

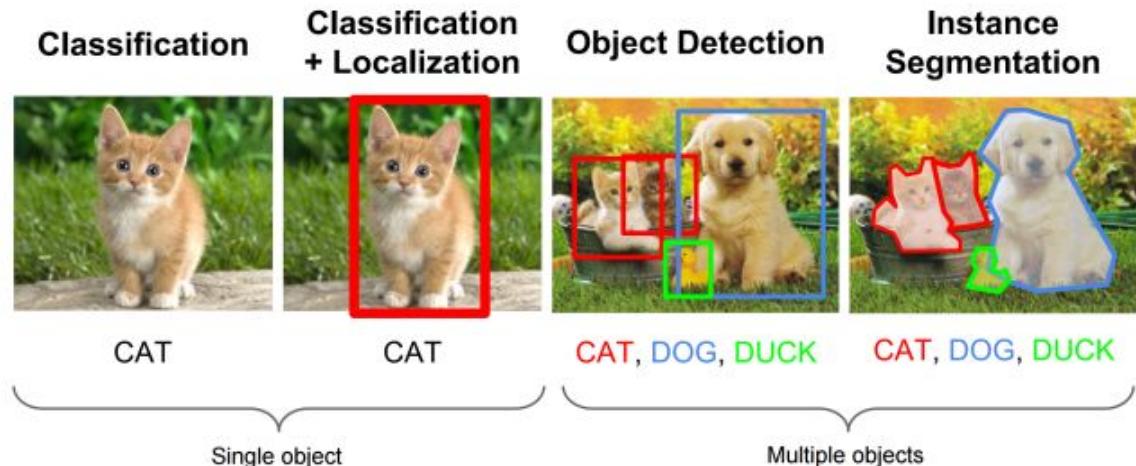
академия
больших
данных



Neural Networks for detection

Boris Lestsov
Mail.Ru Group

Detection task

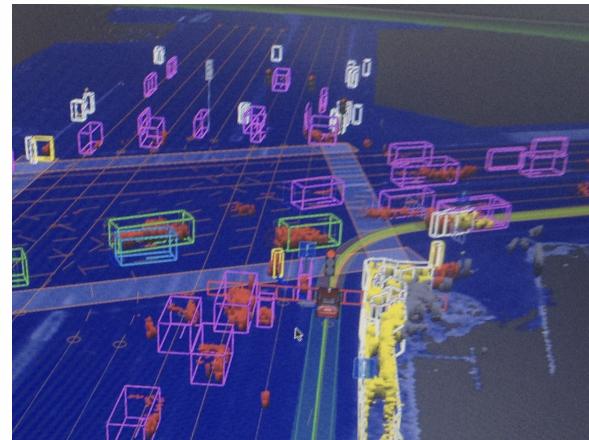
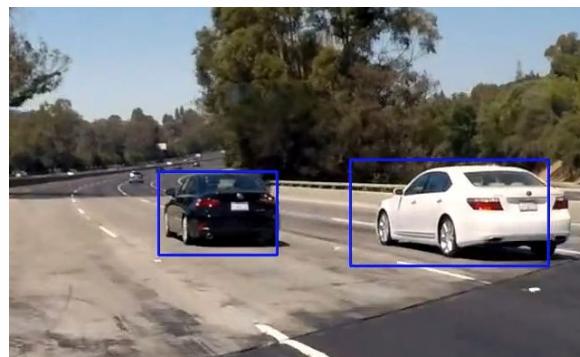
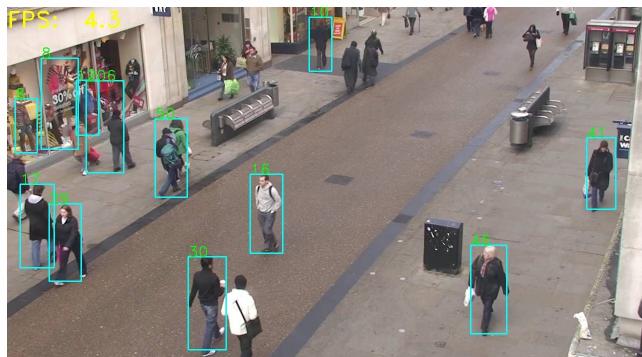
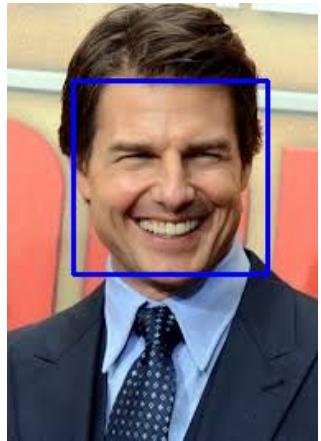


Differences with image classification task:

- 1) One image may contain objects of multiple classes or may not contain any.
- 2) One image may contain multiple objects of a single class.
- 3) Need to localize object on the image by predicting it's Bounding Box (bbox).

Applications

- 1) Face detection:
 - a) Camera autofocus
 - b) Face Recognition
 - 2) Self-driving cars/robots/drones.
 - 3) Vehicle detection
 - 4) Pedestrian detection

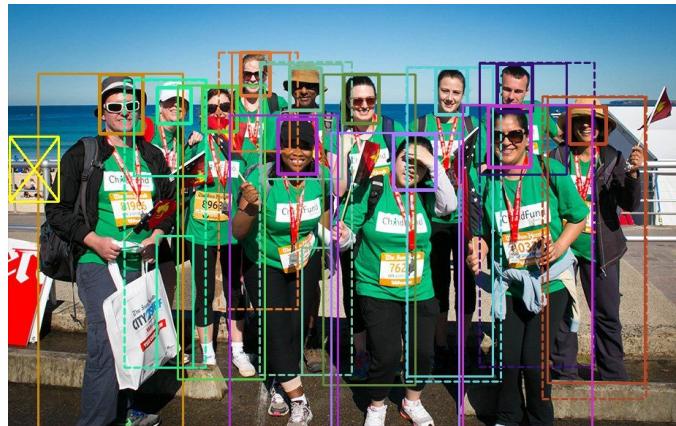


Difficulties

- 1) How to localize objects? How to predict bounding boxes?
- 2) Bounding boxes have completely size.
- 3) Simple resize will not work in many cases.
- 4) Intra-class variation
- 5) Overlapping objects
- 6) Which metrics to use?

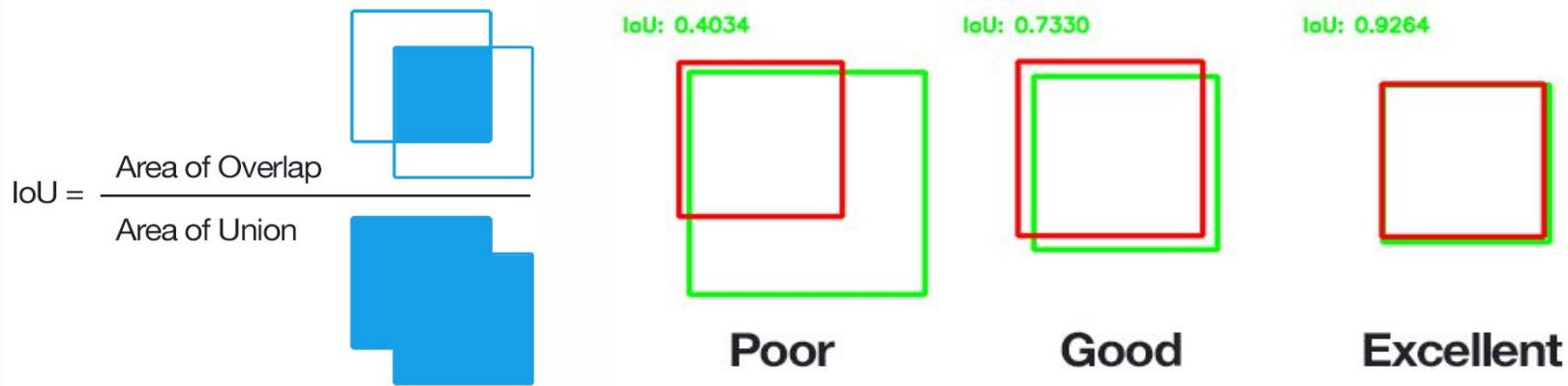


slide credit: Fei-Fei, Fergus



Metrics

- 1) How to check if object is localized accurately? - Intersection over Union (IoU).



If $\text{IoU} >$ threshold (usually 0.5): True Positive

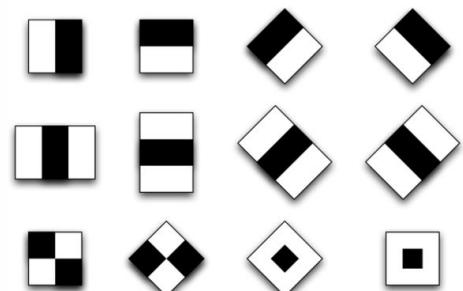
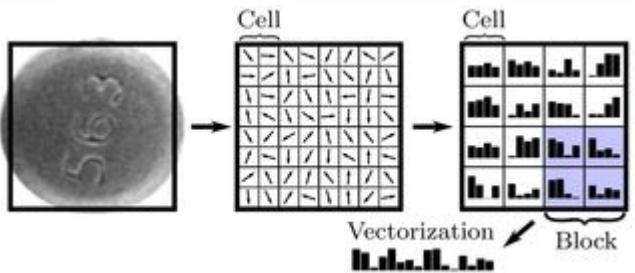
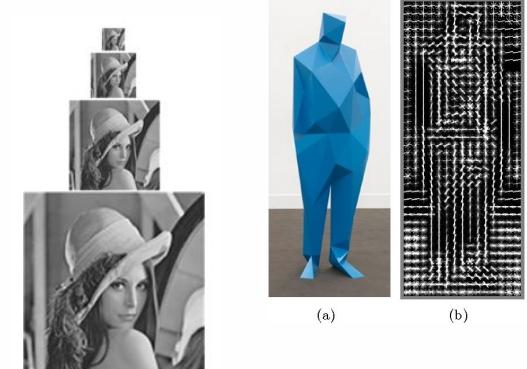
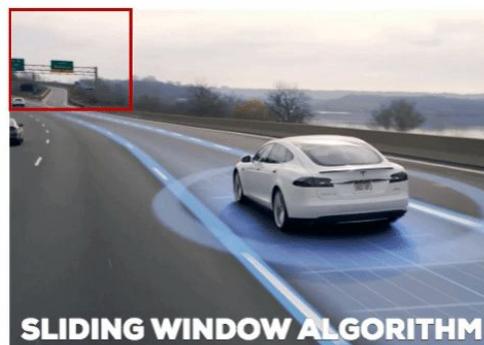
Else: False Positive.

If no prediction for some ground-truth object: False Negative.

Old school approaches

- 1) Histograms of Oriented Gradients (HOG)
- 2) Haar Cascades
- 3) SIFT

Idea: Sliding window + image pyramid



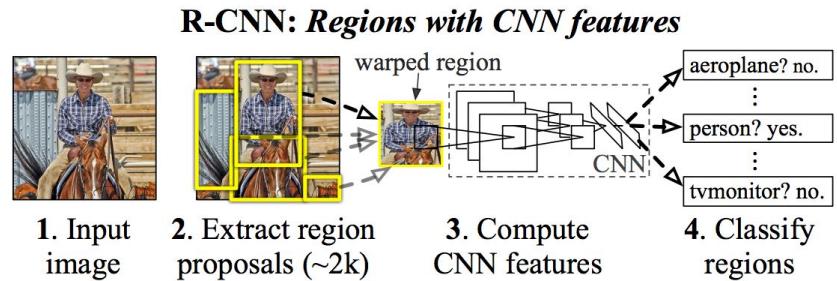
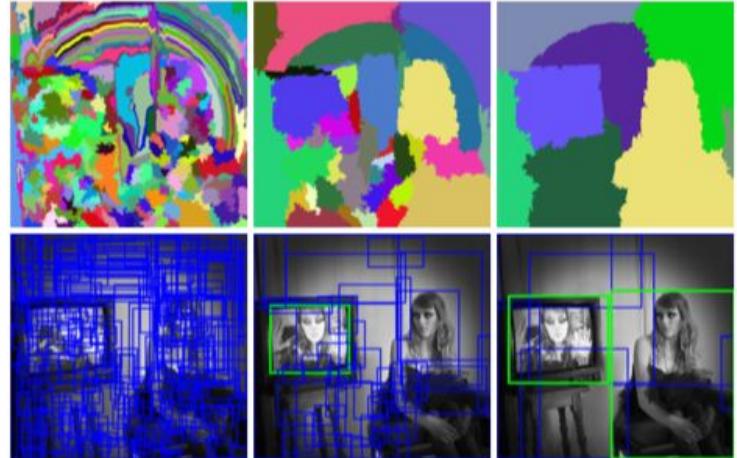


Neural Network approaches

- 1) Region-based
 - a) RCNN
 - b) Fast RCNN
 - c) Faster RCNN
- 2) One-shot:
 - a) SSD
 - b) YOLO
- 3) Cascaded Detectors:
 - a) MTCNN

RCNN (Regions with CNN features)

- 1) Generate region proposals (Selective Search)
- 2) Crop each region, feed to the CNN, trained for image classification.
- 3) Threshold detections and apply Non Maximum Suppression (NMS) to them.



Non Maximum Suppression (NMS)

Problem: many highly overlapping regions for a single object. NMS discards bounding boxes with high overlap with bounding boxes with higher detection score.

NMS algorithm:

1. Sort boxes by detection scores in descending order. Store them in list L.
2. Iterate over list L:
 - a. Pick first box in L.
 - b. Compute its IoU with all other boxes in L.
 - c. Remove boxes that have IoU with B greater than threshold.





RCNN Details

- 1) Classification CNN training:
 - a) Pre-trained CNN for image classification (authors use AlexNet on ImageNet2012)
 - b) Replace last classification layer, add “background” label
 - c) Generate dataset with object crops.
 - d) Retrain network.
- 2) Class-specific SVM:
 - a) Allows to assign multiple labels to a region at test time.
 - b) Trained on CNNs features from pre-last fully connected layer (before classification layer).



Advantages and disadvantages of RCNN

Pros:

1. First “good” object detector (53.7% mAP on PASCAL VOC 2010).

Cons:

1. Too slow: 13s/image on a GPU or 53s/image on a CPU. ~2000 region proposals. Duplication of calculations in case of overlapping region proposals (no computation sharing).
2. Warps image to some standard size, does not care about object scale/ aspect ratio.
3. Training is both computationally and memory expensive.
4. Multiple detection stages which is not very convenient.

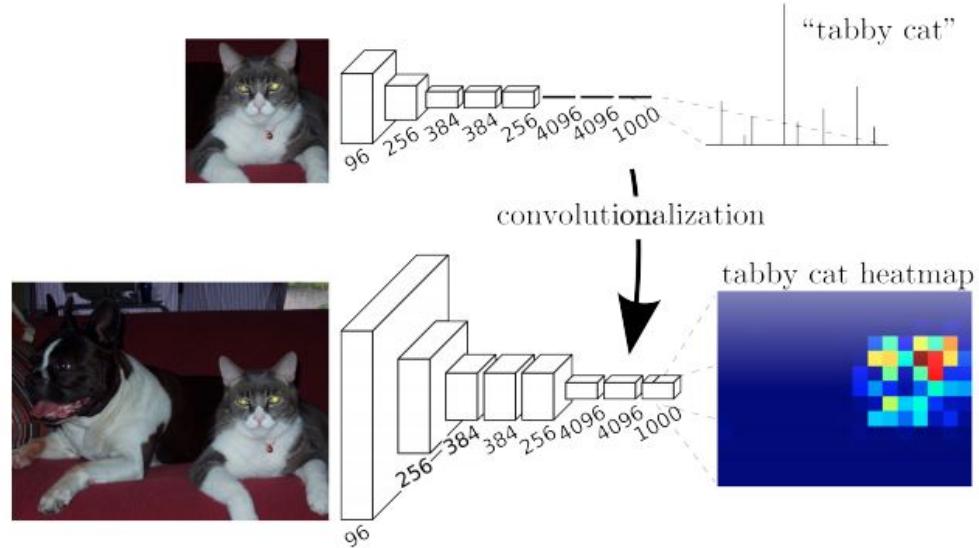
How can we speedup RCNN?

Fully convolutional network (FCN)

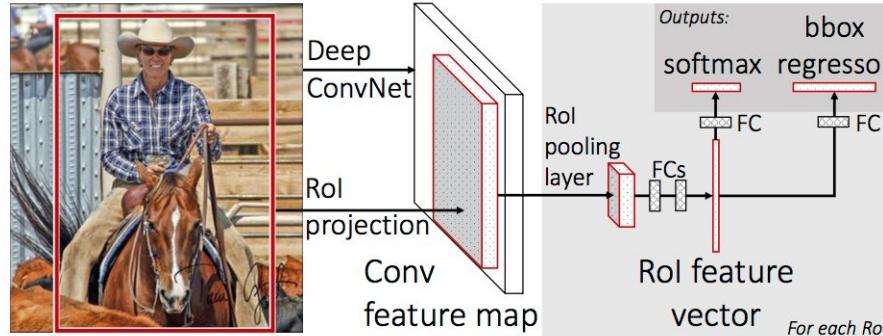
1) Convolution with fixed filter can be applied to a tensor of arbitrary height and width.
Same for max pooling.

2) Only fully connected layers do not allow to apply CNN to image of arbitrary height and width.

BTW, FC layer can be viewed as a convolutional layer with filter that covers entire input tensor.



Fast RCNN



| input | 0.88 | 0.44 | 0.14 | 0.16 | 0.37 | 0.77 | 0.96 | 0.27 |
|-------|------|------|------|------|------|------|------|------|
| 0.19 | 0.45 | 0.57 | 0.16 | 0.63 | 0.29 | 0.71 | 0.59 | 0.70 |
| 0.66 | 0.26 | 0.82 | 0.64 | 0.54 | 0.73 | 0.59 | 0.26 | |
| 0.85 | 0.34 | 0.76 | 0.84 | 0.29 | 0.75 | 0.62 | 0.25 | |
| 0.32 | 0.74 | 0.21 | 0.39 | 0.34 | 0.03 | 0.33 | 0.48 | |
| 0.20 | 0.14 | 0.16 | 0.13 | 0.73 | 0.65 | 0.96 | 0.32 | |
| 0.19 | 0.69 | 0.09 | 0.86 | 0.88 | 0.07 | 0.01 | 0.48 | |
| 0.83 | 0.24 | 0.97 | 0.04 | 0.24 | 0.35 | 0.50 | 0.91 | |

- 1) Same region proposal technique (selective search)
- 2) Feed full image to fully convolutional network, obtain feature map with size $C \times W \times H$.
- 3) Crop regions from this feature map that correspond to region proposals. Apply ROI pooling layer to them. The result is fixed-size feature map of size $C \times W' \times H'$. W' and H' are approximately equal to (W/w) and (H/h) .
- 4) Feed this fixed-size feature map to fully connected layers and predict class of the region and bounding box regression. Bounding box regression is class-specific and predicts offset for region proposal center and warping coefficients for its height and width.
- 5) Adjust the region proposal with bounding box regression, apply nms on obtained bounding boxes.



Fast RCNN Details

- 1) RoI Pooling layer is differentiable, so Fast RCNN can be trained end-to-end.
- 2) Both loss functions for classification and bounding box regression are optimized jointly.
- 3) Convolutions are applied to input image only once, but fully connected layers are applied to every pooled region proposal => authors use truncated SVD on fully connected layers weight matrices => 30% speedup.
- 4) VGG16 instead of AlexNet.

Main disadvantage of Fast RCNN: region proposal technique.



Fast RCNN Details

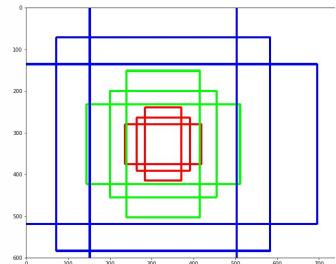
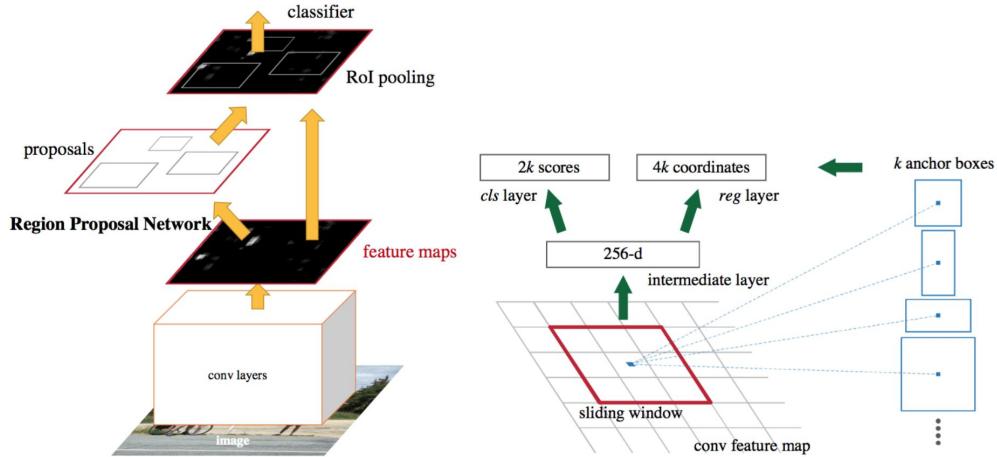
- 1) RoI Pooling layer is differentiable, so Fast RCNN can be trained end-to-end.
- 2) Both loss functions for classification and bounding box regression are optimized jointly.
- 3) Convolutions are applied to input image only once, but fully connected layers are applied to every pooled region proposal => authors use truncated SVD on fully connected layers weight matrices => 30% speedup.
- 4) VGG16 instead of AlexNet.

Main disadvantage of Fast RCNN: region proposal technique.

Let's train a CNN that will give us better region proposals!

Faster RCNN

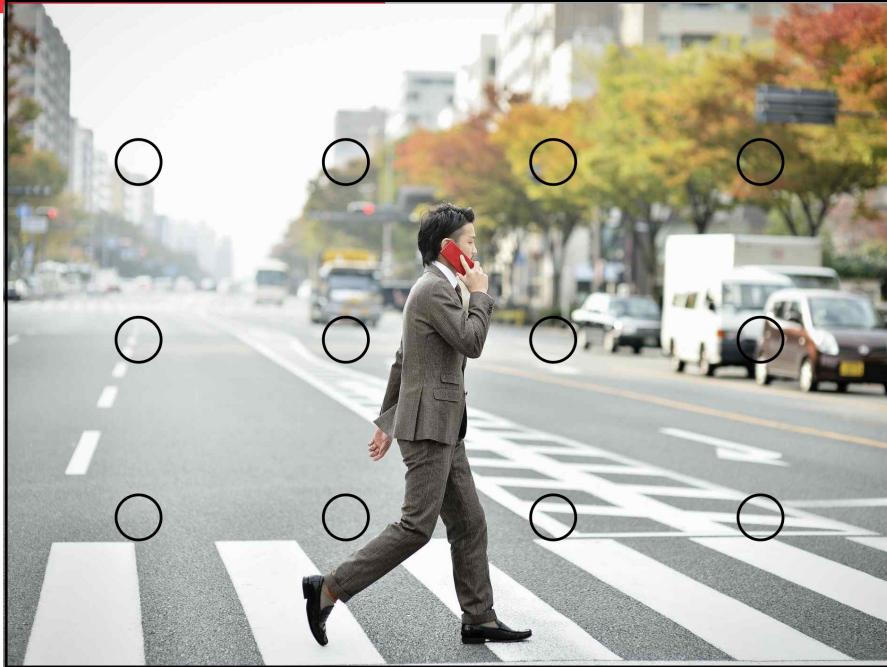
- 1) Additional network for object proposal generation on top of base fully convolutional part - Region Proposal Network (RPN).
- 2) Concept of Anchor Boxes with different size and aspect ratios.
- 3) RPN predicts “objectness”.
- 4) NMS is applied to generated proposals. Corresponding regions of conv feature map are passed to RoI pooling layer and the result is passed to Fast RCNN head.



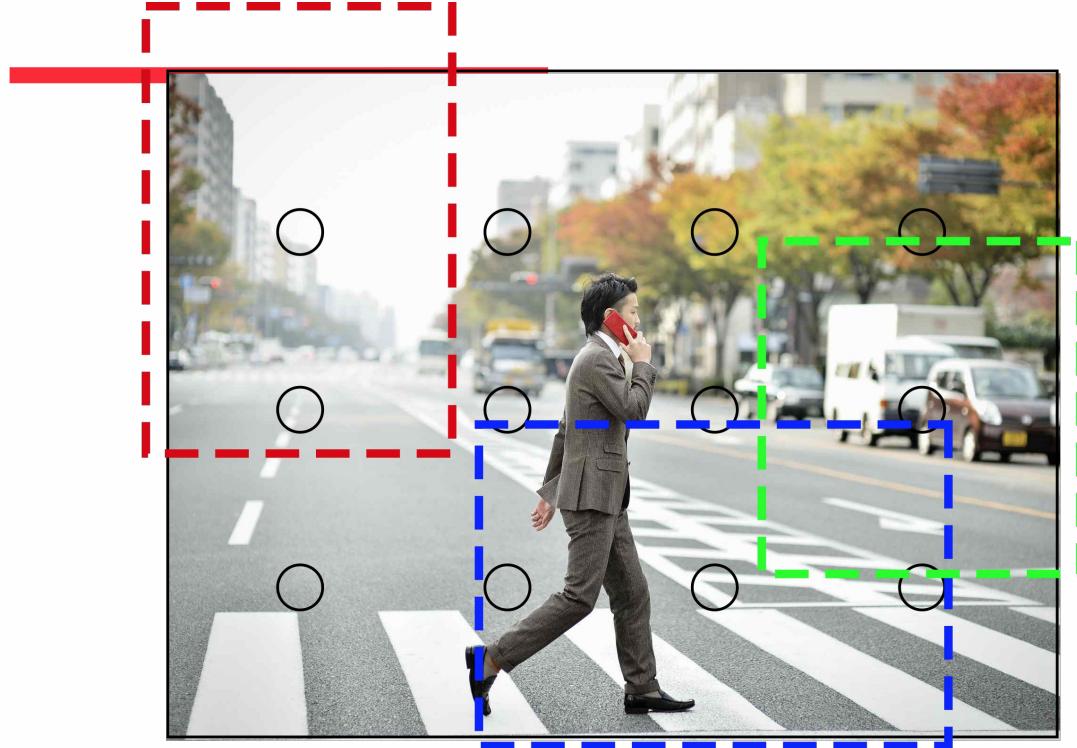
Anchor Boxes



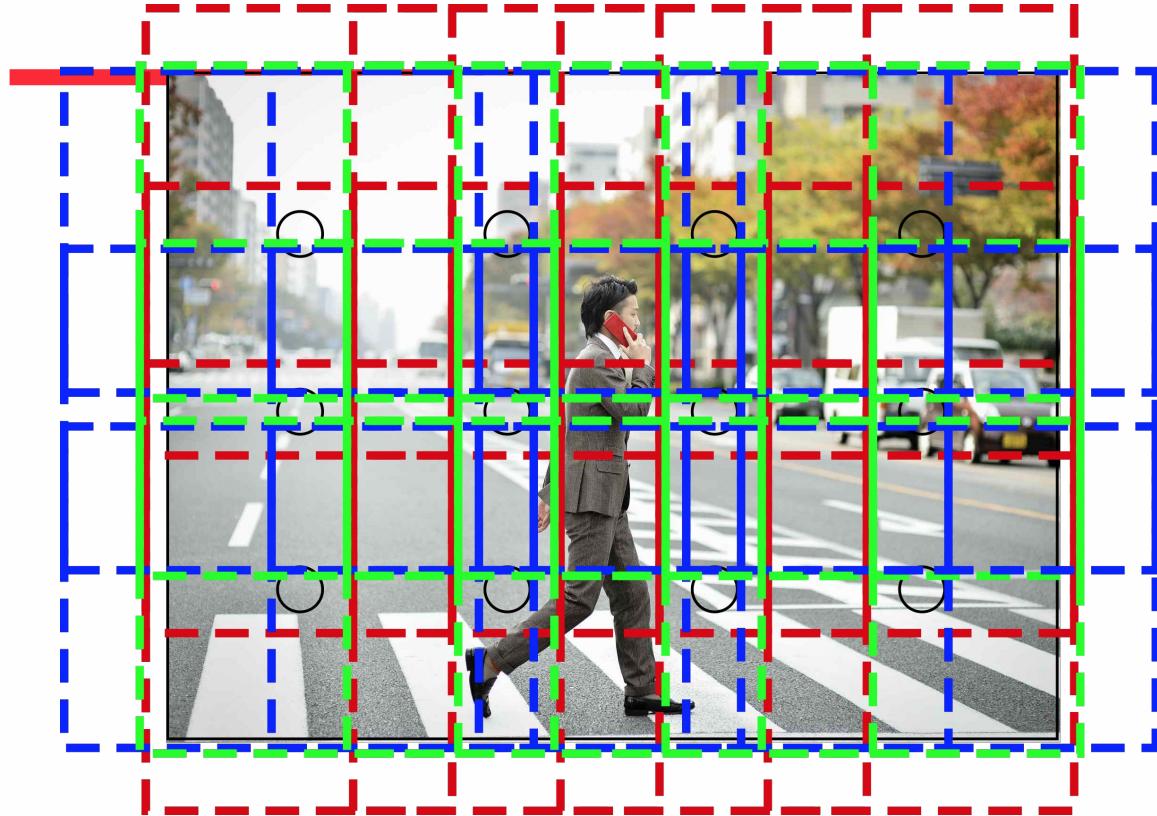
Anchor Boxes



Anchor Boxes

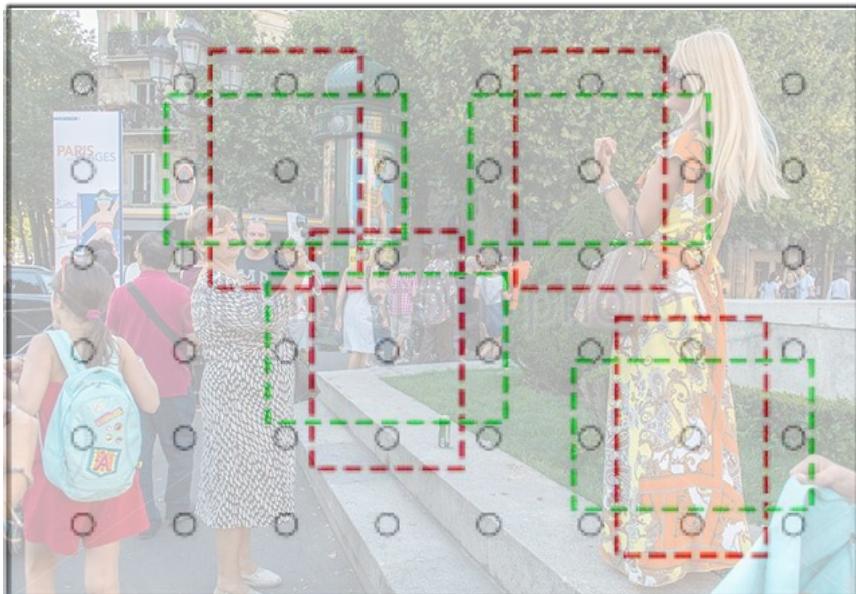


Anchor Boxes

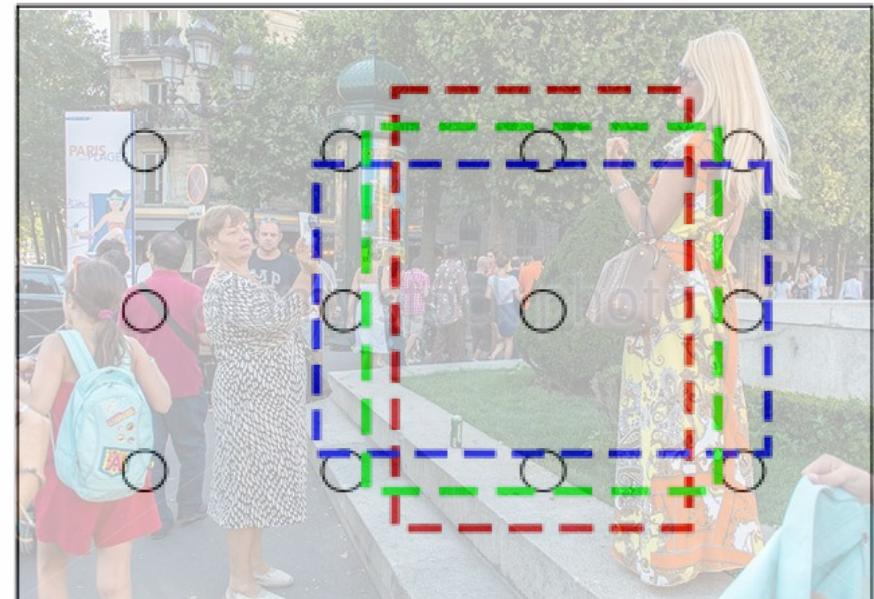




Different Scales



Smaller scale

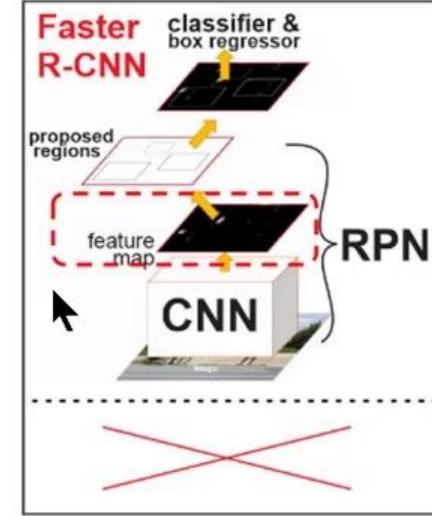
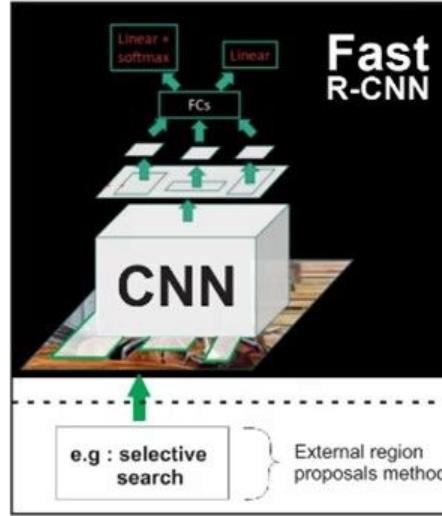
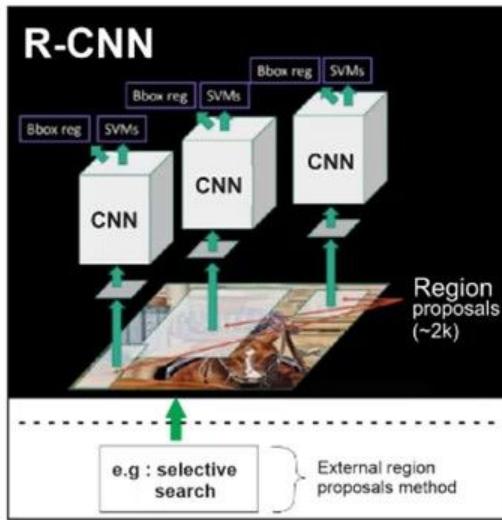


Bigger scale



Faster RCNN Details

- 1) RPN can be viewed as “attention” mechanism of the network.
- 2) Classification and Regression are predicted in RPN in superpixel-wise manner.
- 3) Still used. Mask RCNN is based on Faster RCNN with resnet as a backbone.
- 4) R-FCN is a further development of Faster RCNN.



| | R-CNN | Fast R-CNN | Faster R-CNN |
|---------------------|--------------|-------------------|---------------------|
| Test time per image | 50 seconds | 2 seconds | 0.2 seconds |
| Speed-up | 1x | 25x | 250x |
| mAP (VOC 2007) | 66.0% | 66.9% | 66.9% |

* Standford lecture notes on CNN by Fei Fei Li and Andrej Karpathy



Cascaded detectors

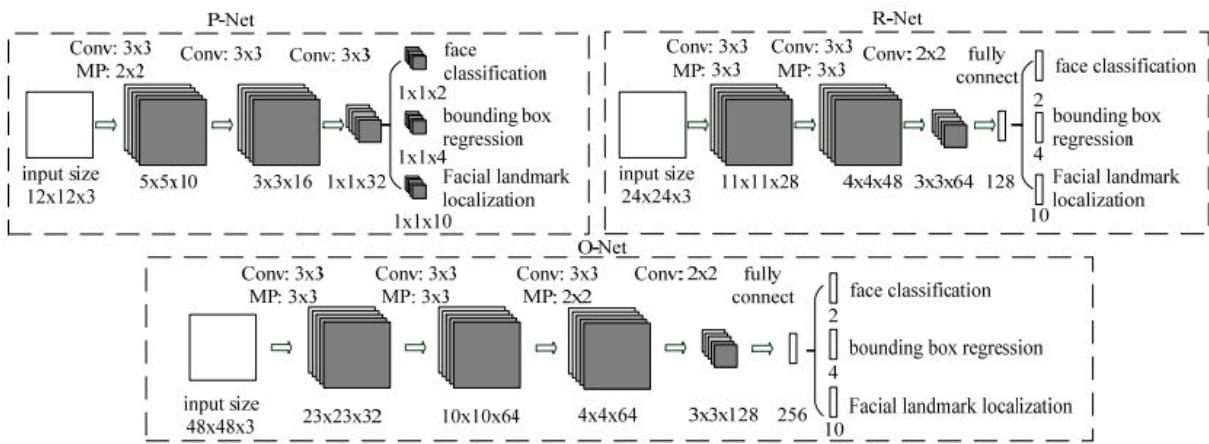
Idea: Process image with many simple and computationally cheap classifiers, effectively discarding most of empty regions on the image. Regions that have not been discarded are processed by more complex classifiers.

Examples:

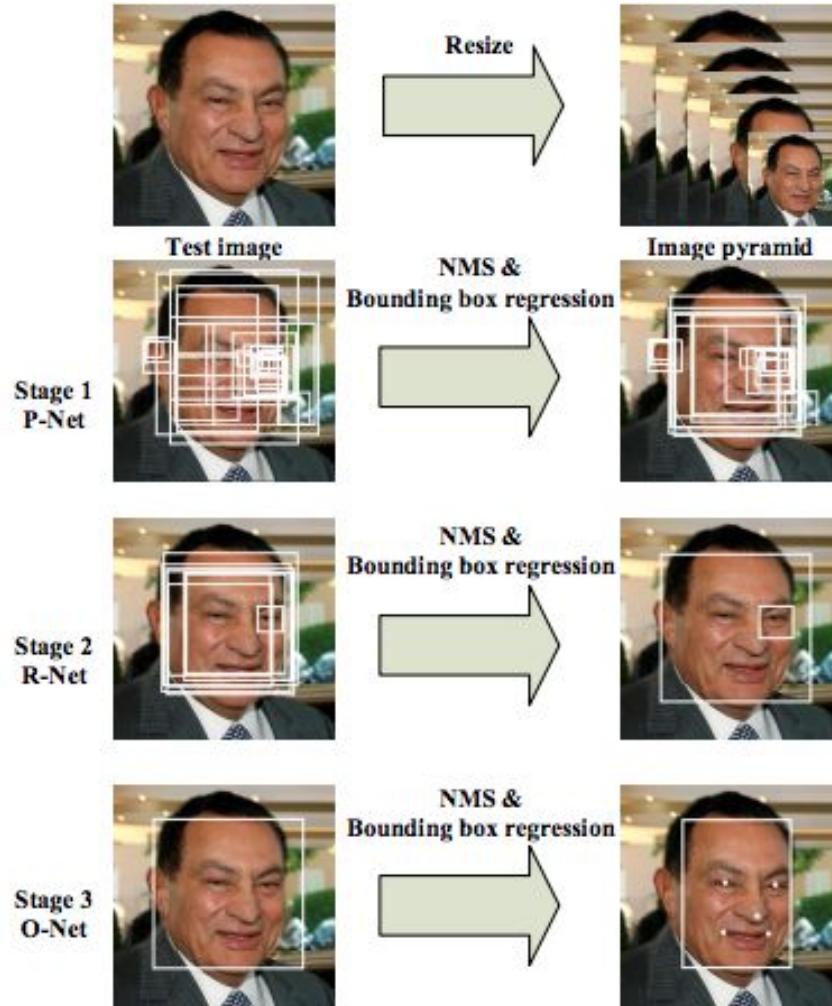
- 1) Viola-Jones (Classic)
- 2) MTCNN

MTCNN

- 1) 3 stages:
 - a) PNet - FCN, generates Proposals.
 - b) RNet, Refines proposals.
 - c) ONet, further refines detections and Outputs facial landmarks.



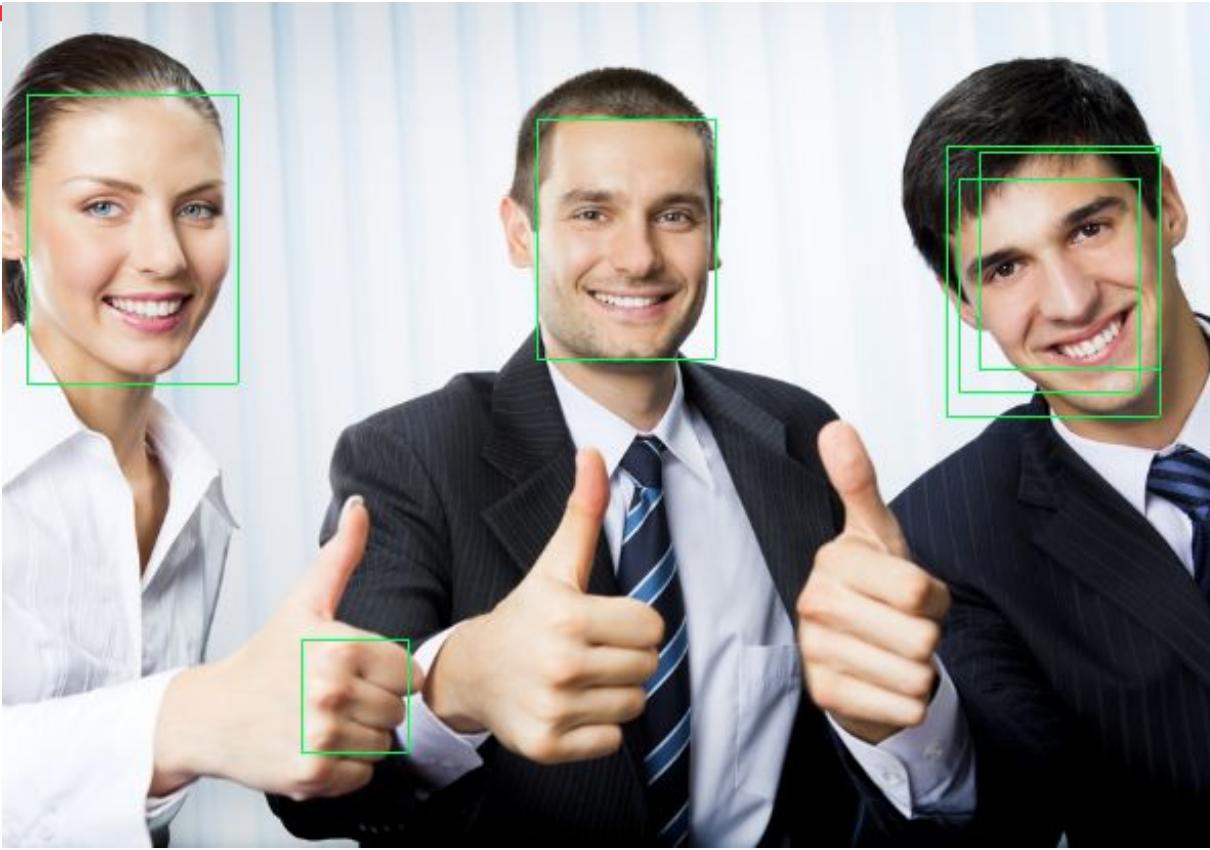
MTCNN Framework



PNet result



RNet result



ONet result





MTCNN details

- 1) Super fast. About 10-20 ms. (GPU) on a 600x600 image for full detection pipeline.
- 2) A good choice for most Facial Recognition algorithms.
- 3) Quality is slightly worse than most others “more complex” detection architectures.



MTCNN training

- 1) Crop image patches with $\text{IoU} > 0.6$ with ground-truth bboxes, patches with $\text{IoU} > 0.4$ as “part” examples, and some others as “negative” examples.
- 2) Train PNet. Use positive and negative examples for classification, positive and “part” examples for bbox regression.
- 3) Run PNet on the whole dataset, again, collect positive, negative and “part” samples from PNet detections.
- 4) Train RNet in the same manner.
- 5) Again, run PNet and RNet on the whole dataset, generate training set for ONet.
- 6) Train ONet to classify proposals, regress bounding boxes and predict facial landmarks.



More good architectures

- 1) R-FCN
 - 2) FCOS
 - 3) Cascade-RCNN (SOTA)
-
- + MaskRCNN

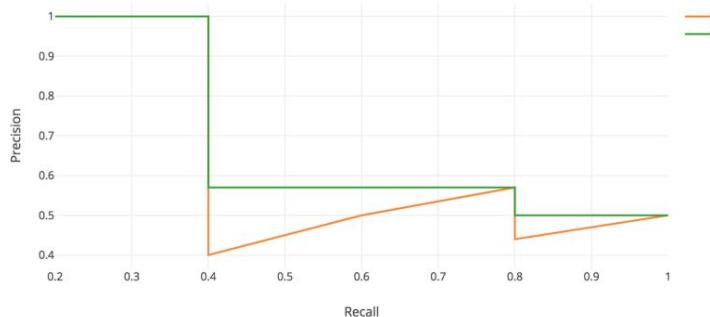
Metrics

Main evaluation metric - mean Average Precision (mAP). Should not be confused with mAP@K metric in ranking task.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$AP = \frac{1}{11} \times (AP_r(0) + AP_r(0.1) + \dots + AP_r(1.0))$$



Vary threshold \Rightarrow obtain different Precision and Recall.

Orange line - PR curve,

Green line - maximum Precision we can get for specific Recall level.

Average Precision for specific class: average maximum Precision we can get for Recall values in [0.0, 0.1, ..., 0.9, 1.0]

mAP metric is mean of APs for all classes in dataset.

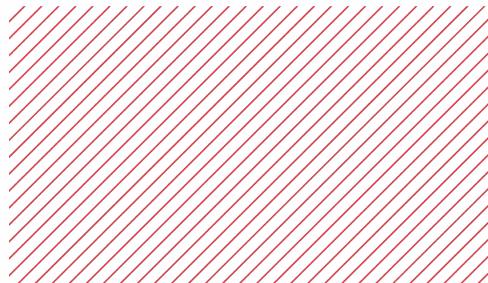
BTW, we can compute multiple mAPs for any IoU threshold. If IoU threshold = 0.75 \Rightarrow mAP metric for this threshold is abbreviated as mAP@.75

академия
больших
данных



mail.ru
group

Лекция начнется в 19:00



Single Shot Approaches



Single shot approaches

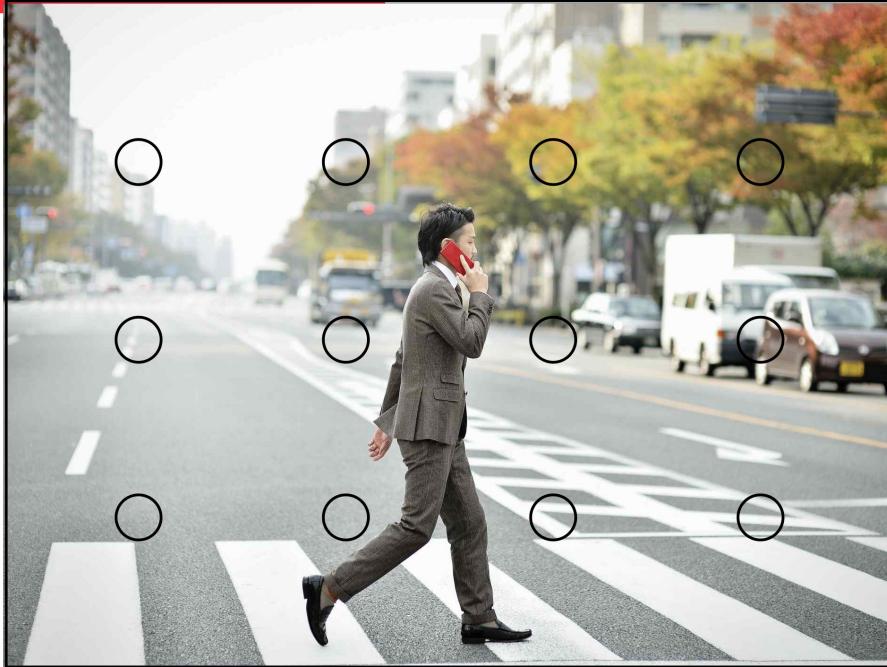
Main idea: No region proposals, no RoI pooling. One network to predict classification and regression scores for each possible object location (anchor-boxes).

BTW, should not be confused with one-shot learning.

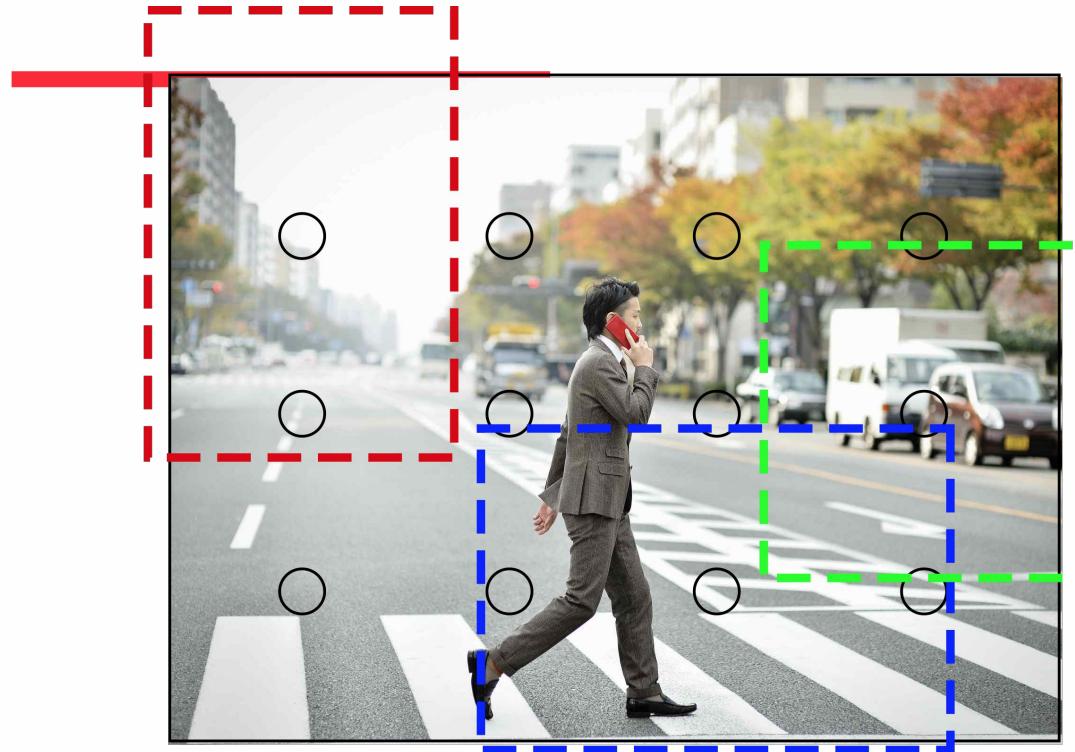
Reminder: Anchor Boxes



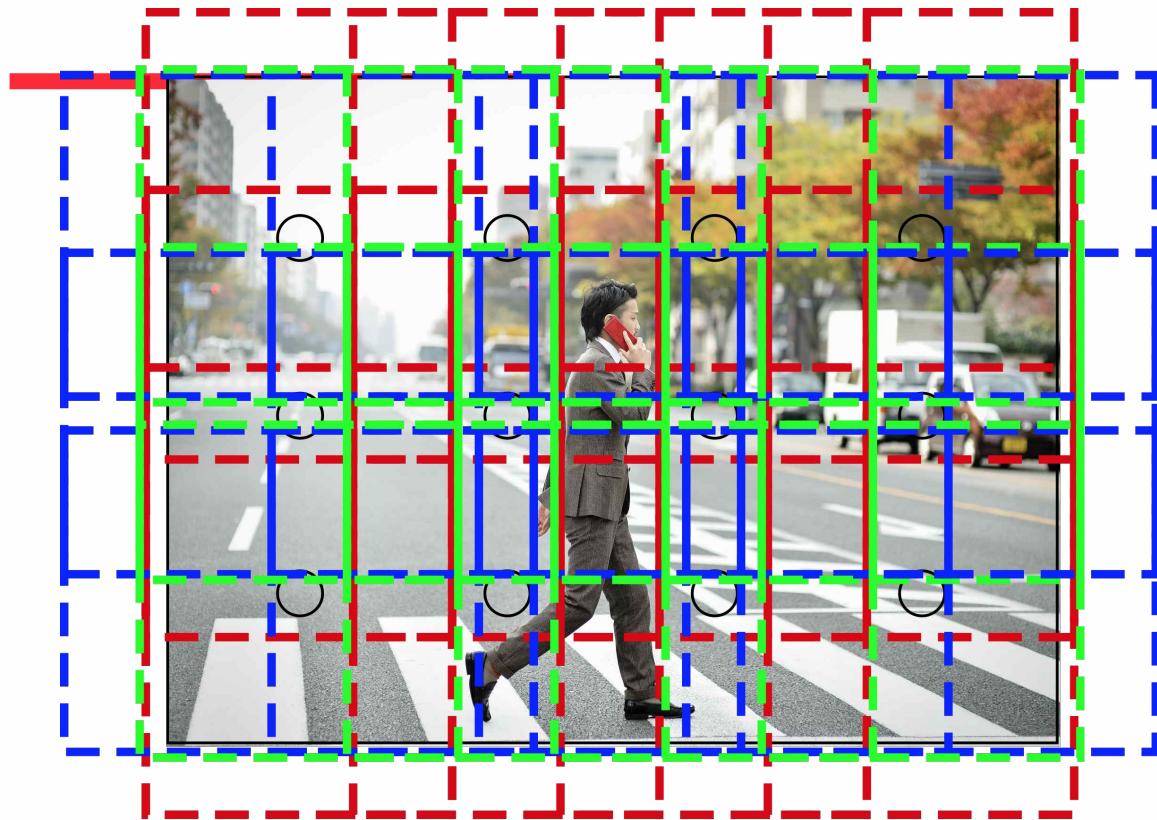
Reminder: Anchor Boxes



Reminder: Anchor Boxes

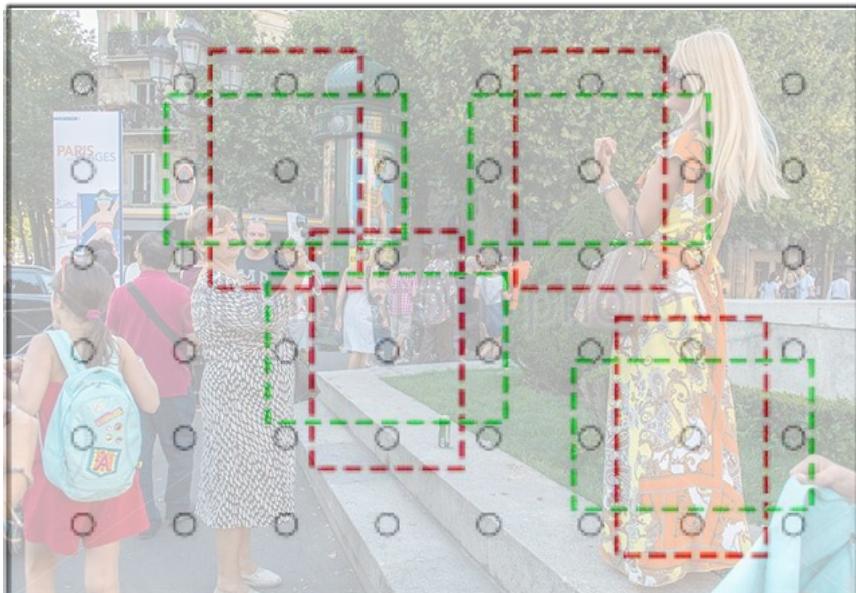


Reminder: Anchor Boxes

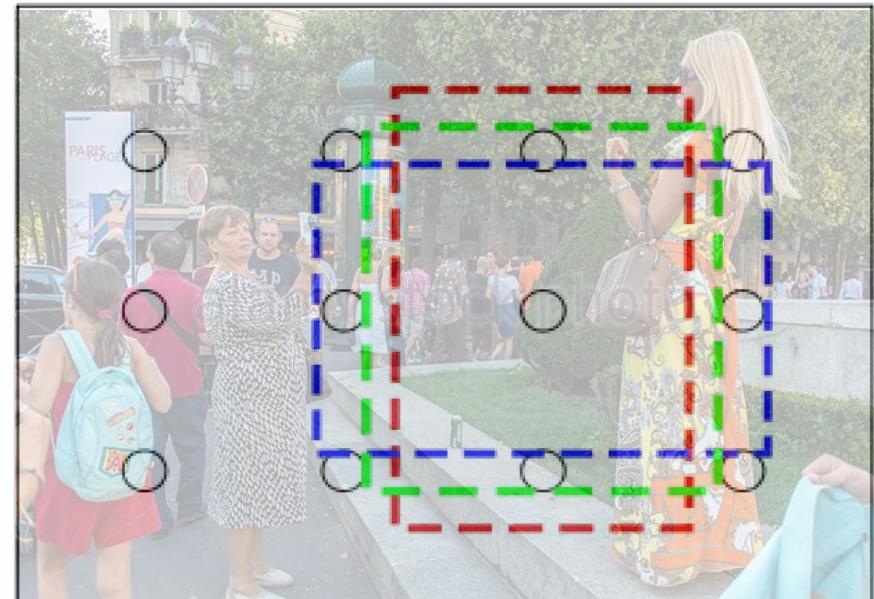




Reminder: Anchor Boxes



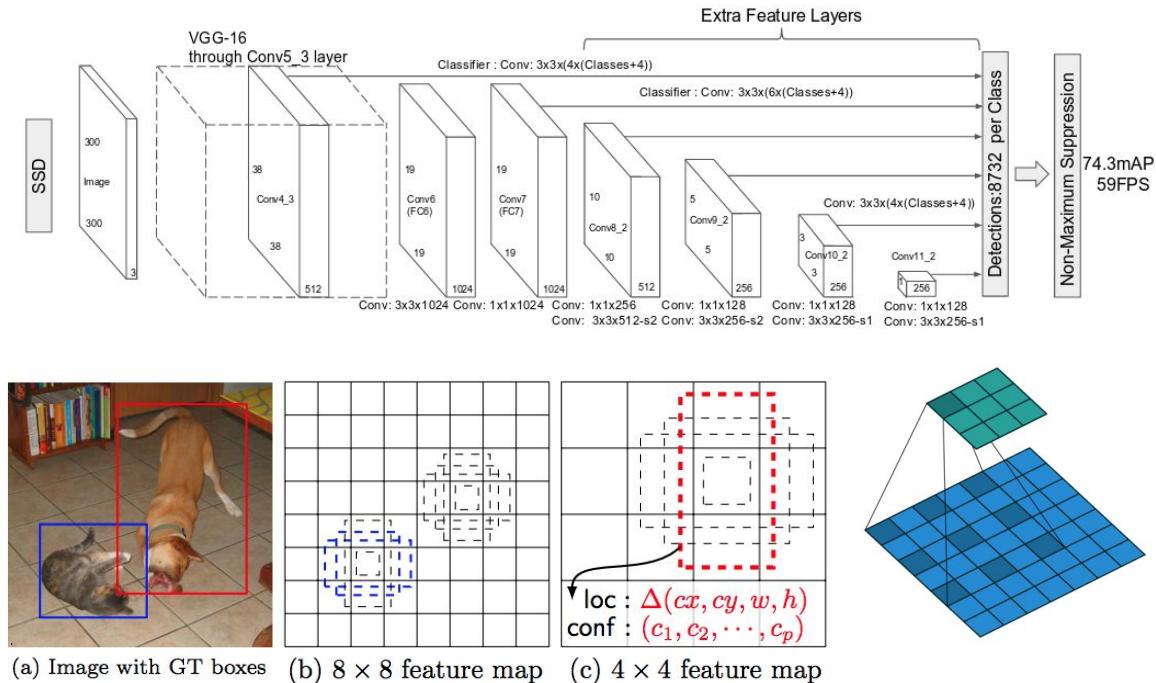
Smaller scale



Bigger scale

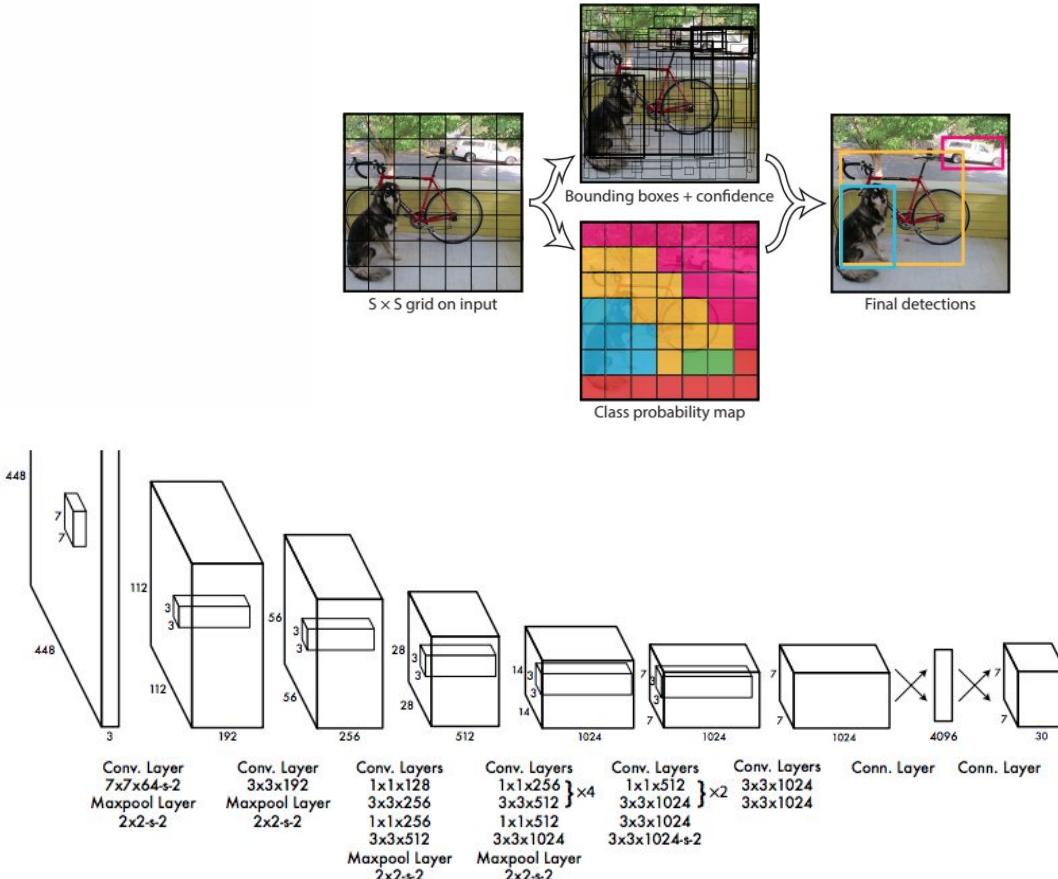
1) Single Shot Detector (SSD)

- 1) Prior boxes concept.
- 2) Fixed size input image processed by VGG16 up to conv5_3. Result is passed to classifier.
- 3) Dilated convolution with dilation=6 to conv5_3 to greatly increase receptive field.
- 4) Apply extra layers to predict bigger boxes.
- 5) Fast (~59 fps for SSD300 and 16 fps for SSD512)

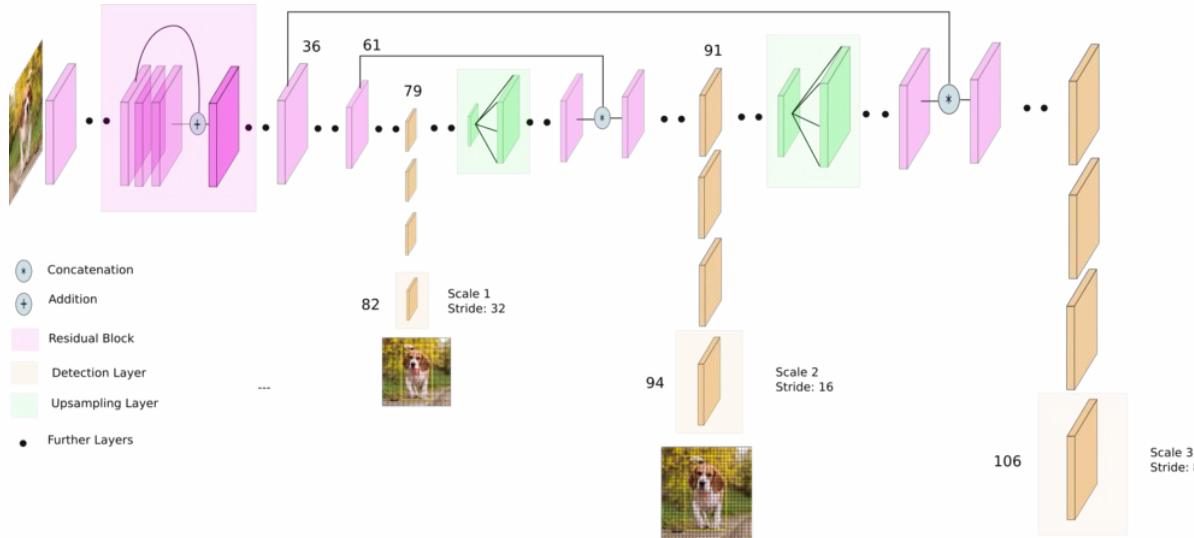


YOLO (You Only Look Once)

- 1) Tensor of size $S \times S \times (B * 5 + C)$. B is amount of anchor boxes for each cell. C is amount of classes. ($B=2$, $S=7$, $C=20$)
- 2) Realtime performance
- 3) Crutches and “Darknet” framework
- 4) YOLO v2 (a.k.a. YOLO9000): more classes.
- 5) YOLO v3: log. reg. for each class. Helps with overlapping objects.
Multiscale anchor boxes.

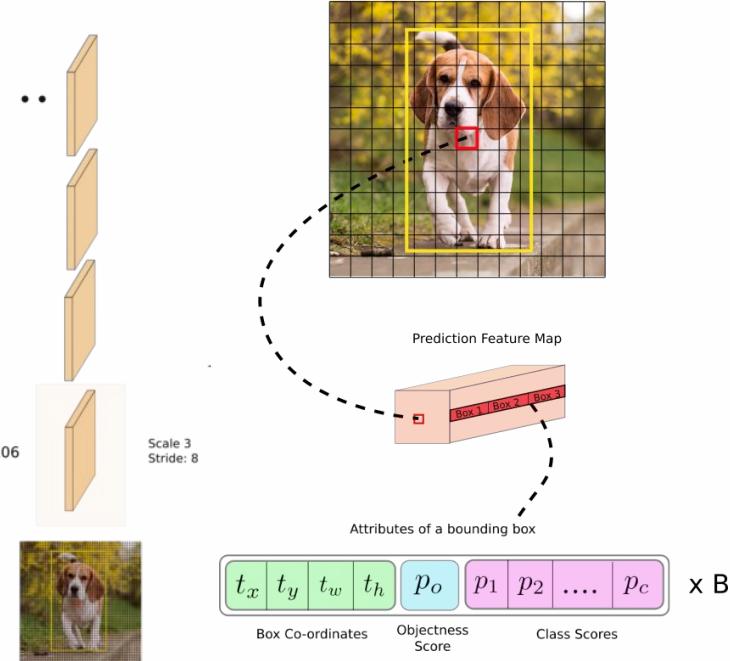


YOLO v3



YOLO v3 network Architecture

Image Grid. The Red Grid is responsible for detecting the dog





RetinaNet

- 1) Backbone - ResNet
- 2) Feature Pyramid Network (FPN)
- 3) FocalLoss against class disbalance



Focal Loss

Problem: class disbalance

Cross Entropy (CE): Focal Loss
(FL): p_t - predicted probability
of g.t. class:

$$\text{CE}(p, y) = \text{CE}(p_t) = -\log(p_t).$$

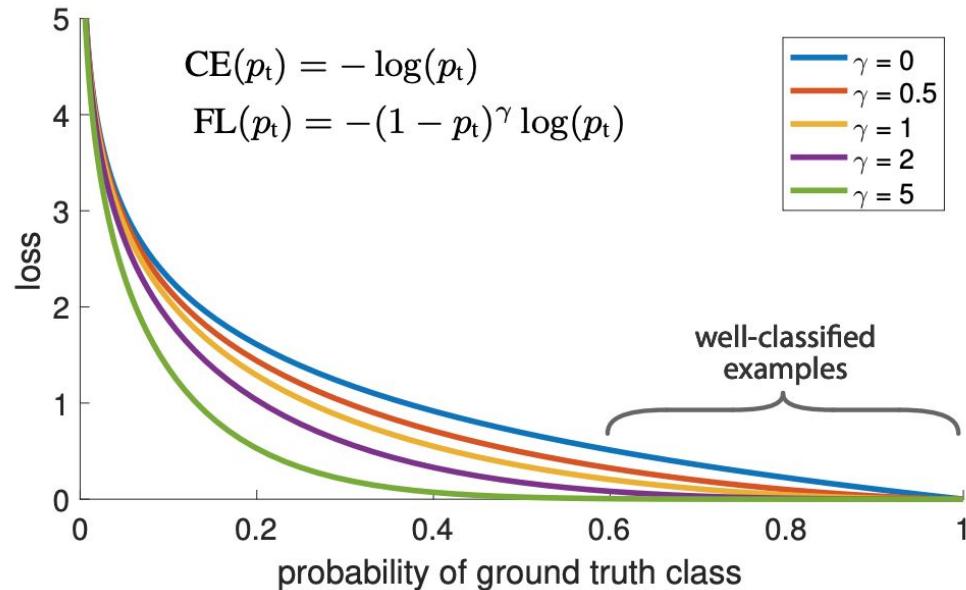
$$\text{FL}(p_t) = \underline{-(1 - p_t)^\gamma} \log(p_t)$$

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise,} \end{cases}$$

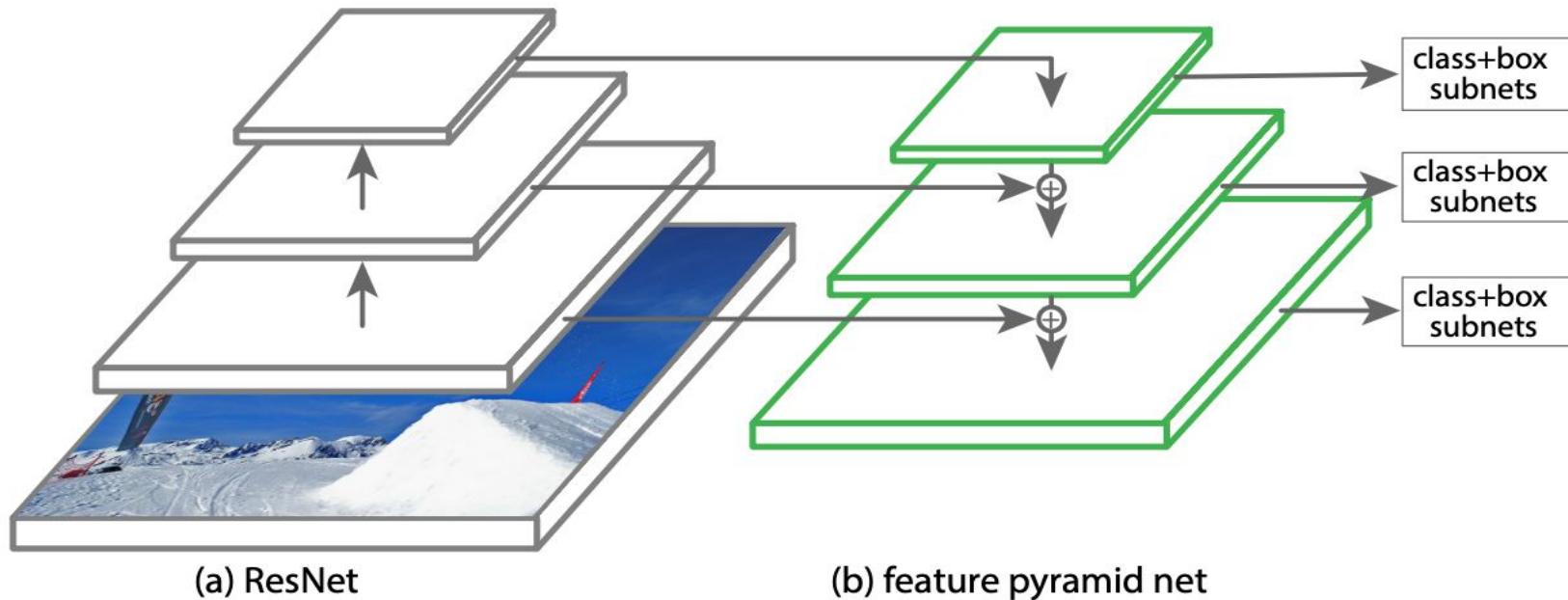
Focal Loss

Well classified examples =>
smaller contribution

Analogue of Online Hard
Example Mining

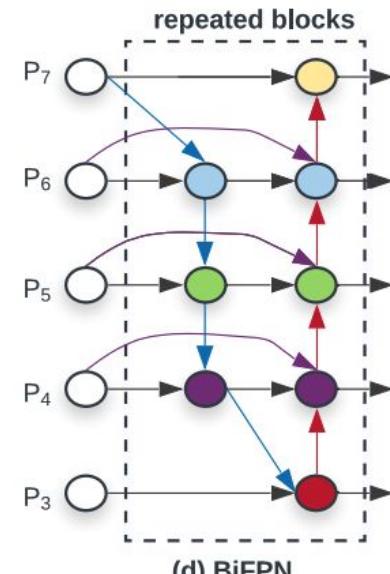
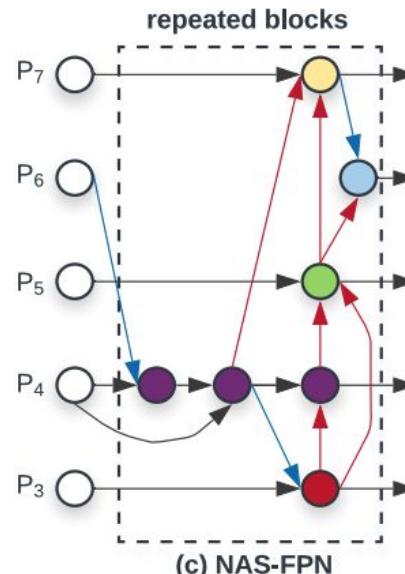
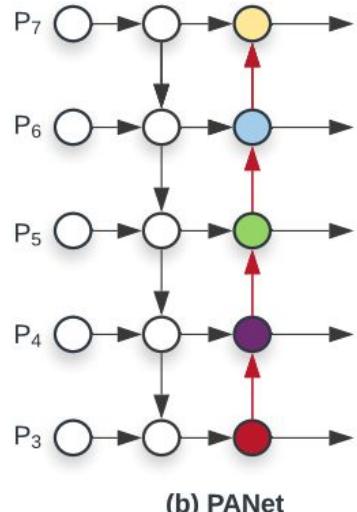
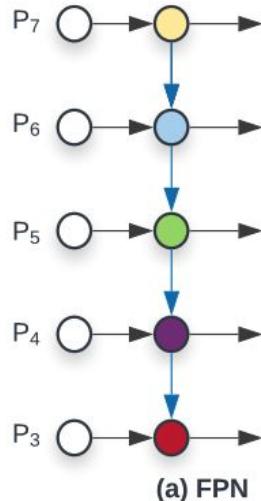


Feature Pyramid Network (FPN)



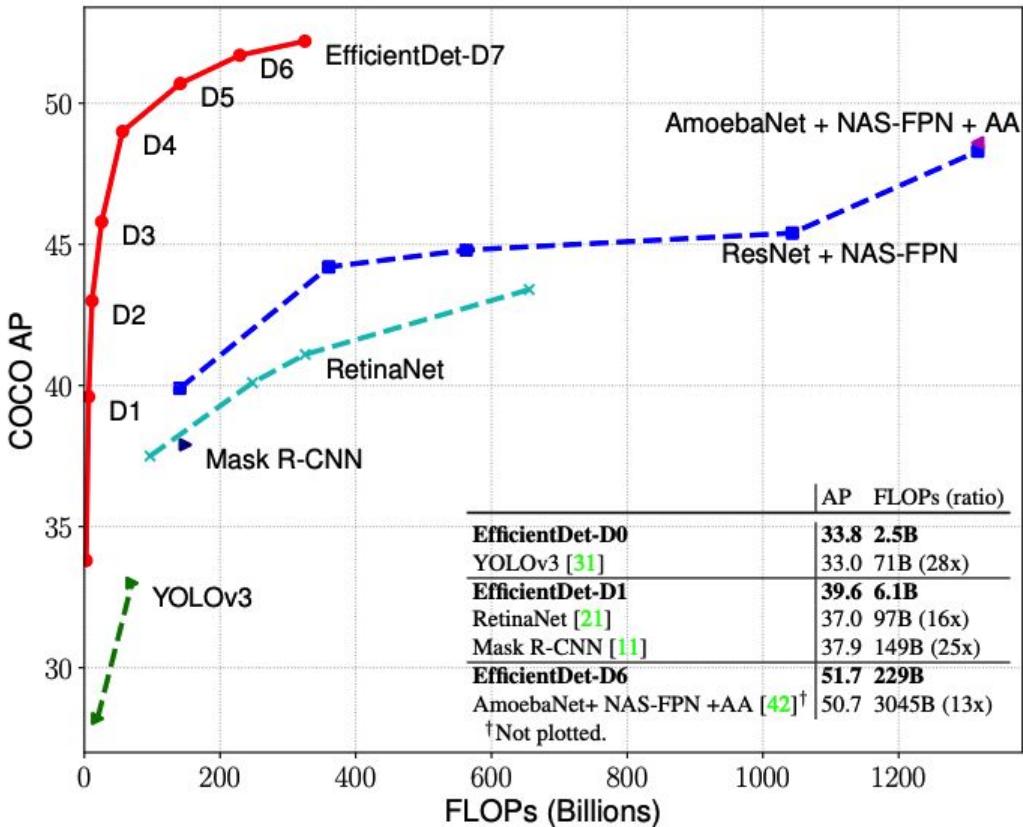
Higher level features for smaller scales

Cooler Feature Pyramid Network (FPN)



EfficientDet

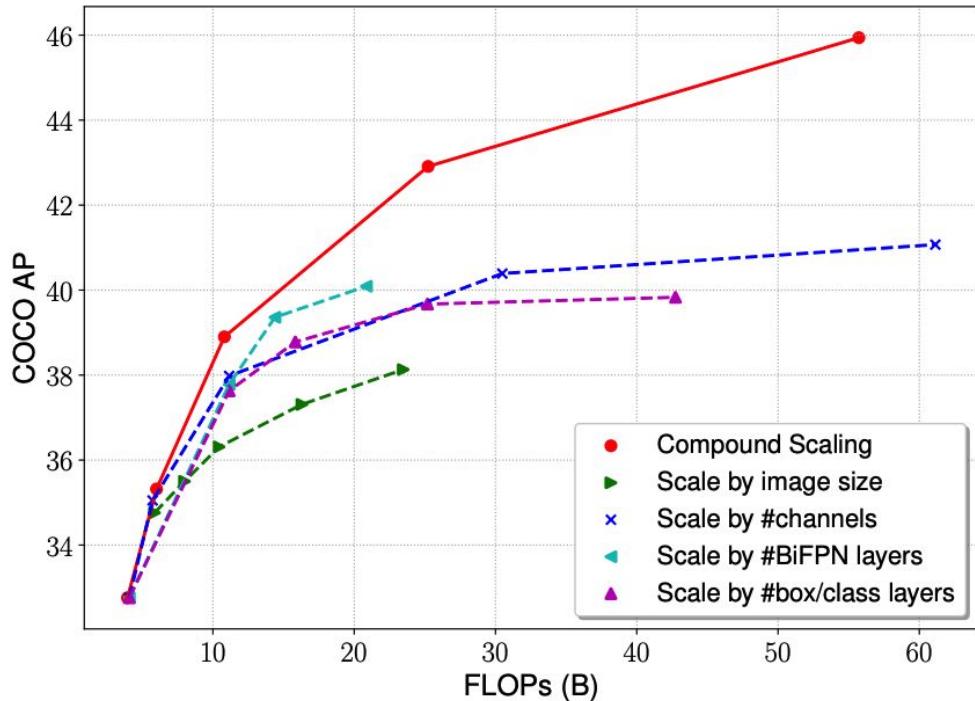
- 1) Backbone: EfficientNet
- 2) Bi-FPN



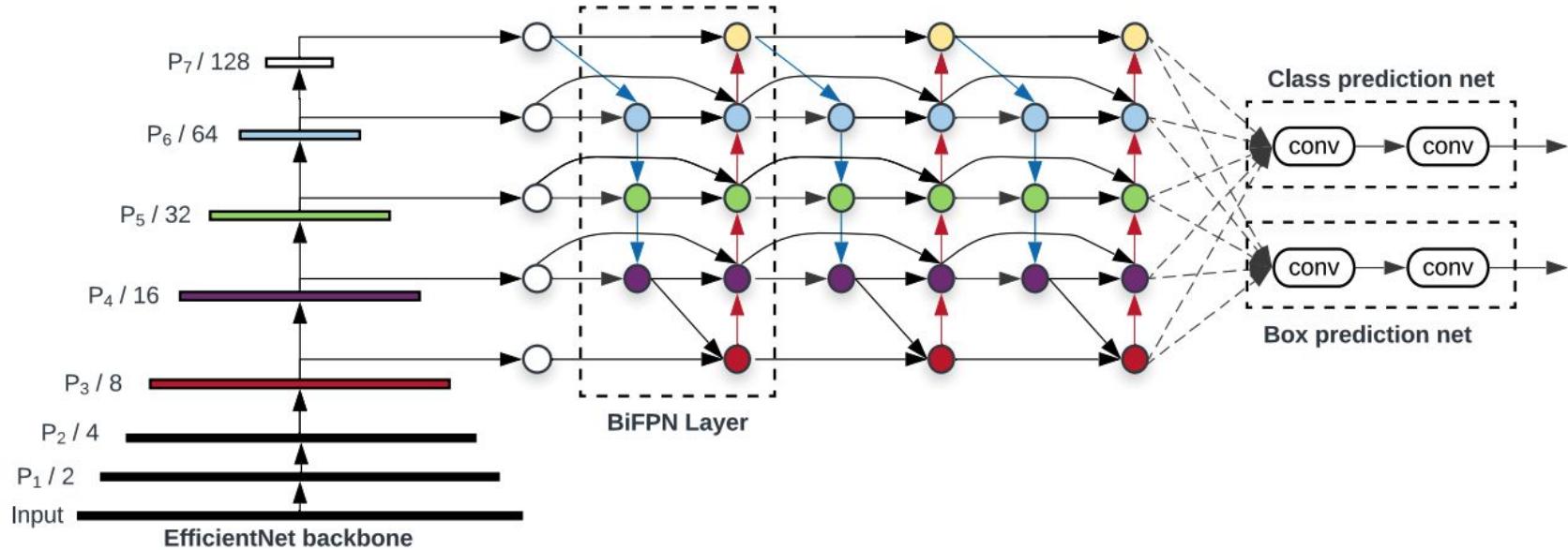
EfficientNet

EfficientNet:

- 1) Neural Architecture Search (NAS)
- 2) Compound Scaling

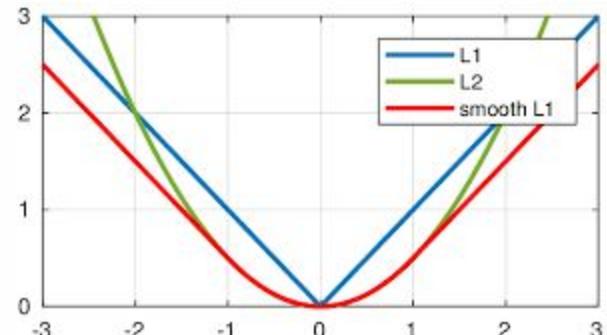


EfficientDet



Tips to boost detection quality

1. More augmentation:
 - 1.1. Classics (mirror flips, hue/gamma modification, etc.)
 - 1.2. Anything that would change the shape, size and location of bounding boxes - rotations, scaling, random cropping, jittering.
2. Better backbone architecture.
3. SoftNMS
4. Smooth-L1 loss for bbox regression
5. IoU Loss, Repulsion Loss



Questions!