Object detection with CNNs

Strasbourg
13.02.2020

Robert Maria

Main Points

1. Two-stage detectors

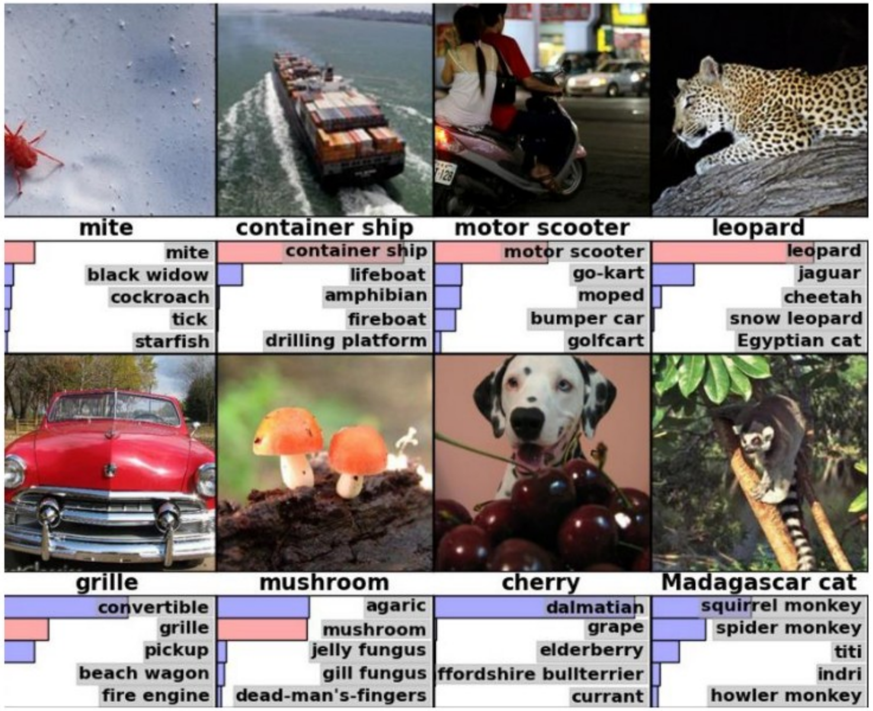
2. Single-stage detectors

3. Demo

Classification vs Segmentation vs Detection

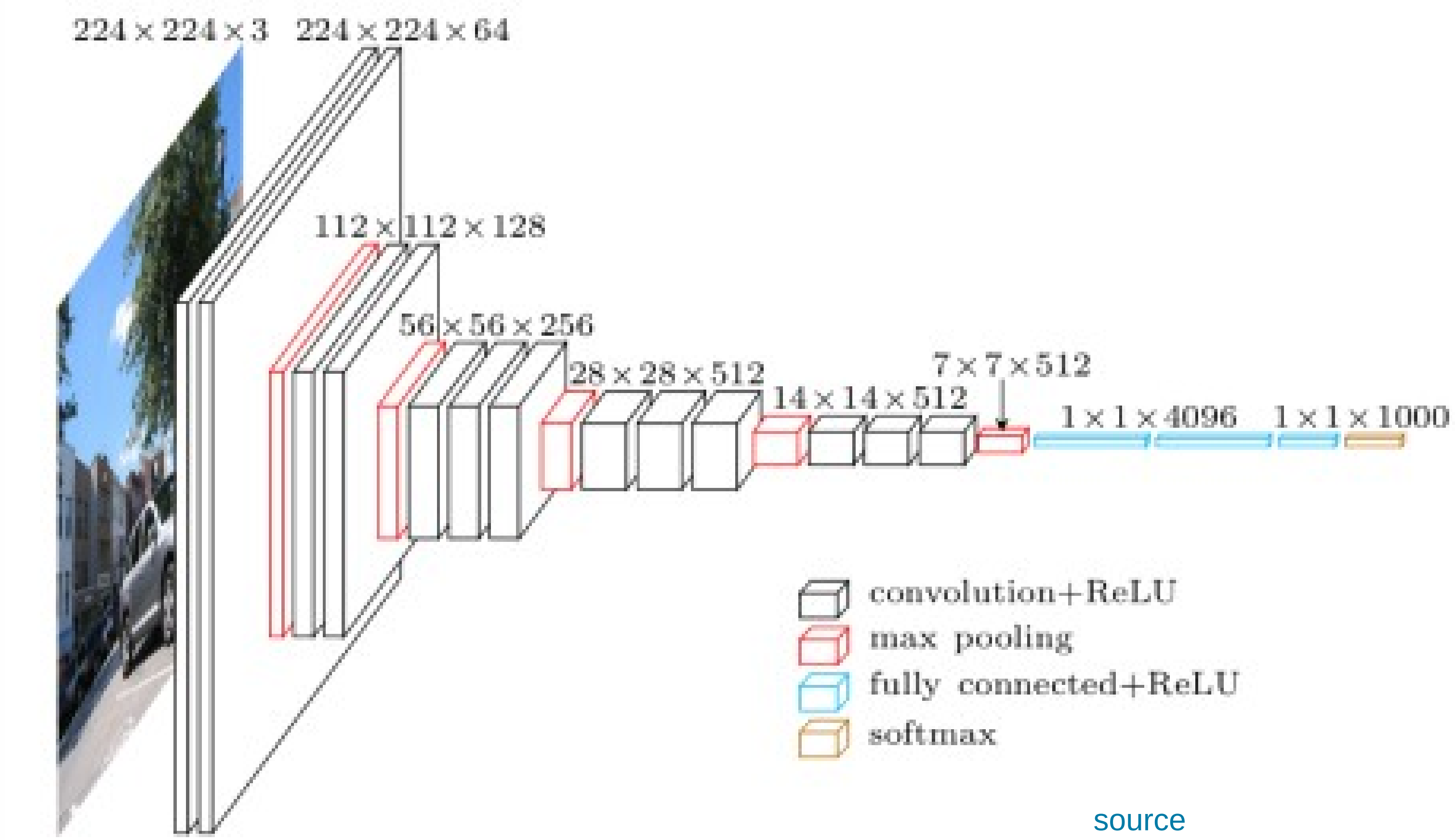
ImageNet – dataset

~ 14 million images with annotations (link)



Classification: For each of the 1000 classes, predicting presence/absence of an example of that class in the test image.

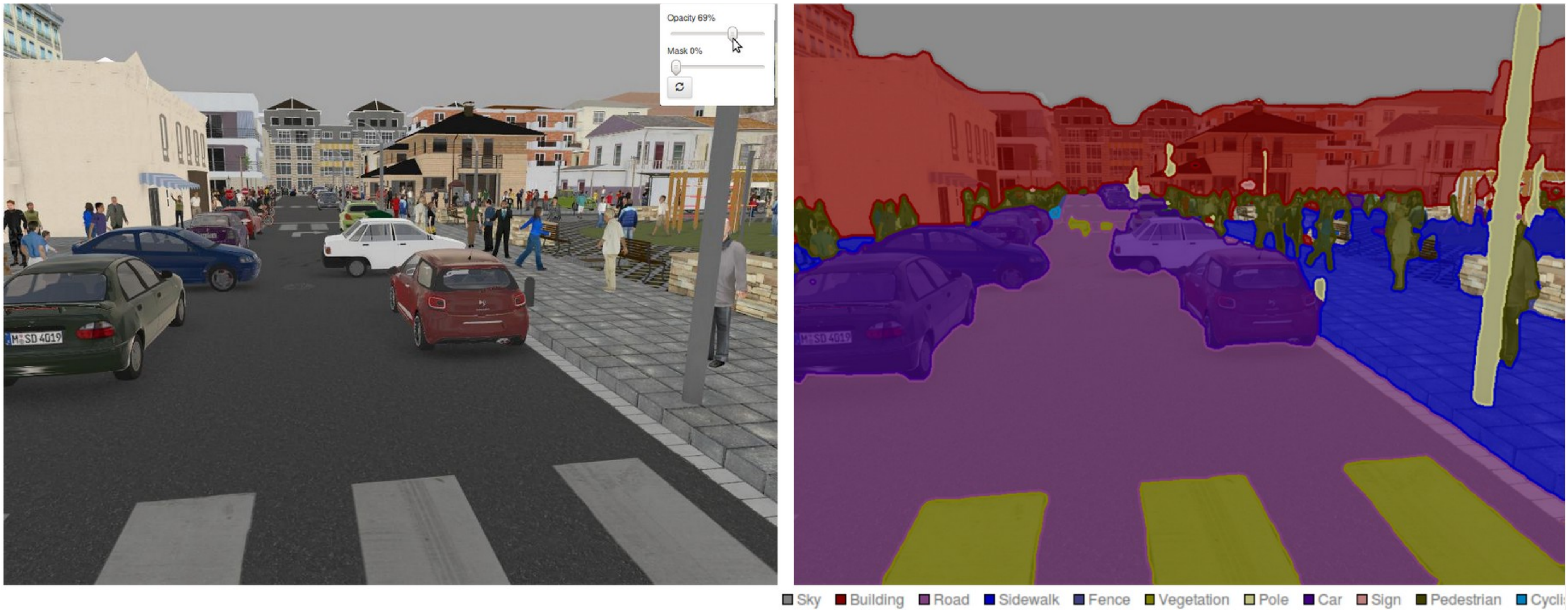
Classification vs Segmentation vs Detection



source

Classification vs **Segmentation** vs Detection

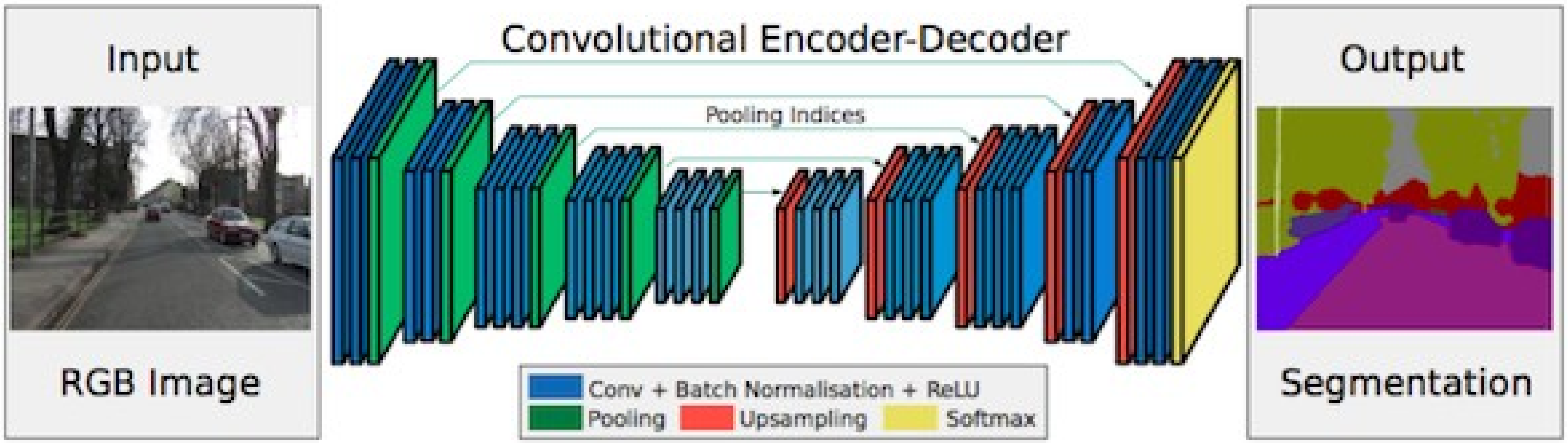
KITTI, CityScaped – datasets for autonomous driving



source

Segmentation: mapping each pixel in an image to an object class

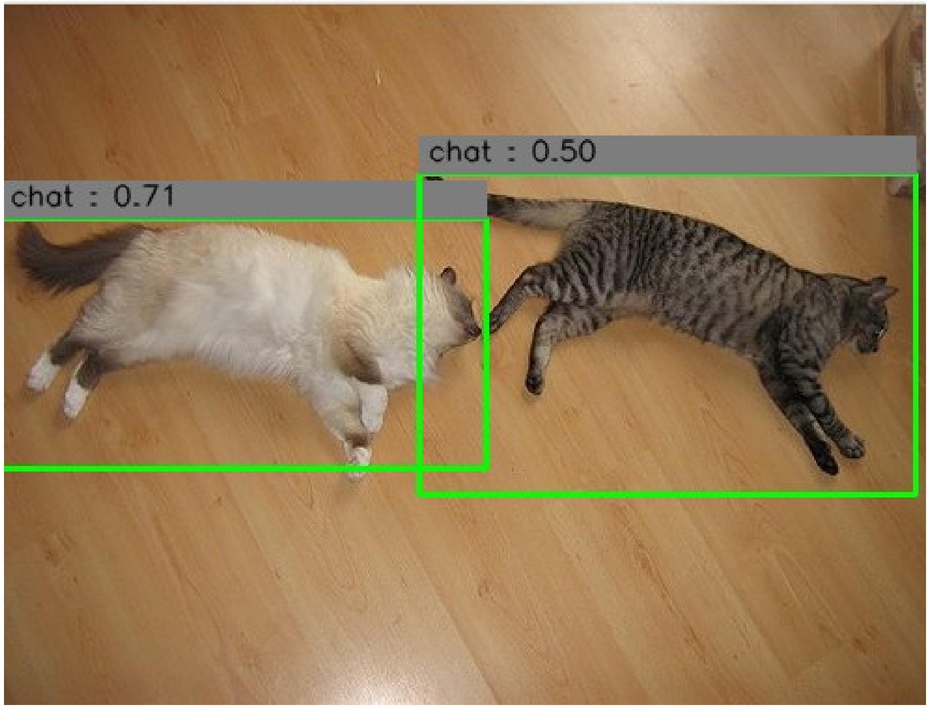
Classification vs **Segmentation** vs Detection



source

Classification vs Segmentation vs **Detection**

Datasets: ImageNet, MS COCO, Pascal VOC



Detection: what is present in an image and where.

Classification vs Segmentation vs **Detection**

Impact of deep learning on Object Detection

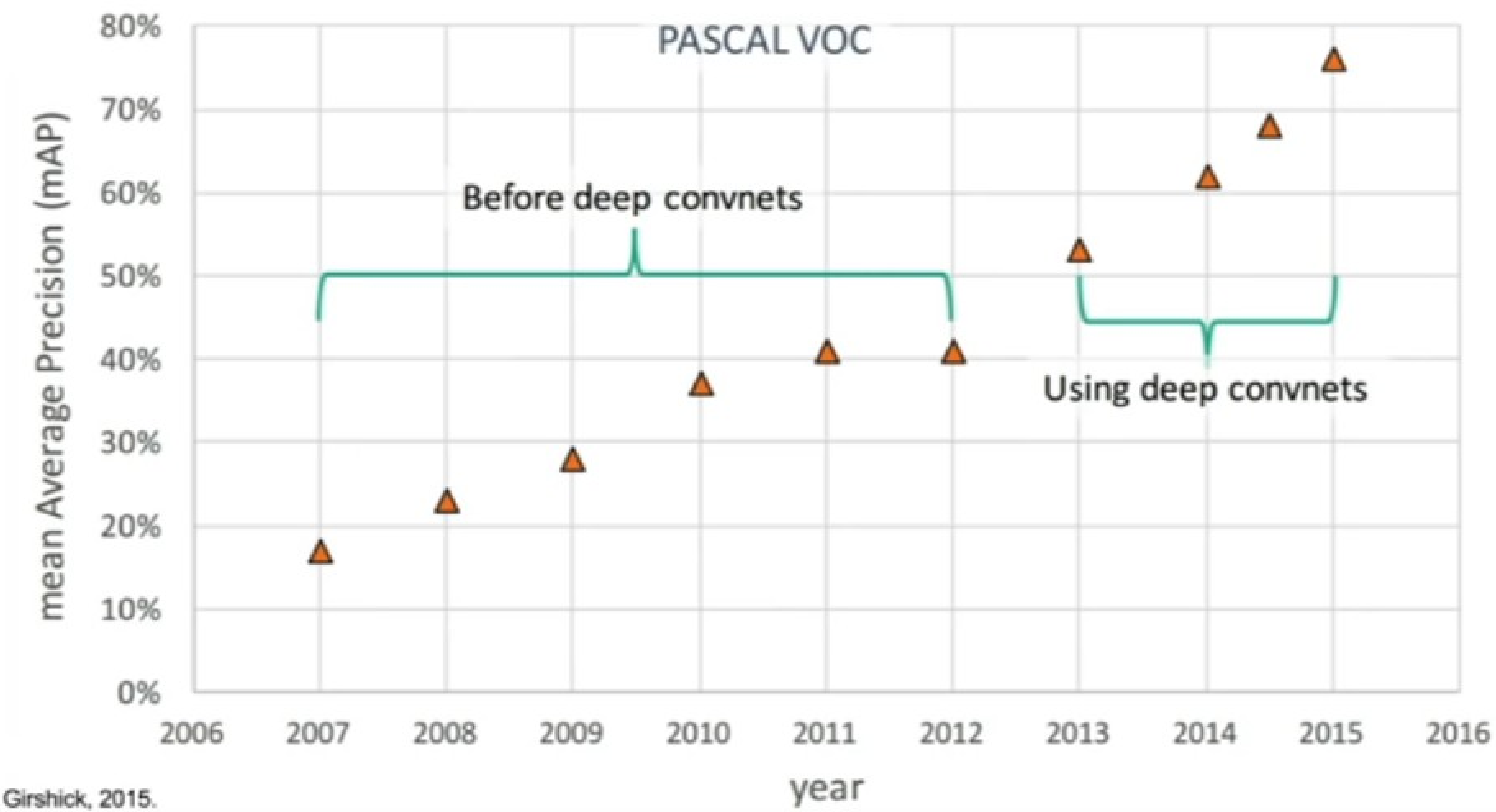


Figure copyright Ross Girshick, 2015.

source

Two-stages Detectors

Region-based detectors: R-CNN, Fast R-CNN, Faster R-CNN

(1) The model proposes a set of regions where an object might be located

(2) A classifier only processes these regions (because classifiers work well!)

Two-stages Detectors

Sliding window approach

Two-stages Detectors

Sliding window approach

Two-stages Detectors

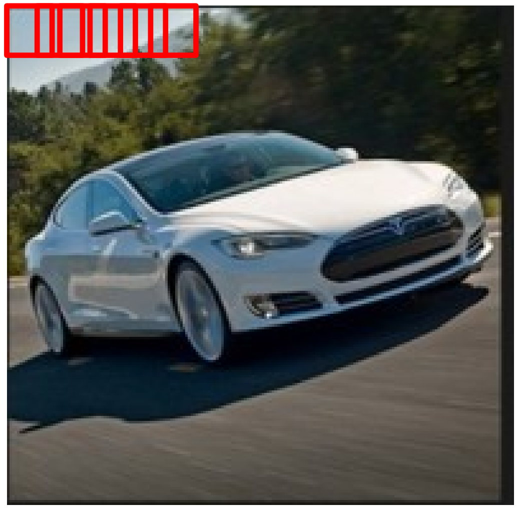
Sliding window approach

Two-stages Detectors

Sliding window approach

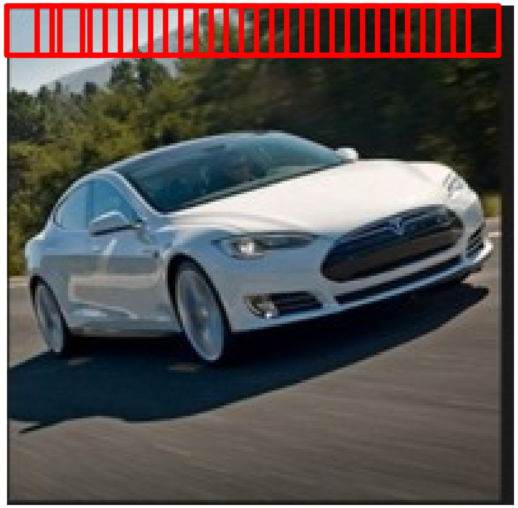
Two-stages Detectors

Sliding window approach



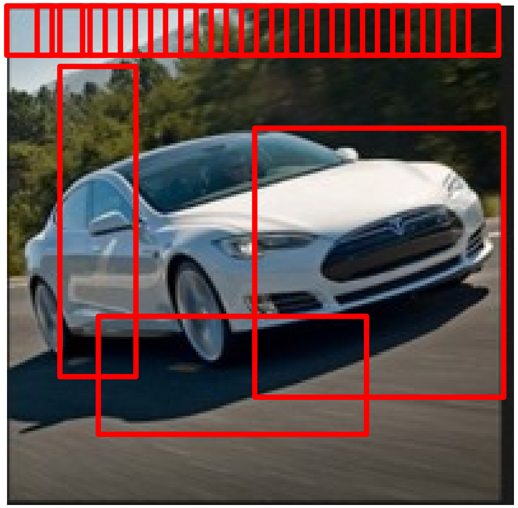
Two-stages Detectors

Sliding window approach



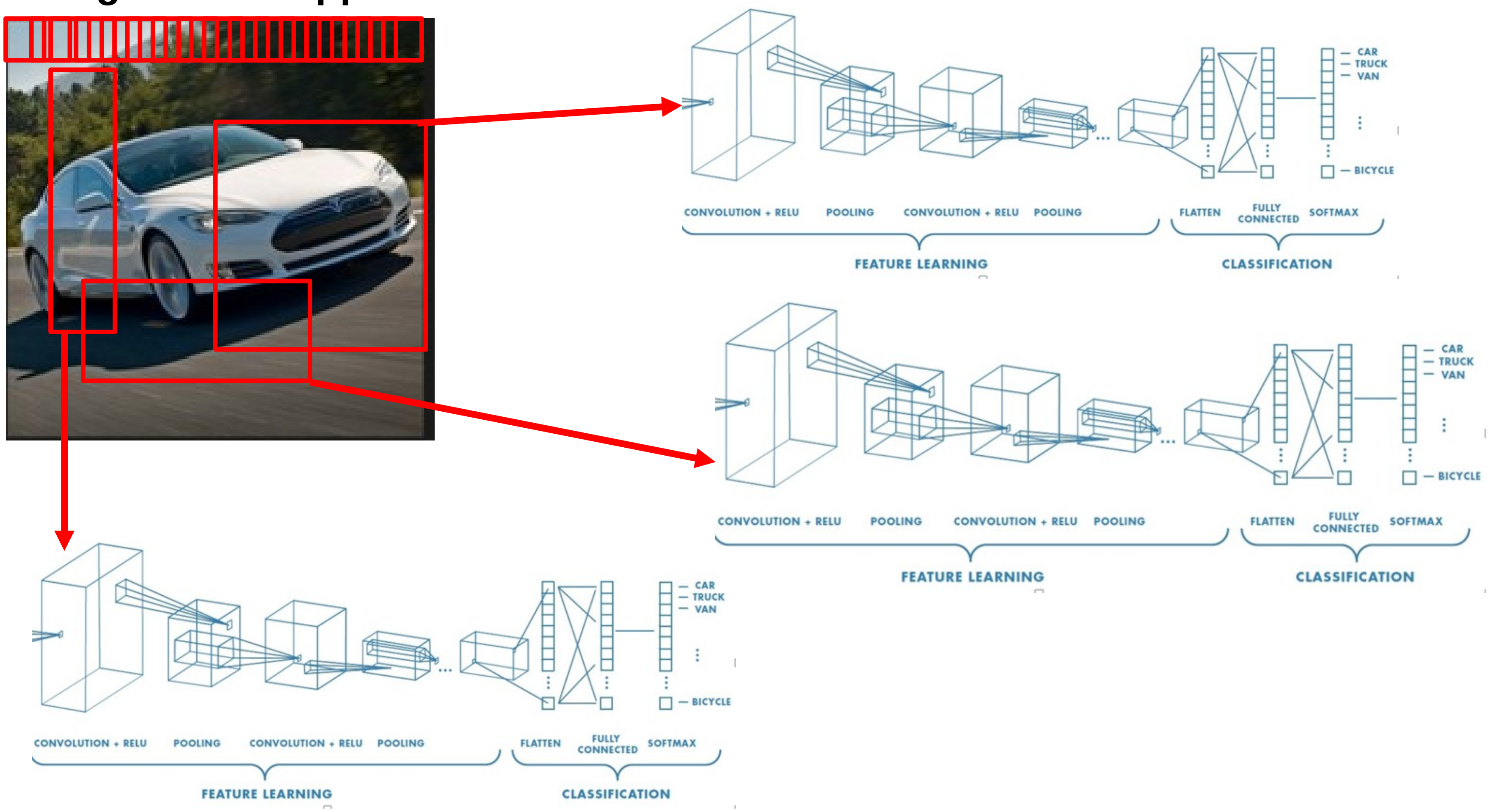
Two-stages Detectors

Sliding window approach



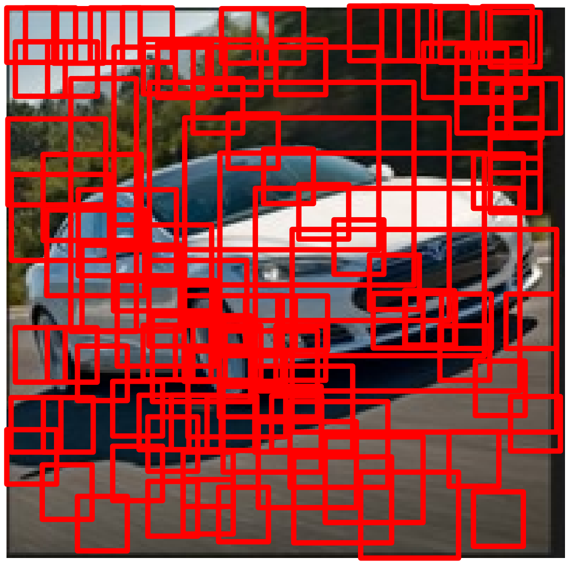
Two-stages Detectors

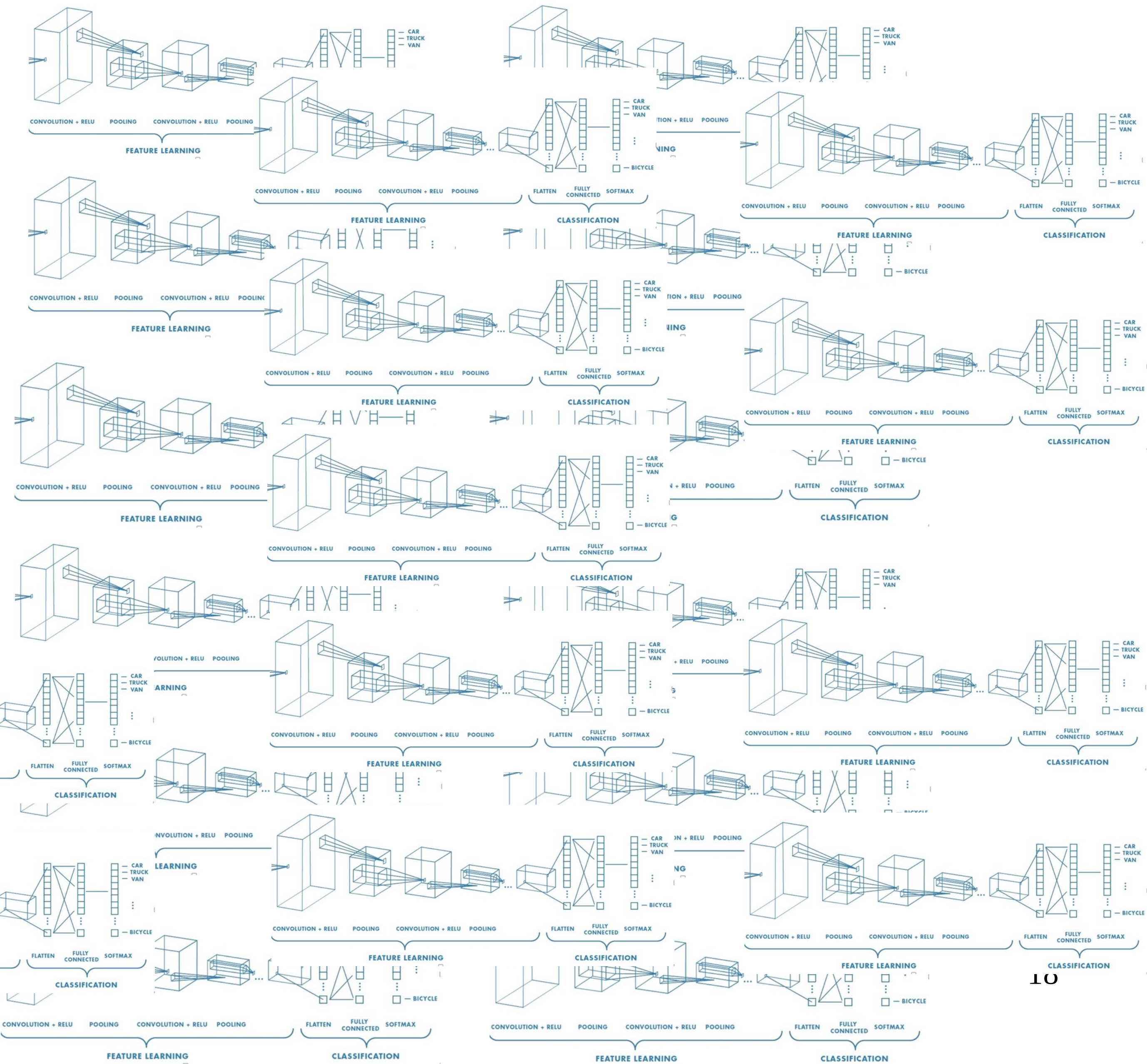
Sliding window approach

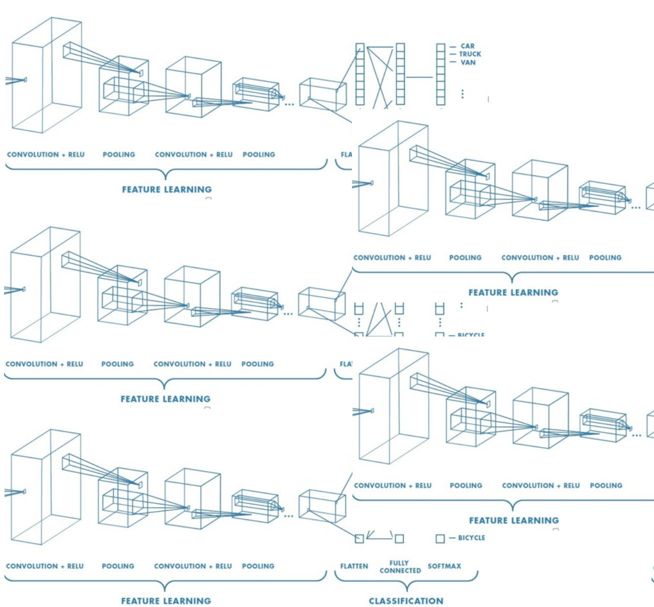


Two-stages Detectors

Sliding window approach







R-CNN (3 stages)

(1) Generate category-independent region proposals

(2) A CNN extracts a fixed length feature vector for each region

(3) A class-specific linear SVM (support vector machine) is used for classification

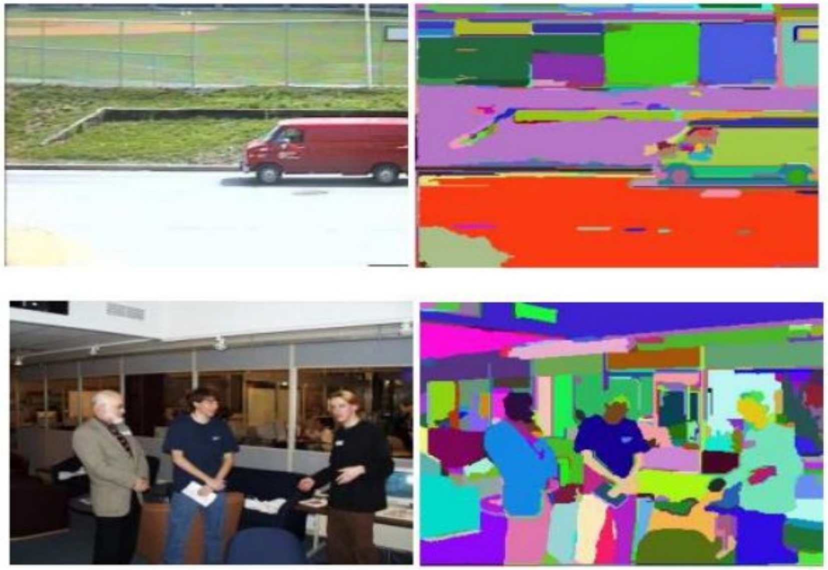
R-CNN (Nov. 2013)

(1) Generate category-independent region proposals

Based on **selective search**, an April 2013 SOTA algorithm for object detection

Won ImageNet object detection competition in 2011 !! And Pascal VOC 2012 competition !!

Selective search is based on “Efficient Graph-Based Image Segmentation”, 2004.



source

R-CNN (Nov. 2013)

(1) Generate category-independent region proposals

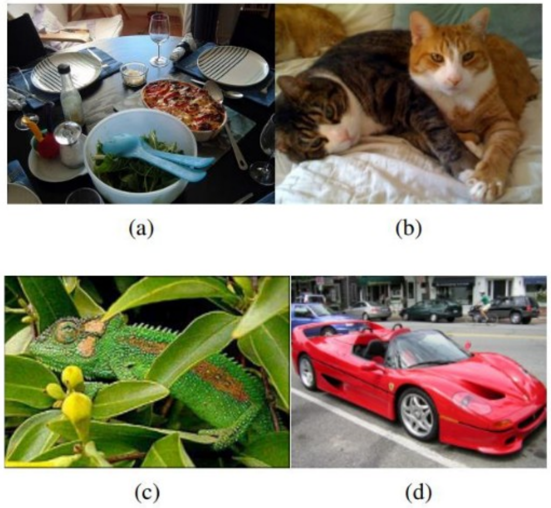
Selective search – starts with over-segmentation (bottom grouping) merges similar regions and produce region proposals

– graph-based image segmentations to get small starting regions

– similarities between neighborhood regions are computed: the most similar regions are grouped together

– new similarities are computed between the resulting regions and the neighbors

– hierarchical grouping to deal with different scales



source

R-CNN (Nov. 2013)

(1) Generate category-independent region proposals

Selective search – produces ~ 2000 region proposals

(2) A CNN extracts a fixed length feature vector for each region

AlexNet architecture (**SOTA in 2012 for image classification**)

Extracts a fixed number of feature: 4096 from each region proposal

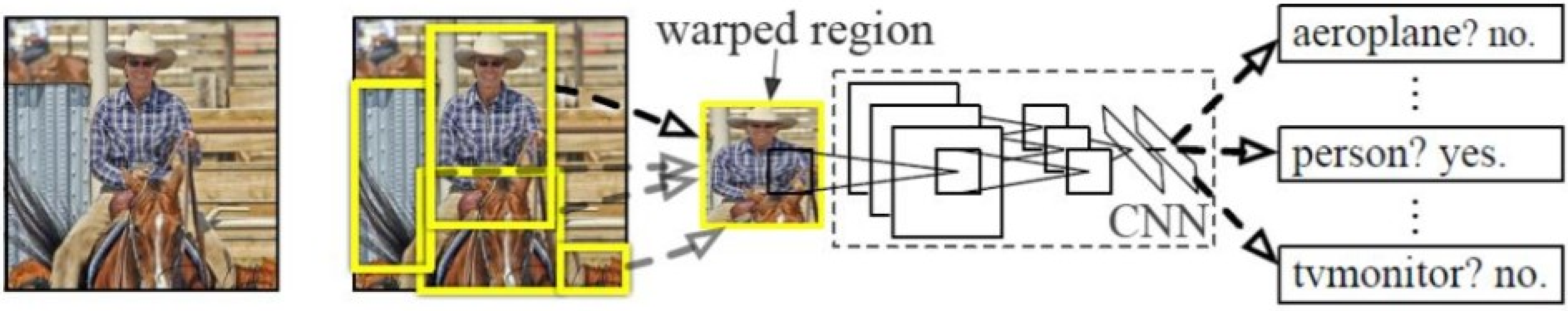
Images with fixed input size: 256 x 256

R-CNN (Nov. 2013)

(1) Generate category-independent region proposals

Selective search – produces ~ 2000 region proposals

(2) A CNN extracts a fixed length feature vector for each region



source

R-CNN (Nov. 2013)

(1) Generate category-independent region proposals

Selective search – produces ~ 2000 region proposals

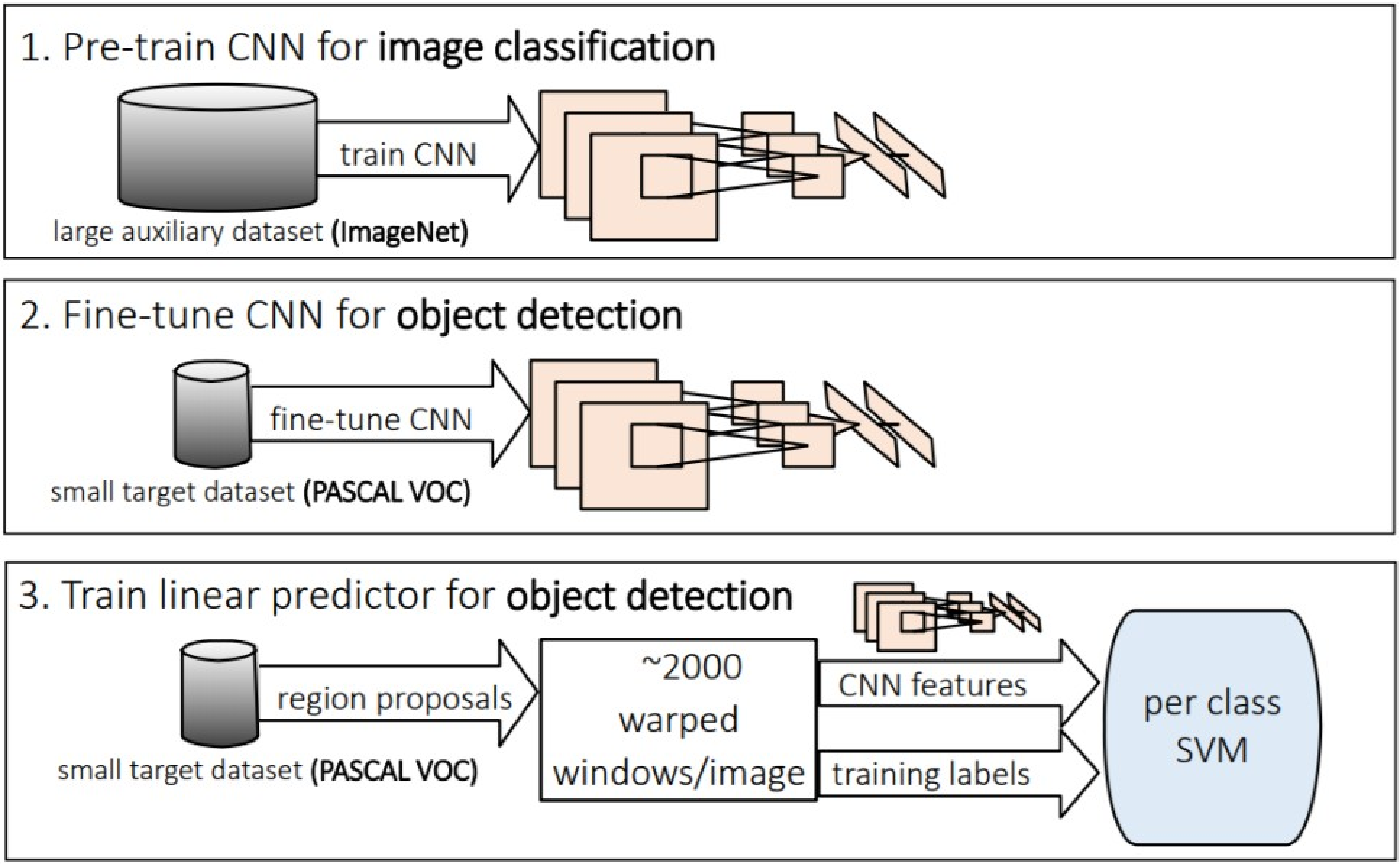
(2) A CNN extracts a fixed length feature vector for each region

(3) A class-specific linear SVM (support vector machine) is used for classification

- dot products between features and SVM weights
- non-maximum suppression
- bounding box regression for small adjustments after classification

R-CNN (Nov. 2013)

Training



R-CNN (Nov. 2013)

Observations

Far from real-time

Training is slow and takes a lot of disk space

Much of the network's representational power comes from the convolutional layers

CNN classification results on ImageNet can generalize to object detection on Pascal VOC

Fast R-CNN (Apr. 2015)

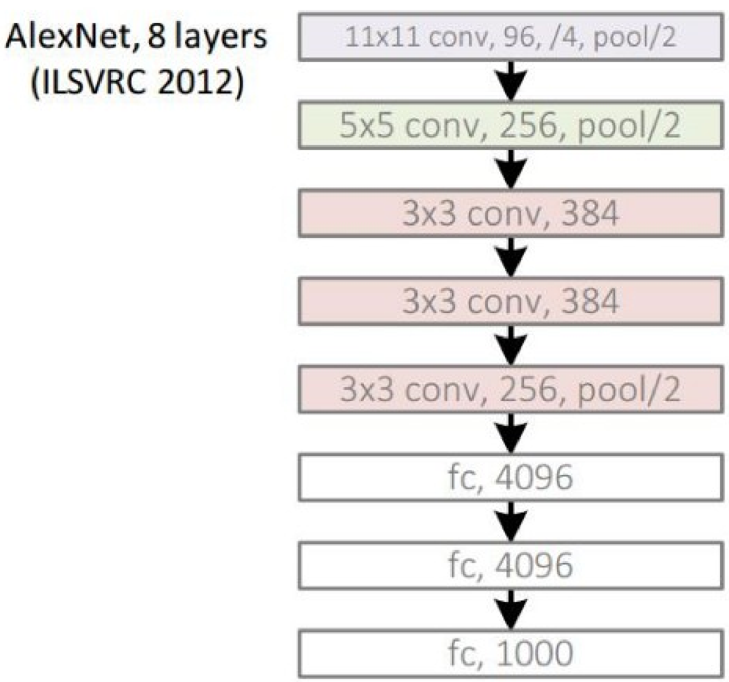
Fast R-CNN (Apr. 2015)

R-CNN	**Fast R-CNN**
(1) Selective search for object proposals	(1) Selective search for object proposals
(2) Run AlexNet for each of the 2000 proposals (**no sharing computations**)	(2) Run VGG16 for the whole image (**sharing computations**)
(3) Refinement for spatial location	(3) **Jointly learns to classify object proposals and refine their spatial locations**

Fast R-CNN (Apr. 2015)

AlexNet was replaced with **2014 SOTA for image classification**, VGG

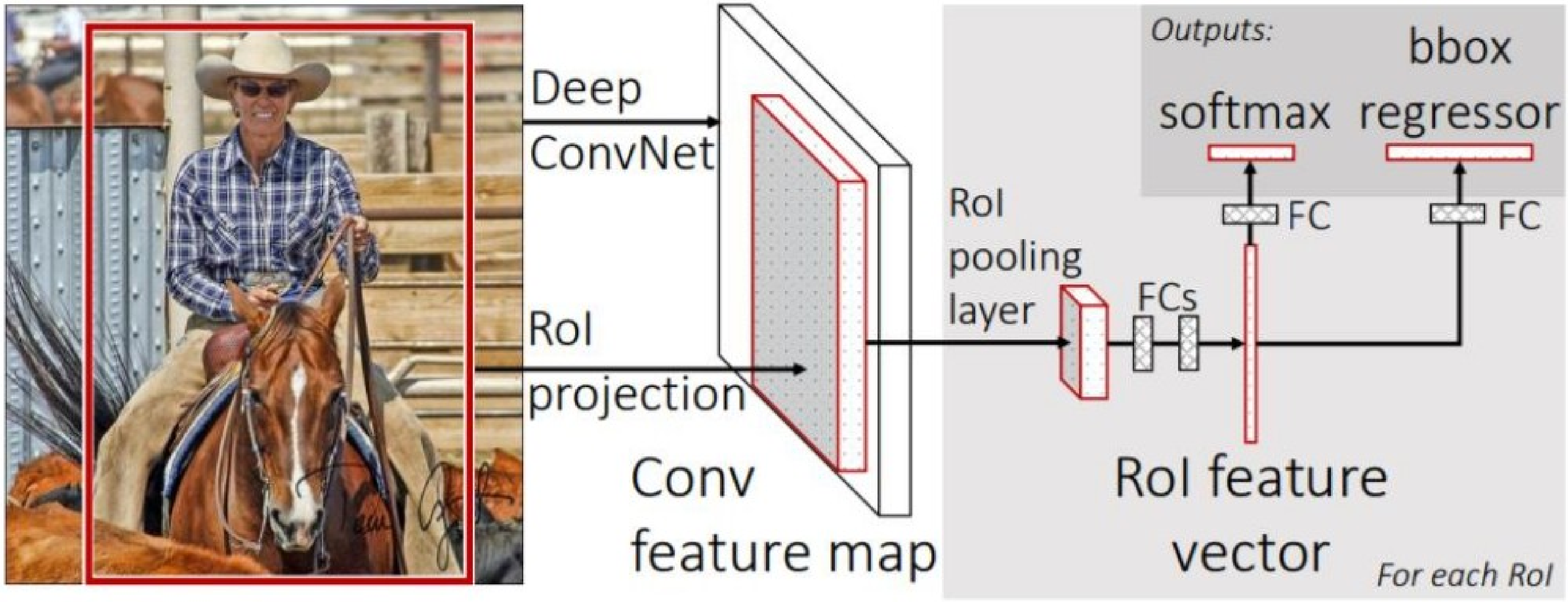
R-CNN



Fast R-CNN

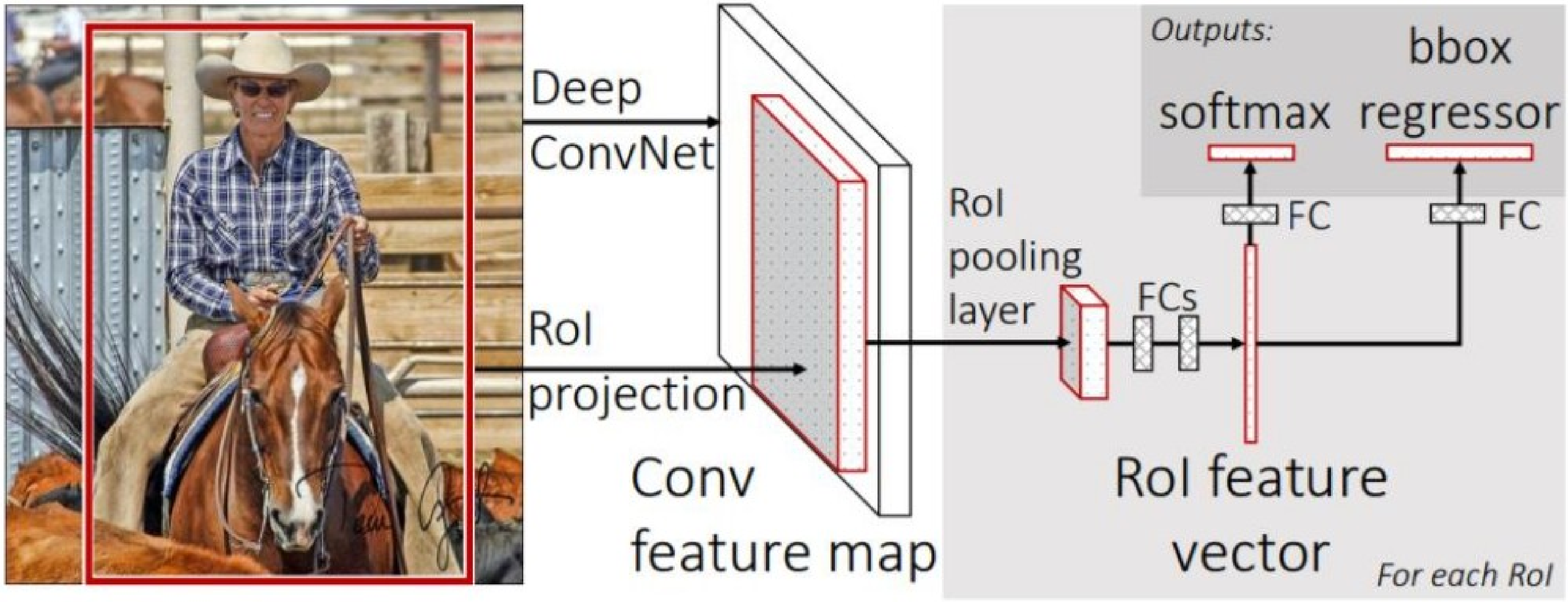
Fast R-CNN (Apr. 2015)

Architecture:



Fast R-CNN (Apr. 2015)

Architecture:



Observation:

Region proposals (selective search) remains the bottleneck at detection time

Fast R-CNN (Apr. 2015)

	R-CNN	Fast R-CNN
	AlexNet	**VGG-16**
SOTA	2012 ILSVRC winner (classification)	**2014 ILSVRC winner (ensemble, classification)**
Train		**9x faster**
Test		**25x faster**
mAP (VOC 2007 test)	58.5 %	**66.9 %**

(*) R-CNN using VGG-16 obtains 66 % mAP

How to make Fast R-CNN faster ?

Faster R-CNN (Jun. 2015)

Region proposals (selective search)

Bottleneck at detection time

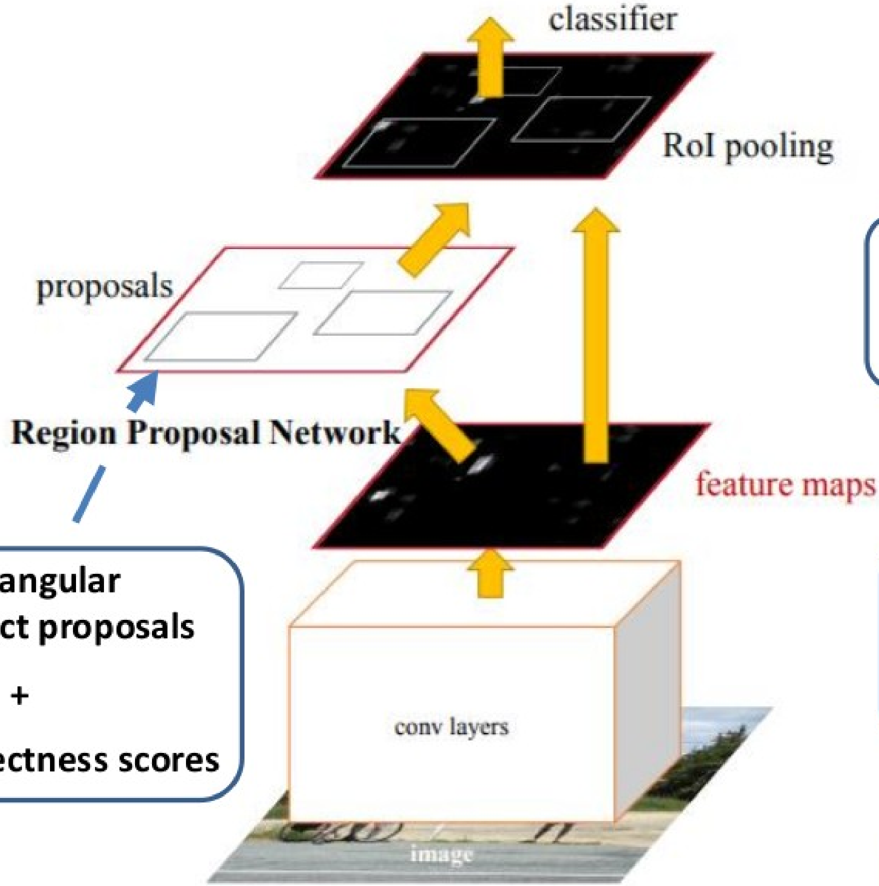
Takes 2 sec / image to compute

SOLUTION:

Compute proposals with a deep convolutional neural network

Use Region Proposal Networks (RPNs)

Faster R-CNN (Jun. 2015)



Rectangular object proposals

+

Objectness scores

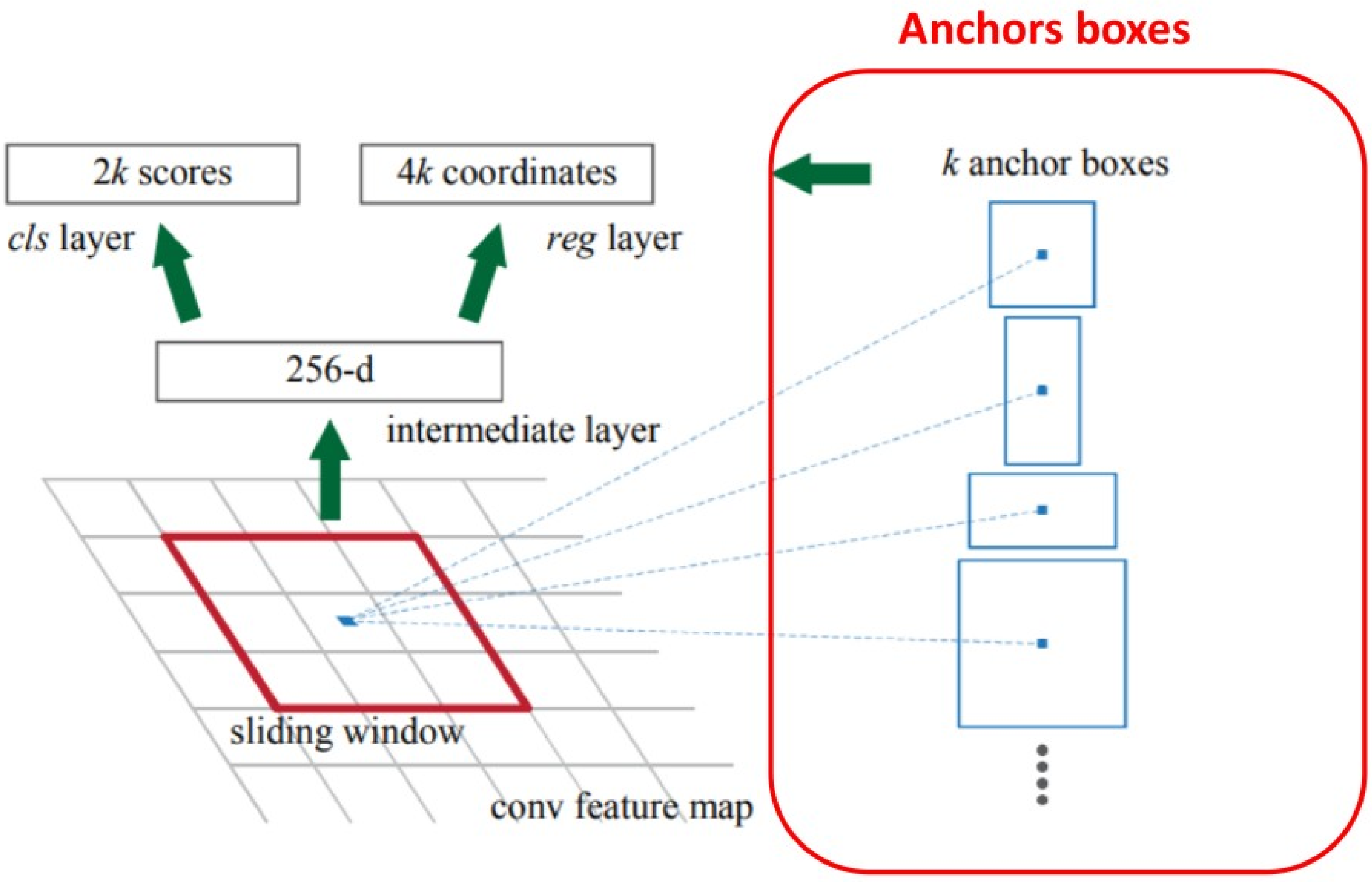
Shares convolutional features with the detection network

Can be used apart from Fast R-CNN or merged with Fast R-CNN into a single network

Takes 10 ms / image to compute

Faster R-CNN (Jun. 2015)

Region proposal networks (RPNs)



Deals with different scales and ratios

Faster R-CNN (Jun. 2015)

The loss function

$$\boxed{L(p_i, t_i) \quad = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)}$$

p_i **- predicted probability of anchor *i* being an object**

p_i^* **- ground truth label, 1 if the anchor is positive, 0 otherwise**

t_i **- predicted bounding-box coordinates**

t_i^* **- ground truth bounding-box coordinates**

N_{cls} **- number of anchors in an image (256)**

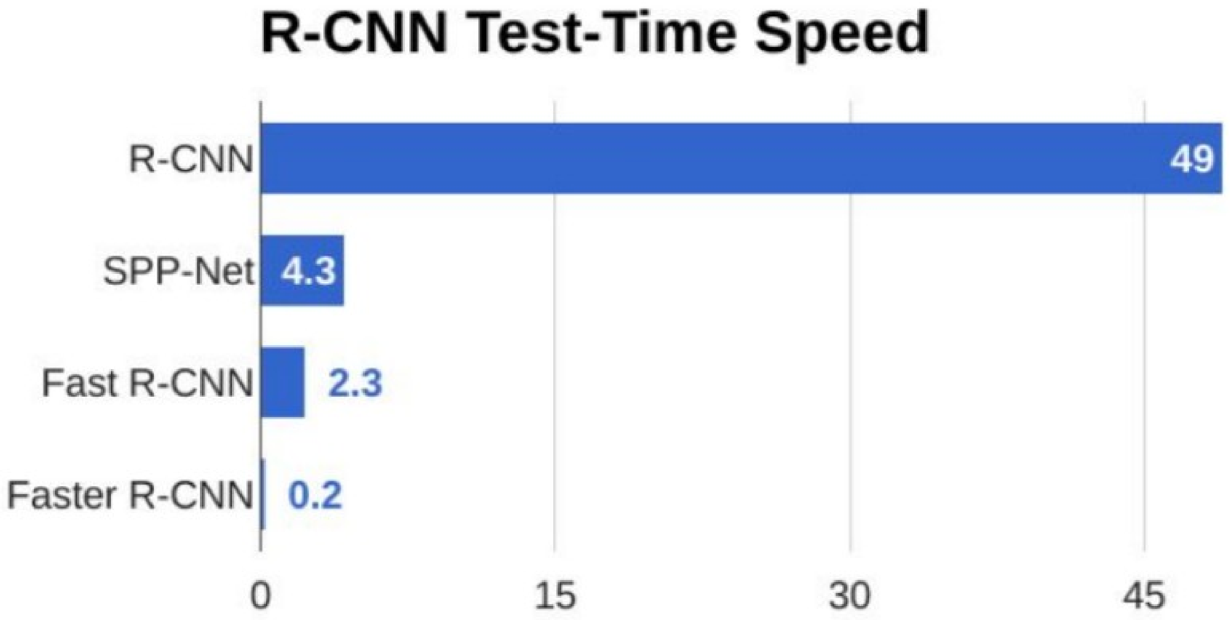
L_{cls}**- log loss over two classes (object vs non-object)**

λ **– balancing parameter**

N_{reg} **– number of anchor locations**

L_{reg}**– regression loss (activated only for positive anchors)**

Comparing inference runtimes



	R-CNN	Fast R-CNN	Faster R-CNN
mAP (VOC 2007 test)	66 %	66.9 %	70 %

Main Points

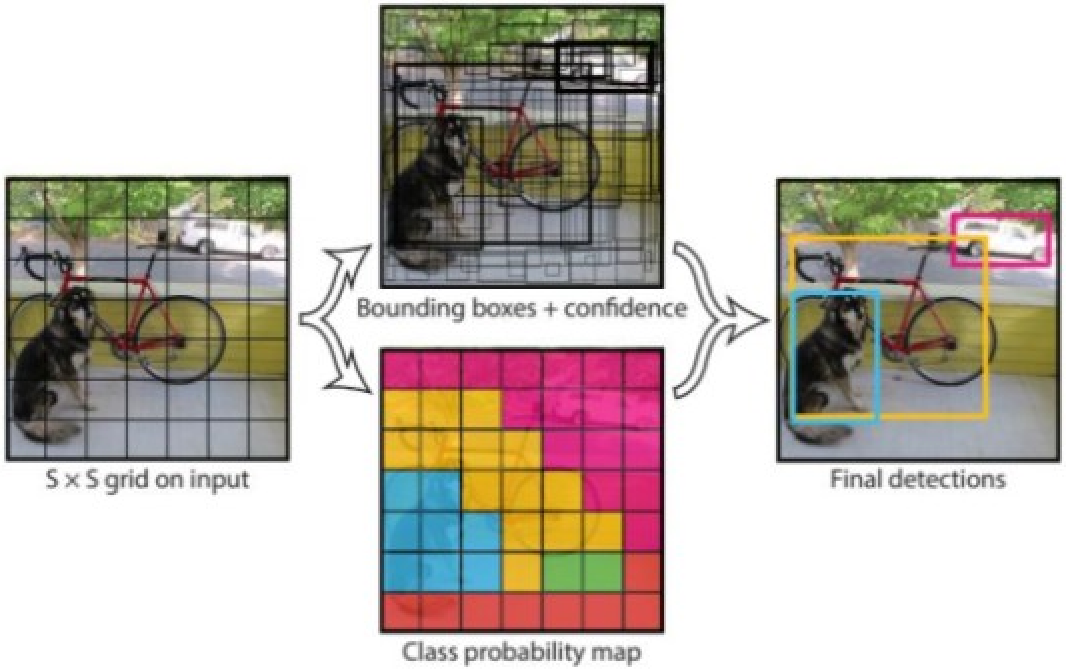
1. Two-stage detectors

2. Single-stage detectors

3. Demo

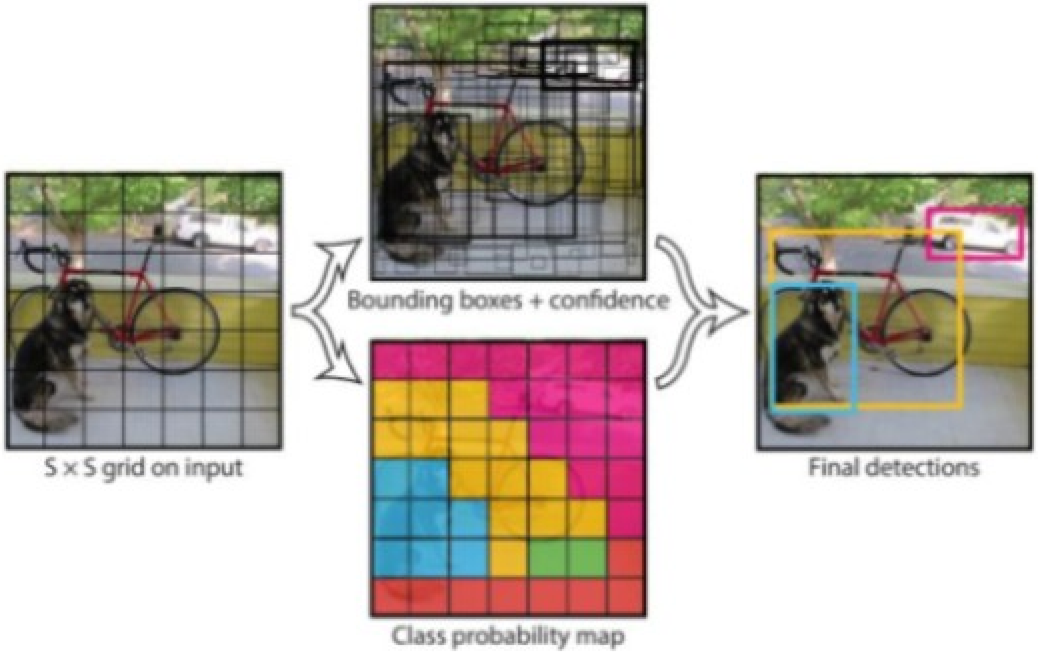
YOLO v1 (Jun. 2015)

Detection as a single regression problem



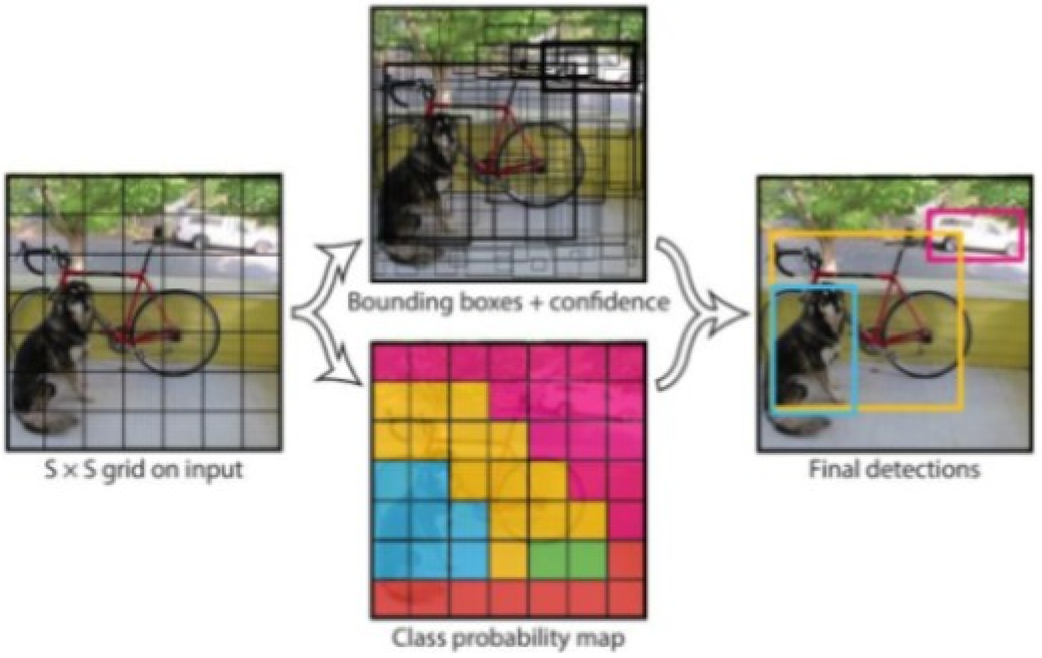
YOLO (You Only Look Once)

YOLO v1 (Jun. 2015)



1. Divide the input image in a SxS grid
2. Each grid predicts: class probability and bounding boxes
3. Each bounding box : 4 coordinates (x,y,w,h) + confidence

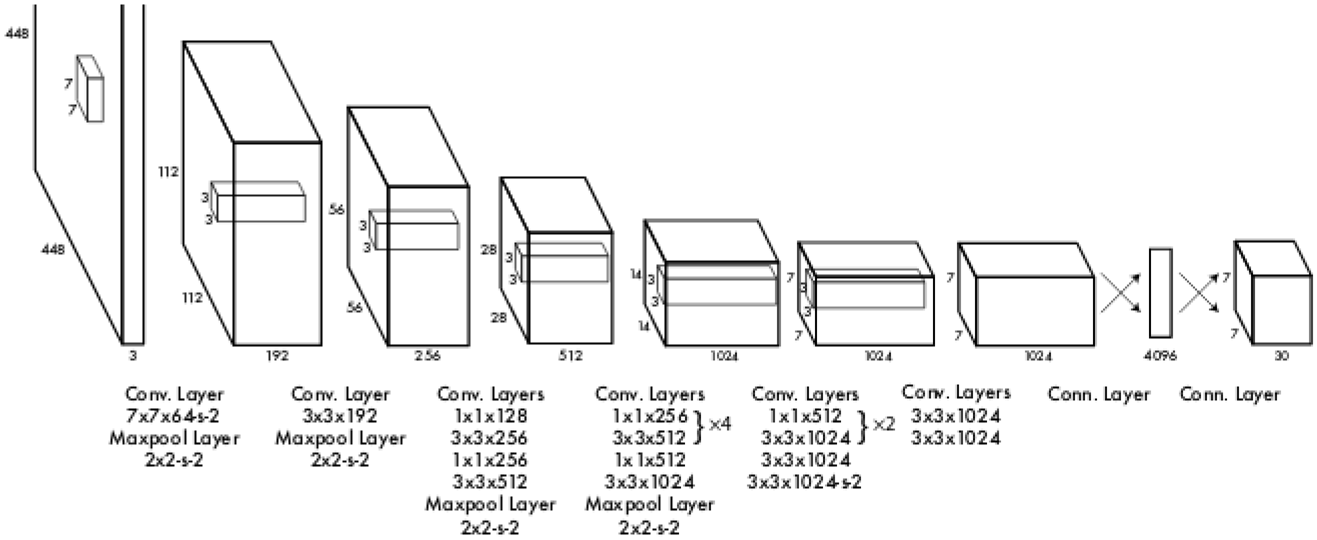
YOLO v1 (Jun. 2015)



1. Divide the input image in a SxS grid
2. Each grid predicts: class probability and bounding boxes
3. Each bounding box : 4 coordinates (x,y,w,h) + confidence

Predictions are encoded in a SxSx(Bx5+C) tensor

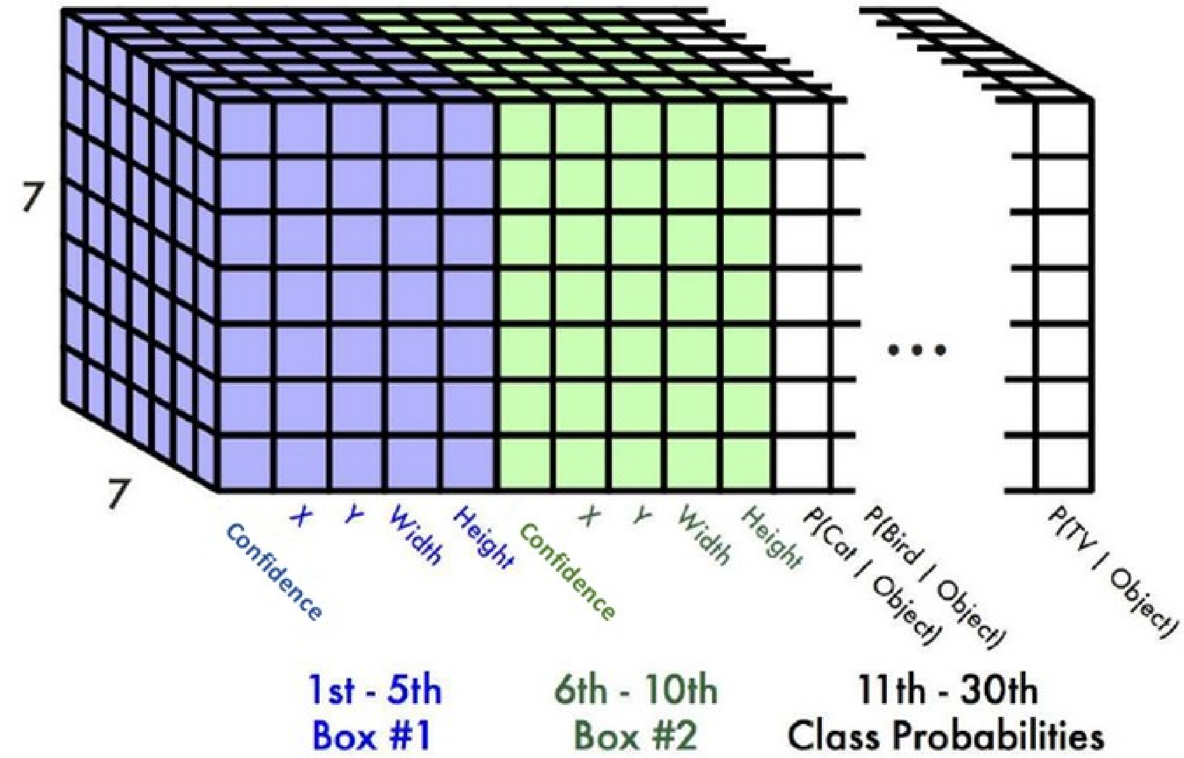
YOLO v1 (Jun. 2015)



1. Divide the input image in a SxS grid
2. Each grid predicts: class probability and bounding boxes
3. Each bounding box : 4 coordinates (x,y,w,h) + confidence

Predictions are encoded in a SxSx(Bx5+C) tensor

YOLO v1 (Jun. 2015)



The output from YOLO

The output size becomes: $7 \times 7 \times (2 \times 5 + 20) = 1470$

source

YOLO v1 (Jun. 2015)

Observations

Problems in detecting small and far objects

Faster than Faster R-CNN but real-time only on GPU

Lower MAP than Faster R-CNN

Proved that we can do classification and detection in a single shot

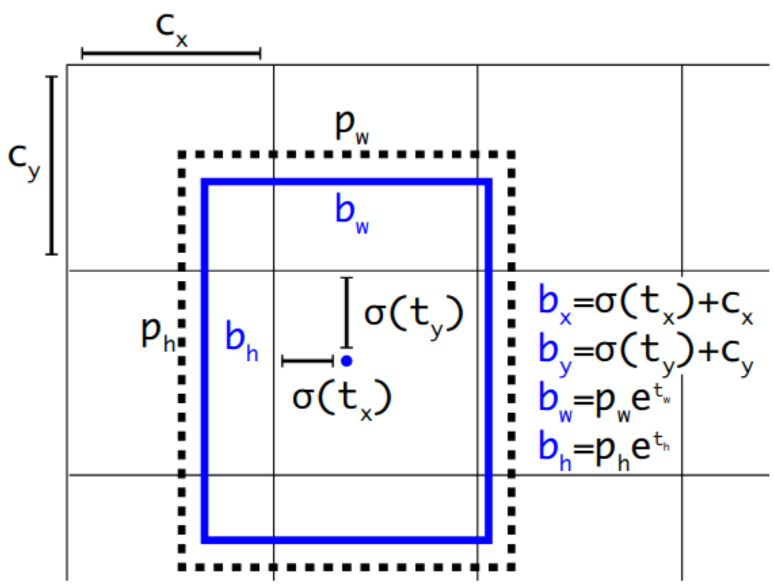
YOLO v2 (Dec. 2016)

Changes:

1. Batch normalization

2. Higher resolution images: from 256x256 to 448x448

3. Introduction of anchor boxes



predict the width and height of the box
as offsets from cluster centroid

YOLO v2 (Dec. 2016)

Changes:

1. Batch normalization

2. Higher resolution images: from 256x256 to 448x448 at training time

3. Introduction of anchor boxes

4. Multi-scale training

   Improves detection of small objects

YOLO v2 (Dec. 2016)

Cost function:

$$L = \lambda_{\text{coord}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{\text{ij}}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{\text{coord}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{\text{ij}}^{\text{obj}} [(\sqrt{w_i} - \sqrt{\widehat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] +$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 + \sum_{i=0}^{s^2} 1_{\text{i}}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

YOLO v2 (Dec. 2016)

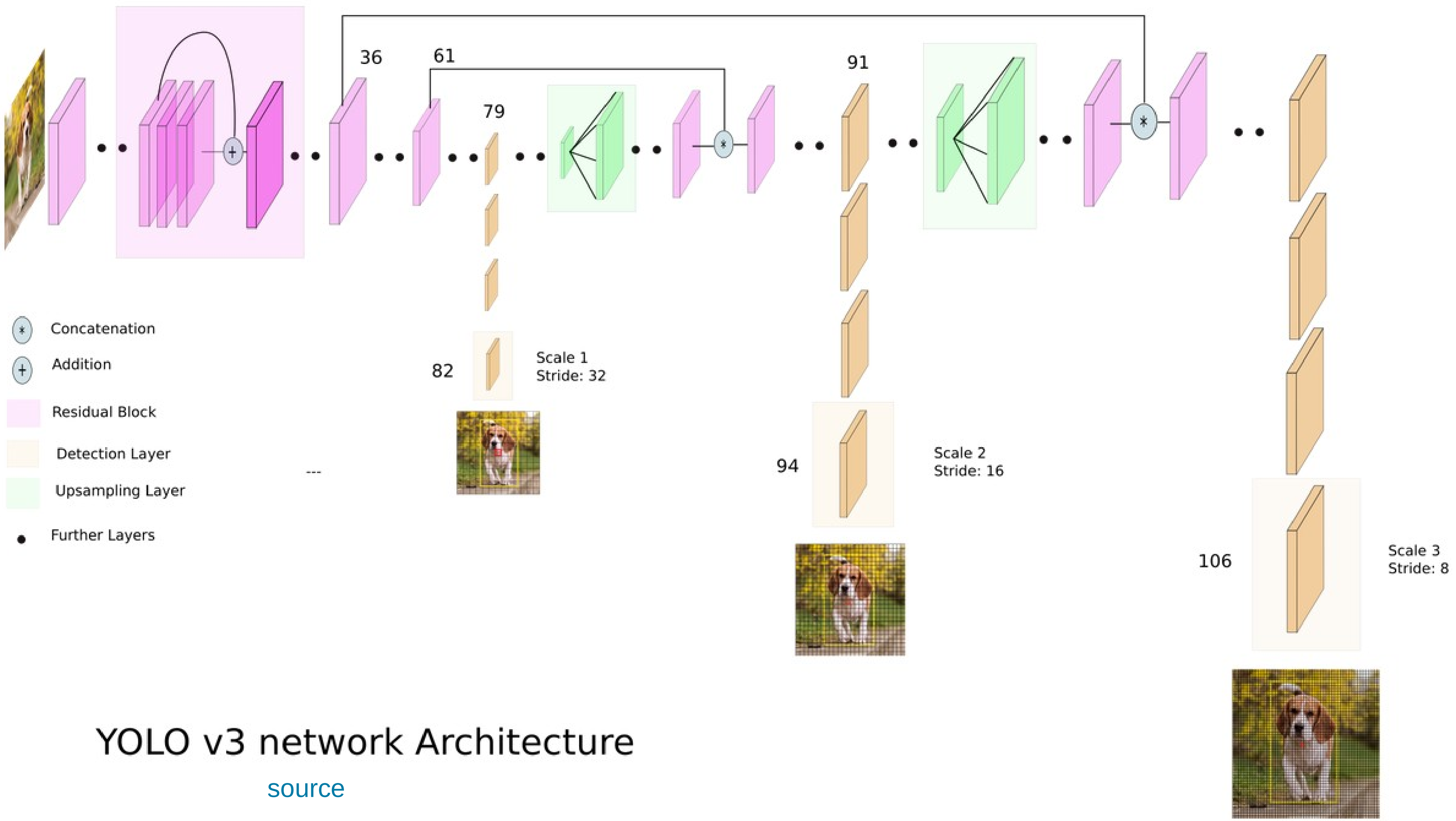
Results on Pascal VOC 2007

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288×288	2007+2012	69.0	91
YOLOv2 352×352	2007+2012	73.7	81
YOLOv2 416×416	2007+2012	76.8	67
YOLOv2 480×480	2007+2012	77.8	59
YOLOv2 544×544	2007+2012	**78.6**	40

FPS are calculated on a Titan X GPU

YOLO v3 (Apr. 2018)

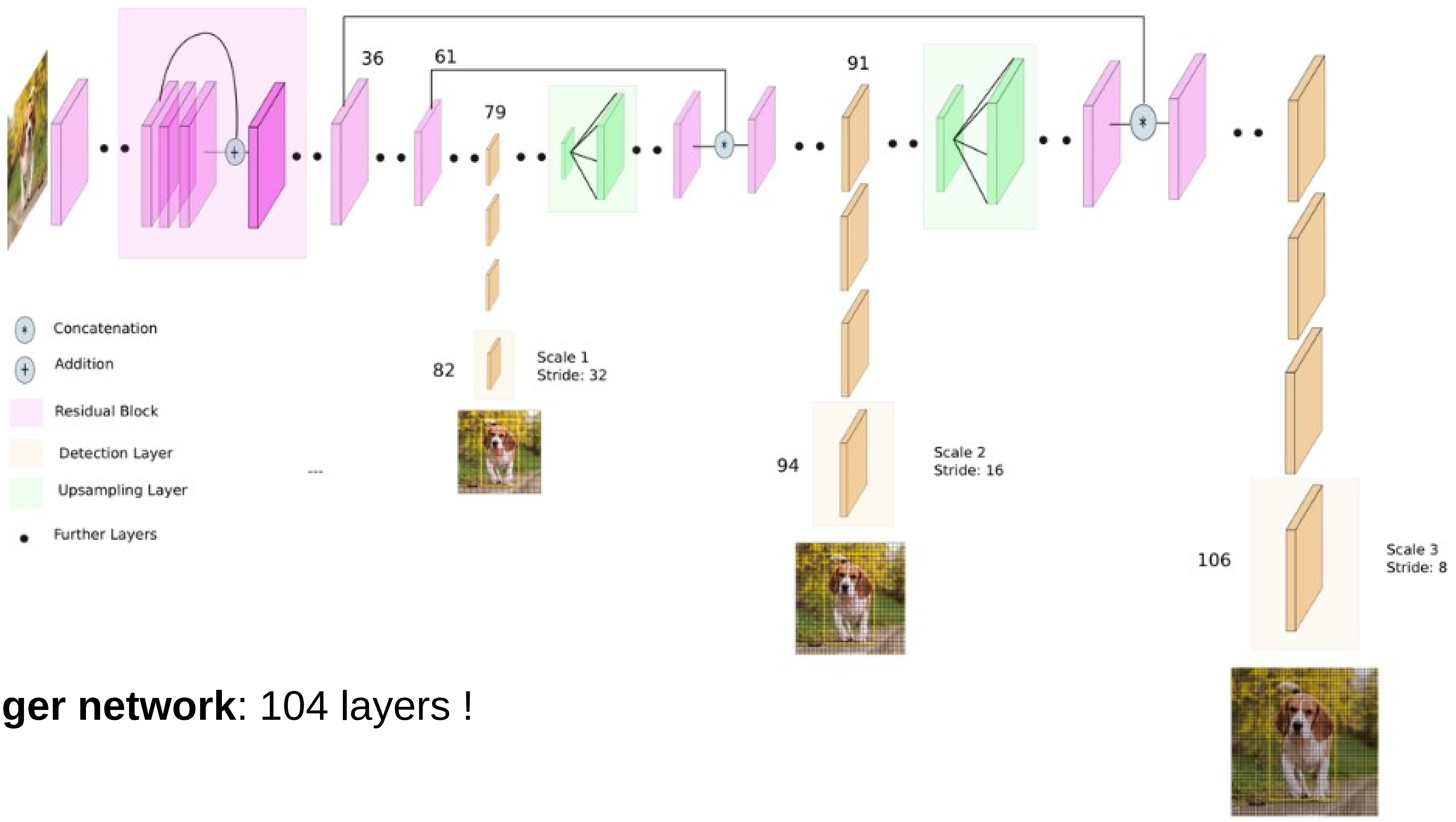
YOLO v3 (Apr. 2018)



YOLO v3 network Architecture

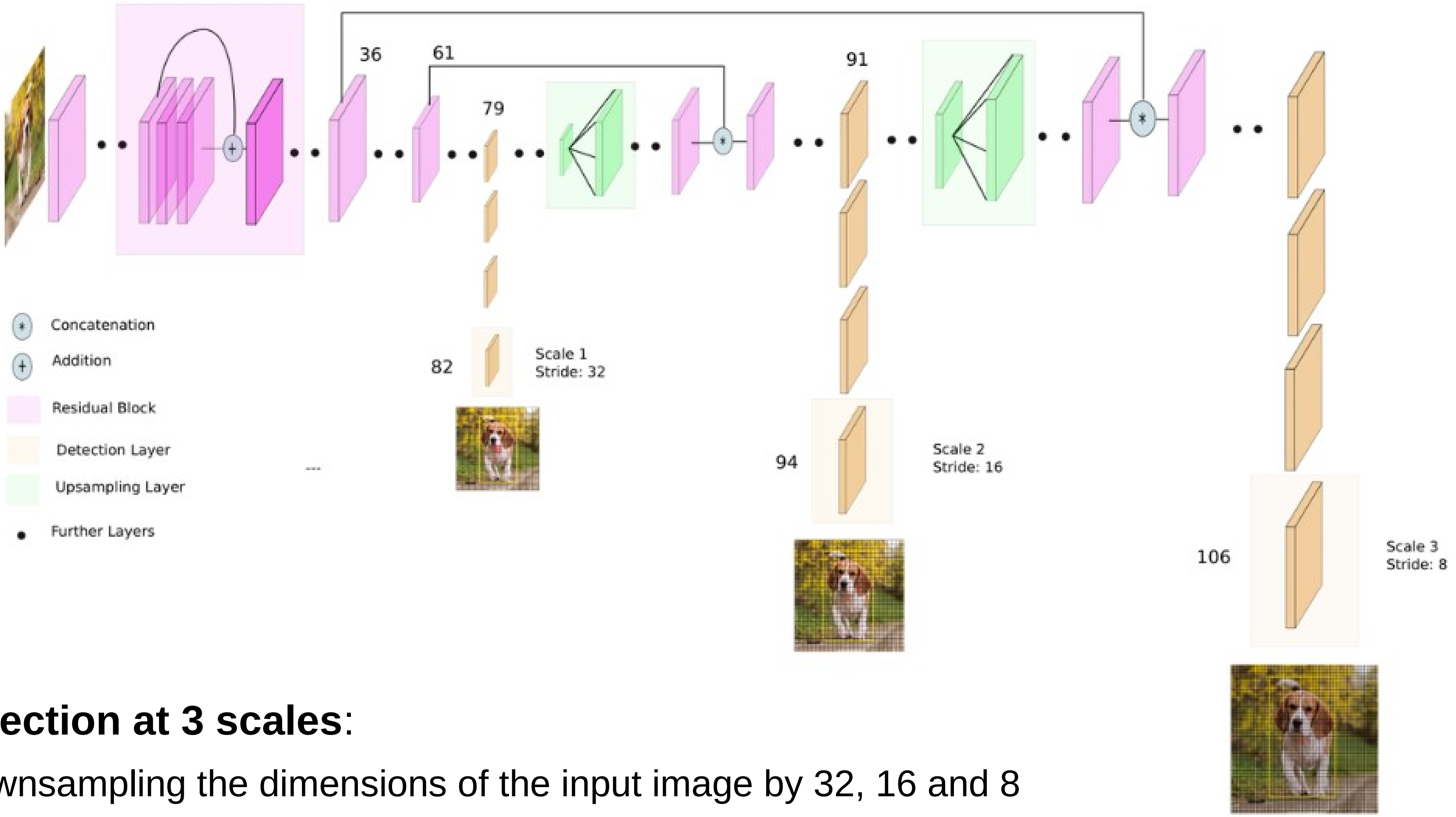
source

YOLO v3 (Apr. 2018)



Bigger network: 104 layers !

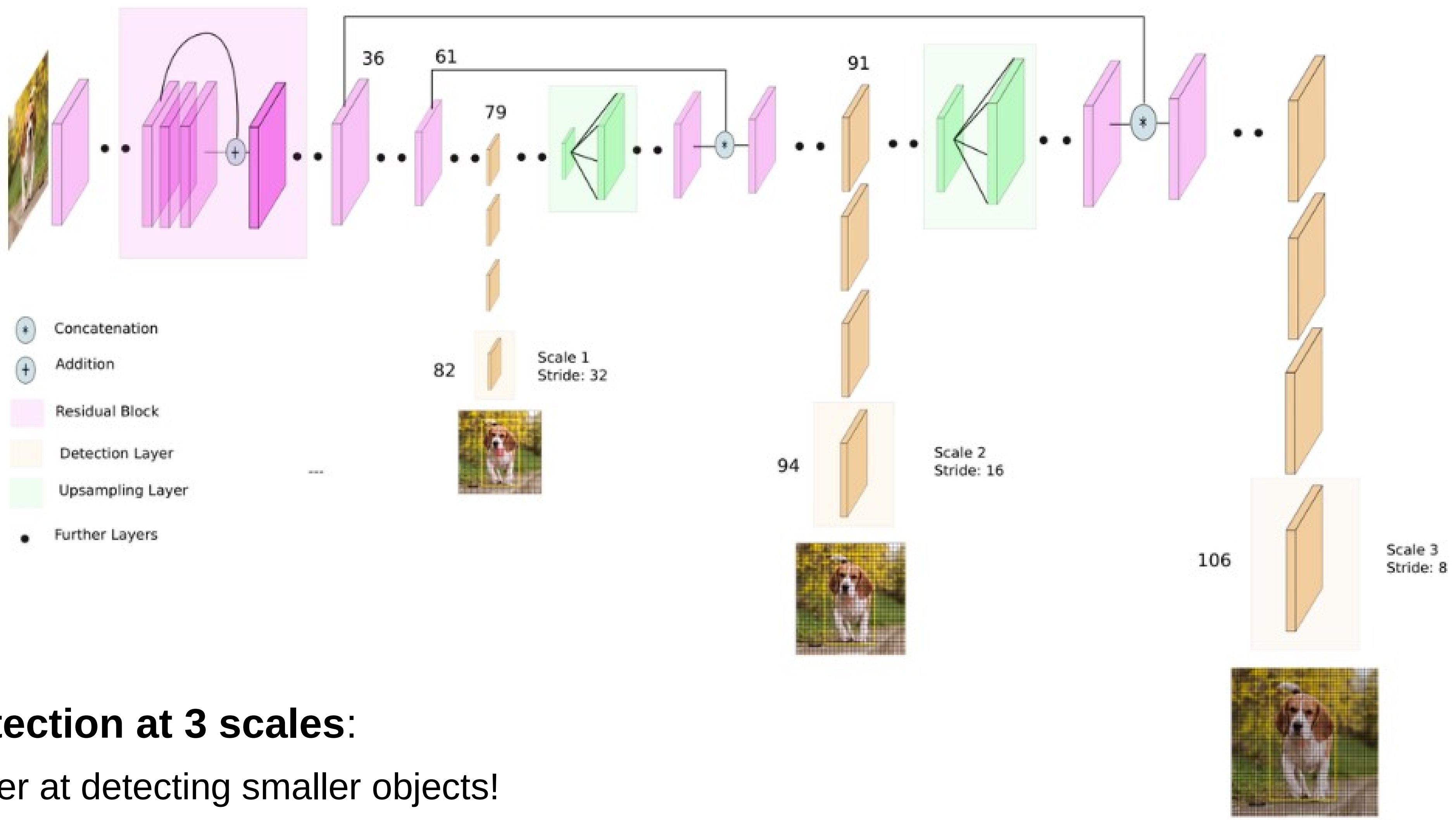
YOLO v3 (Apr. 2018)



Detection at 3 scales:

- downsampling the dimensions of the input image by 32, 16 and 8

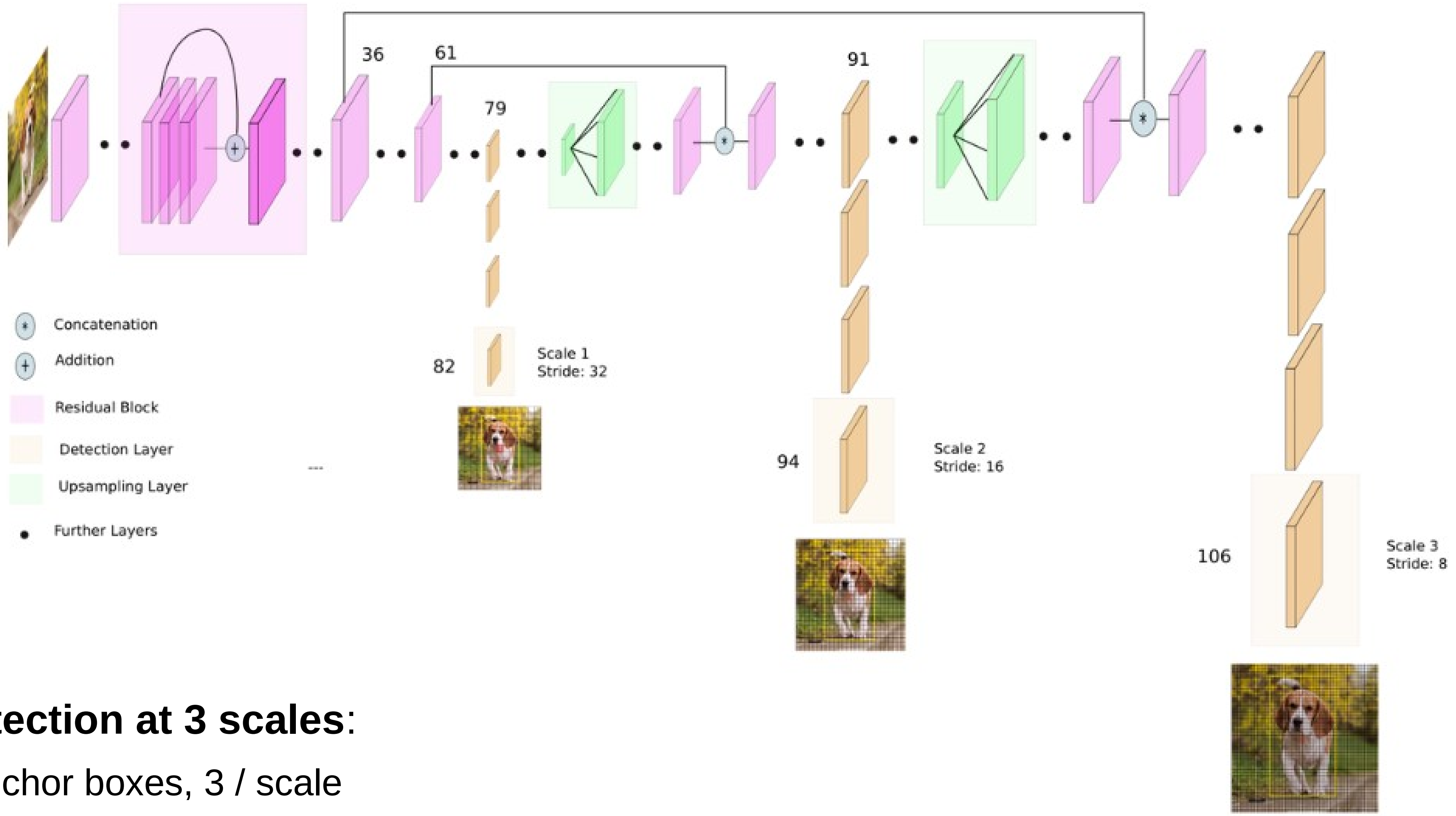
YOLO v3 (Apr. 2018)



Detection at 3 scales:

Better at detecting smaller objects!

YOLO v3 (Apr. 2018)



Detection at 3 scales:

9 anchor boxes, 3 / scale

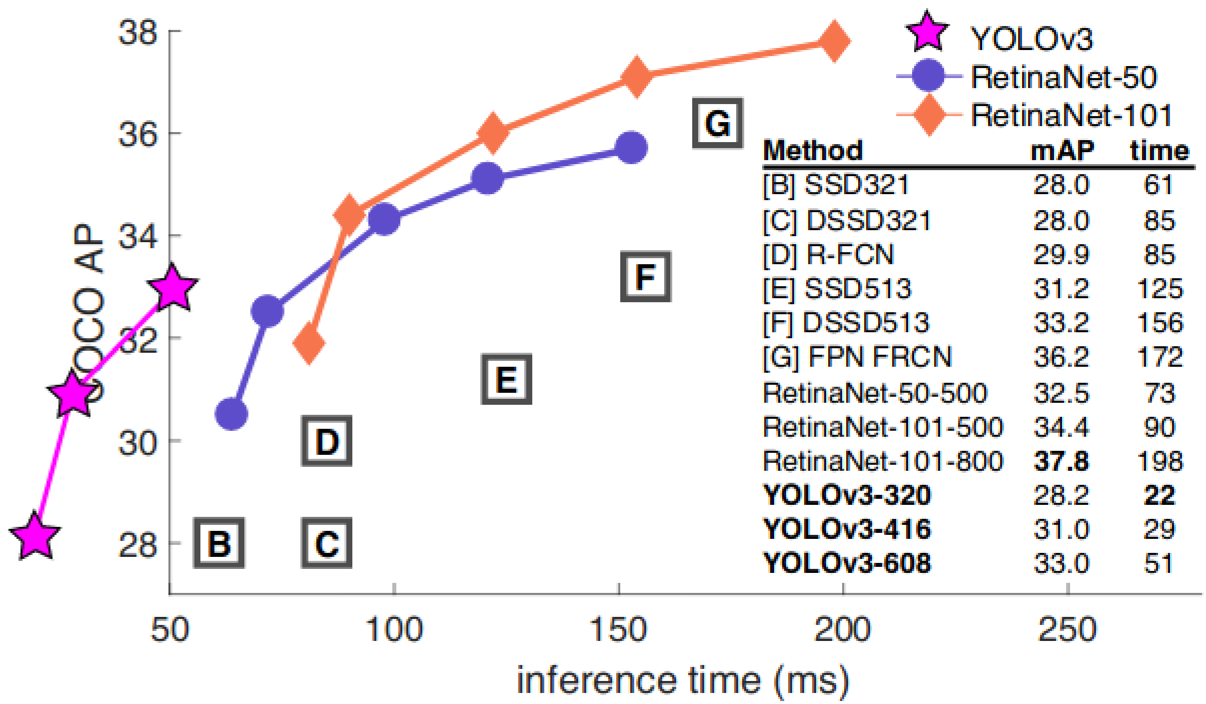
YOLO v3 (Apr. 2018)

Change in the loss function:

- instead of softmax of the class score, use logistic regression

YOLO v3 (Apr. 2018)

Results:



source

Merci pour your attention !!!