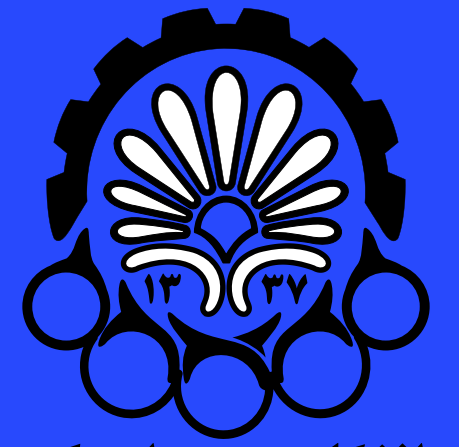
AI in robotic - Object detection

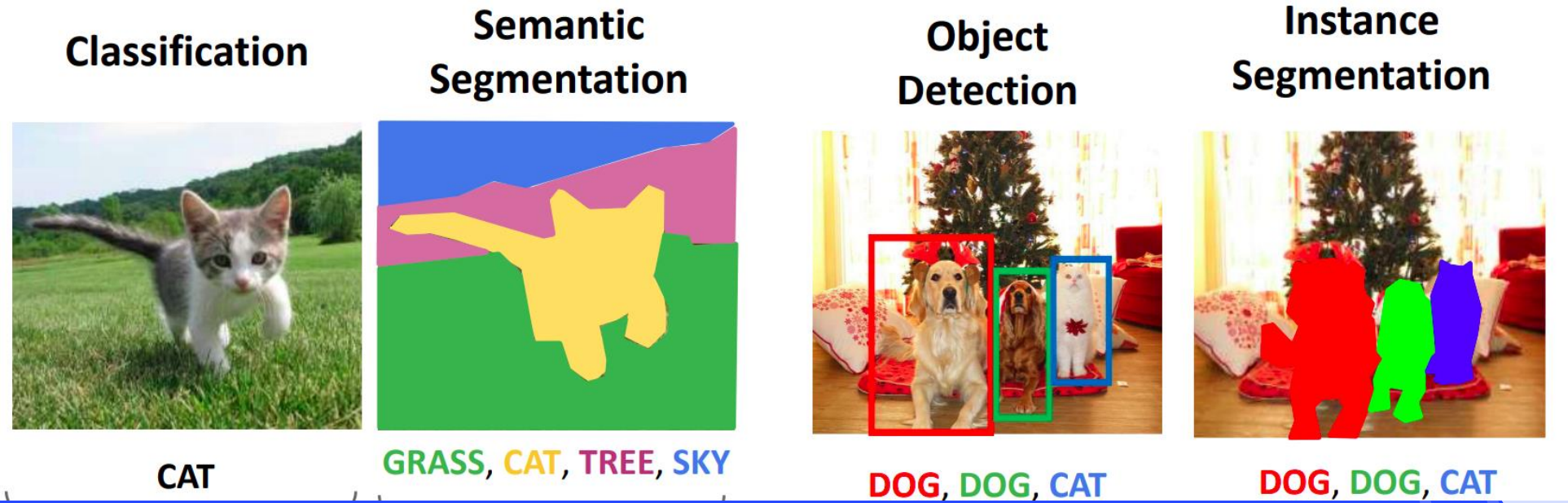


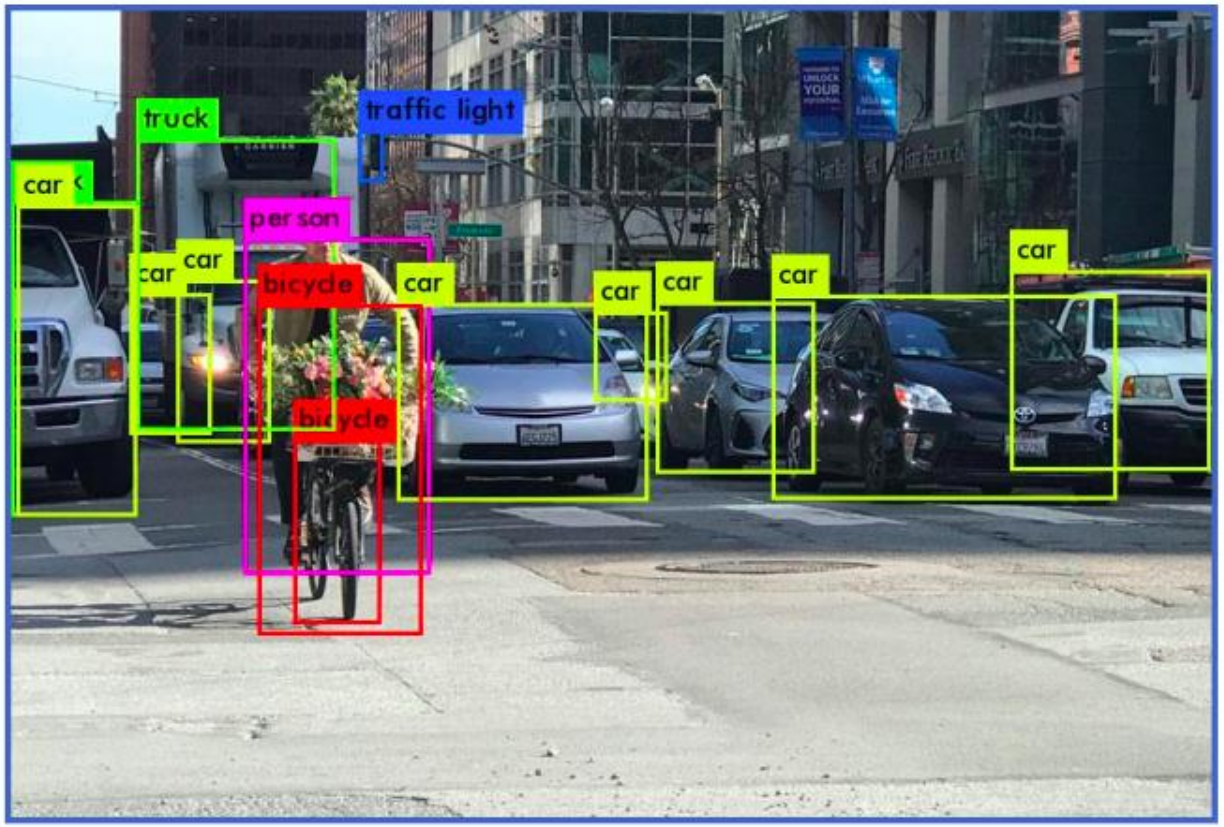
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

Presenter : hassan yousefzade

What is object detection ?

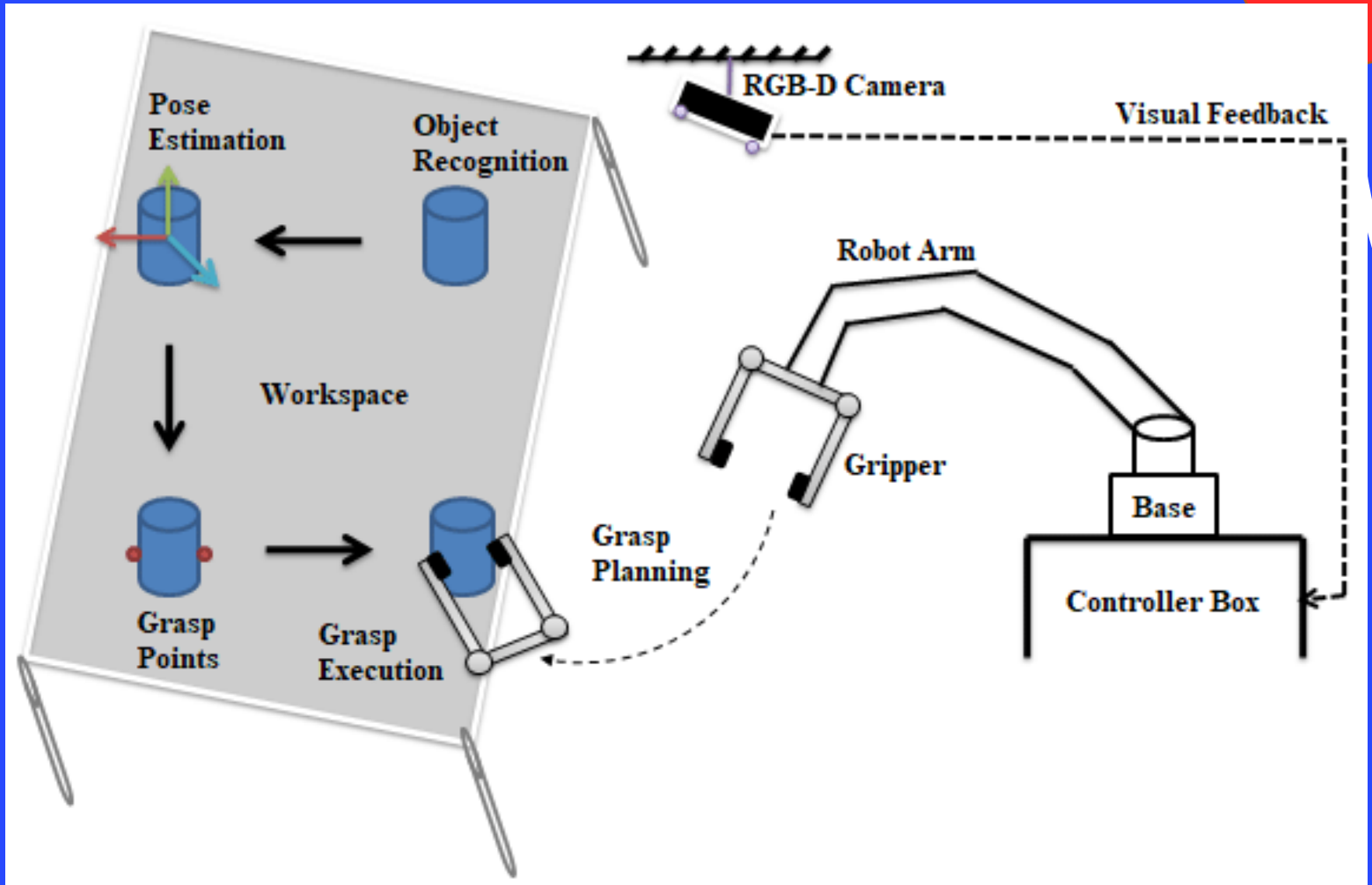
- **Detection vs Recognition**




truck
traffic light
car
person
car
car
bicycle
car
car
car
car
car
bicycle

Object detection in robotic

Object detection in robotic



Object detection

Input:

- Single RGB Image

Output:

- A set of detected objects

For each object predict:

- 1. Category label (from fixed, known set of categories)
- 2. Bounding box (four numbers: x, y, width, height)

Object Detection: Challenges

-Multiple outputs:

- **Need to output variable numbers of objects per image**

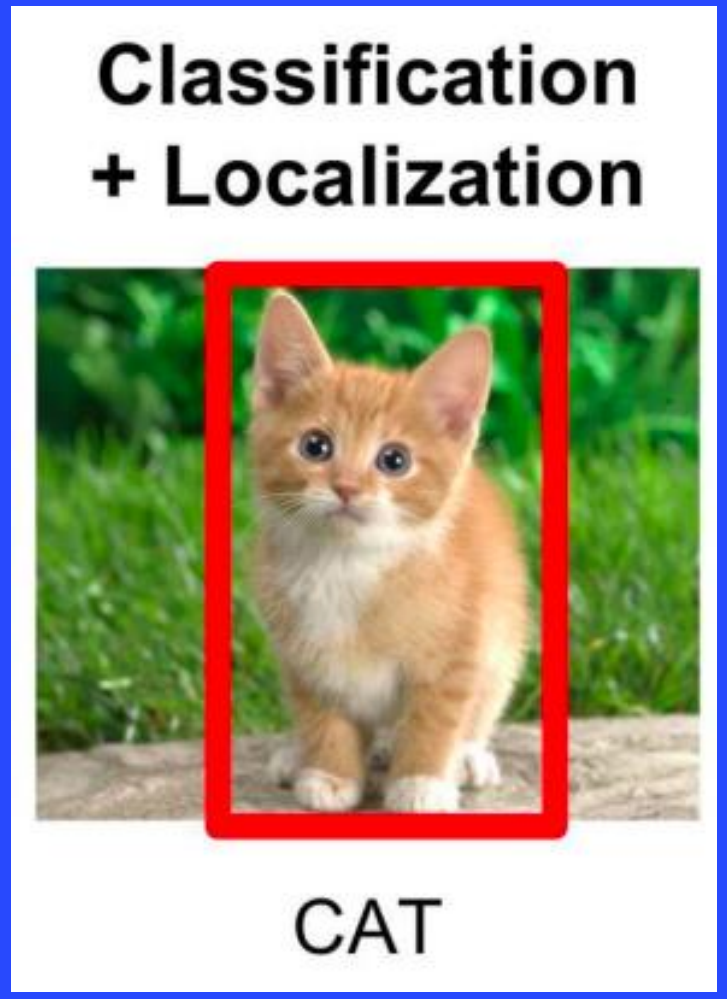
- Multiple types of output:

- **Need to predict "what" (category label) as well as “where” (bounding box)**

- Large images:

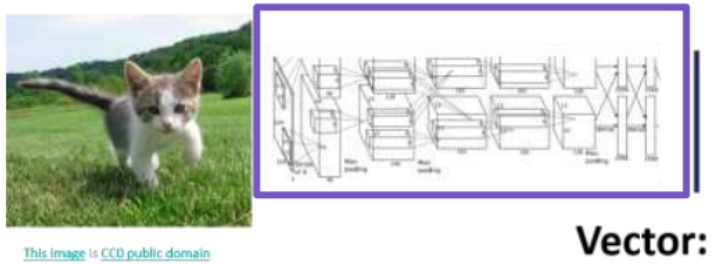
- **Classification works at 224x224; need higher resolution for detection, often ~800x600**

Object localization



How to implement object localization ?

Often pretrained on ImageNet (Transfer learning)



This Image is CC0 public domain

Vector: 4096

"What"

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label: Cat

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

"Where"

Correct box: (x', y', w', h')

L2 Loss

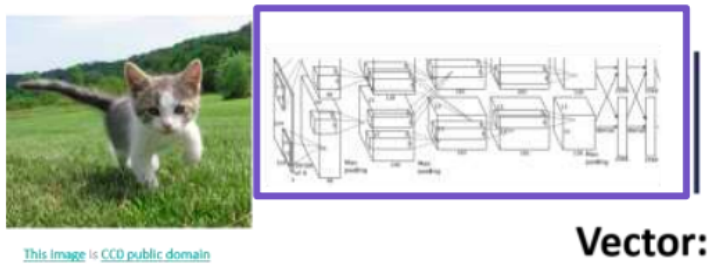
Weighted Sum

Loss

Multitask Loss

01-simple-regression-train.ipynb
02-simple-regression-inference.ipynb
03-object-classification-and-localization.ipynb
04-object-classification-and-localization-infe

Often pretrained on ImageNet (Transfer learning)



This image is CC0 public domain

Vector: 4096

"What"

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label: Cat

Softmax Loss

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

"Where"

Correct box: (x', y', w', h')

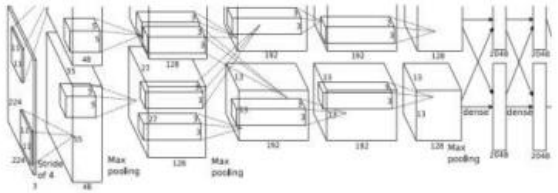
L2 Loss

Weighted Sum → **Loss**

Multitask Loss

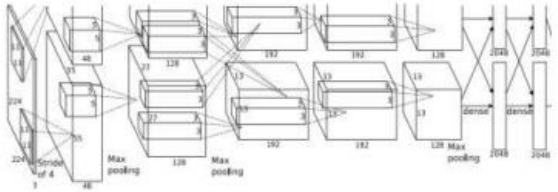
Problem:
Images can have more than one object!

Need different numbers of outputs per image



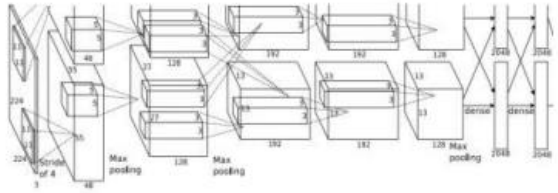
CAT: (x, y, w, h)

4 numbers



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

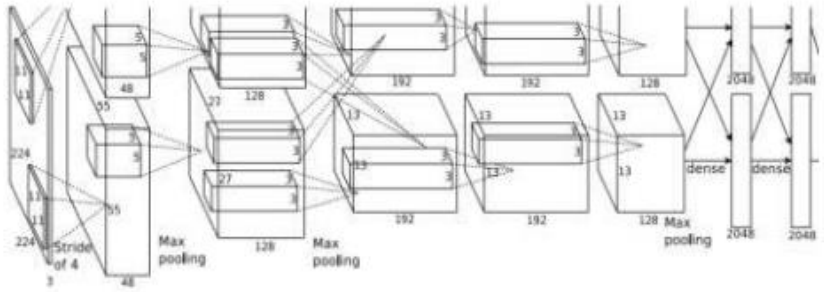
16 numbers



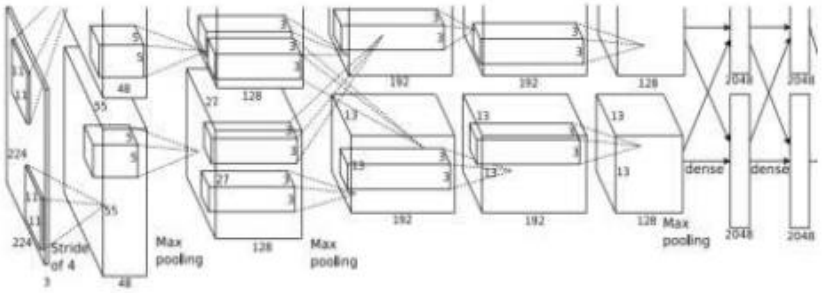
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Detecting Multiple Objects: Sliding Window



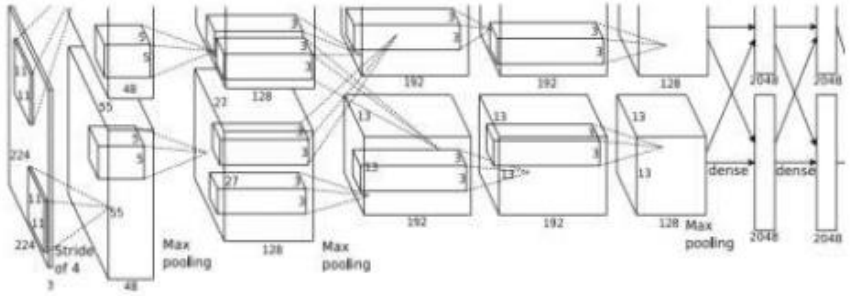
Dog? NO
Cat? NO
Background? YES



Dog? YES

Cat? NO

Background? NO



Dog? YES
Cat? NO
Background? NO

- **Question:**

  How many possible boxes are there in an image of size H x W?

Consider a box of size h x w :

Possible x positions: W – w + 1

Possible y positions: H – h + 1

Possible positions: (W – w + 1) * (H – h + 1)

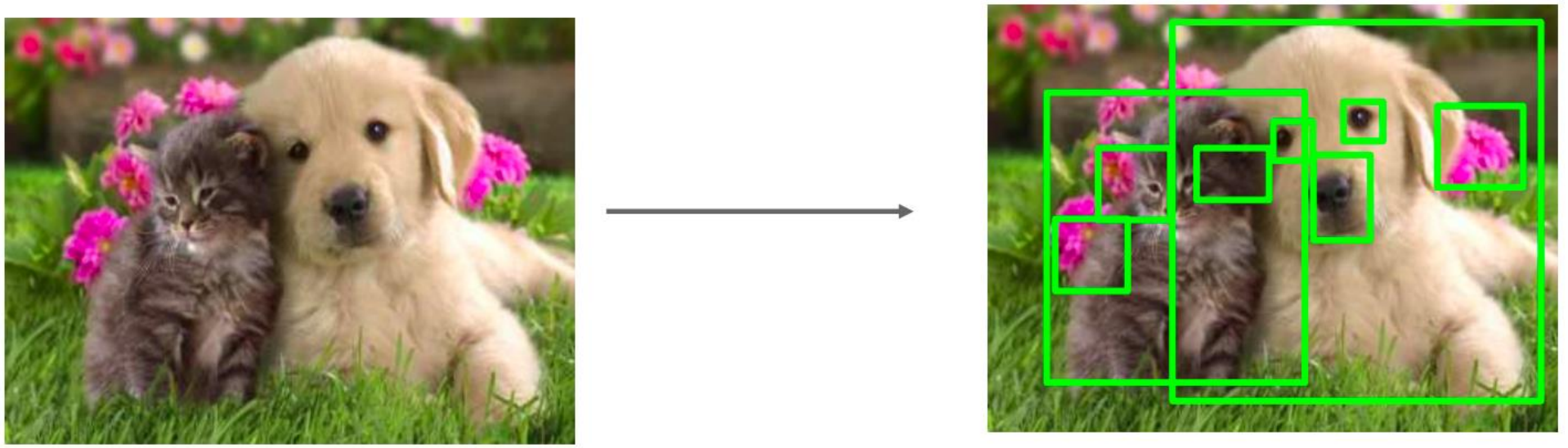
Total possible boxes:

$$\sum_{h=1}^{H}\sum_{w=1}^{W}(W-w+1)(H-h+1)$$

$$=\frac{H(H+1)}{2}\frac{W(W+1)}{2}$$

800 x 600 image has ~58M boxes! No way we can evaluate them all

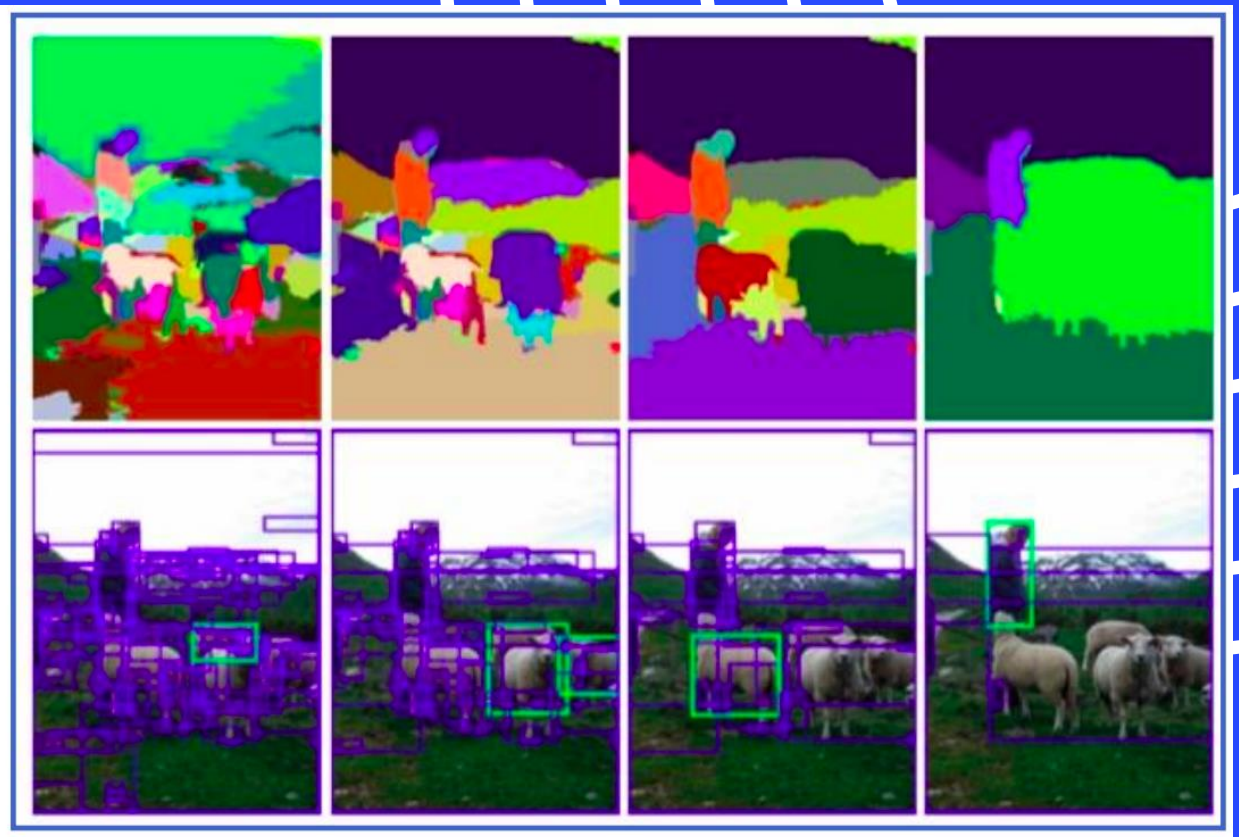
Region Proposals



Selective Search Method

It is based on computing hierarchical grouping of similar regions based on

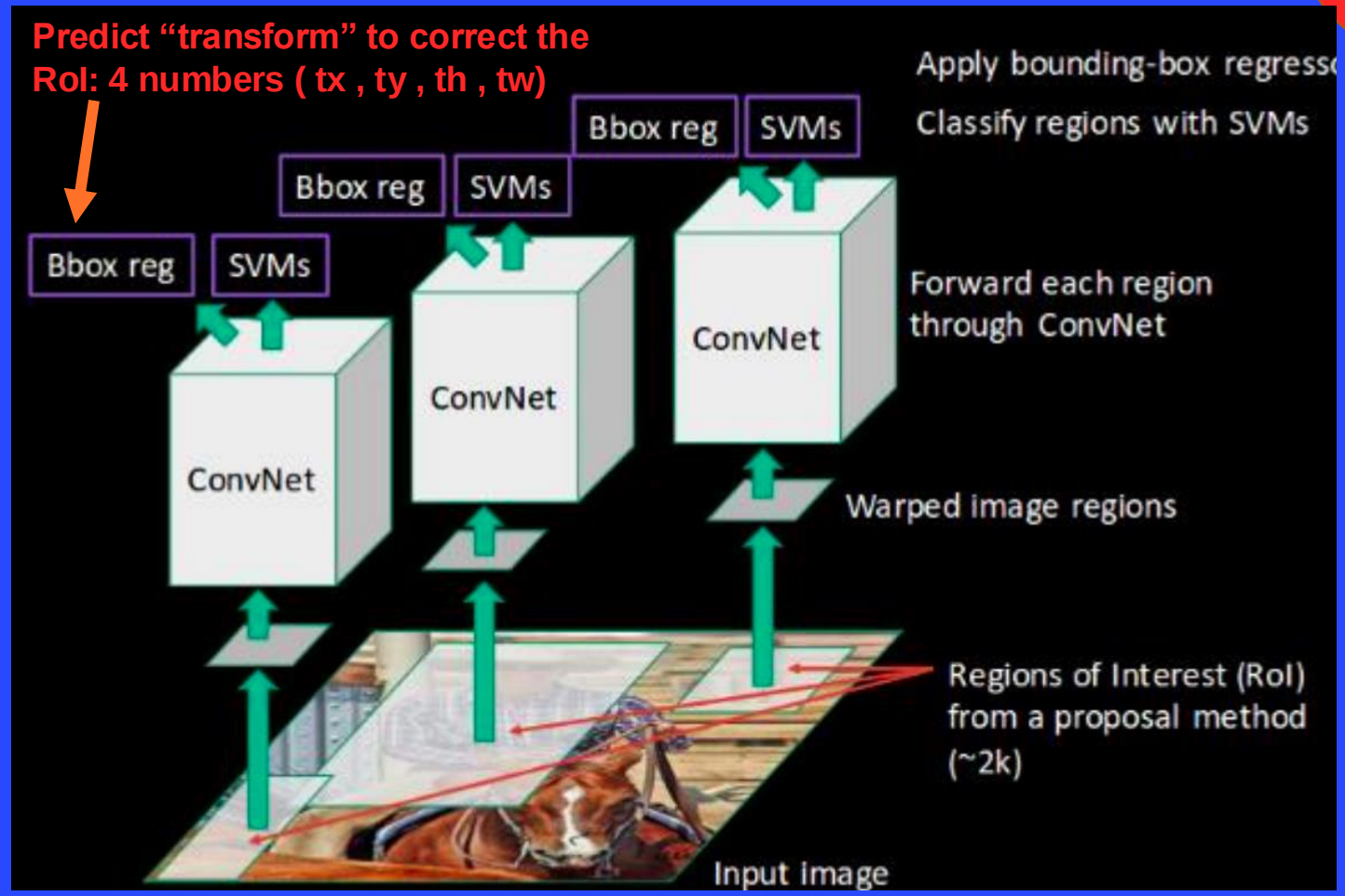
- **Color**
- **Texture**
- **Size**
- **shape compatibility**



05-selective-search.ipynb

06-object-detection-and-bounding-boxes.ipynb

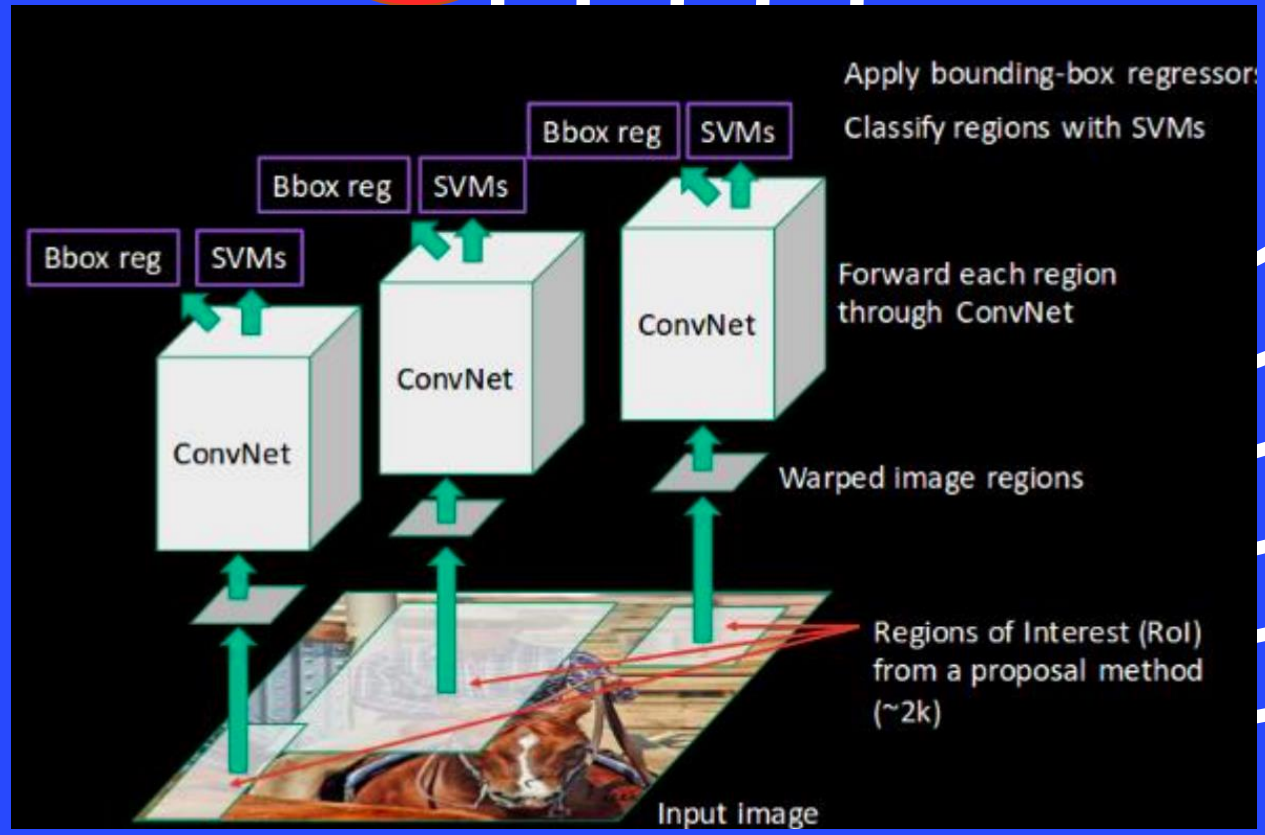
R-CNN: Region-Based CNN



R-CNN: Test-time

Input: Single RGB Image

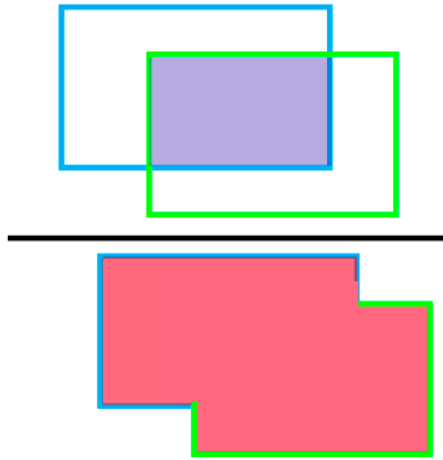
1. Run region proposal method to compute ~2000 region proposals
2. Resize each region to 224x224 and run independently through CNN to predict class scores and bbox transform
3. Use scores to select a subset of region proposals to output
   (Many choices here: threshold on background, or per-category? Or take top K proposals per image?)
4. Compare with ground-truth boxes???



Comparing Boxes: Intersection over Union (IoU)

Comparing Boxes: Intersection over Union (IoU)

$$\frac{\text{Area of Intersection}}{\text{Area of Union}}$$



Comparing Boxes: Intersection over Union (IoU)

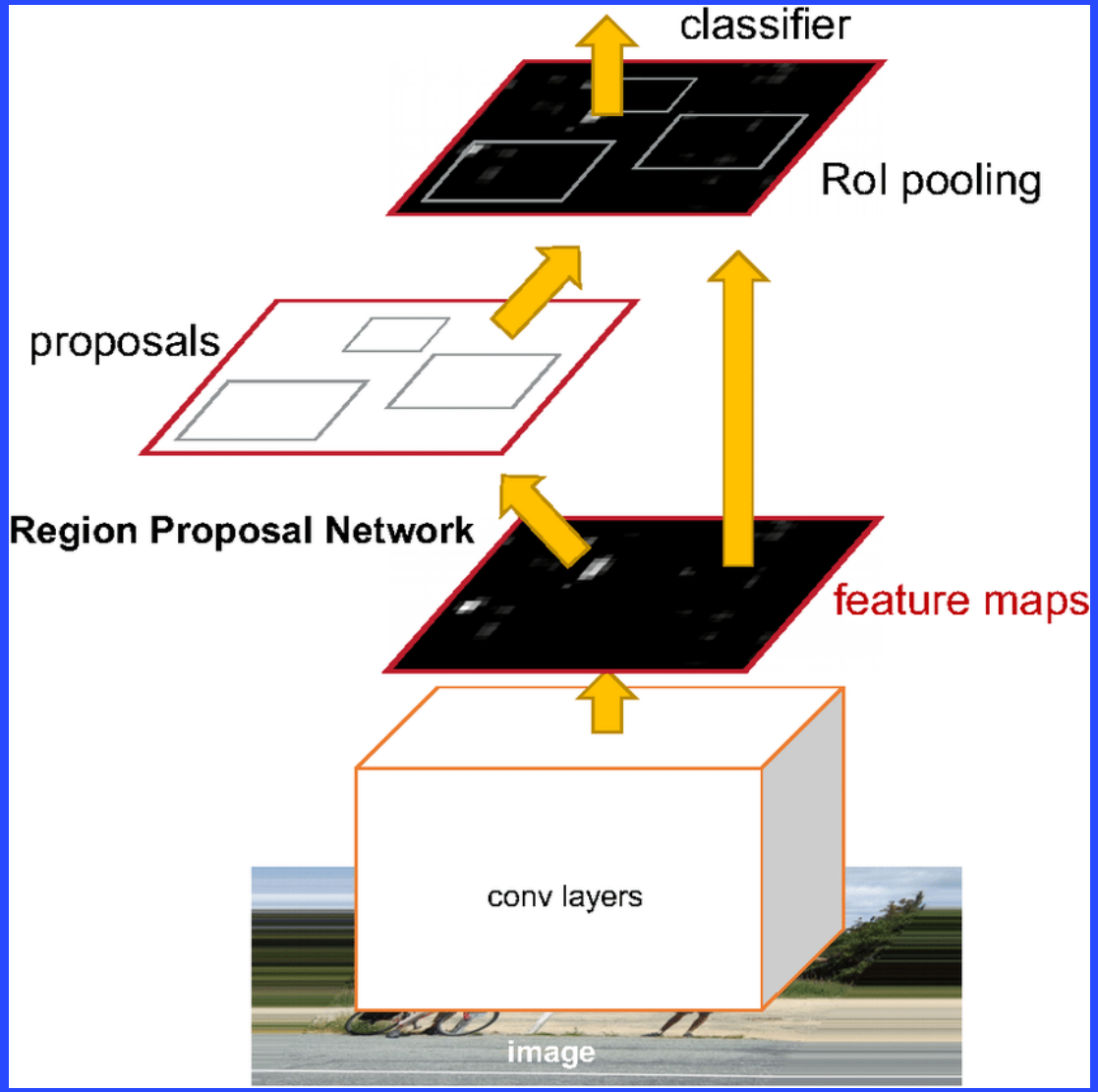
Comparing Boxes: Intersection over Union (IoU)

Comparing Boxes: Intersection over Union (IoU)

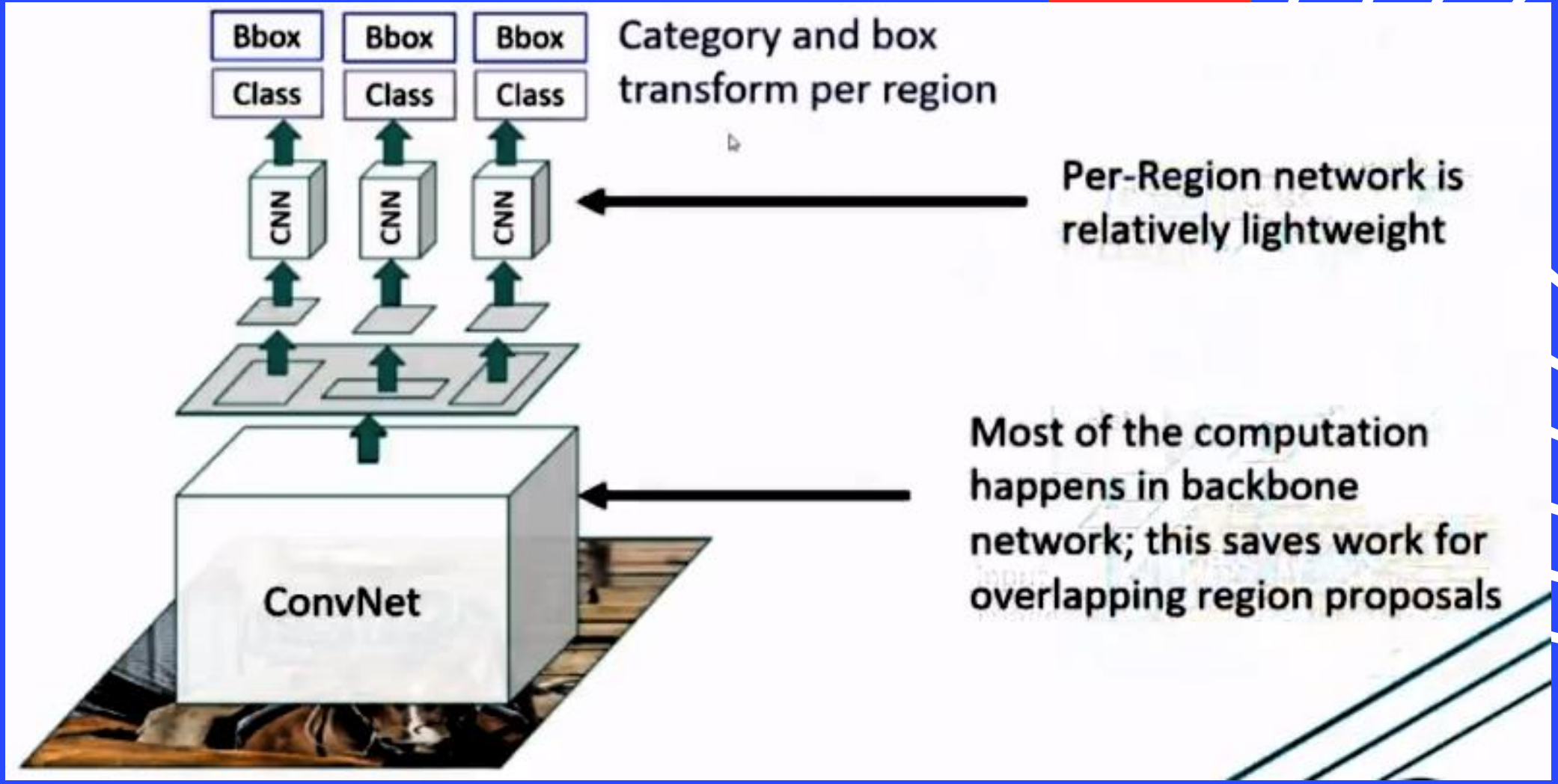
IOU > 0.5 is “decent”
IOU > 0.7 is “pretty good”
IOU > 0.9 is “almost perfect”

07-Intersection-over-Union(IoU).ipynb

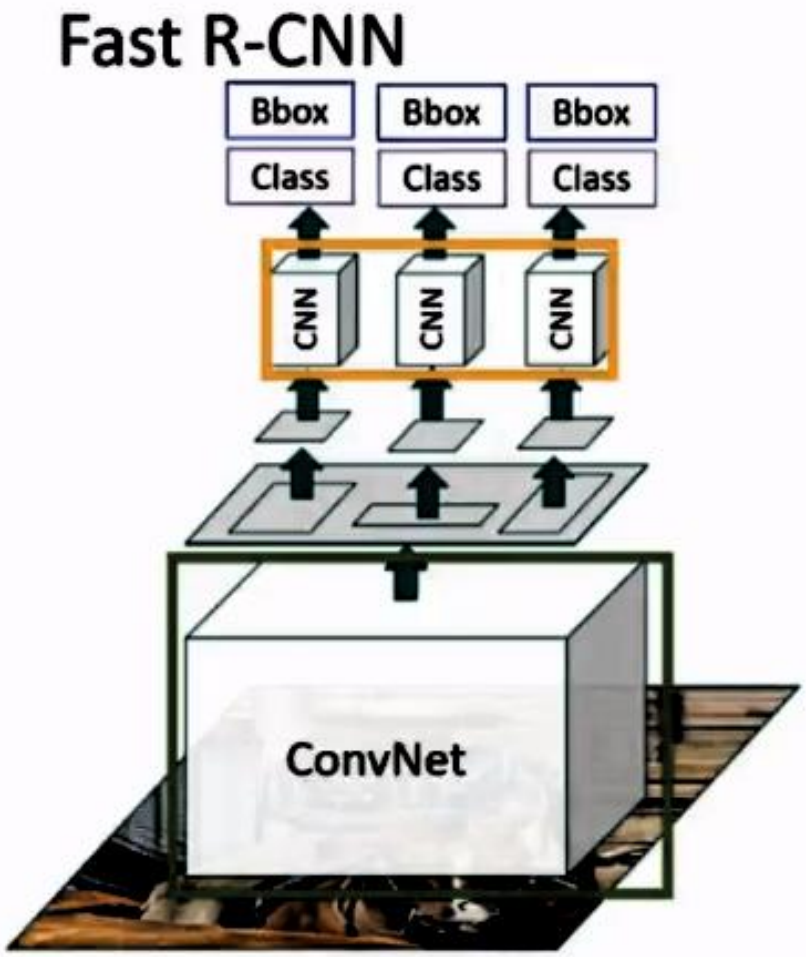
Fast R-CNN

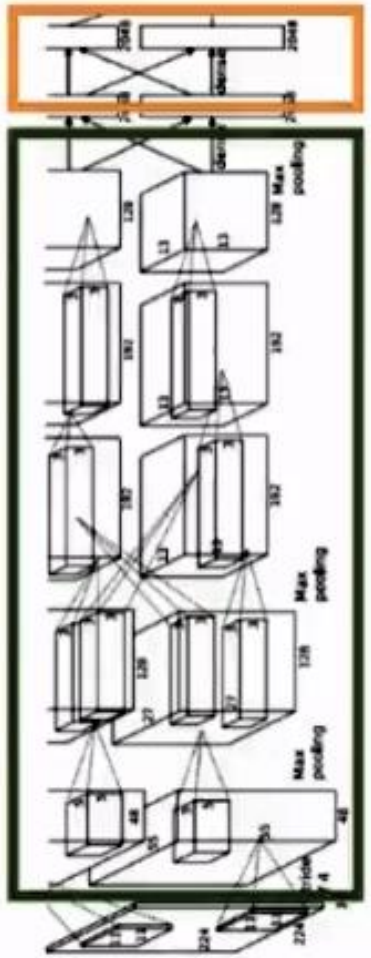

classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image

Fast R-CNN


Bbox
Bbox
Bbox
Class
Class
Class
CNN
CNN
CNN
ConvNet
Category and box transform per region
Per-Region network is relatively lightweight
Most of the computation happens in backbone network; this saves work for overlapping region proposals

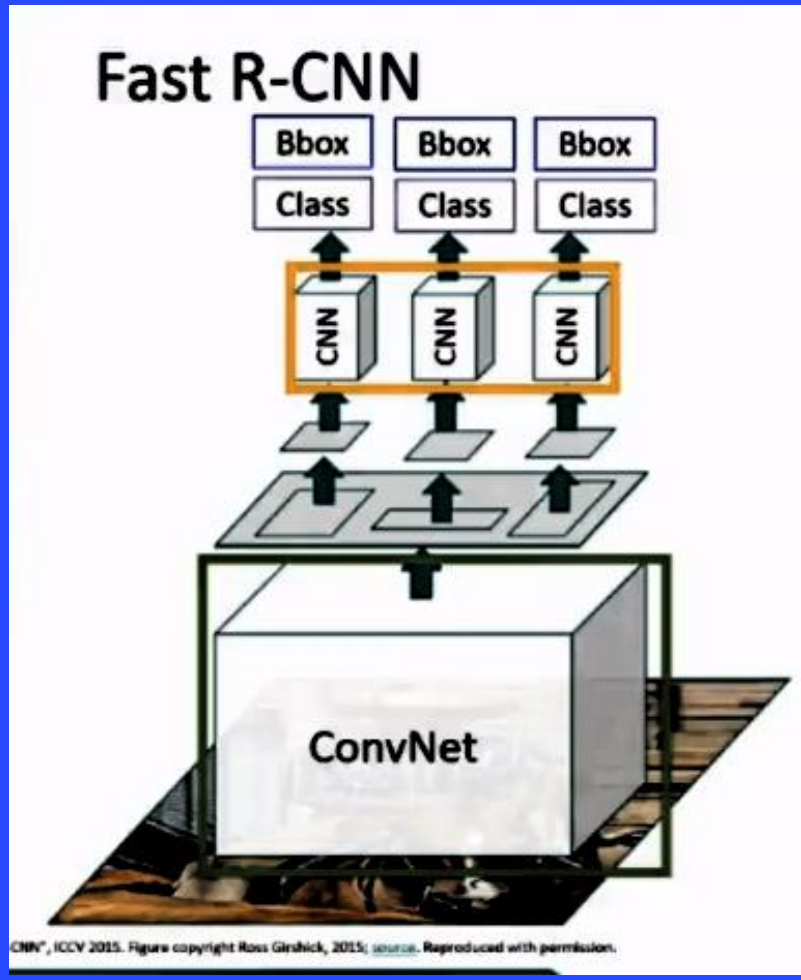
Fast R-CNN

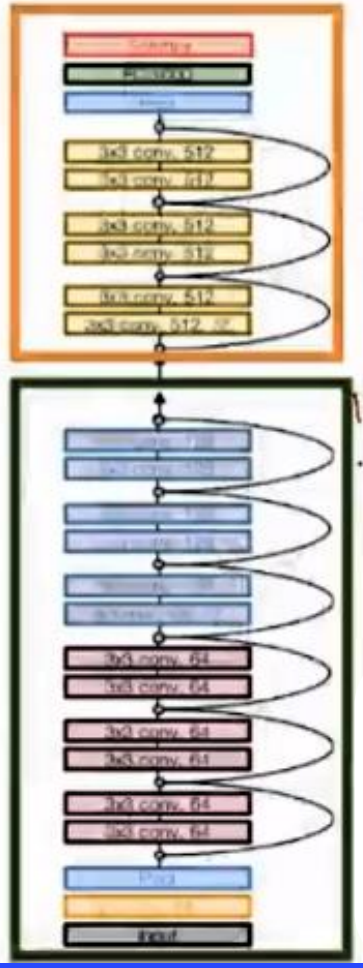




Example:
When using AlexNet for detection, five conv layers are used for backbone and two FC layers are used for per-region network

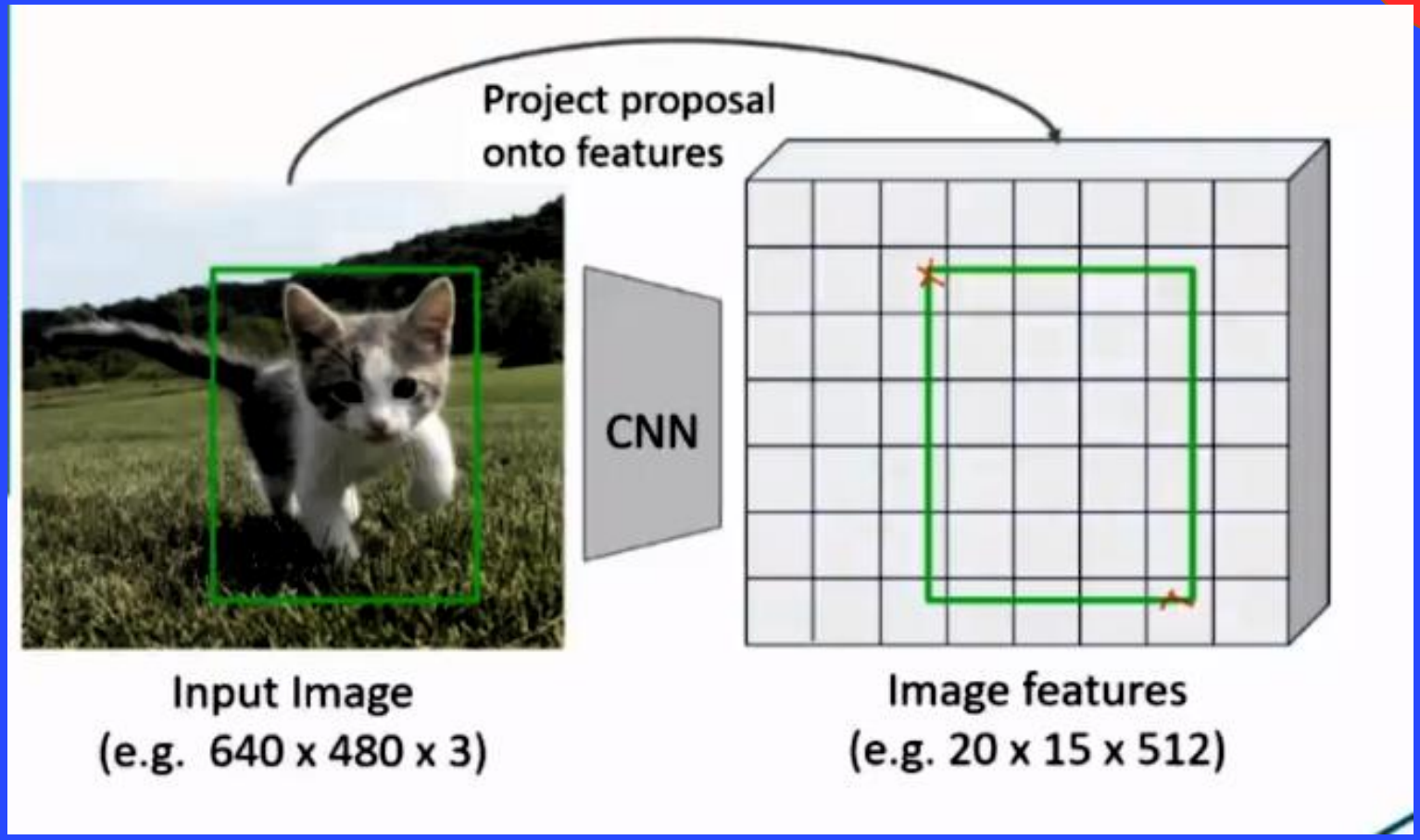
Fast R-CNN



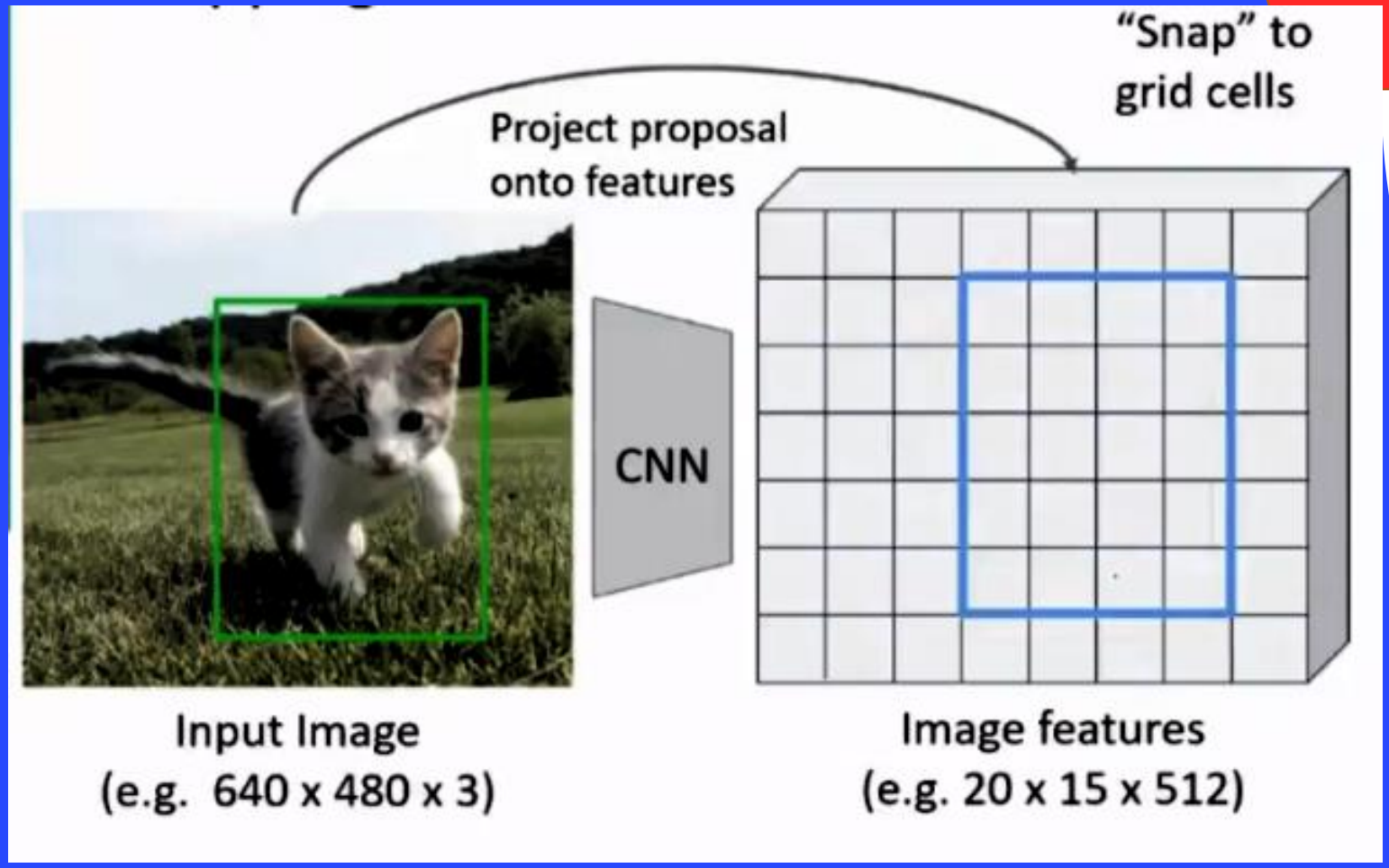


Example:
For ResNet, last stage is used as per-region network; the rest of the network is used as backbone

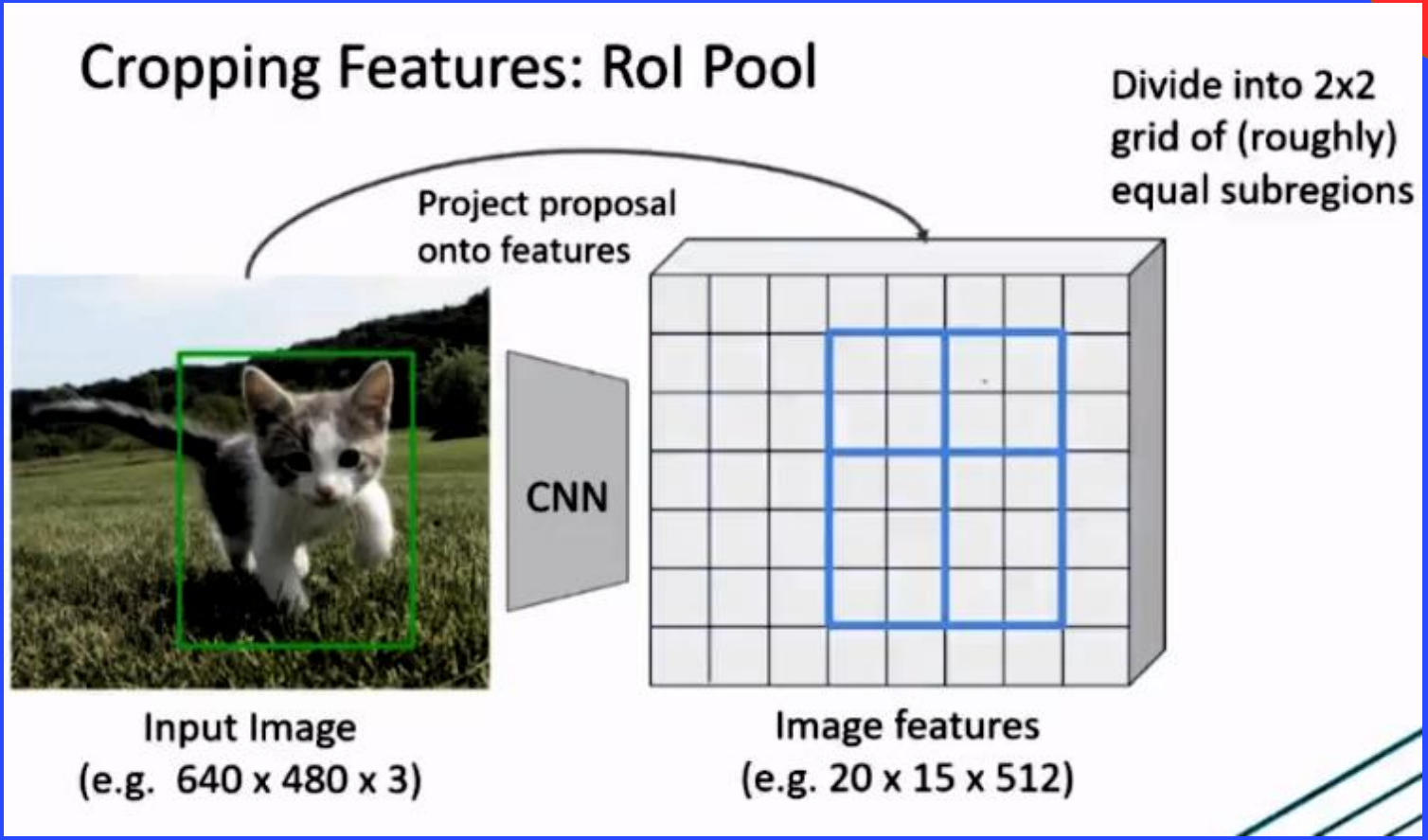
Fast R-CNN


Project proposal
onto features
CNN
Input Image
(e.g. 640 x 480 x 3)
Image features
(e.g. 20 x 15 x 512)

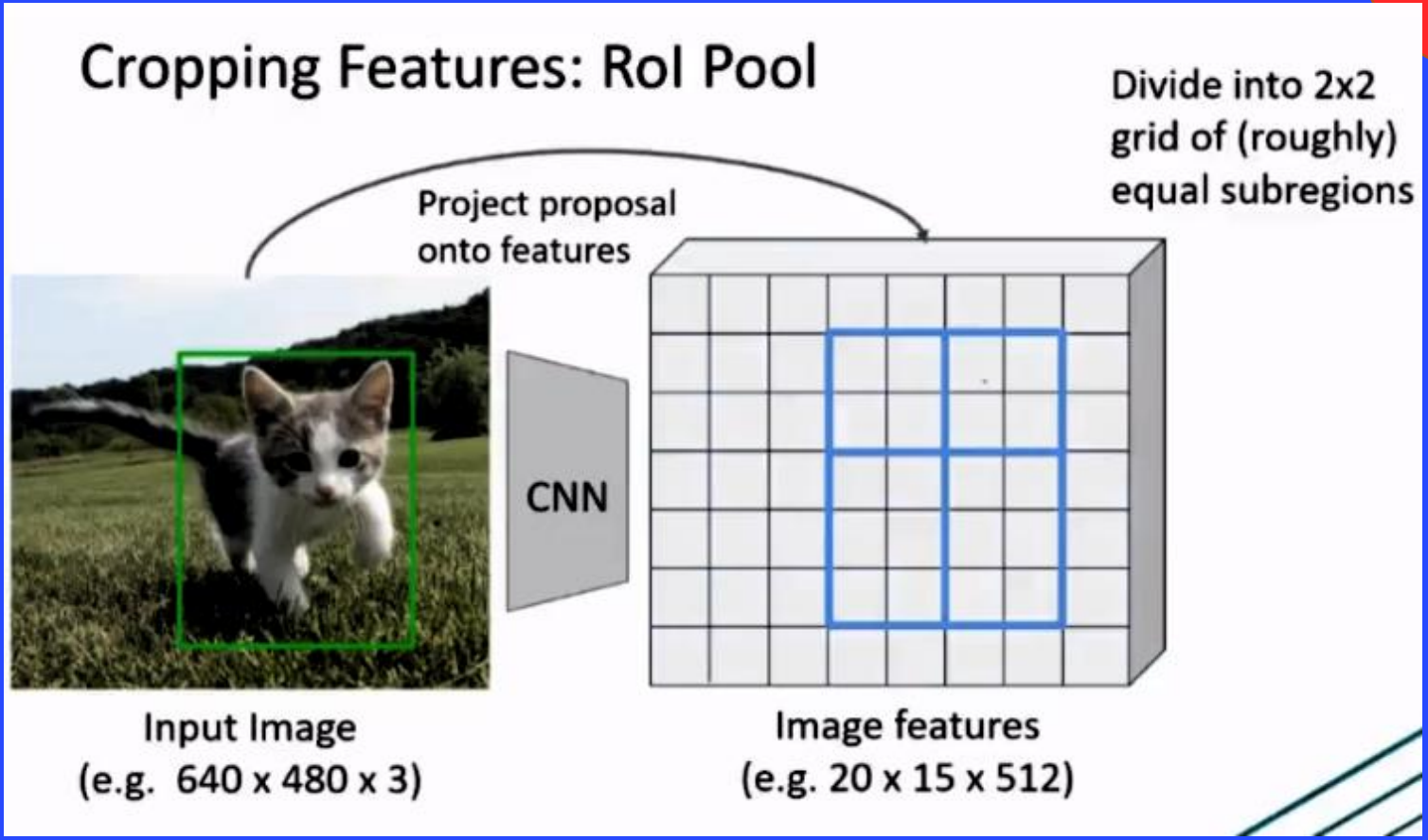
Fast R-CNN


"Snap" to
grid cells
Project proposal
onto features
CNN
Input Image
(e.g. 640 x 480 x 3)
Image features
(e.g. 20 x 15 x 512)

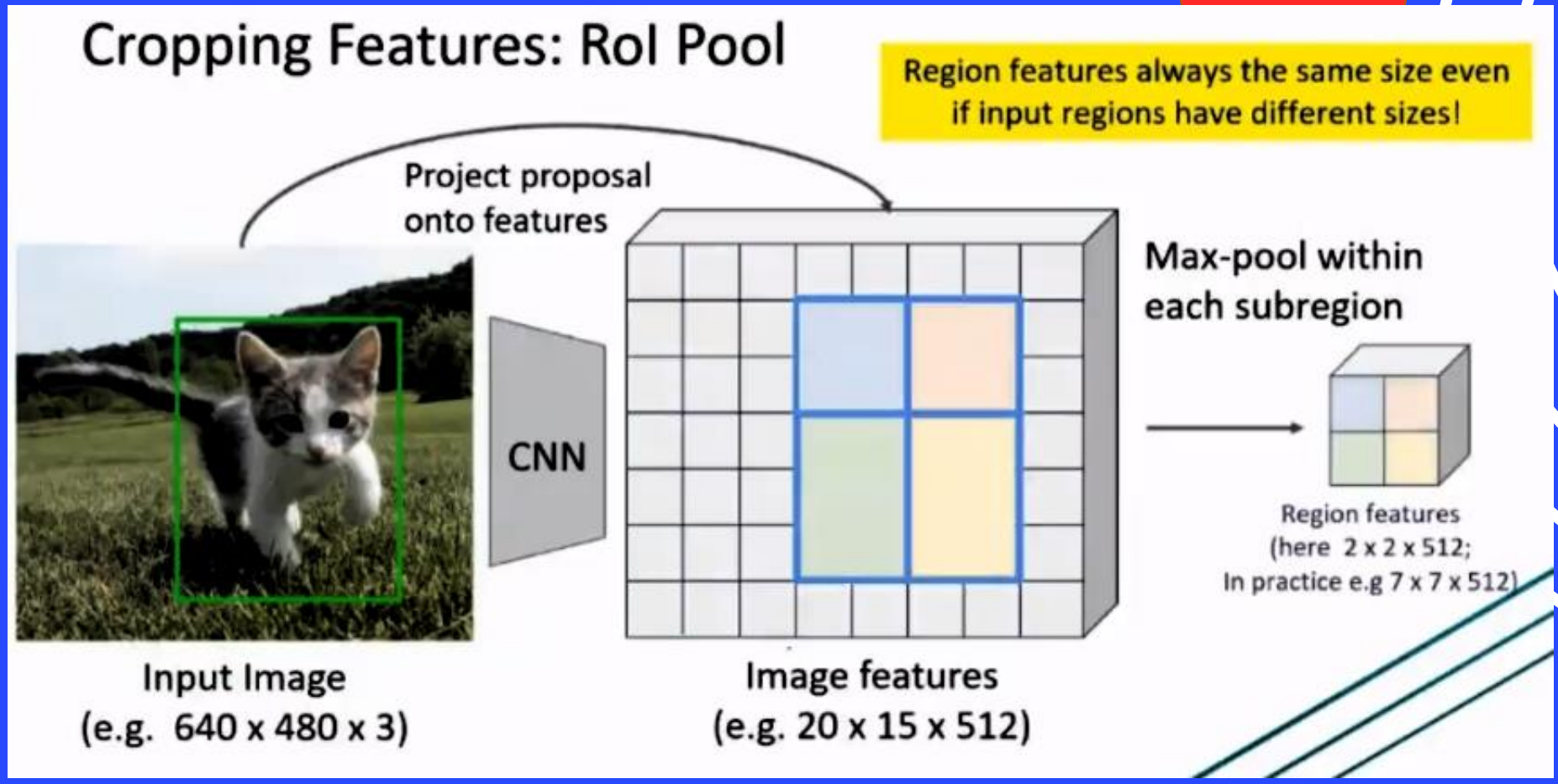
Fast R-CNN


Cropping Features: RoI Pool
Divide into 2x2 grid of (roughly) equal subregions
Project proposal onto features
CNN
Input Image
(e.g. 640 x 480 x 3)
Image features
(e.g. 20 x 15 x 512)

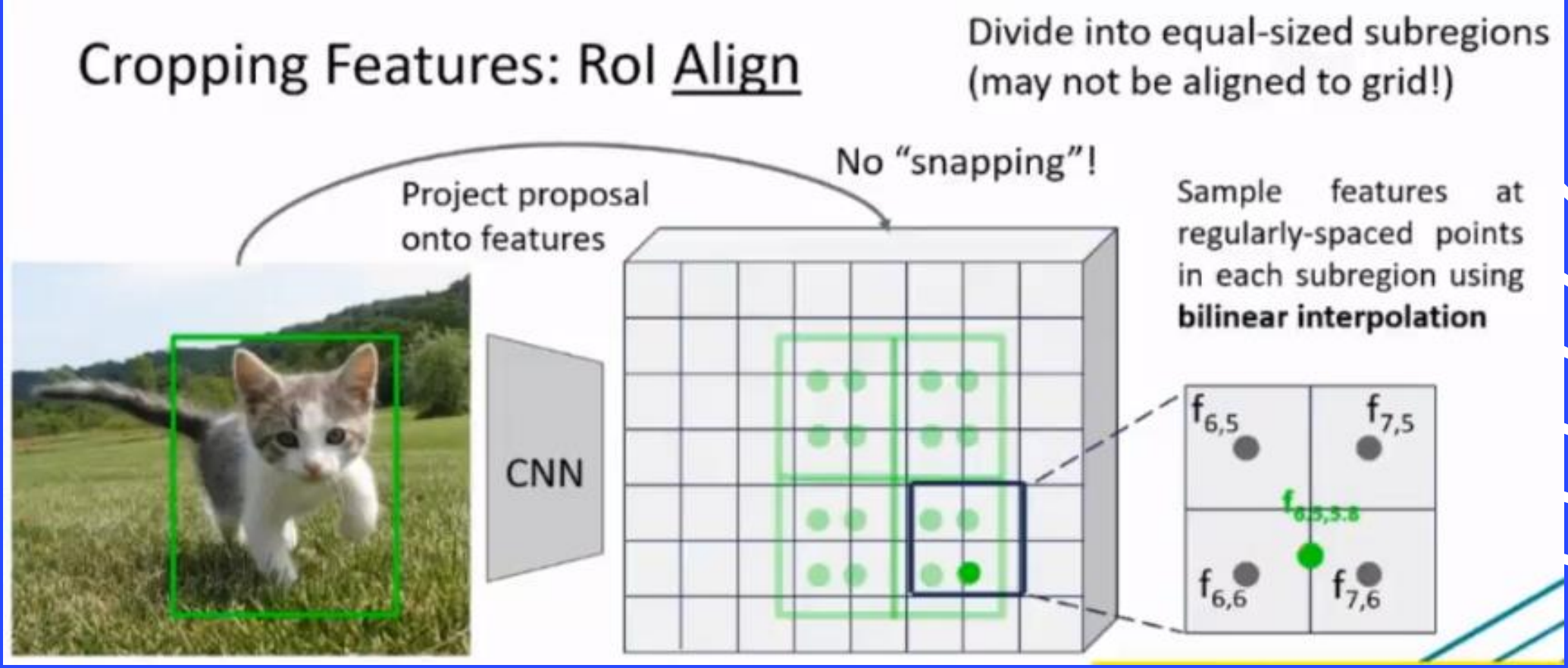
Fast R-CNN


Cropping Features: RoI Pool
Divide into 2x2 grid of (roughly) equal subregions
Project proposal onto features
CNN
Input Image
(e.g. 640 x 480 x 3)
Image features
(e.g. 20 x 15 x 512)

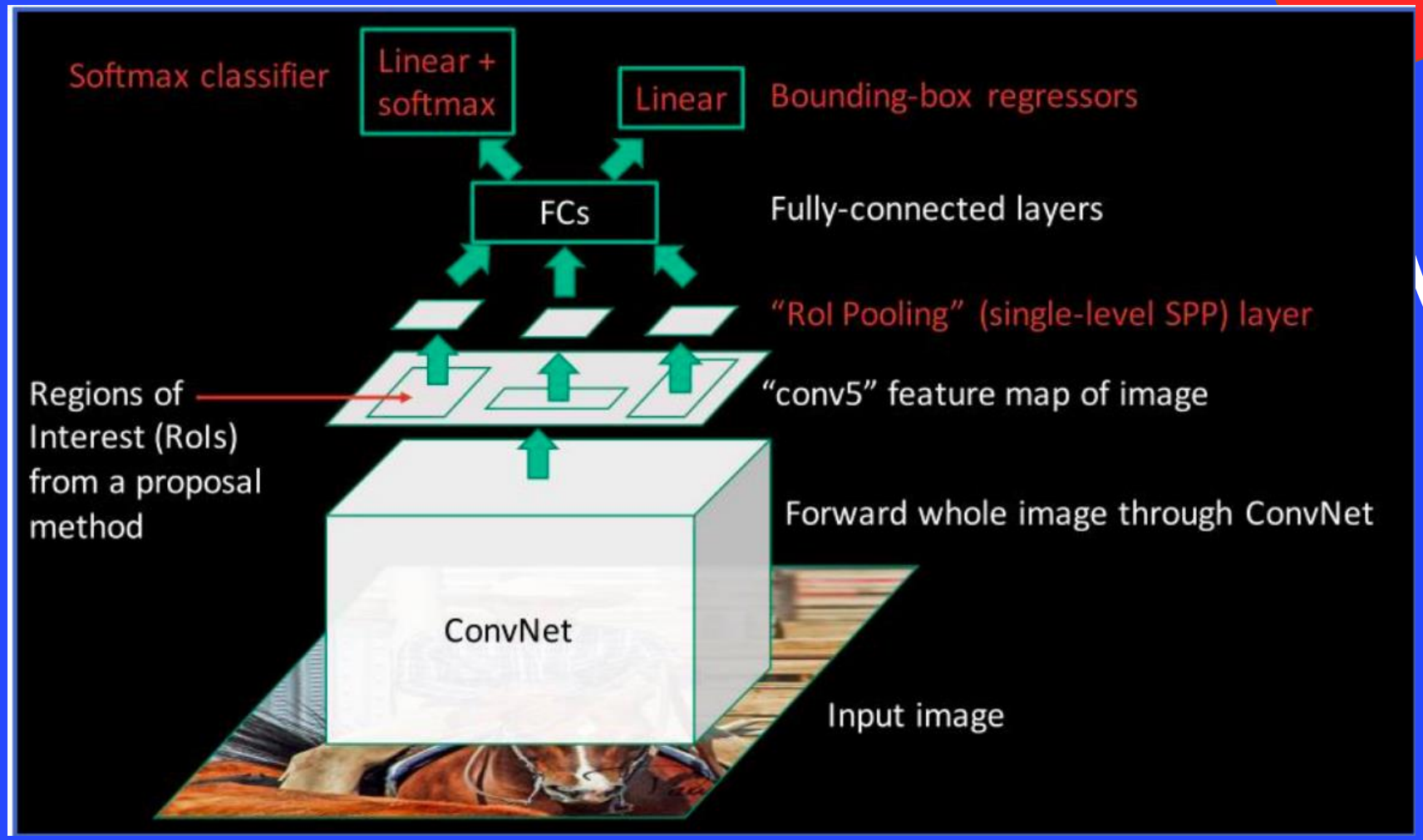
Fast R-CNN

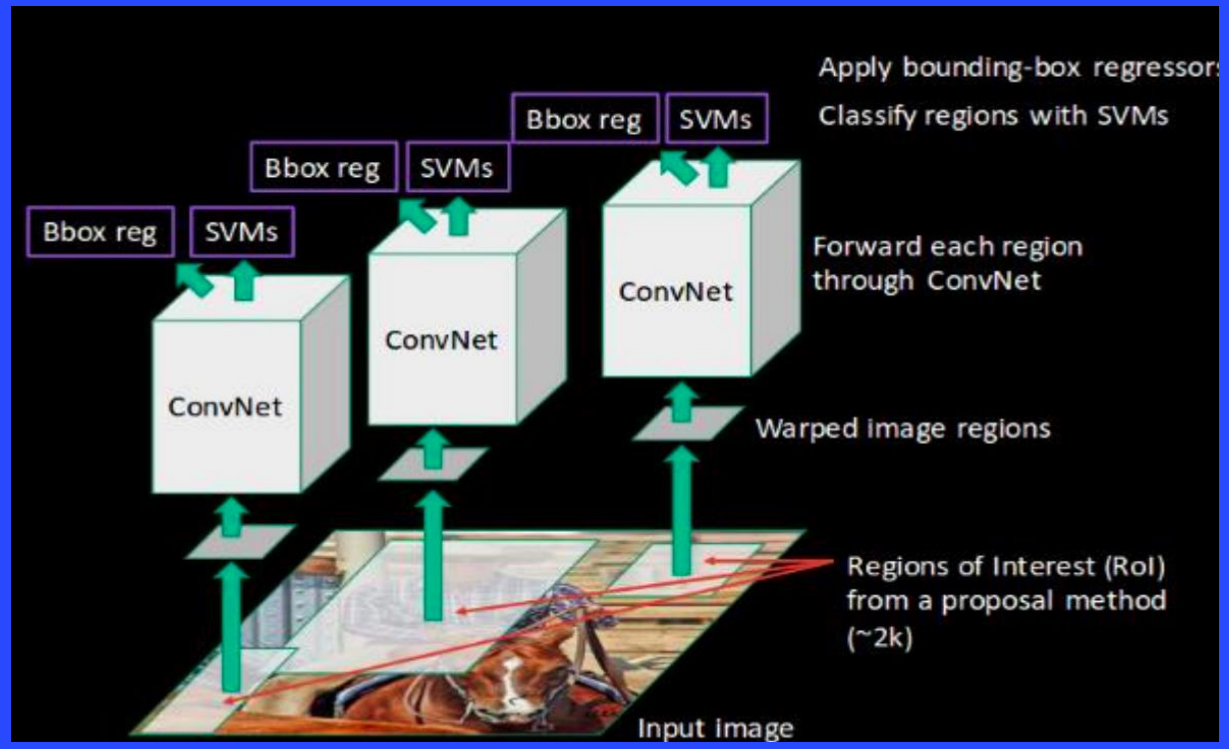

Cropping Features: RoI Pool
Region features always the same size even if input regions have different sizes!
Project proposal onto features
CNN
Max-pool within each subregion
Region features (here 2 x 2 x 512; In practice e.g 7 x 7 x 512)
Input Image (e.g. 640 x 480 x 3)
Image features (e.g. 20 x 15 x 512)

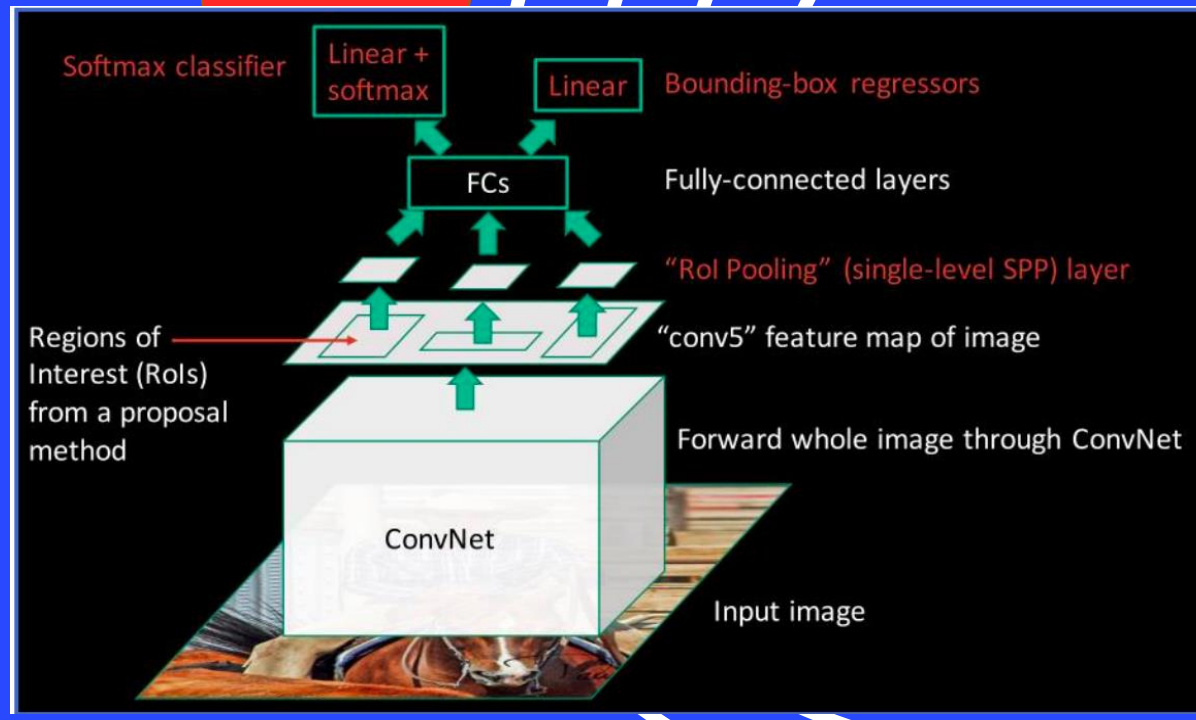
Fast R-CNN


Cropping Features: RoI Align
Divide into equal-sized subregions
(may not be aligned to grid!)
No "snapping"!
Project proposal
onto features
CNN
Sample features at
regularly-spaced points
in each subregion using
bilinear interpolation
$f_{6,5}$
$f_{7,5}$
$f_{6.5,5.8}$
$f_{6,6}$
$f_{7,6}$

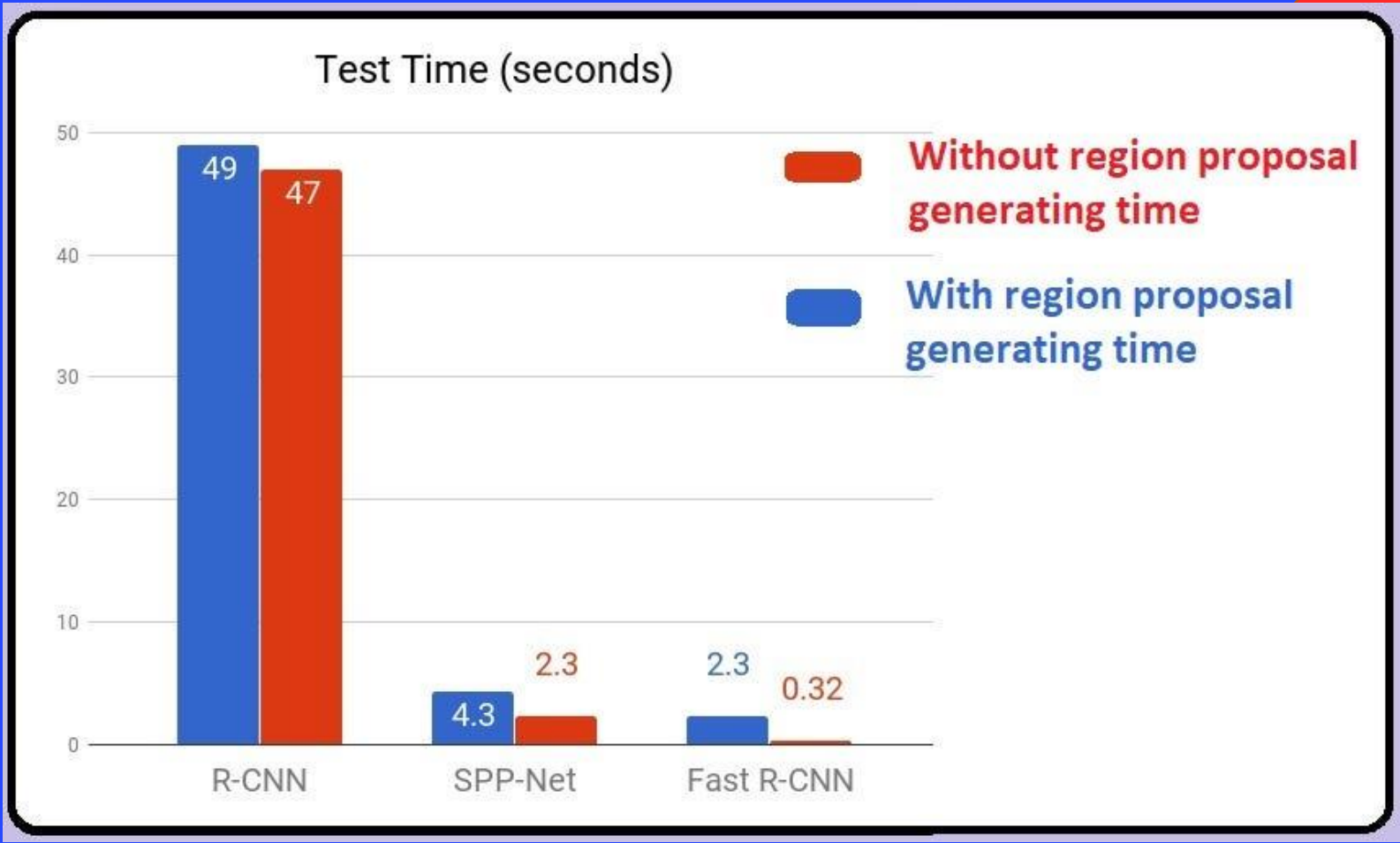
Fast R-CNN


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

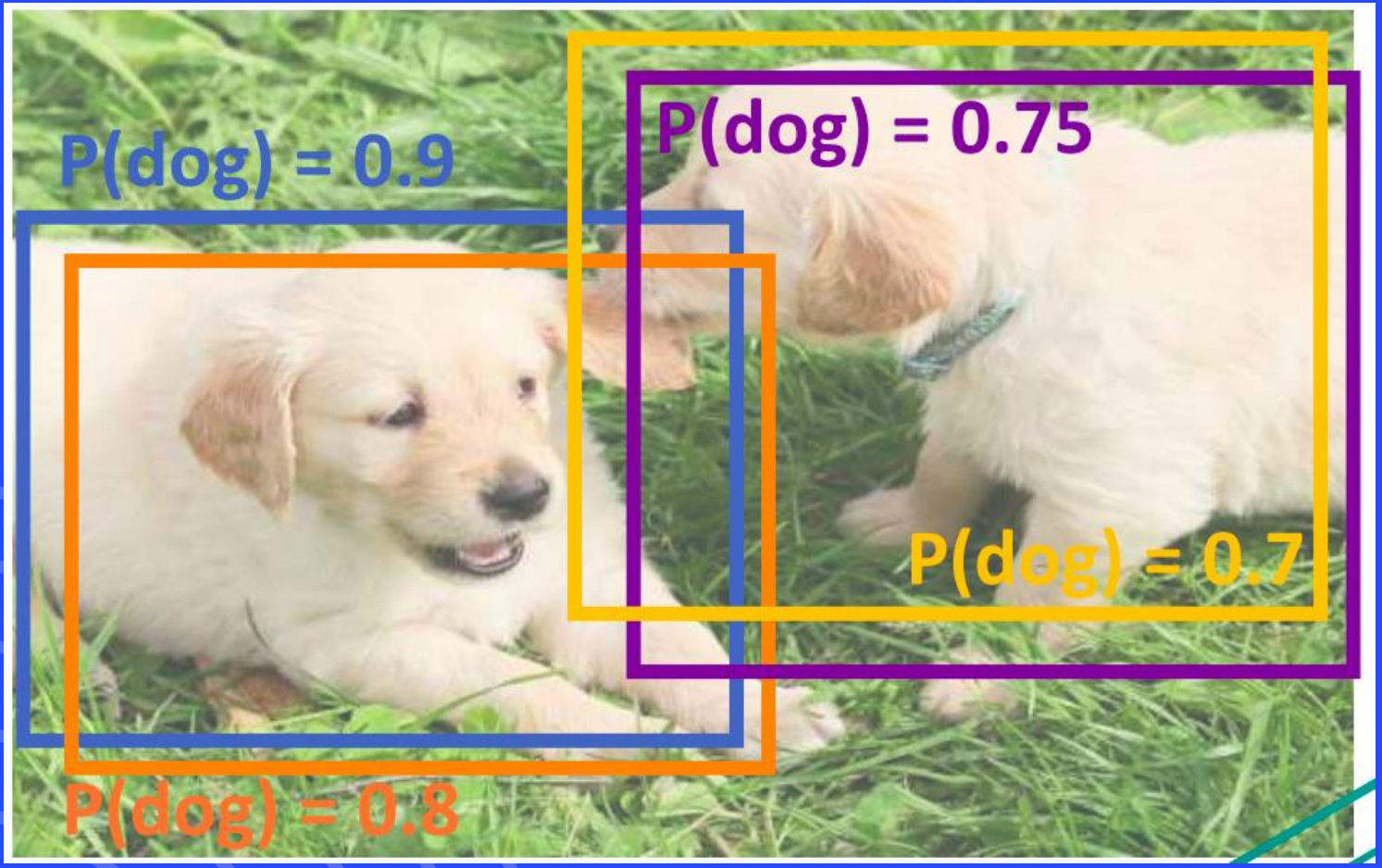




Selective Search gives 2000 region proposals in a few seconds on CPU

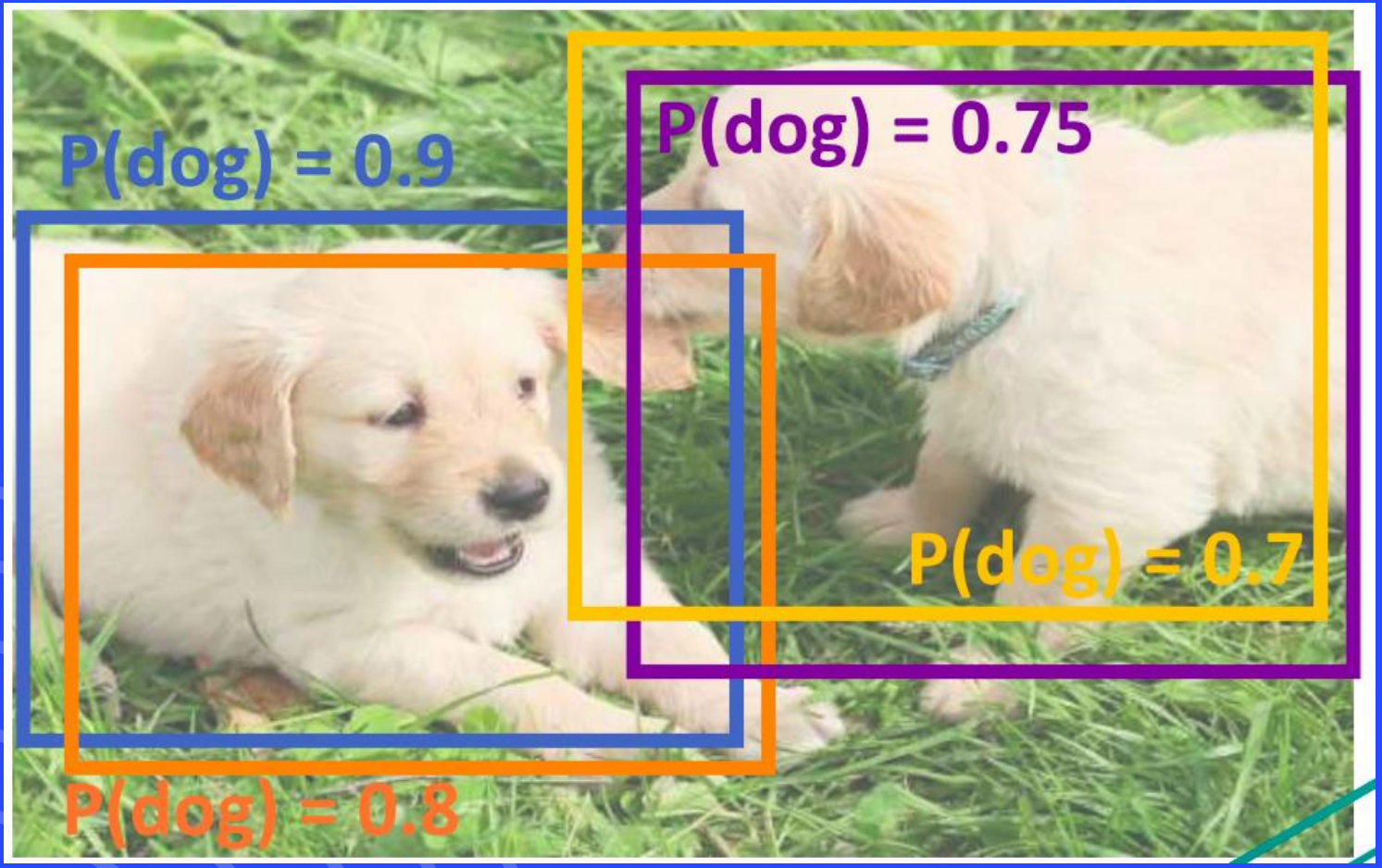

Test Time (seconds)
50
40
30
20
10
0
49
47
4.3
2.3
2.3
0.32
R-CNN
SPP-Net
Fast R-CNN
Without region proposal generating time
With region proposal generating time

Overlapping Boxes: Non-Max Suppression (NMS)



1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

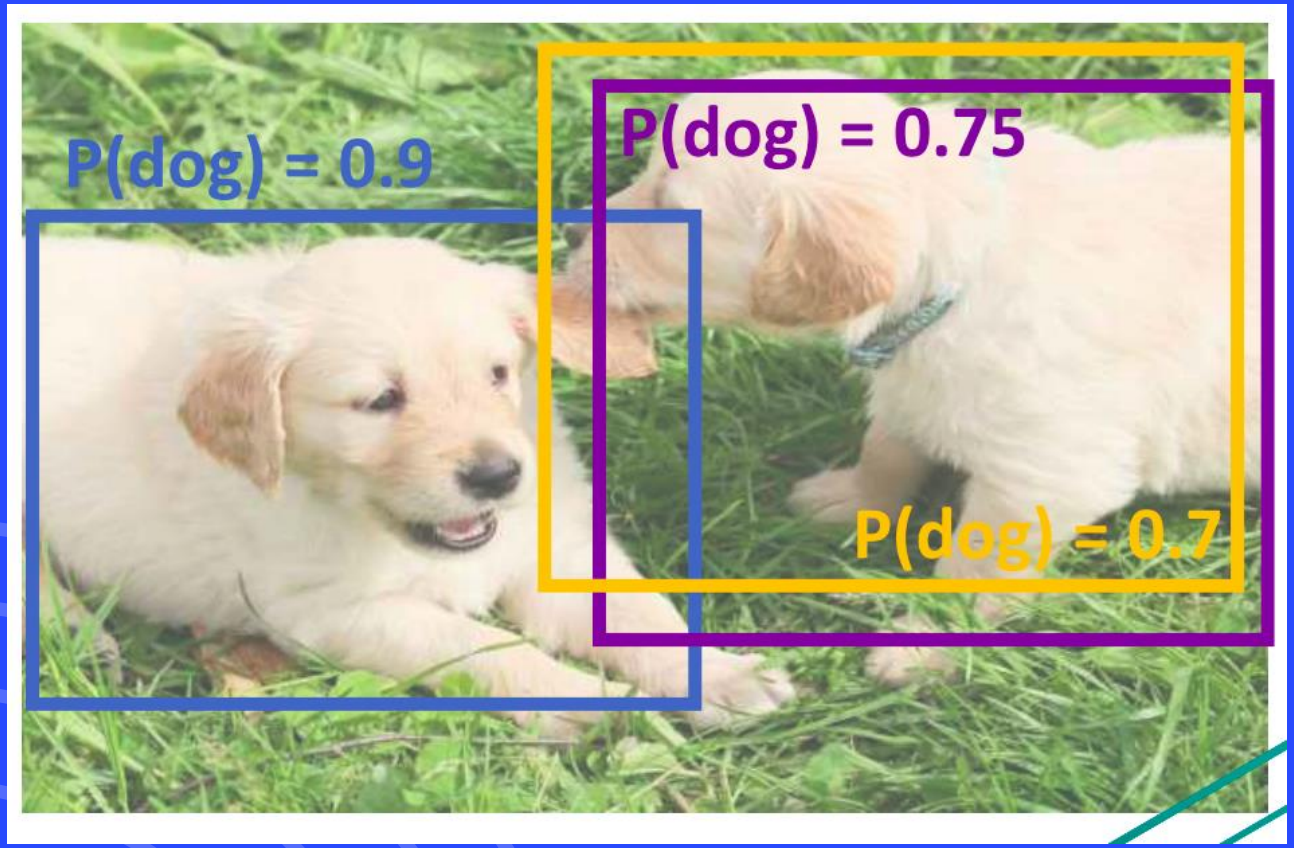
Overlapping Boxes: Non-Max Suppression (NMS)



1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

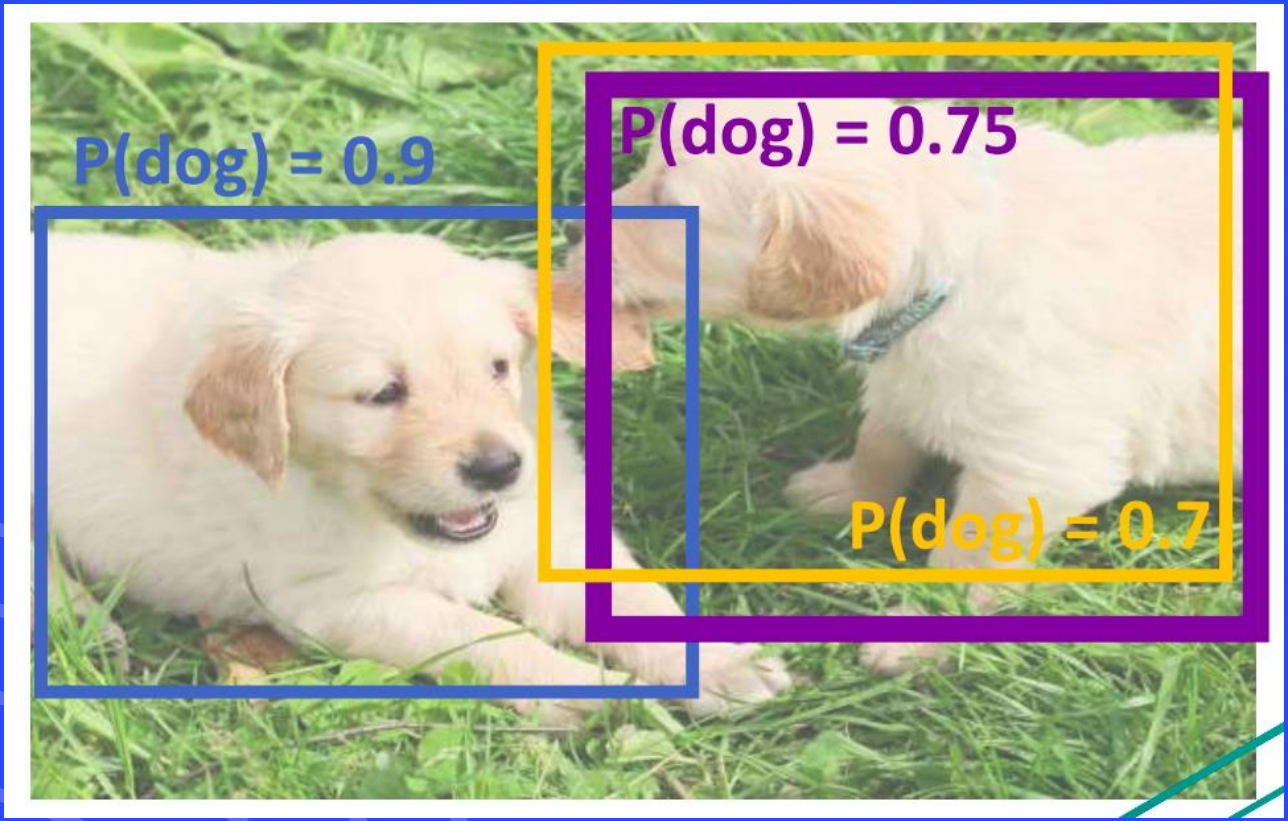
IoU(,) = **0.78**
IoU(,) = 0.05
IoU(,) = 0.07

Overlapping Boxes: Non-Max Suppression (NMS)



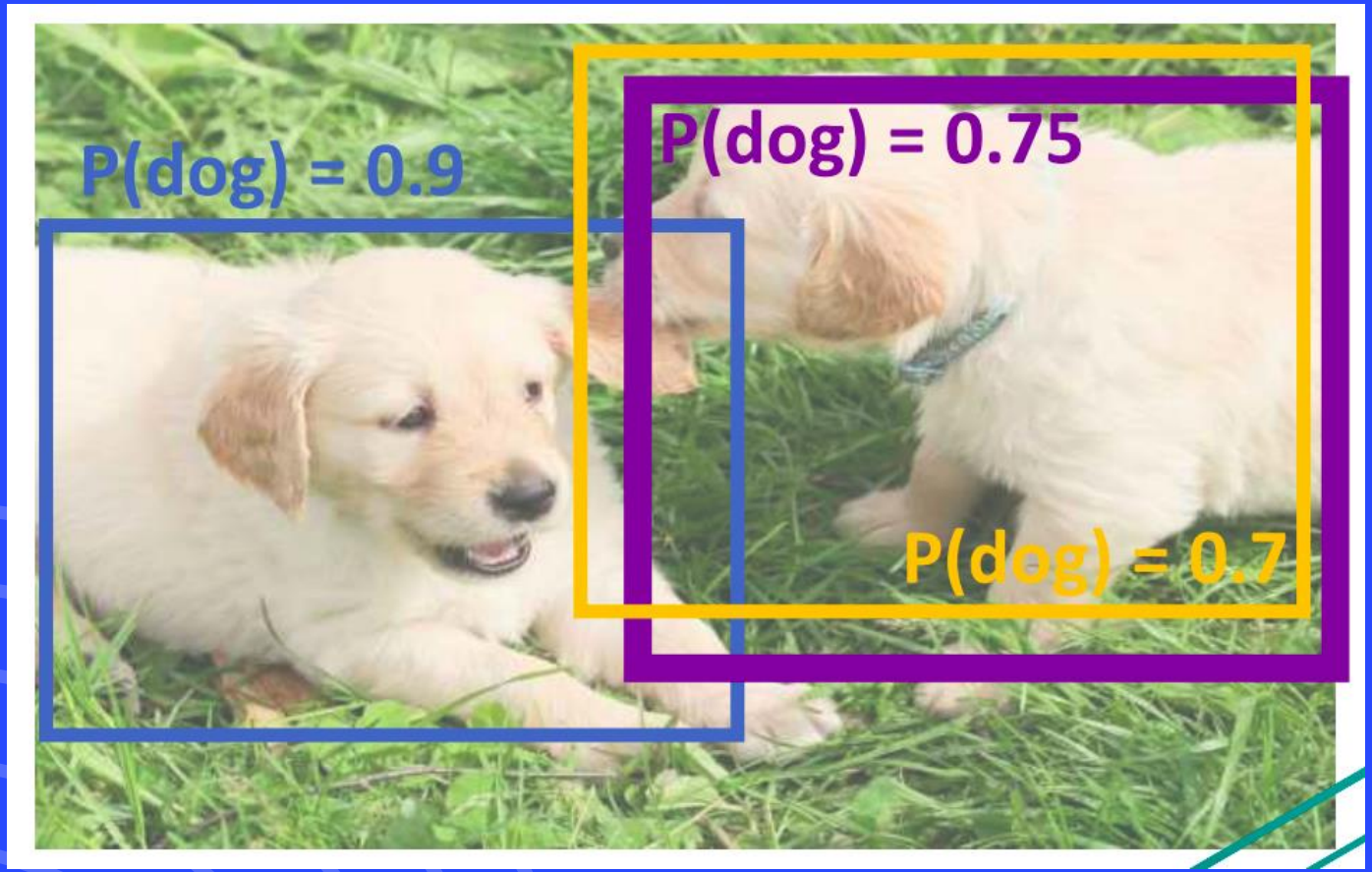
1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

Overlapping Boxes: Non-Max Suppression (NMS)



1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

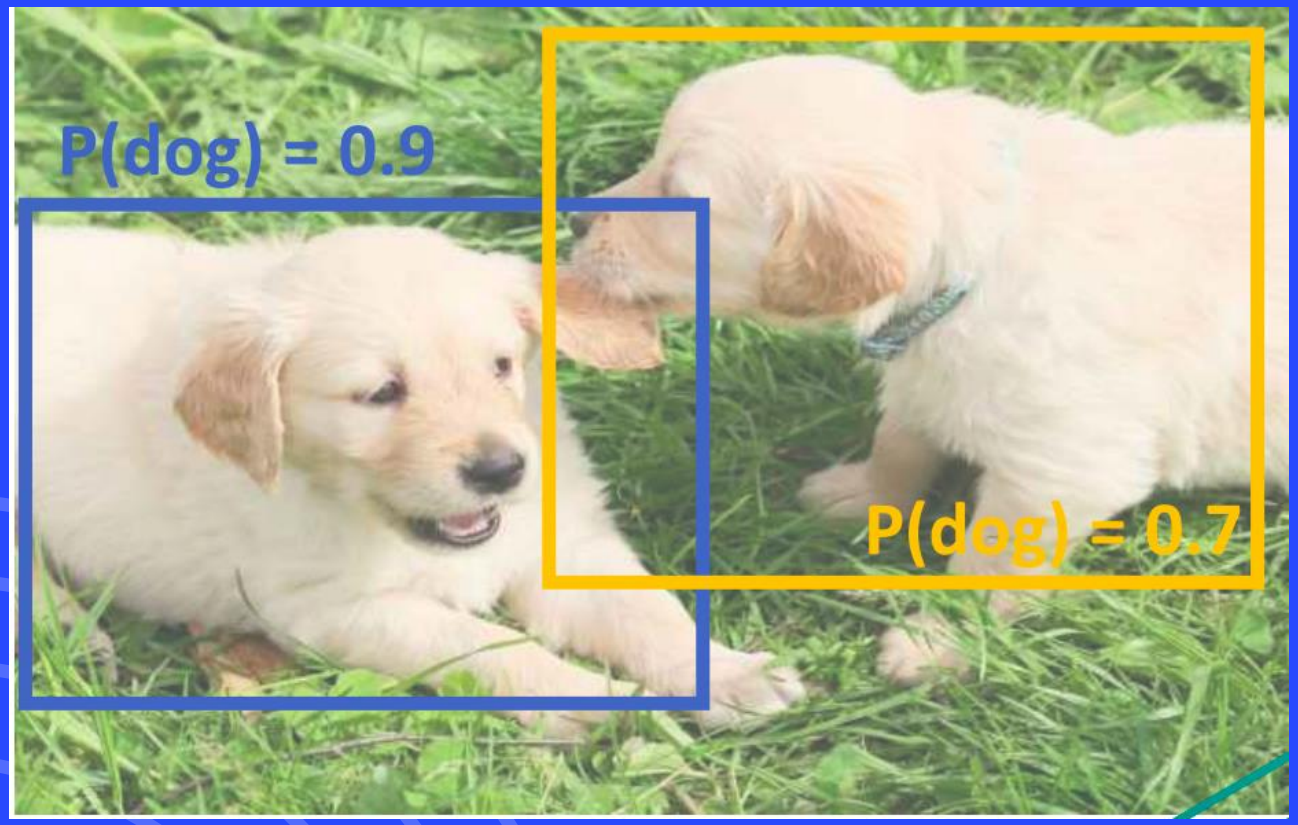
Overlapping Boxes: Non-Max Suppression (NMS)



1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

IoU(■, ■) = 0.74

Overlapping Boxes: Non-Max Suppression (NMS)



1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

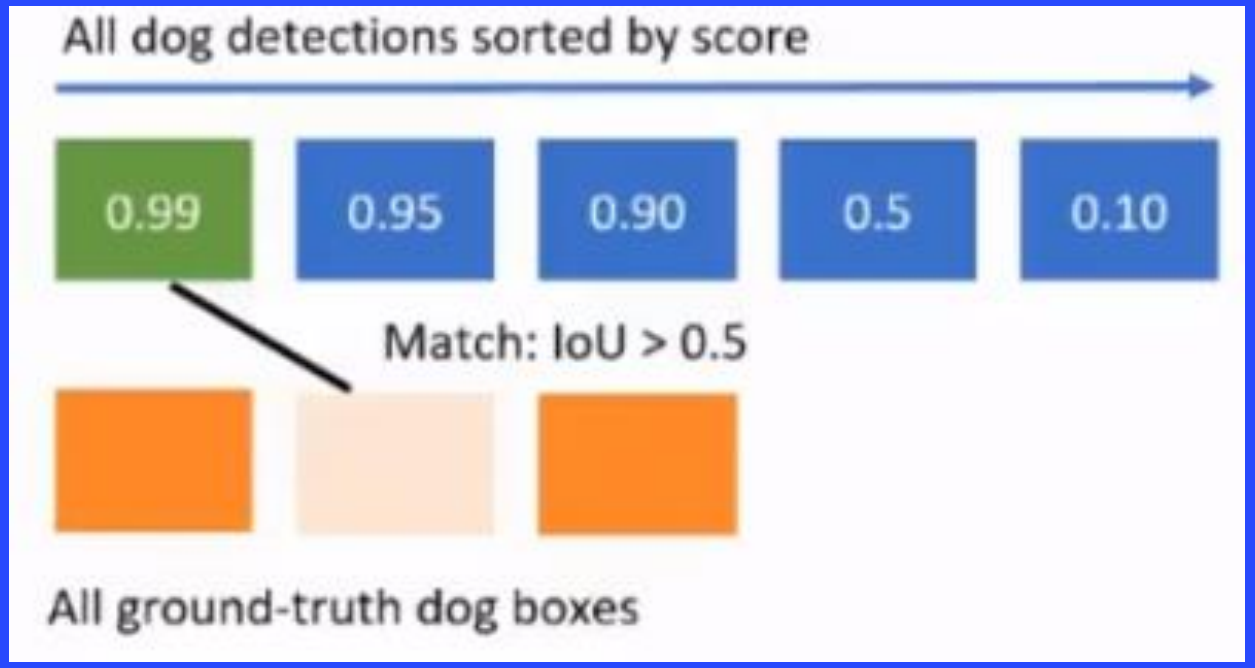
NMS

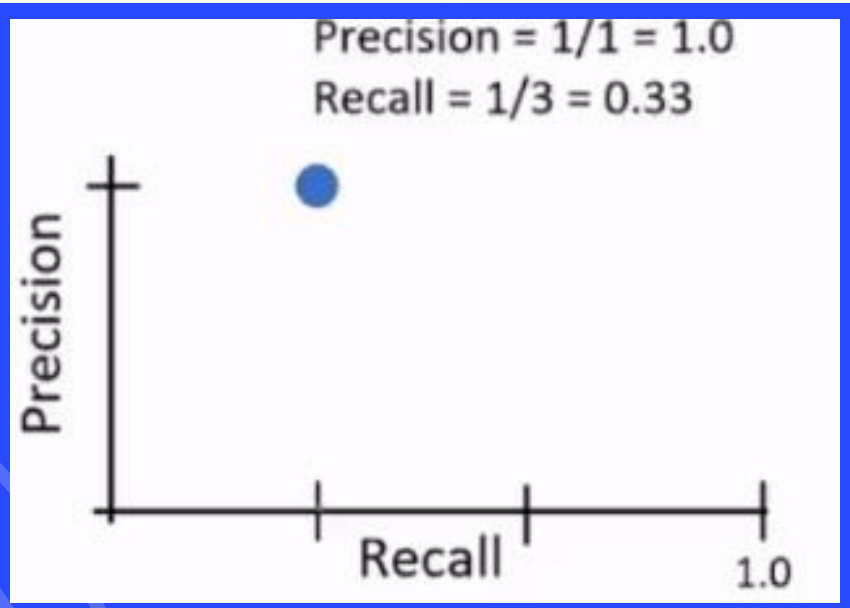
Problem:

NMS may eliminate ”good” boxes when objects are highly overlapping!

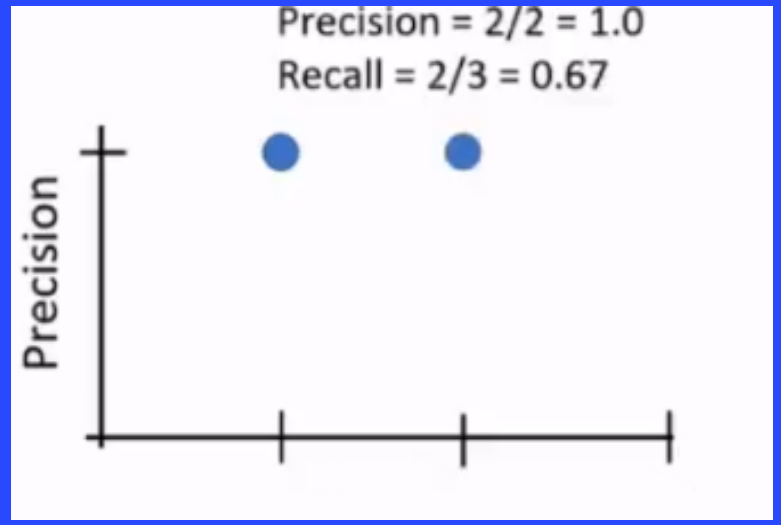
Evaluating Object Detectors: Mean Average Precision (mAP)


All dog detections sorted by score
0.99
0.95
0.90
0.5
0.10
All ground-truth dog boxes

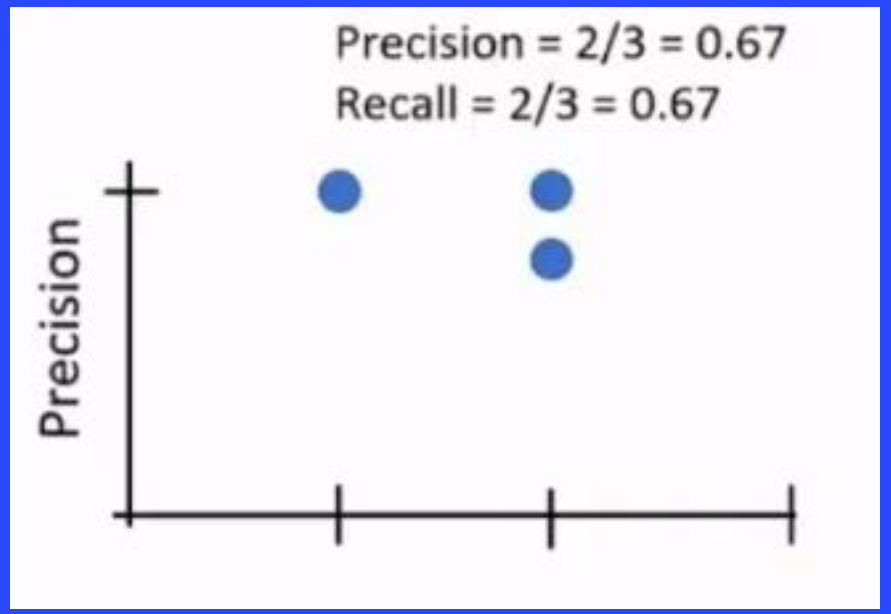

All dog detections sorted by score
0.99
0.95
0.90
0.5
0.10
Match: IoU > 0.5
All ground-truth dog boxes

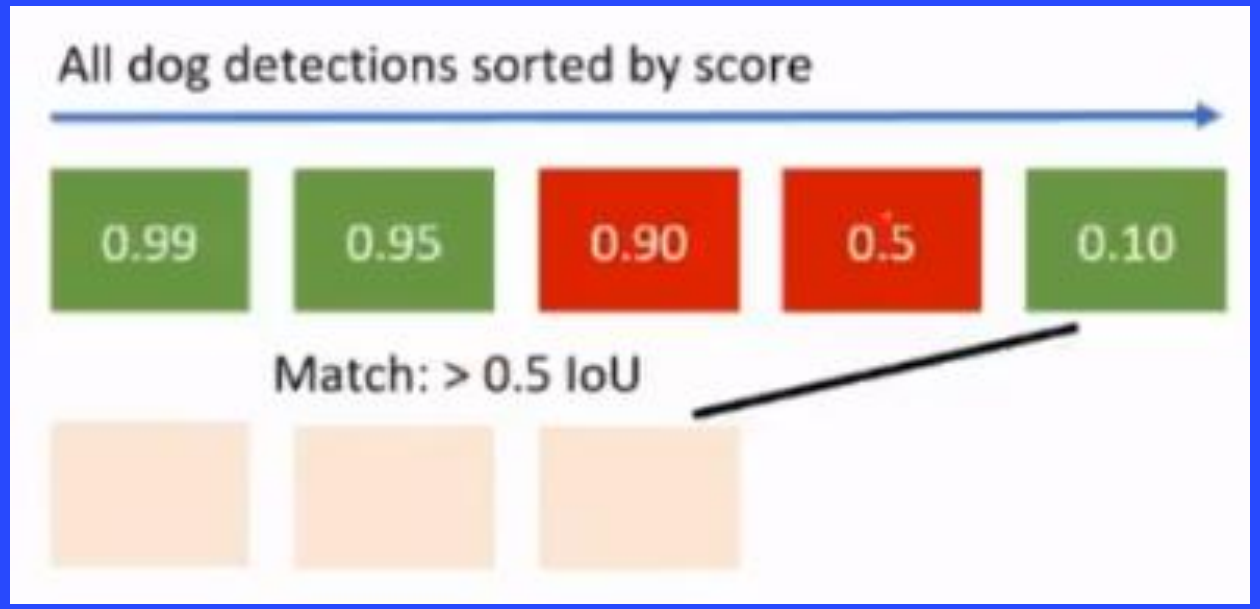

Precision = 1/1 = 1.0
Recall = 1/3 = 0.33
Precision
Recall
1.0

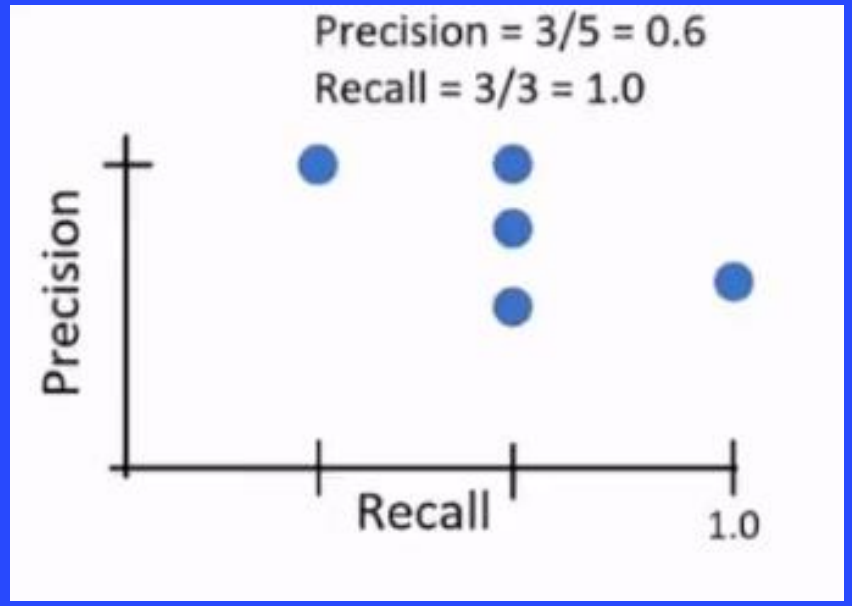

All dog detections sorted by score
0.99
0.95
0.90
0.5
0.10
Match: IoU > 0.5

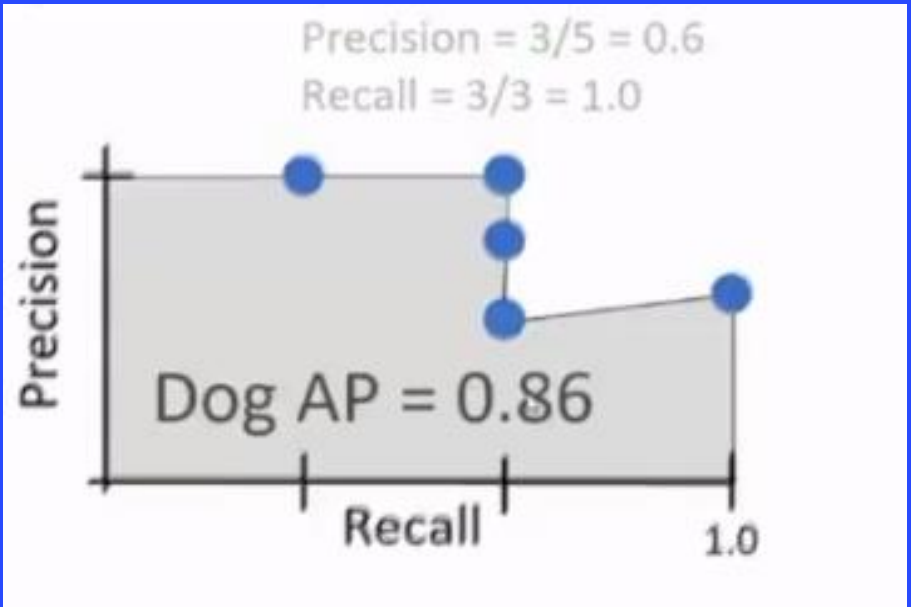

Precision = 2/2 = 1.0
Recall = 2/3 = 0.67
Precision


All dog detections sorted by score
0.99
0.95
0.90
0.5
0.10
No match > 0.5 IoU with GT


Precision = 2/3 = 0.67
Recall = 2/3 = 0.67
Precision


All dog detections sorted by score
0.99
0.95
0.90
0.5
0.10
Match: > 0.5 IoU


Precision = 3/5 = 0.6
Recall = 3/3 = 1.0
Precision
Recall
1.0



Car AP = 0.65

Cat AP = 0.80

Dog AP = 0.86

mAP@0.5 = 0.77

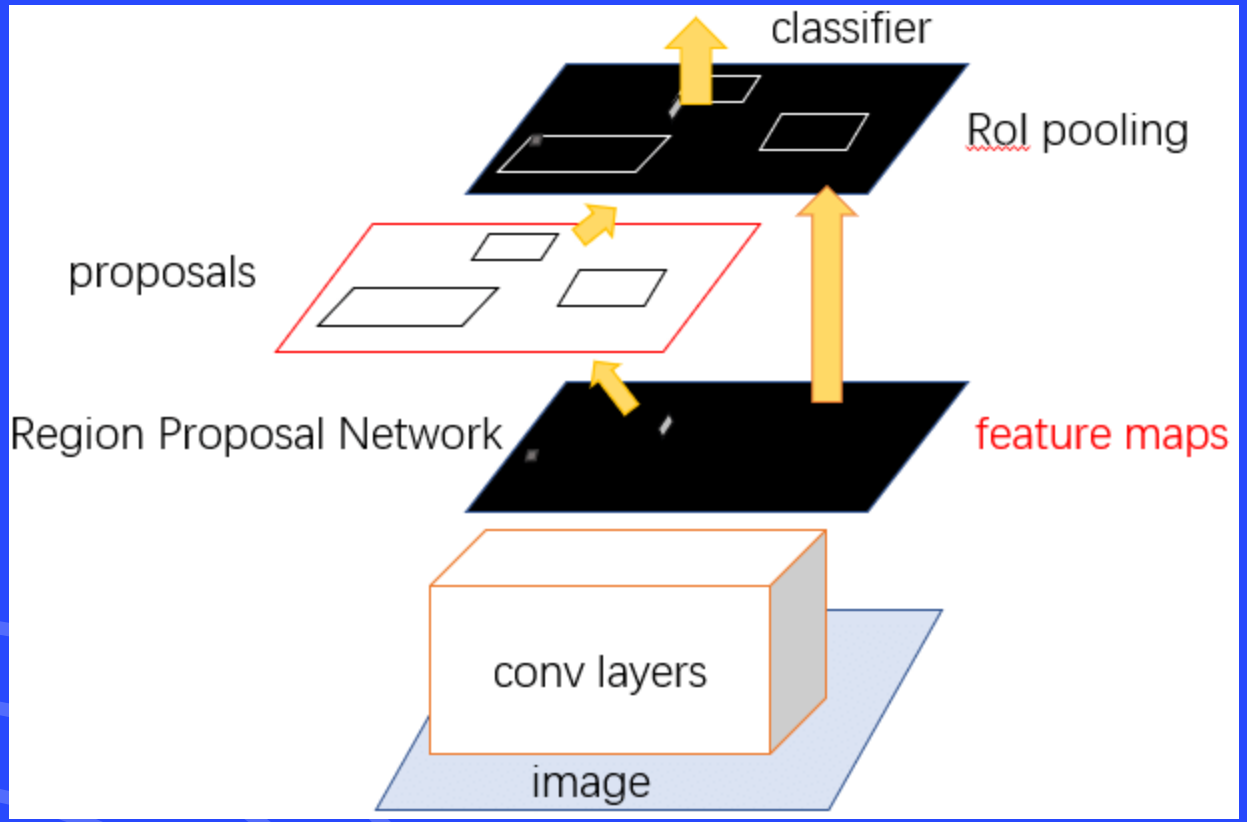
mAP@0.5 = 0.77
mAP@0.55 = 0.71
mAP@0.60 = 0.65
...
mAP@0.95 = 0.2

COCO mAP = 0.4

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve

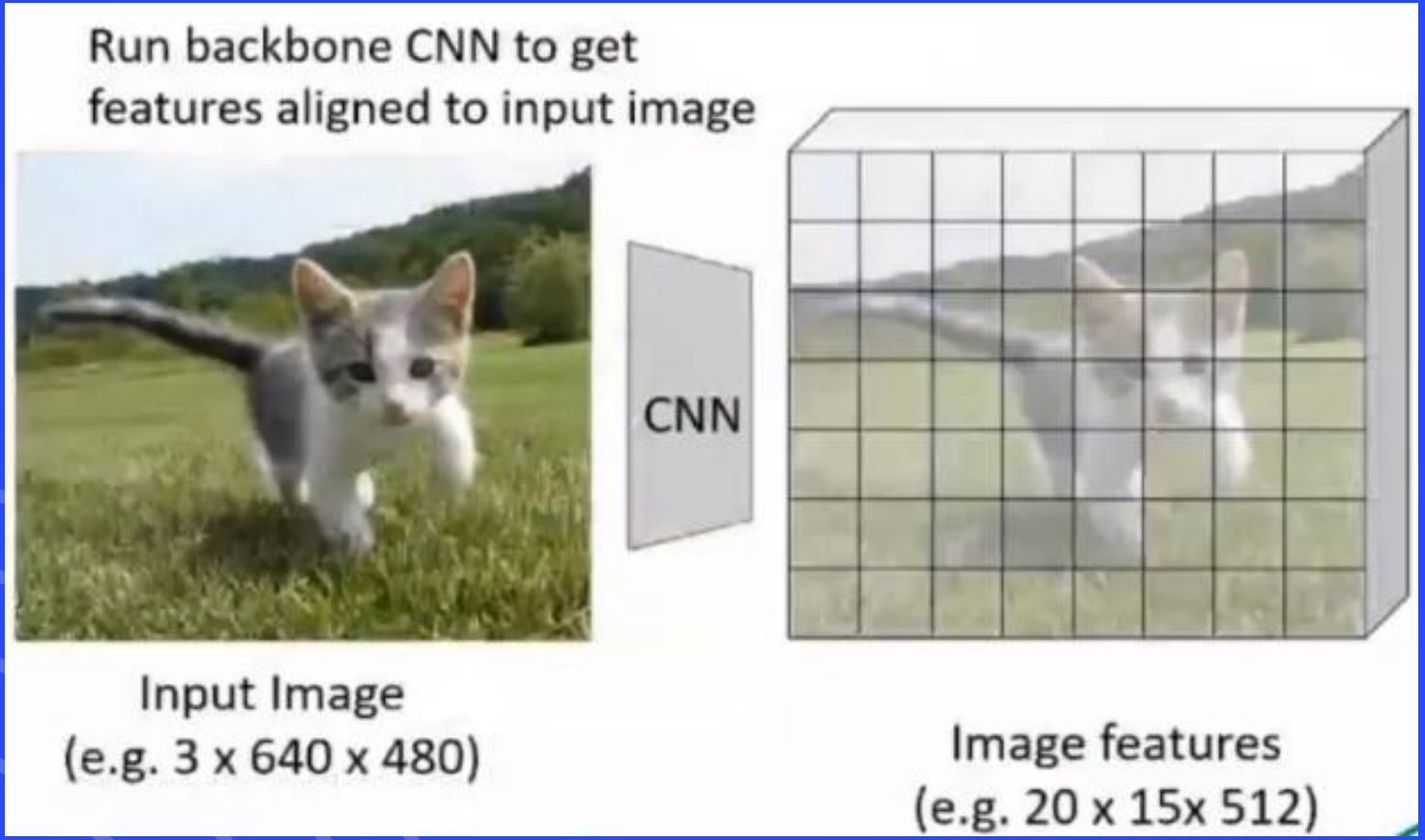
    2. Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category
4. For "COCO mAP": Compute mAP@thresh for each IoU threshold (0.5, 0.55, 0.6, ..., 0.95) and take average

Faster-RCNN

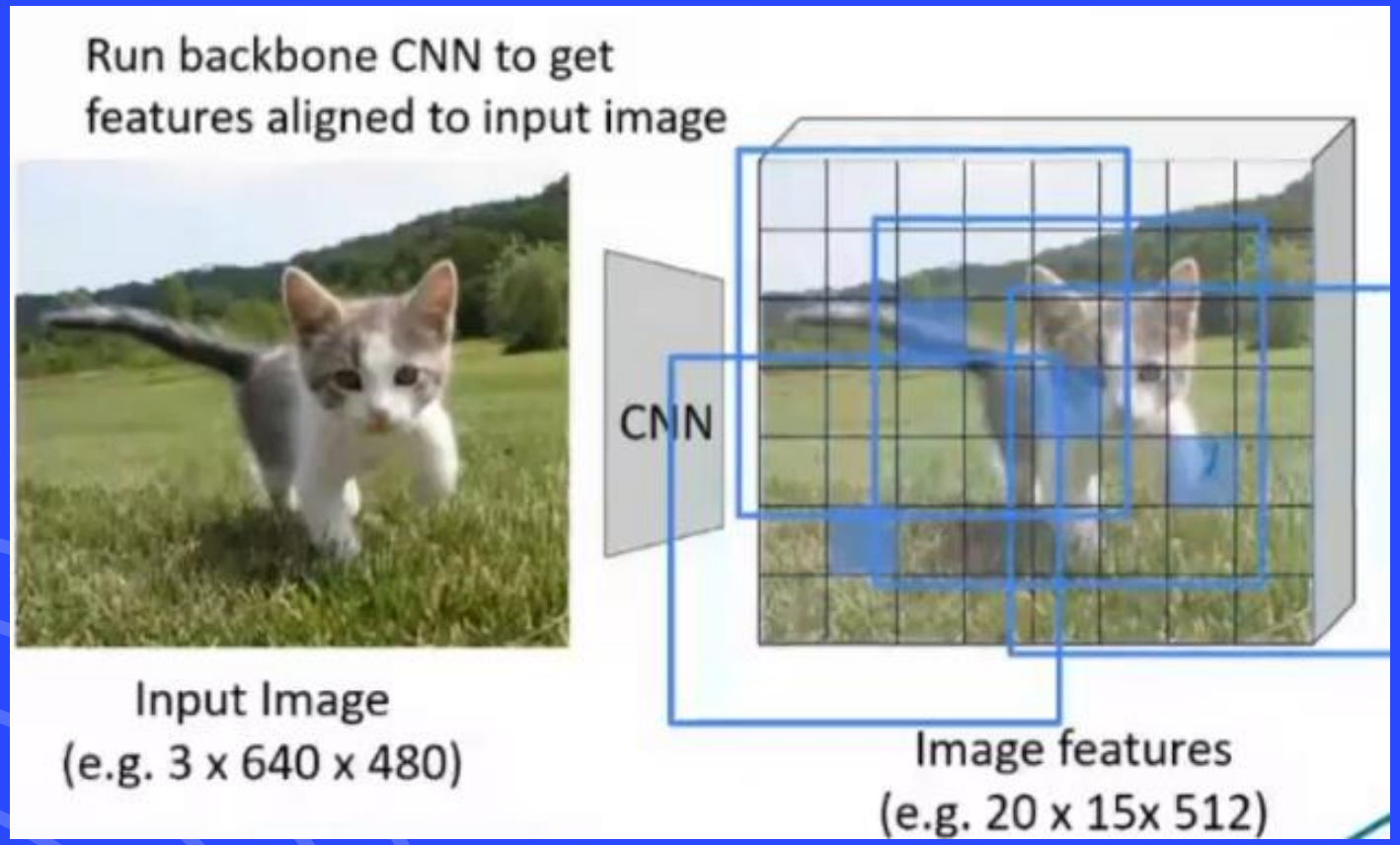


1. **RPN classification**: anchor box is object / not an object
2. **RPN regression**: predict transform from anchor box to proposal box
3. **Object classification**: classify proposals as background / object class
4. **Object regression**: predict transform from proposal box to object box

Faster-RCNN-RPN

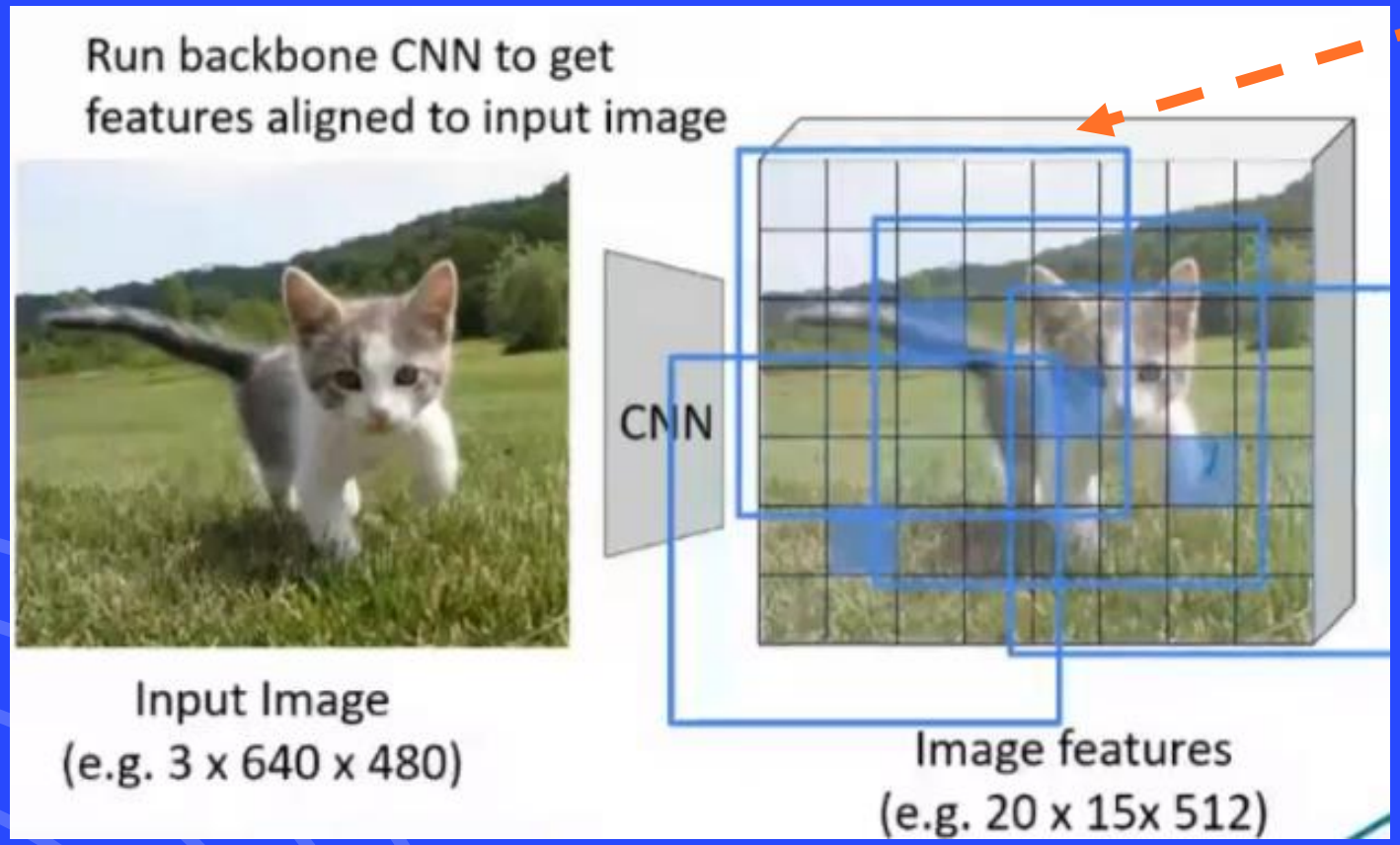


Faster-RCNN-RPN

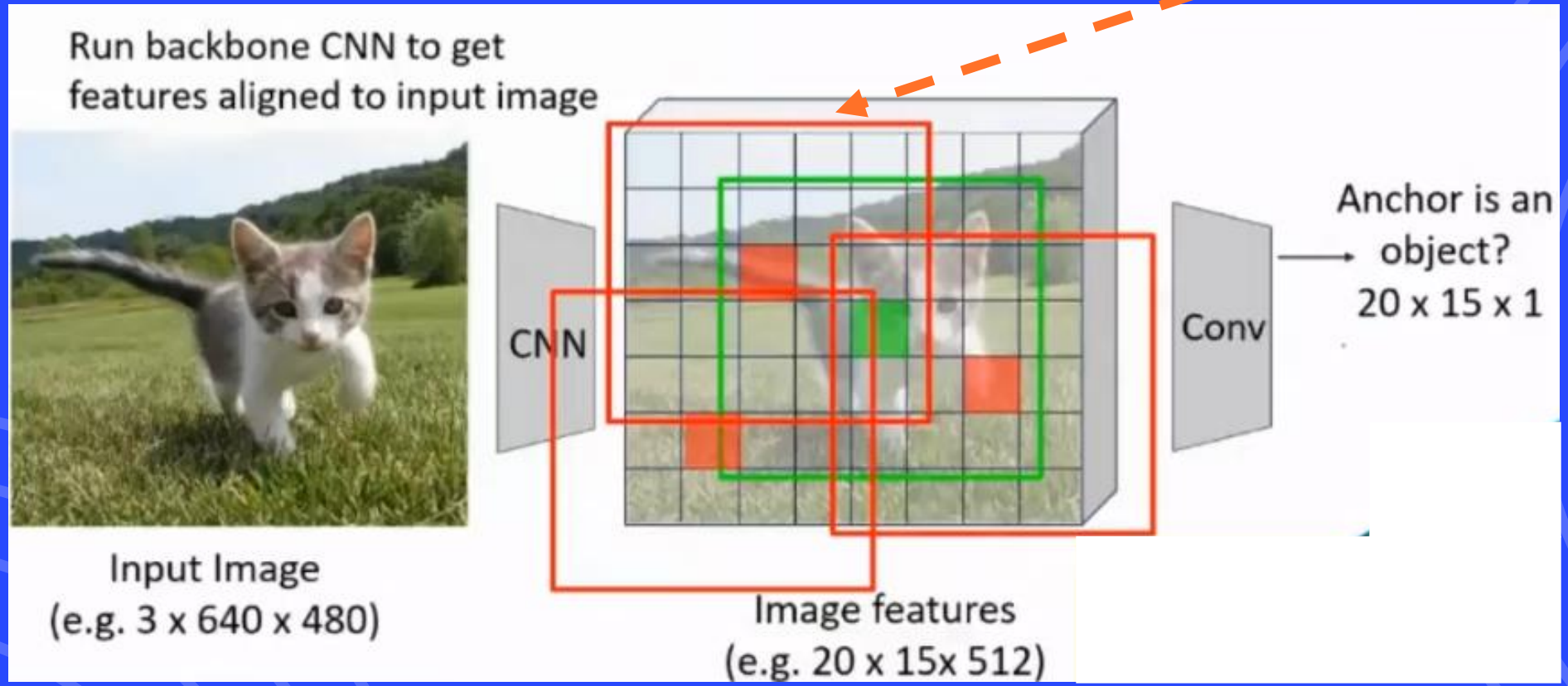


Faster-RCNN-RPN

Anchor box

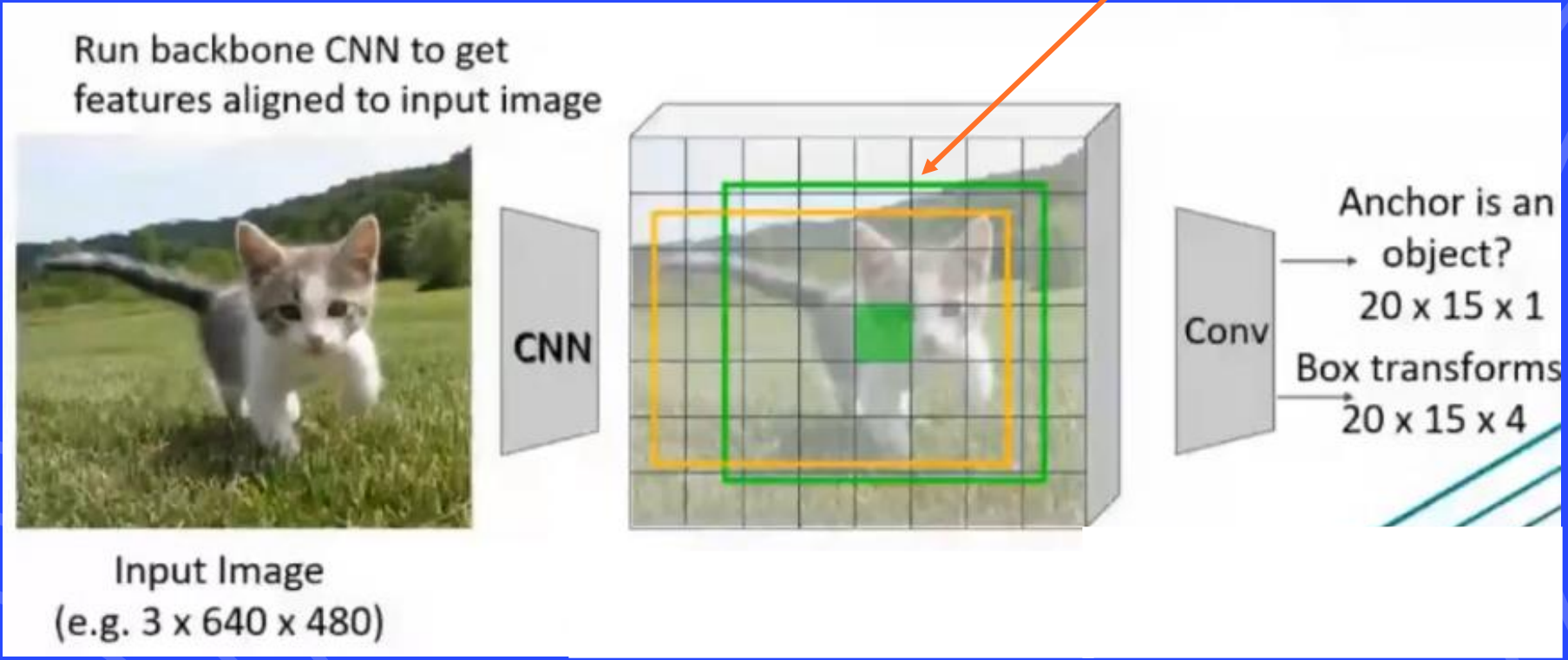


Faster-RCNN-RPN

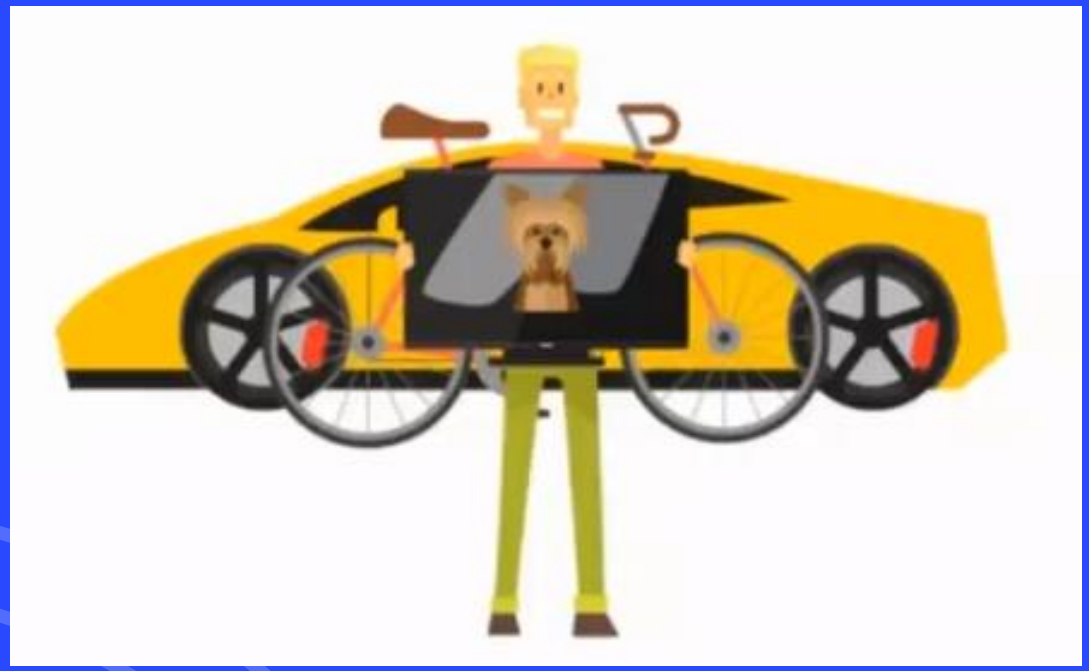


Faster-RCNN-RPN

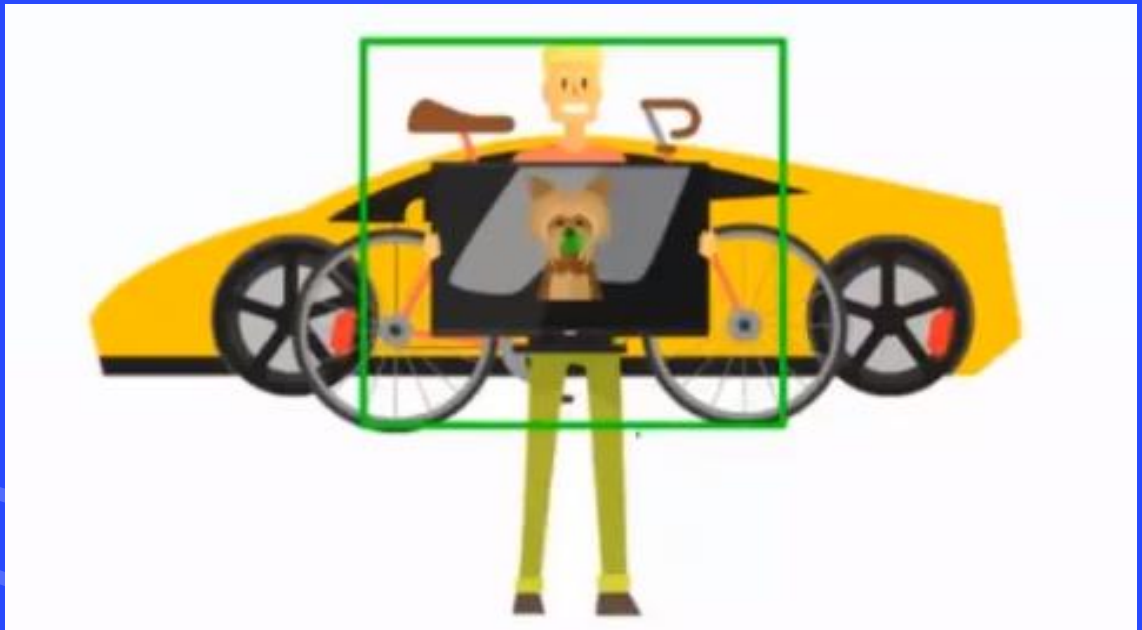
From anchor box to object box



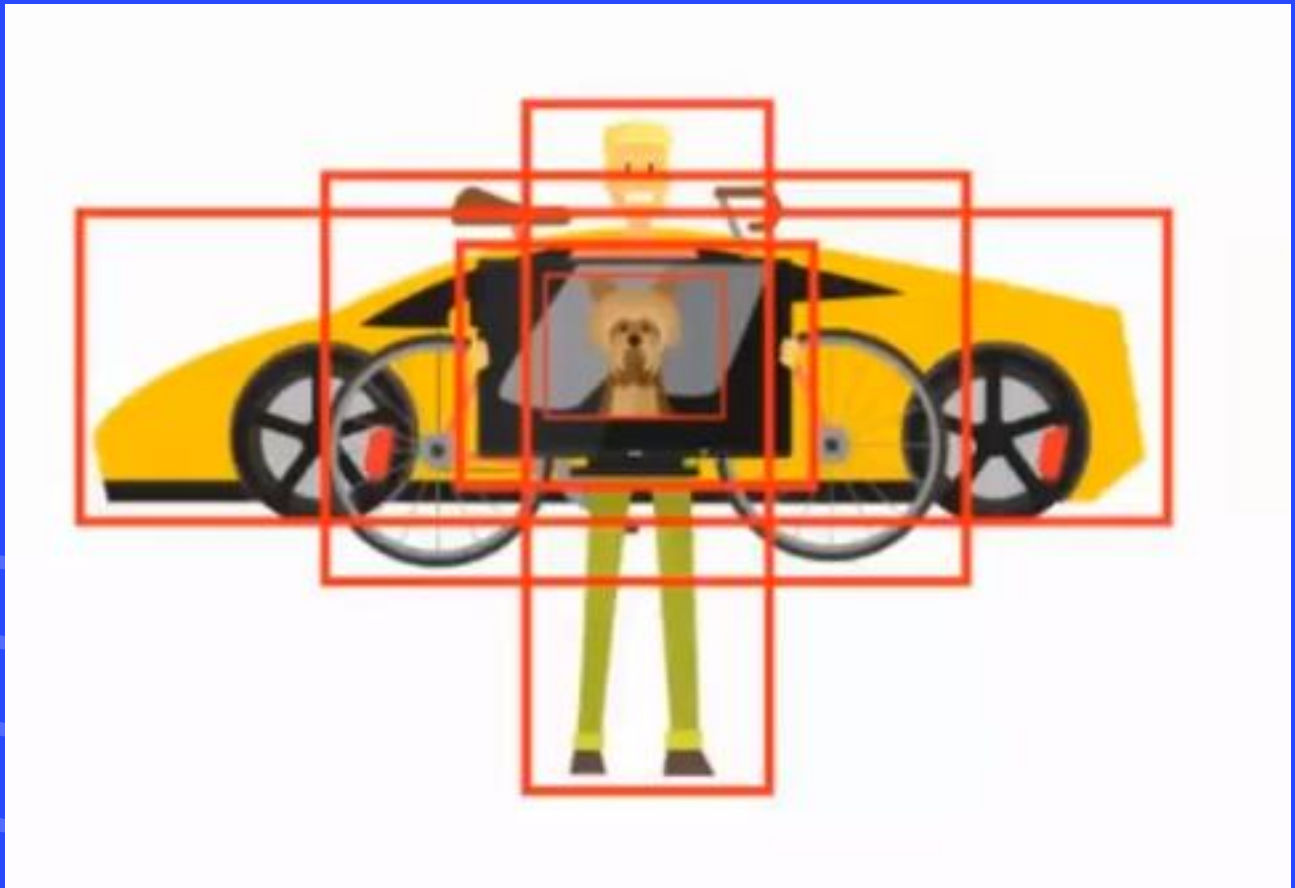
Faster-RCNN-RPN-anchor box



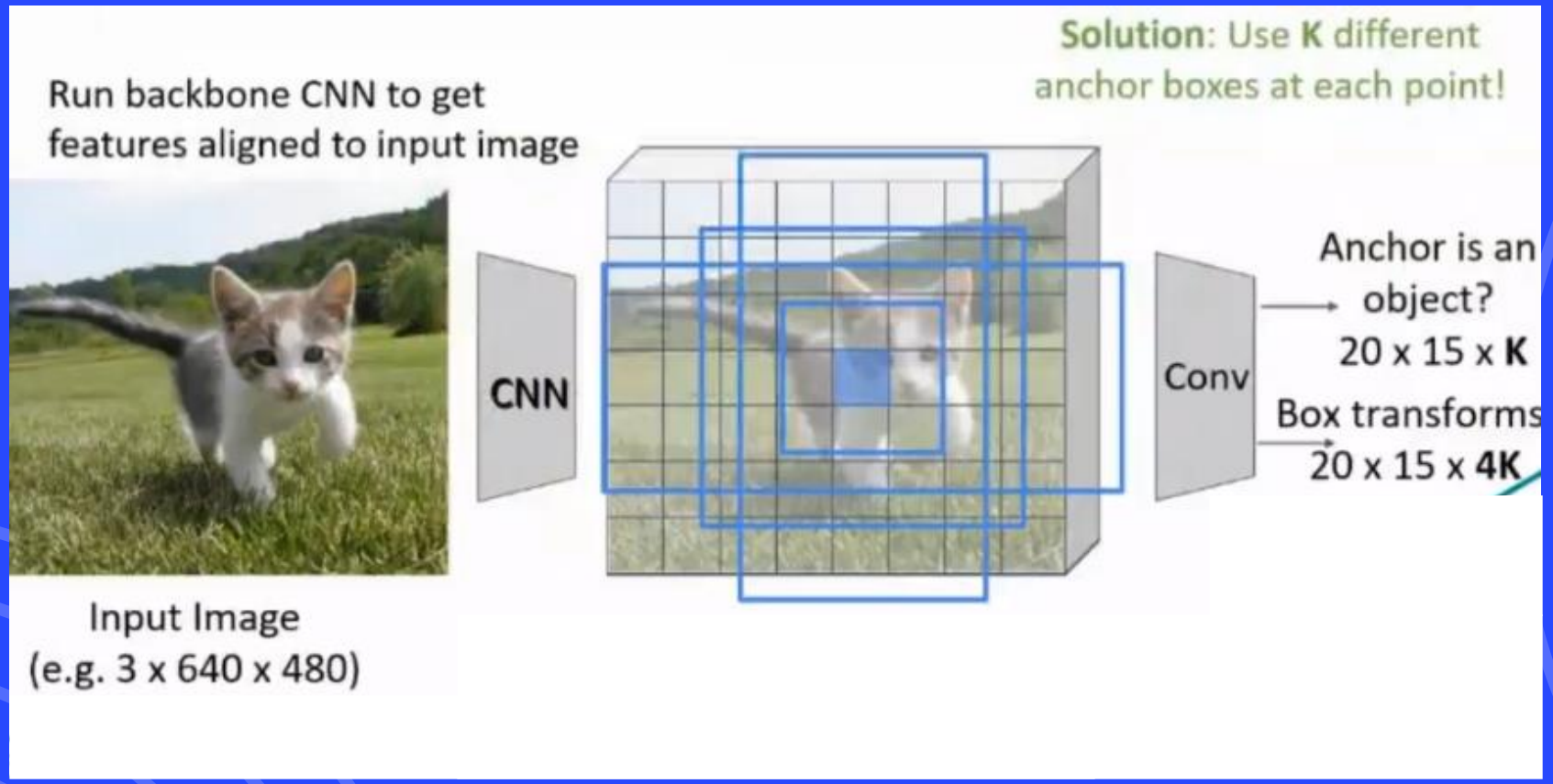
Faster-RCNN-RPN-anchor box



Faster-RCNN-RPN

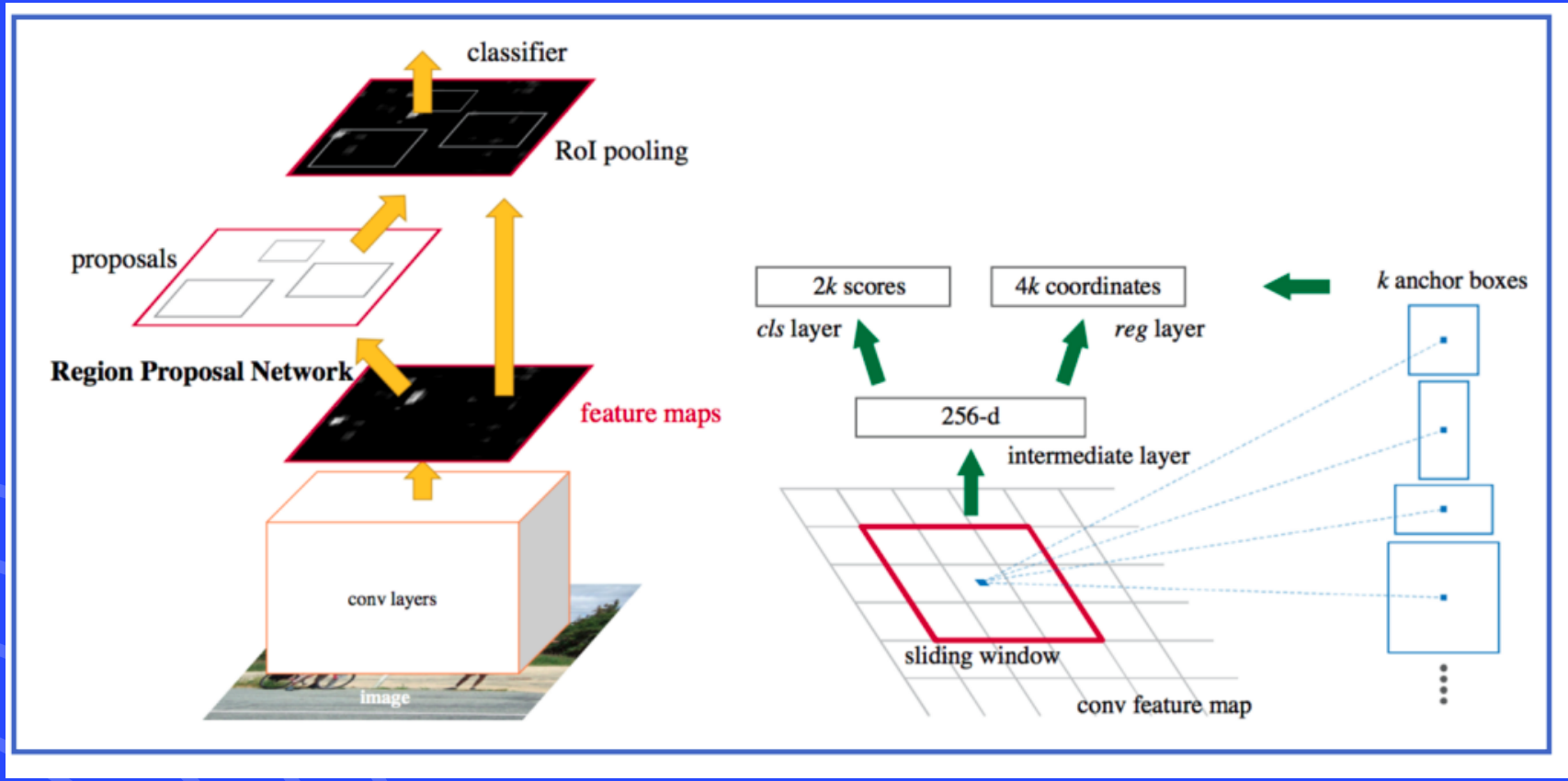


Faster-RCNN-RPN-anchor box

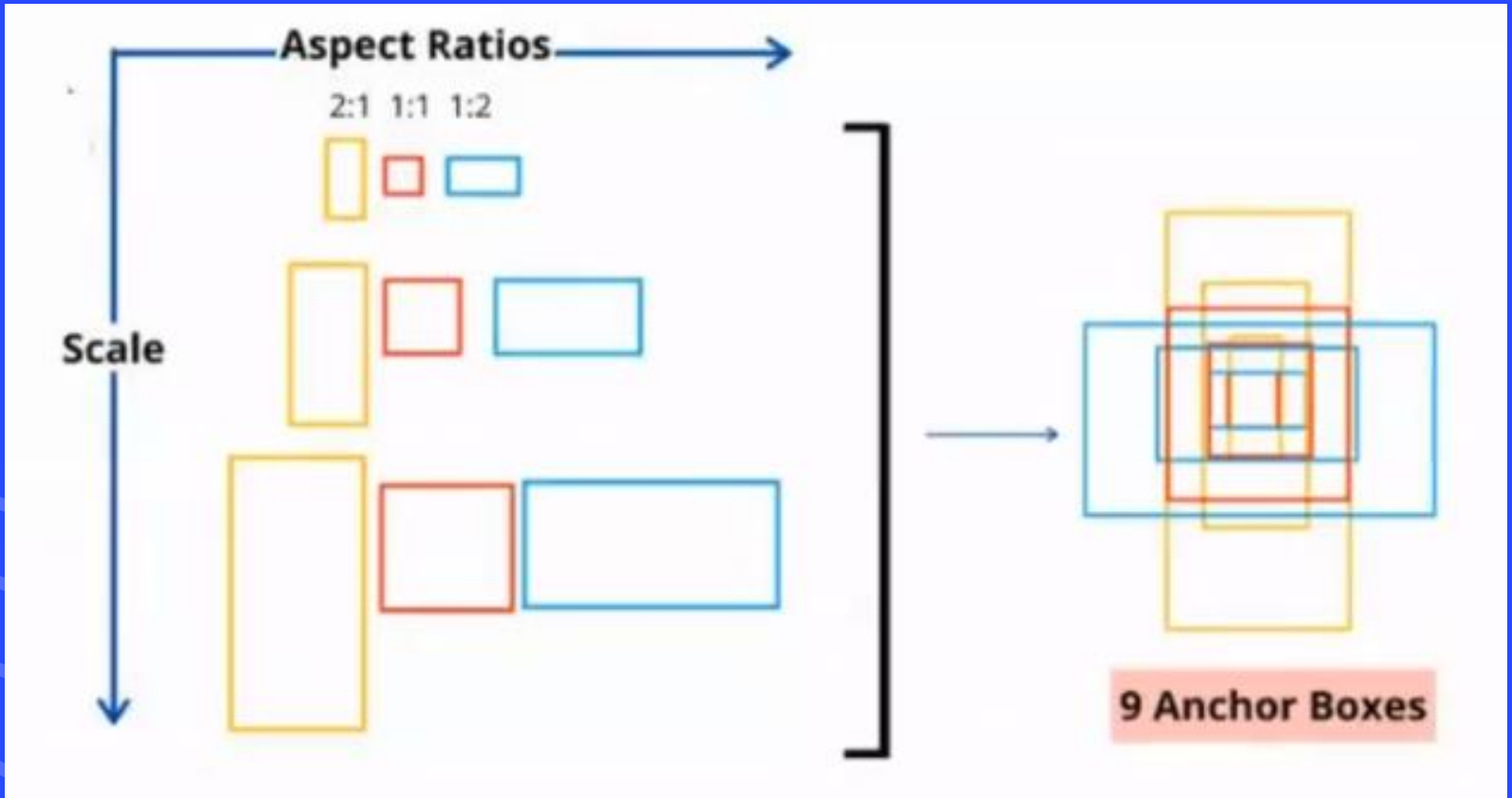


At test time sort all k*15*20 anchor box by their scores and take the top ~300

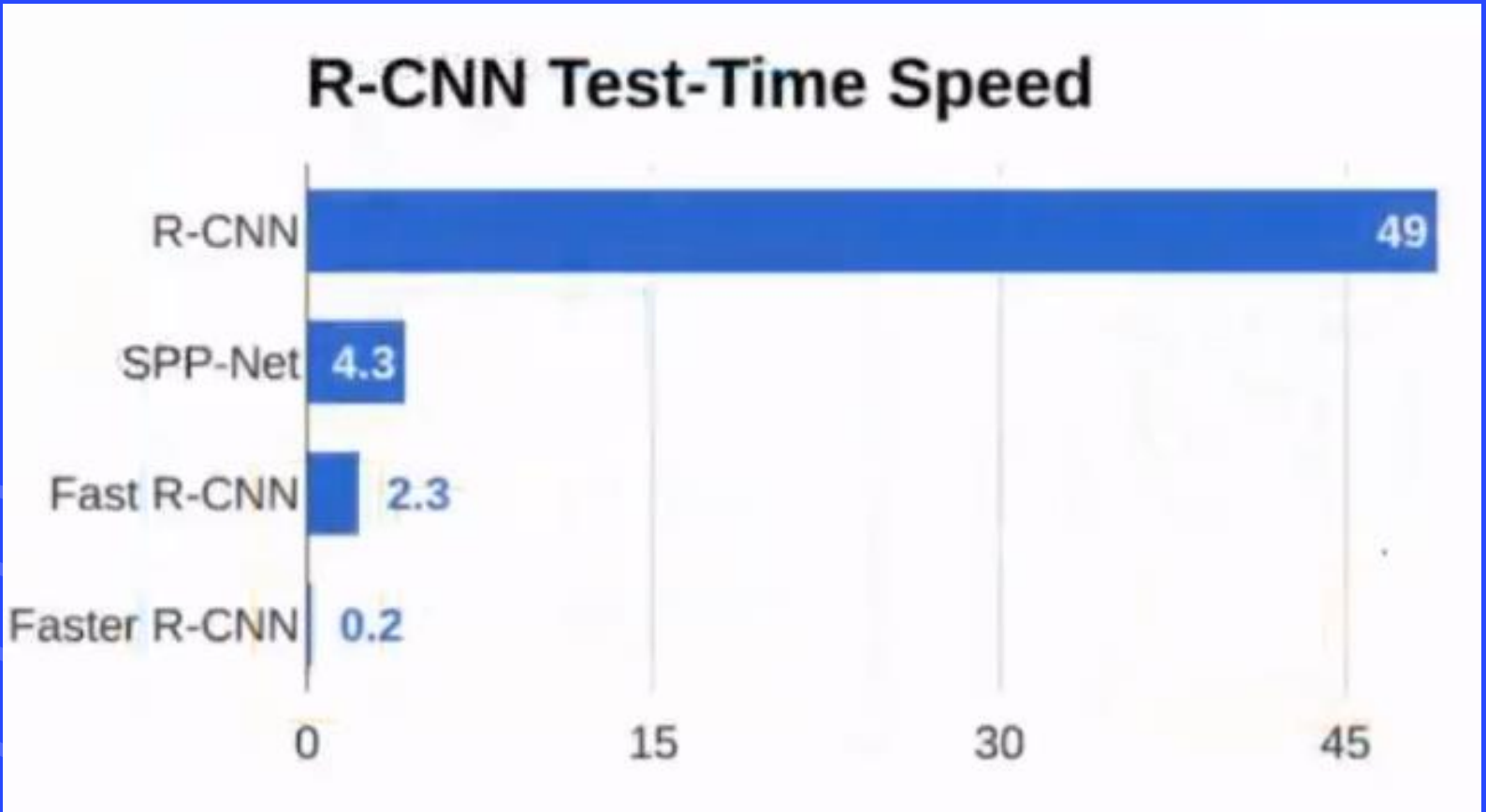
Faster-RCNN-RPN-anchor box



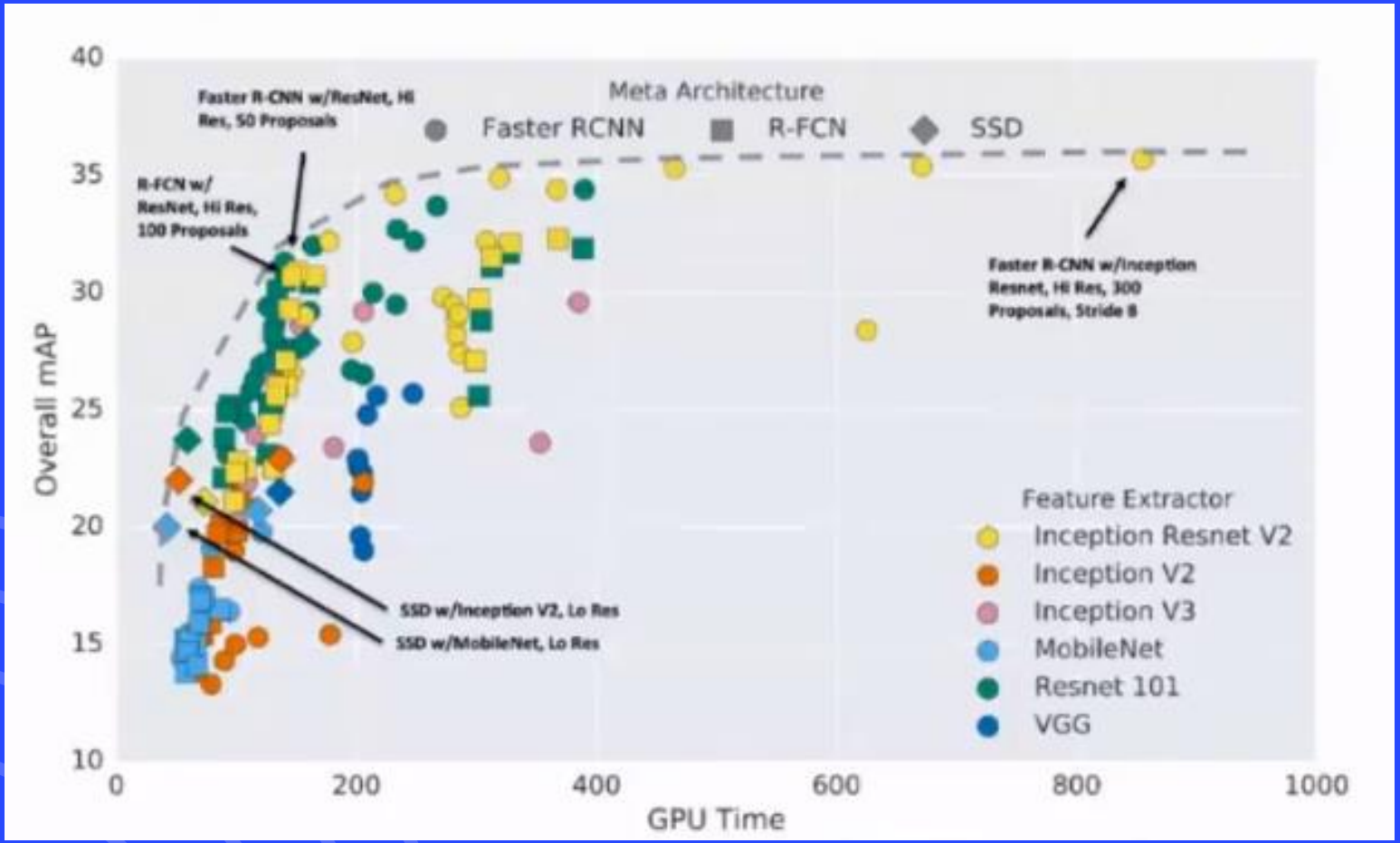
Faster-RCNN-RPN



Faster-RCNN-RPN



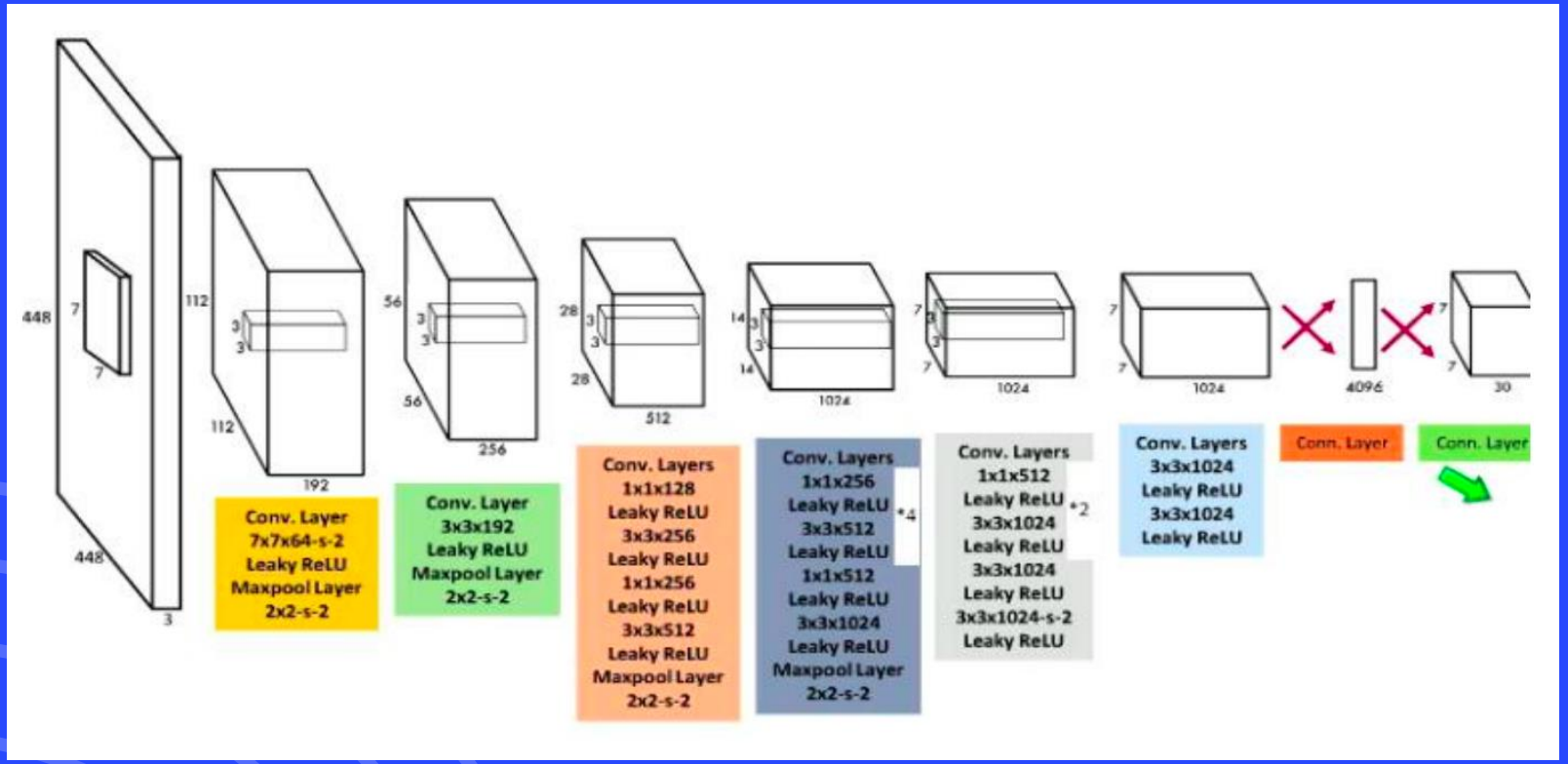
Faster-RCNN-RPN



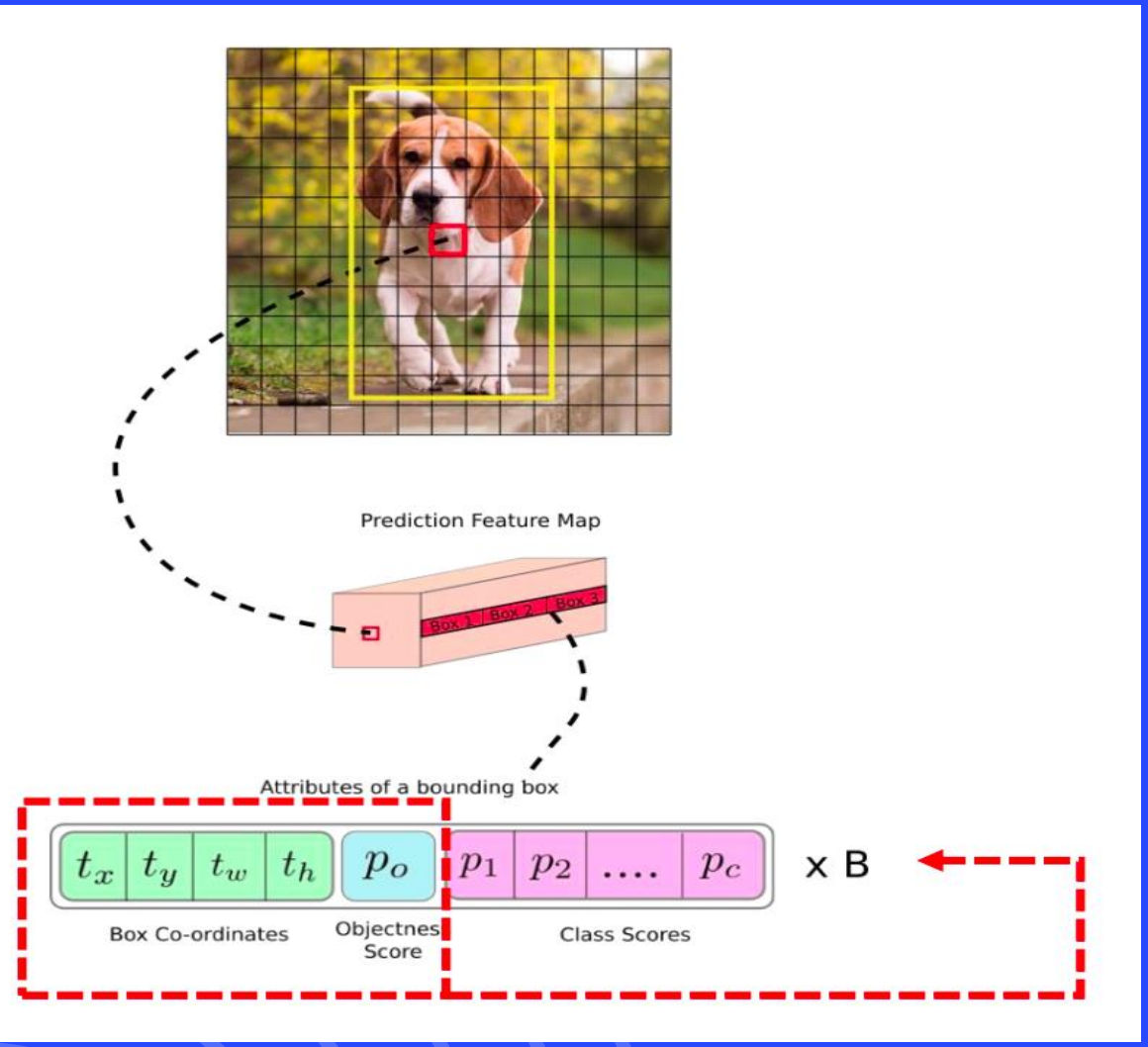
YOLO – You Only Look Once

- **Super fast (21~155 fps)**
- **Finds objects in image grids at parallel**
- **Only slightly worse performance than Faster R-CNN**

YOLO – You Only Look Once

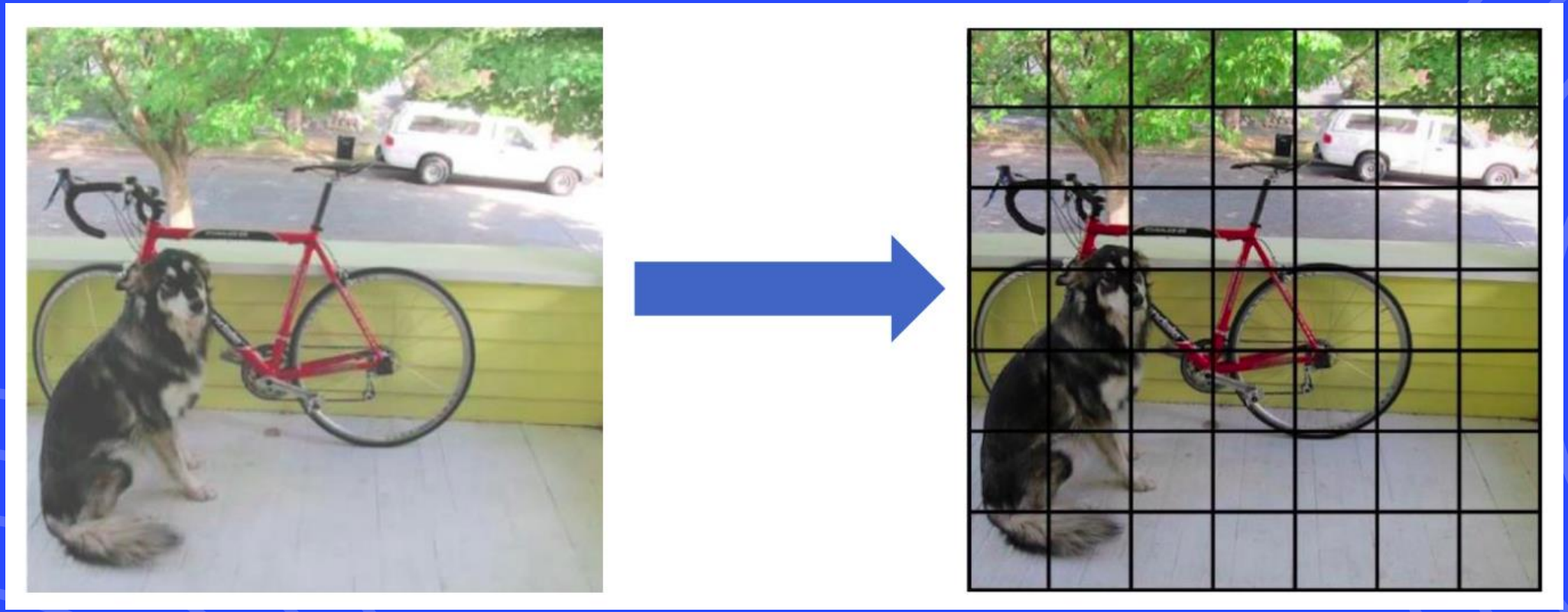


YOLO – You Only Look Once

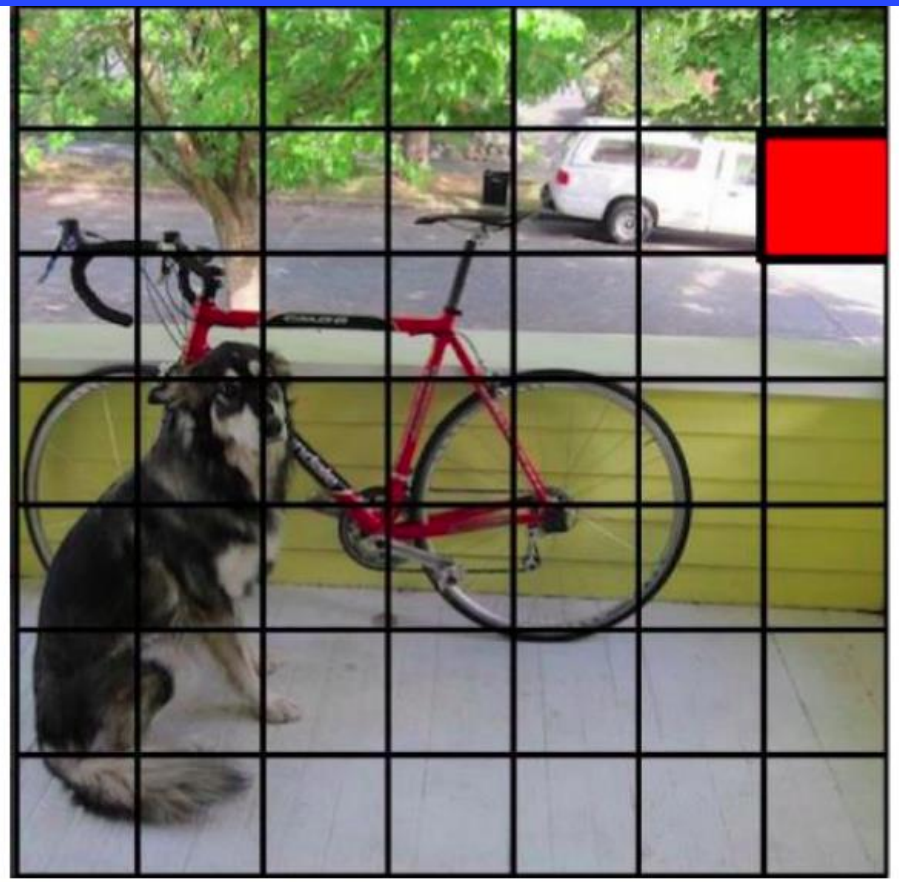


- Give an image and Divide it into Grid S x S
- If the center of image falls in to the grid cell , that grid cell is responsible Each cell predicts B bounding boxes
- Five predictions : x , y , w , h , conf Each cell predicts C class probabilities
- B x 5 bounding boxes

YOLO – You Only Look Once-test



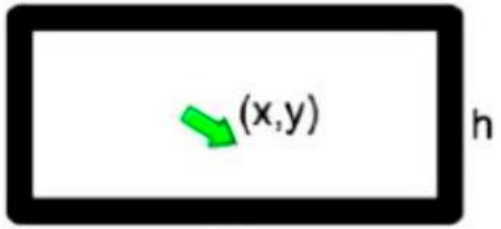
YOLO – You Only Look Once-test



YOLO – You Only Look Once-test

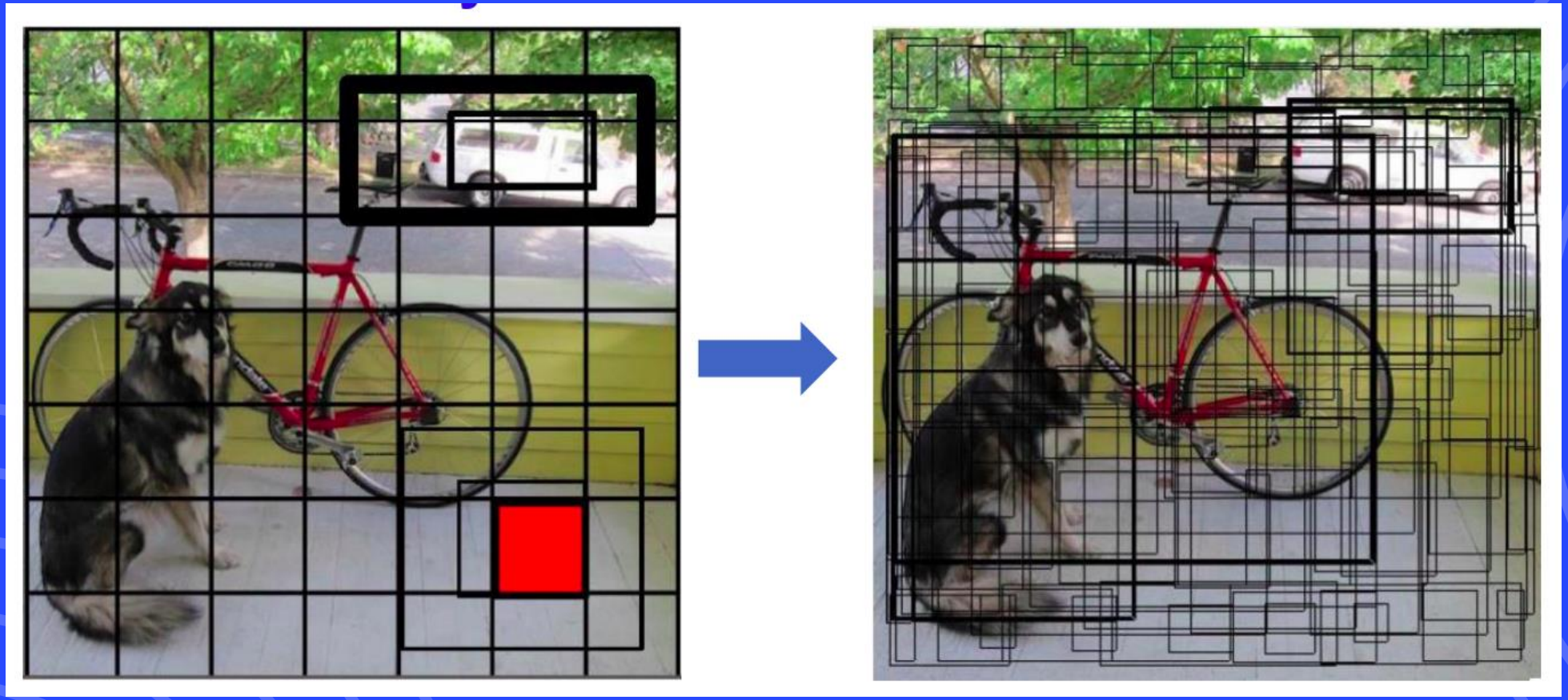
B = 2

each box predict:

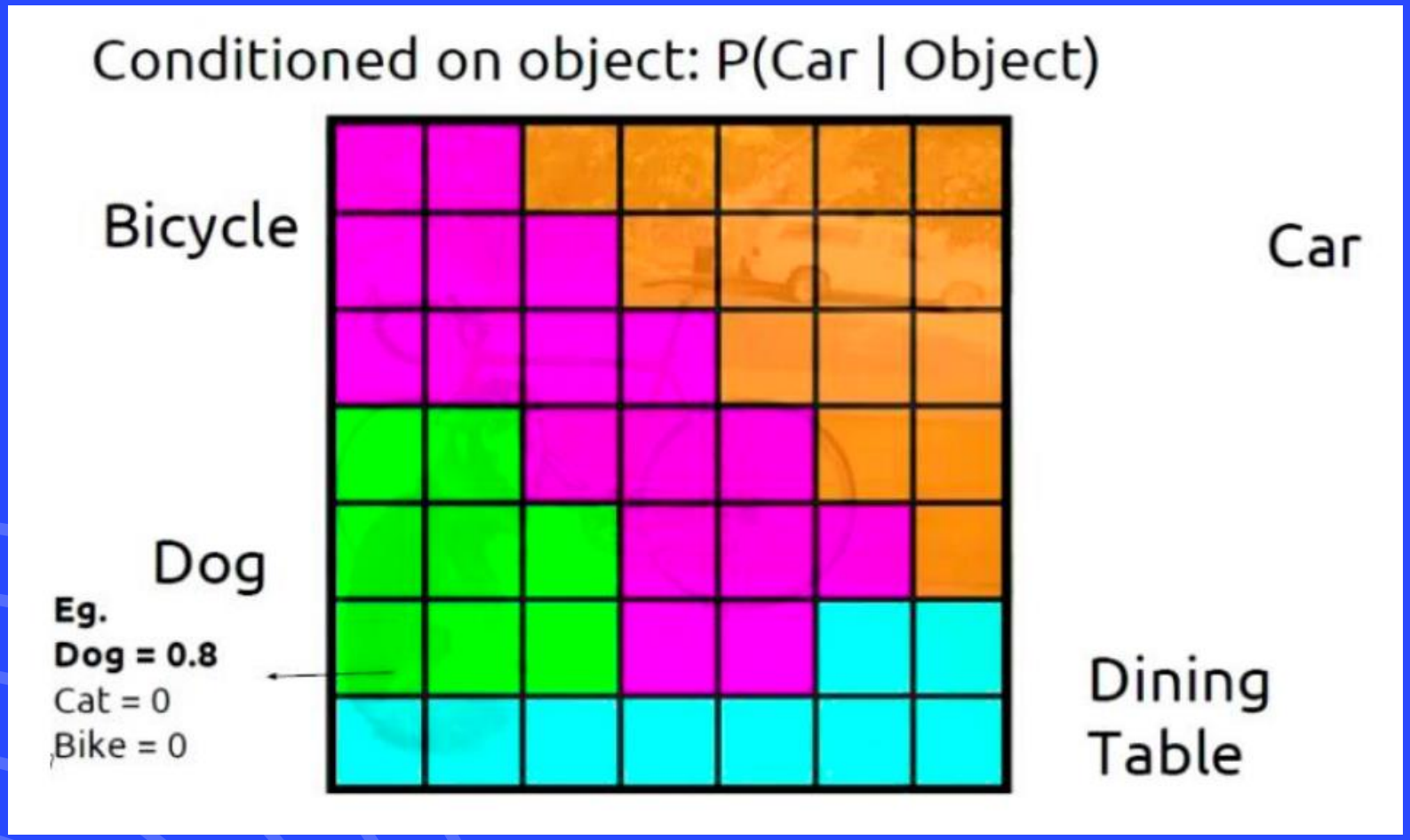


P(Object): probability that the box contains an object

YOLO – You Only Look Once-test

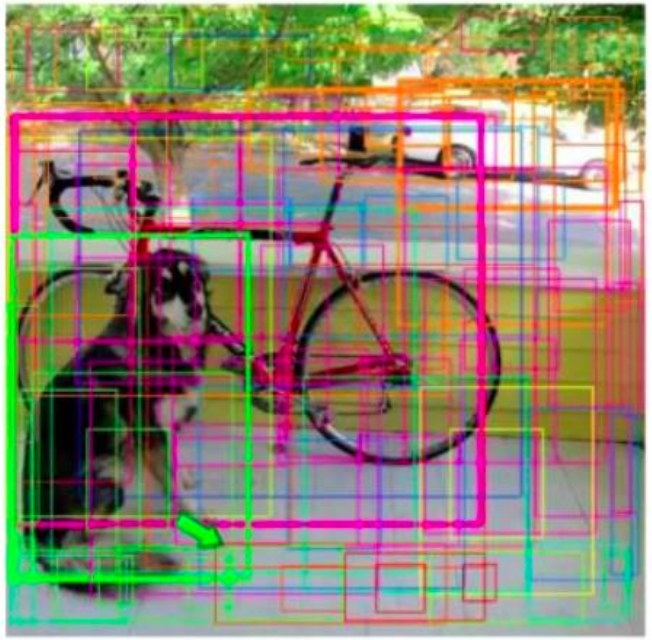


YOLO – You Only Look Once-test

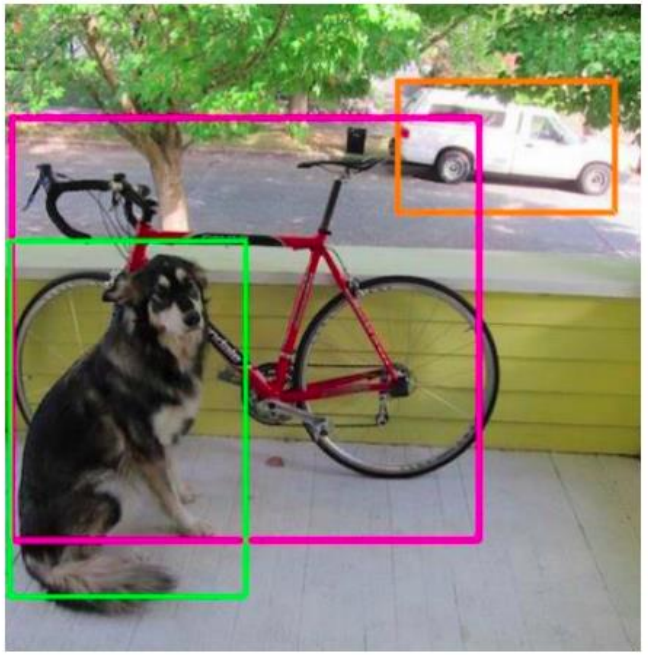


YOLO – You Only Look Once-test

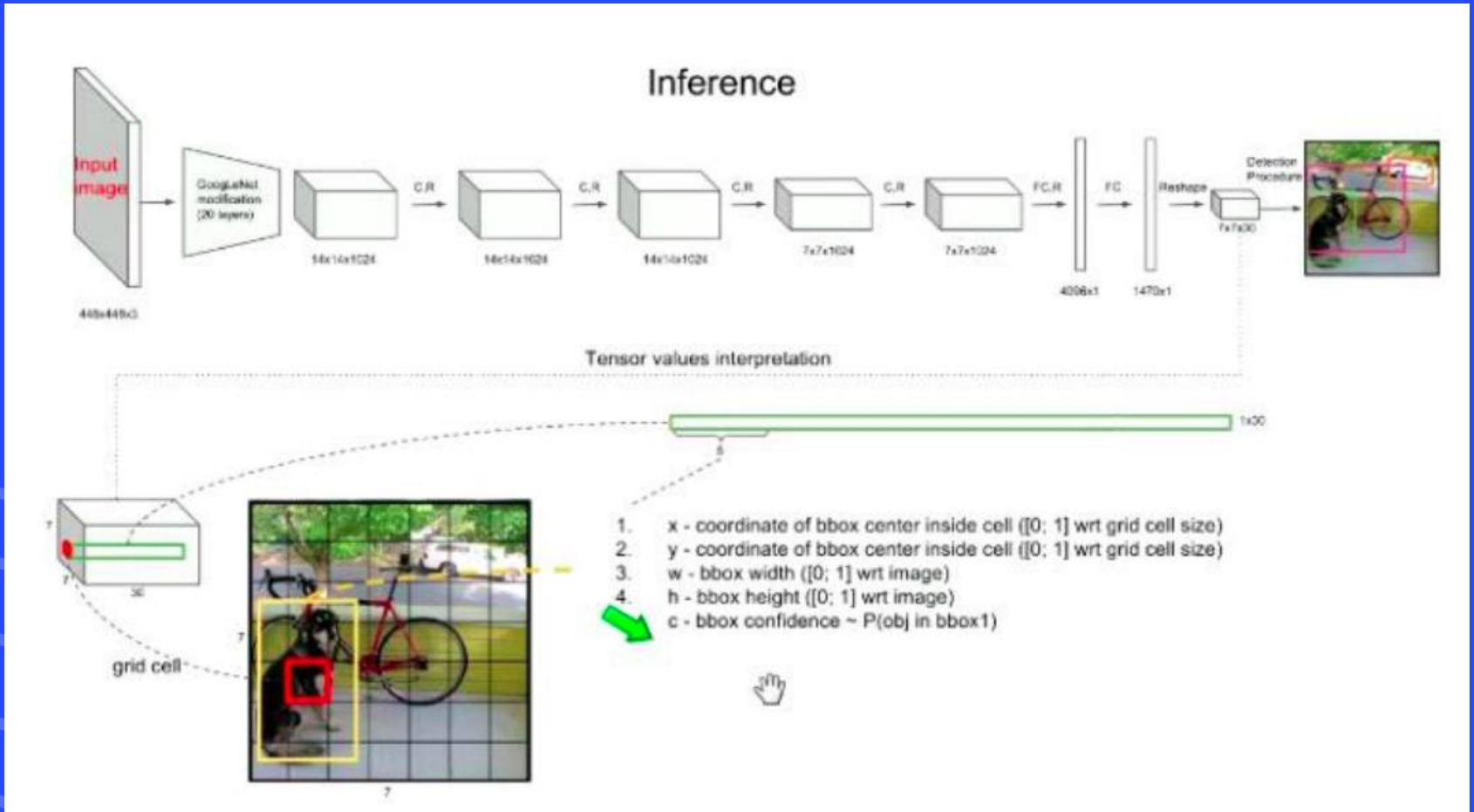
Then we combine the box and class predictions.



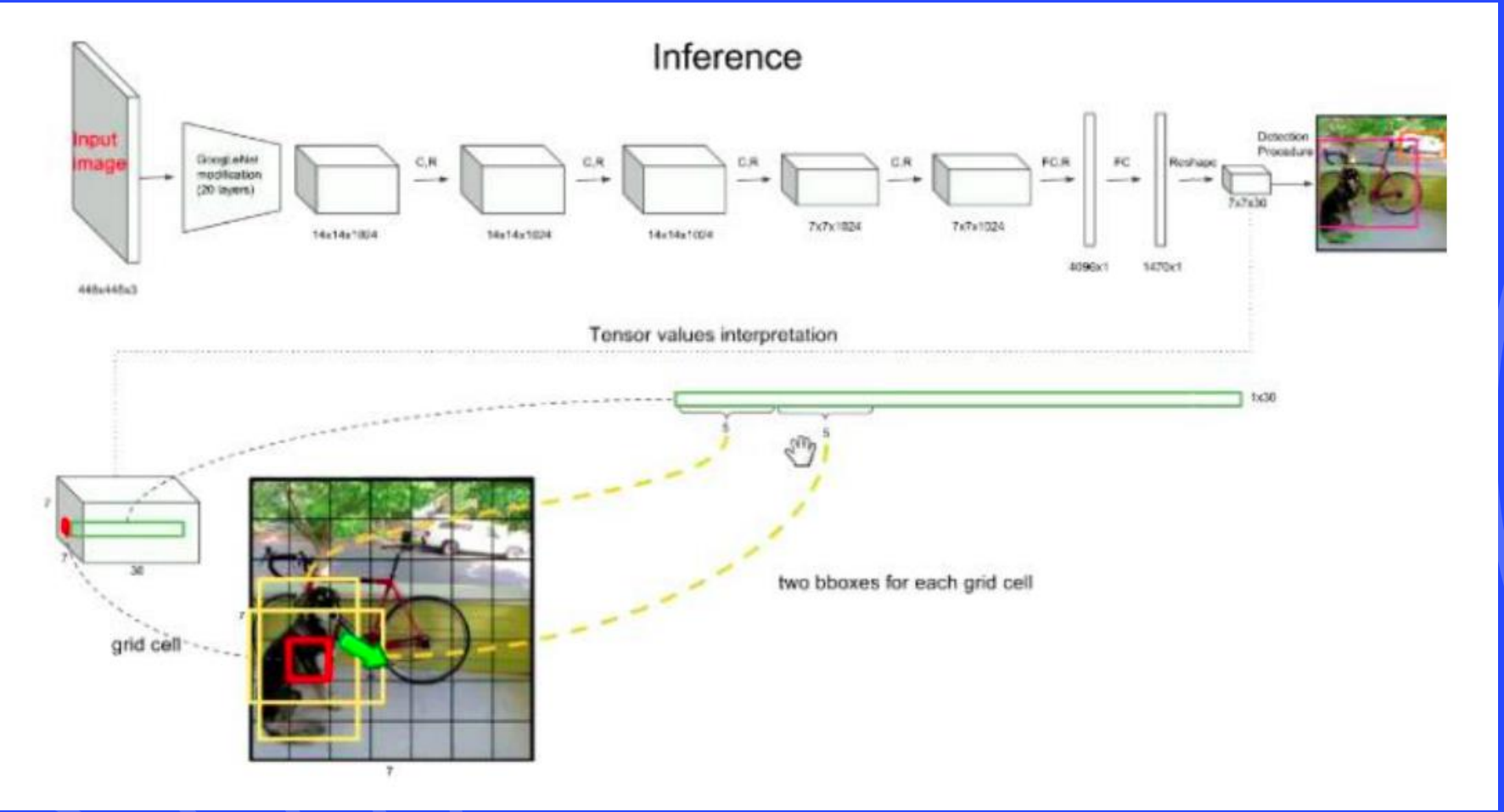
P(class|Object) * P(Object)
=P(class)



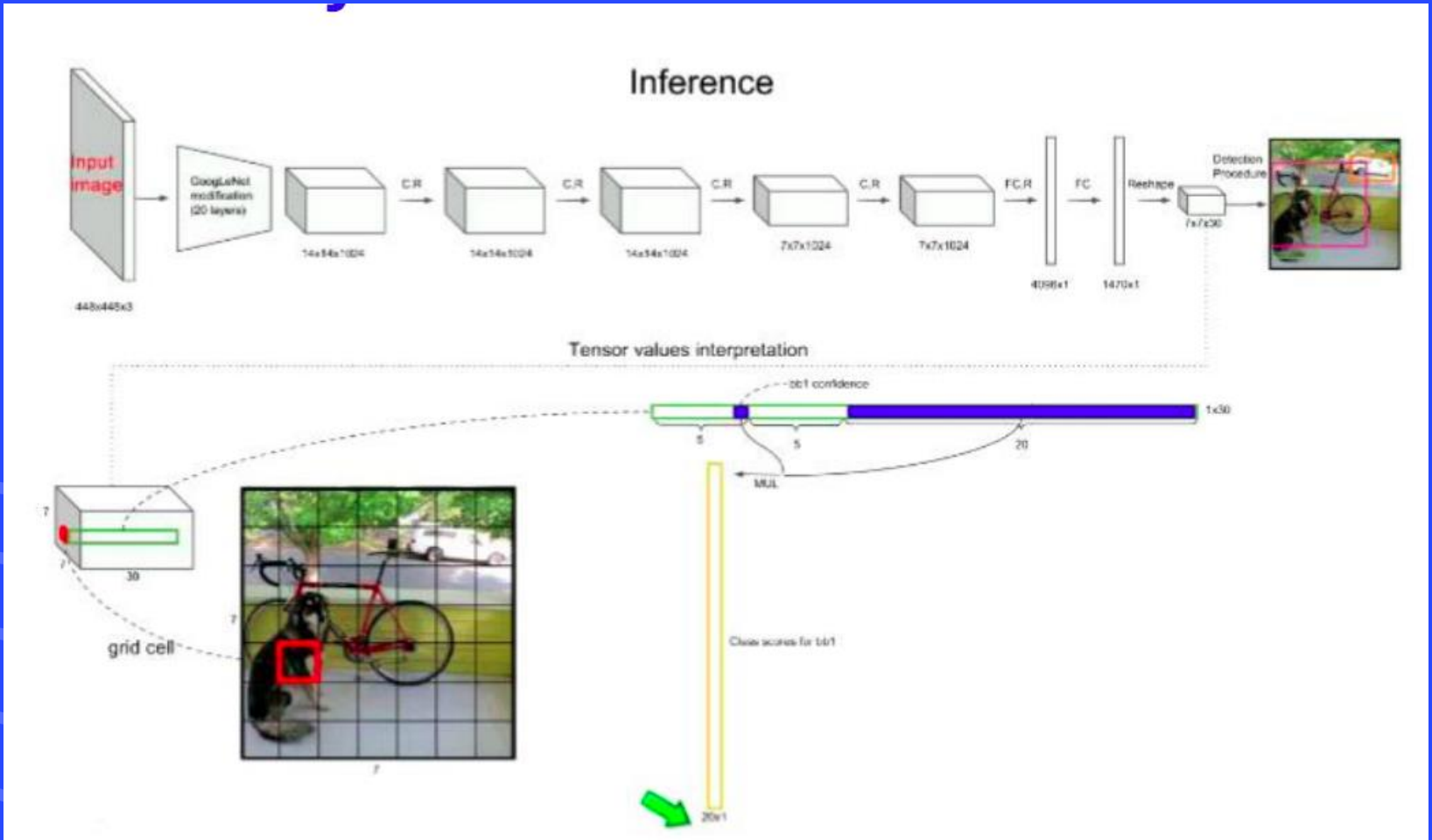
YOLO – You Only Look Once-test



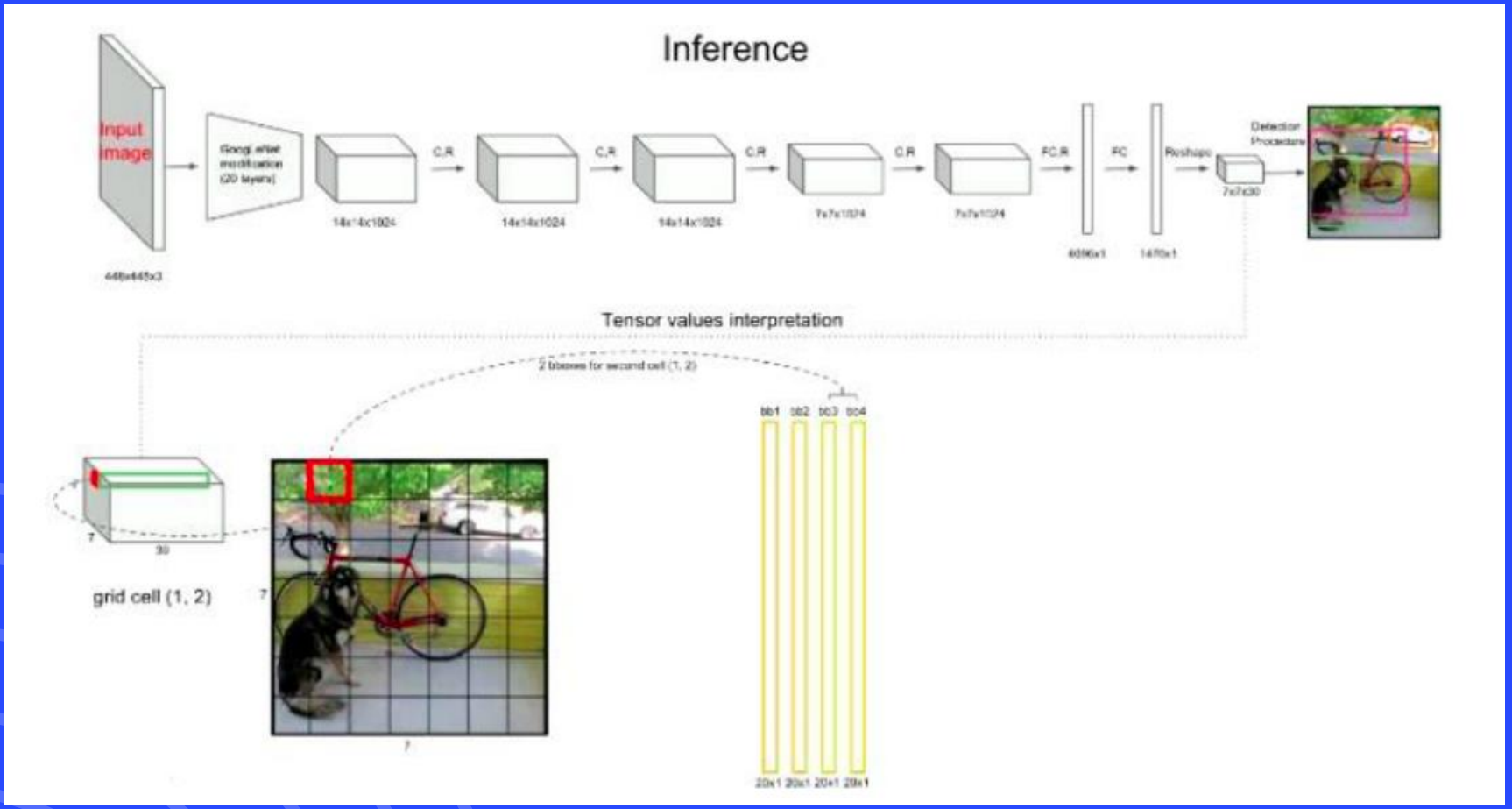
YOLO – You Only Look Once-test



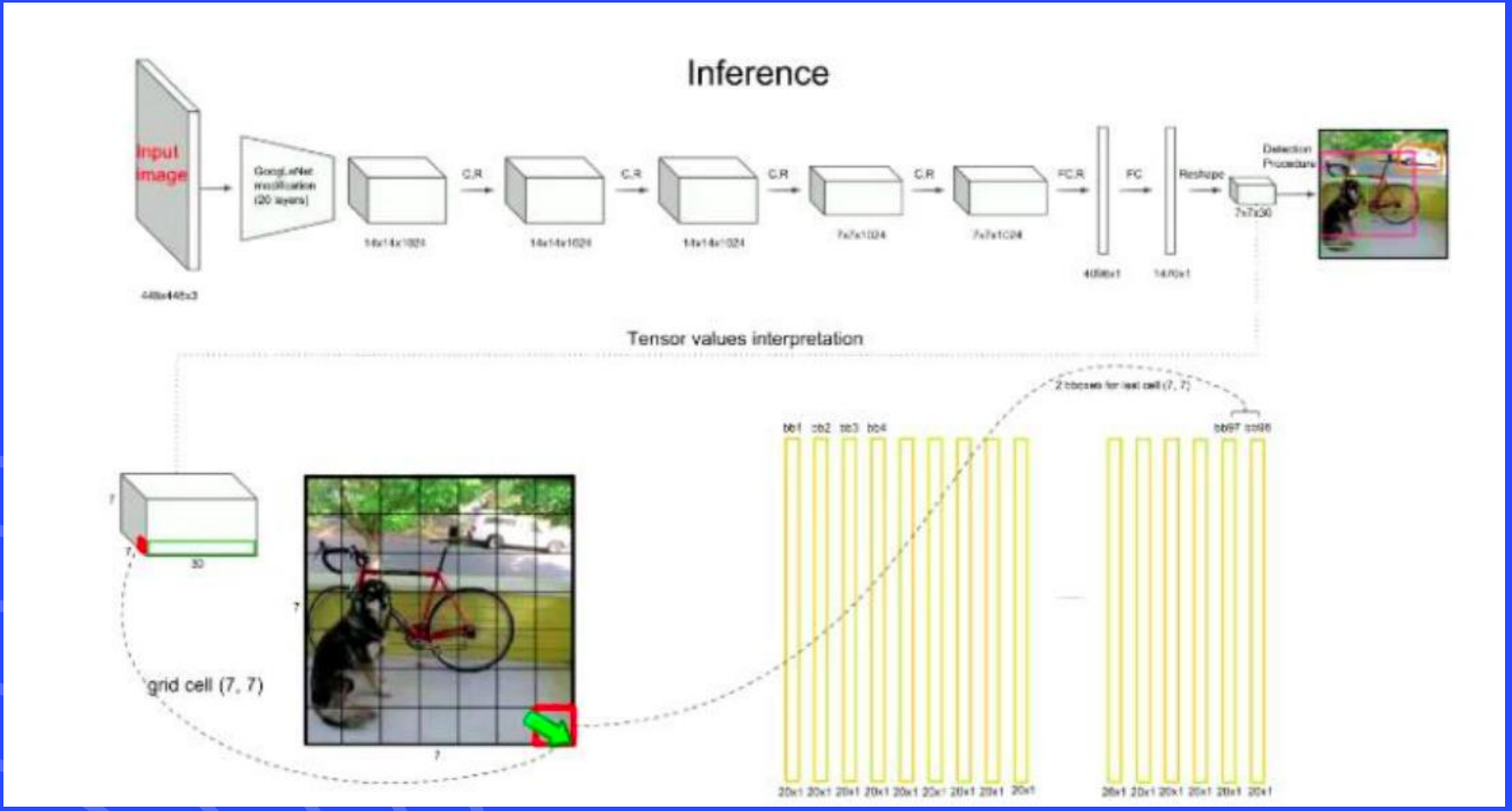
YOLO – You Only Look Once-test



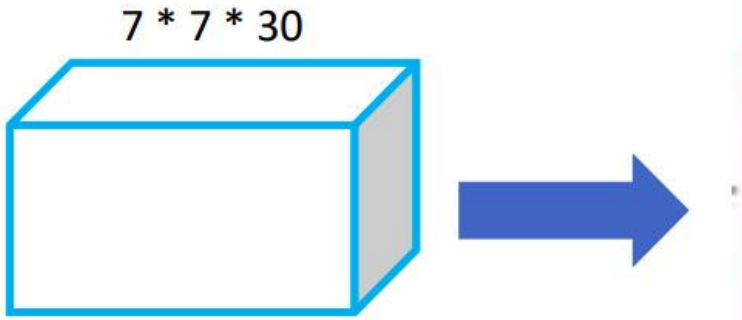
YOLO – You Only Look Once-test

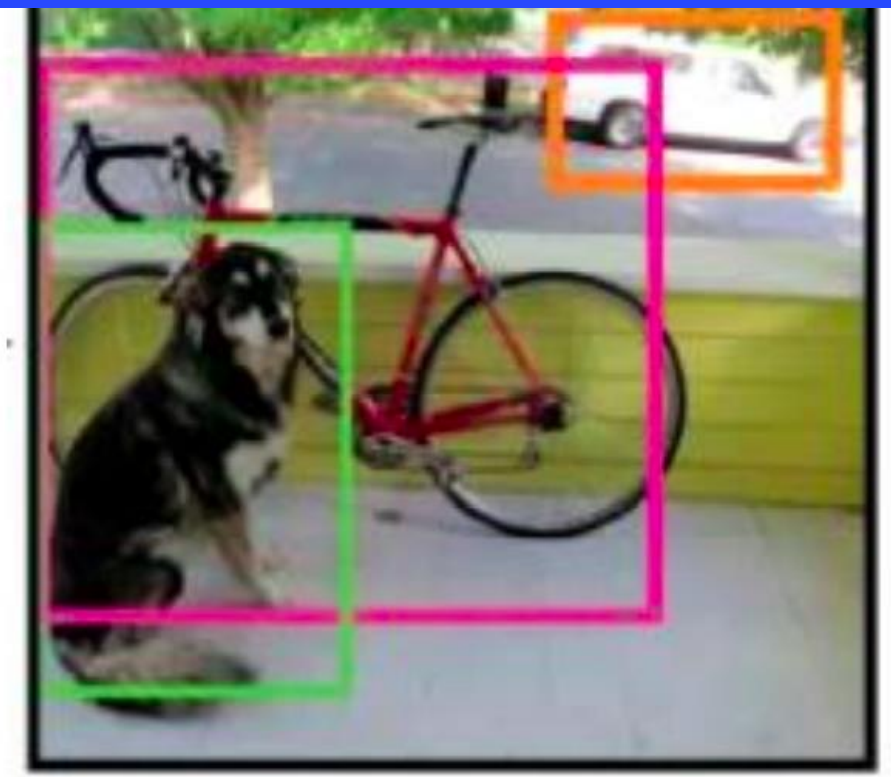


YOLO – You Only Look Once-test

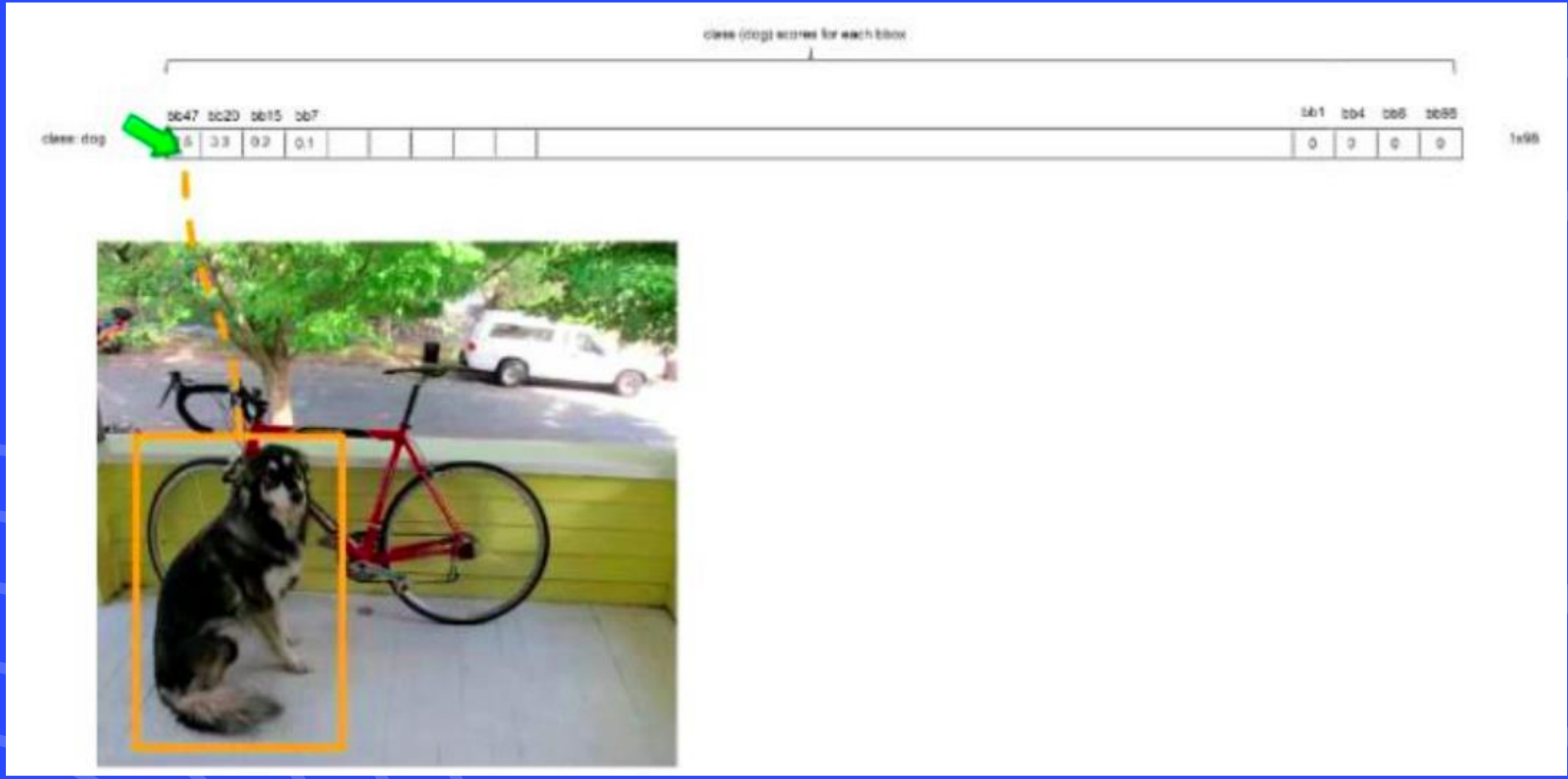


YOLO – You Only Look Once-test

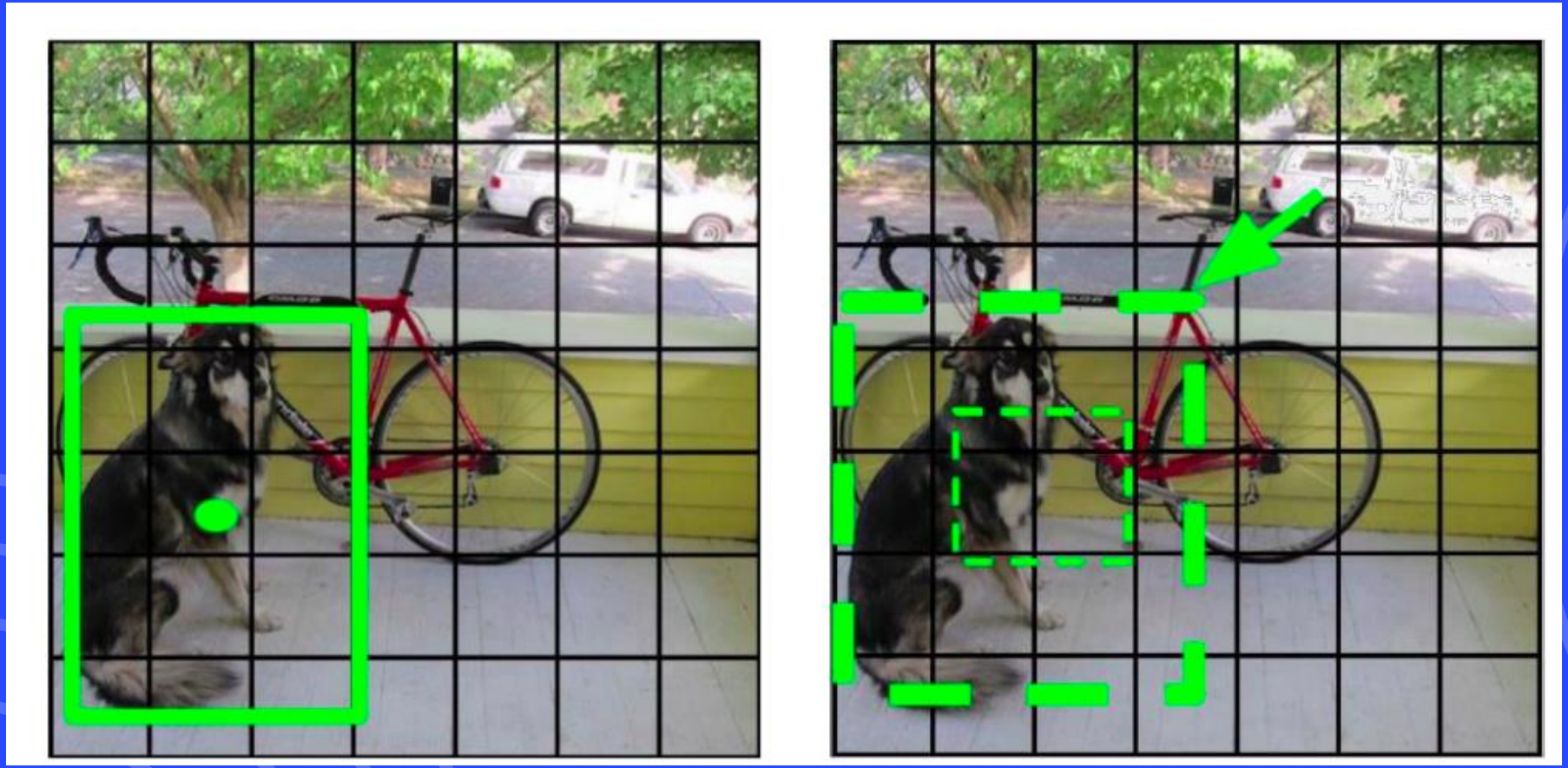




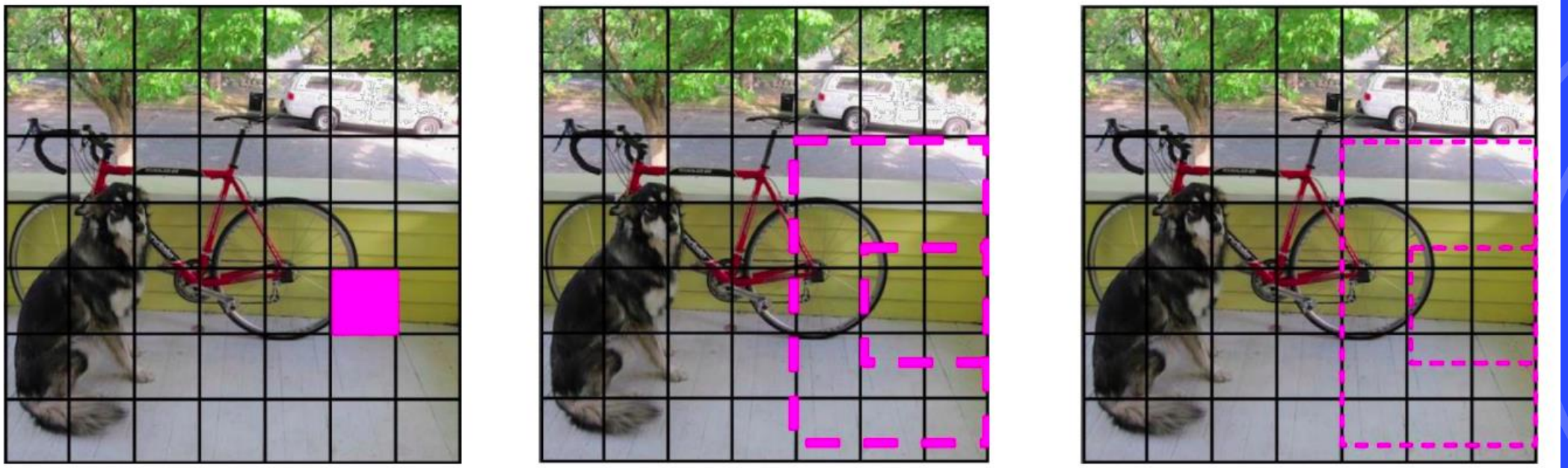
YOLO – You Only Look Once-test-NMS



YOLO – You Only Look Once-train

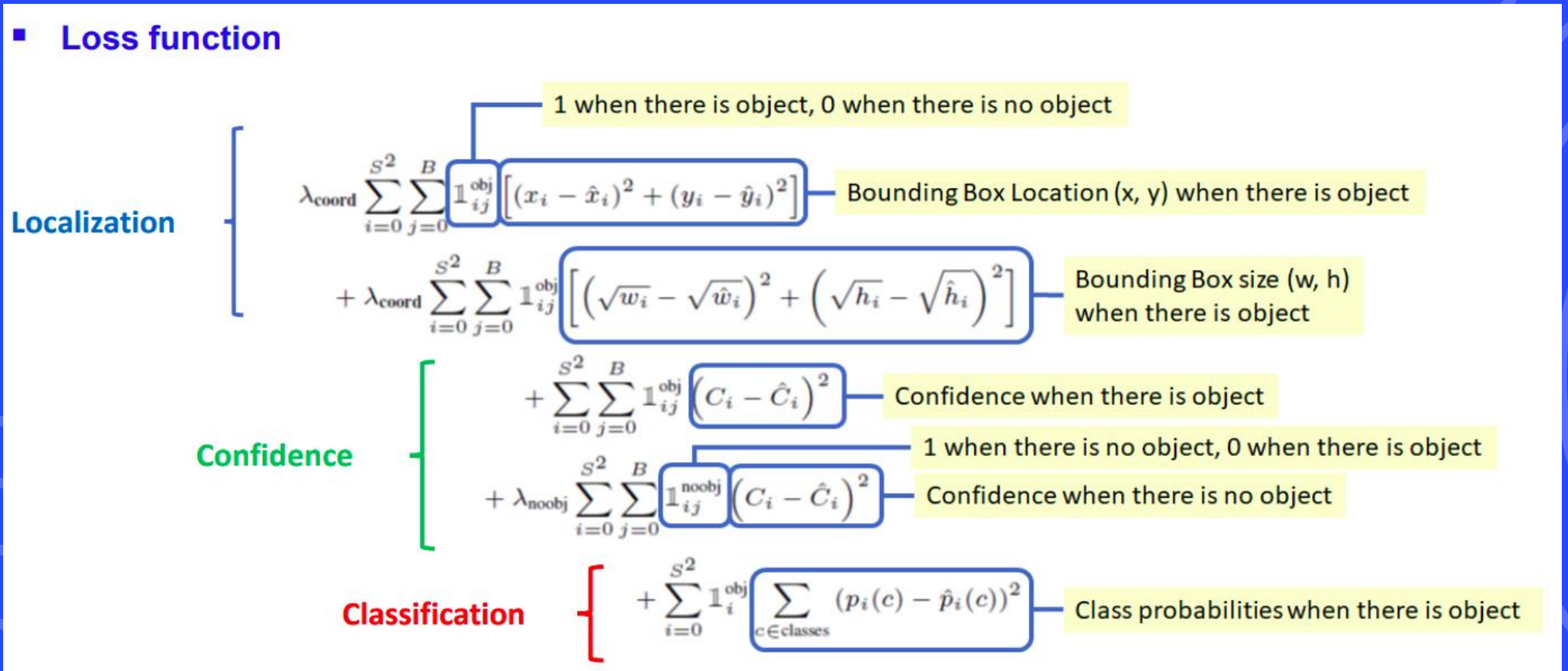


YOLO – You Only Look Once-train



YOLO – You Only Look Once-train

- **Loss function**



YOLO – You Only Look Once-train

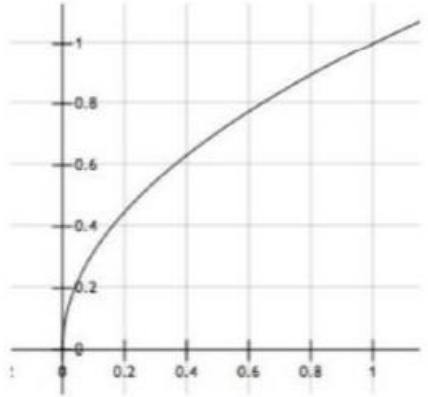
$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

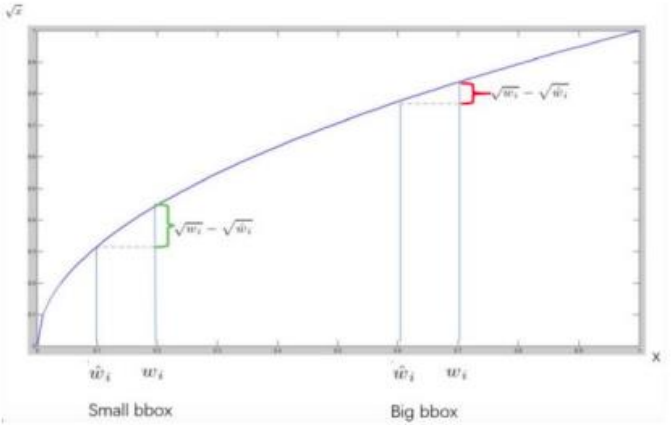
$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)$$





YOLO – You Only Look Once

- Limitation of YOLO
  - Group of small objects
  - Unusual aspect rations
  - Coarse feature
  - Localization of bounding box

YOLO – You Only Look Once

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	63.4	45 FPS	22 ms/img

YOLO – You Only Look Once

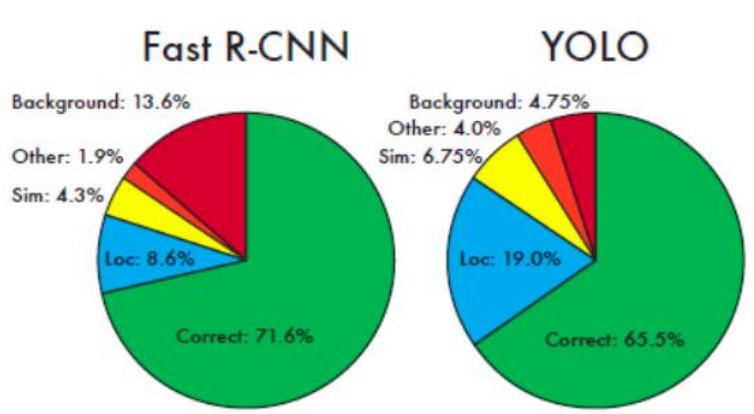


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

YOLO – You Only Look Once

- Strengths
  - Fast
  - 45 fps, smaller version 155 fps
  - End2end training o Background error is low
- Weaknesses
  - Performance is lower than state-of-art
  - Makes more localization error