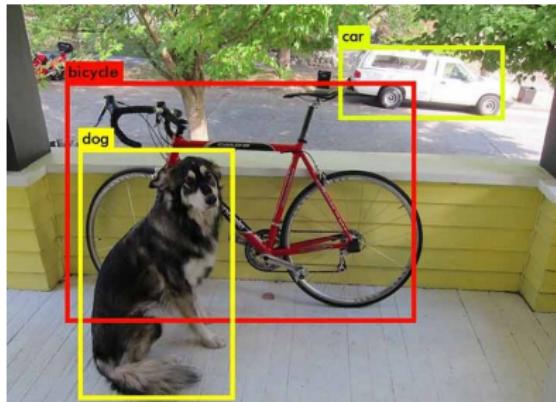


Object detection

Victor Kitov

v.v.kitov@yandex.ru

Problem statement



Training dataset:

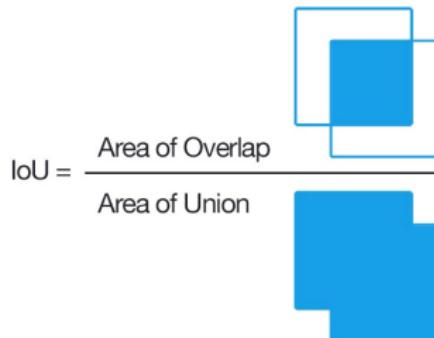
- img1.jpg: X(1), Y(1), X(2), Y(2), class
- img2.jpg: X(1), Y(1), X(2), Y(2), class
-

Popular object detection datasets

Dataset	train		validation		trainval		test	
	images	objects	images	objects	images	objects	images	objects
VOC-2007	2,501	6,301	2,510	6,307	5,011	12,608	4,952	14,976
VOC-2012	5,717	13,609	5,823	13,841	11,540	27,450	10,991	-
ILSVRC-2014	456,567	478,807	20,121	55,502	476,688	534,309	40,152	-
ILSVRC-2017	456,567	478,807	20,121	55,502	476,688	534,309	65,500	-
MS-COCO-2015	82,783	604,907	40,504	291,875	123,287	896,782	81,434	-
MS-COCO-2018	118,287	860,001	5,000	36,781	123,287	896,782	40,670	-
OID-2018	1,743,042	14,610,229	41,620	204,621	1,784,662	14,814,850	125,436	625,282

Quality measures

- Intersection over union (IoU) = Jaccard similarity



- Precision = TP/\hat{P} , Recall = TP/P

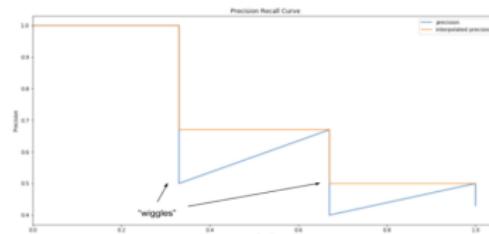


Precision-recall curve

- Fix confidence threshold and minimal IoU threshold.
- Precision-recall curve = Precision(Recall)
- To make it smoother, *interpolated precision* is used:

$$Pr_{interp}(Rec) = \max_{\tilde{Rec} \leq Rec} Pr(\tilde{Rec})$$

	TP/FP	Precision	Recall	Precision_inter
1				
2	TP	1/1 = 1	1/3 = 0.33	1
3	FP	1/2 = 0.5	1/3 = 0.33	1
4	TP	2/3 = 0.67	2/3 = 0.67	0.67
5	FP	2/4 = 0.5	2/3 = 0.67	0.67
6	FP	2/5 = 0.4	2/3 = 0.67	0.67
7	TP	3/6 = 0.5	3/3 = 1	0.5
8	FP	3/7 = 0.43	3/3 = 1	0.5



Average precision, mean average precision

- Average precision (AP) - area under the curve:

$$AP = \frac{1}{K+1} \sum_{Rec \in \{0, \frac{1}{K}, \frac{2}{K}, \dots, \frac{K}{K}\}} Pr_{interp}(Rec)$$

- Mean average precision - averaged AP over object classes:

$$mAP = \frac{1}{C} \sum_{c=1}^C AP(c)$$

Naive approach



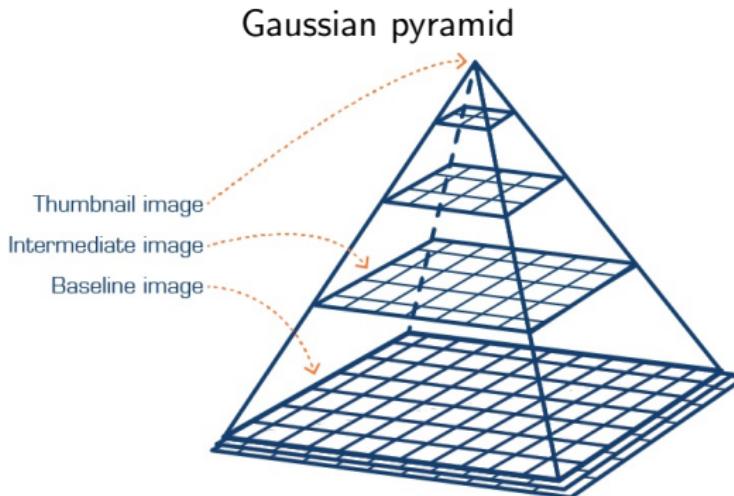
Naive approach: extract patches at all locations at different sizes and apply classifier to each patch.

- inefficient, but represents the idea of later approaches.

Extracting bounding boxes of different size

To extract bounding boxes of different size:

- extract bboxes of the same size applied to rescaled versions of image.



Need also to check bboxes of different shape.

How to solve object detection?

- Selective search algorithm¹ generates region proposals.

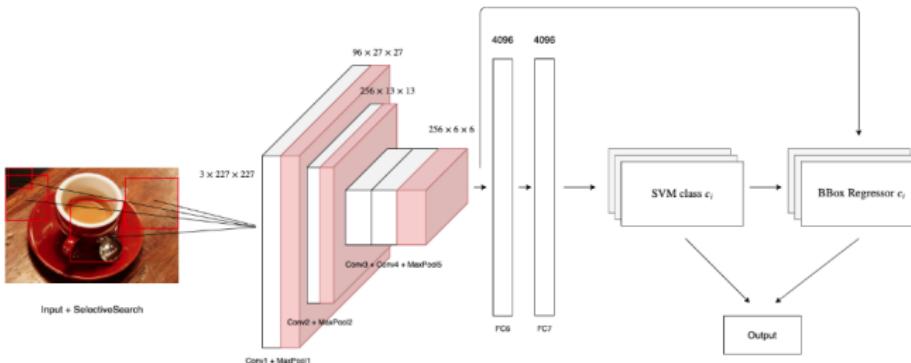


- Idea: cluster RGB+XY information into clusters of similar color and spatial position. Join neighboring segments similar in color. Draw a boundary around each segment.

¹<http://www.huppelen.nl/publications/selectiveSearchDraft.pdf>

R-CNN²

R-CNN scheme:



²Girshik et al. Rich feature hierarchies for accurate object detection and semantic segmentation.

R-CNN algorithm

- ➊ SelectiveSearch algorithm generates region proposals ~ 2000
- ➋ Region proposals rescaled to 224×224 (to match AlexNet)
- ➌ AlexNet convolutional layers (except FC layers) extract 4096 features for each region.
- ➍ SVM classifier is trained on $C + 1$ class (+1 for background)
- ➎ Regression is trained to correct coordinates of region proposal:

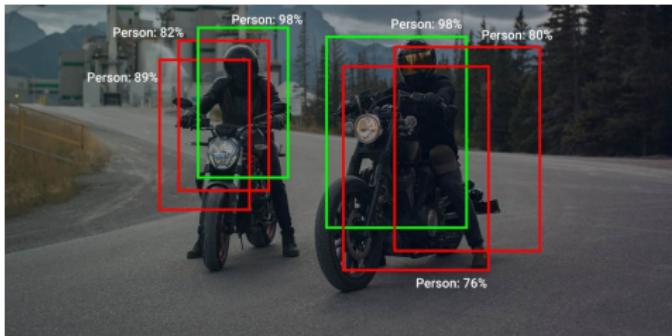
$(\hat{x}, \hat{y}, \hat{h}, \hat{w})$ -predicted bbox, (x, y, h, w) -true bbox

regr. predicts: $\left(\frac{x - \hat{x}}{w}, \frac{y - \hat{y}}{h}, \ln \left(\frac{\hat{w}}{w} \right), \ln \left(\frac{\hat{h}}{h} \right) \right)$

trained on bboxes with $\text{IoU} > 0.3$ with true bbox

- ➏ Eliminate redundant bboxes (non-maximum suppression)

Non-maximum suppression

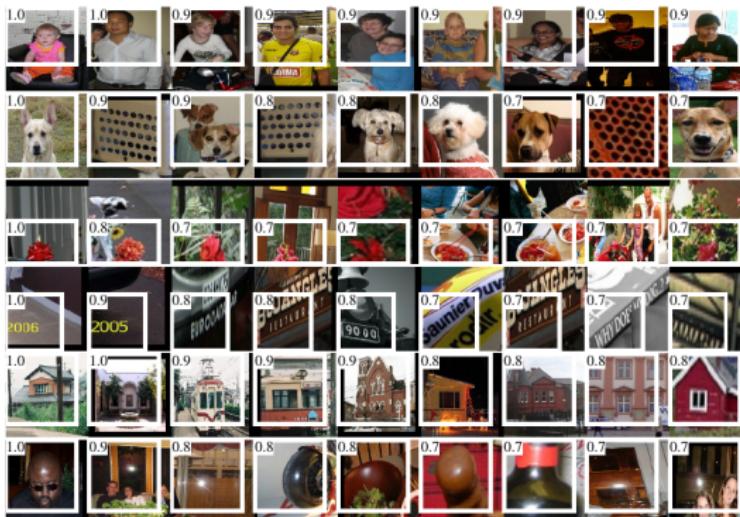


Non-maximum suppression:

- drop low confidence regions
- order bboxes by decreasing confidence
- starting from max confident region downwards:
 - if region has high IoU with other bbox of smaller confidence, drop the latter

Neurons interpretation

Visualization of patches, activating certain neurons inside R-CNN the most.



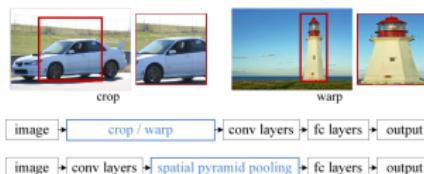
Drawbacks of R-CNN

Drawbacks of R-CNN:

- need separate training for
 - CNN (finetuning)
 - classification (SVM)
 - regression
- for each region proposal need to run CNN
 - and many regions overlap

SPP-net

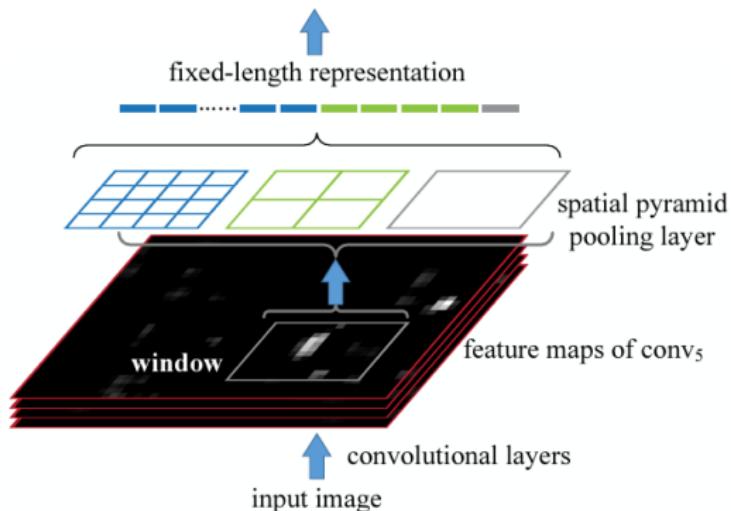
- Problems of R-CNN:
 - CNN reapplied to each region proposal
 - region proposal requires rescaling to 224x224 (image deformation & information loss due to cropping)



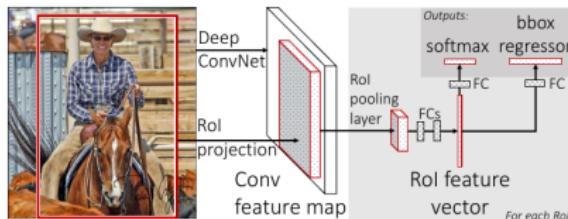
Ideas of SPP-net (spatial pyramid pooling net):

- apply CNN one to whole image (instead of applying for each region proposal)
 - speedup 10x - 100x times.
- spatial pooling allows to output fixed size image representation for image of arbitrary size.

Spatial pyramid pooling

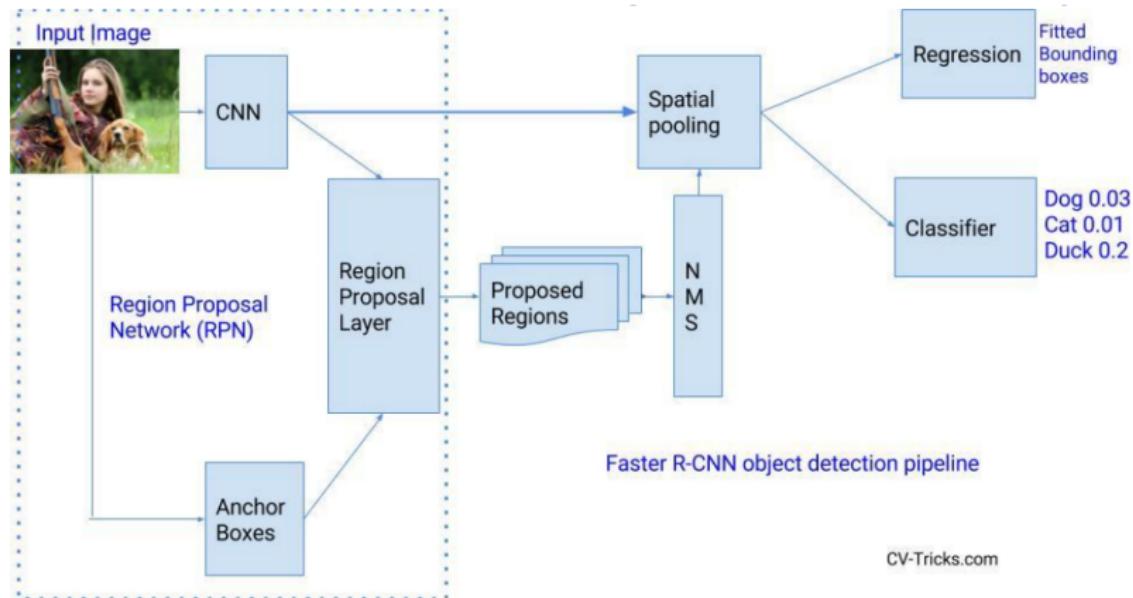


Fast R-CNN



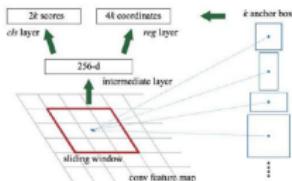
- Regions extracted using SelectiveSearch.
- Image passed through CNN once as a whole.
- Region proposals rescaled to CNN final feature map.
- Pyramid pooling with single layer (grid 7x7) used to match tensor->fixed size representation ("ROI pooling").
- Classifier and bbox regression implemented using additional layers of NN.
 - end2end loss=classification loss+bbox regression loss

Faster R-CNN



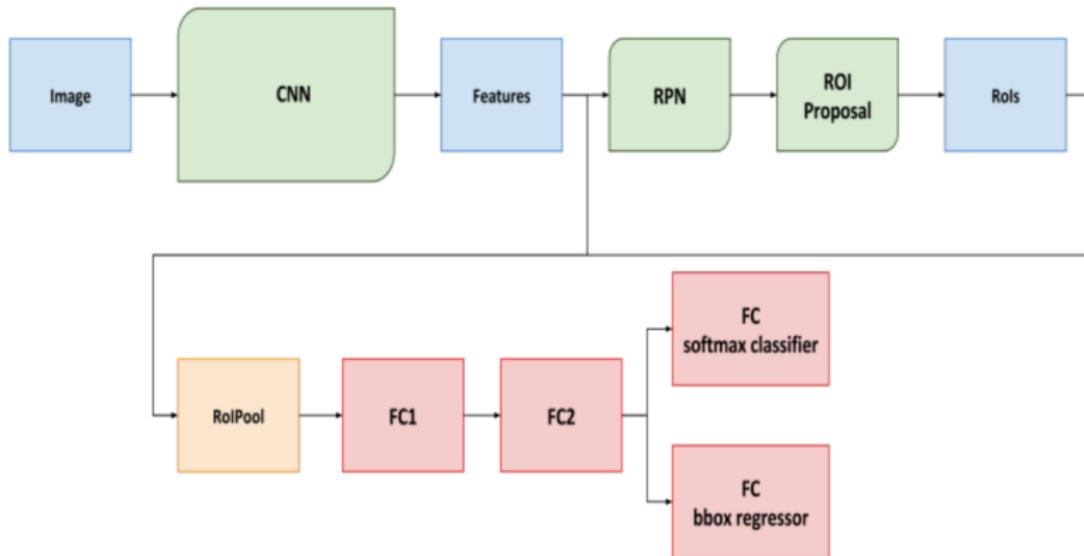
- Fully end2end: region proposals generated by region proposal network RPN inside NN.

Faster R-CNN



- Region proposal network trained inside the whole NN.
- traverse with rolling 3×3 window over the feature map - generate k candidates as possible regions.
- k anchor boxes - some templates of size 128, 256, 512 and aspect ratio 1:1, 1:2, 2:1
- Regions with high confidence are passed further.
- k regions =>
 - $2k$ scores (probability of presence&absence of object)
 - $4k$ coordinates: refined coordinates

Faster R-CNN



Faster R-CNN

- **Learning:** non-trivial of RPN and another part, producing the class and coordinates.
 - may be done iteratively
- **Inference:**
 - image->CNN+RPN
 - leave 6000 perspective regions
 - delete thin and out-of-image regions
 - NMS with high IoU=0.9
 - regions->RoI pooling+regression+classification
 - NMS on top

Speed comparison

