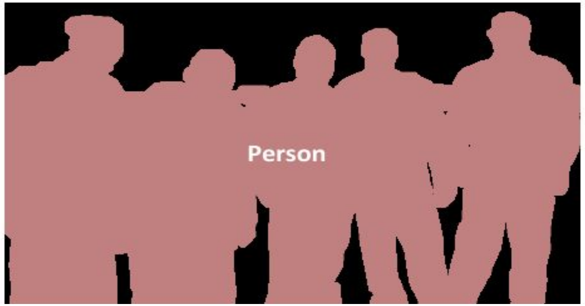
Deep machine learning for Computer Vision

Andrii Liubonko
Samsung R&D Institute Ukraine

Contents

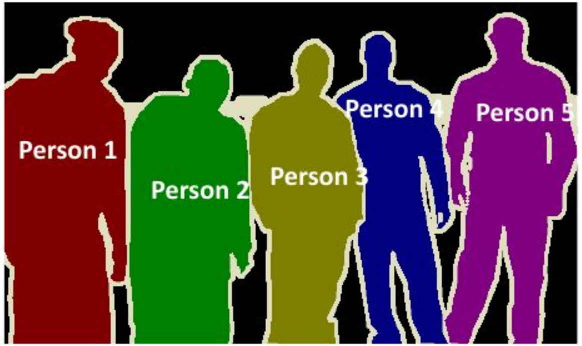
Visual Perception Problems

Classification + Localization



Semantic Segmentation

Object Detection



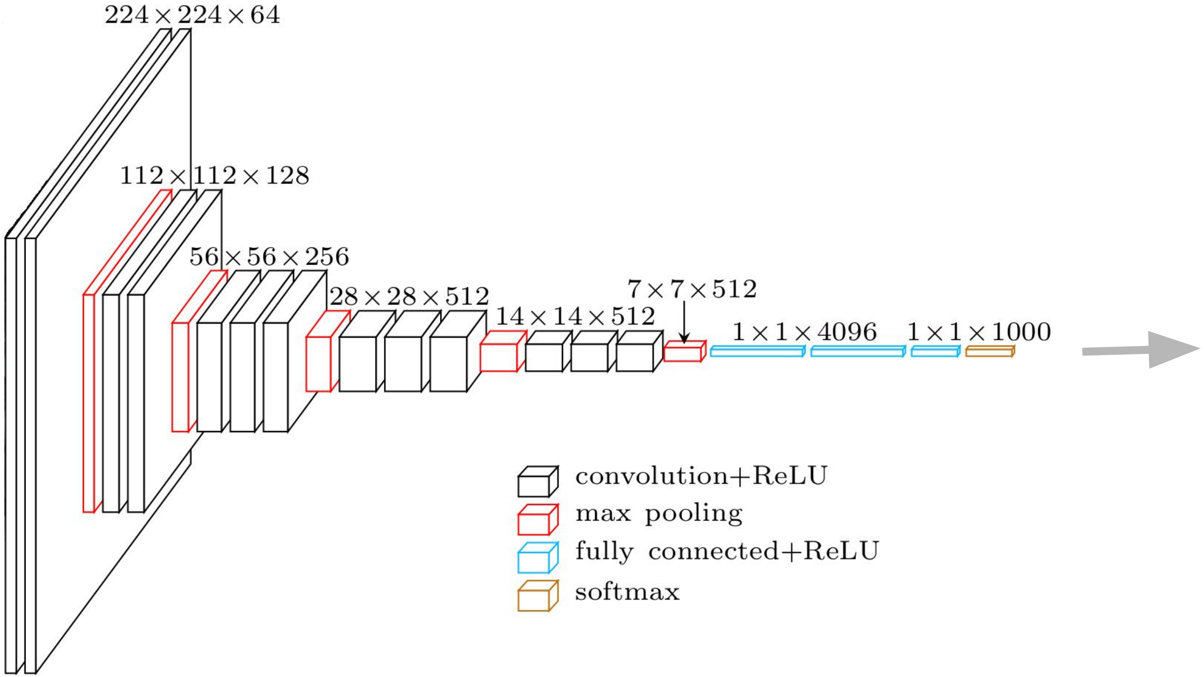
Instance Segmentation

Localization

[Score] dog: 0.98
cow: 0.01
...
[Bounding Box]
(x,y,h,w)

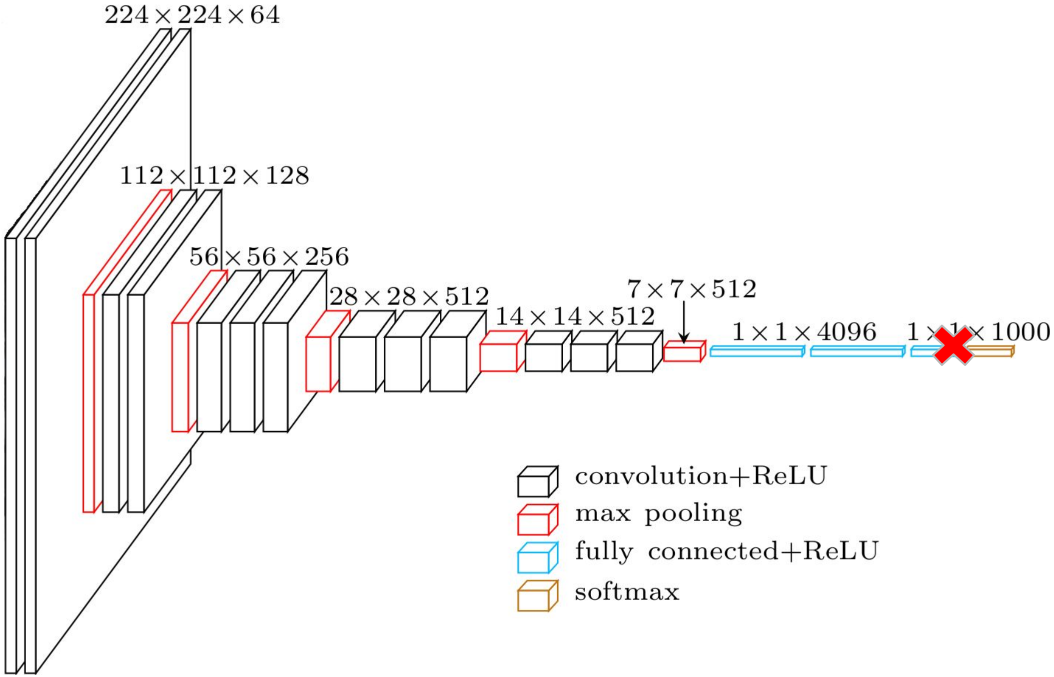
The main assumption is that there is only **one** object in the image

Image Classification



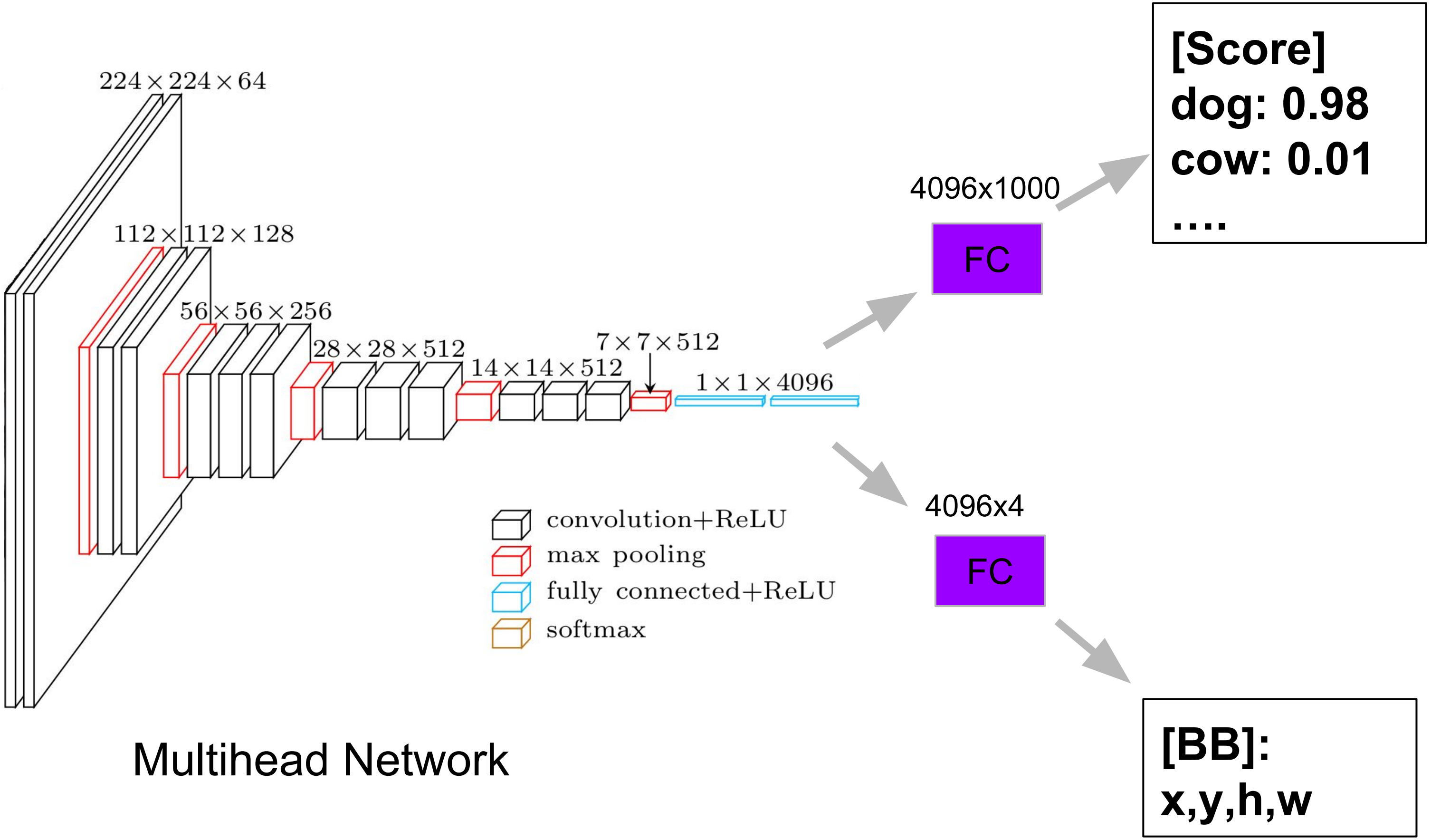
dog: 0.98
cow: 0.01
....

Image Classification -> Classification + Localization



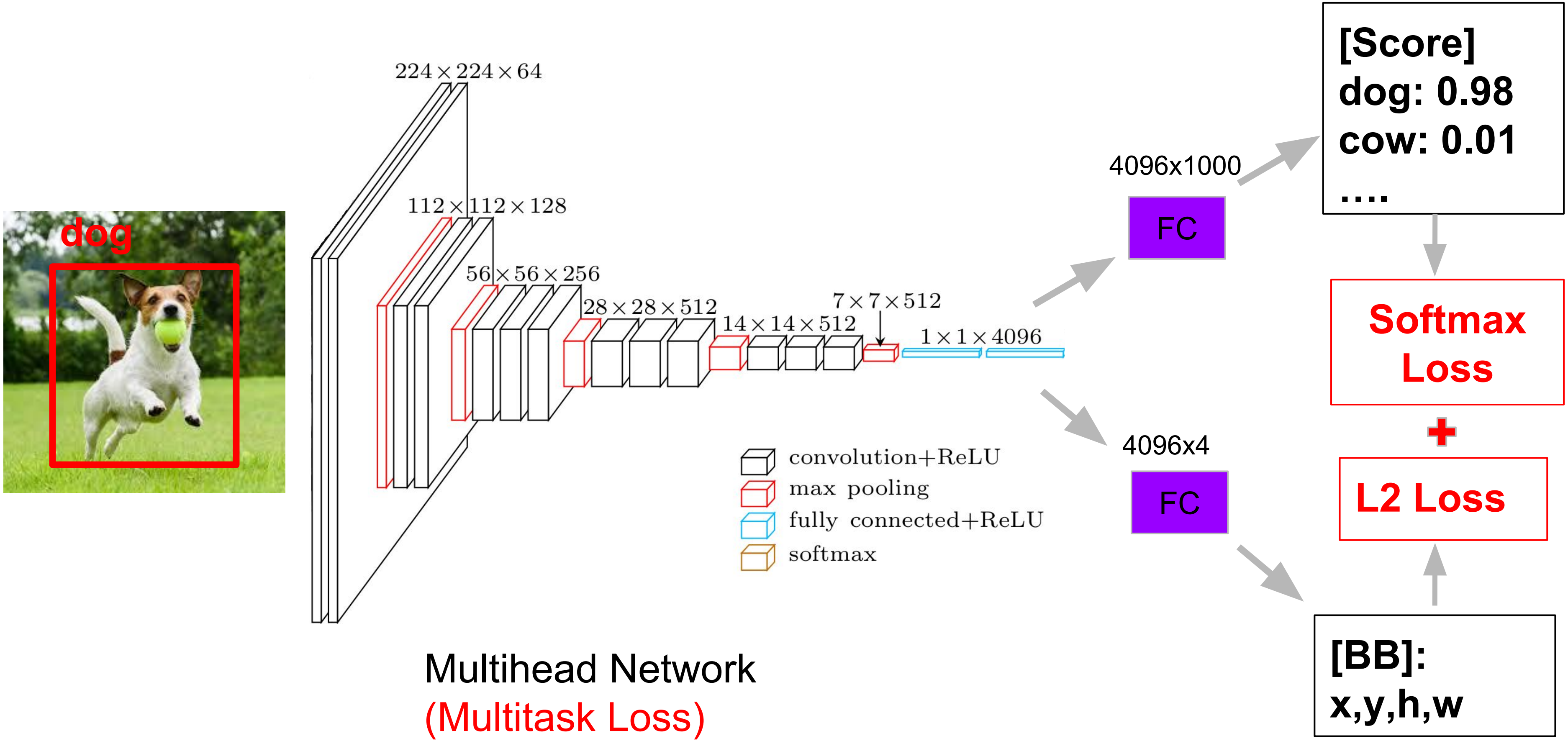
dog: 0.98
cow: 0.01
....

Image Classification -> Classification + Localization

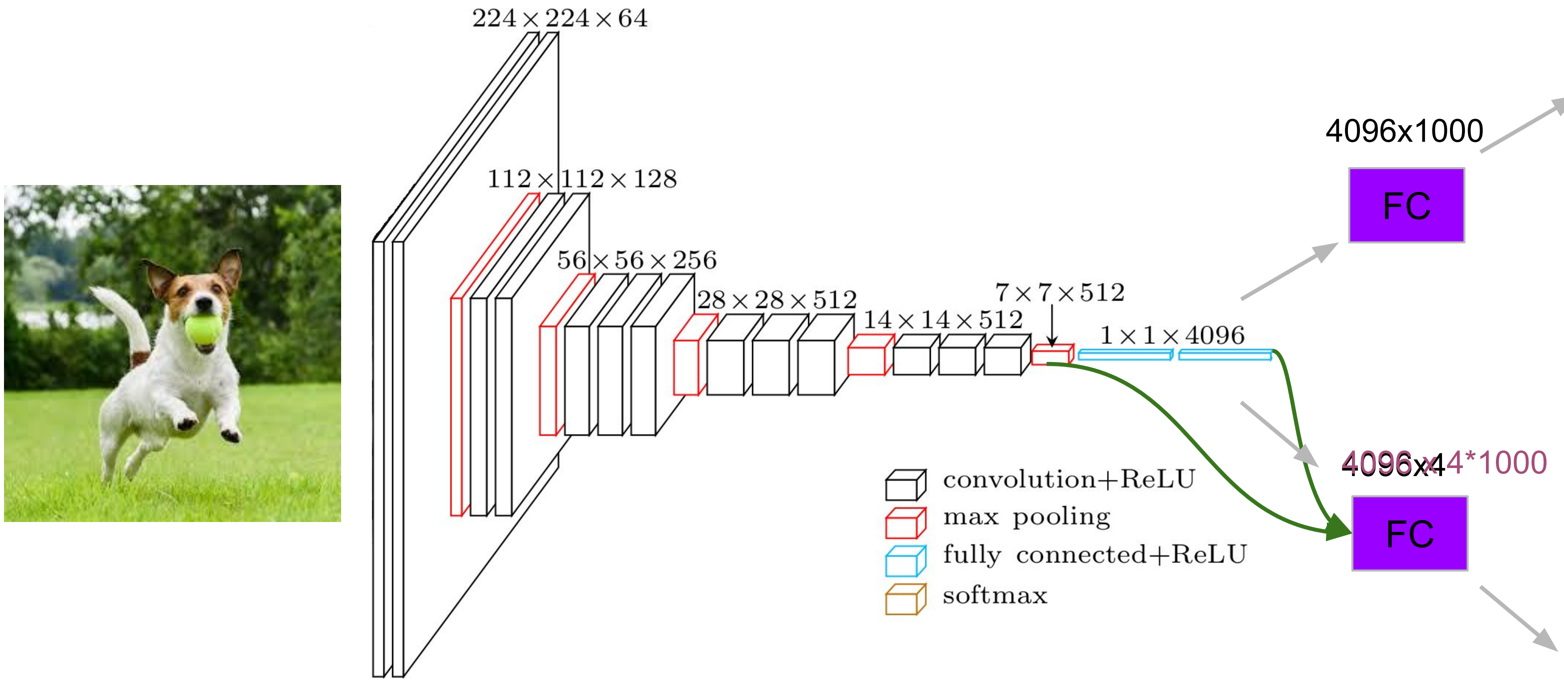


Multihead Network

Localization [Training]



Localization [Variations]



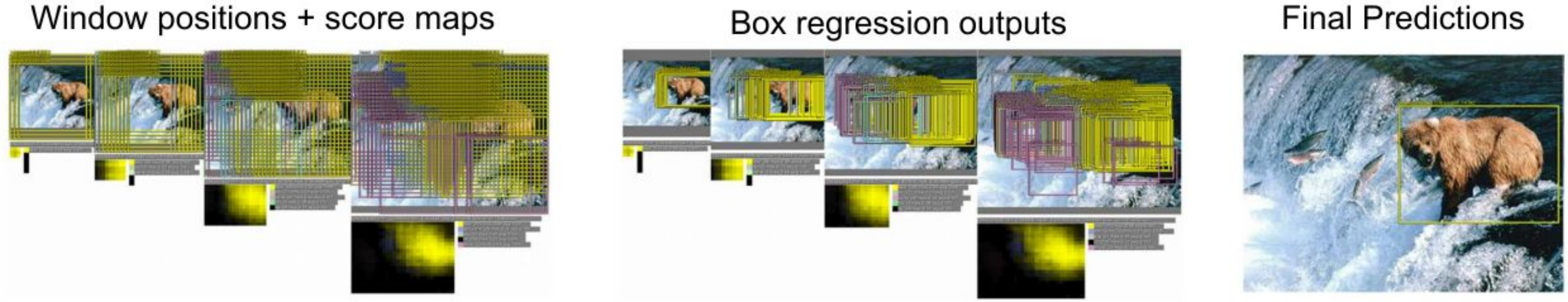
[Score]
dog: 0.98
cow: 0.01
....

[BB]:
x,y,h,w

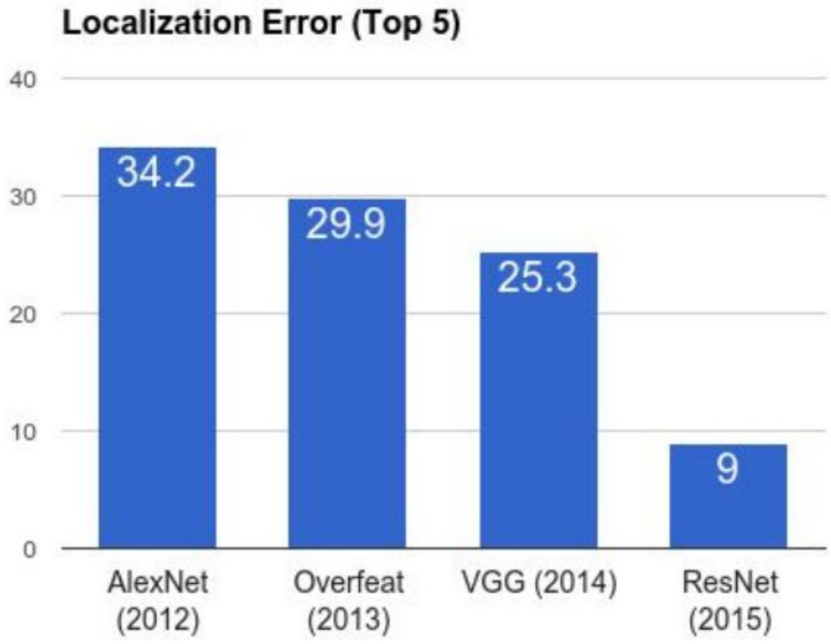
Where to attach the regression head?

Class agnostic vs. Class specific (one box per class)

Localization [Sliding Window]



Localization [Results on ImageNet]



AlexNet: Localization method not published

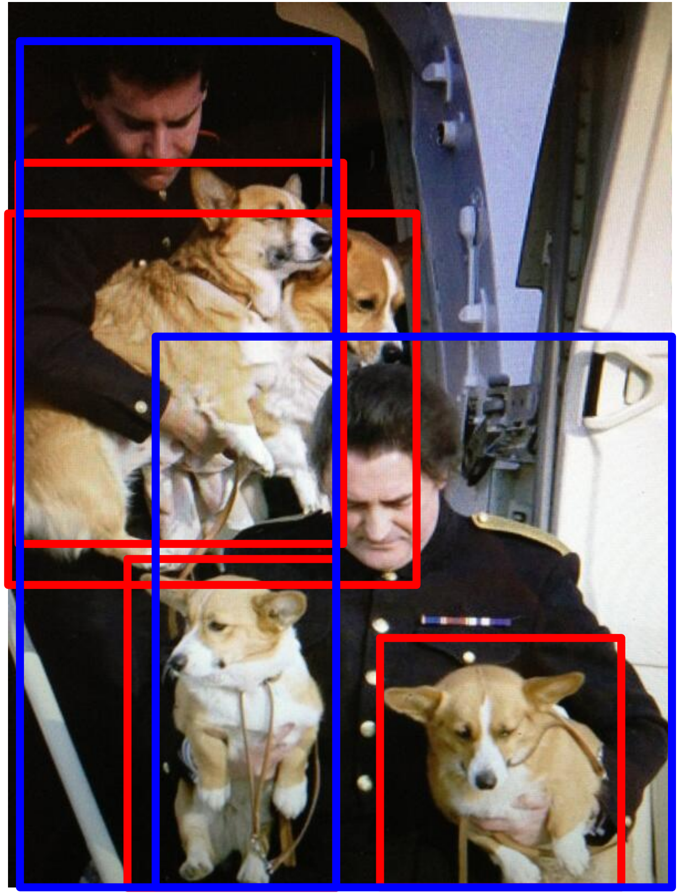
Overfeat: Multiscale convolutional regression with box merging

VGG: Same as Overfeat, but fewer scales and locations; simpler method, gains all due to deeper features

ResNet: Different localization method (RPN) and much deeper features

Contents

Object Detection



Object Detection [Datasets]

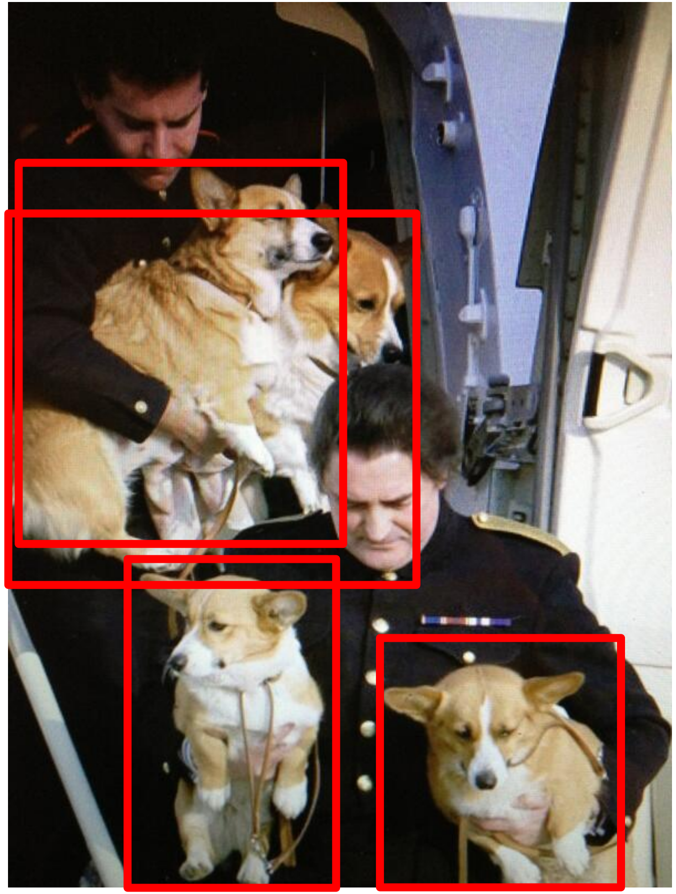
Name	# Images (trainval)	# Classes	Last updated
Open Images Dataset V4	1.74M	600	2018
ImageNet	450k	200	2015
COCO	120K	80	2014
KITTI Vision	7K	3	2014
Pascal VOC	12K	20	2012

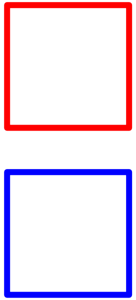
Object Detection [Metrics]

$$IoU = \frac{A \cap B}{A \cup B}$$

A - ground truth (GT)
B - detector result (P)

Object Detection [Metrics]

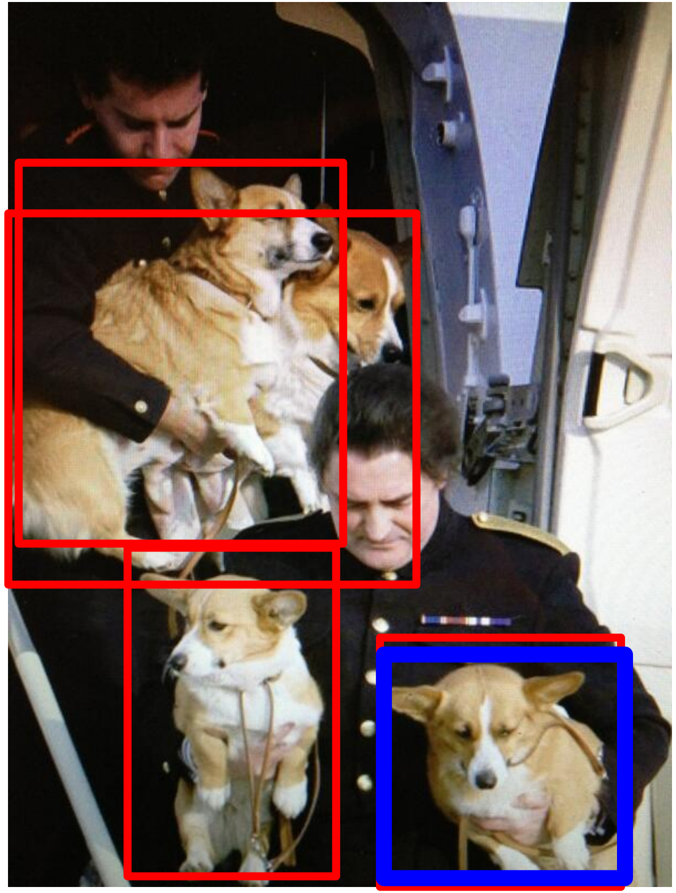




GT

P

Object Detection [Metrics]



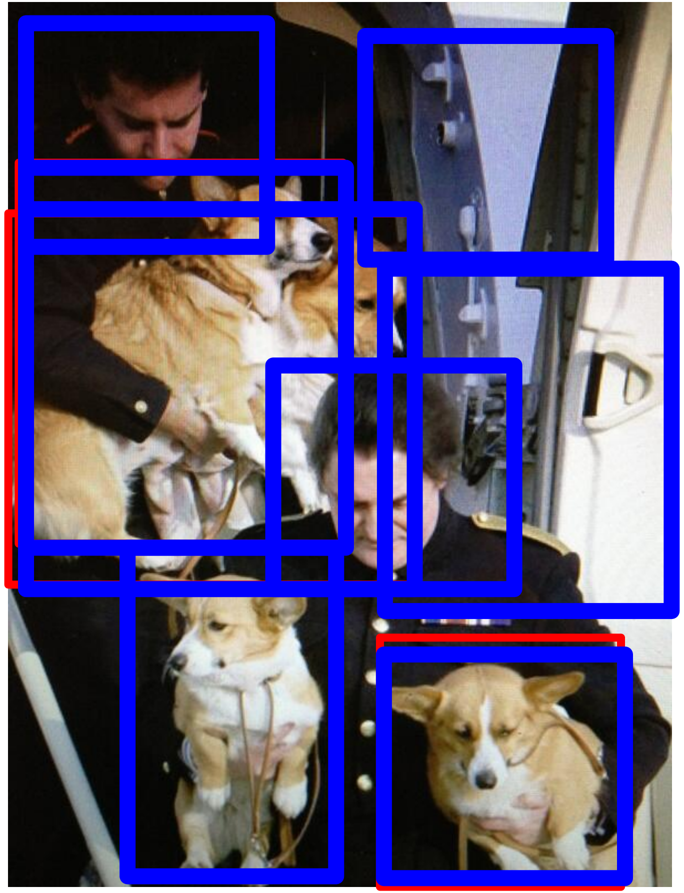
GT

P

Precision ?

Recall ?

Object Detection [Metrics]

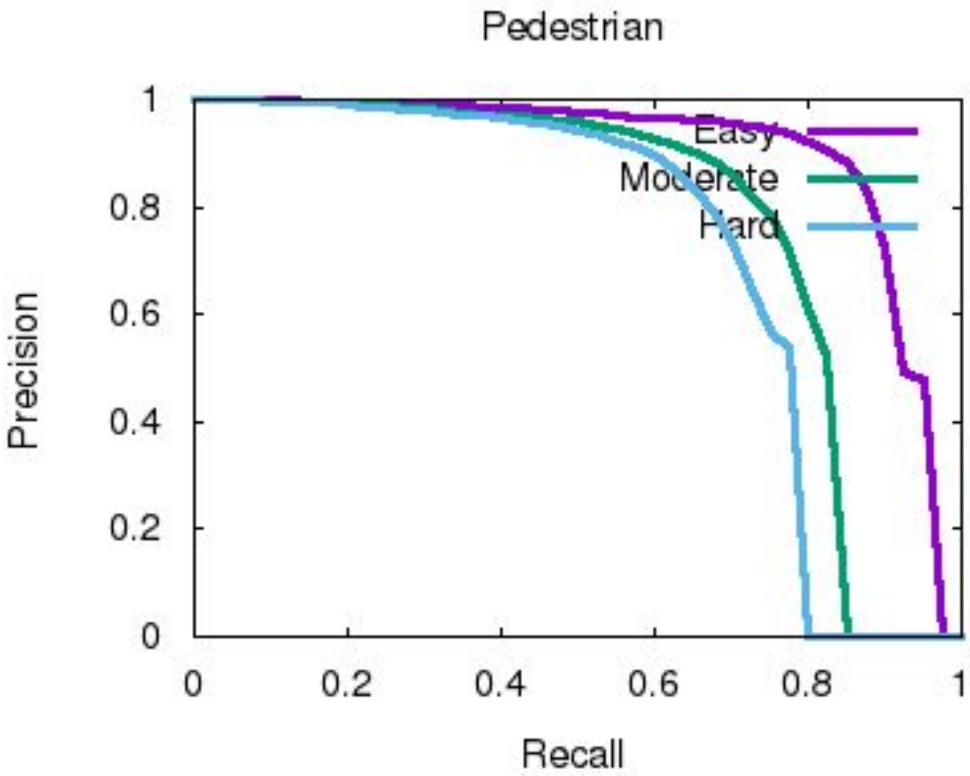


GT

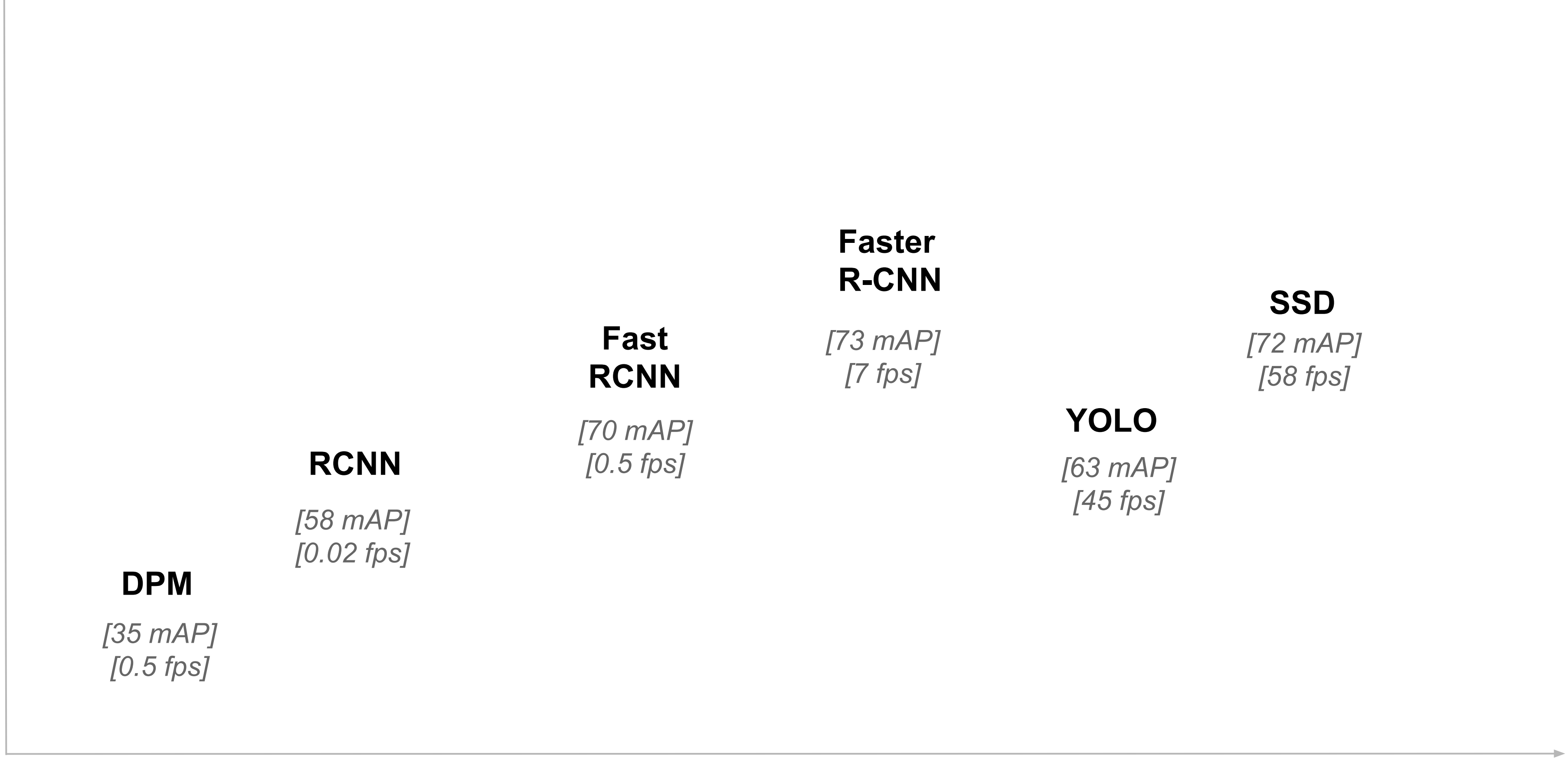
P

Precision ?
Recall ?

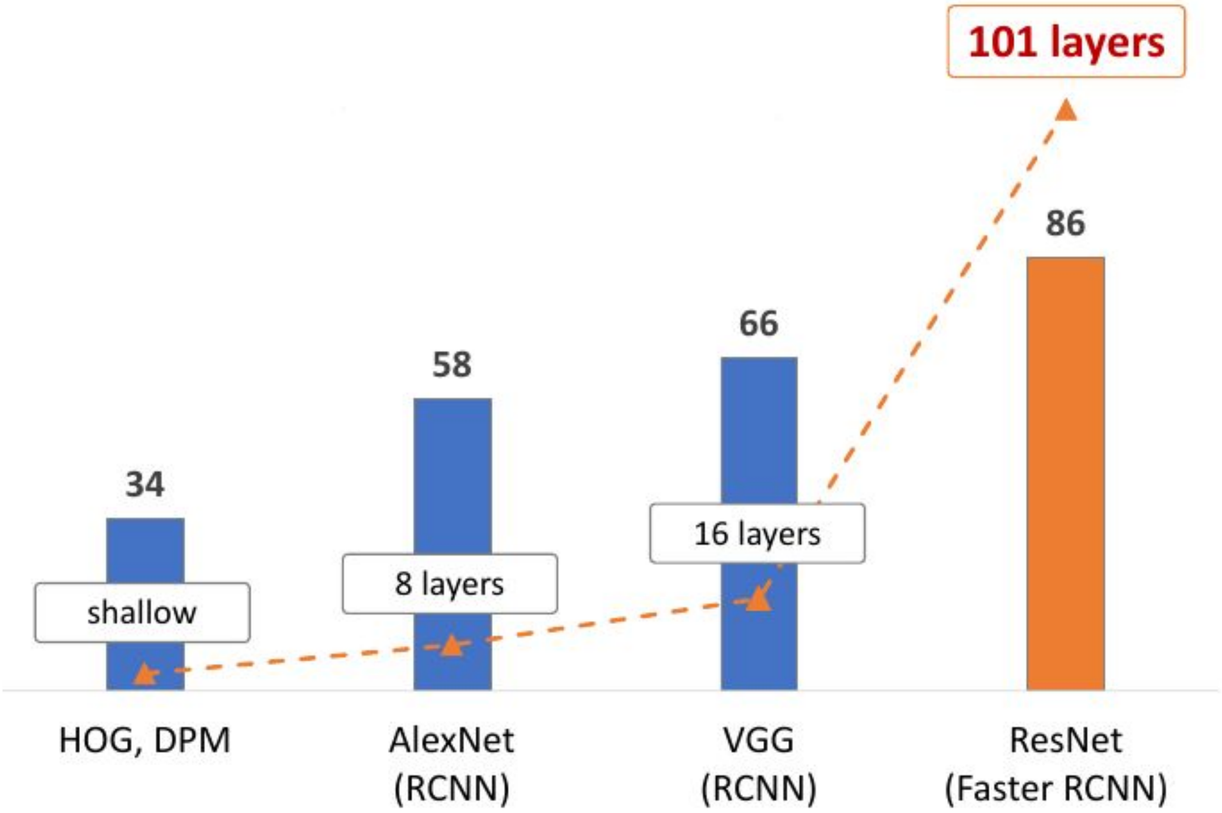
Object Detection [Metrics]



Object Detection [PASCAL VOC 2007 Results, mAP %]



Object Detection [PASCAL VOC 2007 Results, mAP %]

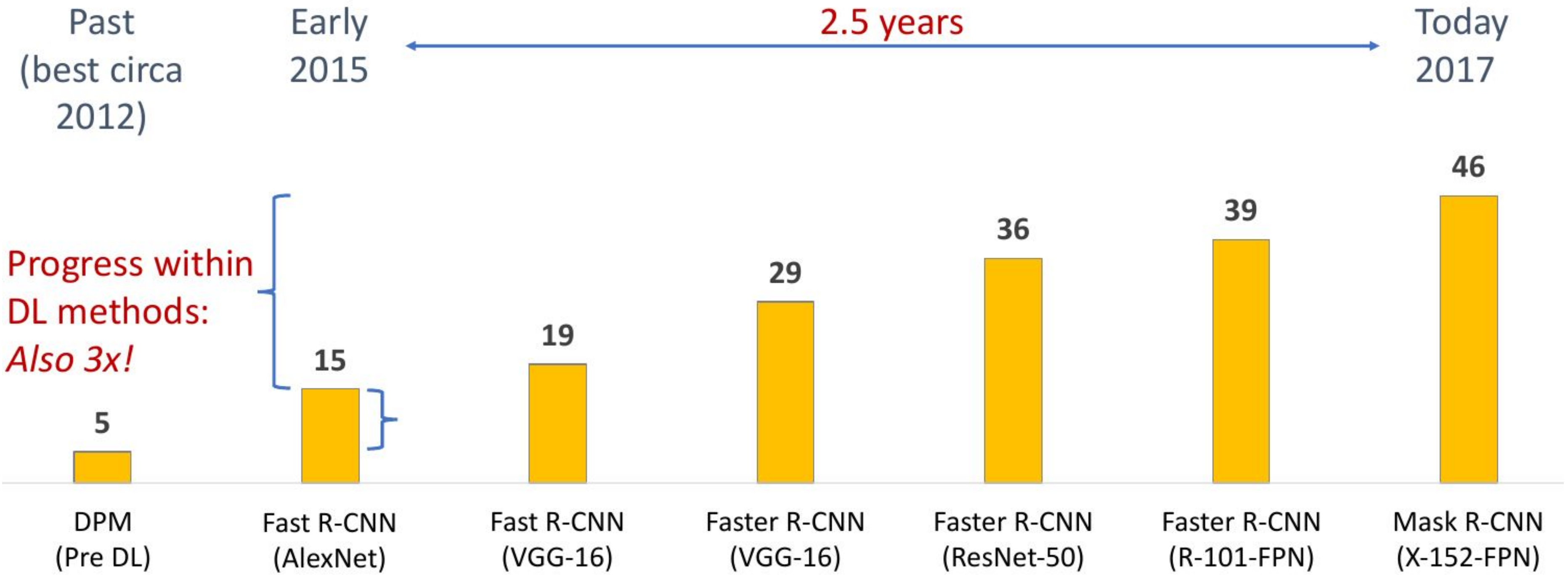


Object Detection [MS COCO Results, AP %]

Past (best circa 2012) | Early 2015

Movement to
Deep Learning methods:
3x improvement in AP

Object Detection [COCO Results, AP %]



ICCV 2017 Ross Girshick Tutorial

Contents

Faster

Fast R-CNN
(Regions with CNN features)

2013 arXiv:1311.2524

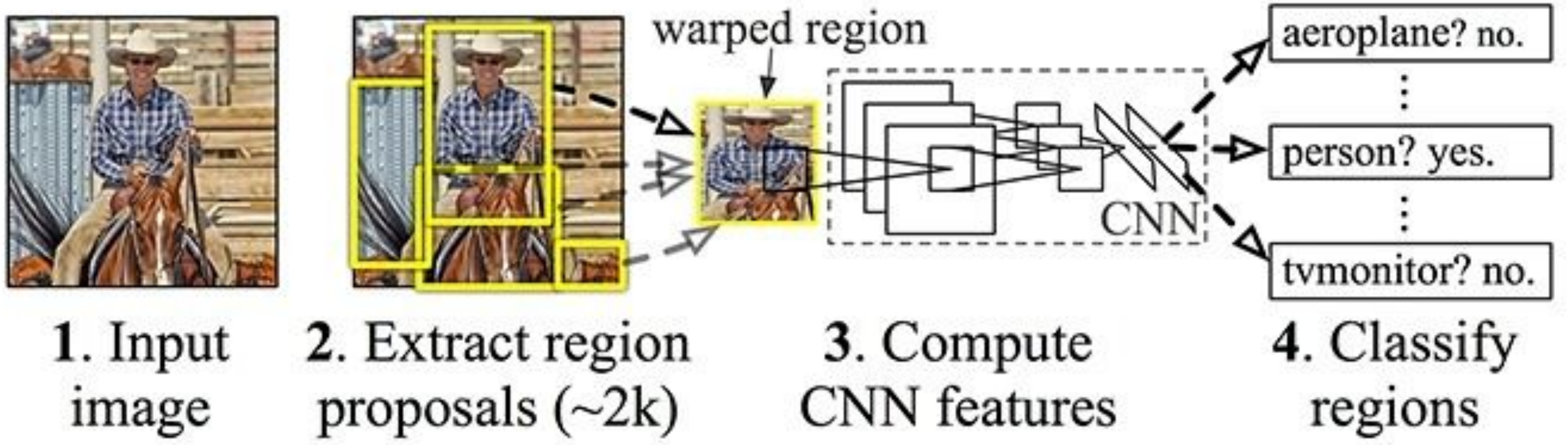
2015 arXiv:1504.08083

2015 arXiv:1506.01497

Ross Girshick
[Facebook AI Research (FAIR)]

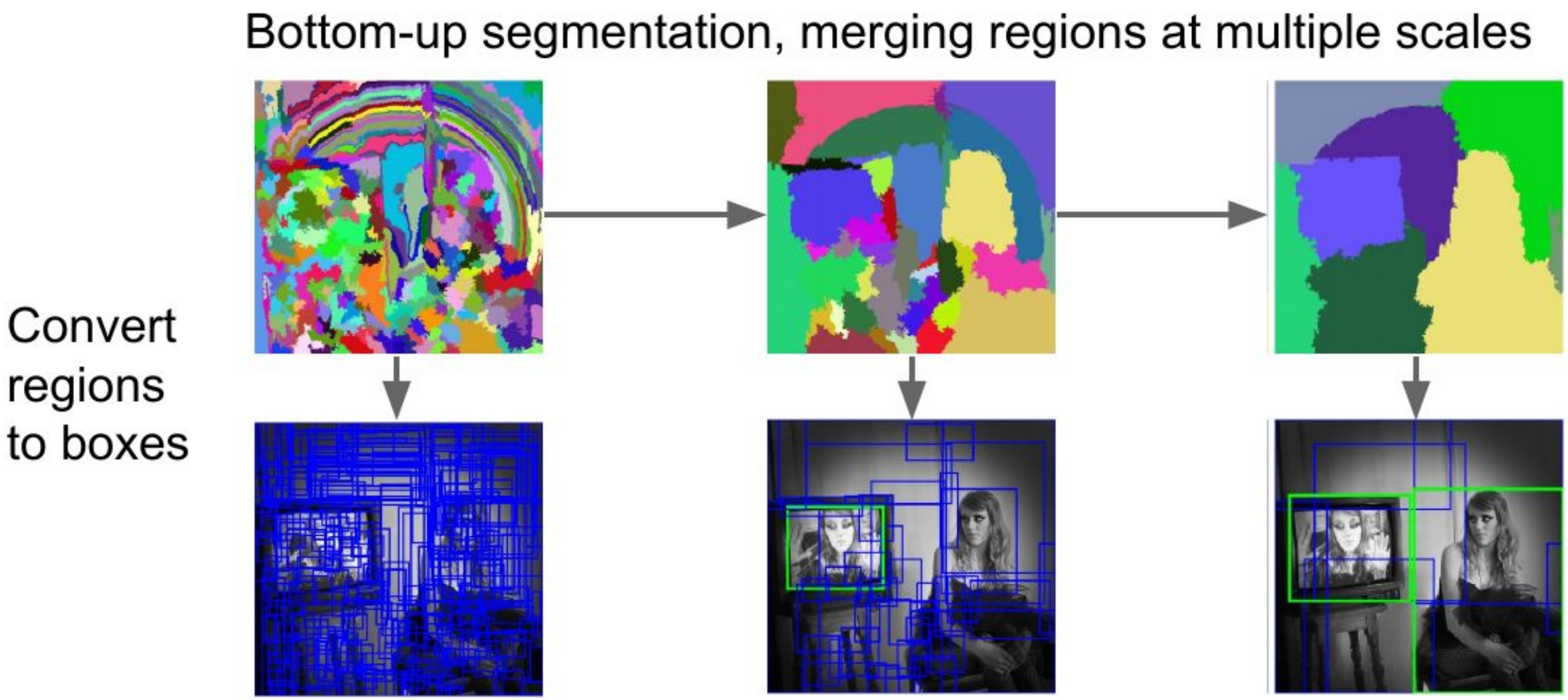
http://www.rossgirshick.info/

R-CNN: Regions with CNN features



- Extract possible objects using a region proposal method (the most popular one being Selective Search).
- Extract features from each region using a CNN.
- Classify each region with SVMs.

Object Detection [Region Proposal]



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

R-CNN Summary

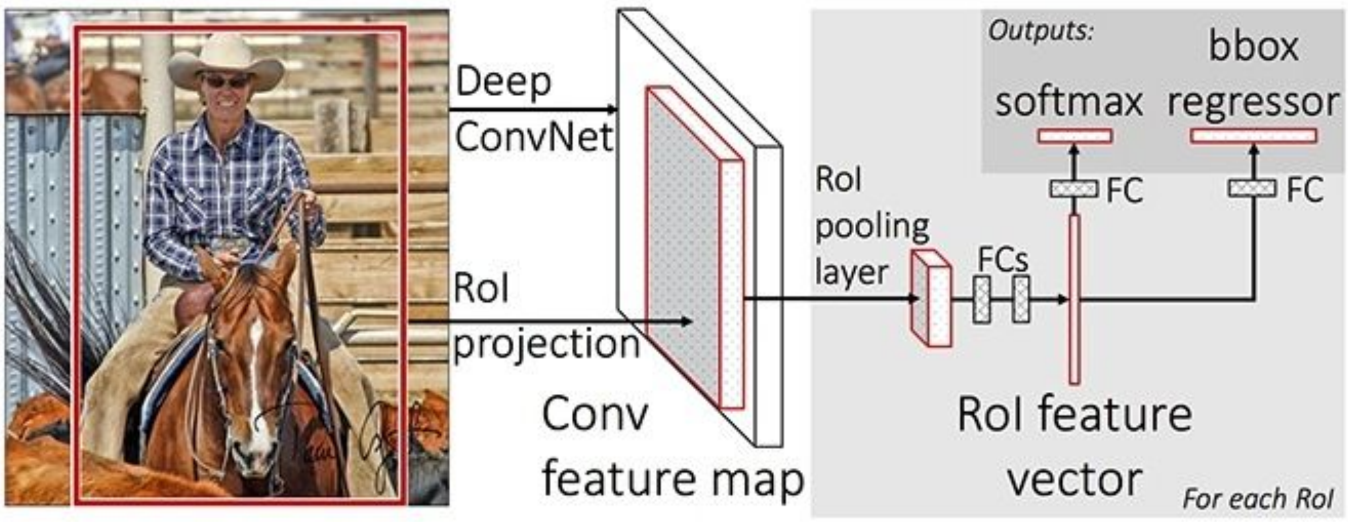
Pros:

- generalization of CNN features for detection problem;
- almost 50% improvement on the object detection challenge;
- “when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost”

Cons:

- time & memory inefficiency;
- complex training;
- relied upon external region proposal generator (Selective Search);

Fast R-CNN



- Use Selective Search to generate object proposals;
- Apply the CNN on the complete image and then used both **Region of Interest (RoI) Pooling** on the feature map;
- Use feed forward network for classification and regression;

Fast R-CNN Summary

Pros:

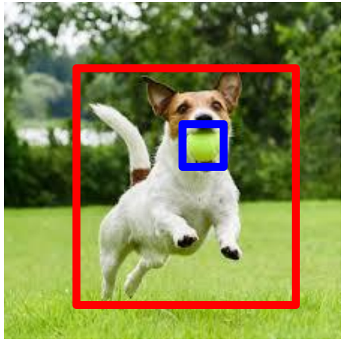
- end-to-end differentiable model, which is easier to train;
- much faster training time (Fast R-CNN trains the very deep VGG16 network 9x faster than R-CNN, is 213x faster at test-time);
- better accuracy than R-CNN

Cons:

- relied upon external region proposal generator (Selective Search), similar to R-CNN;

Contents

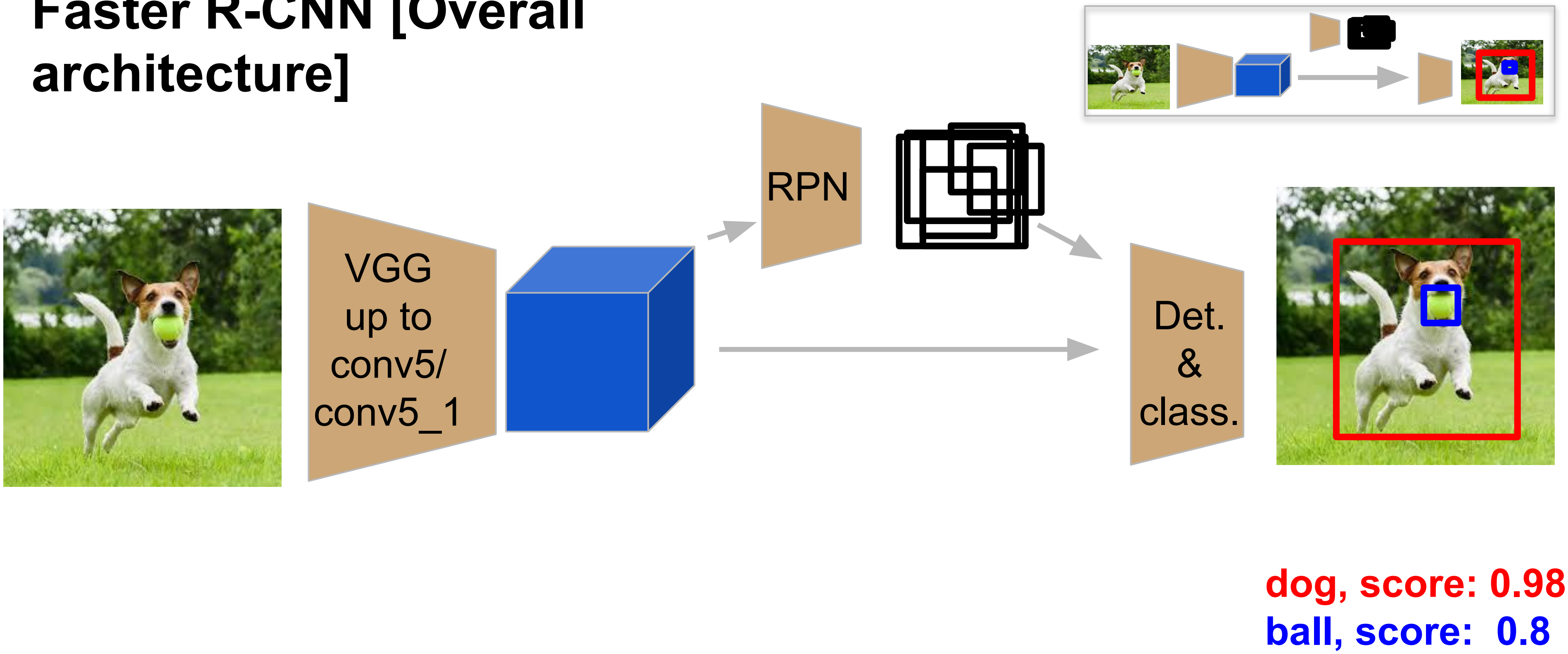
Faster R-CNN (Overall architecture)



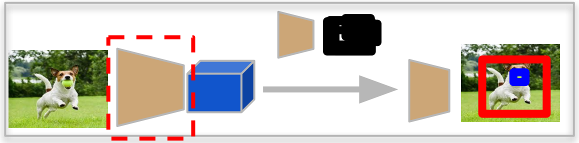
dog, score: 0.98

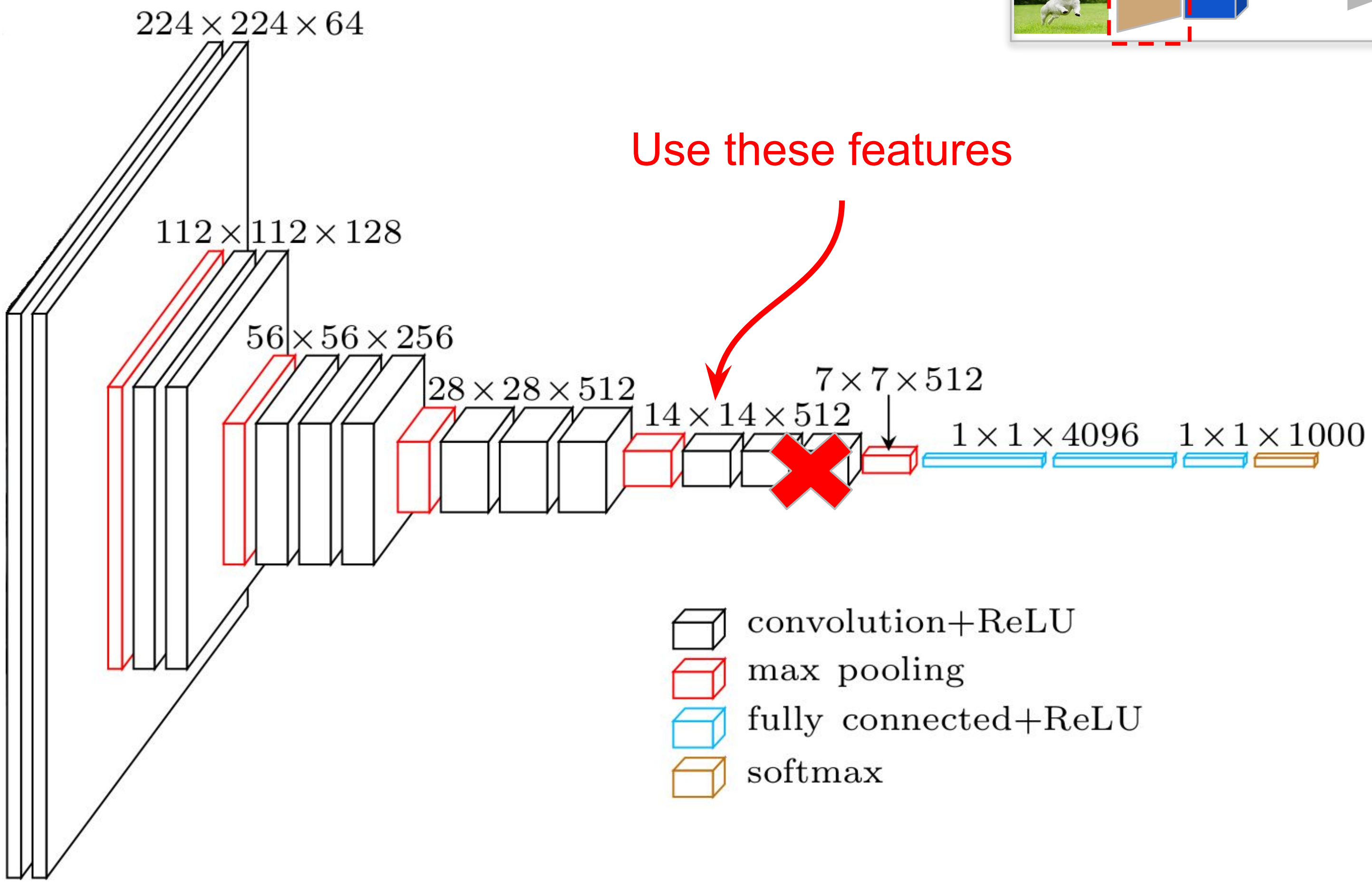
ball, score: 0.6

Faster R-CNN [Overall architecture]

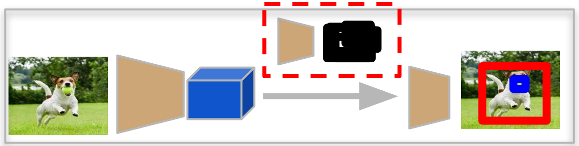


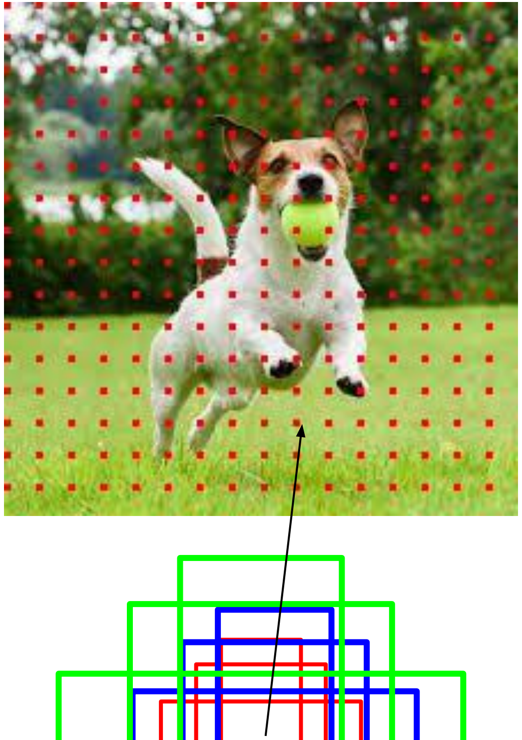
Faster R-CNN [Base Network]





Faster R-CNN [Anchors]

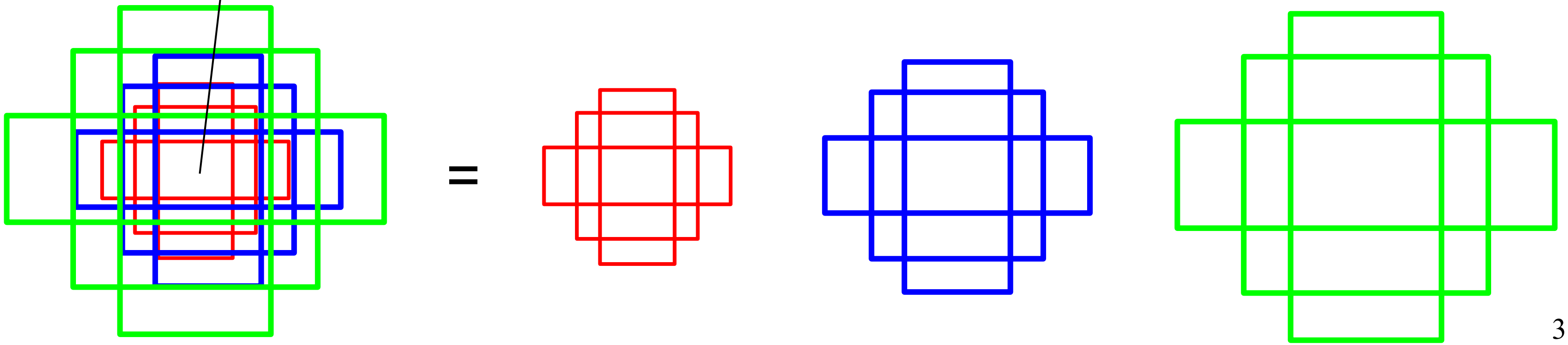




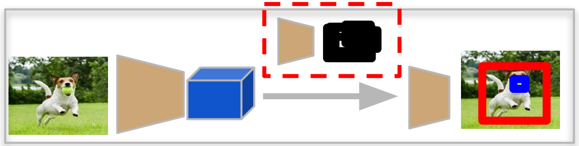
Anchors are fixed bounding boxes that are placed throughout the image with different sizes and ratios that are going to be used for reference when first predicting object locations.

- set of sizes (e.g. 64px, 128px, 256px)
- set of ratios between width and height of boxes (e.g. 0.5, 1, 1.5)

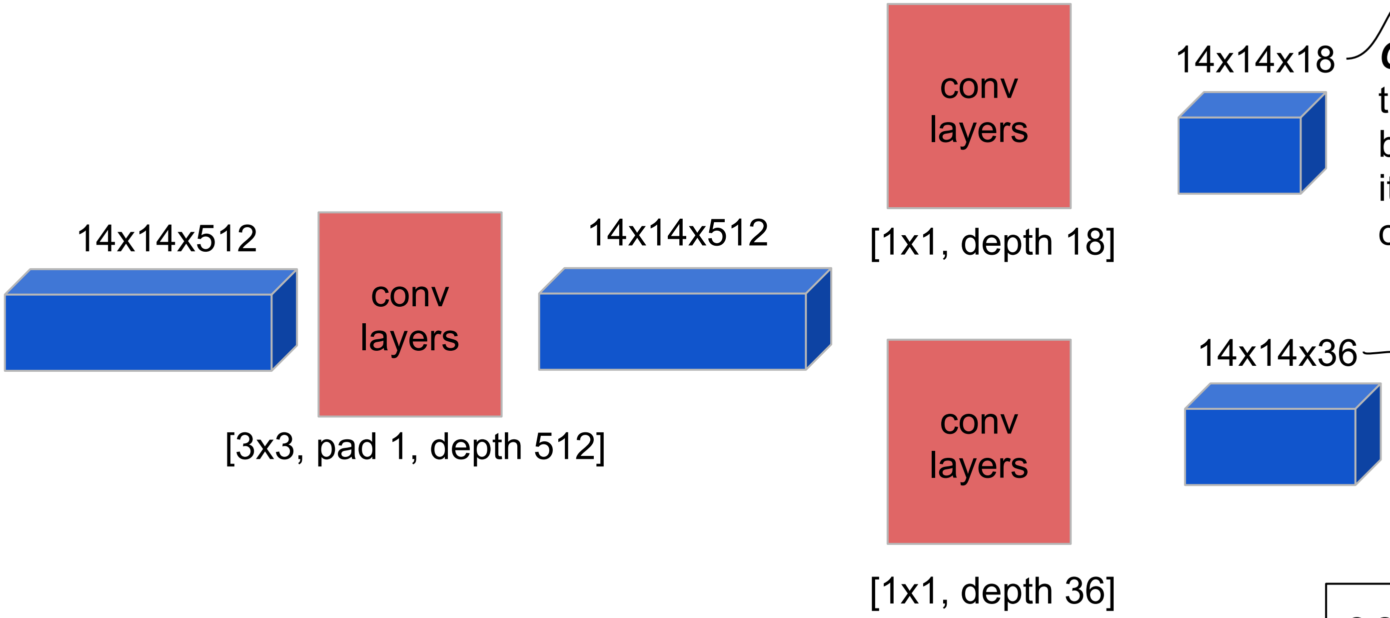
= 9



Faster R-CNN [Region Proposal Network]



RPN takes all the reference boxes (anchors) and outputs a set of good proposals for objects.



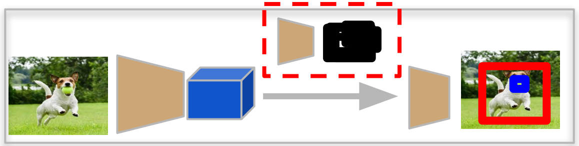
18 = 2 * 9 = **2 * k**

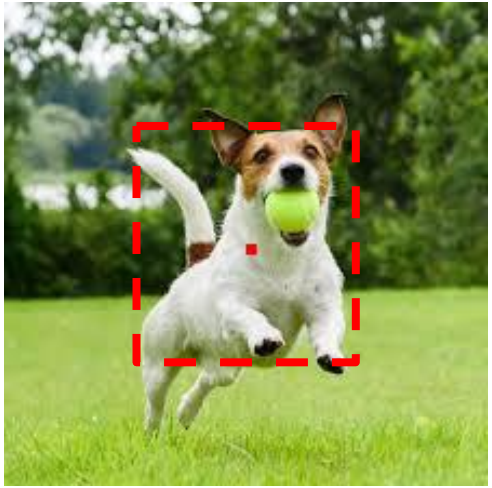
Classification layer
the score of it being background and the score of it being foreground (an actual object)

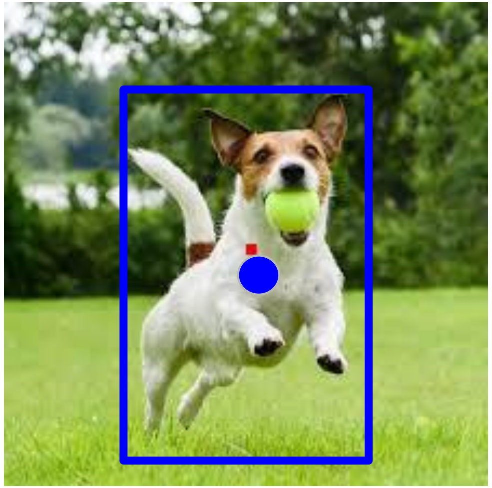
Regression layer
Δxcenter, Δycenter, Δw, Δh, which will be applied to the anchors to get the final proposals.

36 = 4 * 9 = **4 * k**

Faster R-CNN [Region Proposal Network]







RPN output:

score (object) = 0.9
score (background) = 0.1

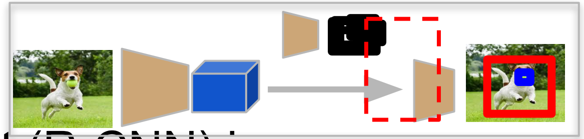
Δxcenter = 3
Δycenter = 10
Δw = 10
Δh = 40

default anchor position and size: *Xa, Ya, Wa, Ha*

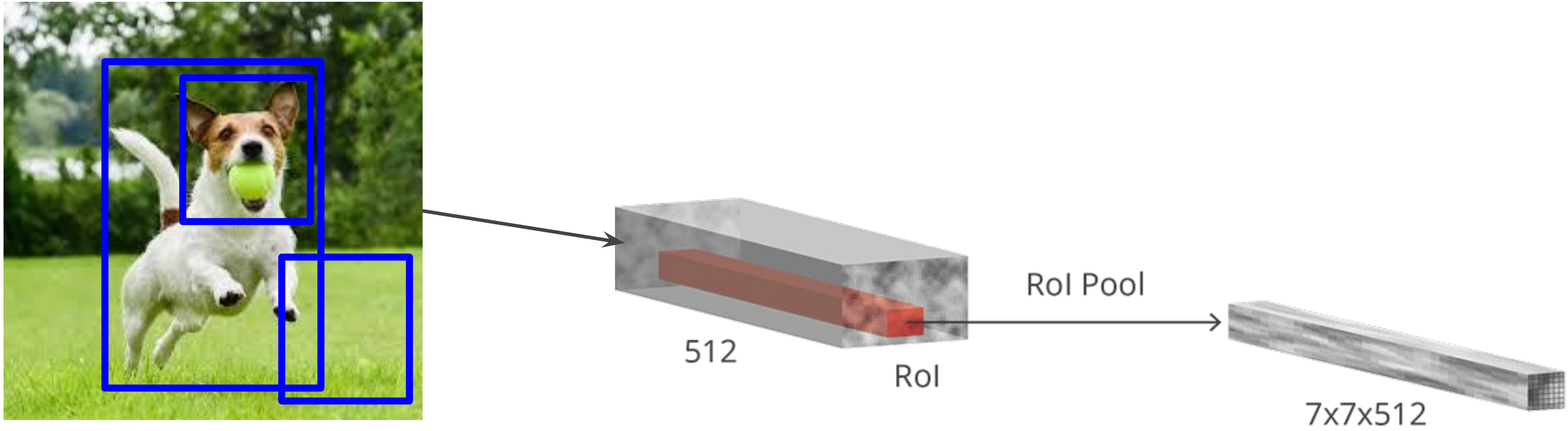
$$X = \Delta xcenter * Wa + Xa$$
$$Y = \Delta ycenter * Wa + Ya$$
$$W = Wa * \exp(\Delta w)$$
$$H = Ha * \exp(\Delta h)$$

proposal box position and size: *X, Y, W, H*

Faster R-CNN [Region of Interest Pooling]

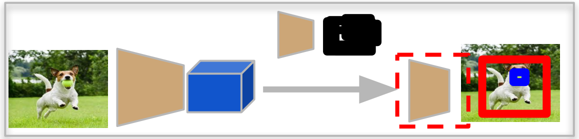


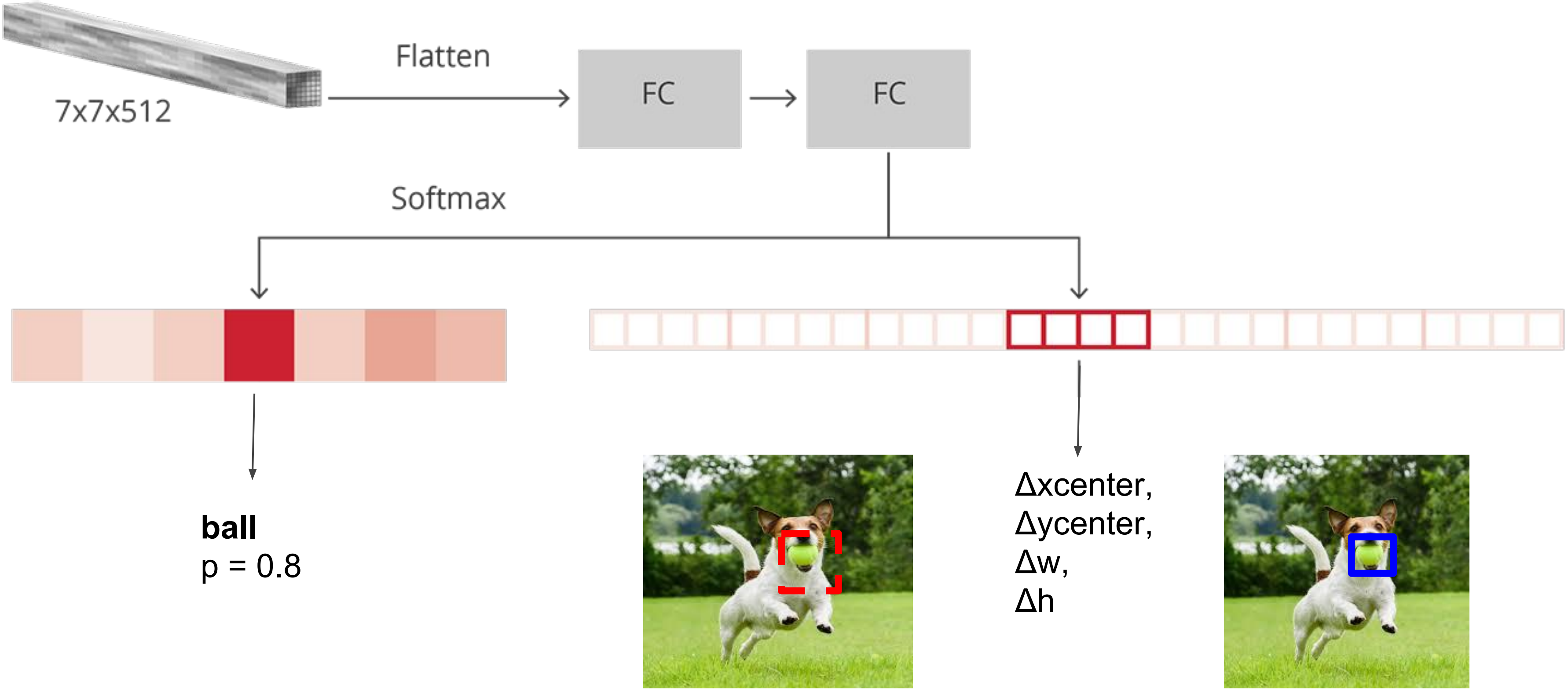
Fixed size feature maps are needed for the det.&class. part (R-CNN) in order to classify them into a fixed number of classes.



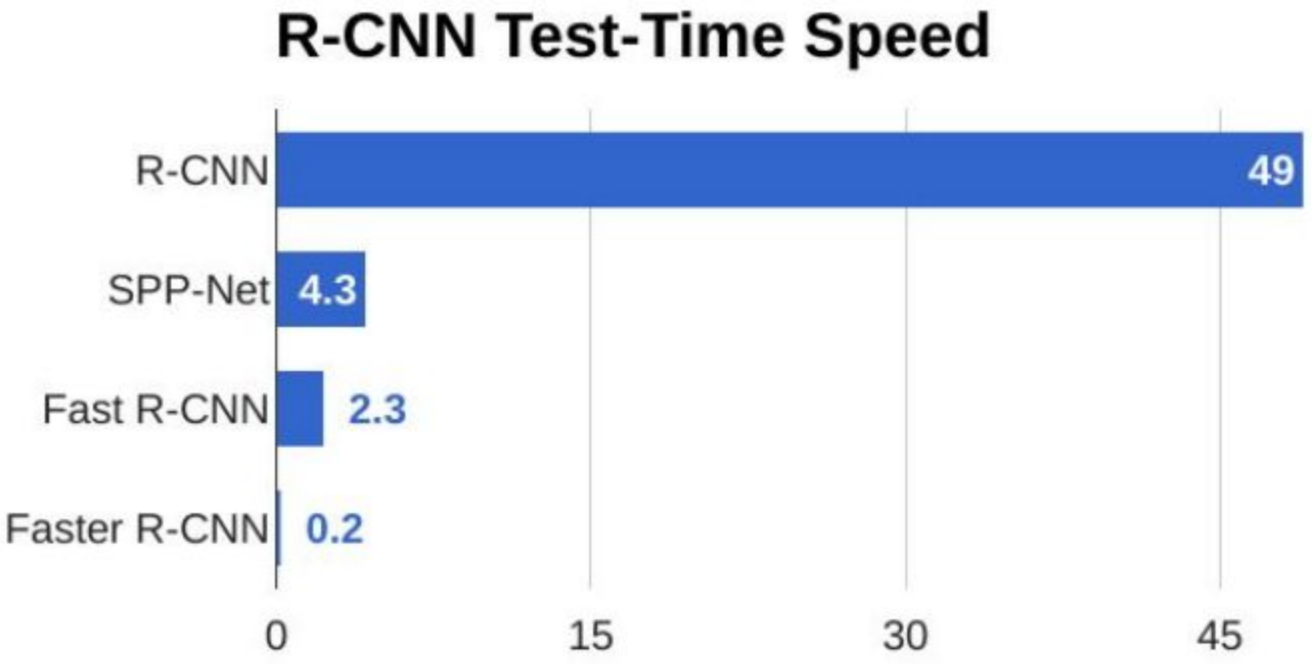
crop the convolutional features using each proposal and then resize to a fixed sized 14×14×D using interpolation (usually bilinear). After cropping, max pooling with a 2x2 kernel is used to get a final 7×7×D for each proposal.

Faster R-CNN [Region-based CNN]





Faster R-CNN [Time]



Faster R-CNN [Training]

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

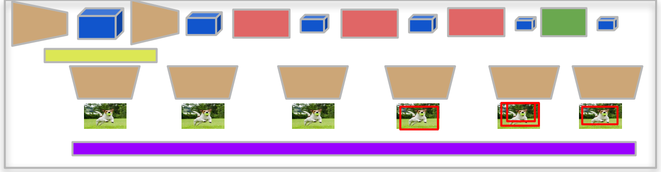
mini batch of size 256 — trying to maintain a balanced ratio between foreground and background anchors.

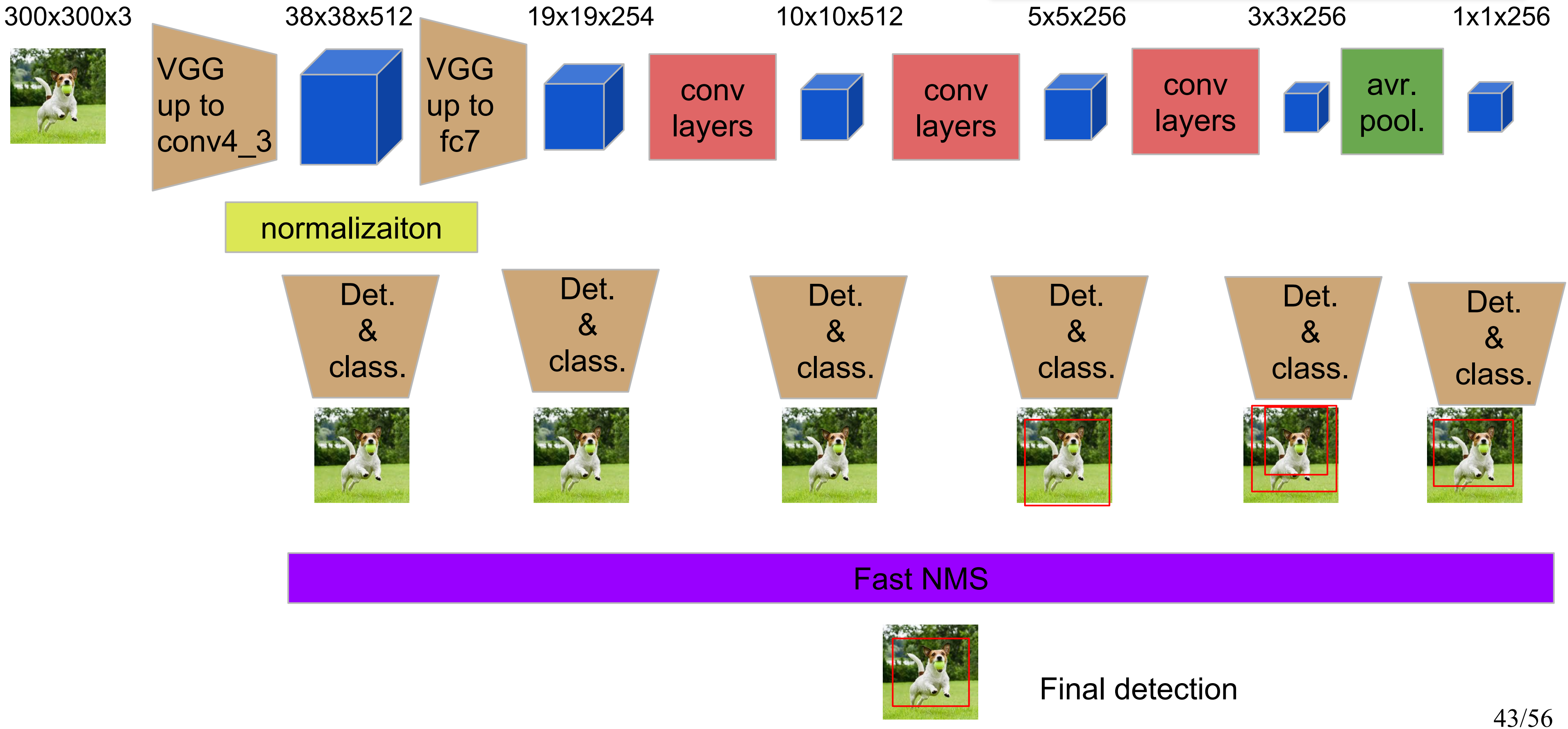
The RPN uses all the anchors selected for the mini batch to calculate the classification loss using binary cross entropy. Then, it uses only those minibatch anchors marked as foreground to calculate the regression loss.

for the regression error, the paper uses Smooth L1 loss. Smooth L1 is basically L1, but when the L1 error is small enough, defined by a certain σ, the error is considered almost correct and the loss diminishes at a faster rate.

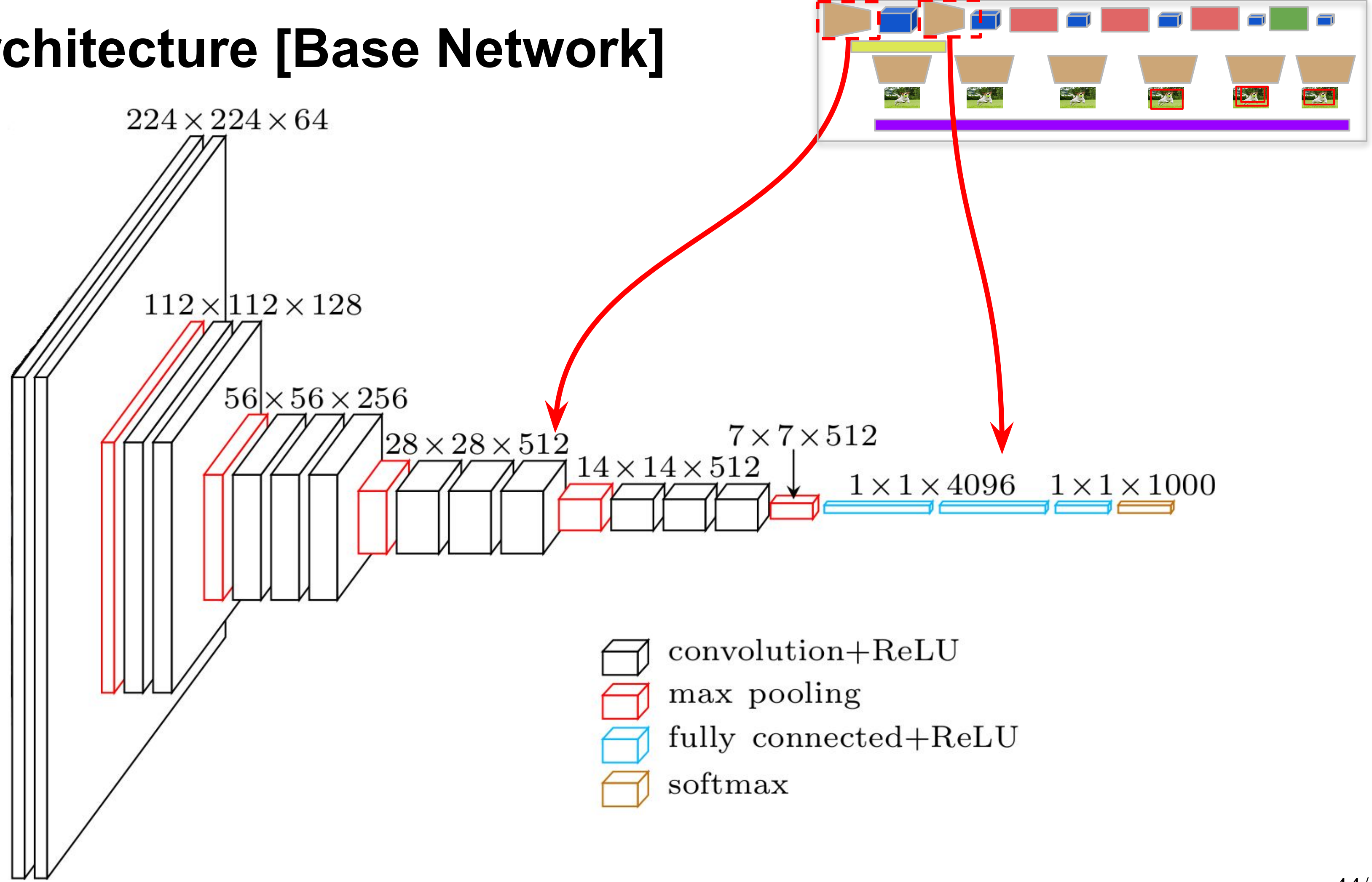
Contents

SSD architecture

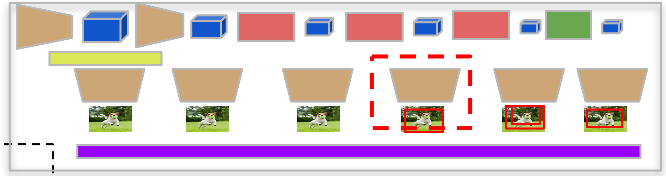


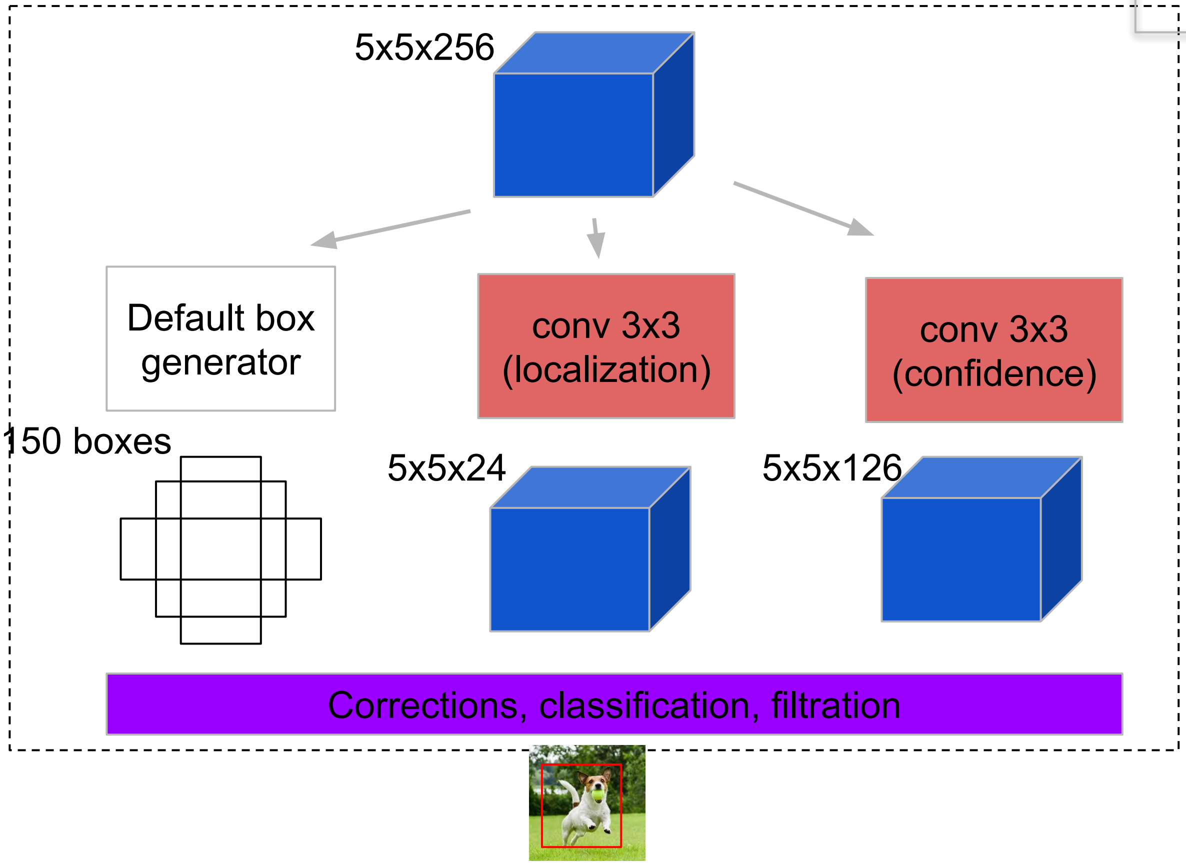


SSD architecture [Base Network]



SSD architecture [Detector & classifier]

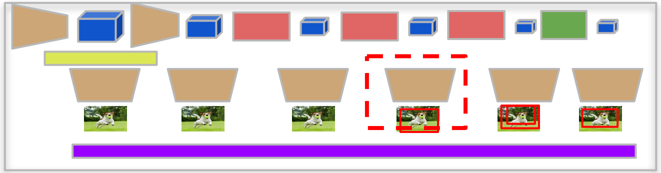


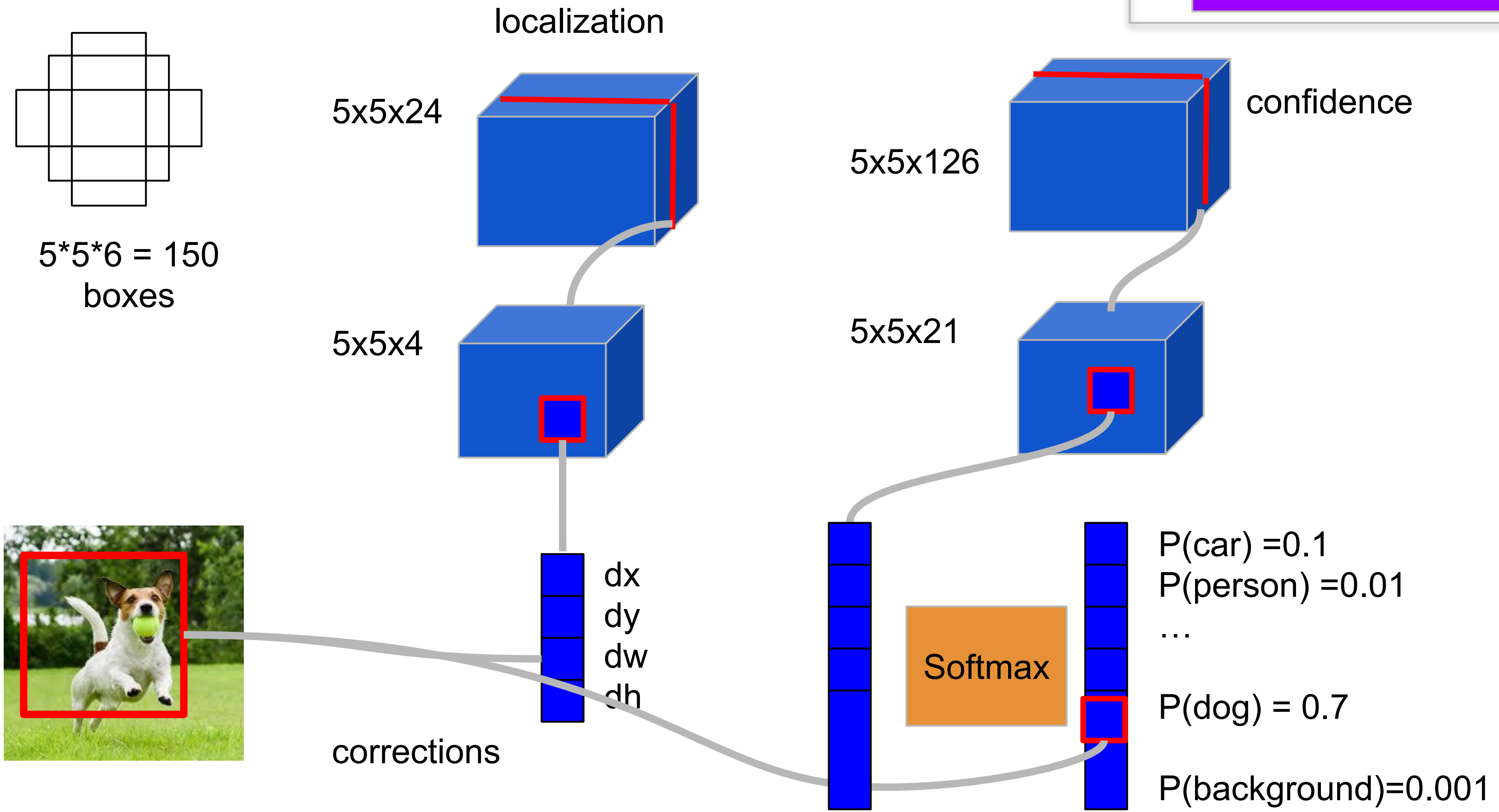


- # default boxes = 6
- 20 + 1 classes
- 4 numbers for BB

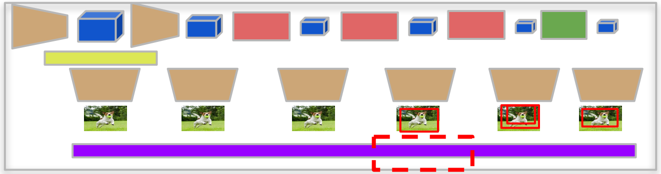
- 126 = 21 x 6
- 24 = 4 x 6
- 150 = 5 x 5 x 6

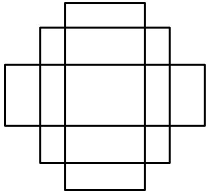
SSD architecture. [Corrections]



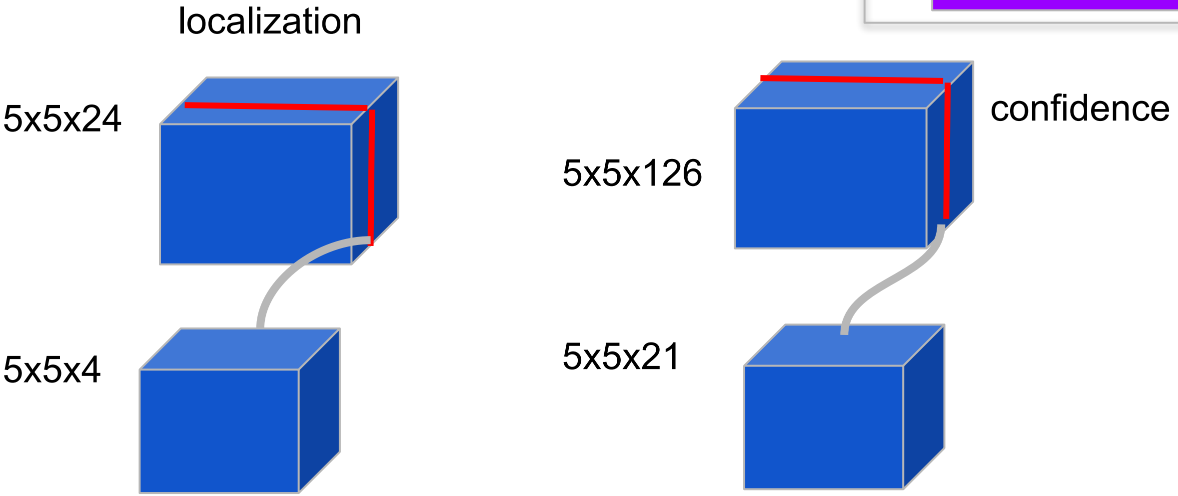


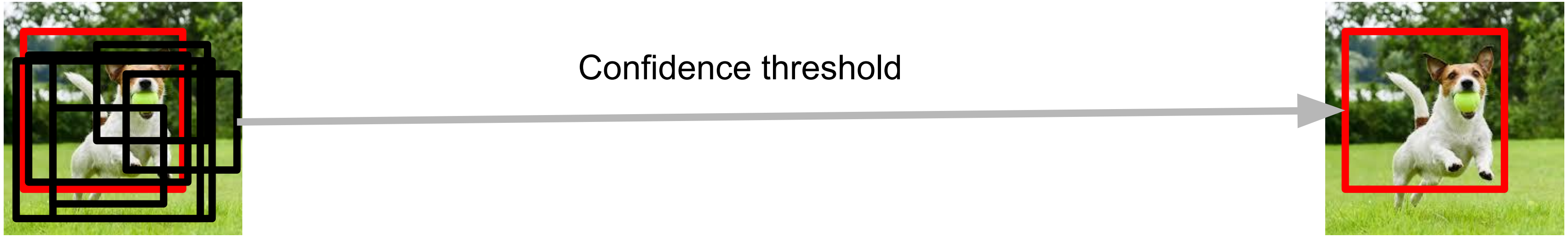
SSD architecture. [Corrections]



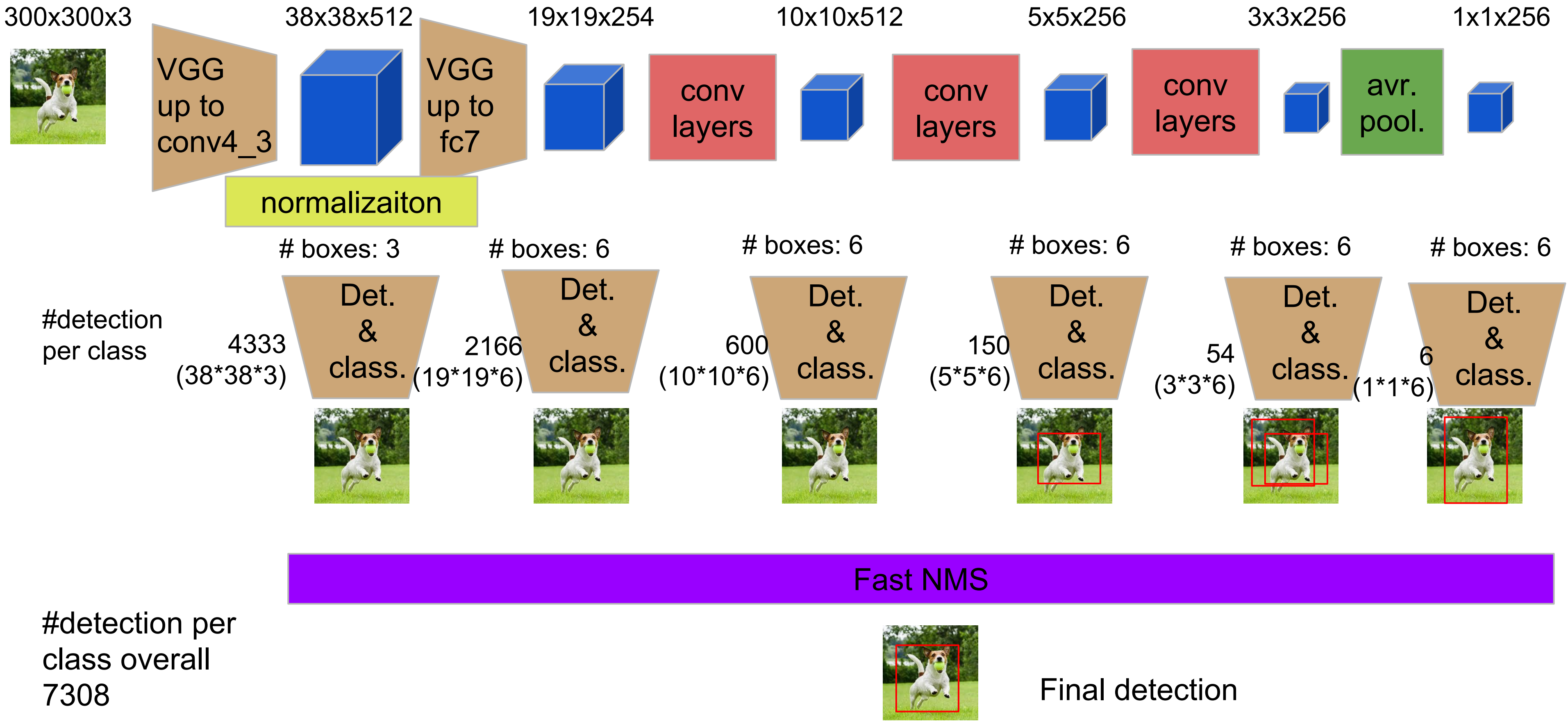


5*5*6 = 150
boxes

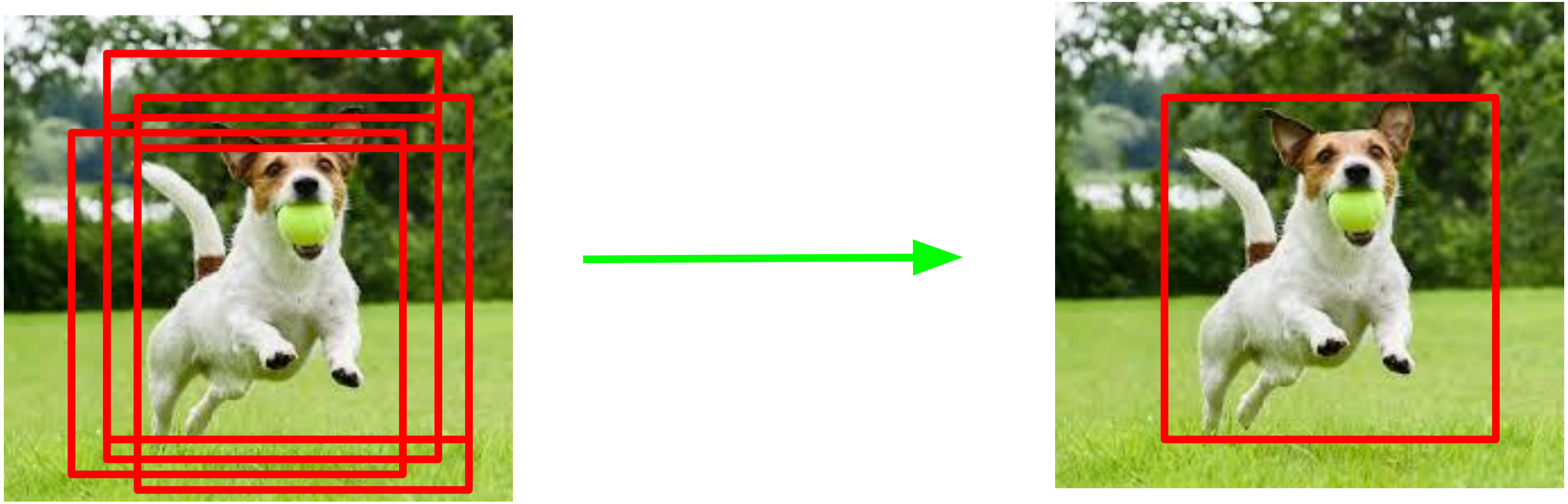




SSD architecture



Non-maximum Suppression



SSD [Training]

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$$

$$L_{conf}(x,c) = -\sum_{i\in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i\in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

$$L_{loc}(x,l,g) = \sum_{i\in Pos}^{N} \sum_{m\in\{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

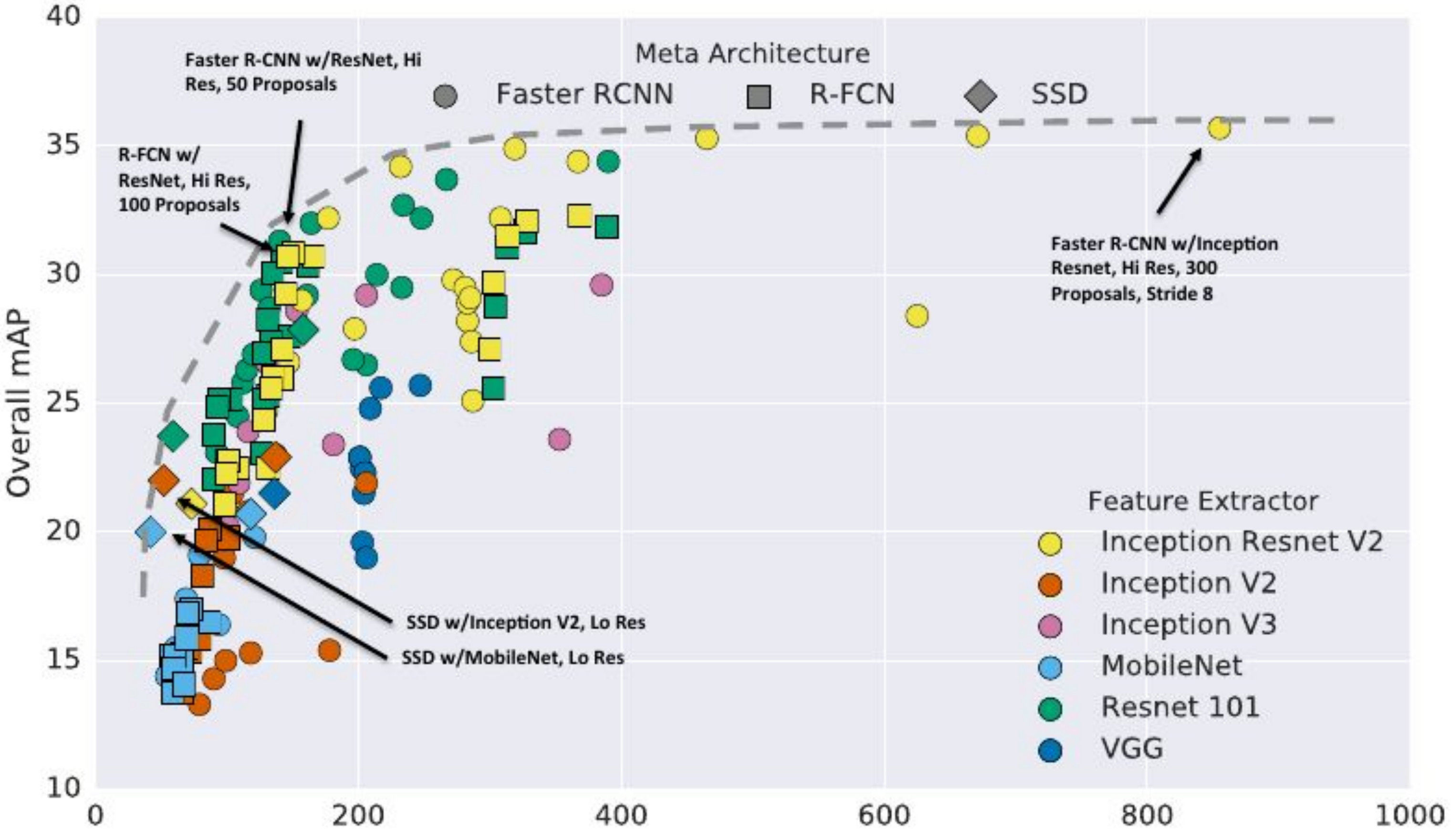
$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^{w} = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^{h} = \log\left(\frac{g_j^h}{d_i^h}\right)$$

Heavy use of data augmentation.

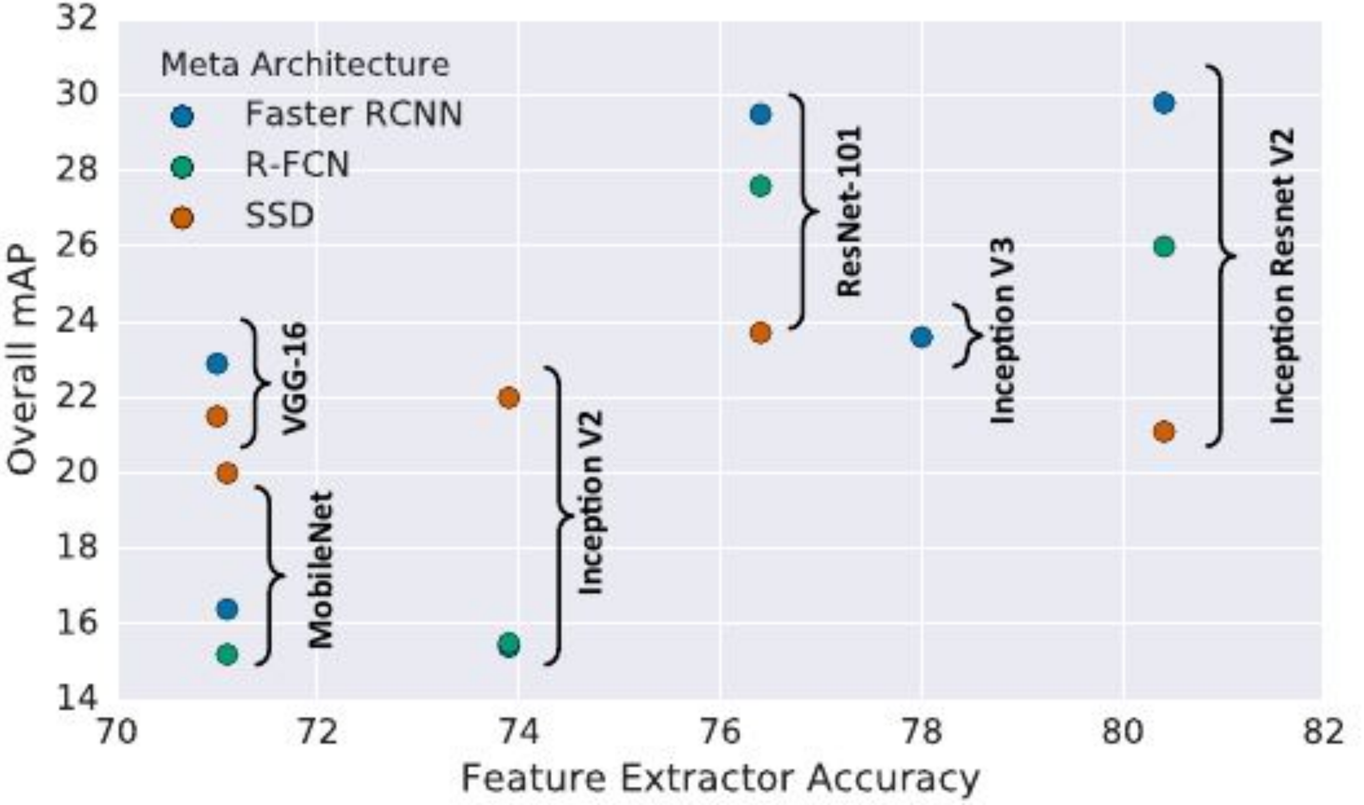
Contents

Object Detection [Comparison]



arXiv:1611.10012

Object Detection [Comparison]



arXiv:1611.10012

Object Detection [Comparison]

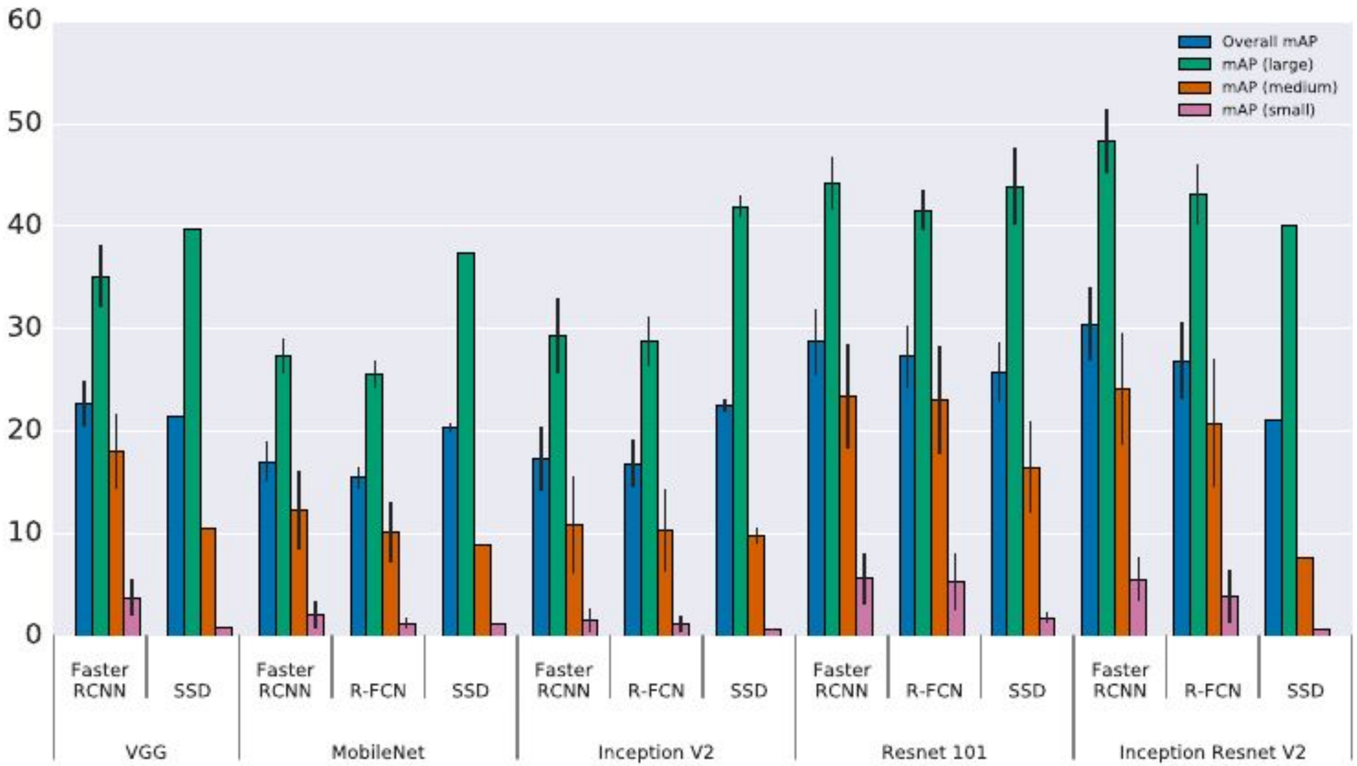


Figure 4: Accuracy stratified by object size, meta-architecture and feature extractor, We fix the image resolution to 300.

arXiv:1611.10012

Contents

Summary

Object Localization, Object Detection tasks

Two stage detector [Faster-RCNN]
- RPN
- End-to-end
- Best accuracy

One stage detector [SSD architecture]
- Use of multiple convolutional maps to deal with scales;
- More default bonding boxes -> the better the result;
- Comparable to Fast*-rCNN family but faster;