CS 437 / CS 5317 / EE412
Deep Learning

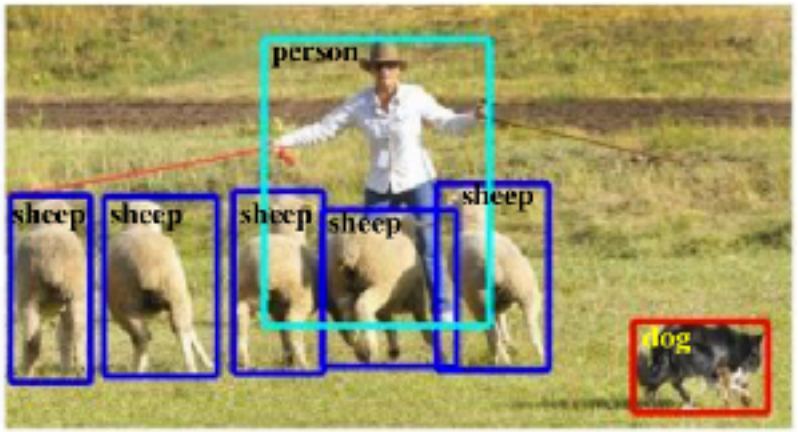
Murtaza Taj
murtaza.taj@lums.edu.pk

Lecture 29: Supp: Object Detection 2

Wed 04th Apr 2022

Recap: From Classification to Object Detection

(a) Object Classification

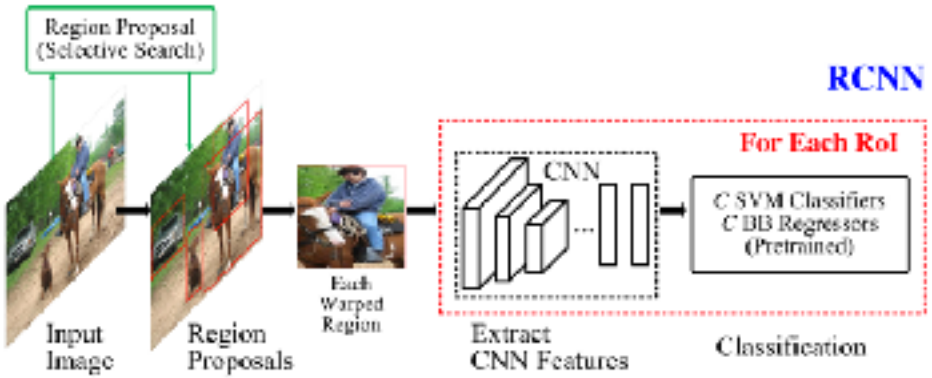


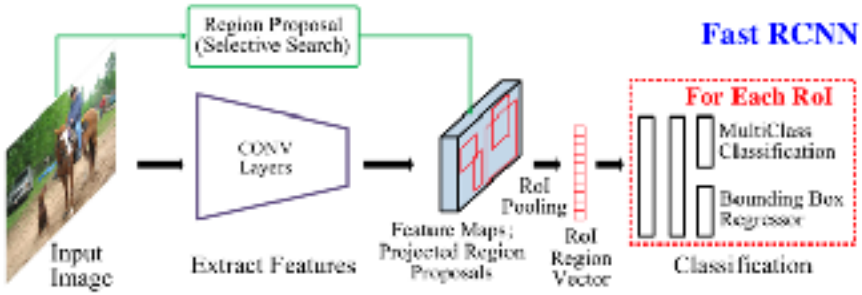
(b) Generic Object Detection
(Bounding Box)

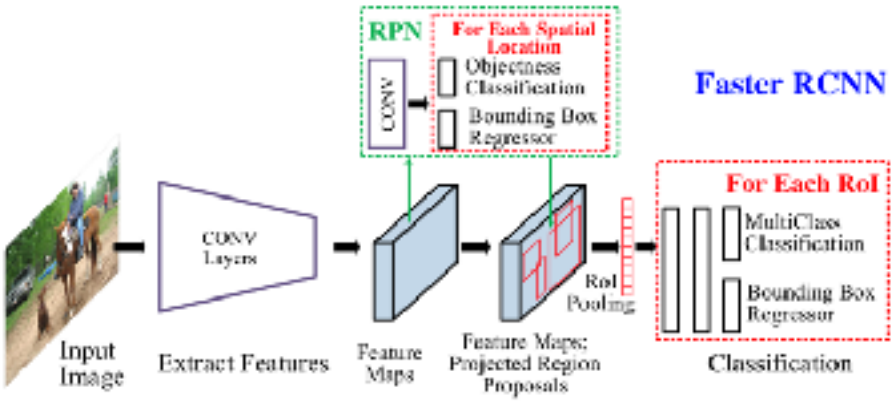
Class

Class
(x,y,w,h)

Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018







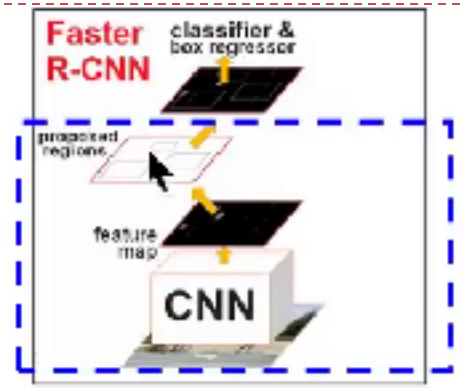
Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018

This Lecture

- Mask RCNN
- Yolo
- SSD
- 3D Object Detection

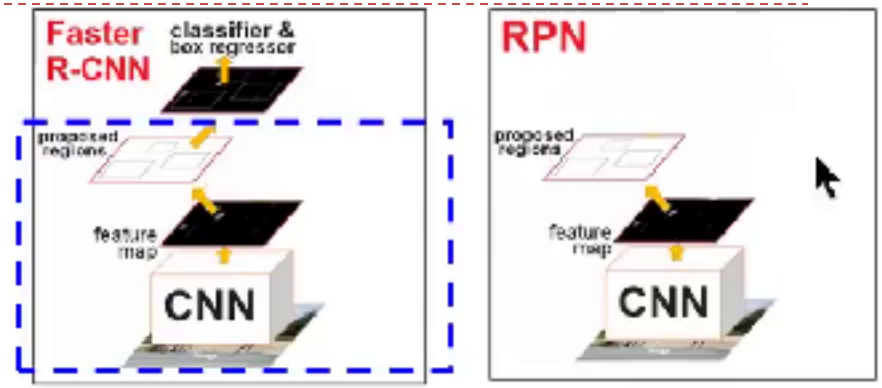
Region Proposal Network

RPN as reg. proposer

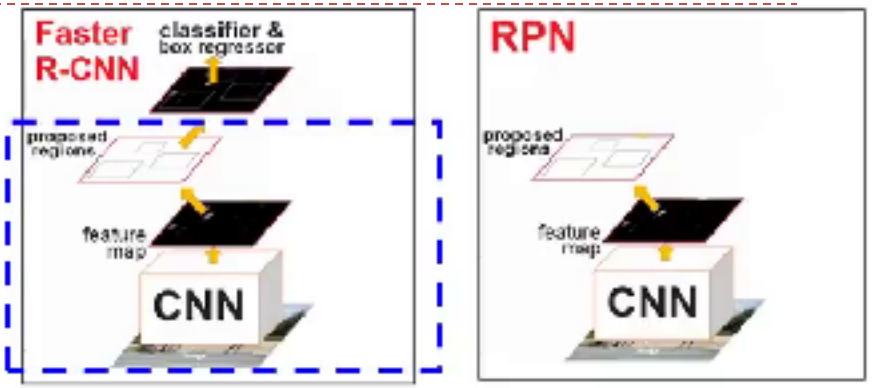


Region Proposal Network

RPN as reg. proposer

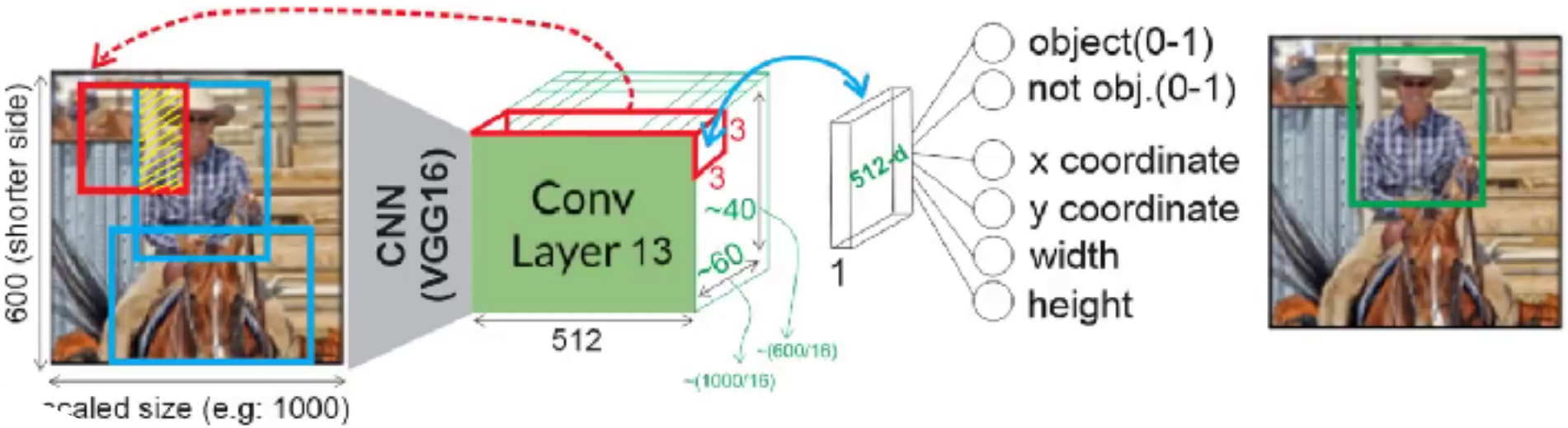


Region Proposal Network

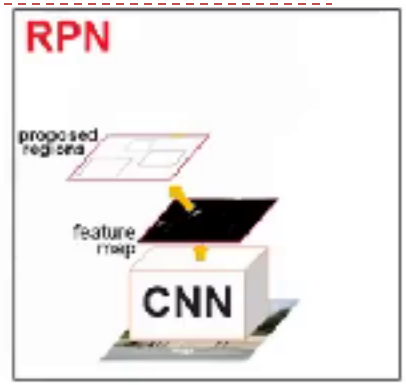


RPN as reg. proposer

$$IoU = \frac{A \cap Gt}{A \cup Gt} \begin{cases} > 0.7 = \text{object} \\ < 0.3 = \text{not object} \end{cases}$$

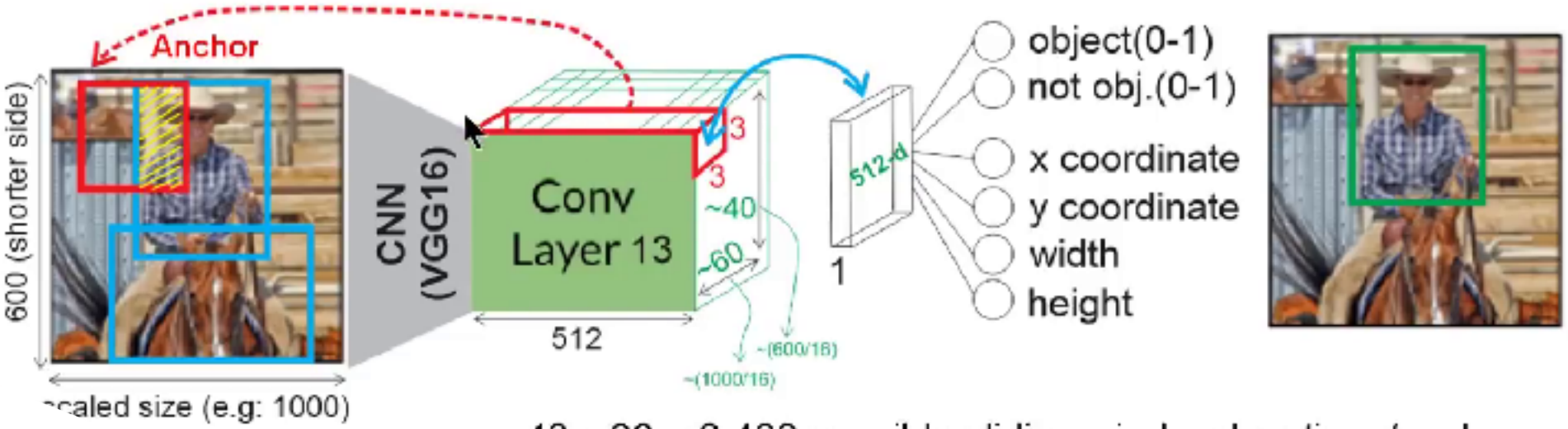


Region Proposal Network



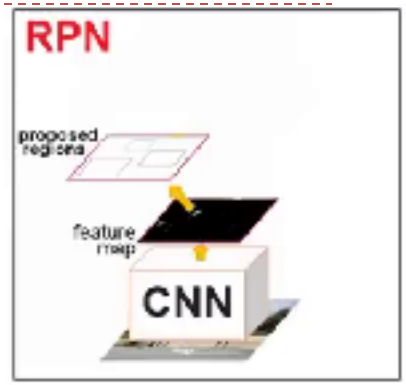
RPN as reg. proposer

$$IoU = \frac{A \cap Gt}{A \cup Gt} \begin{cases} > 0.7 = \text{object} \\ < 0.3 = \text{not object} \end{cases}$$



40 x 60 = 2,400 possible sliding window locations/anchors

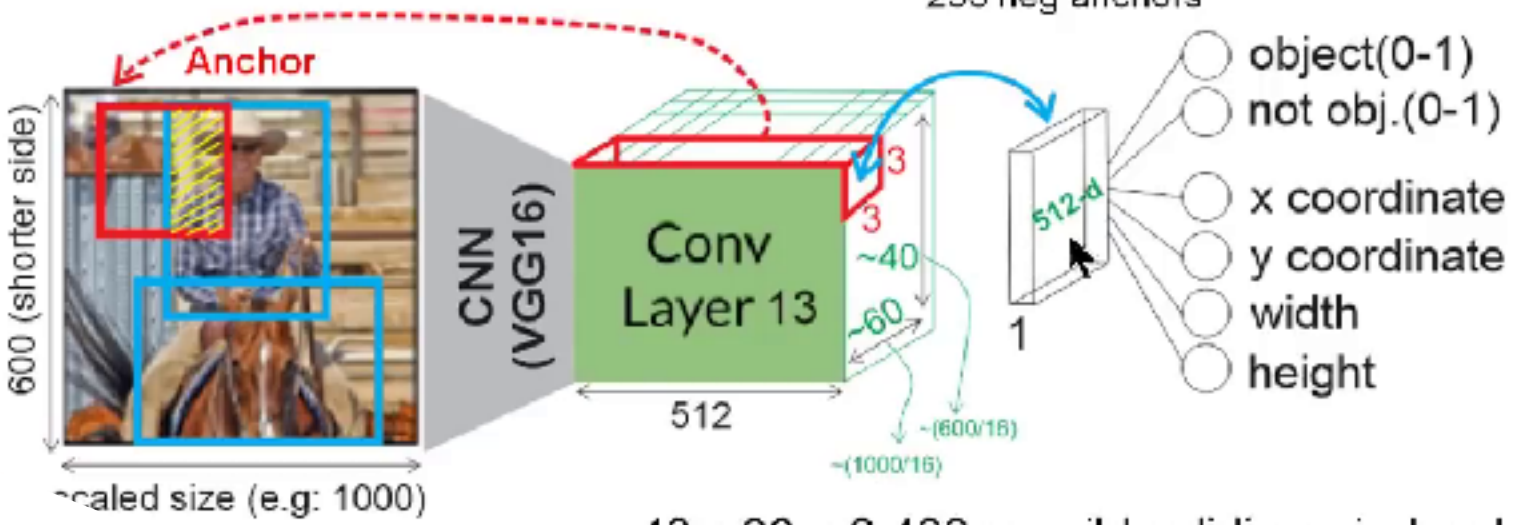
Region Proposal Network

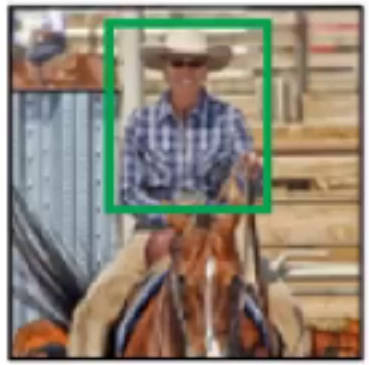


RPN as reg. proposer

$$IoU = \frac{A \cap Gt}{A \cup Gt} \begin{cases} > 0.7 = \text{object} \\ < 0.3 = \text{not object} \end{cases}$$

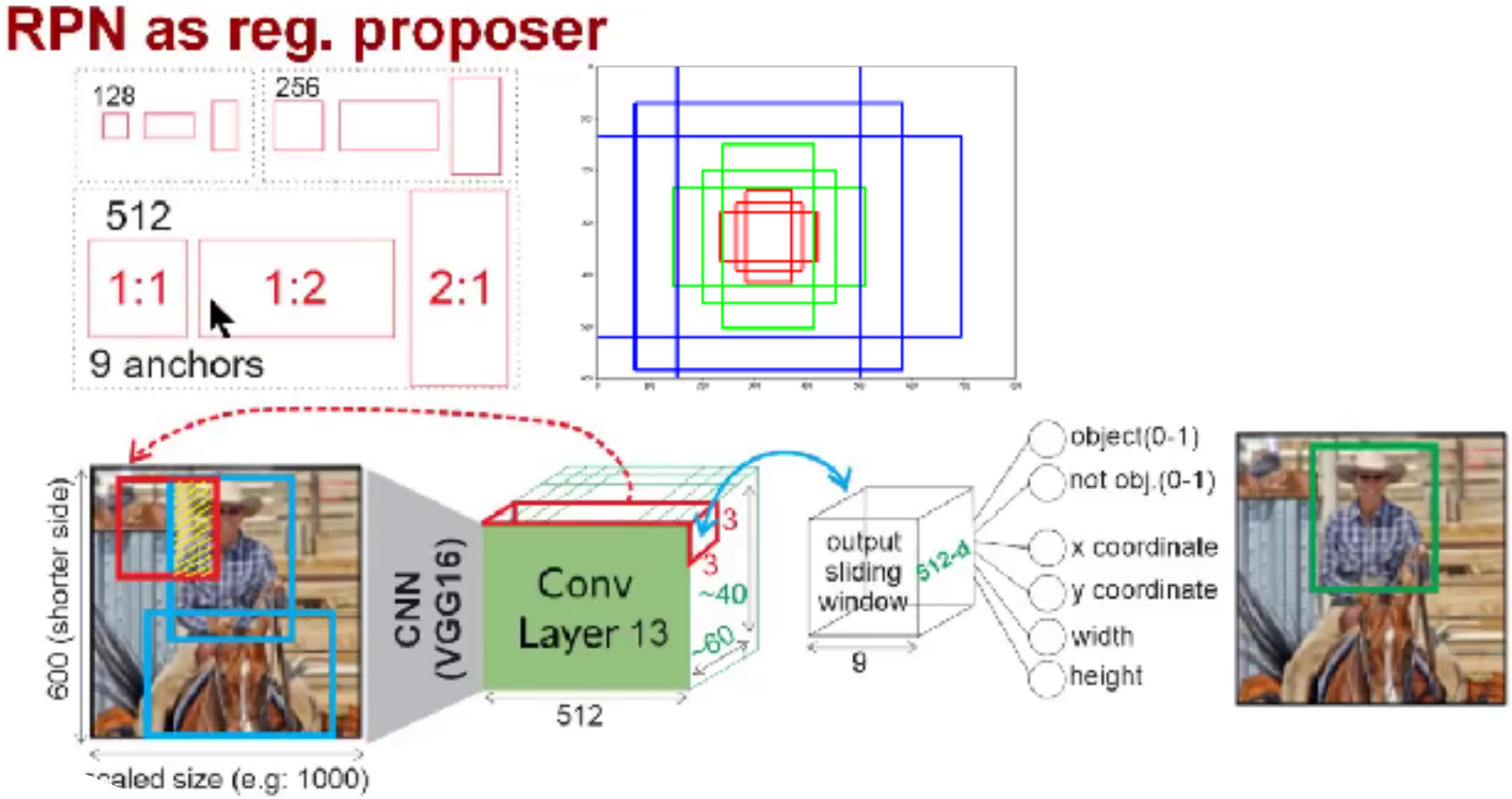
256 pos anchors
256 neg anchors



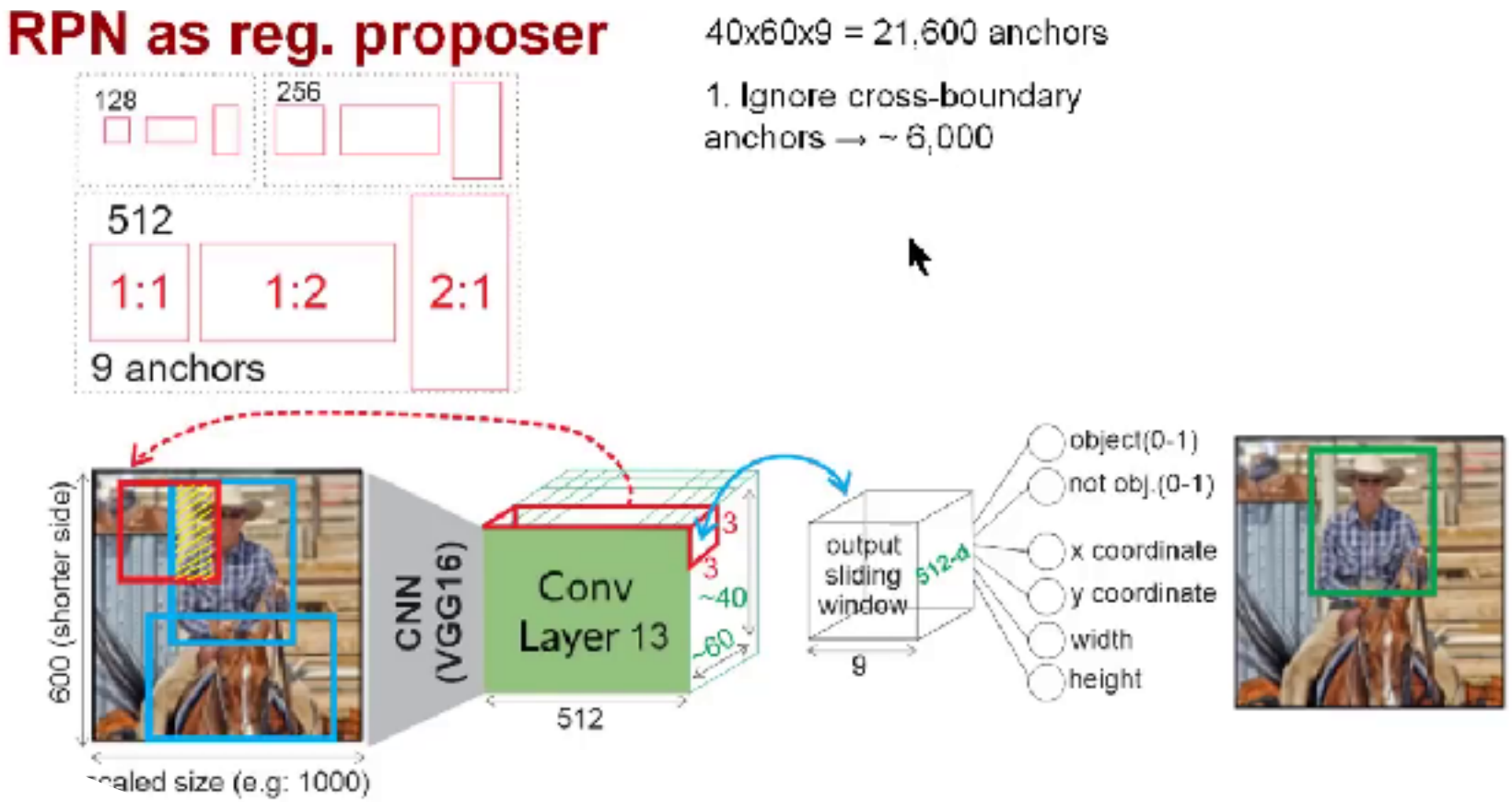


40 x 60 = 2,400 possible sliding window locations/anchors

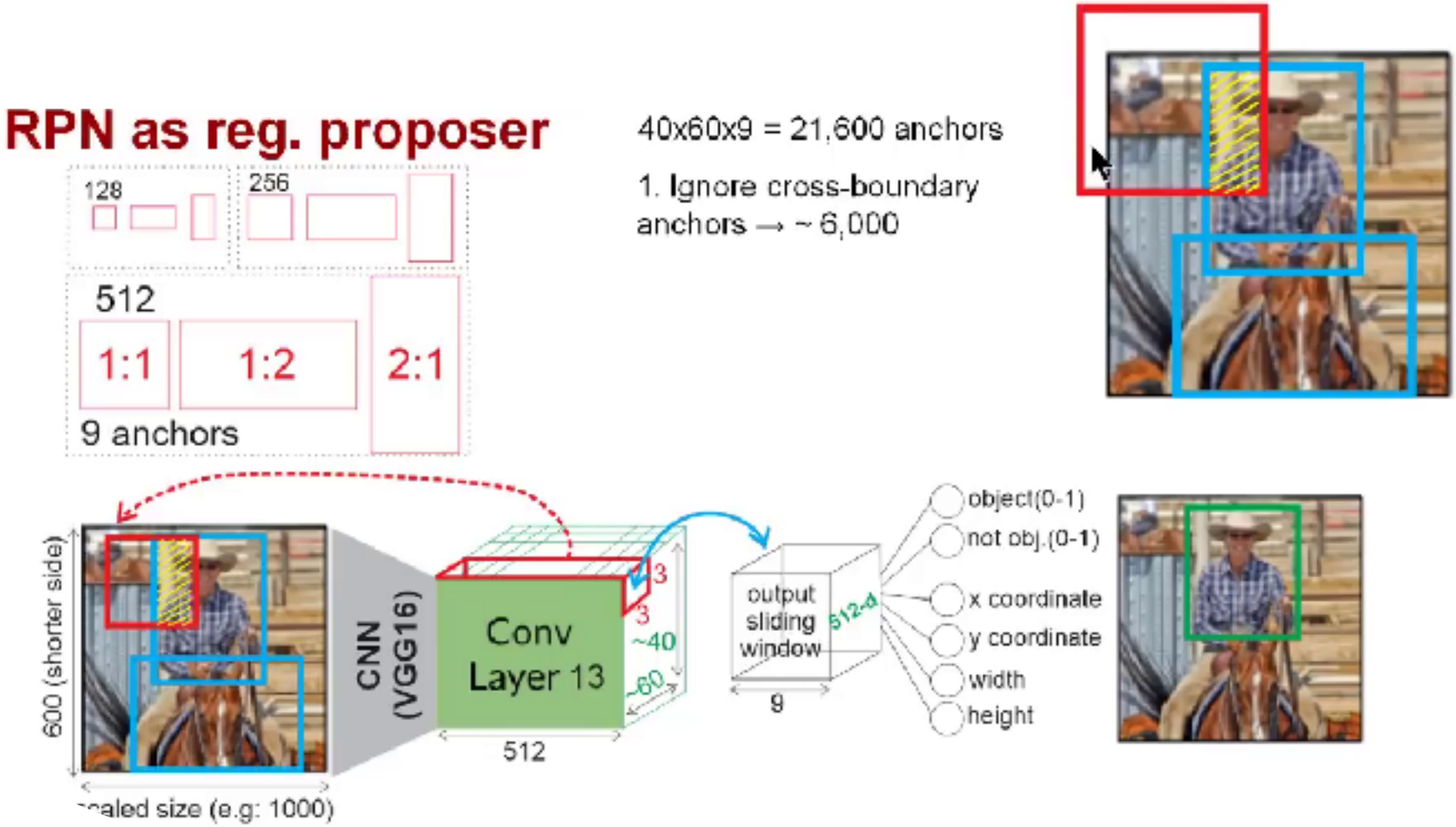
Region Proposal Network



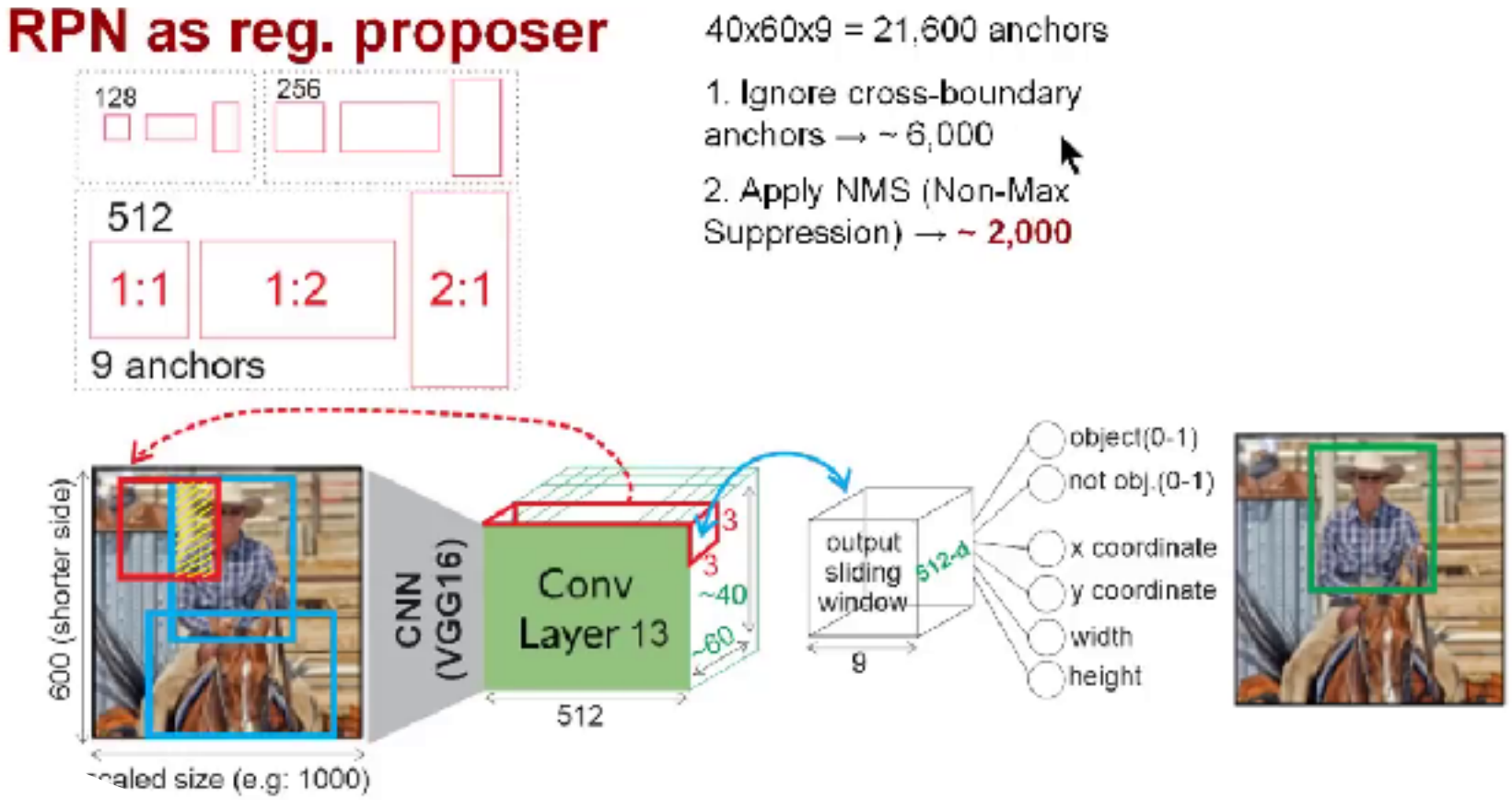
Region Proposal Network



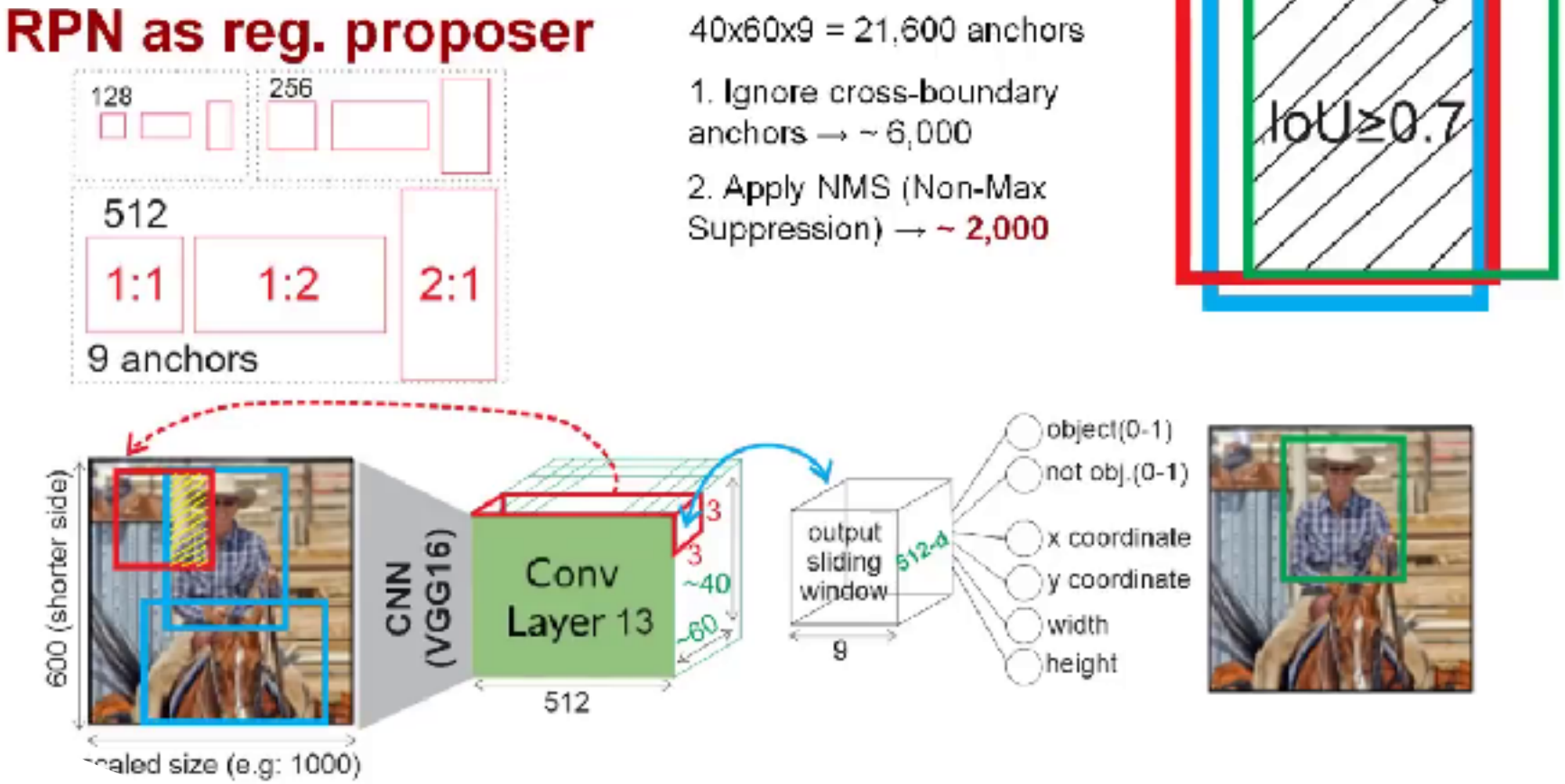
Region Proposal Network



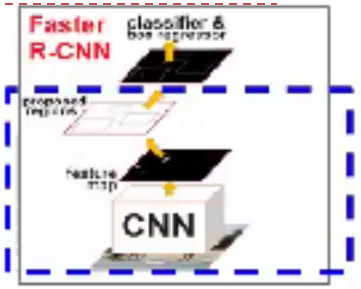
Region Proposal Network



Region Proposal Network



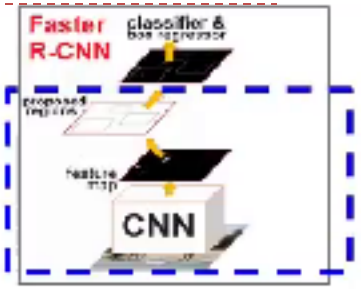
Region Proposal Network



RPN loss function

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

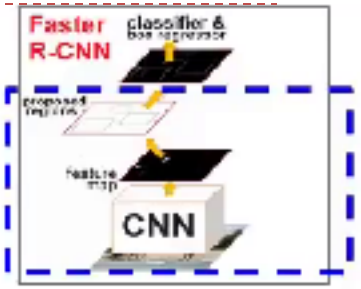
Region Proposal Network



RPN loss function

$$L(\ p_i\ ,t_i\ ) = \underbrace{\frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)}_{\text{object/not object classifier}} + \underbrace{\lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)}_{\text{box regressor}}$$

Region Proposal Network



RPN loss function

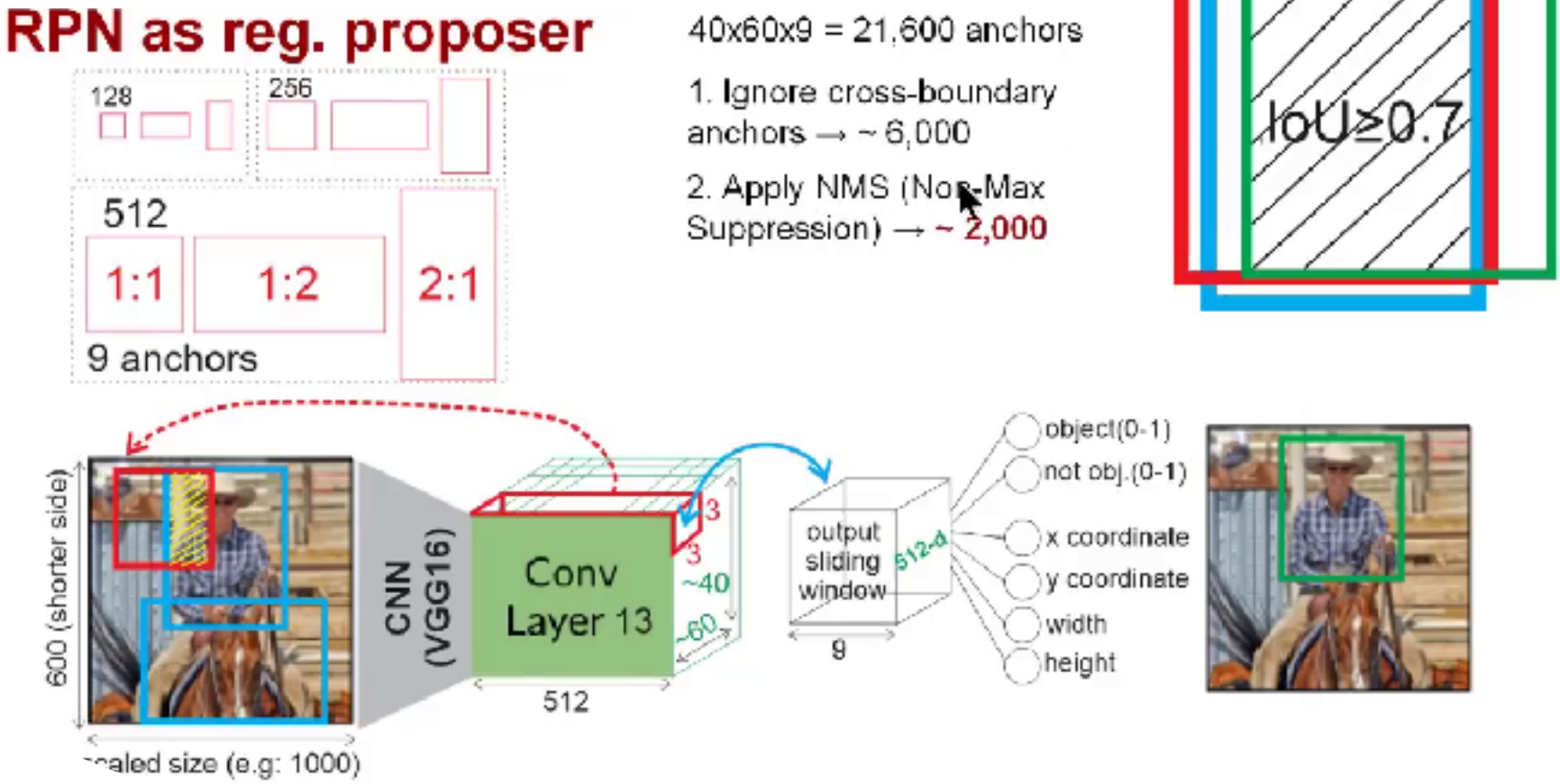
$$L(\ p_i\ ) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)$$

p_i : predicted probability

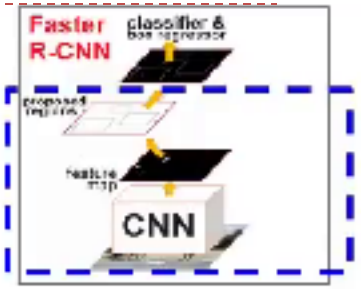
p_i^* { 1 for pos anchor
0 for neg anchor

N_{cls} : numbers of anchors in minibatch (512)

Region Proposal Network



Region Proposal Network



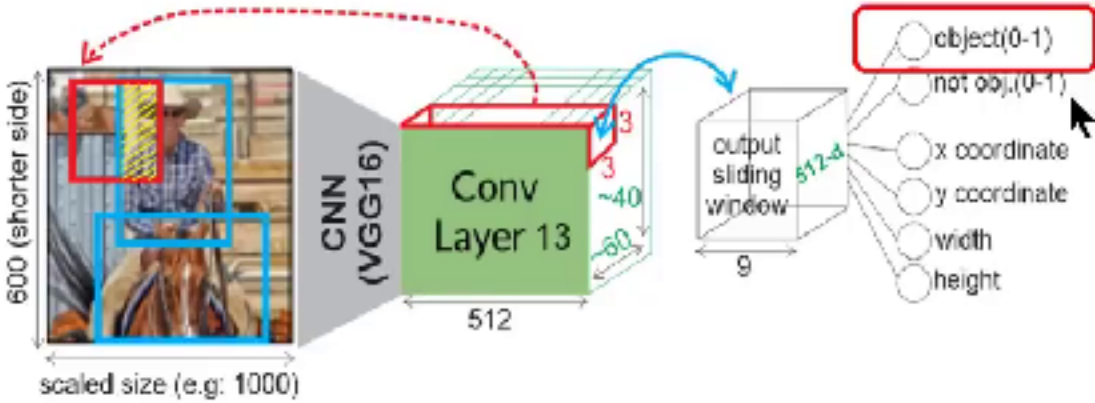
RPN loss function

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

p_i : predicted probability

p_i^* { 1 for pos anchor / 0 for neg anchor

N_{cls} : numbers of anchors in minibatch (512)



Region Proposal Network

RPN loss function

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

λ = constant value

N_{reg} = number of total anchors (~2000)

p_i^* { 1 for pos anchor
0 for neg anchor

Region Proposal Network

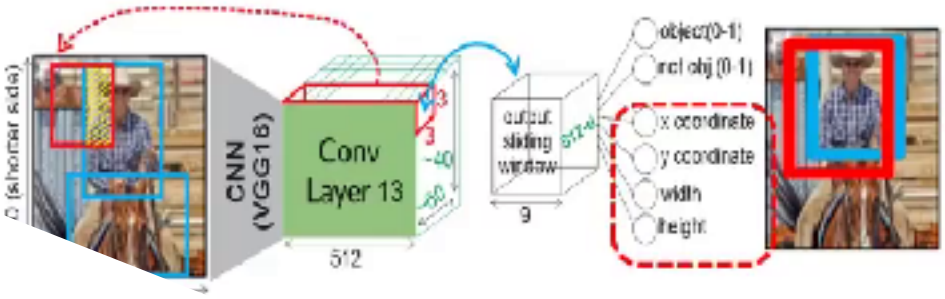
RPN loss function

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

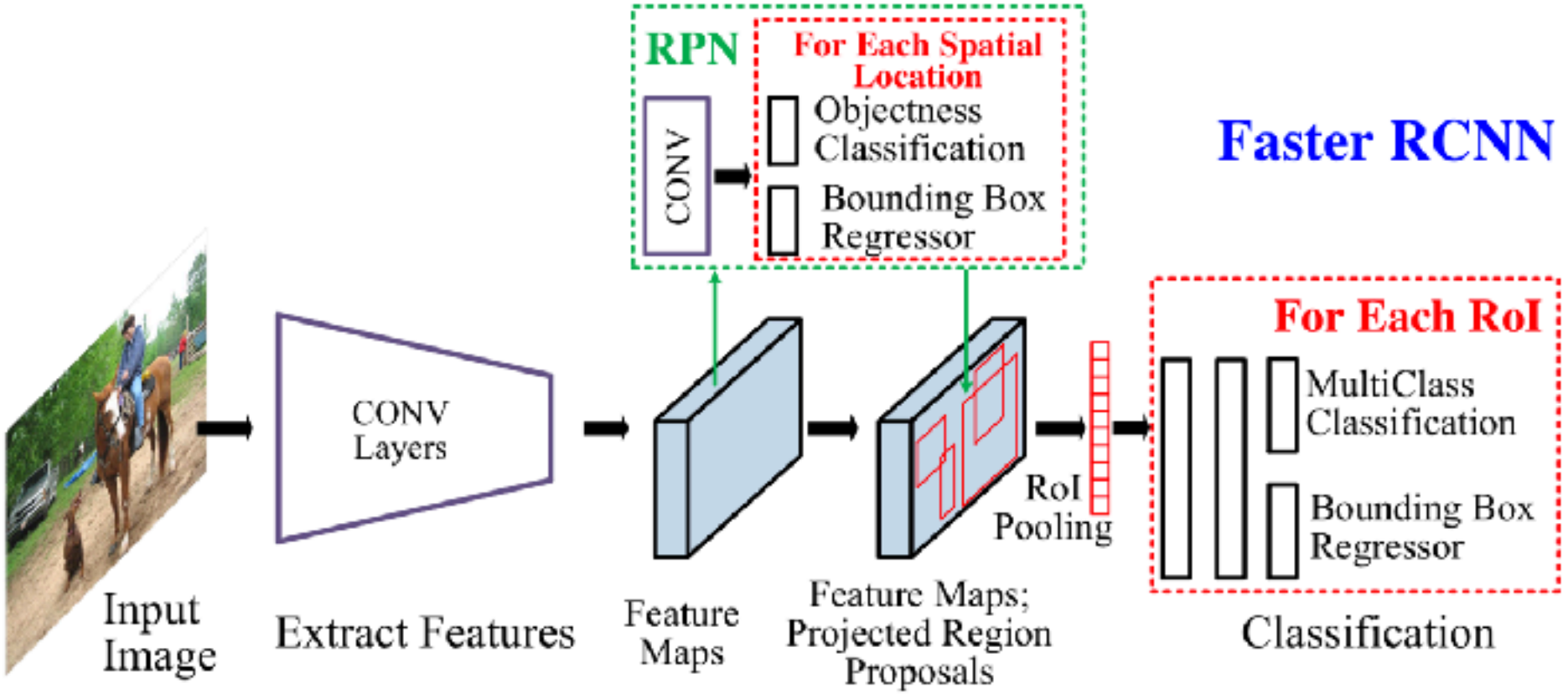
L_{reg} = $\text{smooth}_{L_1}(\mathbf{t}_i - \mathbf{t}_i^*)$ $\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$

$\mathbf{t}_i$ = predicted box; $\mathbf{t}_i^*$ = ground truth box

$$\begin{aligned} t_x &= (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \\ t_w &= \log(w/w_a), \quad t_h = \log(h/h_a), \\ t_x^* &= (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a, \\ t_w^* &= \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a), \end{aligned}$$

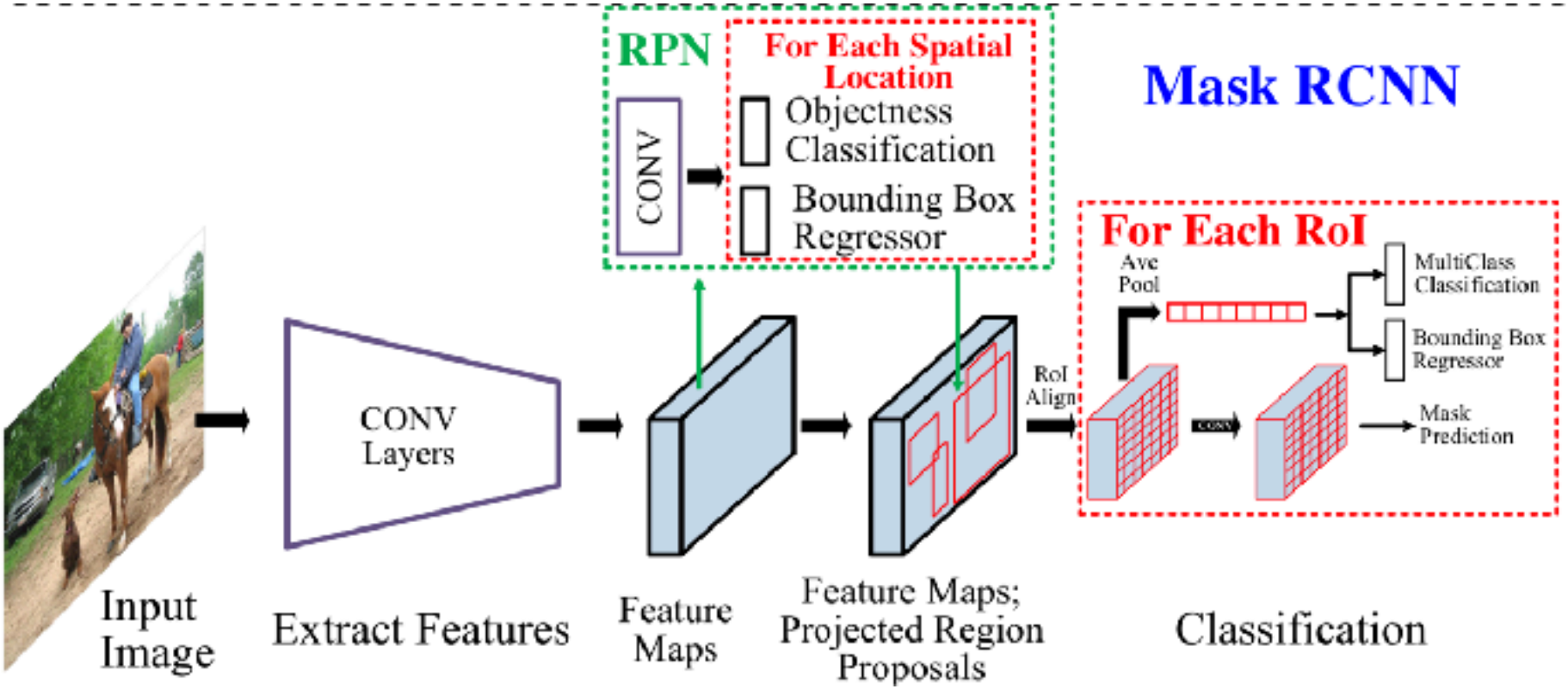


Faster RCNN



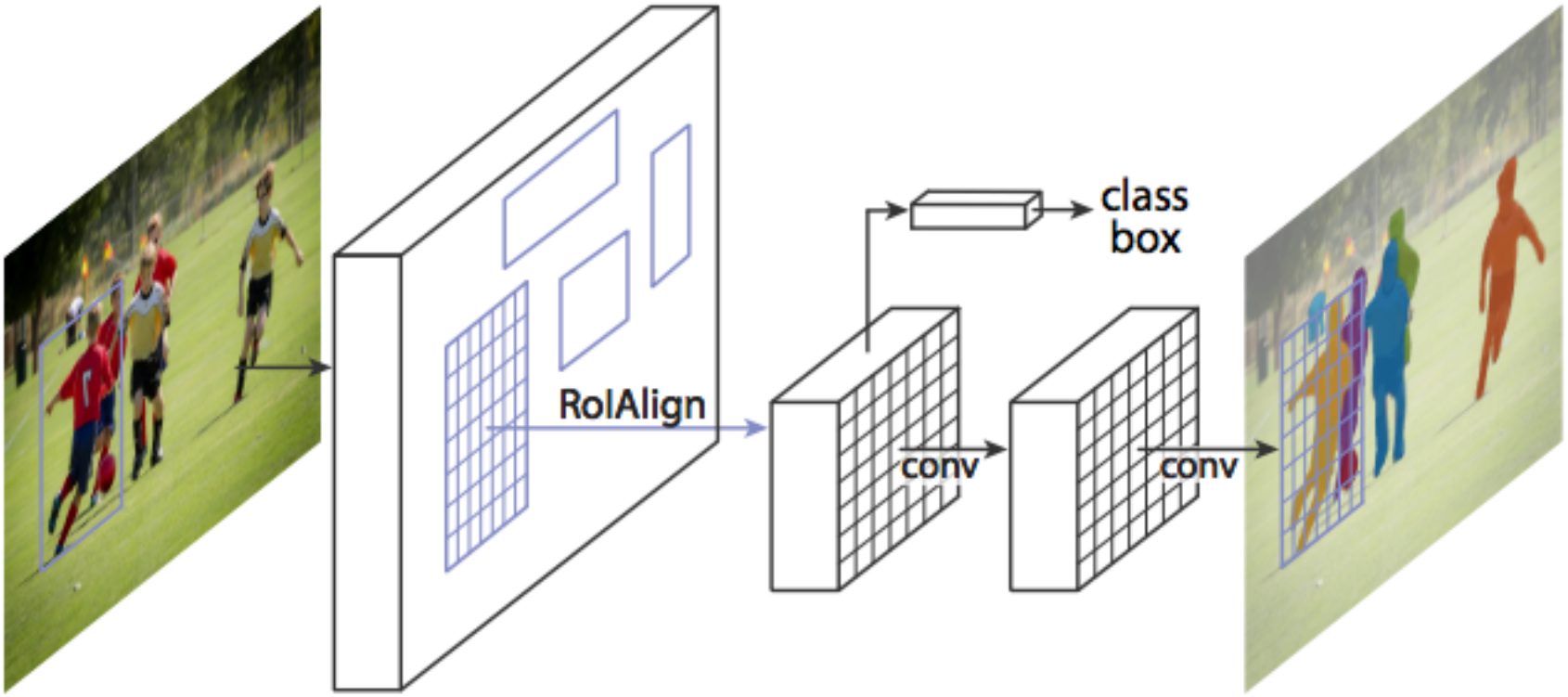
Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018

Mask RCNN



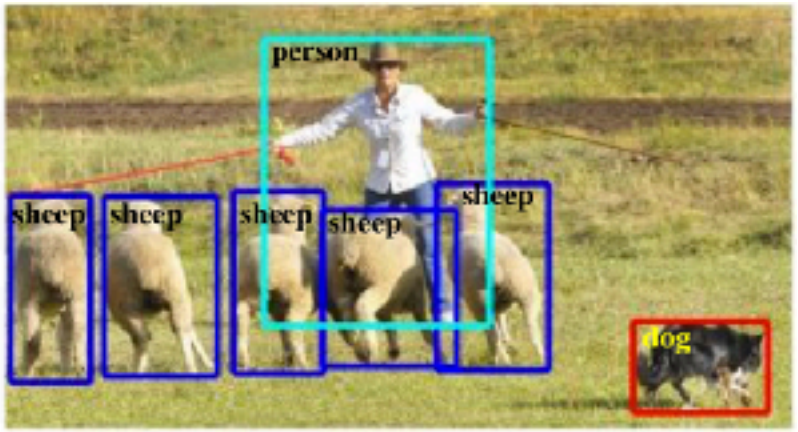
Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018

Mask RCNN: Mask Prediction

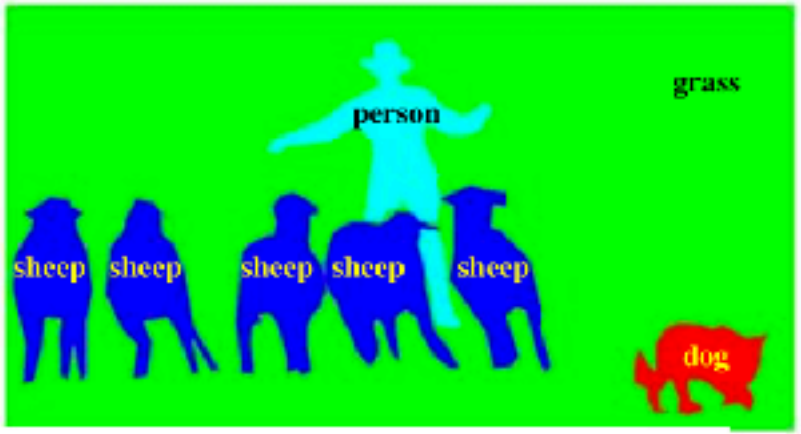


Mask RCNN

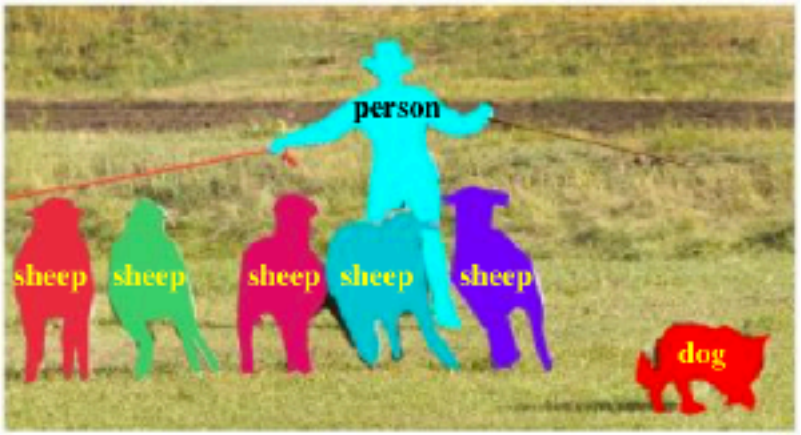
(a) Object Classification



(b) Generic Object Detection (Bounding Box)

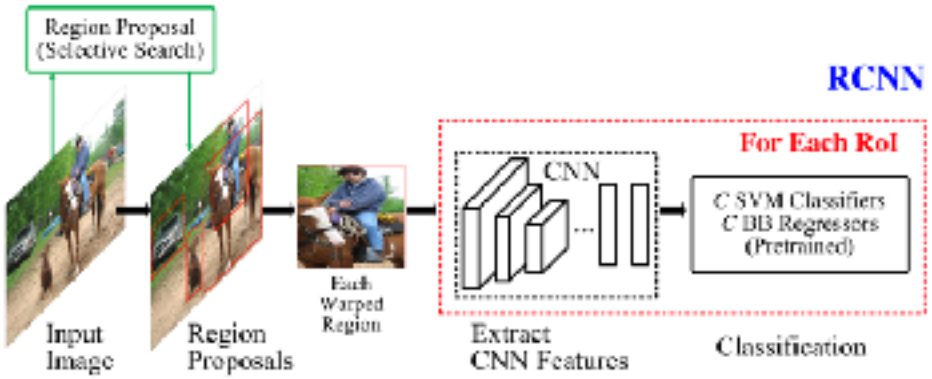


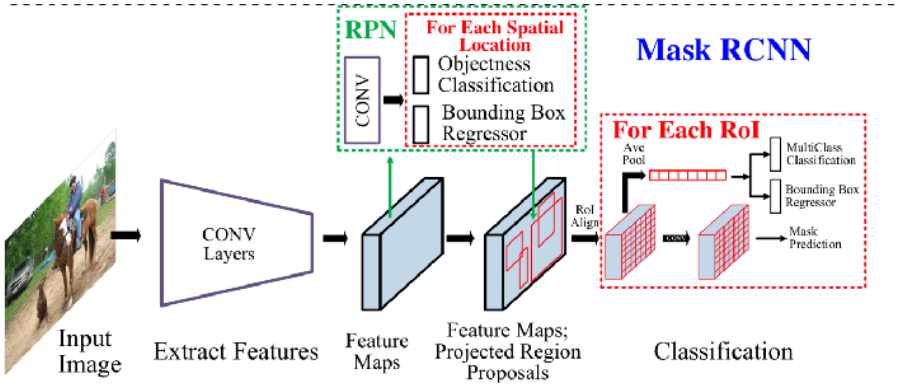
(c) Semantic Segmentation

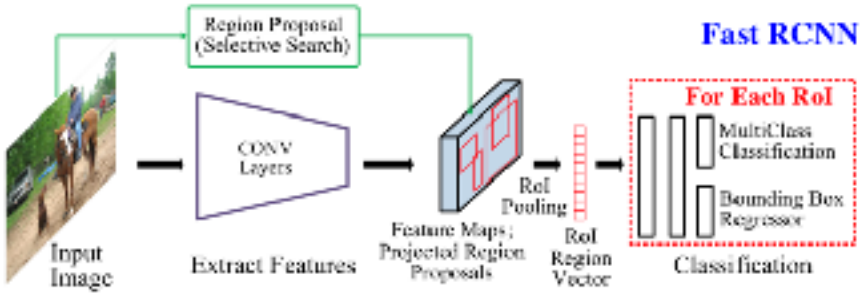


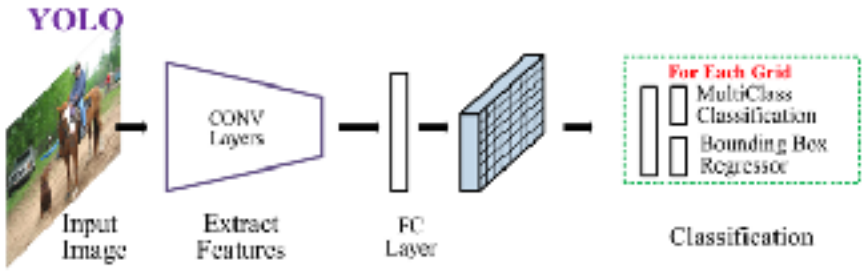
(d) Object Instance Segmetation

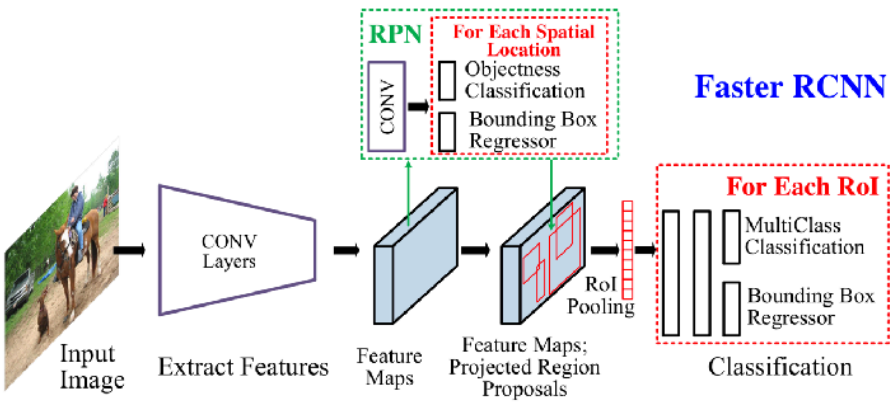
Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018

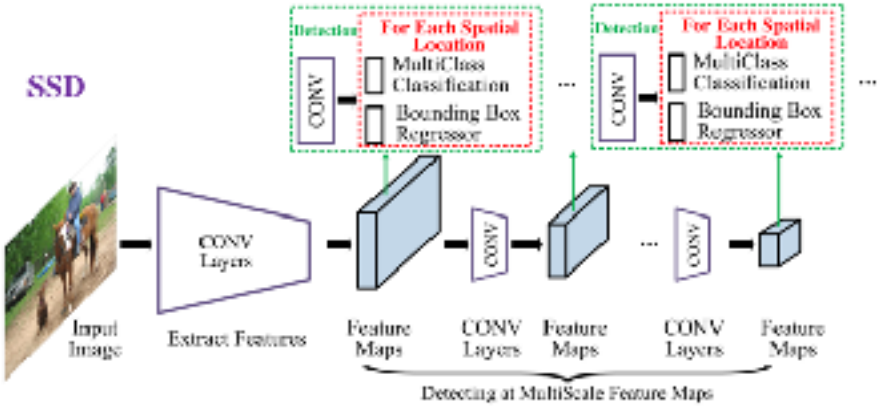






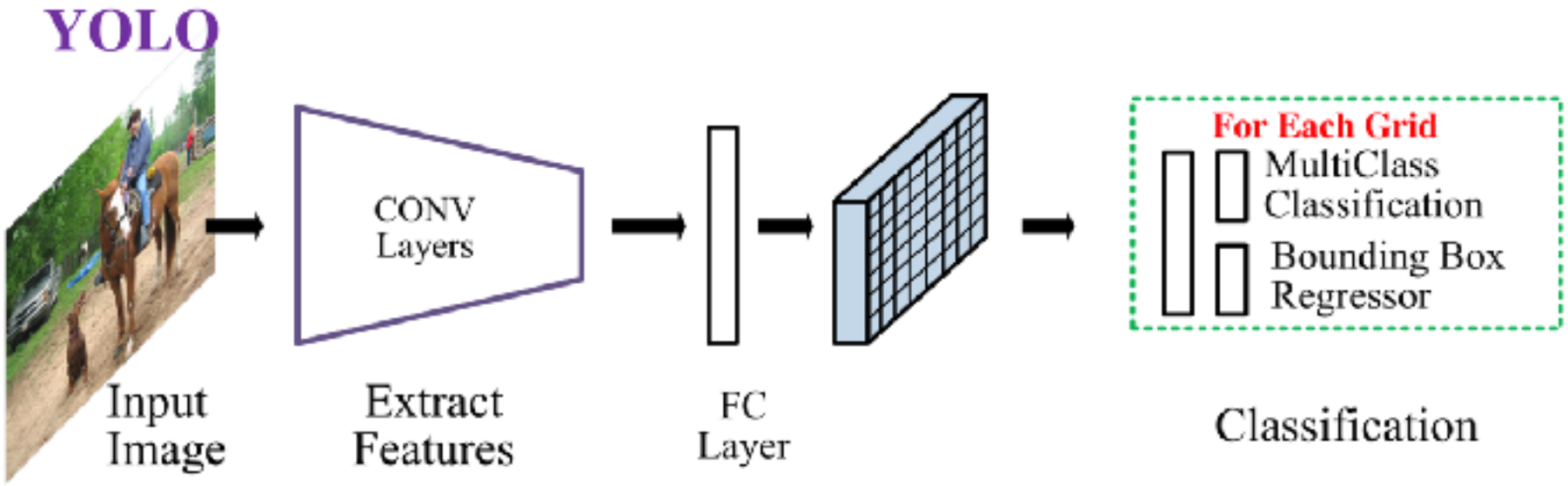






Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018

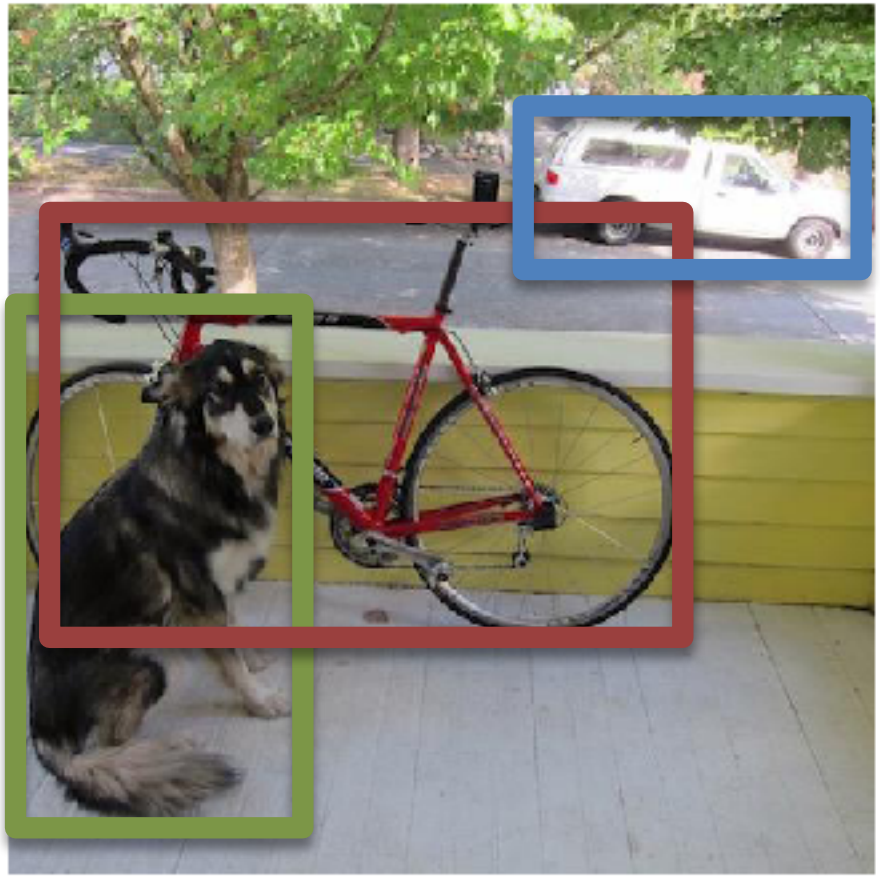
YOLO: You Only Look Once



Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018

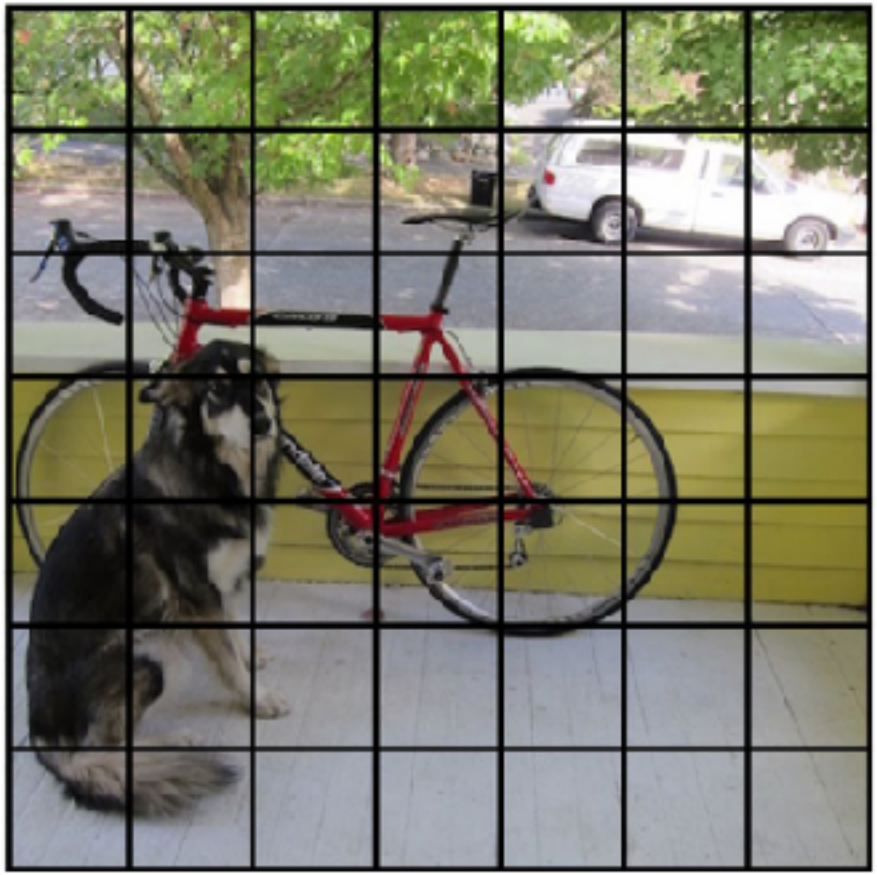
YOLO: You Only Look Once

- Detection Procedure



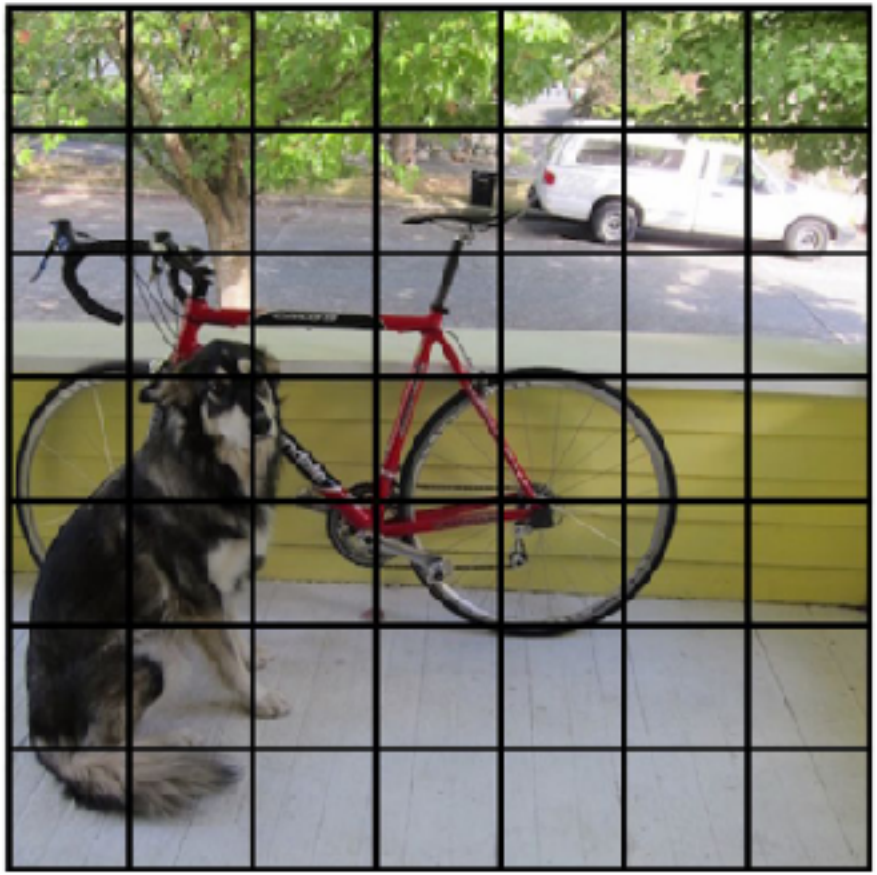
YOLO: You Only Look Once

We split the image into an S*S grid



YOLO: You Only Look Once

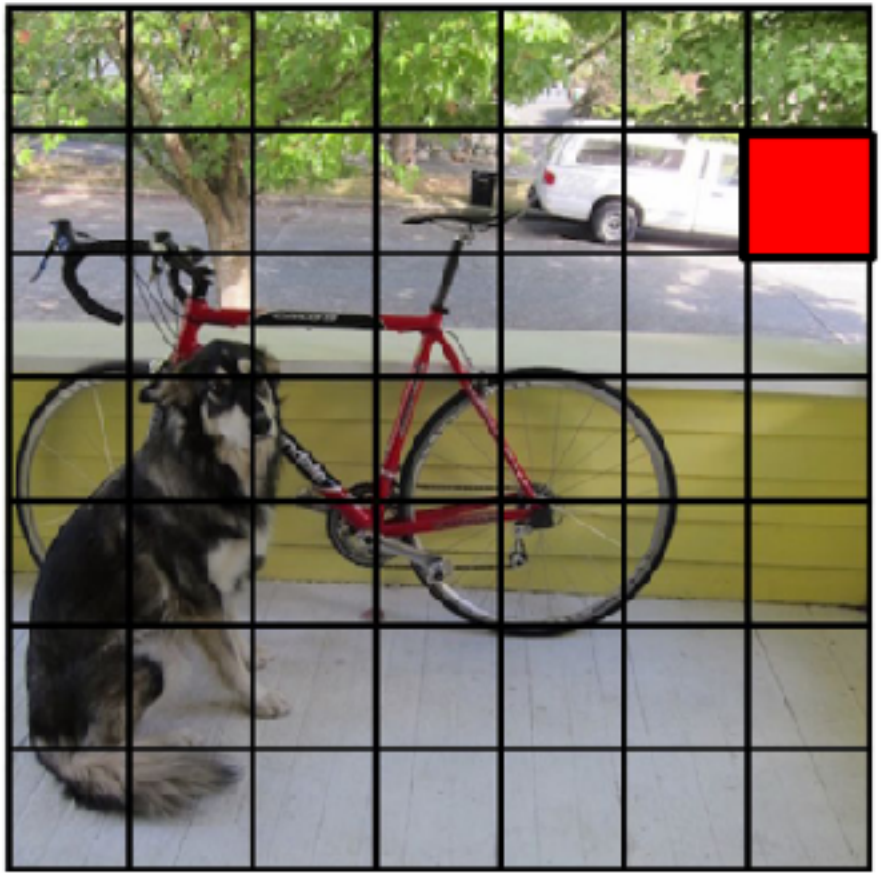
We split the image into an S*S grid



7*7 grid

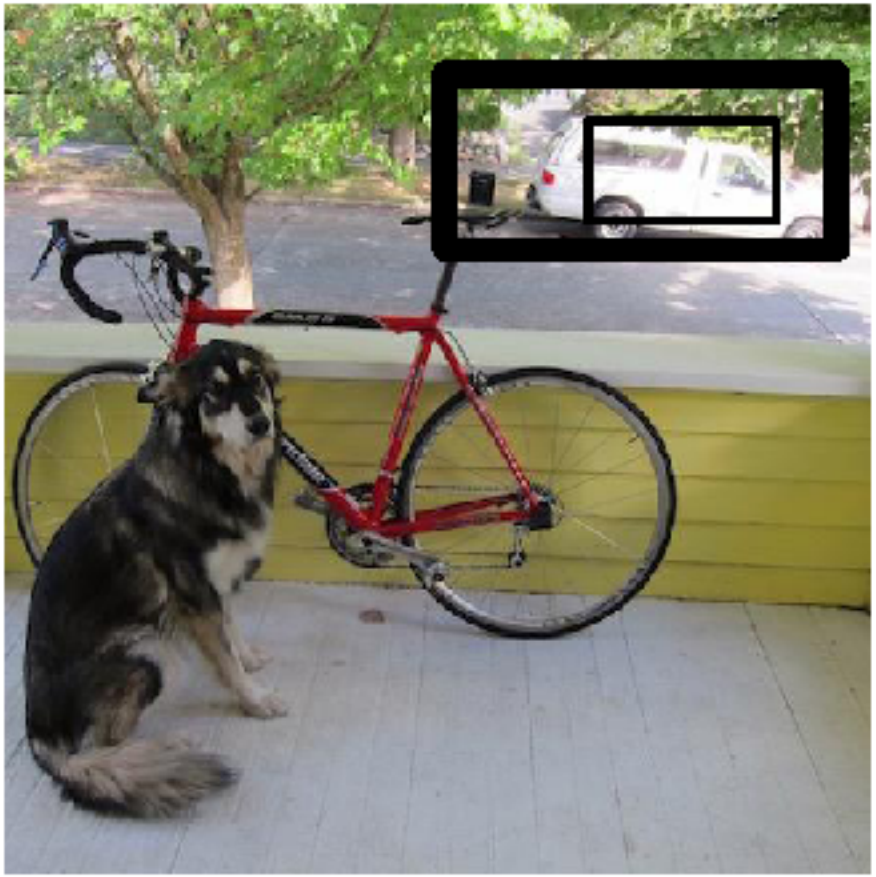
YOLO: You Only Look Once

Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)



YOLO: You Only Look Once

Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)

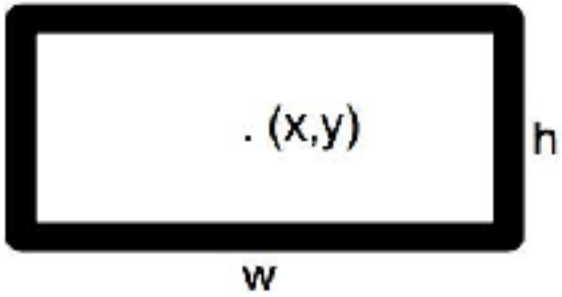


YOLO: You Only Look Once

Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)

B = 2

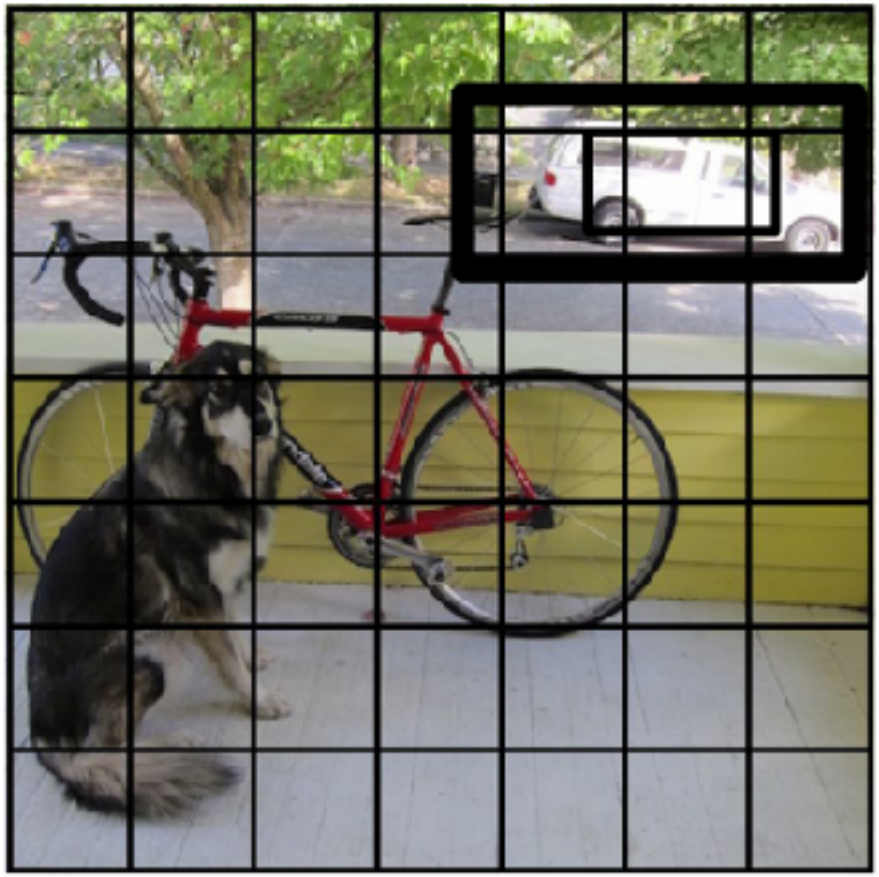
each box predict:



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

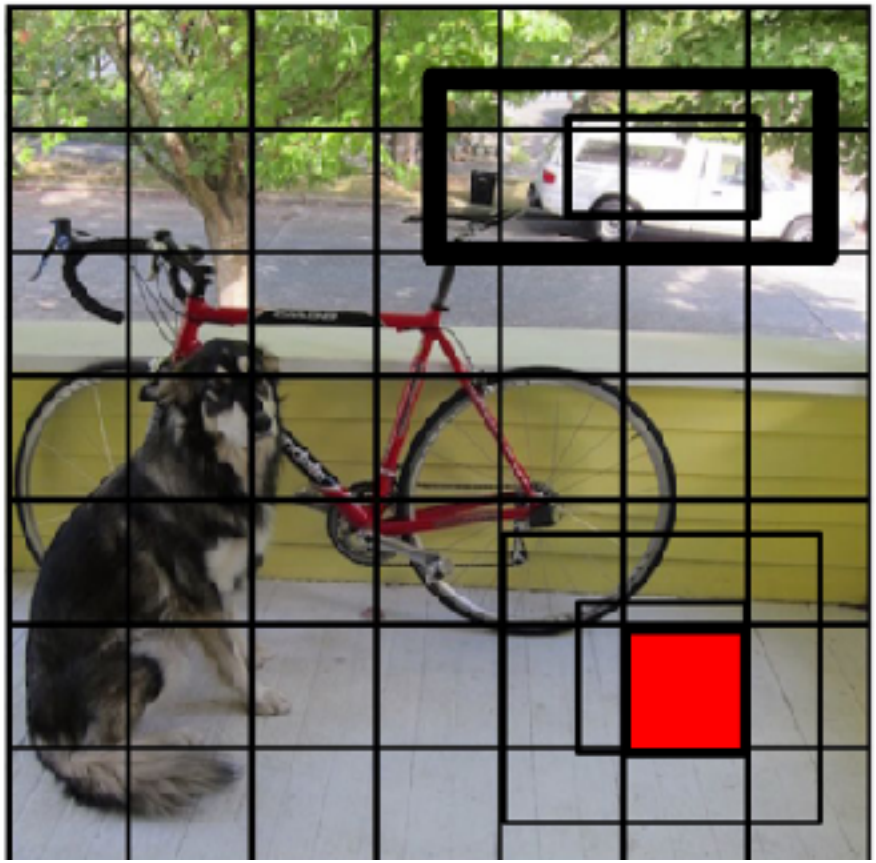
YOLO: You Only Look Once

Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)



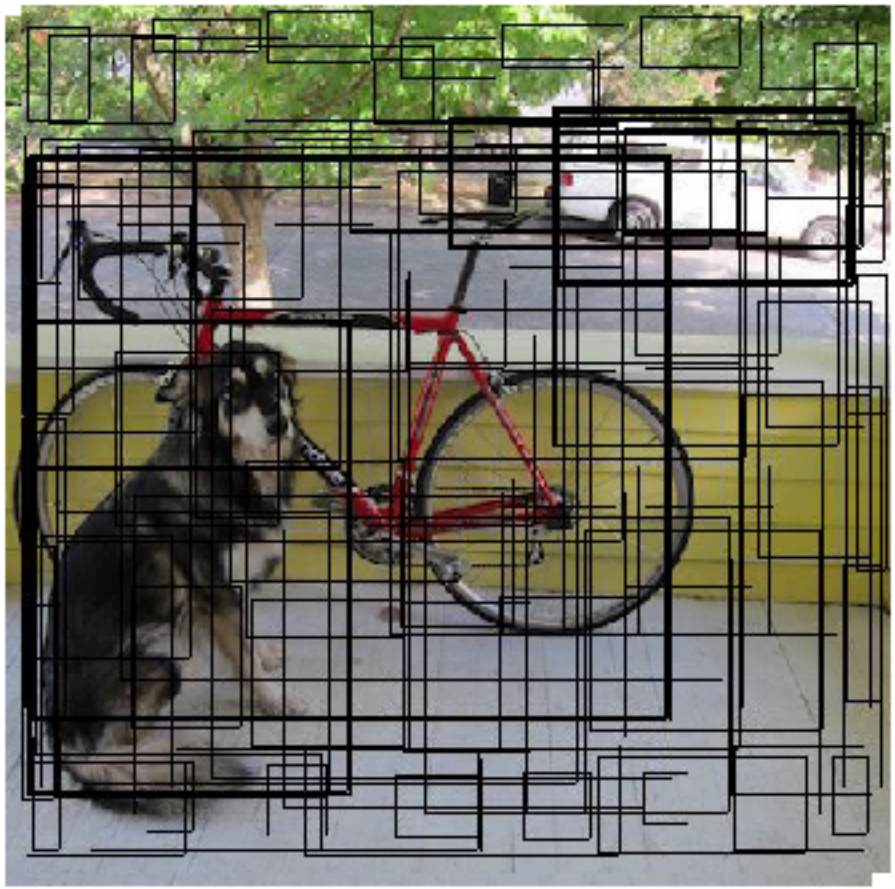
YOLO: You Only Look Once

Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)



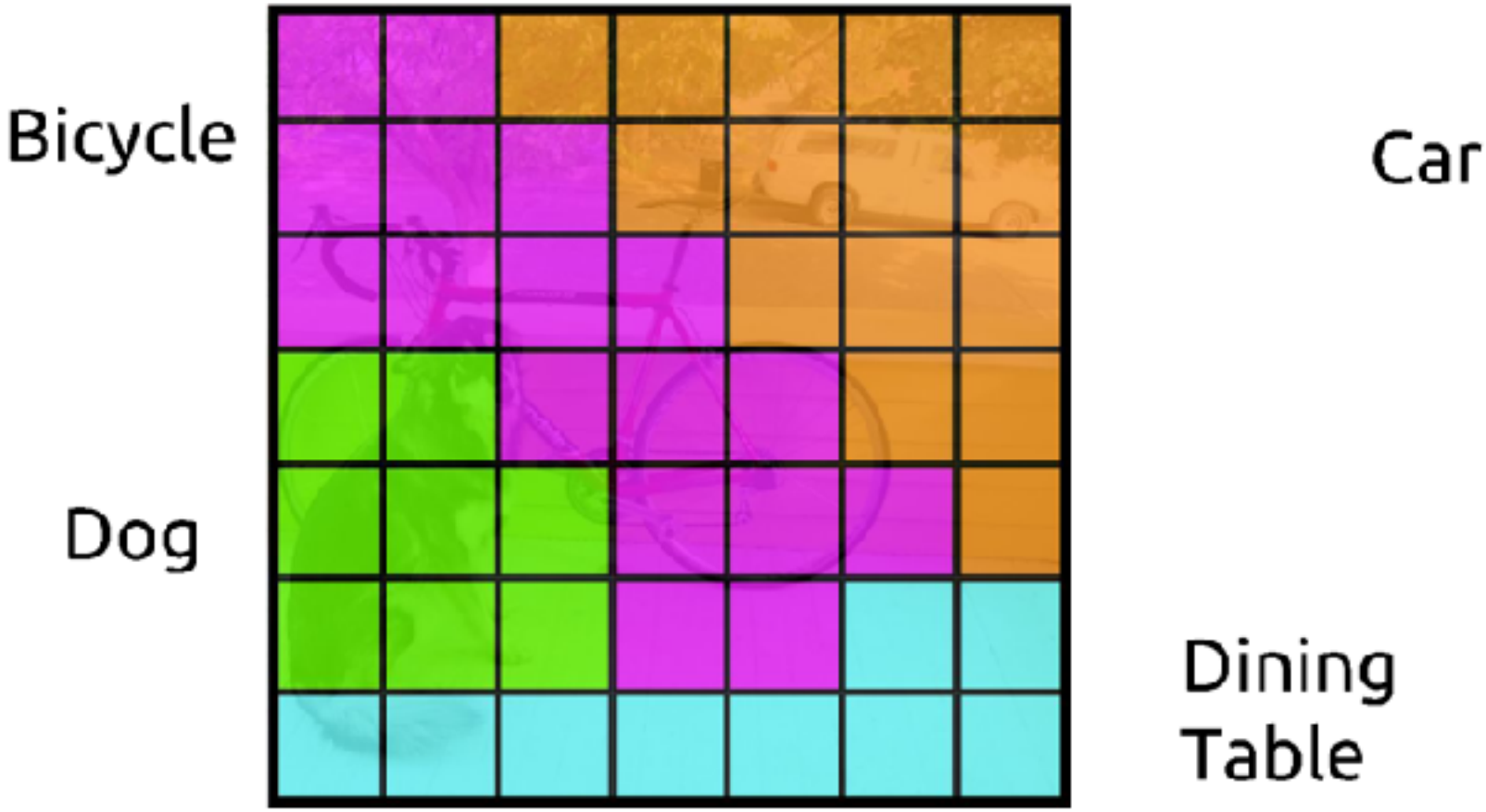
YOLO: You Only Look Once

Each cell predicts boxes and confidences: P(Object)

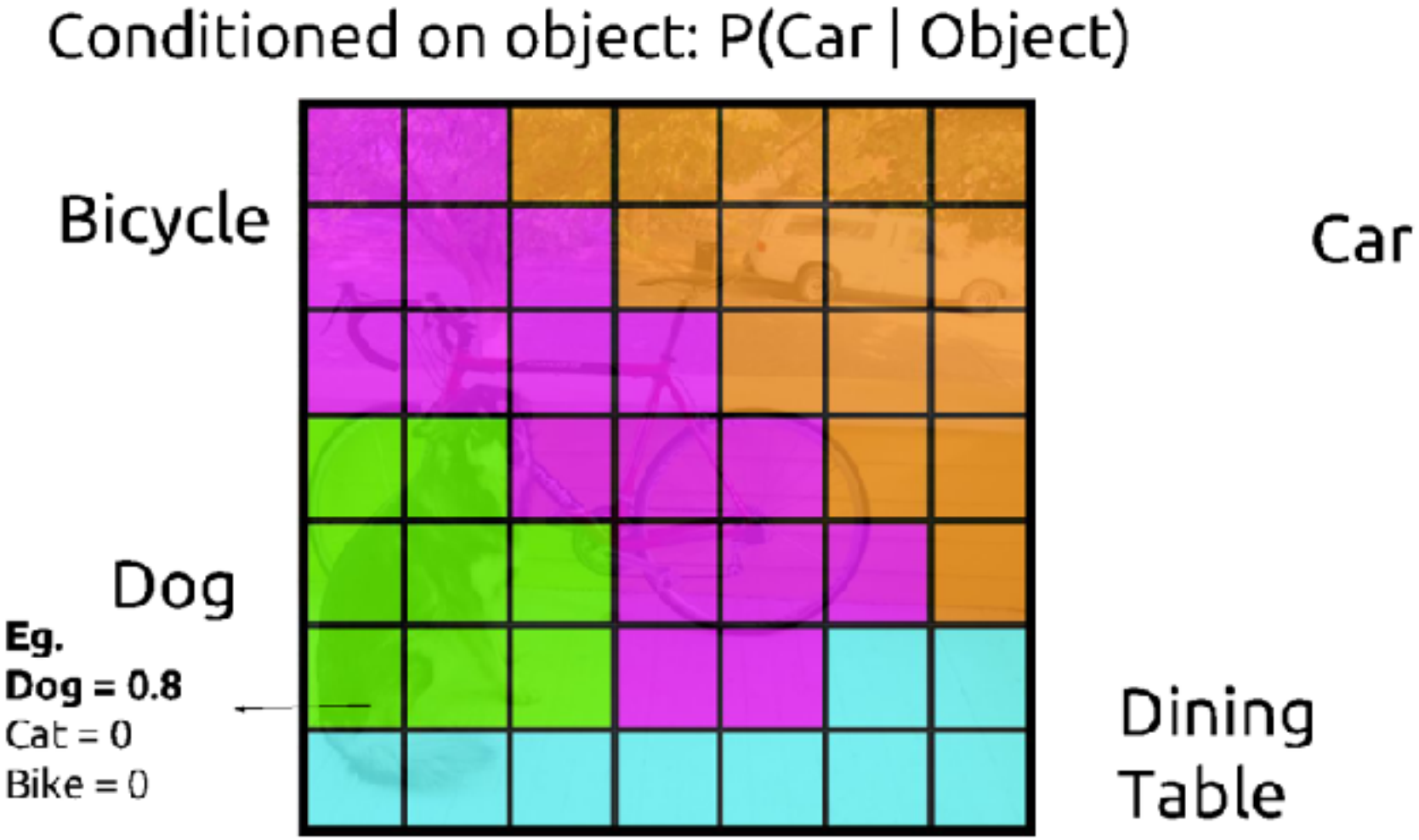


YOLO: You Only Look Once

Each cell also predicts a class probability.

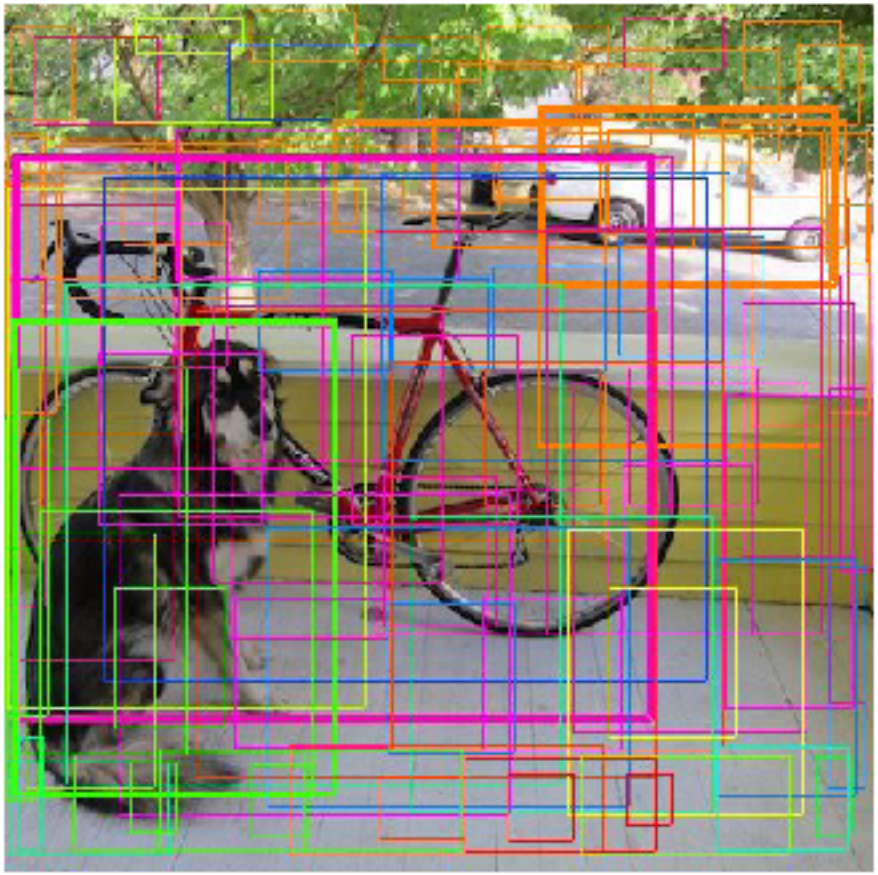


YOLO: You Only Look Once



YOLO: You Only Look Once

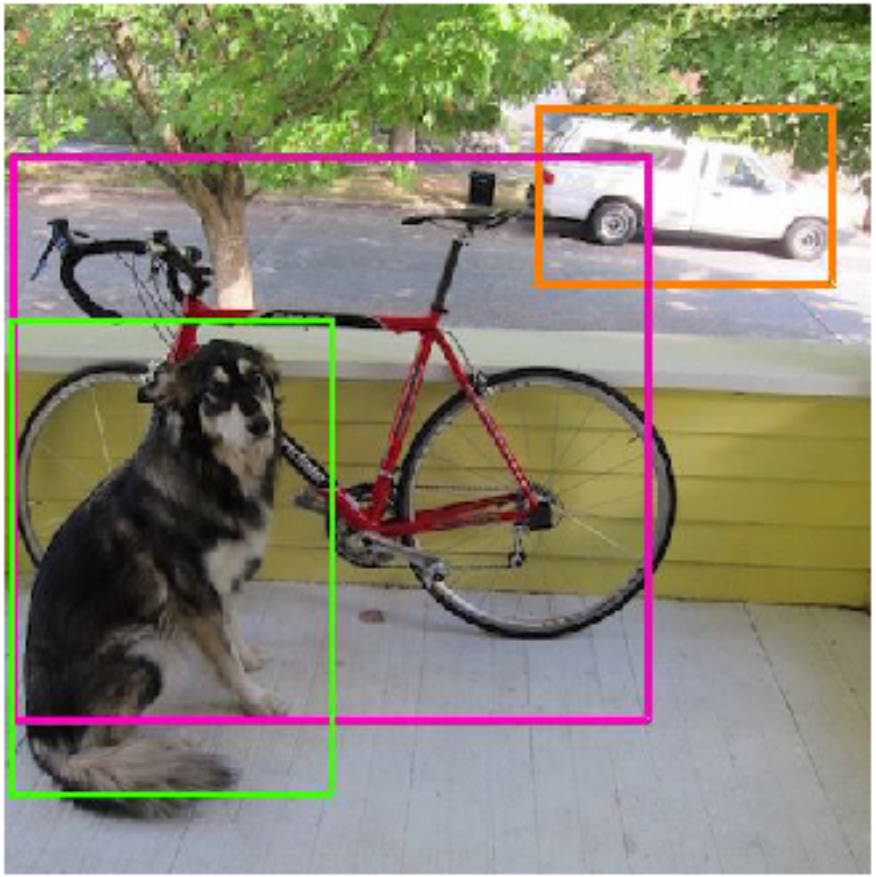
Then we combine the box and class predictions.



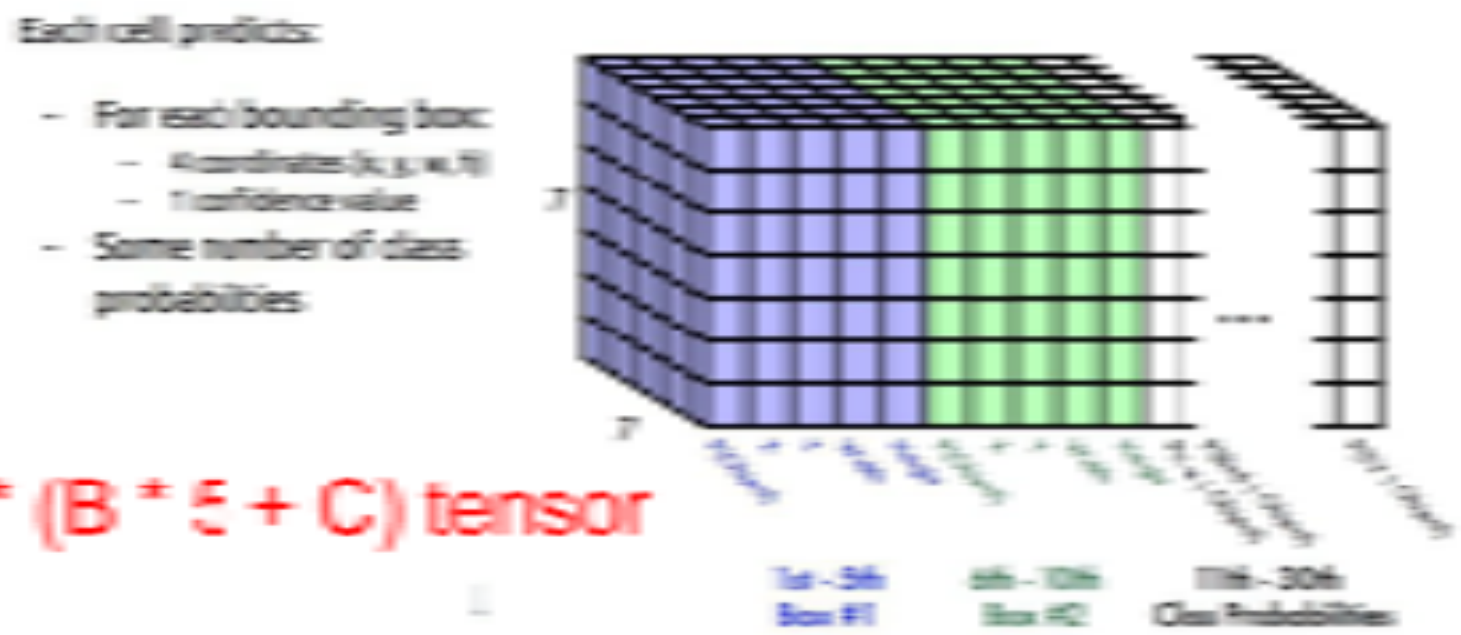
P(class|Object) * P(Object)
=P(class)

YOLO: You Only Look Once

Finally we do threshold detections and NMS



YOLO: You Only Look Once



YOLO: You Only Look Once

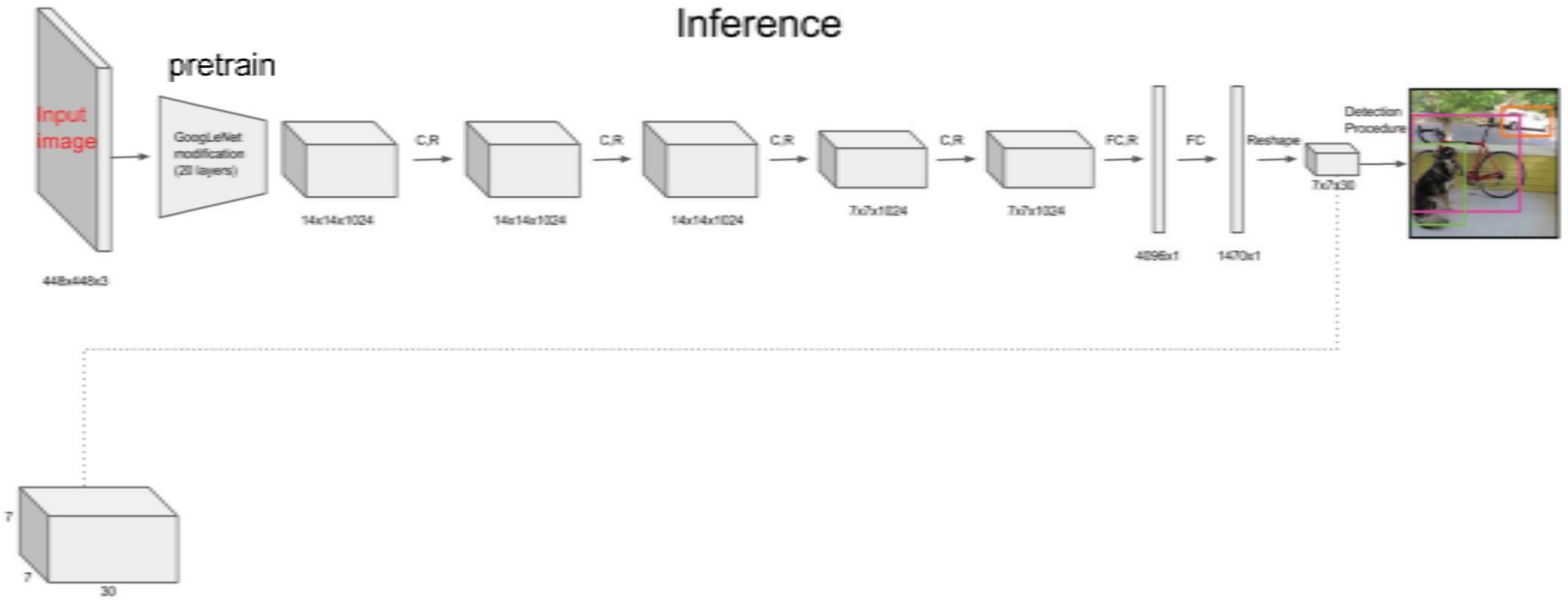
Network

YOLO: You Only Look Once

https://zhuanlan.zhihu.com/p/24916786?refer=xiaoleimlnote

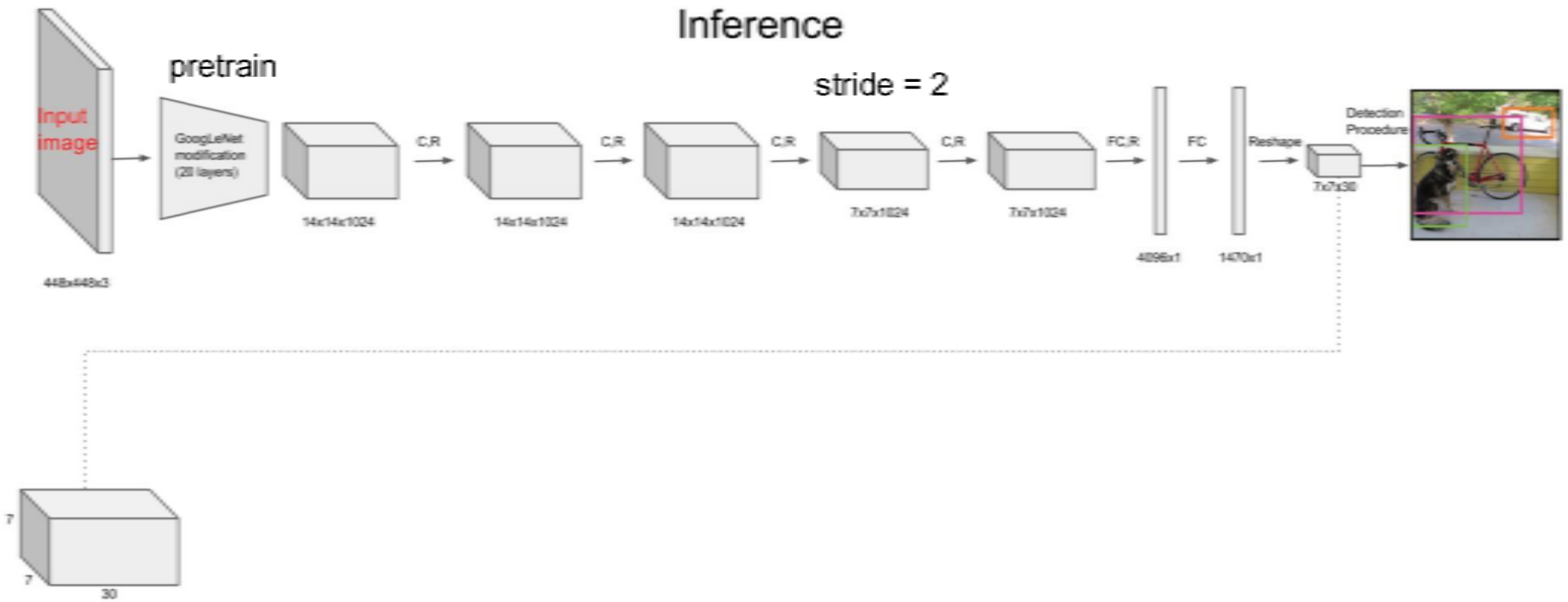
YOLO: You Only Look Once

https://zhuanlan.zhihu.com/p/24916786?refer=xiaoleimlnote



YOLO: You Only Look Once

https://zhuanlan.zhihu.com/p/24916786?refer=xiaoleimlnote

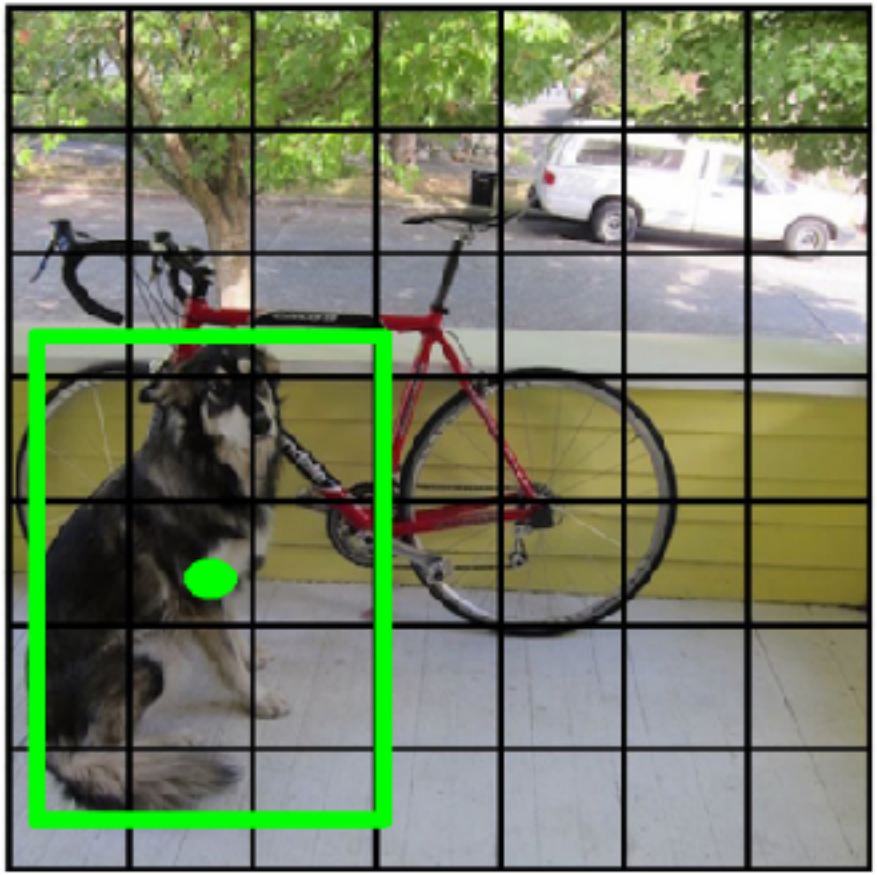


YOLO: You Only Look Once

Train

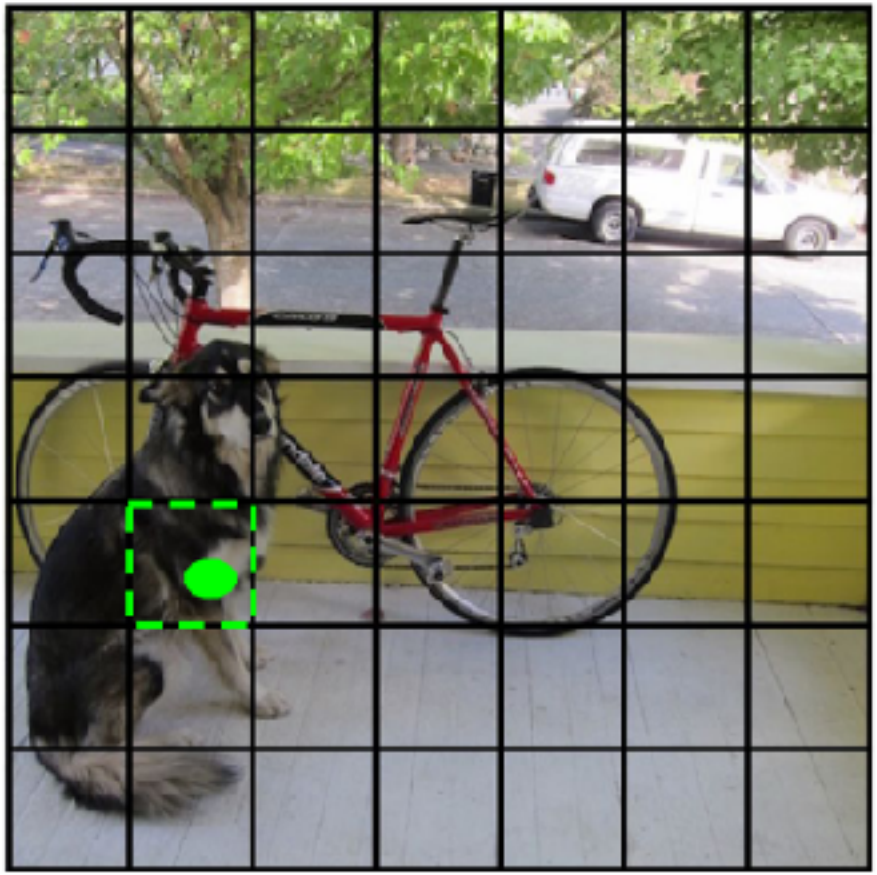
YOLO: You Only Look Once

During training, match example to the right cell



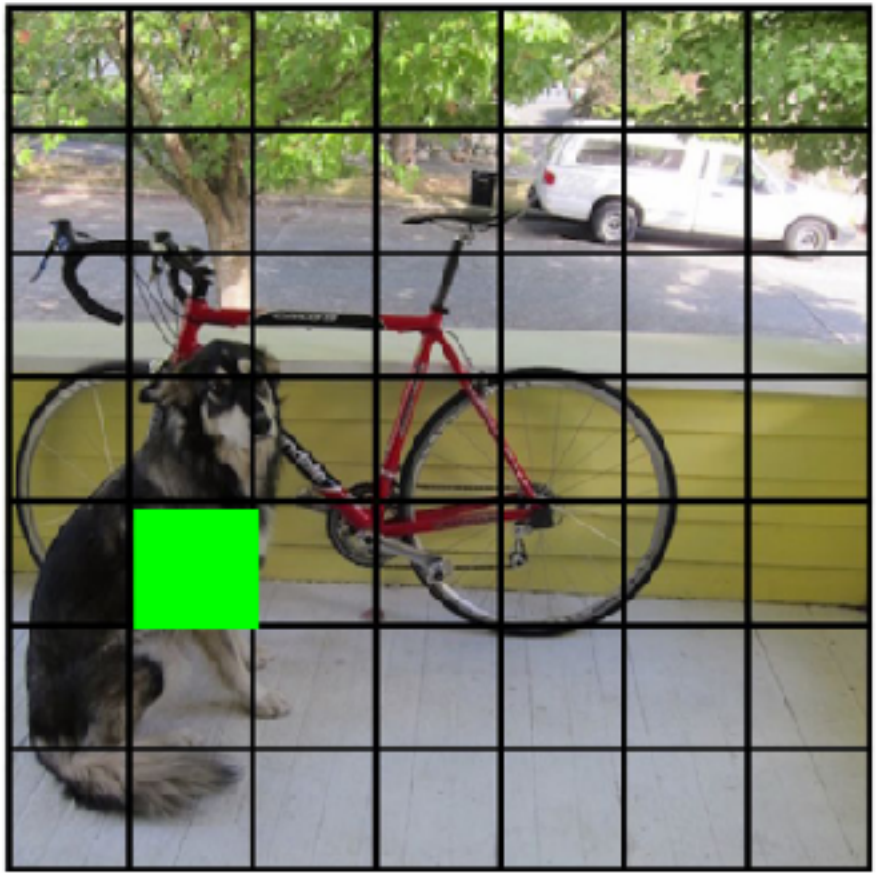
YOLO: You Only Look Once

During training, match example to the right cell



YOLO: You Only Look Once

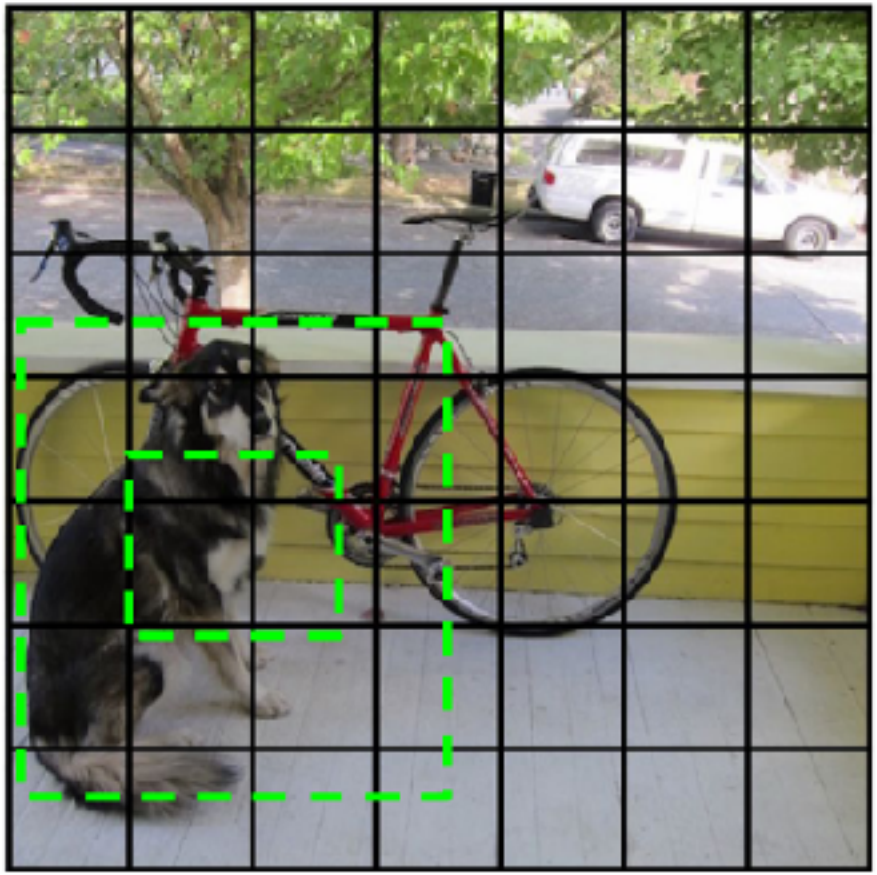
Adjust that cell's class prediction



Dog = 1
Cat = 0
Bike = 0
...

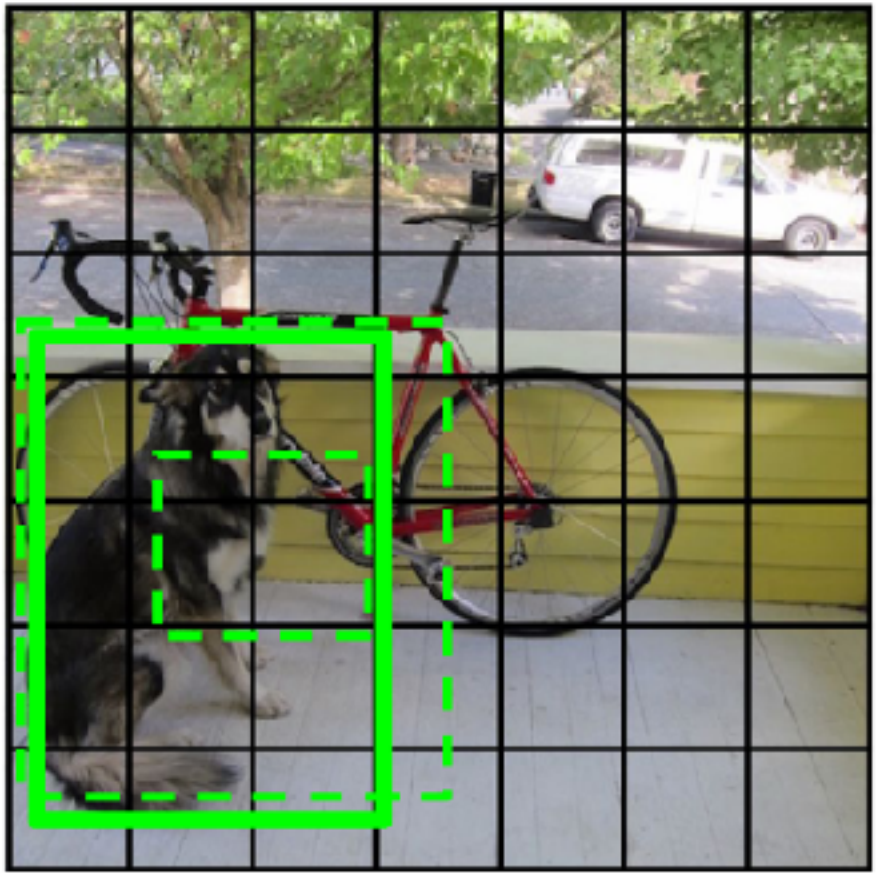
YOLO: You Only Look Once

Look at that cell's predicted boxes



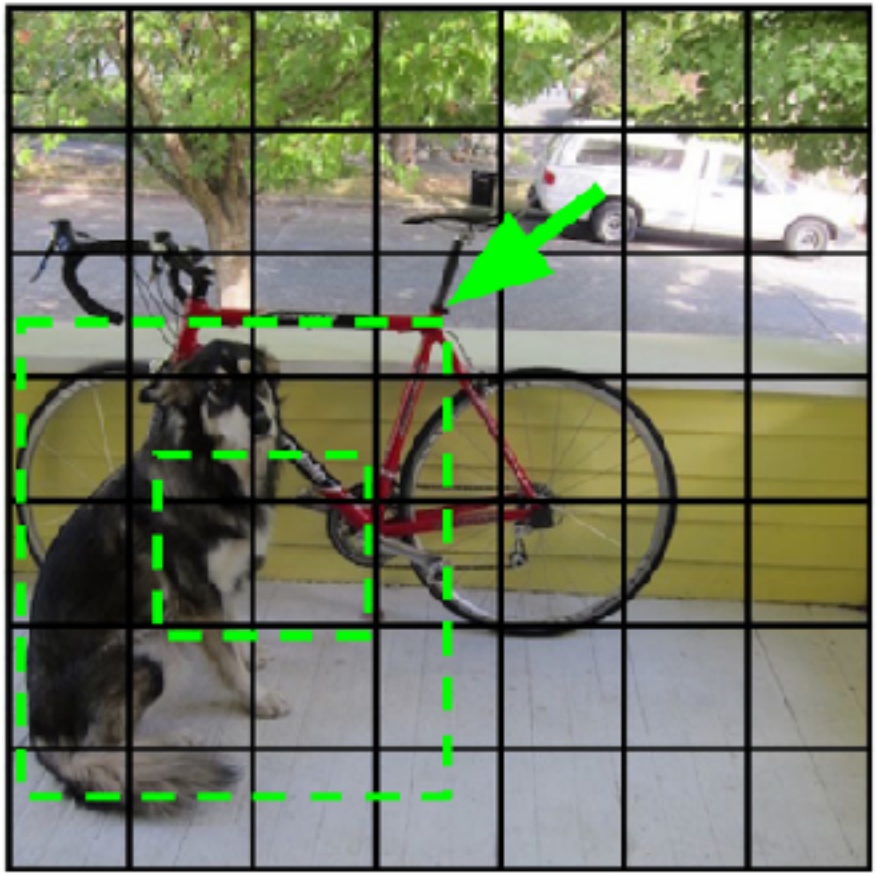
YOLO: You Only Look Once

Find the best one, adjust it, increase the confidence



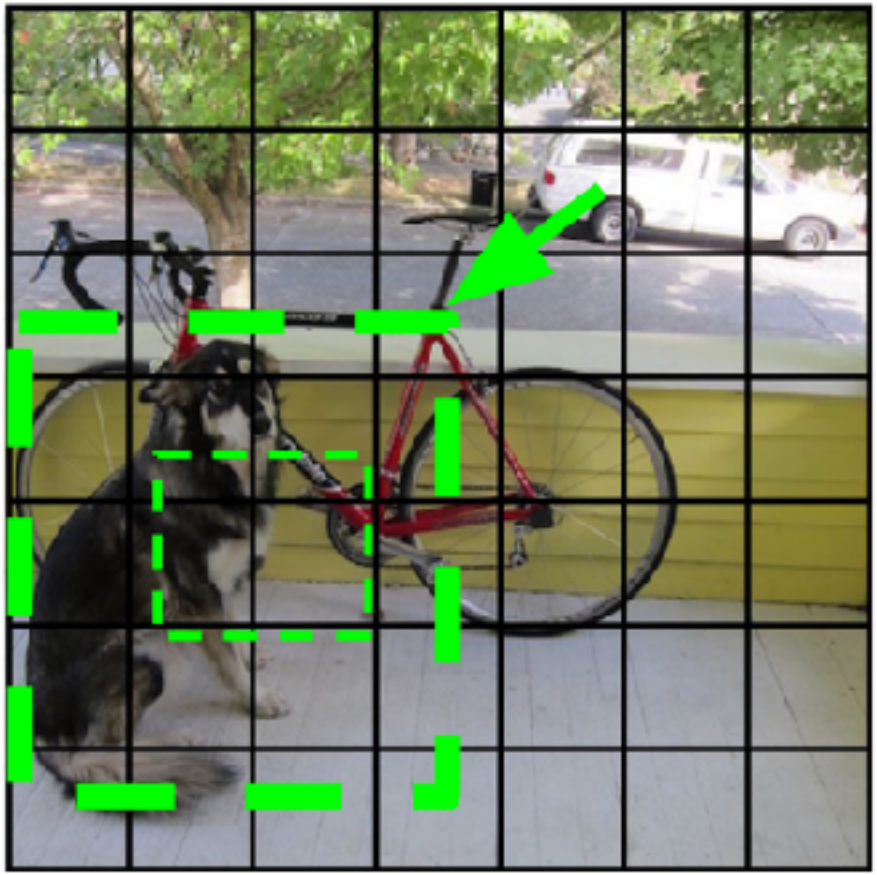
YOLO: You Only Look Once

Find the best one, adjust it, increase the confidence



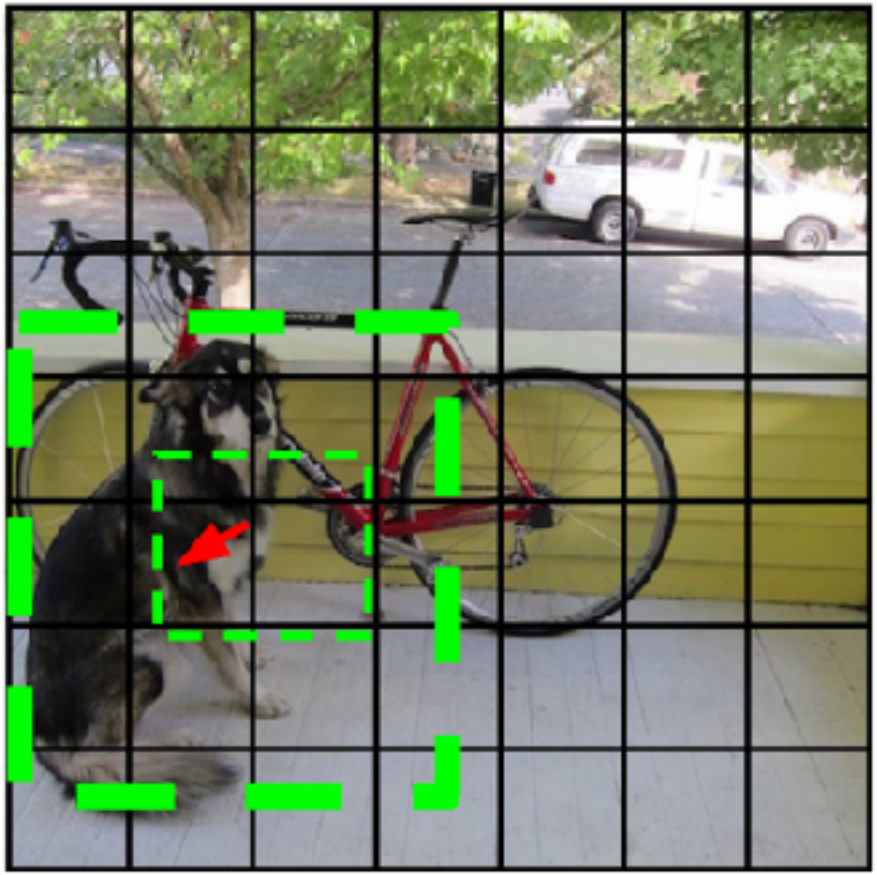
YOLO: You Only Look Once

Find the best one, adjust it, increase the confidence



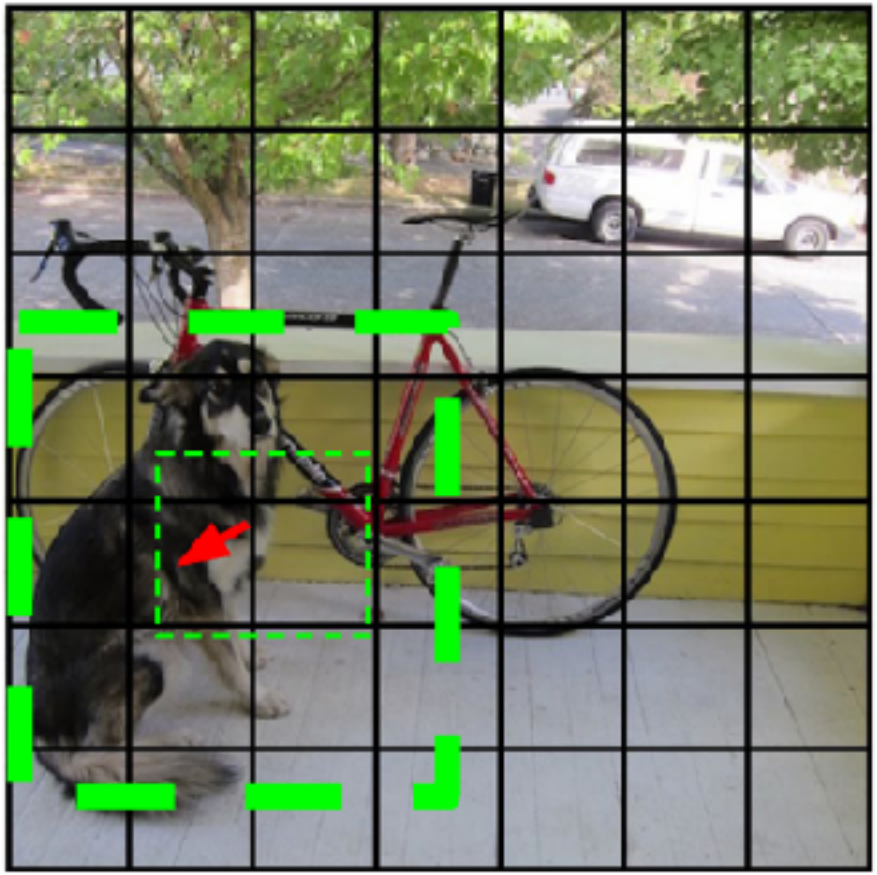
YOLO: You Only Look Once

Decrease the confidence of the other box



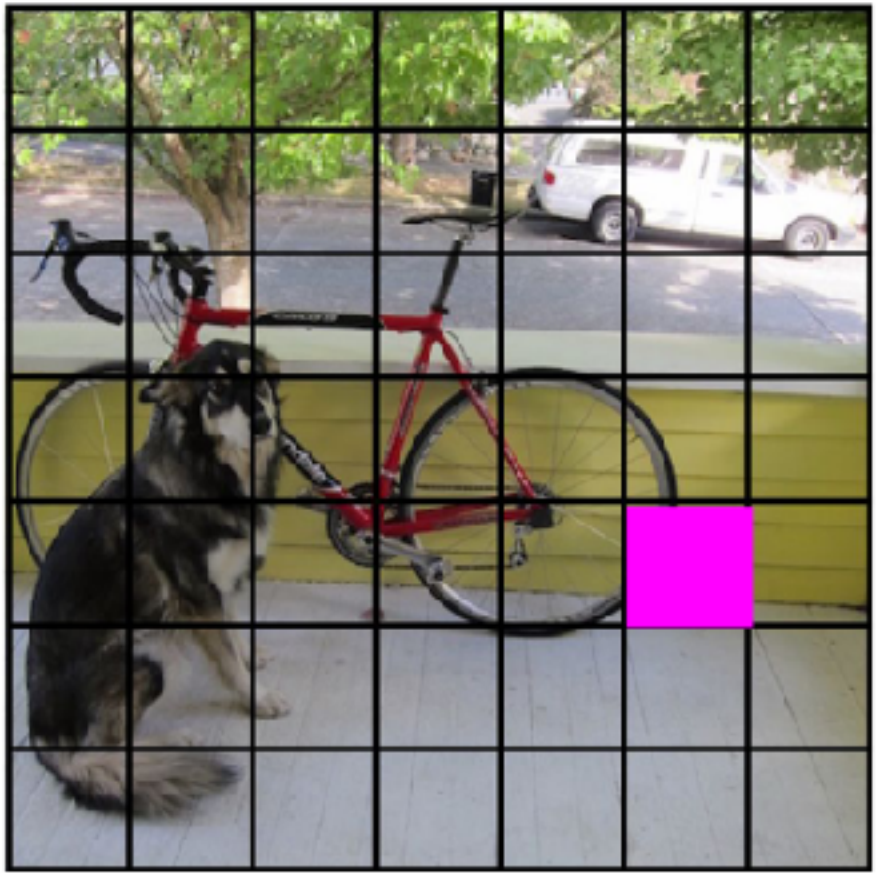
YOLO: You Only Look Once

Decrease the confidence of the other box



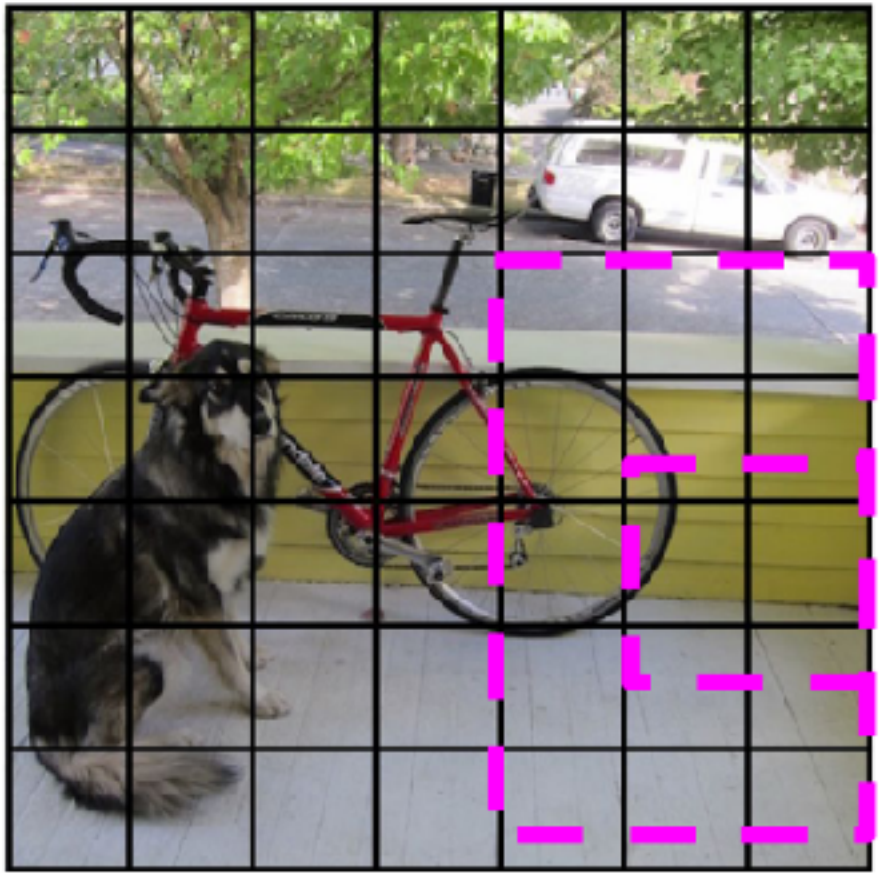
YOLO: You Only Look Once

Some cells don't have any ground truth detections!



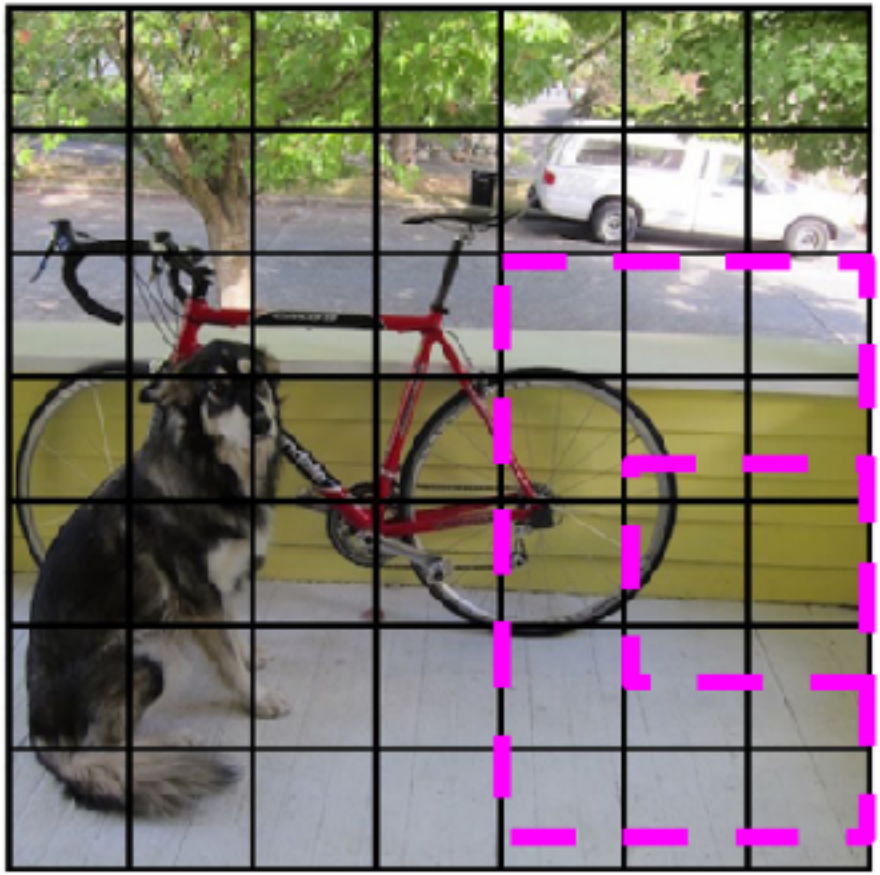
YOLO: You Only Look Once

Some cells don't have any ground truth detections!



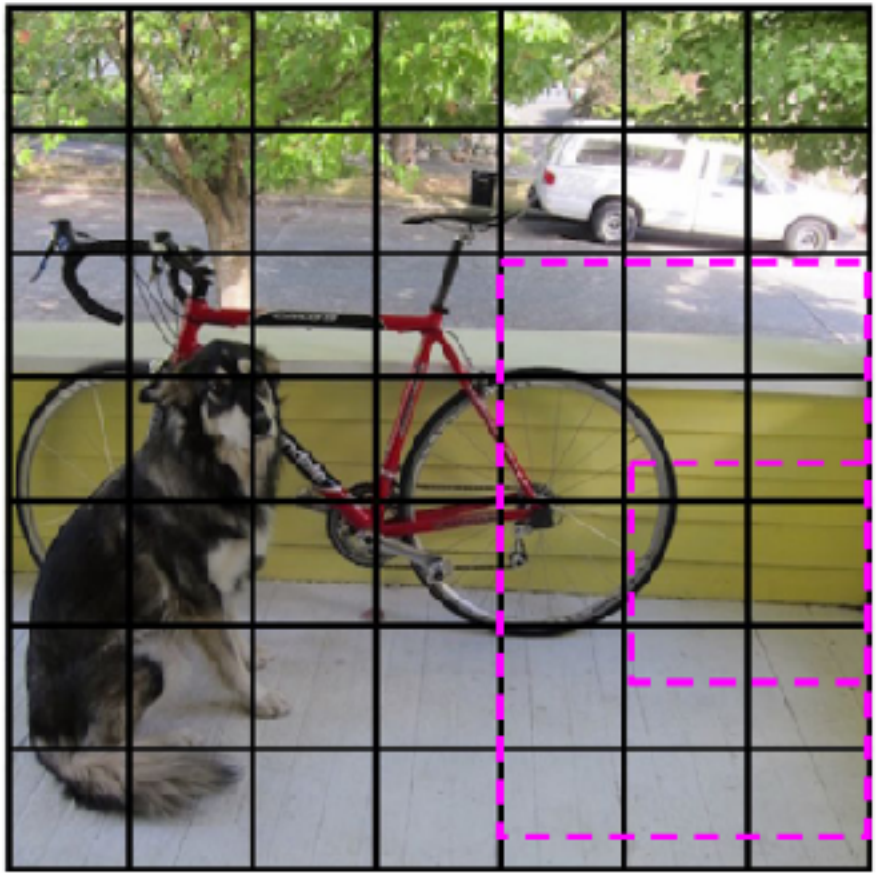
YOLO: You Only Look Once

Decrease the confidence of boxes boxes



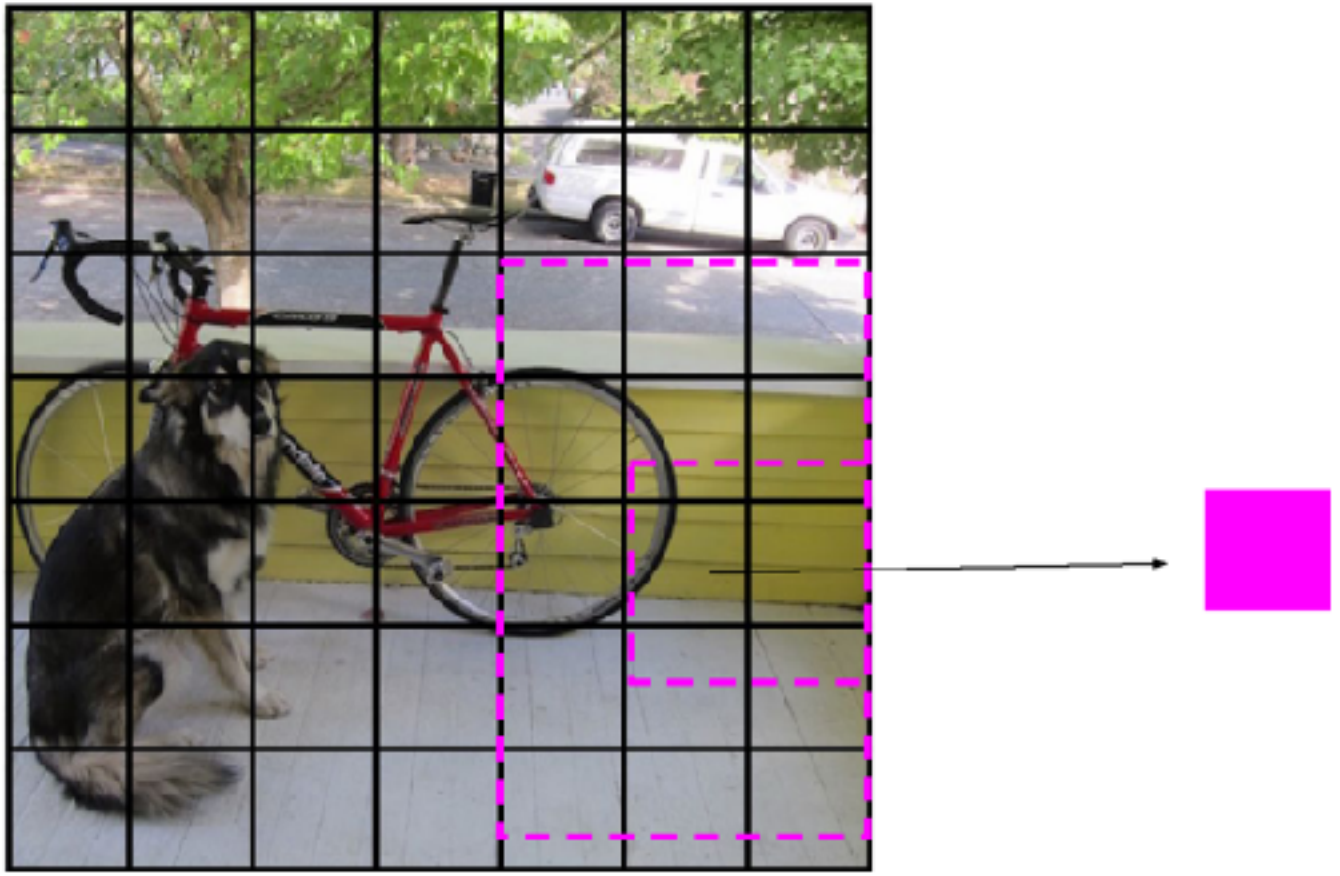
YOLO: You Only Look Once

Decrease the confidence of these boxes



YOLO: You Only Look Once

Don't adjust the class probabilities or coordinates



YOLO: Loss Function

- $\mathbb{1}_i^{obj}$ is 1 if there is an object in cell i and 0 otherwise,
- $\mathbb{1}_{i,j}^{obj}$ is 1 if there is an object in cell i and predicted box j is the most fitting one, 0 otherwise
- $p_{i,c}$ is 1 if there is an object of class c in cell i, and 0 otherwise,
- x_i, y_i, w_i, h_i the annotated object bounding box (defined only if $\mathbb{1}_i^{obj} = 1$, and relative in location and scale to the cell),
- $c_{i,j}$ IOU between the predicted box and the ground truth target.

YOLO: Loss Function

The training procedure first computes on each image the value of the $\mathbf{1}_{i,j}^{obj}$s and $c_{i,j}$, and then does one step to minimize

$$\lambda_{coord} \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbf{1}_{i,j}^{obj} \left((x_i - \hat{x}_{i,j})^2 + (y_i - \hat{y}_{i,j})^2 + \left(\sqrt{w_i} - \sqrt{\hat{w}_{i,j}} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_{i,j}} \right)^2 \right)$$

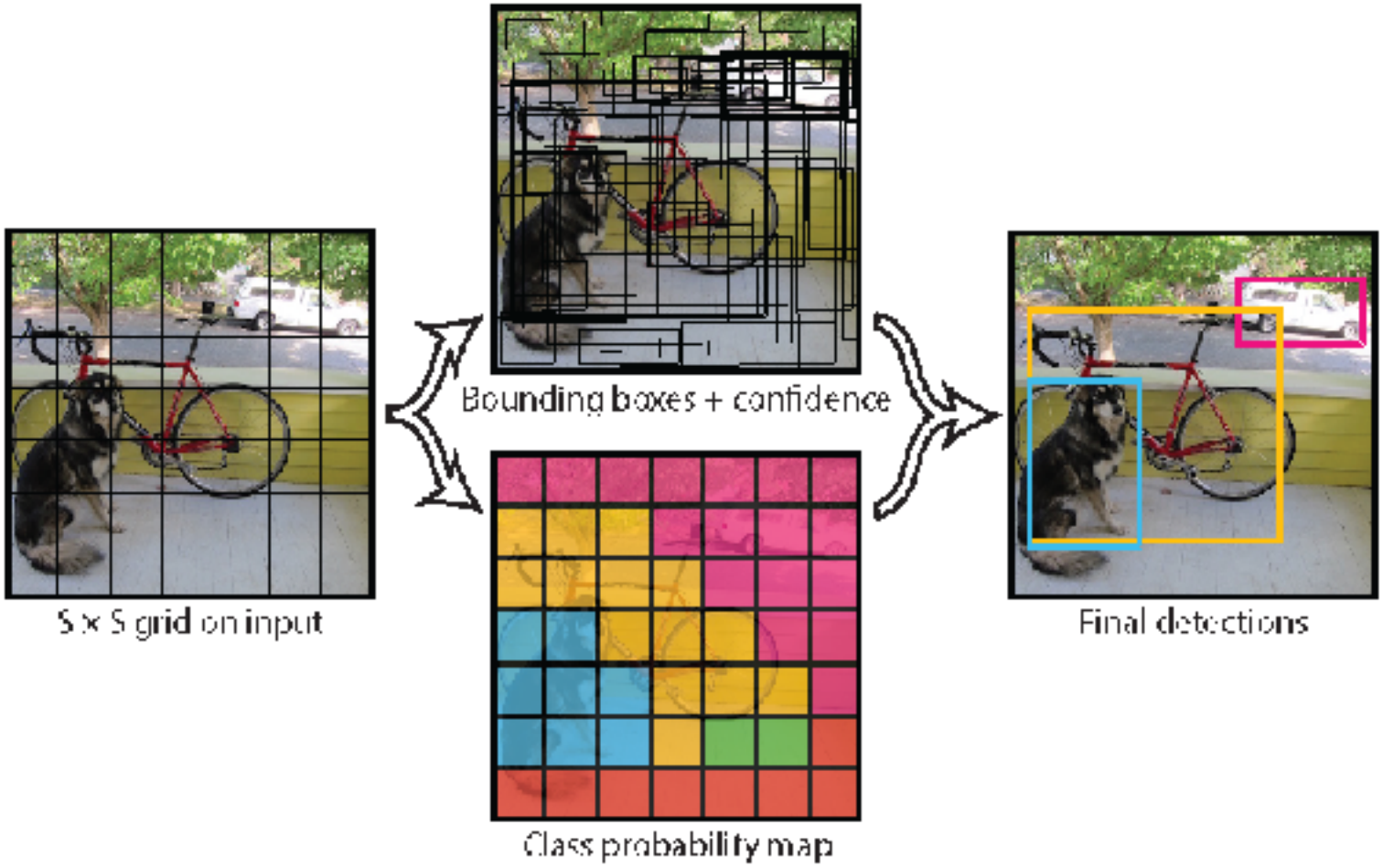
$$+ \lambda_{obj} \sum_{i=1}^{S^2} \sum_{j=1}^{B} \mathbf{1}_{i,j}^{obj} (c_{i,j} - \hat{c}_{i,j})^2 + \lambda_{noobj} \sum_{i=1}^{S^2} \sum_{j=1}^{B} \left(1 - \mathbf{1}_{i,j}^{obj} \right) \hat{c}_{i,j}^2$$

$$+ \lambda_{classes} \sum_{i=1}^{S^2} \mathbf{1}_{i}^{obj} \sum_{c=1}^{C} \left(p_{i,c} - \hat{p}_{i,c} \right)^2 .$$

where $\hat{p}_{i,c}, \hat{x}_{i,j}, \hat{y}_{i,j}, \hat{w}_{i,j}, \hat{h}_{i,j}, \hat{c}_{i,j}$ are the network's outputs.

(slightly re-written from Redmon et al. 2015)

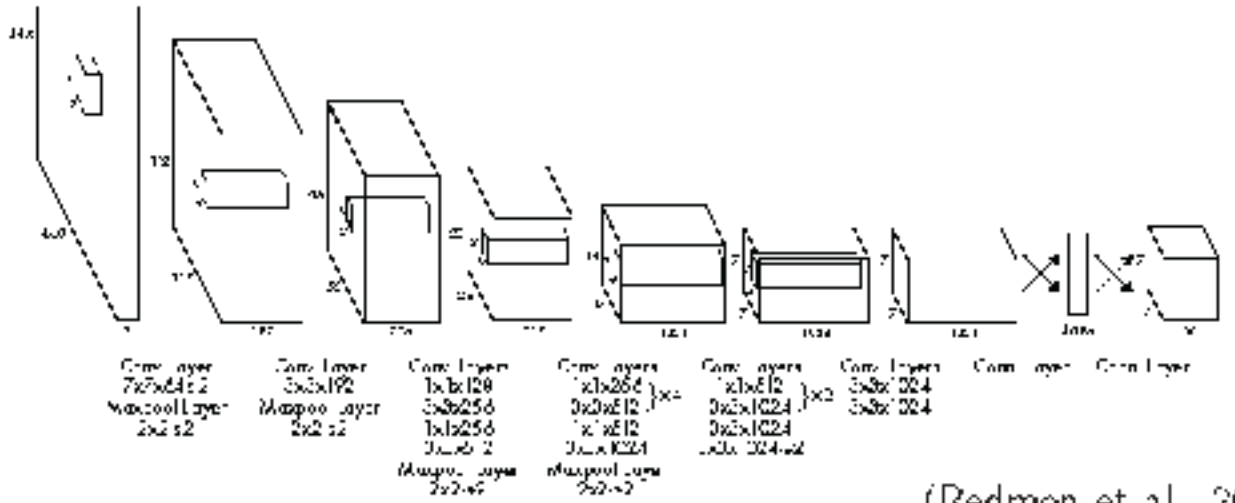
YOLO: You Only Look Once



(Redmon et al., 2015)

YOLO: You Only Look Once

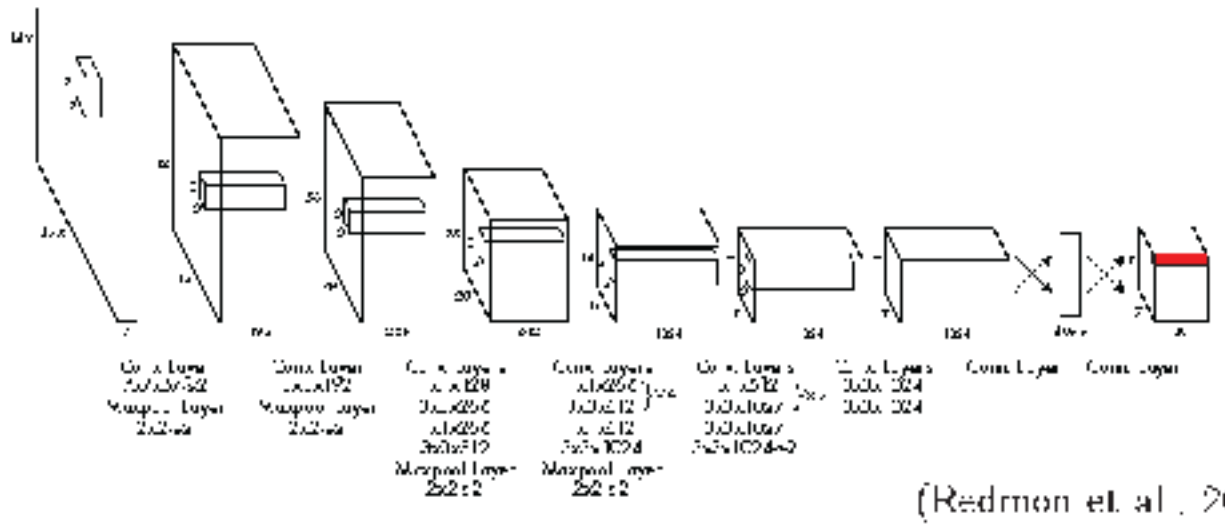
The output corresponds to splitting the image into a regular $S \times S$ grid, with $S = 7$



(Redmon et al., 2015)

YOLO: You Only Look Once

The output corresponds to splitting the image into a regular $S \times S$ grid, with $S = 7$, and for each cell, to predict a 30d vector

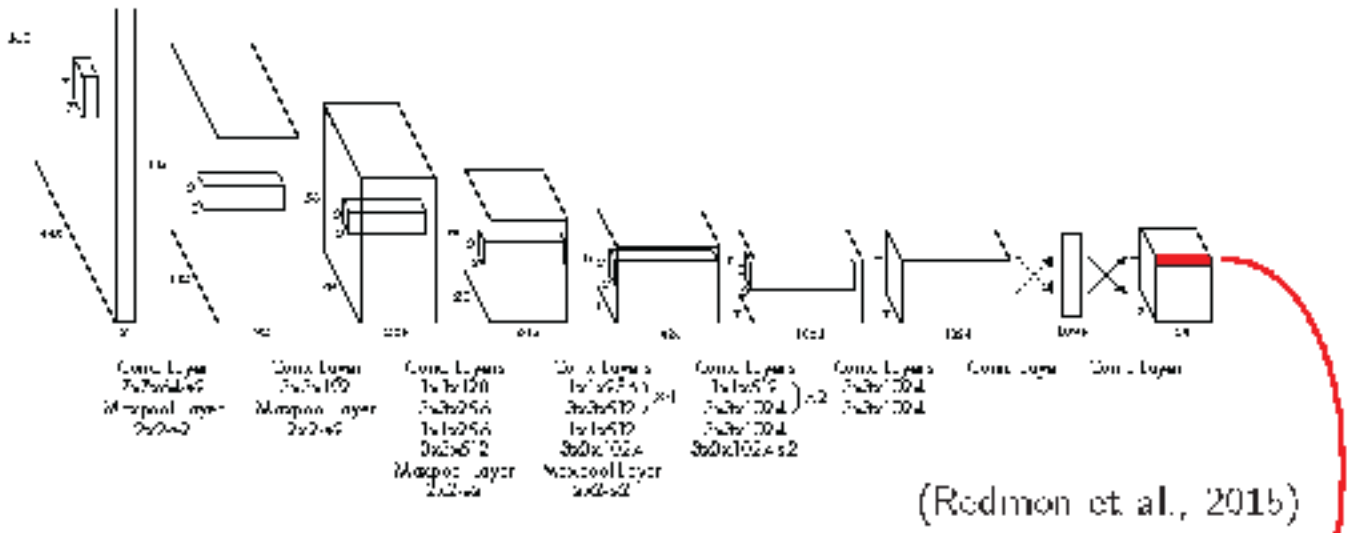


(Redmon et al., 2015)

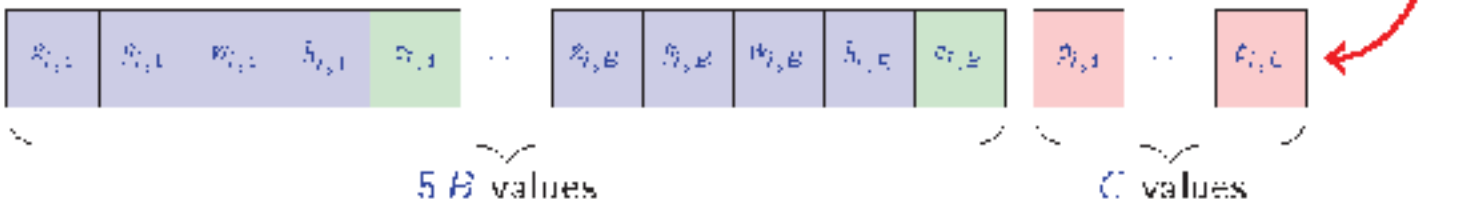
YOLO: You Only Look Once

The output corresponds to splitting the image into a regular $S \times S$ grid, with $S = 7$, and for each cell, to predict a 30d vector:

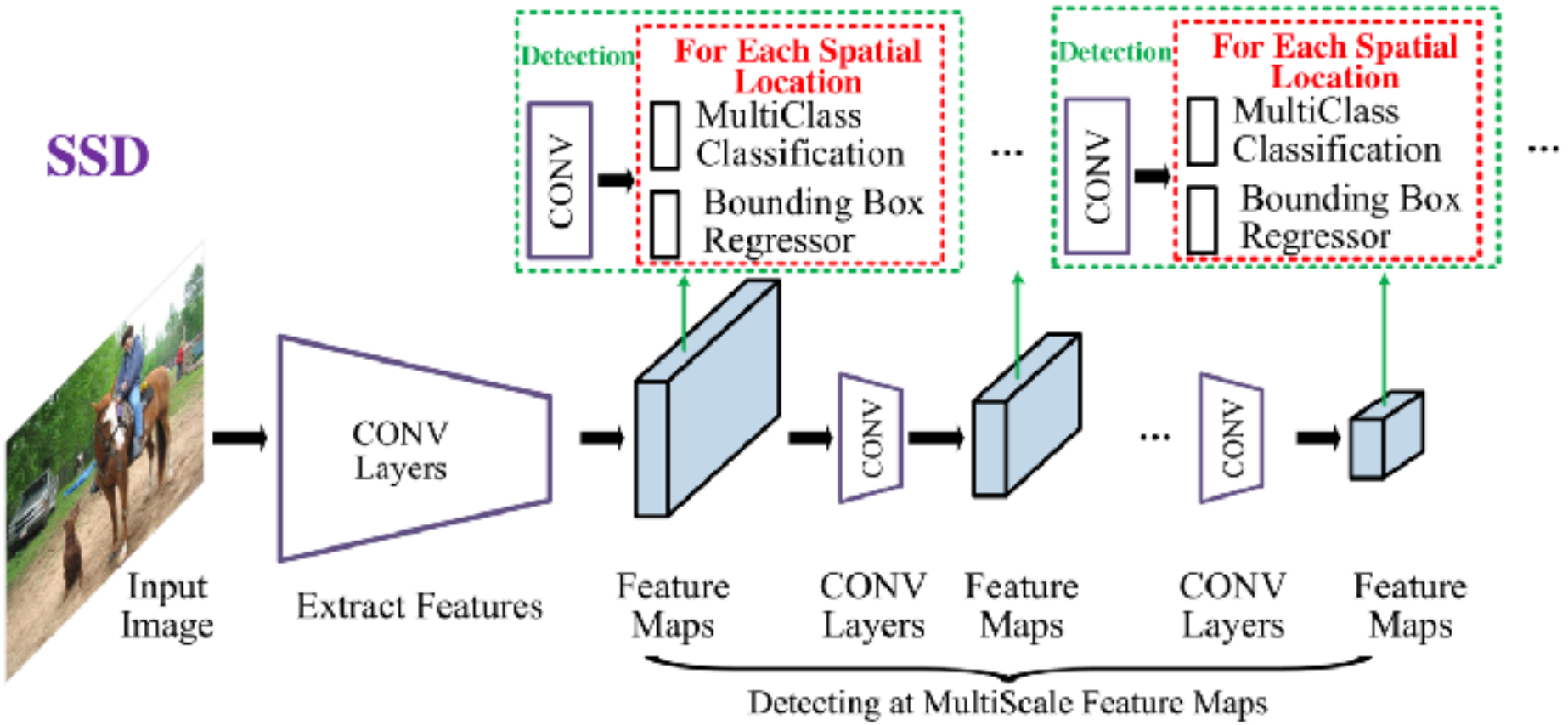
- $B = 2$ bounding boxes coordinates and confidence,
- $C = 20$ class probabilities, corresponding to the classes of Pascal VOC



(Redmon et al., 2015)

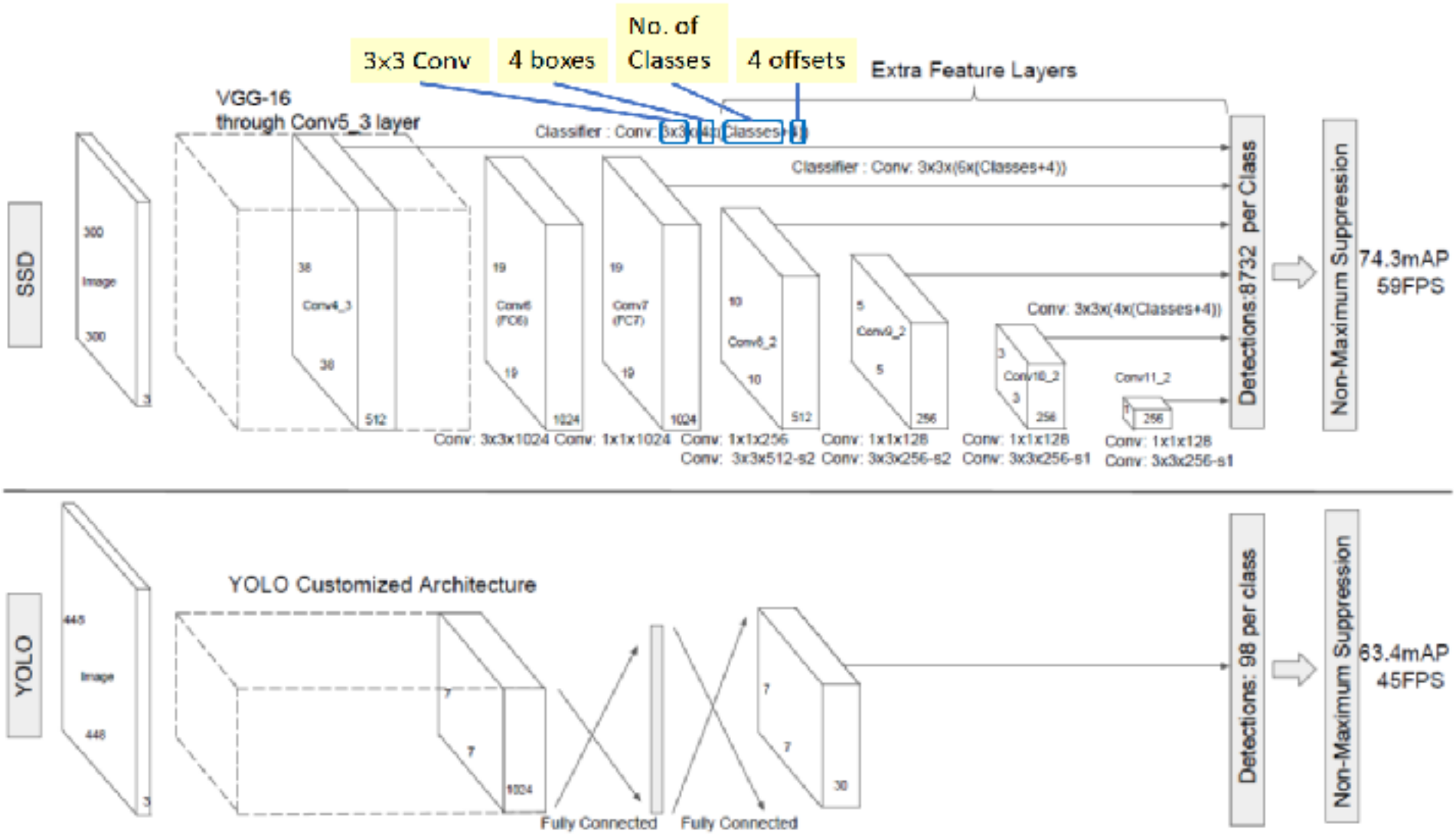


SSD: Single Shot Multibox Detector



Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018

SSD: Single Shot Multibox Detector



Li Liu et al. Deep Learning for Generic Object Detection: A Survey 2018

SSD: Single Shot Multibox Detector

- After going through a certain of convolutions for feature extraction, we obtain **a feature layer of size *m*×*n* (number of locations) with *p* channels**, such as 8×8 or 4×4 above. And a 3×3 conv is applied on this *m*×*n*×*p* feature layer.

- **For each location, we got *k* bounding boxes.** These k bounding boxes have different sizes and aspect ratios. The concept is, maybe a vertical rectangle is more fit for human, and a horizontal rectangle is more fit for car.

- For each of the bounding box, we will compute *c* class scores and 4 offsets relative to the original default bounding box shape.

- Thus, we got **(*c*+4)*kmn* outputs.**

https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11

SSD: Single Shot Multibox Detector

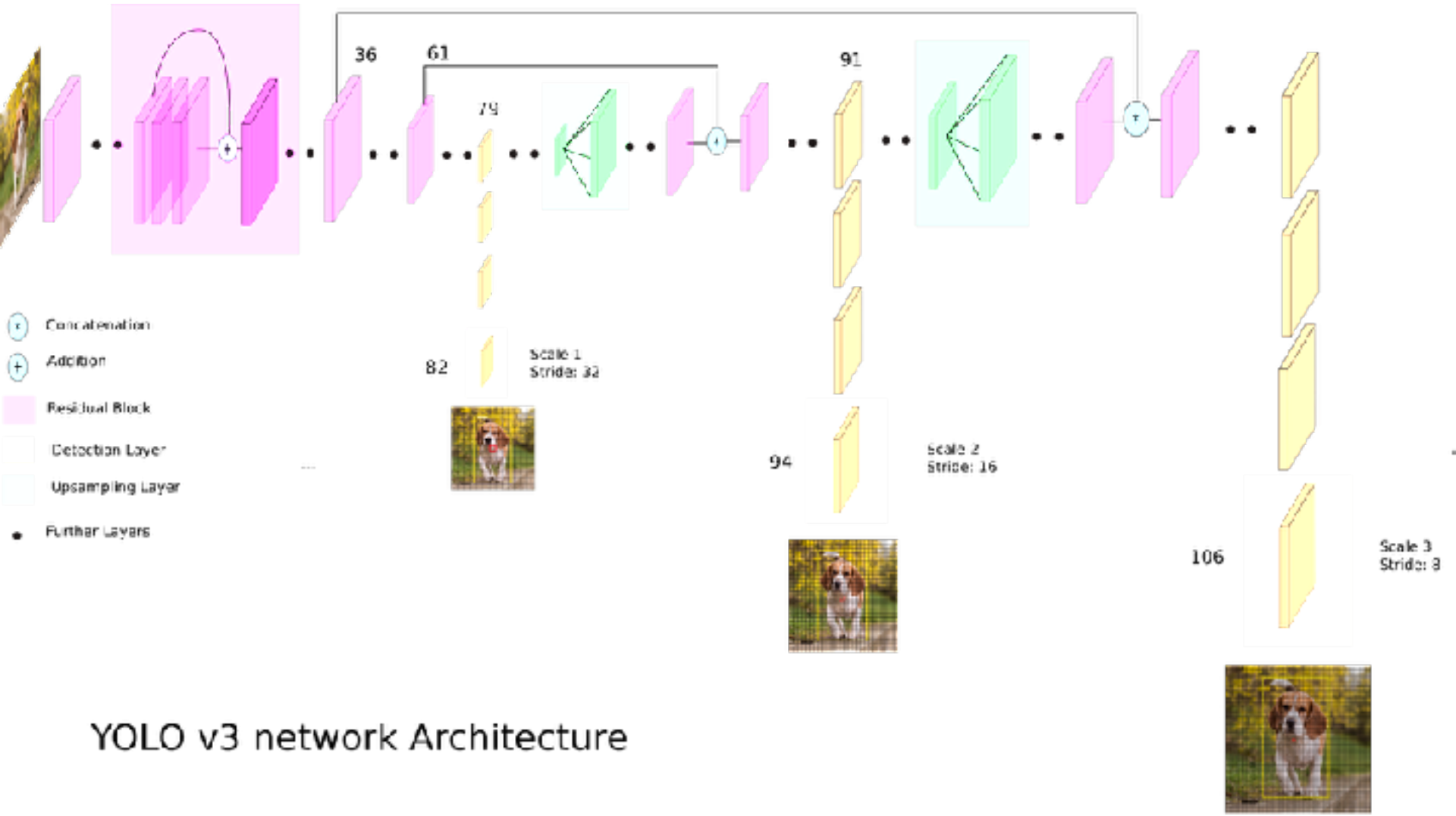
- At **Conv4_3,**
  - size 38×38×512
  - 4 bounding boxes (classes + 4) outputs.
  - the output is 38×38×4×(c+4).
  - For 20 object classes plus one background class
  - 38×38×4×(21+4) = 144,400.
  - there are 38×38×4 = 5776 bounding boxes.

- Similarly for other conv layers:
  - Conv7: 19×19×6 = 2166 boxes (6 boxes for each location)
  - Conv8_2: 10×10×6 = 600 boxes (6 boxes for each location)
  - Conv9_2: 5×5×6 = 150 boxes (6 boxes for each location)
  - Conv10_2: 3×3×4 = 36 boxes (4 boxes for each location)
  - Conv11_2: 1×1×4 = 4 boxes (4 boxes for each location)

5776 + 2166 + 600 + 150 + 36 +4 = **8732 boxes in total**.
YOLO only got 7×7×2 = 98 boxes.

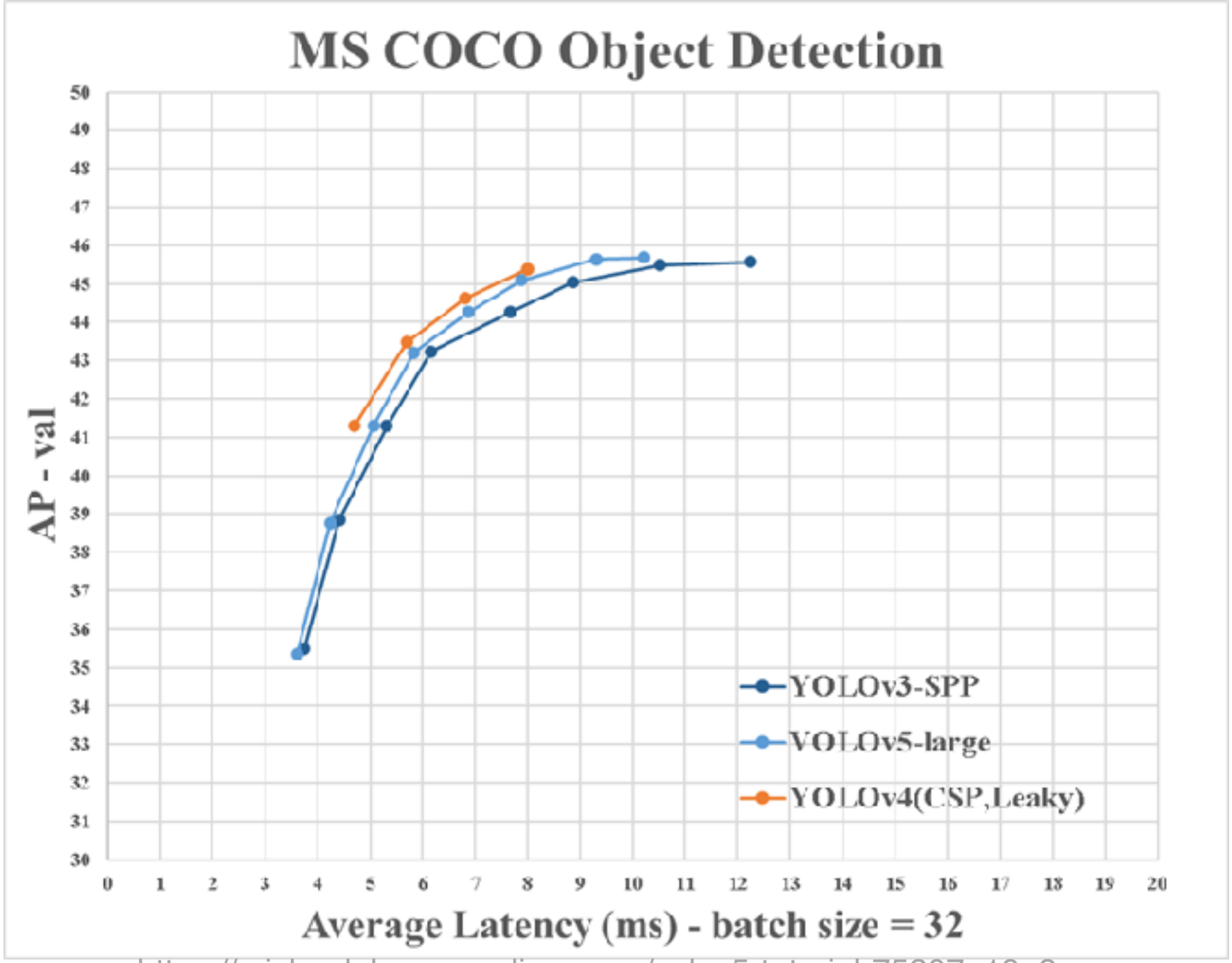
https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11

YoloV3



YOLO v3 network Architecture

https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b



https://michaelohanu.medium.com/yolov5-tutorial-75207a19a3aa

YOLOv5

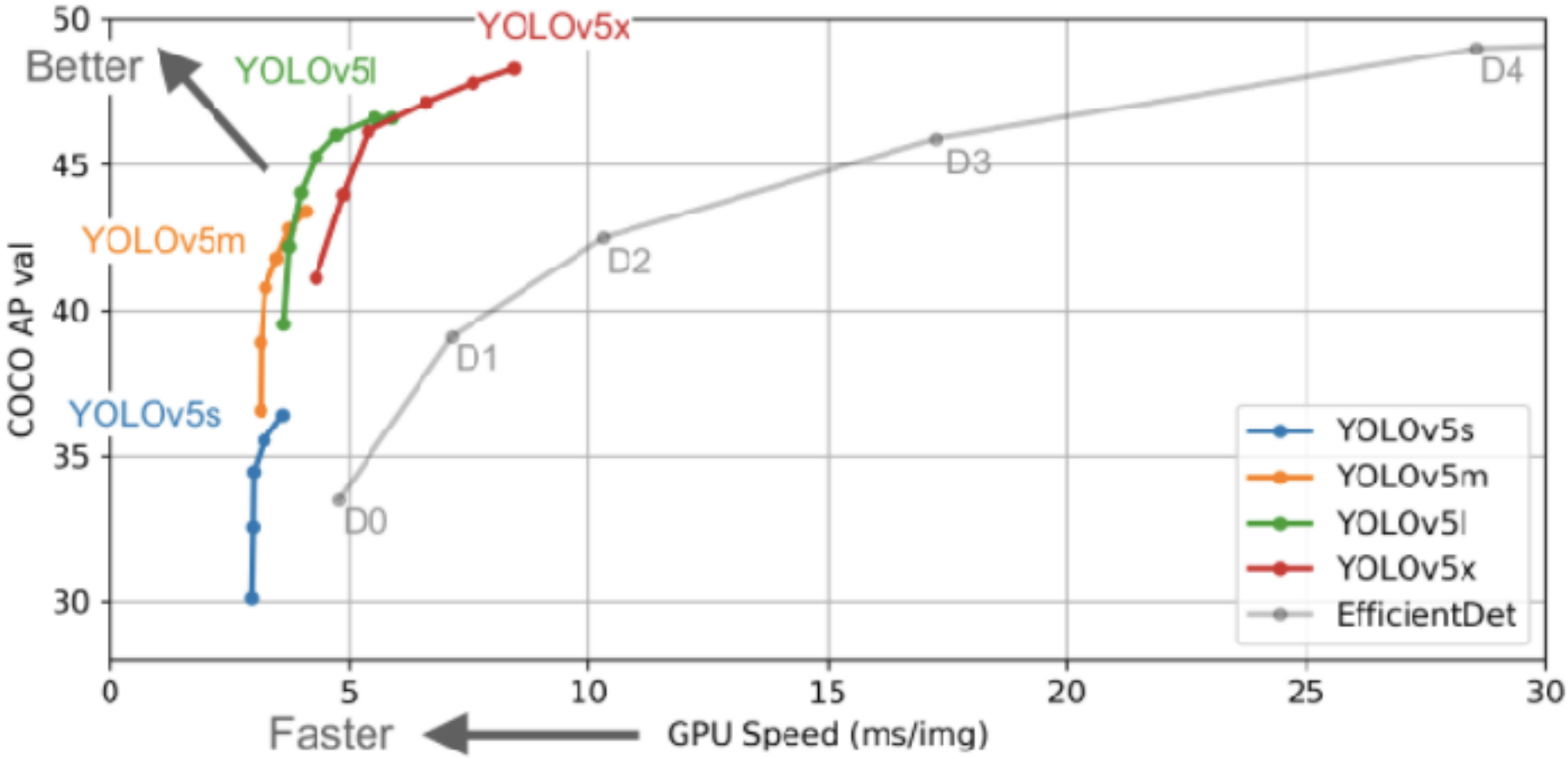
Jocher's YOLOv5 implementation differs from prior releases in a few notable ways.

First, Jocher did not (yet) publish a paper to accompany his release.

Second, Jocher implemented YOLOv5 natively in the Ultralytics PyTorch framework, which is very intuitive to use and inferences very fast where as all prior models in the YOLO family leveraged Darknet (an open source neural network framework written in C and CUDA, It is fast, easy to install, and supports CPU and GPU computation).

https://michaelohanu.medium.com/yolov5-tutorial-75207a19a3aa

Yolov5

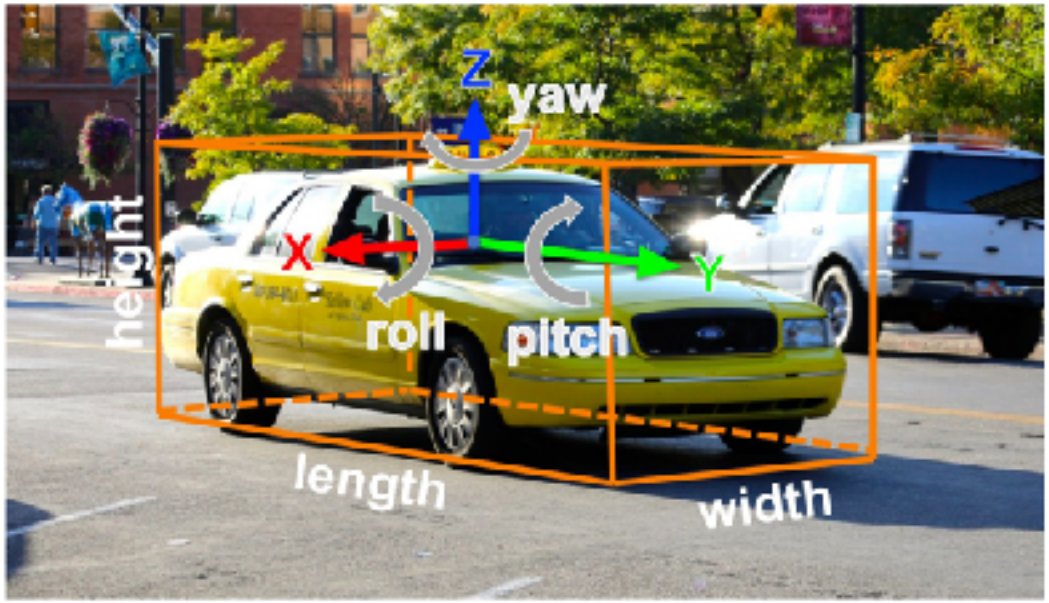


https://michaelohanu.medium.com/yolov5-tutorial-75207a19a3aa

YoloV3Keras.ipynb

3D Object Detection

3D Object Detection



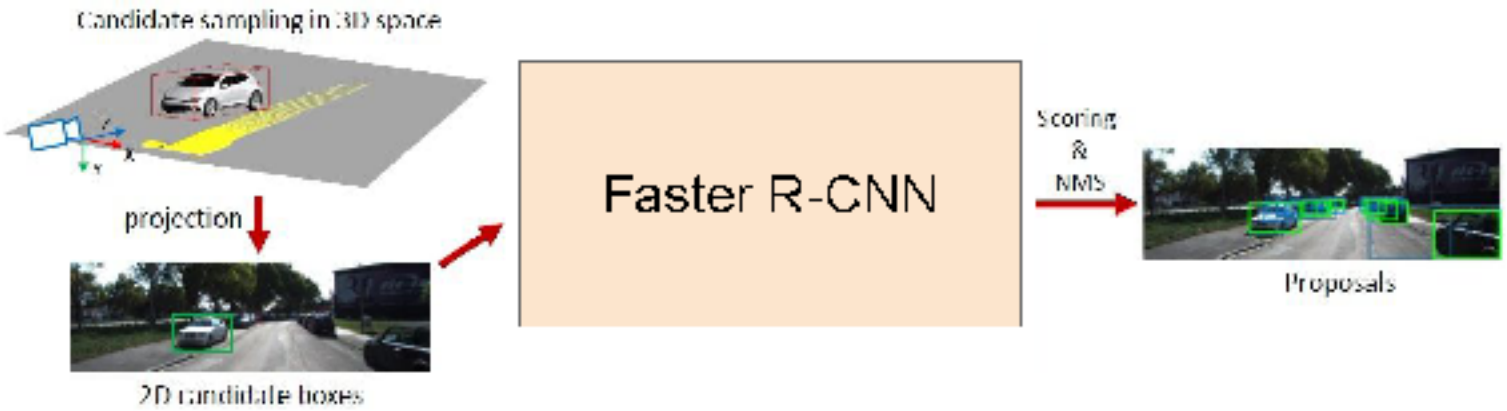
2D Object Detection:
2D bounding box
(x, y, w, h)

3D Object Detection:
3D oriented bounding box
(x, y, z, w, h, l, r, p, y)

Simplified bbox: no roll & pitch

Much harder problem than 2D object detection!

3D Object Detection: Monocular Camera

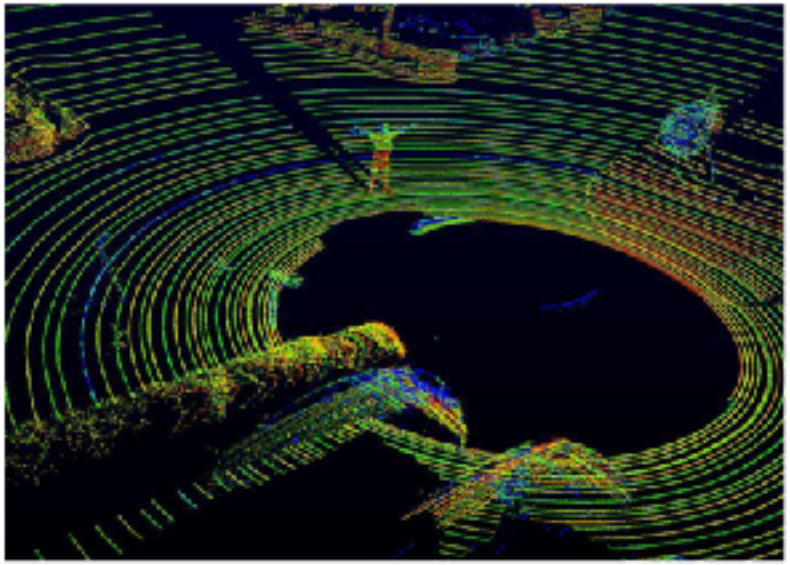


- Same idea as Faster RCNN, but proposals are in 3D
- 3D bounding box proposal, regress 3D box parameters + class score

Chen, Xiaozhi, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. "Monocular 3d object detection for autonomous driving." CVPR 2016.

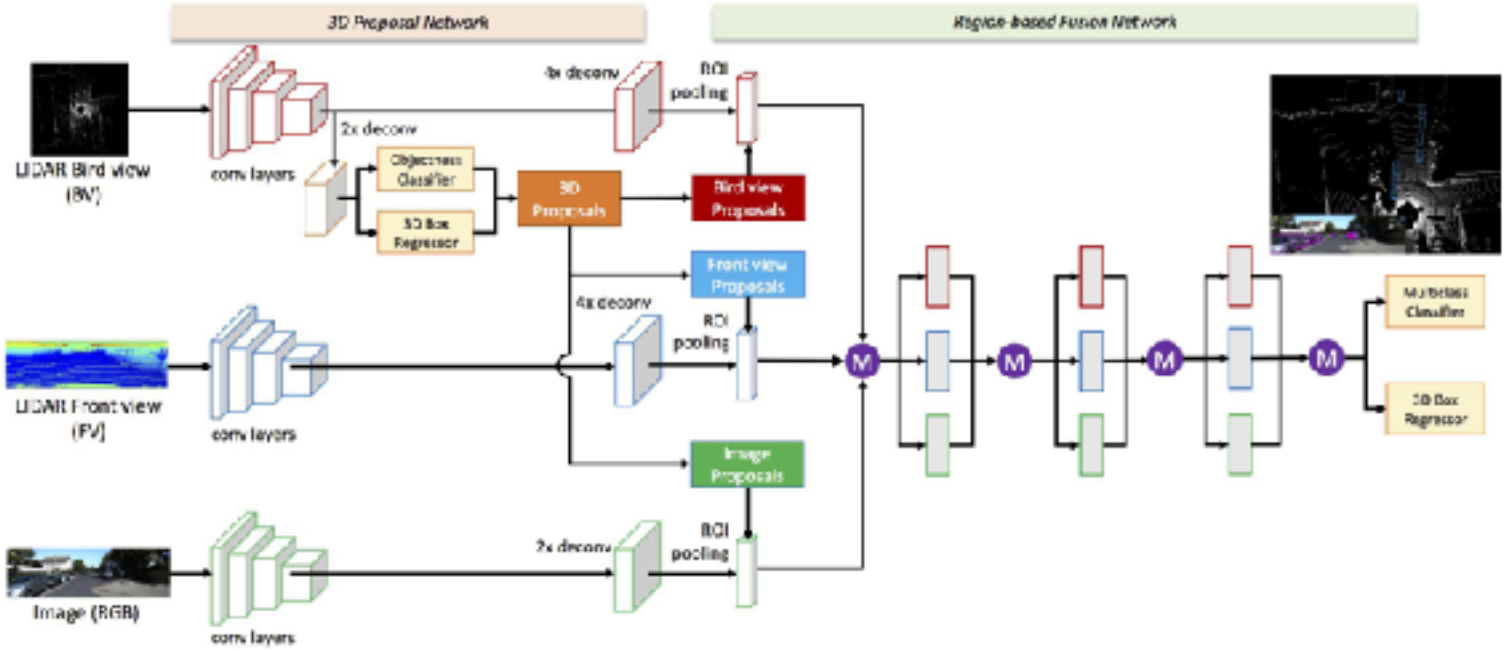
3D Object Detection: Camera + LiDAR

Velodyne (HDL-64e)



3D Point Cloud

3D Object Detection: Camera + LiDAR

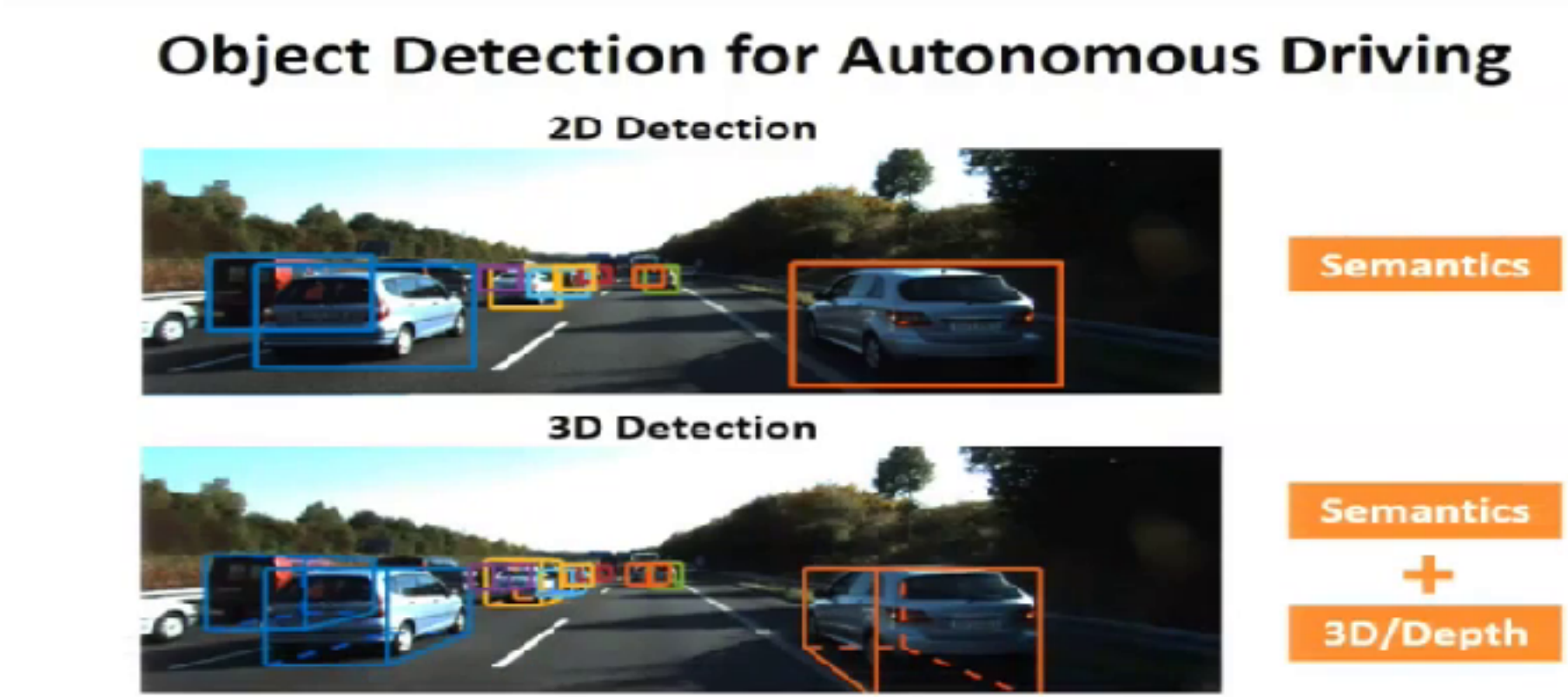


- Combine 3D proposals from multiple views & sensors
- regress 3D box parameters + class score

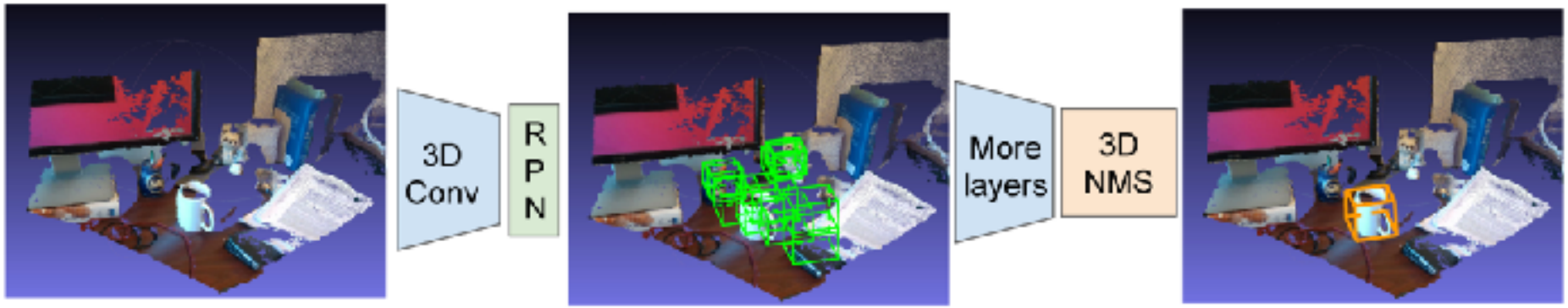
Chen, Xiaozhi, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. "Multi-view 3d object detection network for autonomous driving." *CVPR* 2017

3D Object Detection: Camera + LiDAR

3D Object Detection: Camera + LiDAR



3D Object Detection: RGB-Depth Camera



"Faster RCNN in 3D"

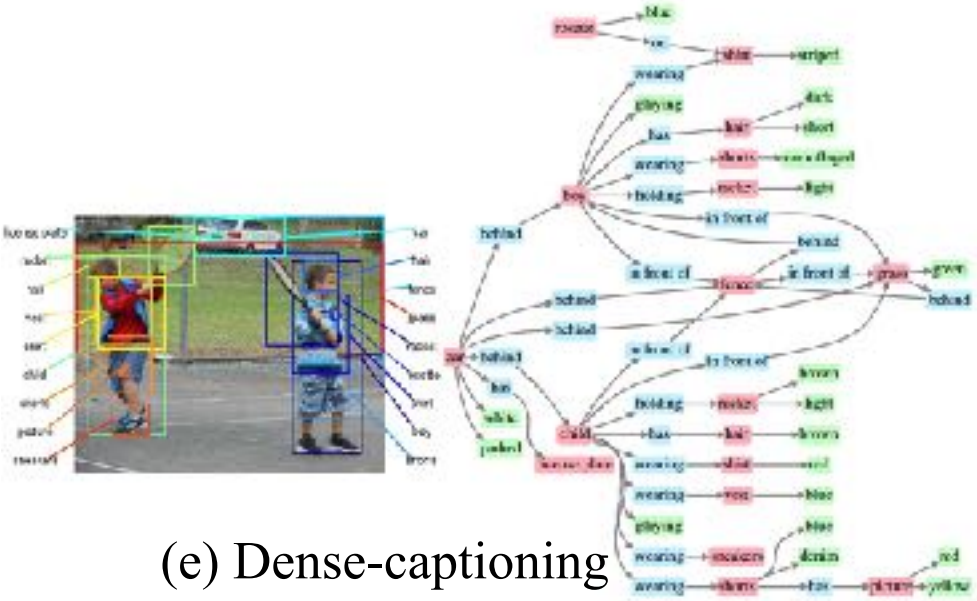
S. Song, and J. Xiao. Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images. CVPR 2016

Beyond Object Detection

A person riding a motorcycle on a dirt road

Two dogs play in the grass

(e) Auto-captioning



(e) Dense-captioning

Next…

- Graph Neural Networks/Geometry Deep Learning
  - Invited Talk: by Usman Nazir