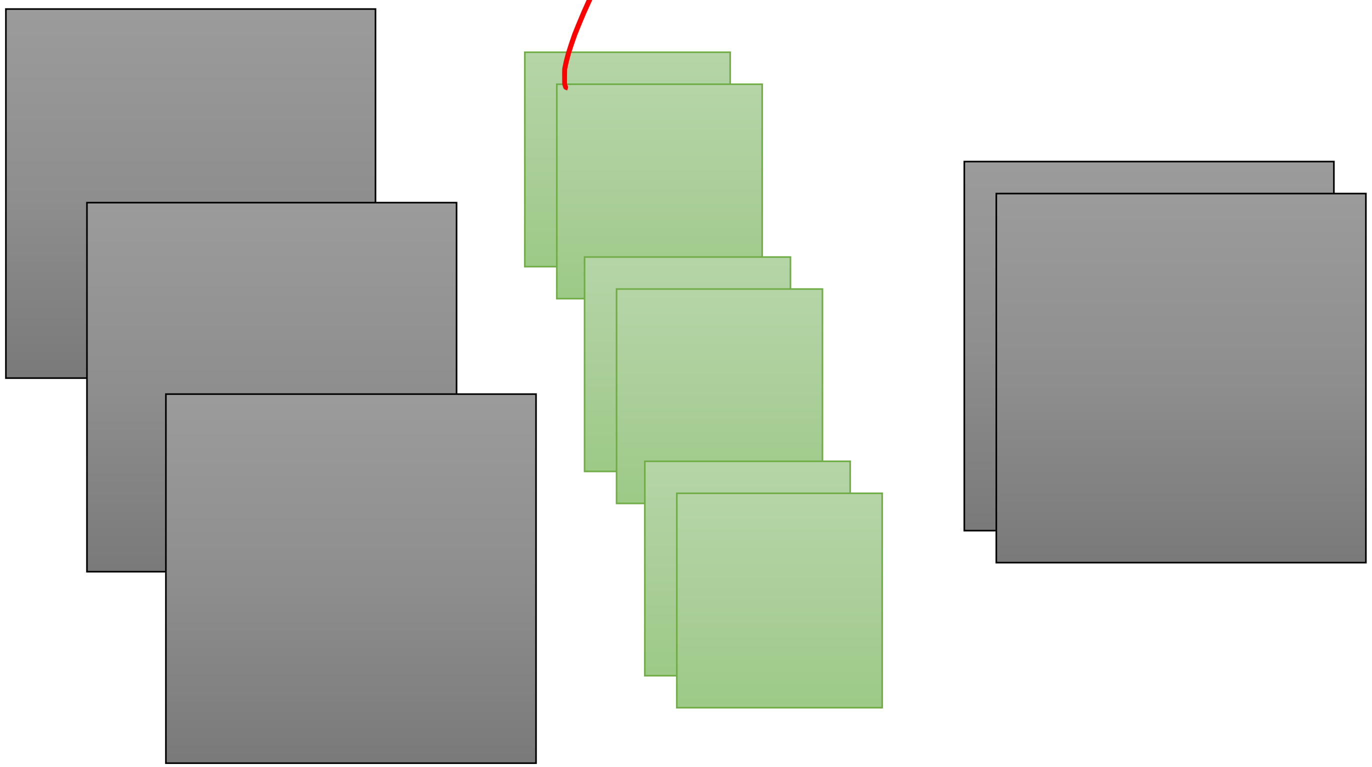


CNN Arithmetic



CNN Arithmetic

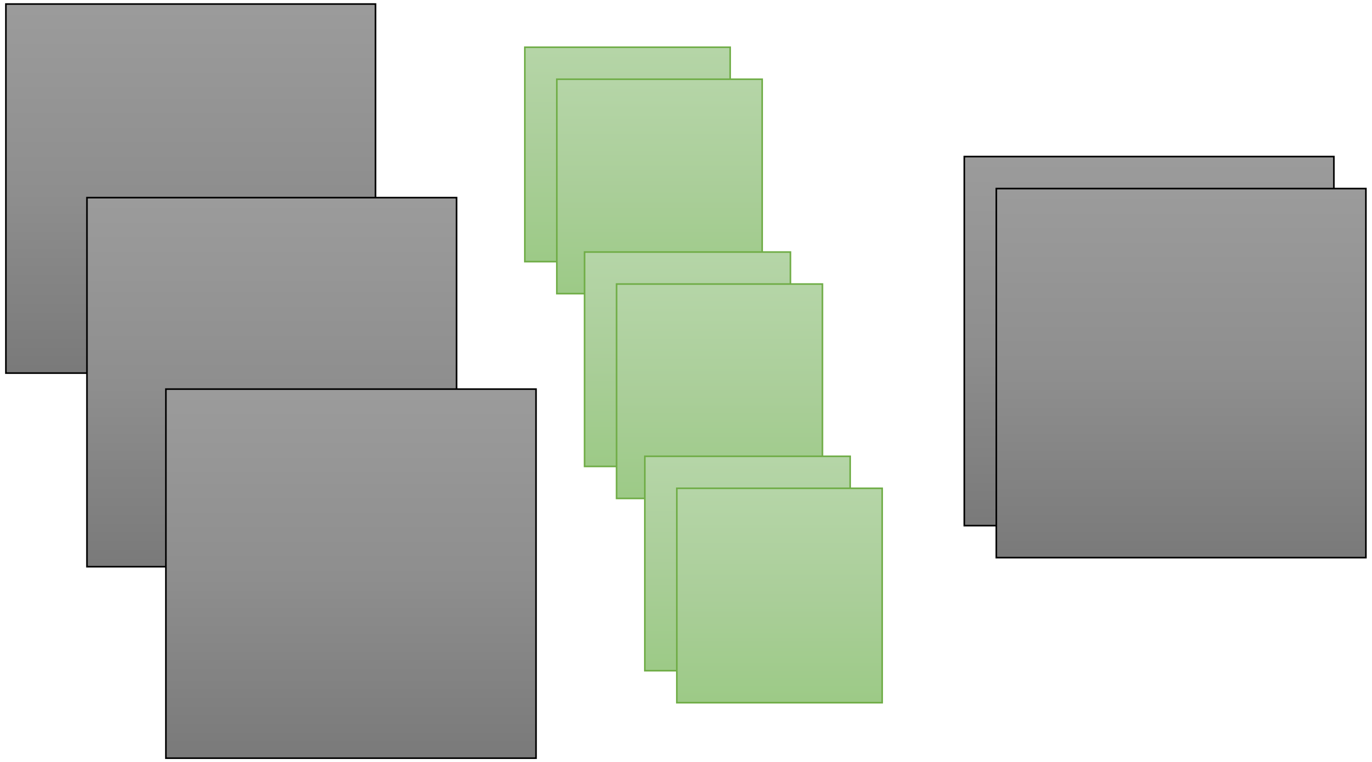
3x3 $9 \times 6 = 54$



3 input maps 3x2 3x3 kernels 2 output maps

Quiz: how many parameters does this layer have?

CNN Arithmetic

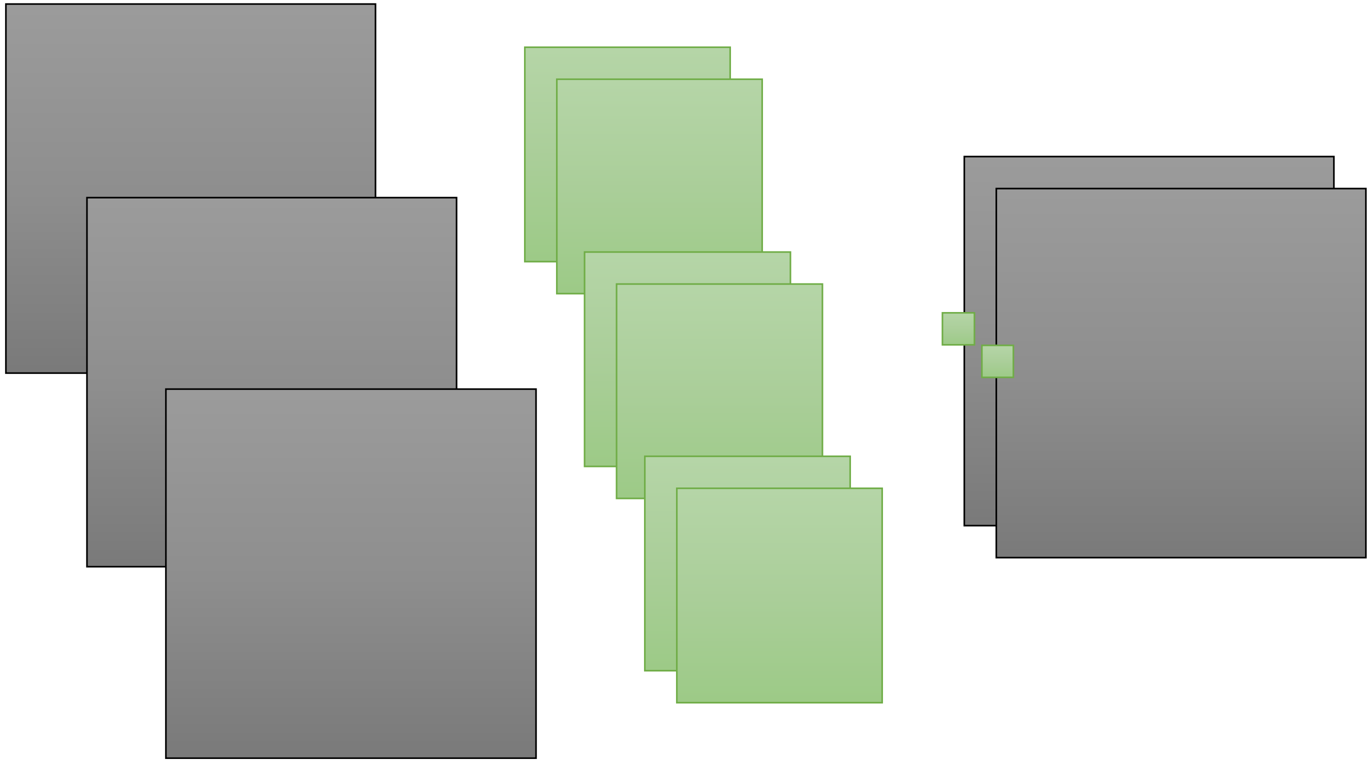


3 input maps

3x2 3x3 kernels
= 54 ...

2 output maps

CNN Arithmetic

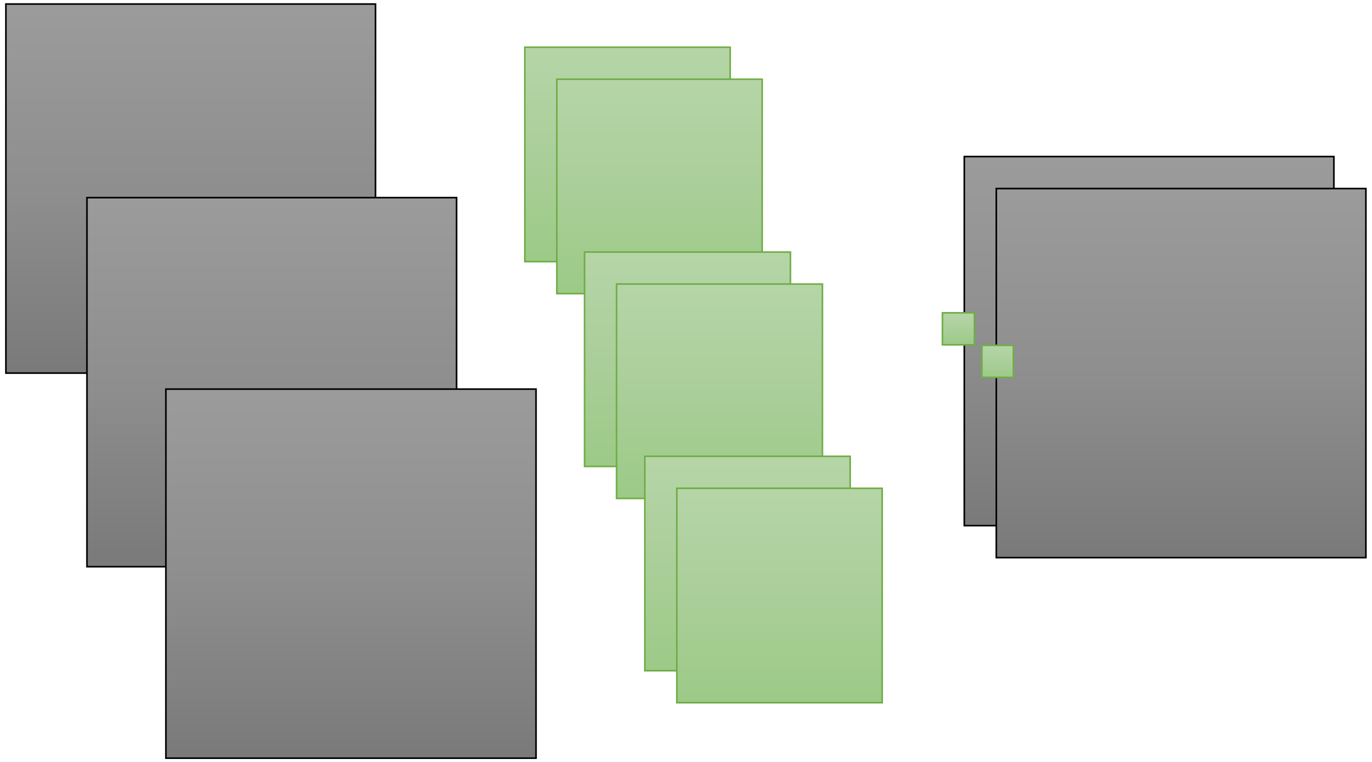


3 input maps

3x2 3x3 kernels
= 54 ...

2 output maps
+ 2 biases

CNN Arithmetic



3 input maps   3x2 3x3 kernels   2 output maps

= 54 ...   + 2 biases

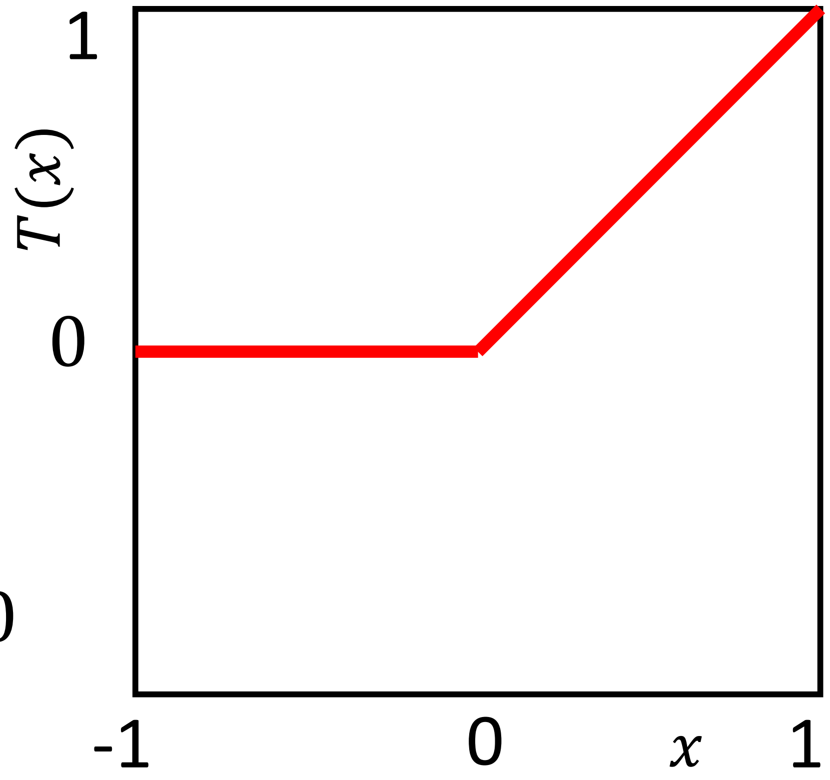
= 56 trainable parameters (weights)

Activation Layers

Introduce nonlinearities in the network, otherwise the CNN might be equivalent to a linear classifier…

RELU (Rectifier Linear Units): it's a thresholding on the feature maps, i.e., a $\max(0,\cdot)$ operator.

- By far the most popular activation function in deep NN (since when it has been used in AlexNet)
- Dying neuron problem: a few neurons become insensitive to the input (vanishing grandient problem)



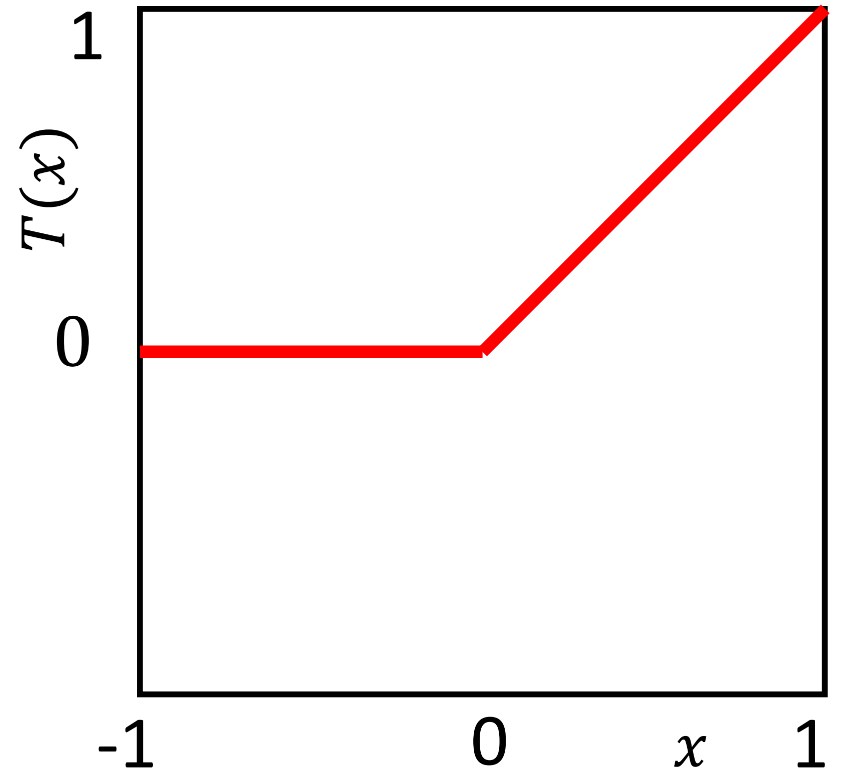
$$T(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$$

Activation Layers

Introduce nonlinearities in the network, otherwise the CNN might be equivalent to a linear classifier…

LEAKY RELU: like the relu but include a small slope for negative values

$$T(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0.01 * x & \text{if } x < 0 \end{cases}$$



Activation Layers

Introduce nonlinearities in the network, otherwise the CNN might be equivalent to a linear classifier…

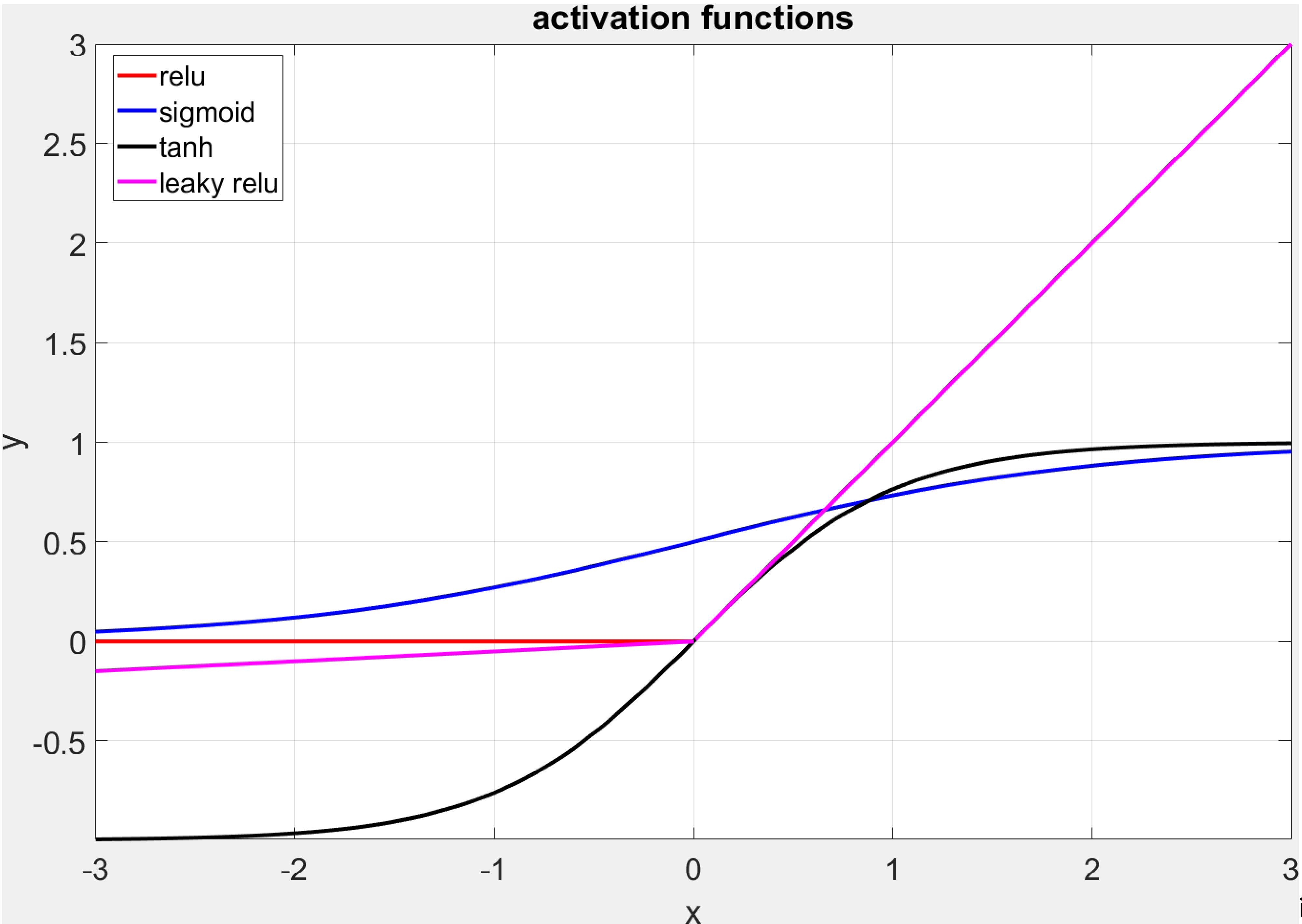
TANH (hyperbolic Tangent): has a range (-1,1), continuous and differentiable

$$T(x) = \frac{2}{1 + e^{-2x}} - 1$$

SIGMOID: has a range (0,1), continuous and differentiable

$$S(x) = \frac{1}{1 + e^{-2x}}$$

These activation functions are mostly popular **in MLP architectures**



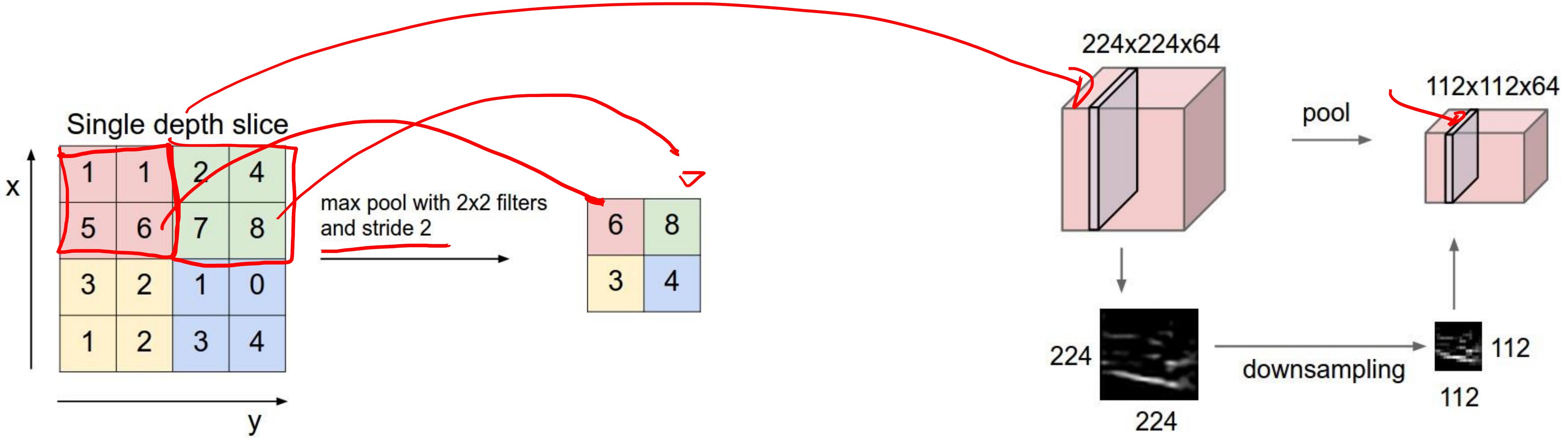
iacomo Boracchi

Pooling Layers

Pooling Layers reduce the **spatial** size of the volume.

The Pooling Layer operates **independently on every depth slice** of the input and **resizes it spatially, often using the MAX operation.**

In a 2x2 support it discards 75% of samples in a volume



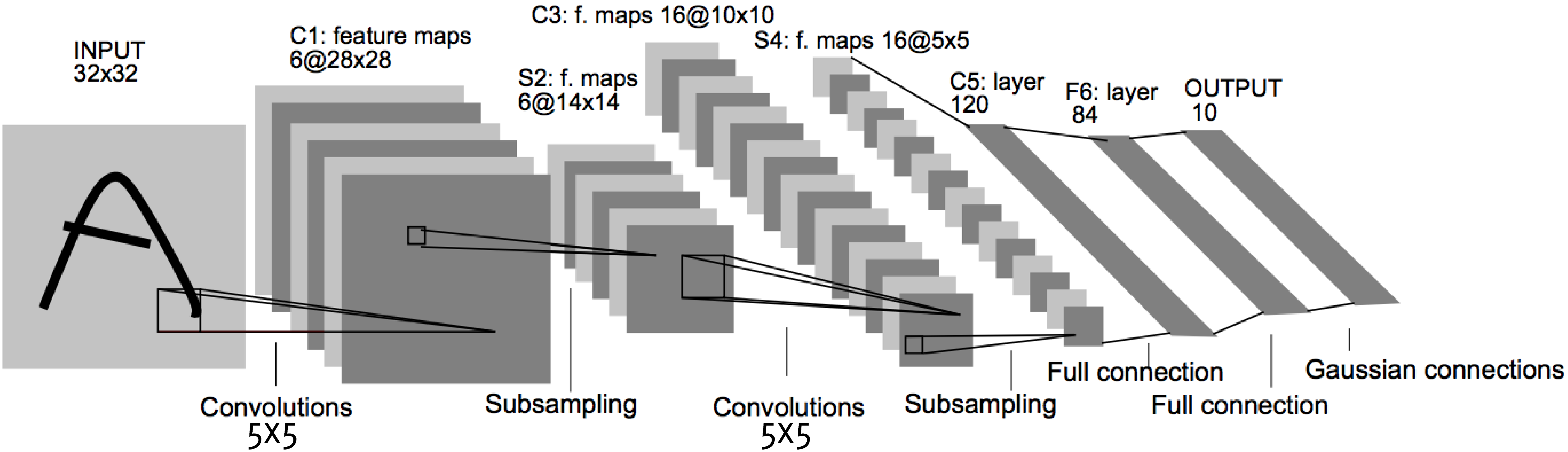
CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

The First CNN

LeNet-5 (1998)

Stack of Conv2D + RELU + ~~MAX~~ AVG-POOLING

A TRADITIONAL MLP



LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998)

The first CNN

Do not use each pixel as a separate input of a large MLP, because:

- images are highly spatially correlated,
- using individual pixel of the image as separate input features would not take advantage of these correlations.

The first convolutional layer: 6 filters 5x5

The second convolutional layer: 16 filters 5x5

model.summary()

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 6)         156
_________________________________________________________________
average_pooling2d_1 (Average (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 16)        2416
_________________________________________________________________
average_pooling2d_2 (Average (None, 5, 5, 16)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_1 (Dense)              (None, 120)               48120
_________________________________________________________________
dense_2 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_3 (Dense)              (None, 10)                850
=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
```

model.summary()

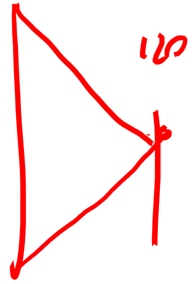
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 6)         156 (6 x 5 x 5 + 6)
_________________________________________________________________
average_pooling2d_1 (Average (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 16)        2416 (16 x 5 x 5 x 6 + 16)
_________________________________________________________________
average_pooling2d_2 (Average (None, 5, 5, 16)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_1 (Dense)              (None, 120)               48120
_________________________________________________________________
dense_2 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_3 (Dense)              (None, 10)                850
=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
```

5x5

6

120 x 400 + 120

400



Input is a grayscale image

The input is a volume having depth = 6

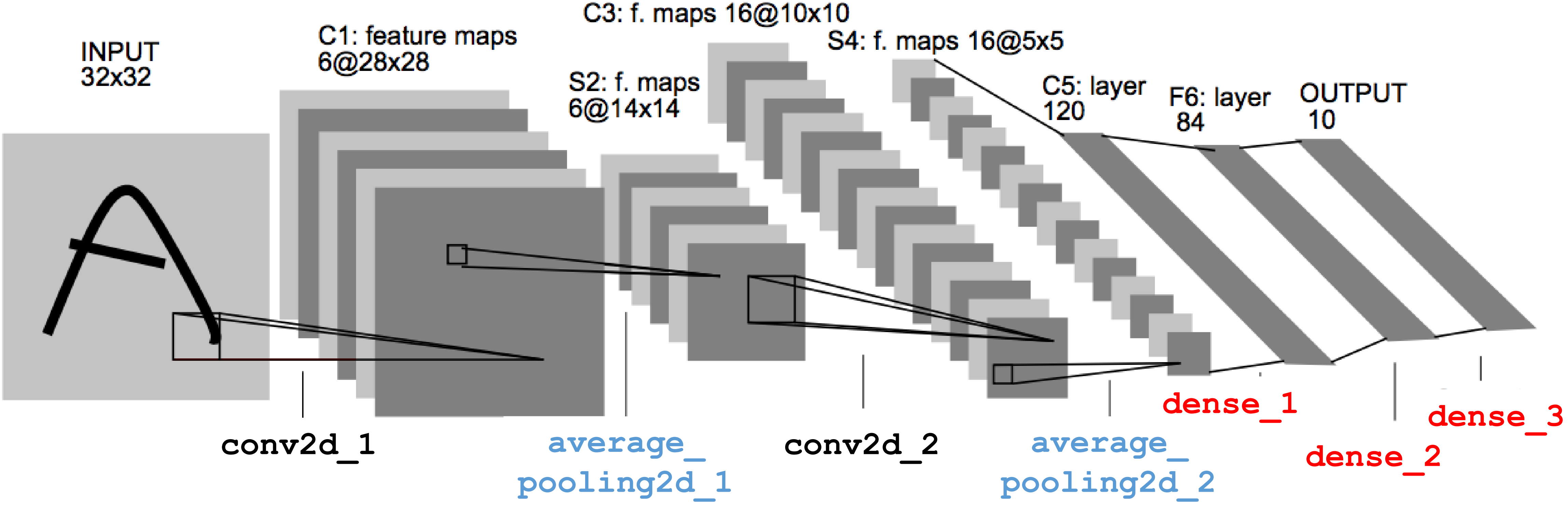
Most parameters are still in the MLP

Giacomo Boracchi

LeNet-5 (1998)

Stack of Conv2D + RELU + MAX-POOLING

A TRADITIONAL MLP



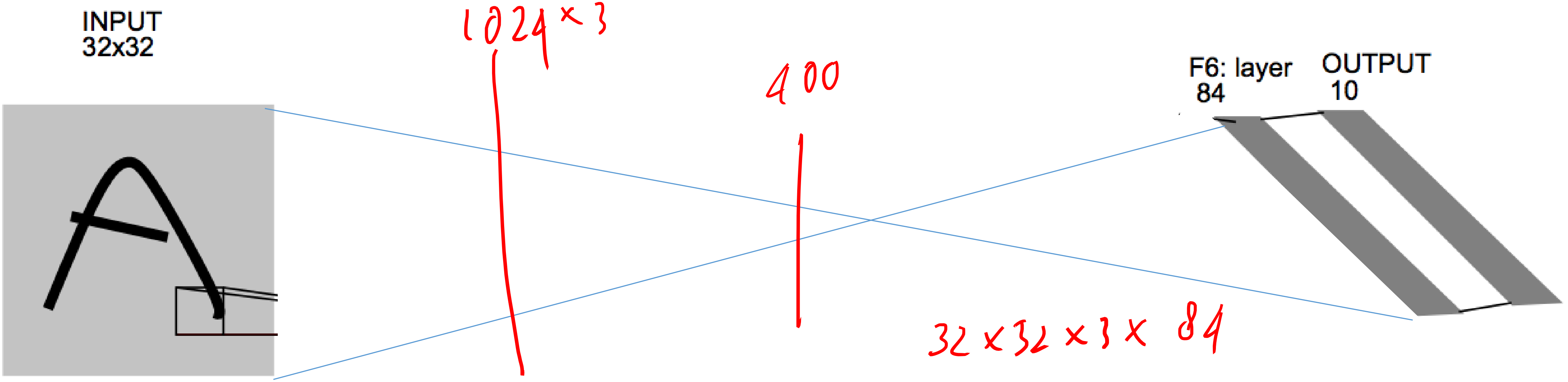
LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998)

Most of parameters are in MLP

What about a MLP taking as input the whole image?

Input 32 x 32 = 1024 pixels, fed to a 84 neurons (the last FC layers of the network) -> 86950 parameters: 1024 * 84 + 84 + 84 * 10 + 10

60K



Most of parameters are in MLP

What about a MLP taking as input the whole image?

Input 32 x 32 = 1024 pixels, fed to a 84 neurons (the last FC layers of the network) -> 86950 parameters

But.. If you take an RGB input: 32 x 32 x 3,

6×5×5

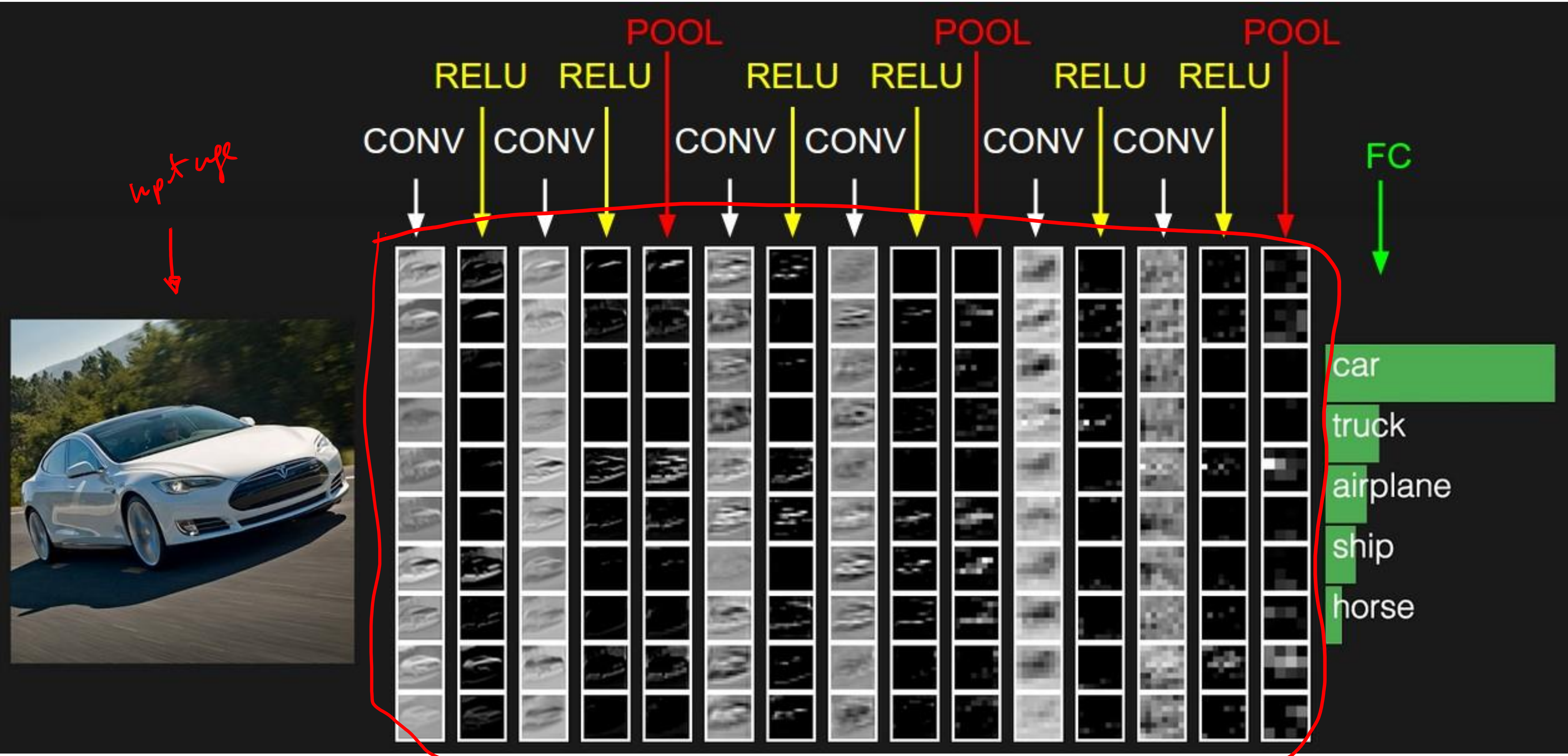
6×5×5×3

CNN: only the nr. of parameters in the first filters increases 156 -> 456

MLP: everything increases by a factor 3

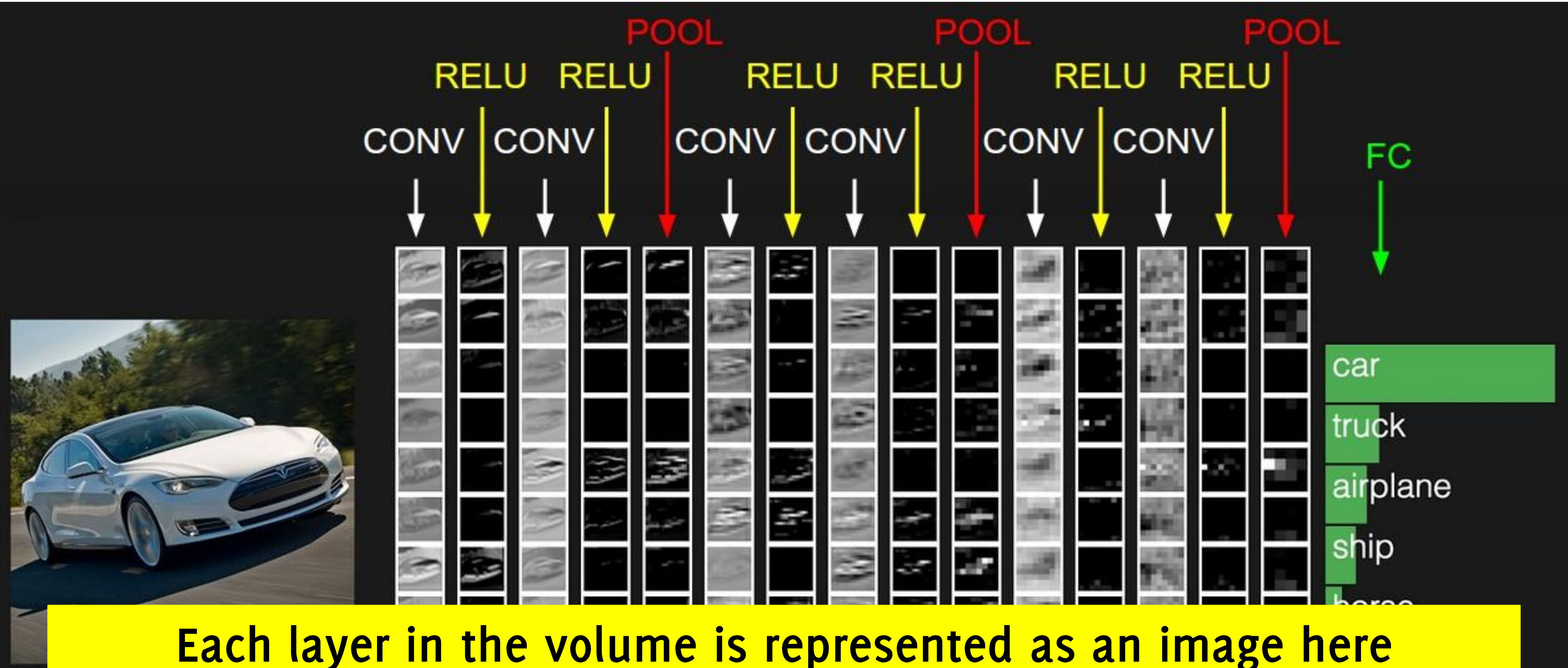
CNN «in action»

Activations in a convolutional network



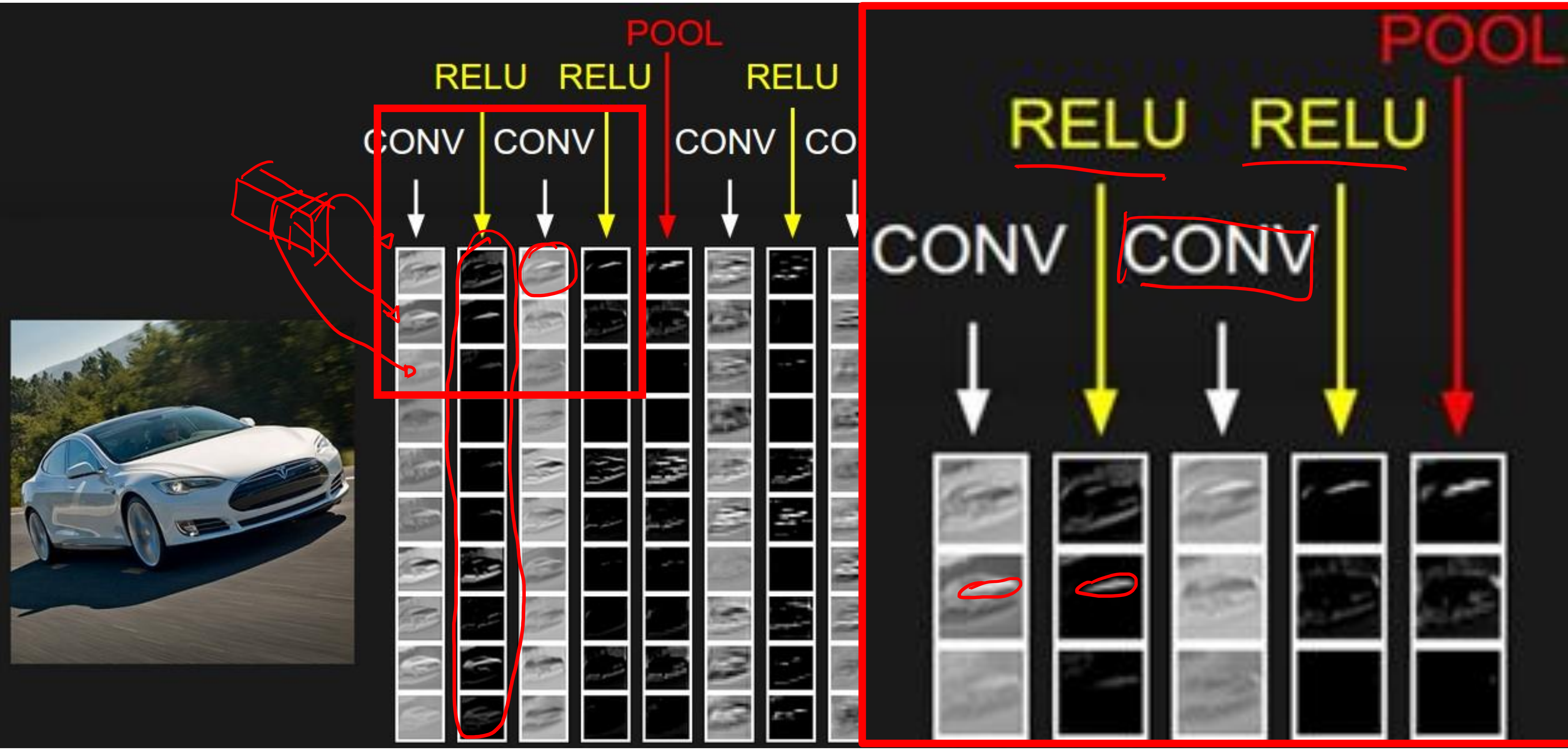
CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Activations in a convolutional network



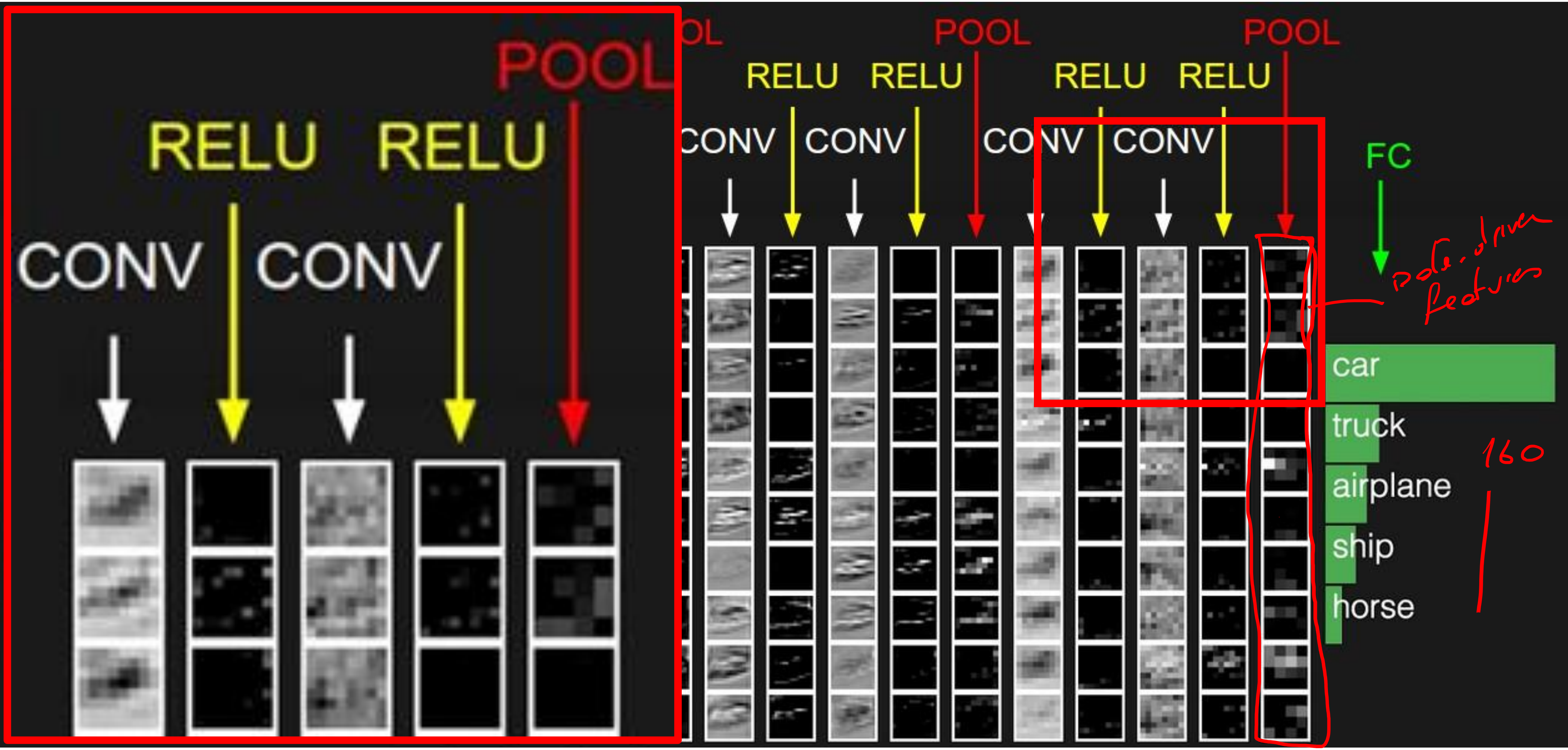
Each layer in the volume is represented as an image here
(using the same size but different resolution for visualization sake)

Activations in a convolutional network



[CS231n: Convolutional Neural Networks for Visual Recognition](http://cs231n.github.io/) http://cs231n.github.io/

Activations in a convolutional network

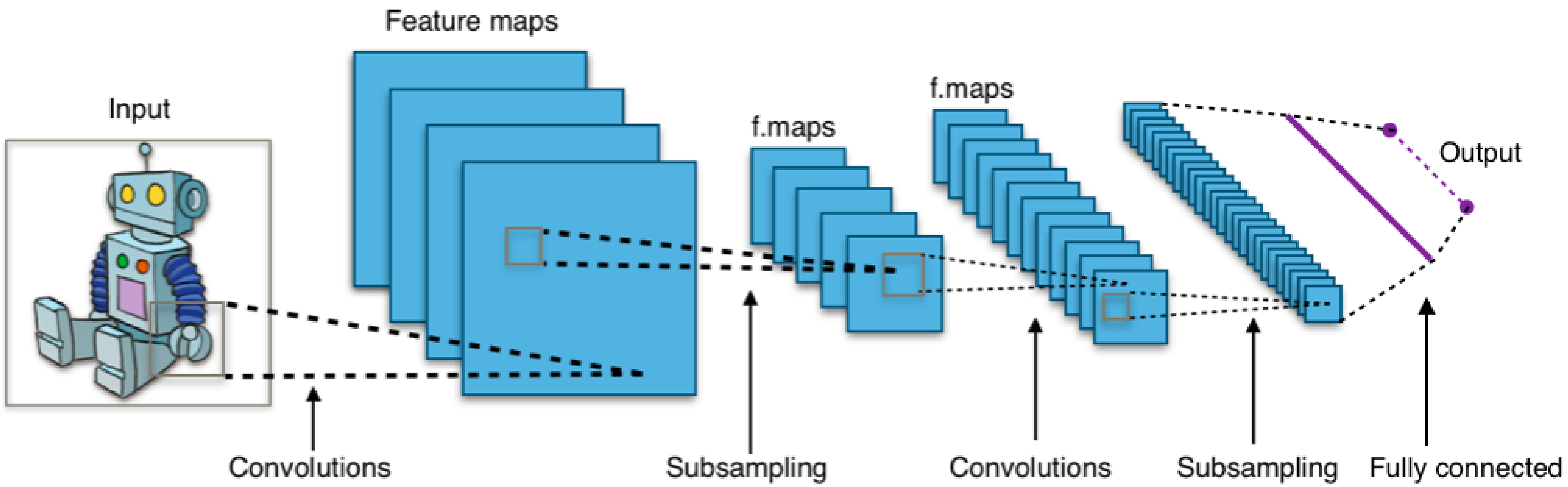


[CS231n: Convolutional Neural Networks for Visual Recognition](http://cs231n.github.io/) http://cs231n.github.io/

CNNs as data-driven feature extractors

Convolutional Neural Networks (CNN)

The typical architecture of a convolutional neural network



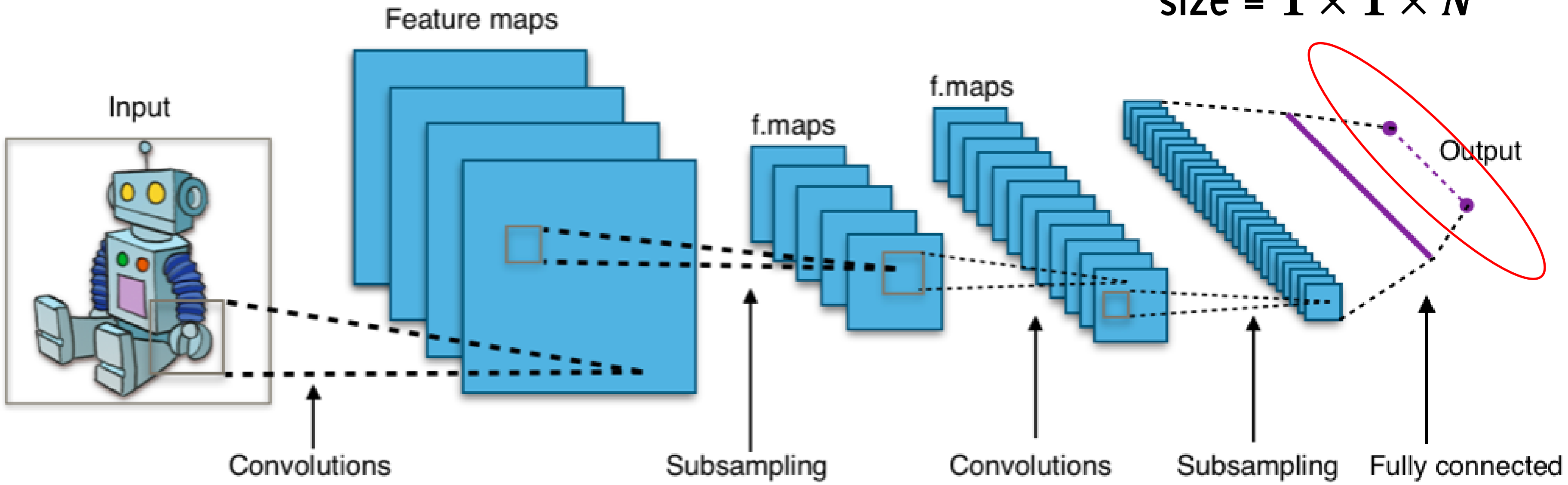
By Aphex34 - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=45679374

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. *"Gradient-based learning applied to document recognition"* Proceedings of the IEEE, 1998 86(11), 2278-2324.

Convolutional Neural Networks (CNN)

At the end, there is a FC layer, namely a neural network

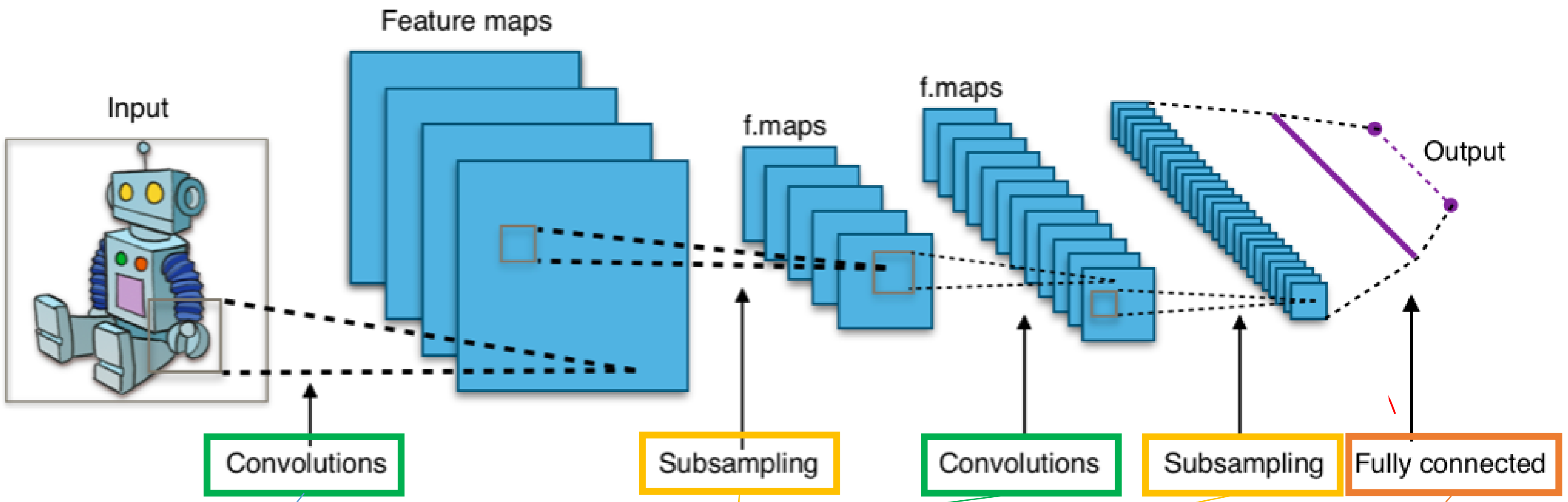
size = $\mathbf{1 \times 1 \times} \boldsymbol{N}$



The output of the **fully connected (FC) layer** has the same size as the **number of classes**, and **provides a score** for the input image to belong to each class

Convolutional Neural Networks (CNN)

At the end, there is a FC layer, namely a neural network

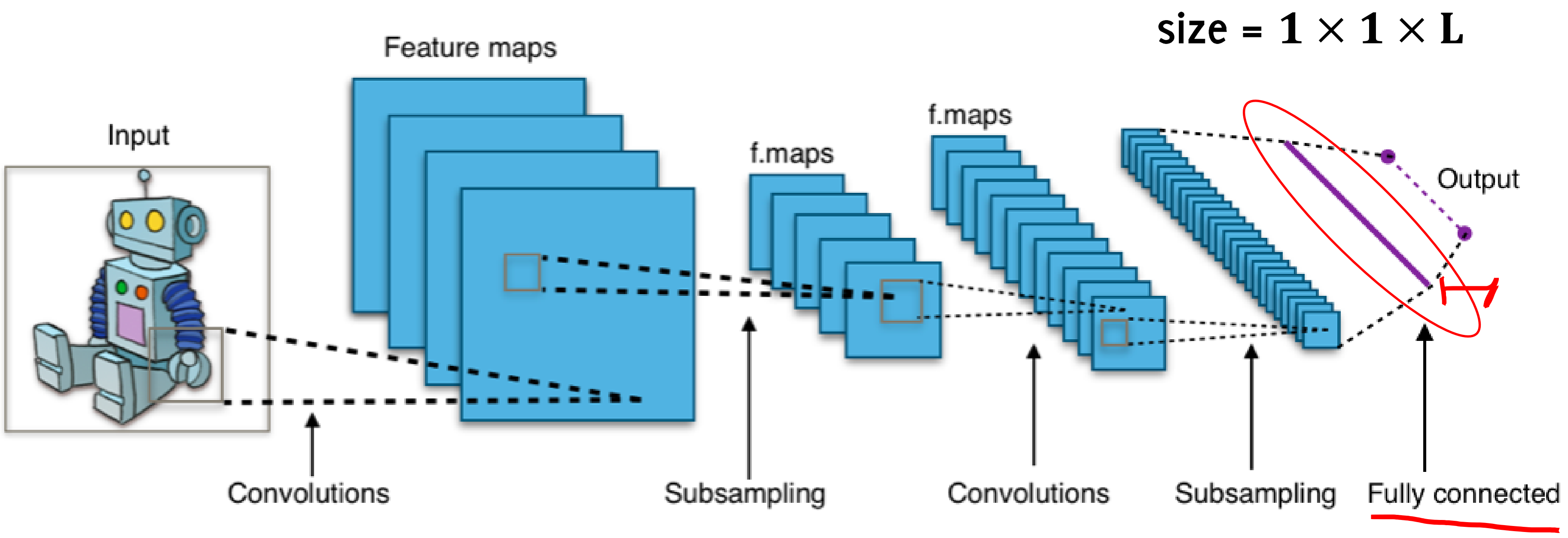


Convolution filters are learned for the classification task at hand

Thresholding + Downsampling (ReLu + Maxpooling)

Fully connected Neural Network providing as output class probabilities

Convolutional Neural Networks (CNN)

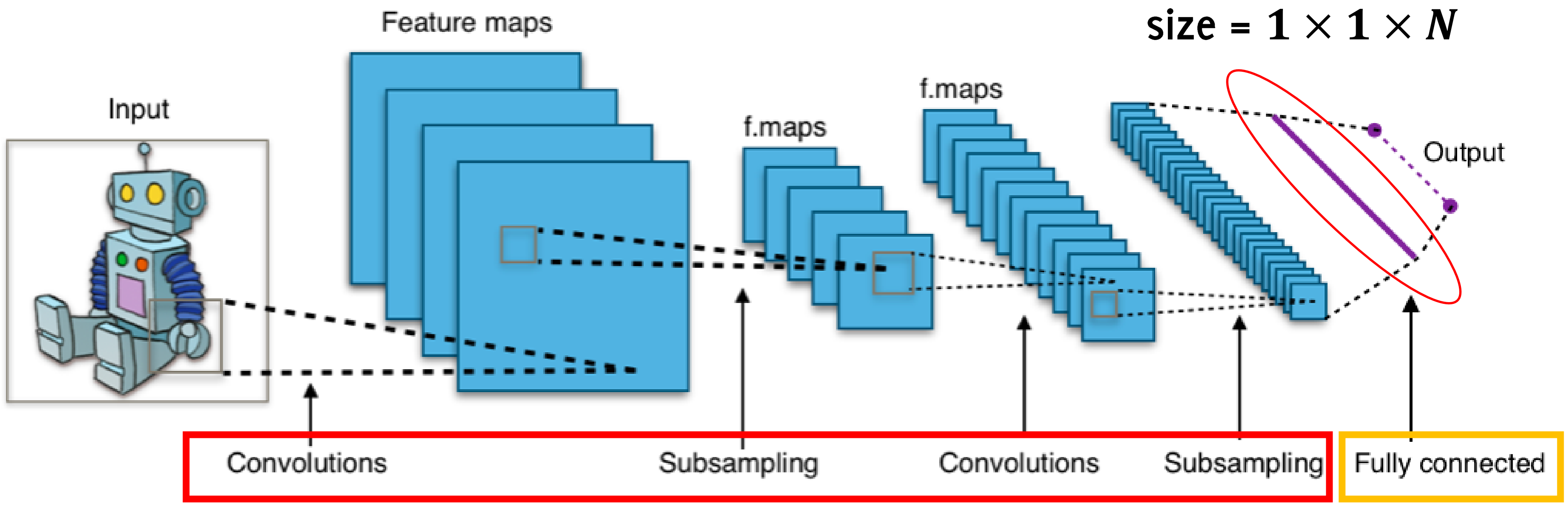


The input of the fully connected layer can be seen as a data-driven descriptor of the input image, i.e. a feature vector that is:

- defined to maximize classification performance
- Trained via backpropagation as the NN they are fed

Convolutional Neural Networks (CNN)

The typical architecture of a convolutional neural network



Data-driven feature extraction

Feature Classifier

Latent representation in CNNs

CIFAR-10 dataset

The CIFAR-10 dataset contains 60000 images:

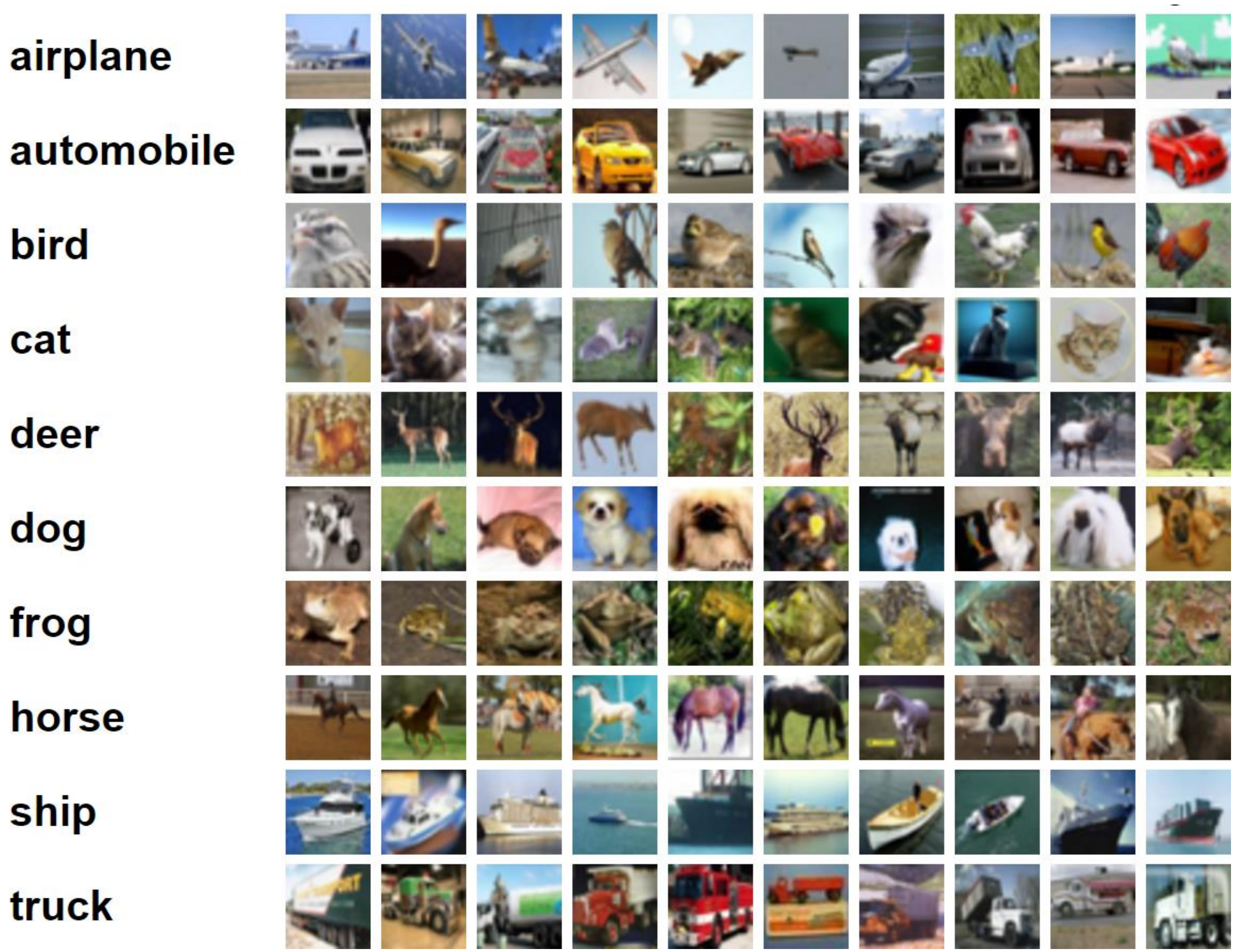
Each image is 32x32 RGB

Images are in 10 classes

6000 images per class

Extremely small images, but high-dimensional:

$$d = 32 \times 32 \times 3 = 3072$$



Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.

Dataset visualization

T-SNE: technique for data visualization that is particularly well suited for the visualization of high-dimensional dataset

It arranges images (data) on a lower dimensional space (2D plane) and images that are nearby on the plane are considered to be close based on some distance metric

Let's do t-SNE using as distance the ℓ^2 distance between input data

$$d(\underline{I_1}, \underline{I_2}) = \|I_1 - I_2\|_2$$

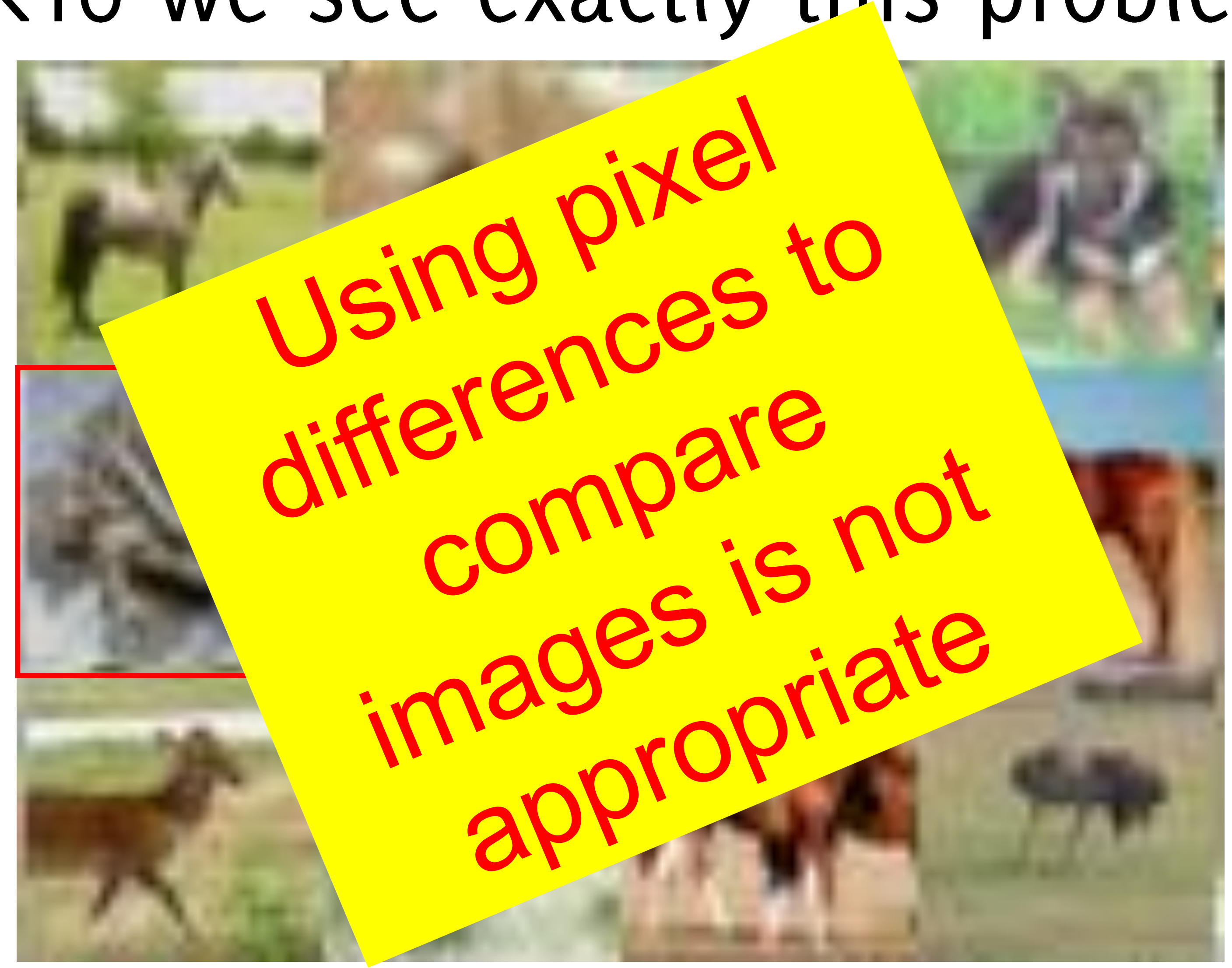
$$\| \square - \square \|_2$$

T-SNE on the whole CIFAR10

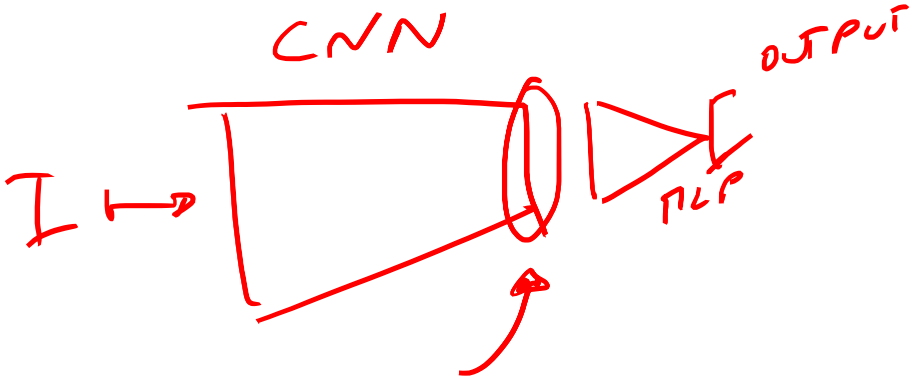
On CIFAR10 we see exactly this problem

[CS231n: Convolutional Neural Networks for Visual Recognition](http://cs231n.github.io/) http://cs231n.github.io/

On CIFAR10 we see exactly this problem



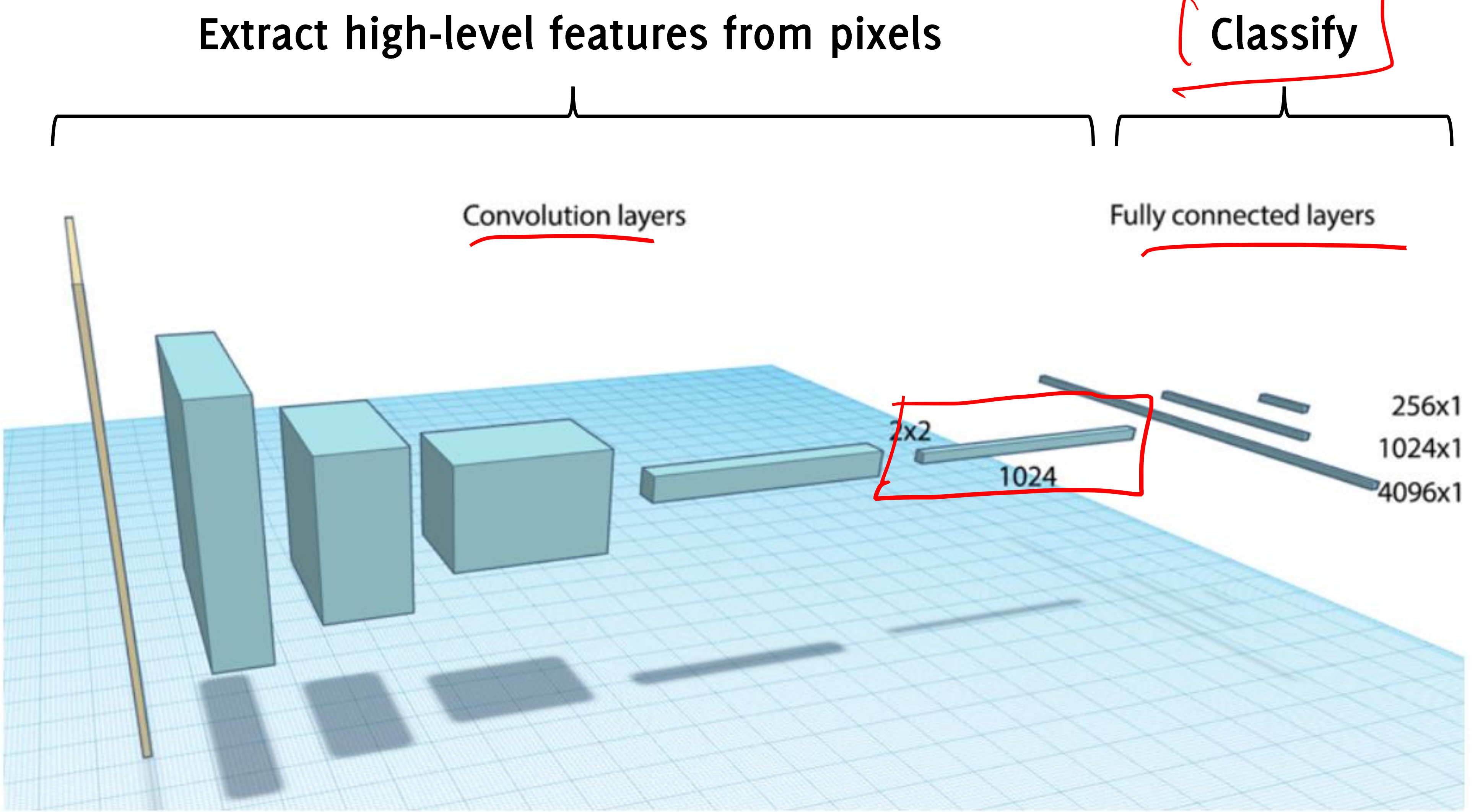
[CS231n: Convolutional Neural Networks for Visual Recognition](http://cs231n.github.io/) http://cs231n.github.io/

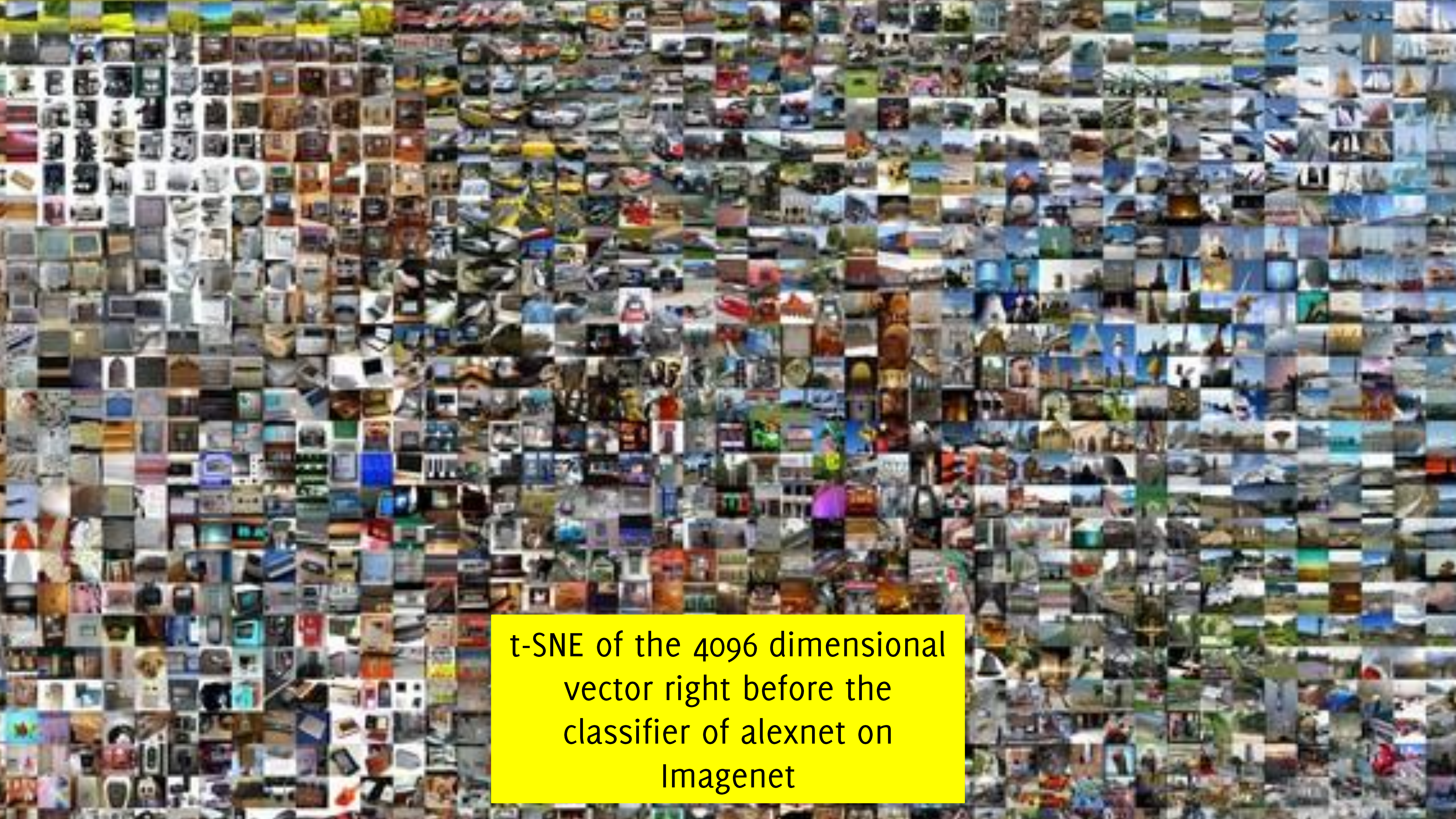


Repeat t-SNE using the latent representation of the CNN (denoted by $\mathrm{conv}_x(\cdot)$)

$$d(I_1, I_2) = \|\underline{\mathrm{conv}_x(I_1)} - \underline{\mathrm{conv}_x(I_2)}\|_2$$

A Typical Architecture





t-SNE of the 4096 dimensional vector right before the classifier of alexnet on Imagenet


CIFAR
100

Distances in the latent representation of a CNN are much more meaningful than data itself

Peculiarities of CNNs

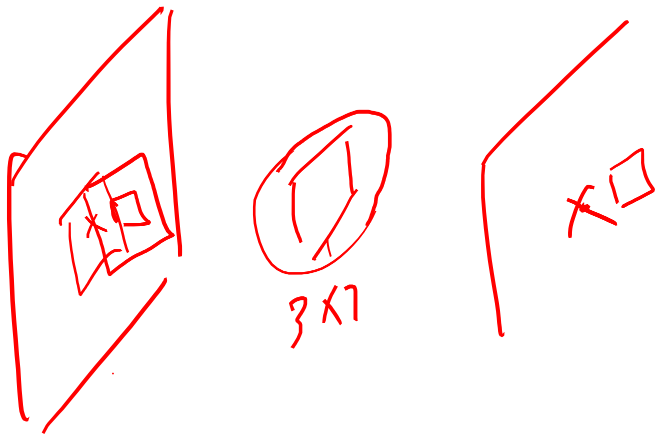
Parameters Definition

Convolutions as MLP

Convolution is a linear operation!

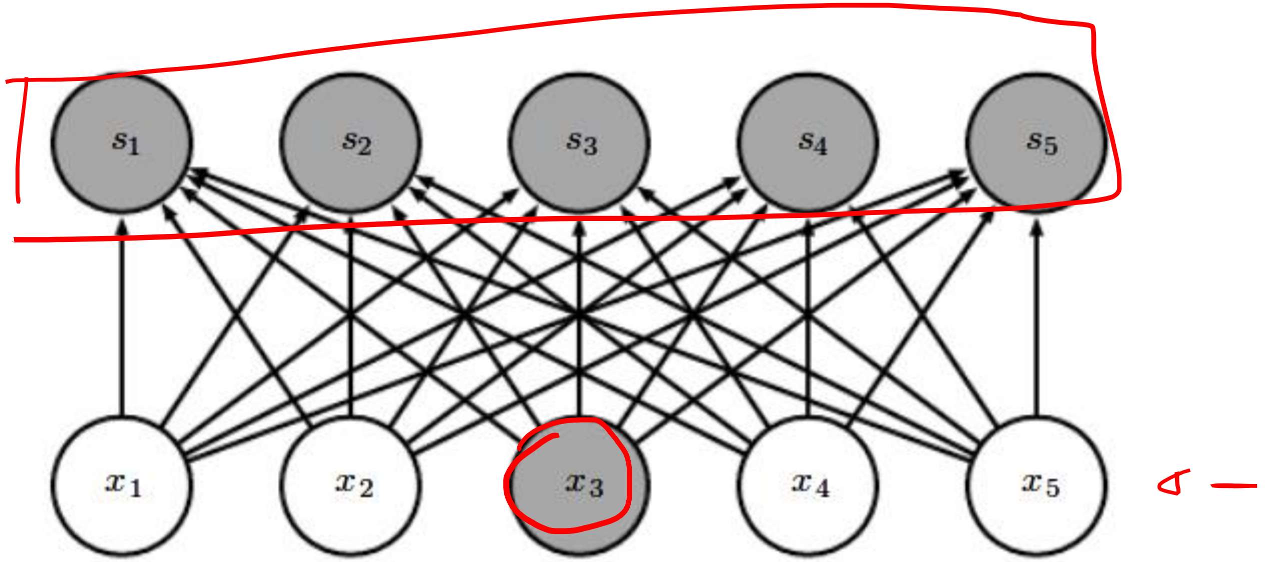
Therefore, If you unroll the input image to a vector, you can consider convolution weights as the weights of a Multilayer Perceptron Network

What are the differences between MLP and CNNs then?

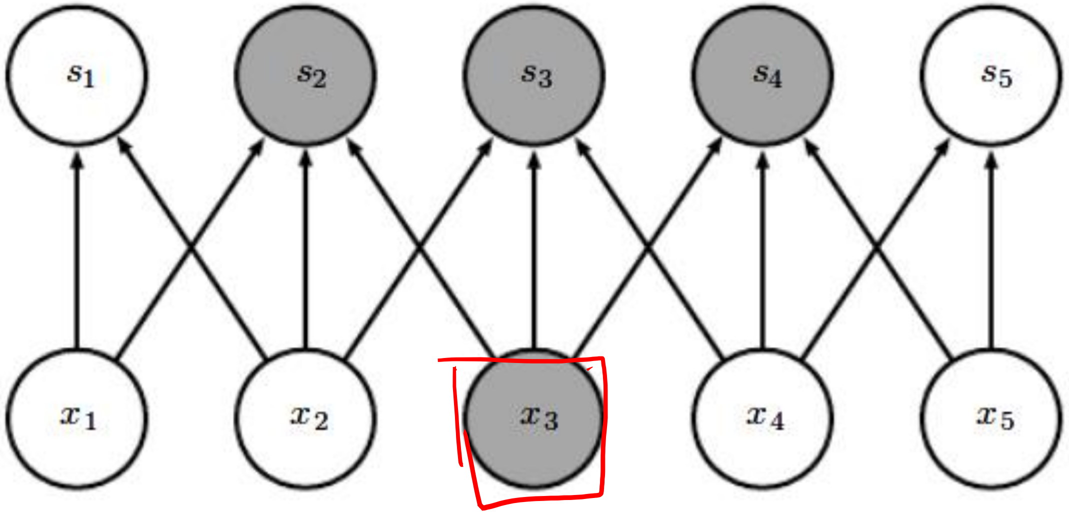


CNNs Feature Sparse Connectivity

Fully connected



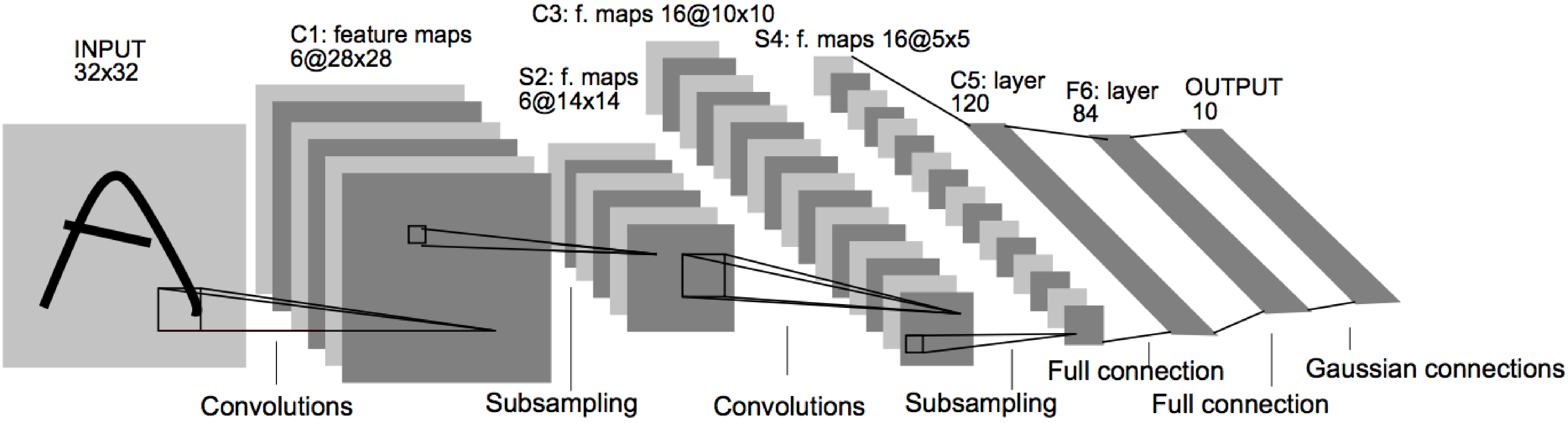
3x1 convolutional



Weight Sharing / Spatial Invariance

In a CNN, all the neurons in the same slice of a feature map use the same weights and bias: this reduces the nr. of parameters in the CNN.

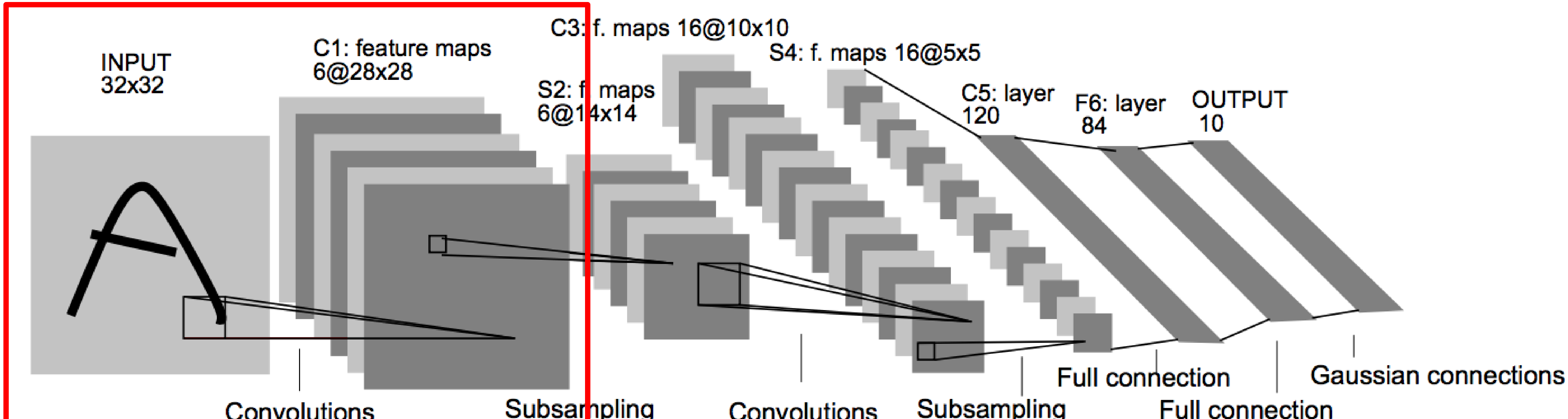
Underlying assumption: **if one feature is useful** to compute **at some spatial position** (x,y), **then it should also** be useful to compute at a **different position** (x2,y2)



Weight Sharing / Spatial Invariance

If the first layer were a MLP:

- it should have had 28 x 28 x 6 neurons in the output
- 5x5 connectivity each neuron
- It would be a 28 x 28 x 6 x 25 weights + 28 x 28 x 6 biases (122 304)
  CNN share the same weights among a few neurons of the same layer

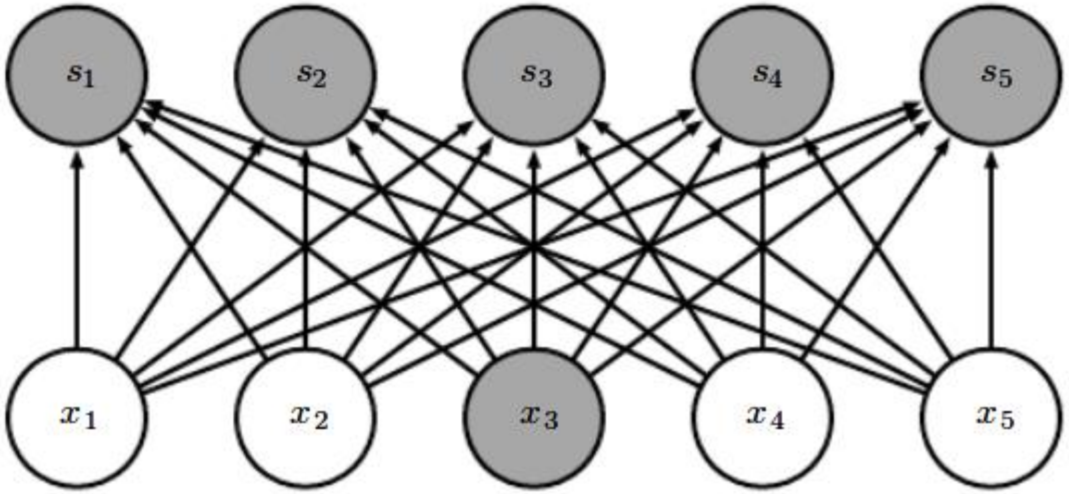


Parameter sharing

Fully connected

5x5 = 25 weights

(+ 5 bias)

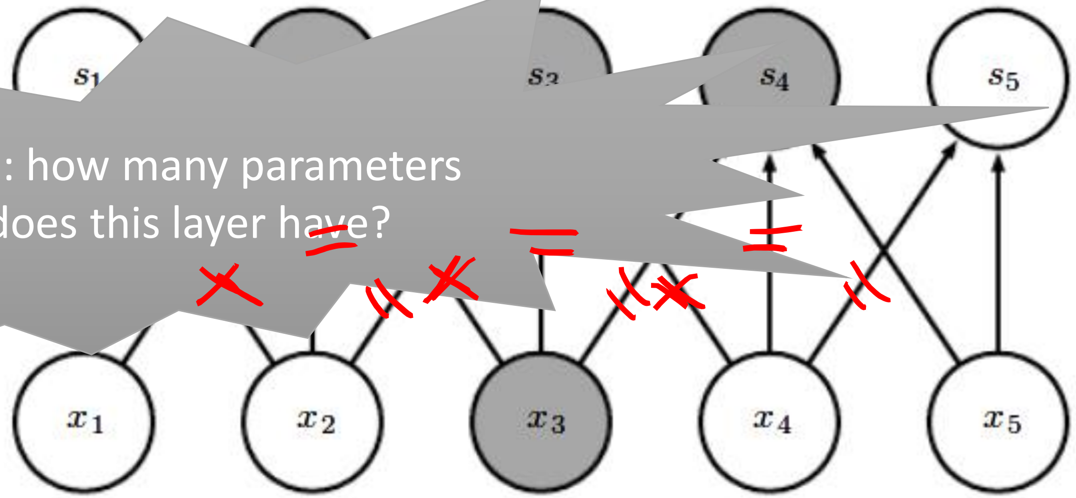


3x1 conv

Quiz: how many parameters does this layer have?

3 weights!

(+ 1 bias)



The Receptive Field
A very important aspect in CNNs

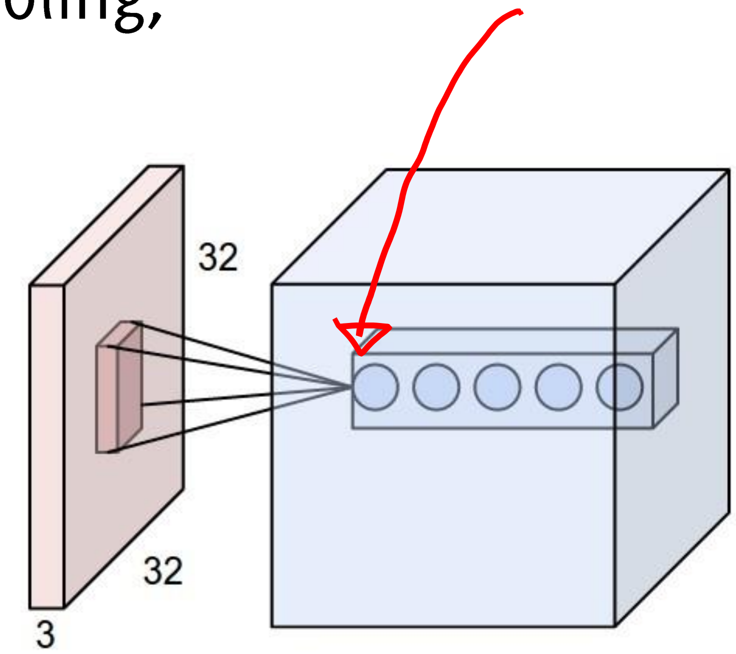
The Receptive Field

One of the basic concepts in deep CNNs.

Unlike in FC layers, where the value of each output depends on the entire input, **in CNN an output only depends on a region of the input. This region in the input is the receptive field for that output**

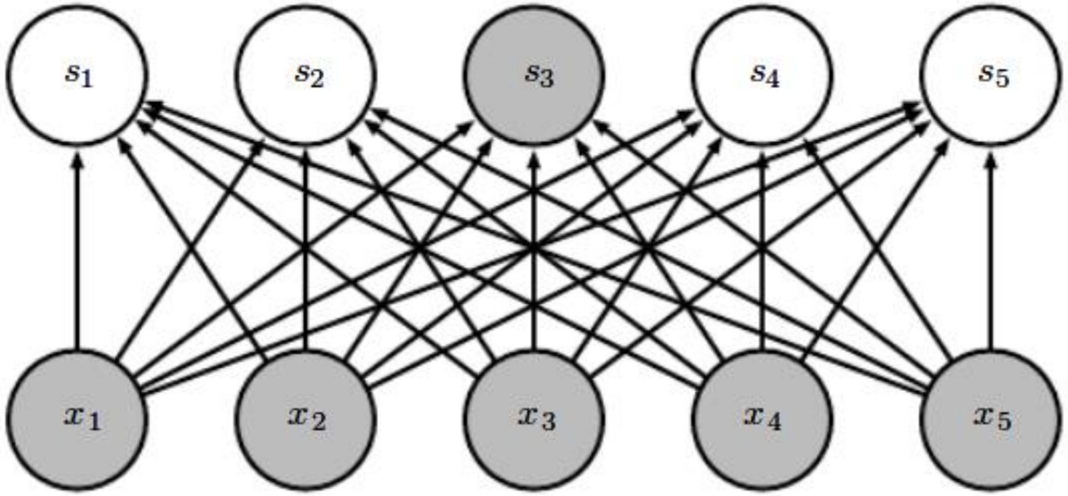
The deeper you go, the wider the receptive field: maxpooling, convolutions and stride > 1 increase the receptive field

Usually, the receptive field refers to the final **output unit** of the network in relation to the network input, but the same definition holds for intermediate volumes



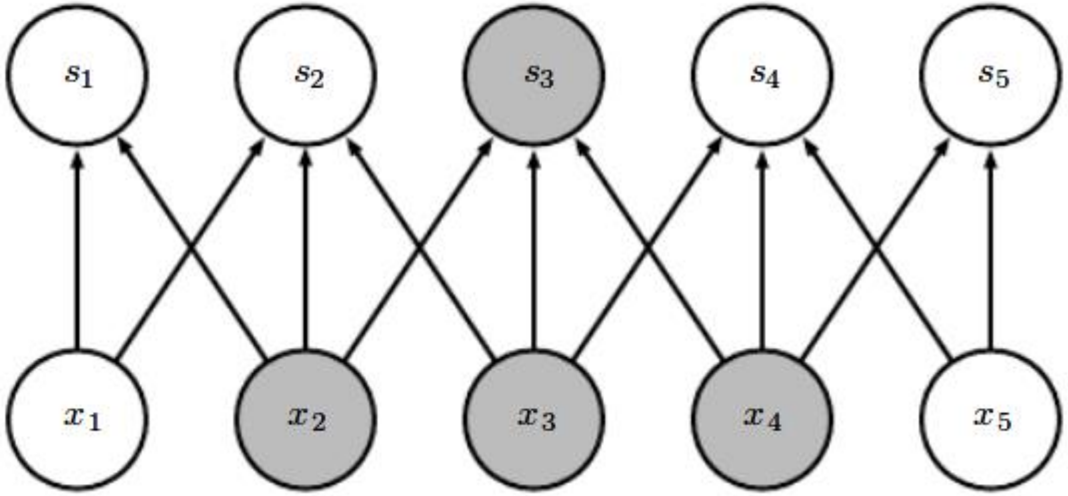
Receptive fields

Fully connected



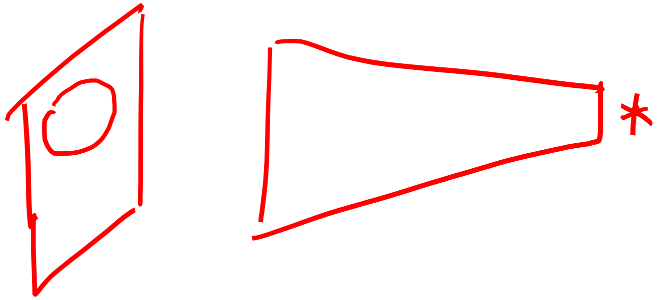
5 inputs

3x1 convolutional



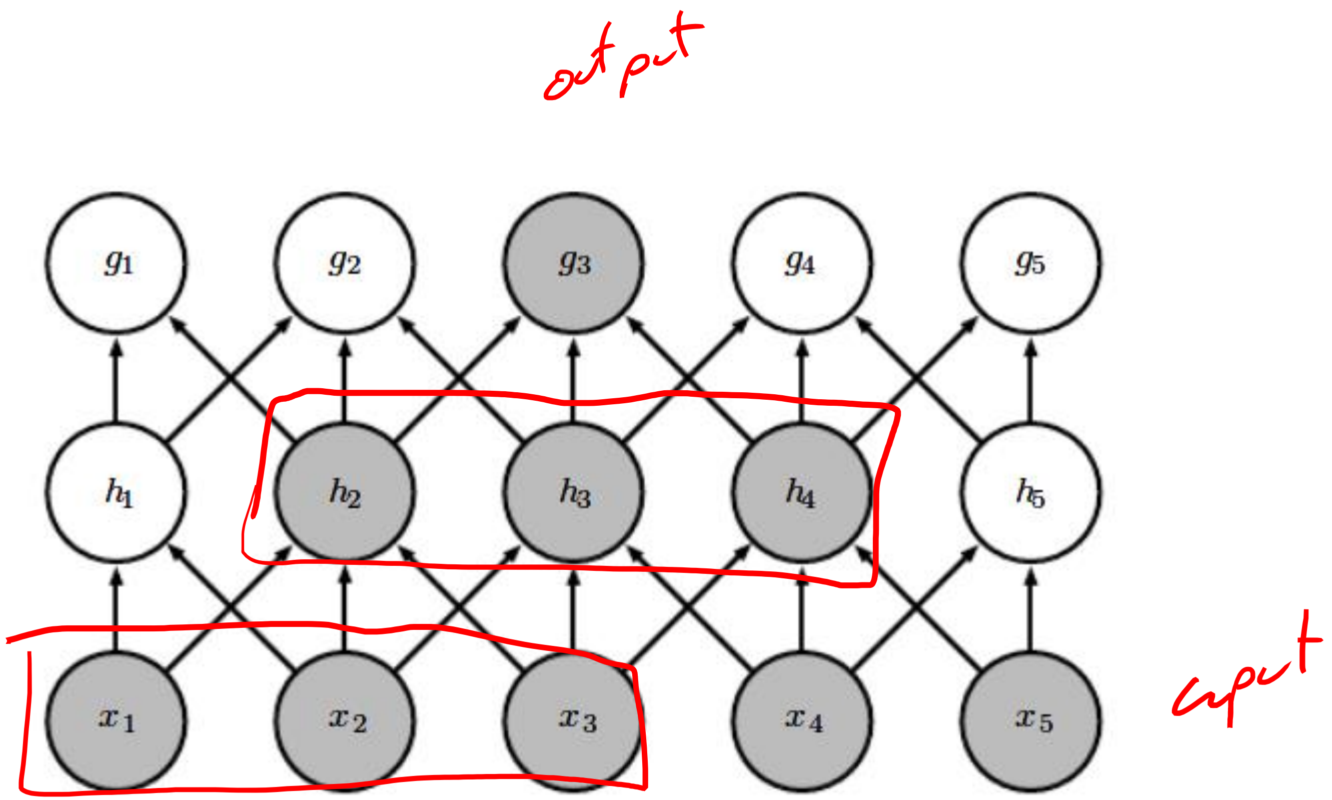
Receptive fields

Deeper neurons depend on wider patches of the input (convolution is enough to increase receptive field, no need of maxpooling)

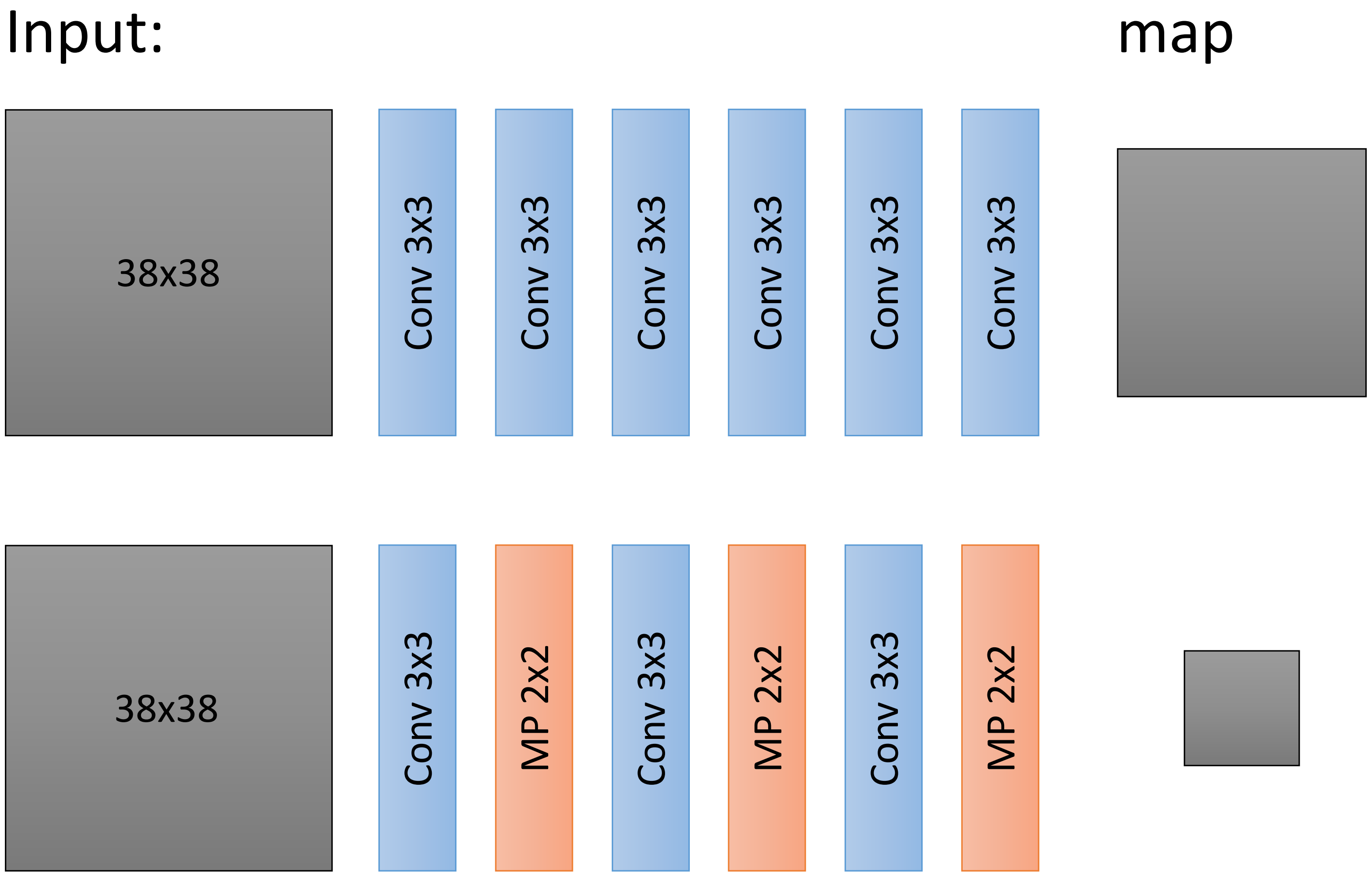


3x1 convolutional

3x1 convolutional

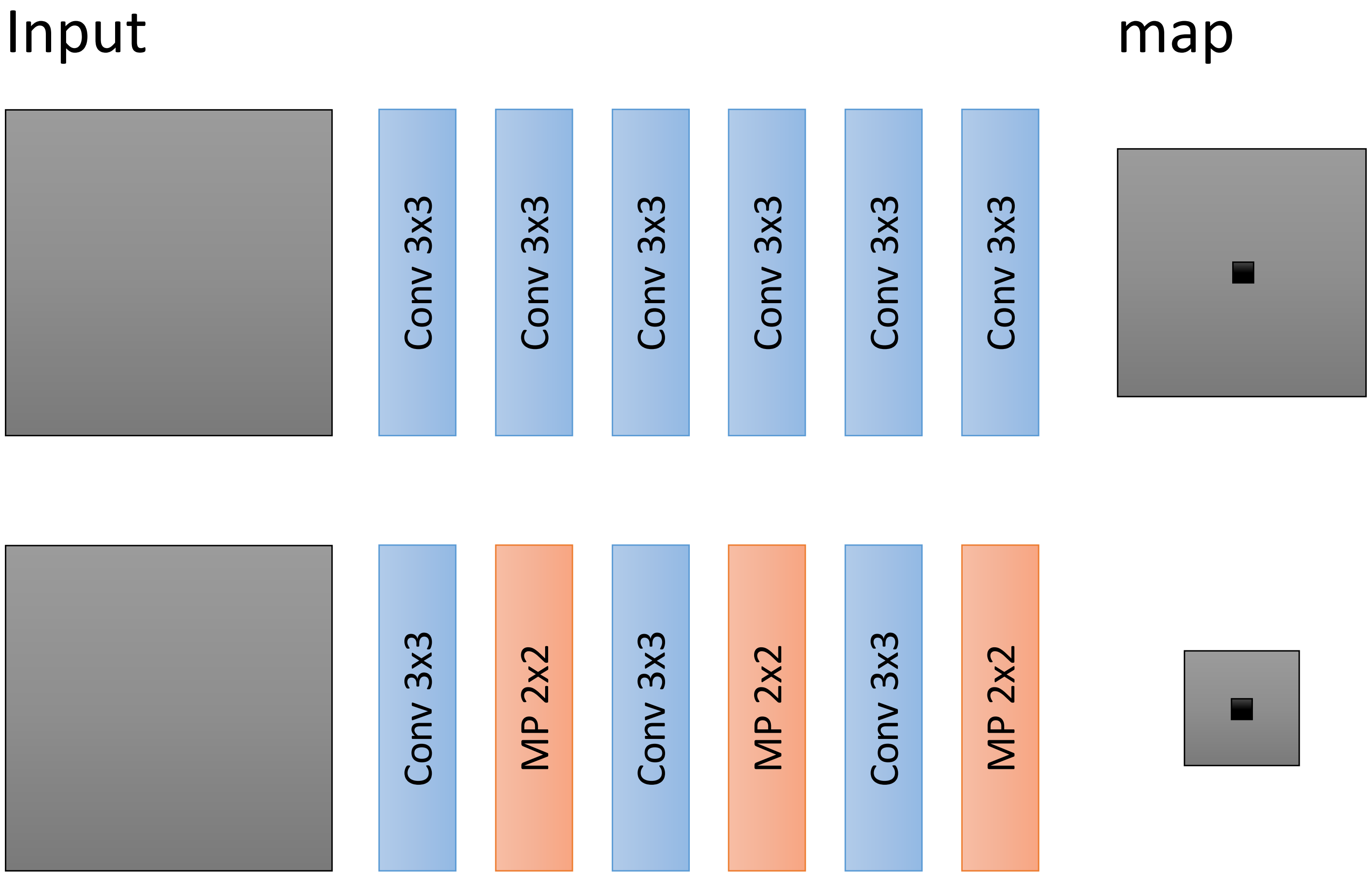


Exercise

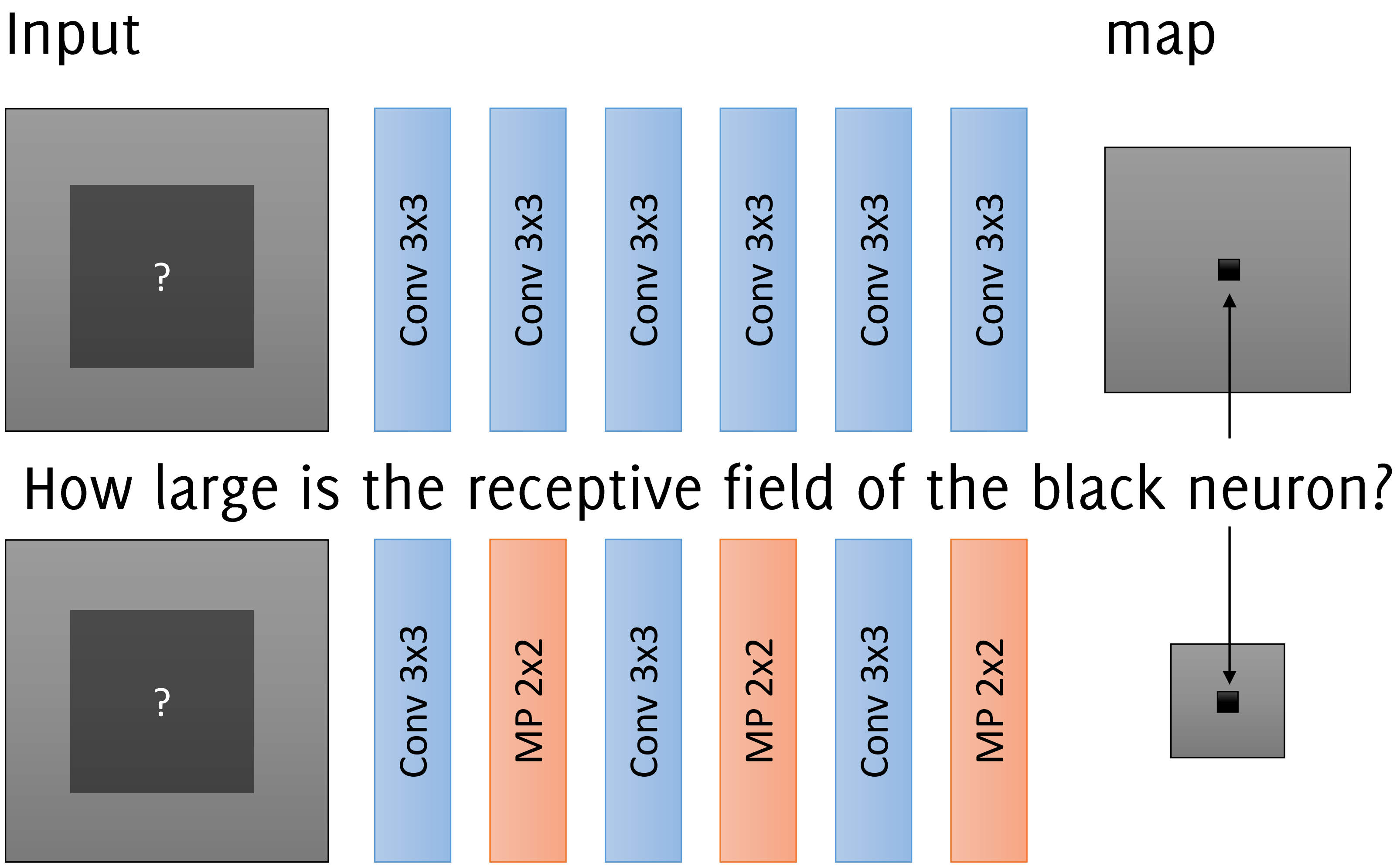


Input:
map
38x38
Conv 3x3
Conv 3x3
Conv 3x3
Conv 3x3
Conv 3x3
Conv 3x3
38x38
Conv 3x3
MP 2x2
Conv 3x3
MP 2x2
Conv 3x3
MP 2x2

Receptive fields



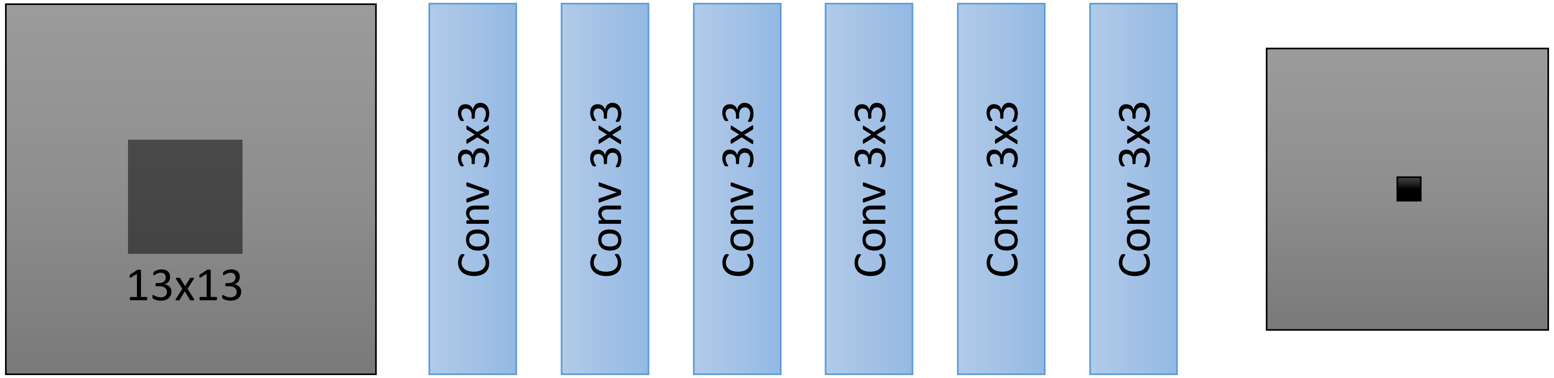
Receptive fields



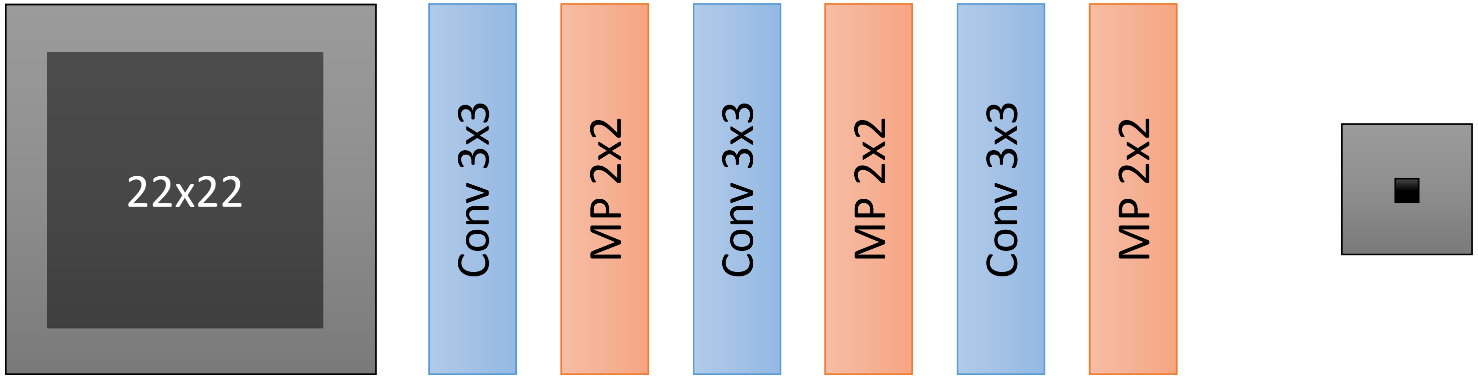
Receptive fields

Input

map



How large is the receptive field of the black neuron?



Maxpooling

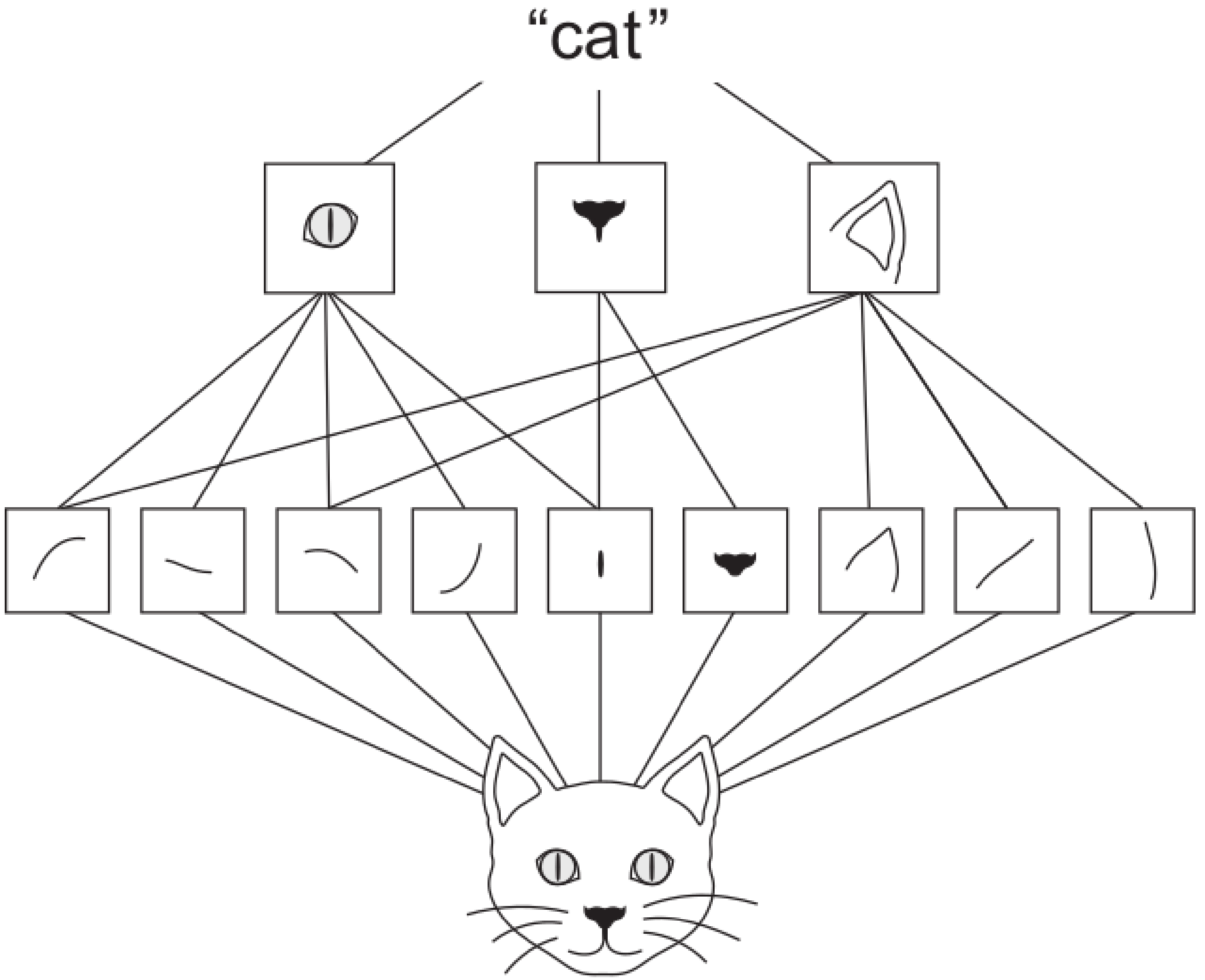
As we move deeper…

As we move to deeper layers:

- spatial resolution is reduced
- the number of maps increases

We search for **higher-level patterns**, and **don’t care too much about their exact location**.

There are more high-level patterns than low-level details!



Deep Learning

Giacomo Boracchi

CVPR USI, May 15 2020

giacomo.boracchi@polimi.it

https://boracchi.faculty.polimi.it/

Discussion on Grading Criteria

I understand

- Timing issues due to the exam session
- Large workload due to project/assignments

Constraints

Everything should be submitted by the end of the lectures (May 29)

- No point/time to open a new homework assignment

I'd like you to develop a (even a small) project on your own

- You need to practice CVPR "out of the beaten roads"
- I want to grade your initiative and expertise

Oral exam cannot just be limited to a project discussion

- You need to master the whole class material to get 10/10

Previous Grading Criteria

Assignments [45 points + 10 bonus]

- First assignment on Single and Multi-View Geometry [25 points + 5 bonus]
- Second assignment on Template matching [10 points + 5 bonus]
- Third assignment on image denoising [10 points]
- You will be given 3 weeks for solving each.

Project [30 points]

- You will choose a project on multiple template detection (multiple instances, multiple templates) [30 points]
- You will give a short presentation of your project during the last week of lecture
- A few days before the exam, you will have to submit a short presentation (including all the results) and a two page abstract (template will be provided and references and figures are not included in the page limit) to describe your project.
- Project will be briefly discussed during the exam day.

Oral exam [25 points]

- A short interview where you have to prove proficiency either in the theory and in the practical (programming) aspects of the course.

Grading Criteria

Assignments [35 points + 10 bonus]

- First assignment on Single and Multi-View Geometry [25 points + 5 bonus]
- Second assignment on Template matching [10 points + 5 bonus]

You will be given 3 weeks for solving each.

Small Project [20 points] 25

- You define your team project (1 – 2 persons). Project concerns multiple template detection (multiple instances, multiple templates)
- You will give a short presentation of your project on **Friday May 29**

60

Oral exam [45 points] 40

40

- An interview where you have to prove proficiency either in the theory and in the practical (programming) aspects of the course.

Project Assessment Criteria

20 points (10 bonus)

- Identifying another scenario for template matching to work
- Expanding the solution of homework 2 to work in multiple template settings

90/100

- Identifying additional challenges / margin for improvement
- Identifying a sound solution based on the course materials
- Clearly describe this solution in your presentation and your report
- Provide a proof of concept. The more sound, the better!
- Understand your experimental outcomes and provide a convincing discussion

Project Timeline

1. Register your team for the project (Monday May 18)
    - We need to know how many champions we have ☺
2. [optional] book a slot for a 3 minutes pitch to gather feedback during Tuesday May 19 or Friday May 22.
3. Prepare your 8' presentation and give it on May 29
4. Prepare your
    - one page abstract
    - Your code

    by May 29 and (extensions till May 31° will be granteed upon request)

~~longer~~ lecture

We will provide templates for pitch / presentation and for the 1 abstract page. Take for instance papers at this workshop

Outline

- Deep Learning when data are scarce
- Fully convolutional CNN
- CNN for semantic segmentation
- CNN for Localization
- CNN for Object Detection

Data Scarcity

Training a CNN with Limited Aumont of Data

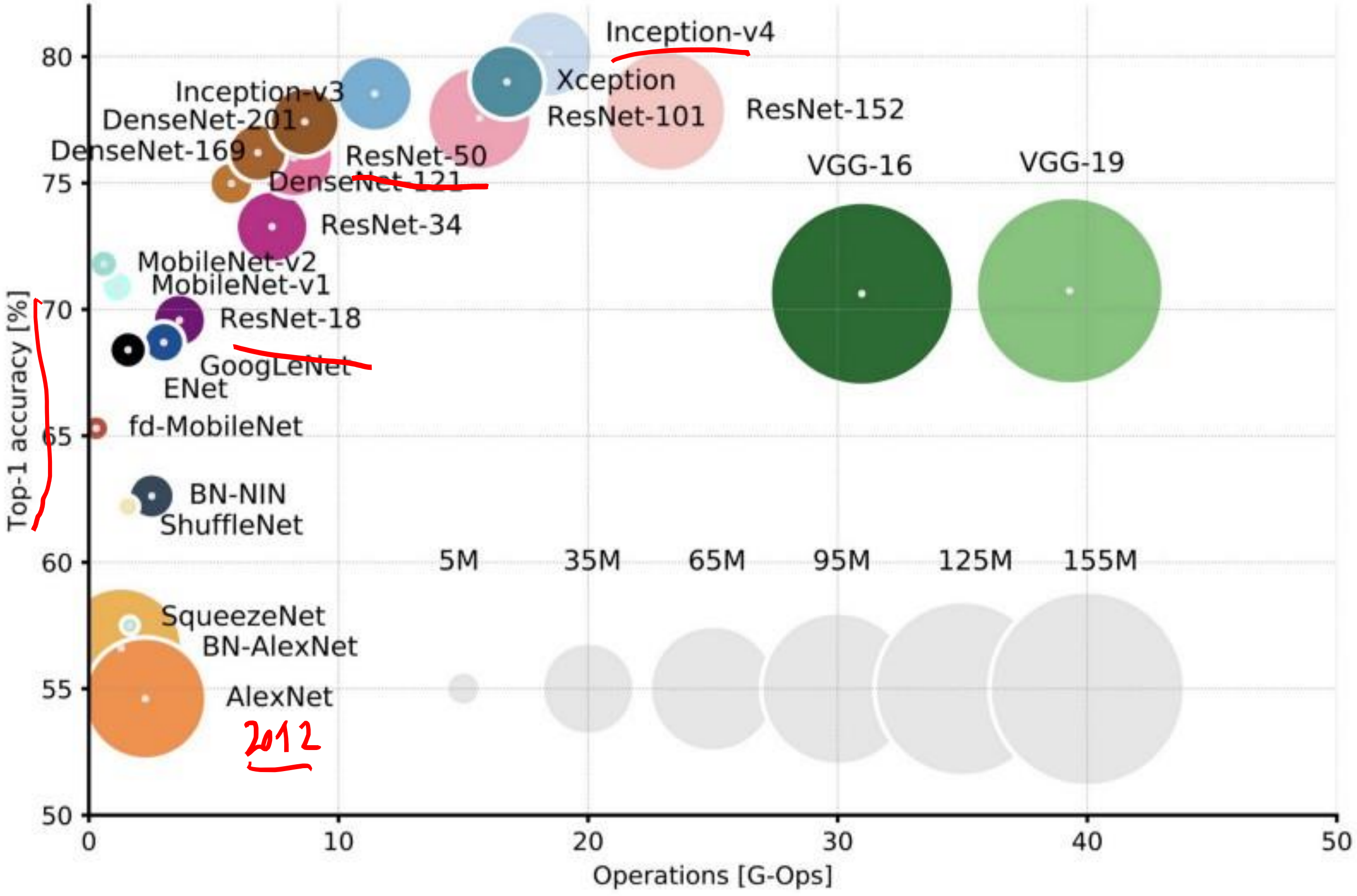
The need of data

Deep learning models are very data hungry.

Networks such as AlexNet have been trained on ImageNet datasets containing tens of thousands of images over hundreds of classes

The need of data

This is necessary to define millions of parameters characterizing these networks



Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." *arXiv preprint arXiv:1605.07678* (2016).

The need of data

Deep learning models are very data hungry.

… watch out: each image in the training set have to be annotated!

How to train a deep learning model with a few training images?

- Data augmentation
- Transfer Learning

Limited Amount of Data: Data Augmentation

Training a CNN with Limited Aumont of Data


Canada
Aleutian Islands
Stati Uniti
Messico
Steller sea lions in the western Aleutian Islands have declined 94% in the last 30 years.
3D

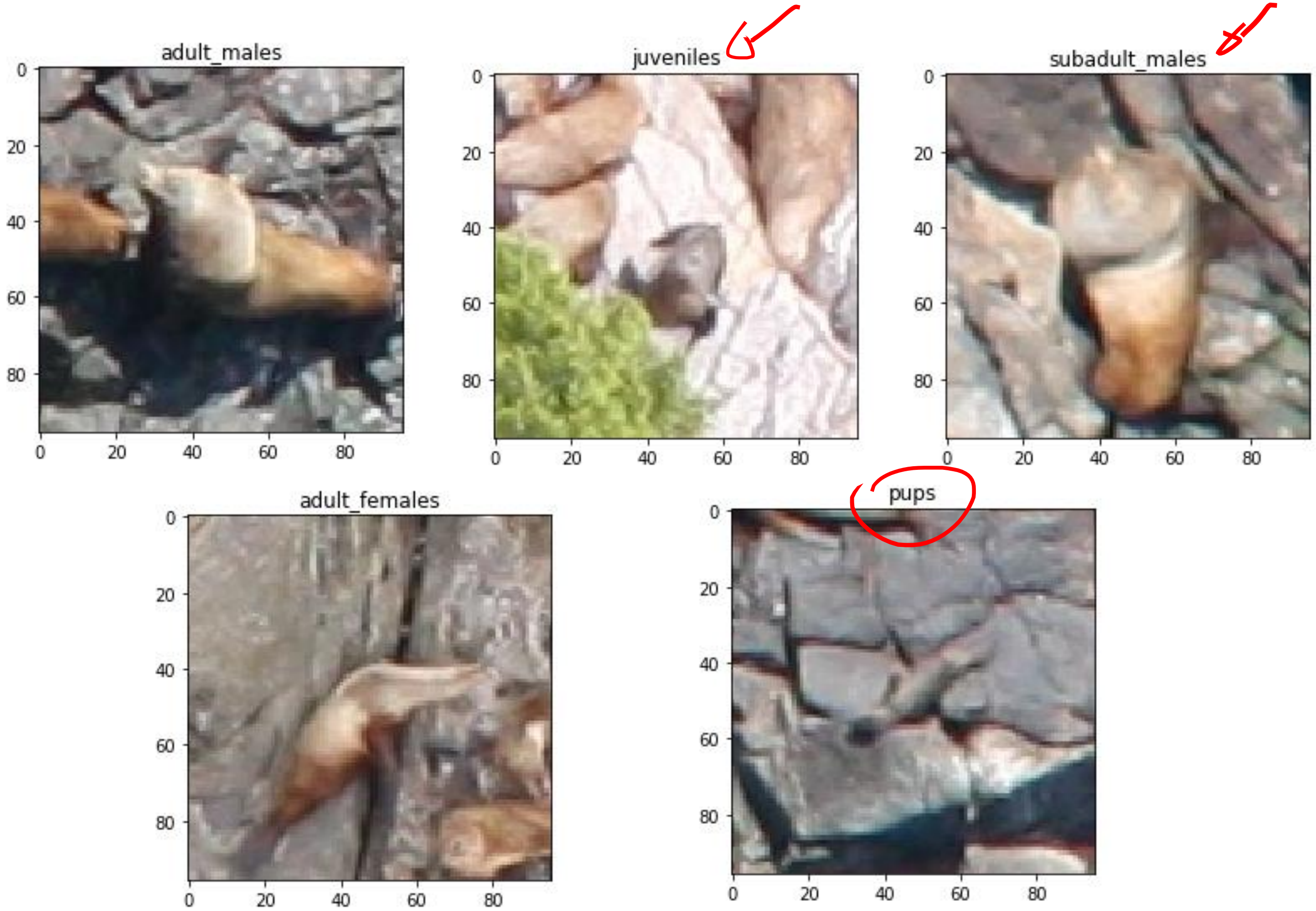
Kaggle in 2017 have opened a competition to develop algorithms which accurately count the number of sea lions in aerial photographs

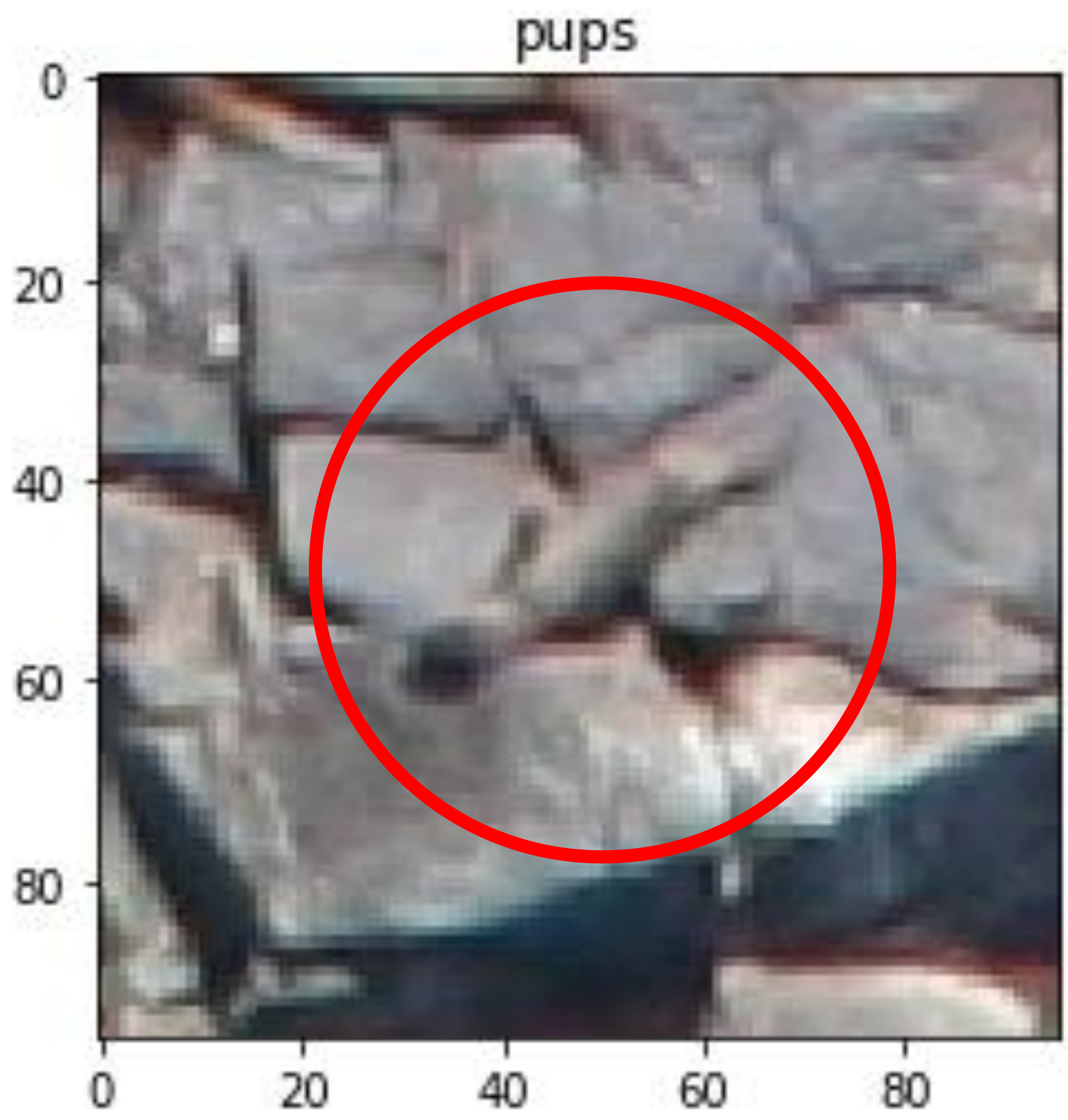
Credits Yinan Zhou

https://github.com/marioZYN/FC-CNN-Demo

The Challenge

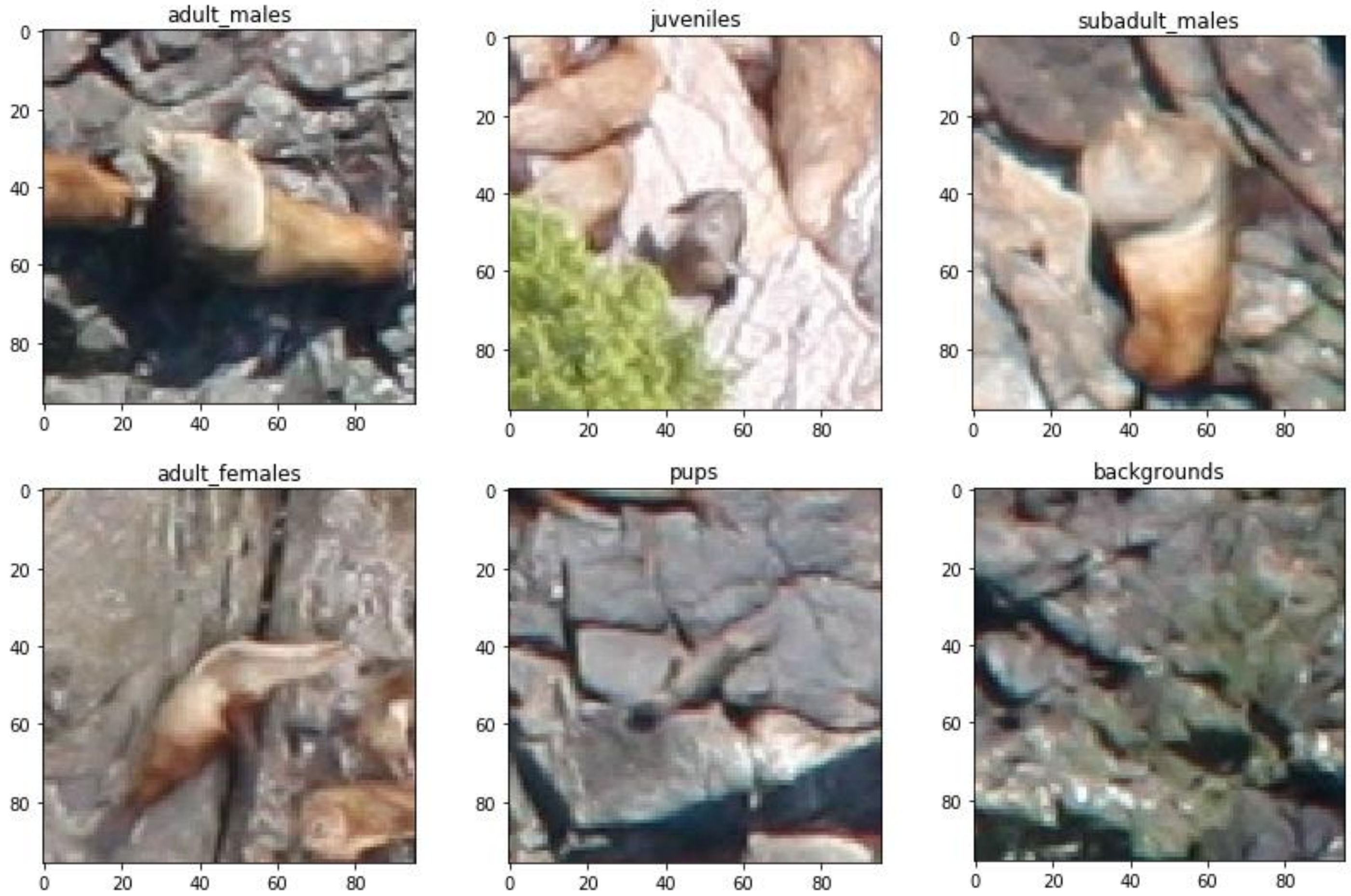
In very large aerial images (≈ 5K x 4K) shot by drones, automatically count the number of sealions per each category



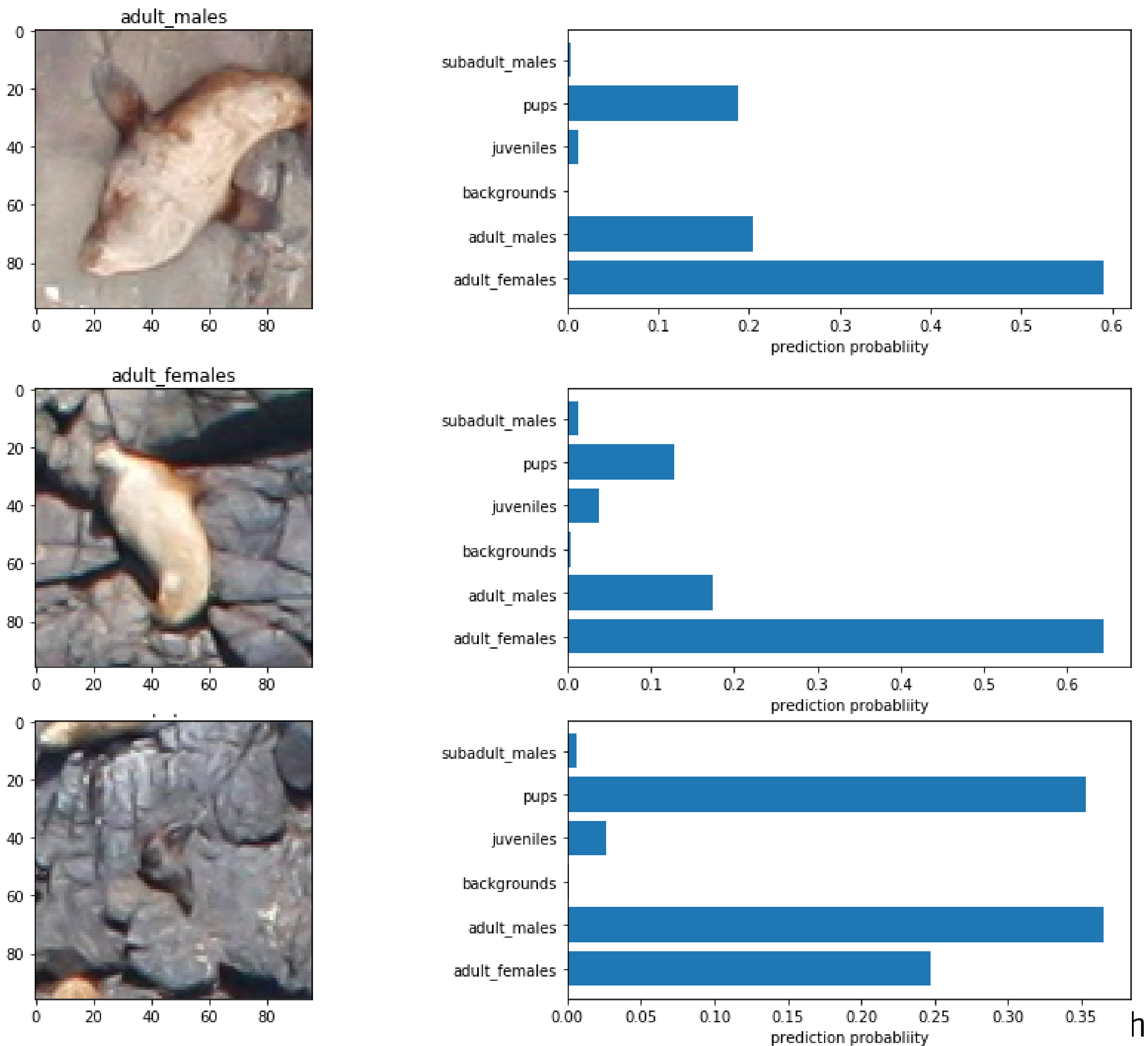

pups
0
20
40
60
80
0
20
40
60
80

The Challenge

This problem can be naively casted in a patch-by-patch 6-class classification problem, where we include also background



An Example of CNN predictions

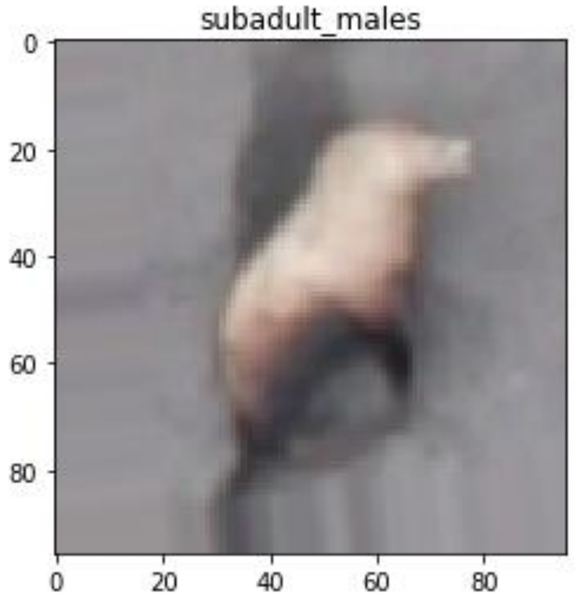


Credits Yinan Zhou
https://github.com/marioZYN/FC-CNN-Demo

Data Augmentation

Often, each annotated image represents a class of images that are all likely to belong to the same class

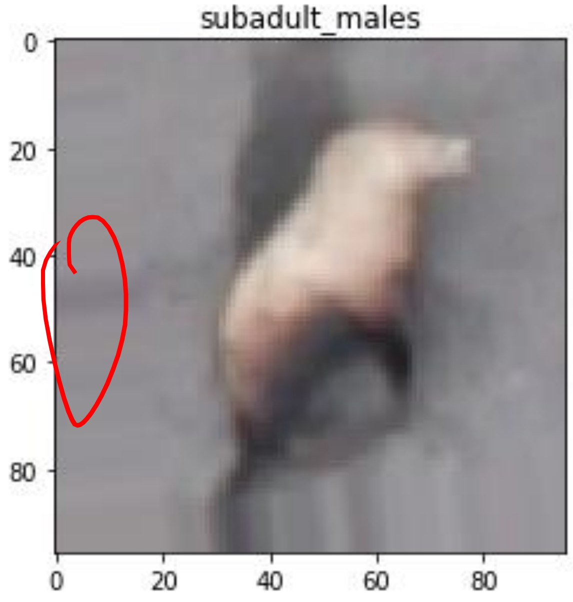
In aereal photograps, for instance, it is normal to have rotated, shifted or scaled images without changing the label

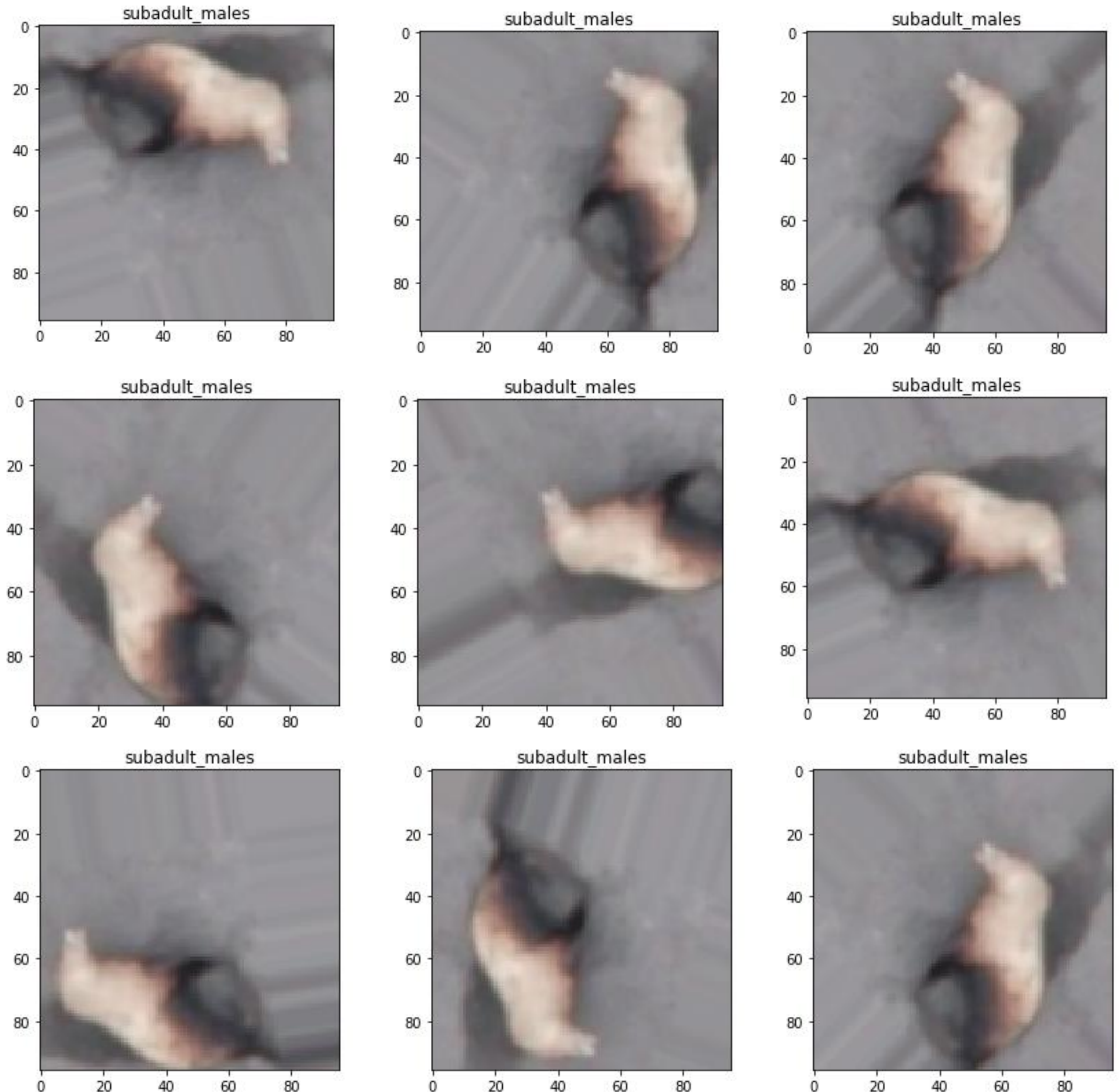


Data Augmentation

Augmented Images

Original image





Data Augmentation

Data augmentation is typically performed by means of

Geometric Transformations:

- Shifts /Rotation/Affine/perspective distortions
- Shear
- Scaling
- Flip

Photometric Transformations:

- Adding noise
- Modifying average intensity
- Superimposing other images
- Modifying image contrast

Augmentation in Keras

```
from keras.preprocessing.image import
ImageDataGenerator

ImageDataGenerator(
rotation_range=0,
width_shift_range=0.0, height_shift_range=0.0,
brightness_range=None, shear_range=0.0,
zoom_range=0.0, channel_shift_range=0.0,
fill_mode='nearest',
horizontal_flip=False, vertical_flip=False,
rescale=None,
preprocessing_function=None)
```

Augmentation in Keras: flow from images

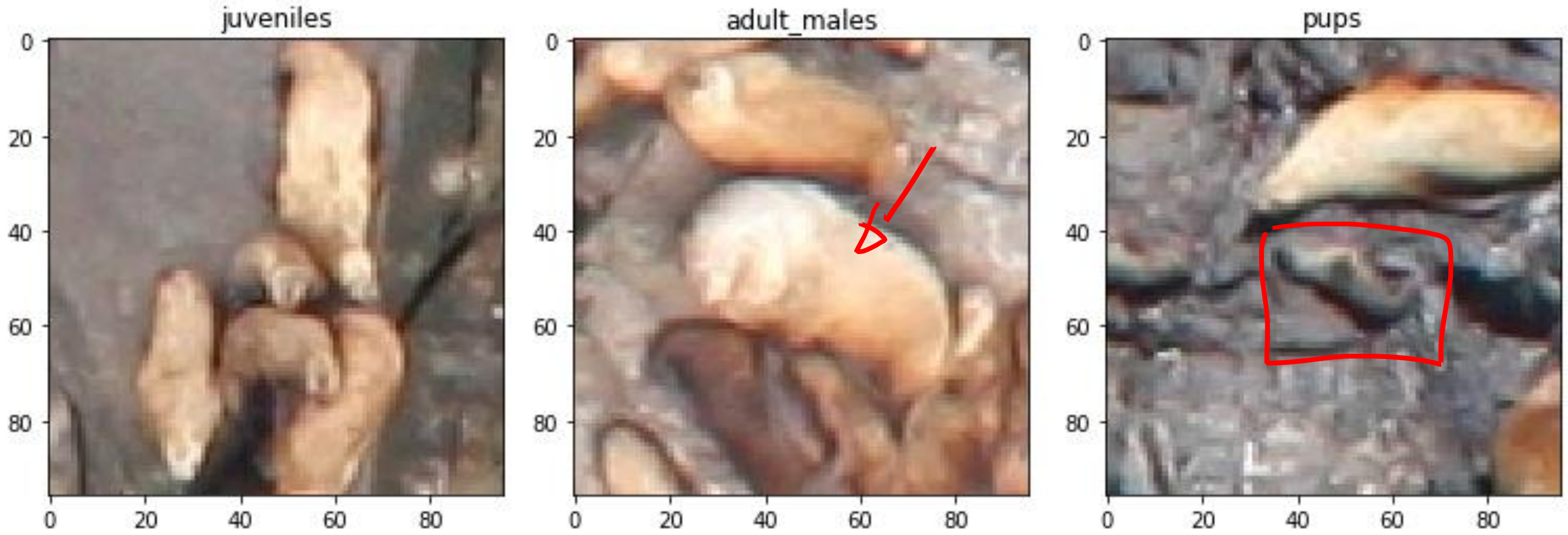
The Image generator has a method **`flow_from_directory`** that allows to load images in folder where different classes are arranged in subfolders.

```
ImageDataGenerator.flow_from_directory(
    directory=PATCH_PATH + 'train/',
    target_size=(img_width, img_width),
    batch_size=batch_size,
    shuffle=True)
```

However…

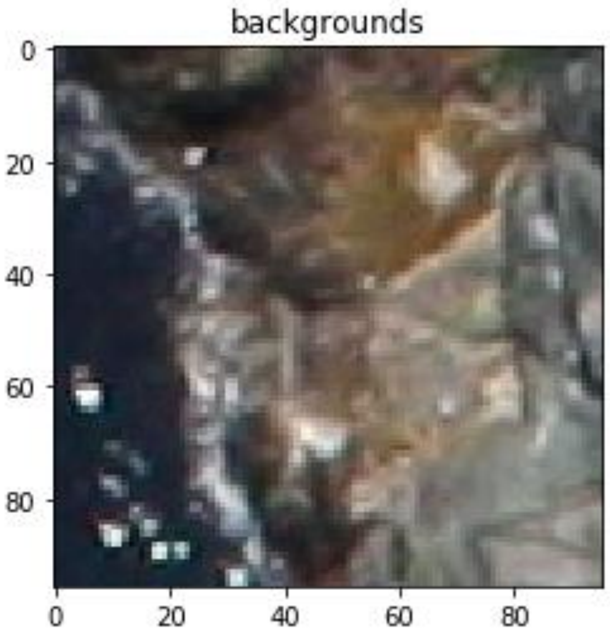
This sort of data augmentation might not be enough to capture the inter-class variability of images…

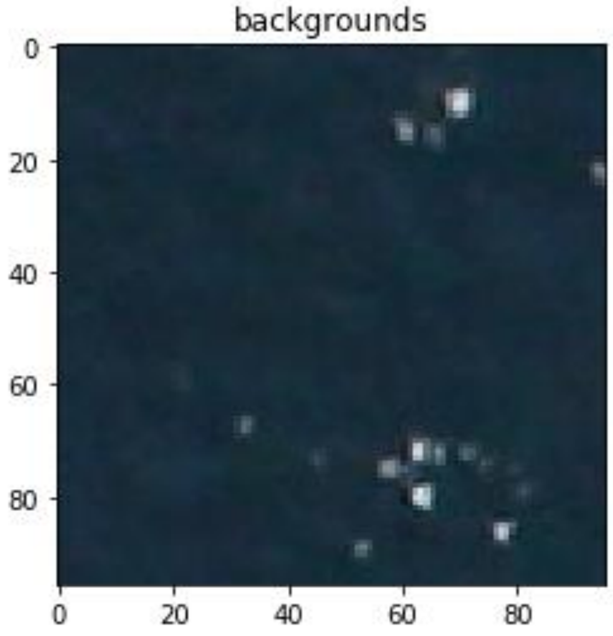
Superimposition of targets

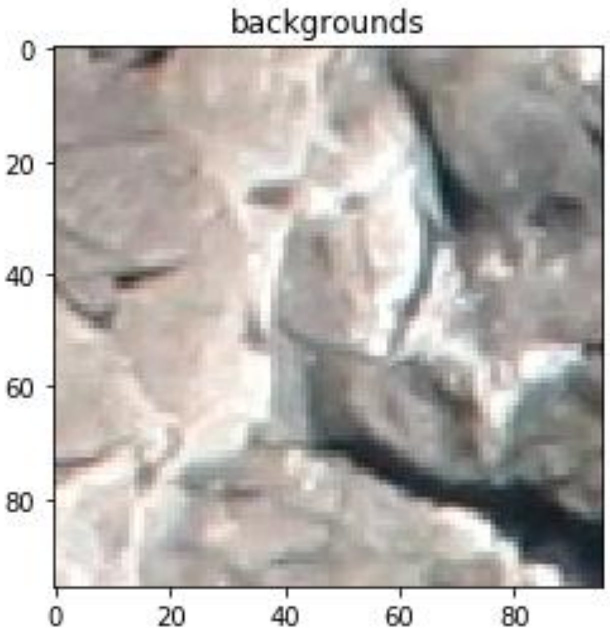


However…

Background variations:

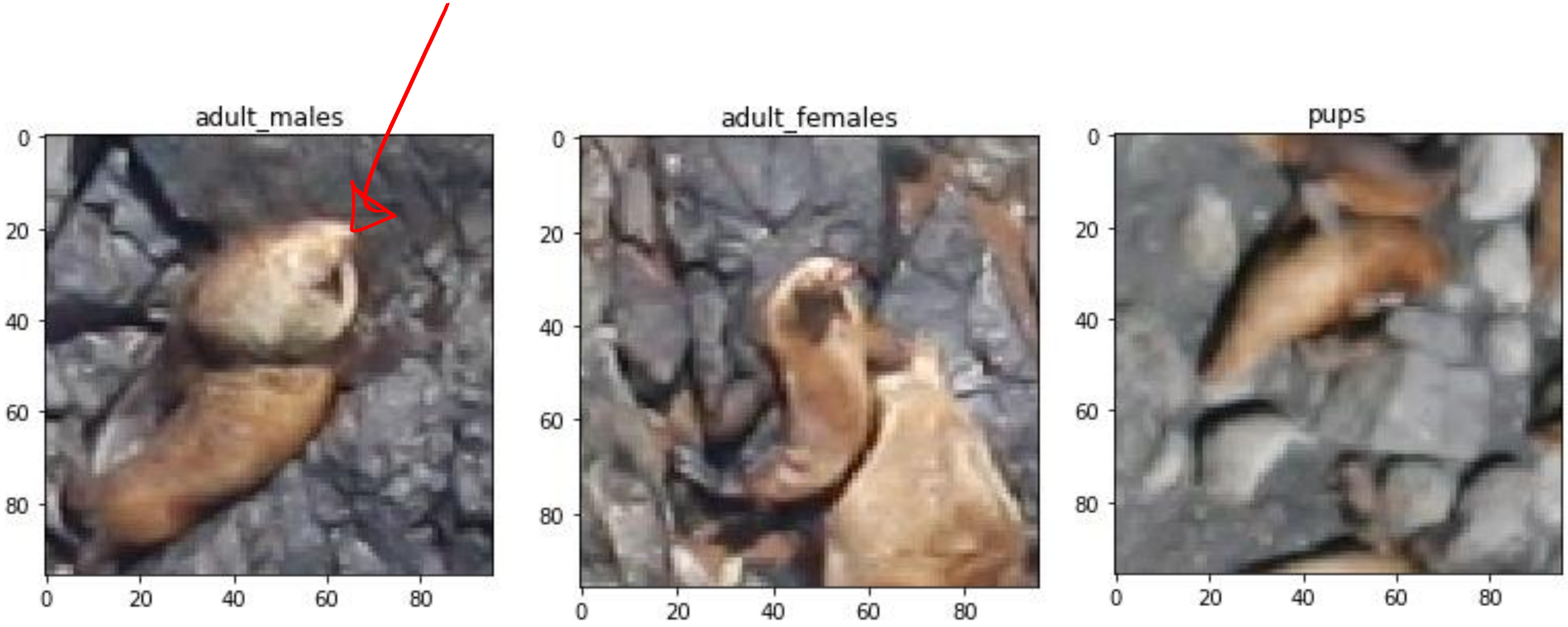






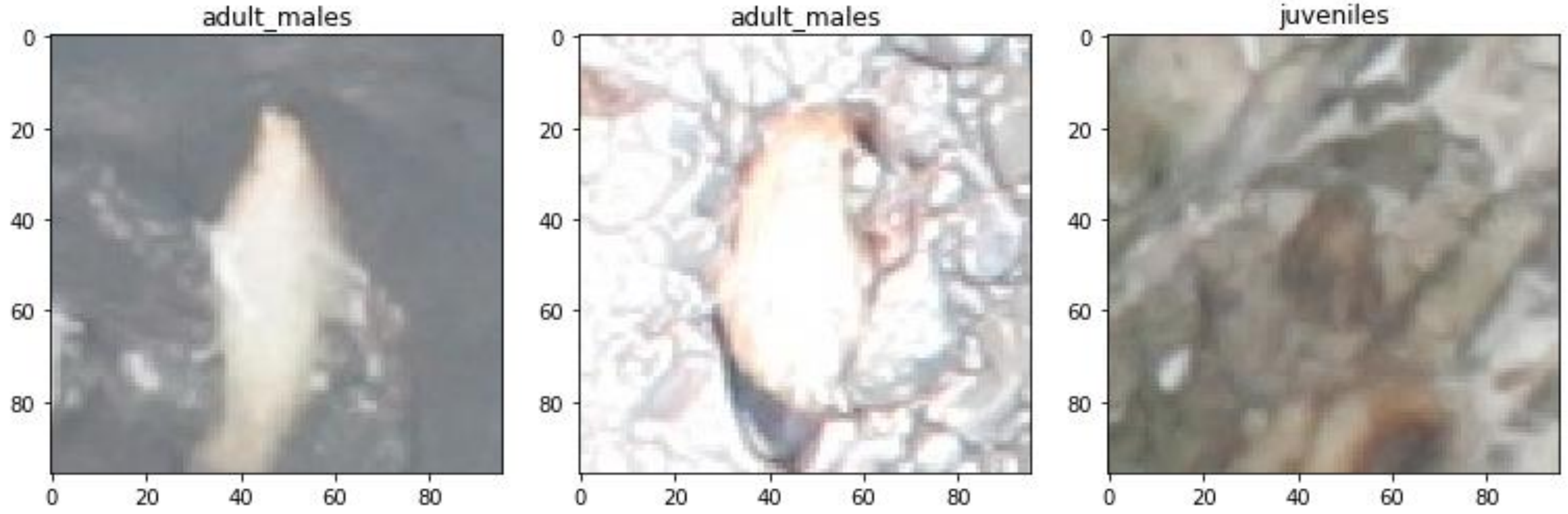
However…

Background variations



However…

Out of focus, bad exposure



Giacomo Boracchi

Augmentation in Keras

```
from keras.preprocessing.image import
ImageDataGenerator

ImageDataGenerator(
rotation_range=0,
width_shift_range=0.0, height_shift_range=0.0,
brightness_range=None, shear_range=0.0,
zoom_range=0.0, channel_shift_range=0.0,
fill_mode='nearest',
horizontal_flip=False, vertical_flip=False,
rescale=None,
preprocessing_function=None)
```

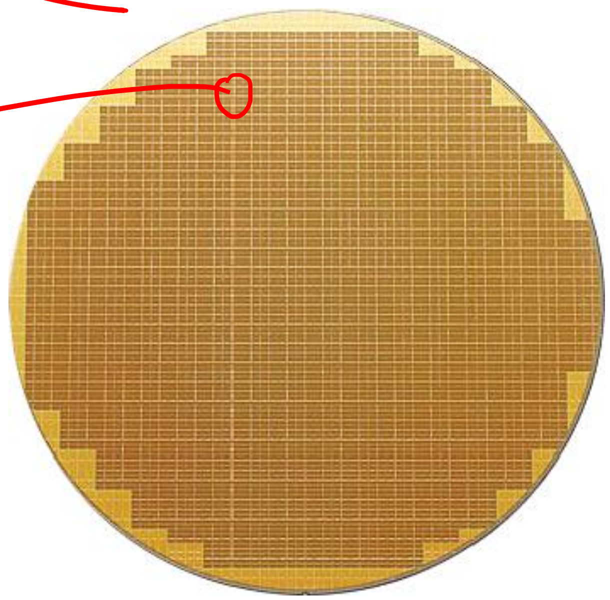
… in case you need some extra flexibility

Data Augmentation is often key..

Collaboration with

For instance to monitor wafer manufacturing

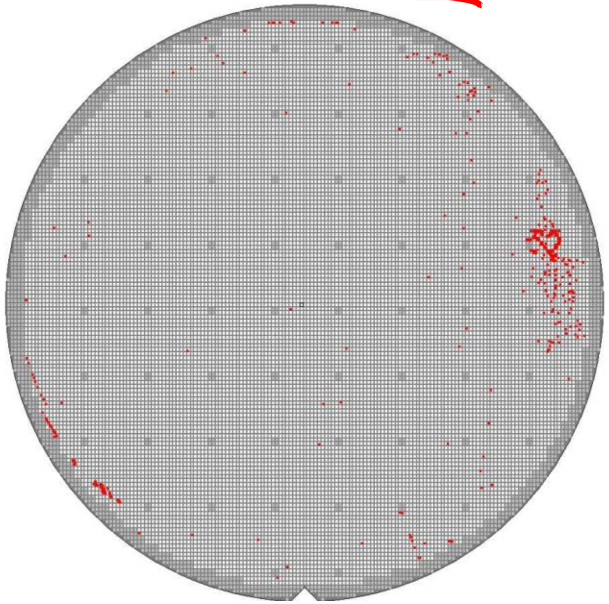
Wafer (containing chips)



Inspection tool

Wafer Defect Map

20K x 20K

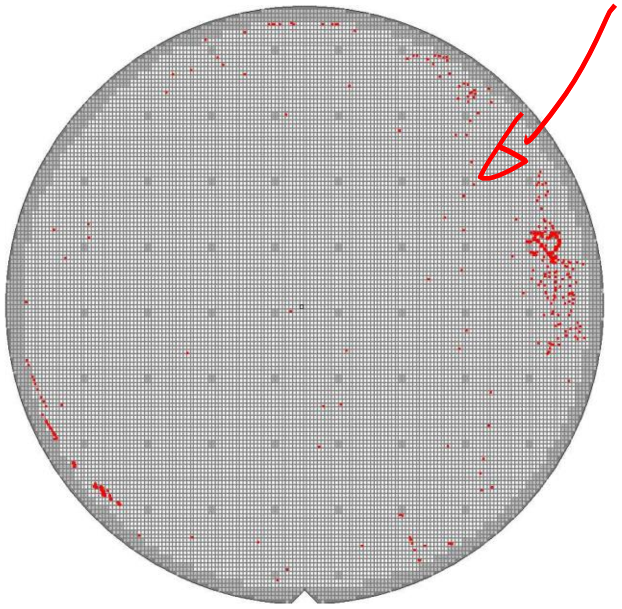


R. di Bella, D. Carrera, B. Rossi, P. Fragneto, G. Boracchi «Wafer Defect Map Classification Using Sparse Convolutional Networks" ICIAP 2019

Data Augmentation is often key..

Collaboration with

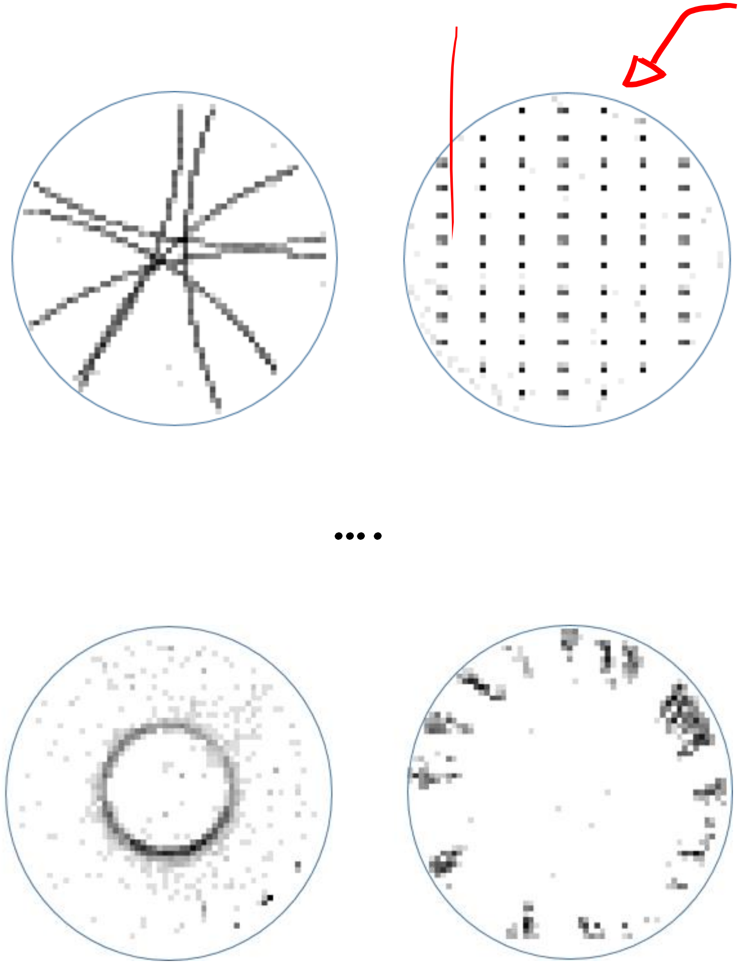
Train a deep learning model to identify defective patterns



Wafer Defect Map

Deep Learning

Defect Patterns

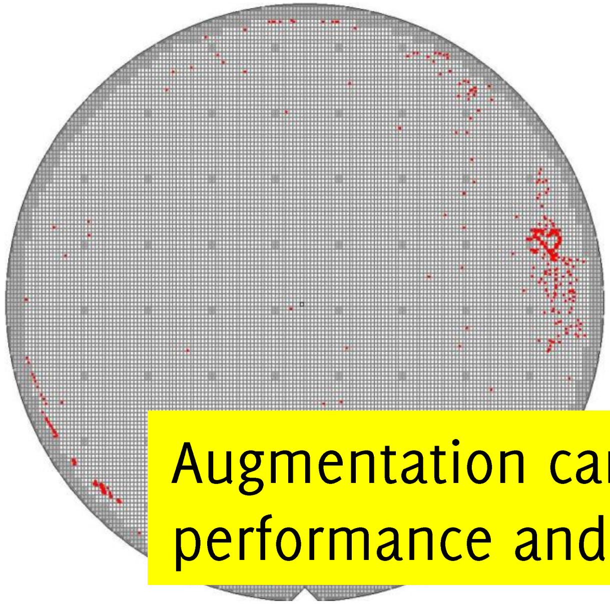


R. di Bella, D. Carrera, B. Rossi, P. Fragneto, G. Boracchi «Wafer Defect Map Classification Using Sparse Convolutional Networks” ICIAP 2019

Data Augmentation is often key..

Collaboration with

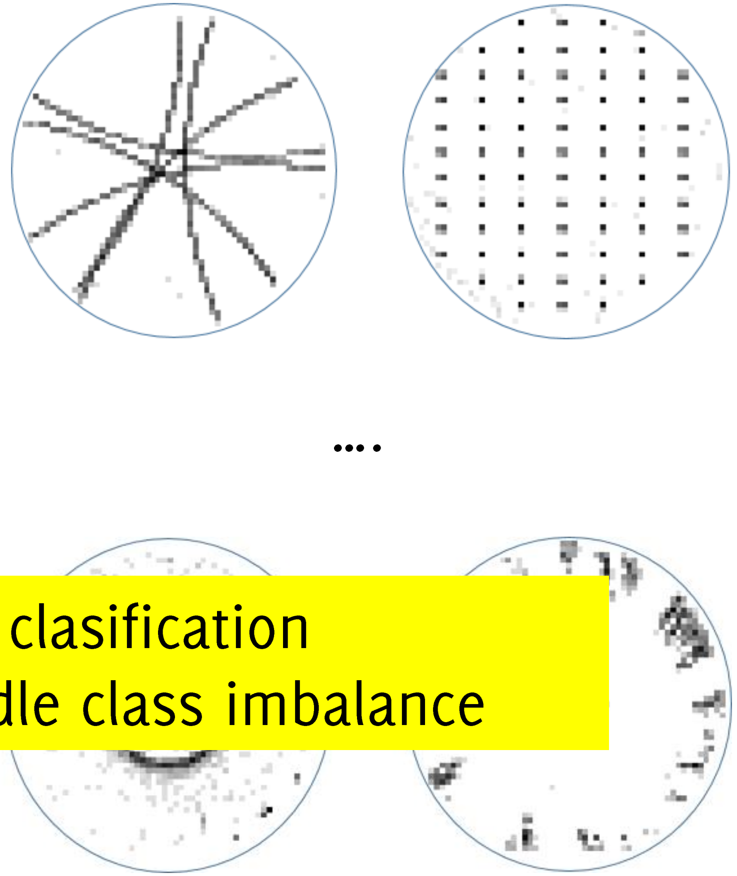
Train a deep learning model to identify defective patterns



Wafer Defect Map

Deep Learning

Defect Patterns

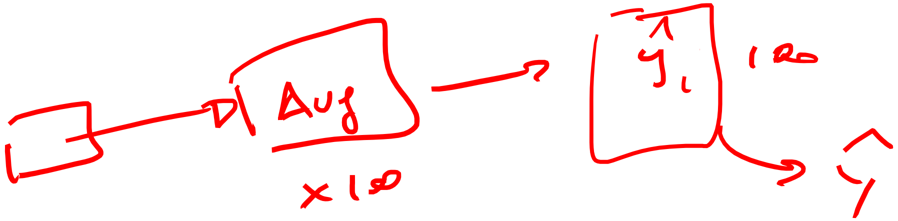


....

Augmentation can greatly improve clasification performance and successfully handle class imbalance

R. di Bella, D. Carrera, B. Rossi, P. Fragneto, G. Boracchi «Wafer Defect Map Classification Using Sparse Convolutional Networks” ICIAP 2019

Test Time Augmentation (TTA)



Even if the CNN is trained using augmentation, **it won't achieve perfect invariance** w.r.t. considered transformations

Test time augmentation (TTA): augmentation can be also performed at test time to improve prediction accuracy.

- Perform random augmentation of each test image I
- Average the predictions of each augmented image
- Take the average vector of posterior for defining the final guess

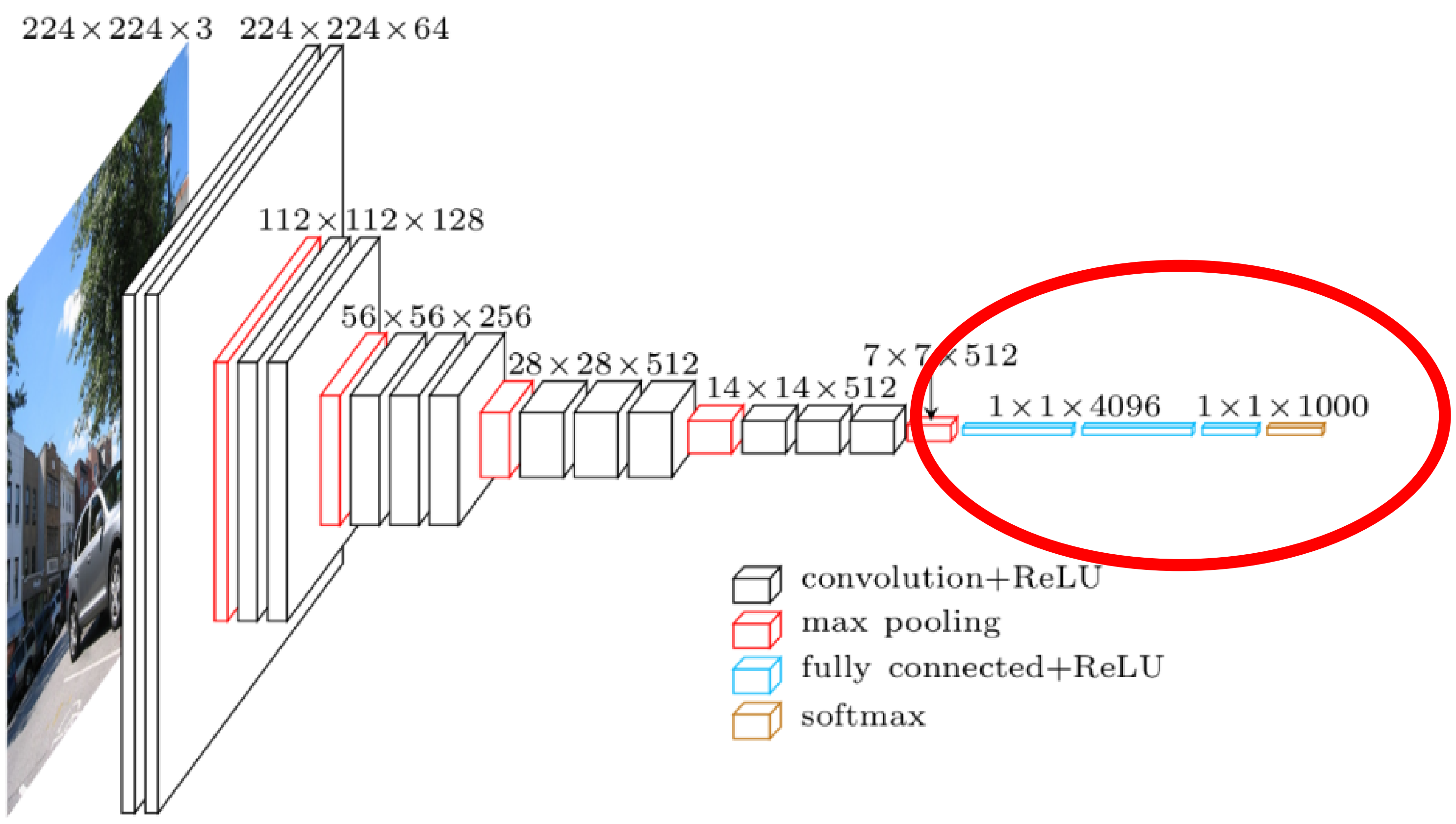
Test Time Augmentation is particularly useful for test images where the model is pretty unsure.

Limited Amount of Data: Transfer Learning

Training a CNN with Limited Aumont of Data

Transfer Learning

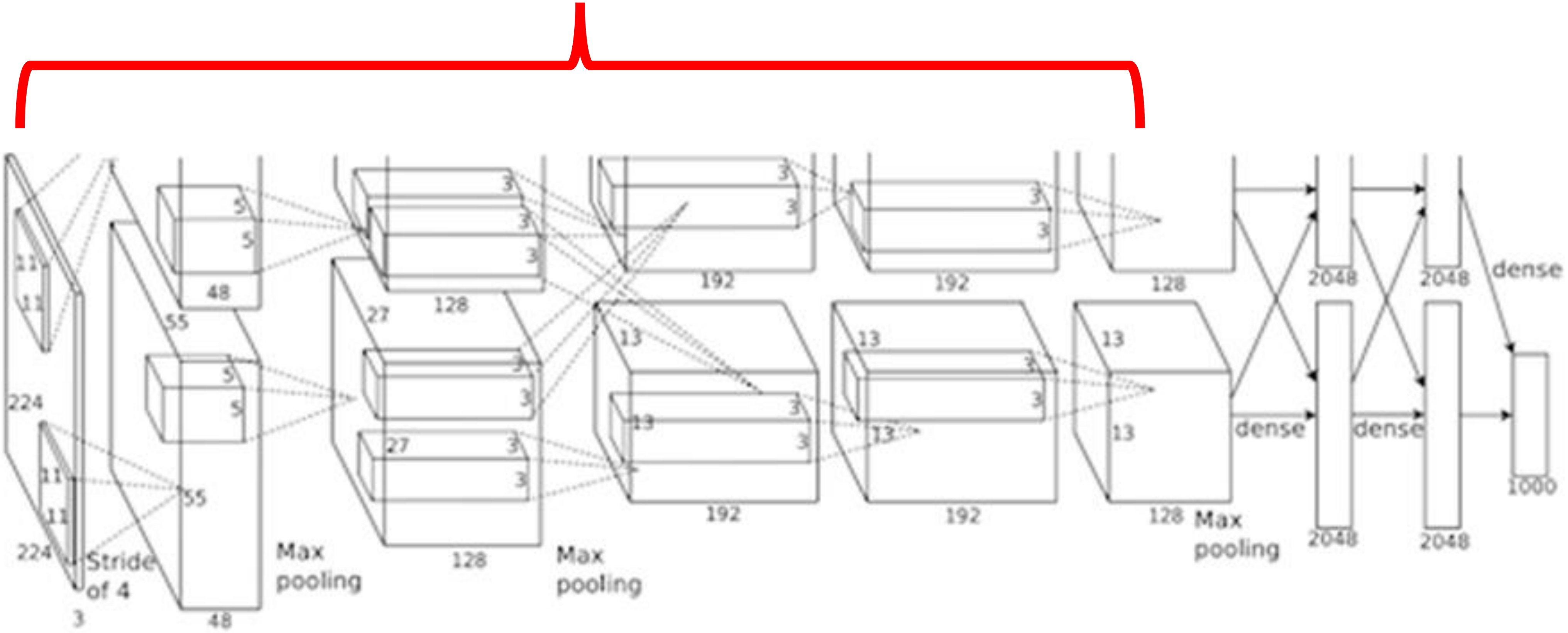
How to use pre-trained models to solve different problems



Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

Deep architectures

Deep Networks are great at extracting
(learned) features from images

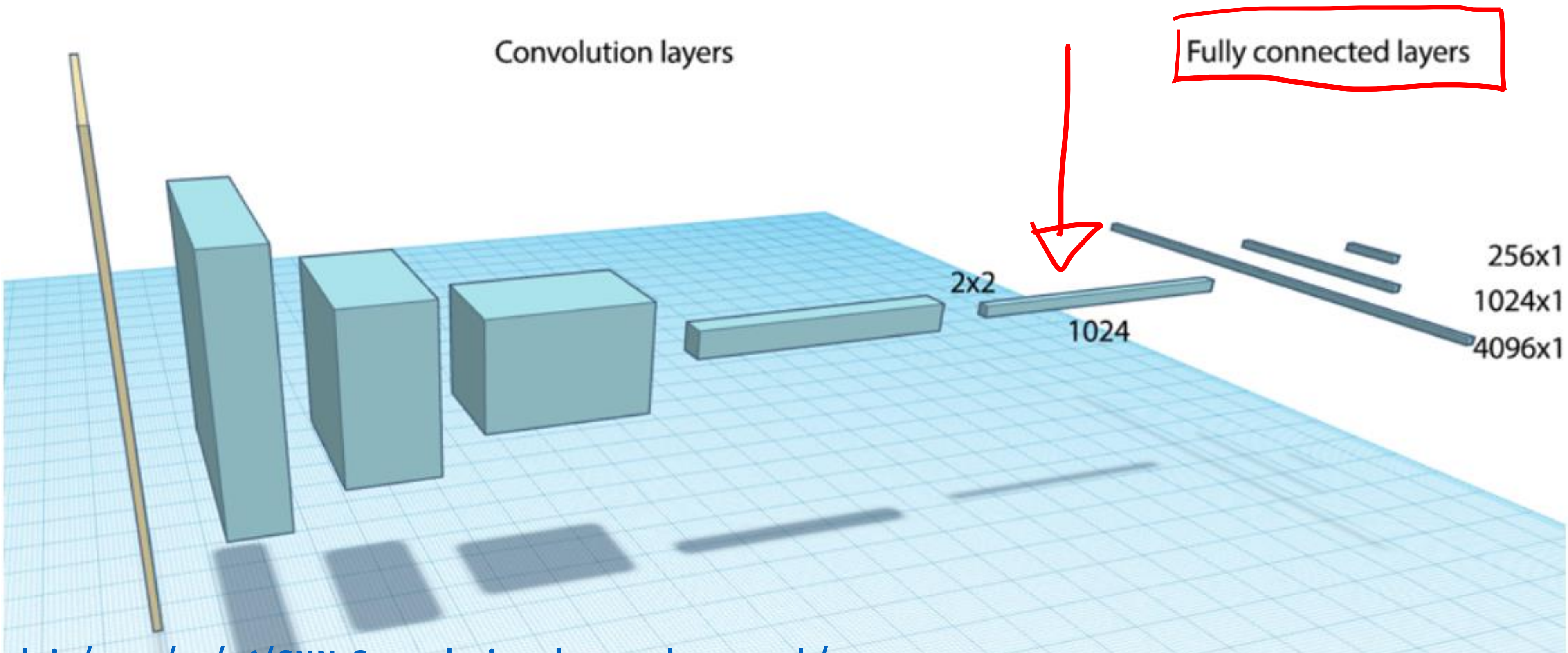


AlexNet architecture (May look weird because there are two different "streams". This is because the training process was so computationally expensive that they had to split the training onto 2 GPUs)

A typical architecture

Extract high-level features from pixel data

Classify

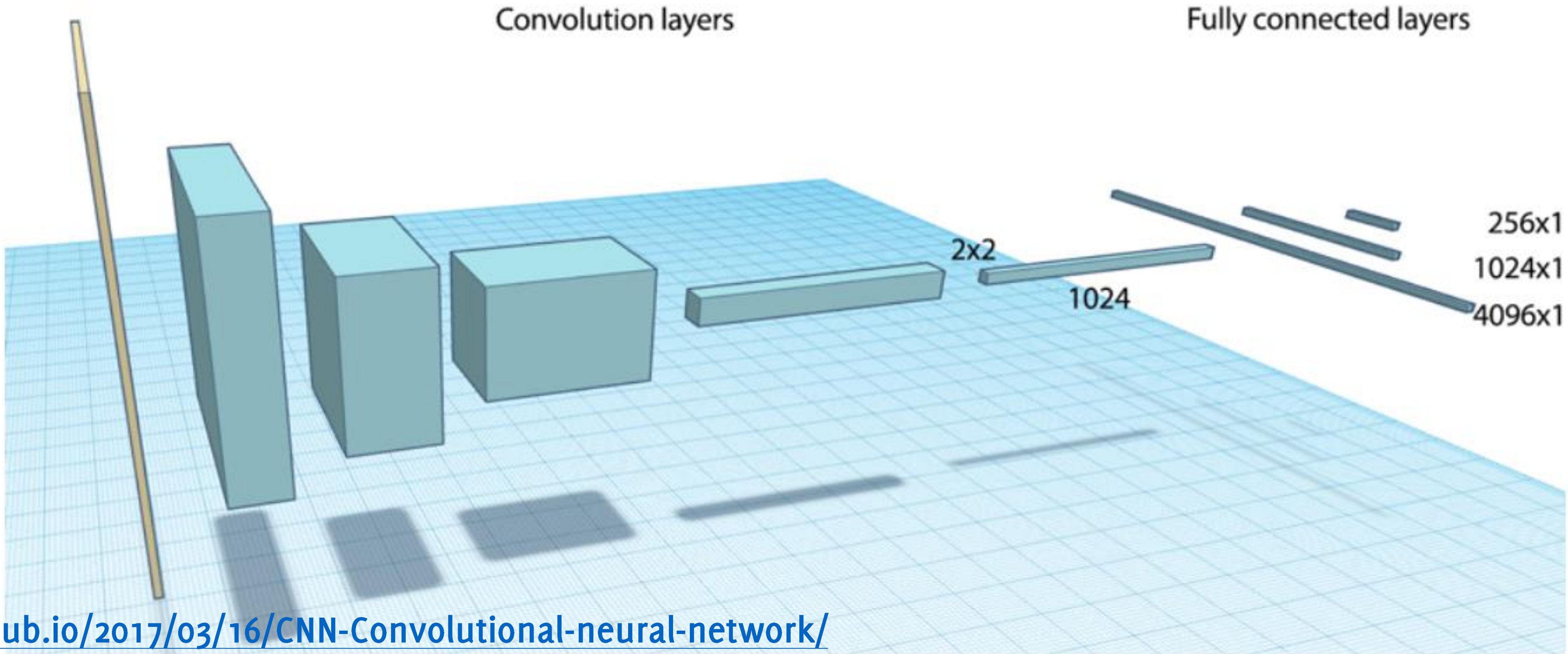


https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

A typical architecture

As we move to deeper layers:

- spatial resolution is reduced
- the number of maps increases

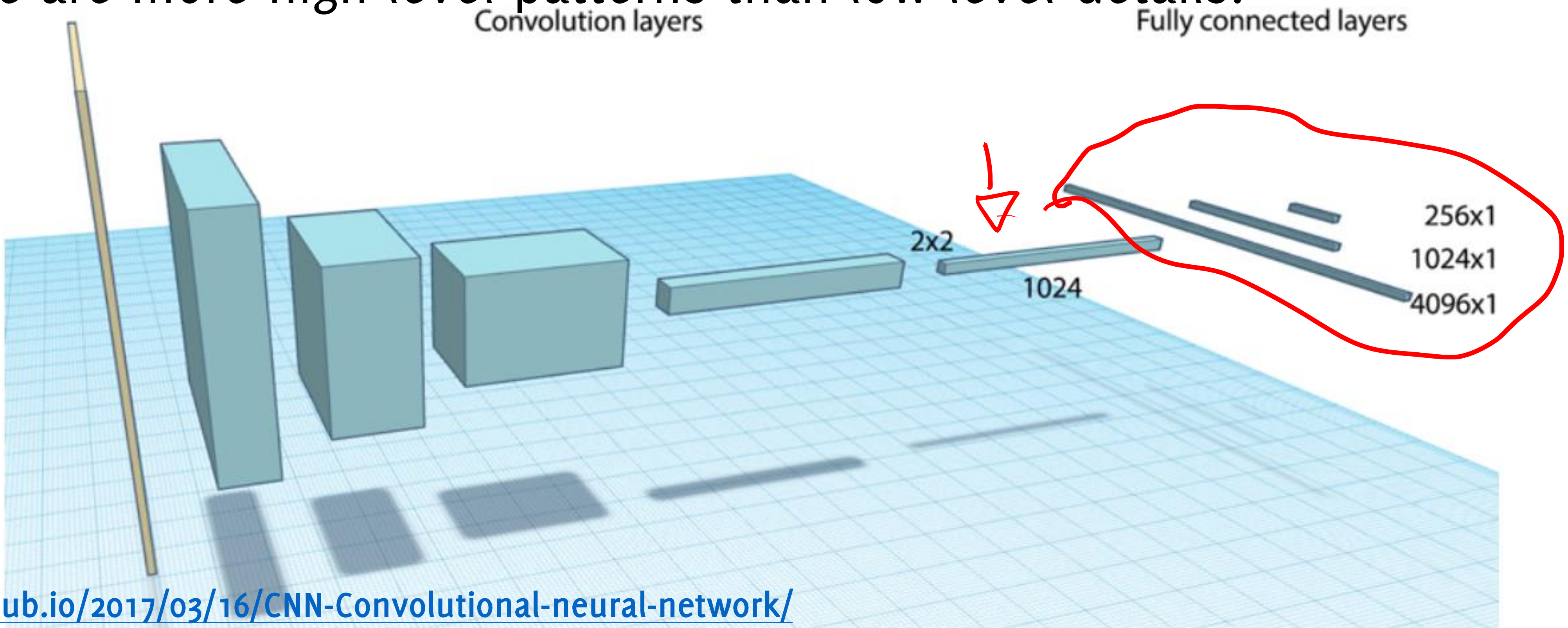


https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

A typical architecture

As we move to deeper layers:

- We search for higher-level patterns, and don't care too much about their exact location.
- There are more high-level patterns than low-level details!

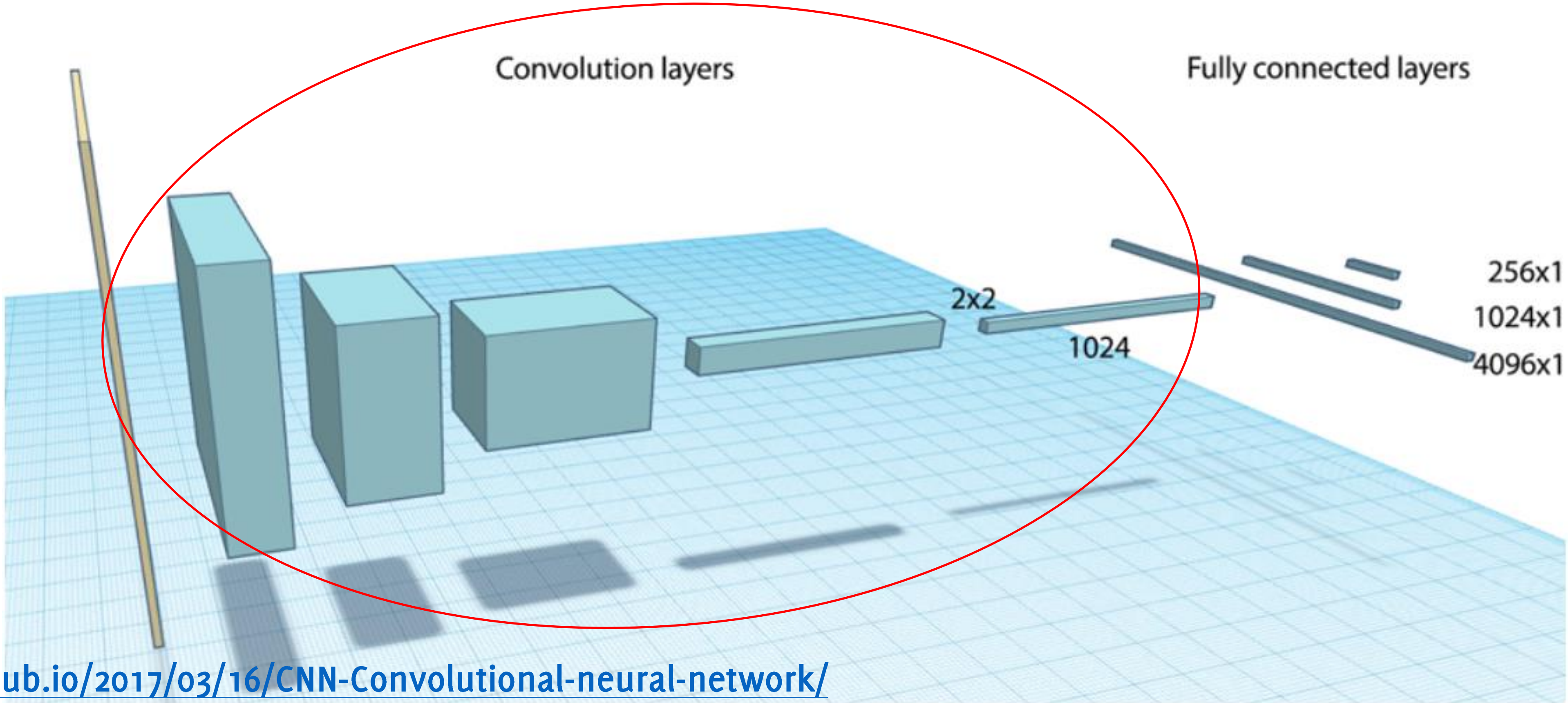


https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

A typical architecture: Convolutional Block

In VGG or Alexnet network that was trained to classify 1000 classes from Imagenet…

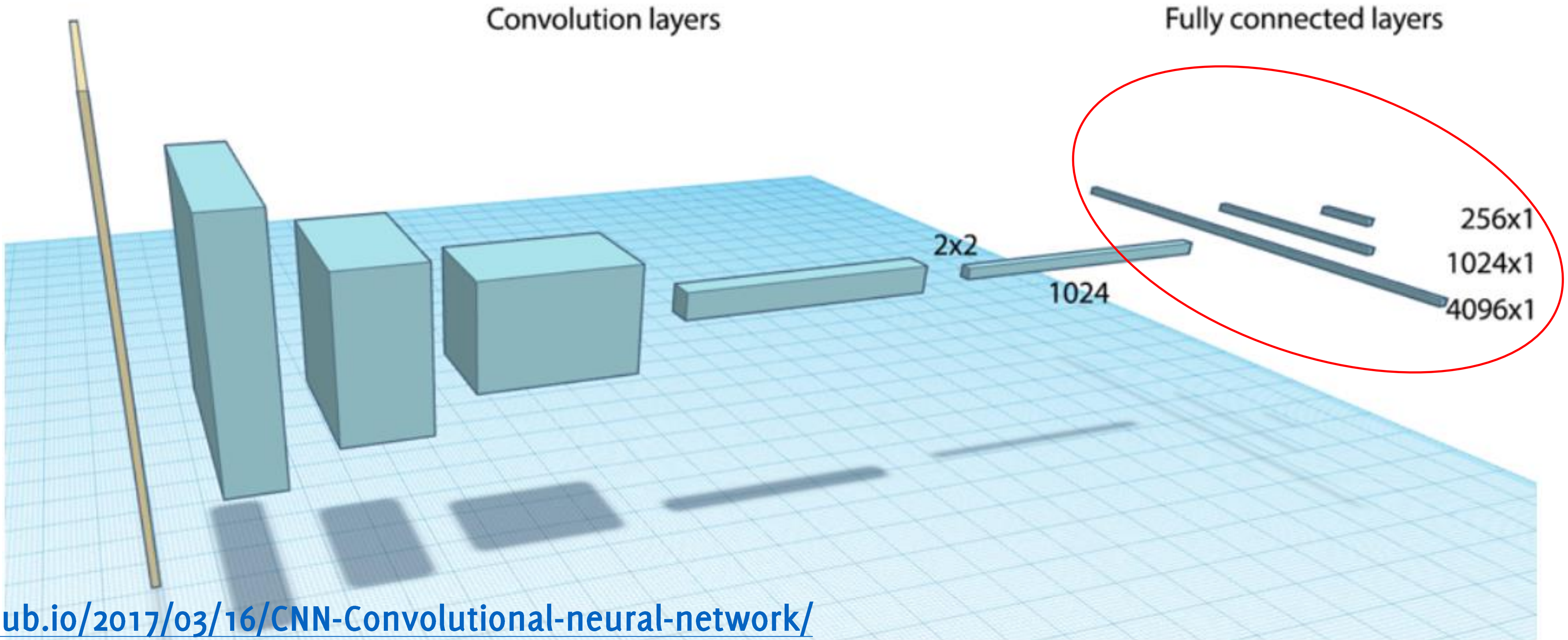
the **convolutional part should be a very general feature extractor**



https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

A typical architecture: FC layers

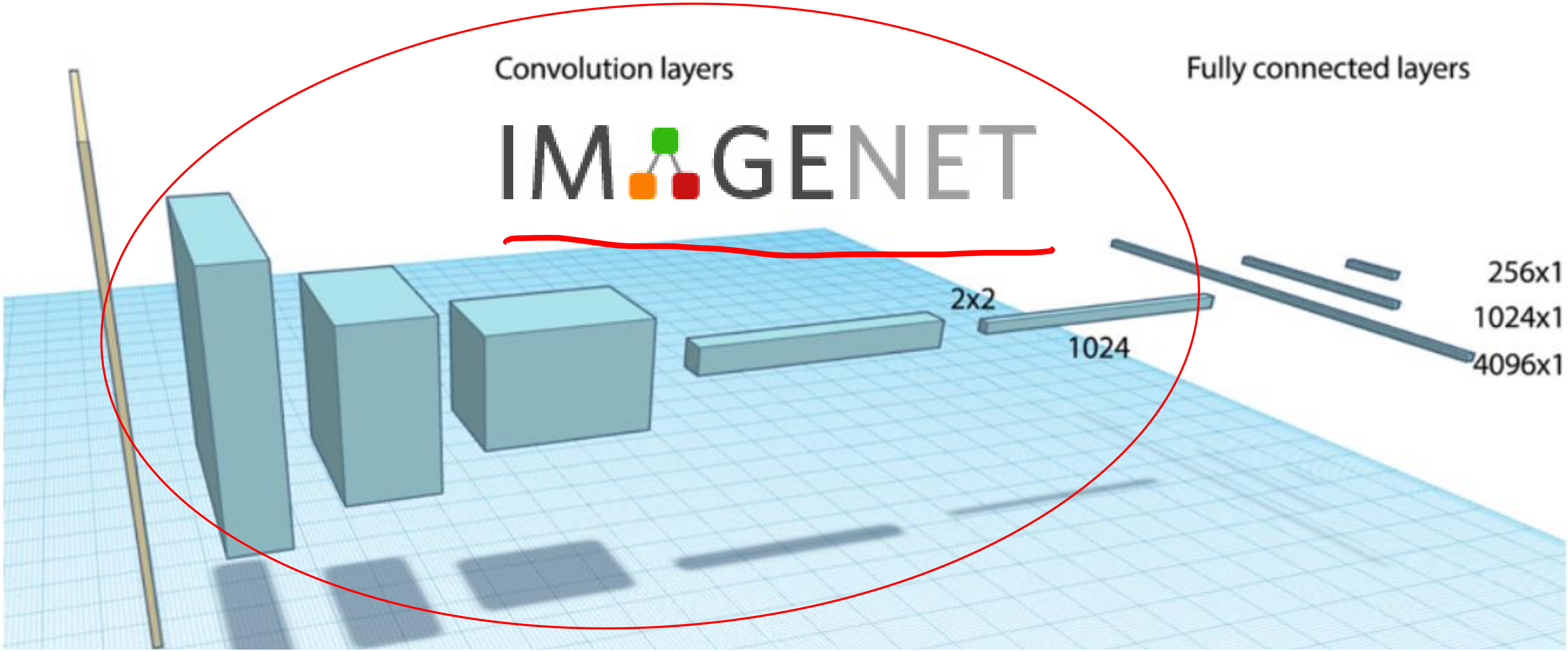
The **FC layers are instead very custom**, as they are meant to solve the specific classification task at hand



https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

Transfer Learning

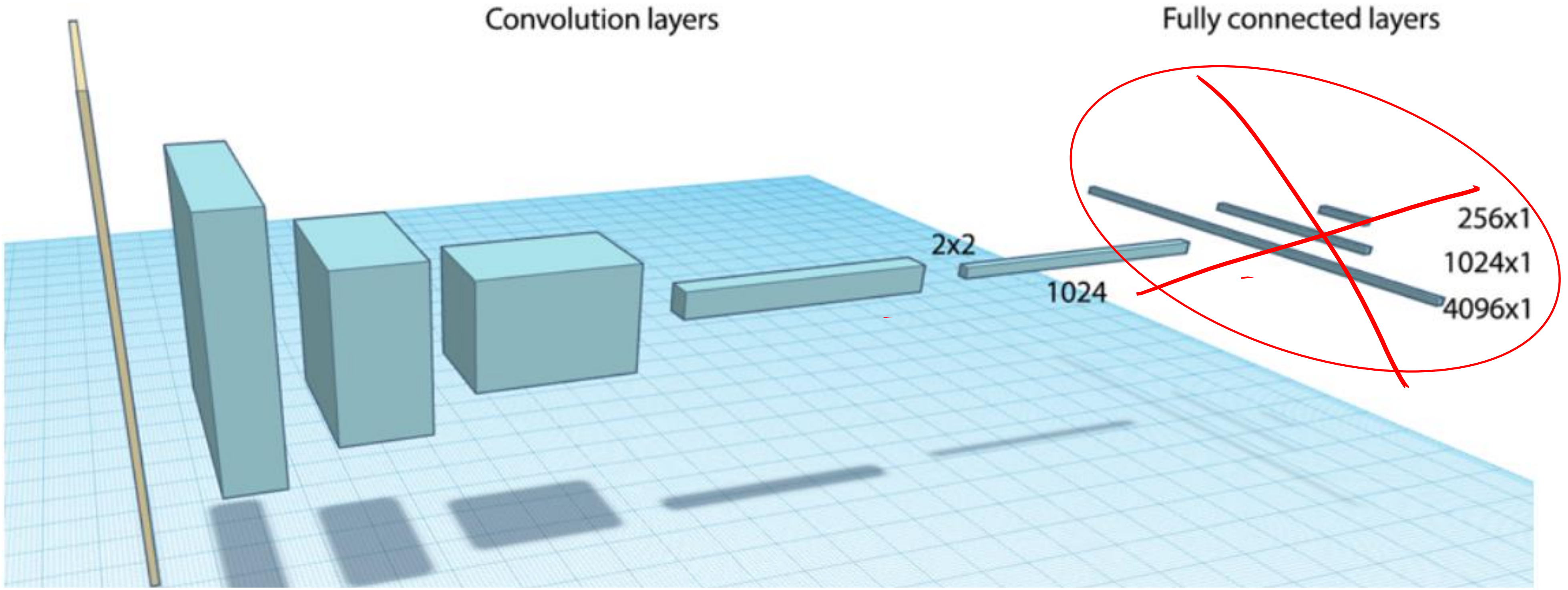
- Take a successful pre-trained model such as VGG



https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

Transfer Learning

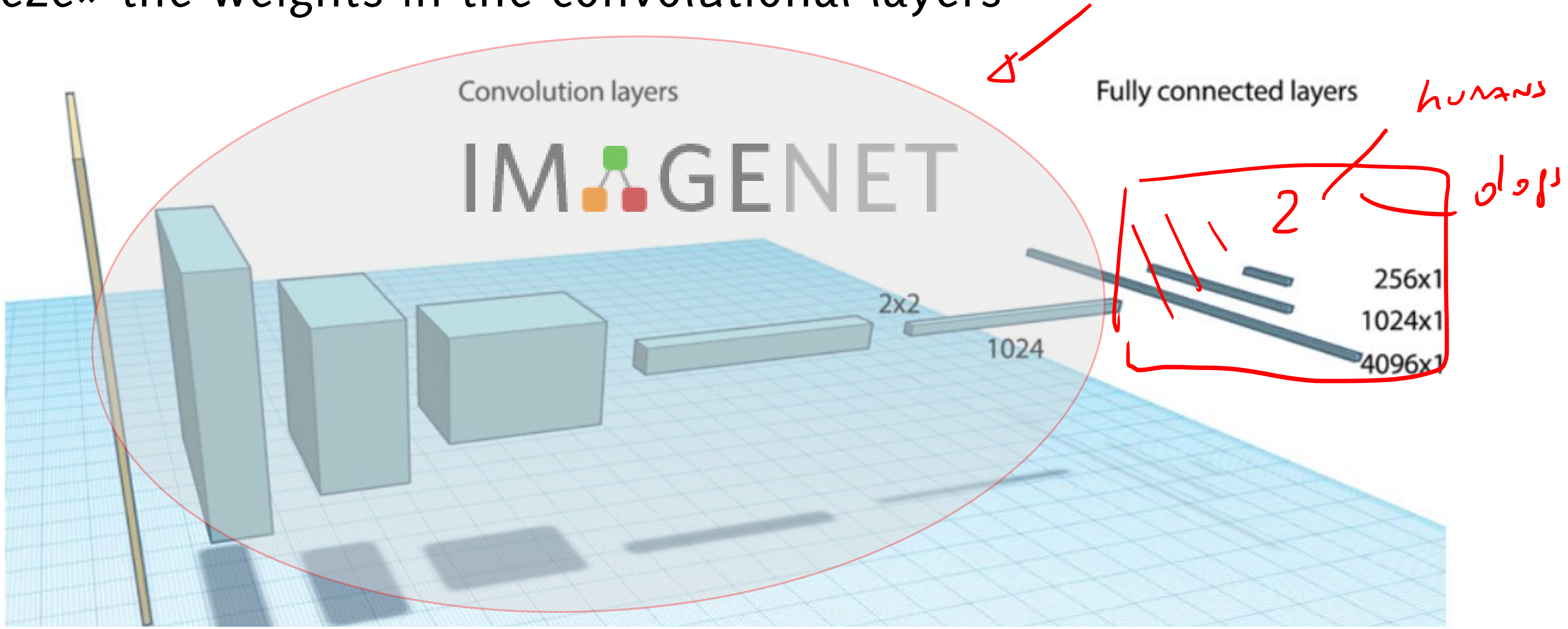
- Take a successful pre-trained model such as VGG
- Remove and modify the fully connected layers for the problem at hand



https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

Transfer Learning

- Take a successful pre-trained model such as VGG
- Remove and modify the fully connected layers for the problem at hand
- «Freeze» the weights in the convolutional layers

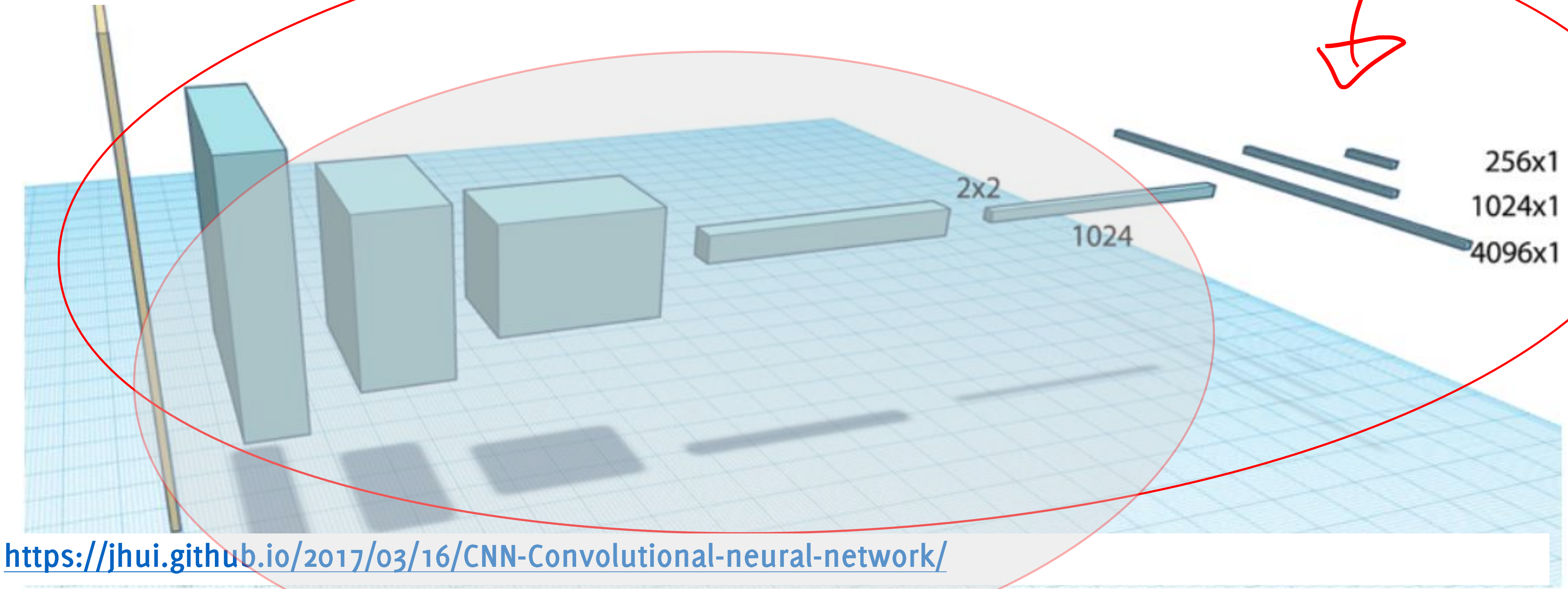


https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

Transfer Learning

- Take a successful pre-trained model such as VGG
- Remove and modify the fully connected layers for the problem at hand
- «Freeze» the weights in the convolutional layers
- Train the whole network on the new training data

TR



https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

In keras…

Pre-trained models are available, typically in two ways:

- **`include_top = True`:** provides the entire network, including the fully convolutional layers. This network can be used to solve the classification problem it was trained for
- **`include_top = False`:** contains only the convolutional layers of the network, and it is specifically meant for transfer learning.

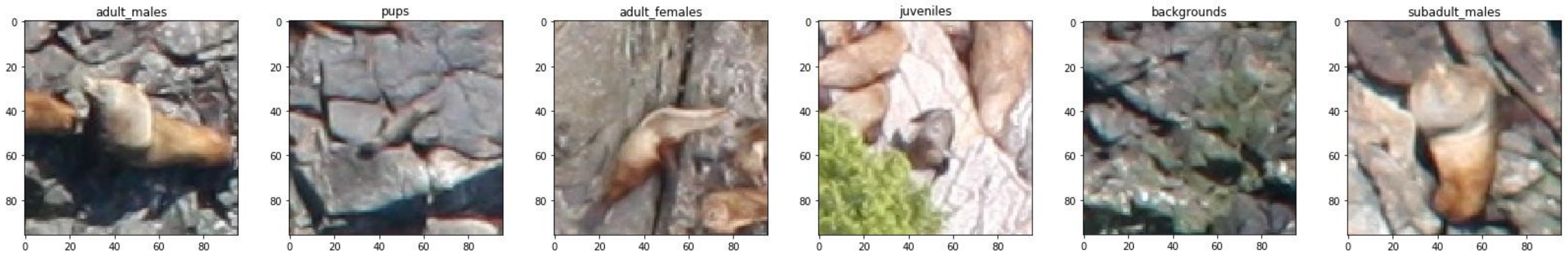
Have a look at the size of these models in the two options!

VGG16 in Keras…

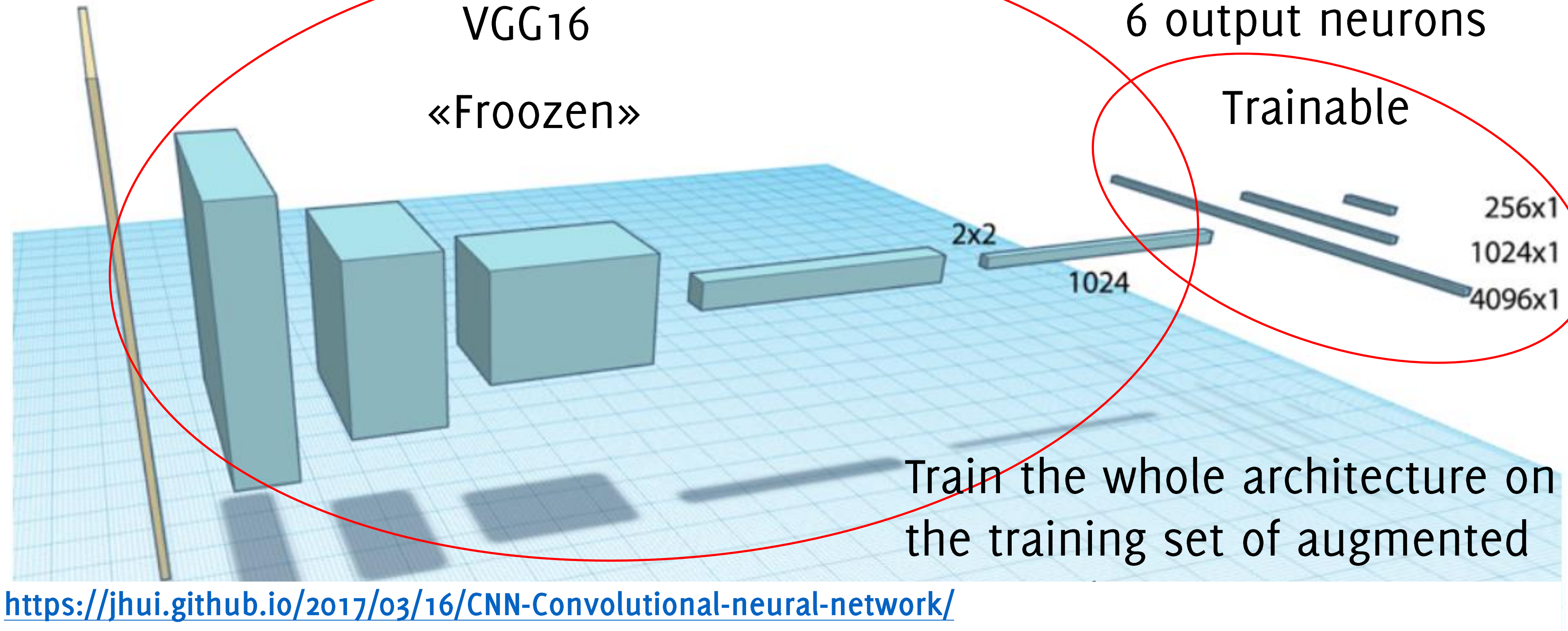
```
from keras import applications

base_model = applications.VGG16(weights =
"imagenet", include_top=False, input_shape =
(img_width, img_width, 3))
```

Transfer Learning in the Sealion Case



FC Layers having 6 output neurons



VGG16

«Froozen»

Trainable

Train the whole architecture on the training set of augmented

https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

Transfer Learning in Keras…

Requires a bit of TensorFlow Backend to add the modified Fully connected layer at the top of a pretrained model

Then, before training it is necessary to loop through the network layers (they are in **`model.layers`**) and then modify the trainable property

```
for layer in model.layers[: lastFrozen]:
    layer.trainable=False
```

Transfer Learning

Different Options:

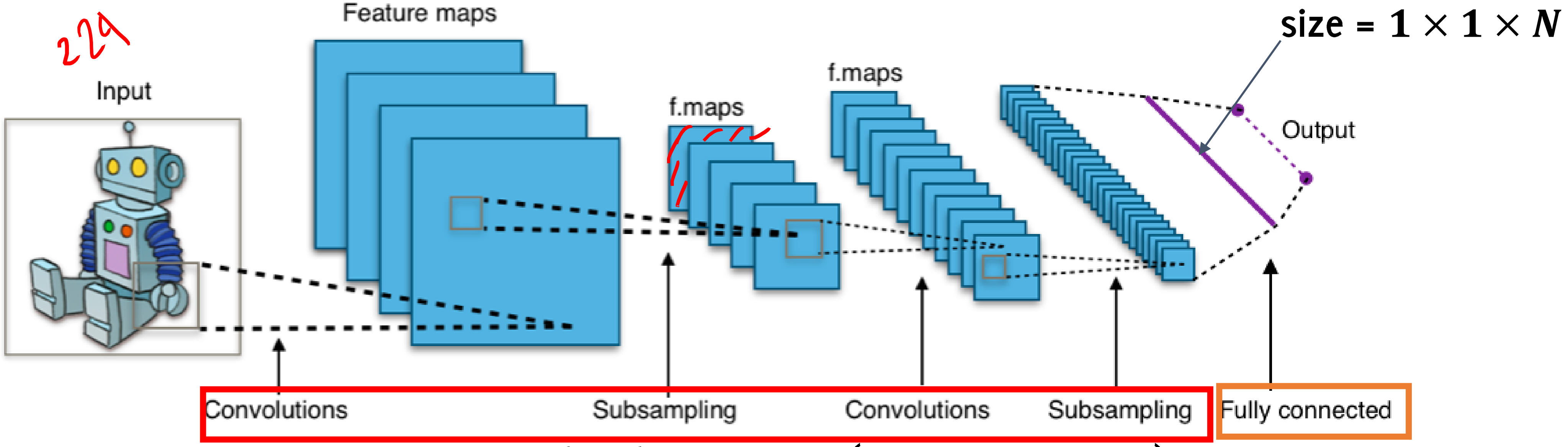
- **Transfer Learning: only the FC** layers are being trained. A good option when **little training data** are provided and the **pre-trained model** is expected to **match** the problem at hand
- **Fine tuning:** the whole **CNN is retrained**, but the **convolutional layers** are **initialized to the pre-trained model**. A good option when **enough training data** are provided or when the **pre-trained model is not expected to match** the problem at hand

Fully Convolutional Networks

What happens when chaning the input image size?

Convolutional Neural Networks (CNN)

The typical architecture of a convolutional neural network



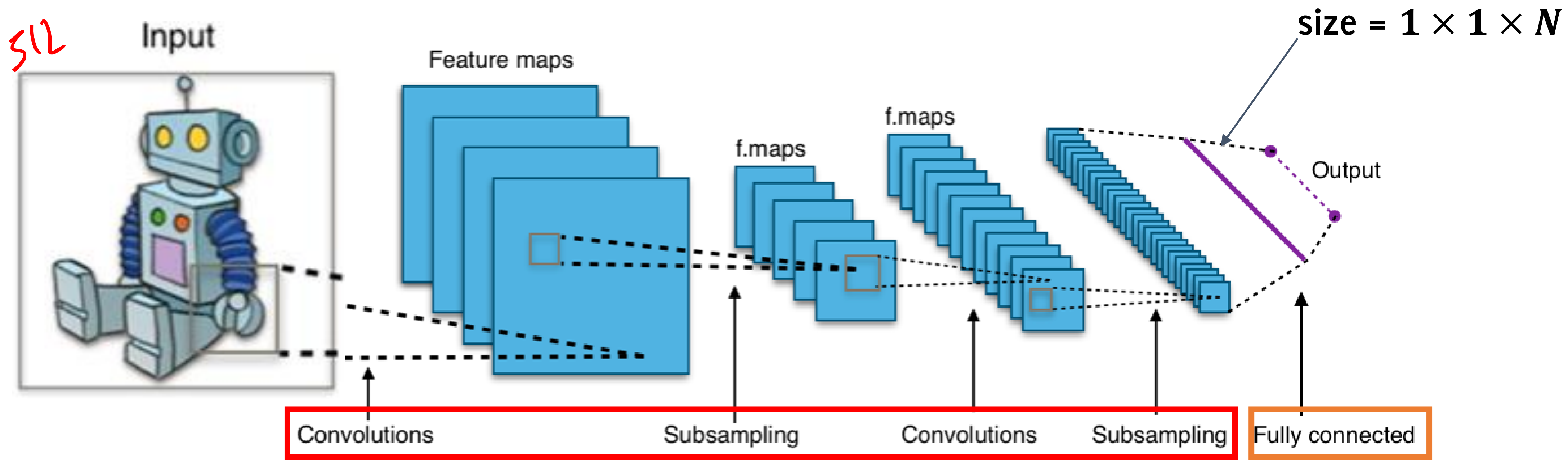
CNNs are meant to process input of a fixed size (e.g. 200 x 200).

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.

Convolutional Neural Networks (CNN)

The typical architecture of a convolutional neural network



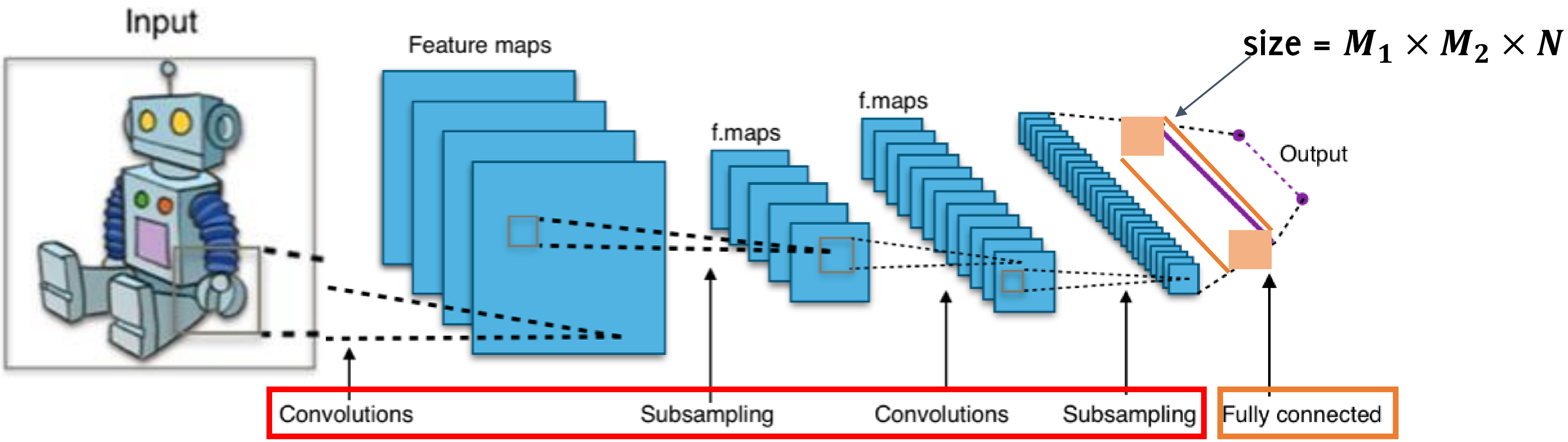
What happens when we feed a larger image to the network?

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 requires a fixed input size

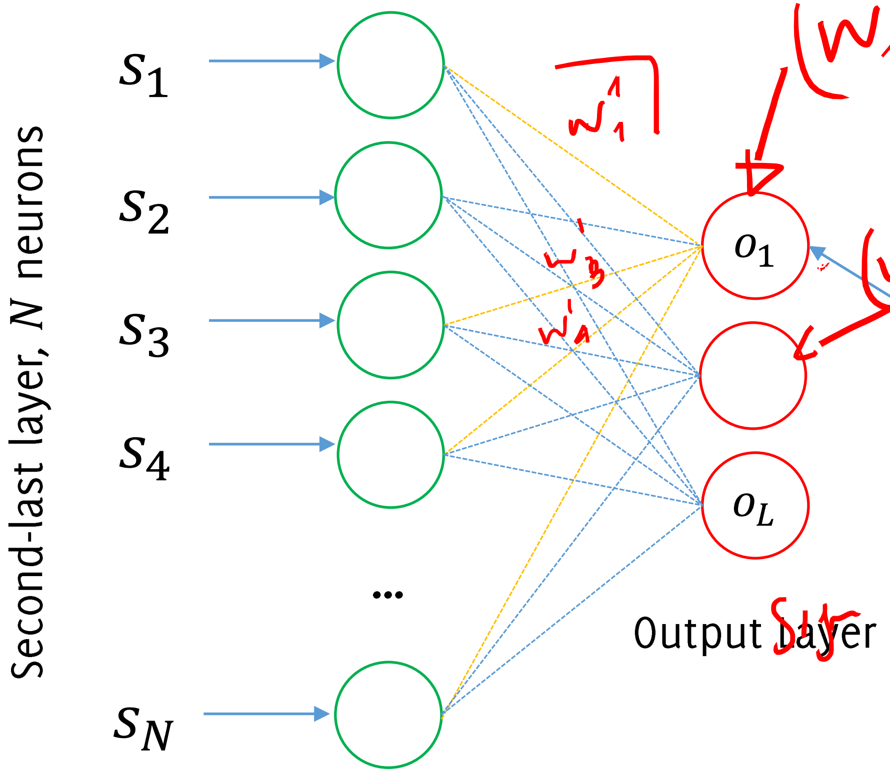
Thus, CNN cannot compute class scores, yet can extract features!



Fully Convolutional Neural Networks (f-CNN)

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

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



$$o_i = \sum_{j=1:N} w_j^i s_j + b_i$$

$$o_i = (\boldsymbol{w^i})' \, \boldsymbol{s} + b_i$$

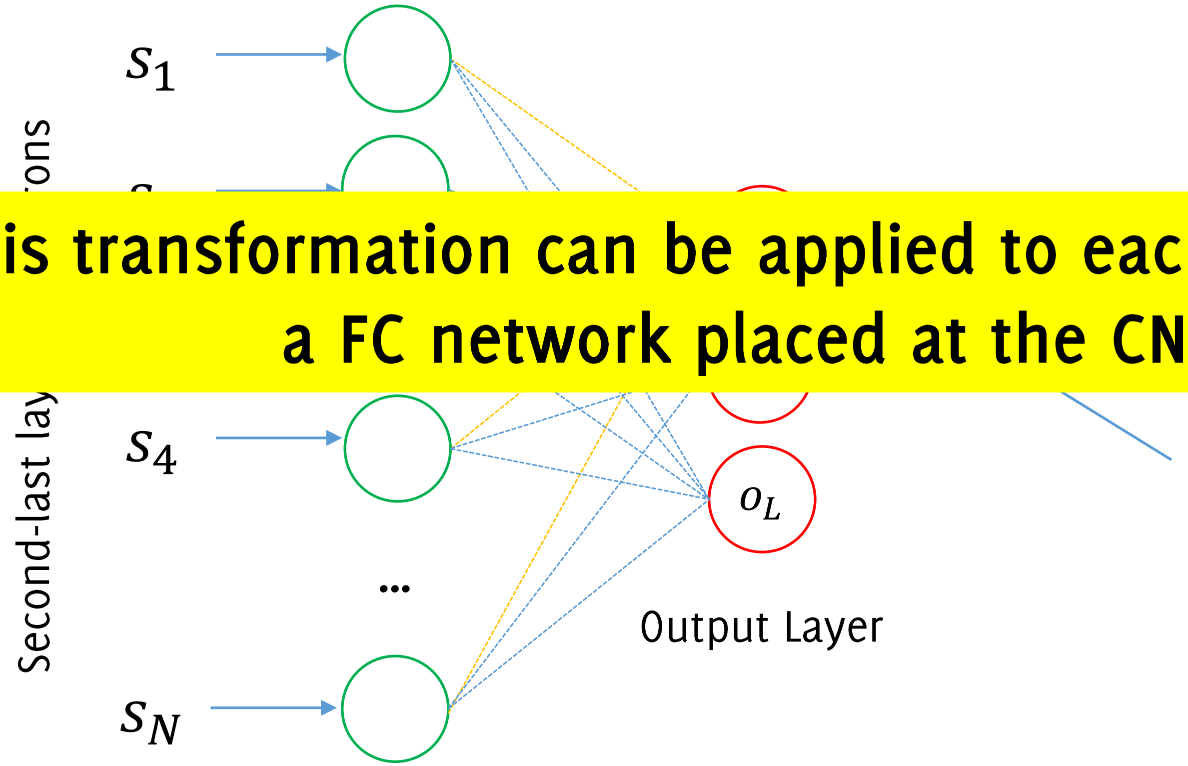
$$o_i = \boxed{(\boldsymbol{w_i} \circledast \boldsymbol{s})(0{,}0)} + b_i$$

A FC layer of L outputs is a 2DConv Layer against $\boldsymbol{L}$ **filters having size** $\boldsymbol{1 \times 1 \times N}$

Fully Convolutional Neural Networks (f-CNN)

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

Weights associated to output neuron i : $\boldsymbol{w_i} = \{w_{i,j}\}_{i=j:N}$



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

$$o_i = \left(\boldsymbol{w}^{\boldsymbol{i}}\right)' \boldsymbol{s} + b_i$$

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

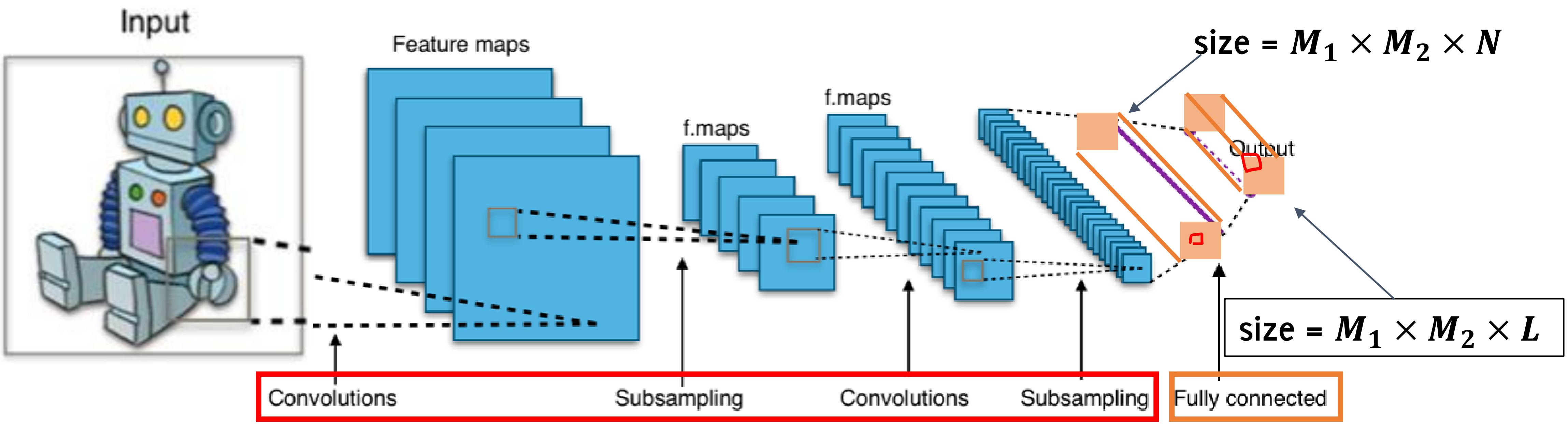
A FC layer of L outputs is a 2DConv Layer against $\boldsymbol{L}$ **filters having size** $\mathbf{1 \times 1 \times} \boldsymbol{N}$

Fully Convolutional Neural Networks (f-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

convolution $1\times1\times N$ → Activation →

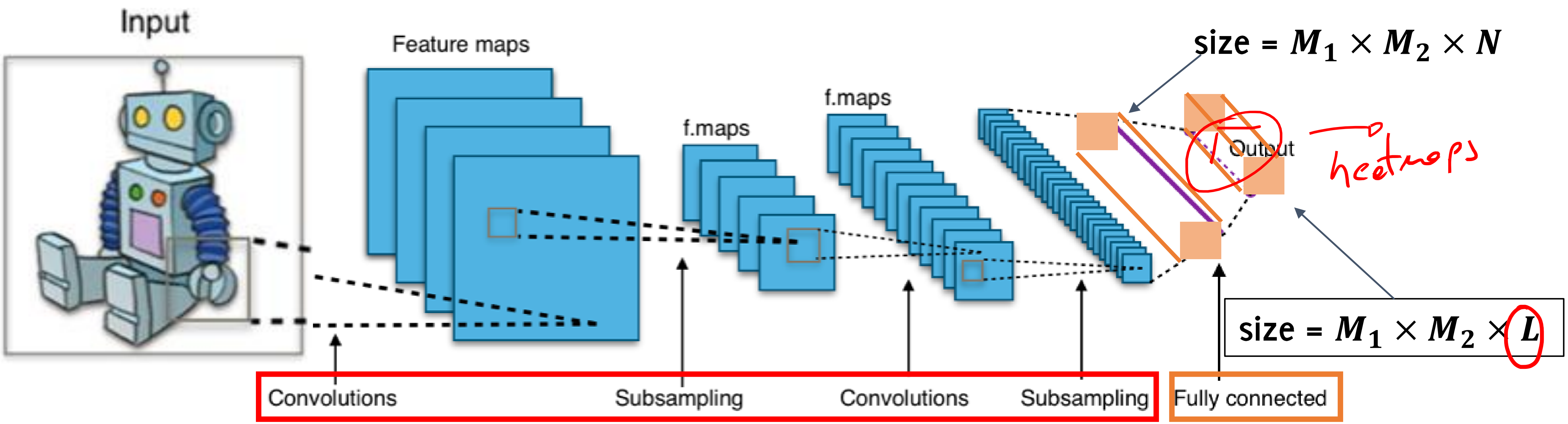


Long, J., Shelhamer, E., Darrell, T. "Fully convolutional networks for semantic segmentation". CVPR 2015

Fully Convolutional Neural Networks (f-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



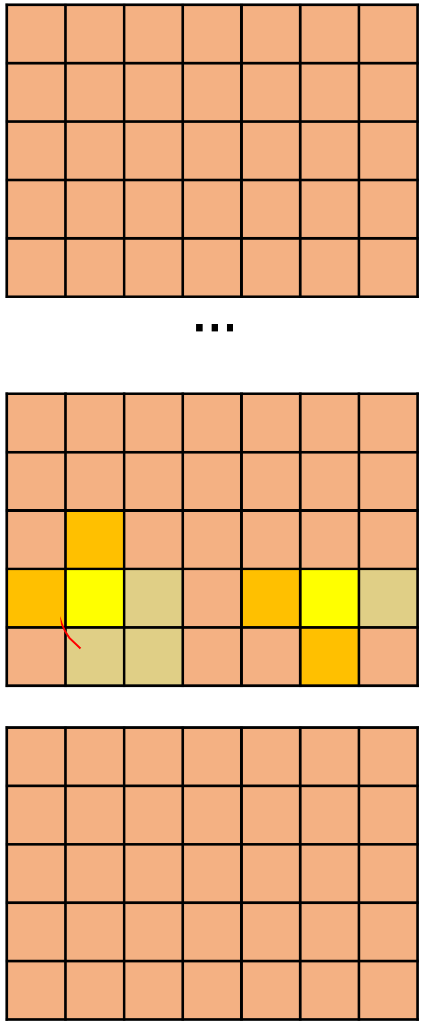
Long, J., Shelhamer, E., Darrell, T. "Fully convolutional networks for semantic segmentation". CVPR 2015

Output of a FCNN as heatmaps

A larger image than those used for training the network

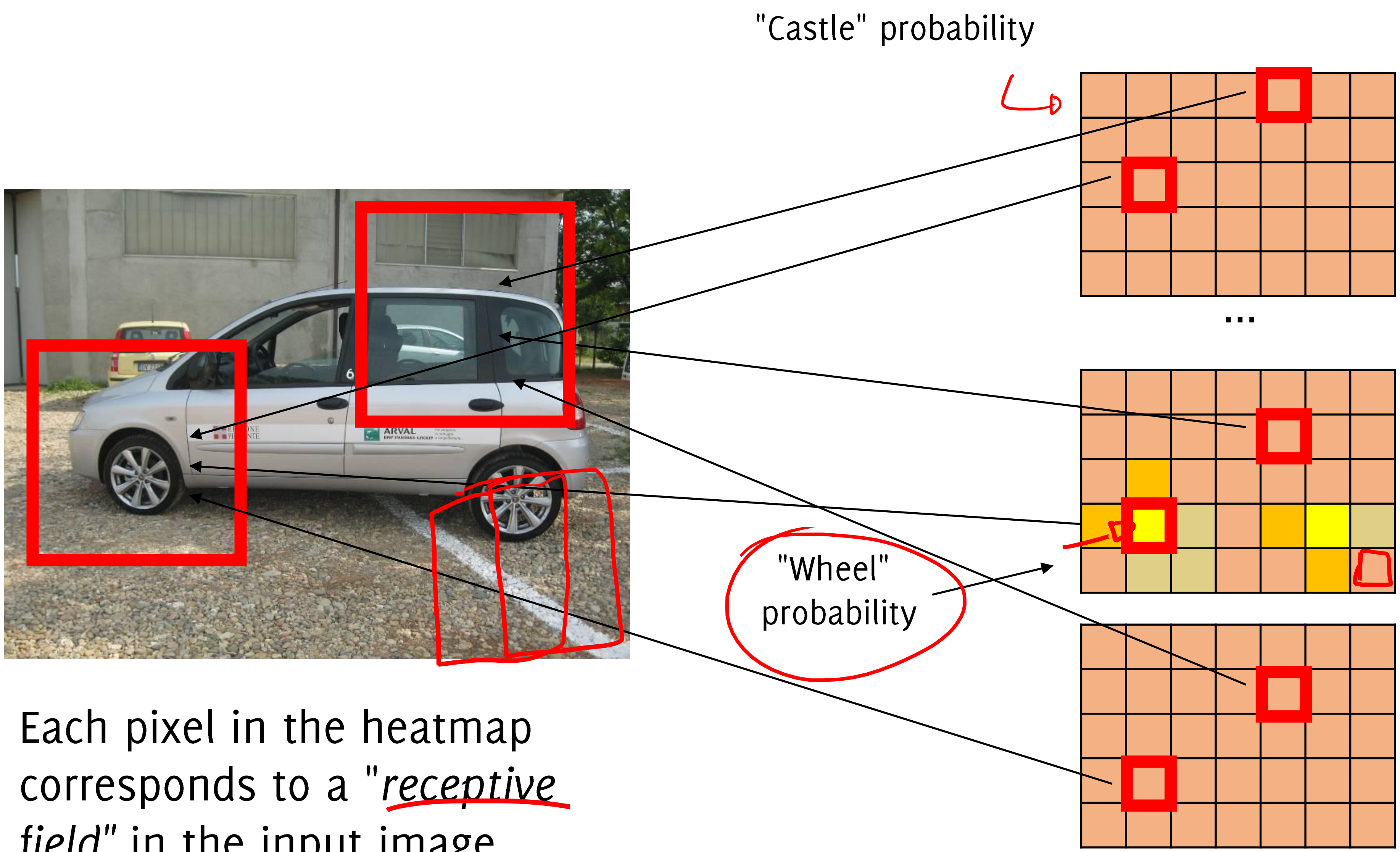
Trained Fully CNN

"Castle" probability



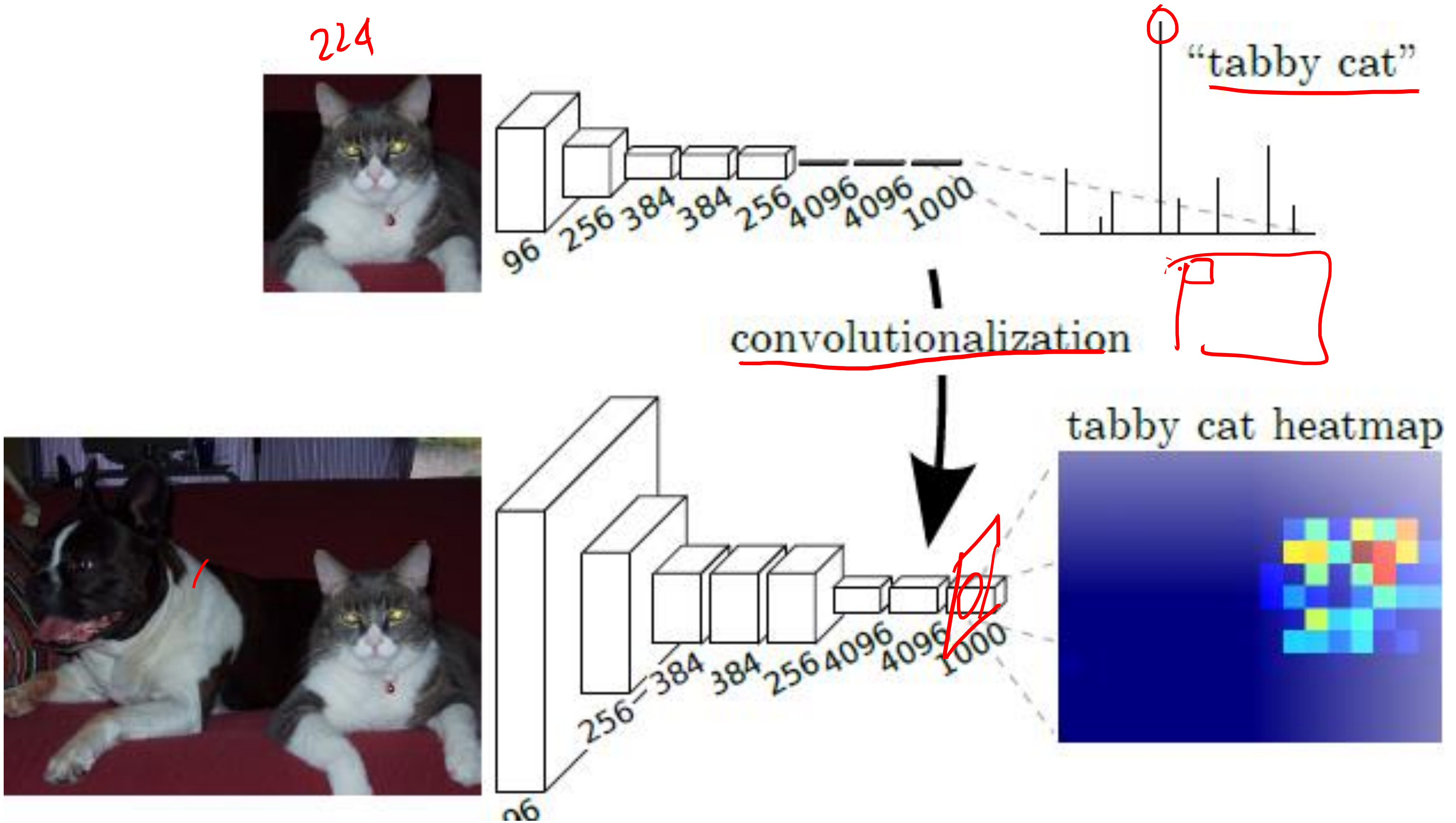
"Wheel" probability

Output of a FCNN as heatmaps



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

Migration to FCNN of a pretrained model

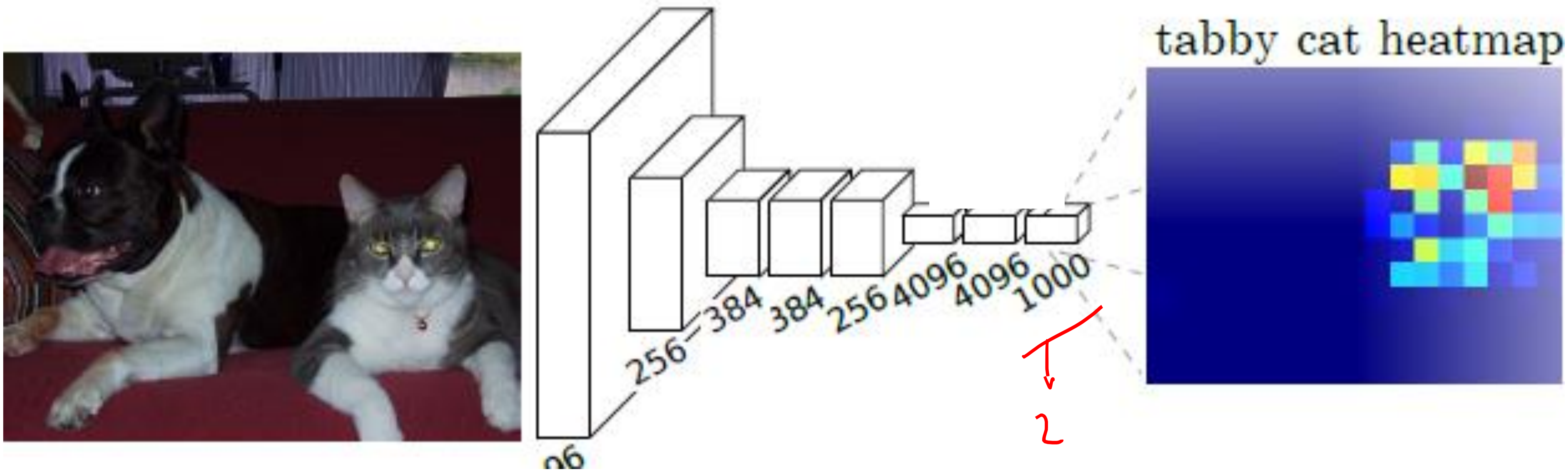


Long, Jonathan, *Evan* Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

Migration to FCNN of a pretrained model

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

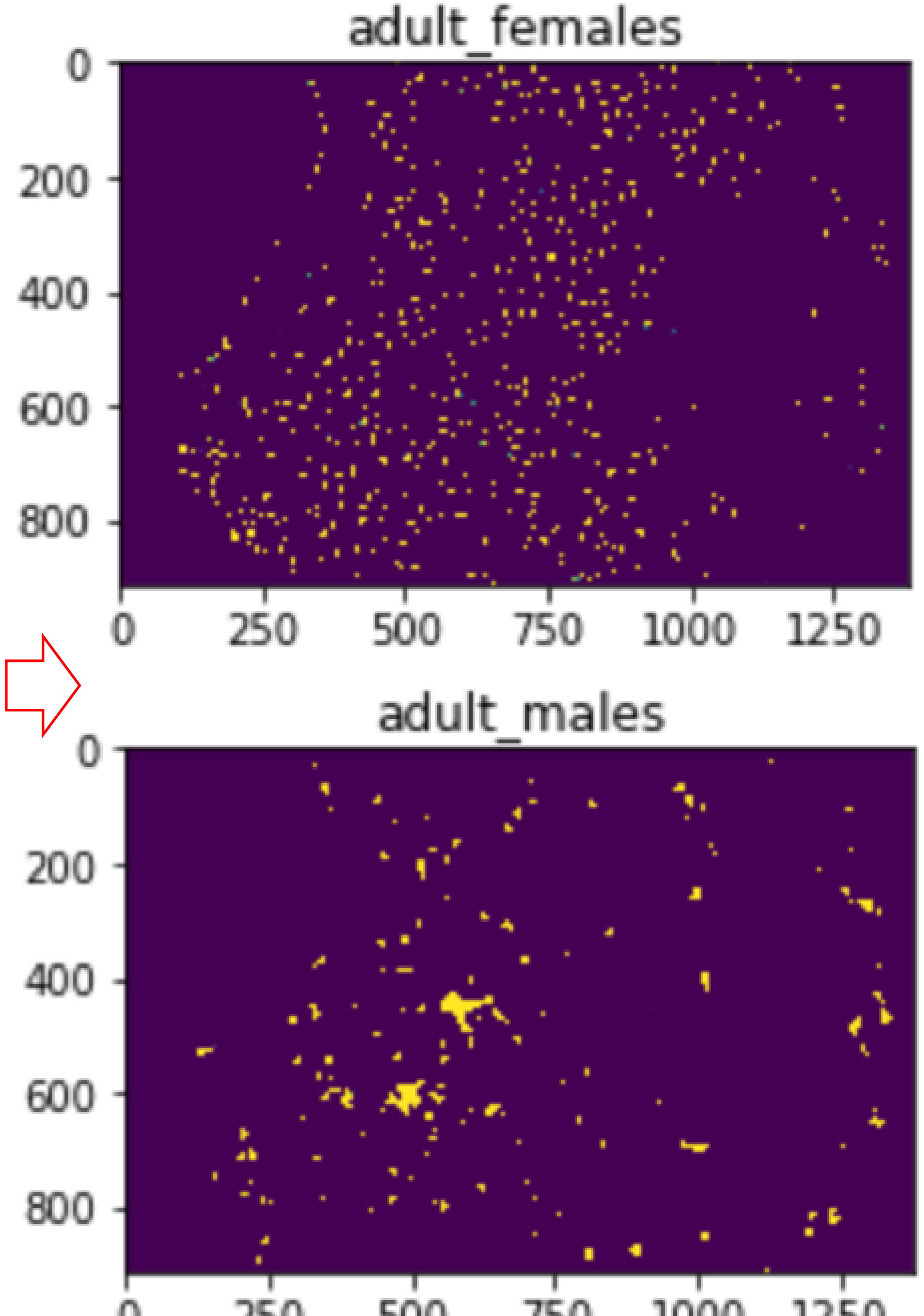
Significantly more efficient than patch extraction and classification (avoids multiple repeated computations within overlapping patches)



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

Sealion HeatMaps

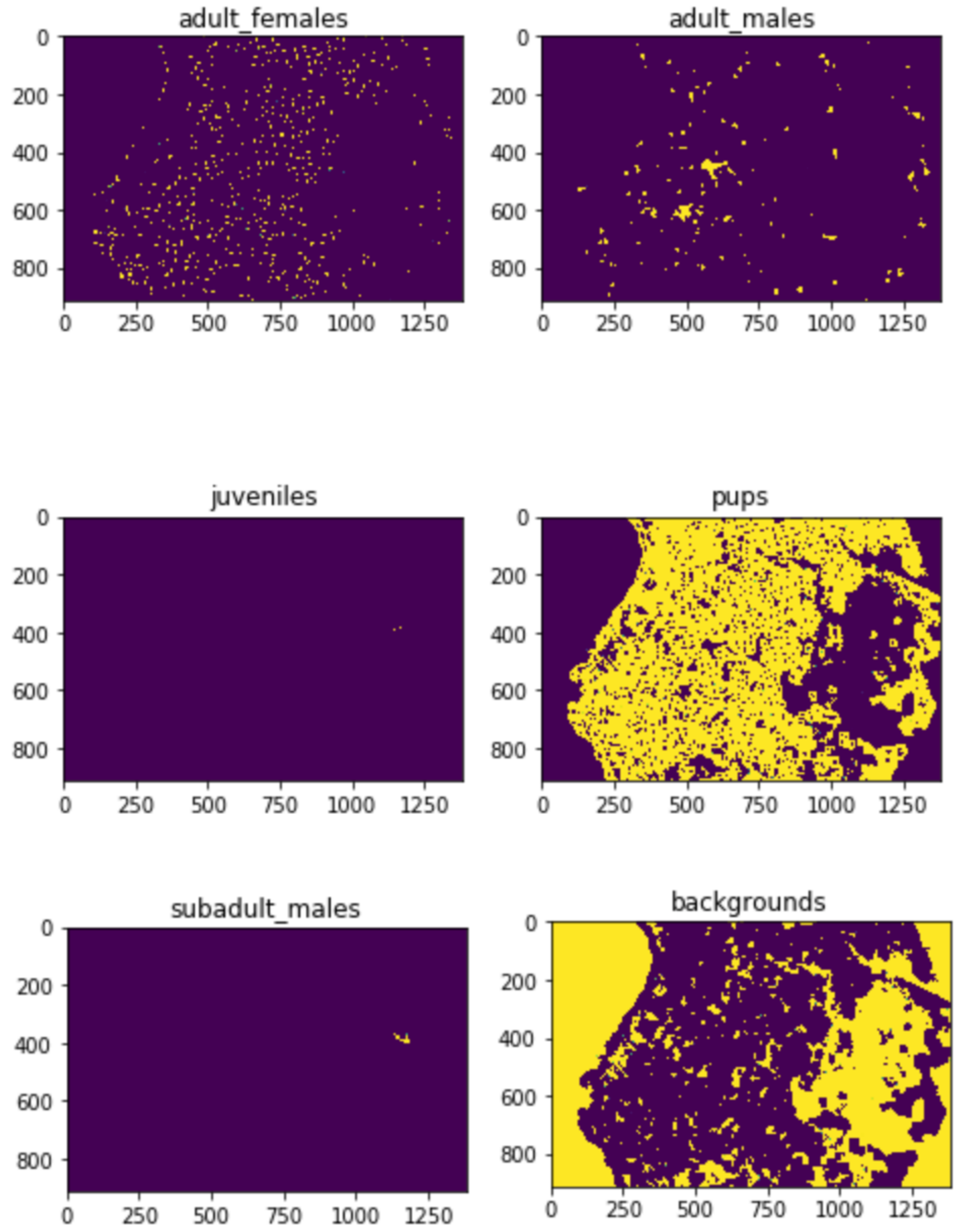
FCNN



Credits Yinan Zhou

https://github.com/marioZYN/FC-CNN-Demo

Sealion HeatMaps



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 a convolution

```
w7.reshape(20, 20, 10, 256)
```

- assign these weights to the FCNN architecture

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

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.

$S(r,c) \in \Lambda$

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

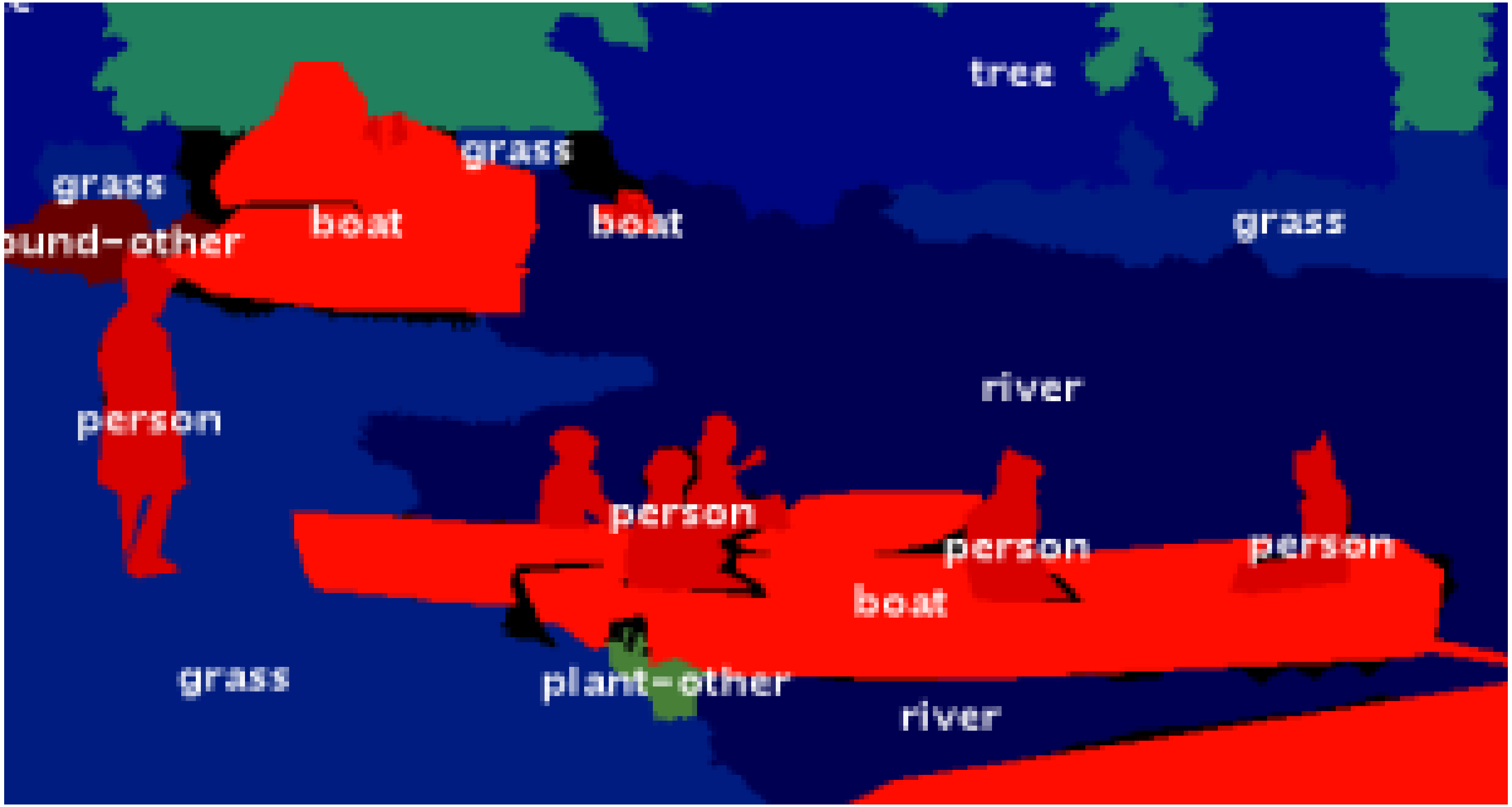
I

GT

http://cocodataset.org/

Training Set

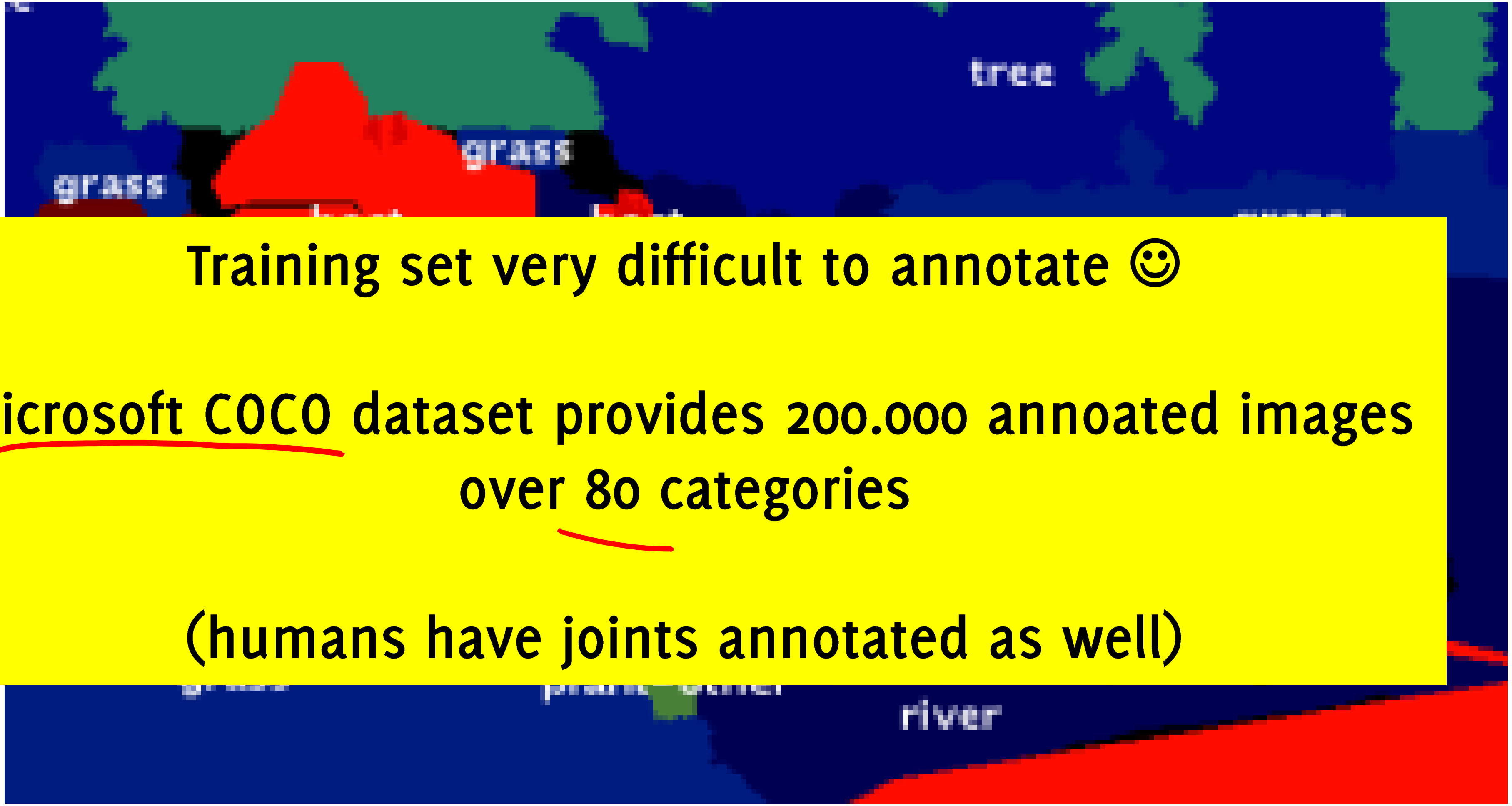
The training set is made of pairs (I, GT), where the GT is a pixel-wise annotated image over the categories in Λ

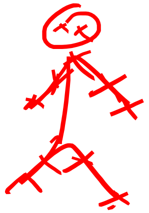


http://cocodataset.org/

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 annotated images over 80 categories

(humans have joints annotated as well)

http://cocodataset.org/

Semantic Segmentation by Fully Convolutional Neural Networks

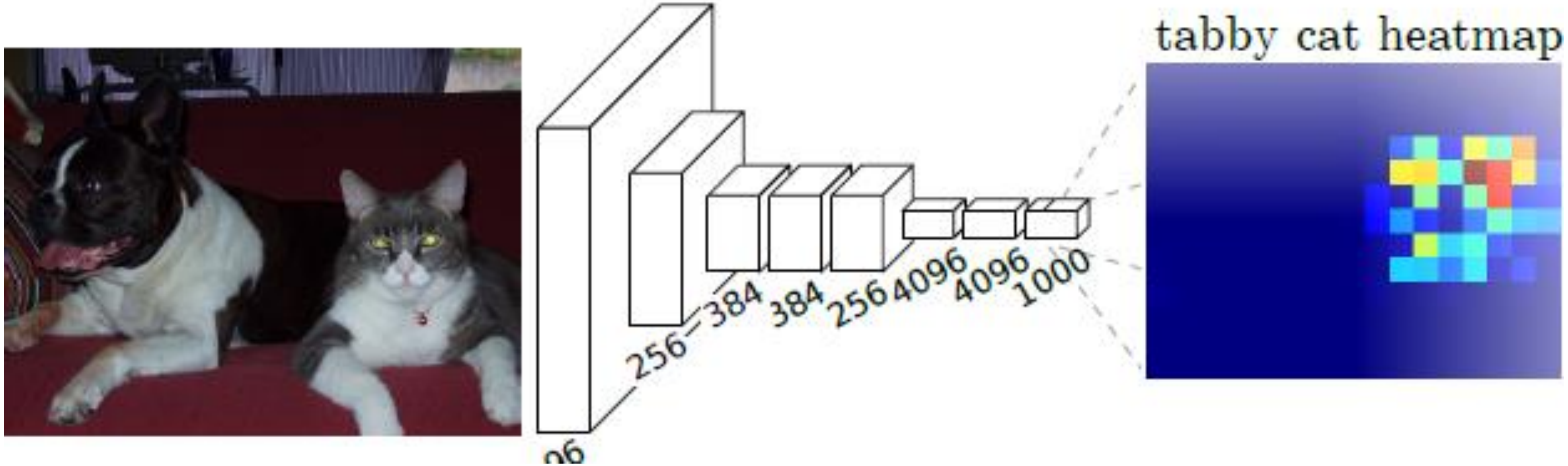
Predicting dense outputs for abritrary-sized inputs

Fully Convolutional Networks for Semantic Segmentation

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

Simple Solution (1): Direct Heatmap Predictions

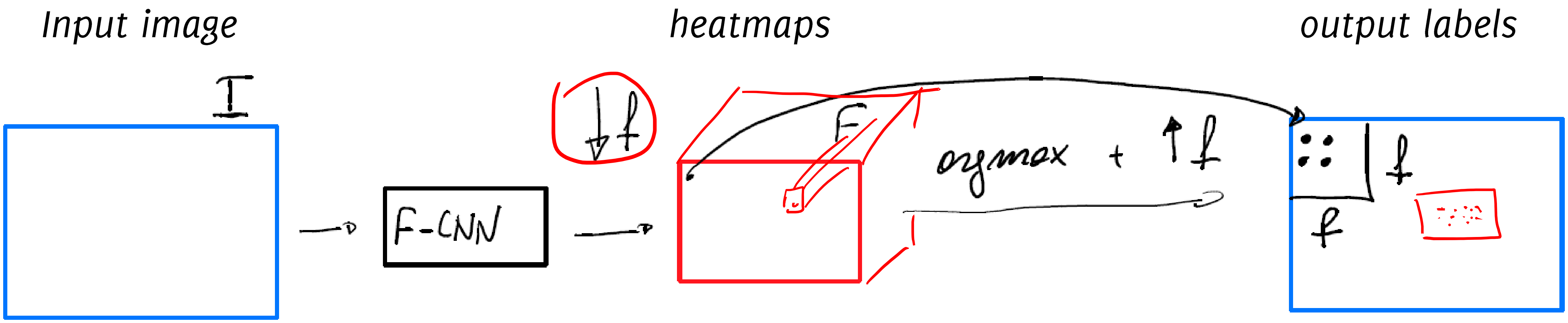
We can assign the predicted label in the heatmap to the whole receptive field, however that would be a **very coarse estimate**



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

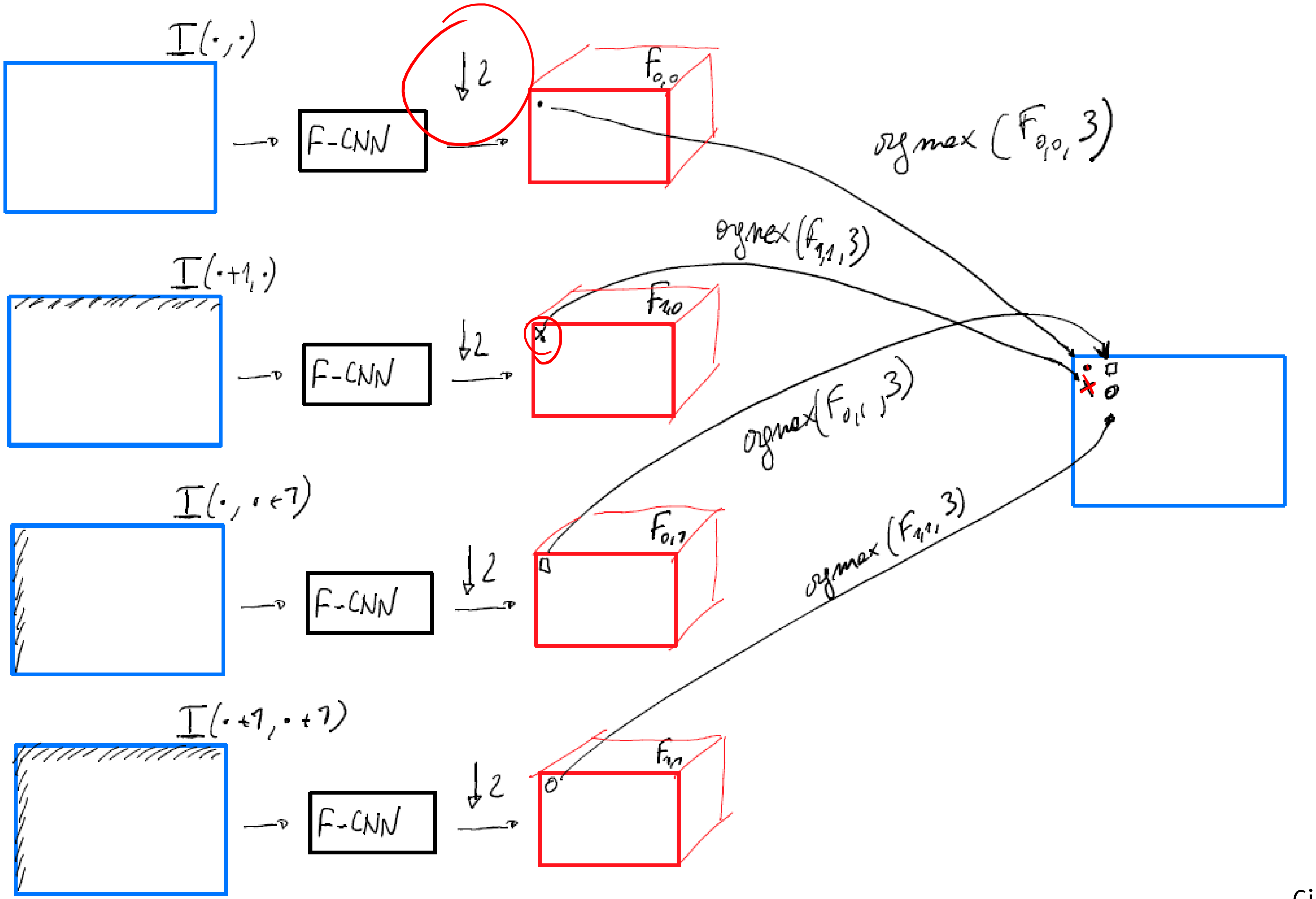
However, **the upsampling method is very rigid**

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

Input image

heatmaps

output labels

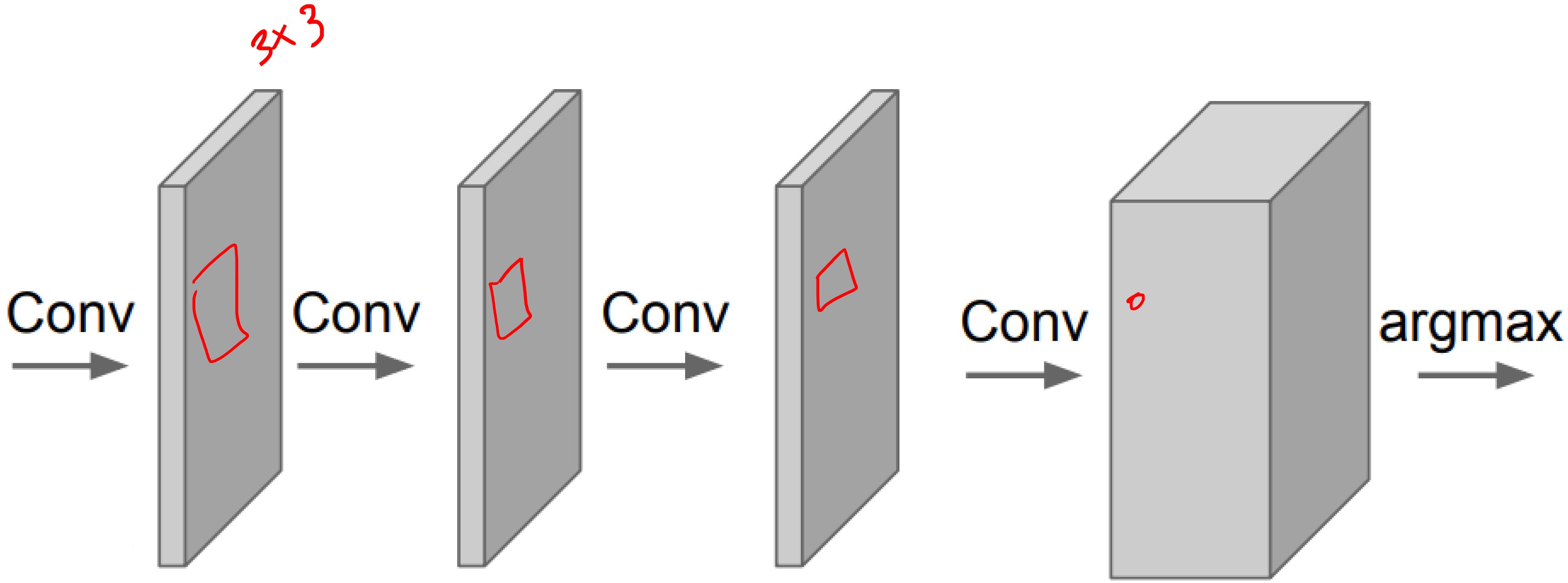


Giacomo Boracchi

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:

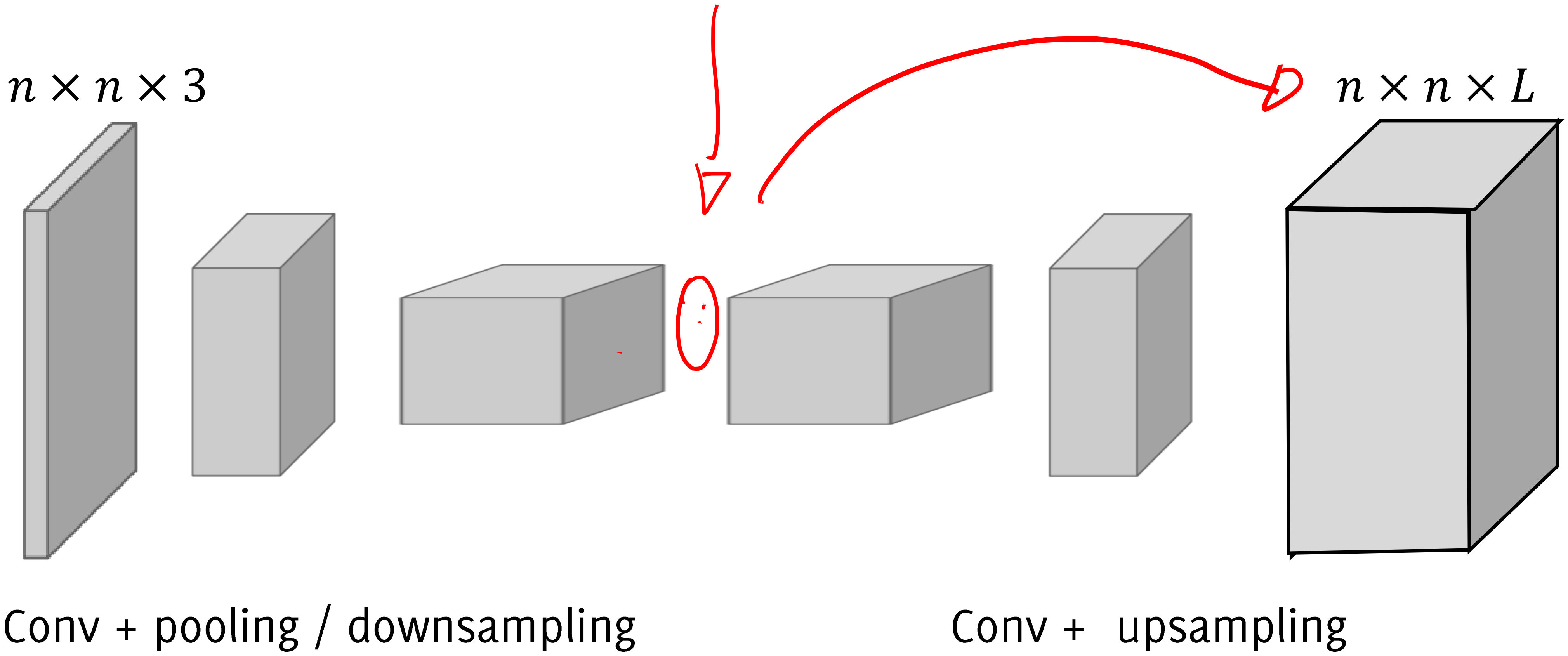
- ***global information resolves what**, while*
- ***local information resolves where***

***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

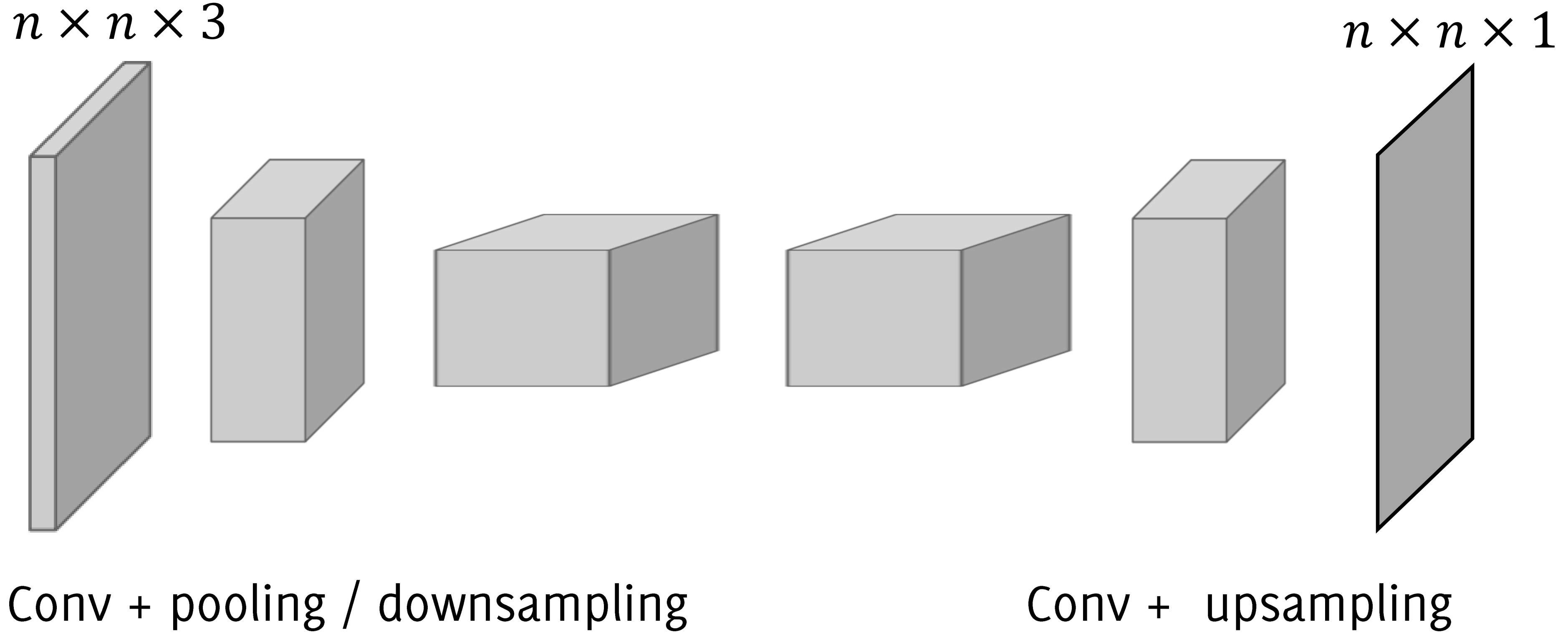
Reduce latent representation dimension

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



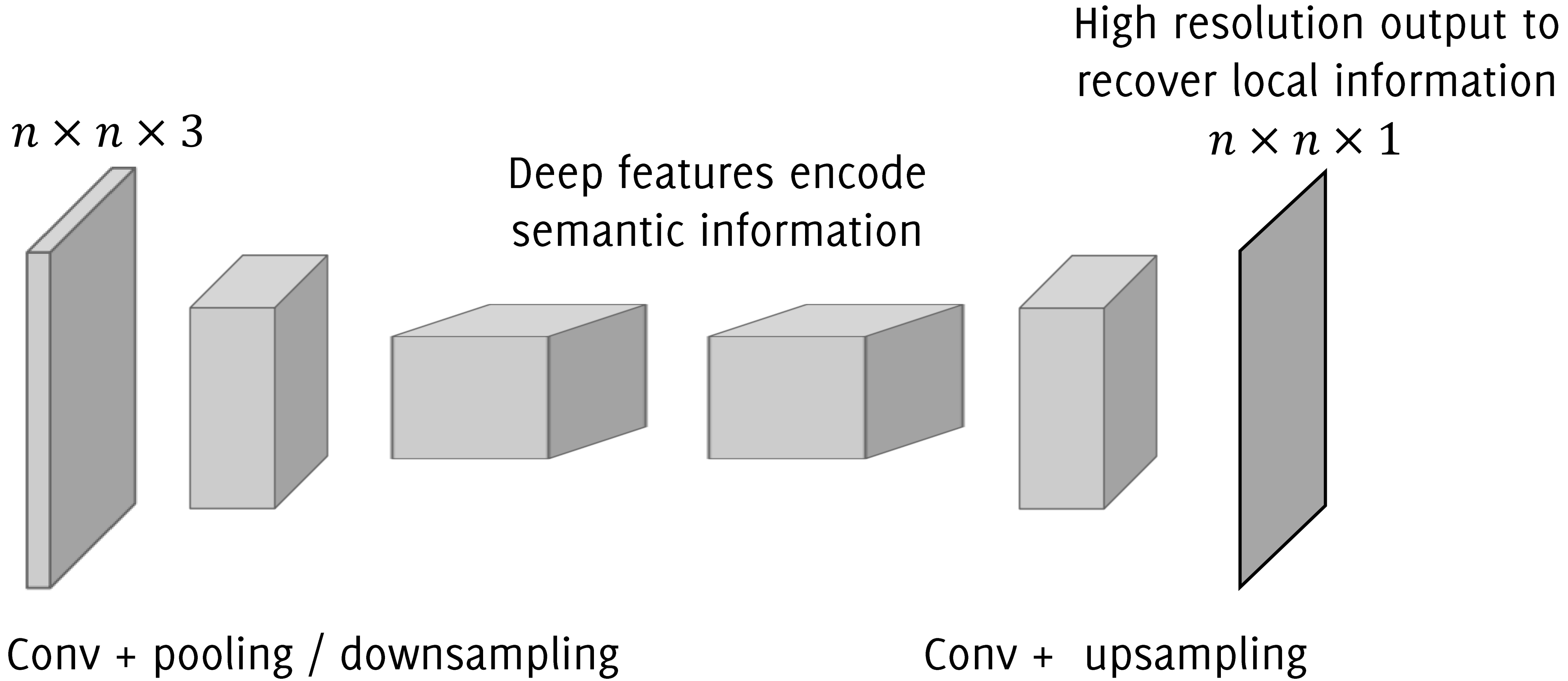
Reduce latent representation dimension

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



Reduce latent representation dimension

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

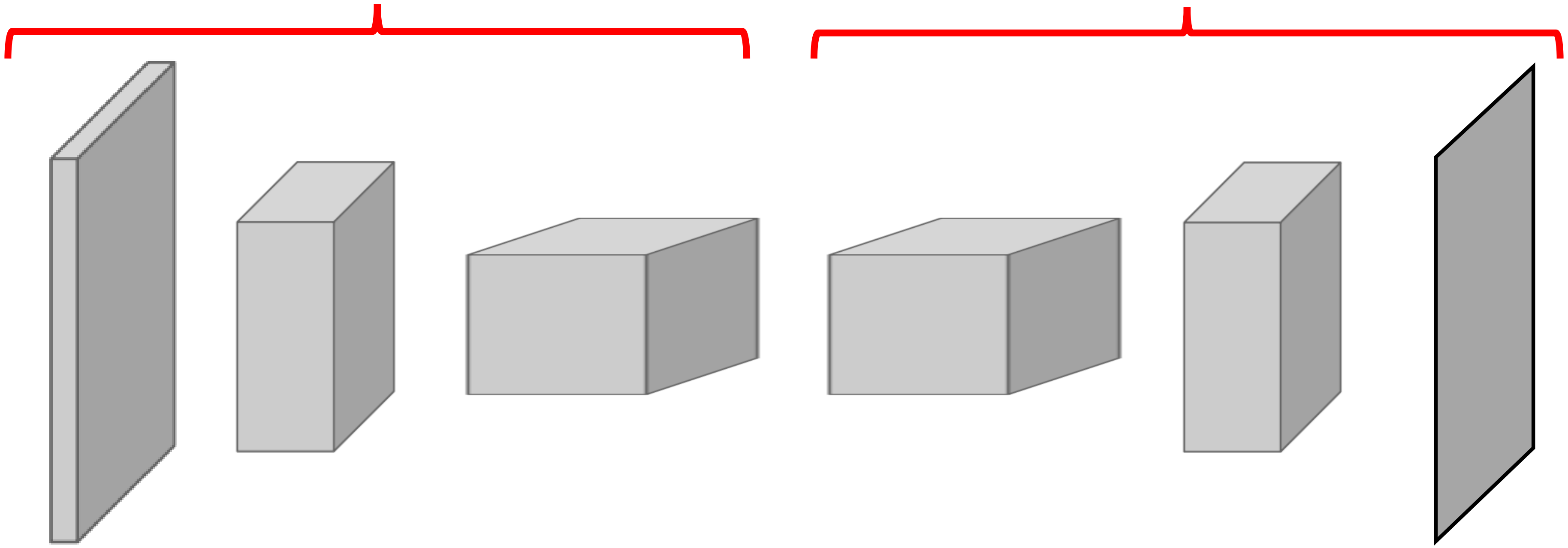


Reduce latent representation dimension

An architecture like the following would be probably better for semantic segmentation

The first half is the same of a classification network

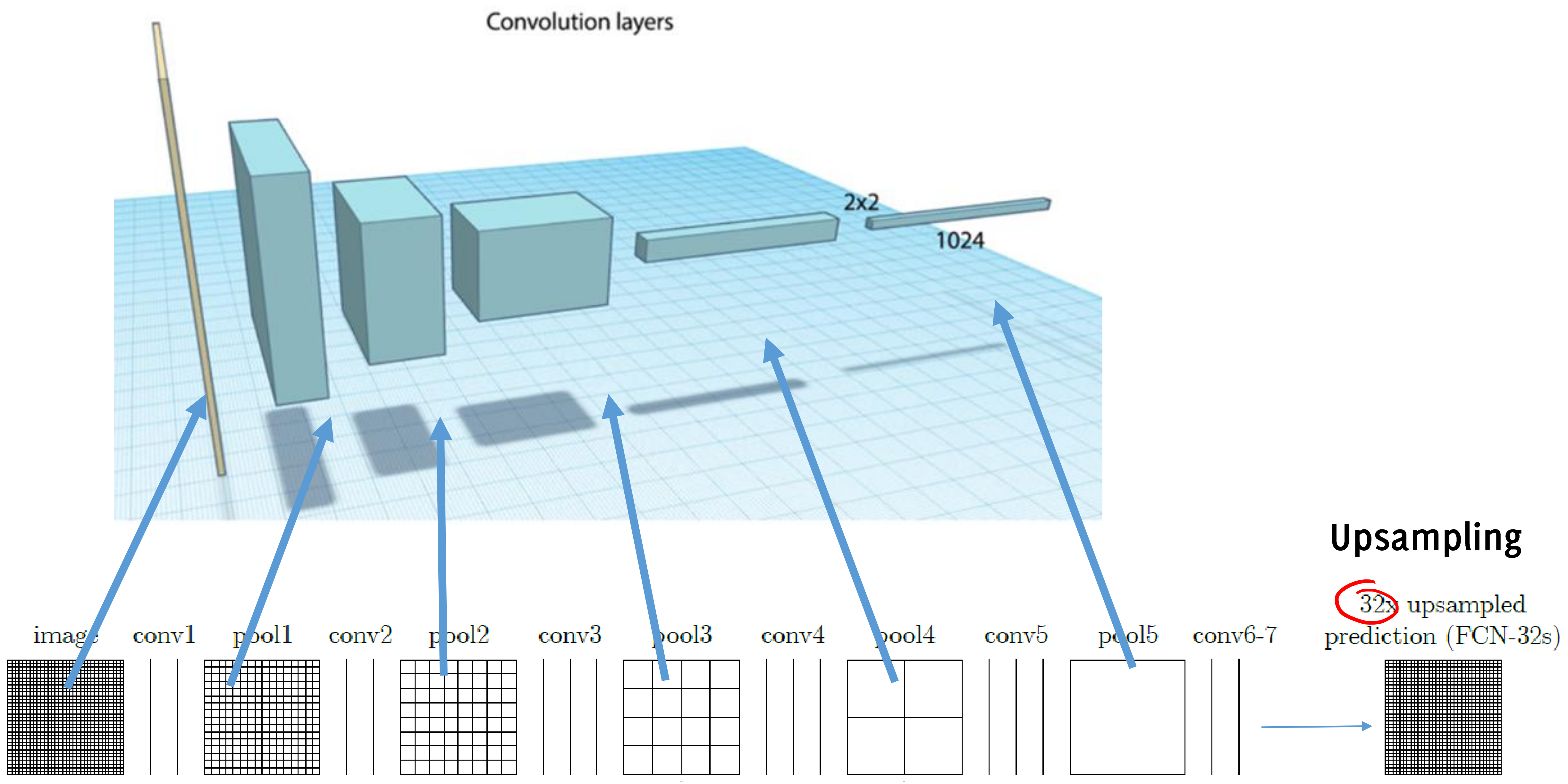
The second half is meant to upsample the predictions to cover each pixel in the image



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

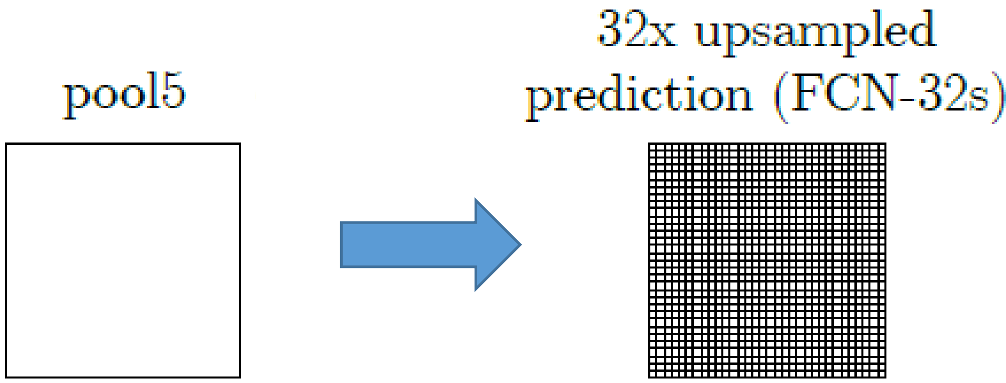
Conv + pooling / downsampling

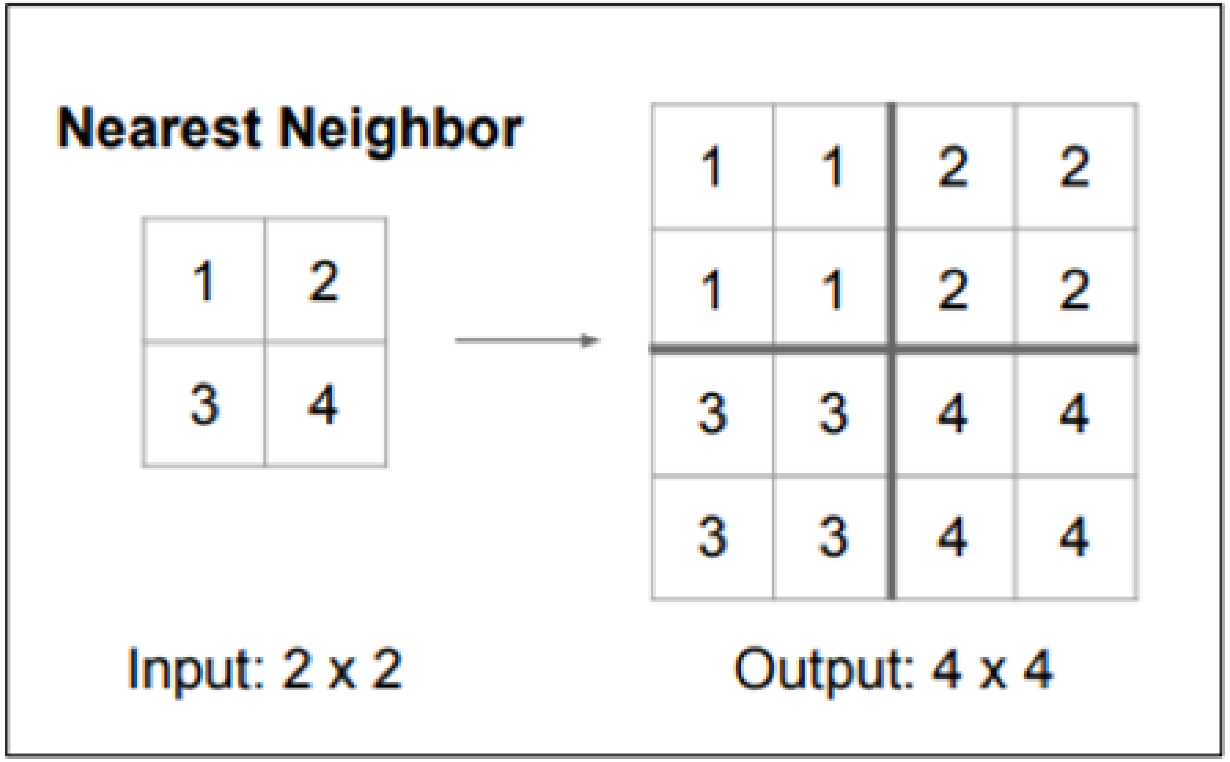
Conv + upsampling

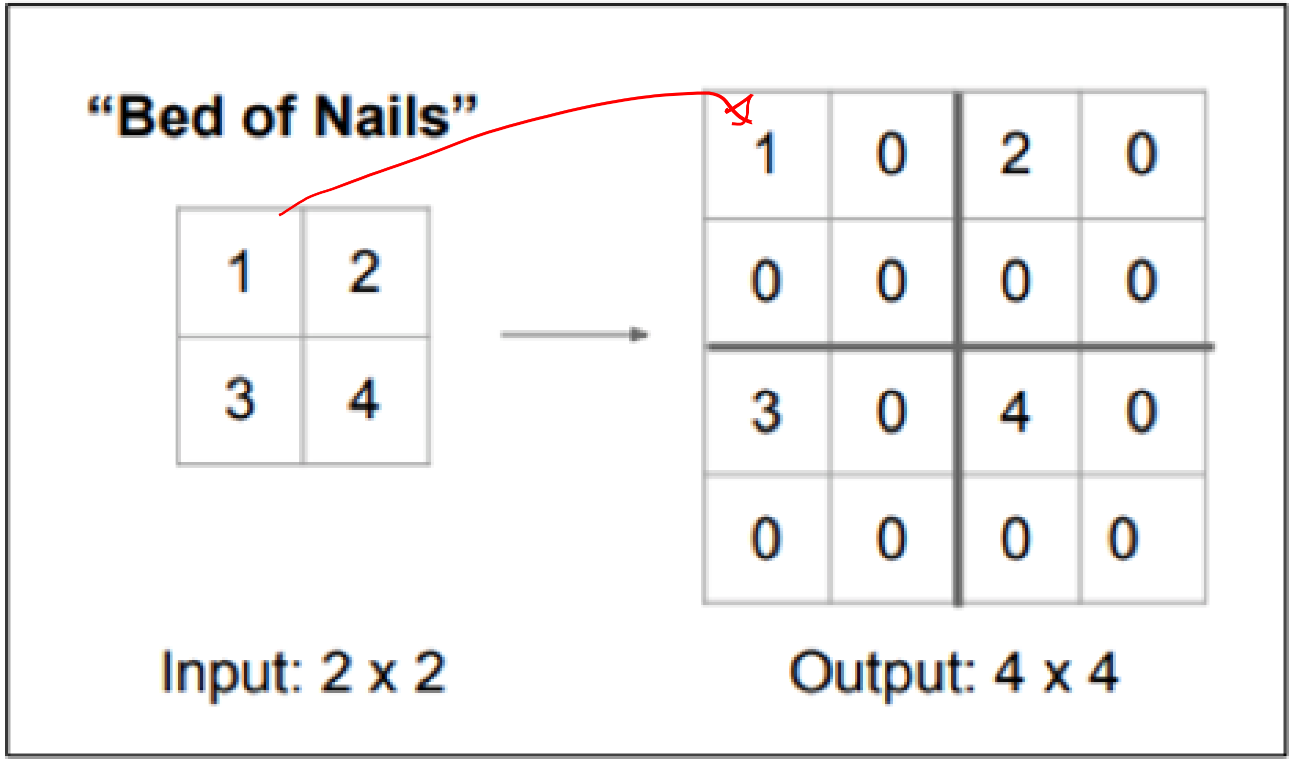


Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

How to perform upsampling?

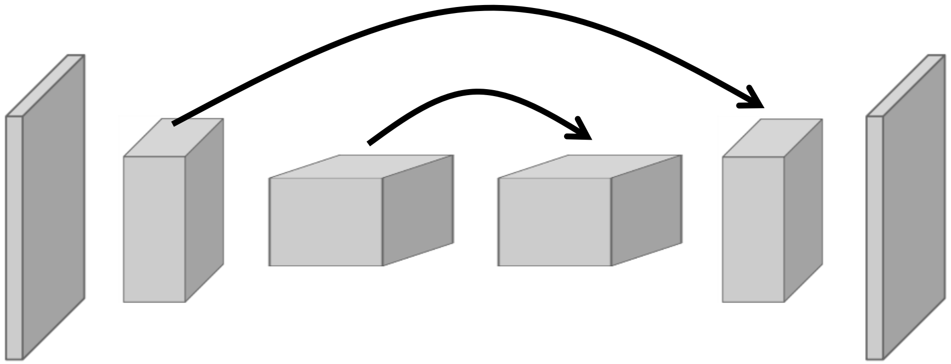


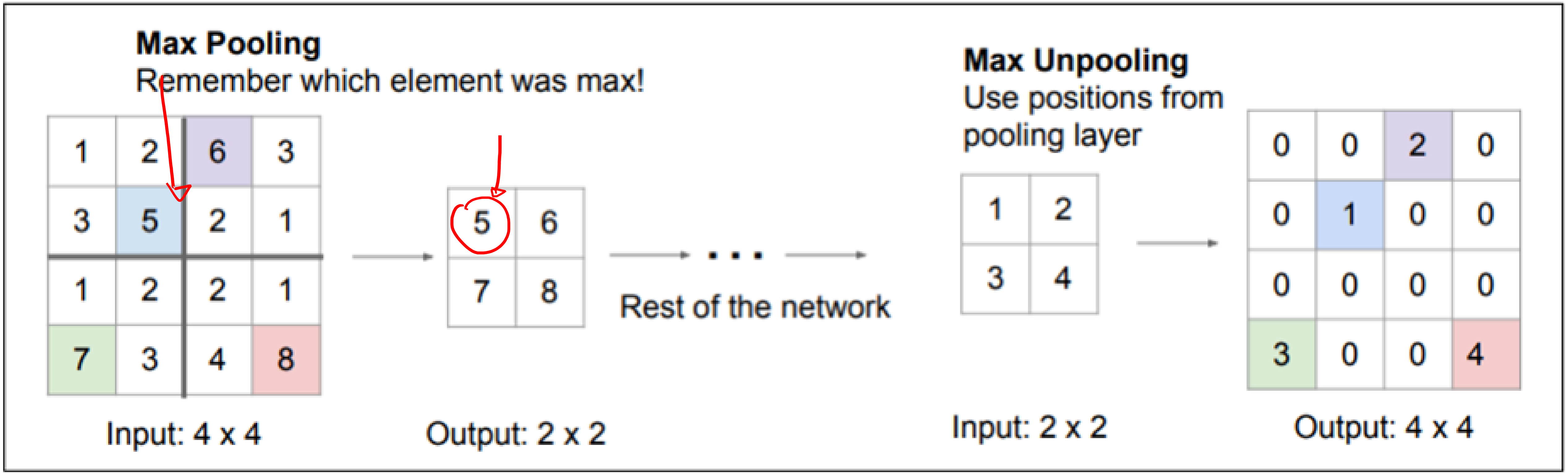




Max Unpooling

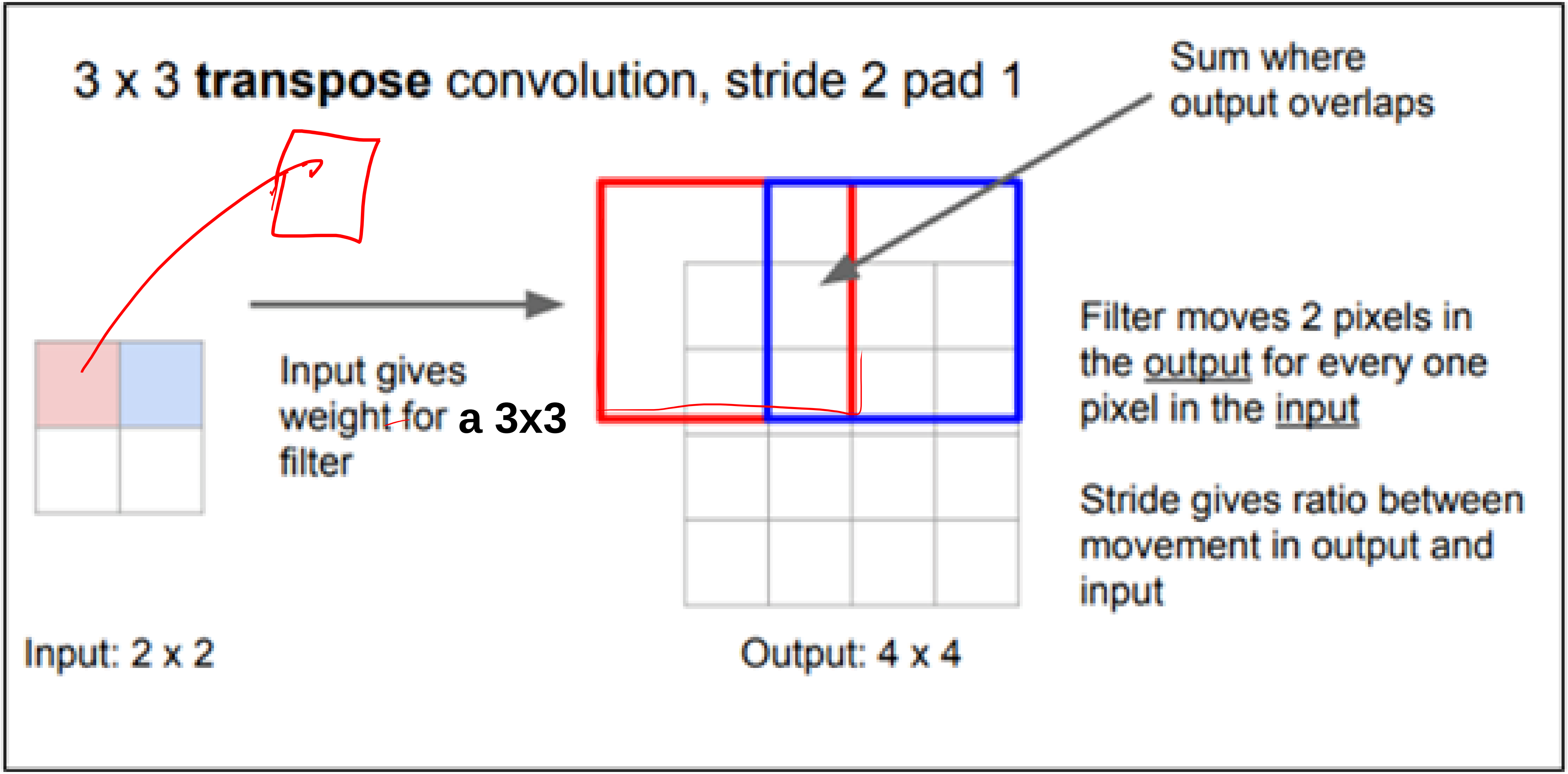
You have to keep track of the locations of the max during maxpooling





[CS231n: Convolutional Neural Networks for Visual Recognition](http://cs231n.github.io/) http://cs231n.github.io/

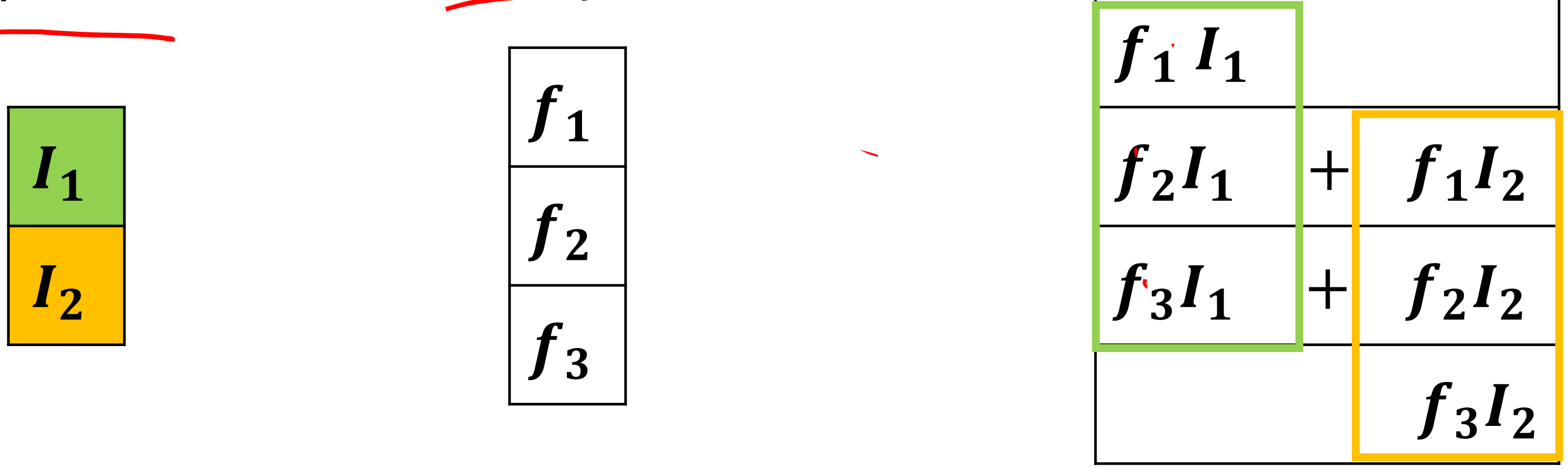
Transpose Convolution



Transpose Convolution

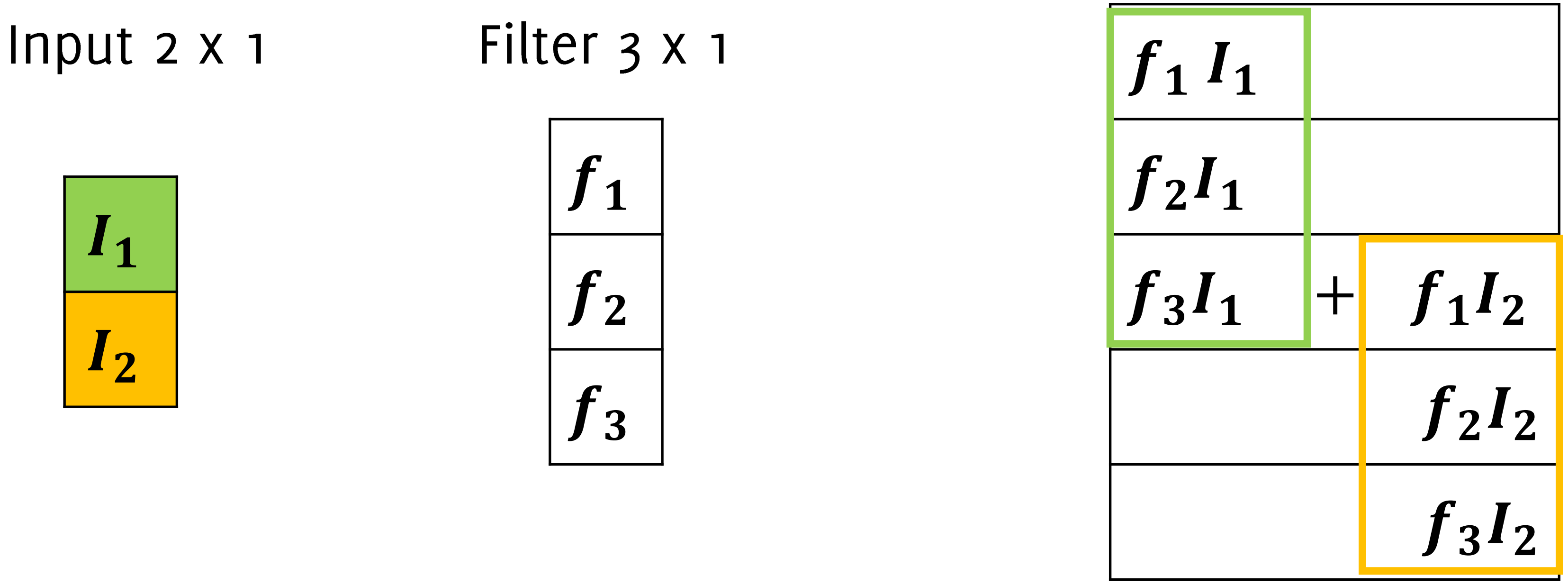
Transpose convolution with stride 1

Input 2 x 1 Filter 3 x 1



Transpose Convolution

Transpose convolution with stride 2

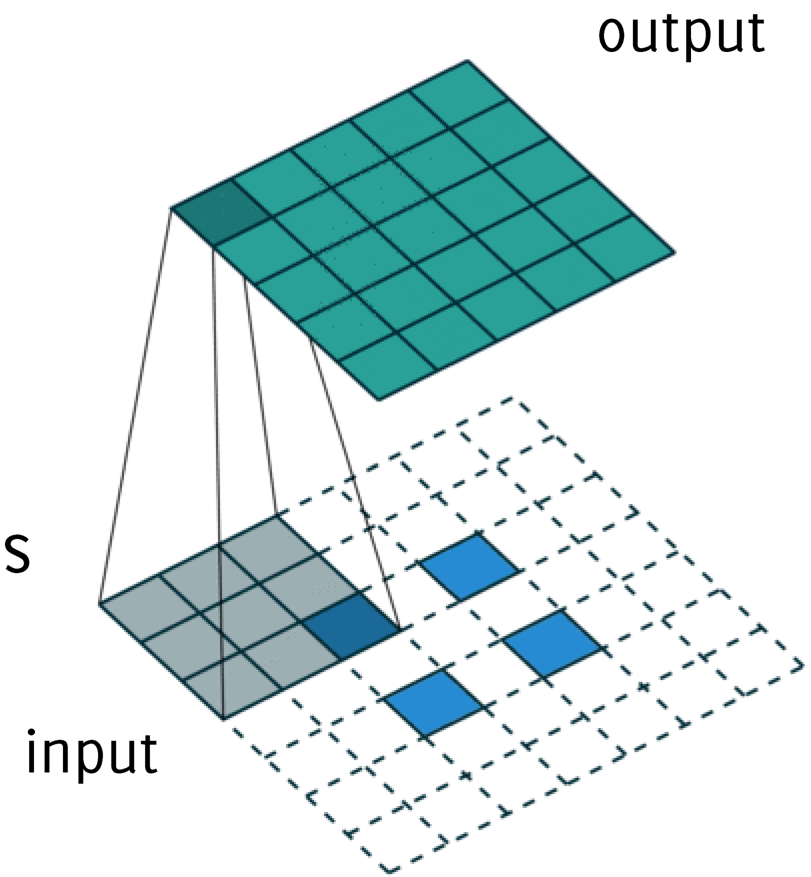


How to perform upsampling?

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
filters can be learned!



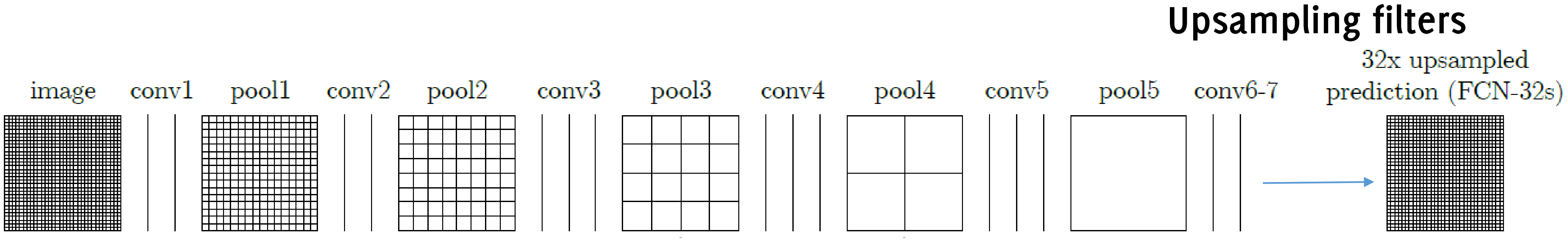
https://github.com/vdumoulin/conv_arithmetic

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.

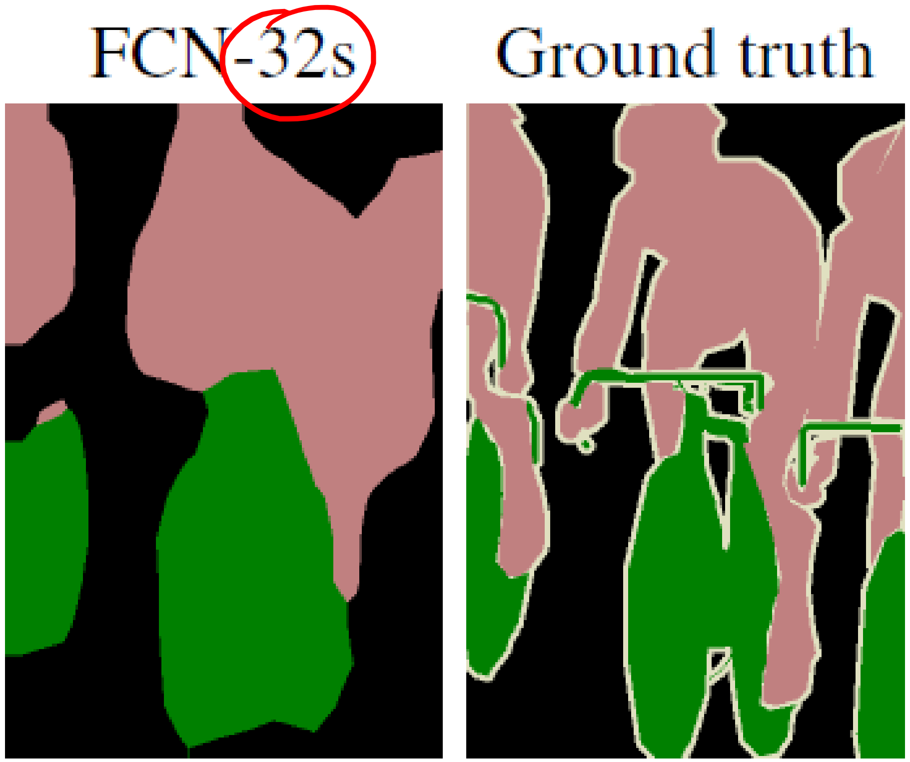


Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." CVPR 2015

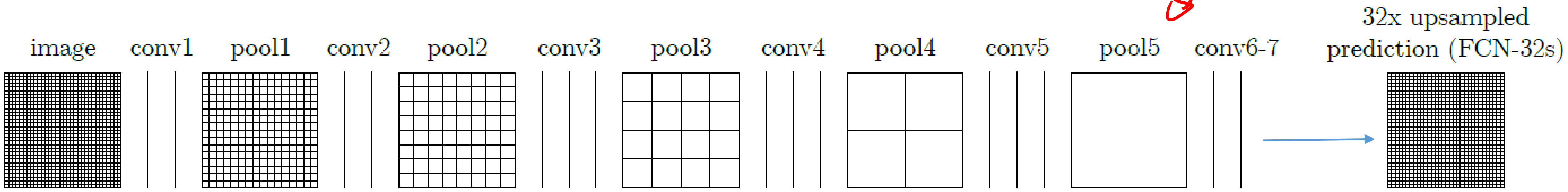
Prediction Upsampling

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

Upsampling filters are learned with initialization equal to the bilinear interpolation

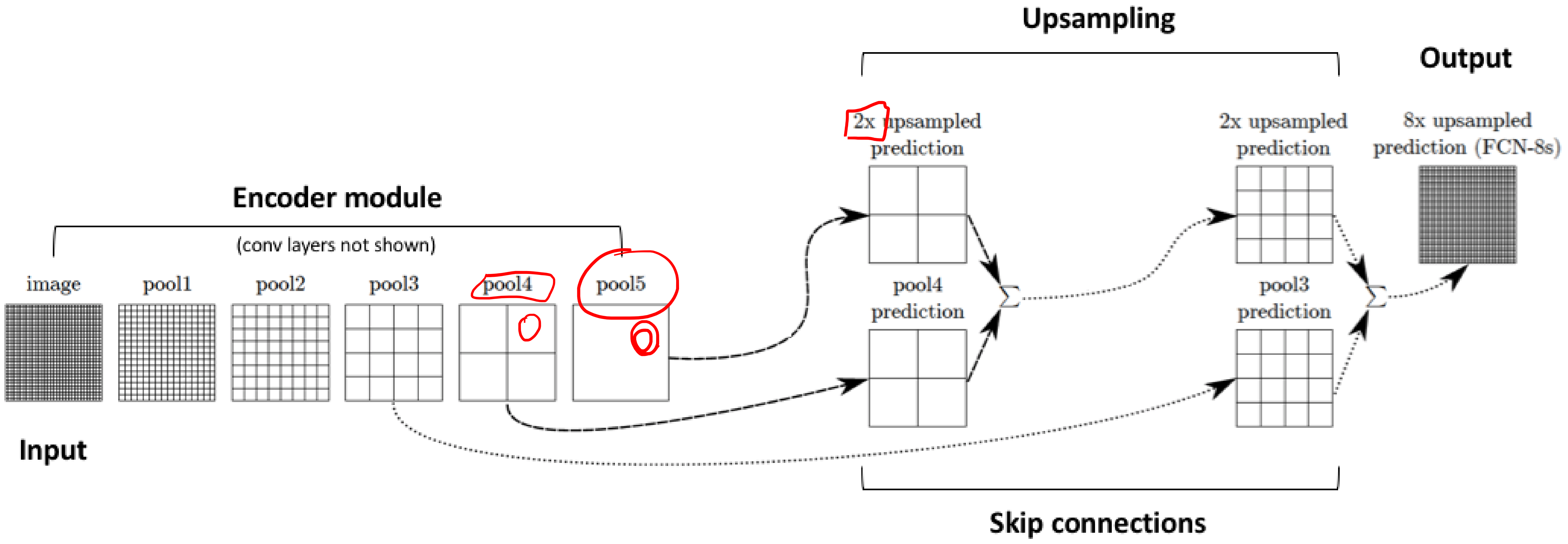


upsampling



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

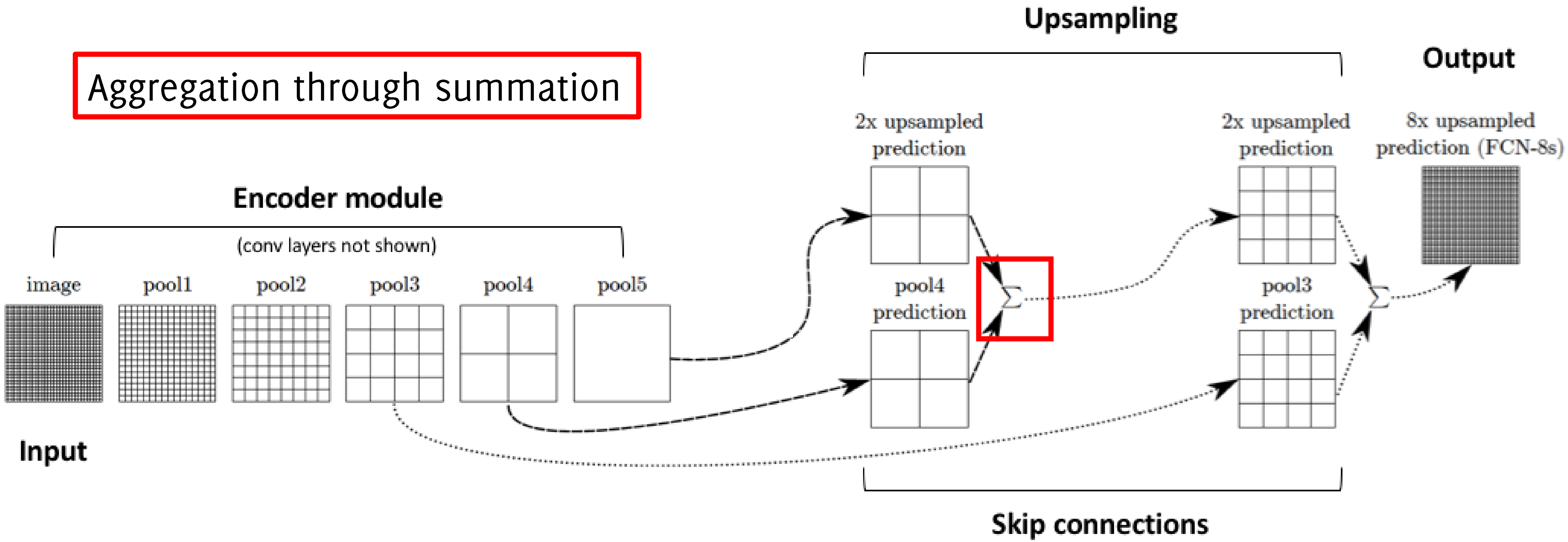
Solution: Skip Connections!



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

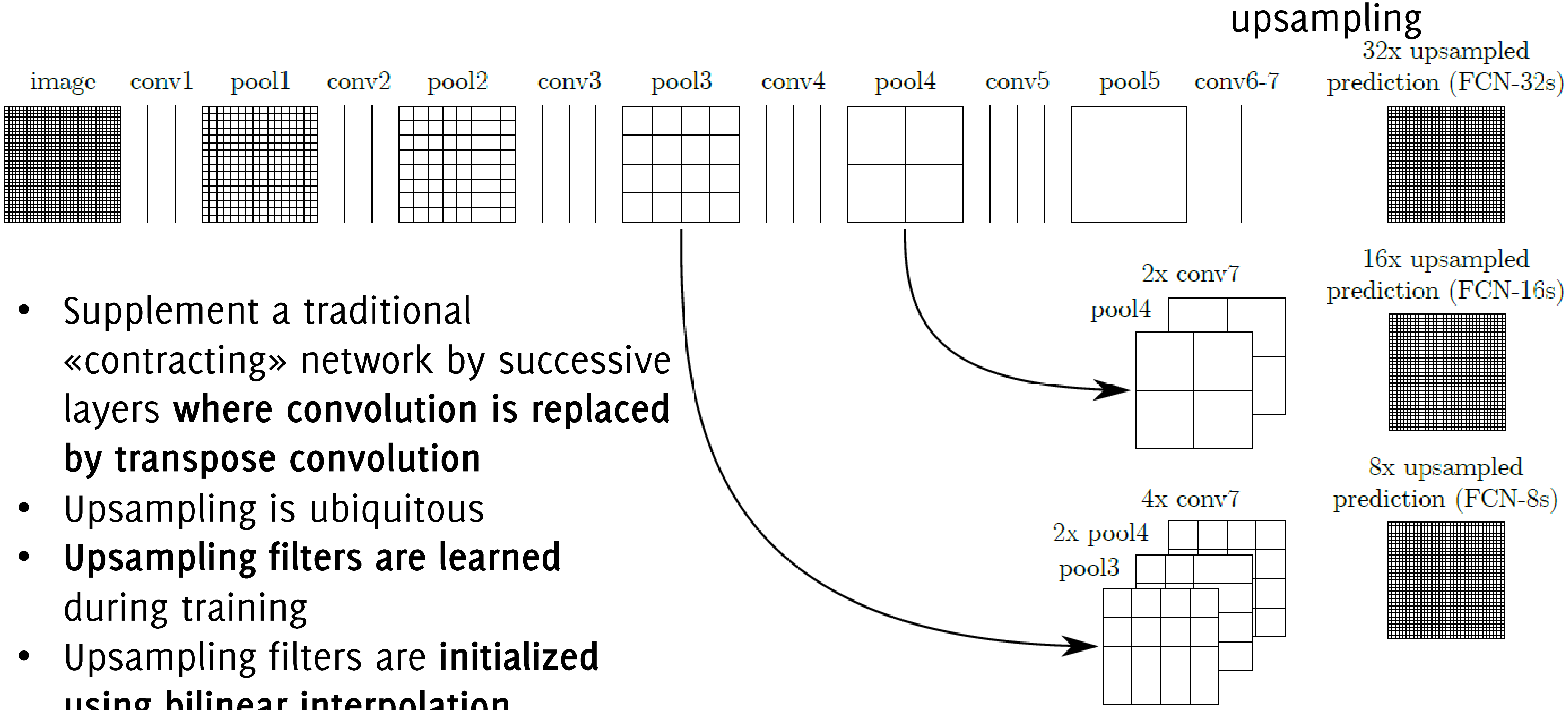
Solution: Skip Connections!

Aggregation through summation



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

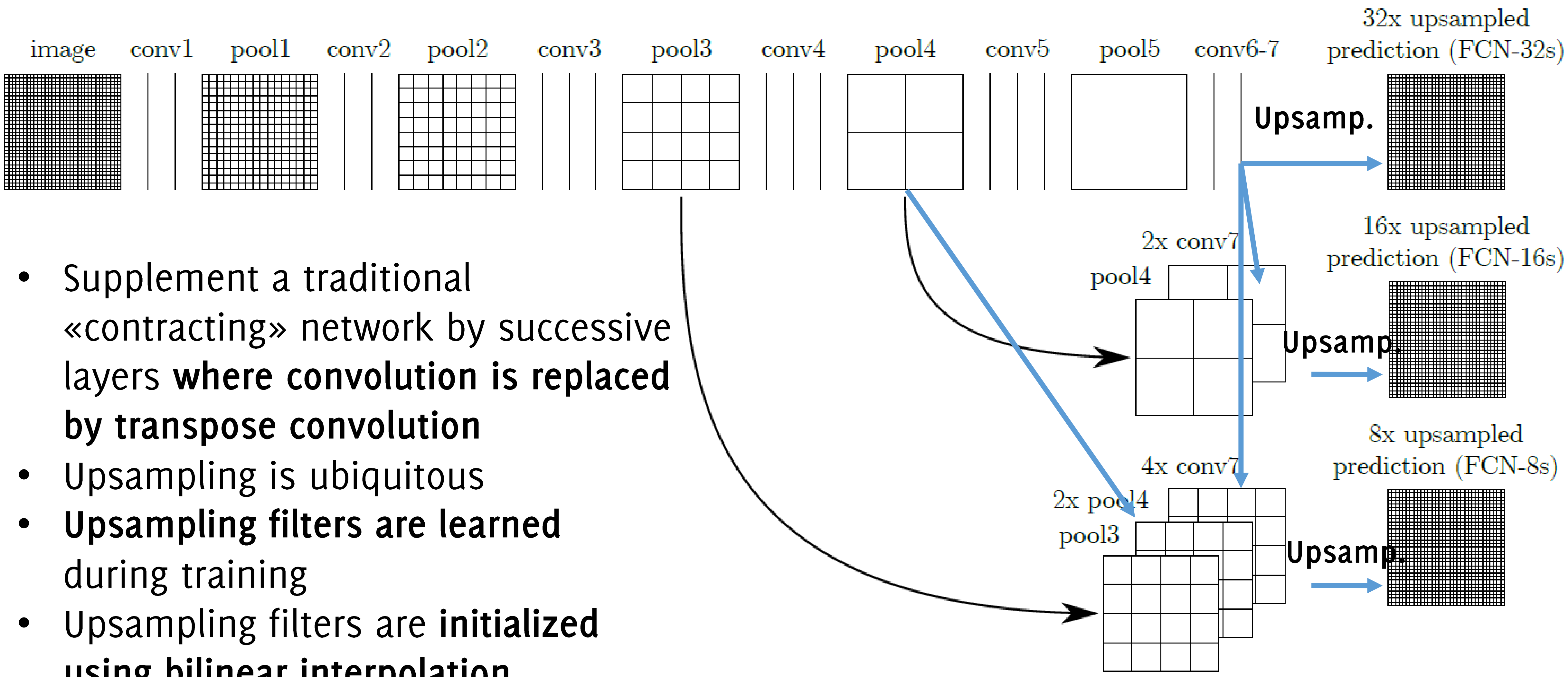
Solution: Skip Connections!



- 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

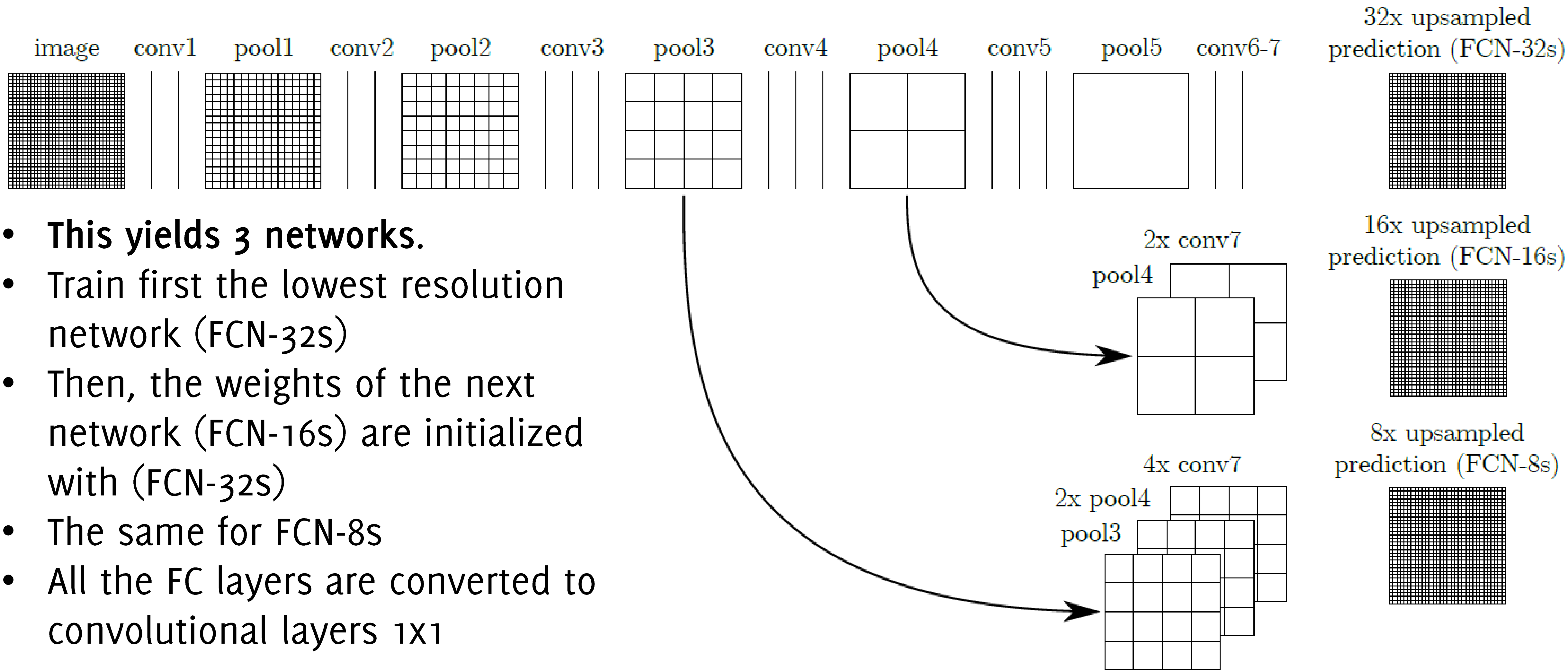
Solution: Skip Connections!



- 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!

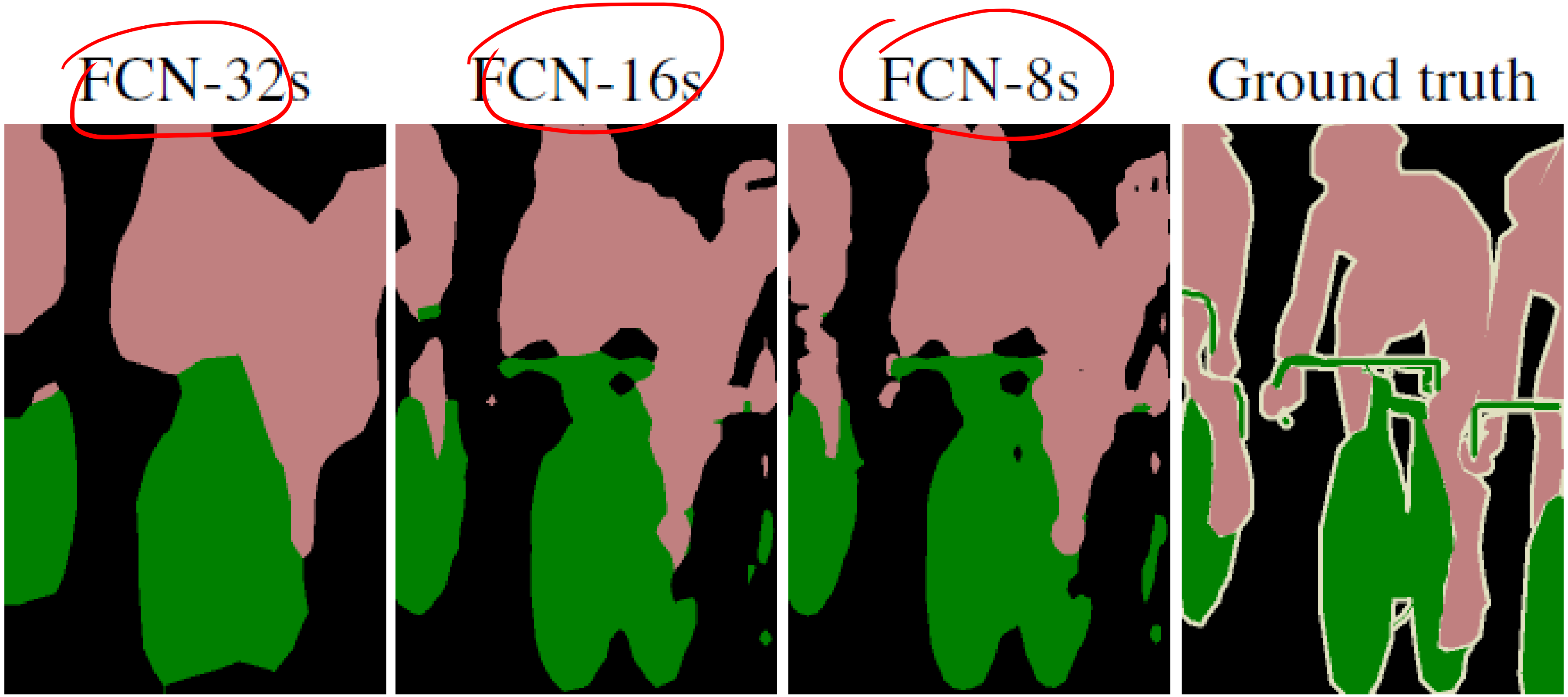


- **This yields 3 networks.**
- Train first the lowest resolution network (FCN-32s)
- Then, the weights of the next network (FCN-16s) are initialized with (FCN-32s)
- The same for FCN-8s
- All the FC layers are converted to convolutional layers 1x1

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
- 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:

$$\text{minimize} \sum_{\boldsymbol{x}_j} \ell(\boldsymbol{x}_j, \theta)$$

Where $\boldsymbol{x}_j$ **are all the pixels in a region of the input image** and the loss is evaluated over the corresponding labels.

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

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

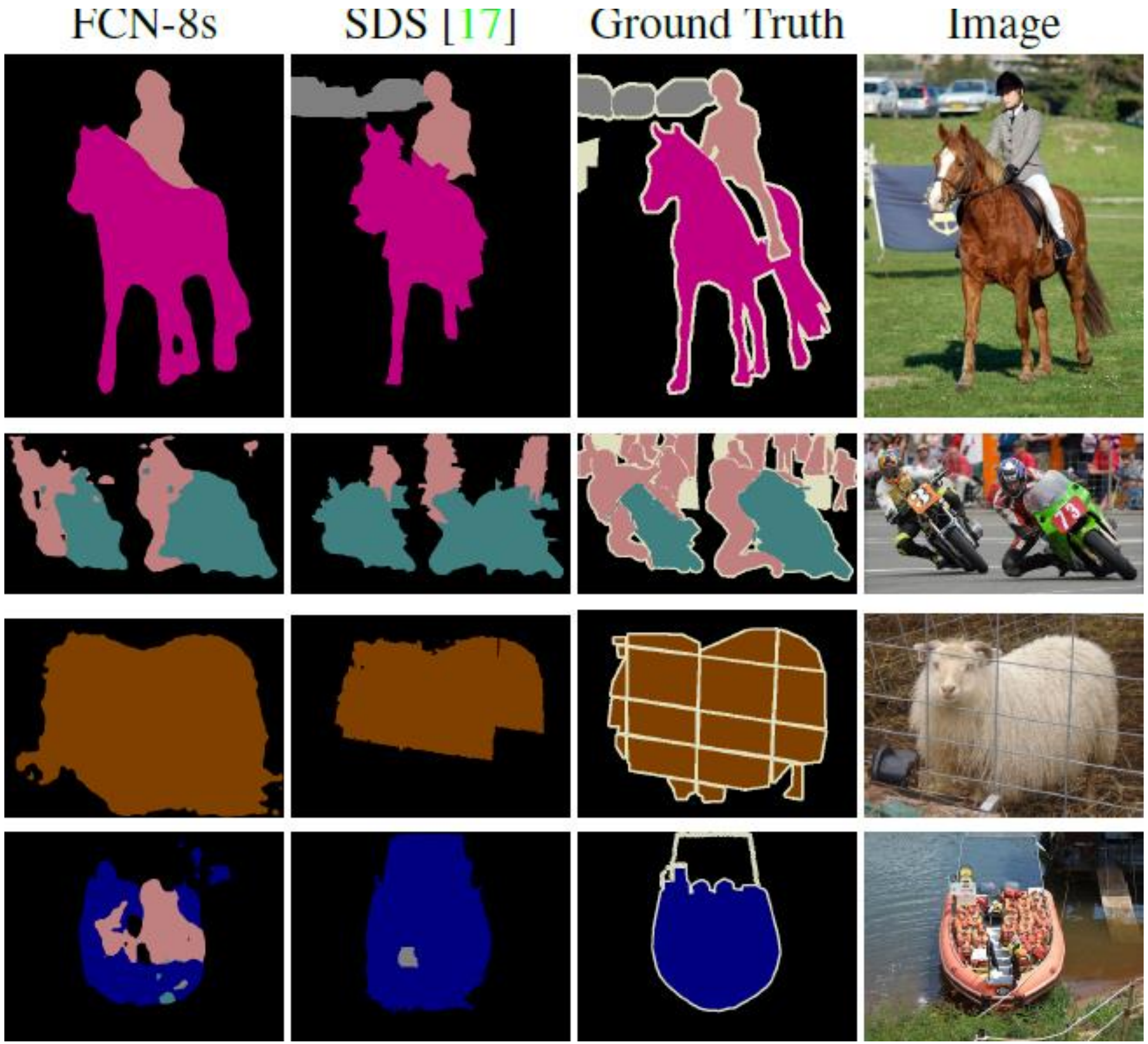
being $M(\boldsymbol{x_j})$ a binary random variable

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

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

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

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/

U-Net

Network formed by:

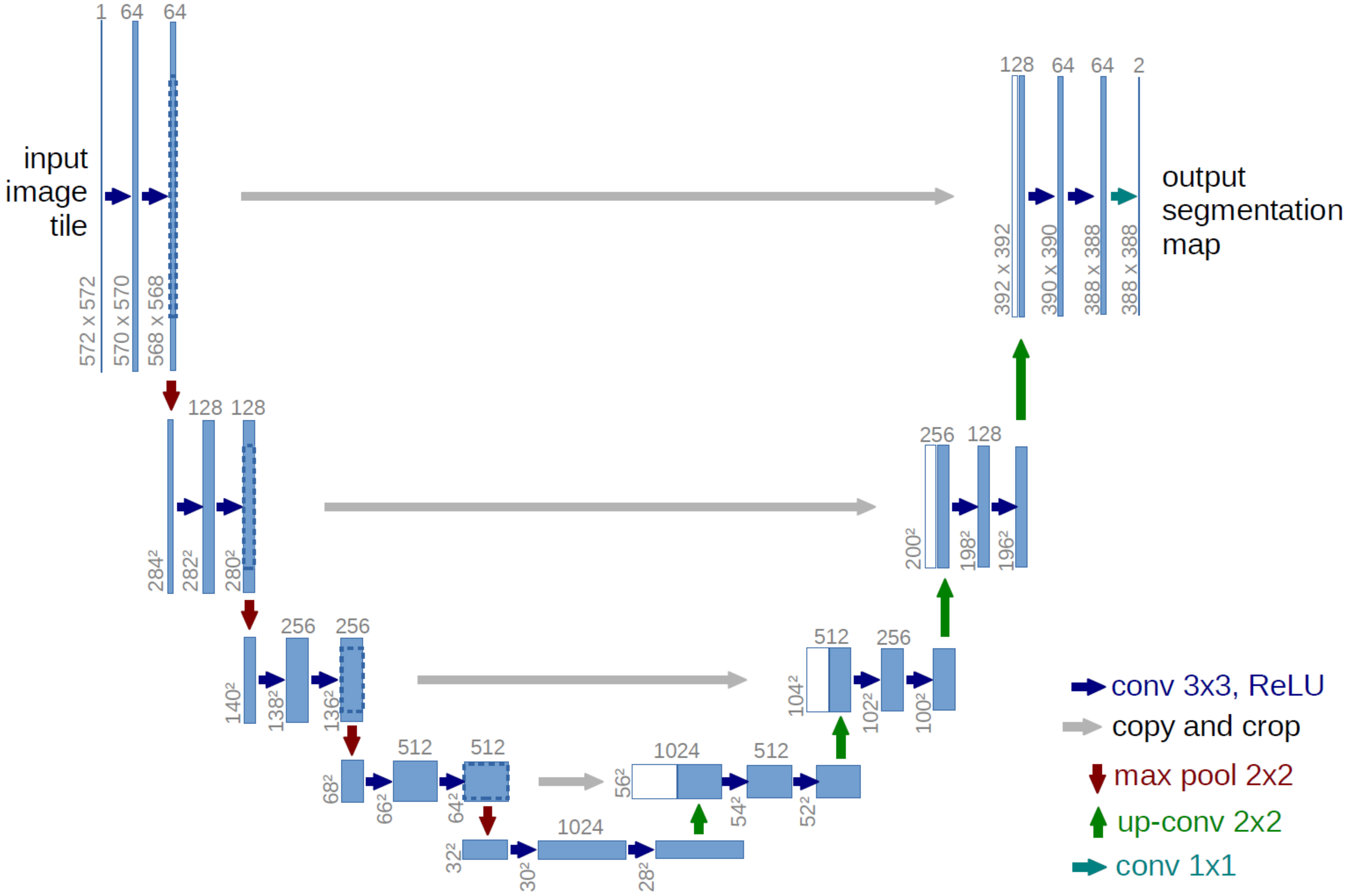
- A contracting path
- An expansive path

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.

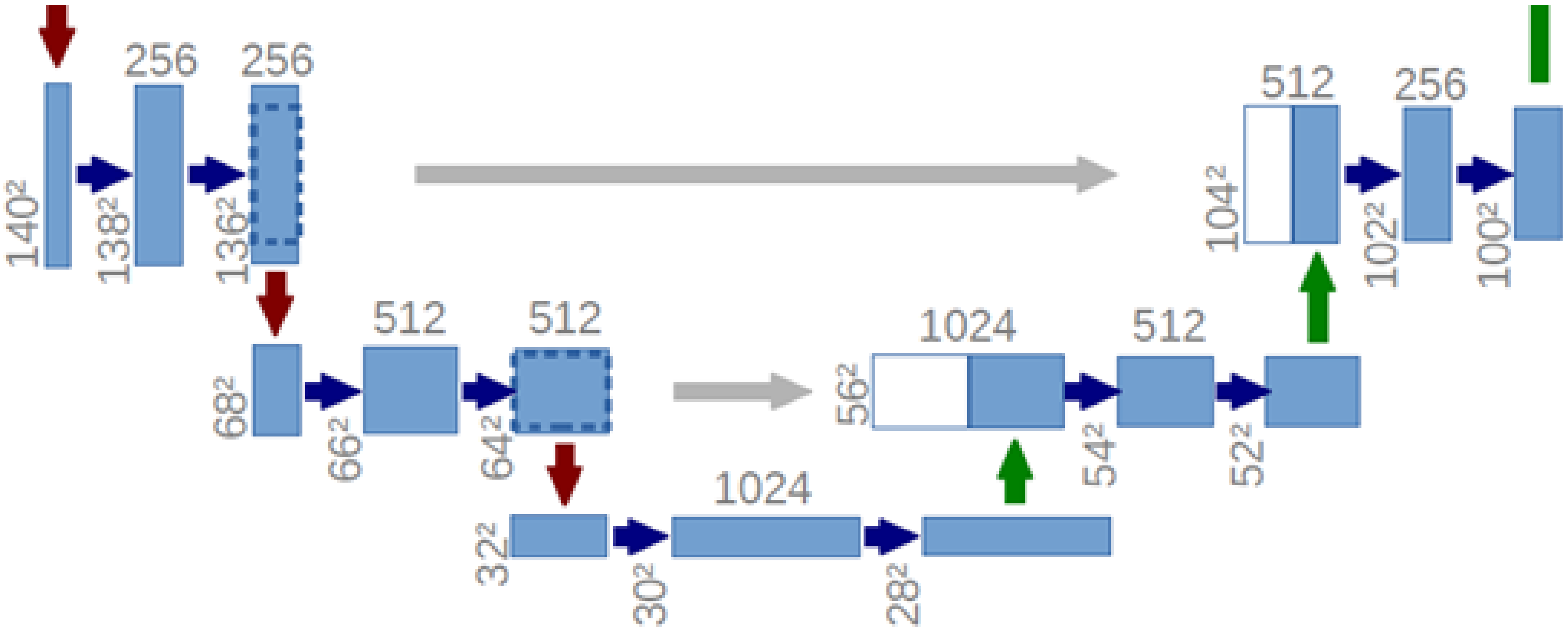
U-Net: Contracting path

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



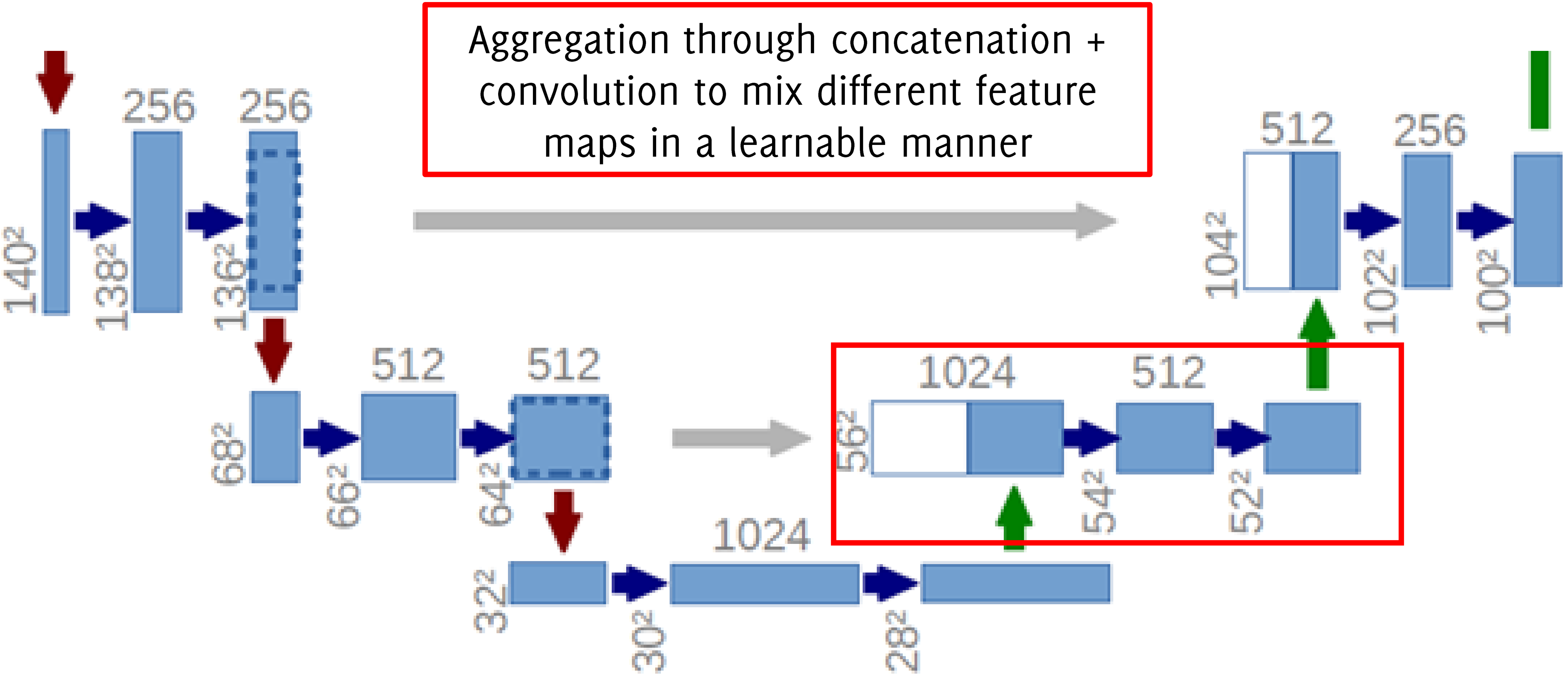
Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *MICCAI*, 2015.

U-Net: Skip connections

Aggregation through concatenation

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

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

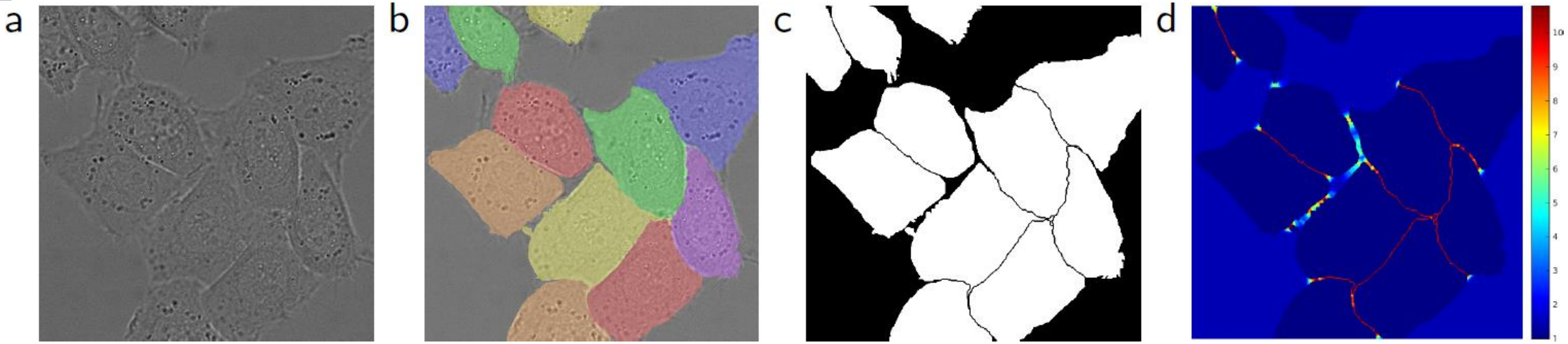
- w_c is used to balance class proportions (remember no patch resampling in full-image training)
- d_1 is the distance to the border of the closest cell
- d_2 is the distance to the border of the second closest cell

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



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

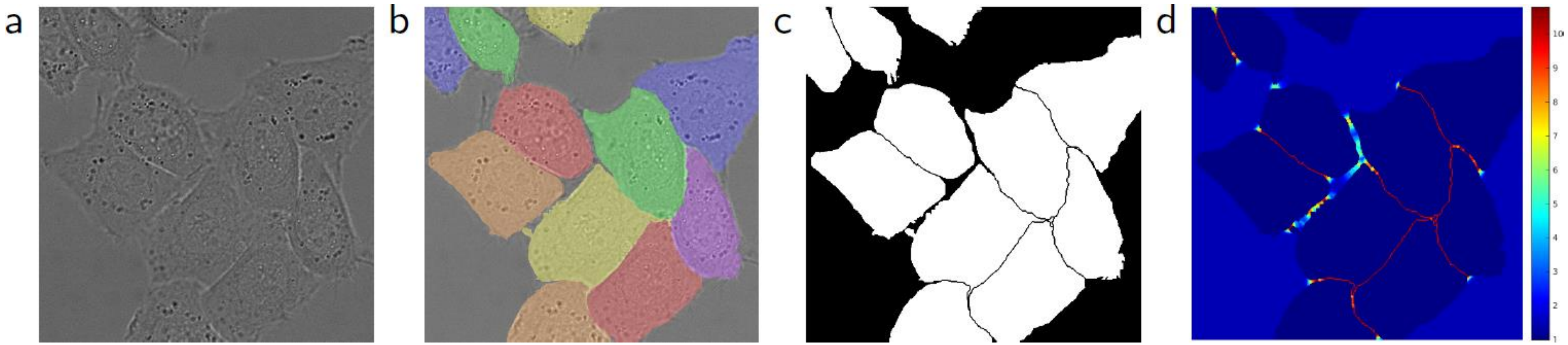
U-net: Training

Full-image training by a weighted loss function

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

Takes into account class unbalance in the training set

Enhances classification performance at borders of different objects



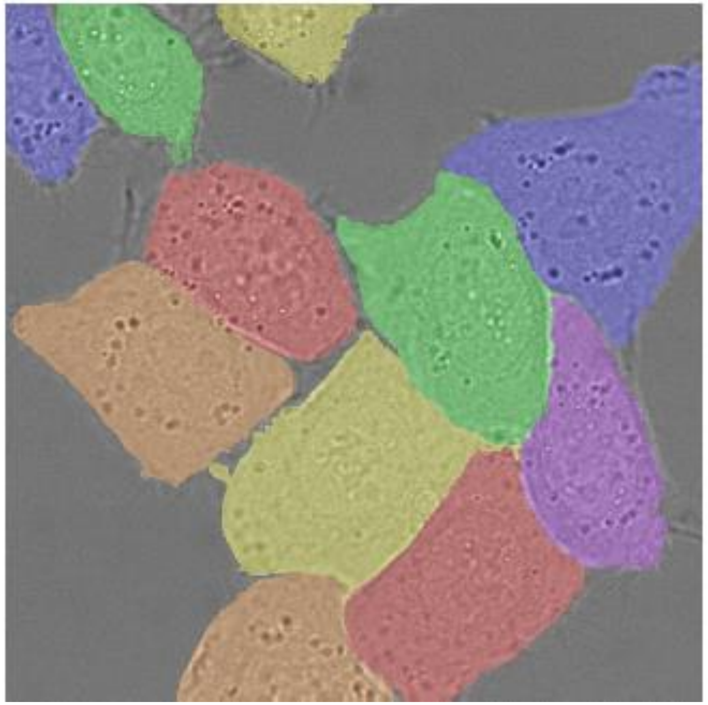
Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *MICCAI*, 2015.

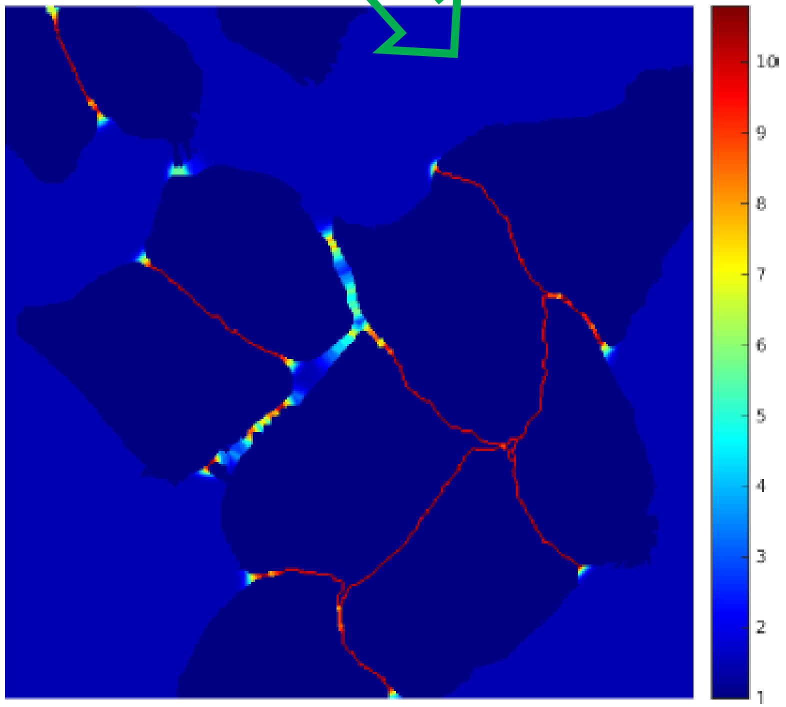
U-net: Training

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

Full-image training by a weighted loss function

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





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

Global Averaging Pooling

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

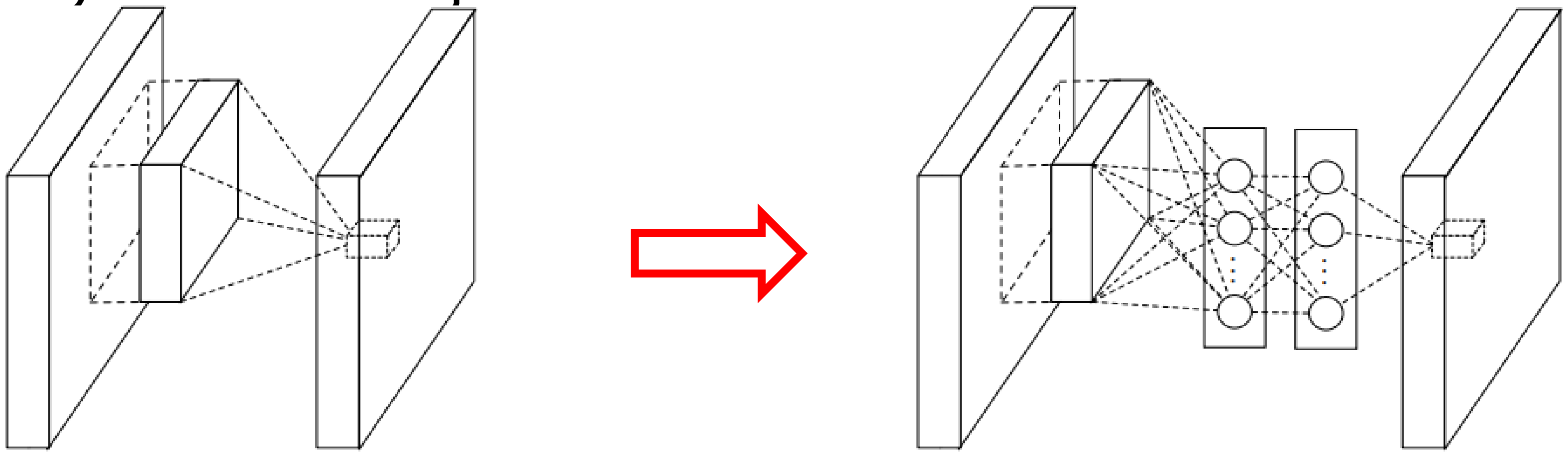
Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

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



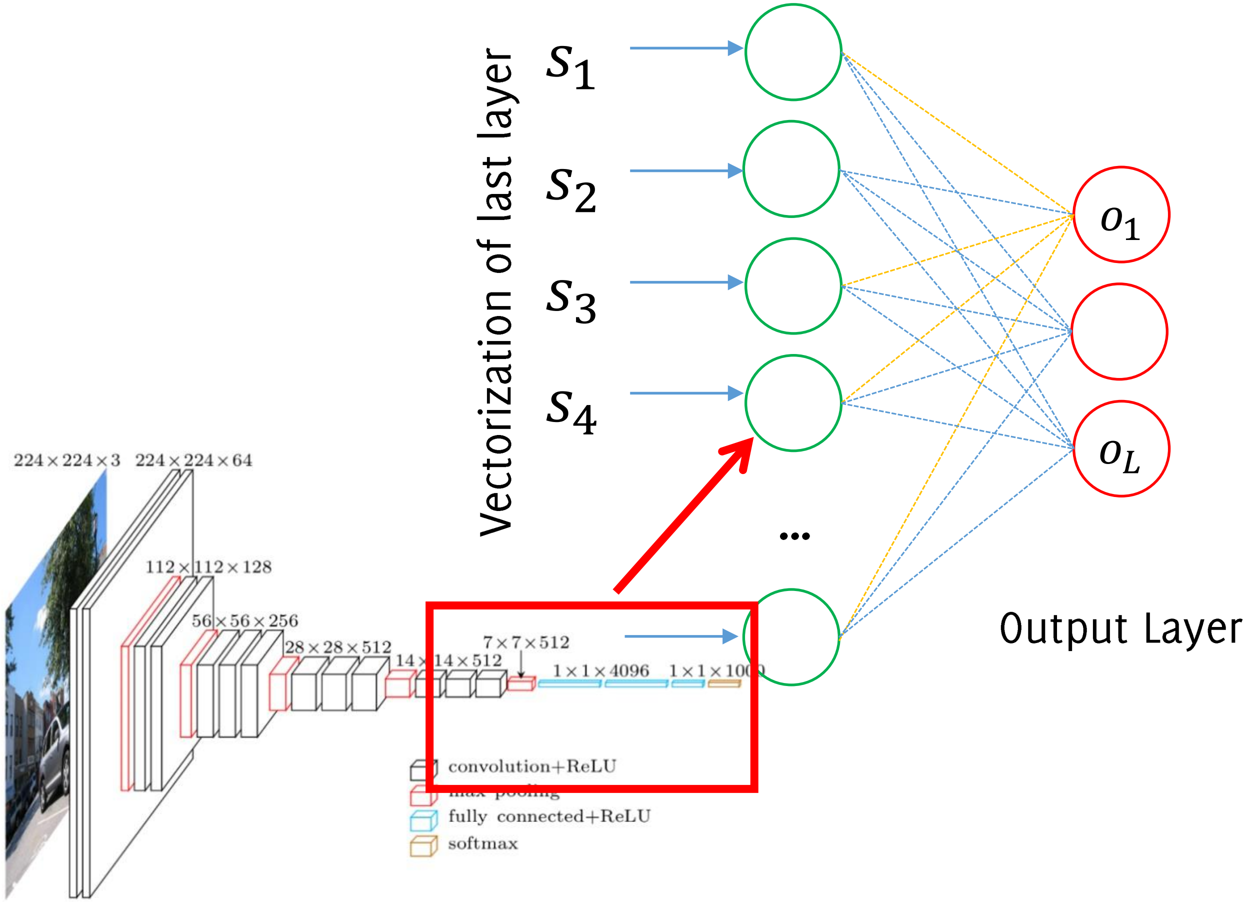
(a) Linear convolution layer (b) Mlpconv layer

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

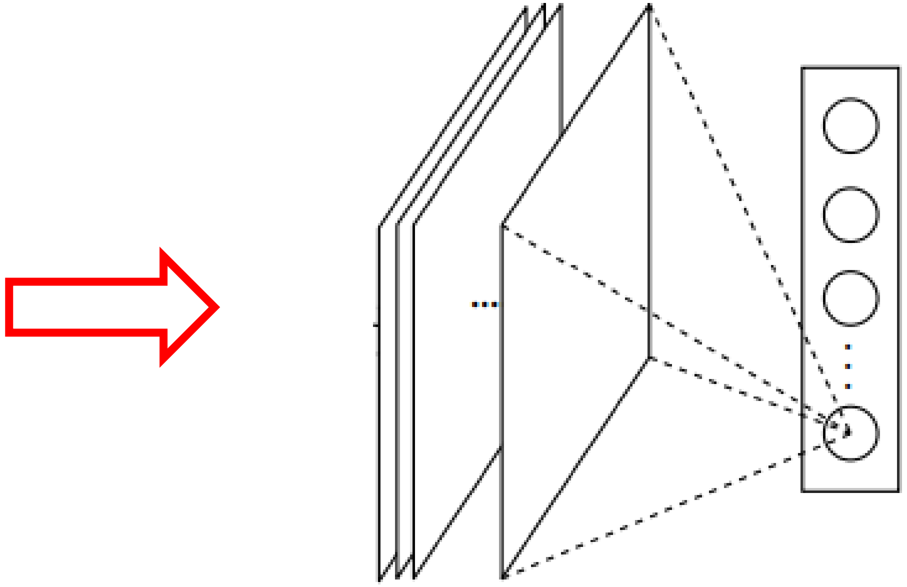
Network in Network

They introduce Global Averaging Pooling Layers

Fully Connected Layer



Global Averaging Pooling Layer

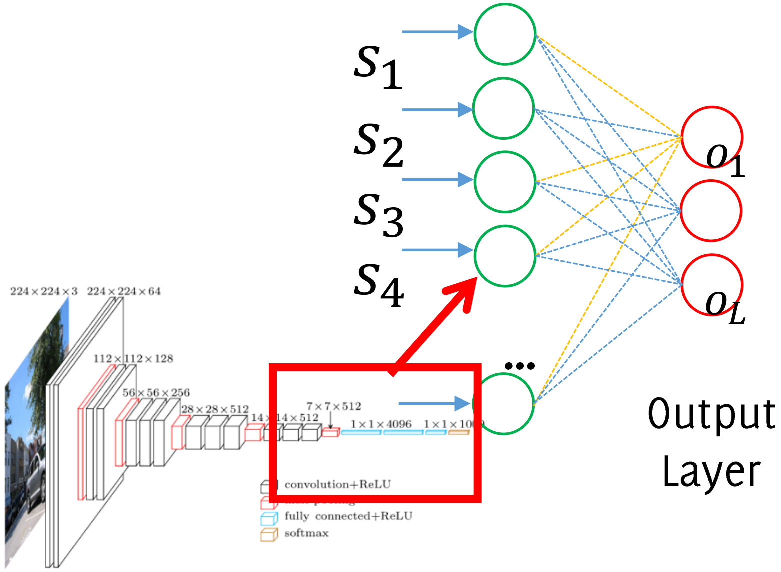


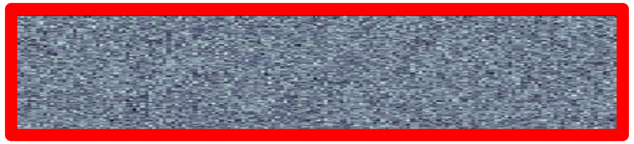
Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

Network in Network: GAP

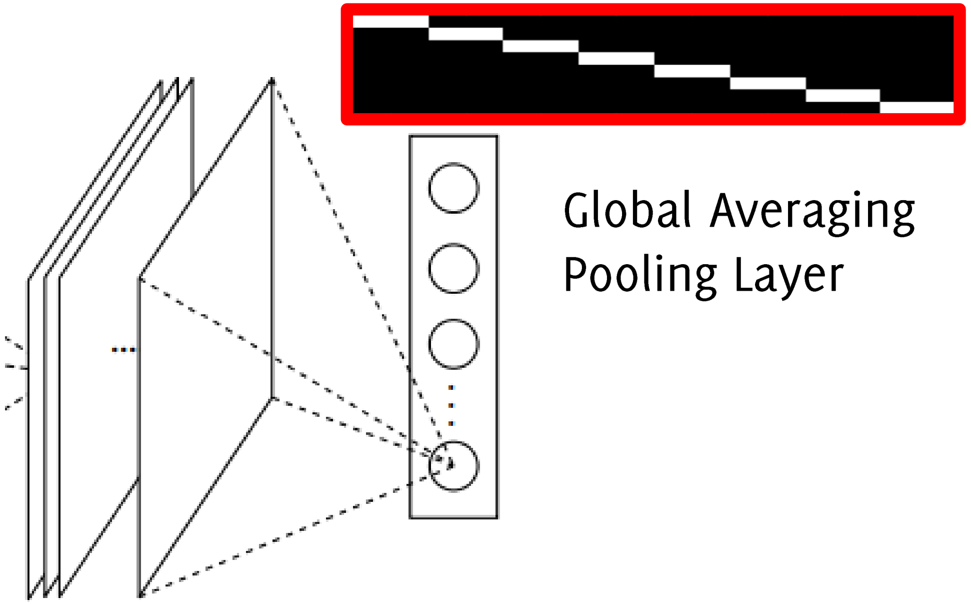
Global Averaging Pooling Layers: instead of a FC layer at the end of the network, compute the average of each feature map.

- 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



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 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

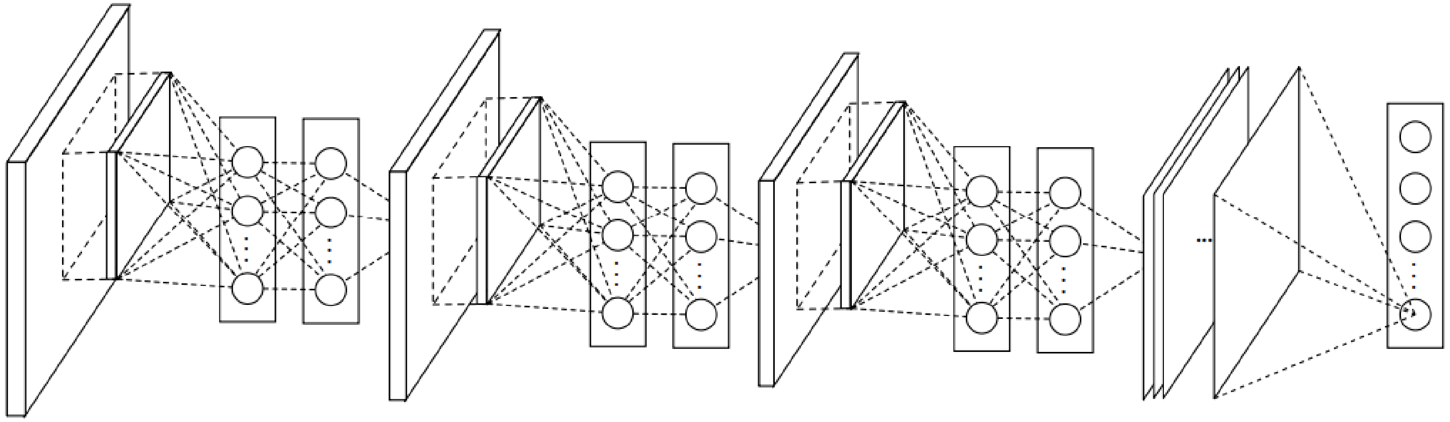
The whole NiN stacks

- mlpconv layers (RELU) + dropout
- Maxpooling
- Global Averaging Pooling (GAP) layer
- Softmax

A few layers of these

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

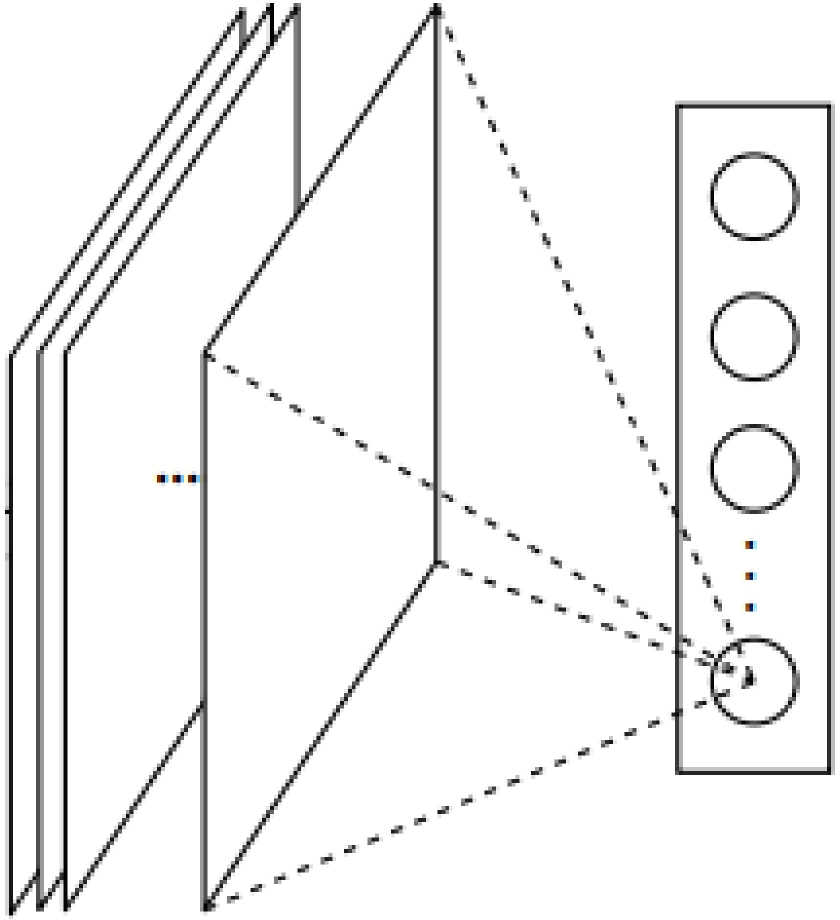


Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *ICLR 2014*

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

Giacomo Boracchi

Weakly supervised localization

Perform localization over an image without images with annotated bounding box

- Training set provided as for classification with pairs $(\boldsymbol{x}, y)$ where only the image label 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

Giacomo 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

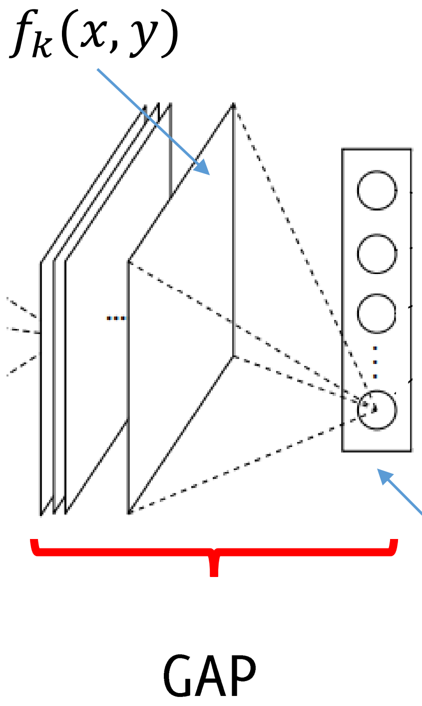
Brushing teeth

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.

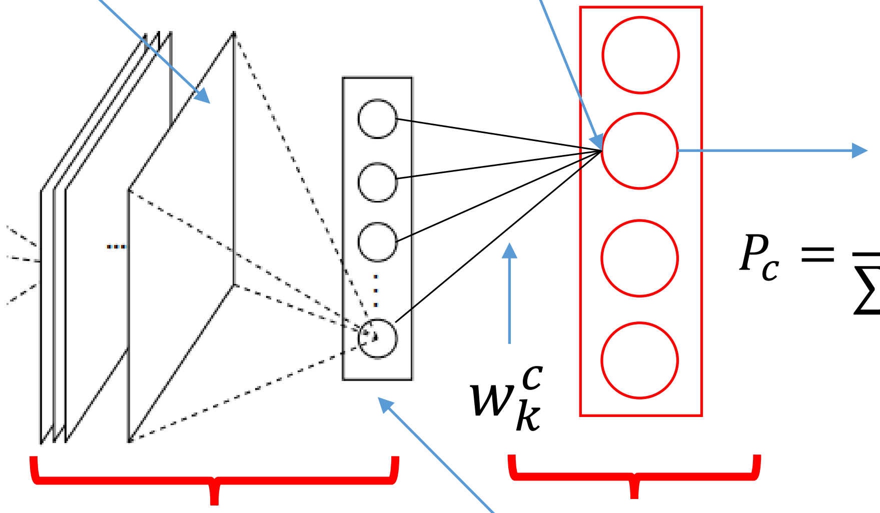
The FC computes S_c for each class c as the weighted sum of $\{F_k\}$, where weights are defined during training

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

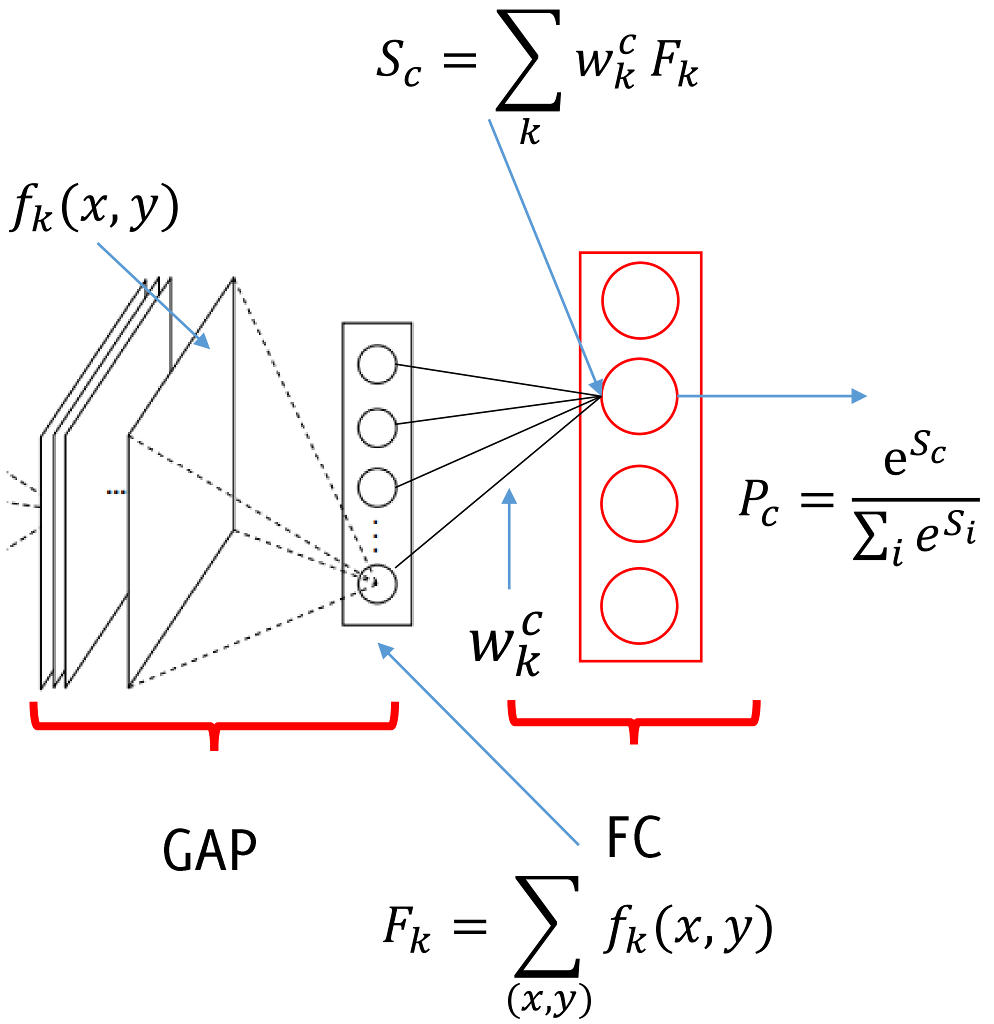
$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \boxed{\sum_k w_k^c f_k(x,y)}$$

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

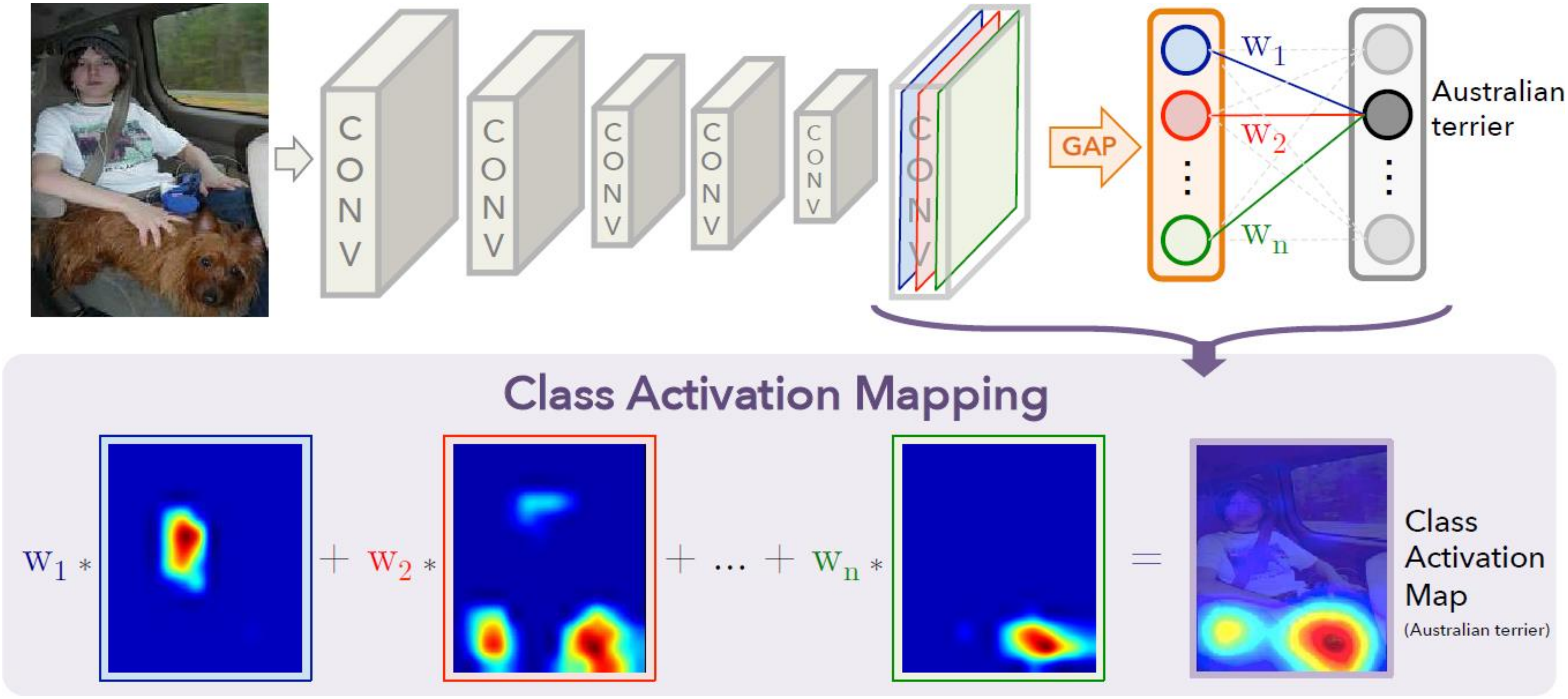
Thanks to the softmax, the depth of the last convolutional volume can be different from the number of classes



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

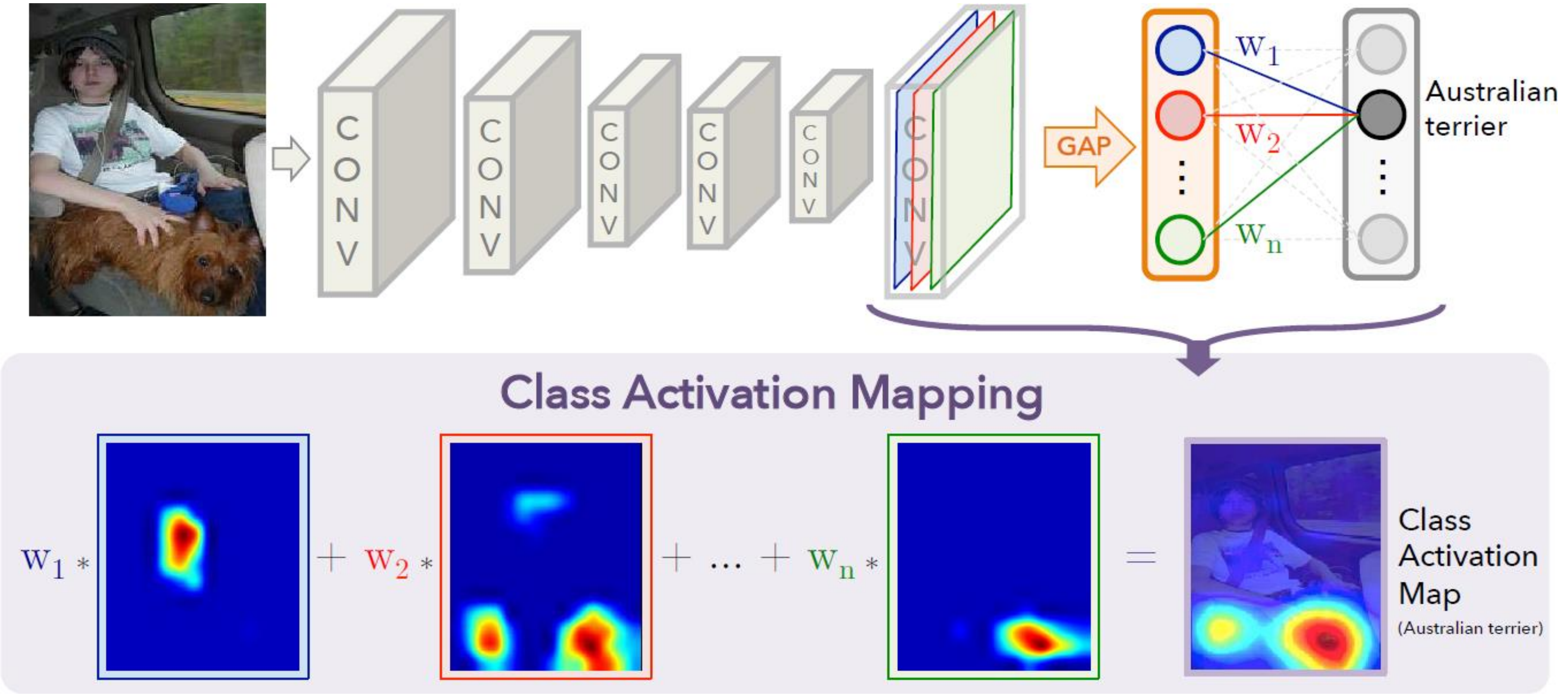
Class Activation Mapping

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



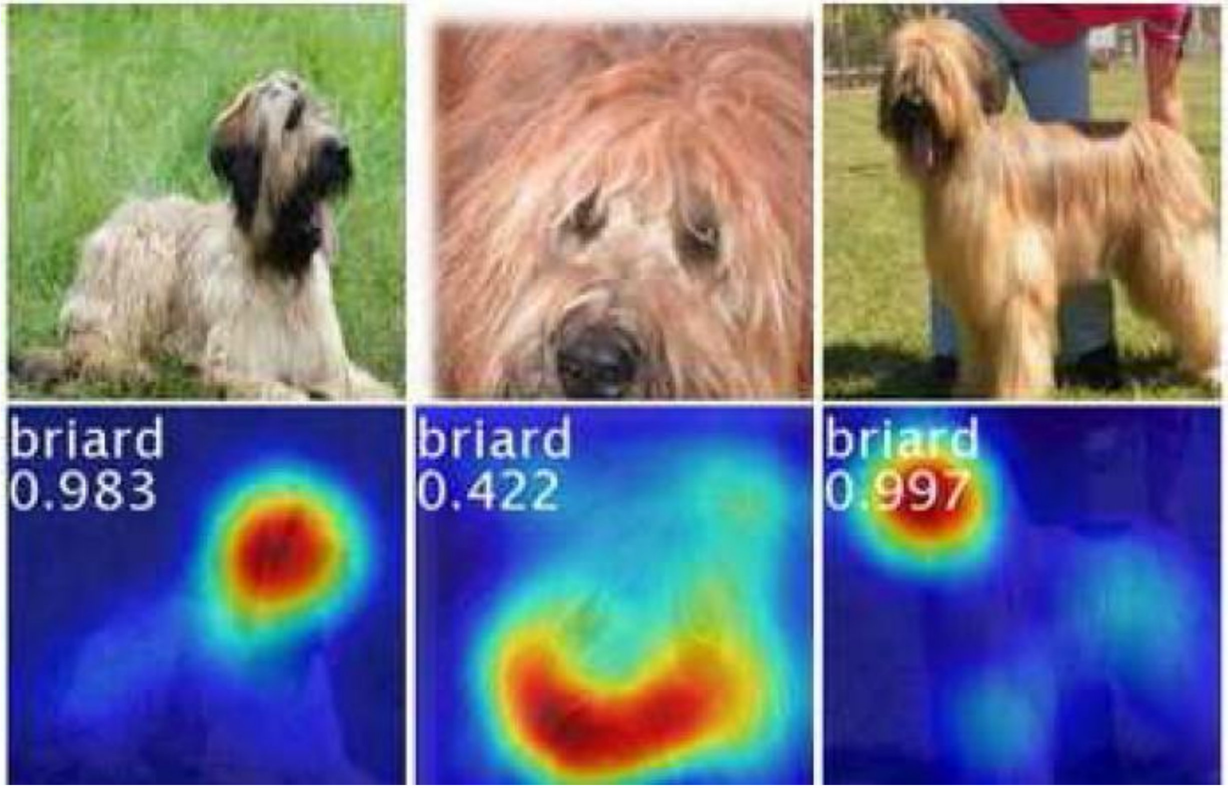
Zhou, Bolei, et al. "Learning deep features for discriminative localization." CVPR 2016.

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.

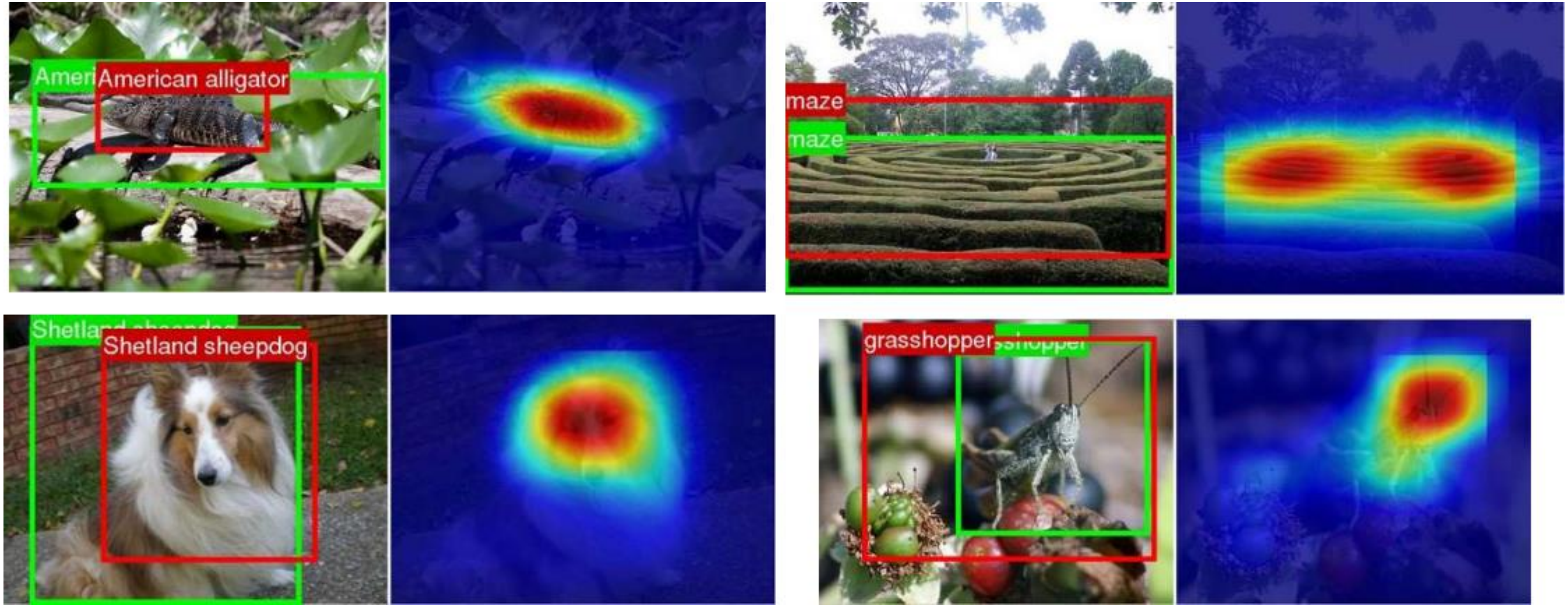
Remarks

- 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.

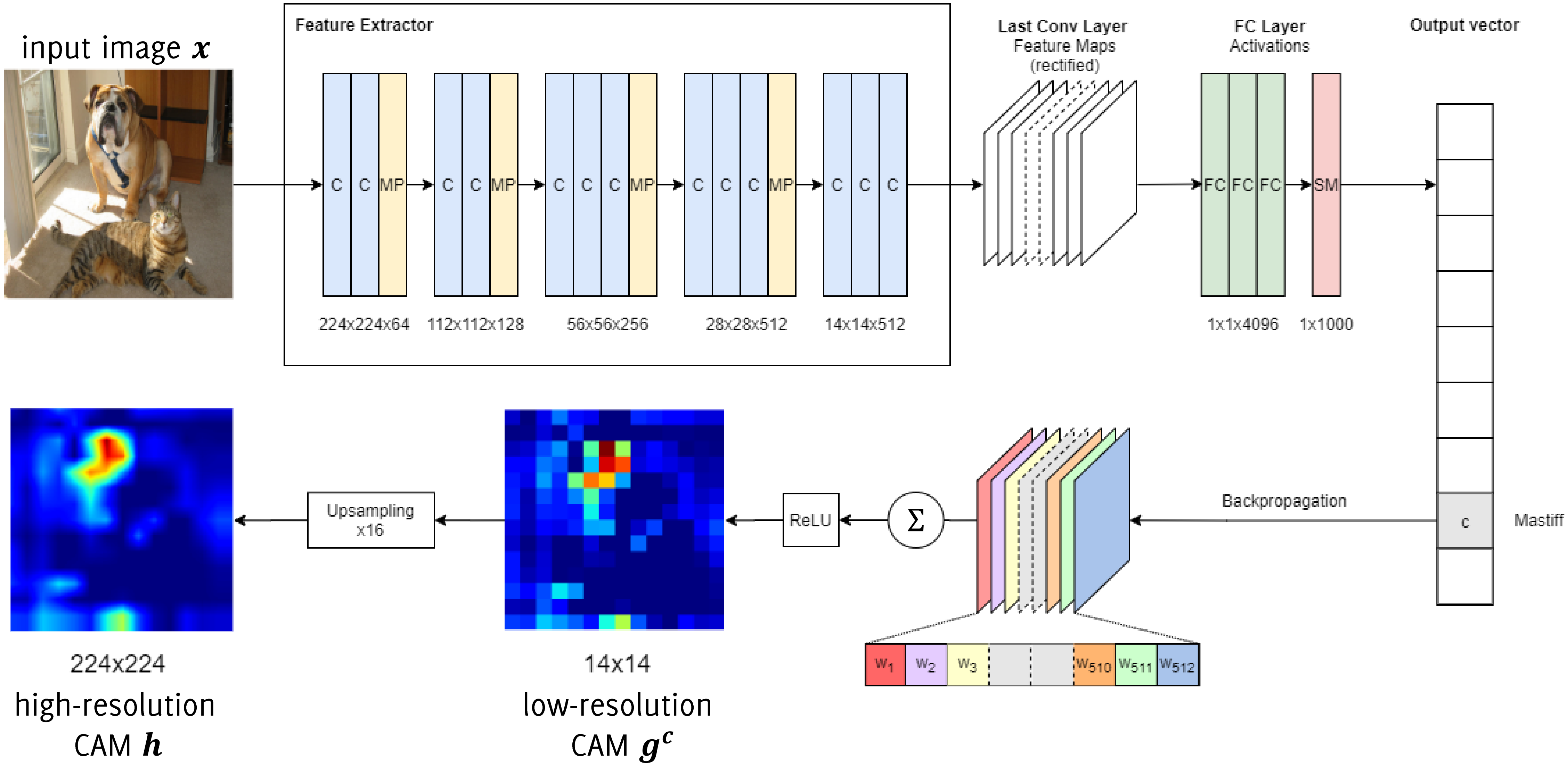
Weakly Supervised Localization

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



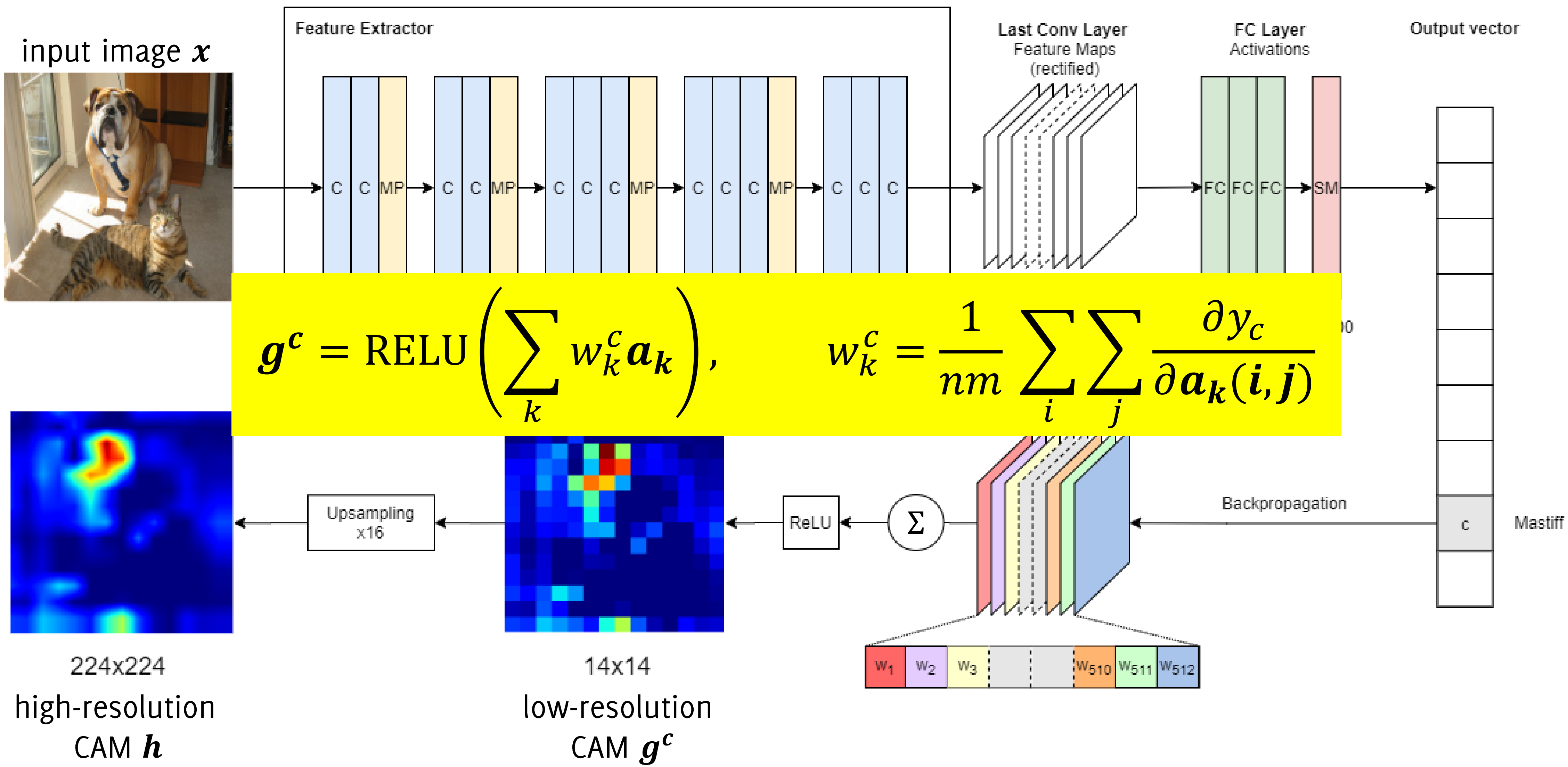
Zhou, Bolei, et al. "Learning deep features for discriminative localization.“ CVPR 2016

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*

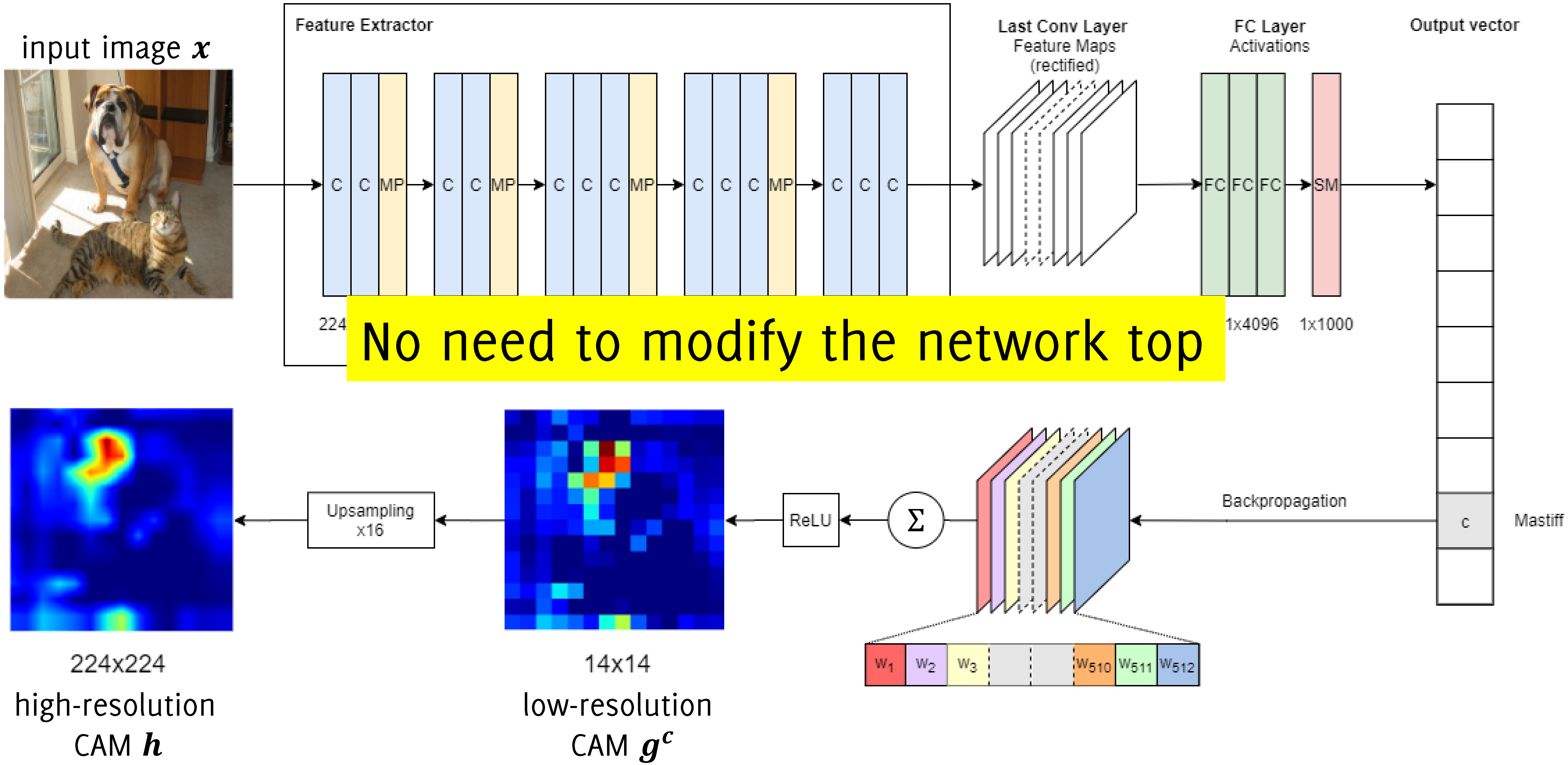
Grad-CAM and CAM-based techniques



$$\boldsymbol{g}^c = \mathrm{RELU}\left(\sum_k w_k^c \boldsymbol{a}_k\right), \qquad w_k^c = \frac{1}{nm}\sum_i \sum_j \frac{\partial y_c}{\partial \boldsymbol{a}_k(\boldsymbol{i},\boldsymbol{j})}$$

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)

first layers depth last layers

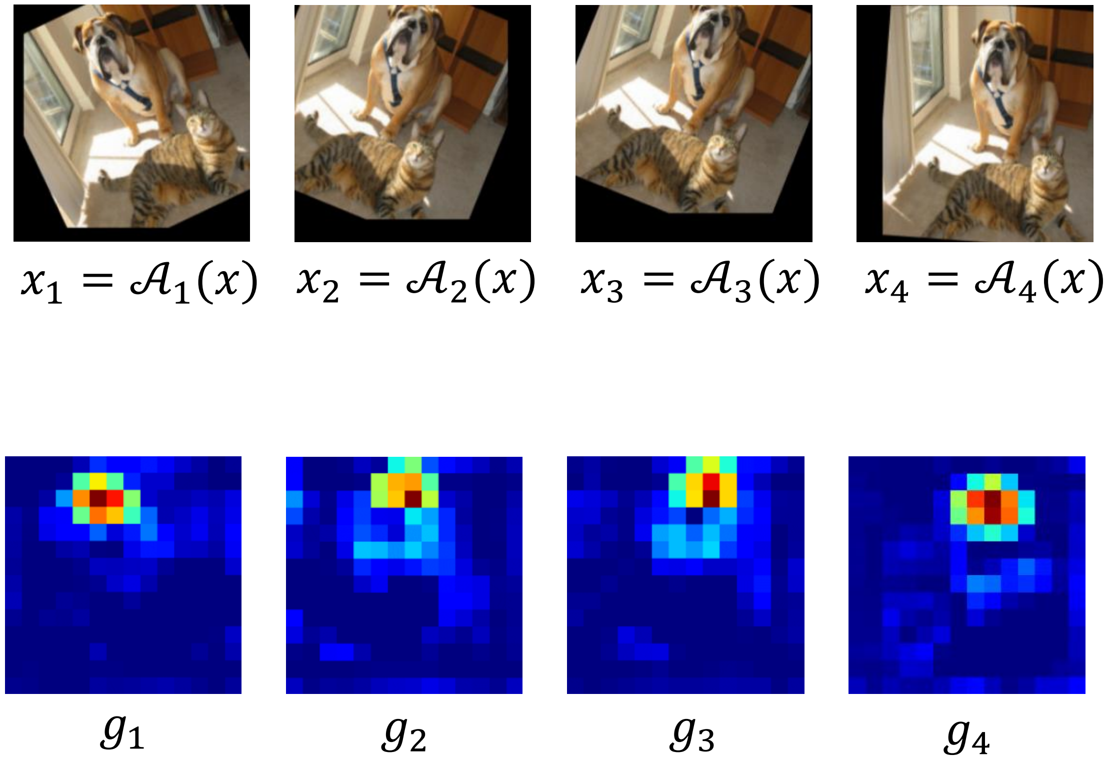
less informative more informative

Augmented Grad-CAM

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

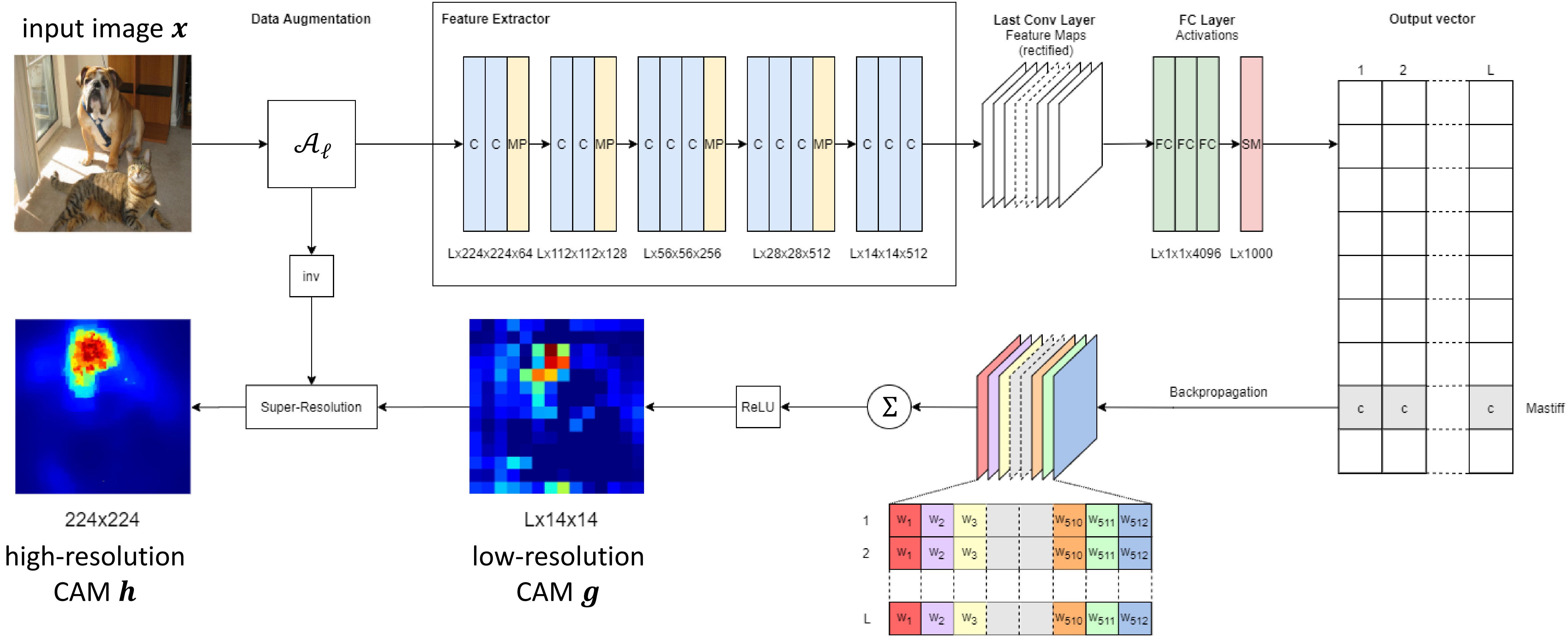
Augmented Grad-CAM: increase heat-maps resolution through image augmentation

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 $\boldsymbol{h}$



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

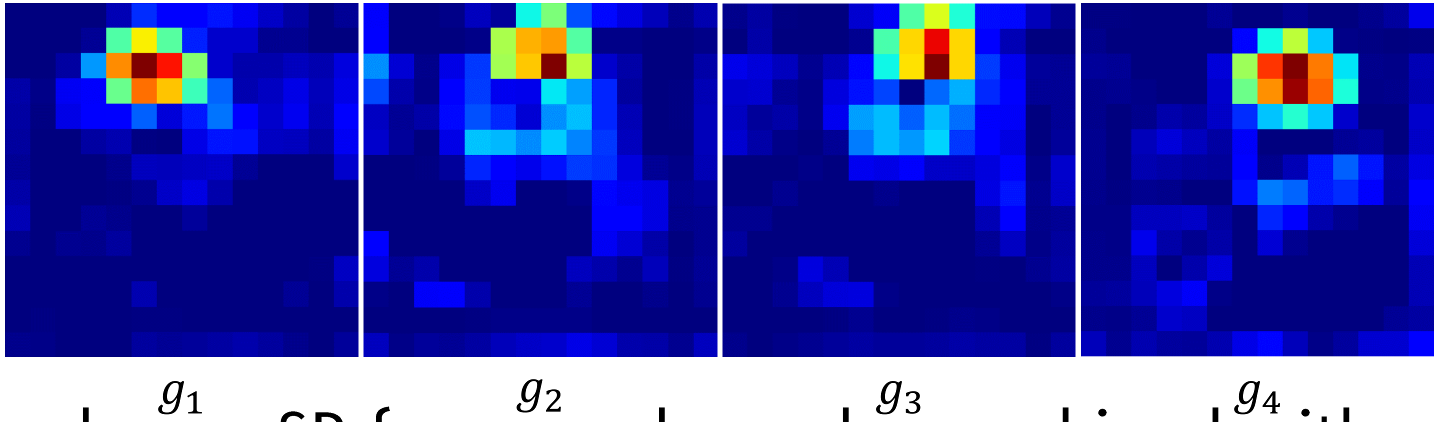


Morbidelli, P., Carrera, D., Rossi, B., Fragneto, P., & Boracchi, G. (2020, May). Augmented Grad-CAM: Heat-Maps Super Resolution Through Augmentation. In *ICASSP 2020*

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



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)

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

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*

Localization

Giacomo Boracchi,

Advanced Neural Networks and Deep Learning

http://home.deib.polimi.it/boracchi/

The Localization Task

The input image contains a single relevant object to be classified in a fixed set of categories

The task is to:

1) assign the object class to the image

hawk

The Localization Task

The input image contains a single relevant object to be classified in a fixed set of categories

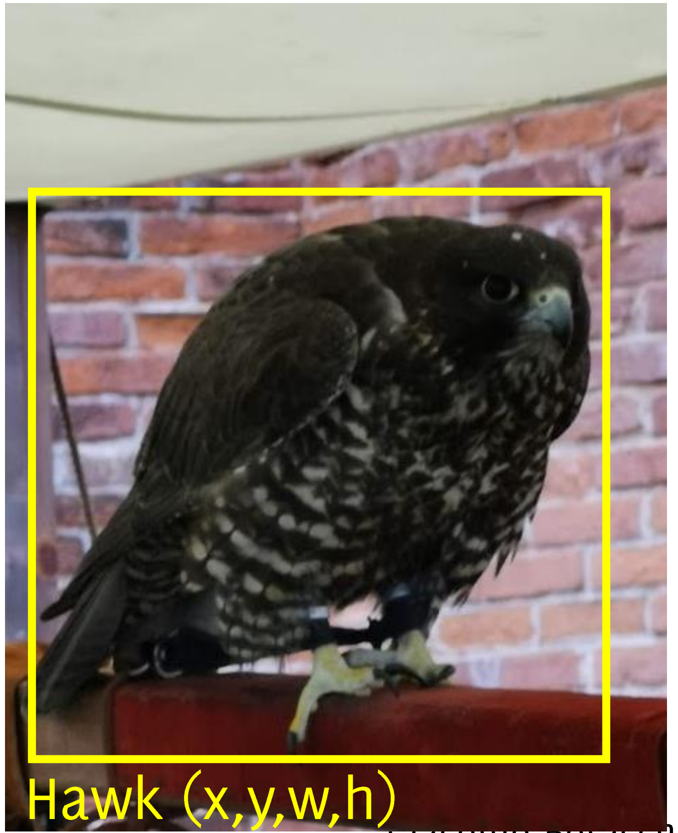
The task is to:

1) assign the object class to the image
2) locate the object in the image by its bounding box

A training set of annotated images with **label and a bounding box** around each object is required

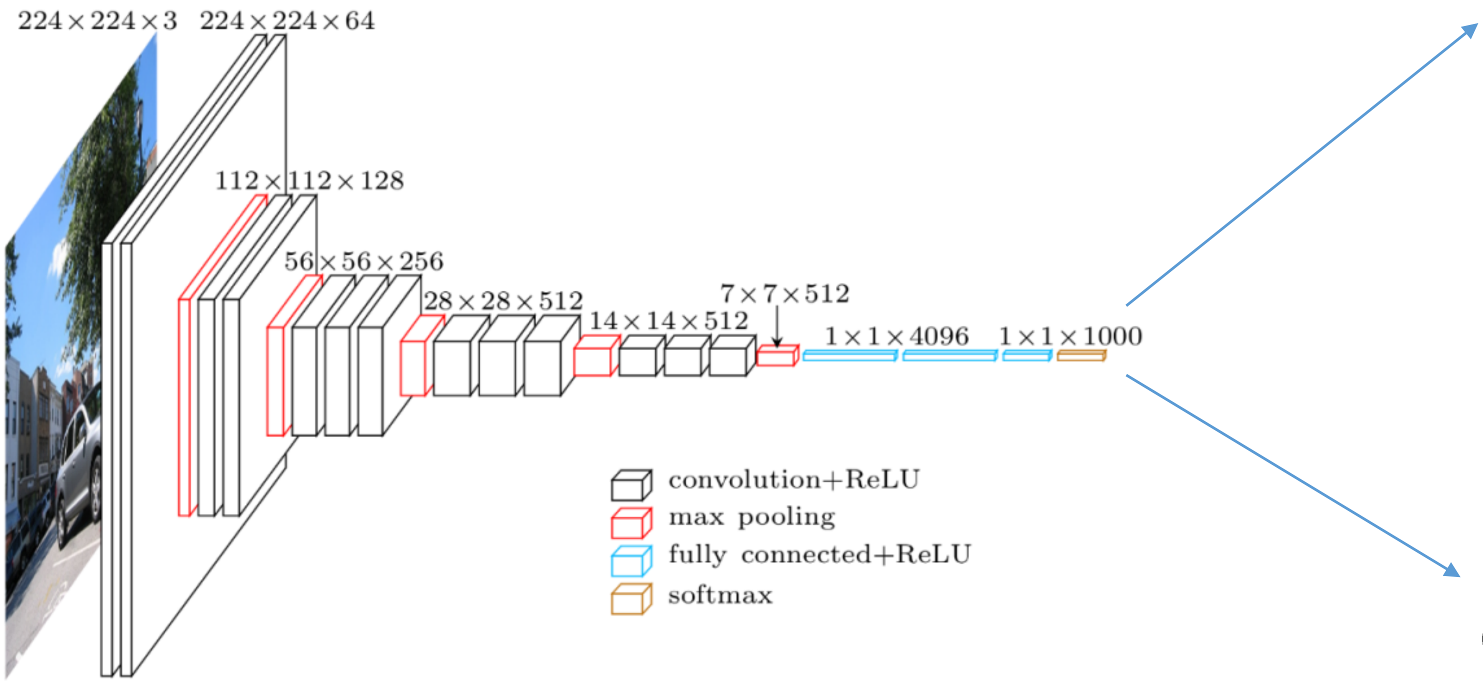
Extended localization problems involve regression over more complicated geometries (e.g. human skeleton)

hawk



The Simplest Solution

Train a network to predict both the class label and the bounding box



Target class
Soft-max loss $\mathcal{S}$

Bounding Box Coordinates
(x, y, w, h)
Regression loss $\mathcal{R}$, e.g. the $\ell^2, \ell^1, ..$ bounding box.

The simplest solution

The training loss has to be a single scalar since **we compute gradient of a scalar function** with respect to network parameters.

So one tend to **minimize a multitask loss to merge two losses**:

$$\mathcal{L}(x) = \alpha\, \mathcal{S}(x) + (1 - \alpha)\mathcal{R}(x)$$

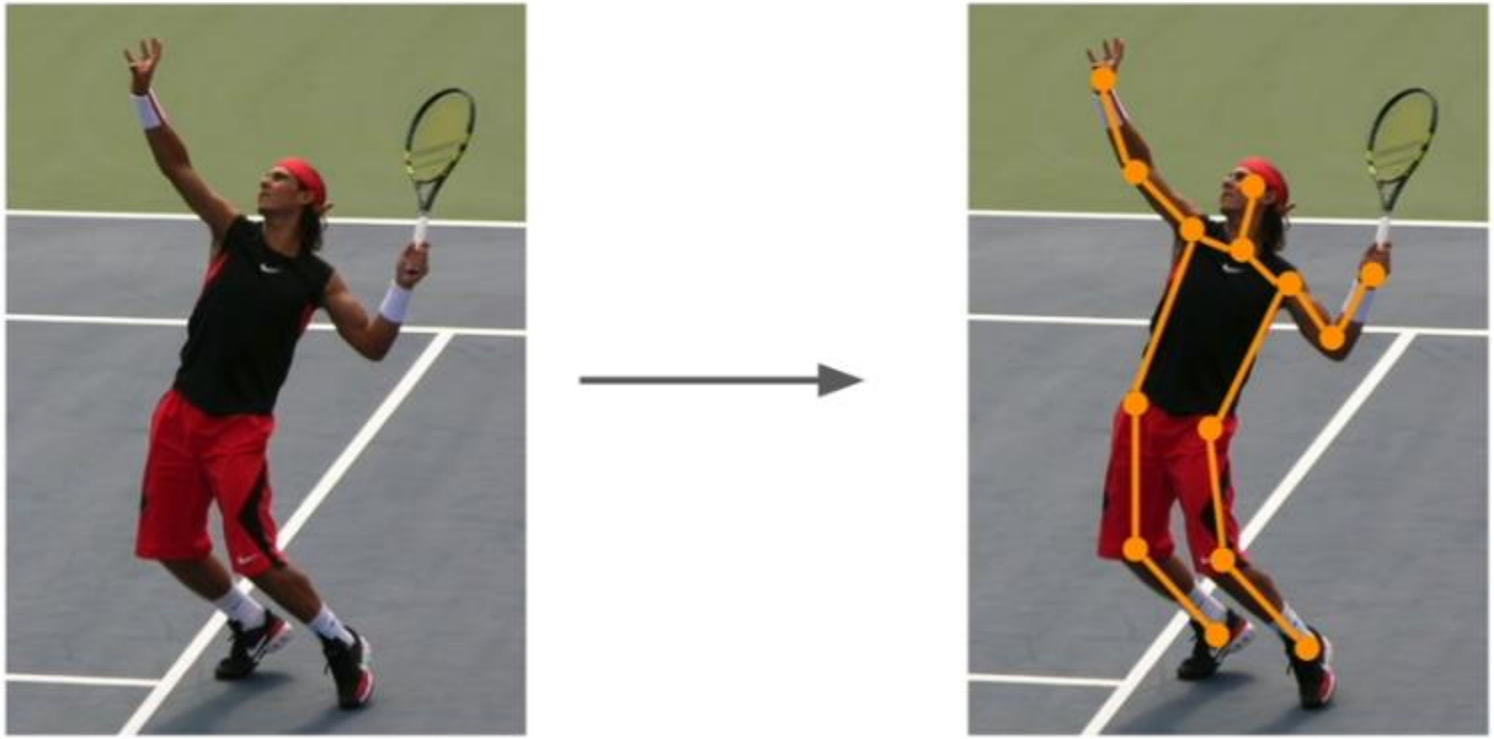
and α is an hyper parameter of the network.

Watch out that $\boldsymbol{\alpha}$ **directly influences the loss definition**, so tuning might be difficult, better to do cross-validation looking at some other loss.

It is also possible to **adopt a pre-trained model** and then train the two FC separately… however it is always better to perform at least some fine tuning to train the two jointly.

Extension to Human Pose Estimation

Pose estimation is formulated as a **CNN-regression problem towards body joints.**



Represent pose as a set of 14 joint positions:

Left / right foot
Left / right knee
Left / right hip
Left / right shoulder
Left / right elbow
Left / right hand
Neck
Head top

Alexander Toshev and Christian Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

Extension to Human Pose Estimation

Pose estimation is formulated as a **CNN-regression problem towards body joints.**

- **The network receives as input the whole image, capturing the full-context of each body joints.**
- The approach is **very simple to design and train.** Training problems can be **alleviated by transfer learning** of existing classification networks

Pose is defined as a vector of k joints location for the human body, possibly normalized w.r.t. the bounding box enclosing the human

Train a CNN to predict **a $2k$ vector as output** by using an Alexnet-like architecture

Alexander Toshev and Christian Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

Training Human Pose Estimation Networks

Adopt a ℓ^2 regression loss of the estimated pose parameters over the annotations.

- This can be also defined when a few joints are not visible

Reduce overfitting by augmentation (translation and flips)

Multiple networks have been trained to improve localization by refining joint localtions in a crop around the previous detection

Alexander Toshev and Christian Szegedy, “DeepPose: Human Pose Estimation via Deep Neural Networks”, CVPR 2014

Open Pose

Cao, Z., Hidalgo, G., Simon, T., Wei, S.E. and Sheikh, Y.. OpenPose: realtime multi-person 2D pose estimation using Part Affinity

Object Detection

Object Detection Task

Given a fixed set of categories and an input image which contains an unknown and varying number of instances

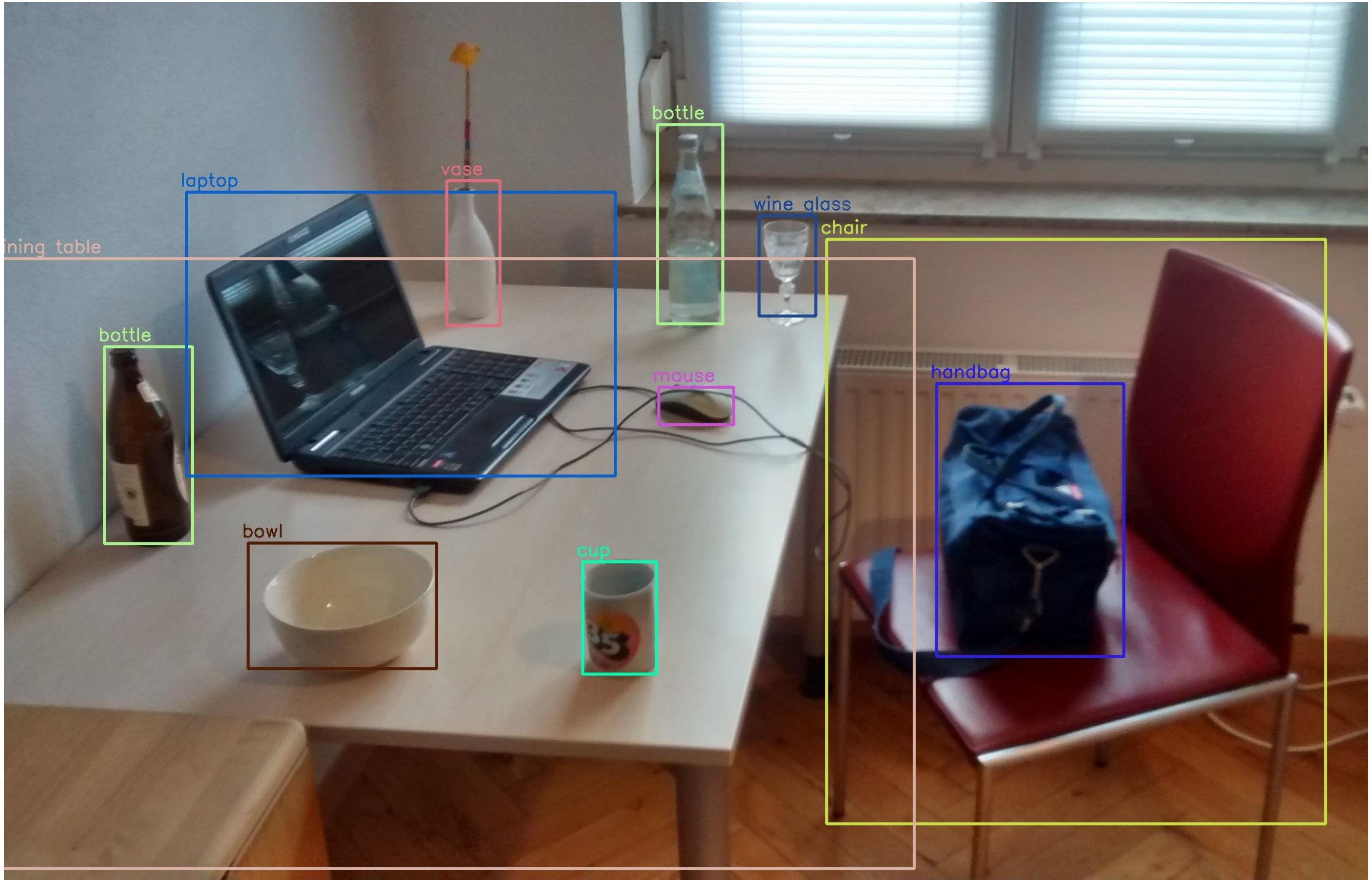
Draw a bounding box on each object instance

A training set of annotated images with labels and bounding boxes for each object is required

Each image requires **a varying number of outputs**

MAN: (x,y,h,w)
KID: (x,y,h,w)
GLOVE: (x,y,h,w)

Annotated Dataset for Object Detection

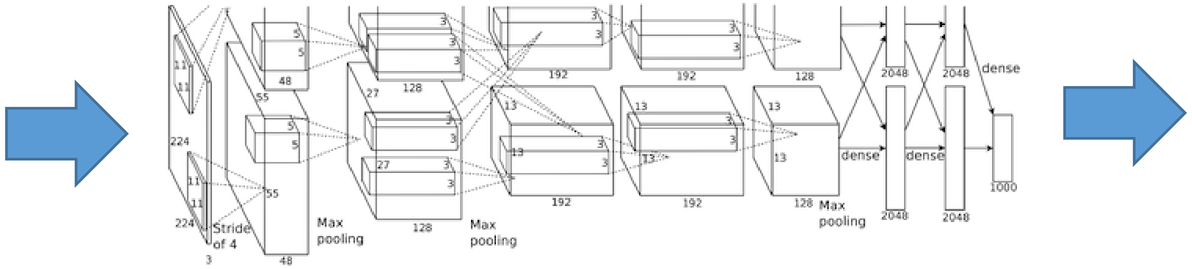


The Straightforward Solution: Sliding Window

1000 x 2000 pixels

- Similar to the sliding window for semantic segmentation
- A pretrained model is meant to process a fixed input size (e.g. 224 x 224 x 3)
- Slide on the image a window of that size and classify each region.
- Assign the predicted label to the central pixel

Adopt the whole machinery seen so far to each crop of the image



car

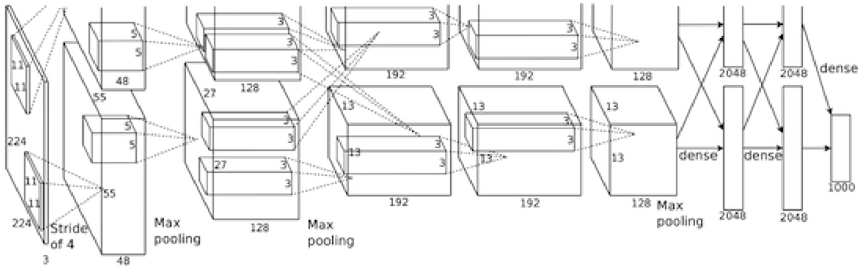
Giacomo Boracchi

The Straightforward Solution: Sliding Window

1000 x 2000 pixels

- Similar to the sliding window for semantic segmentation
- A pretrained model is meant to process a fixed input size (e.g. 224 x 224 x 3)
- Slide on the image a window of that size and classify each region.
- Assign the predicted label to the central pixel

Adopt the whole machinery seen so far to each crop of the image



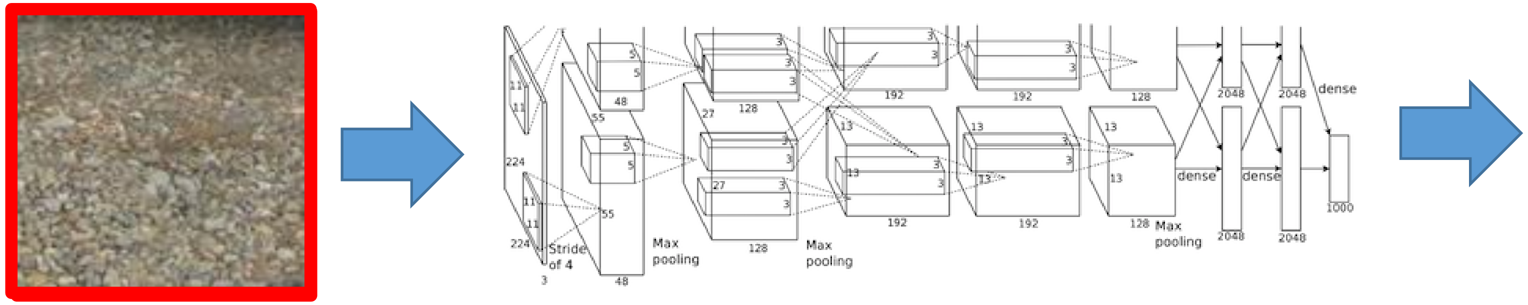
wheel

The Straightforward Solution: Sliding Window

1000 x 2000 pixels

- Similar to the sliding window for semantic segmentation
- A pretrained model is meant to process a fixed input size (e.g. 224 x 224 x 3)
- Slide on the image a window of that size and classify each region.
- Assign the predicted label to the central pixel

Adopt the whole machinery seen so far to each crop of the image
The background class has to be included!



background

Many drawbacks…

Cons:

- **Very inefficient!** Does not re-use features that are «shared» among overlapping crops
- How to choose the crop size?
- Difficult to detect objects at different scales!
- A huge number of crops of different sizes should be considered….

Plus:

- No need of retraining the CNN

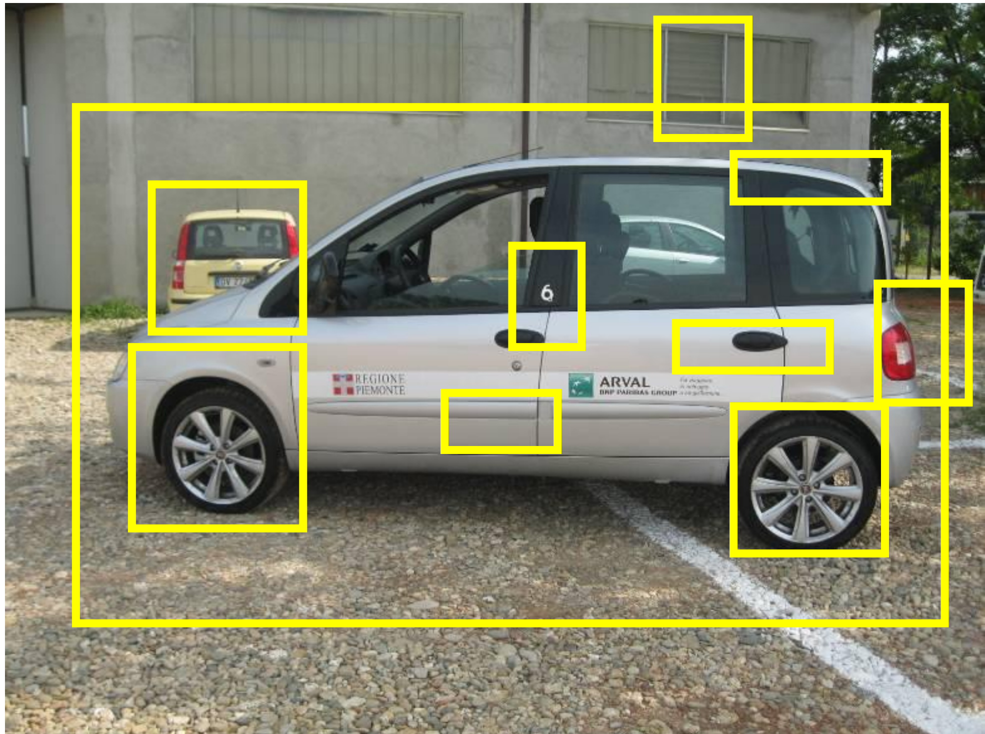
Region Proposal

Region proposal algorithms (and networks) are meant **to identify bounding boxes** that correspond to a **candidate object** in the image.

Algorithms with **very high recall** (but low precision) were there before the deep learning advent

The idea is to:

- Apply a region proposal algorithm
- Classify by a CNN the image inside each proposal regions



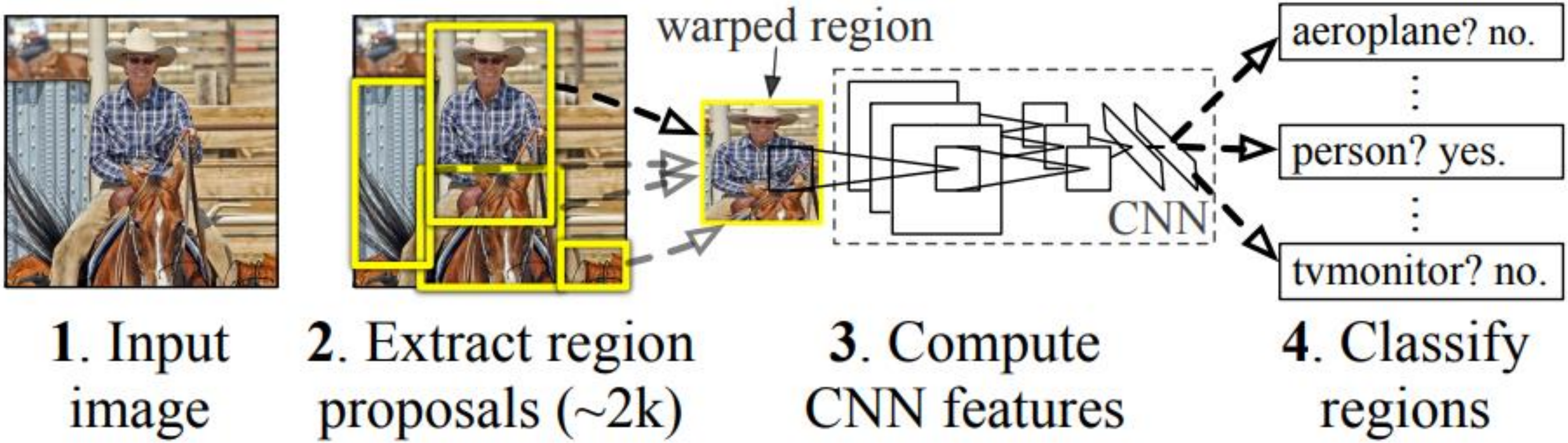
Rich feature hierarchies for accurate object detection and semantic segmentation

Ross Girshick[1] Jeff Donahue[1,2] Trevor Darrell[1,2] Jitendra Malik[1]
[1]UC Berkeley and [2]ICSI
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

R-CNN

Object detection by means of region proposal (R stands for **regions**)

SVM +
BB regressor



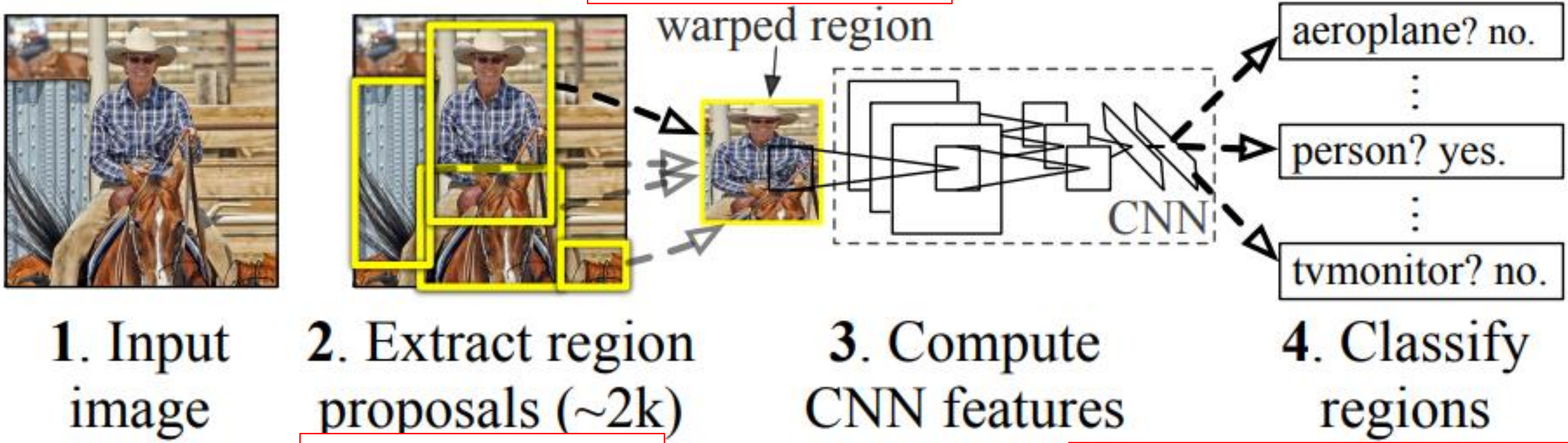
Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *CVPR 2014*

R-CNN

Object detection by means of region proposal

Warping to a predefined size is necessary since the CNN has a FC layer

SVM + BB regressor



There is no learning in the region proposal algorithm, **very high recall**

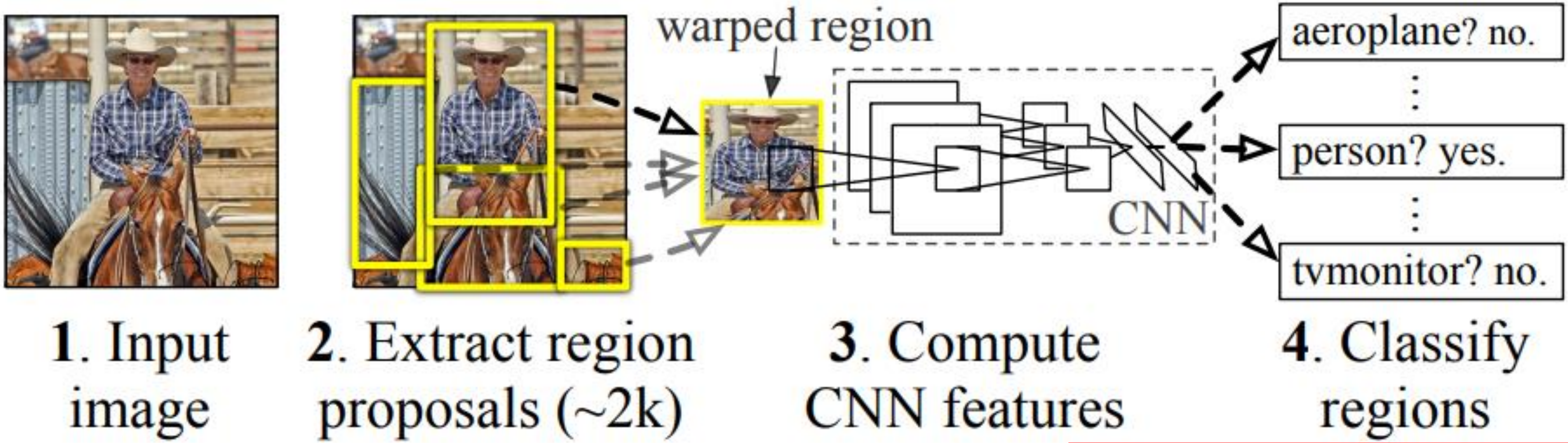
The regions are refined by a regression network to correct the bounding box estimate from RP.

R-CNN

Object detection by means of region proposal

SVM +
BB regressor

This is a pre-trained network



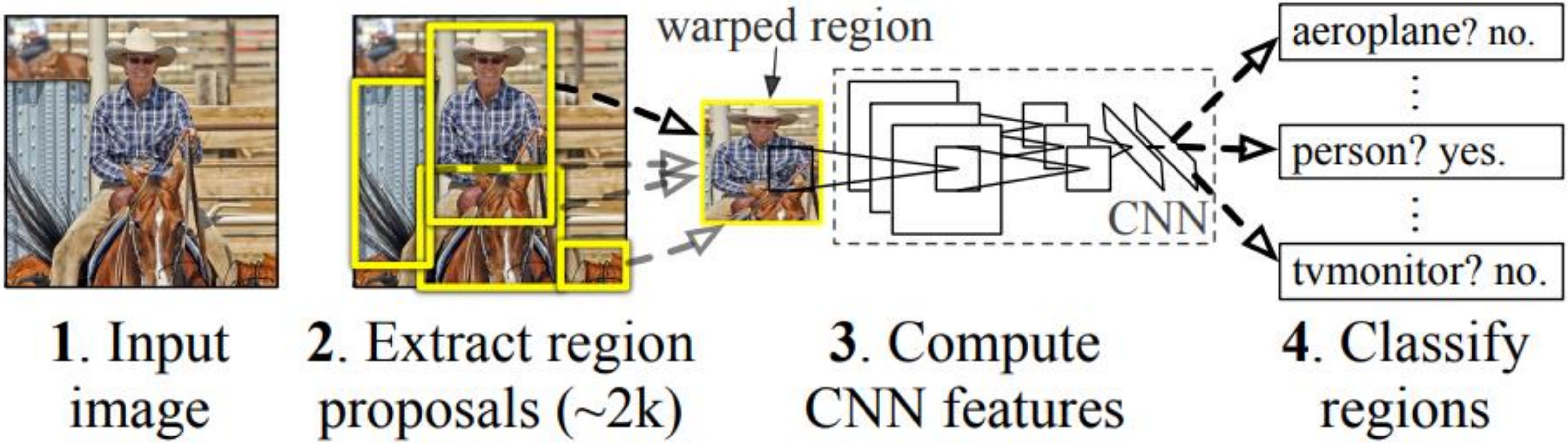
Trained with multi-task loss
Region of interest can exceed image boundaries

R-CNN

Object detection by means of region proposal

The CNN is fine-tuned during training

SVM +
BB regressor



Include **a background class** to get rid of those regions not corresponding to an object

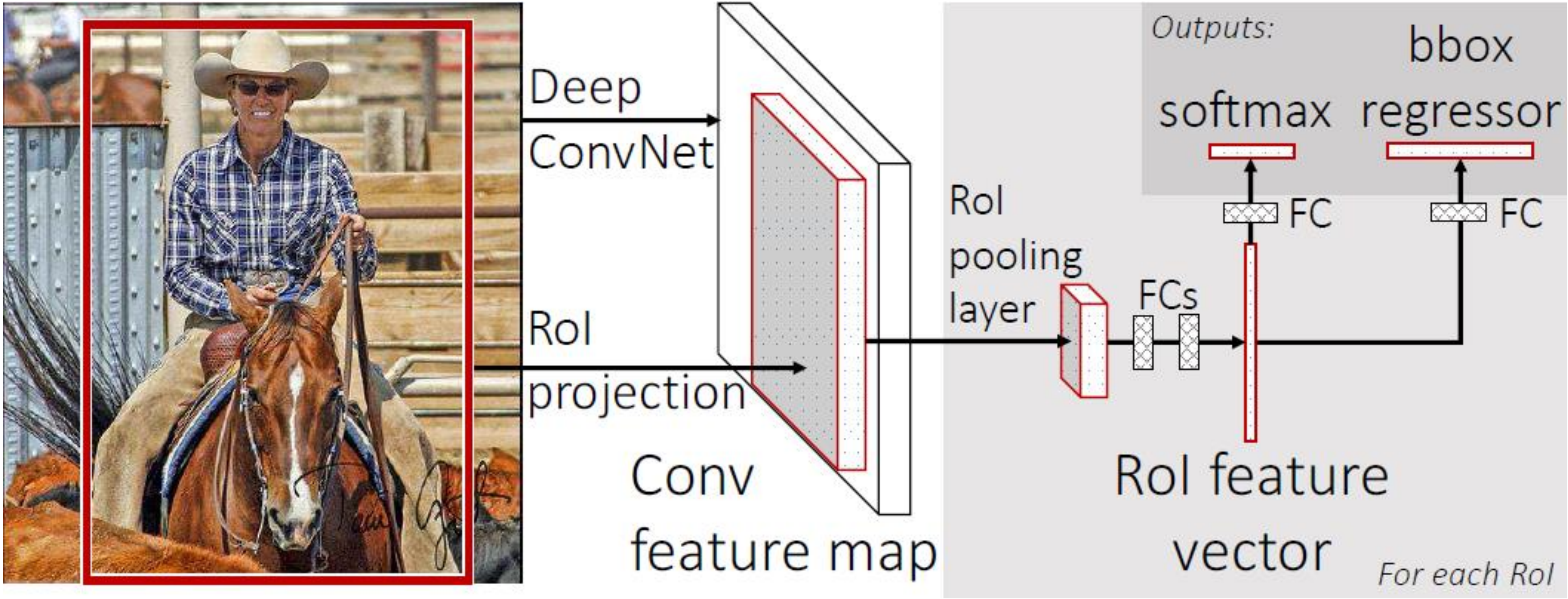
R-CNN limitations

- **Ad-hoc training objectives** and not an end-to-end training
    - Fine-tune network with softmax classifier (log loss)
    - Train post-hoc linear SVMs (hinge loss)
    - Train post-hoc bounding-box regressions (least squares)
- **Region proposals are from a different algorithm** and that part has not been optimized for the detection by CNN
- **Training is slow** (84h), takes **a lot of disk space to store features**
- **Inference** (detection) **is slow** since the CNN has to be executed on each region proposal (**no feature re-use**)
    - 47s / image with VGG16

Fast R-CNN

Ross Girshick
Microsoft Research
rbg@microsoft.com

Fast R-CNN

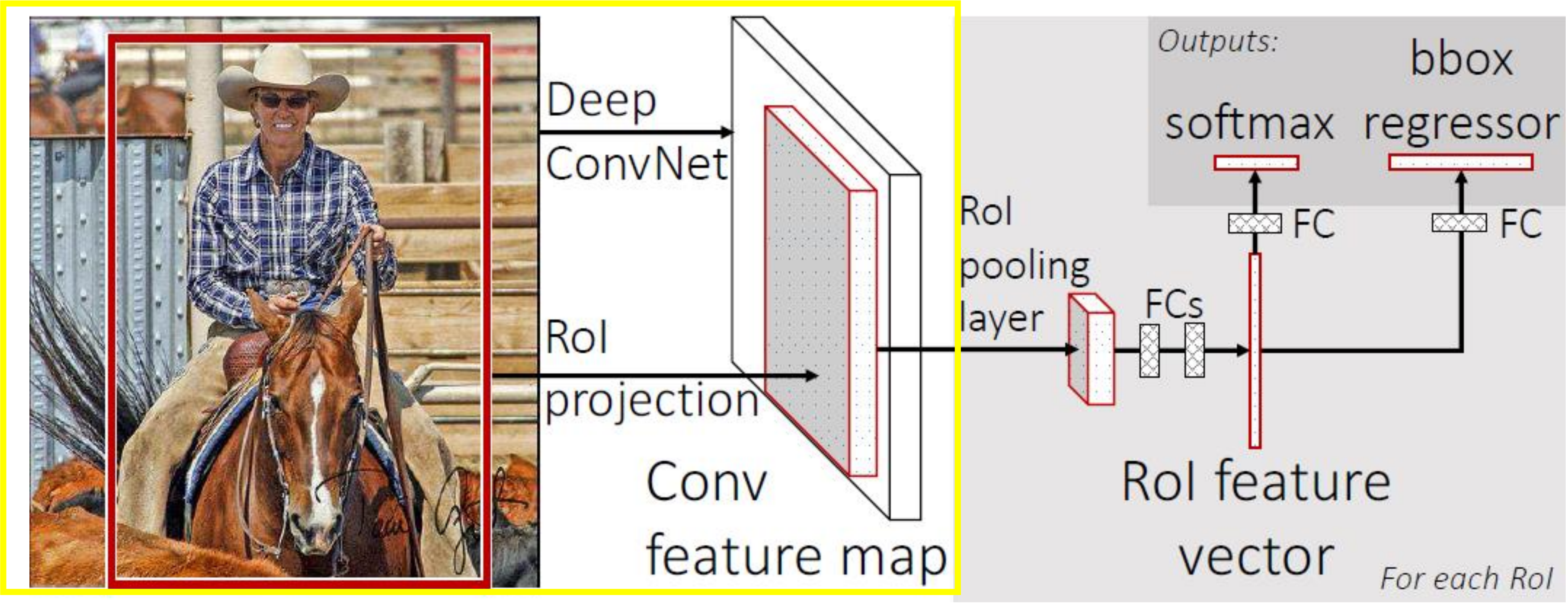


«conv5»

Girshick, Ross. "Fast r-cnn." ICCV 2015

Fast R-CNN

1. The whole image is fed to a CNN that extracts feature maps.
2. **Region proposals** are identified from the image and **projected into the feature maps.** Regions are directly cropped form the feature maps, instead from the image: →**re-use convolutional computation.**

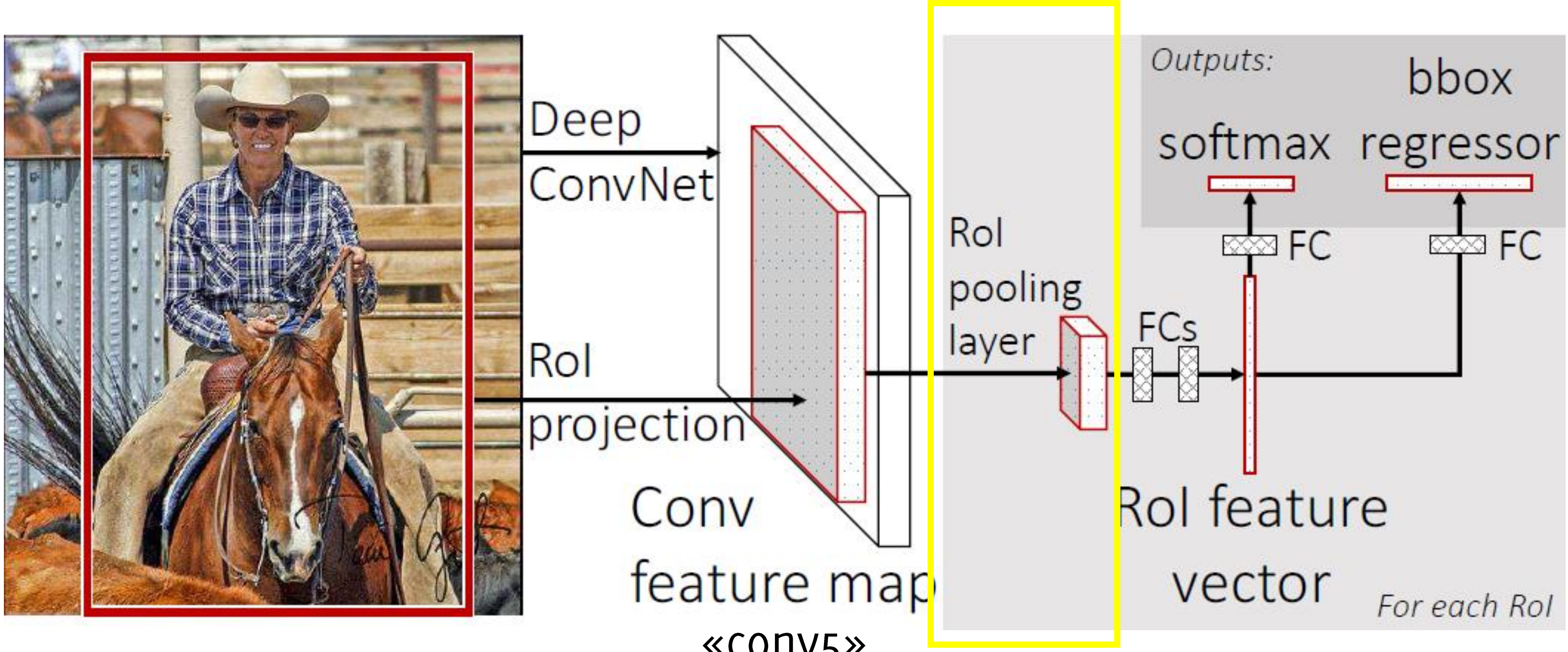


«conv5»

Girshick, Ross. "Fast r-cnn." ICCV 2015

Fast R-CNN

3. Fixed size is still required to feed data to a fully connected layer. ROI pooling layers extract a **feature vector of fixed size $H \times W$** from each region proposal. Each **ROI in the feature maps is divided in a $H \times W$ grid** and then **maxpooling** over this provides the feature vector



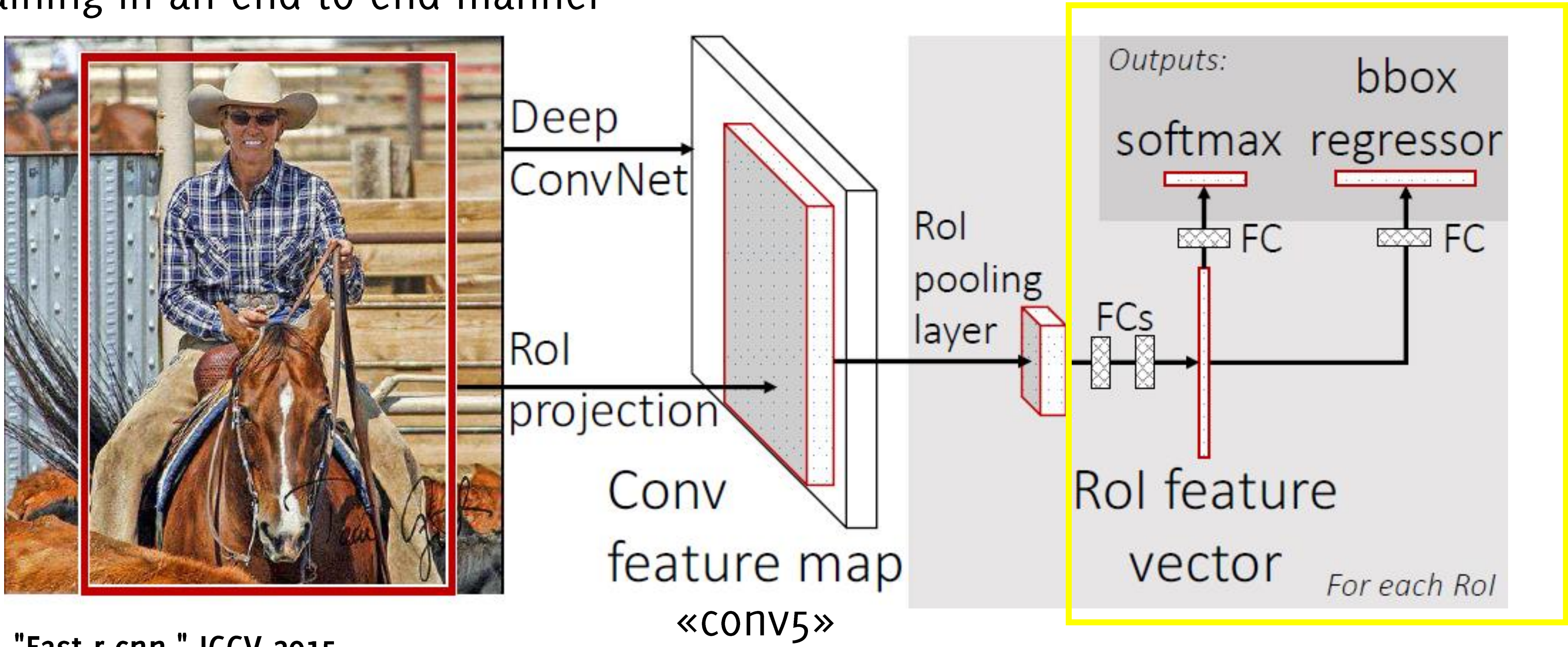
«conv5»

Girshick, Ross. "Fast r-cnn." ICCV 2015

Fast R-CNN

4. The **FC layers estimate both classes and BB location** (bb regressor)
A **convex combination of the two** is used as a **multitask loss** to be optimized (as in R-CNN, but no SVM here).

5. Training in an end-to-end manner



Girshick, Ross. "Fast r-cnn." ICCV 2015

Fast R-CNN

In this new architecture it is possible to **back-propagate through the whole network**, thus train the whole network in an end-to-end manner

It becomes **incredibly faster than R-CNN during testing.**

Now that convolutions are not repeated on overlapping areas, **the vast majority of test time is spent on ROI extraction**

Girshick, Ross. "Fast r-cnn." ICCV 2015

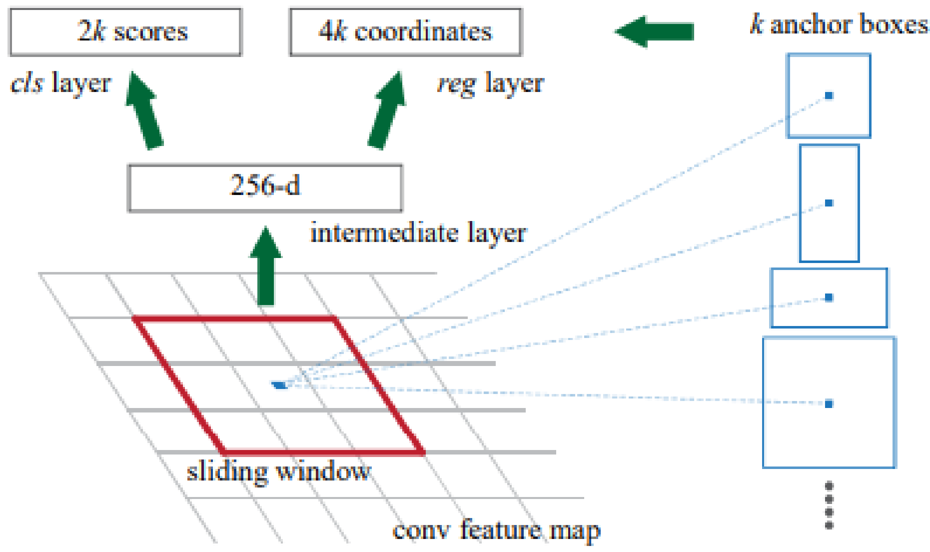
Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren* **Kaiming He** **Ross Girshick** **Jian Sun**
Microsoft Research
{v-shren, kahe, rbg, jiansun}@microsoft.com

Faster R-CNN

- Instead of the ROI extraction algorithm, a region proposal network (RPN), which is a F-CNN (3x3 filter size)
- **RPN operates on feature maps** of the last conv layers
- Then, operations are very similar to a Fast R-CNN

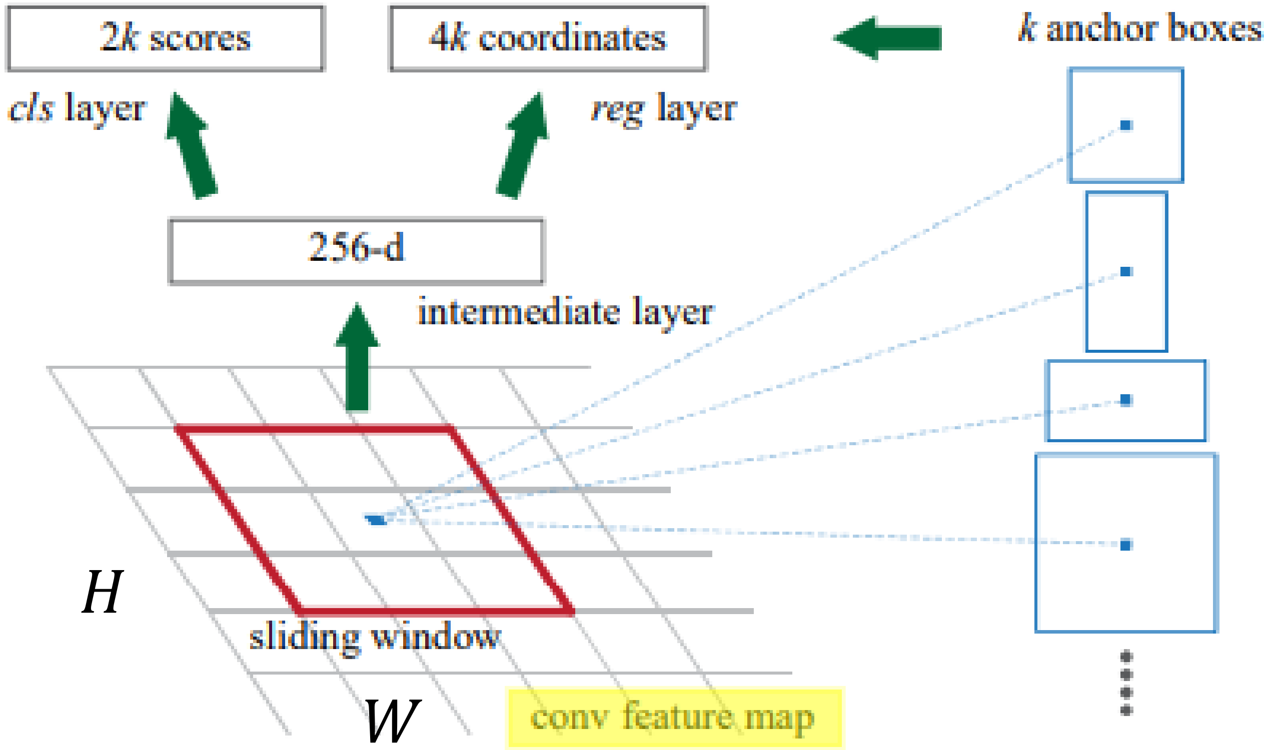
RPN



Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

RPN

- Takes as input a 3×3 region in the feature maps
- Maps (by 1×1 convolutions) the region to a lower dim. vector
- In each point consider k anchor boxes, i.e. different ROI size/proportions
- $H \times W \times k$ candidate anchors and estimate scores for each of these anchors

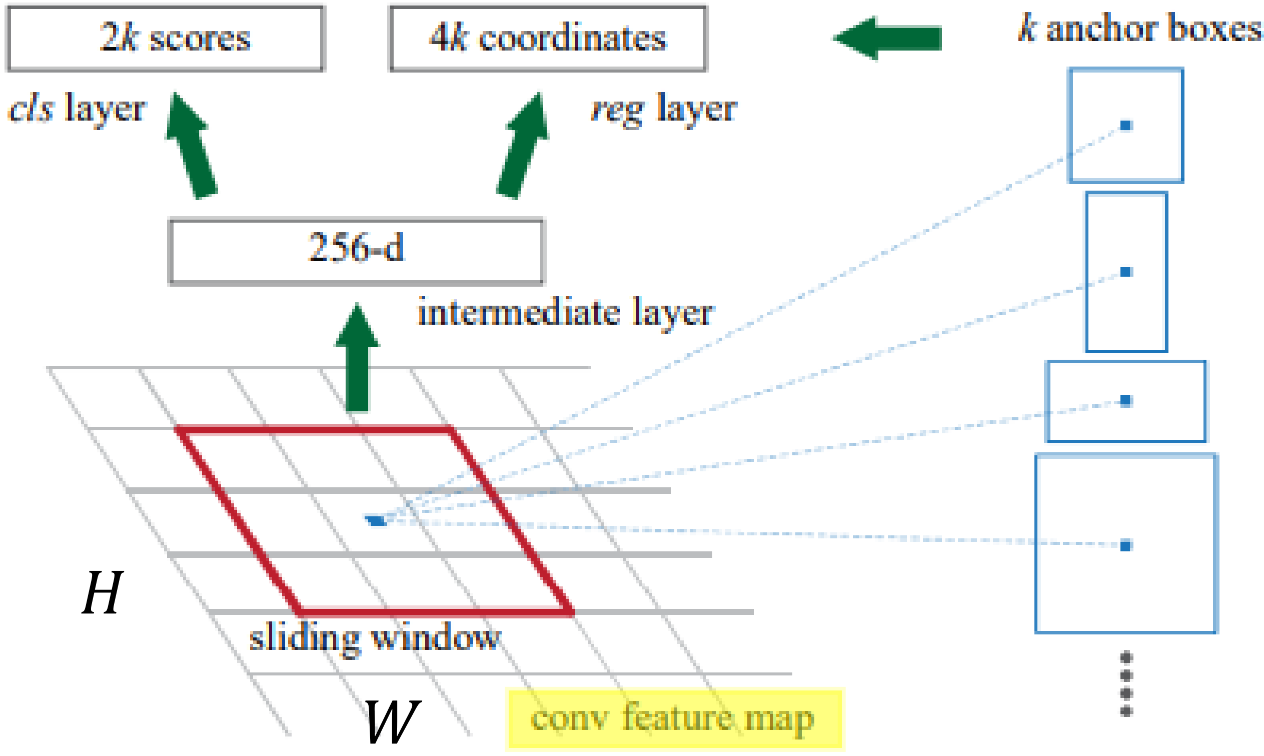


Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

RPN using k anchors

- The **cls network** is trained to predict the *object probability,* i.e. that each anchor contains and object *[contains an object / does not contain an object]* $\rightarrow 2k$ probability estimates

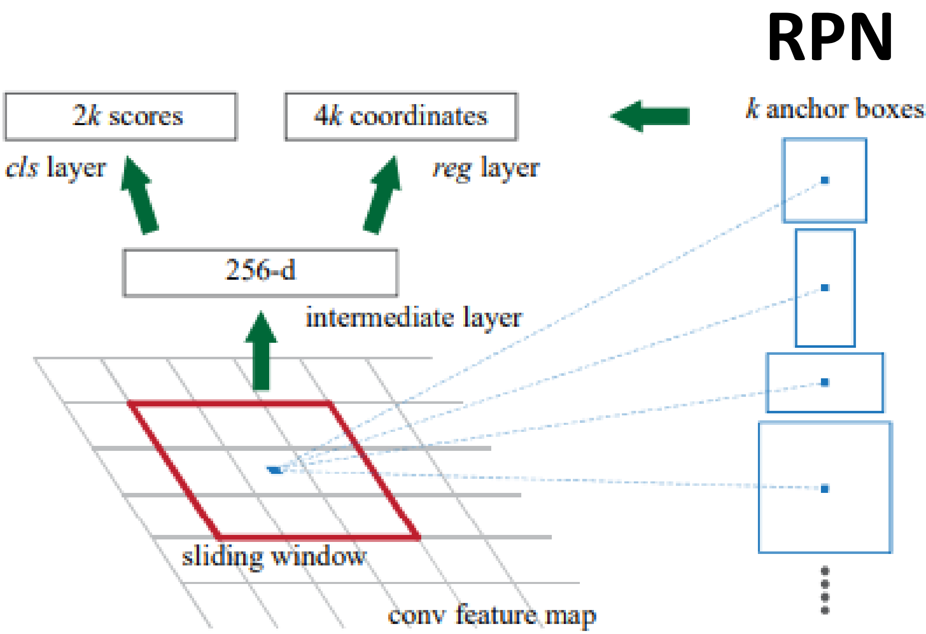
- The **reg network** is trained to *adjust* the anchor to the object ground truth $\rightarrow 4k$ estimates



- If you want to change the anchors, there is no need to design different RPN, bust just to **define different labels when training the RPN**

Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

Faster R-CNN



- **Training** now involves 4 losses:
  - RPN classify object/non object
  - RPN regression coordinates
  - Final classification score
  - Final BB coordinates
- During training, object/non object ground truth is defined by measuring the overlap with annotated BB
- The network becomes much faster (0.2s test time per image)

Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

Faster R-CNN

At test time,

- Take the top $\sim$ 300 anchors according to their object scores
- Consider the refined bounding box location of these 300 anchors
- These are the ROI to be fed to a Fast R-CNN
- Classify ROI

- Therefore, Faster R-CNN provides to each image a set of BB with their *objectness score*

Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." ***NIPS 2015***

Faster R-CNN

It’s still a **two stage detector**

First stage:

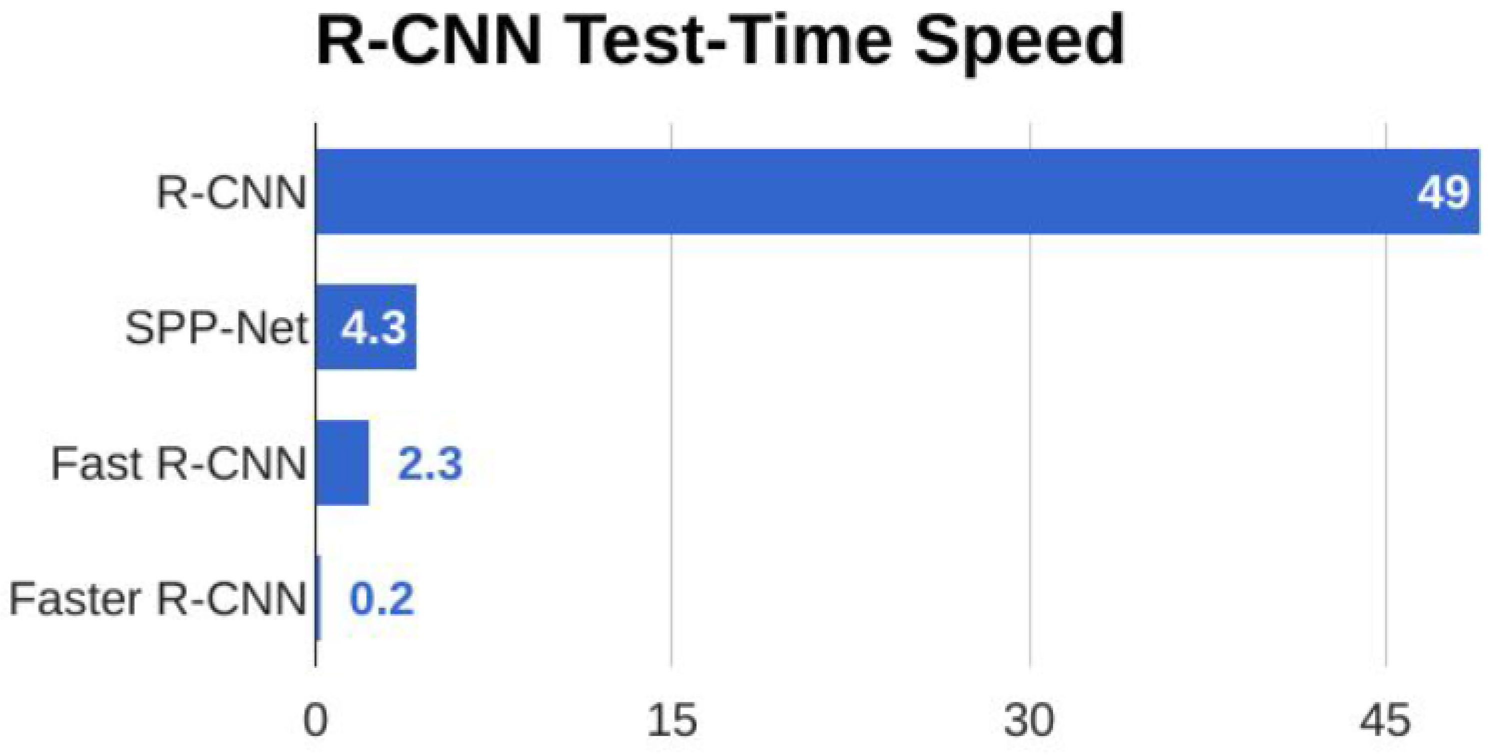
- run a **backbone network** (e.g. VGG16) to **extract featues**
- run the **Region Proposal Network** to **estimate** ~ 300 **ROI**

Second stage (the same as in Fast R-CNN):

- **Crop Features** through ROI pooling (with alignment)
- **Predict object class** using FC + softmax
- **Predict bounding box offset** to improve localization using FC + softmax

Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

Faster R-CNN



Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS 2015*

You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Giacomo Boracchi

YOLO/SSD

R-CNN methods are based on region proposals

There are also region-free methods, like:

Yolo: You Only Look Once

SSD: Single Shot Detectors

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." CVPR 2016.

The rationale

Detection networks are indeed a pipeline of multiple steps.

In particular, **region-based methods make it necessary to have two steps** during inference

This can be slow to run and hard to optimize, because each individual component must be trained separately.

In Yolo "***we reframe the object detection as a single regression problem,*** *straight from image pixels to bounding box coordinates and class probabilities"*

And solve these regression problems **all at once**, with a large CNN

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *CVPR* 2016.
Liu, Wei, et al. "SSD: Single shot multibox detector." *European conference on computer vision*. Springer, Cham, 2016.

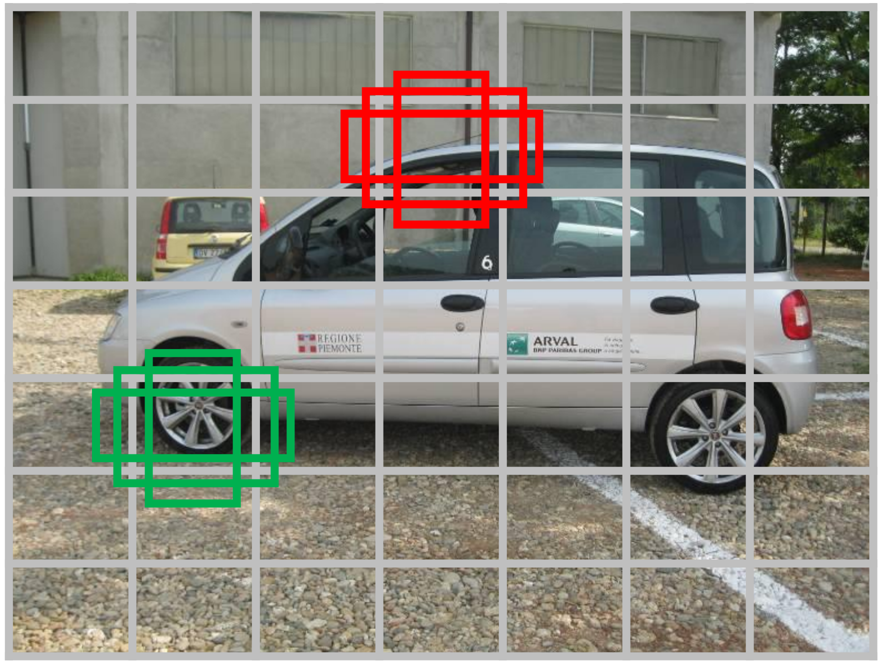
YOLO

1. divide the image in a coarse grid (e.g. 7x7)

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *CVPR* 2016.

YOLO

1. divide the image in a coarse grid (e.g. 7x7)
2. each grid cell contains B *base-bounding boxes* associated



Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *CVPR* 2016.

YOLO

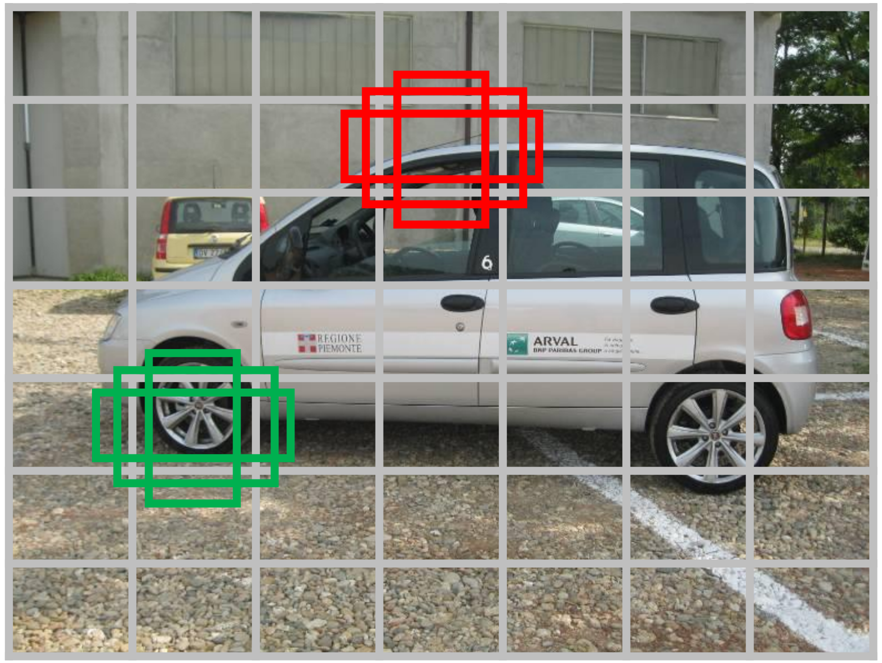
3. For each cell and *base bounding box* we want to predict:
   - The **offset of the base bounding box**, to better match the object:

     $$(\mathrm{dx}, \mathrm{dy}, \mathrm{dh}, \mathrm{dw}, \mathrm{objectness\_score})$$

   - The **classification score of the base-bounding box** over the C considered categories (including background)

So, the output of the network has dimension

$$7 \times 7 \times B \times (5 + C)$$



Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *CVPR* 2016.

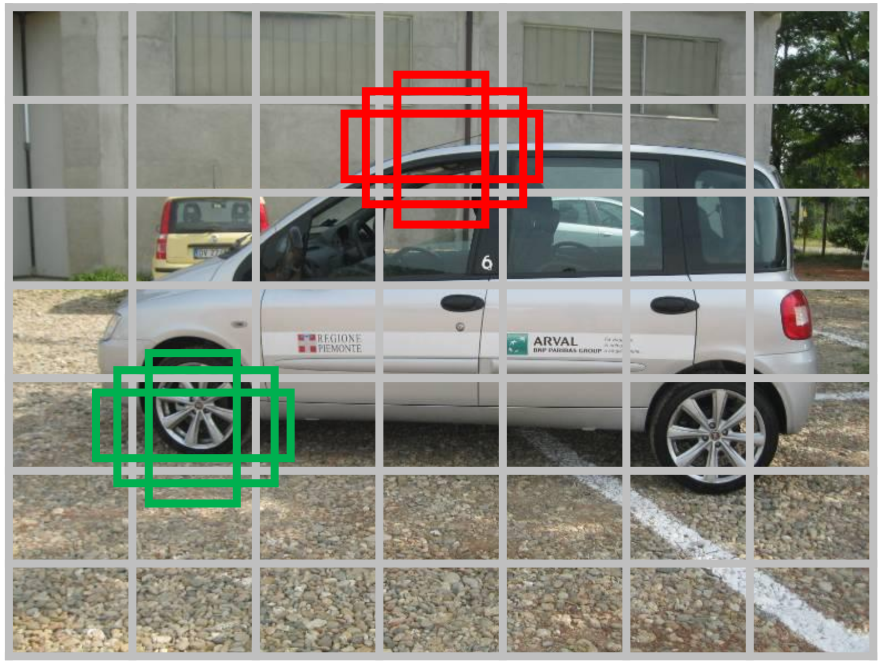
YOLO

The whole **prediction is performed in a single forward pass** over the image, **by a single convolutional network**

Training this network is sort of **tricky to assess the** loss (matched / not matched)

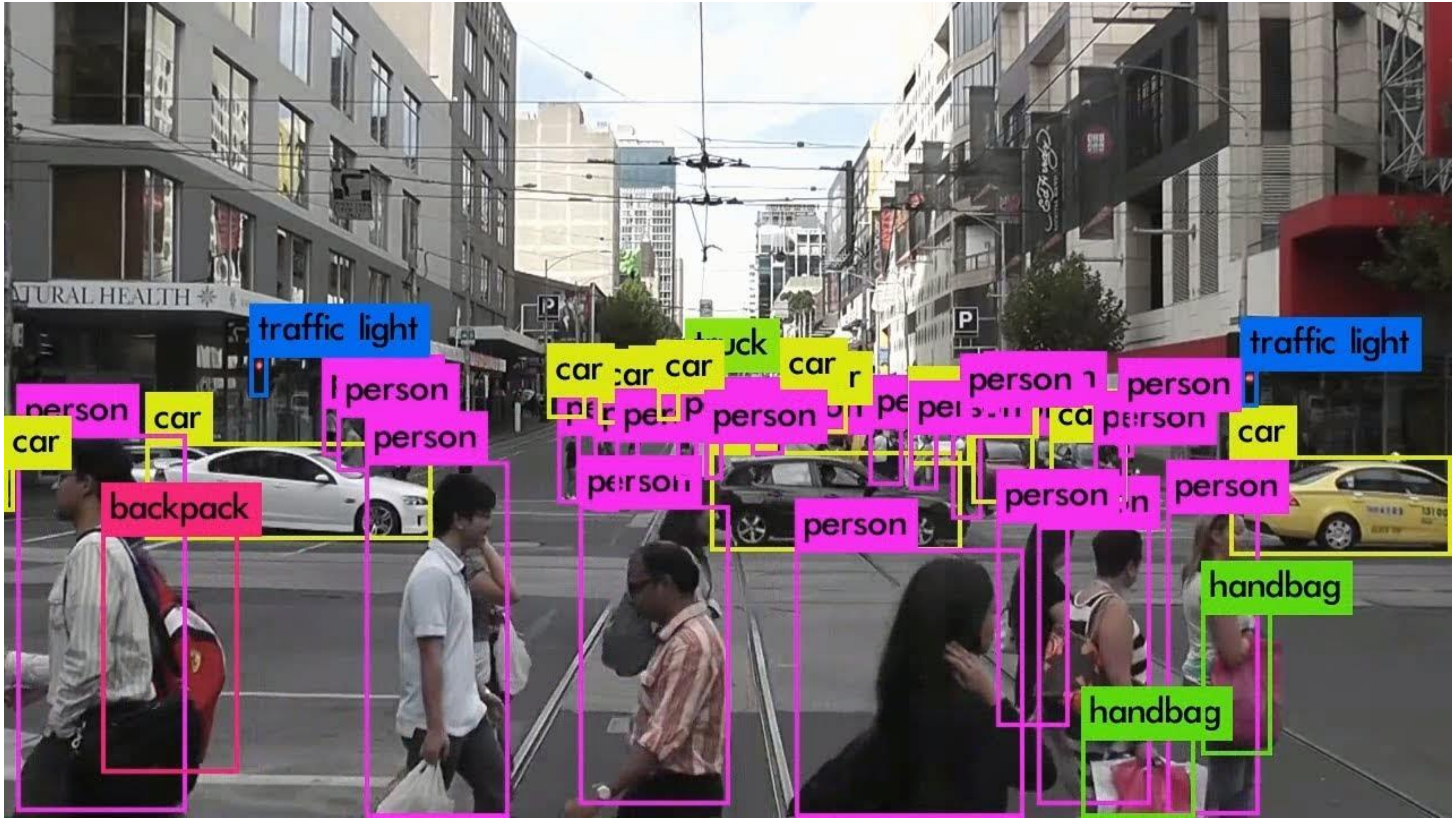
YOLO/SSD shares a similar ground of the RPN used in Faster R-CCN

Typically networks based on region-proposal are more accurate, single shot detectors are faster but less accurate



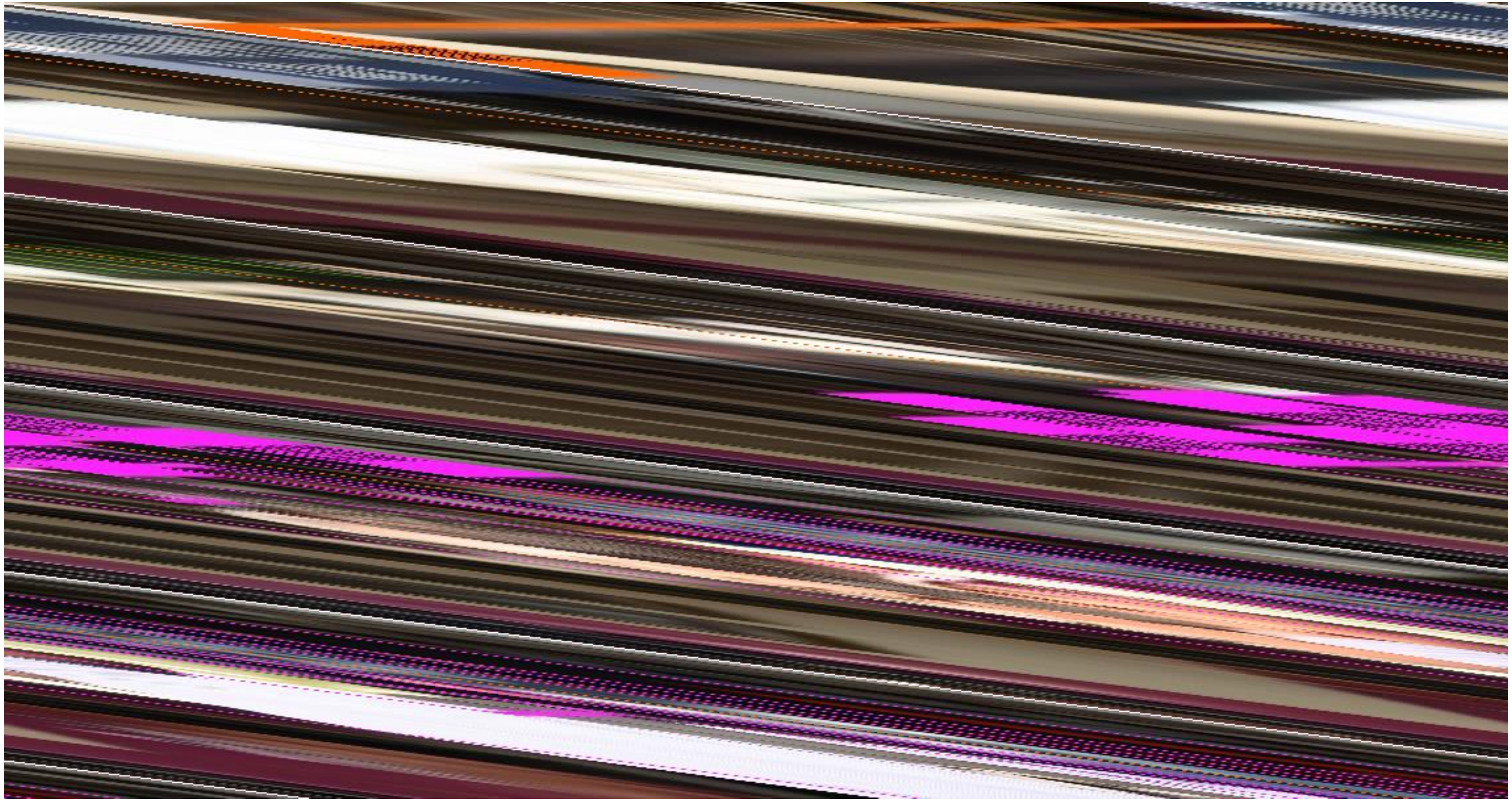
Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *CVPR* 2016.

Object Detection



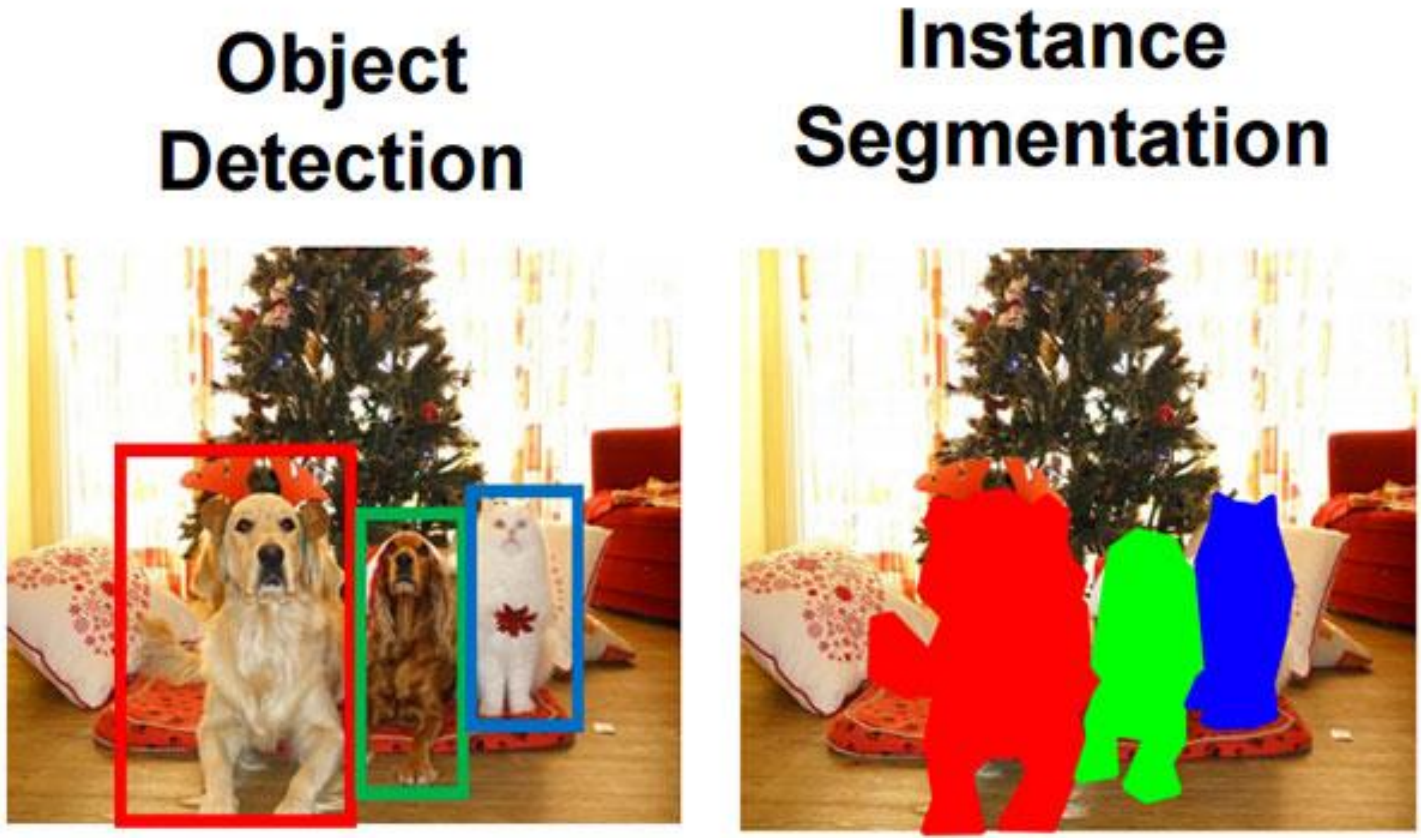
Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Object Detection



Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Object Detection vs Instance Segmentation



[CS231n: Convolutional Neural Networks for Visual Recognition](http://cs231n.github.io/) http://cs231n.github.io/

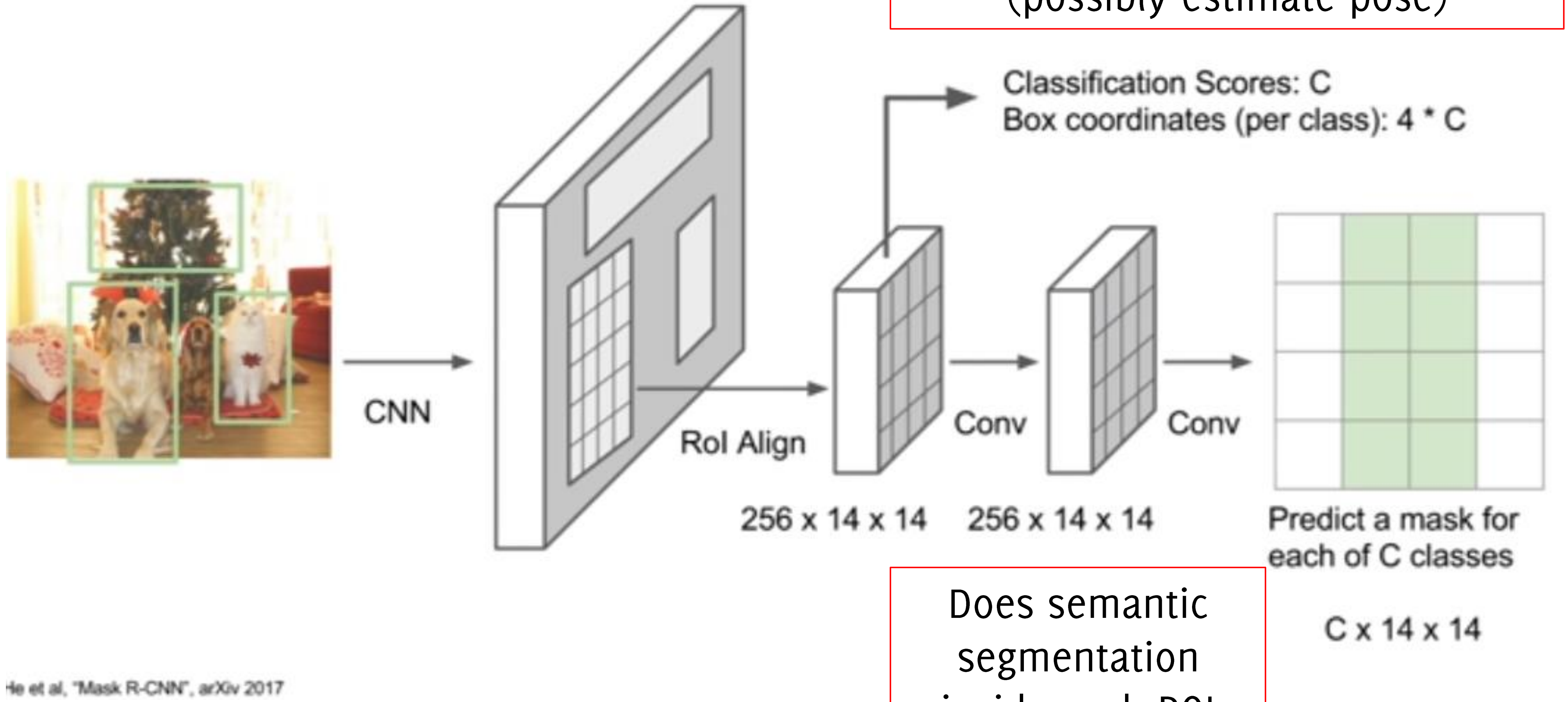
Instance Segmentation

It combines the challenges of:

- **Object detection** (multiple instances present in the image)
- **Semantic segmentation** (associate a label to each pixel) separating each object instance

Mask R-CNN

As in Fast R-CNN classify the whole ROI and regress the bounding box (possibly estimate pose)

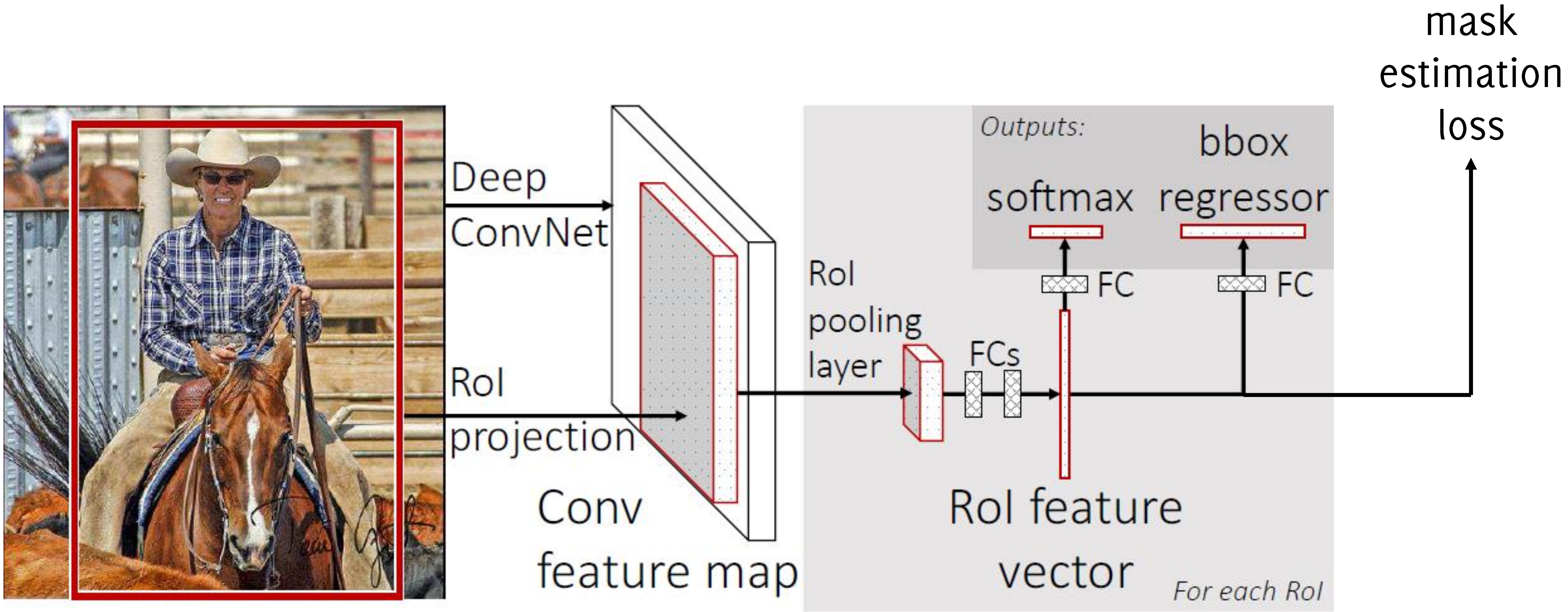


Does semantic segmentation inside each ROI

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

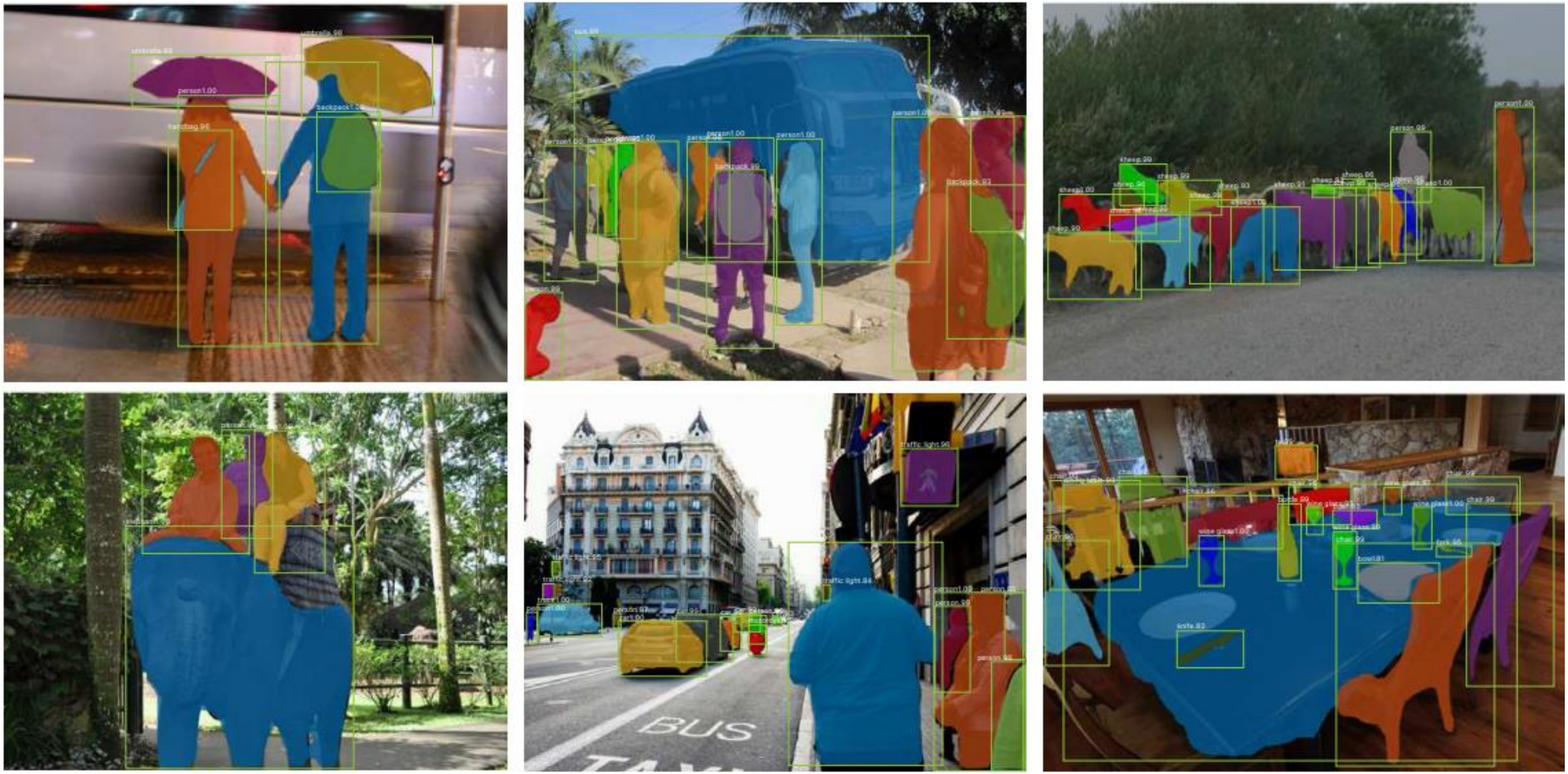
Mask R-CNN

Mask is estimated for each ROI and each class



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

Mask R-CNN (end-to-end training)



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

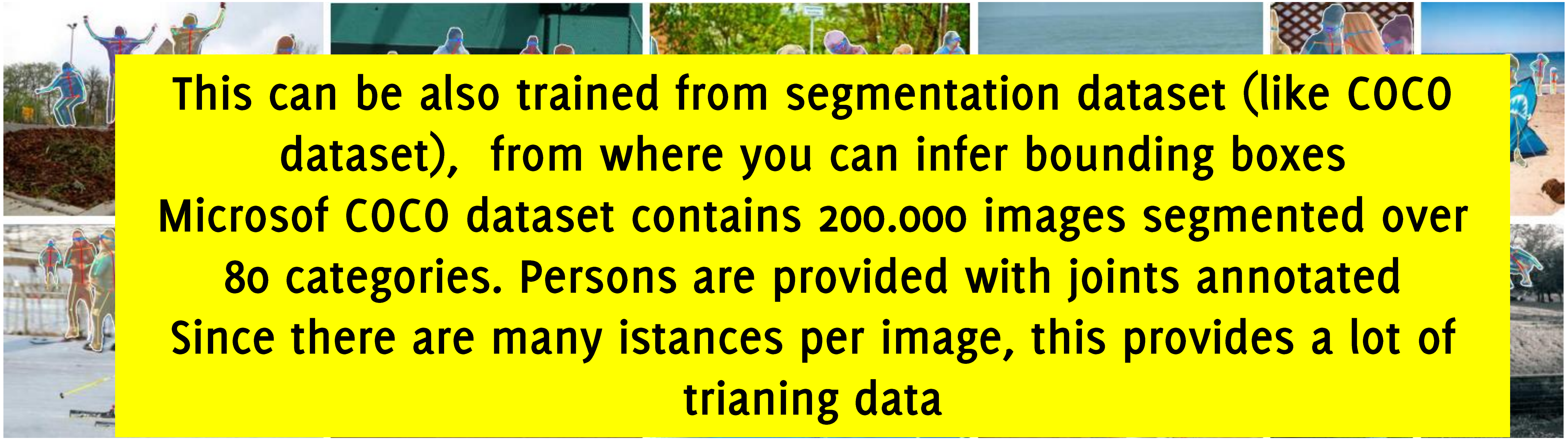
Mask R-CNN (including Pose estimation)

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

Mask R-CNN (including Pose estimation)

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

Mask R-CNN (including Pose estimation)



This can be also trained from segmentation dataset (like COCO dataset), from where you can infer bounding boxes
Microsof COCO dataset contains 200.000 images segmented over 80 categories. Persons are provided with joints annotated
Since there are many istances per image, this provides a lot of trianing data

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