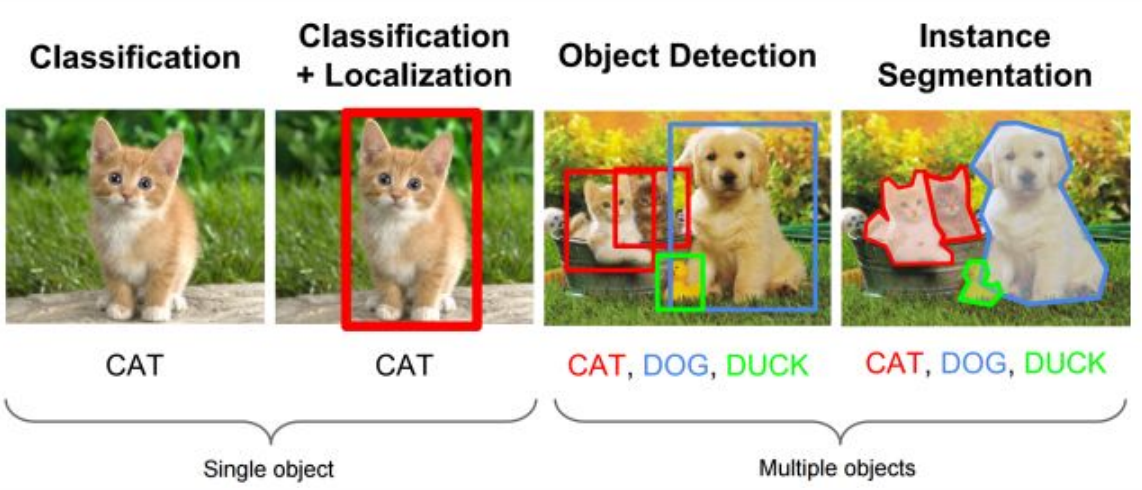
академия больших данных

mail.ru group

Neural Networks for detection

Boris Lestsov
Mail.Ru Group

Detection task

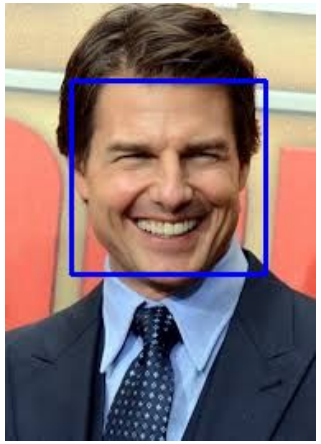


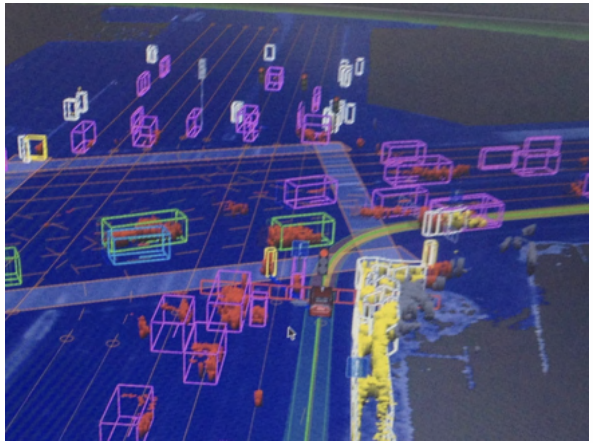
Differences with image classification task:

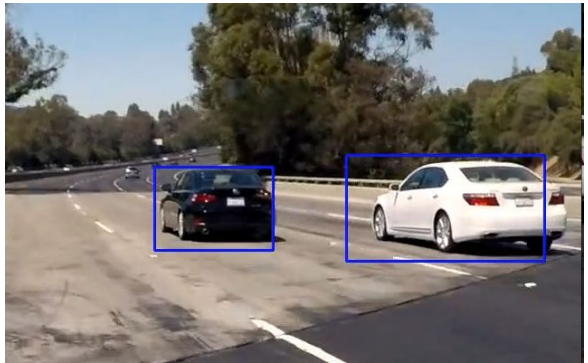
1) Image may contain objects of multiple classes or may not contain any.
2) Image may contain multiple objects of a single class.
3) Need to localize object on the image by predicting it's Bounding Box (bbox).

Applications

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





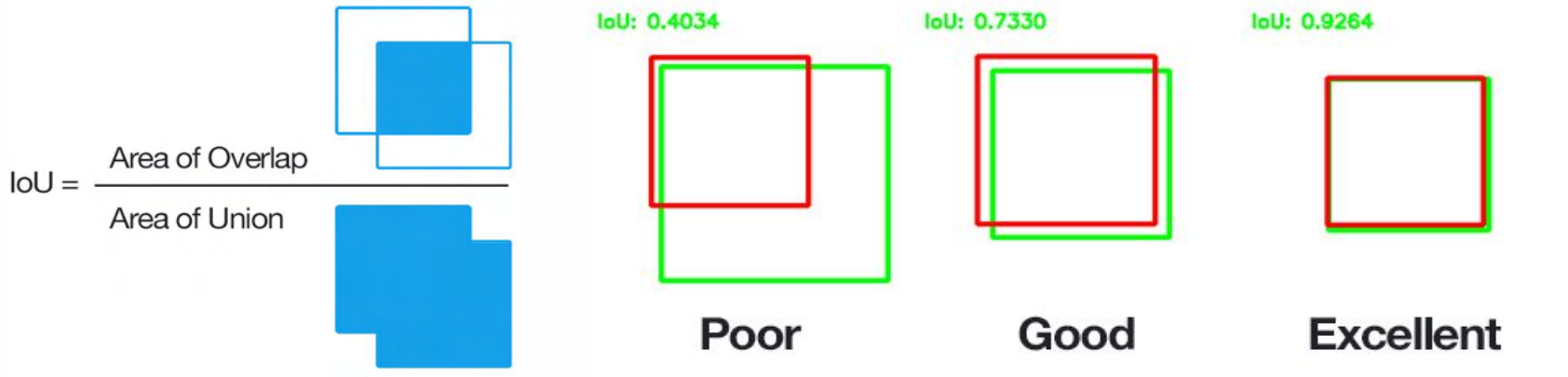


Difficulties

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

Metrics

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



If IoU > threshold (usually 0.5): **True Positive**

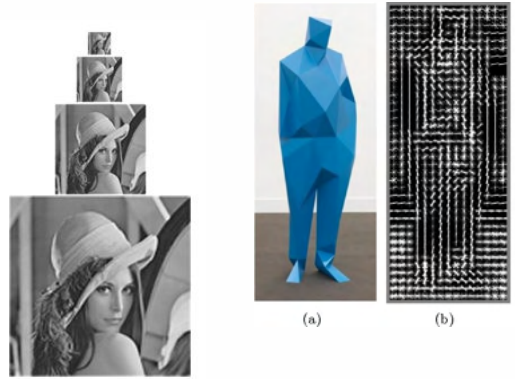
Else: **False Positive.**

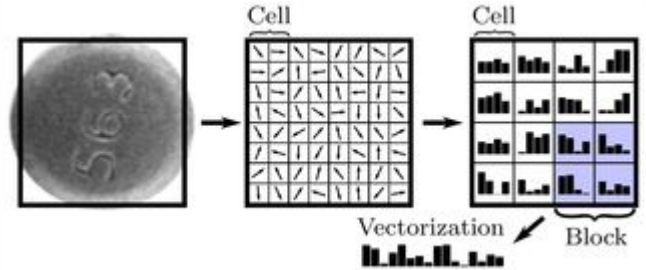
If no prediction for some ground-truth object: **False Negative.**

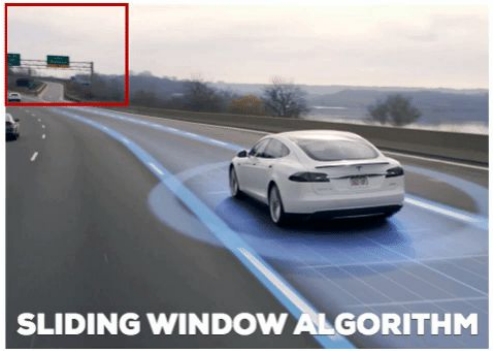
Old school approaches

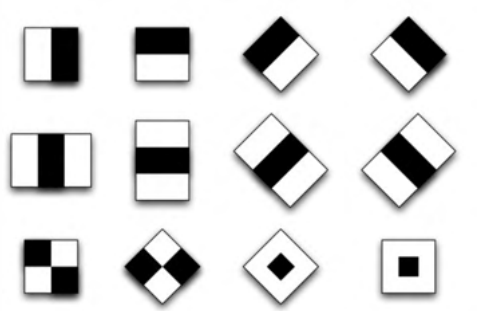
1) Histograms of Oriented Gradients (HOG)
2) Haar Cascades
3) SIFT

Idea: Sliding window + image pyramid







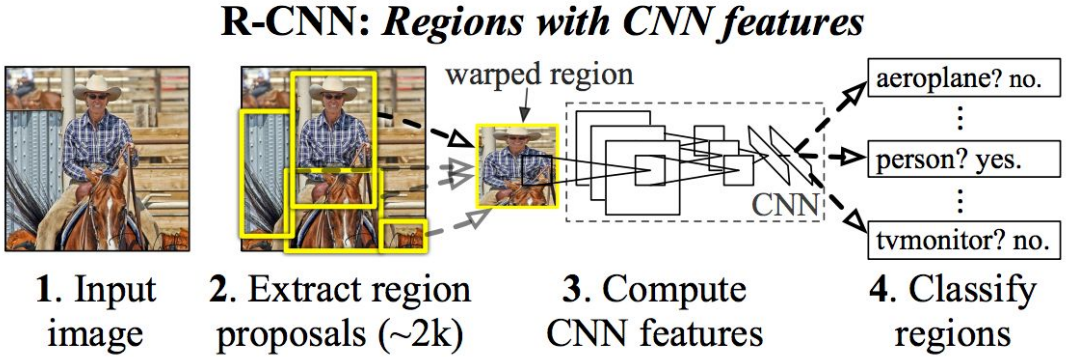


Neural Network approaches

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 trained on ImageNet)
   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?

Question:

What prevents us from applying CNN to image of arbitrary size?

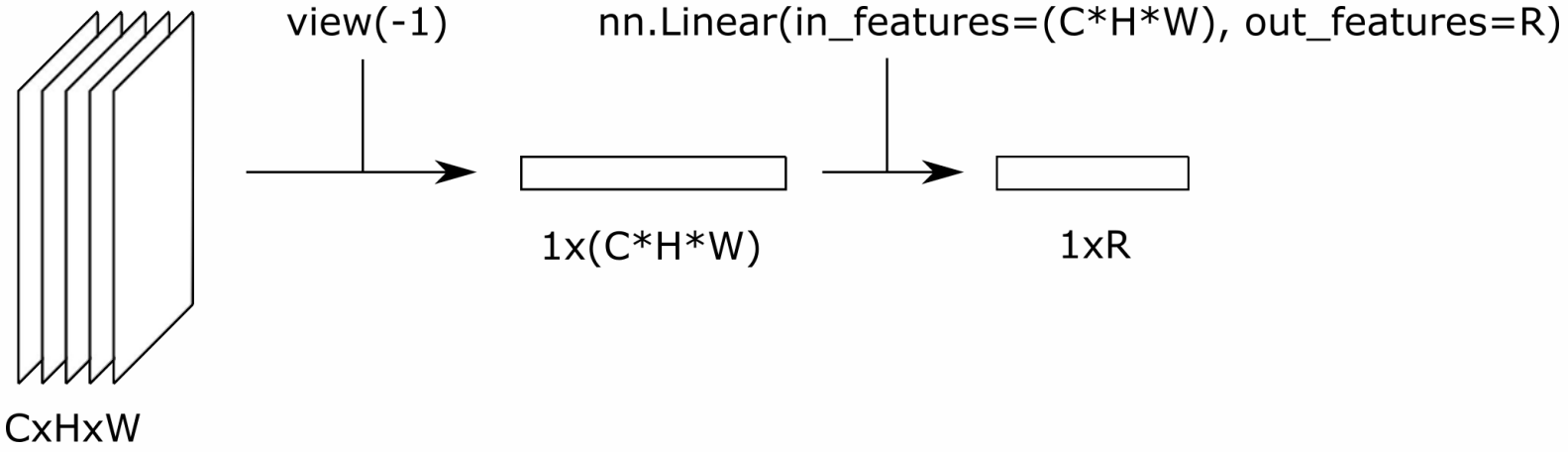
Question:

What prevents us from applying CNN to image of arbitrary size?

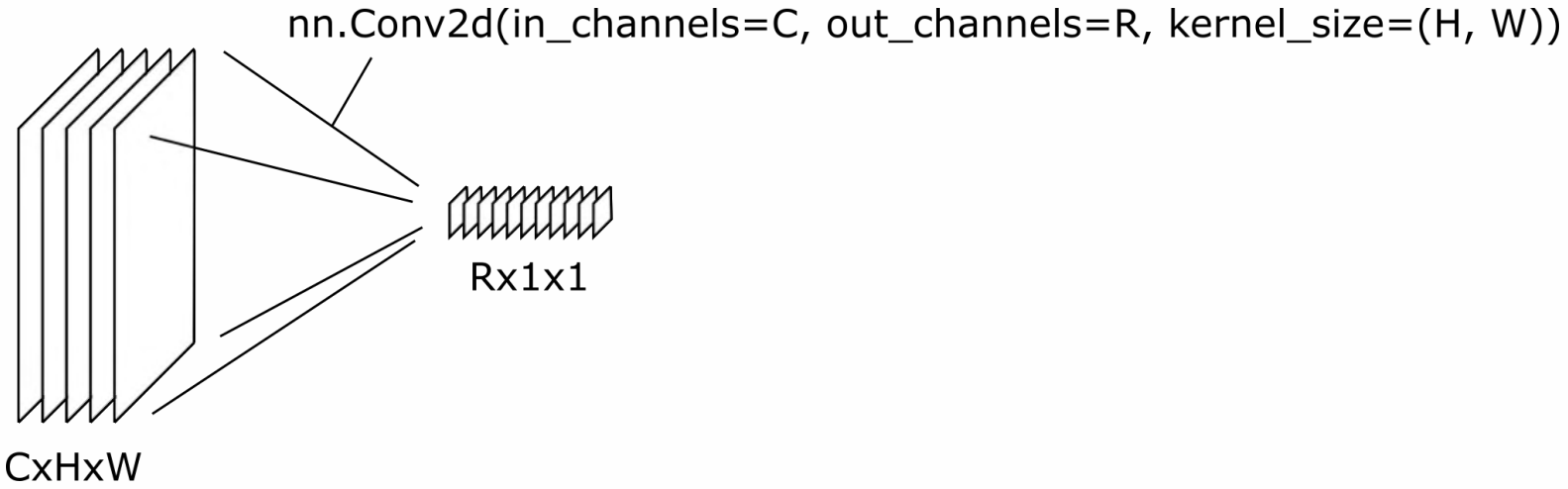
Answer:

Fully Connected layers!

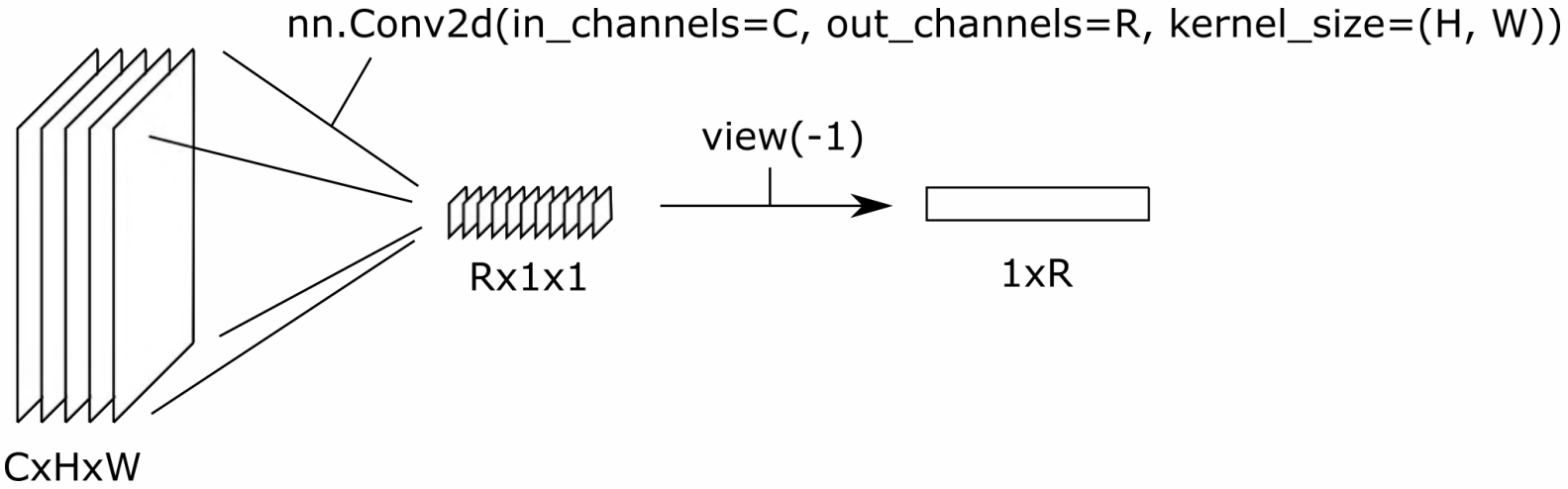
How to mimic Linear layer with Conv2d?



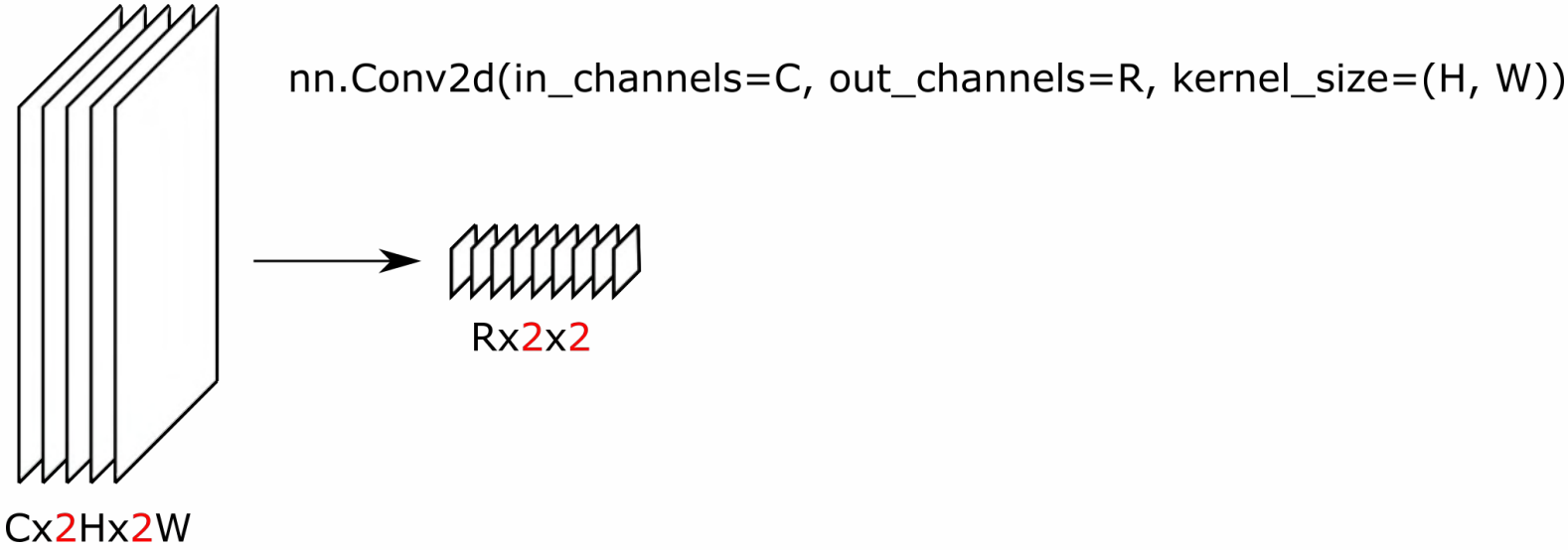
How to mimic Linear layer with Conv2d?



How to mimic Linear layer with Conv2d?



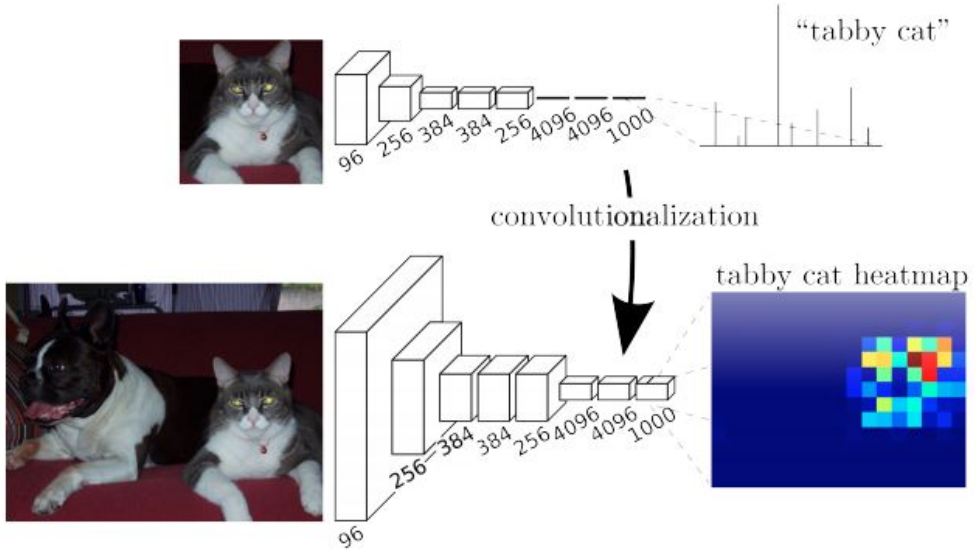
How to mimic Linear layer with Conv2d?



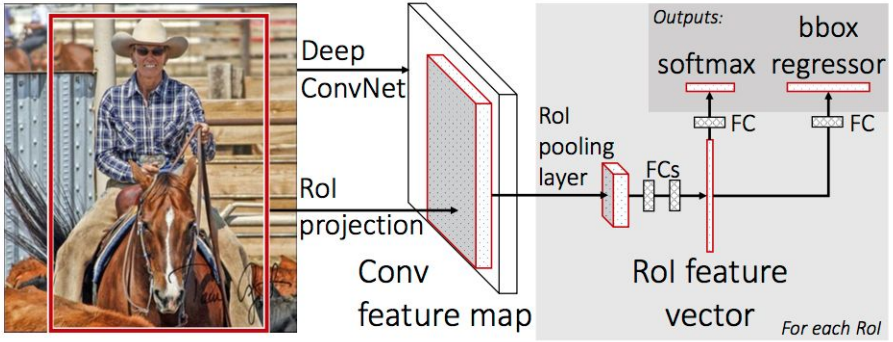
Fully convolutional network (FCN)

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.

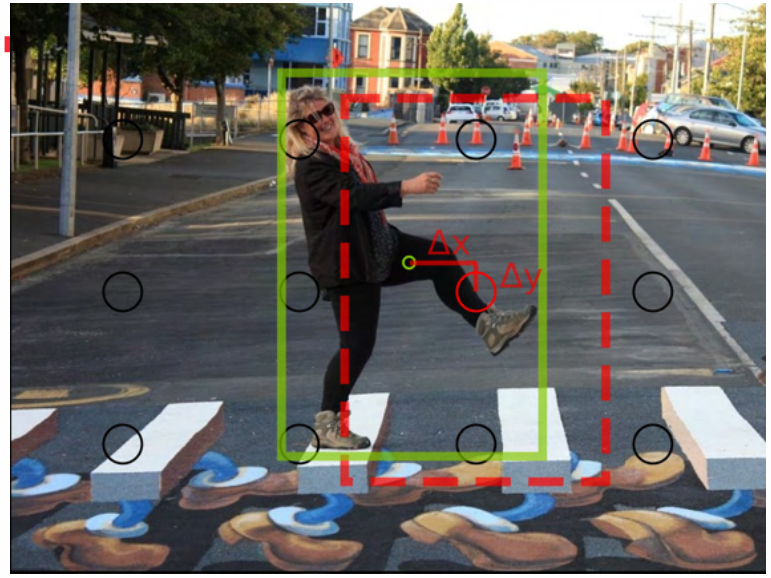


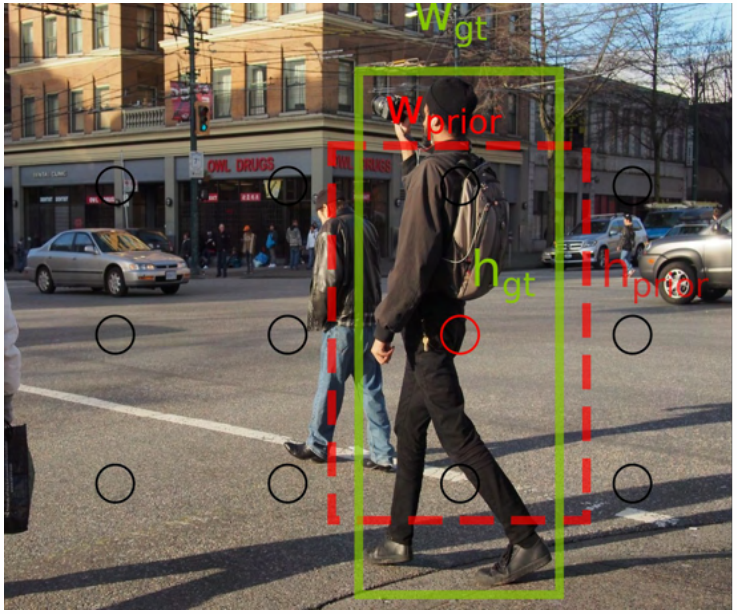
Fast RCNN



1) Same region proposal technique (selective search).
2) Feed full image to fully convolutional network, obtain feature map with size CxWxH.
3) For each proposal:
   a) Crop regions from feature map that corresponding to proposal using RoI Pooling. The result is fixed-size feature map of size approximately ($W_{image}/w_{proposal}$) and ($H_{image}/h_{proposal}$).
   b) Feed pooled result to fully connected layers, predict object class and bounding box regression (Δx, Δy, k_h, k_w).
   c) Adjust bbox with bounding box regression vector.
4) Apply Non Maximum Suppression.

BBox Regression





BBox regression vector: $(\Delta x, \Delta y, k_h, k_w)$.

$$k_h = h_{gt} / h_{prior}$$

$$k_w = w_{gt} / w_{prior}$$

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

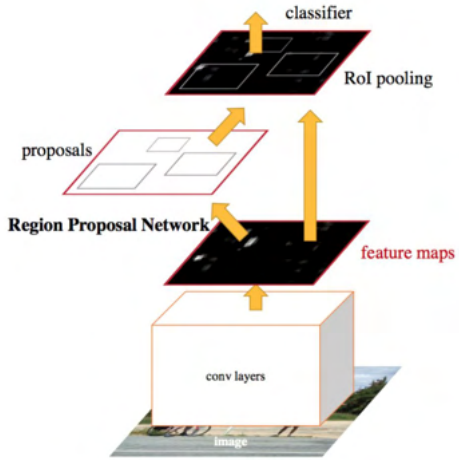
1) RoI Pooling layer is differentiable, so Fast RCNN can be trained end-to-end.
2) Both loss functions for classification and bounding box regression are optimized jointly.
3) Convolutions are applied to input image only once, but fully connected layers are applied to every pooled region proposal =>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

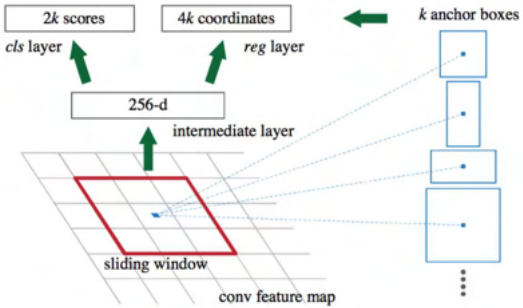
Main disadvantage of Fast RCNN: region proposal technique.

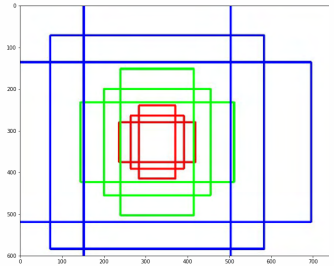
Let's train a CNN that will give us better region proposals!

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.

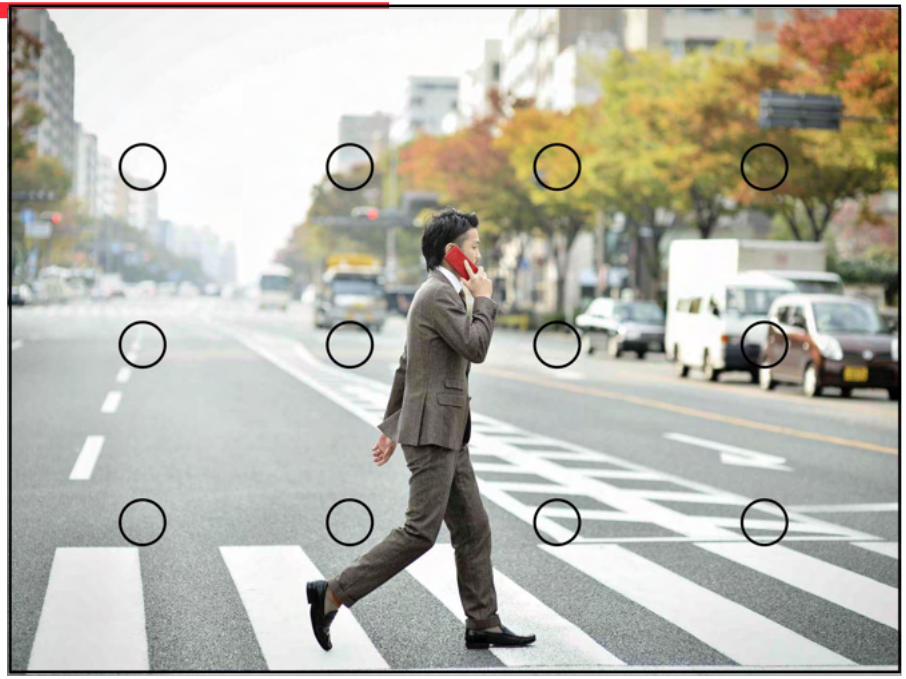




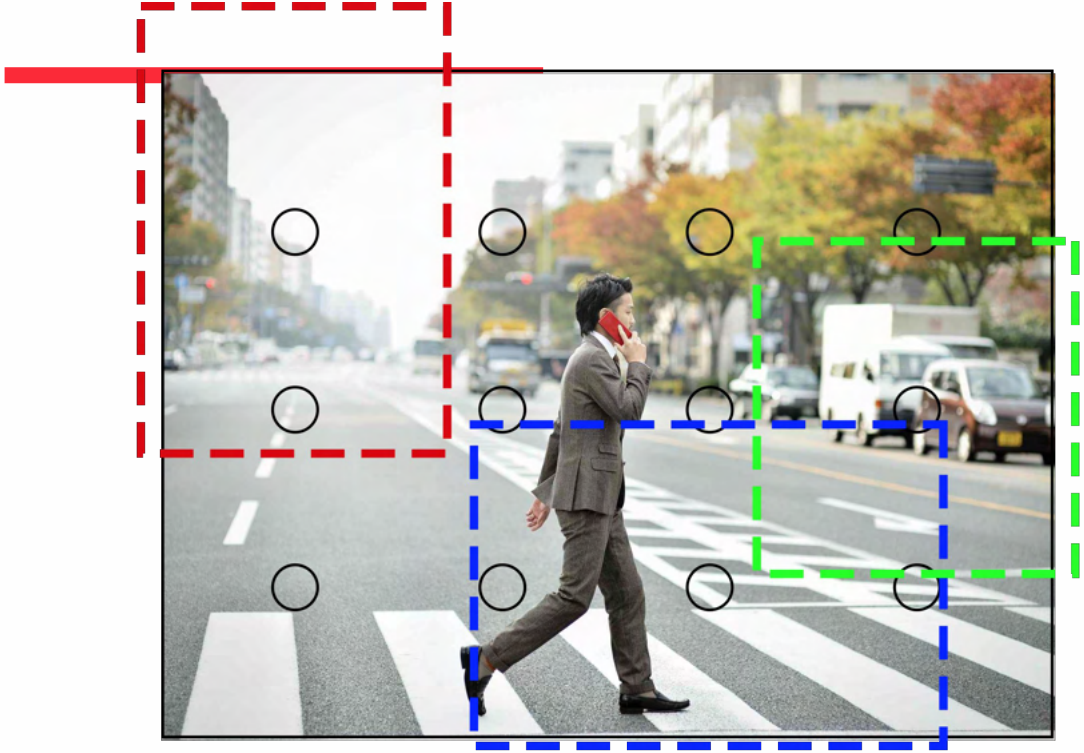


Anchor Boxes

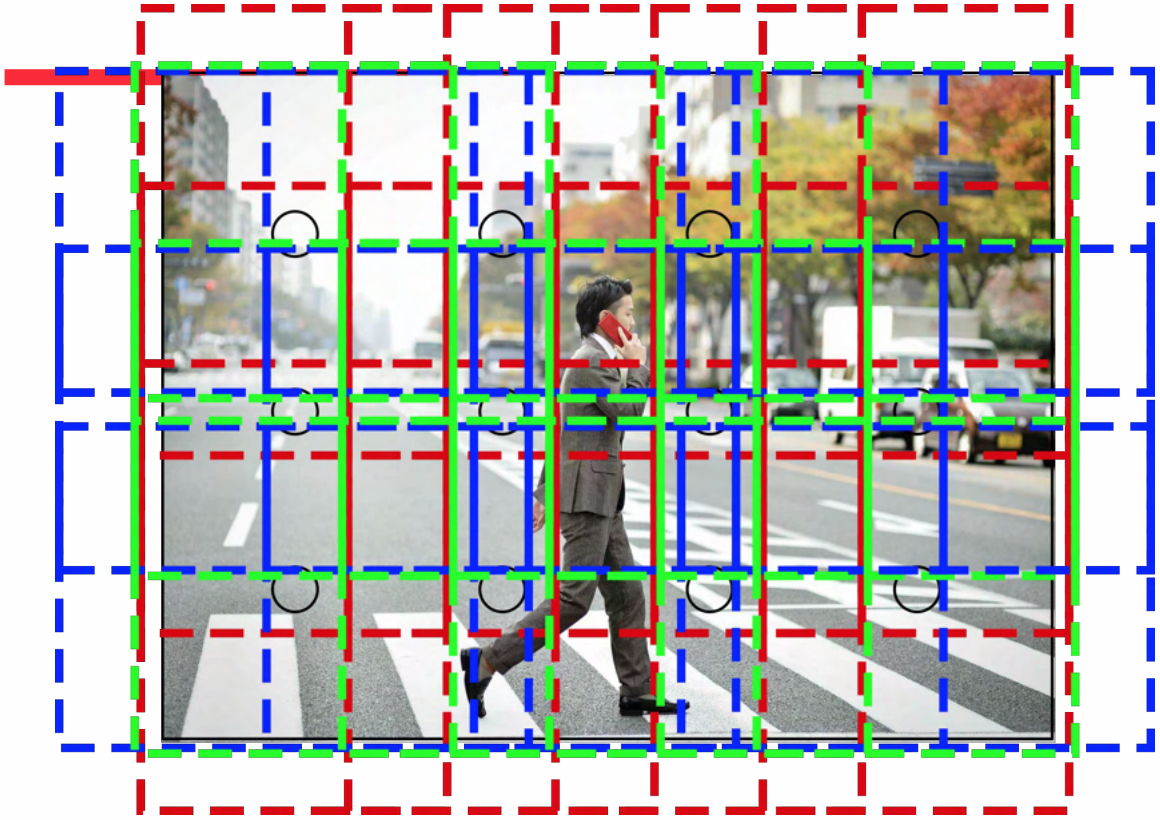
Anchor Boxes



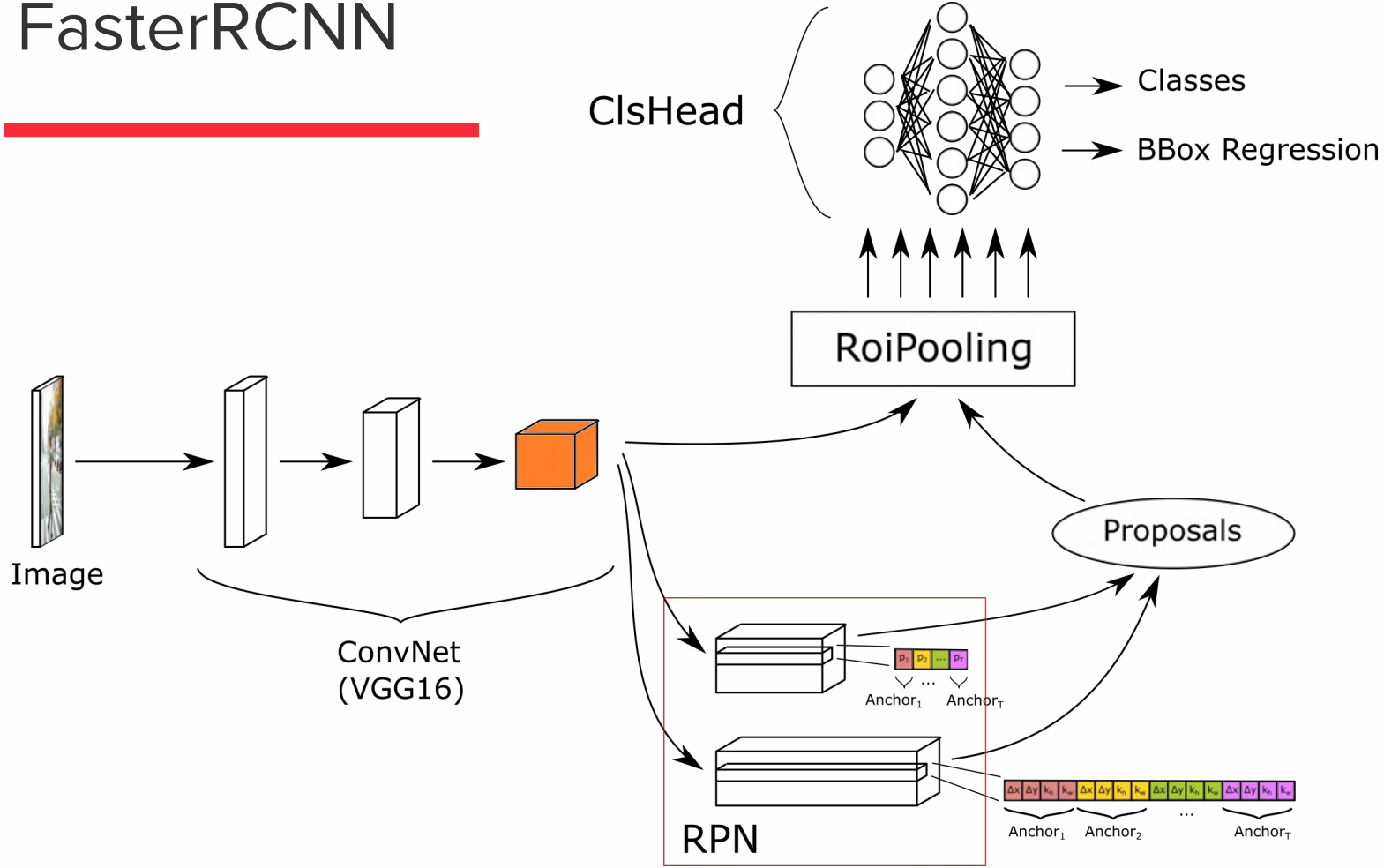
Anchor Boxes



Anchor Boxes



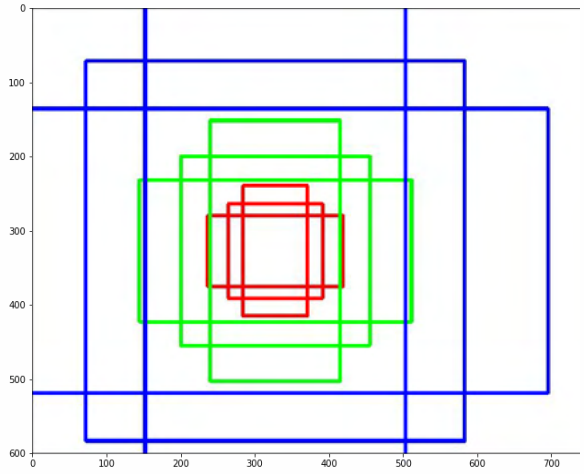
FasterRCNN


ClsHead
Classes
BBox Regression
RoiPooling
Proposals
Image
ConvNet
(VGG16)
RPN
Anchor1
AnchorT
Anchor1
Anchor2
AnchorT

Scale problem

Problem:

For each “location” we have same number of big and small boxes. Smaller boxes are more “sparse”. But the image can fit more small objects than big ones.



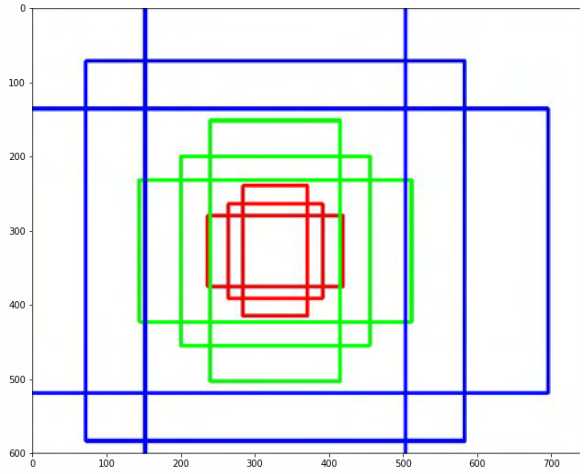
Scale problem

Problem:

For each “location” we have same number of big and small boxes. Smaller boxes are more “sparse”. But the image can fit more small objects than big ones.

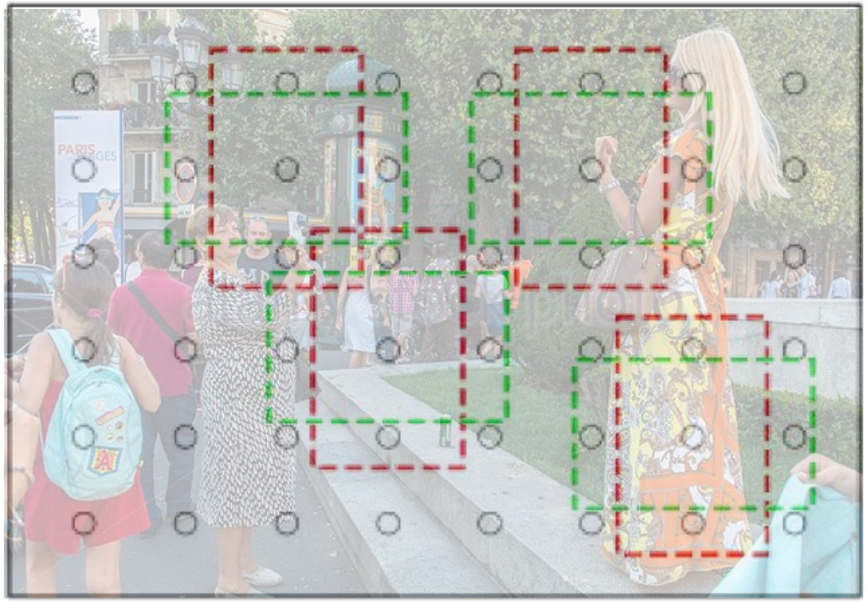
Solution:

Let’s detect at different scales!

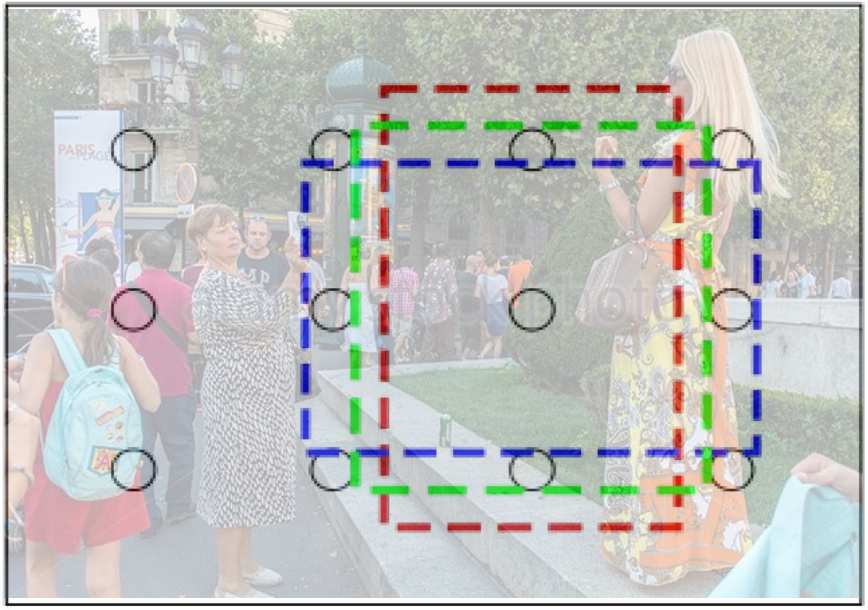



PARIS
PLAGES

Different Scales

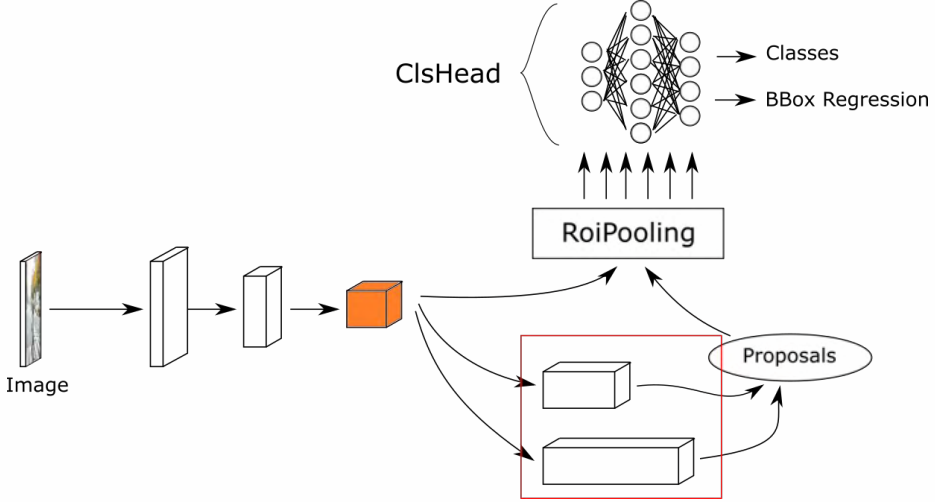


Smaller scale

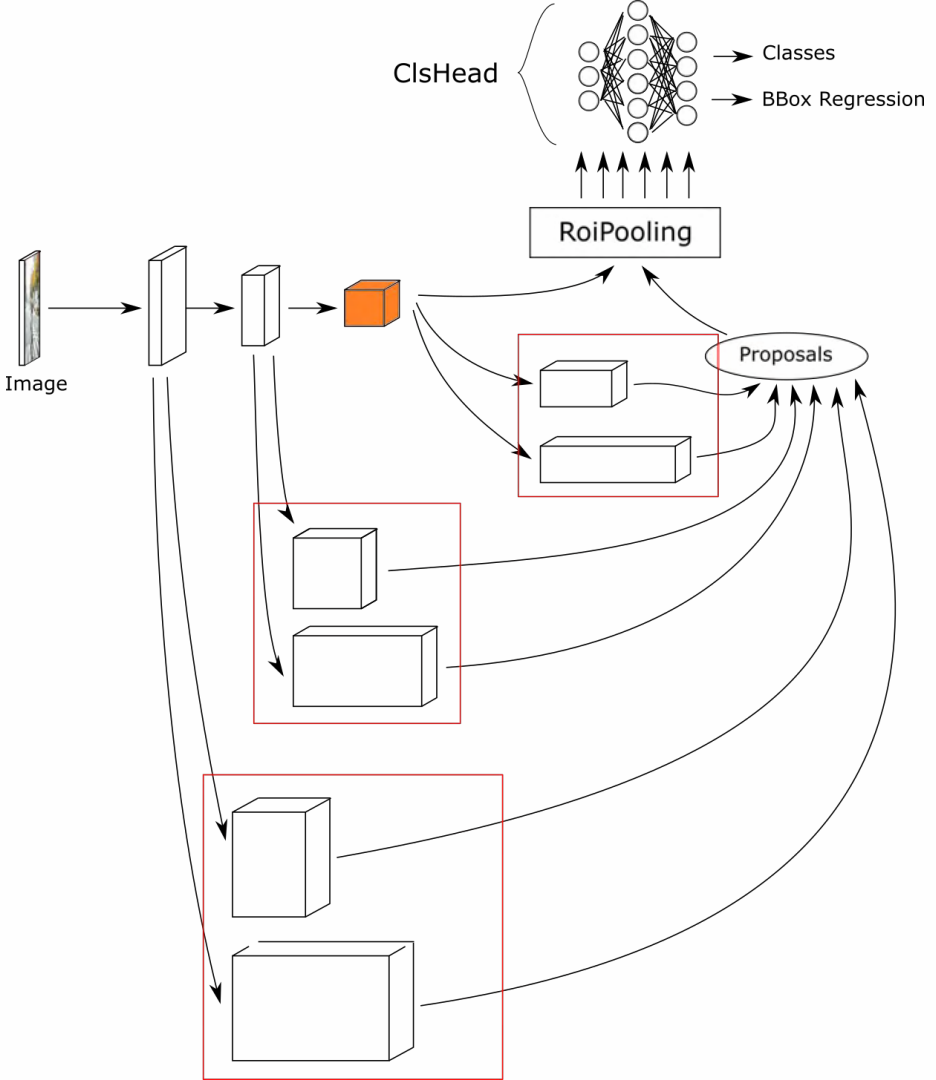


Bigger scale

Multiple Scales

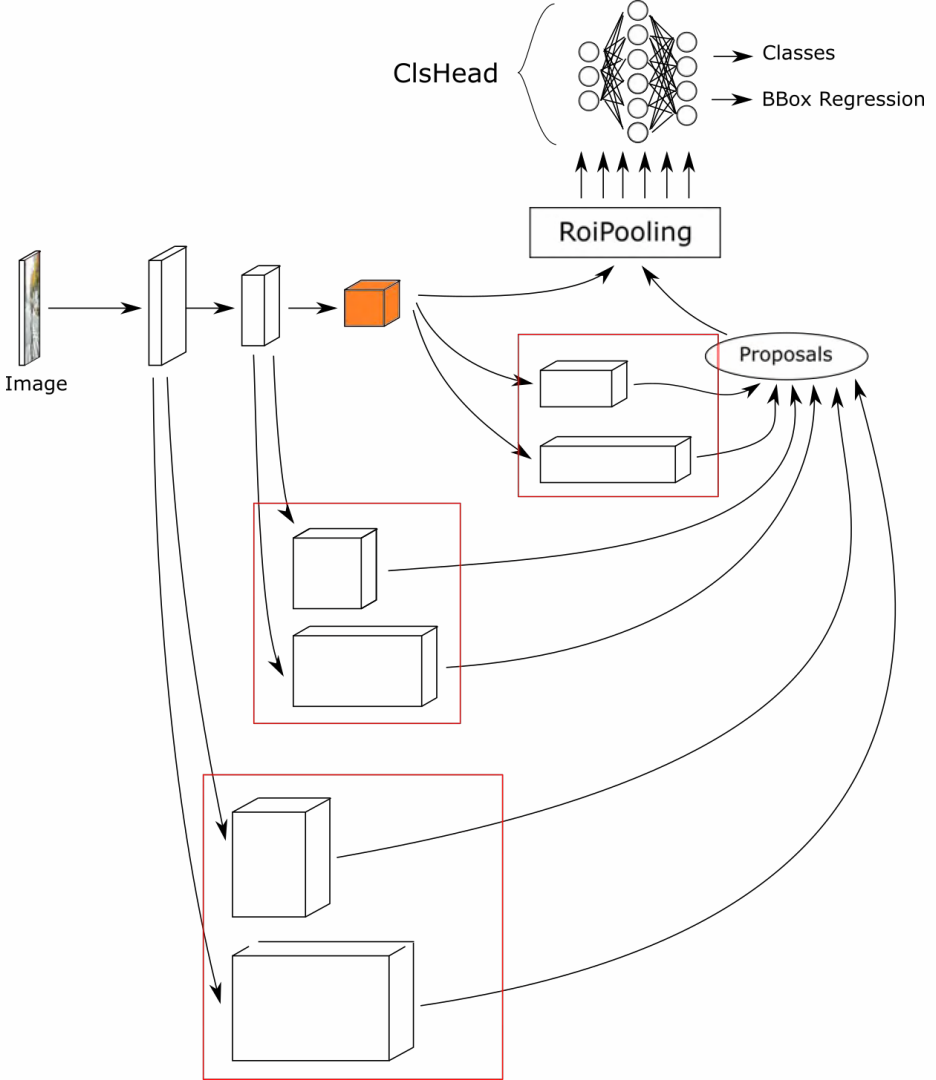


Multiple Scales



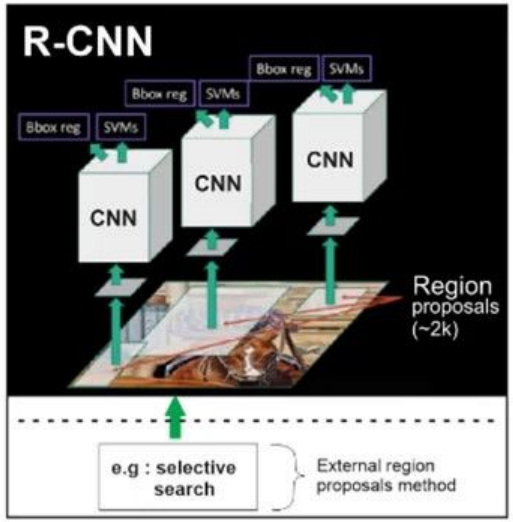
Multiple Scales

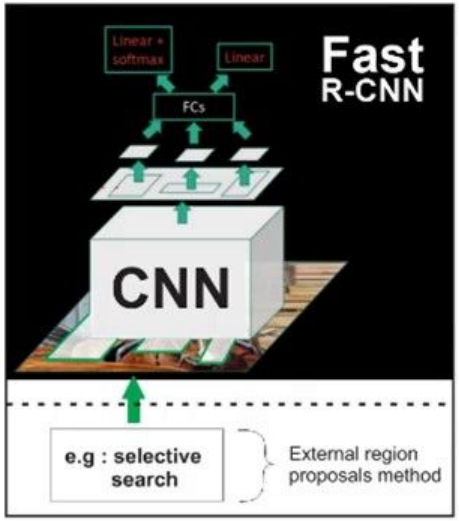
1) Previous CNN tensors are used for proposal generation
2) Proposals from different scales are stored in one array.

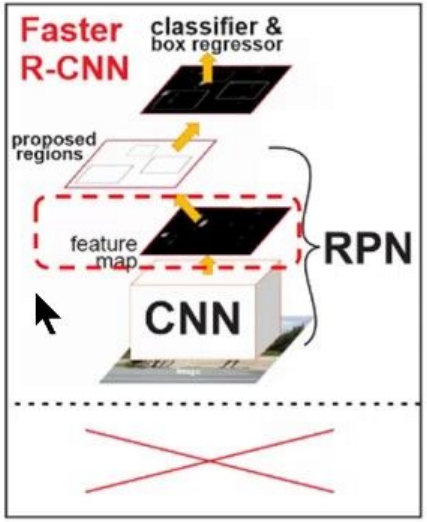


Faster RCNN Details

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

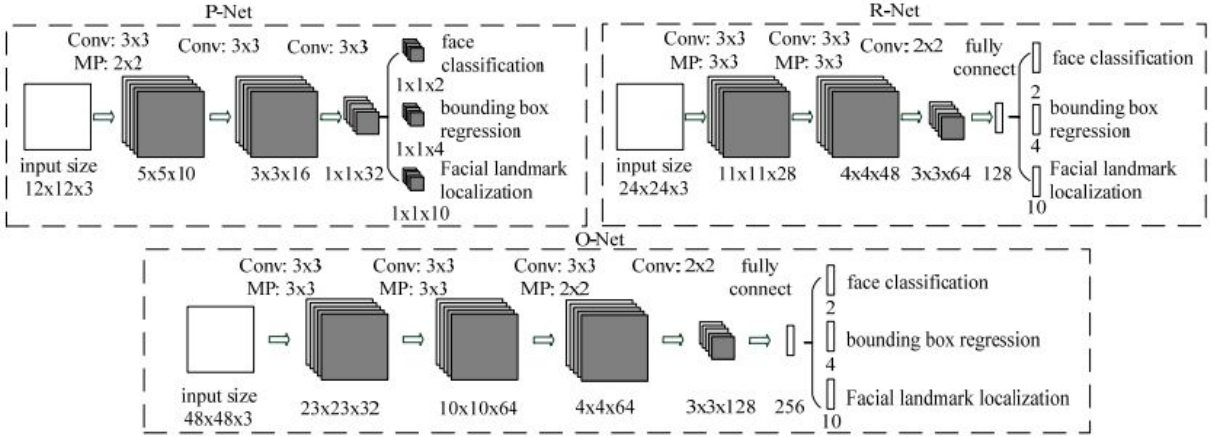
Idea: Process image with many simple and computationally cheap classifiers, effectively discarding most of empty regions on the image. Regions that have not been discarded are processed by more complex classifiers.

Examples:

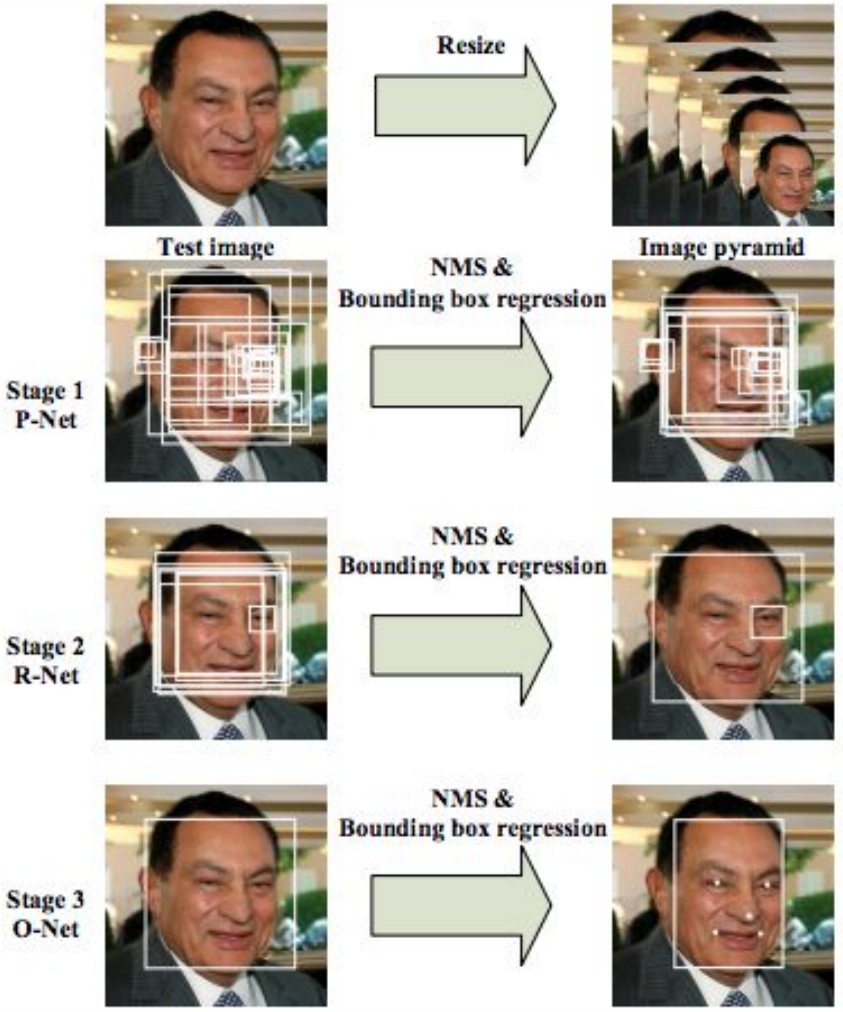
1) Viola-Jones (Classic)
2) MTCNN

MTCNN

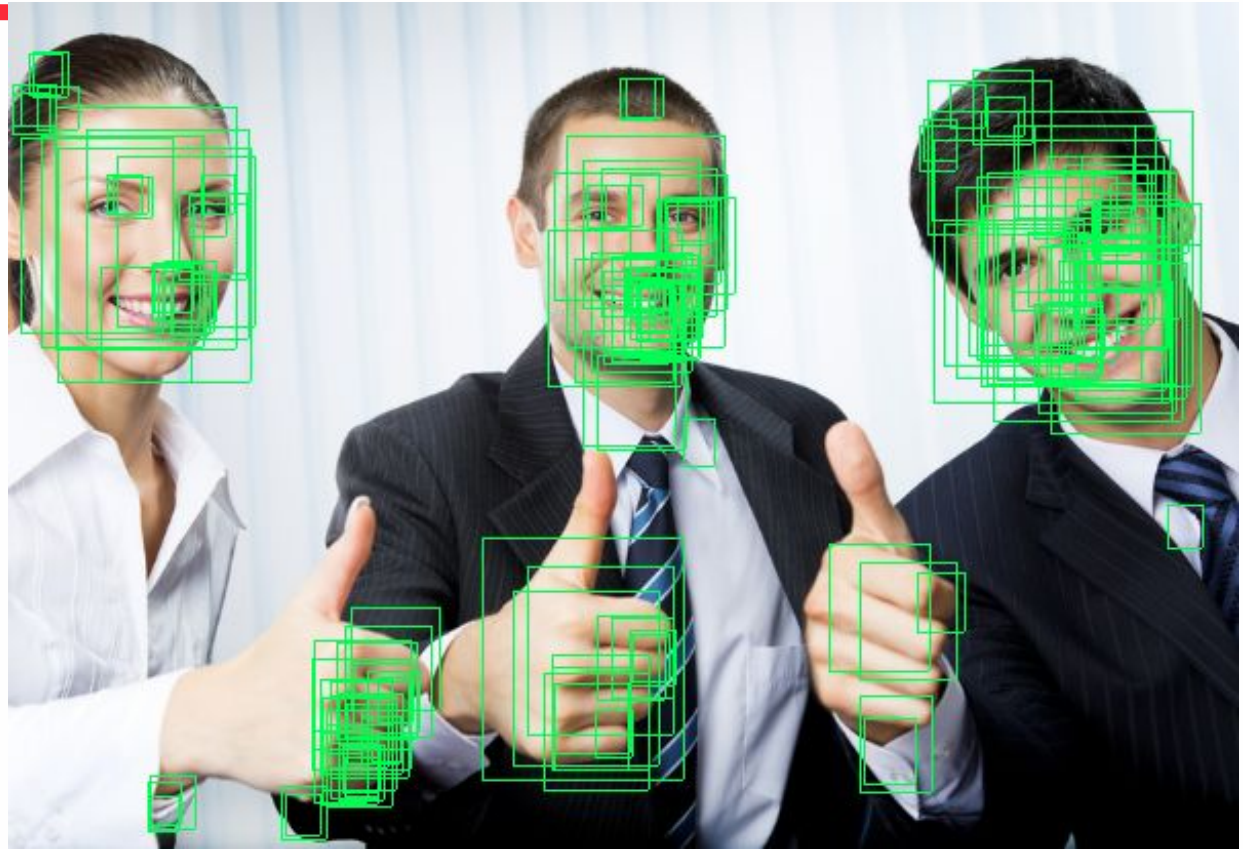
1) 3 stages:
   a) PNet - FCN, generates **P**roposals.
   b) RNet, **R**efines proposals.
   c) ONet, further refines detections and **O**utputs facial landmarks.



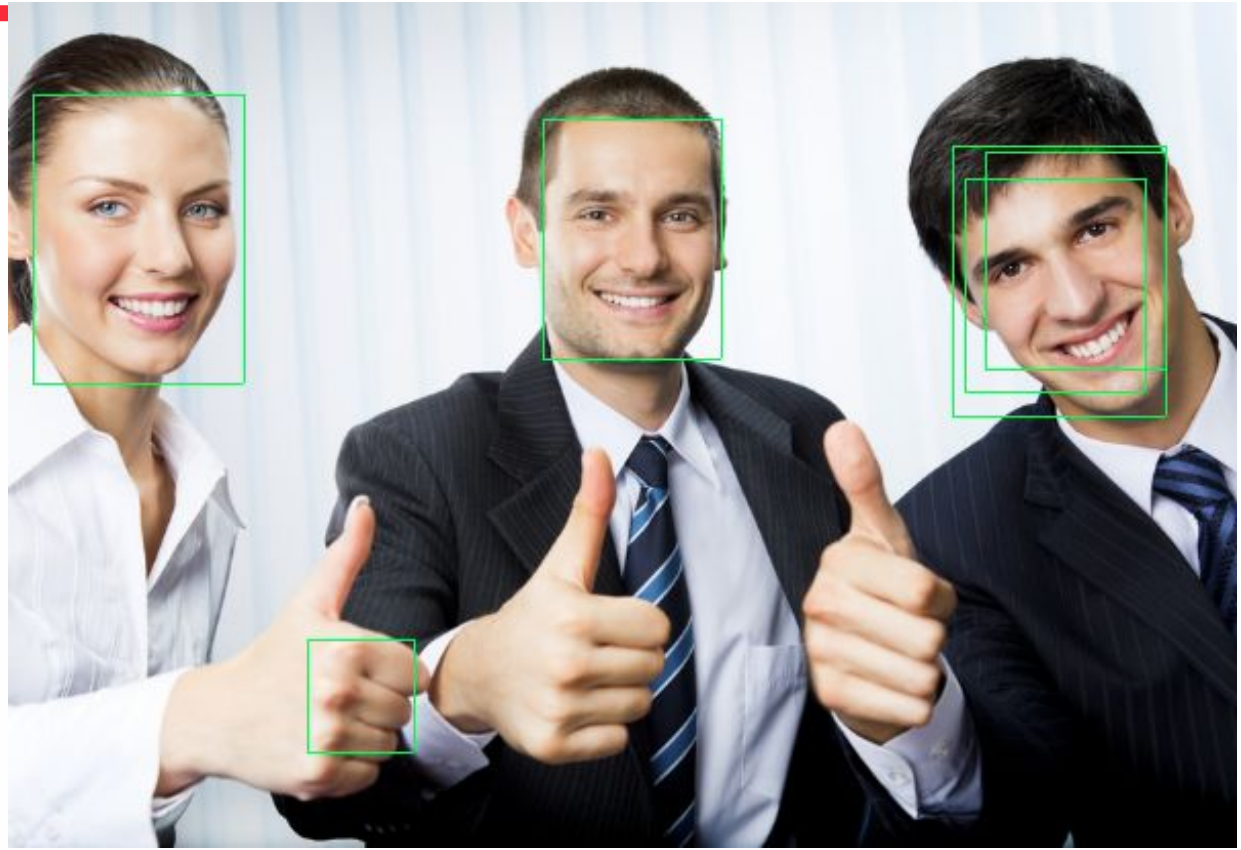
MTCNN Framework



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.

More good architectures

1) R-FCN
2) FCOS
3) Cascade-RCNN

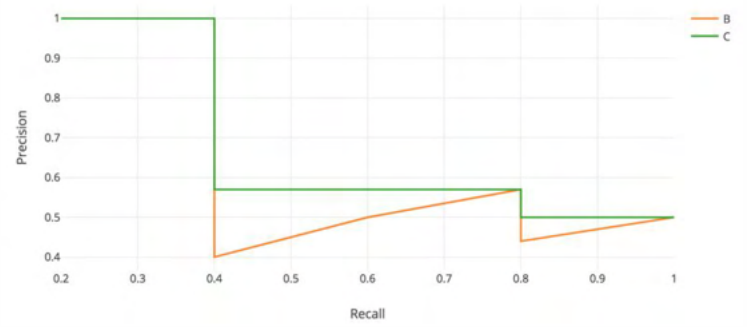
\+ MaskRCNN

Metrics

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

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

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



Vary threshold => obtain different Precision and Recall.

Orange line - PR curve,

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

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

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.

академия больших данных

mail.ru group

Neural Networks for detection 2

Boris Lestsov
Mail.Ru Group

Single Shot Approaches

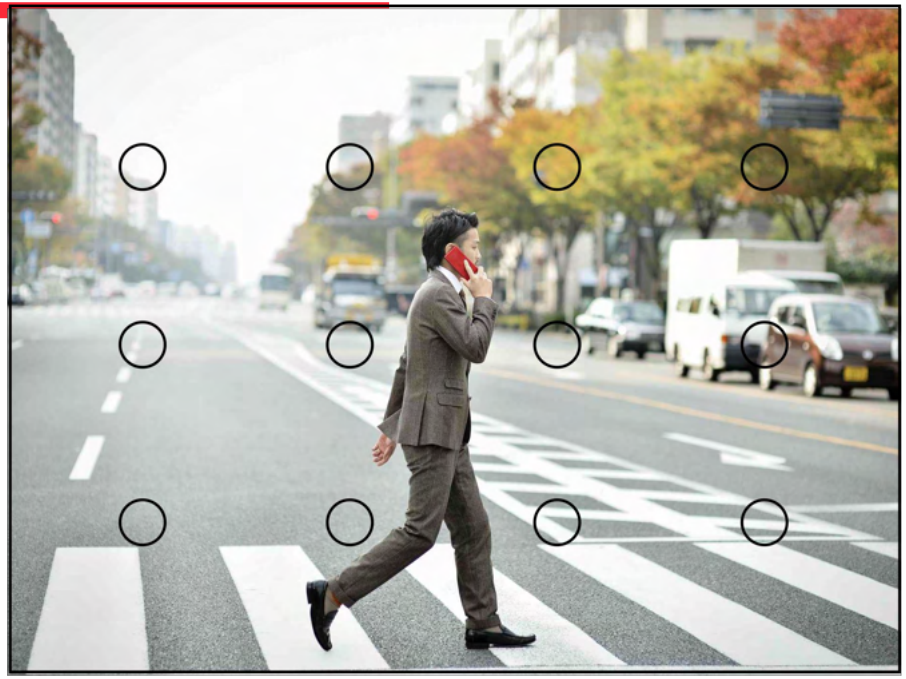
Single shot approaches

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

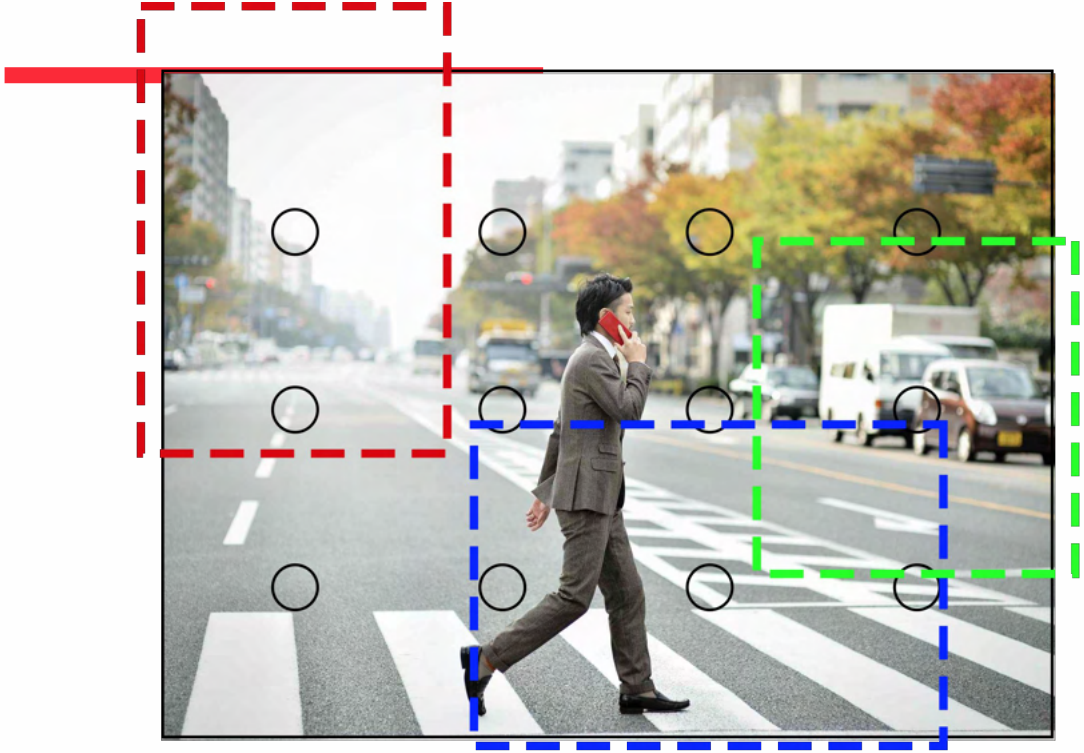
BTW, should not be confused with one-shot learning.

Reminder: Anchor Boxes

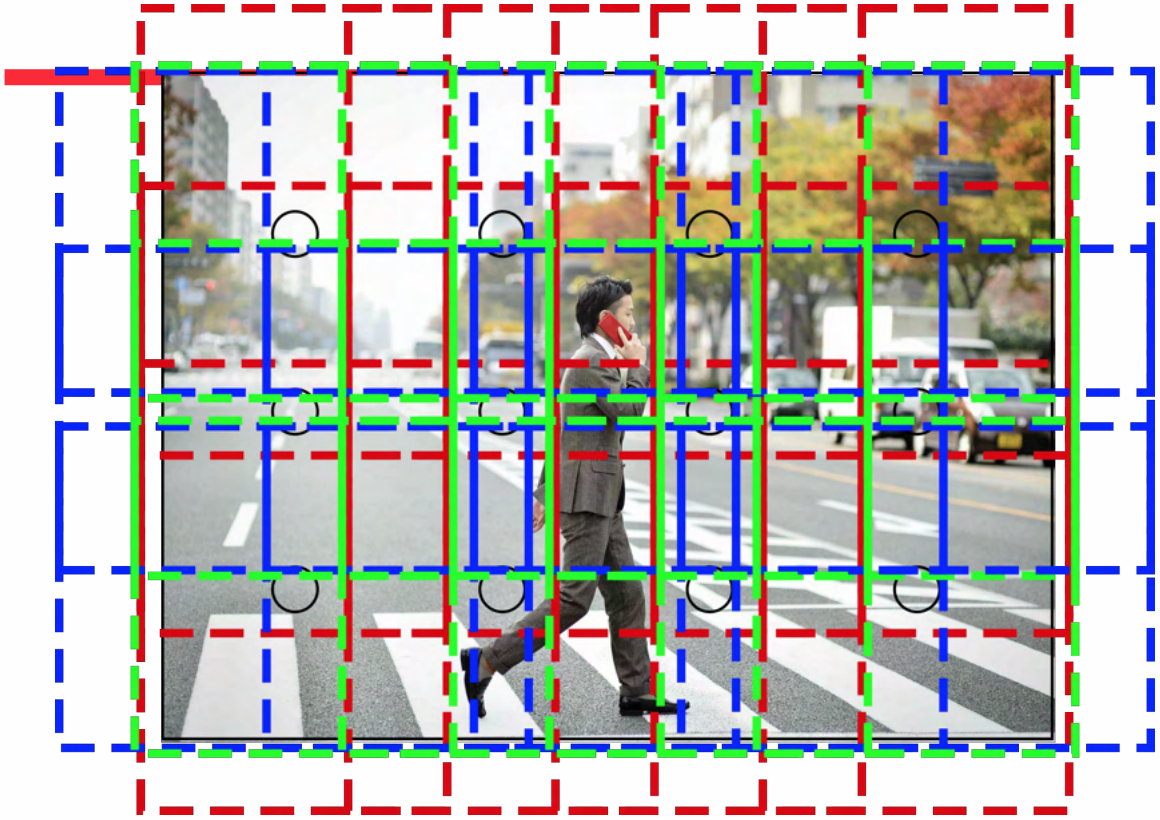
Reminder: Anchor Boxes



Reminder: Anchor Boxes

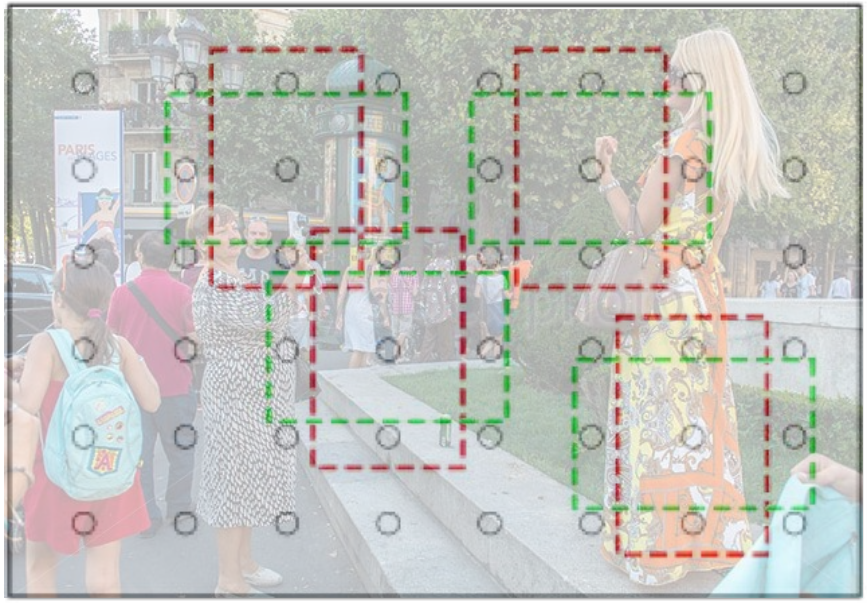


Reminder: Anchor Boxes

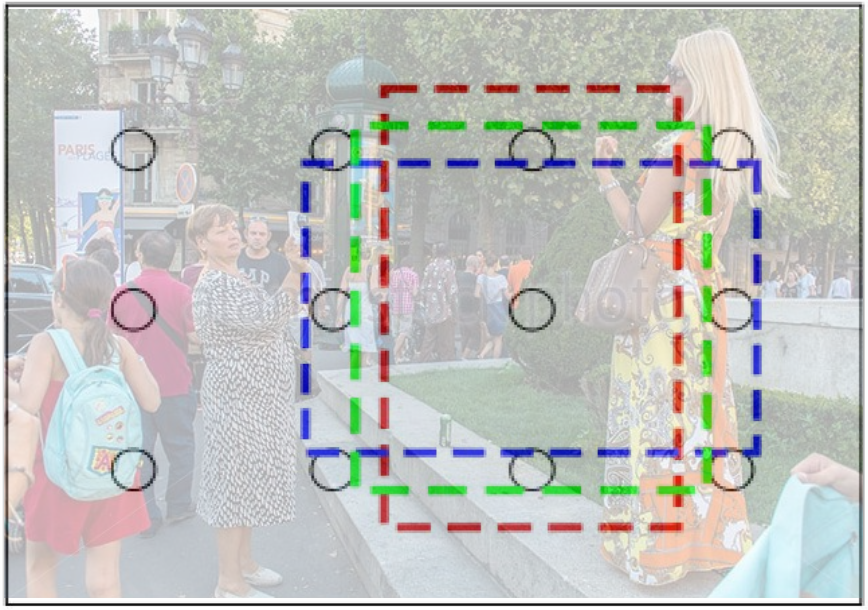



PARIS
PLAGES

Reminder: Anchor Boxes



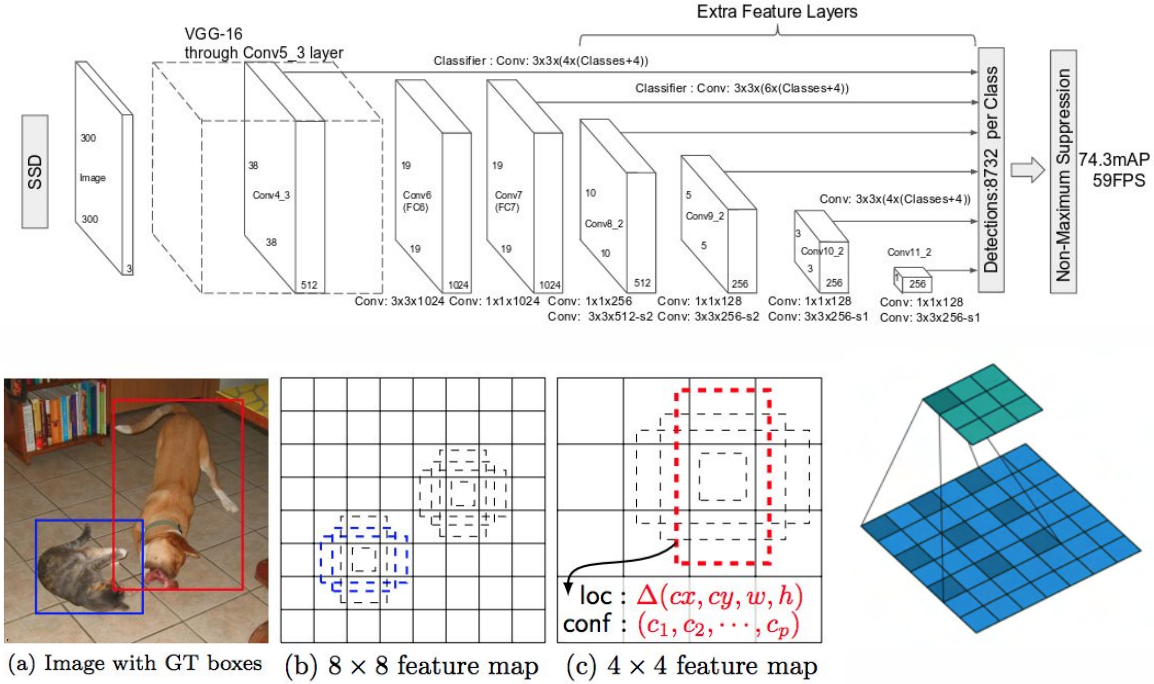
Smaller scale



Bigger scale

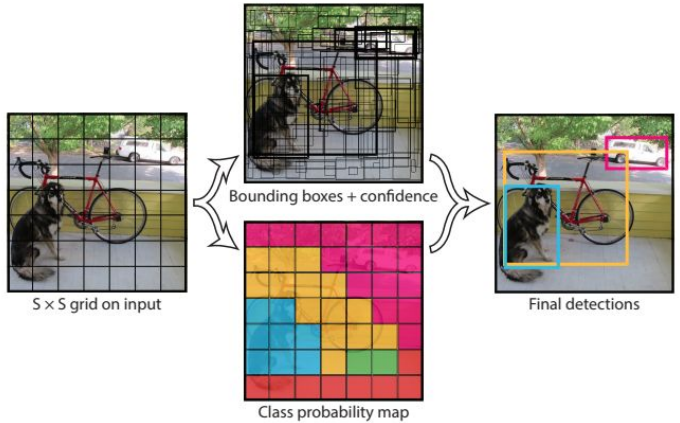
1) Single Shot Detector (SSD)

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)



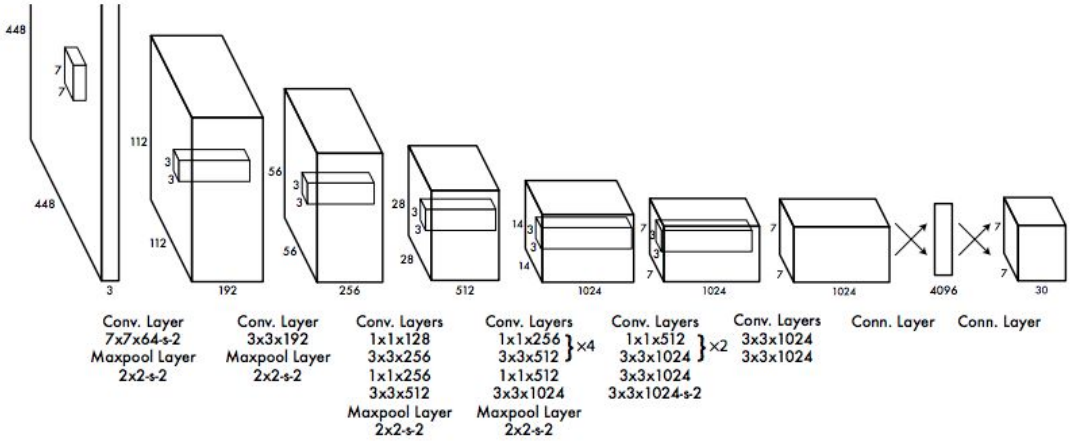
YOLO (You Only Look Once)

1) Tensor of size SxSx(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) “Darknet” framework

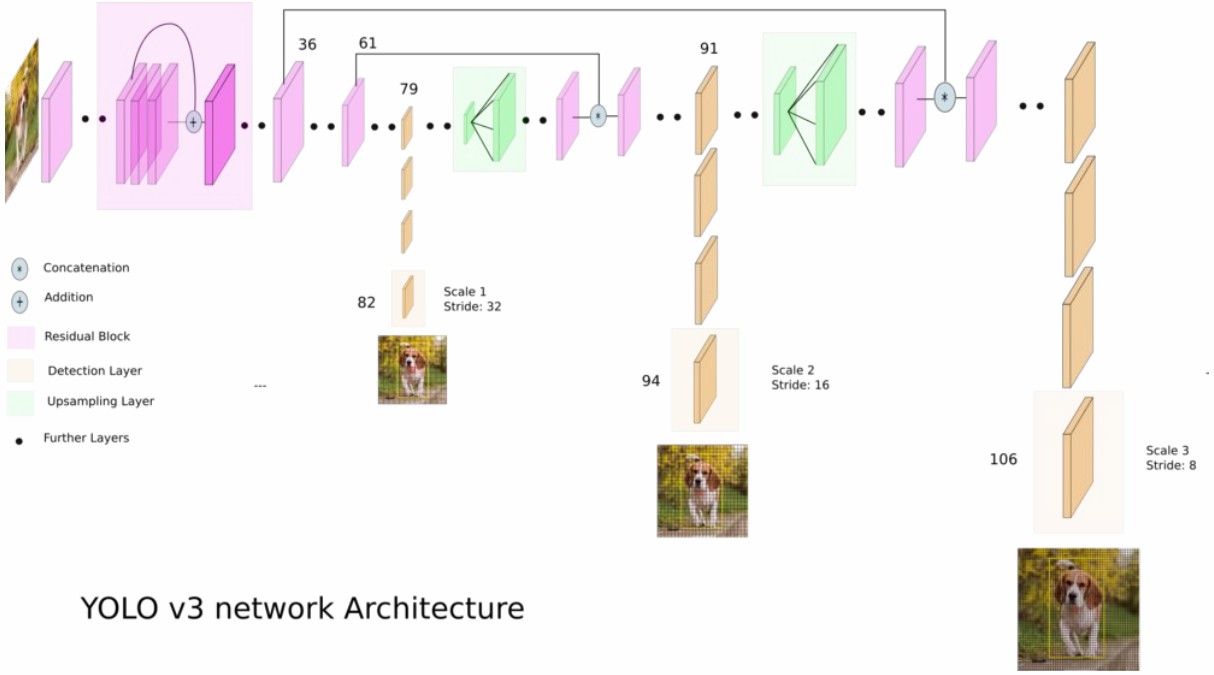


Further reading on YOLO:

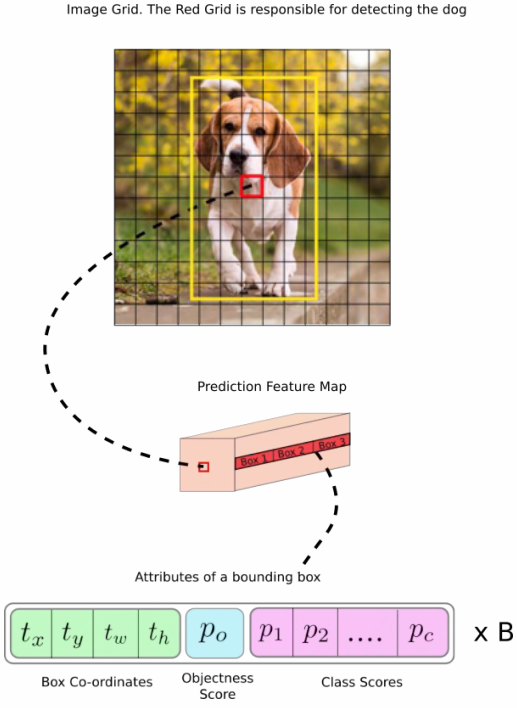
1) YOLO v2 (a.k.a. YOLO9000): more classes.
2) YOLO v3: log. reg. for each class. Helps with overlapping objects. Multiscale anchor boxes.
3) YOLO v4...



YOLO v3



YOLO v3 network Architecture



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): pt - predicted probability of g.t. class:

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

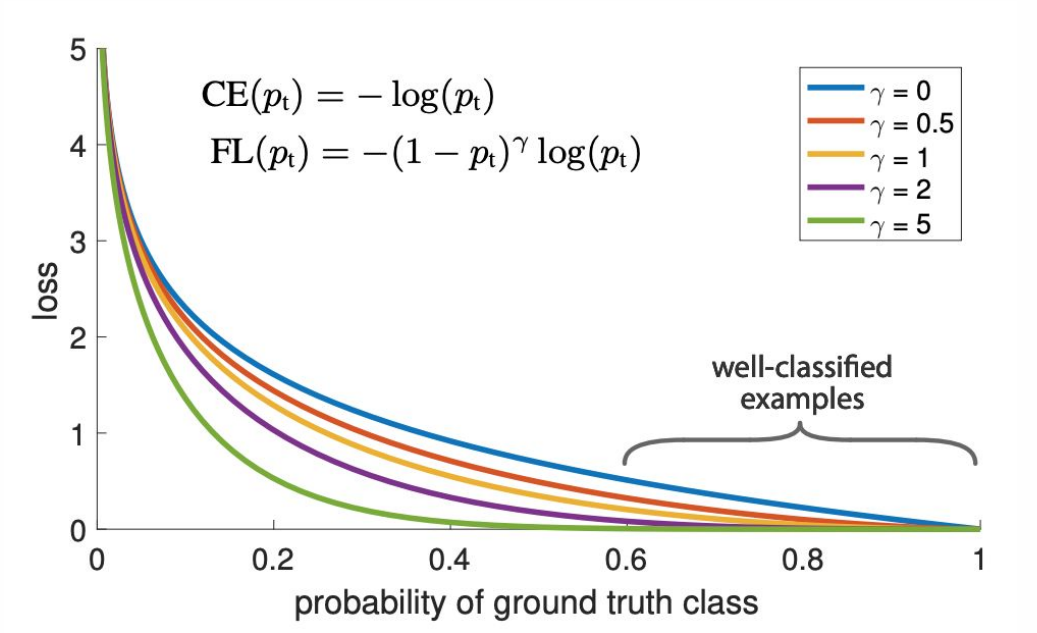
$$\mathrm{FL}(p_t) = -(1 - p_t)^{\gamma} \log(p_t)$$

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

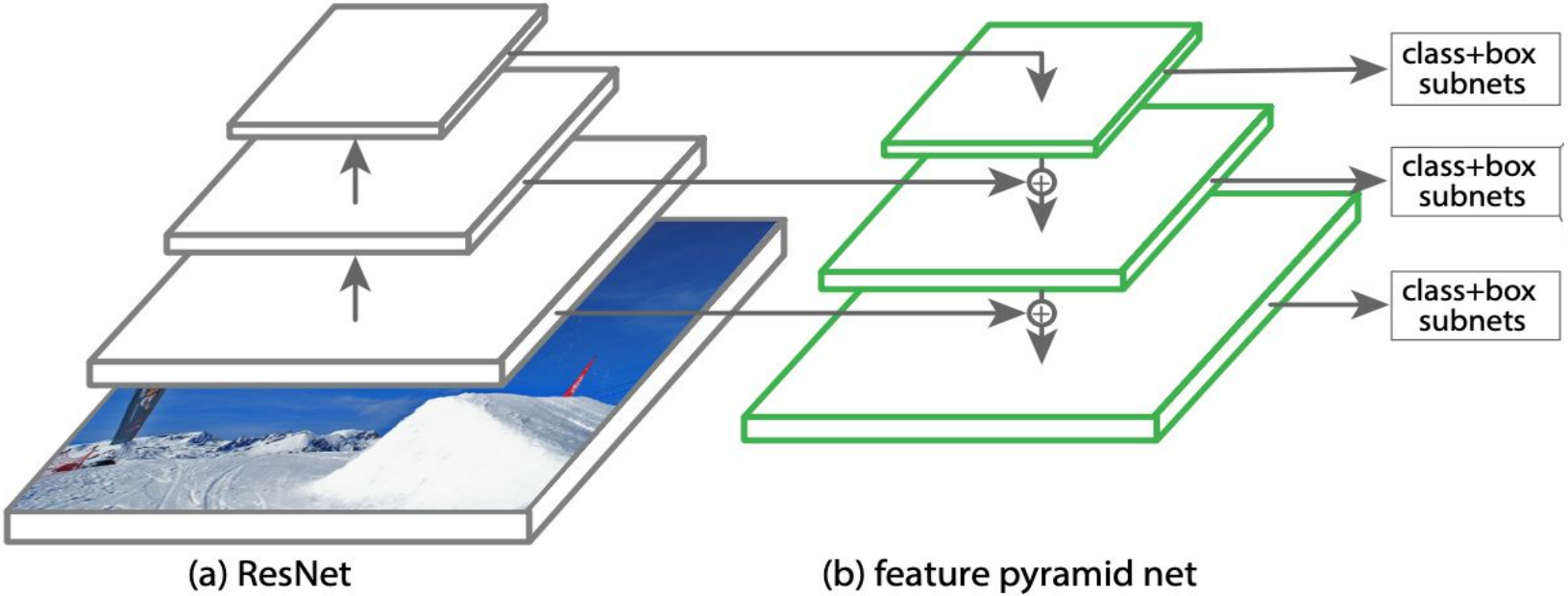
Focal Loss

Well classified examples => smaller contribution

Analogue of Online Hard Example Mining

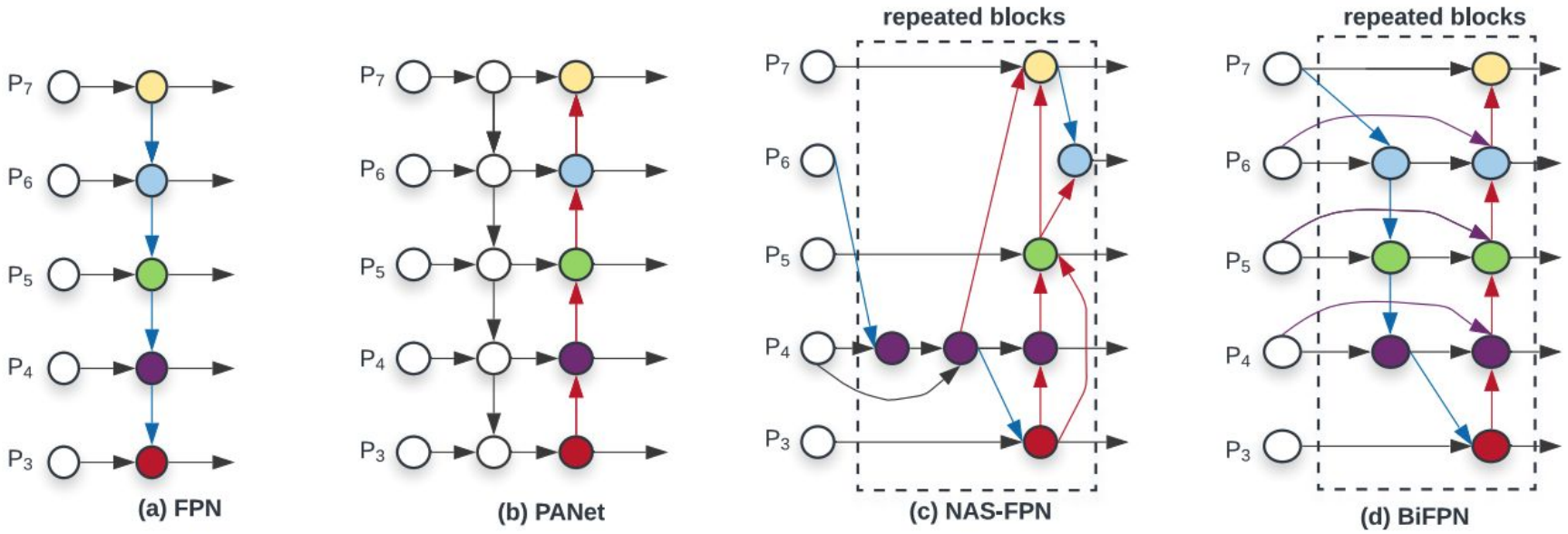


Feature Pyramid Network (FPN)



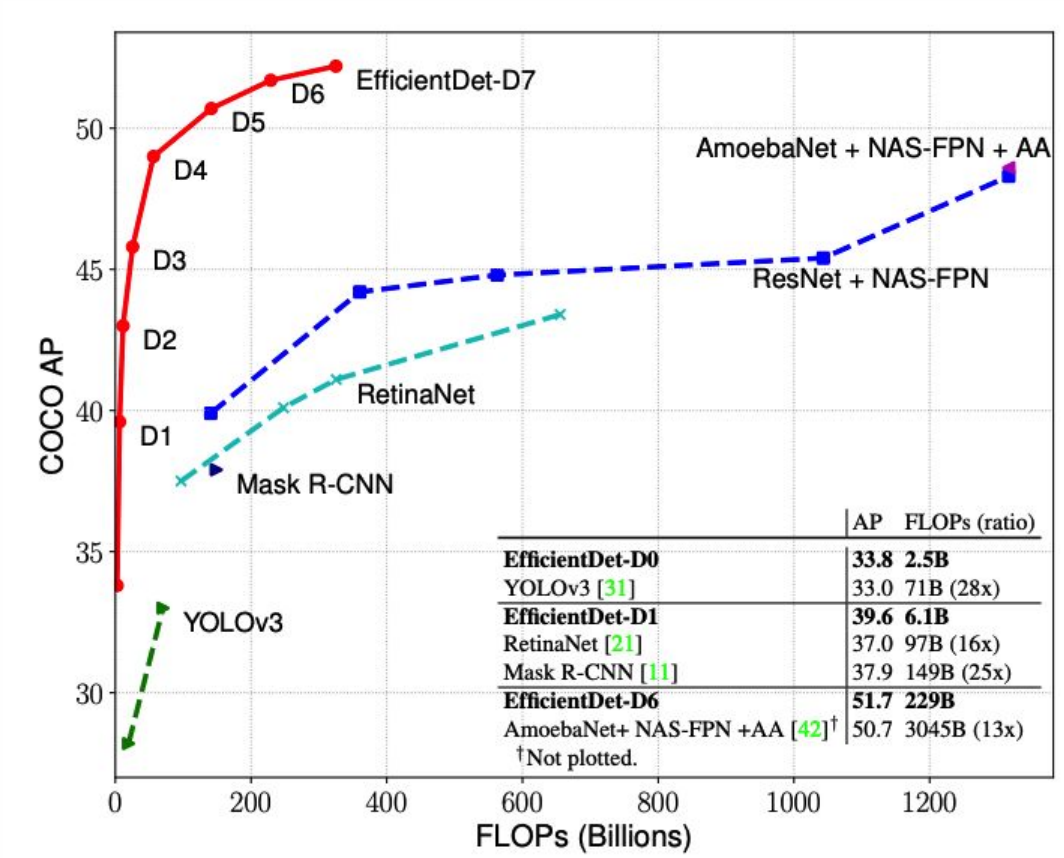
Higher level features for smaller scales

Cooler Feature Pyramid Network (FPN)



EfficientDet

1) Backbone: Efficient**N**et
2) Bi-FPN

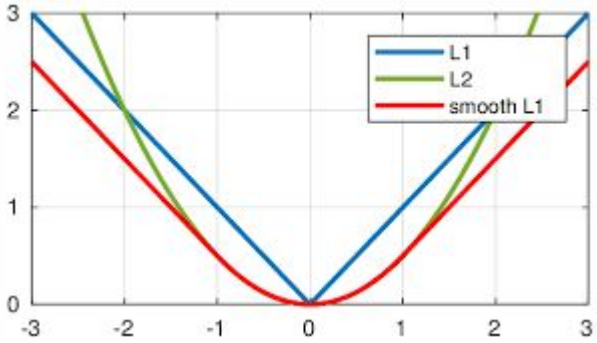


	AP	FLOPs (ratio)
EfficientDet-D0	**33.8**	**2.5B**
YOLOv3 [31]	33.0	71B (28x)
EfficientDet-D1	**39.6**	**6.1B**
RetinaNet [21]	37.0	97B (16x)
Mask R-CNN [11]	37.9	149B (25x)
EfficientDet-D6	**51.7**	**229B**
AmoebaNet+ NAS-FPN +AA [42]†	50.7	3045B (13x)

†Not plotted.

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

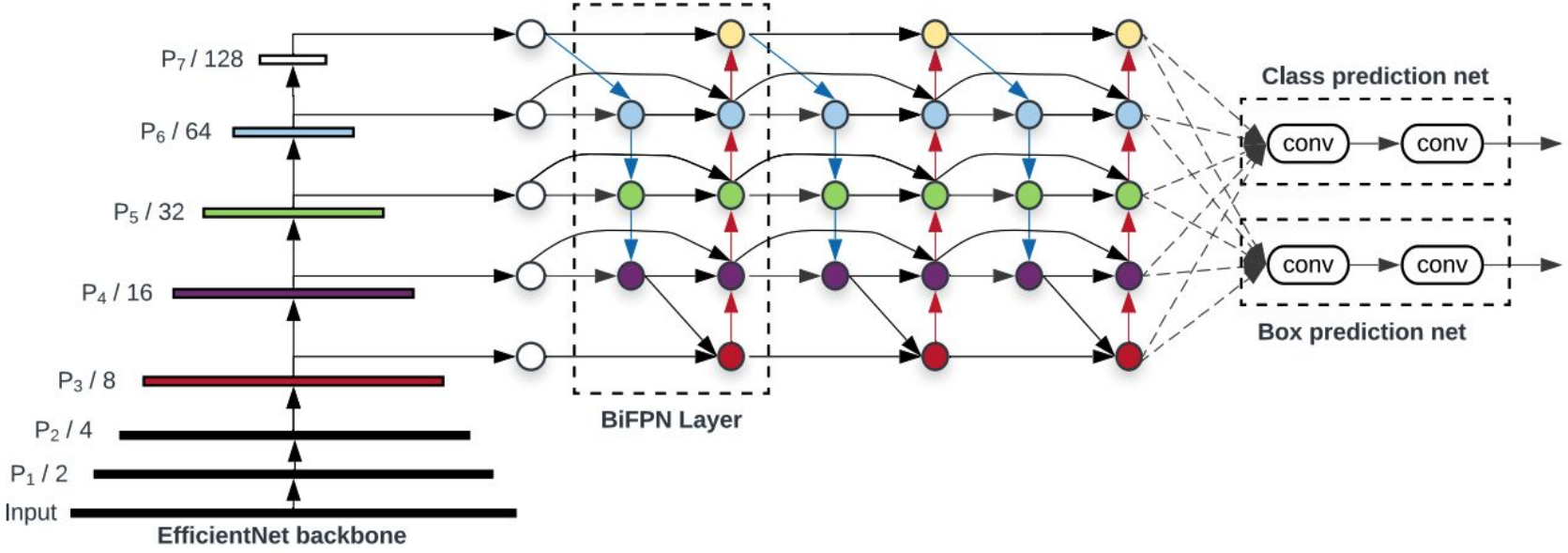


More Architectures

1) EfficientDet
2) FCOS
3) CenterNet
4) YOLOv4
5) CascadeRCNN

Questions!

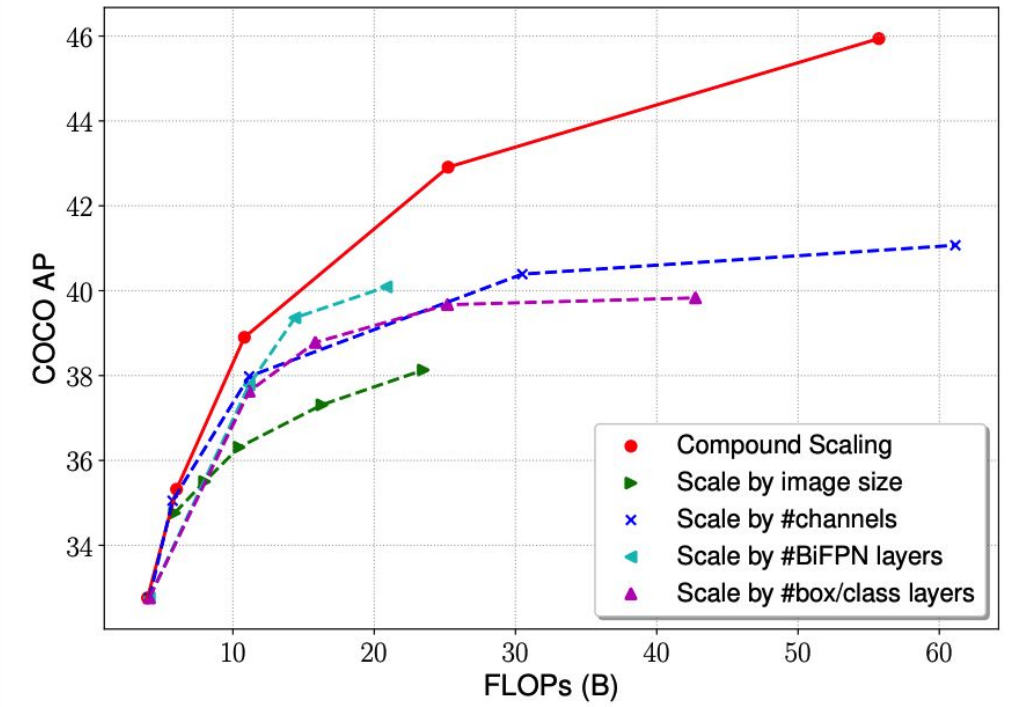
EfficientDet



EfficientNet

EfficientNet:

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



MTCNN training

1) Crop image patches with IoU>0.6 with ground-truth bboxes, patches with 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.