Processamento e Análise de Imagens

Deep Learning in Computer Vision

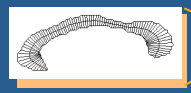
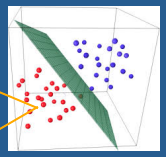
Prof. Alexei Machado

PUC Minas

Classical Computer Vision Process



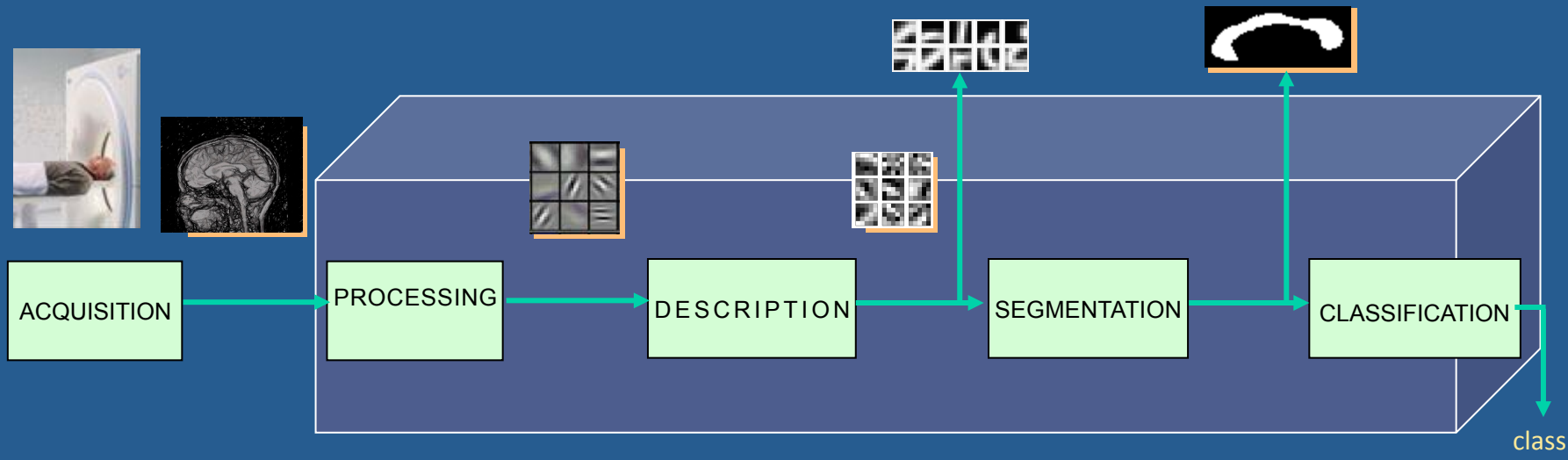


ACQUISITION → PROCESSING → SEGMENTATION → DESCRIPTION → CLASSIFICATION → class

Xray, CT, MRI, PET, US...	Noise reduction, edge and contrast enhancement...	Thresholding, Region growing, Active contour, Watershed...	Shape and Texture features, Wavelets, PCA...	Euclidean, Mahalanobis, Decision trees, SVM, NN…

Computer Vision with DL



Xray, CT, MRI, PET, US...

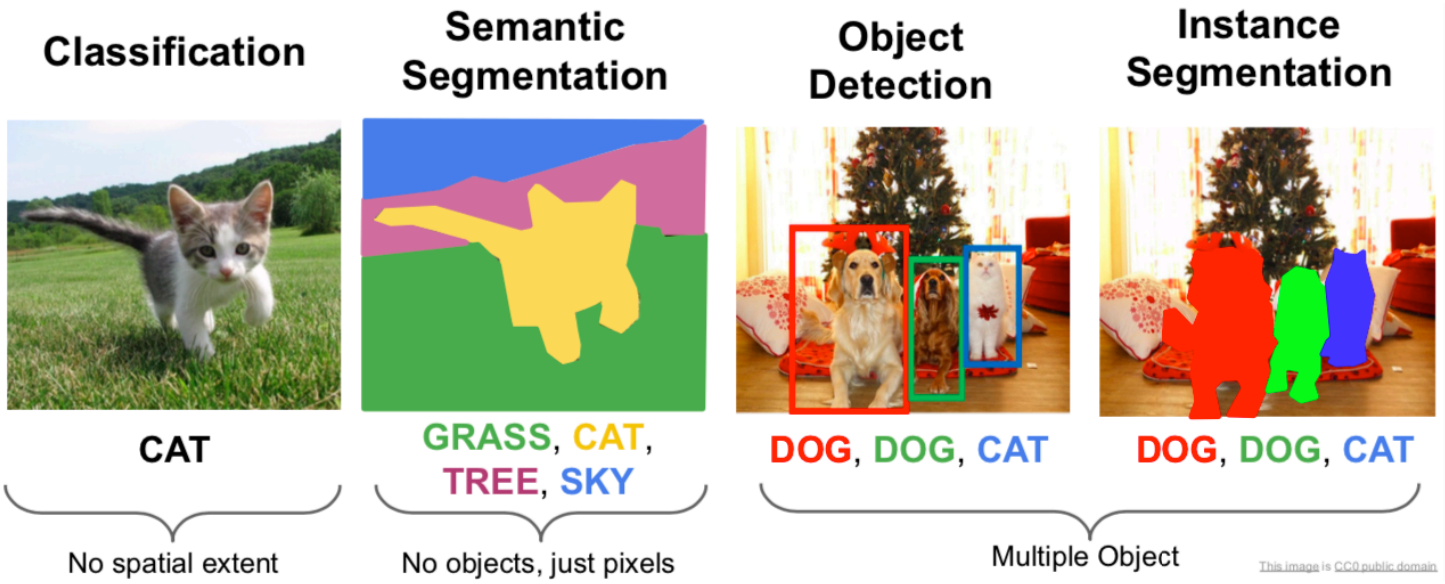
Convolutional layers

Bottleneck layers

Up-sampling layers

Fully-connected layers

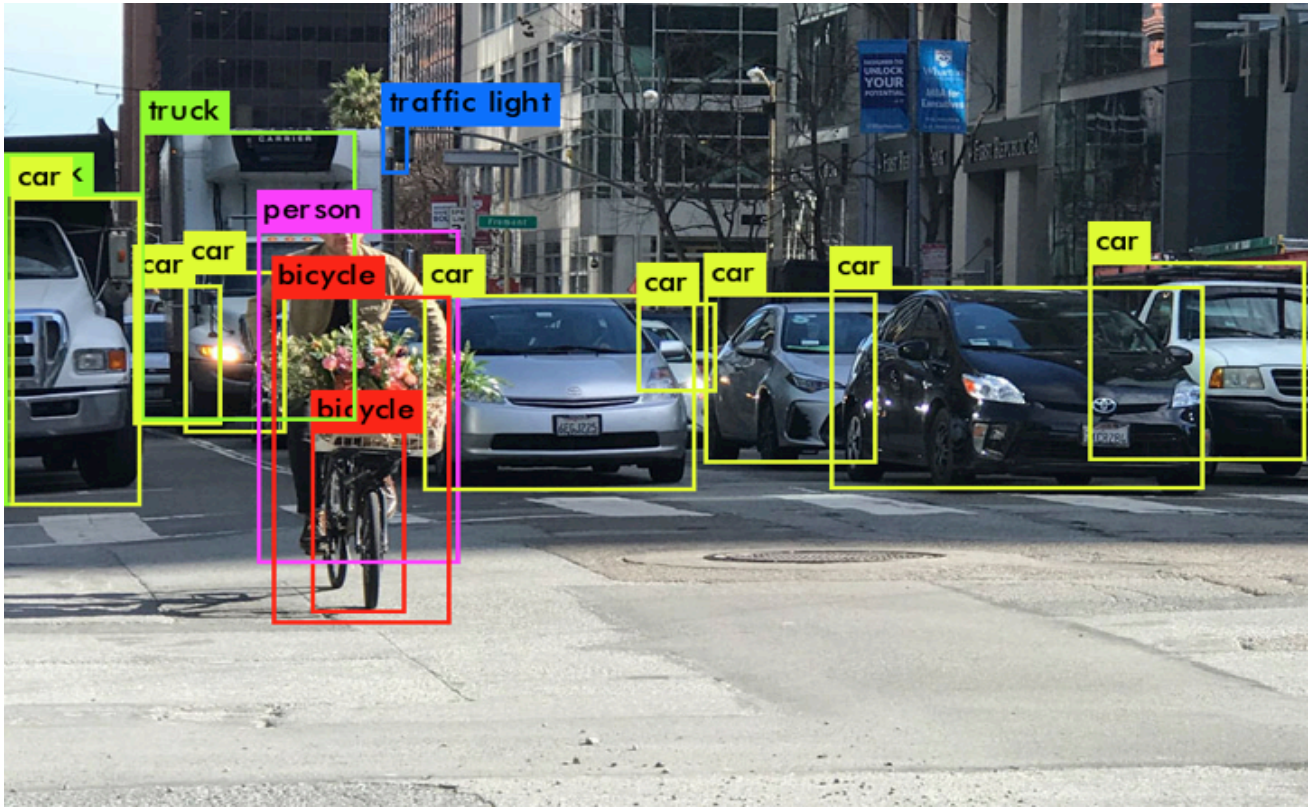
Computer Vision Tasks



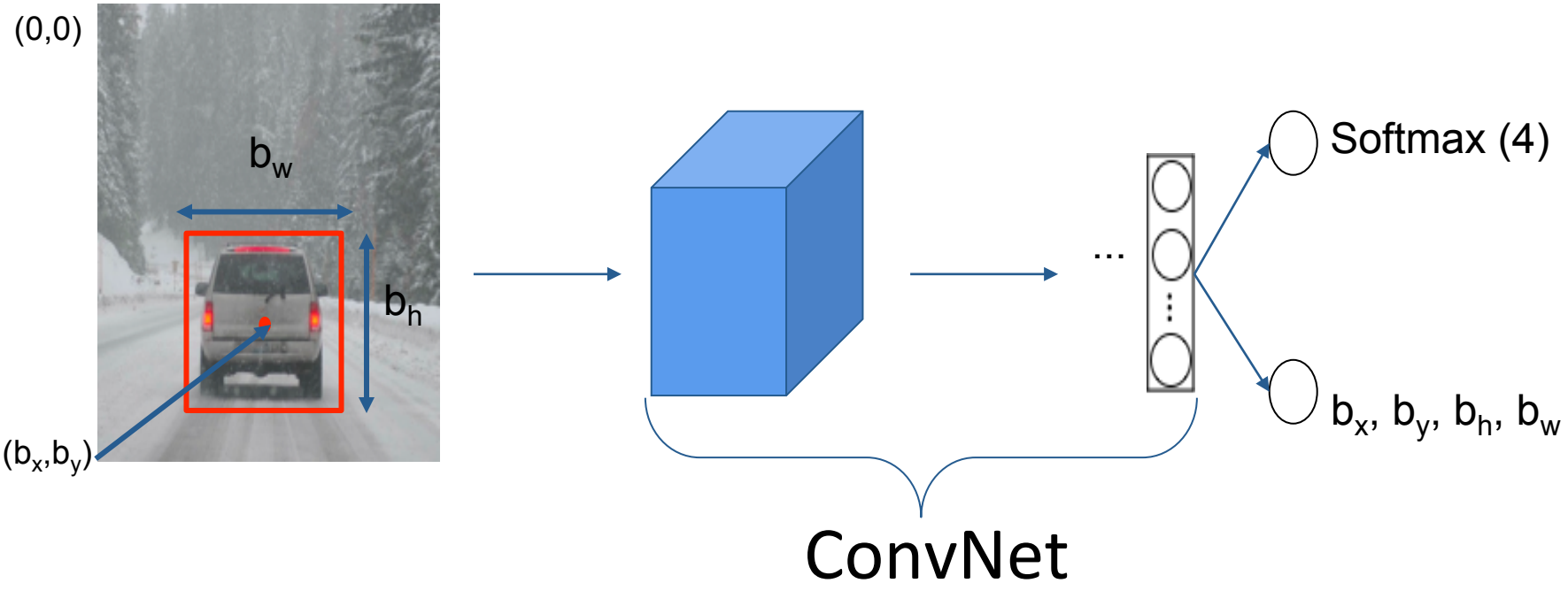
Computer Vision Tasks

- Classification: returns class associated with the image
- Semantic segmentation: associates each pixel with a class; does not distinguish between instances of the same class
- Object detection: returns class and location of each object in the image
- Instance segmentation: assigns each pixel to an instance and classifies the instance

Object Detection



Classification with Localization



Classes

1. Pedestrian
2. Car
3. Motorcycle
4. Background

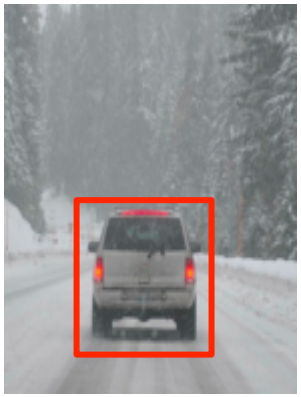
Bounding box

b_x= 0.5
b_y = 0.7
b_h = 0.3
b_w = 0.4

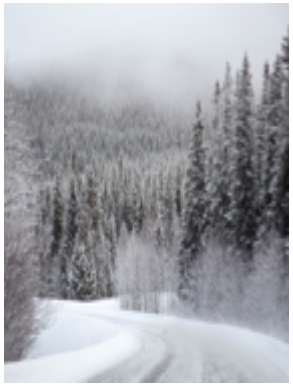
Classification with Localization

Return: b_x, b_y, b_h, b_w, class label (1-4)

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$ → Is there an object?

x= 

$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

x= 

$$y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

?=Don't care

Classes

1. Pedestrian
2. Car
3. Motorcycle
4. Background

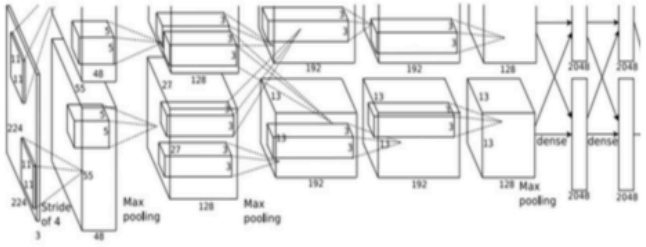
Possible loss function (with quadratic error):

$$L(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \dots (\hat{y}_8 - y_8)^2 & \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_1 = 0 \end{cases}$$

In practice, error is measured with a different function in each coordinate: logistic regression loss (p_c), squared error (bounding box), log-likelihood loss (class)

Detection with sliding windows

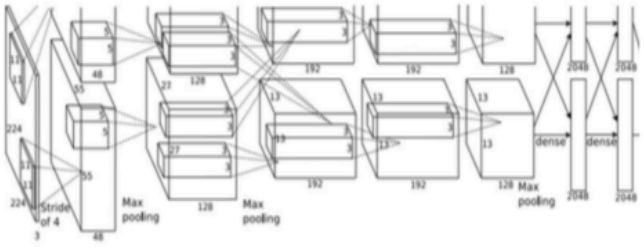
- Apply a CNN to different clippings in the image, CNN classifies each clip as object or background



Dog? NO
Cat? NO
Background? YES

Detection with sliding windows

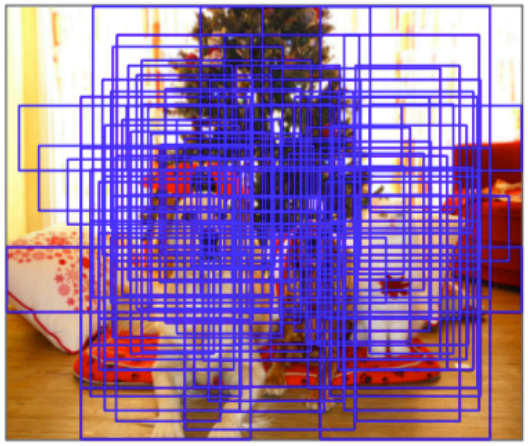
- Apply a CNN to different clippings in the image, CNN classifies each clip as object or background

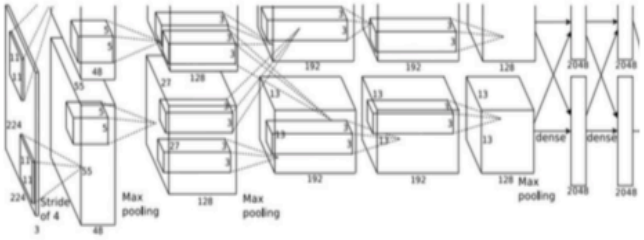


Dog? YES
Cat? NO
Background? NO

Detection with sliding windows

- Apply a CNN to different clippings in the image, CNN classifies each clip as object or background



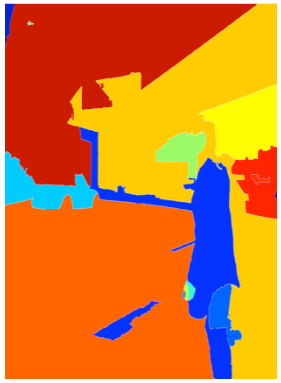


Dog? NO
Cat? YES
Background? NO

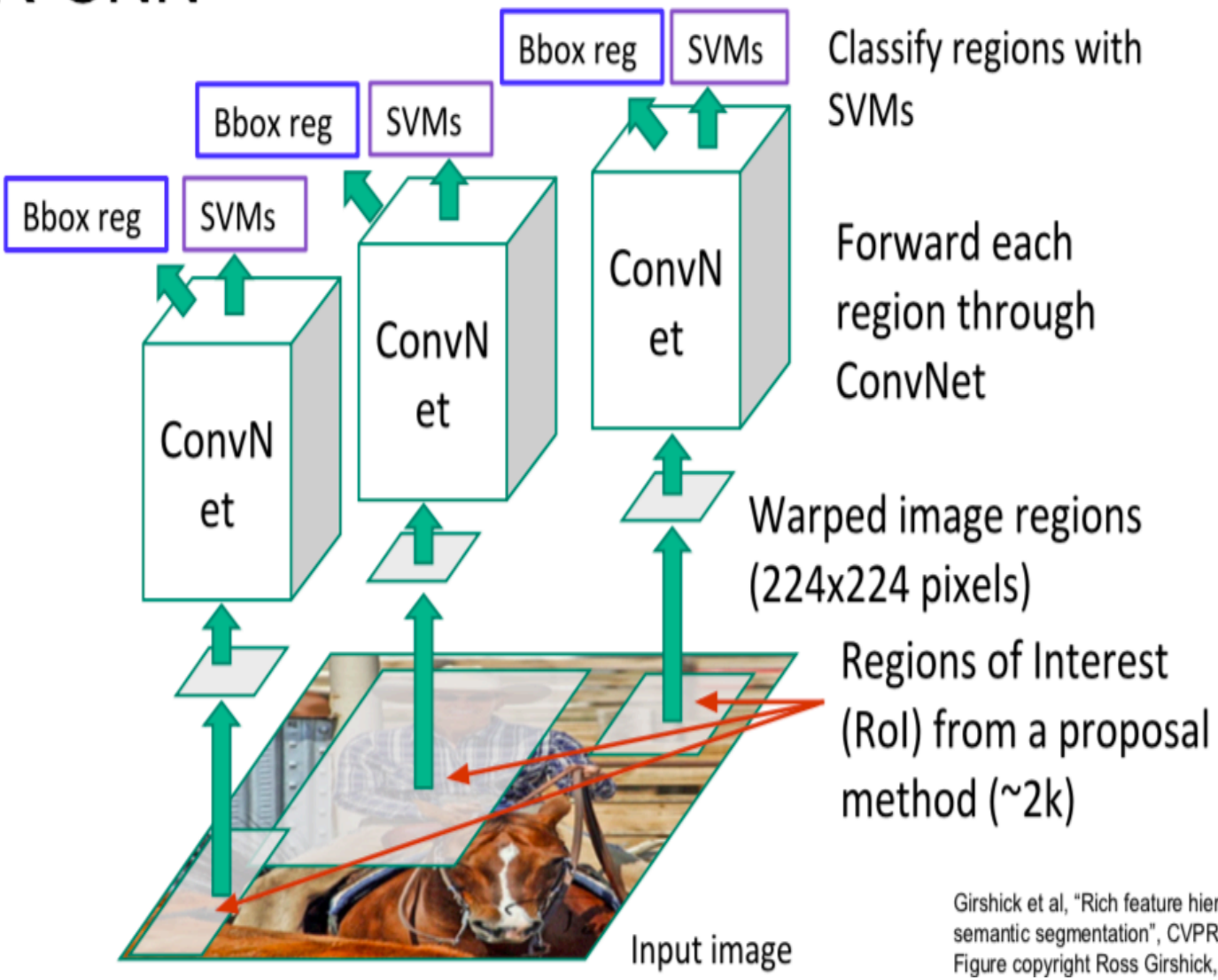
Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

R-CNN

- Object detection with sliding windows is inefficient: most regions do not contain objects
- R-CNN: finds "blobs" (regions of interest) and ranks them
- Uses selective search (segmentation algorithm) to find borders that separate regions of the image (returns ~ 2000 blobs)
- Particularly efficient when objects have many different shapes / scales



R-CNN

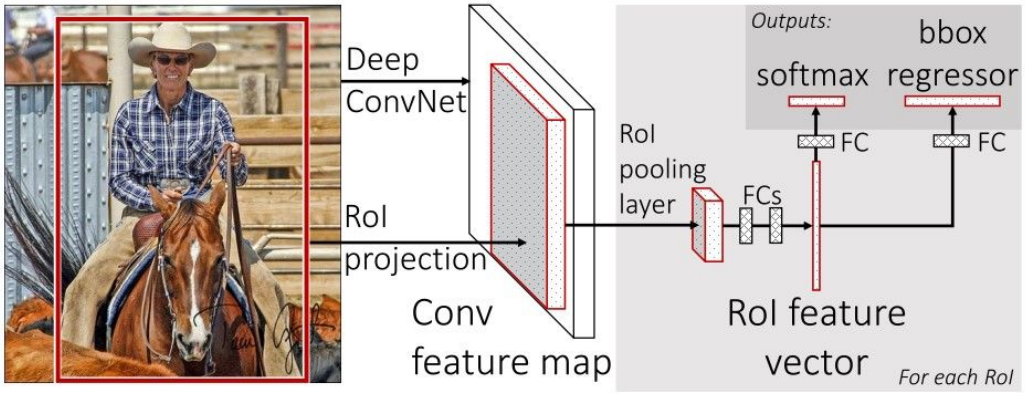


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Faster Algorithms

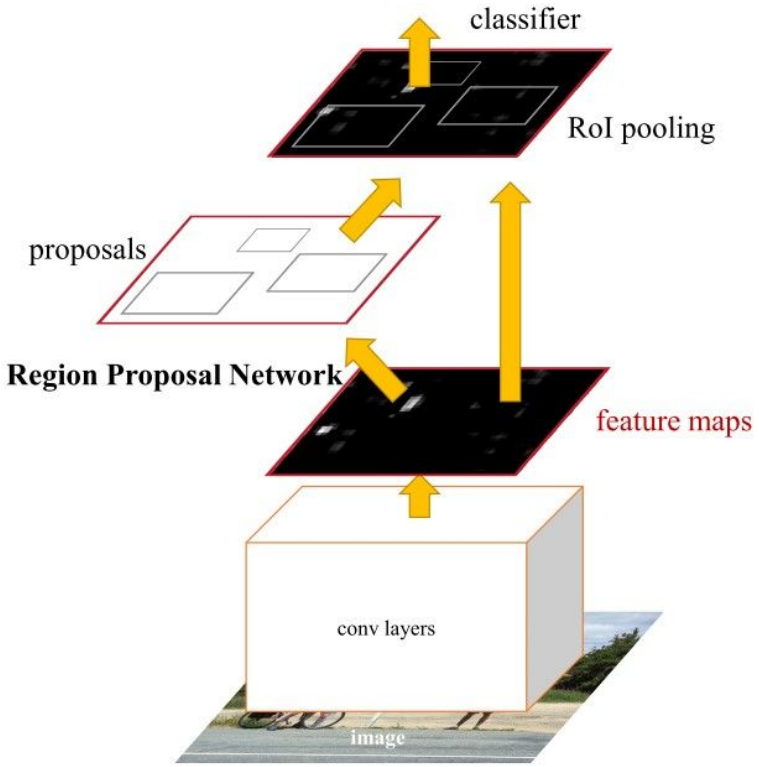
R-CNN: Propose regions; Classify proposed regions one at a time with SVM; Output label and bounding boxes

Fast R-CNN: Propose regions; Use a single CNN on the entire image; Compress the features of each ROI; Classify ROI.

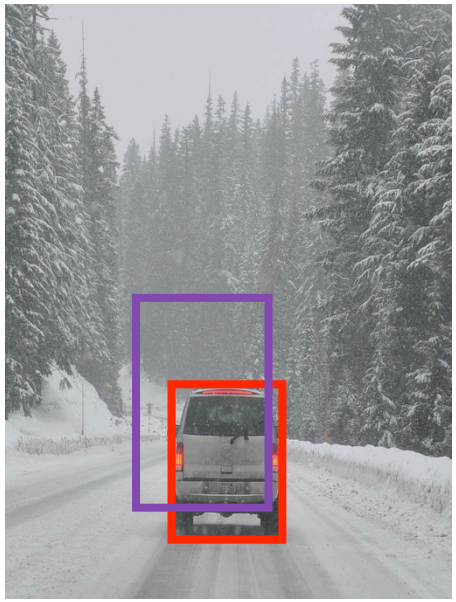


Faster Algorithms

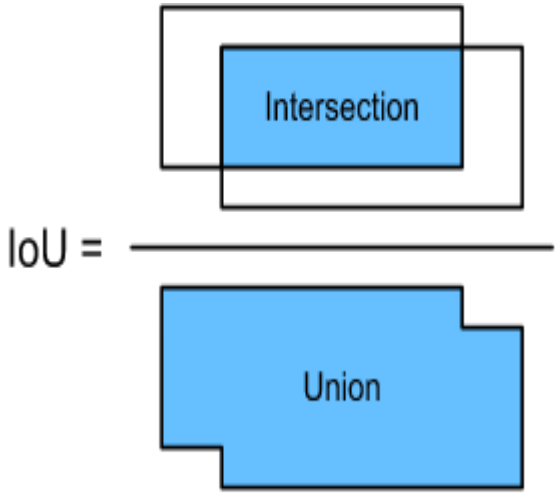
Faster R-CNN: Use the CNN to propose regions (attention map)



Evaluation



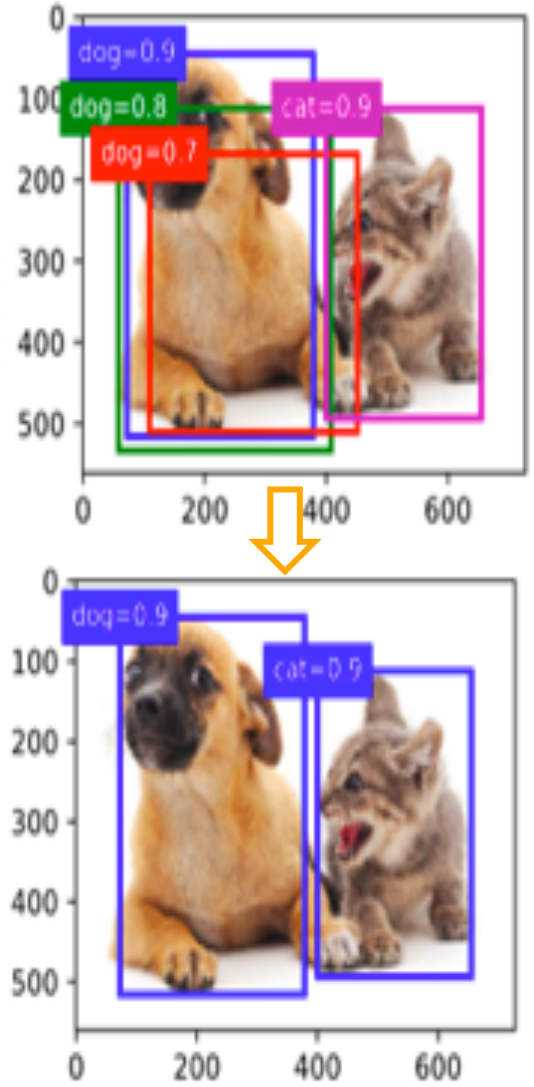
Intersection over Union (IoU)



"Correct" if IoU ≥ 0.5

Multiple boxes

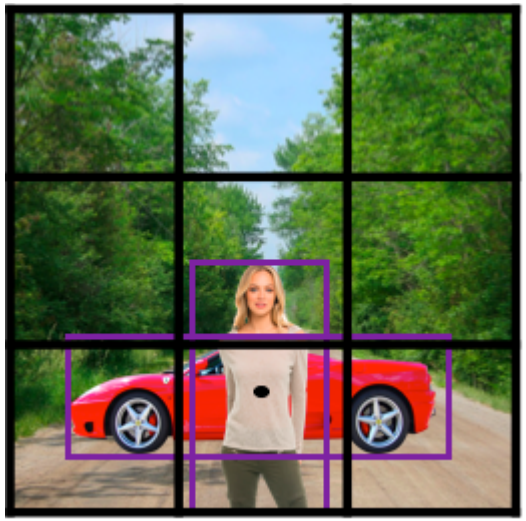
- Methods may find multiple bounding boxes referring to the same object
- Non-max suppression allows to detect the object once



Non-max suppression

- Non-max suppression aims to return only one box per object
- First, evaluate pc for each of the bounding boxes, discard boxes with pc<0.5
- Get the box with largest pc (probable object)
- All boxes with high IoU (>0.5) are discarded
- Keep picking the bounding box with the largest pc among the rest, until each box has been selected or removed

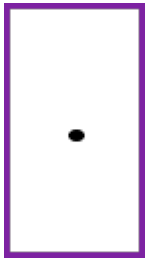
Multiple objects in the same window

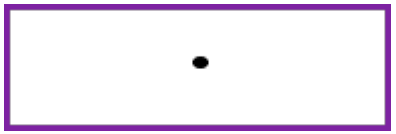


Anchor boxes are bounding boxes of multiple pre-defined sizes and aspect ratios

Anchor box 1

Anchor box 2





$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ \vdots \\ c_3 \end{bmatrix}$$

Anchor box 1

Anchor box 2

…

Anchor boxes

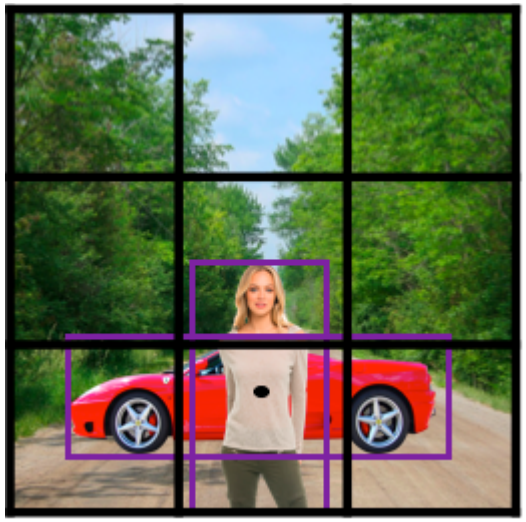
Before (without anchor boxes): Each object in the training image was associated with the grid cell that contains the object's midpoint

Output y:
3 x 3 x 8

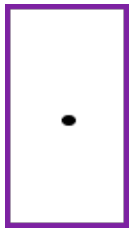
Now (with 2 anchor boxes per region):
Each object in the training image is assigned to the grid cell that contains the midpoint and to the anchor box with the highest IoU.

Output y:
3 x 3 x 16
or
3 x 3 x 2
x 8

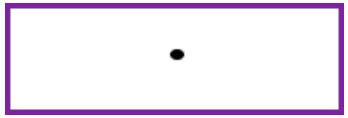
Anchor boxes example



Anchor box 1



Anchor box 2



This image

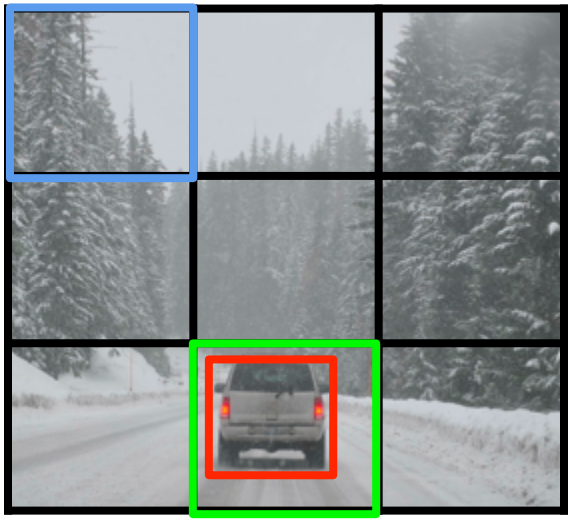
If only a car

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

YOLO

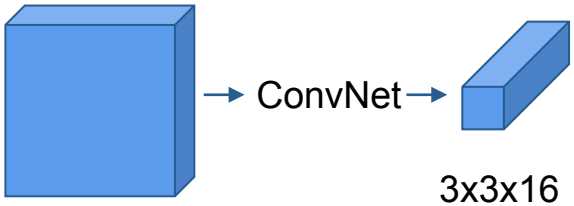
- Sliding windows are expensive
- Sliding windows and region proposal algorithms share a lot of computation
- Most of the proposed regions have no objects
- Why not partitioning the image into grid cells and processing all together as a convolutional network?

YOLO Training

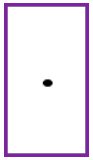


Y is 3x3x2x8

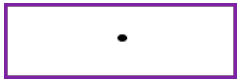
1 - pedestrian
2 - car
3 - motorcycle



Anchor box 1



Anchor box 2



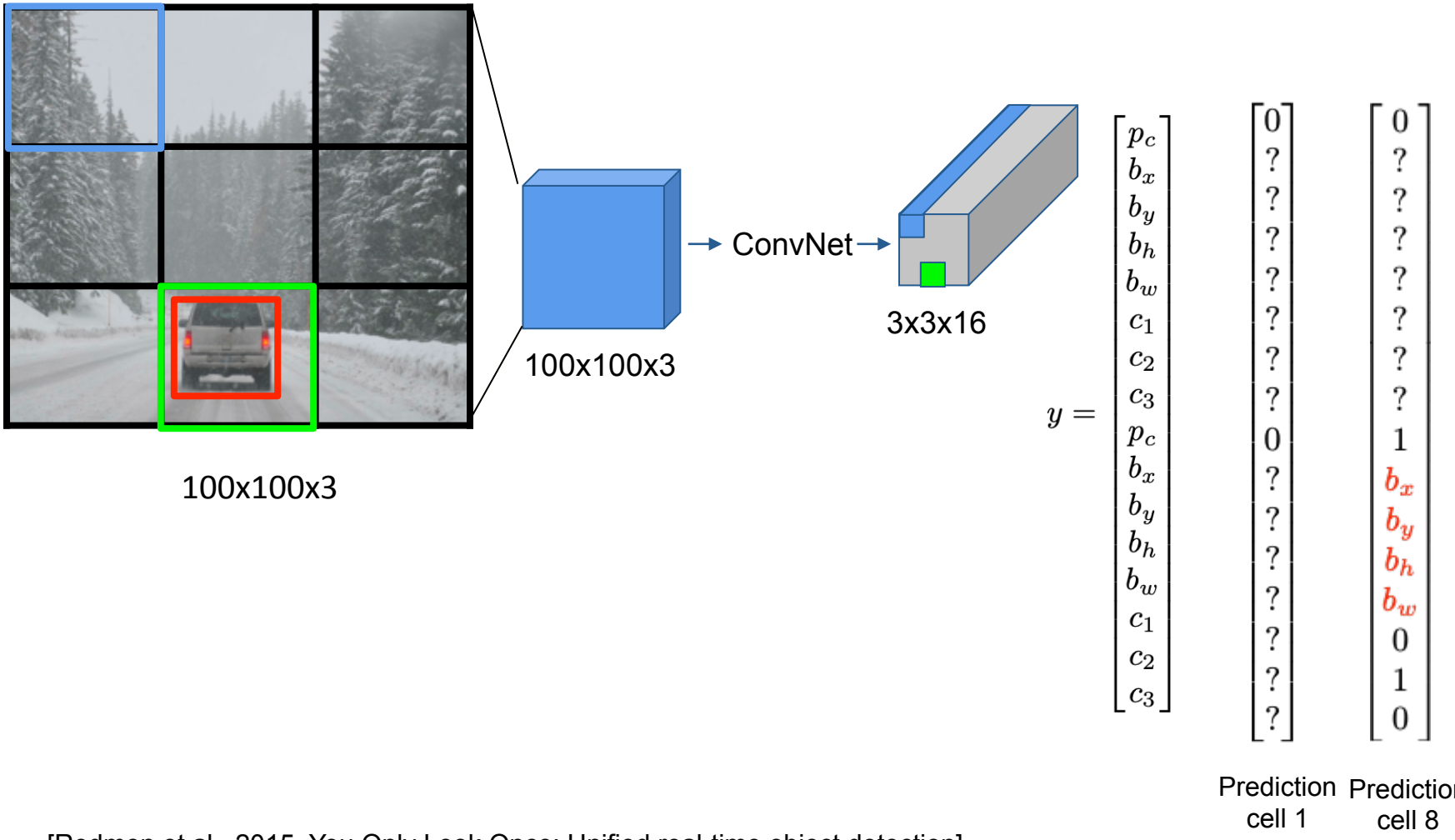
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Cell 1 Cell 8

What if
… finer grid? 19x19x2x8
… 5 anchor boxes? 19x19x5x8

[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

YOLO Testing



[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

YOLO Example

In the previous example

- Training:
  - Cell 1 has no object
  - Cell 8 has a car, with larger IoU with anchor box 2
- Prediction:
  - Cell 1: pc = 0 is expected, other entries don't matter (don't care)
  - Cell 8: anchor box 2 is expected to match the car

YOLO Limitations

Limitations:

- Grid cell with more objects than anchor boxes
- Grid cell with 2 objects matching the same anchor box

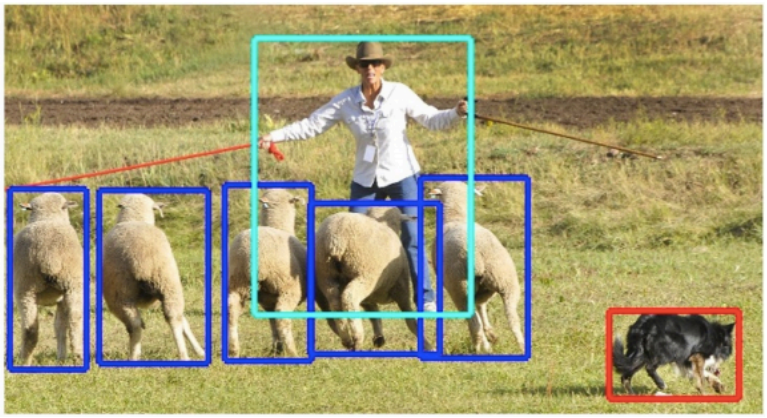
Options:

- Increase number of grid cells
- Increase number of anchor boxes

Semantic Segmentation

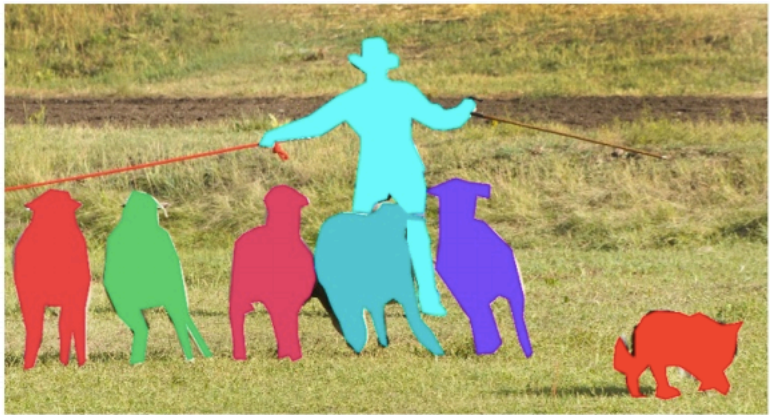
Sparse X Dense Labeling

image classification



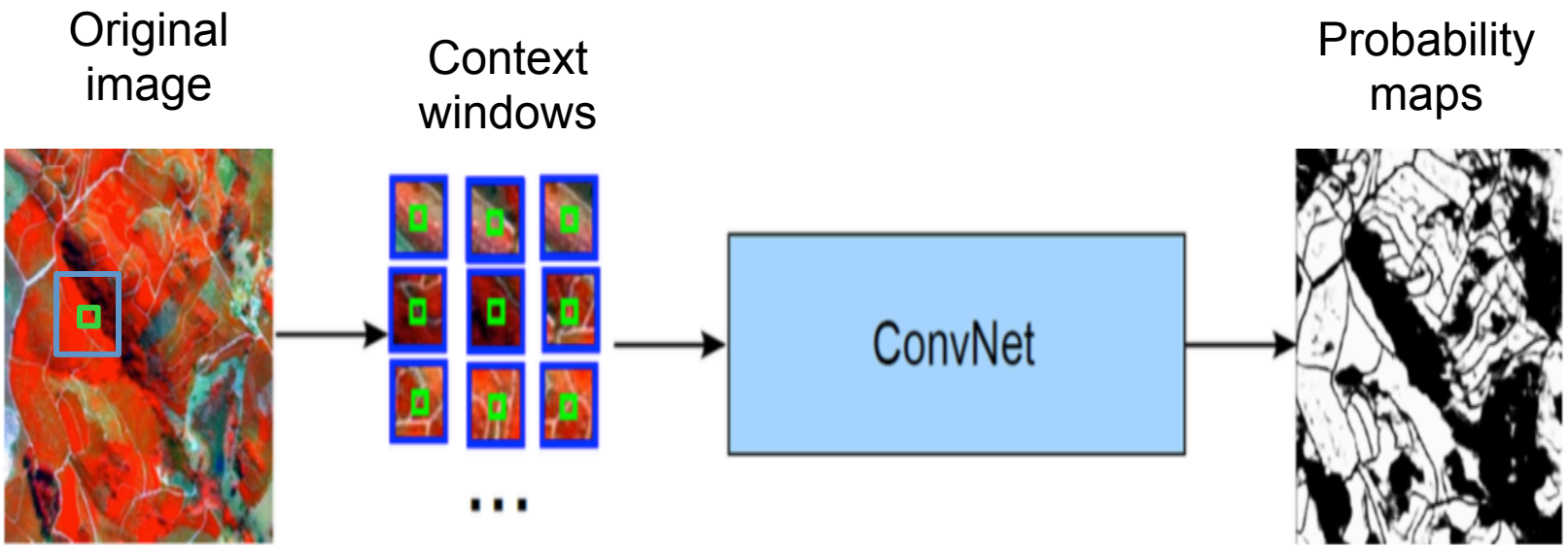
object detection

semantic segmentation



instance segmentation

Semantic Segmentation as Pixel Classification

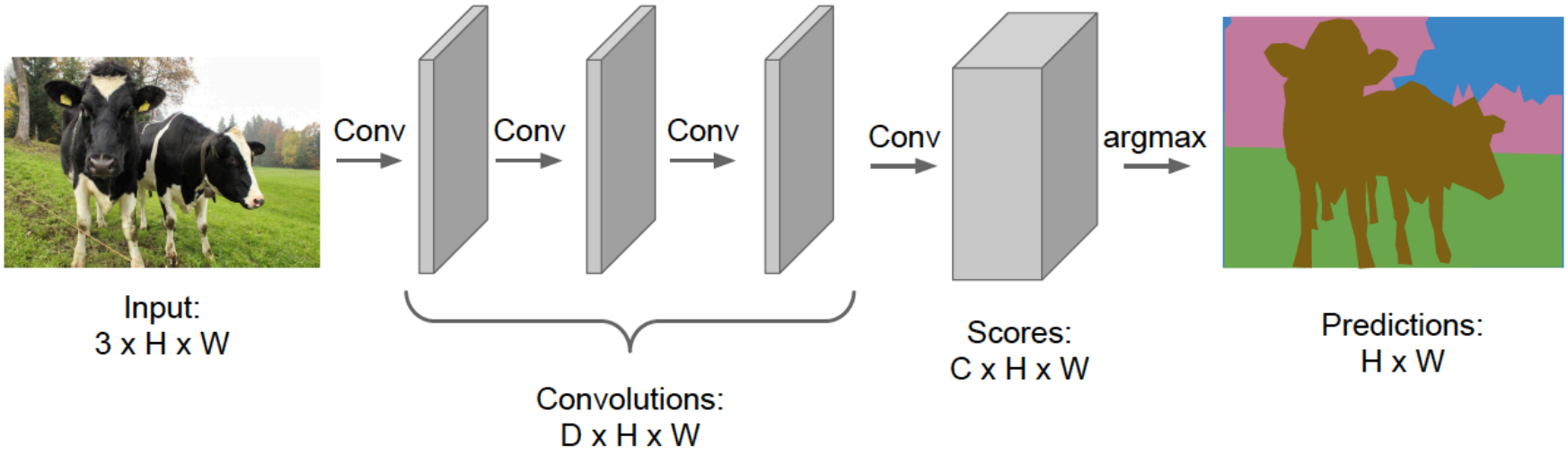


Expensive, regions share information!

Nogueira et al. Learning to semantically segment high-resolution remote sensing images. ICPR 2016.

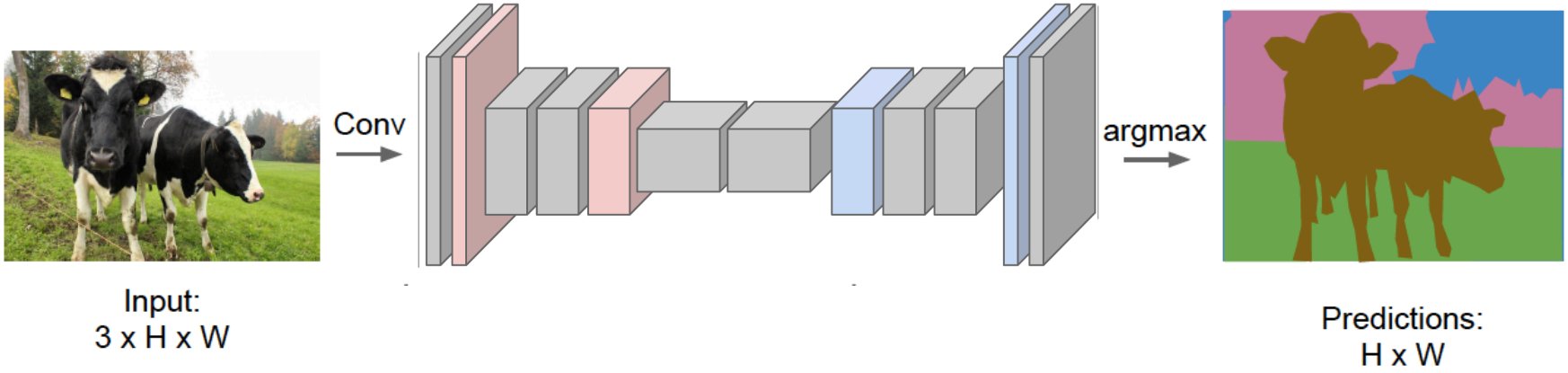
Fully convolutional networks

- Design a network with only convolutional layers, make predictions for all pixels at once
- Ideally, we want convolutions at full image resolution, but implementing that naively is too expensive

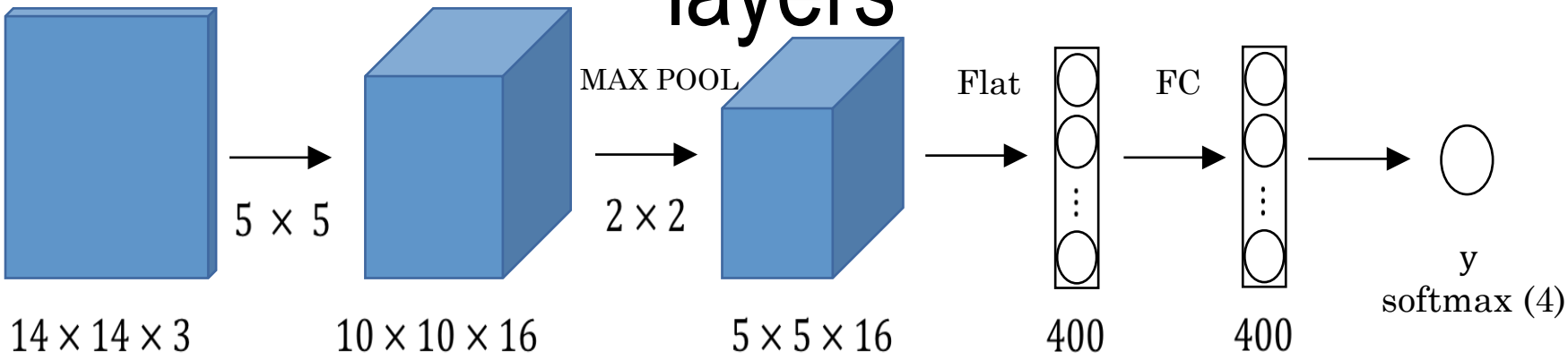


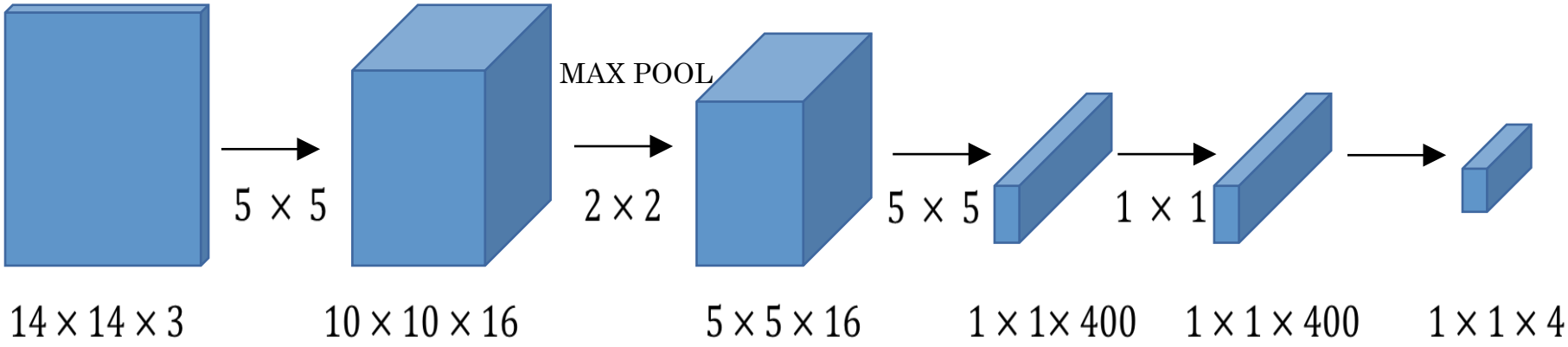
Fully convolutional networks

- Design a network with only convolutional layers, make predictions for all pixels at once
- Ideally, we want convolutions at full image resolution, but implementing that naively is too expensive
- Solution: first downsample, then upsample



Turning FC layer into convolutional layers





How to do upsampling in-network (unpooling)?

Given the following 2 x 2 input, how to get a 4 x 4 output?

Input 2x2

1	2
3	4

Output: 4x4

Nearest Neighbor

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

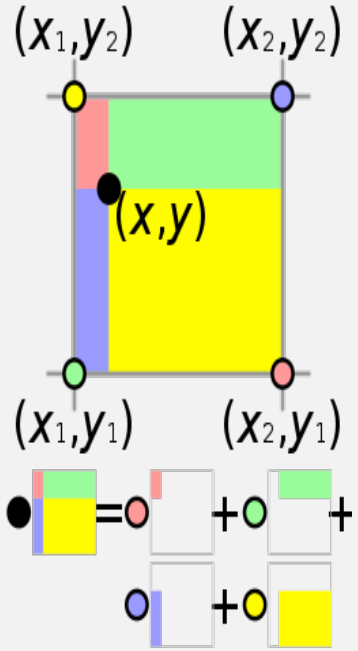
"Nail Bed"

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

Bilinear interpolation

1		2	
3		4	

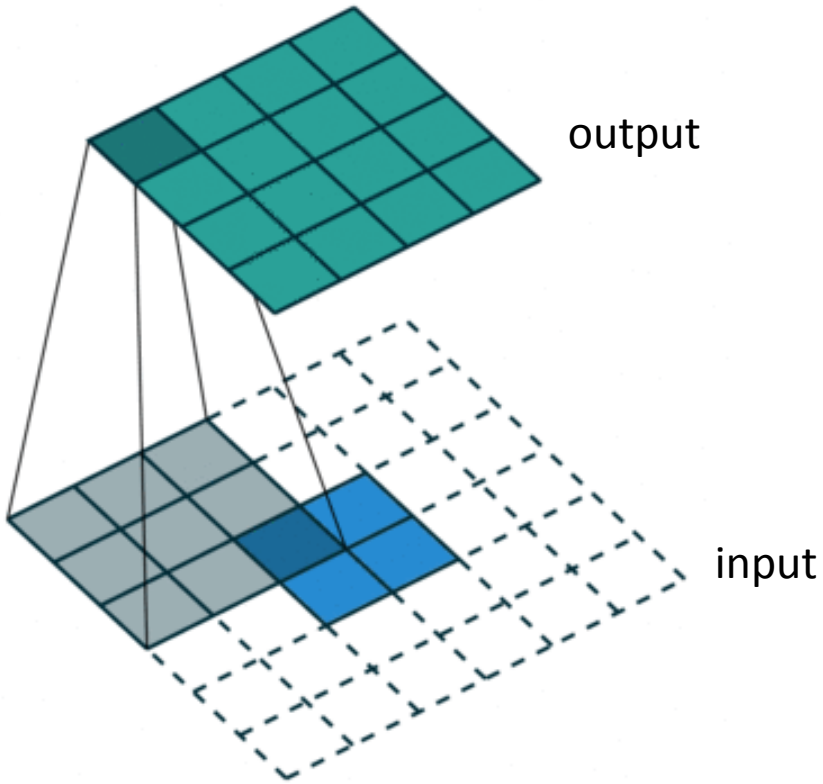
Bilinear interpolation (Wikipedia).



Upsampling in a deep network

Transposed convolution:

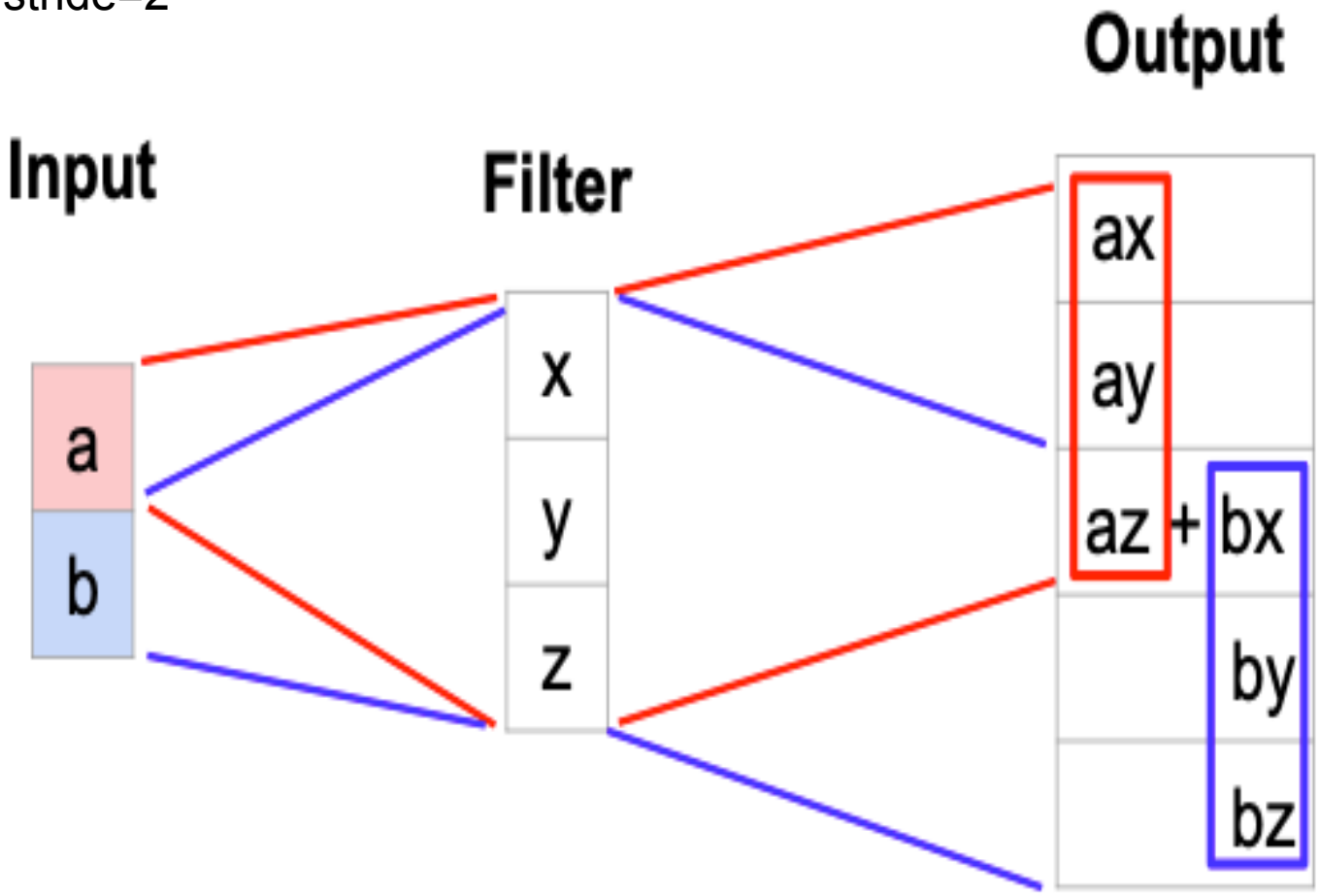
- Deconvolution
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution



V. Dumoulin and F. Visin, A guide to convolution arithmetic for deep learning, arXiv 2018

Transpose convolution Example

stride=2

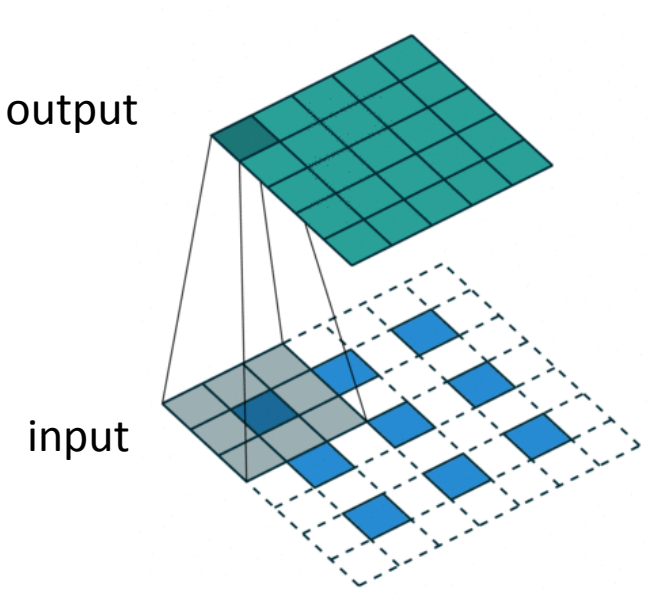


Output contains input-weighted copies of the filter, adding where the output overlaps

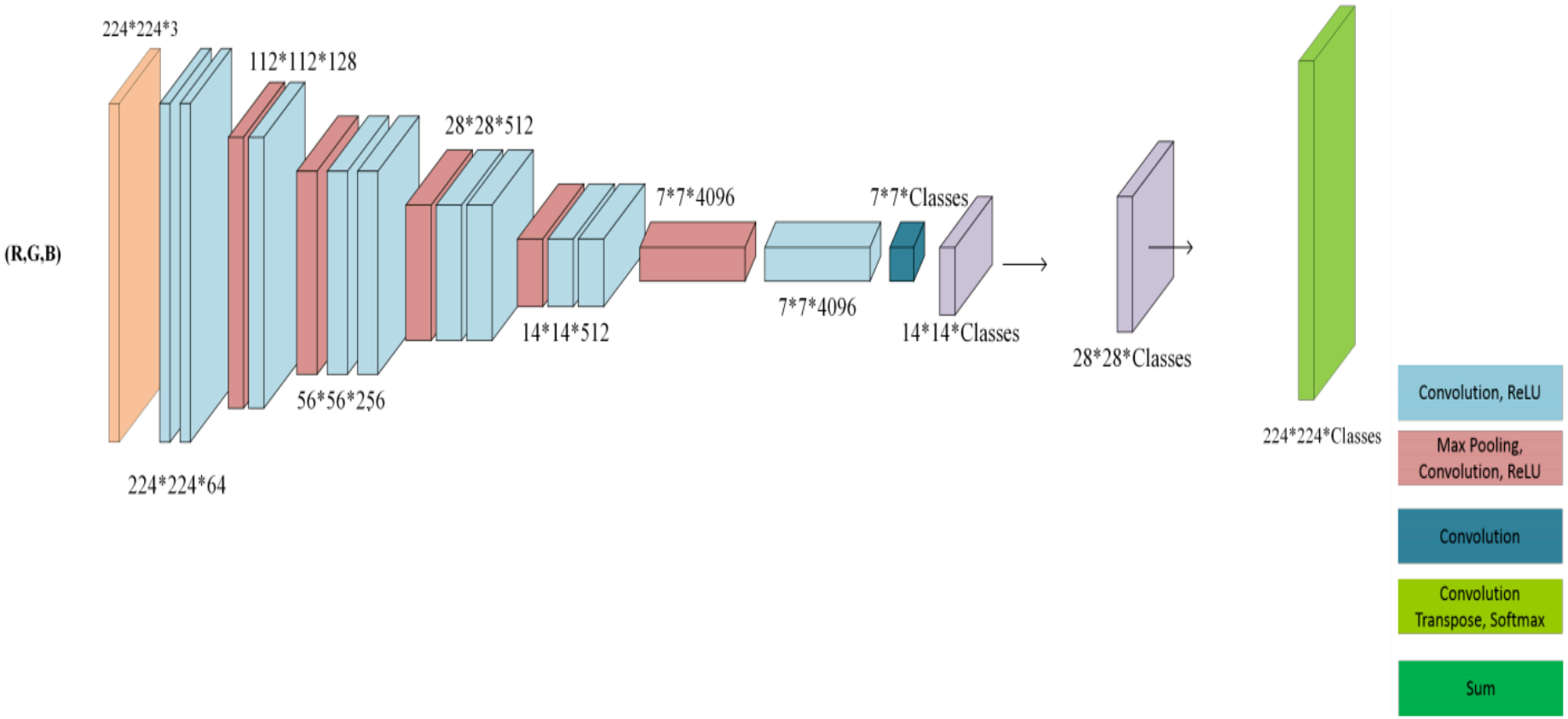
Upsampling in a deep network

Backwards-strided convolution: to increase resolution, use *output stride* > 1

- For stride 2, dilate the input by inserting rows and columns of zeros between adjacent entries, convolve with flipped filter
- Sometimes called convolution with *fractional input stride* 1/2

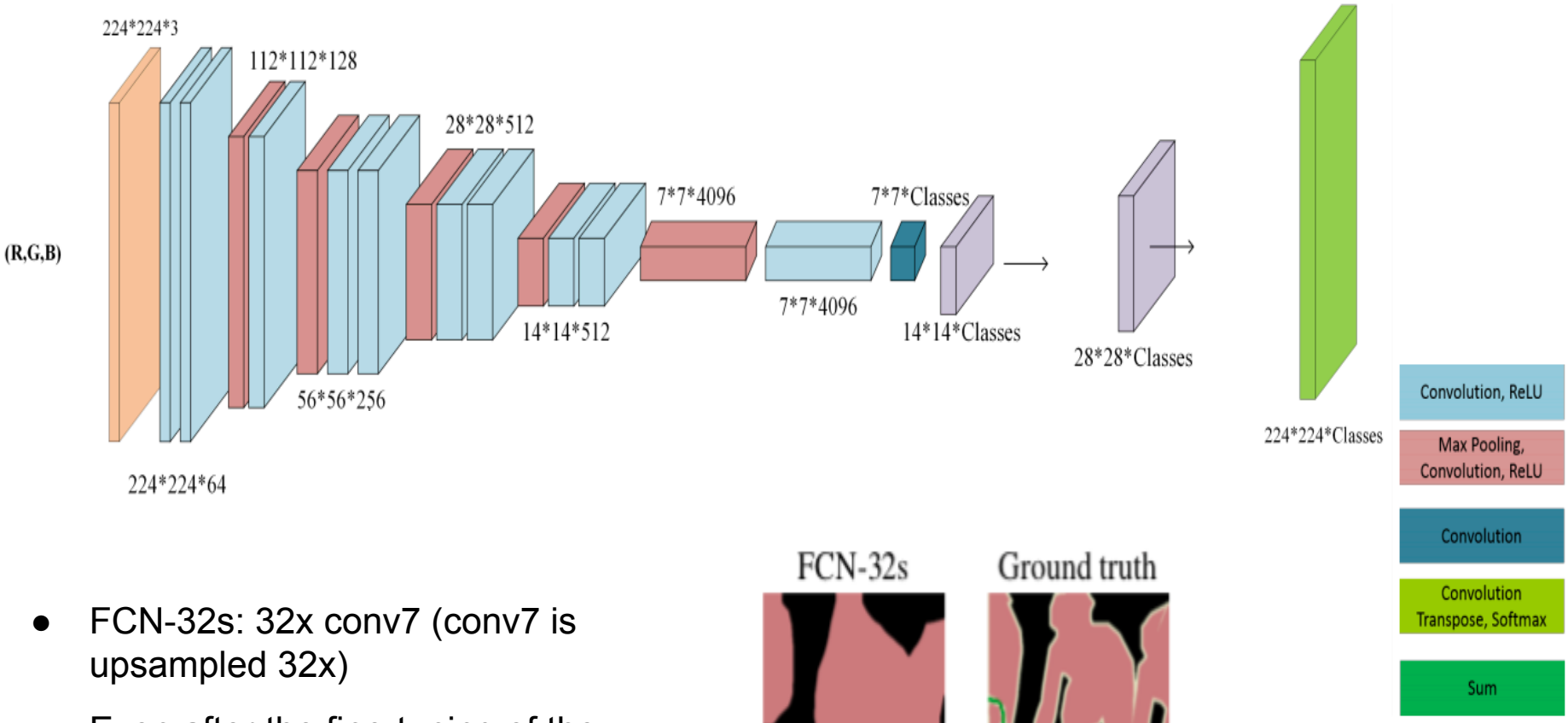


FCN-32s Architecture

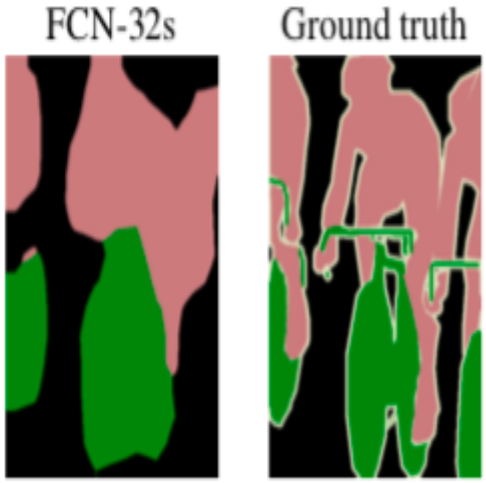


Long et al. Fully convolutional networks for semantic segmentation. In CVPR 2015.

FCN-32s Architecture

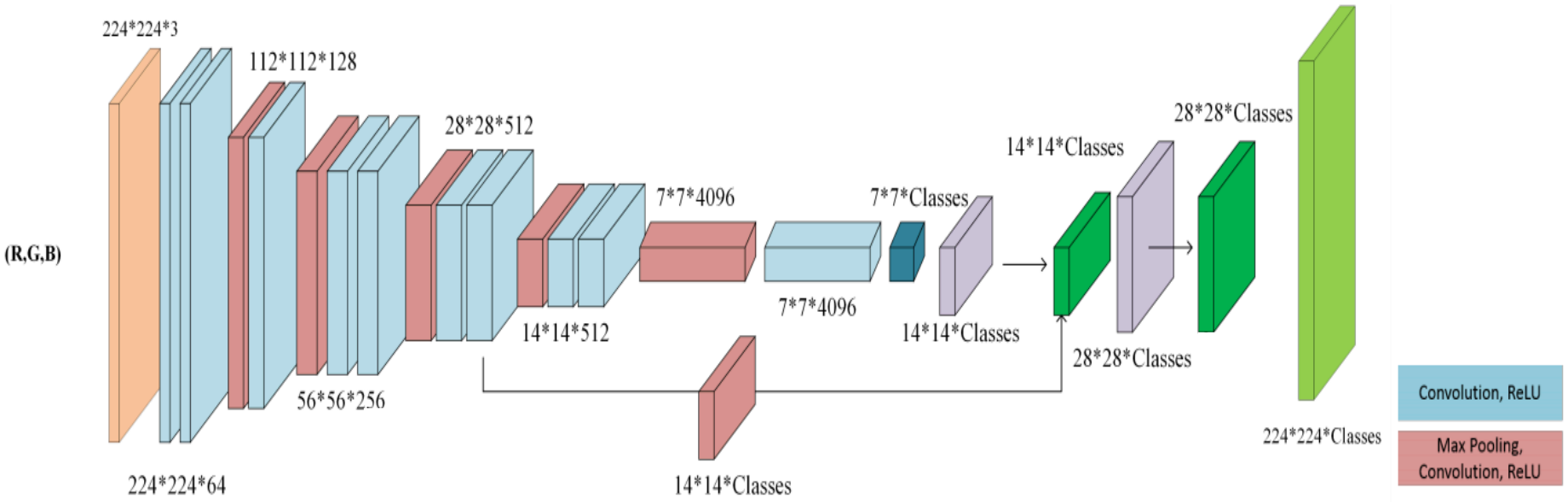


- FCN-32s: 32x conv7 (conv7 is upsampled 32x)
- Even after the fine-tuning of the pre-trained networks, the results are unsatisfactorily coarse

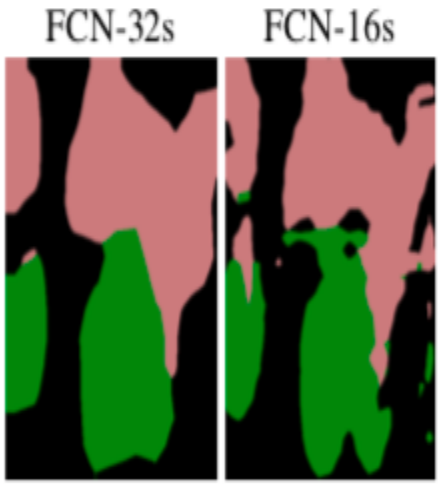


Long et al. Fully convolutional networks for semantic segmentation. In CVPR 2015.

FCN-16s Architecture

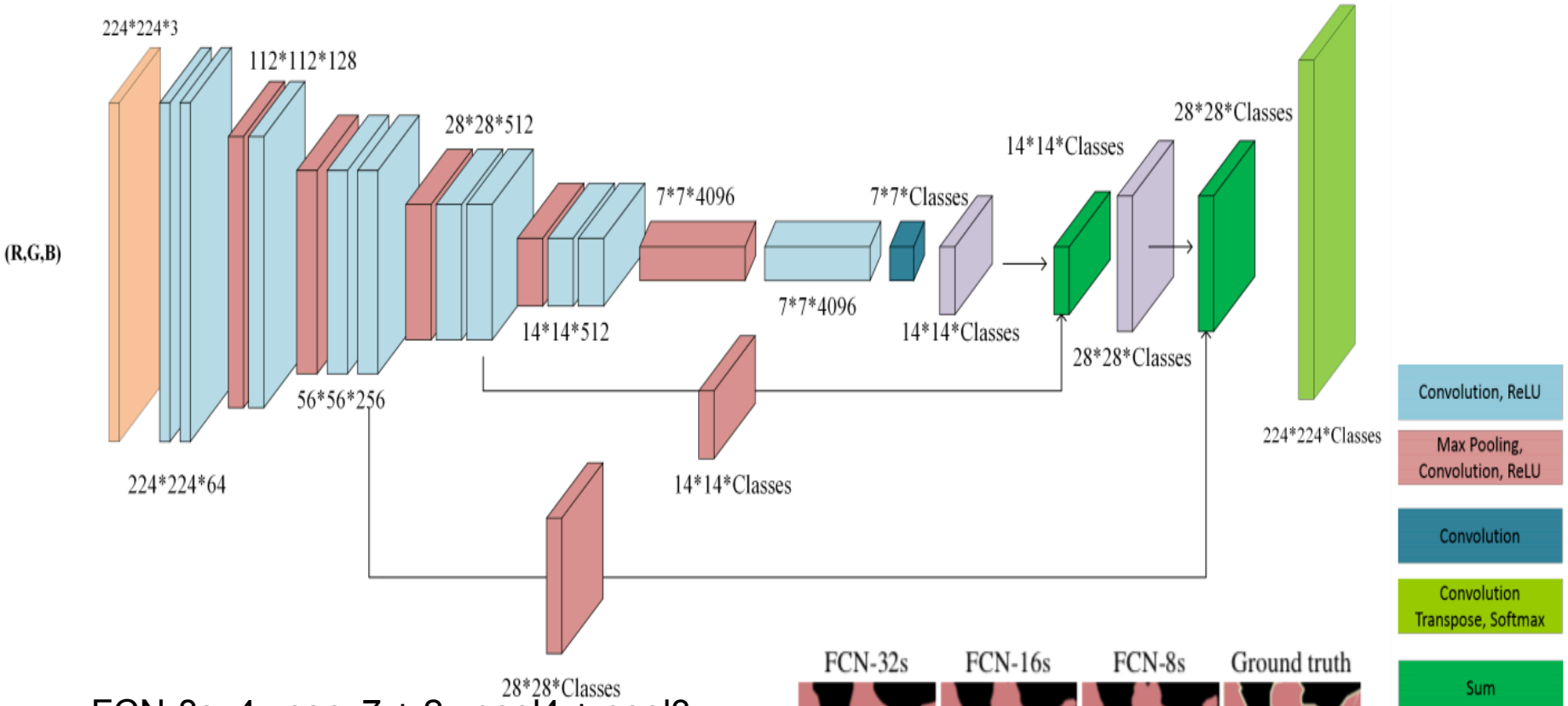


- Skip Connections to link pool4 layer
- FCN-16s: 2x conv7 + pool4; result is upsampled 16x
- Advantages:
  - Fusion of high semantic level information with low semantic level information
  - Mitigation of the Vanishing Gradient problem in larger networks

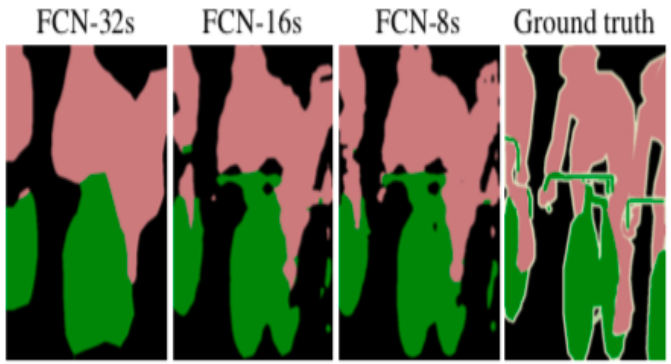


Long et al. Fully convolutional networks for semantic segmentation. In CVPR 2015.

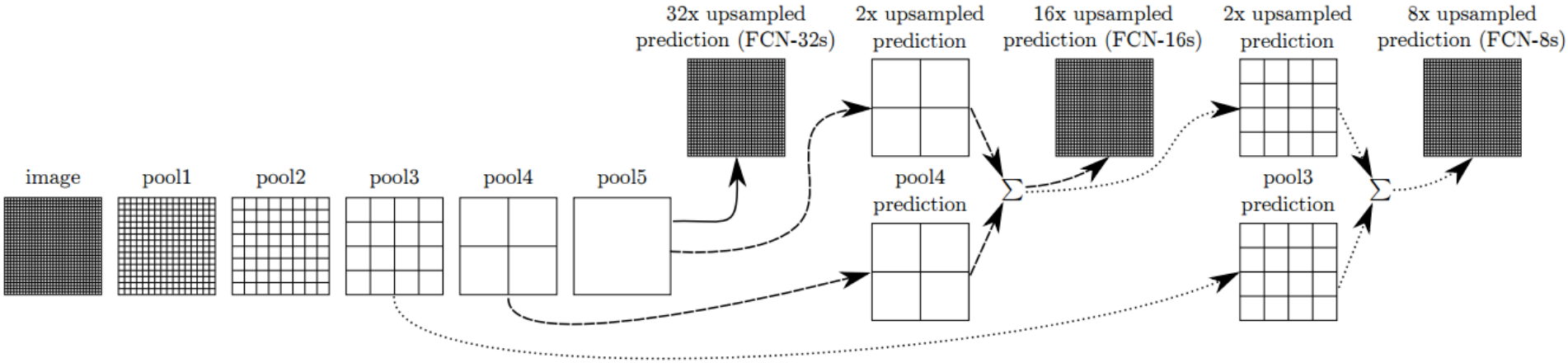
FCN-8s Architecture

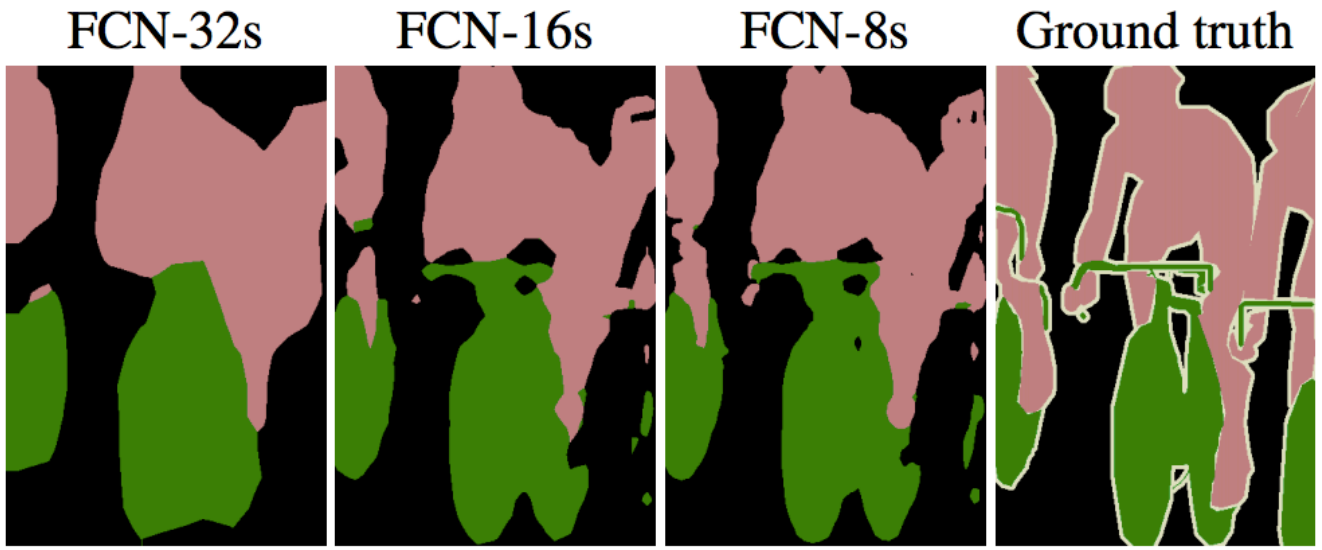


- FCN-8s: 4x conv7 + 2x pool4 + pool3; result is upsampled 8x
- To retrieve finer spatial information in the final layers of the decoder, it is necessary to connect with shallower layers of the encoder



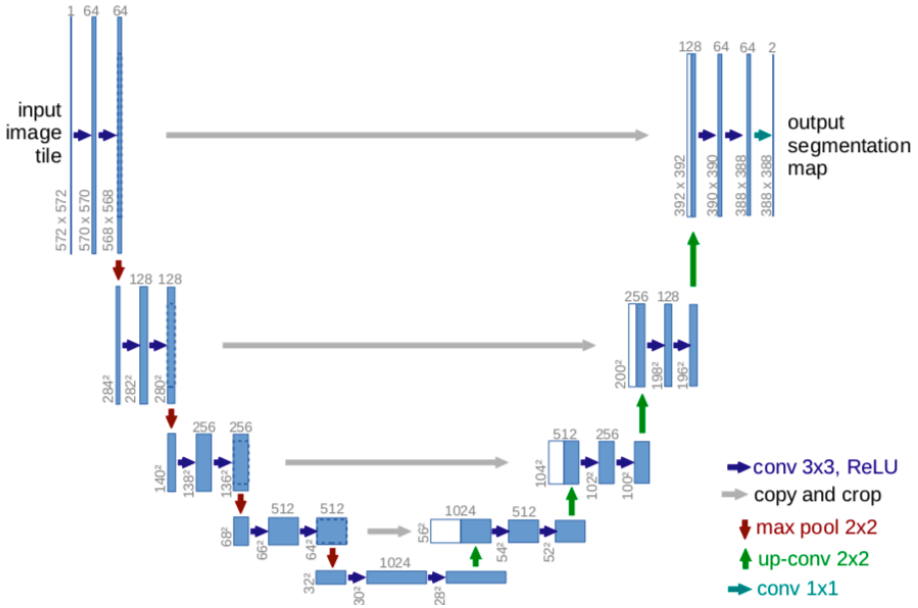
Fully convolutional networks



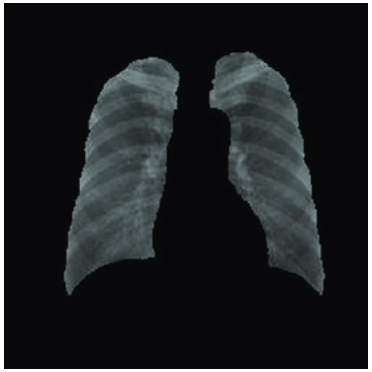


Long et al. Fully convolutional networks for semantic segmentation. In CVPR 2015.

U-Nets



datascience.stackexchange.com

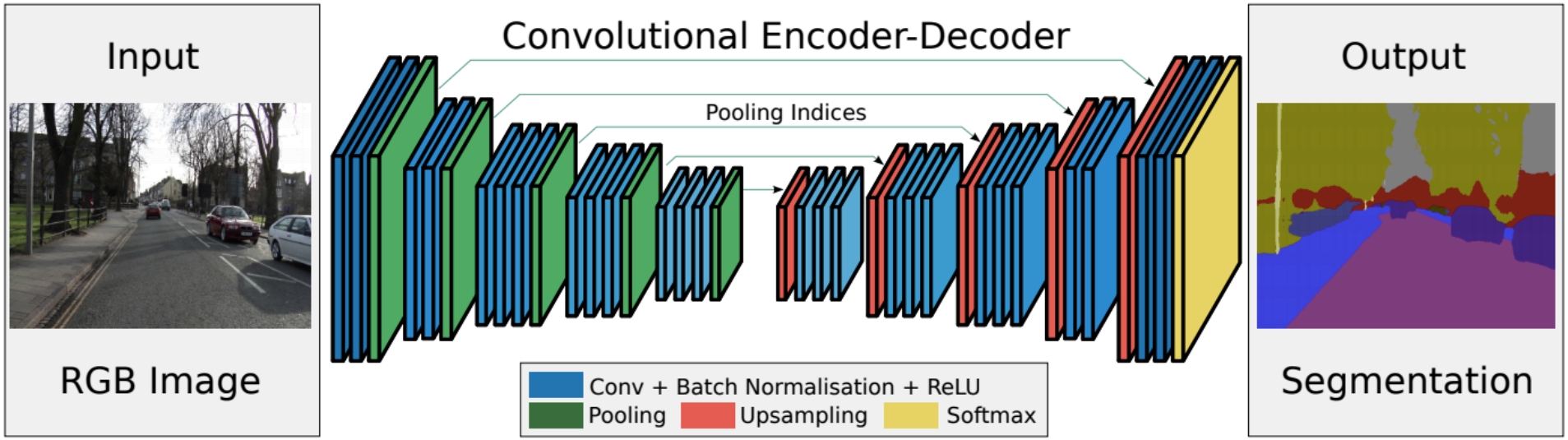


Heo 2019

- Like FCN, fuse upsampled higher-level feature maps with higher-res, lower-level feature maps
- Unlike FCN, fuse by concatenation, predict at the end

O. Ronneberger et al. U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015

SegNets

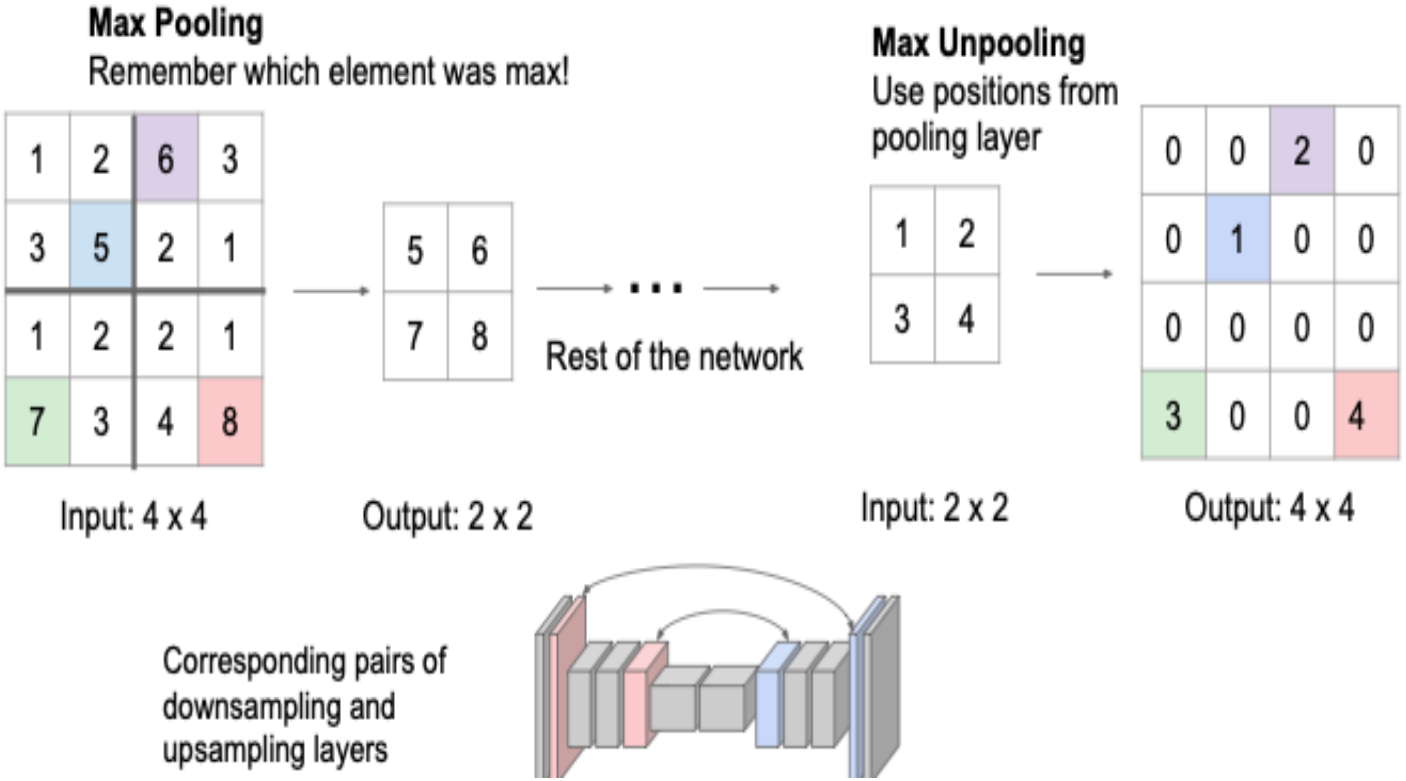


- Encoder-Decoder Architectures
- Use of Encoder pooling indexes for the reconstruction of the Decoder

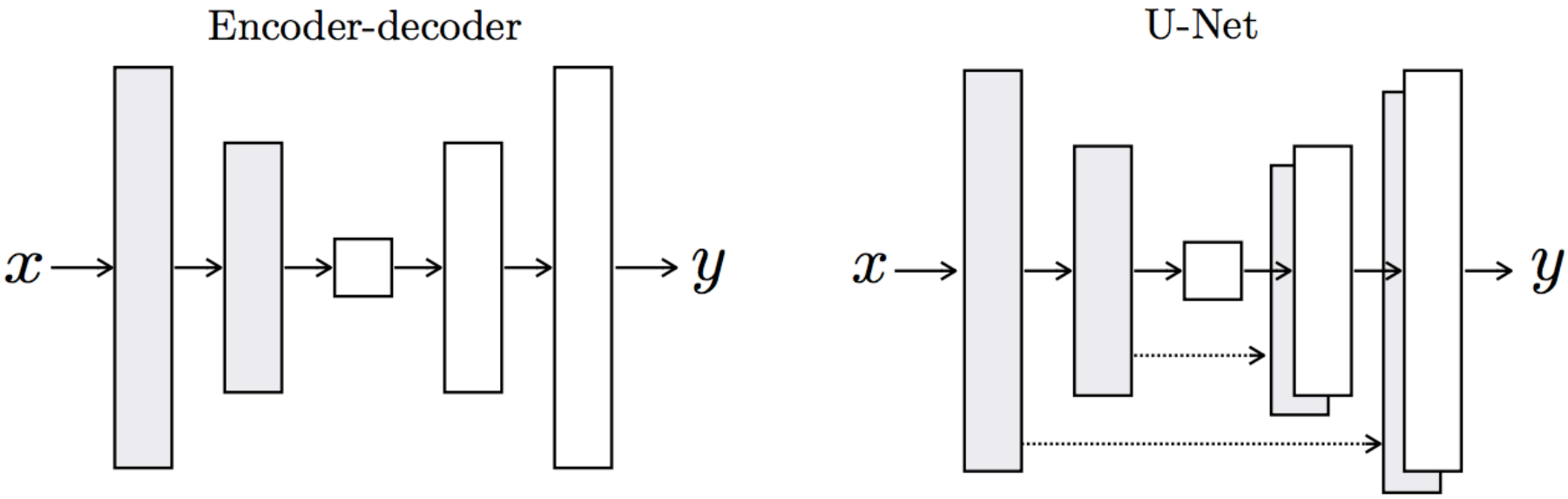
V. Badrinarayanan, A. Kendall and R. Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, PAMI 2017

SegNets

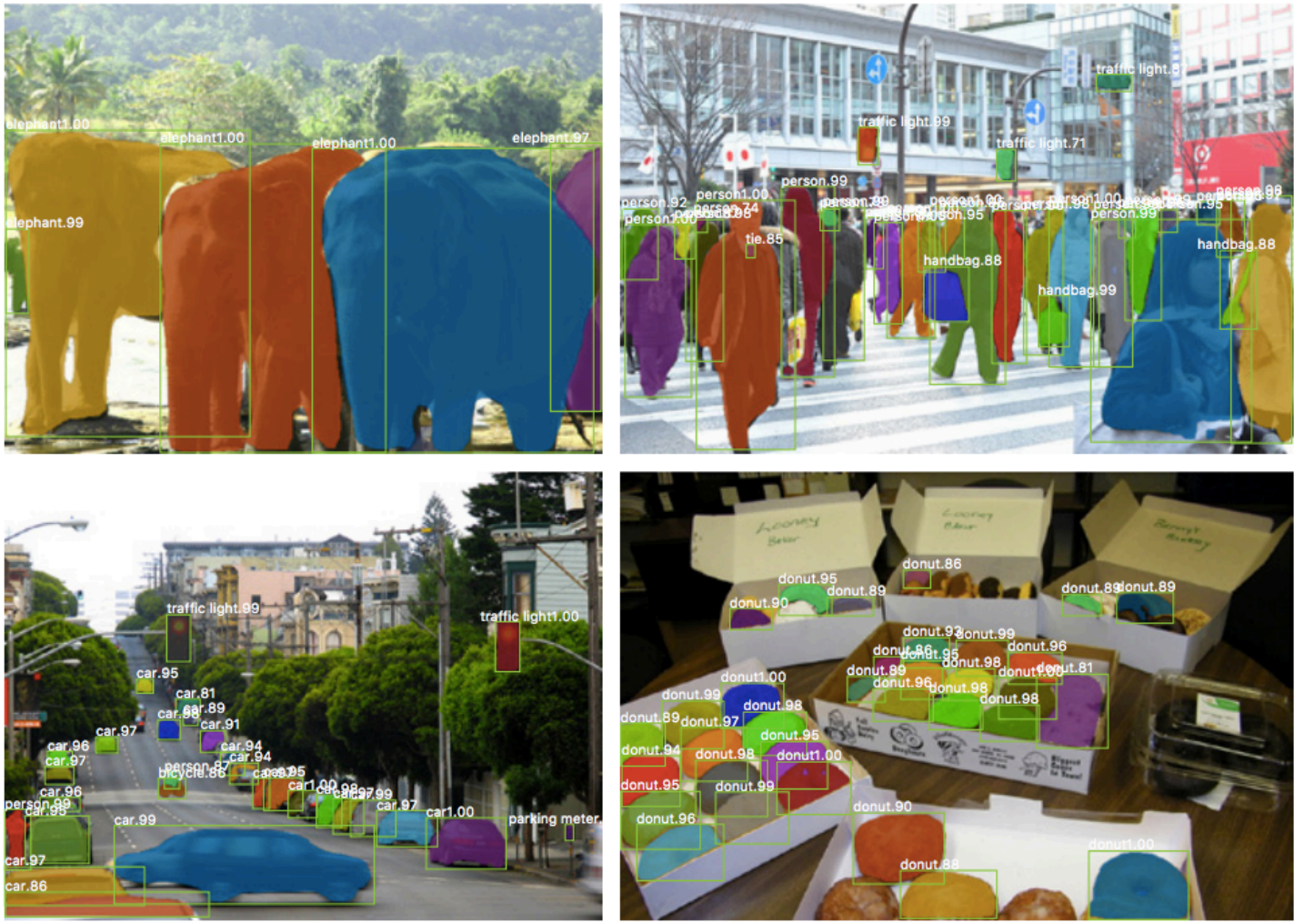
- When we do max pooling, we lose information about which layer element was used. We can save these positions and use them later
- Similar to the nail bed, but places the corresponding elements in the original positions



Summary of upsampling architectures

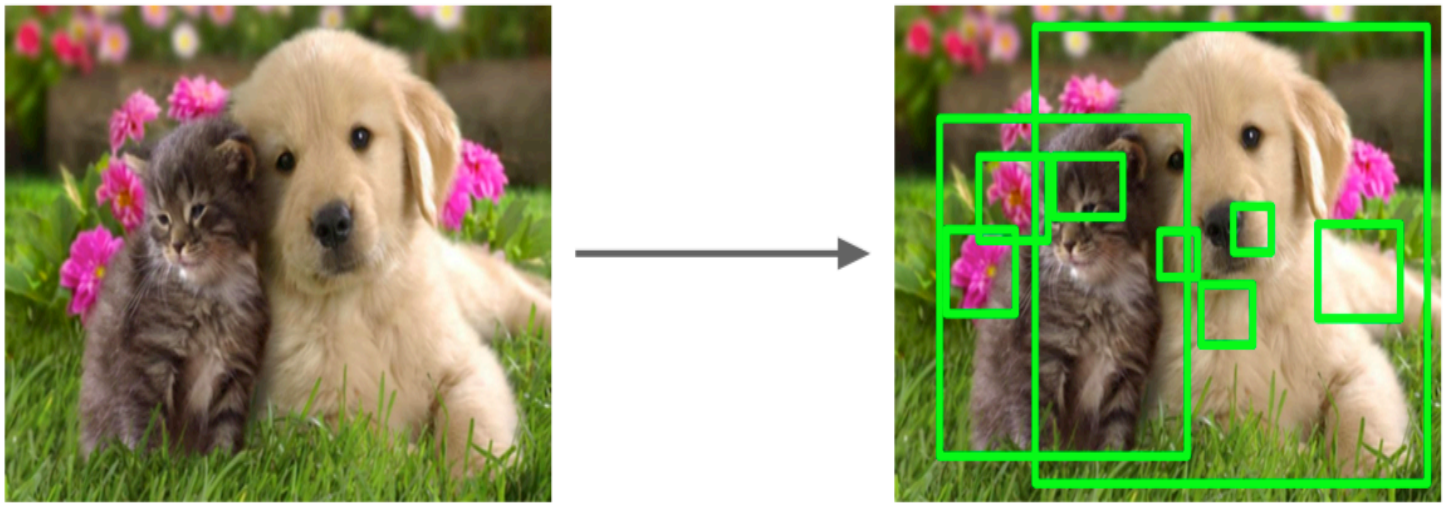


Instance Segmentation



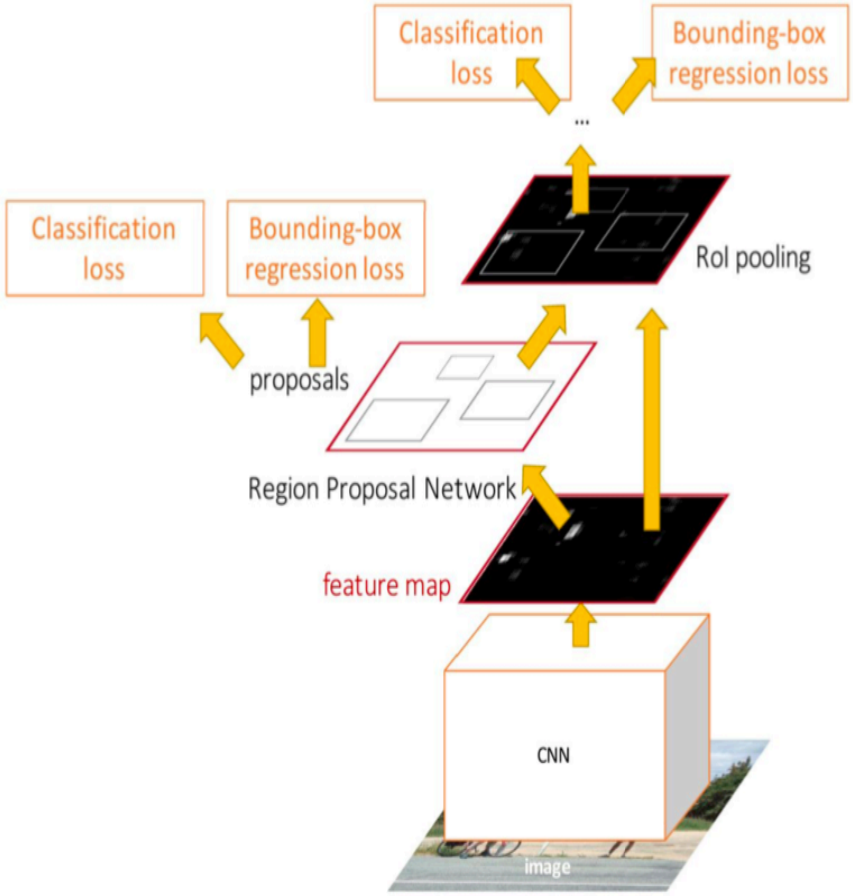
Remembering Region Proposal

- Find "blobs" (regions of the image) that probably contain objects
- R-CNN and Fast R-CNN use Selective Search to return ~ 2000 RoIs
- Faster R-CNN uses region proposal network (RPN)

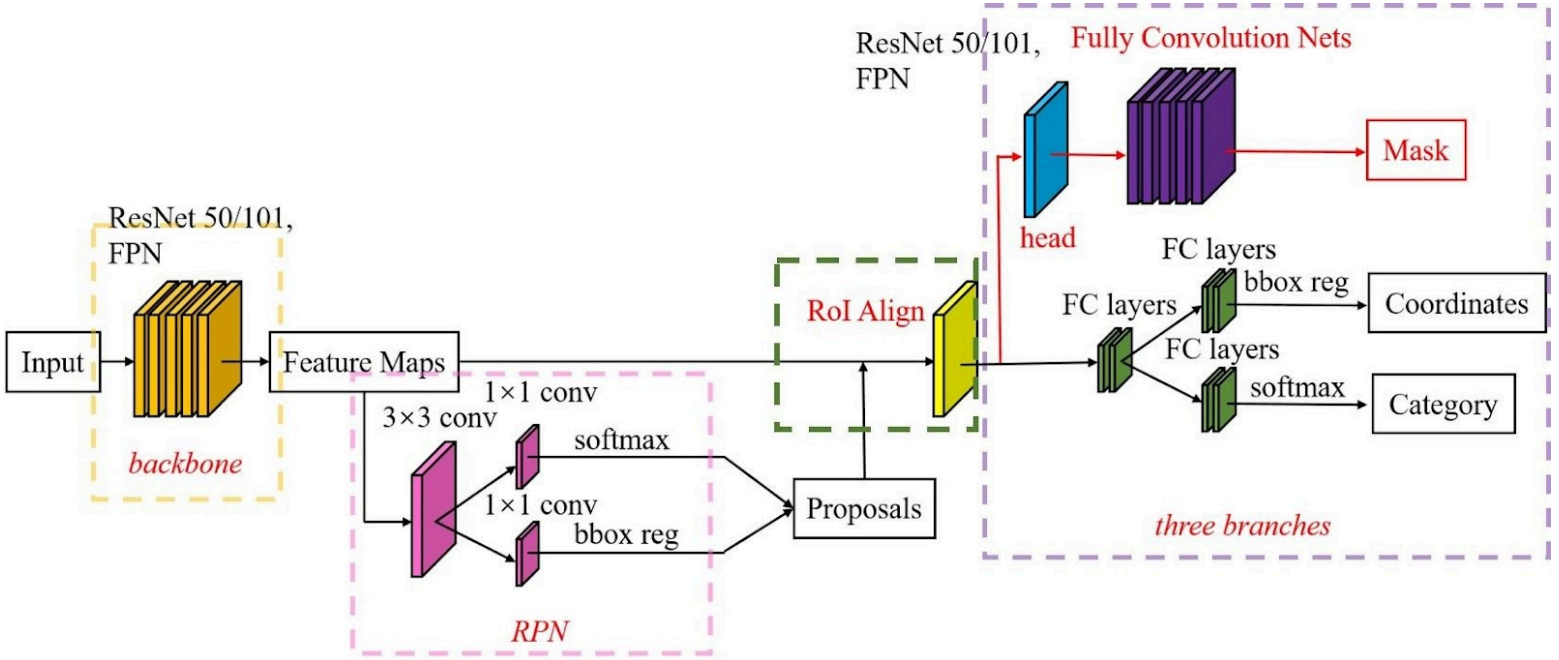


Faster R-CNN

- Proposals made from the feature map
- Insert Region Proposal Network (RPN) to return proposals directly from the features
- Train together with 4 losses:
  - Object / non-object RPN
  - RPN coordinates bounding box
  - Final classification score (object classes)
  - Final bbox coordinates



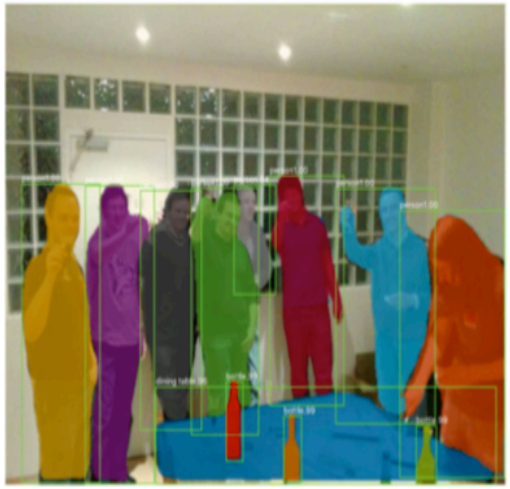
Mask R-CNN



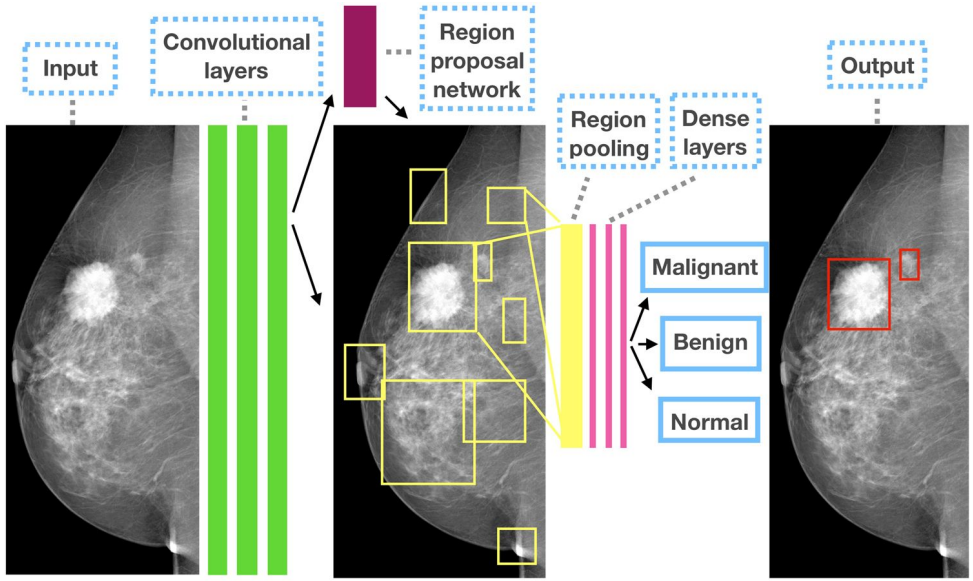
- Mask R-CNN extends Faster R-CNN for instance segmentation by adding a branch to provide segmentation masks parallel to the existing branch for classification and bbox regression

Mask R-CNN

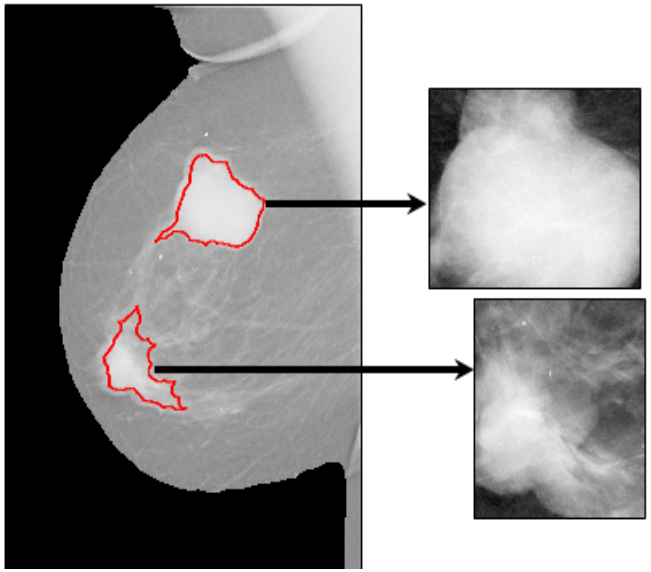




Detection and Segmentation



A2b Healthcare



bioimage.khu.ac.kr

References and acknowledgements

Some of these slides were inspired or adapted from courses and presentations given by Andrew Ng, Camila Laranjeira, Fei-Fei Li, Flávio Figueiredo, Hugo Oliveira, Jefersson dos Santos, Justin Johnson, Keiller Nogueira, Pedro Olmo, Renato Assunção, Serena Yeung.

Reference courses include *Machine Learning* and *Deep Learning* CS230 and CS231 from Stanford University, *Deep Learning* and *Hands-on Deep Learning* from UFMG, *Deep Learning* CS498 from Un. Of Illinois.