Objects detection

Diane Lingrand and many contributors

UNIVERSITÉ
CÔTE D'AZUR
Master Data Science M1

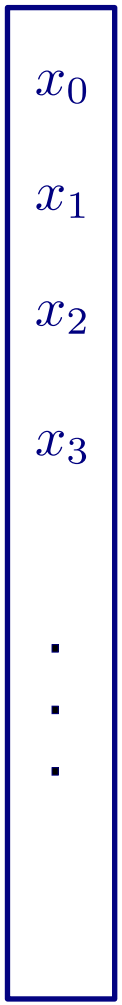
2020 - 2021

Outline

Outline

Back to image classification

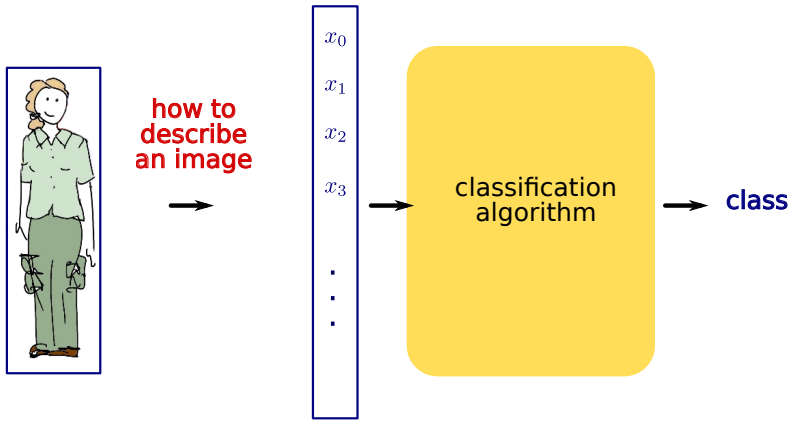
how to describe an image



classification algorithm

class

Image classification



- image description
  - fixed size
  - SIFT, SURF, HOG + BoW
  - CNN (VGG, AlexNet, ResNet, ...)
- classification algorithm
  - SVM
  - fully connected layers (ANN)
  - random forest
  - logistic regression

image description by CNN

- CNN trained with fixed size images for fixed image descriptor
  - VGG16, VGG19, ResNet : 224x224
  - Inception V3, InceptionResNet V2, : 299x299
  - NasNet : 331x331
- need for image transformation
  - resize/warp, crop, padding

- convolution does not need fixed size images
  - Spatial Pyramid Pooling (SPP) layer after convolution and before classification `https://arxiv.org/pdf/1406.4729.pdf`

Pyramid Pooling layer (from He etal TPAMI 2015)

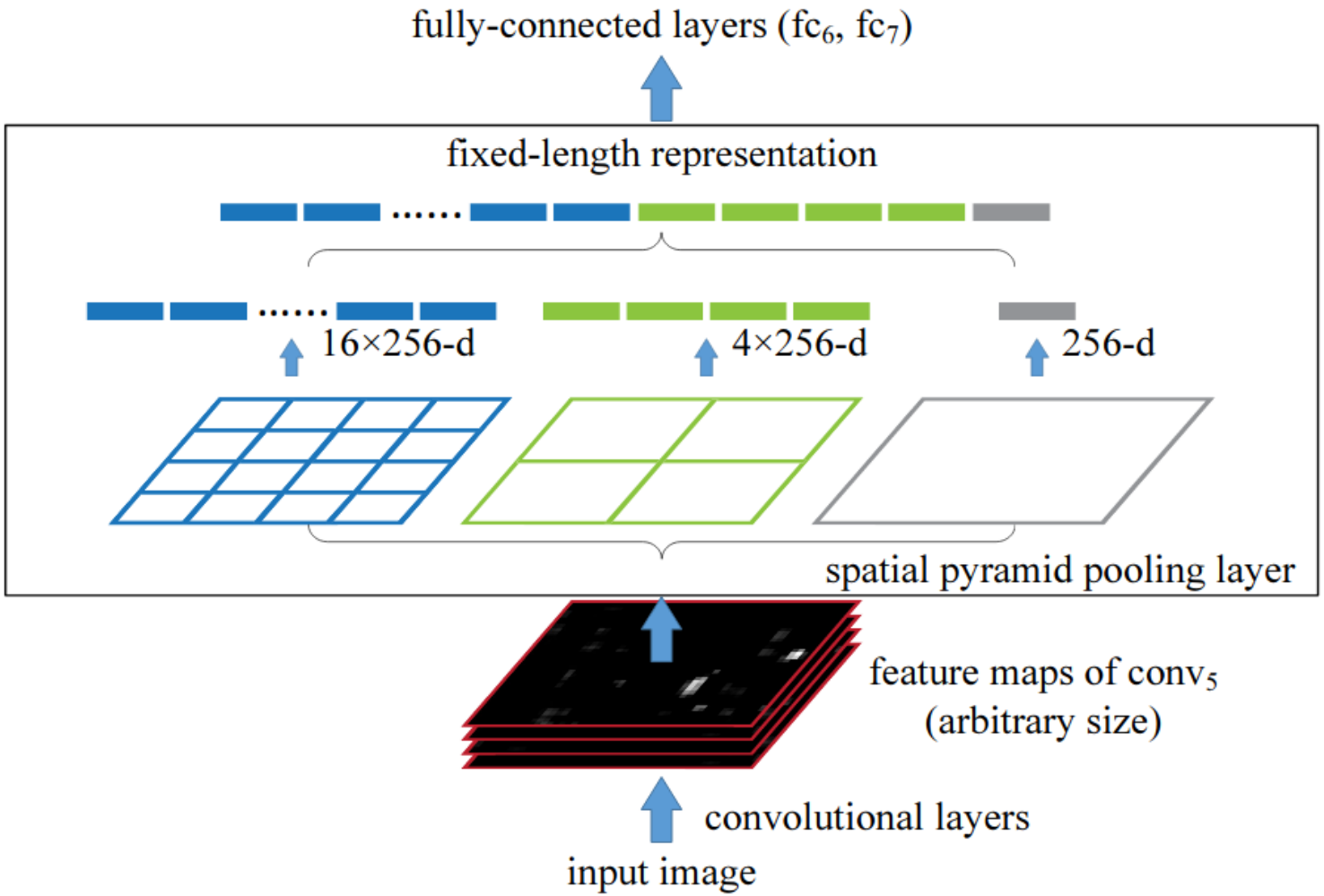
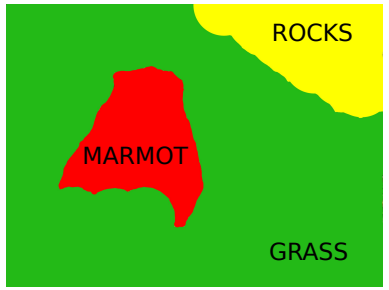
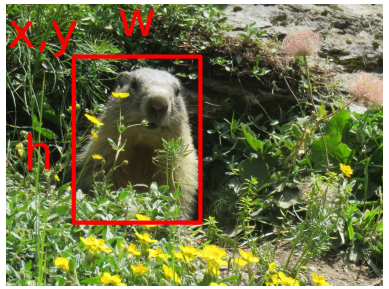


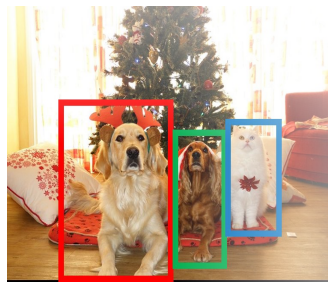
Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

Detection

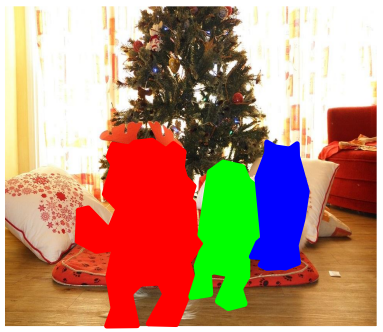
- Semantic segmentation :
  - label each pixel of the image
- Classification and Localisation : find one object in an image
  - label, position, size
- Object detection :
  - find all objects in an image
  - for each object : label, position, size
- Instance segmentation
  - segment all objects in an image :
  - label each pixel of the image. Labels for 2 different objects are different







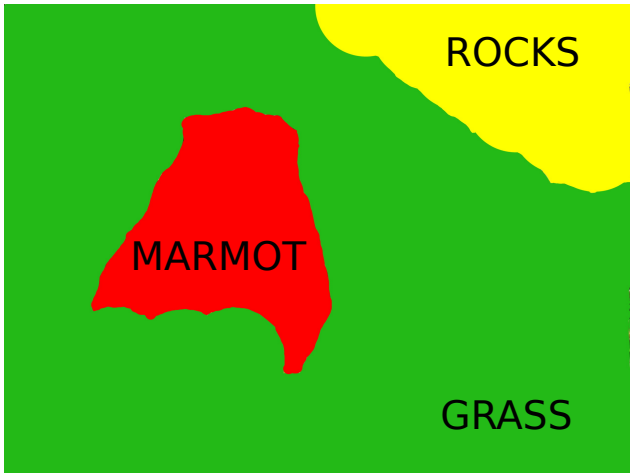
DOG DOG CAT



DOG DOG CAT

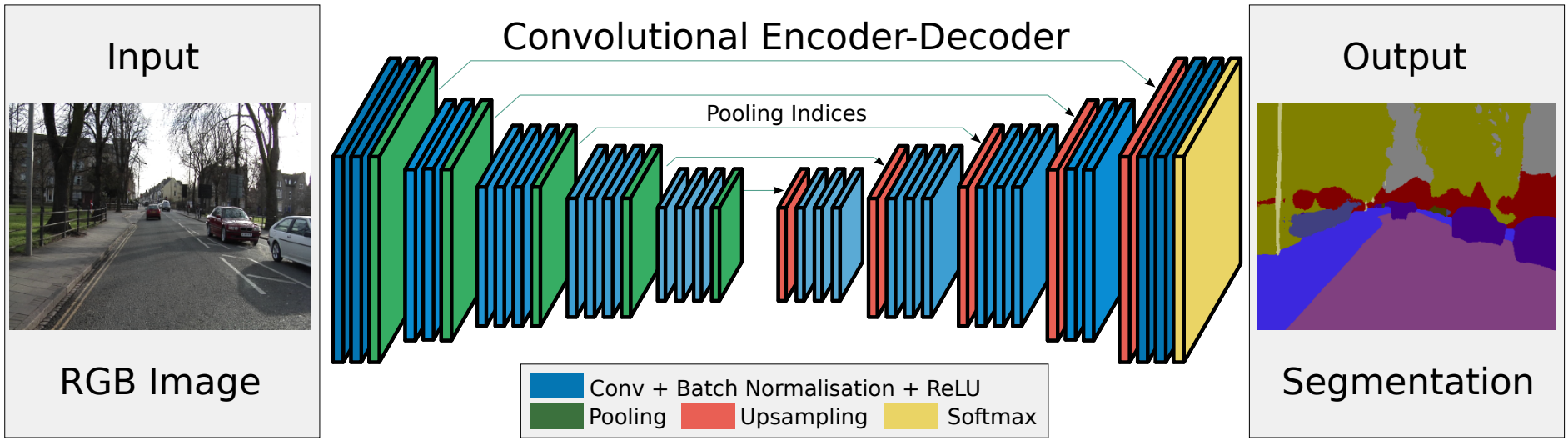
Outline

Semantic segmentation

From  to 

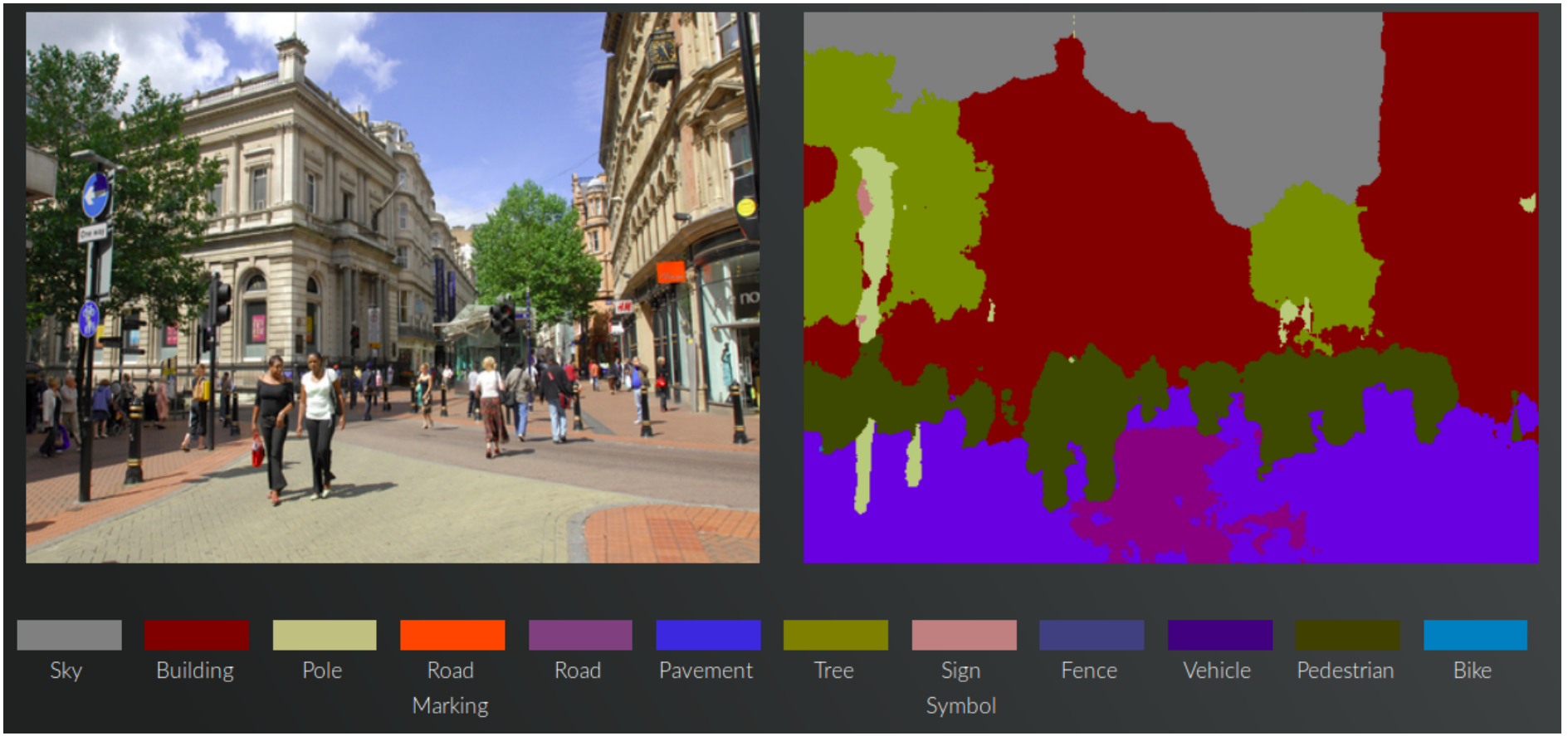
Convolution Autoencoder

- simply replace dense layers by convolutional layers
  - in keras, replace `Dense` by `Conv2D`
- application to segmentation : SegNet, 2015



from Badrinarayanan etal, arXiv :1511.00561v3

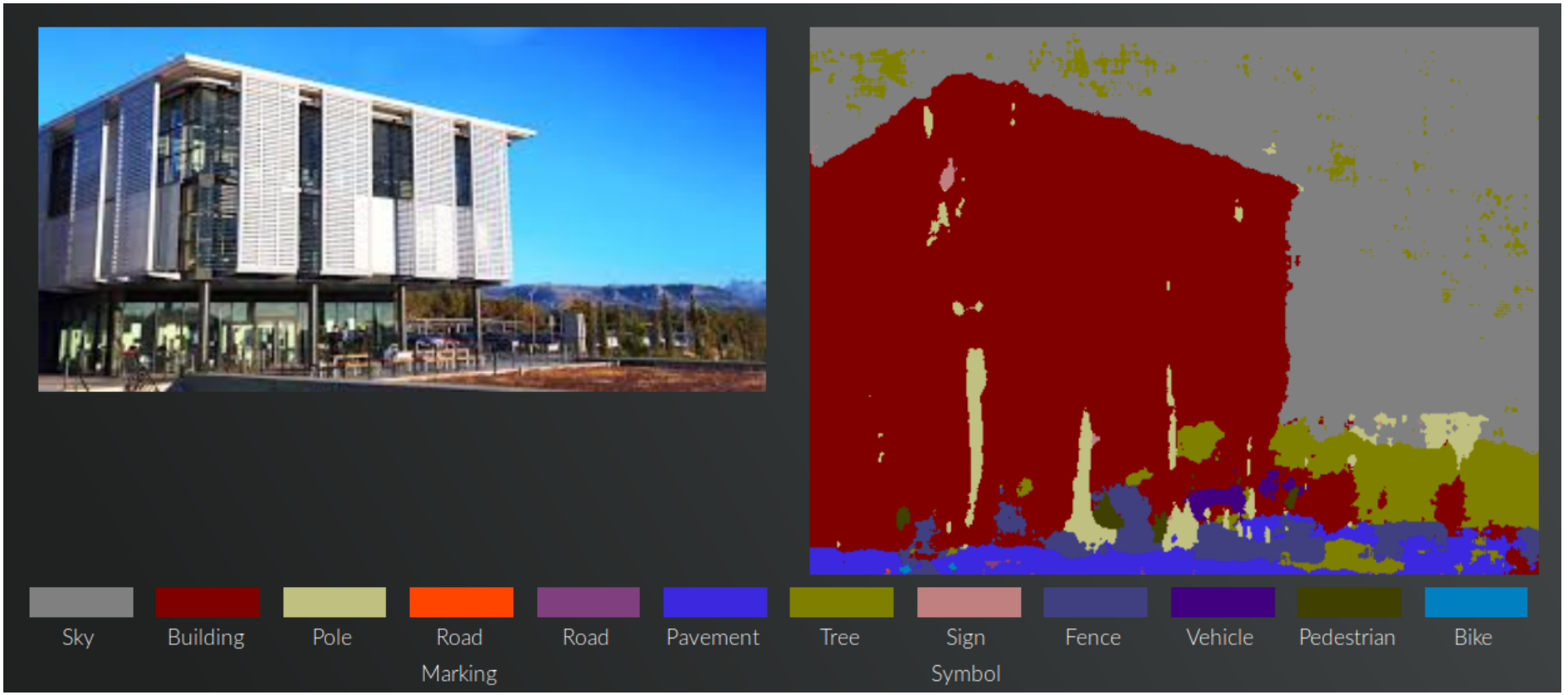
SegNet example I



also on youtube : `https://www.youtube.com/watch?v=CxanE_W46ts`

SegNet example II

try it yourself :
`http://mi.eng.cam.ac.uk/projects/segnet/demo.php`

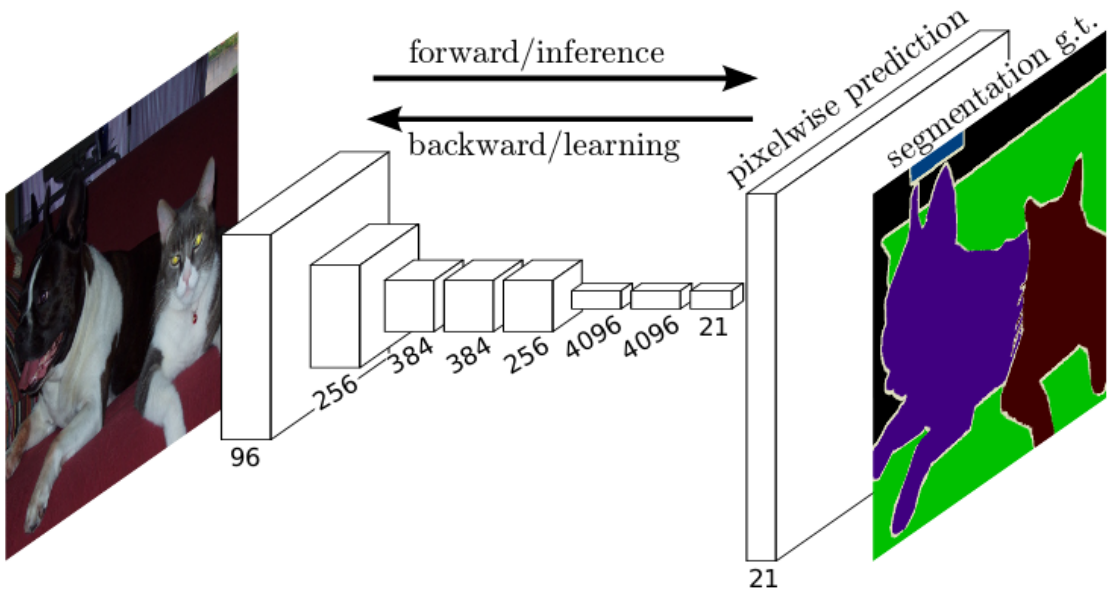


Fully Convolutional Networks (FCN), 2015

Fully Convolutional Networks for Semantic Segmentation

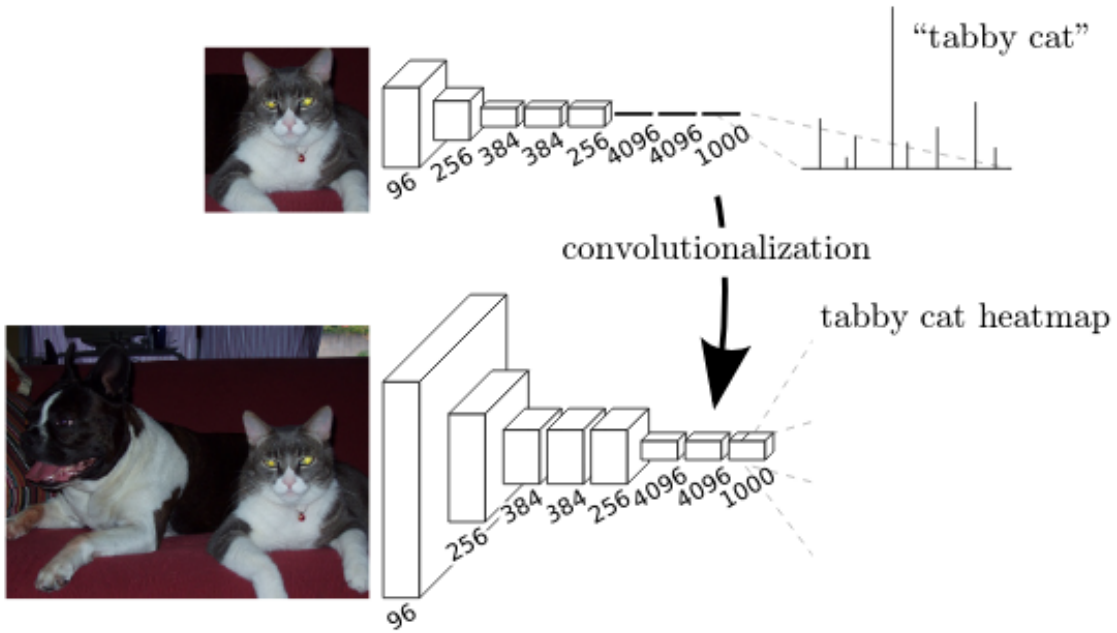
Jonathan Long* Evan Shelhamer* Trevor Darrell
UC Berkeley

https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf



FCN : extending a convnet to arbitrary-sized inputs

- CNN :
  - fixed input size
  - non spatial output (class)
  - image description after conv./pooling layers
- any size
  - only for the conv./pooling layers of CNN



upsampling : transposed convolution

- Example : gaussian filter, dim 3x3, stride 1, paddind VALID (discard borders) applied one a 4x4 grey levels image :

120	110	80	84
105	80	60	70
110	90	54	62
112	94	62	44

$*$

1	2	1
2	4	2
1	2	1

$/16 \Rightarrow$

88.375	72.125
90	64.5

  - From

88.375	72.125
90	64.5

and the filter, how to recover the 4x4 image ?

upsampling : transposed convolution (2)

- also called *backwards convolution* or *deconvolution*
- rewrite the convolution operation :

$$\underbrace{\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 & 0 & 2 & 4 & 2 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 2 & 4 & 2 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 2 & 4 & 2 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 2 & 4 & 2 & 0 & 1 & 2 & 1 \end{bmatrix}}_{C} \cdot \underbrace{\begin{bmatrix} 120 \\ 110 \\ 80 \\ 84 \\ 105 \\ 80 \\ 60 \\ 70 \\ 110 \\ 90 \\ 54 \\ 62 \\ 112 \\ 94 \\ 62 \\ 44 \end{bmatrix}}_{I_1} = \underbrace{\begin{bmatrix} 88.375 \\ 72.125 \\ 90 \\ 64.5 \end{bmatrix}}_{I_2}$$

- transpose the matrix C :

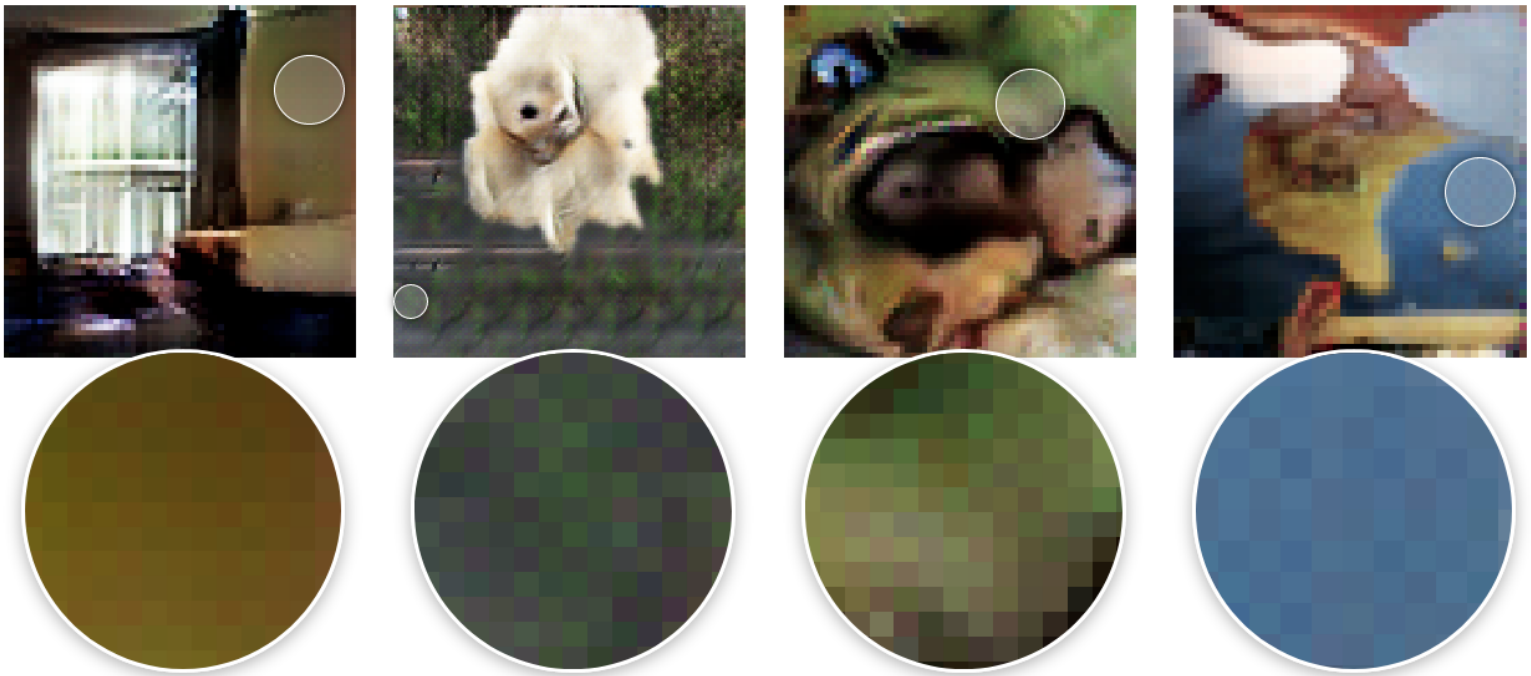
$$I_1 = C^T I_2$$

  - C : not the same coefficients but learnable
  - transpose convolution : upsampling
  - checkerboard artifacts

transposed convolution artefacts

From `https://distill.pub/2016/deconv-checkerboard/`

> When we look very closely at images generated by neural networks, we often see a strange checkerboard pattern of artifacts. It's more obvious in some cases than others, but a large fraction of recent models exhibit this behavior.



Radford, et al., 2015 [1] Salimans et al., 2016 [2] Donahue, et al., 2016 [3] Dumoulin, et al., 2016 [4]

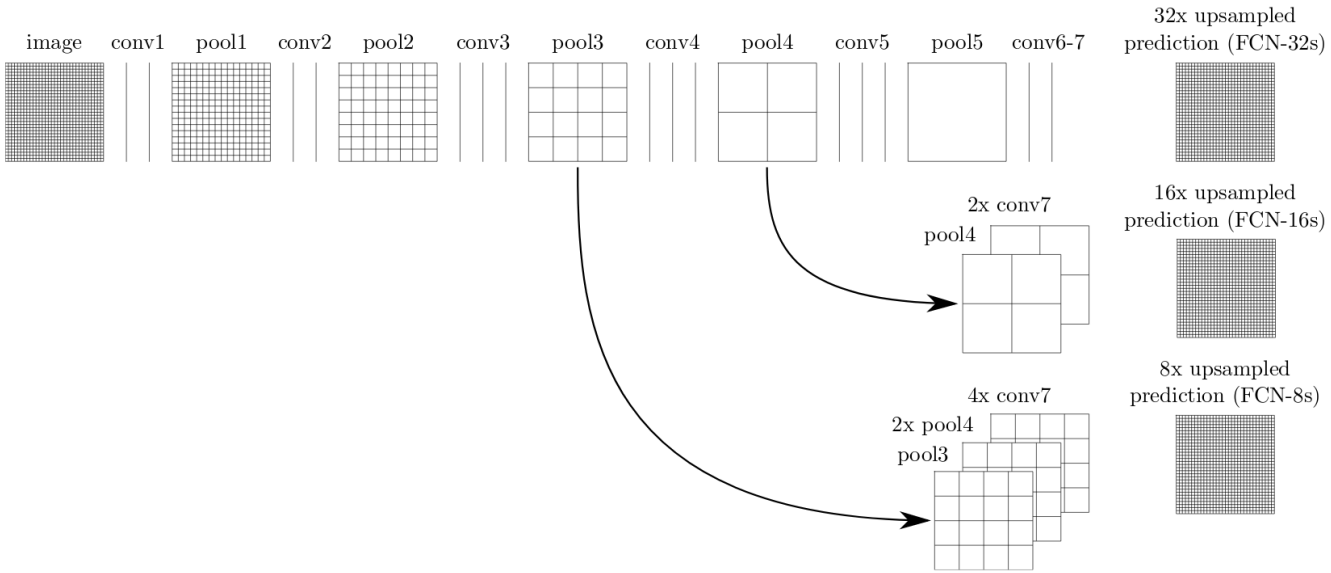


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic

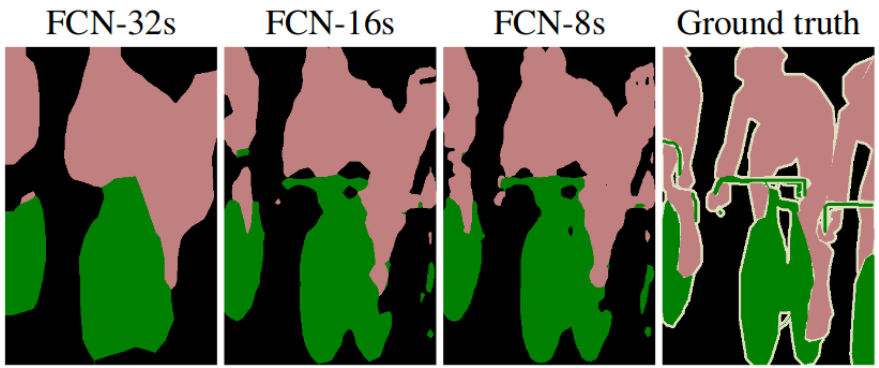


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

U-Net

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

https://arxiv.org/pdf/1505.04597.pdf

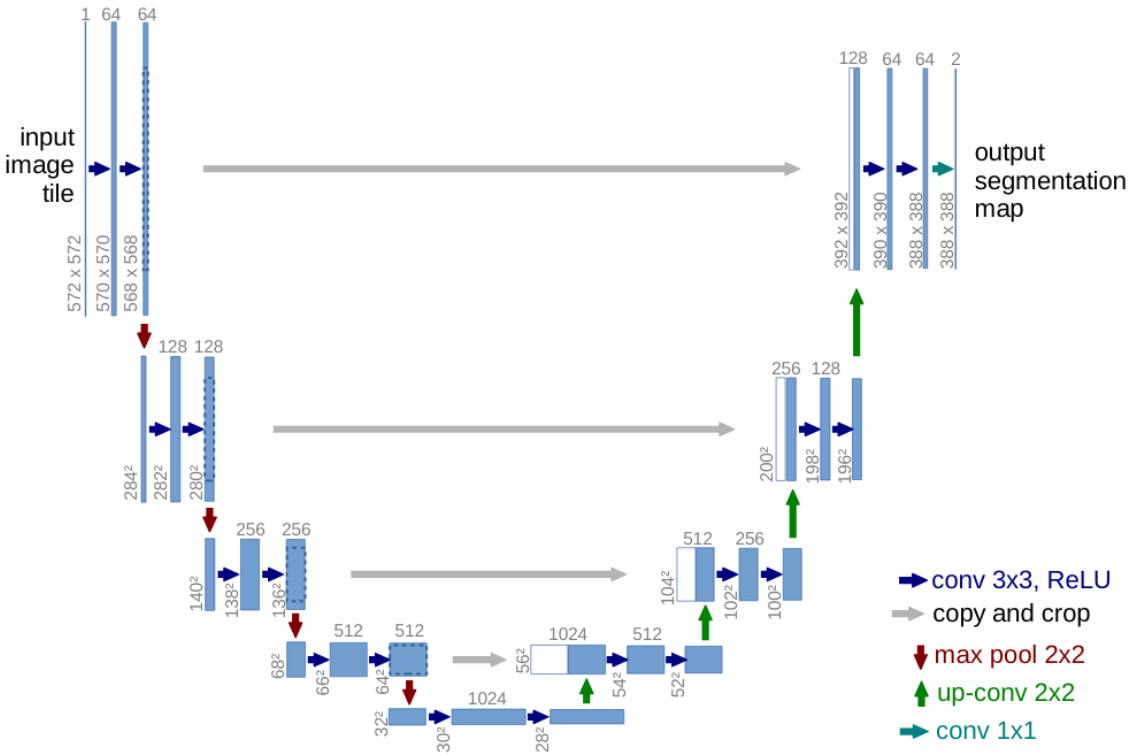


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations

PSPNet Pyramid Scene Parsing Network (2016-17)

Pyramid Scene Parsing Network

Hengshuang Zhao[1] Jianping Shi[2] Xiaojuan Qi[1] Xiaogang Wang[1] Jiaya Jia[1]
[1]The Chinese University of Hong Kong [2]SenseTime Group Limited

https://arxiv.org/pdf/1612.01105.pdf

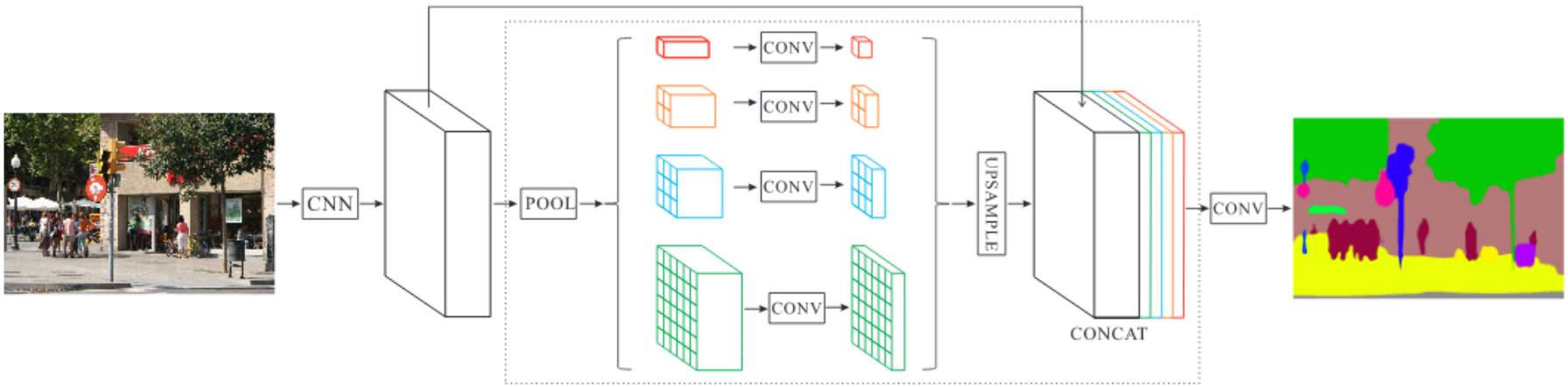
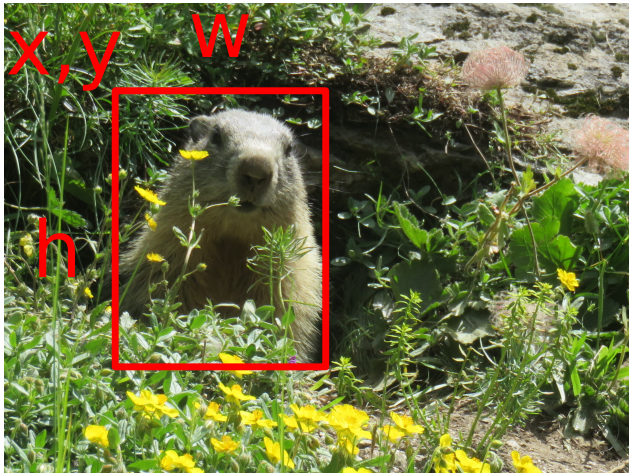
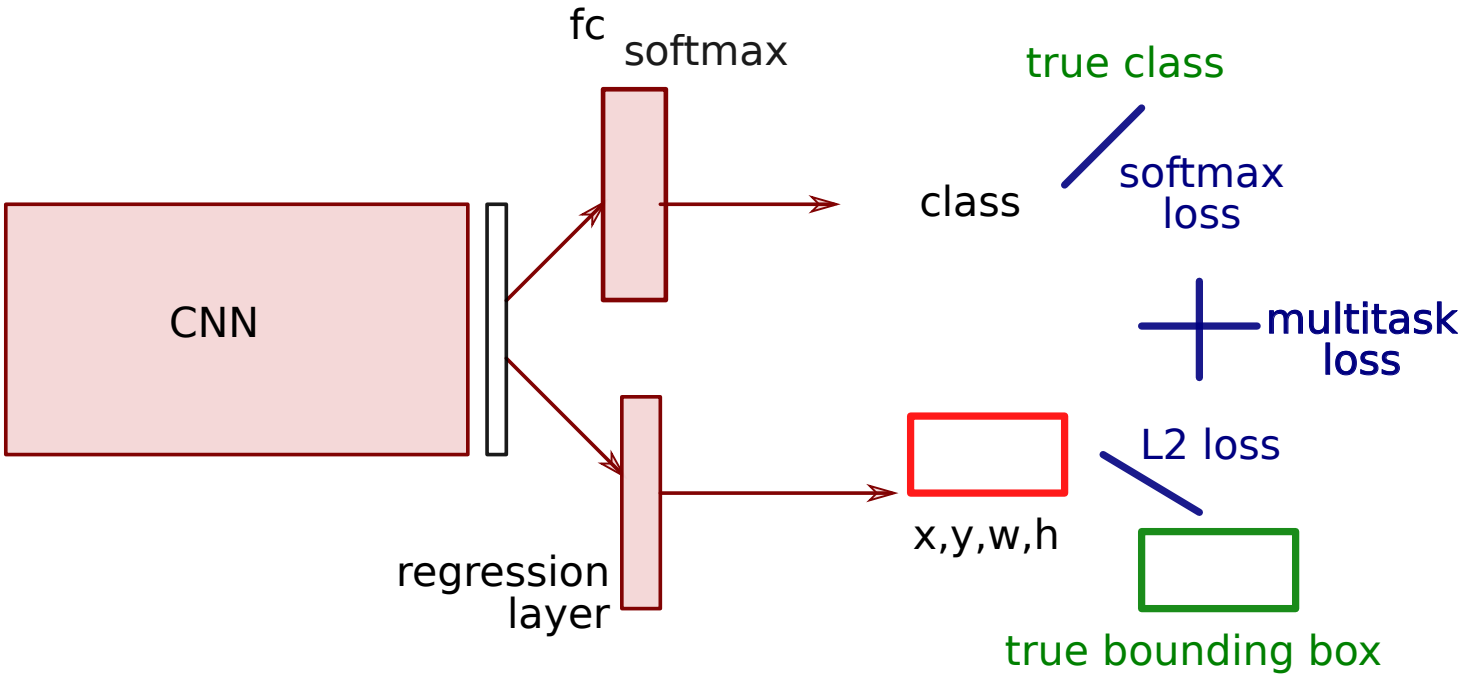


Figure 3. Overview of our proposed PSPNet. Given an input image (a), we first use CNN to get the feature map of the last convolutional layer (b), then a pyramid parsing module is applied to harvest different sub-region representations, followed by upsampling and concatenation layers to form the final feature representation, which carries both local and global context information in (c). Finally, the representation is fed into a convolution layer to get the final per-pixel prediction (d).

Outline

Classification and Localisation





Outline

Object detection

- Focus on some classes and add the *background* class
- You don't know how many objects you expect to find
- First idea : sliding window
  - huge number of locations and scales
- Region proposals
  - using basic image processing (edges,...)
  - select a "small" numbers of windows (around 2000)
  - most windows are noisy windows
  - Then apply classification on each window

Slow R-CNN

Regions of Interest (RoI) from a proposal method (~2k)

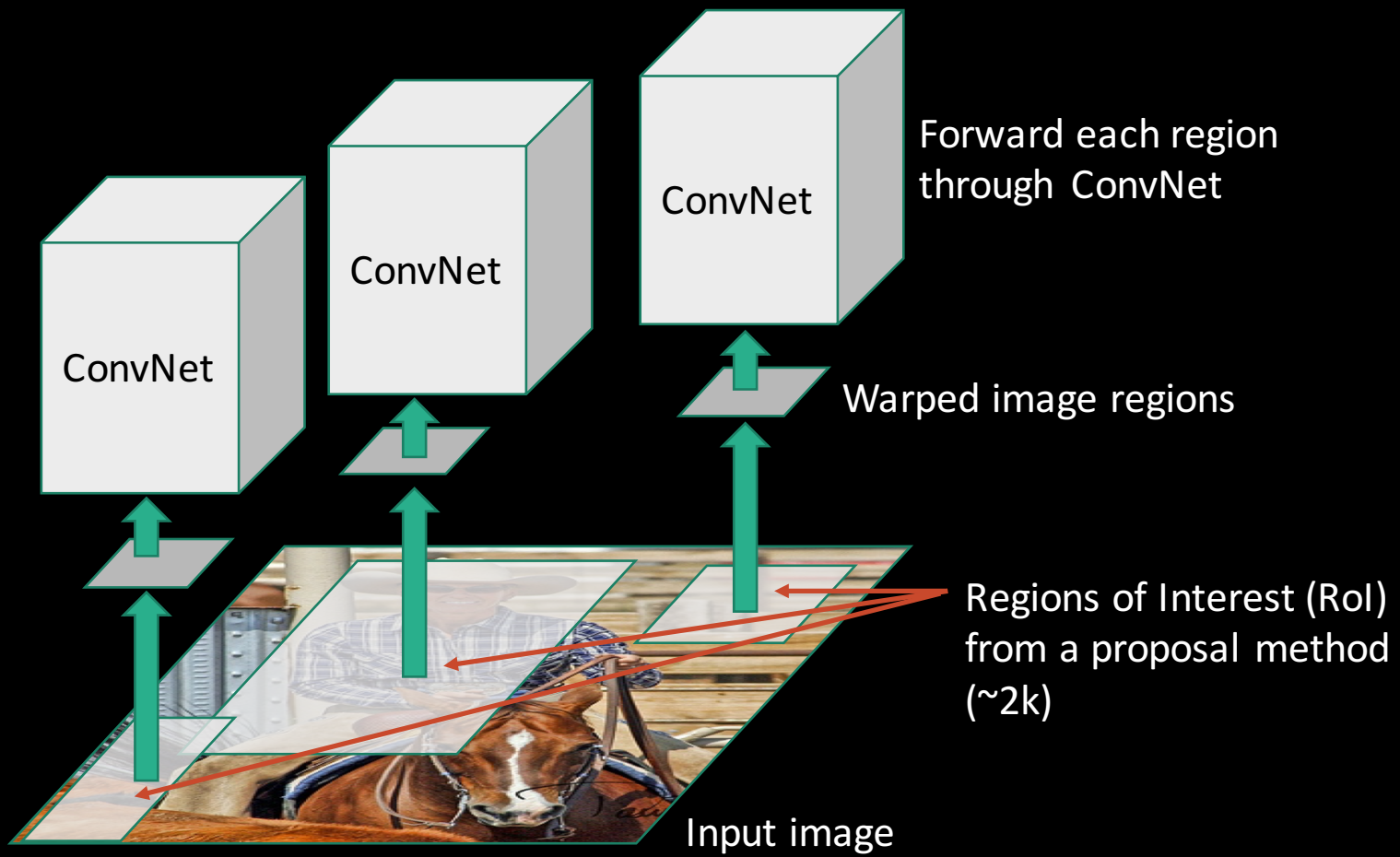
Input image

Girshick et al. CVPR14.

Slow R-CNN

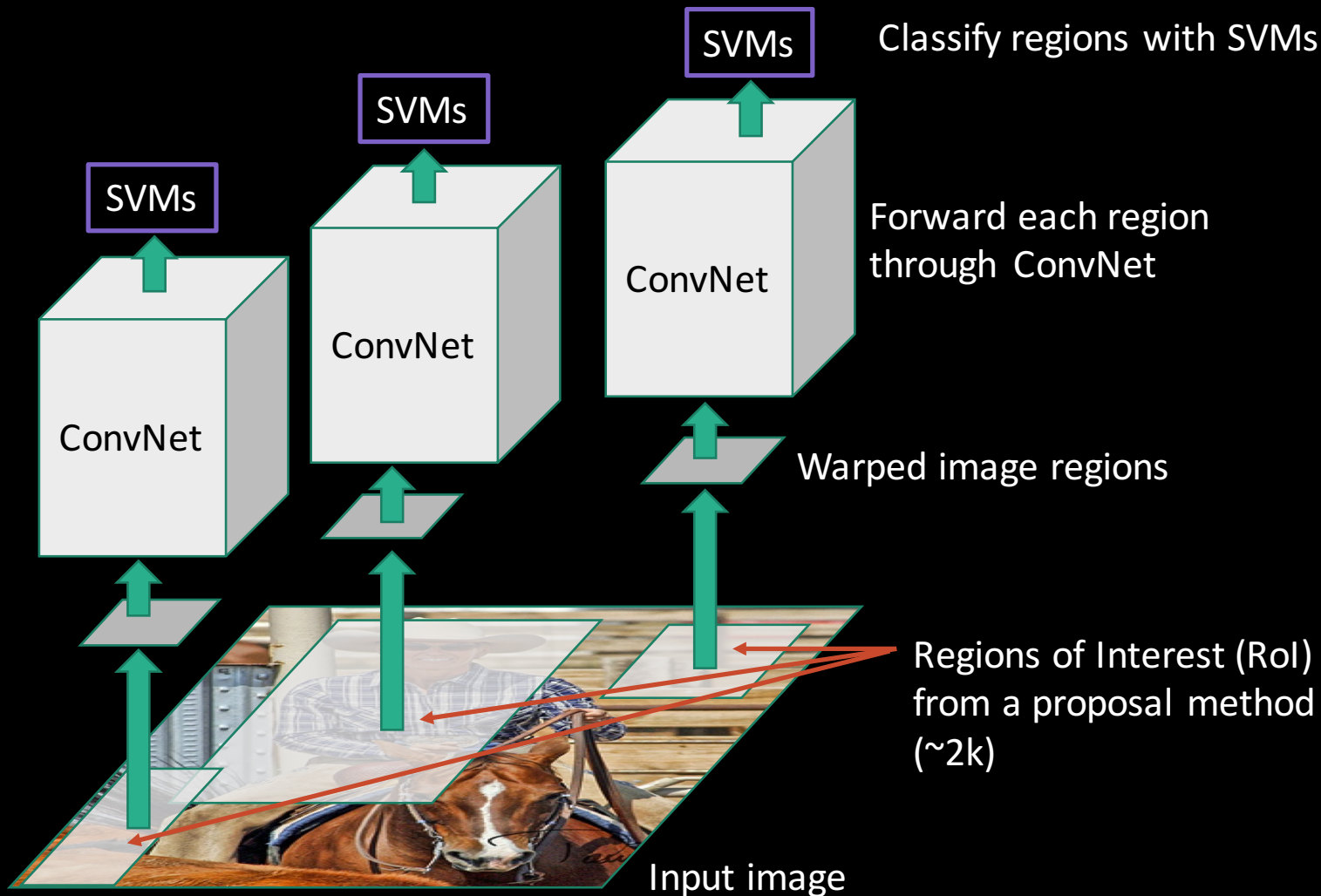
Girshick et al. CVPR14.

Slow R-CNN



Girshick et al. CVPR14.

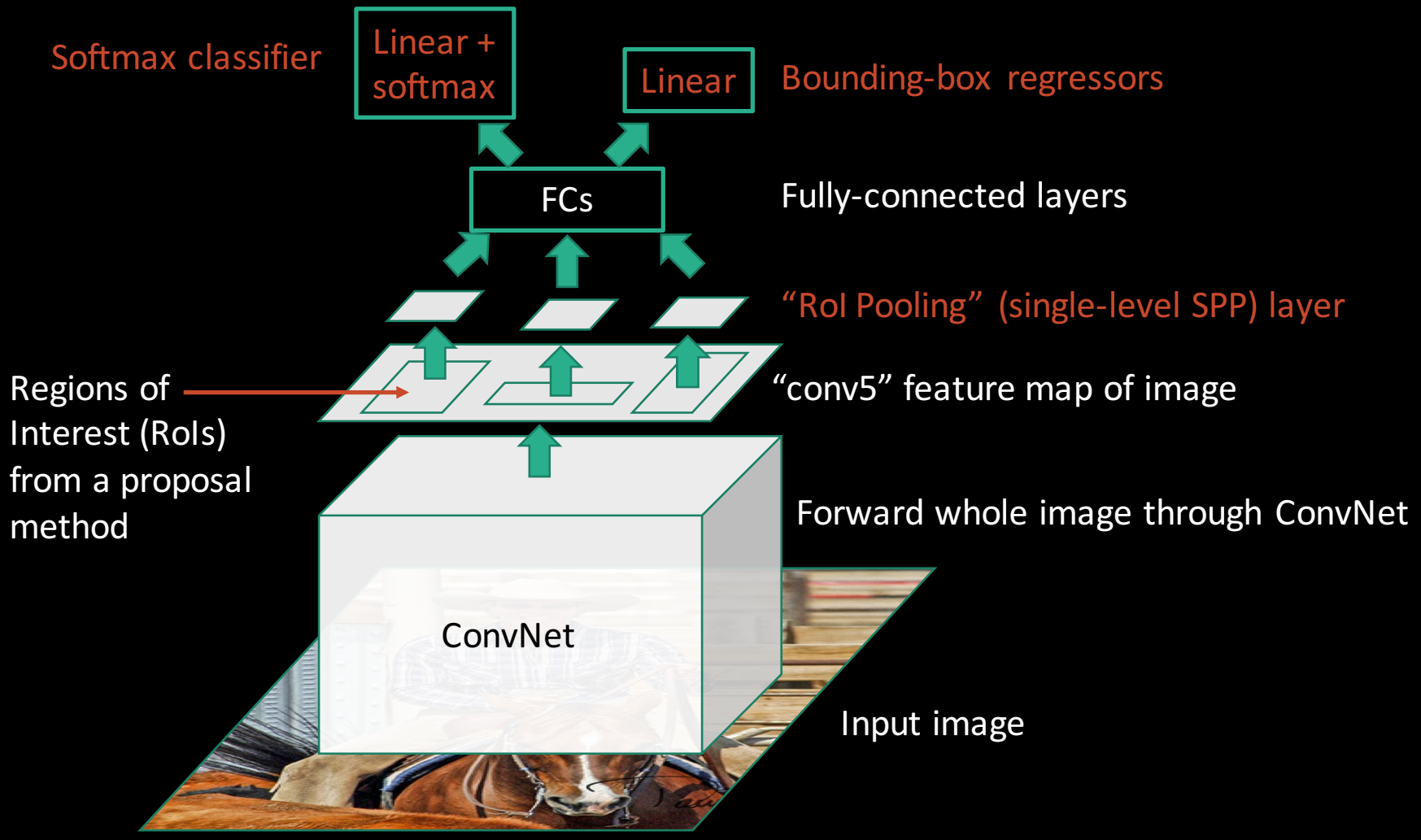
Slow R-CNN



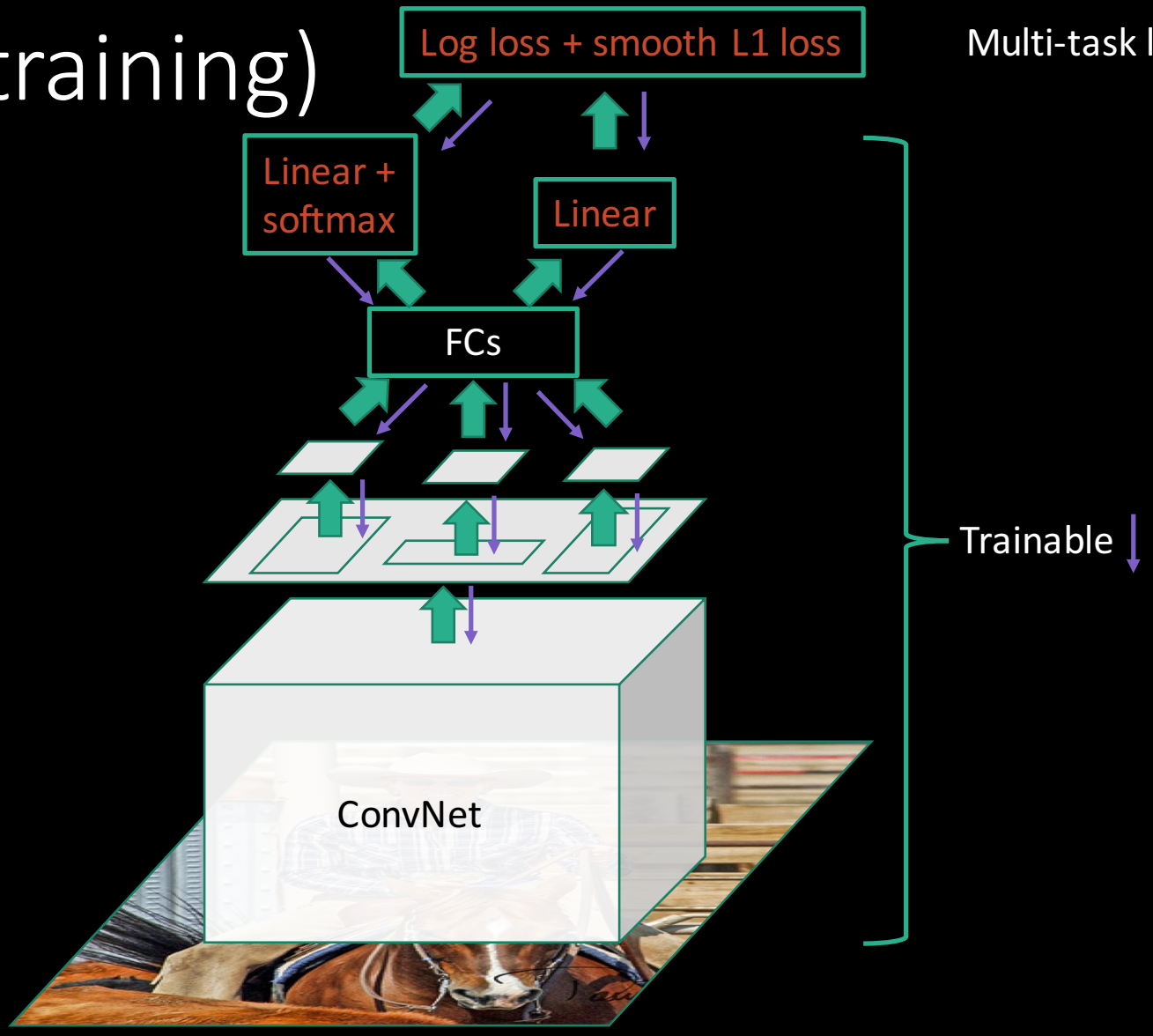
Girshick et al. CVPR14.

Post hoc component

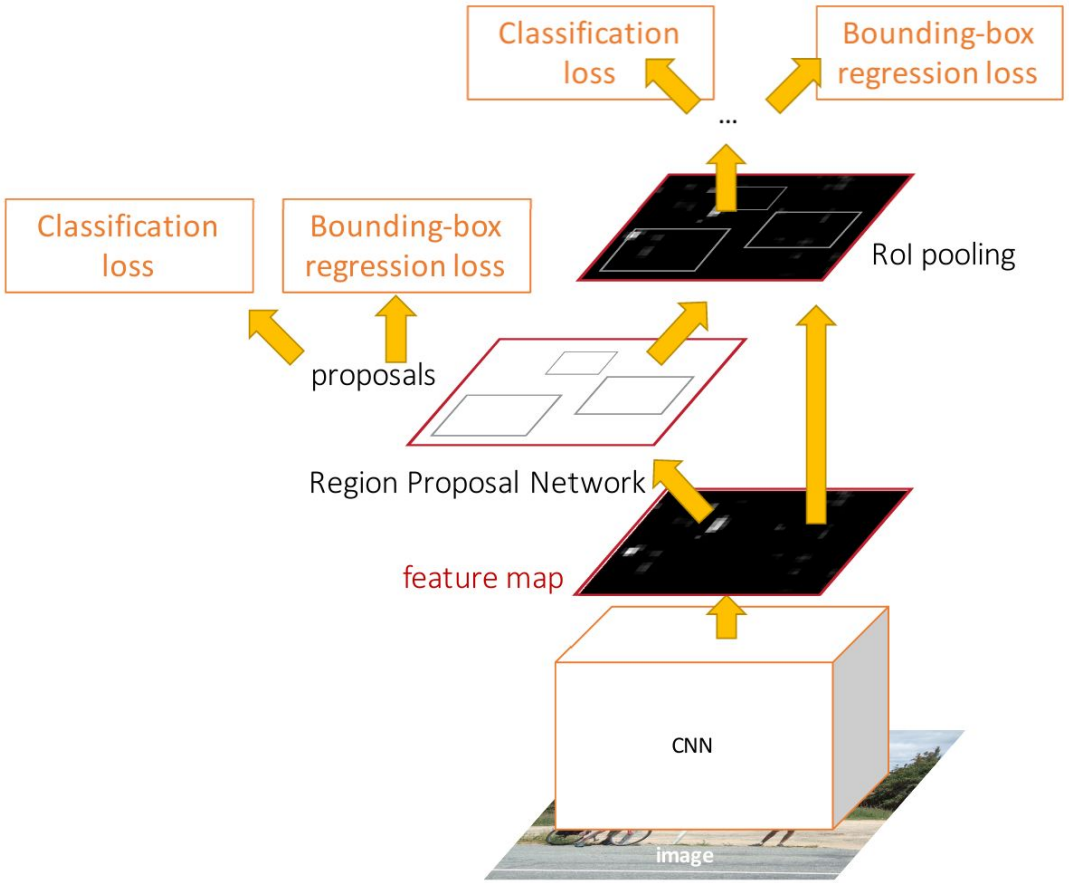
Fast R-CNN (test time)



Fast R-CNN (training)



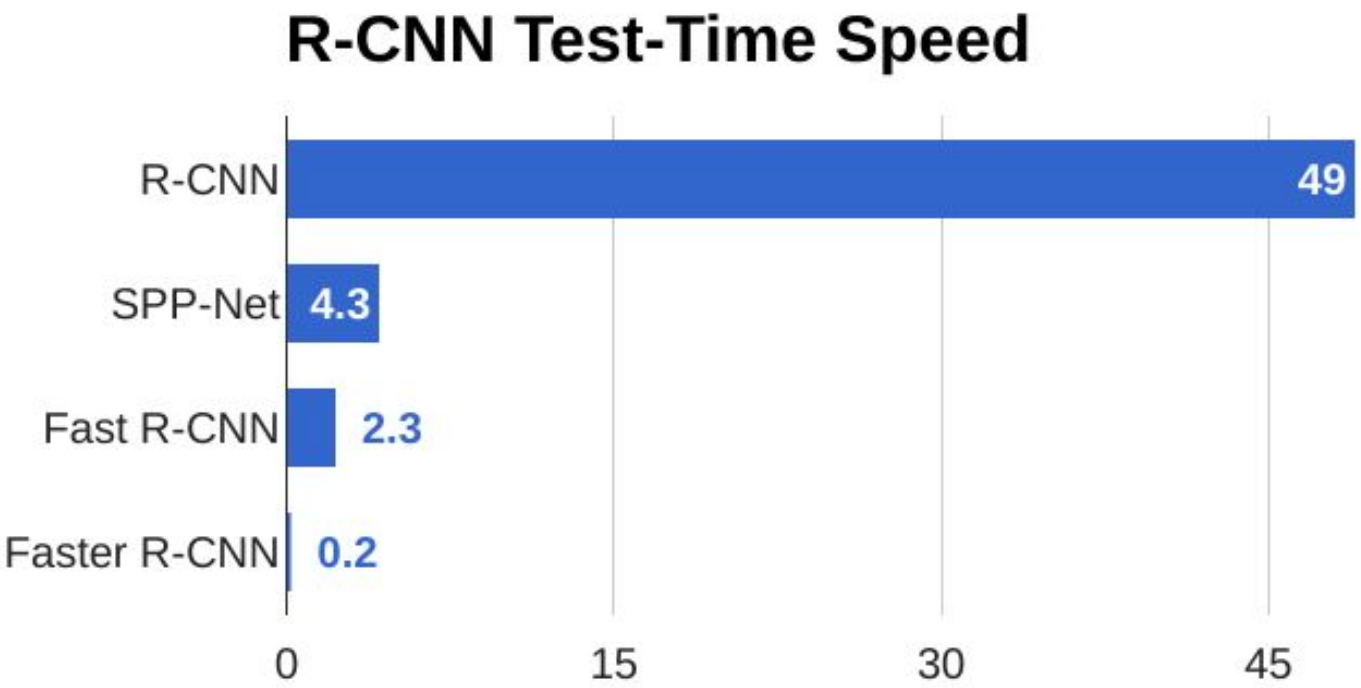
Faster R-CNN : Let CNN make proposals



from `https://arxiv.org/pdf/1506.01497.pdf`

- Insert Region Proposal Network (RPN) to predict proposals from features
- Jointly train with 4 losses :
  - 1. RPN classify object / not object
  - 2. RPN regress box coordinates
  - 3. Final classification score (object classes)
  - 4. Final box coordinates

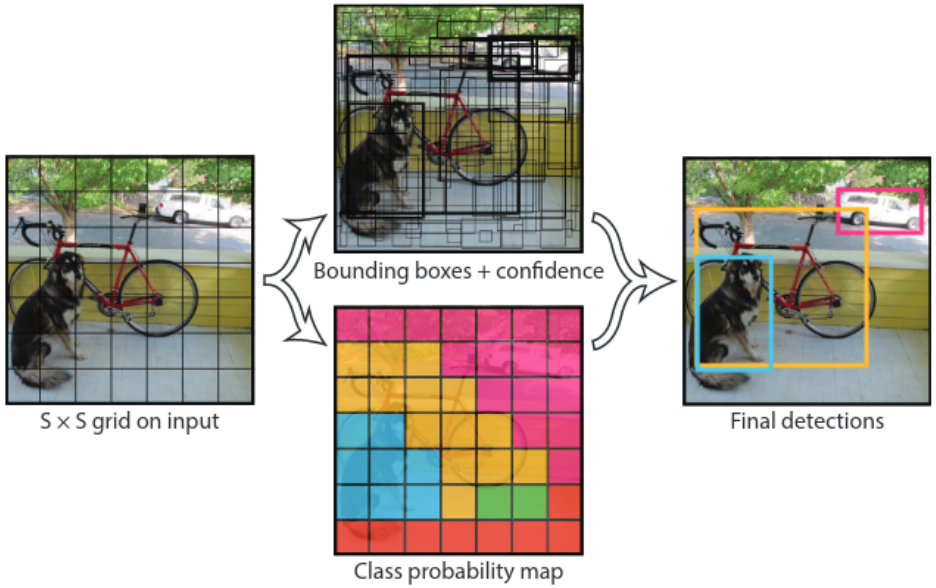
Faster R-CNN


R-CNN Test-Time Speed
R-CNN
SPP-Net
Fast R-CNN
Faster R-CNN
49
4.3
2.3
0.2
0
15
30
45

Faster methods

- two stage methods :
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- one stage methods :
  - YOLO (v1, v2, v3, v9000)
  - SSD
  - RetinaNet

YOLO : You Only Look Once

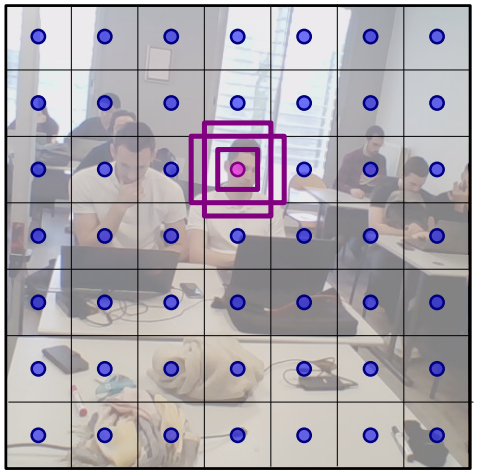


from `https://arxiv.org/pdf/1506.02640.pdf`

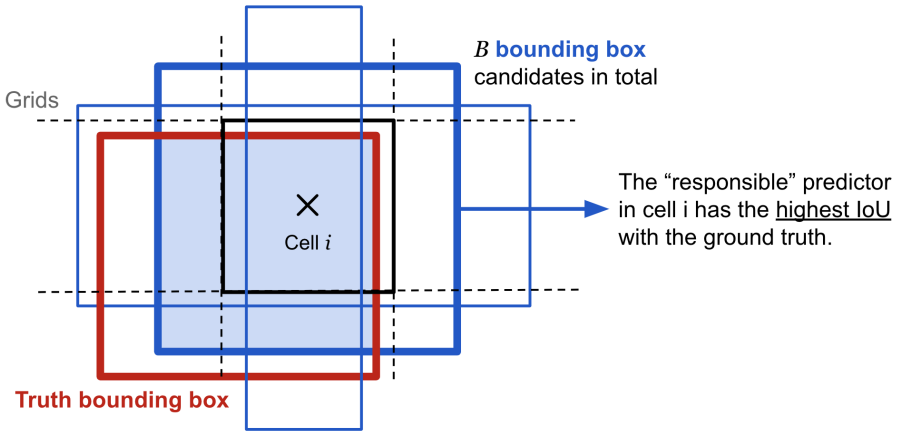
- image resized to a fixed size 448x448
- image divided into a SxS grid (e.g. 7x7)
- for each grid :
  - predict B bounding boxes (e.g. B=2) initially centered on the grid, each box with confidence, and C class probabilities (C = number of classes+bg)
  - neural network with 24 conv. layers and 2 fully connected layers
- all predictions in a tensor of dimensions $SxSx(B*5+C)$

YOLO : anchors and boxes

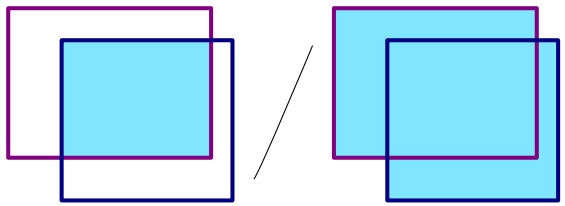
- anchors are initial guesses for bounding boxes (S=7, B=3)



- for training : only one bounding box predictor is responsible for each objet based on the highest IoU with the ground truth.



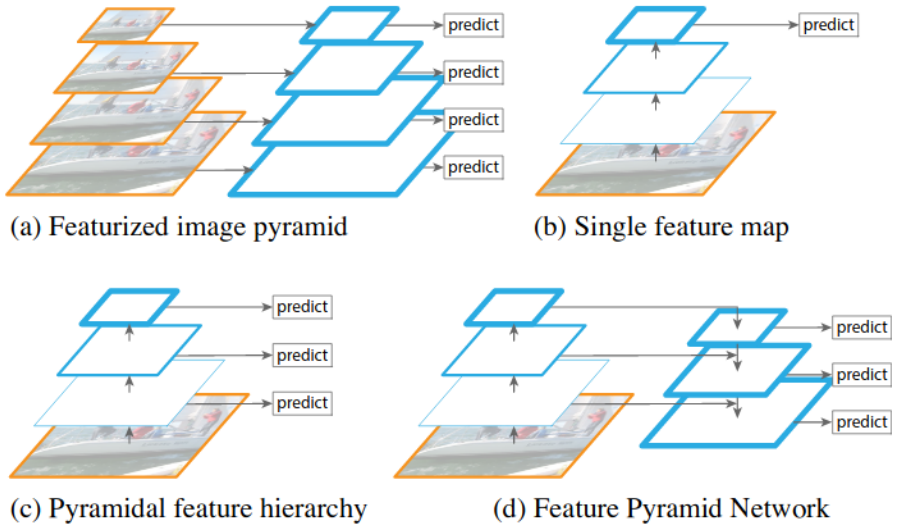
IoU = Intersection over Union



YOLO improvements

- YOLO v2
  - DarkNet (30 layers)
  - increase the mAP (mean Average Precision) and small object detection
- YOLO v3
  - Darknet-53 (2*53 conv. layers using skip connections)
  - Feature Pyramid : 3 scales (32, 16, 8)
  - multilabel classification (no more softmax) for non exclusive classes
  - demo at `https://pjreddie.com/darknet/yolo/`
- YOLO 9000
  - trained using 9000 classes

Feature Pyramid Network



(a) Featurized image pyramid (b) Single feature map

(c) Pyramidal feature hierarchy (d) Feature Pyramid Network

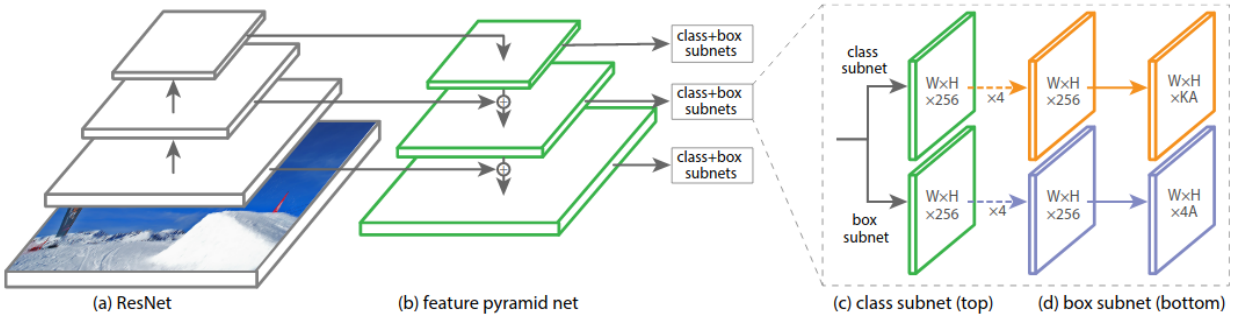
from `https://arxiv.org/pdf/1612.03144.pdf`

- (a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow.
- (b) Single scale features for faster detection.
- (c) An alternative is to reuse the pyramidal feature hierarchy computed by a ConvNet as if it were a featurized image pyramid.
- (d) FPN is fast like (b) and (c), but more accurate. In this figure, feature maps are indicate by blue outlines and thicker outlines denote semantically stronger features.

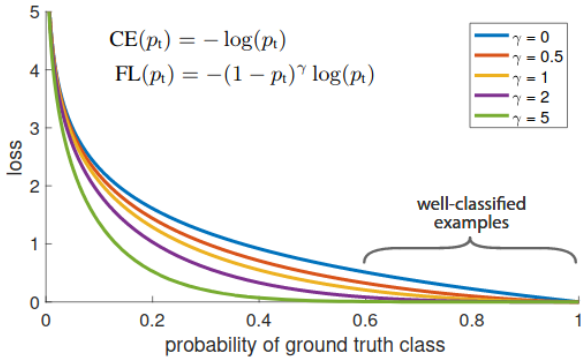
SSD : Single Shot Detector

- algorithm similar to YOLO :
  - single pass
  - anchors
  - prediction of classes and bounding boxes
- but :
  - starts with VGG 16 + conv. filters
  - independent object detections from multiple feature maps (different resolutions)

RetinaNet



- deal with unbalanced classes : background vs objets
- Focal loss : $-(1-p)^{\gamma}\log(p)$ with parameter $\gamma \geq 0$

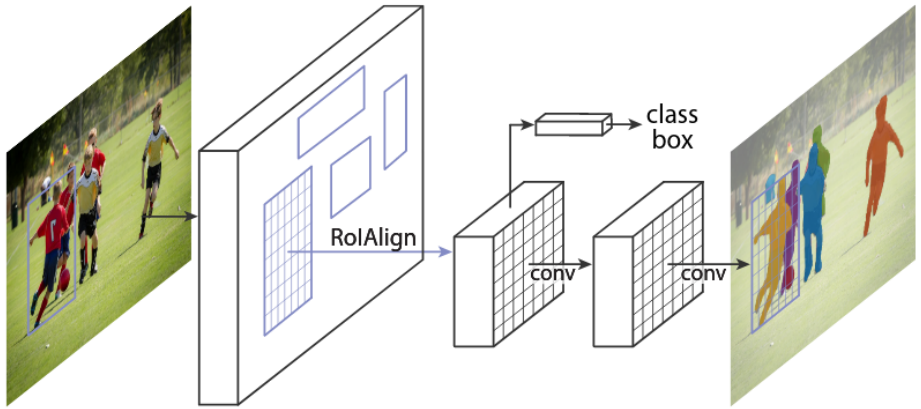


- if γ increases, the loss decreases for well-classified examples ($p > 0.5$) : more effort are done one difficult or missclassified example
- default value : $\gamma = 2$ (usually from 0.5 to 5)

illustrations from `https://arxiv.org/pdf/1708.02002.pdf`

Outline

Mask-RCNN



from `https://arxiv.org/pdf/1703.06870.pdf`

- algorithm
  - image features : CNN (ResNet) + FPN
  - region proposal : RPN + RoI Align
    - RoIAlign : align regions of the feature map with pixels of the image (interpolation)
  - in parallel :
    - classification of proposals (anchors)
    - bounding box and masks computation

Mask-RCNN : example from the paper

Mask-RCNN : example at Polytech