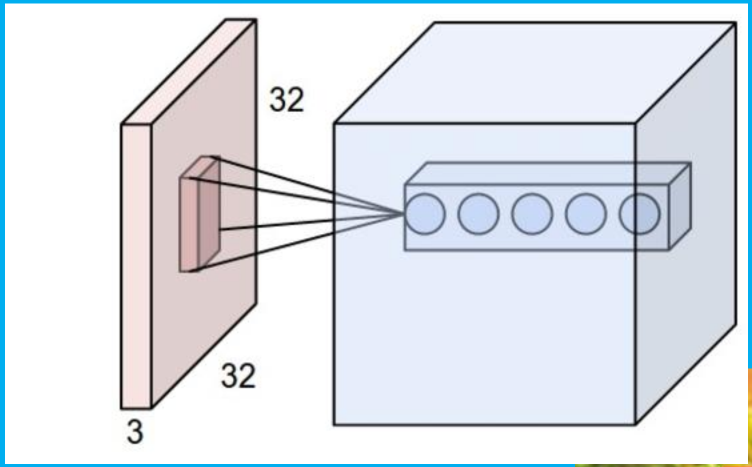
Course Presentation

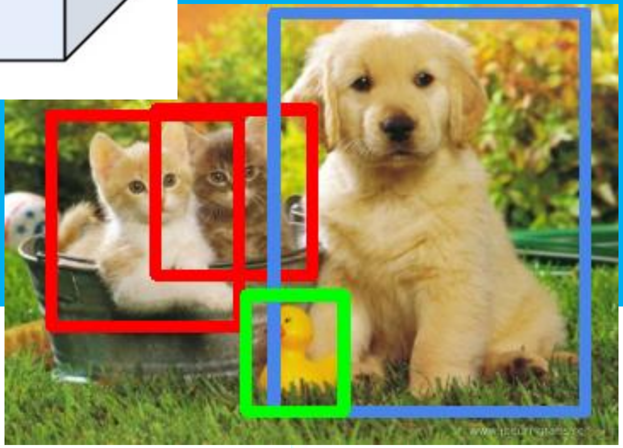
Deep Learning for Object Detection

Yiqi Yan

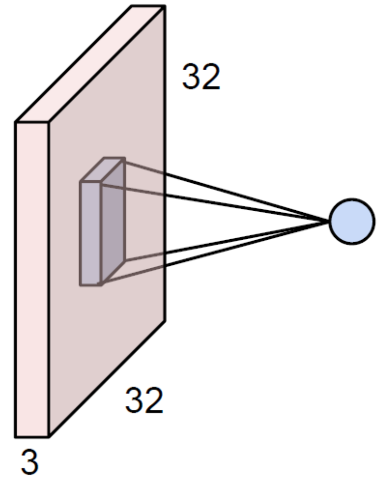
May 10, 2017

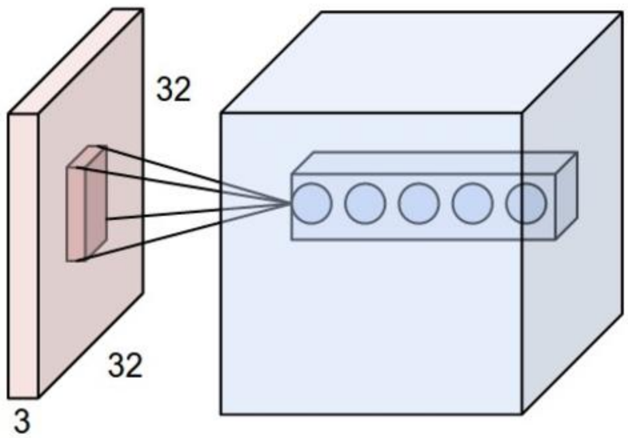
Part I
Fundamental Backgrounds





● Convolution





Single Filter

Multiple Filters

Convolution: case study, 2 filters

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	1	1	1	2	1	0
0	0	0	0	1	0	0
0	2	1	2	2	2	0
0	1	1	2	0	1	0
0	0	1	0	0	2	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	0	0	1	0
0	2	2	2	1	1	0
0	0	0	1	0	2	0
0	1	0	1	0	0	0
0	0	2	1	1	2	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	0	1	1	2	0
0	0	1	2	2	0	0
0	1	2	2	1	1	0
0	1	0	1	2	1	0
0	0	1	2	0	0	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

1	1	-1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
-1	0	1
-1	1	1

w0[:,:,2]

1	-1	-1
0	1	1
1	0	-1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

0	-1	-1
-1	1	1
1	1	-1

w1[:,:,1]

-1	0	0
0	0	0
0	-1	-1

w1[:,:,2]

0	-1	-1
1	1	1
1	-1	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

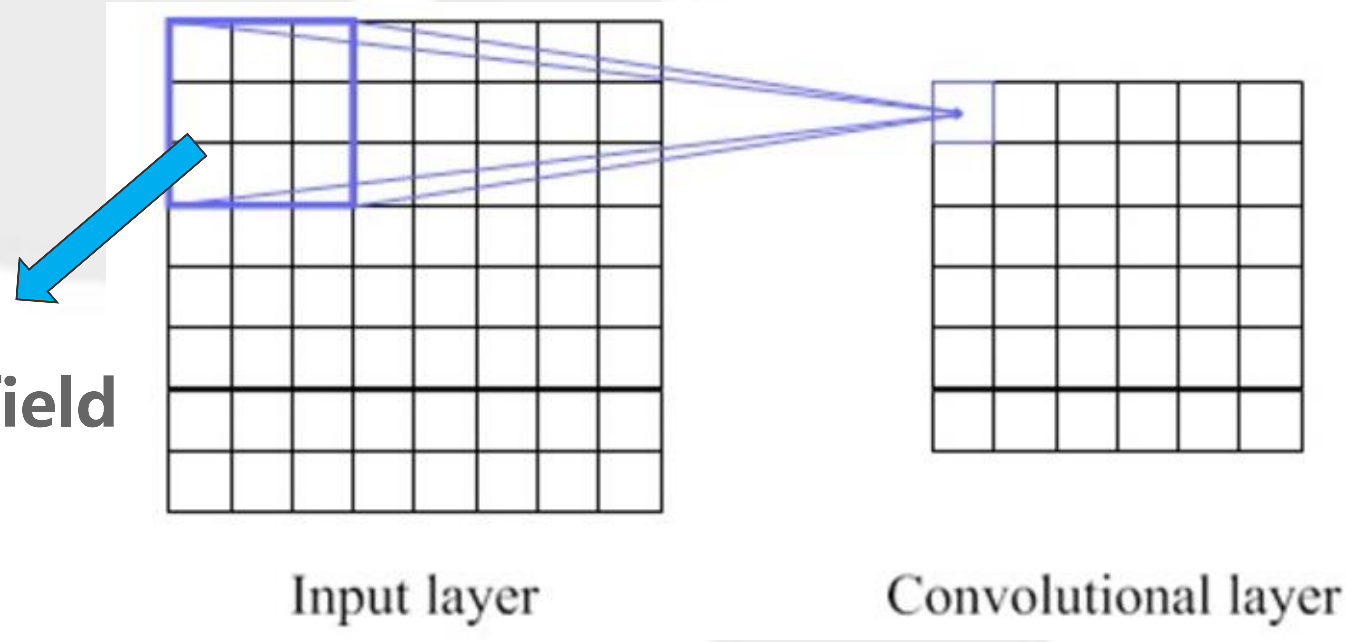
o[:,:,0]

8	5	6
7	-1	5
4	2	2

o[:,:,1]

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

● Convolution: receptive field



receptive field

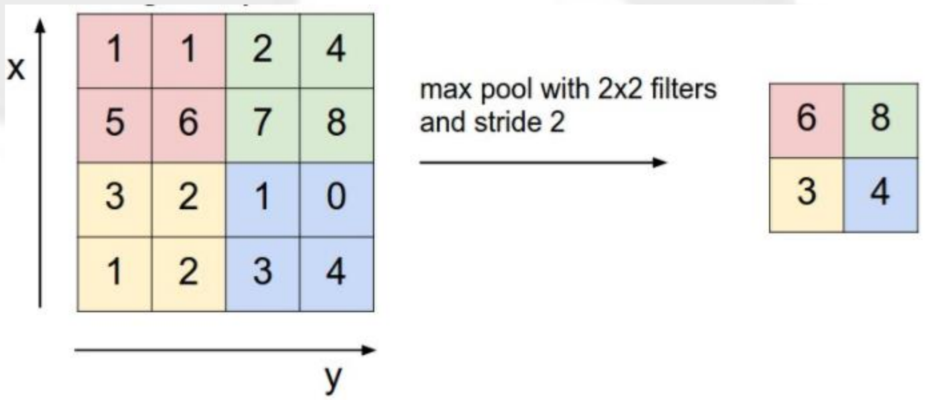
3 X 3

The deeper the network goes,

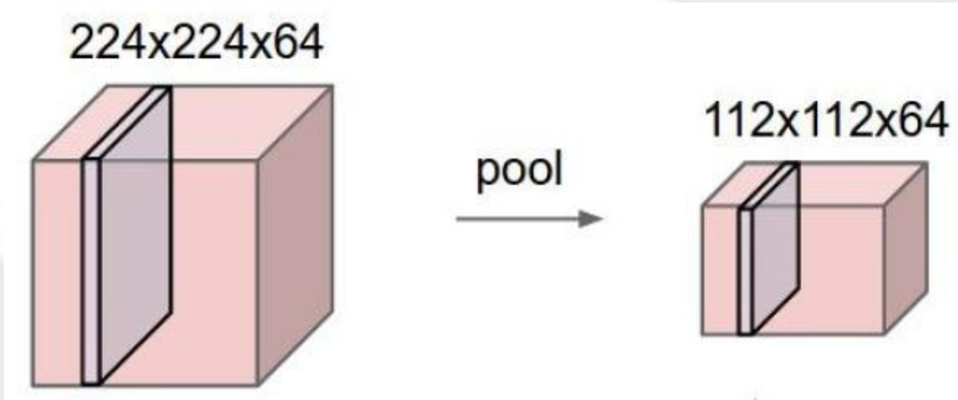
the larger the receptive fields will be

Pooling

Pooling on one single channel



Pooling on the whole feature map



Computer Vision Tasks

Classification

Input: Image
Output: Class label
Evaluation metric: Accuracy

"this is a cat"

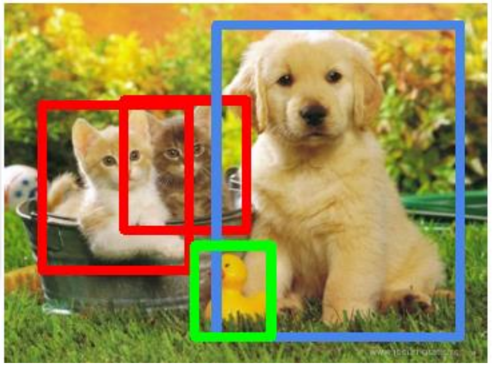
Classification + Localization

Input: Image
Output: Box in the image (x, y, w, h)
Evaluation metric: Intersection over Union

"here is a cat"

● Computer Vision Tasks

Object Detection



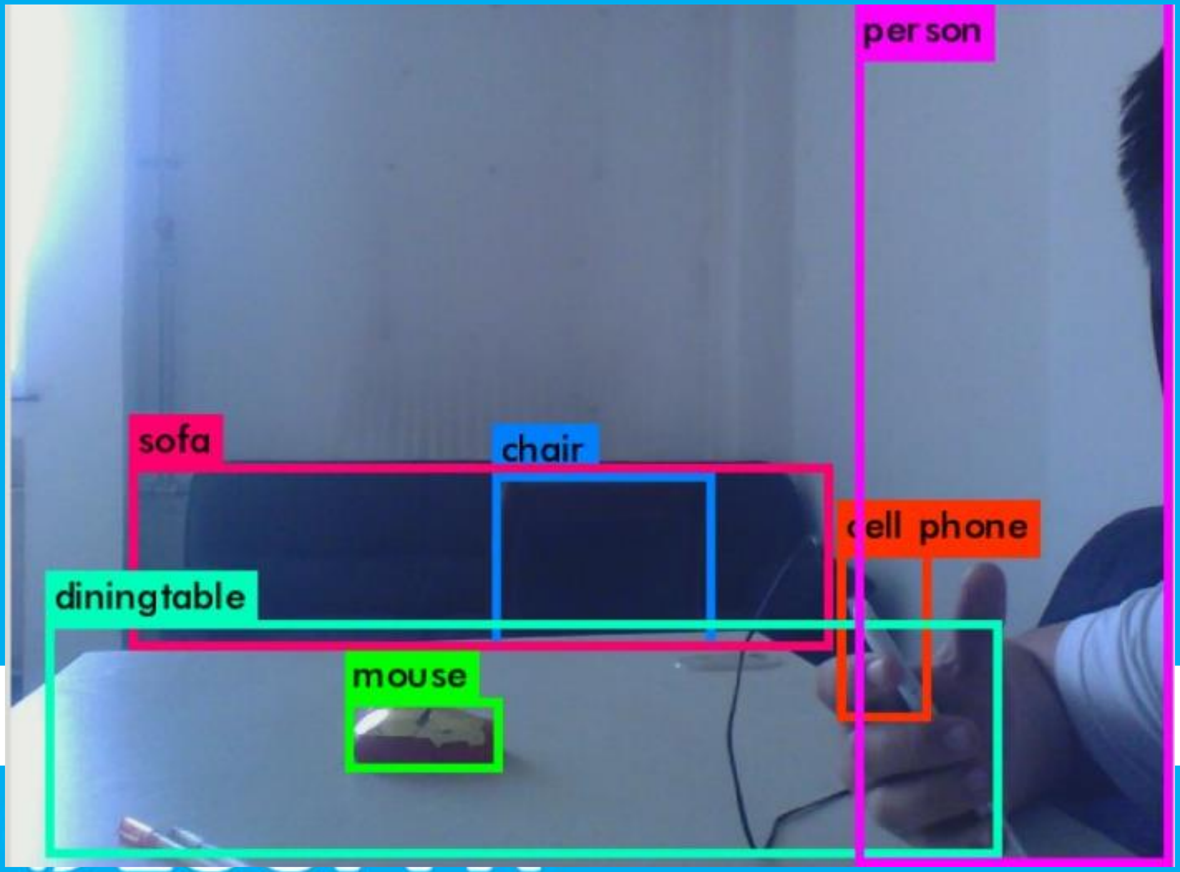
DOG, (x, y, w, h)
CAT, (x, y, w, h)
CAT, (x, y, w, h)
DUCK (x, y, w, h)

Multiple Objects in one image!

Instance Segmentation

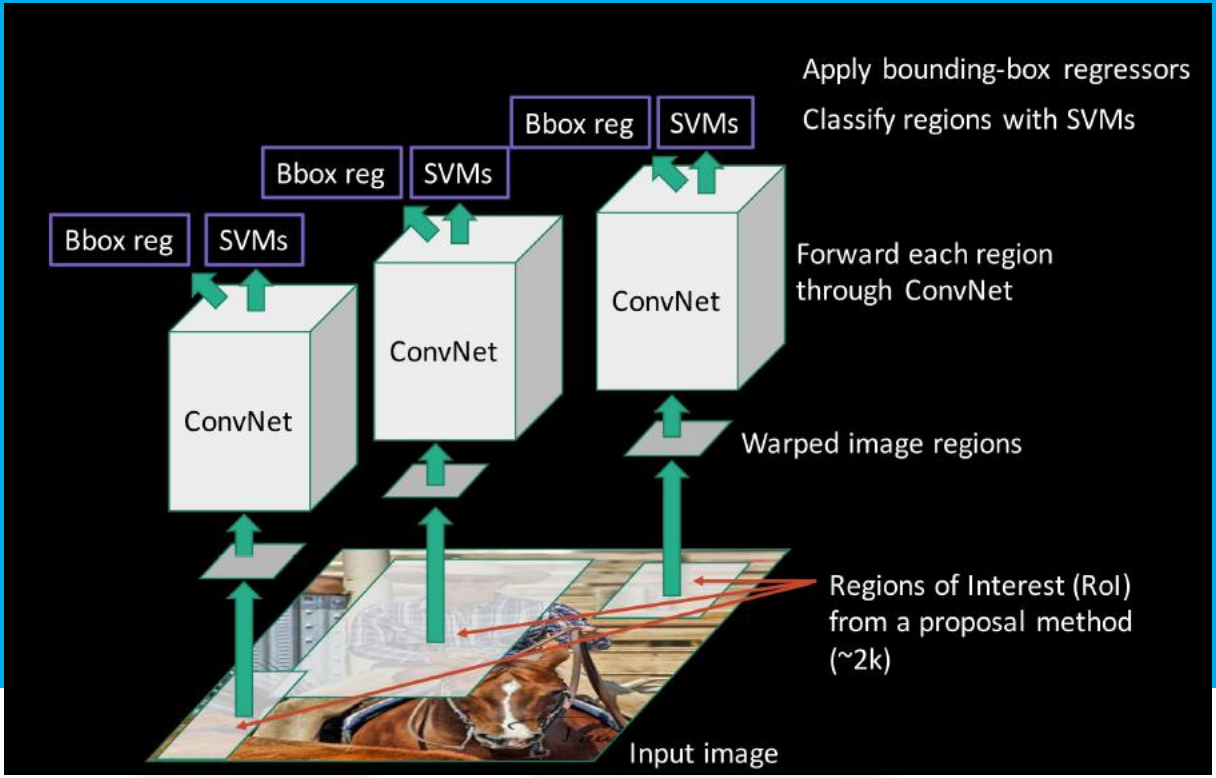
Pixel-wise Classification!

Part II
State-of-art methods for object detection



- Group One: RCNN & Modifications
  - (CVPR 2014) RCNN: Region Based CNN *(TOO SLOW!)*
  - (ECCV 2014) SPP-net: Spatial Pyramid Pooling in CNN
  - (ICCV 2015) Fast RCNN
  - (NIPS 2015) Faster RCNN *(Online Object Detection)*
- Group Two: Fast! *Online* Object Detection
  - (ECCV 2016) SSD: Single Shot MultiBox Detector
  - (CVPR 2016) YOLO
  - (CVPR 2017) YOLO9000
- Group Three: Deformable Convolutional Filters
  - (CVPR 2015) DeepID-Net
  - (arXiv 2017) Deformable Convolutional Networks
- Group Four: Detection + Segmentation
  - (arXiv 2017) Mask R-CNN

Part III
Region Based CNN



LOOPPA

● Motivation

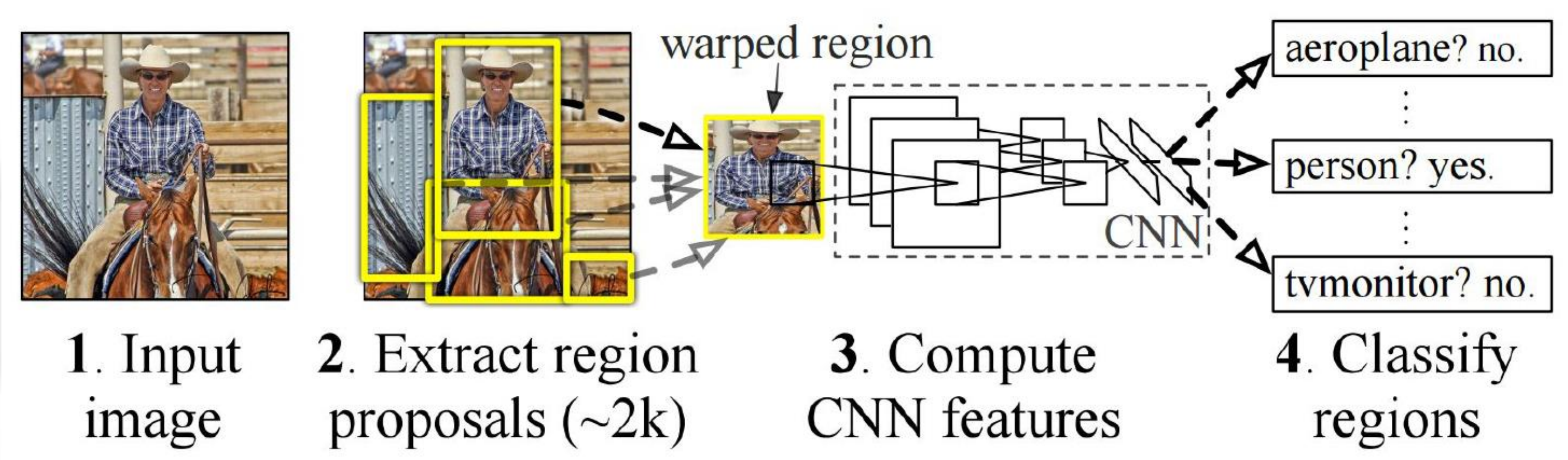
Problem One: CNN seems unsuited for Object Detection

- **Unlike image classification, detection requires *localizing objects* within an image**
- **Deep CNNs have very large receptive fields, which makes precise localization very challenging**

Problem Two: Deep networks need large dataset to train

	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**

● Framework



Solve Problem One

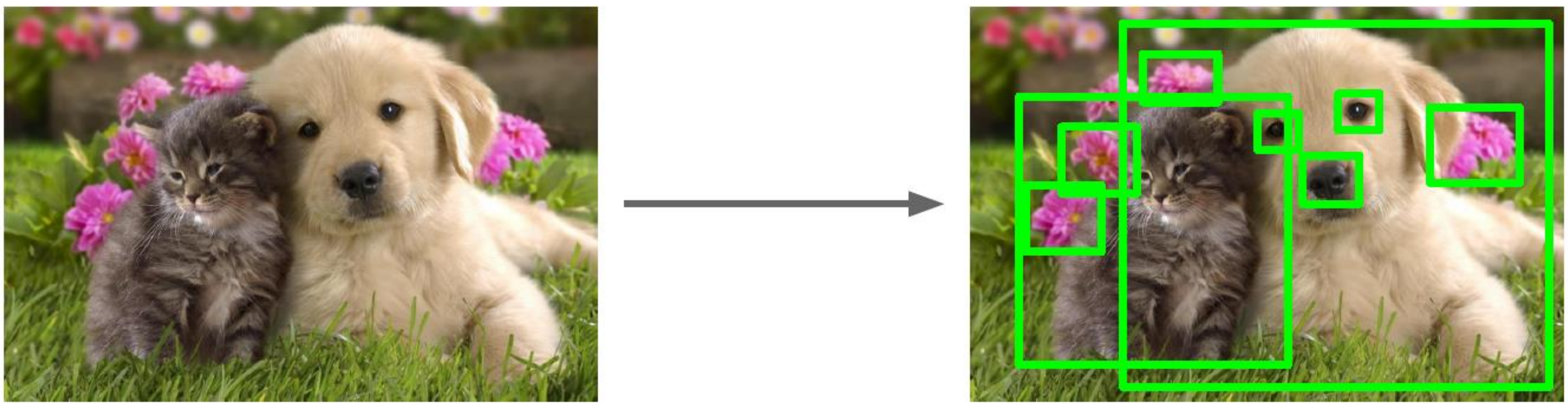
- Extract region proposals
- Recognition using regions

Solve Problem Two

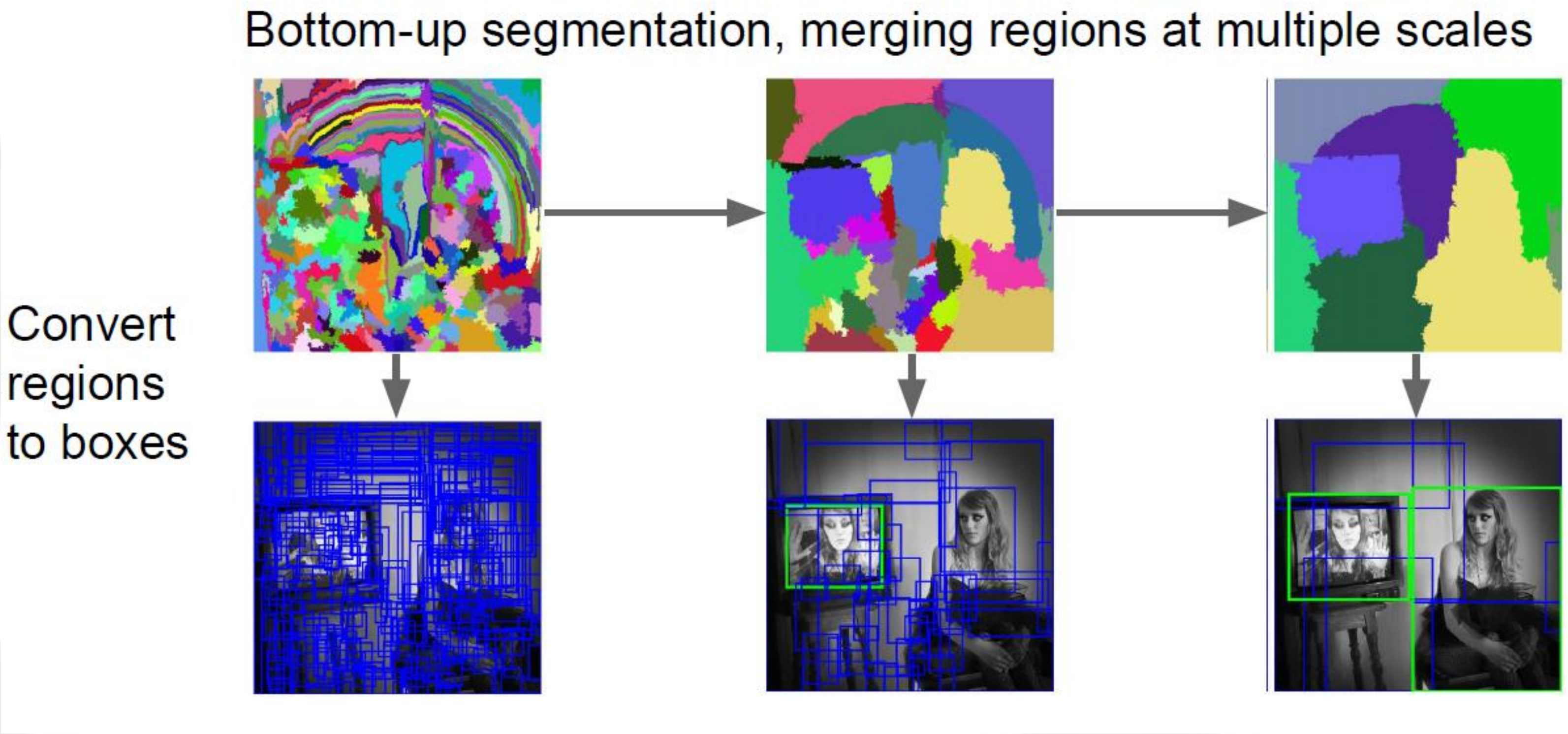
- Supervised Pre-training on ImageNet
- Domain-specified fine-tuning

● Region Proposals

Find “blobby” image regions that are likely to contain objects



Region Proposals: Selective Search



over-segmentation → region-merging

Uijlings et al, "Selective Search for Object Recognition", IJCV 2013

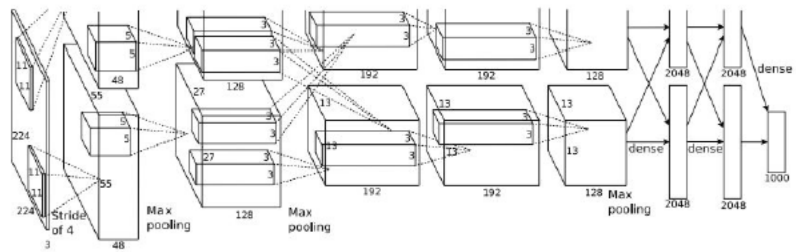
● Training method

Step 1: Supervised Pre-training

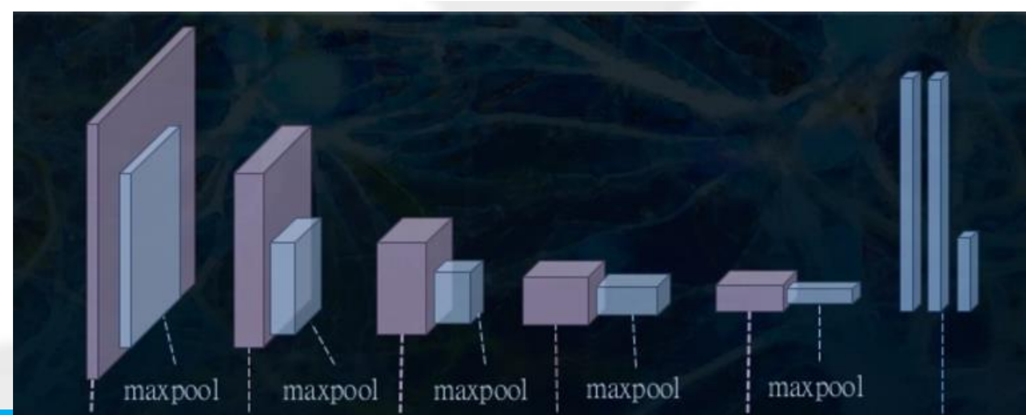
Train a *classification* model for ImageNet (AlexNet, VGGNet, etc.)

No localization can be done, thus this is just pre-training the parameters.

AlexNet
(2012)



VGG-19
(2015)



- **Training method**

Step 2: Domain-specified fine-tuning

Change network architecture

Instead of 1000 ImageNet classes, want N object classes + background (N+1 classes)

Need to reinitialize the soft-max layer

Fine-tuning using Region Proposals

Keep training model using positive / negative regions from detection images

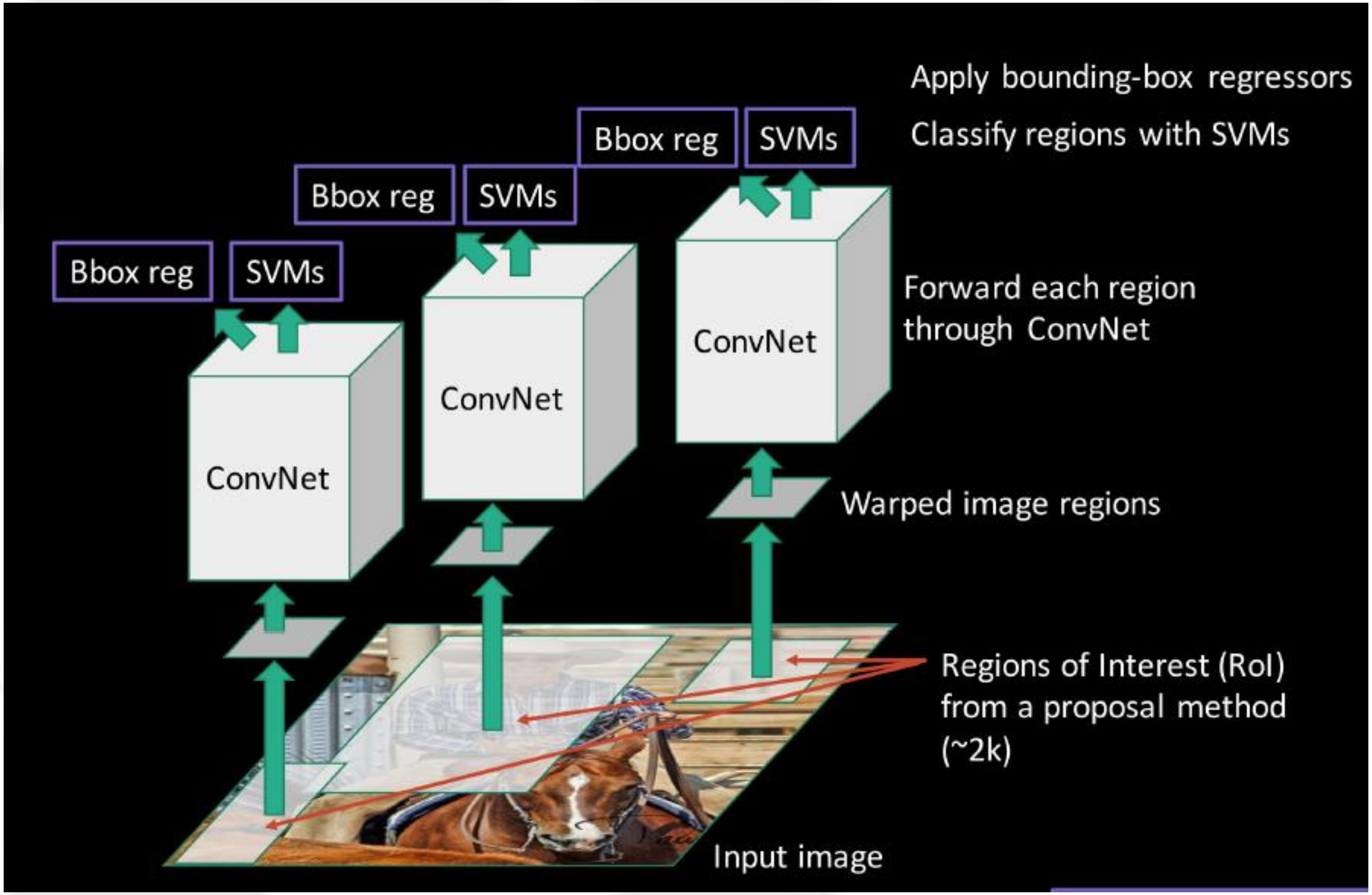
This time, use ***Detection Datasets*** (VOC, ILCVRC, COC, etc.)

- # Run Detection

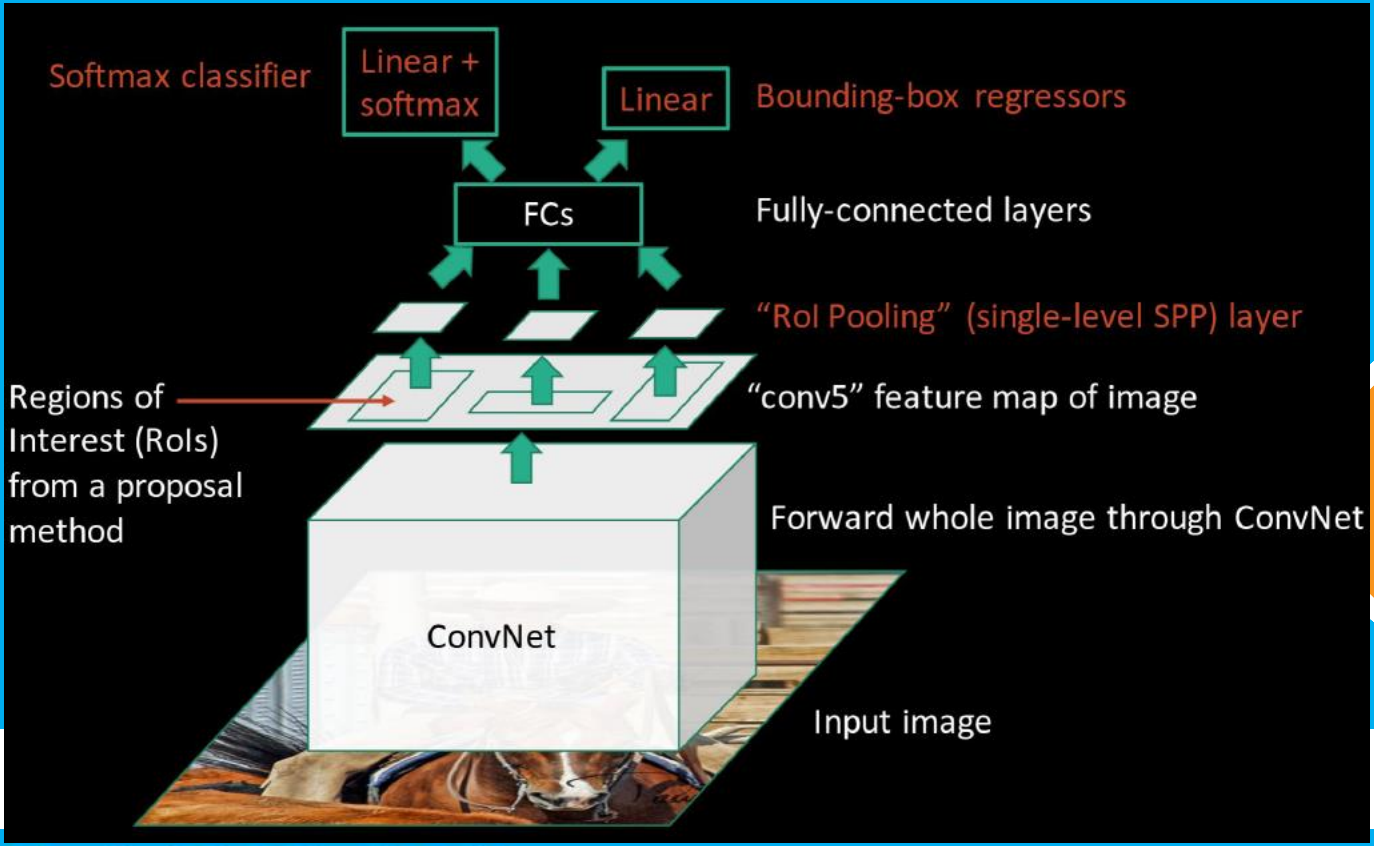
Now we get the trained network, let us test it!

- **Step 1: Extract *region proposals* for all images**
- **Step 2: (for each region) run through CNN, save *pool5* features**
- **Step 3: use binary SVM to classify region features *(WHY NOT just use soft-max)***
- **Step 5: bounding box regression: For each class, train a *linear regression model* to make up for "slightly wrong" proposals**

Run Detection



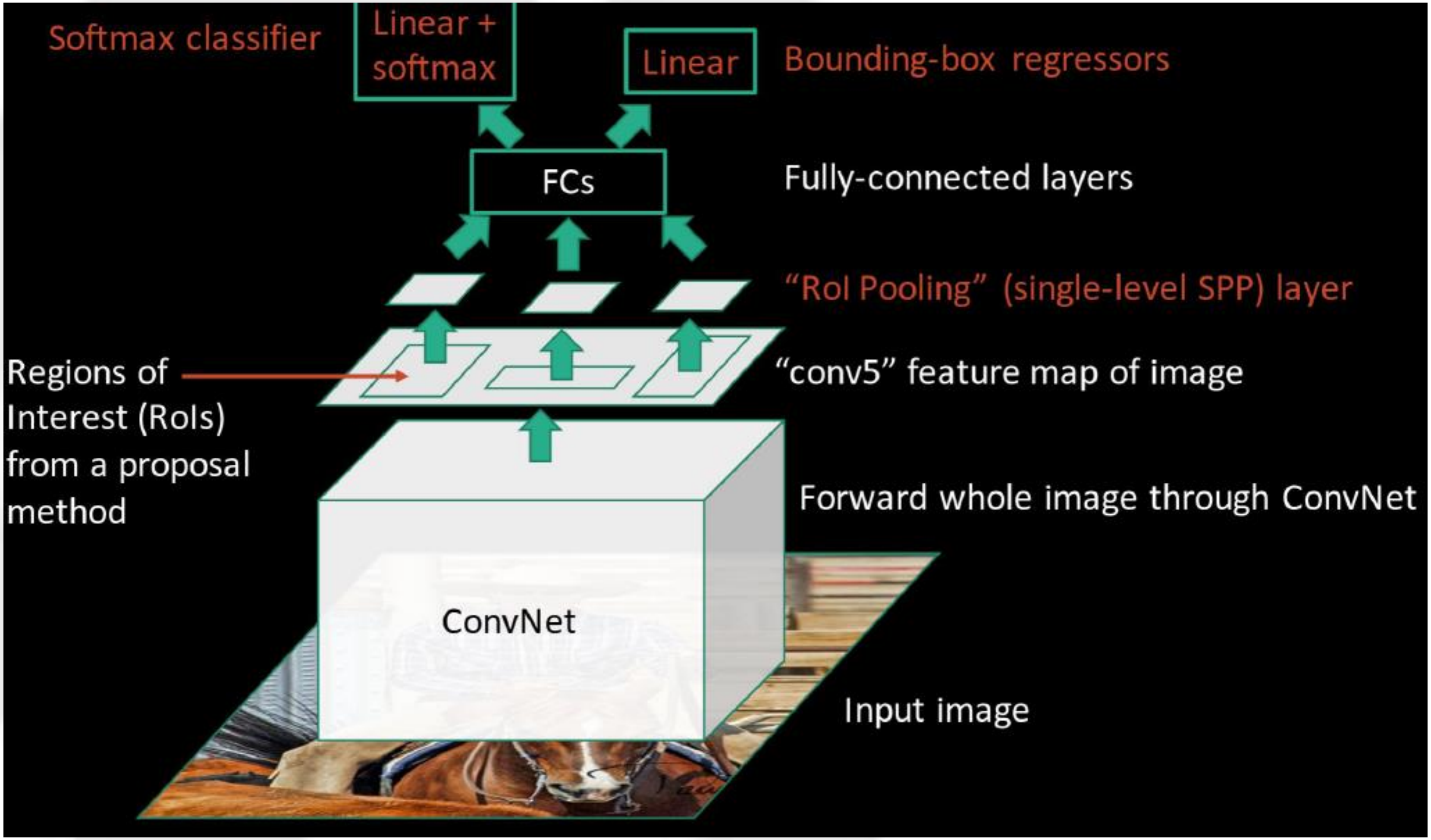
Part IV
Fast RCNN



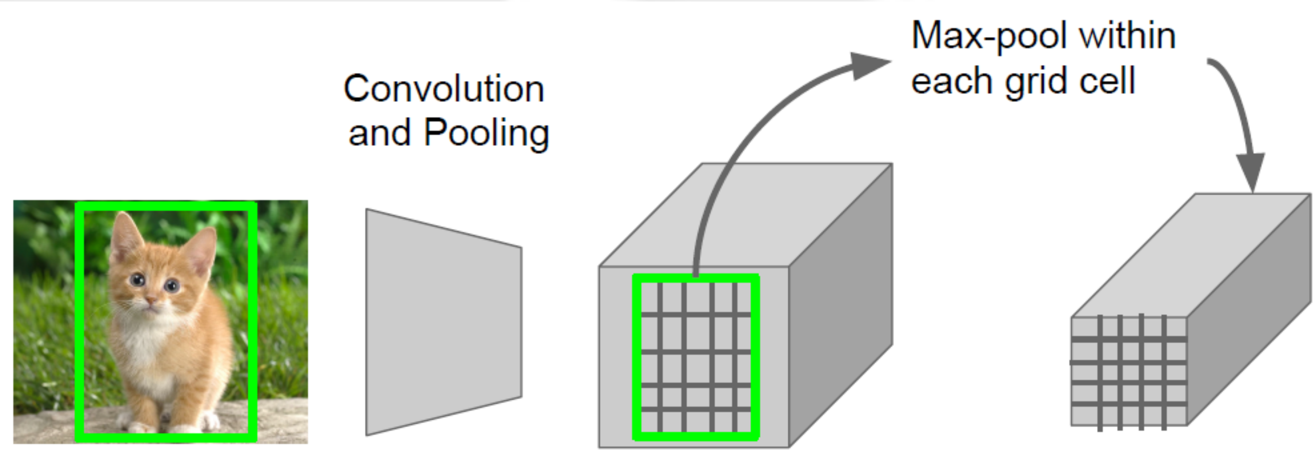
- # R-CNN Problems: too slow!

  - **Training is a multi-stage pipeline:**
    RCNN⟶SVMs⟶bounding-box regression
  - **Training is expensive in space and time**
    CNN features are stored **for use of training SVMs and regression**
    ~200GB disk place for PASCAL dataset!
  - **Object detection is slow:**
    features are extracted from *each object proposal*
    47s / image on a GPU!

Fast R-CNN Framework



ROI (region of interest) Pooling



- **Pooling each region into a *fixed size* (7 X 7 in the paper)**
- **Back propagate similar to traditional max pooling**

- # Multi-task loss

The network outputs two vectors per ROI

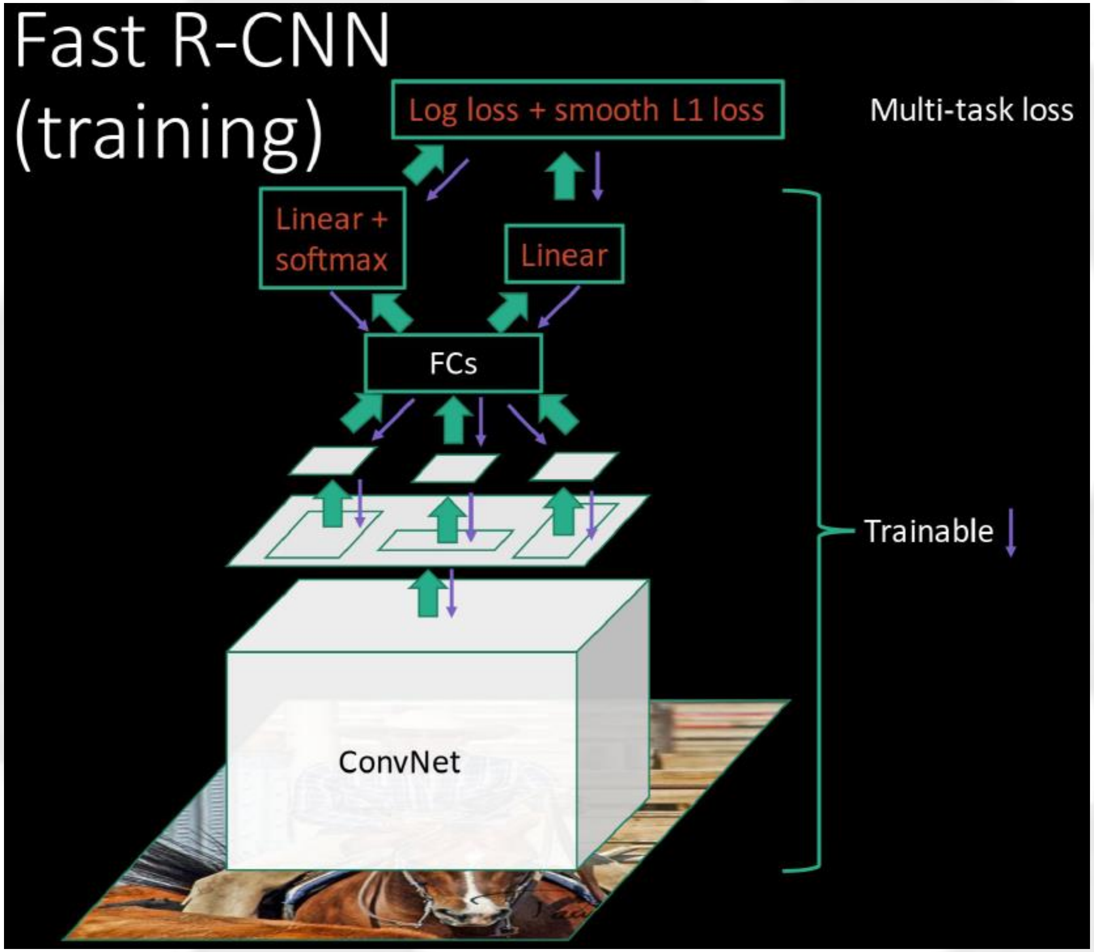
1. Soft-max probabilities for ***classification***
2. Per-class ***bounding-box regression*** offsets

$$L(p, u, t^u, v) = \boxed{L_{\text{cls}}(p, u)} + \lambda[u \geq 1]\boxed{L_{\text{loc}}(t^u, v)}$$

1st term: traditional cross-entropy loss for soft-max

2nd term: error between predicted and true bounding-box

- **Why superior to RCNN: problem #1 & #2**

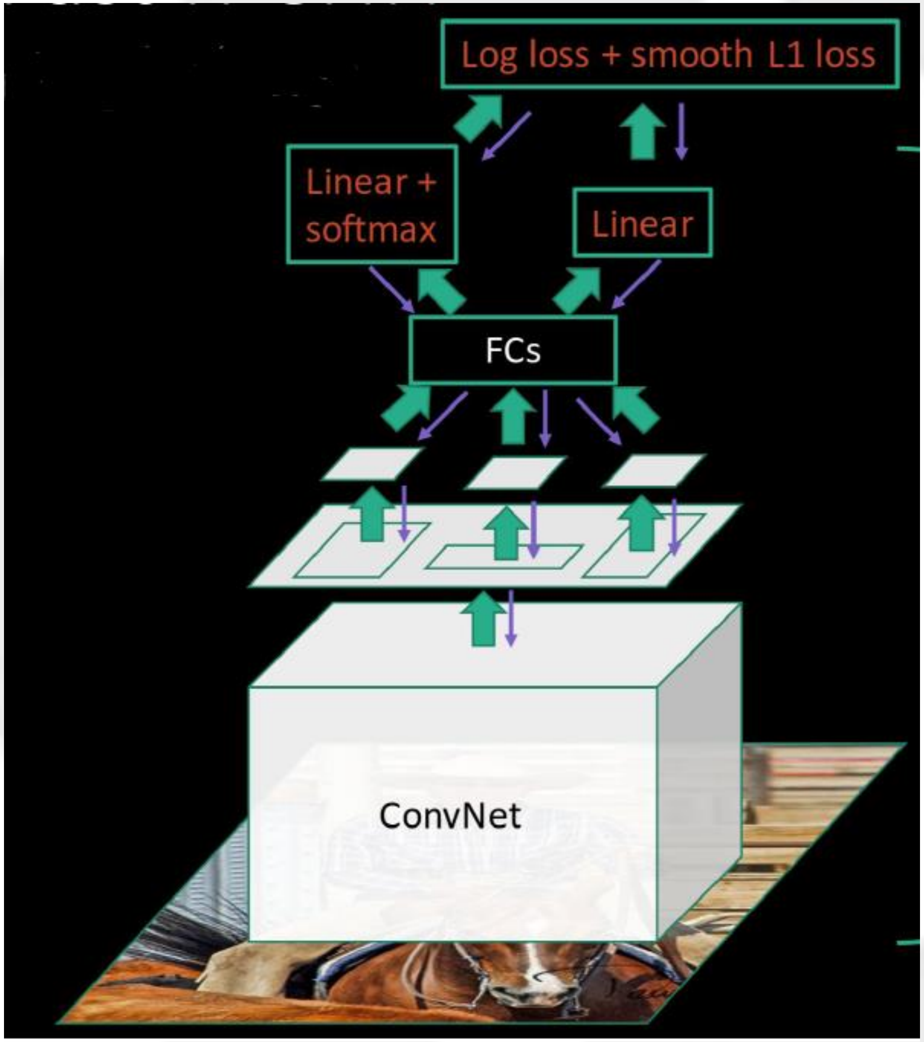


RCNN problem #1: Training is a multi-stage pipeline

RCNN problem #2: Training is expensive in space and time

Faster RCNN: end-to-end training; no need to store features

Why superior to RCNN: problem #3



RCNN problem #3: Object detection is slow because features are extracted from each object proposal

Fast RCNN: just run the whole image through CNN; regions are extracted from feature map

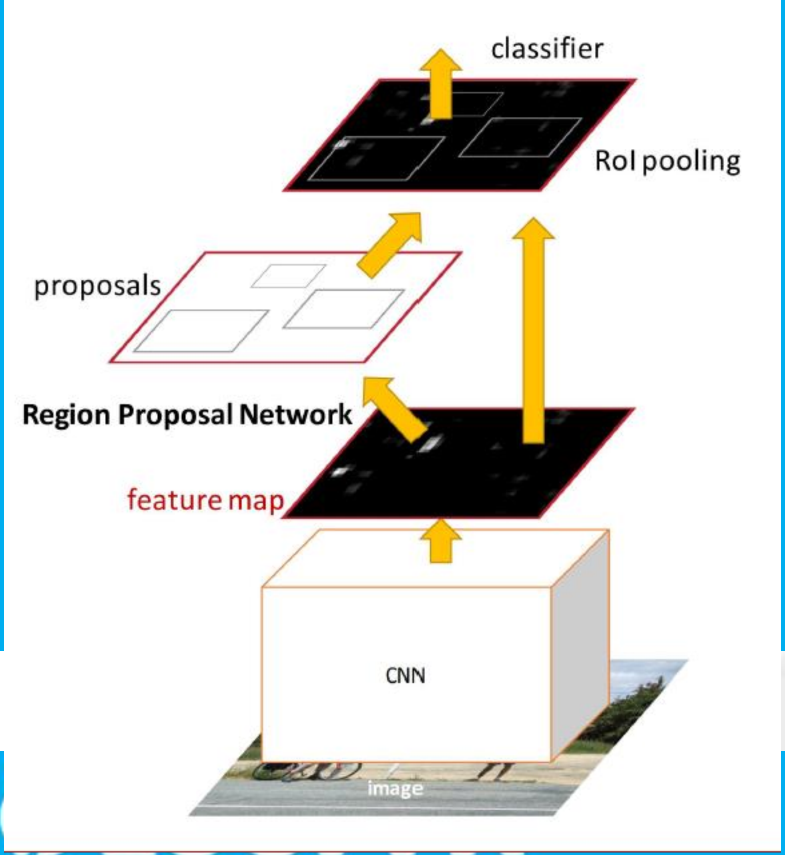
Comparison

	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 RCNN is not fast enough!

Bottleneck： Selective Search Region Proposal

Part V
Faster RCNN
Online Detection!



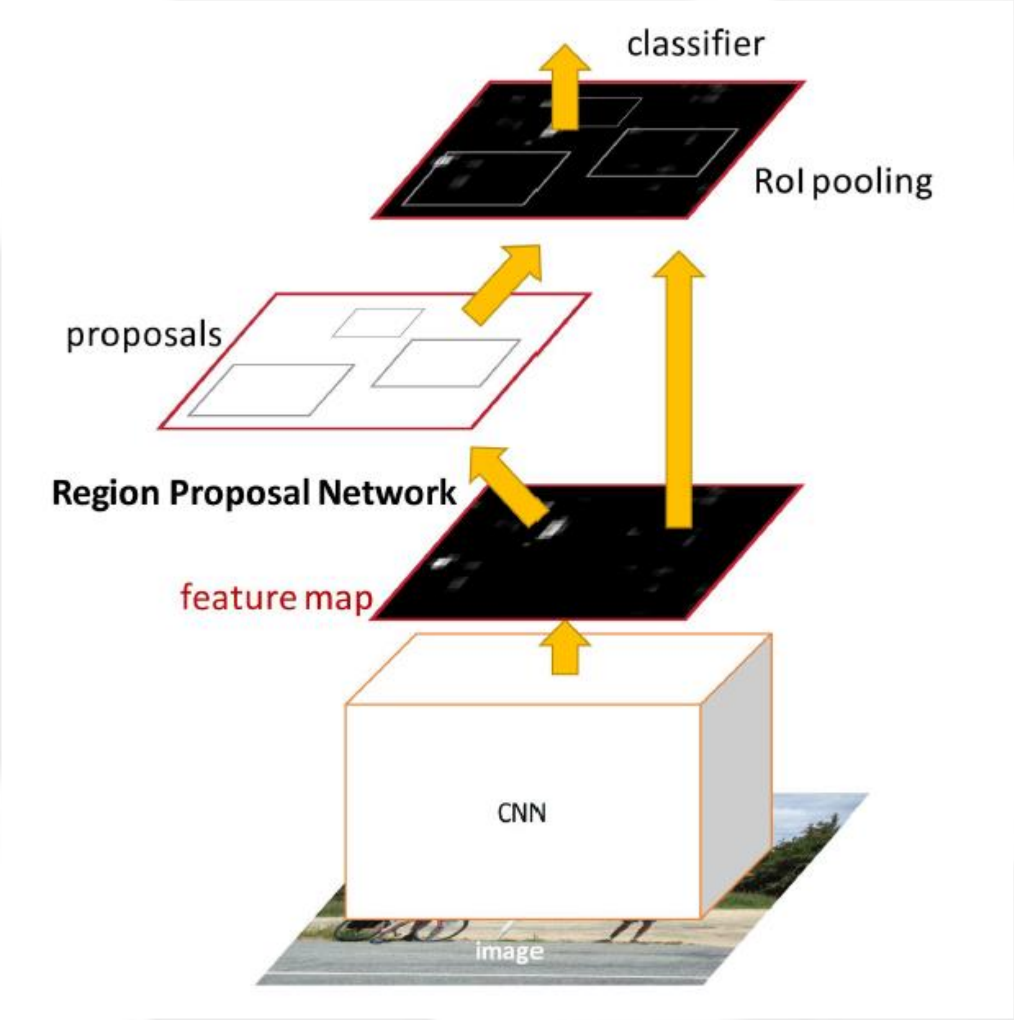
- # Fast RCNN is not fast enough

Test time per image	**0.32 seconds**
(Speedup)	**146x**
Test time per image with Selective Search	**2 seconds**
(Speedup)	**25x**

Main bottleneck : Selective Search Region Proposal

Faster RCNN: Why not just *make the CNN do region proposals too!*

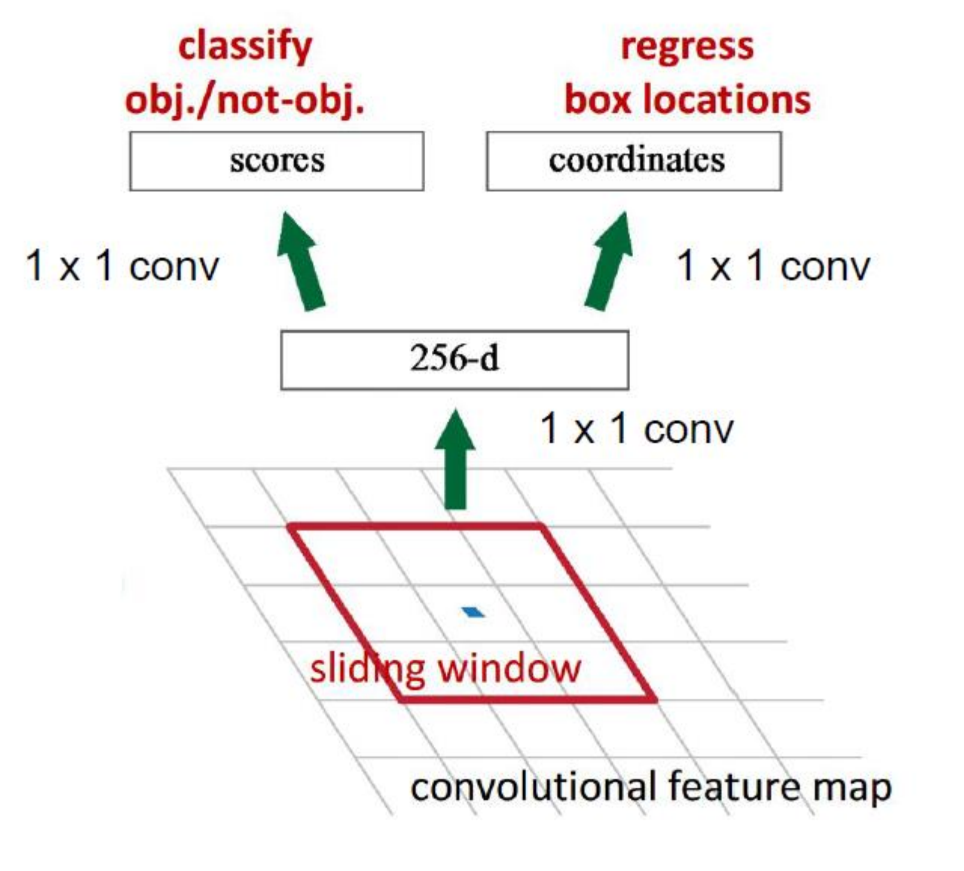
Faster RCNN framework



Insert a ***Region Proposal Network (RPN)*** trained to produce region proposals directly

ROI Pooling, soft-max classifier and bounding box regression are just like Fast RCNN

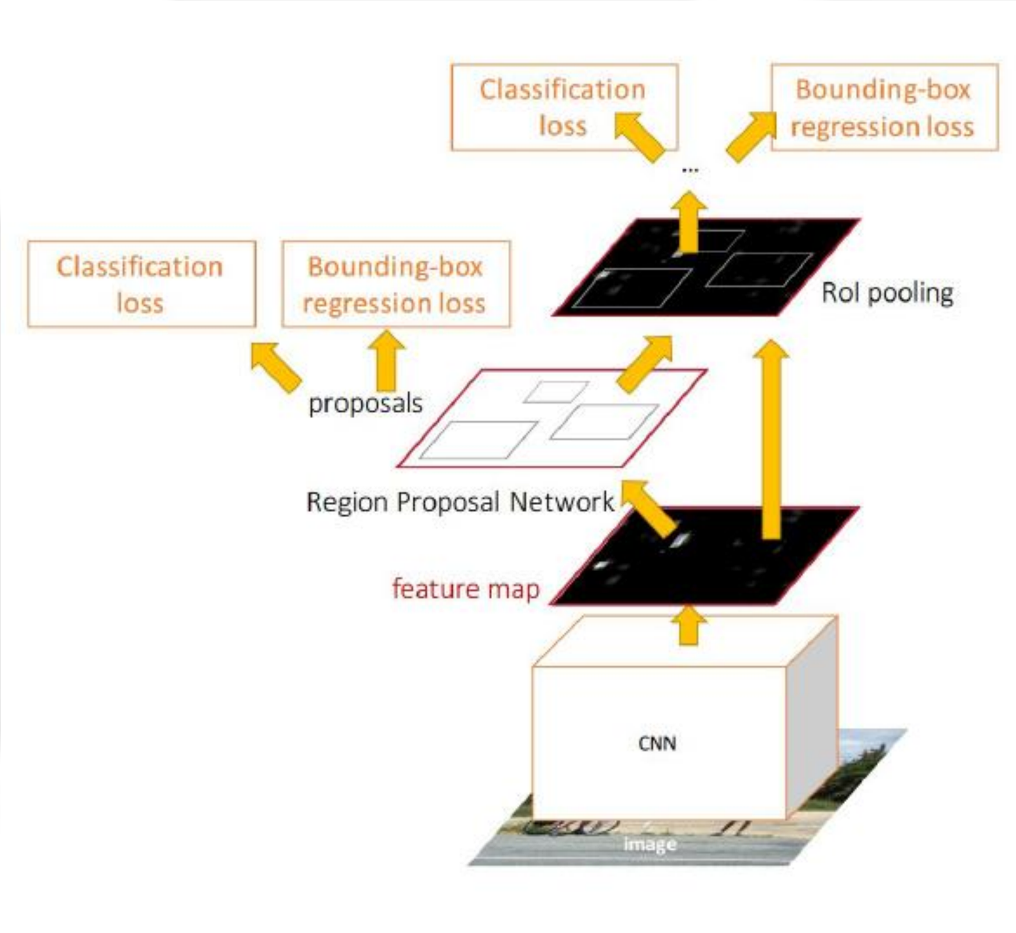
Region Proposal Network



Similar to the multi-task training in fast RCNN: ***classification + bounding box prediction***.

The difference is that we only need two-class classification here: ***object & not object***

- **End-to-end joint training!**

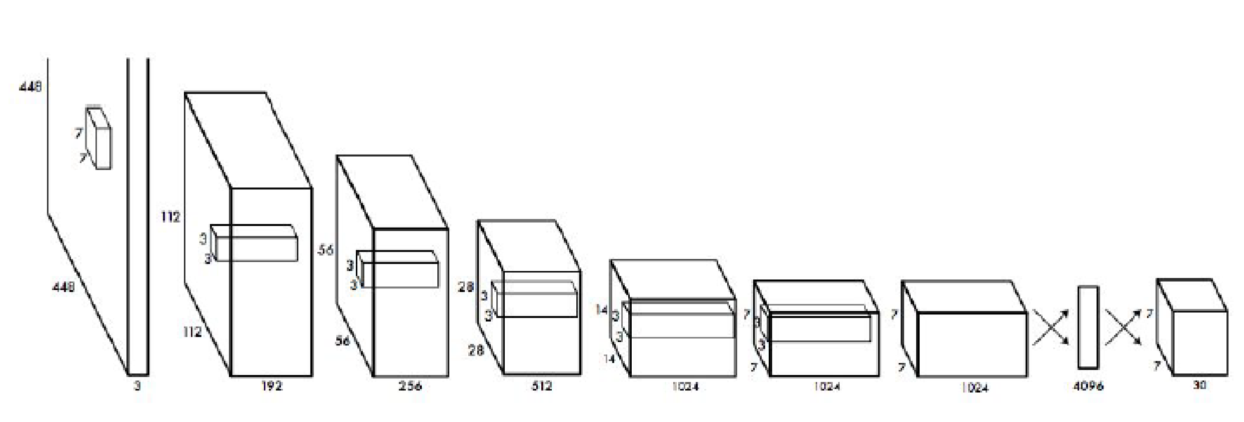


- **RPN classification**
- **RPN bbx regression**
- **Fast RCNN classification**
- **Fast R-CNN bbx regression**

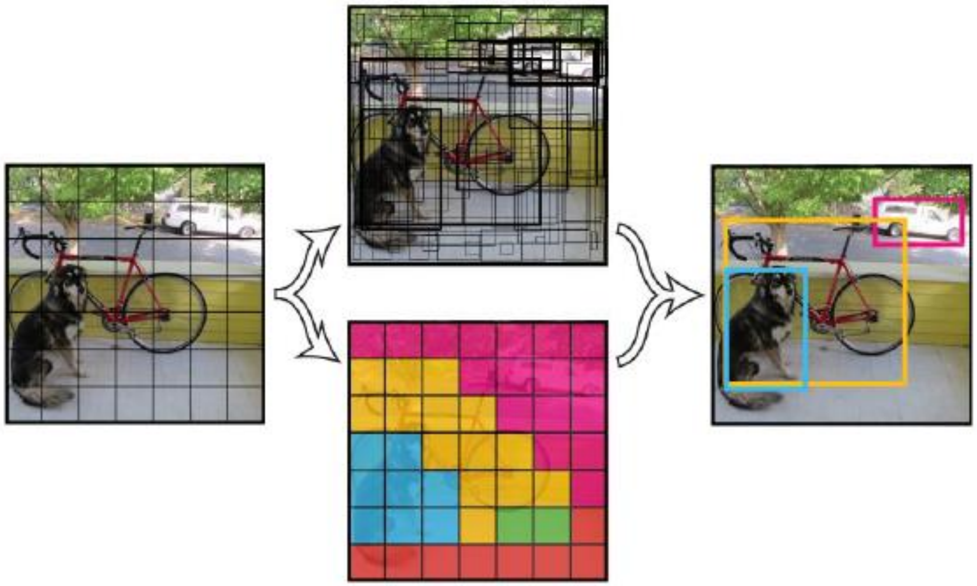
Comparison

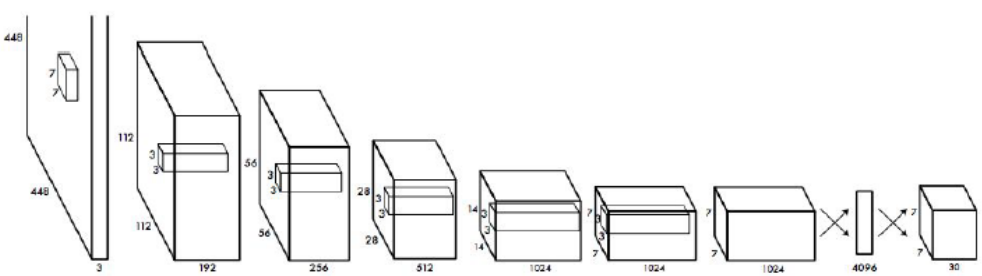
	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

Part VI
YOLO:
You Only Look On Once



YOLO Framework





- Divide image into S x S grid (7 X 7 in the parper)
- Within each grid cell predict:

  B Boxes: 4 coordinates + confidence

  C Class scores
- Regression from image to _7 x 7 x (5 * B + C)_ tensor
- Direct prediction using a CNN

Comparison

Faster than Faster R-CNN, but not as good

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18

- Great Contributors
  - Ross Girshick: http://www.rossgirshick.info/
  - Kaiming He: http://kaiminghe.com/
  - Joseph Chet Redmon: https://pjreddie.com/
- Code
  - R-CNN

    (Cafffe+MATLAB): https://github.com/rbgirshick/rcnn
  - 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
  - YOLO: http://pjreddie.com/darknet/yolo/