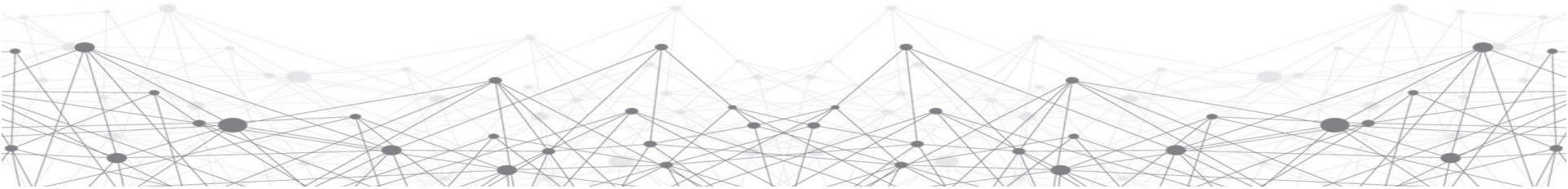
Detecting objects with deep learning hammer

Tao Kong

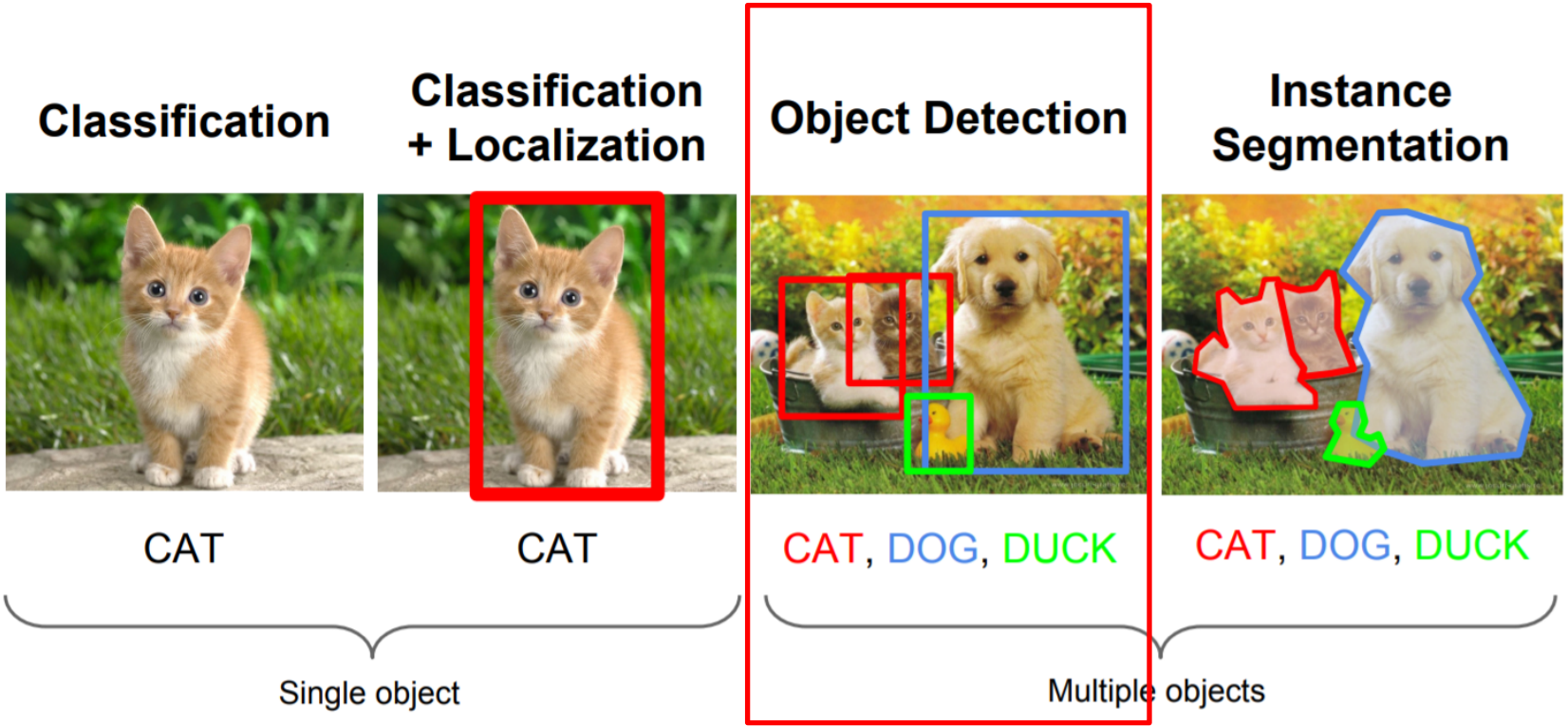
http://www.taokong.org/



Outline

1. The object detection problem
2. Traditional methods (before 2014)
3. Modern object detection frameworks
4. Other useful resources

Basic computer vision tasks



Detection problem

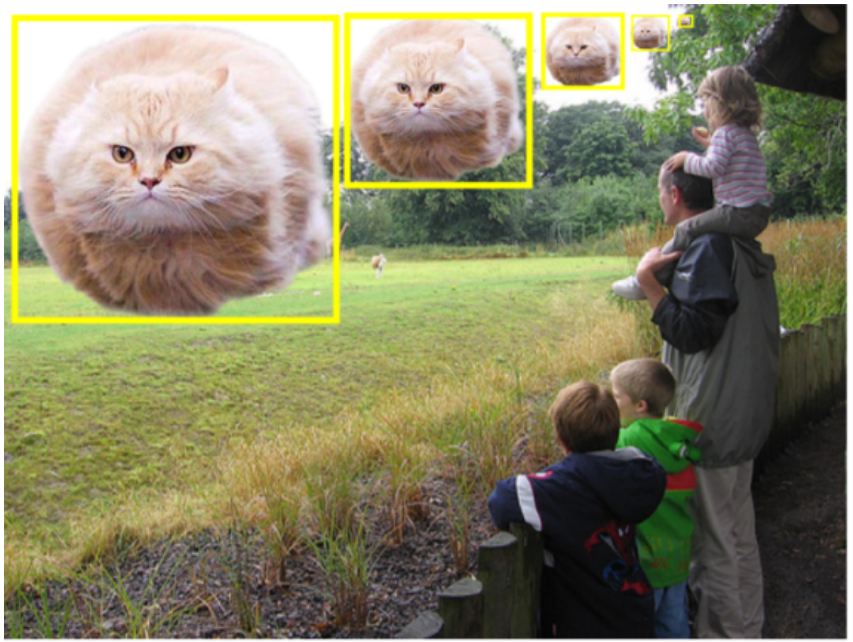
1. Classifier must generalize over all exemplars of one class.

2. Negative class consists of everything else.

3. High accuracy (small FP rate) required for most applications.

Detection challenges

1. Object scales?



2. Negative spaces?

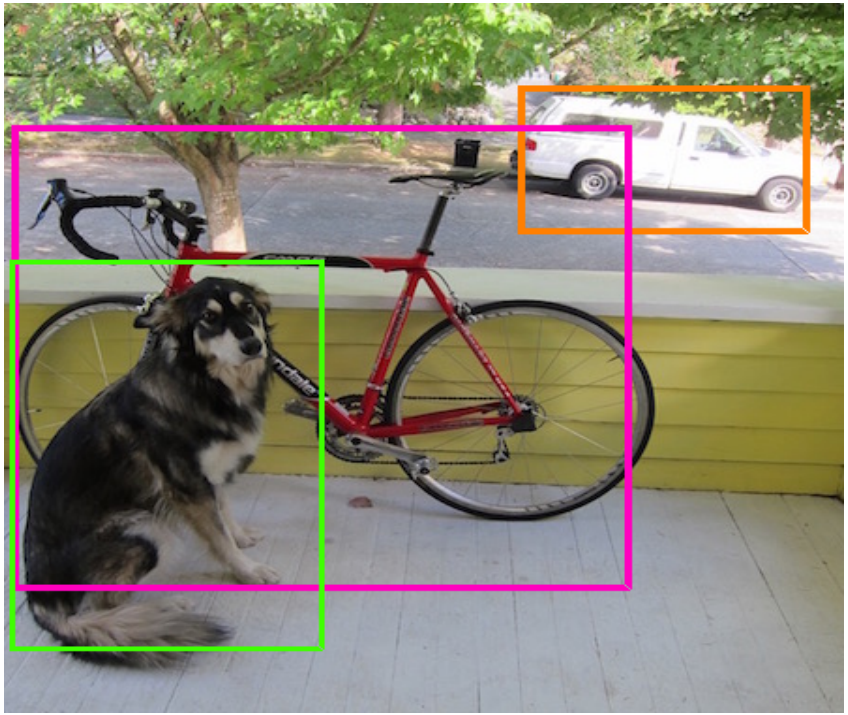
3. Object variations?

Detection challenges

1. Object scales?

2. Negative spaces?

3. Object variations?



Detection challenges

1. Object scales?

2. Negative spaces?

3. Object variations?

Detection challenges

1. Object scales?

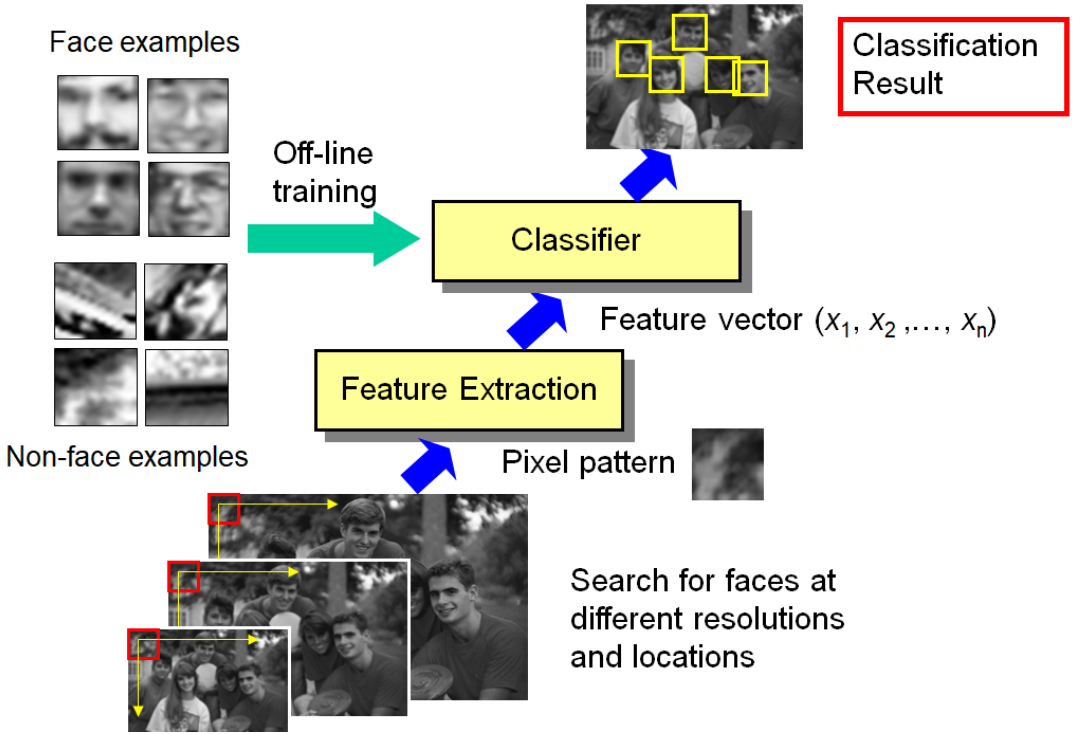
2. Negative spaces?

3. Object variations?

Face Detection – basic scheme

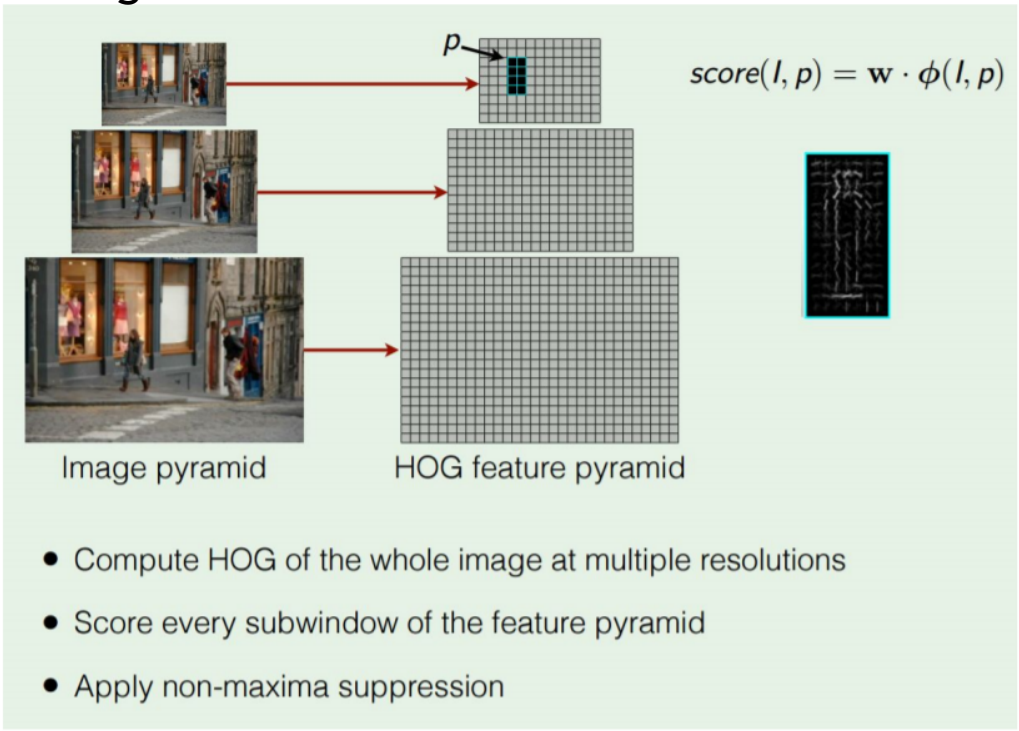
- Sliding windows
- Off-line training

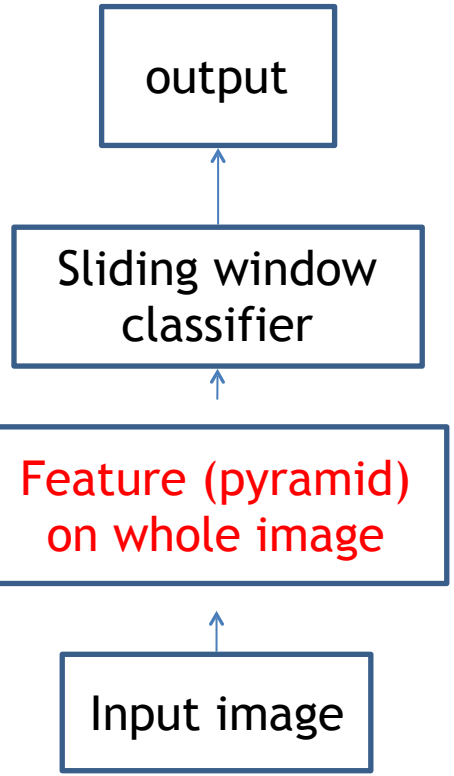
✓ Better Features
✓ Better Classifiers
✓ Better sliding windows



Face Detection – basic scheme

Histogram of Oriented Gradients

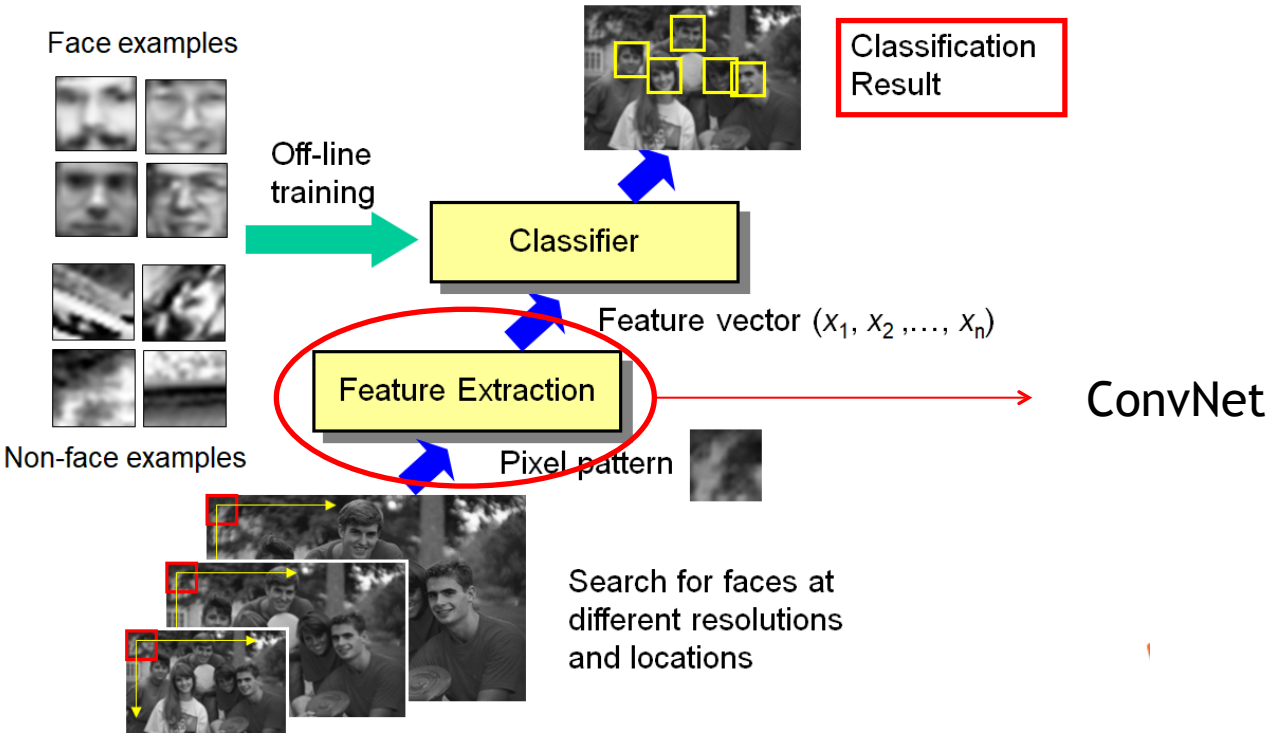


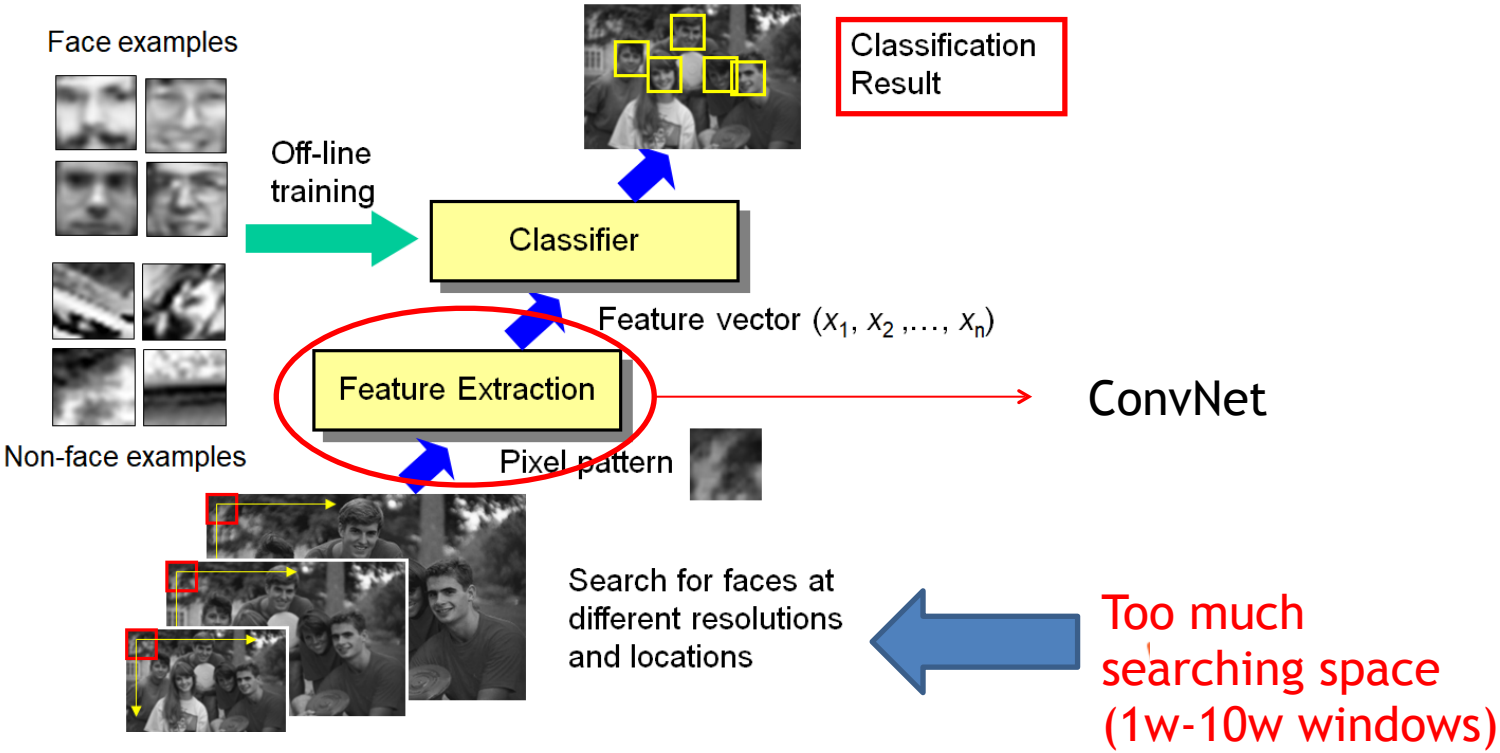


Object detection: benchmarks

	PASCAL VOC (2010)	ImageNet Detection (ILSVRC 2014)	MS-COCO (2014)
Number of classes	20	**200**	80
Number of images (train + val)	~20k	**~470k**	~120k
Mean objects per image	2.4	1.1	**7.2**

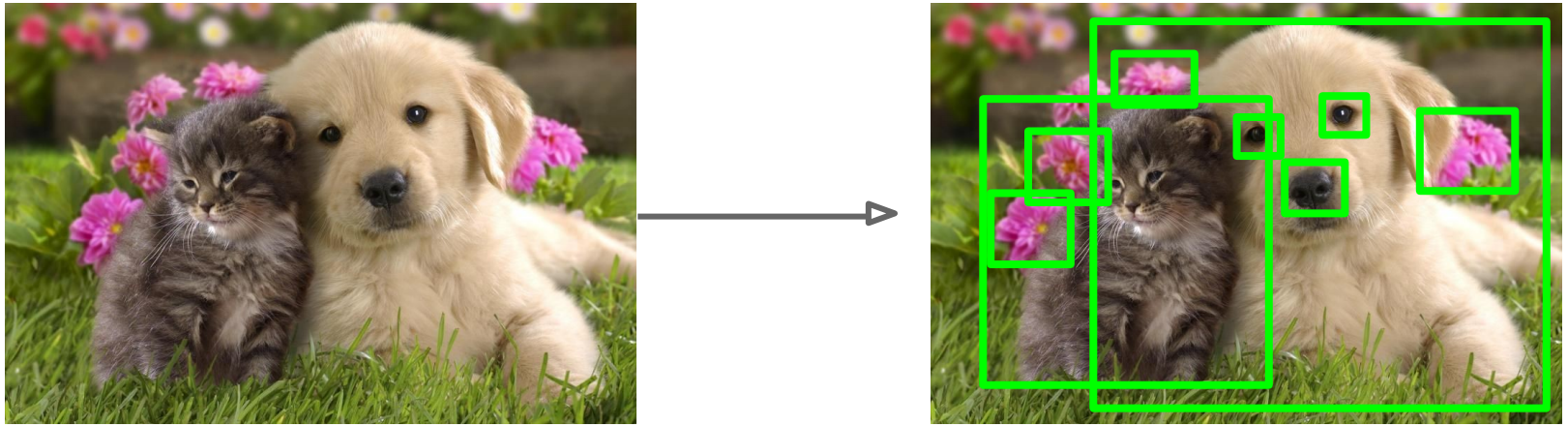
How to do object detection with deep leaning(ConvNet)?


Face examples
Non-face examples
Off-line training
Classifier
Classification Result
Feature vector $(x_1, x_2, \ldots, x_n)$
Feature Extraction
ConvNet
Pixel pattern
Search for faces at different resolutions and locations


Face examples
Non-face examples
Off-line training
Classifier
Classification Result
Feature vector (x_1, x_2 ,…, x_n)
Feature Extraction
ConvNet
Pixel pattern
Search for faces at different resolutions and locations
Too much searching space (1w-10w windows)

Region proposals

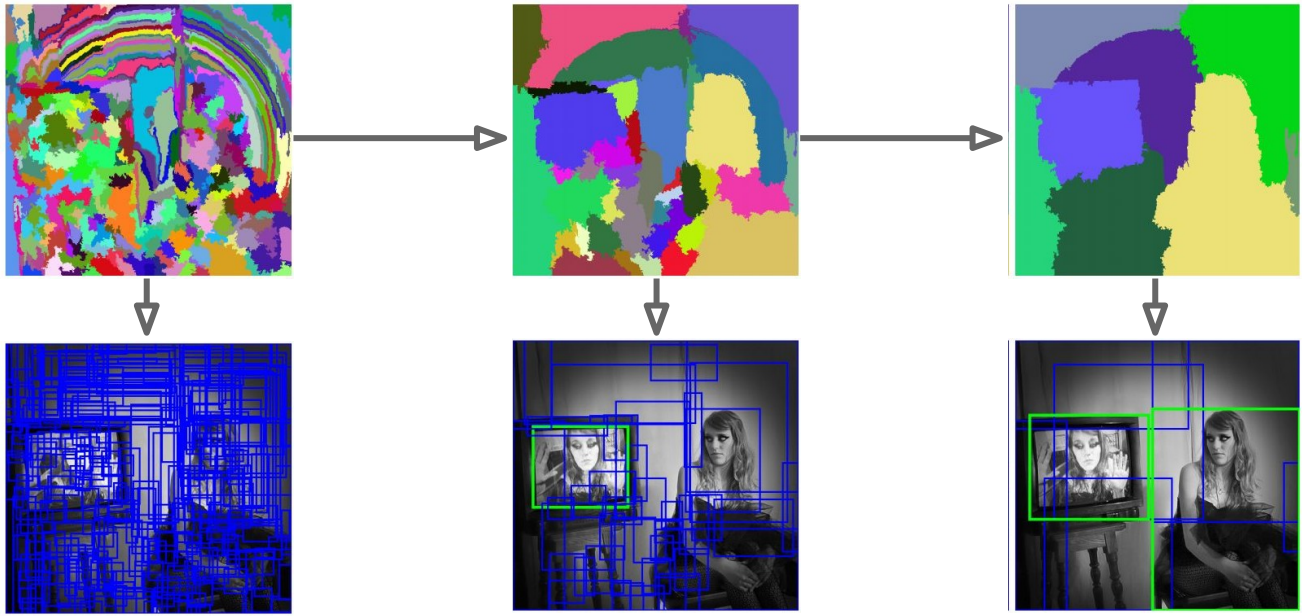
- Find image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



Region proposals: selective search

Bottom-up segmentation, merging regions at multiple scales

Convert regions to boxes



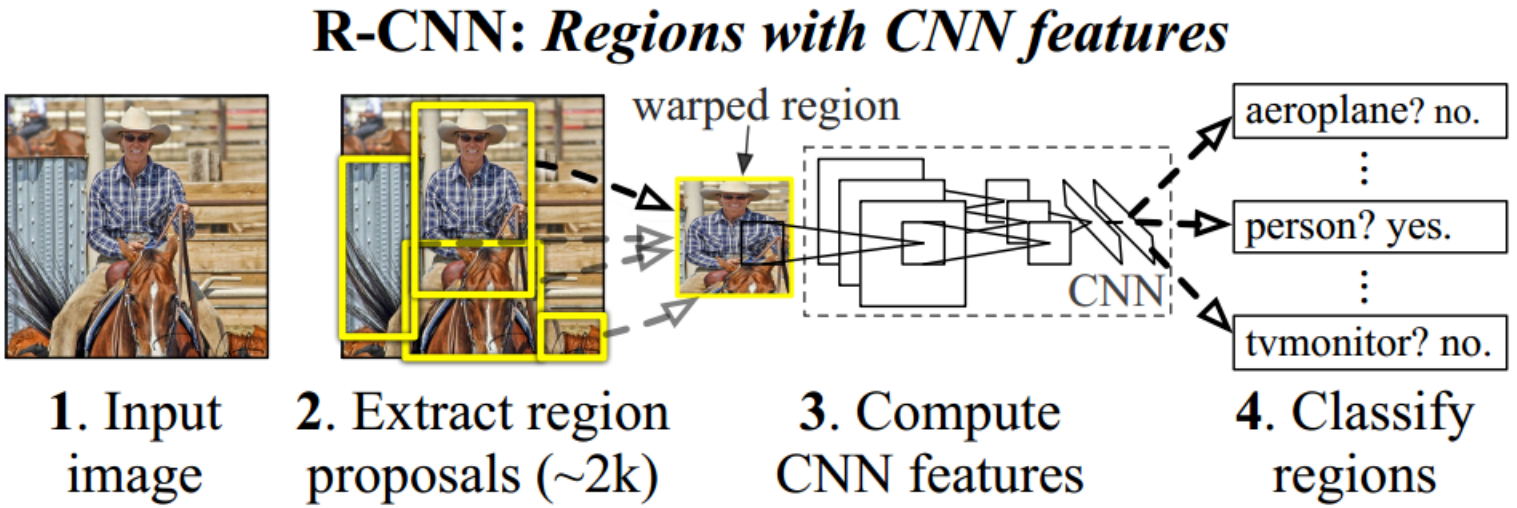
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013

Region proposals: other choices

Method	Approach	Outputs Segments	Outputs Score	Control #proposals	Time (sec.)	Repea-tability	Recall Results	Detection Results
Bing [18]	Window scoring		✓	✓	0.2	⋆⋆⋆	⋆	·
CPMC [19]	Grouping	✓	✓	✓	250	-	⋆⋆	⋆
EdgeBoxes [20]	Window scoring		✓	✓	0.3	⋆⋆	⋆⋆⋆	⋆⋆⋆
Endres [21]	Grouping	✓	✓	✓	100	-	⋆⋆⋆	⋆⋆
Geodesic [22]	Grouping	✓		✓	1	⋆	⋆⋆⋆	⋆⋆
MCG [23]	Grouping	✓	✓	✓	30	⋆	⋆⋆⋆	⋆⋆⋆
Objectness [24]	Window scoring		✓	✓	3	·	⋆	·
Rahtu [25]	Window scoring		✓	✓	3	·	·	⋆
RandomizedPrim's [26]	Grouping	✓		✓	1	⋆	⋆	⋆⋆
Rantalankila [27]	Grouping	✓		✓	10	⋆⋆	·	⋆⋆
Rigor [28]	Grouping	✓		✓	10	⋆	⋆⋆	⋆⋆
SelectiveSearch [29]	Grouping	✓	✓	✓	10	⋆⋆	⋆⋆⋆	⋆⋆⋆
Gaussian				✓	0	·	·	⋆
SlidingWindow				✓	0	⋆⋆⋆	·	·
Superpixels		✓			1	⋆	·	·
Uniform				✓	0	·	·	·

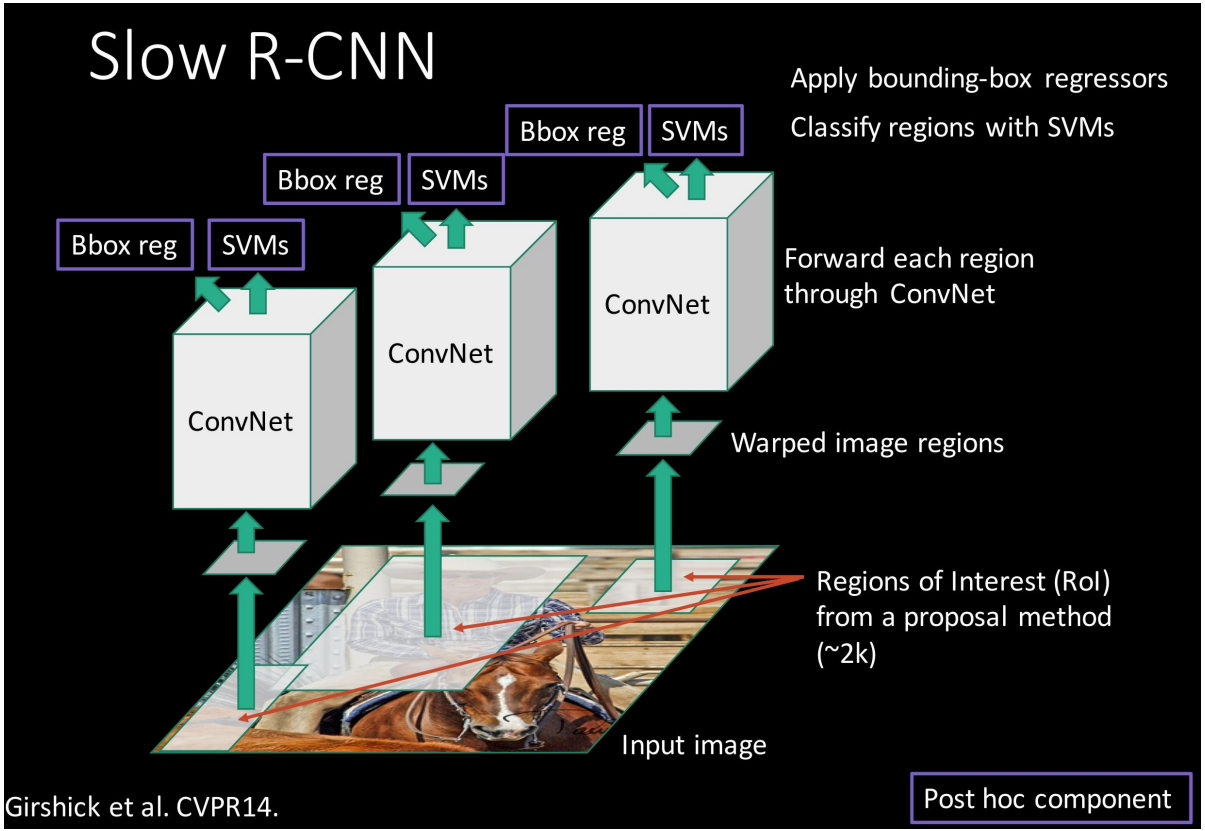
Hosang et al, “What makes for effective detection proposals?”, PAMI 2015

R-CNN:



R. Girshick, J. Donahue, T. Darrell, J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR 2014*. (oral)

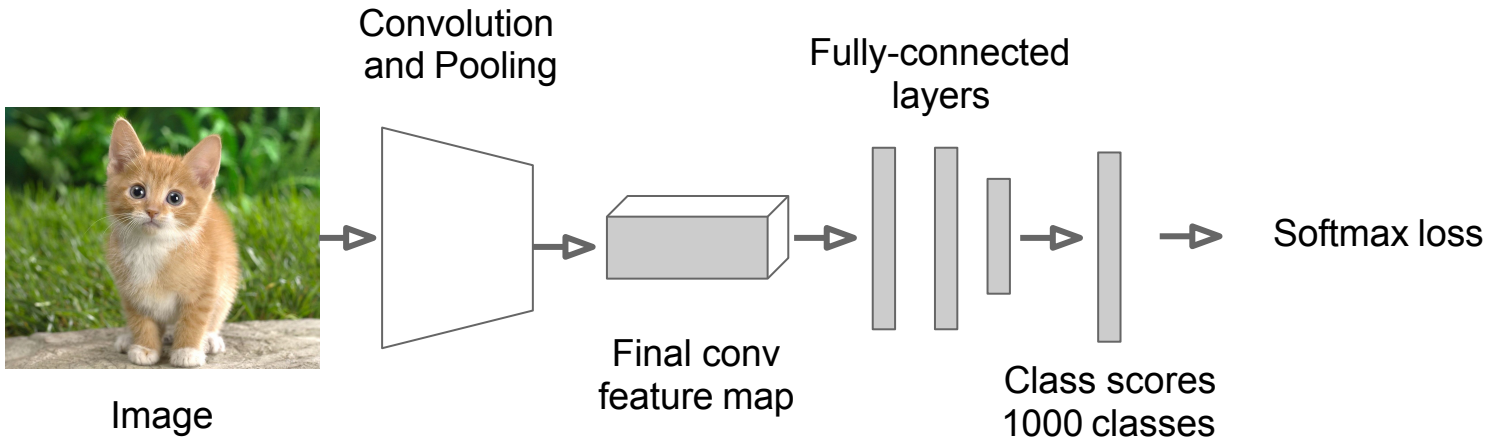
R-CNN: Another view



Girshick et al. CVPR14.

R-CNN: Training

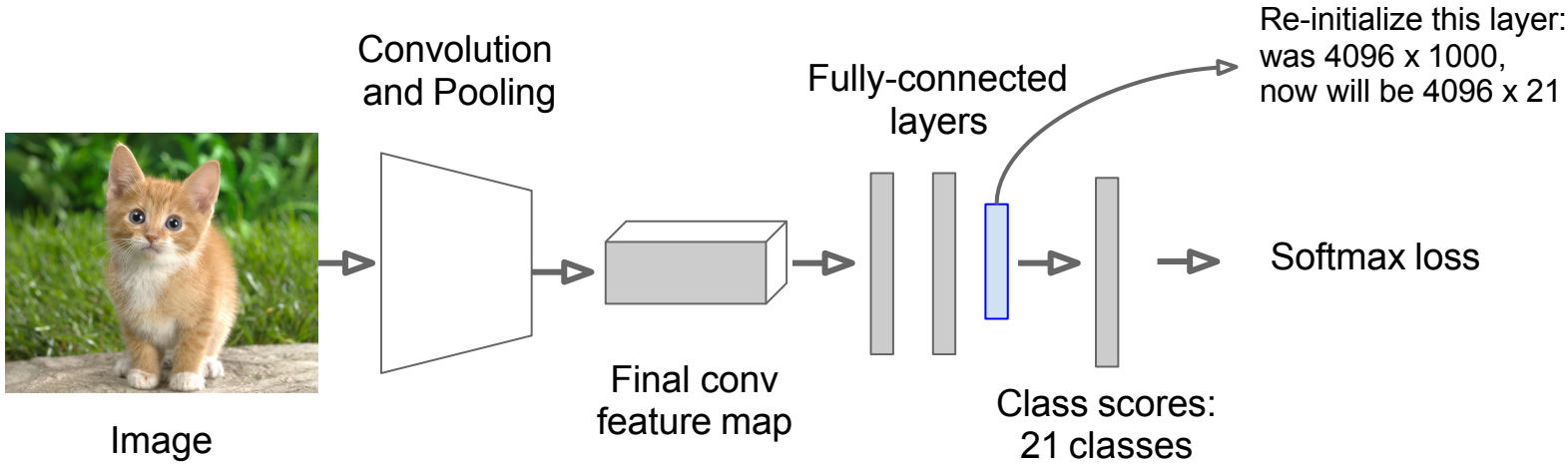
Step 1: Train (or download) a classification model for ImageNet (AlexNet)



R-CNN: Training

Step 2: Fine-tune model for detection

- Instead of 1000 ImageNet classes, want 20 object classes + background
- Throw away final fully-connected layer, reinitialize from scratch
- Keep training model using positive / negative regions from detection images

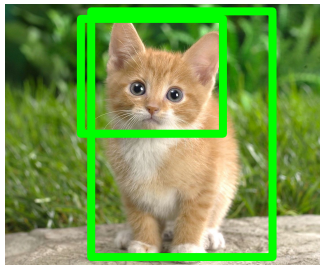


R-CNN: Training

Step 3: Extract features

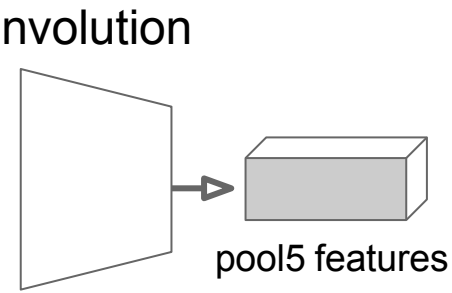
- Extract region proposals for all images
- For each region: warp to CNN input size, run forward through CNN, save pool5 features to disk
- Have a big hard drive: features are ~200GB for PASCAL dataset!

Image

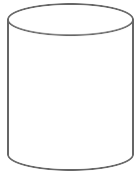


Region Proposals

Crop + Warp



Forward pass



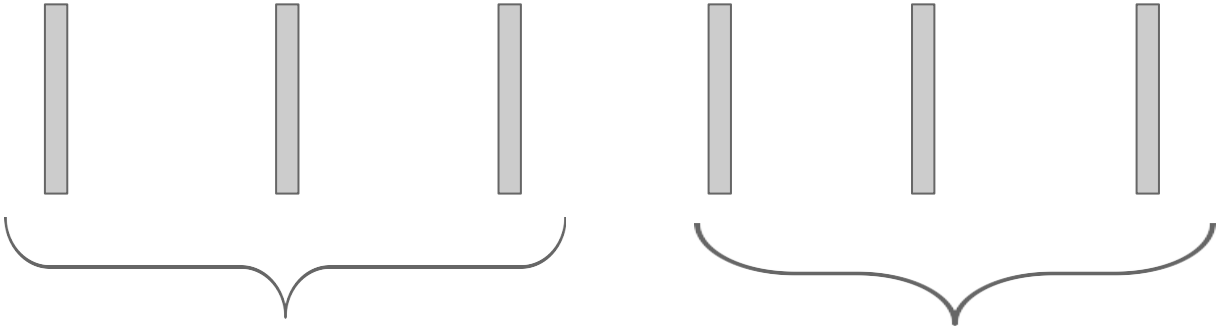
Save to disk

R-CNN: Training

Step 4: Train one binary SVM per class to classify region features

Training image regions

Cached region features



Positive samples for cat SVM

Negative samples for cat SVM

R-CNN: Training

Step 5 (bbox regression): For each class, train a linear regression model to map from cached features to offsets to GT boxes to make up for “slightly wrong” proposals

Training image regions

Cached region features

Regression targets
(dx, dy, dw, dh)
Normalized coordinates

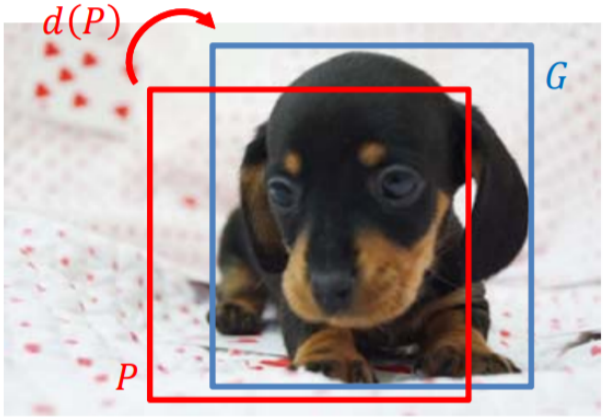
(0, 0, 0, 0)
Proposal is good

(.25, 0, 0, 0)
Proposal too far to left

(0, 0, -0.125, 0)
Proposal too wide

R-CNN: Bounding Box Regression

- Learning a transformation of bounding box
  - Region proposal: $P = (P_w, P_h, P_w, P_h)$
  - Ground-truth: $G = (G_x, G_y, G_w, G_h)$
  - Transformation: $d(P) = (t_x, t_y, t_w, t_h)$



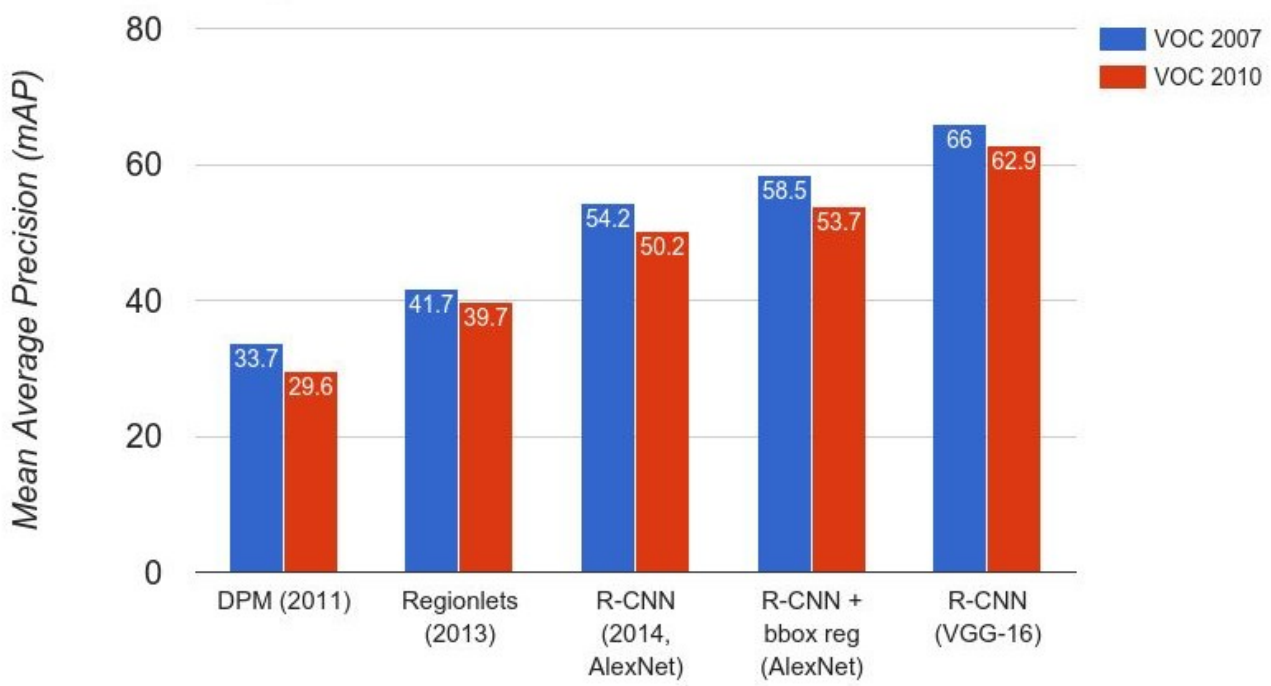
$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp\big(d_w(P)\big)$$
$$\hat{G}_h = P_h \exp\big(d_h(P)\big)$$

$$d_i(P) = \boldsymbol{w}_i^{\mathrm{T}} \phi_5(P)$$

CNN pool5 feature

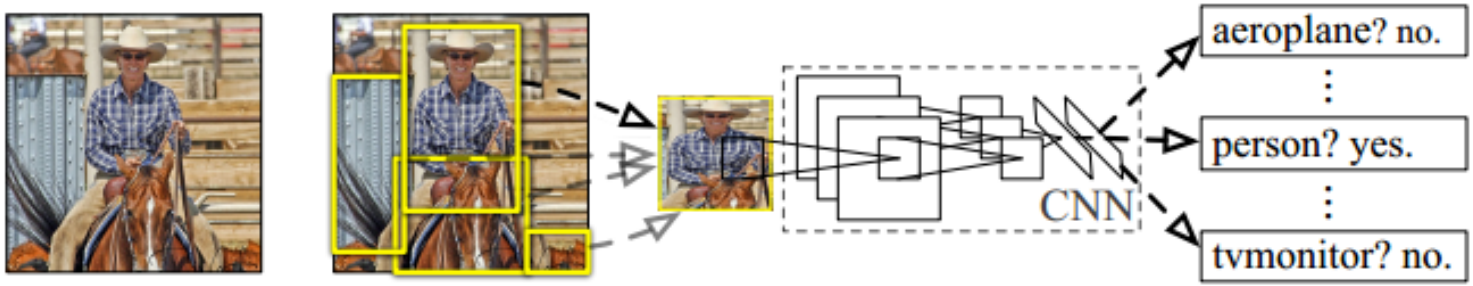
$$\boldsymbol{w}_i^* = \underset{\boldsymbol{w}_i}{\operatorname{argmin}} \sum_{k=1}^{N} \left(t_i^k - \boldsymbol{w}_i^{\mathrm{T}} \phi_5\big(P^k\big)\right)^2 + \lambda \|\boldsymbol{w}_i\|^2$$

R-CNN: Results

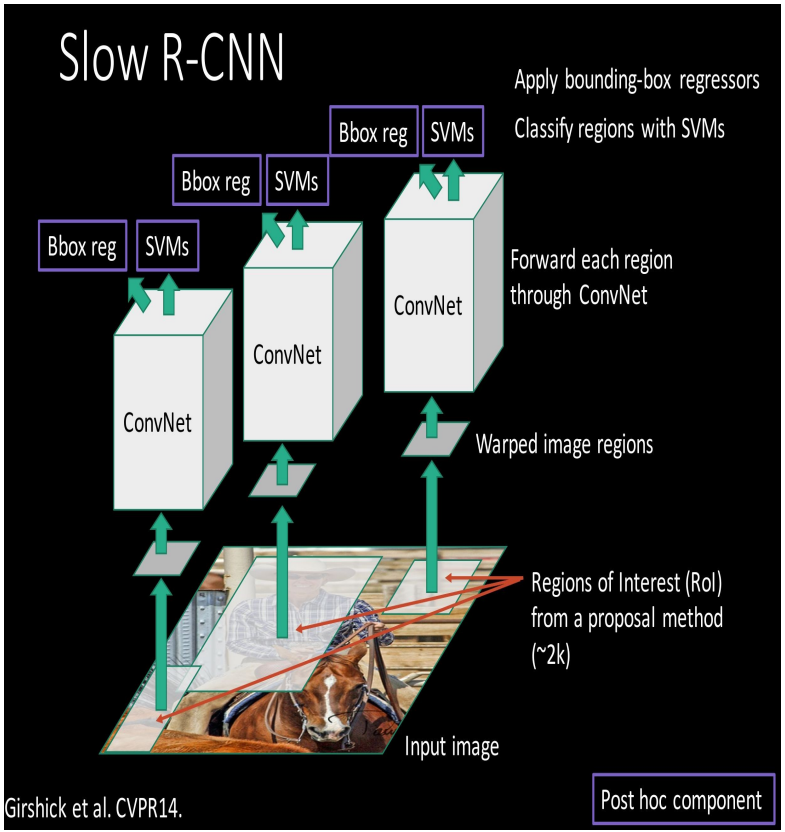


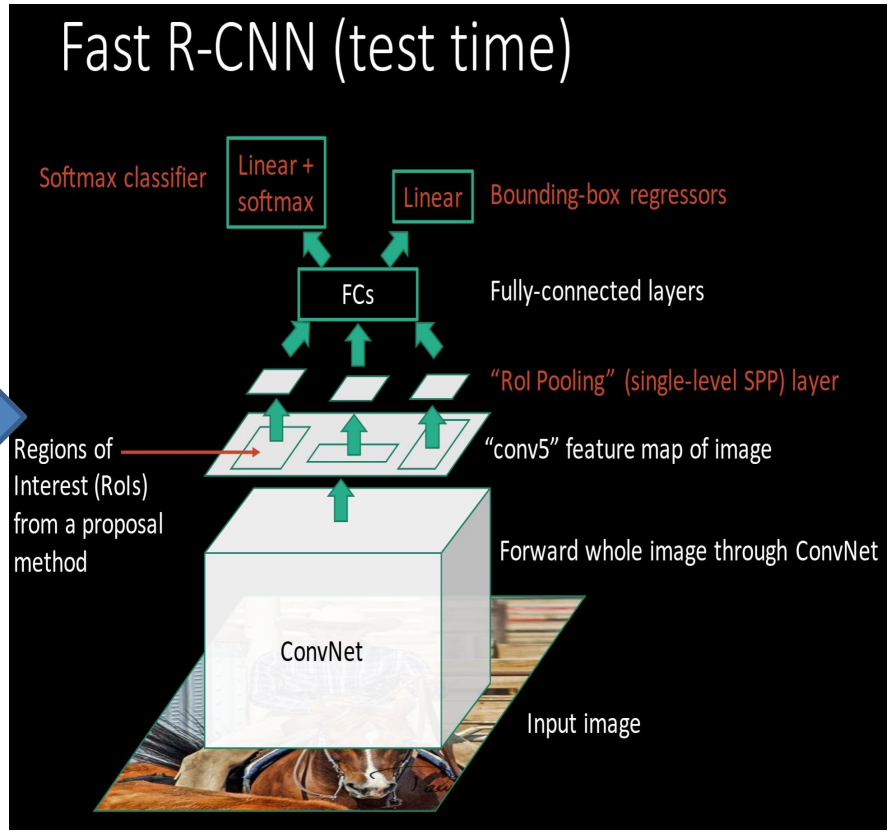
R-CNN: Disadvantages

- Training and testing is expensive in space and time.
  - Why: for each image, one need to extract CNN features for about 2k object proposals and make classification. But the features are shared in space.
- Relays on the region proposal method(selective search)
- Training is a multi-stage pipeline

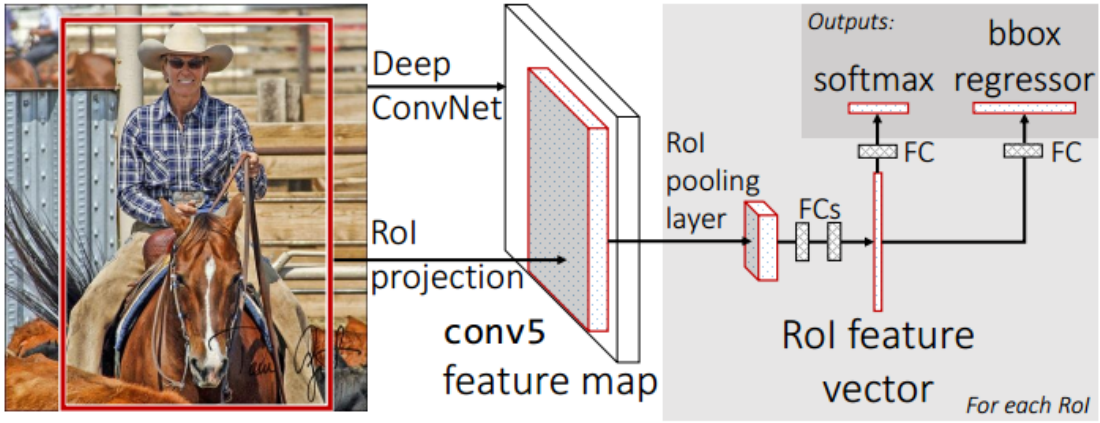


Fast R-CNN

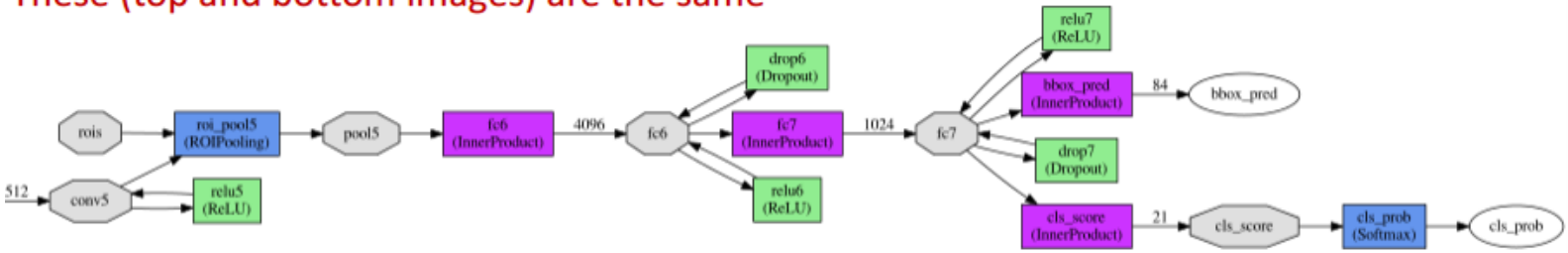

Slow R-CNN
Apply bounding-box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image
Girshick et al. CVPR14.
Post hoc component


Fast R-CNN (test time)
Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" (single-level SPP) layer
"conv5" feature map of image
Regions of Interest (RoIs) from a proposal method
Forward whole image through ConvNet
ConvNet
Input image

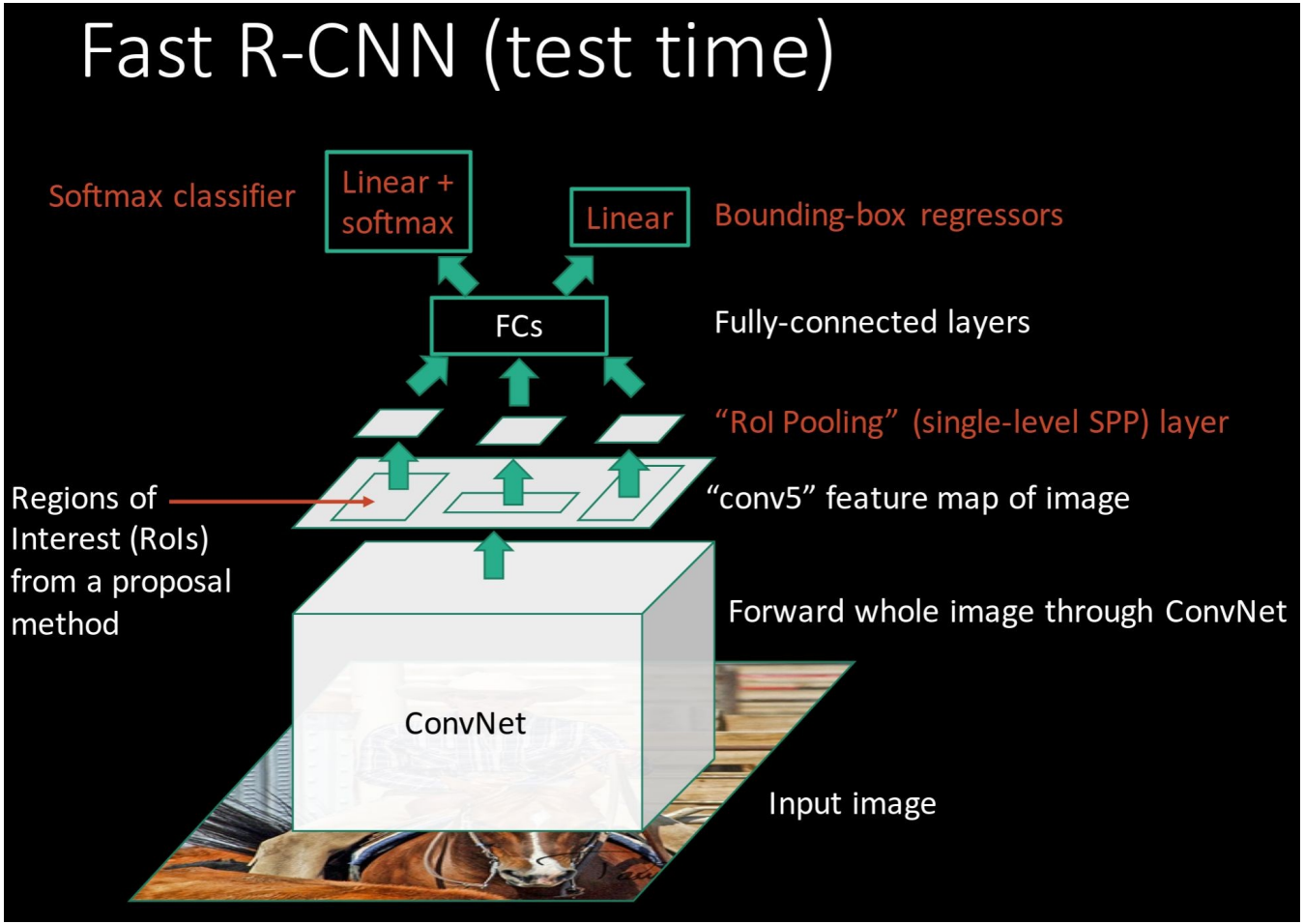
Another view of the same thing



These (top and bottom images) are the same

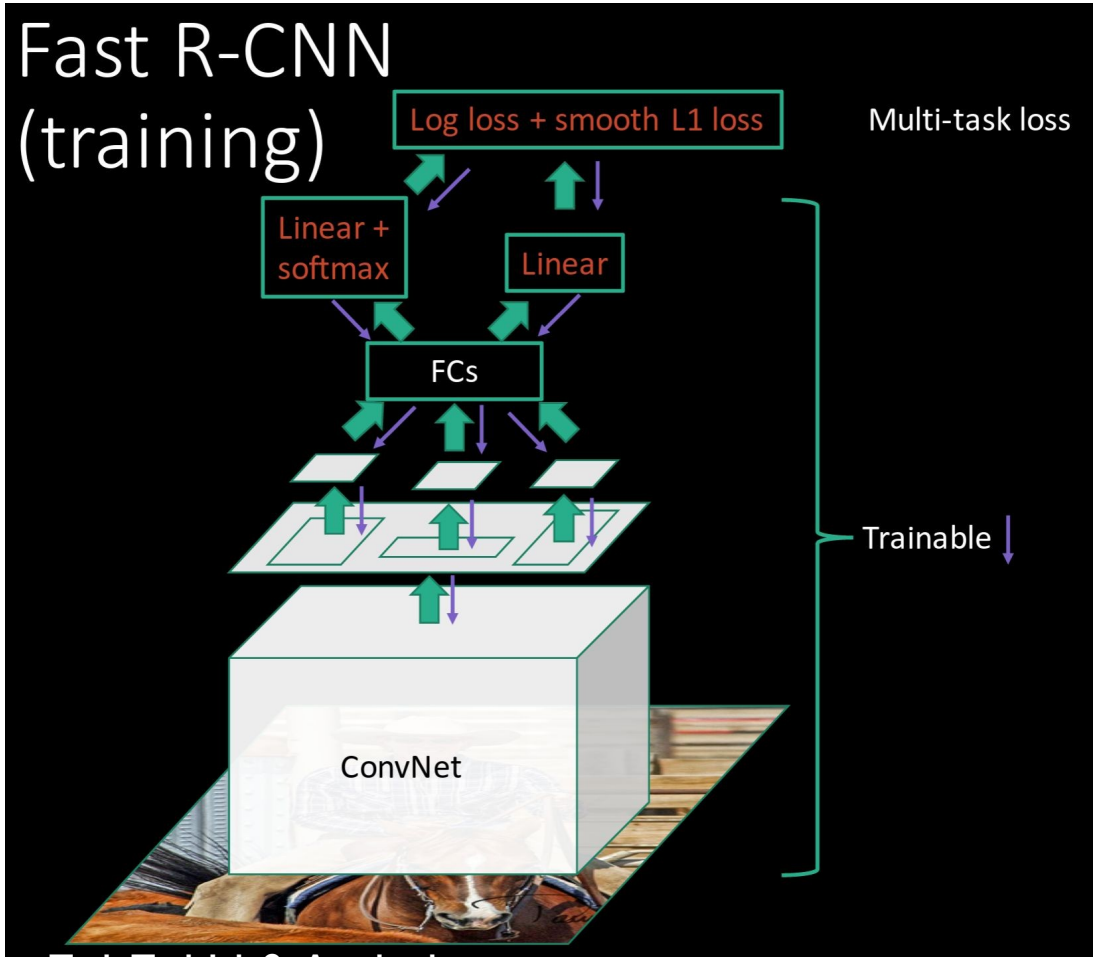


Fast R-CNN (test time)



R-CNN Problem #1:
Slow at test-time due to independent forward passes of the CNN

Solution:
Share computation of convolutional layers between proposals for an image



R-CNN Problem #2:
Post-hoc training: CNN not updated in response to final classifiers and regressors

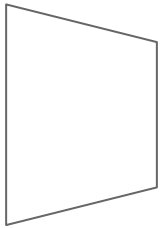
R-CNN Problem #3:
Complex training pipeline

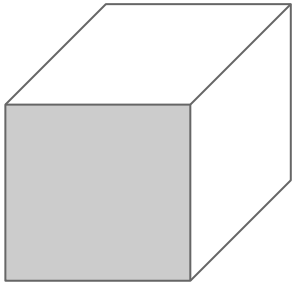
Solution:
Just train the whole system end-to-end all at once!

Slide credit: Ross Girschick

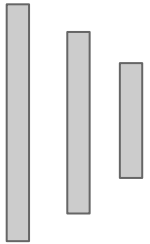
Fast R-CNN: Region of Interest Pooling

Convolution and Pooling





Fully-connected layers



Hi-res input image:
3 x 800 x 600
with region proposal

Hi-res conv features:
C x H x W
with region proposal

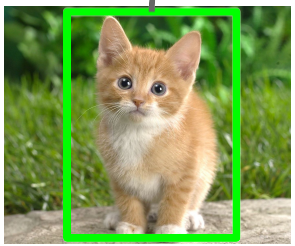
Problem: Fully-connected layers expect low-res conv features: C x h x w

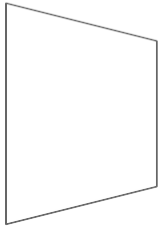
Fast R-CNN: Region of Interest Pooling

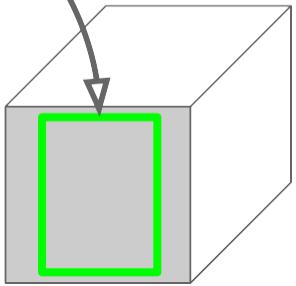
Project region proposal
onto conv feature map

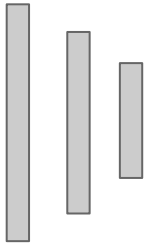
Convolution
and Pooling

Fully-connected
layers









Hi-res input image:
3 x 800 x 600
with region
proposal

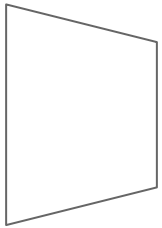
Hi-res conv features:
C x H x W
with region proposal

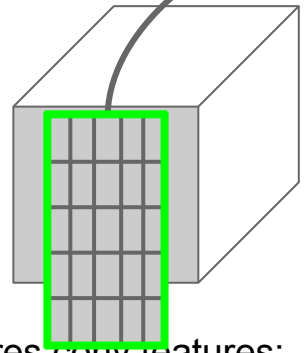
Problem: Fully-connected layers expect low-res conv features: C x h x w

Fast R-CNN: Region of Interest Pooling

Hi-res input image:
3 x 800 x 600
with region
proposal

Convolution
and Pooling

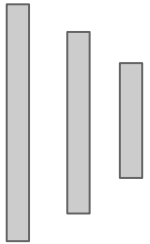




Hi-res conv features:
C x H x W
with region proposal

Divide projected
region into h x w grid

Fully-connected
layers

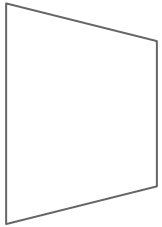


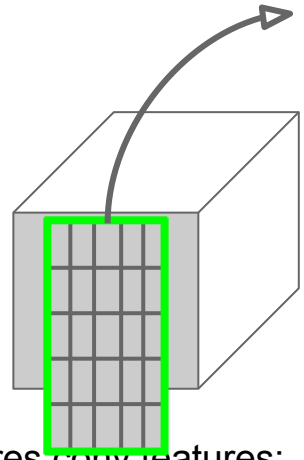
Problem: Fully-connected layers expect low-res conv features: C x h x w

Fast R-CNN: Region of Interest Pooling

Hi-res input image:
3 x 800 x 600
with region
proposal

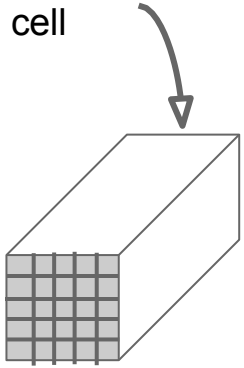
Convolution
and Pooling





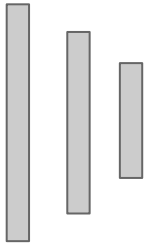
Hi-res conv features:
C x H x W
with region proposal

Max-pool within
each grid cell



RoI conv features:
C x h x w
for region proposal

Fully-connected
layers

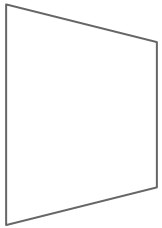


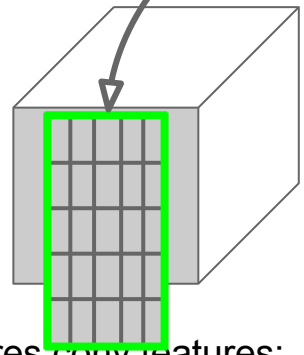
Fully-connected layers expect
low-res conv features:
C x h x w

Fast R-CNN: Region of Interest Pooling

Hi-res input image:
3 x 800 x 600
with region
proposal

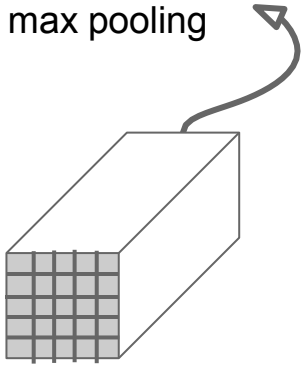
Convolution
and Pooling





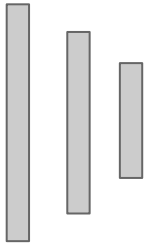
Hi-res conv features:
C x H x W
with region proposal

Can back propagate
similar to max pooling



RoI conv features:
C x h x w
for region proposal

Fully-connected
layers



Fully-connected layers expect
low-res conv features:
C x h x w

Fast R-CNN Results

	R-CNN	Fast R-CNN
Training Time:	84 hours	**9.5 hours**
(Speedup)	1x	**8.8x**
Test time per image	47 seconds	**0.32 seconds**
(Speedup)	1x	**146x**
mAP (VOC 2007)	66.0	**66.9**

Faster!

FASTER!

Better!

Using VGG-16 CNN on Pascal VOC 2007 dataset

Fast R-CNN Problem:

Test-time speeds don't include region proposals

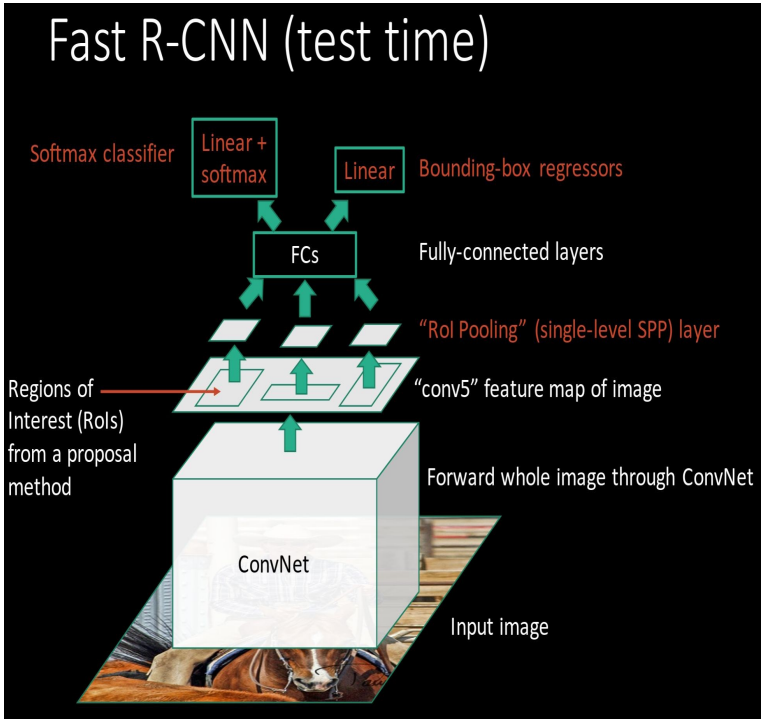
	R-CNN	**Fast R-CNN**
Test time per image	47 seconds	**0.32 seconds**
(Speedup)	1x	**146x**
Test time per image with Selective Search	50 seconds	**2 seconds**
(Speedup)	1x	**25x**

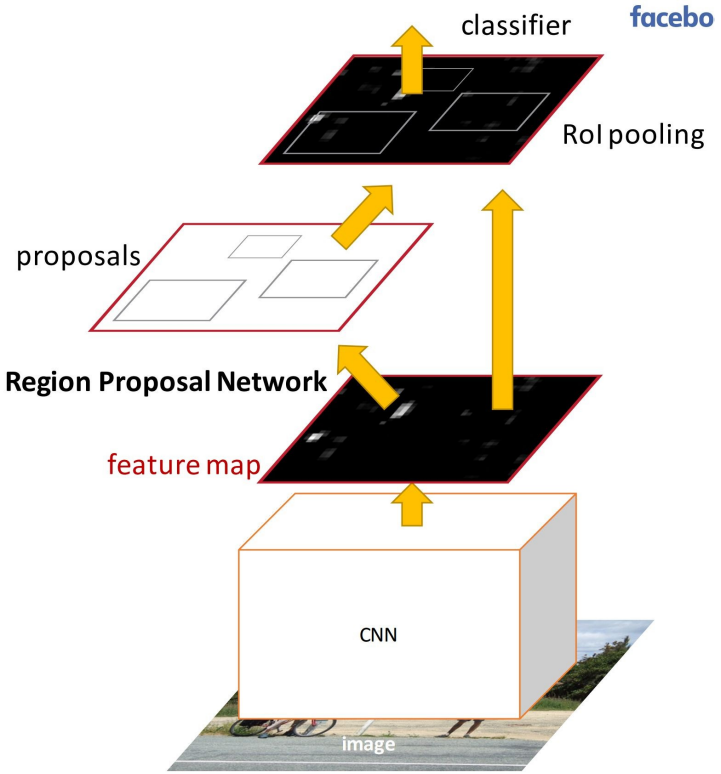
Fast R-CNN ~~Problem~~ Solution:

Test-time speeds don't include region proposals
Just make the CNN do region proposals too!

	R-CNN	**Fast R-CNN**
Test time per image	47 seconds	**0.32 seconds**
(Speedup)	1x	**146x**
Test time per image with Selective Search	50 seconds	**2 seconds**
(Speedup)	1x	**25x**

Faster R-CNN

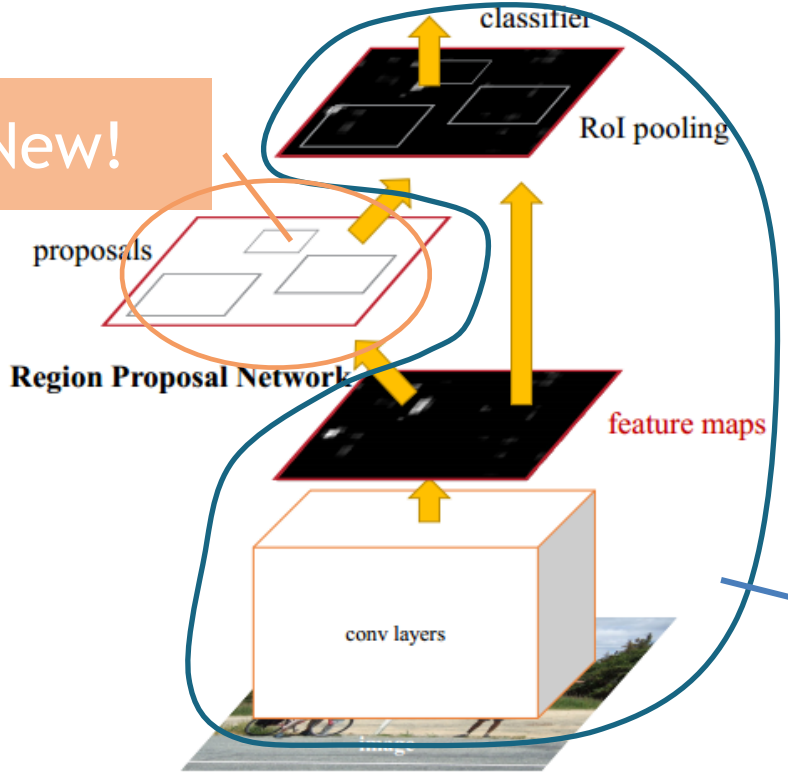

Fast R-CNN (test time)
Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image


classifier
facebo
RoI pooling
proposals
Region Proposal Network
feature map
CNN
image

Faster R-CNN

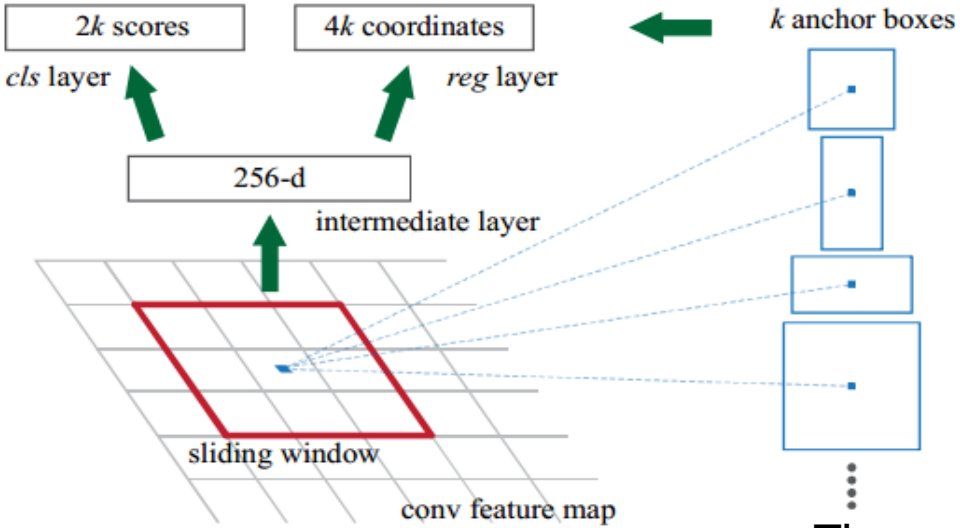
Region Proposal Net + Fast RCNN

This part is New!



There are simple Fast RCNN

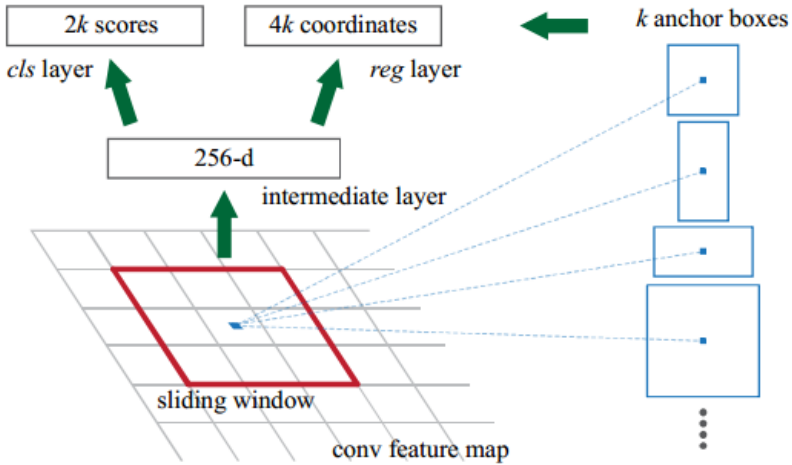
Faster R-CNN: **Region Proposal Network**



- Input an image of any size
- Generate conv feature map
- Map to a lower-dimensional feature
- Output objectness score and bounding box

The region proposal network is a FCN which outputs K*(4+2) sized vectors.

Faster R-CNN: **Region Proposal Network**



LOSS

- Positive: Among K anchors, one with highest IOU (IOU>=0.7)
- Negative:IOU<=0.3
- Others: Do not contribute

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

The mini-batch size(256)

The number of anchor locations

Faster R-CNN: Results

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

Object Detection State-of-the-art: ResNet 101 + Faster R-CNN + some extras

training data	COCO train		COCO trainval	
test data	COCO val		COCO test-dev	
mAP	@.5	@[.5, .95]	@.5	@[.5, .95]
baseline Faster R-CNN (VGG-16)	41.5	21.2		
baseline Faster R-CNN (ResNet-101)	48.4	27.2		
+box refinement	49.9	29.9		
+context	51.1	30.0	53.3	32.2
+multi-scale testing	53.8	32.5	**55.7**	**34.9**
ensemble			**59.0**	**37.4**

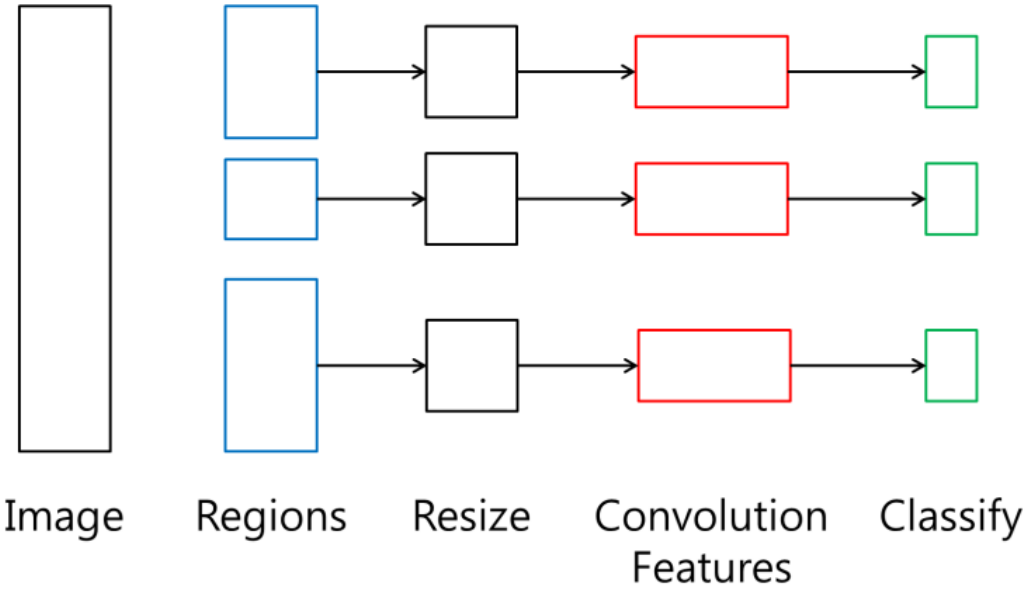
He et. al, “Deep Residual Learning for Image Recognition”, arXiv 2015

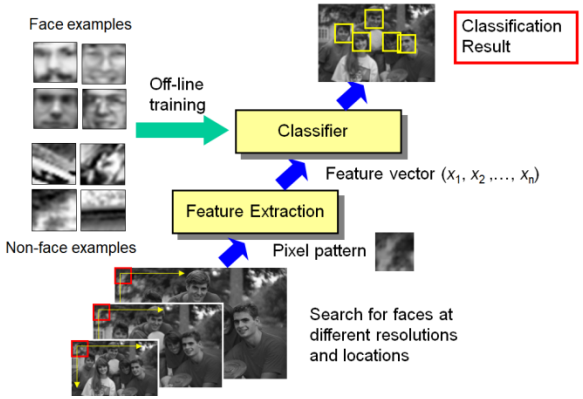
Faster R-CNN: Results

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

R-CNN, Fast R-CNN and Faster R-CNN

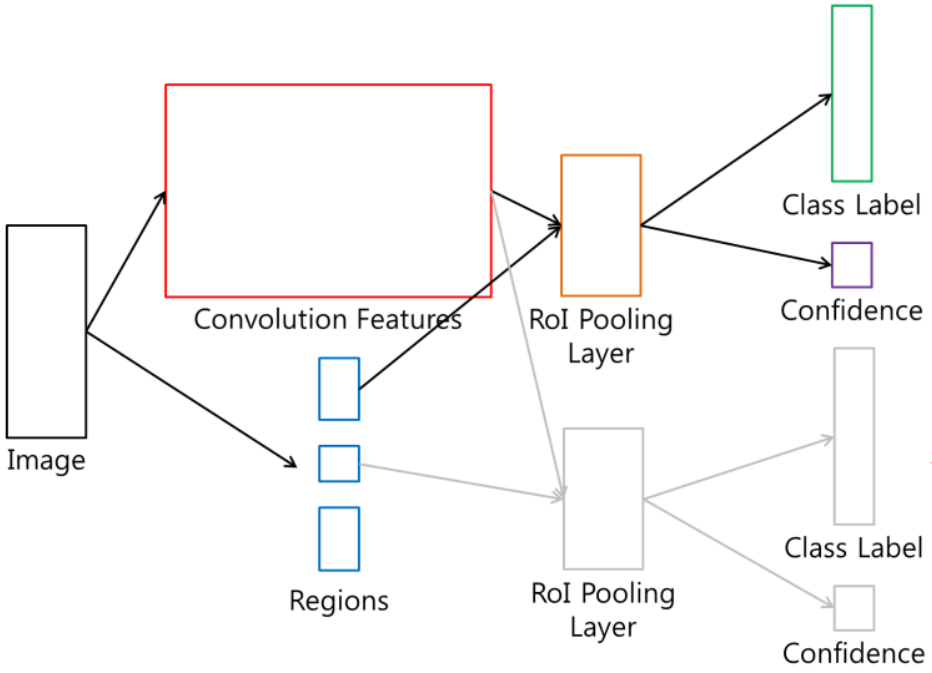
R-CNN

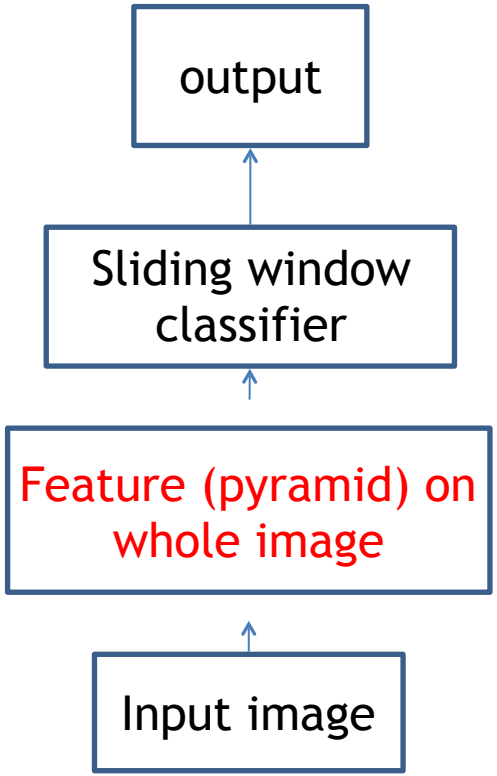




R-CNN, Fast R-CNN and Faster R-CNN

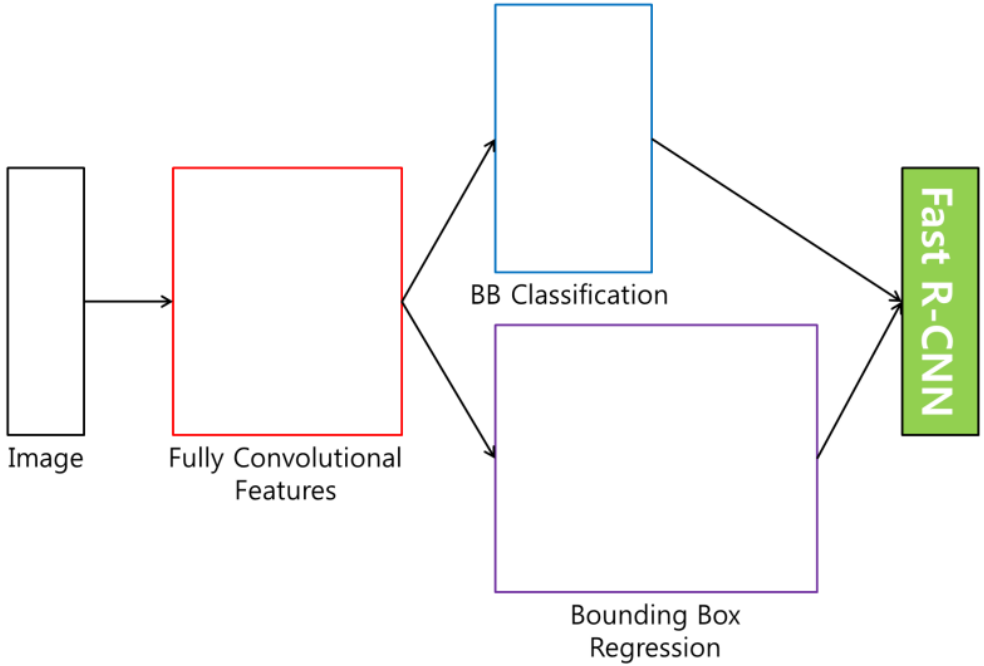
Fast R-CNN

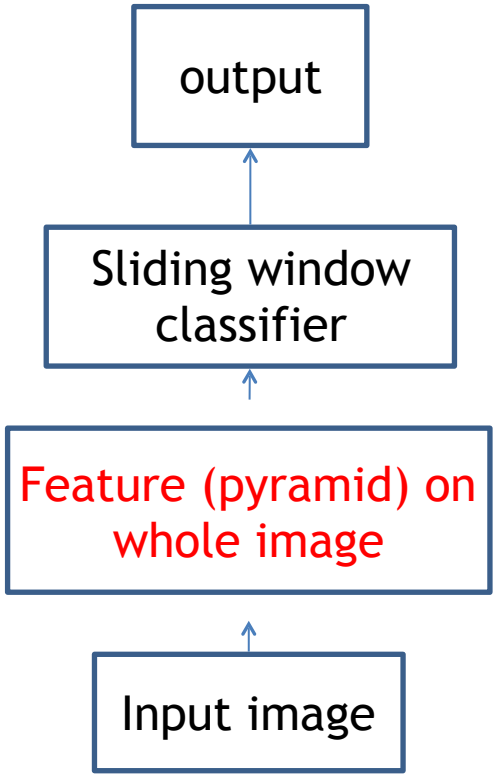




R-CNN, Fast R-CNN and Faster R-CNN

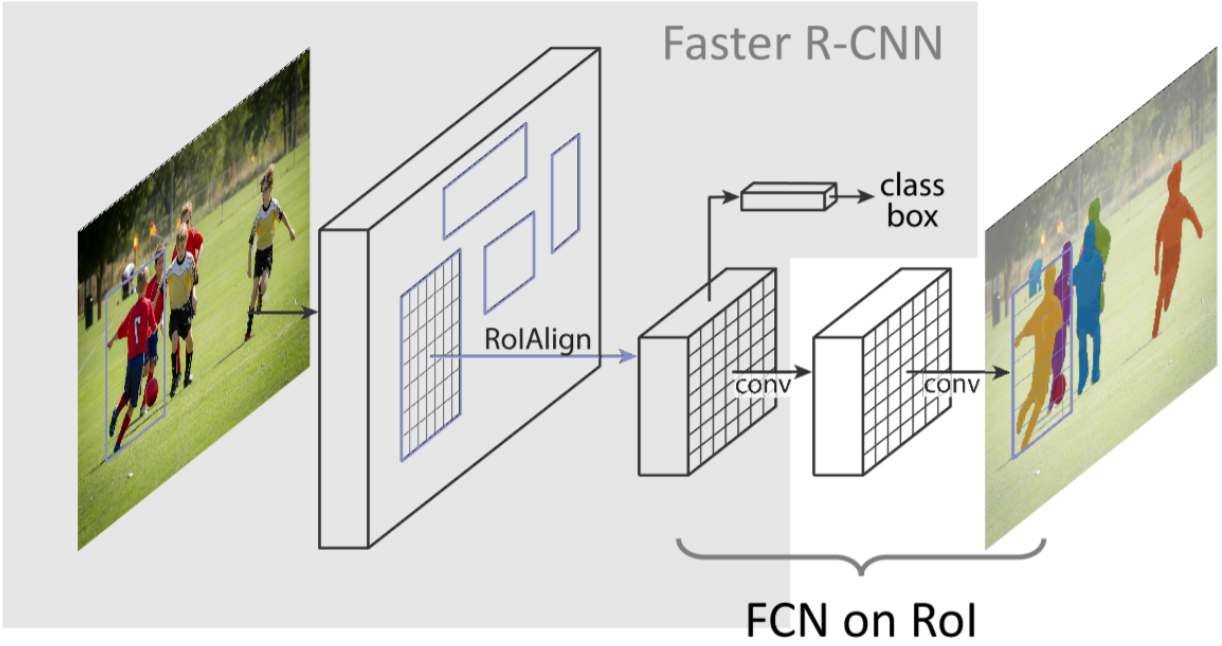
Faster R-CNN





Mask R-CNN

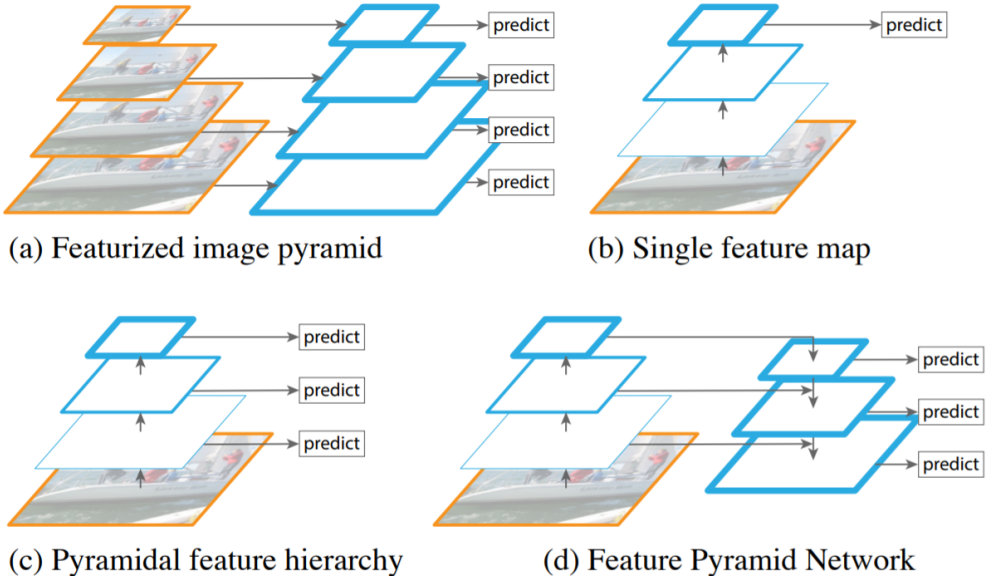
- Mask R-CNN = **Faster R-CNN** with **FCN** on RoIs

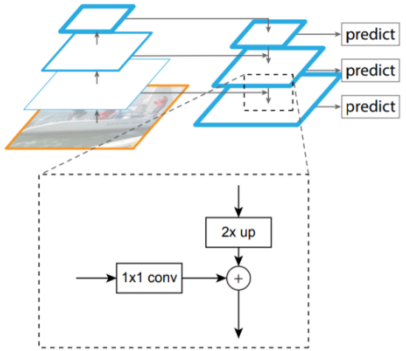


What is the latest news?

Feature pyramid networks

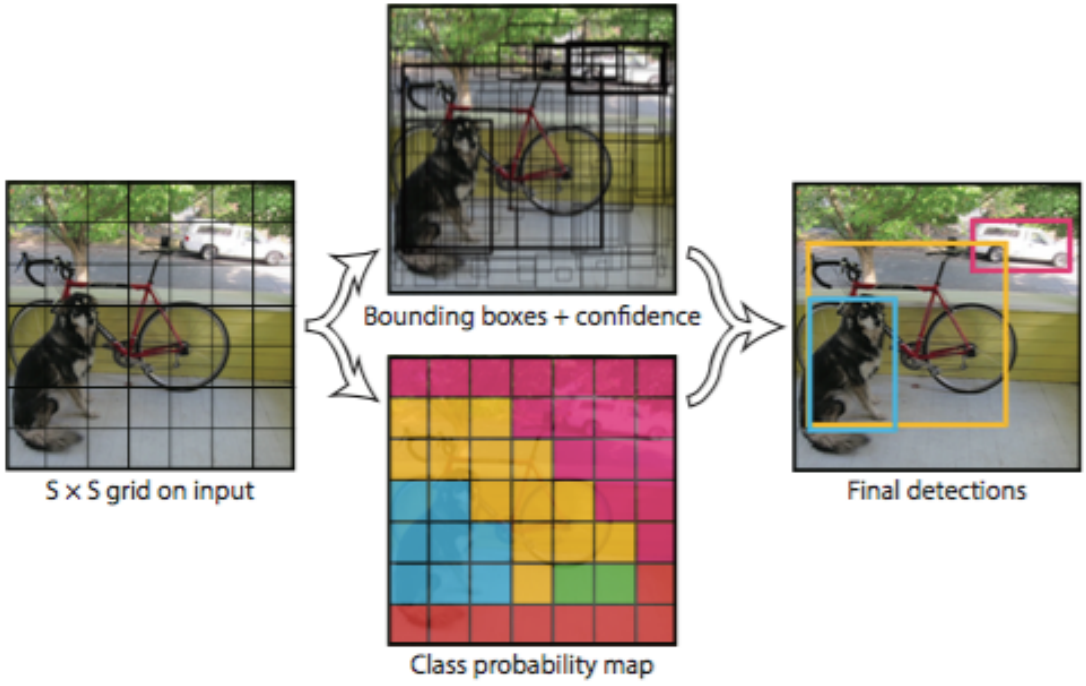
- FPN(Lin, et al.cvpr2017), RON(kong et al. cvpr2017), DSSD(Fu et al. 2017)





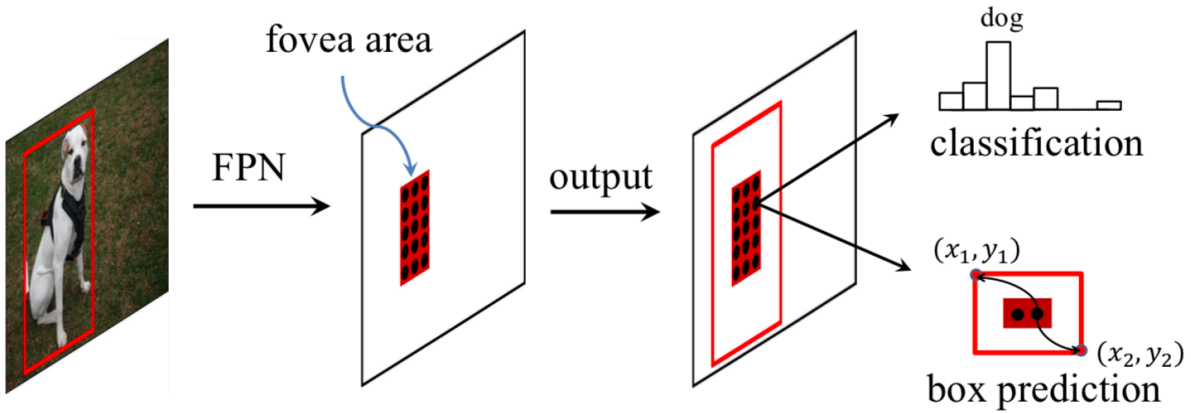
Single stage object detector

- SSD, YOLO, RetinaNet



Anchor-free detector

- CornerNet, FoveaBox, CenterNet



Object Detection code links:

R-CNN
(Cafffe + MATLAB): https://github.com/rbgirshick/rcnn
Probably don't use this; too slow

Fast R-CNN
(Caffe + MATLAB): https://github.com/rbgirshick/fast-rcnn

Faster R-CNN
(Caffe + MATLAB): https://github.com/ShaoqingRen/faster_rcnn
(Caffe + Python): https://github.com/rbgirshick/py-faster-rcnn

Useful links for learning detection

- RCNN/Fast R-CNN/Faster R-CNN: https://github.com/rbgirshick
- YOLO/YOLOv2: https://pjreddie.com/darknet/yolo/
- SSD: https://github.com/weiliu89/caffe/tree/ssd
- R-FCN: https://github.com/daijifeng001/R-FCN
- Tensorflow detector: https://github.com/tensorflow/models/tree/master/research/object_detection
- Facebook Detectron: https://github.com/facebookresearch/Detectron
- MMLab Detection: https://github.com/open-mmlab/mmdetection