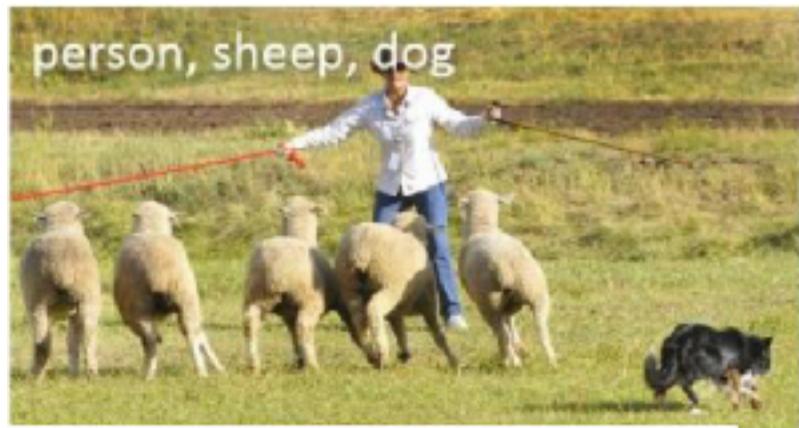
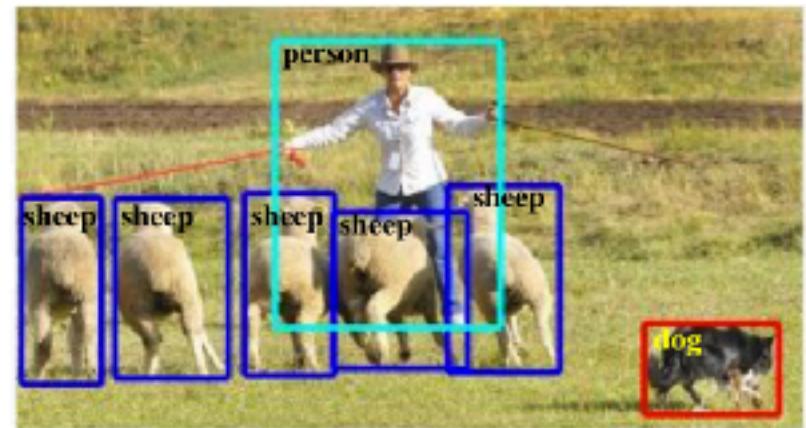


Recap: From Classification to Object Detection



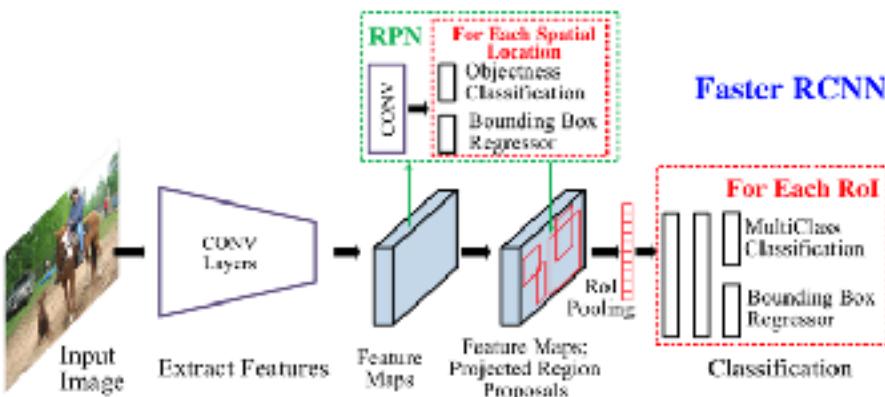
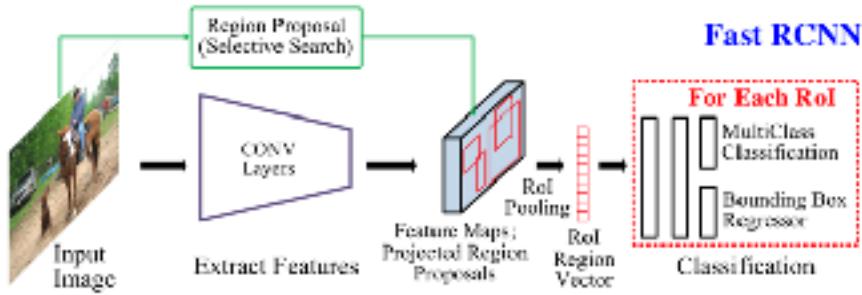
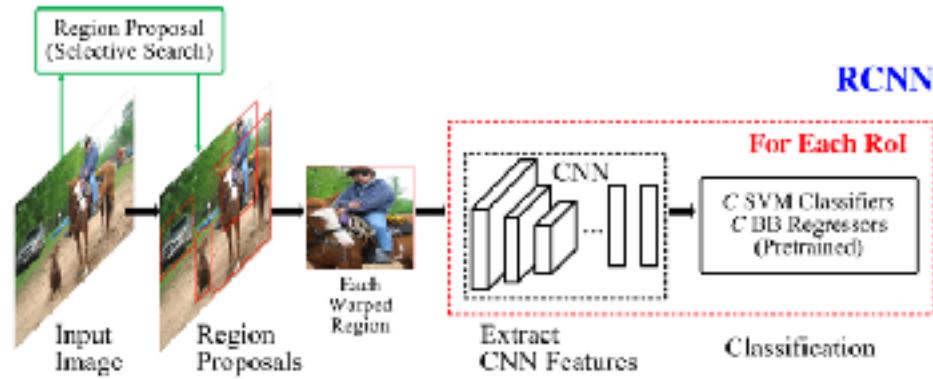
(a) Object Classification



(b) Generic Object Detection
(Bounding Box)

Class

Class
 (x,y,w,h)

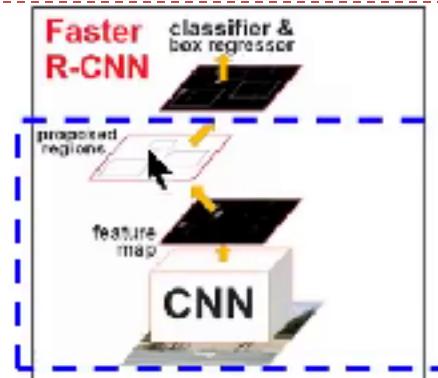


This Lecture

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

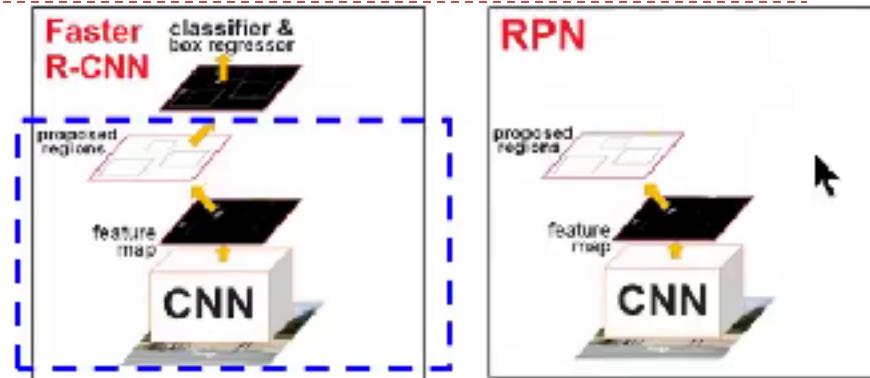
Region Proposal Network

RPN as reg. proposer



Region Proposal Network

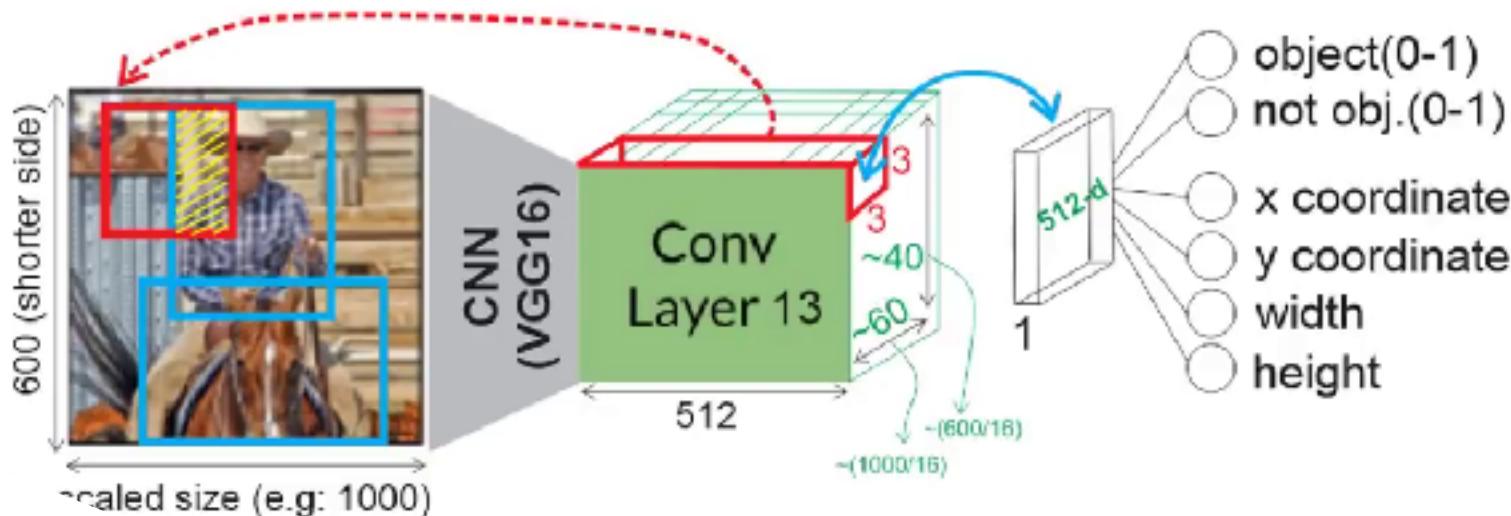
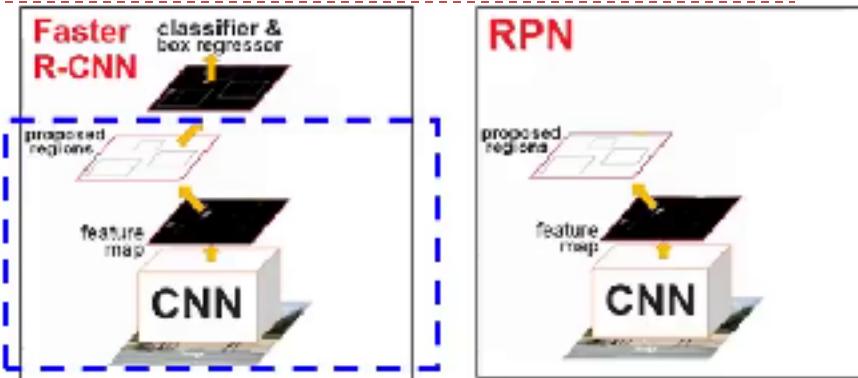
RPN as reg. proposer



Region Proposal Network

RPN as reg. proposer

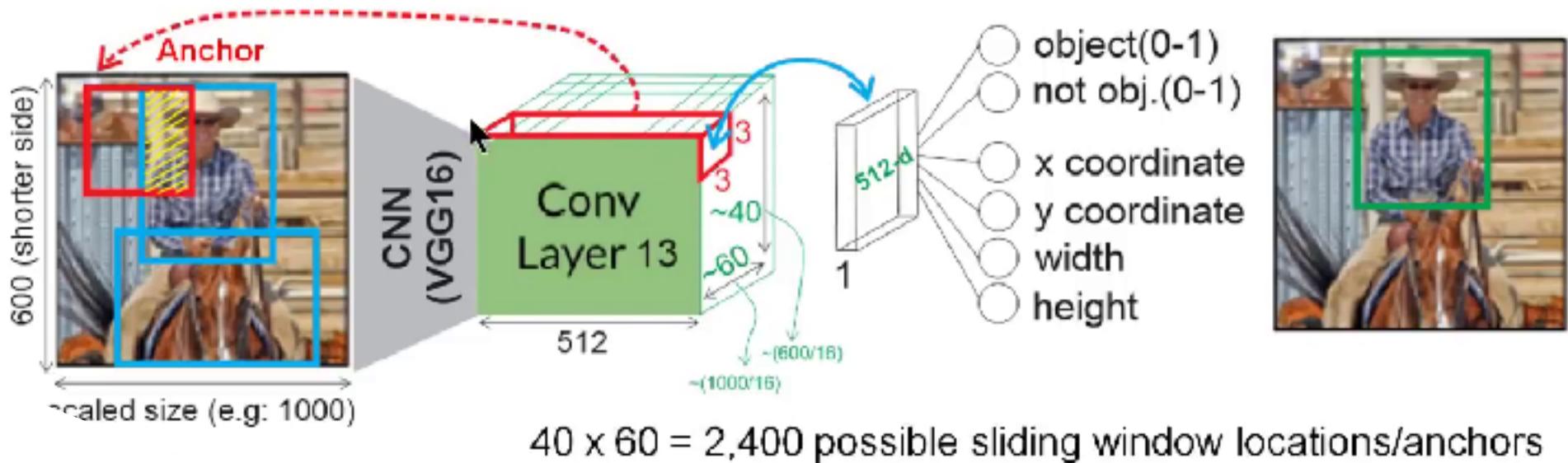
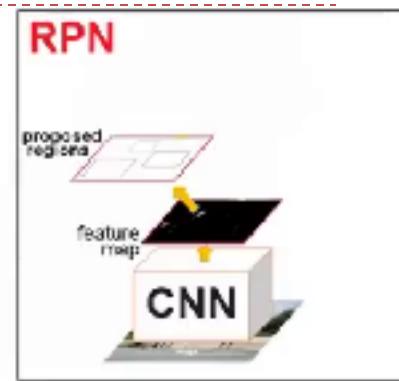
$$\text{IoU} = \frac{A \cap Gt}{A \cup Gt} \quad \left\{ \begin{array}{l} > 0.7 = \text{object} \\ < 0.3 = \text{not object} \end{array} \right.$$



Region Proposal Network

RPN as reg. proposer

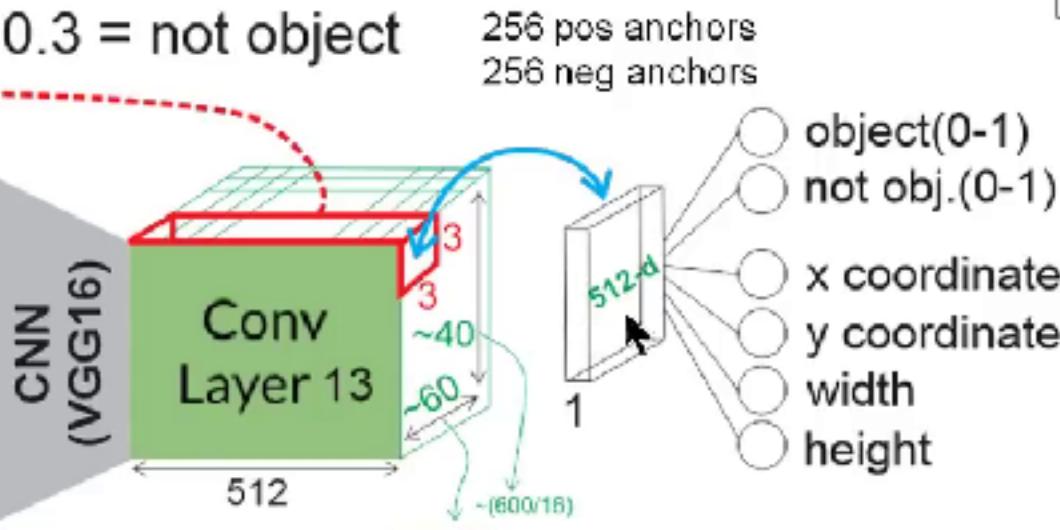
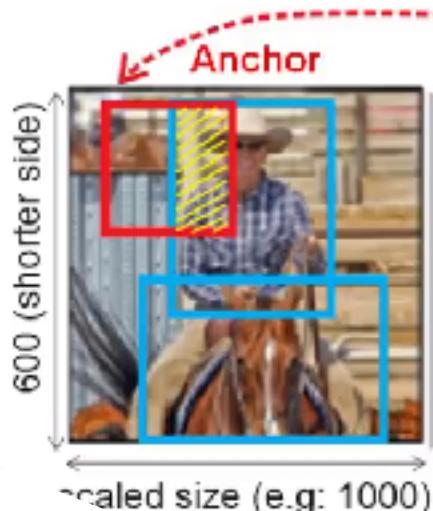
$$\text{IoU} = \frac{A \cap Gt}{A \cup Gt} \quad \left\{ \begin{array}{l} > 0.7 = \text{object} \\ < 0.3 = \text{not object} \end{array} \right.$$



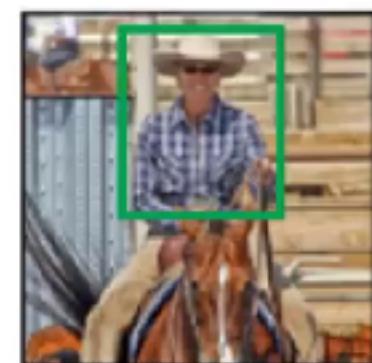
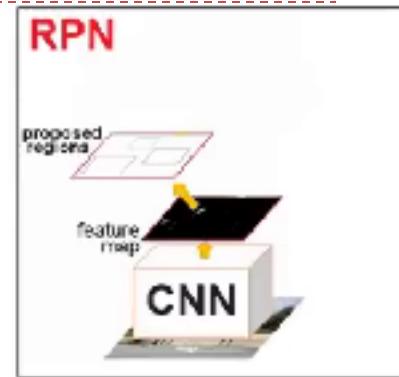
Region Proposal Network

RPN as reg. proposer

$$\text{IoU} = \frac{A \cap Gt}{A \cup Gt} \quad \left\{ \begin{array}{l} > 0.7 = \text{object} \\ < 0.3 = \text{not object} \end{array} \right.$$

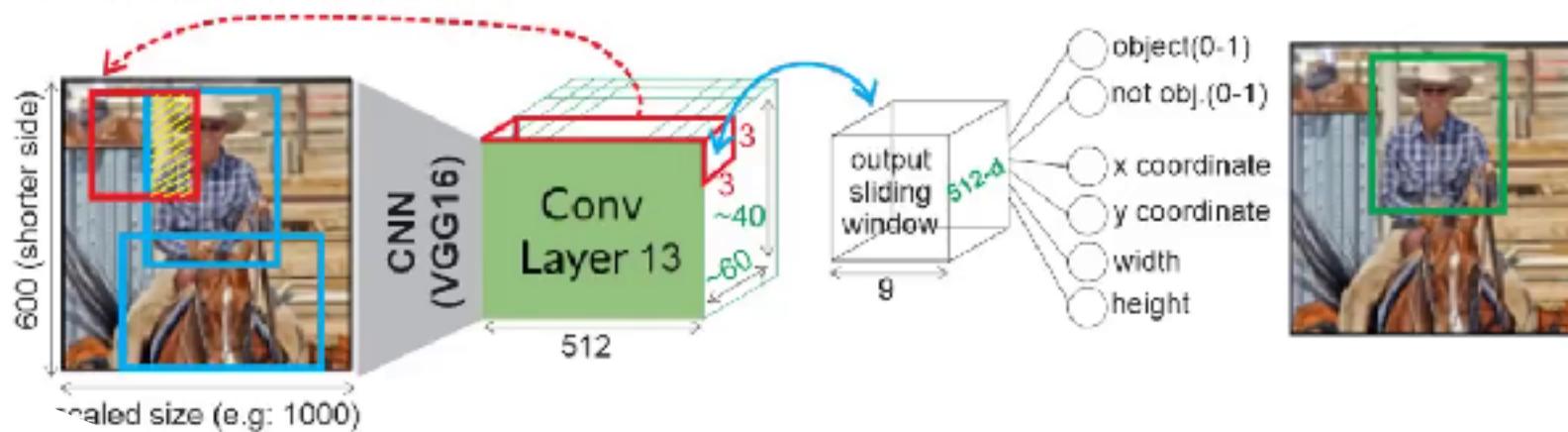
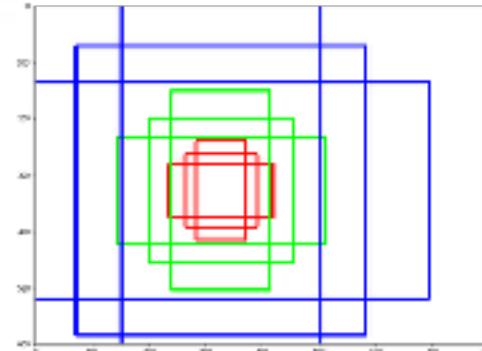
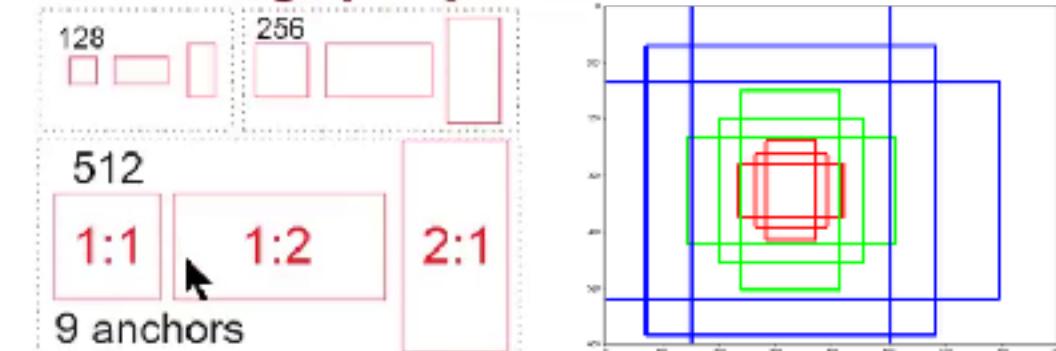


$40 \times 60 = 2,400$ possible sliding window locations/anchors



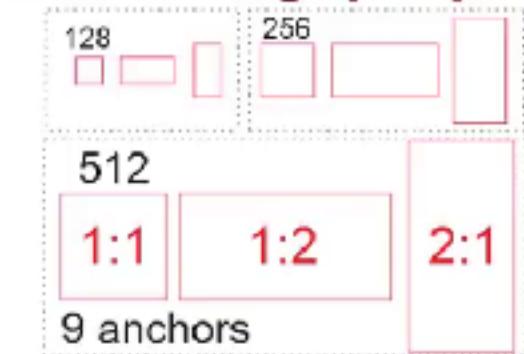
Region Proposal Network

RPN as reg. proposer



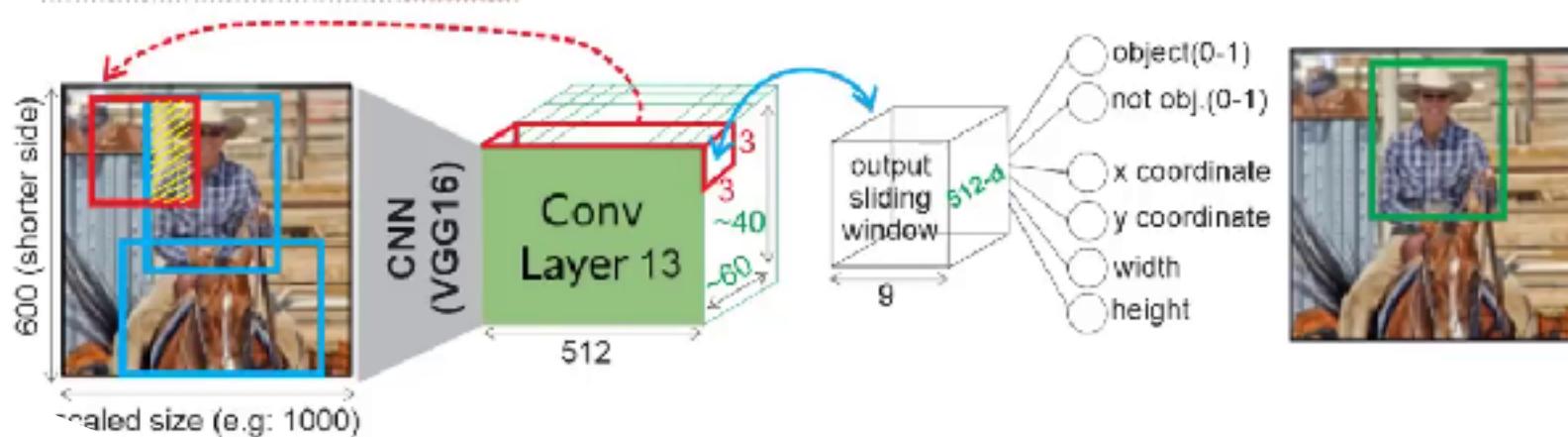
Region Proposal Network

RPN as reg. proposer



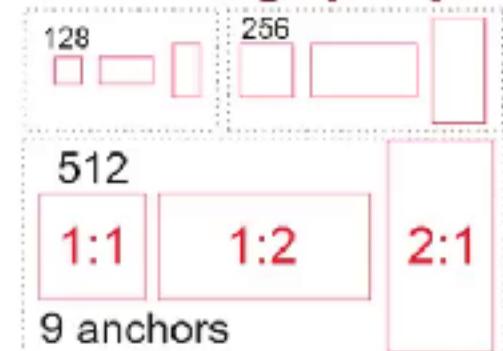
$$40 \times 60 \times 9 = 21,600 \text{ anchors}$$

1. Ignore cross-boundary anchors $\rightarrow \sim 6,000$



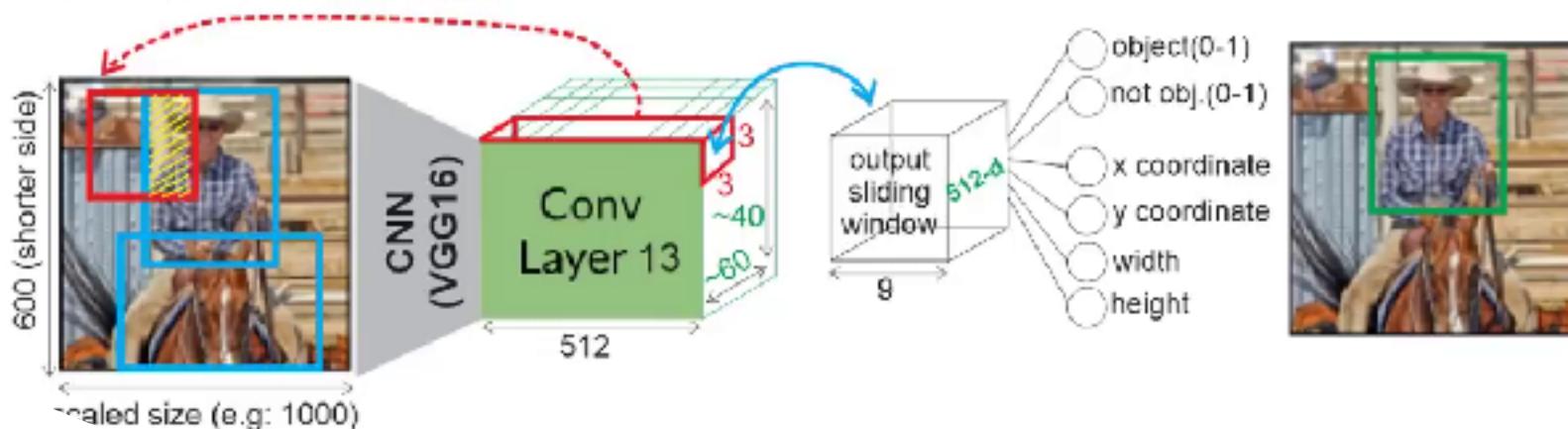
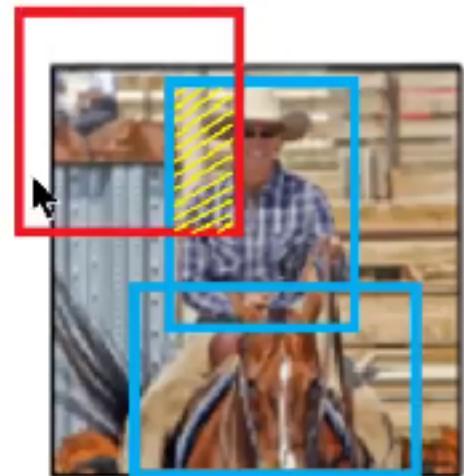
Region Proposal Network

RPN as reg. proposer



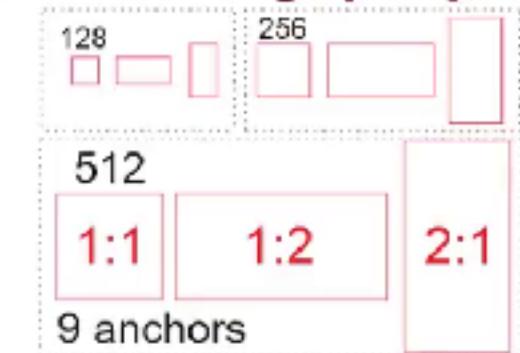
$$40 \times 60 \times 9 = 21,600 \text{ anchors}$$

1. Ignore cross-boundary anchors $\rightarrow \sim 6,000$



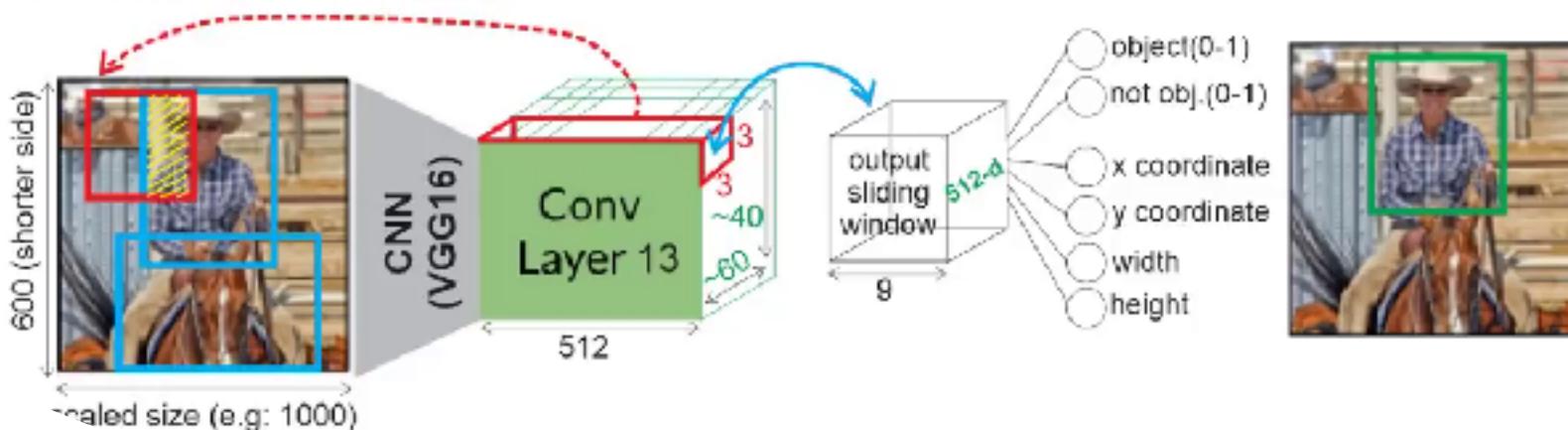
Region Proposal Network

RPN as reg. proposer



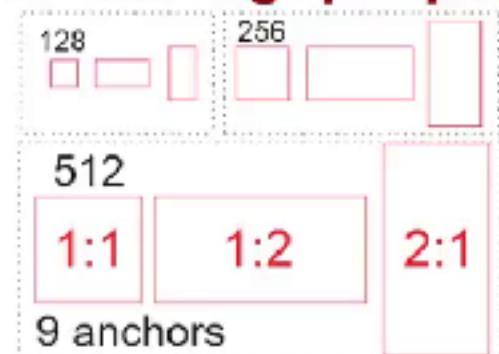
$$40 \times 60 \times 9 = 21,600 \text{ anchors}$$

1. Ignore cross-boundary anchors $\rightarrow \sim 6,000$
2. Apply NMS (Non-Max Suppression) $\rightarrow \sim 2,000$



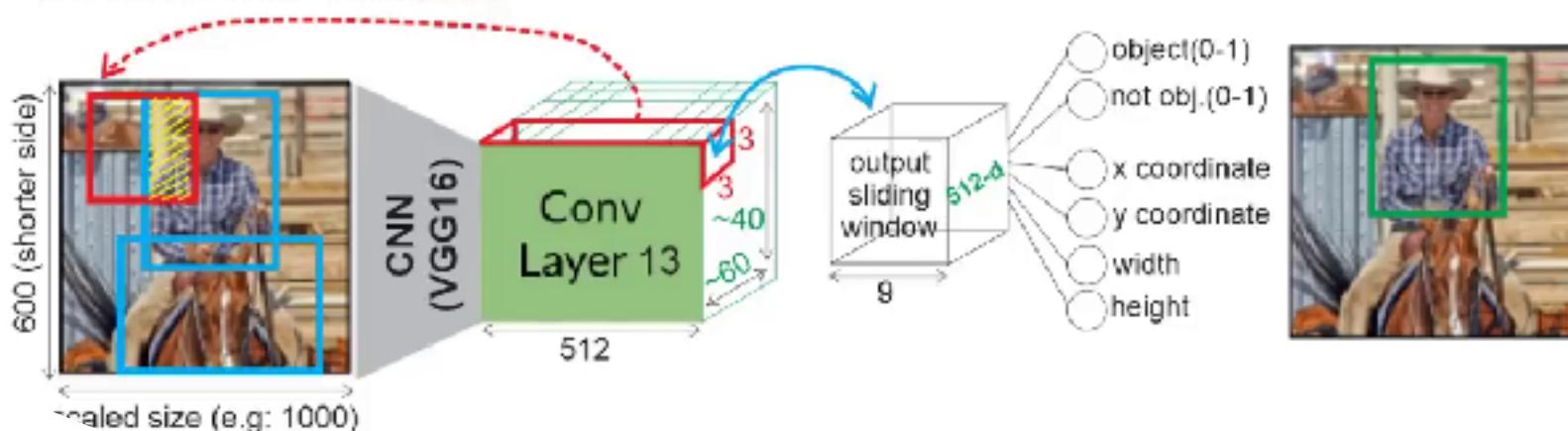
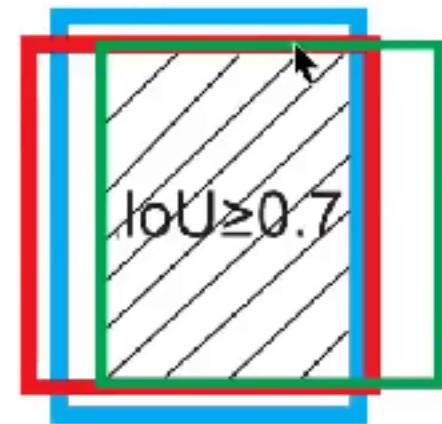
Region Proposal Network

RPN as reg. proposer



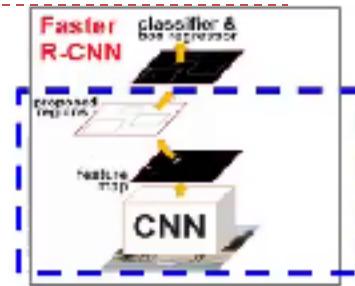
$$40 \times 60 \times 9 = 21,600 \text{ anchors}$$

1. Ignore cross-boundary anchors $\rightarrow \sim 6,000$
2. Apply NMS (Non-Max Suppression) $\rightarrow \sim 2,000$



Region Proposal Network

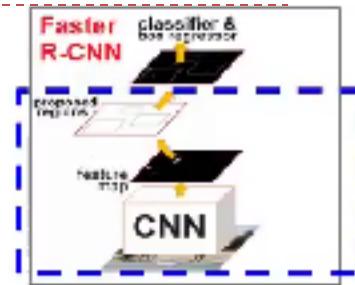
RPN loss function



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

Region Proposal Network

RPN loss function



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

object/not object classifier box regressor

Region Proposal Network

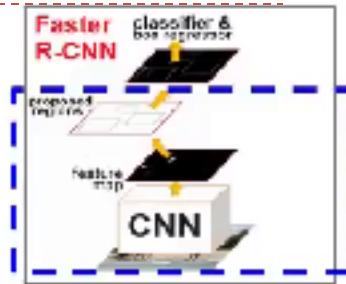
RPN loss function

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

p_i : predicted probability

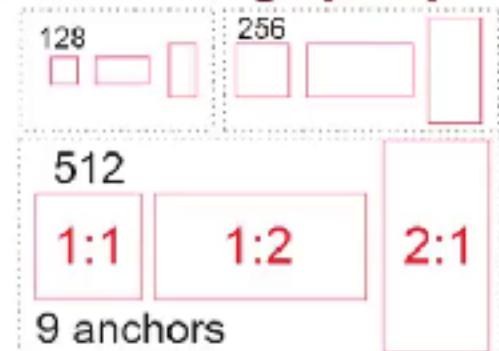
p_i^* { 1 for pos anchor
0 for neg anchor

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



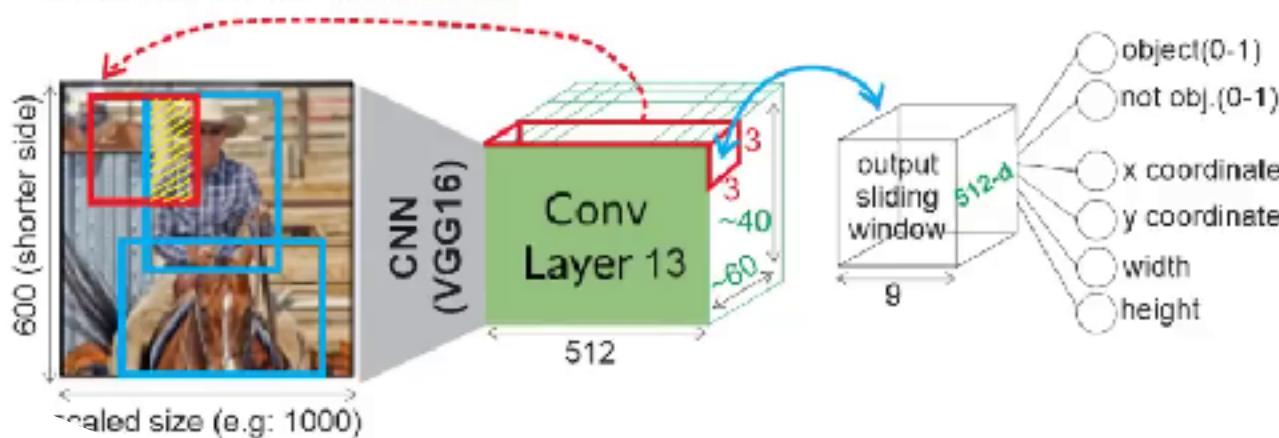
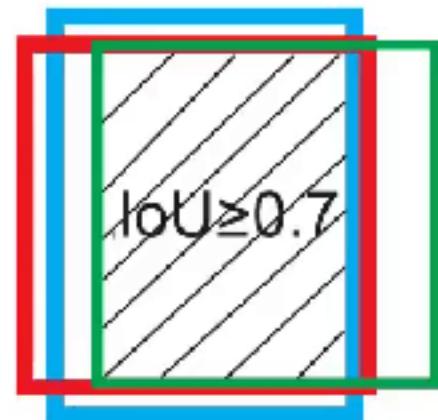
Region Proposal Network

RPN as reg. proposer



$$40 \times 60 \times 9 = 21,600 \text{ anchors}$$

1. Ignore cross-boundary anchors $\rightarrow \sim 6,000$
2. Apply NMS (Non-Max Suppression) $\rightarrow \sim 2,000$



Region Proposal Network

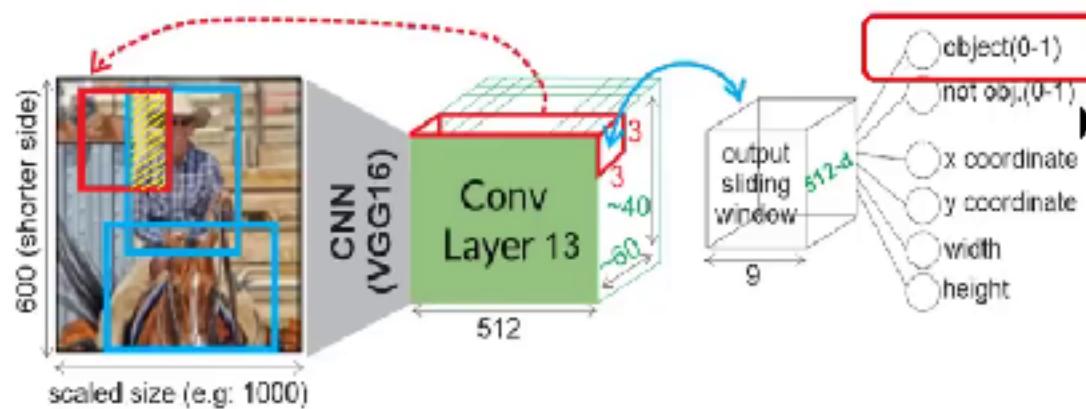
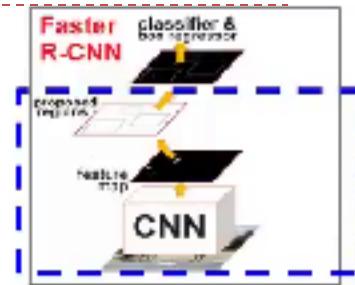
RPN loss function

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

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

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

λ = constant value

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

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

Region Proposal Network

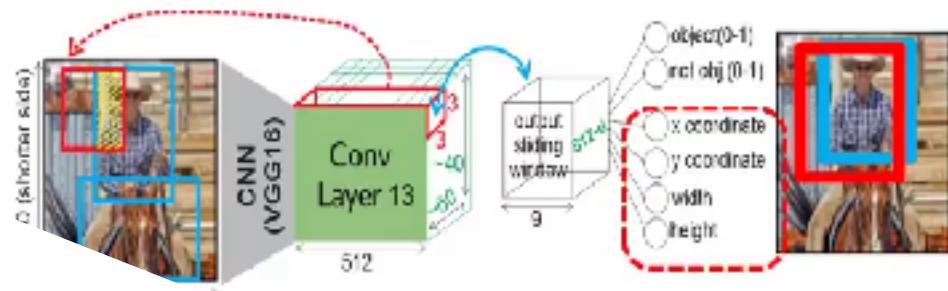
RPN loss function

$$L(t_i) =$$

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

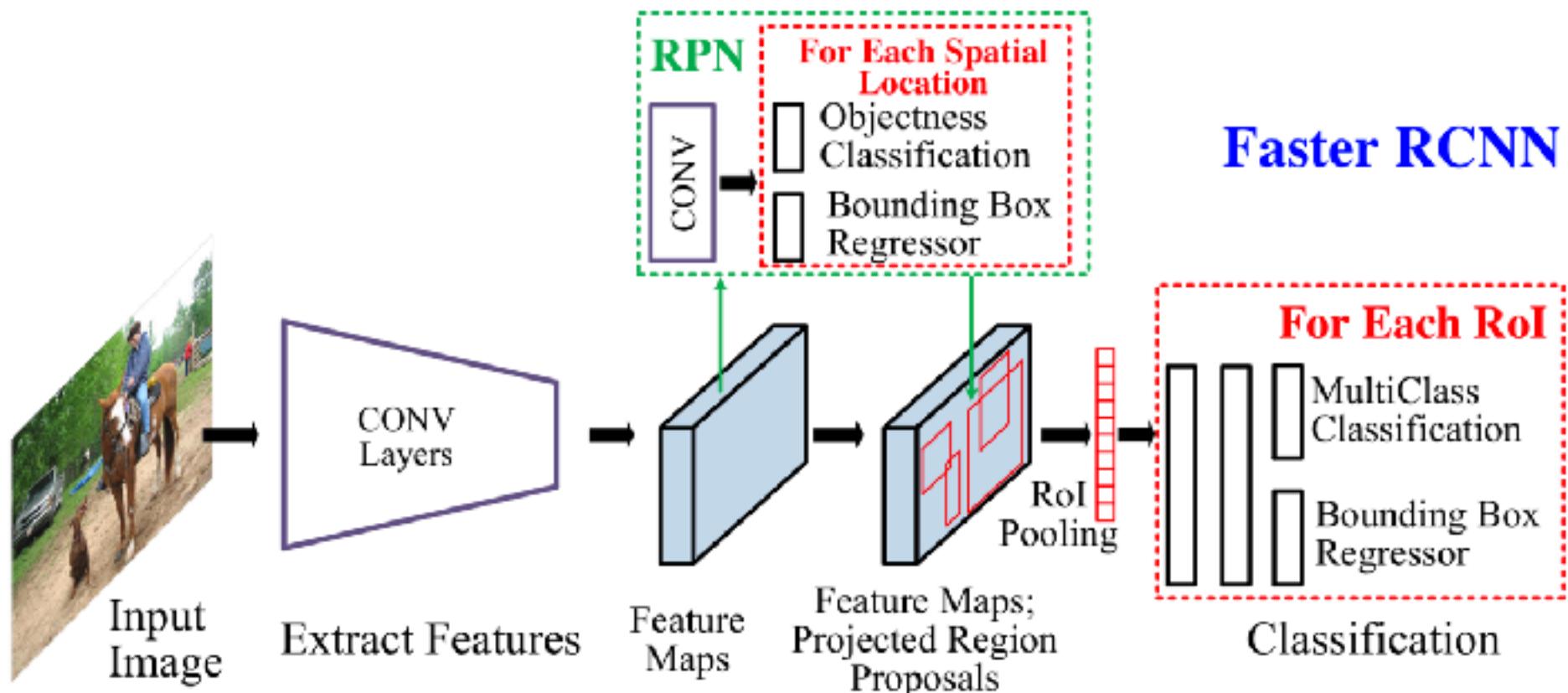
$$L_{reg} = \text{smooth}_{L_1}(\mathbf{t}_i - \mathbf{\hat{t}}_i) \quad \text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

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

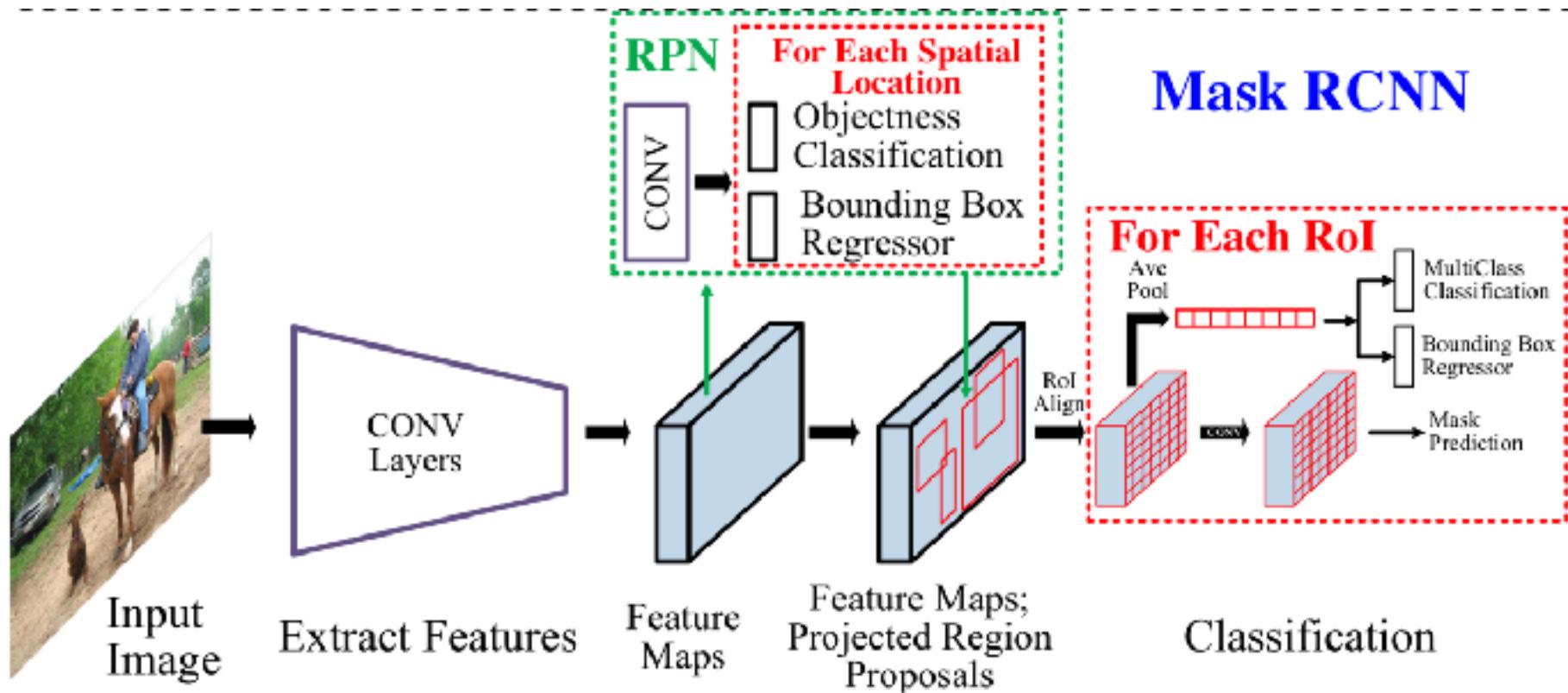


$$\begin{aligned} t_x &= (x - x_a)/w_a, & t_y &= (y - y_a)/h_a, \\ t_w &= \log(w/w_a), & t_h &= \log(h/h_a), \\ t_x^* &= (x^* - x_a)/w_a, & t_y^* &= (y^* - y_a)/h_a, \\ t_w^* &= \log(w^*/w_a), & t_h^* &= \log(h^*/h_a), \end{aligned}$$

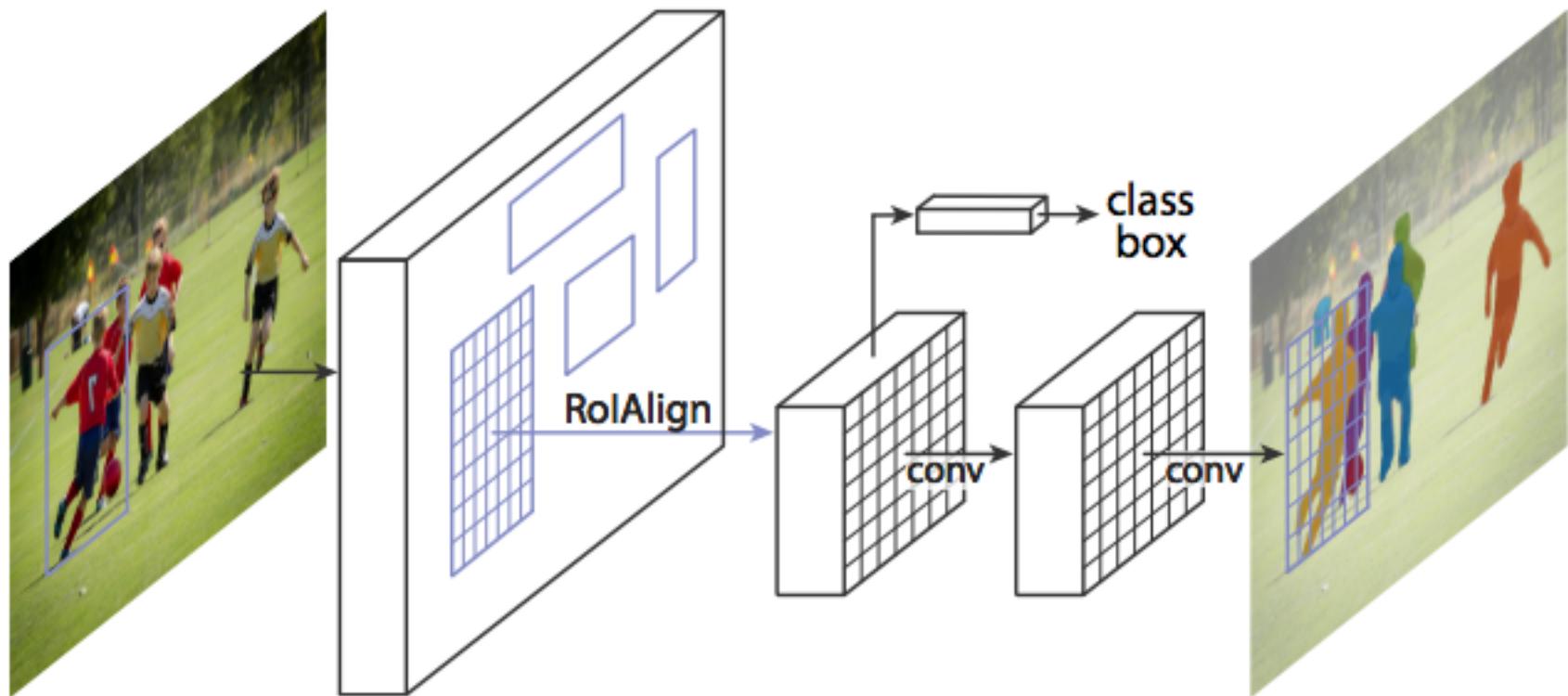
Faster RCNN



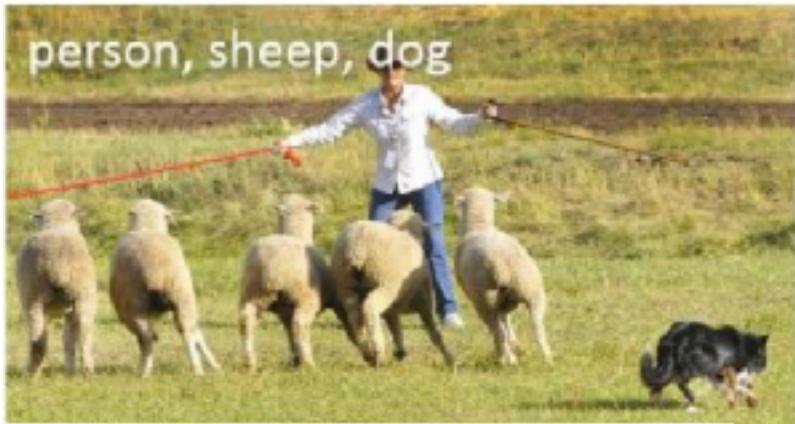
Mask RCNN



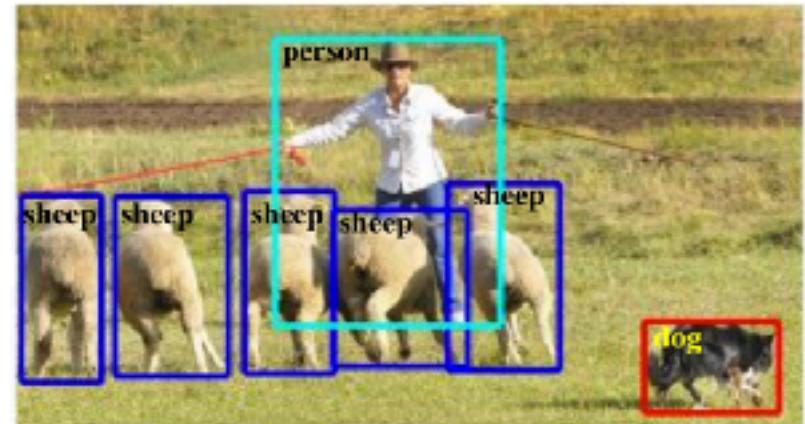
Mask RCNN: Mask Prediction



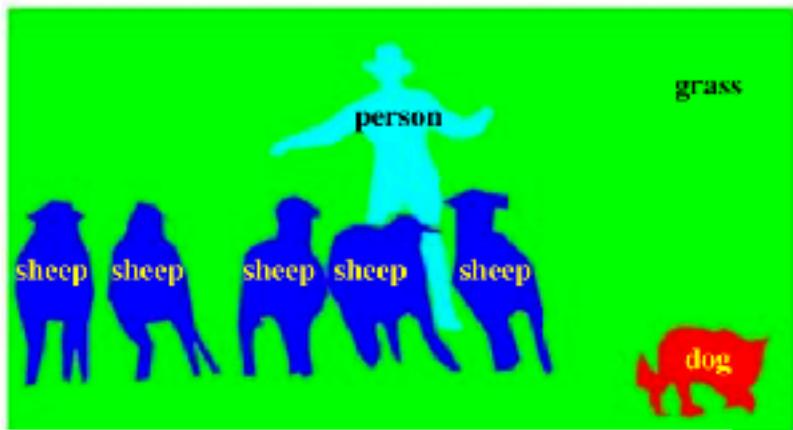
Mask RCNN



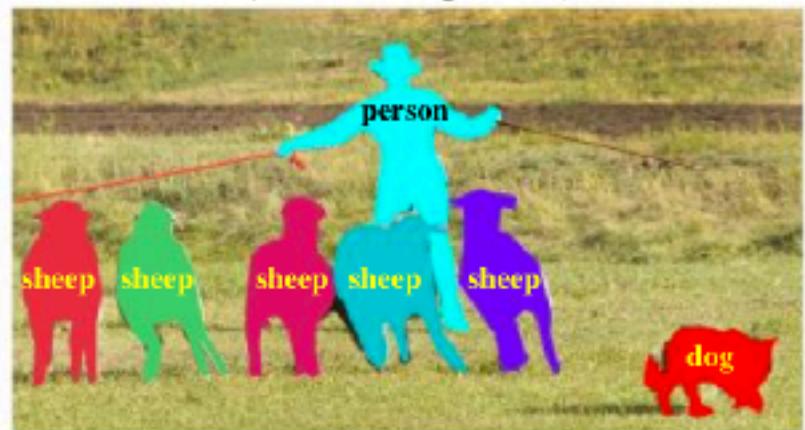
(a) Object Classification



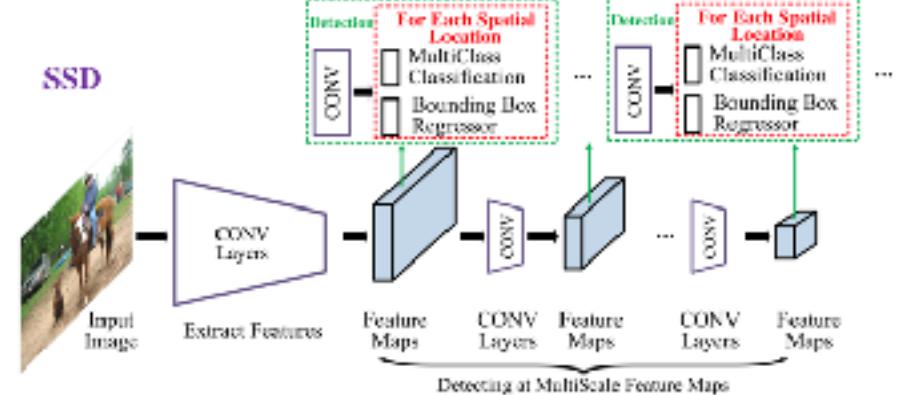
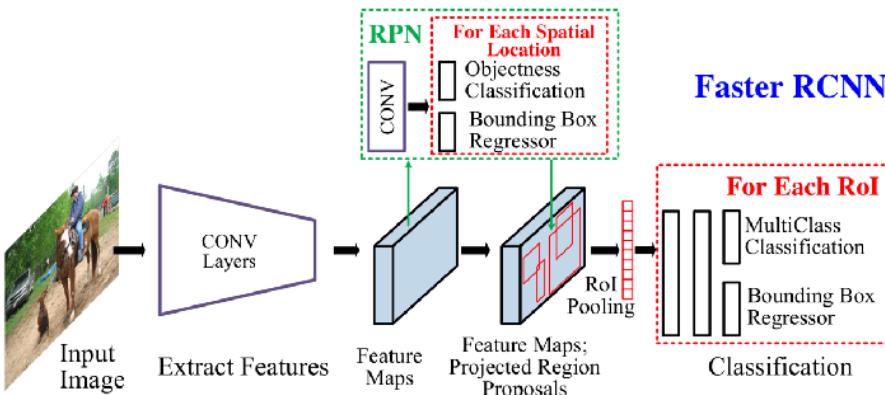
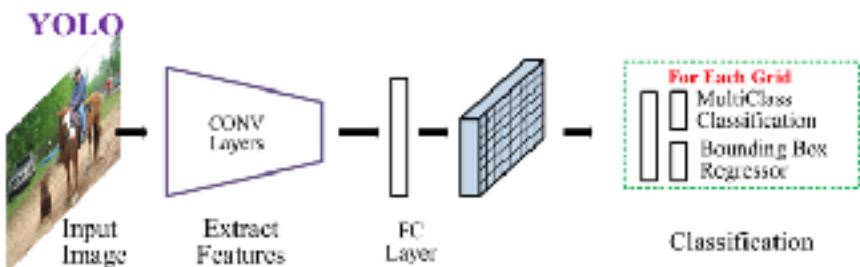
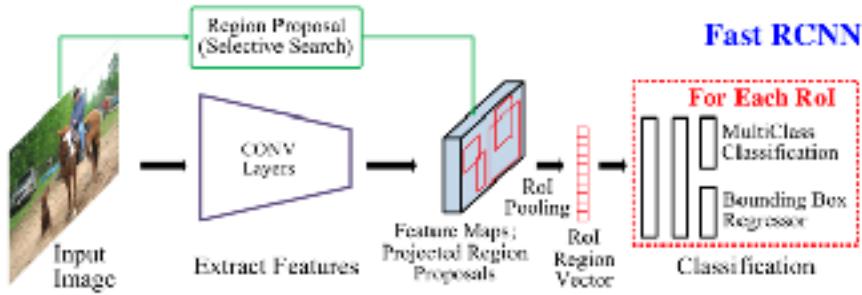
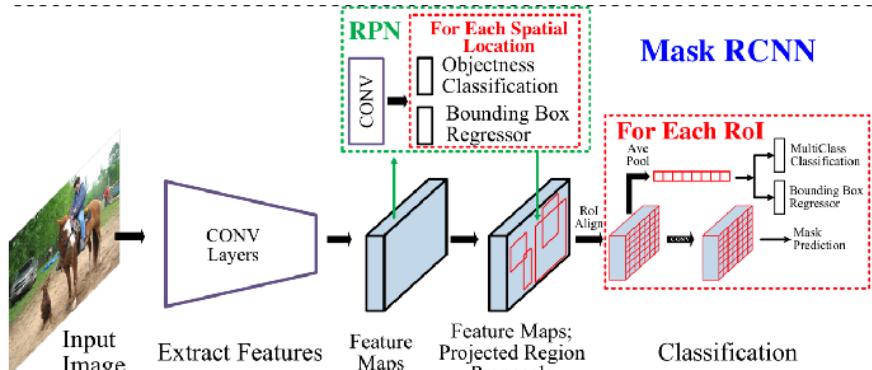
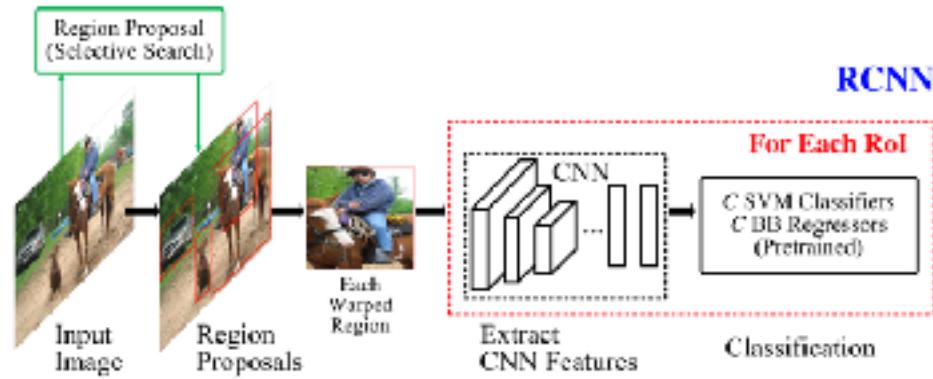
(b) Generic Object Detection
(Bounding Box)



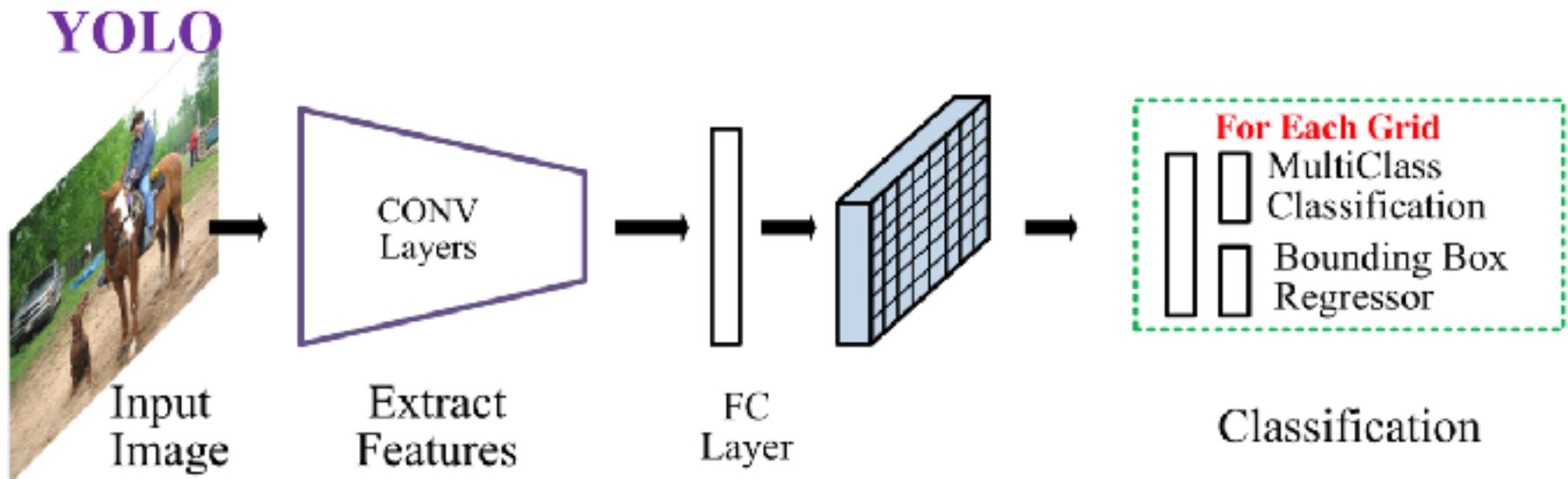
(c) Semantic Segmentation



(d) Object Instance Segmentation

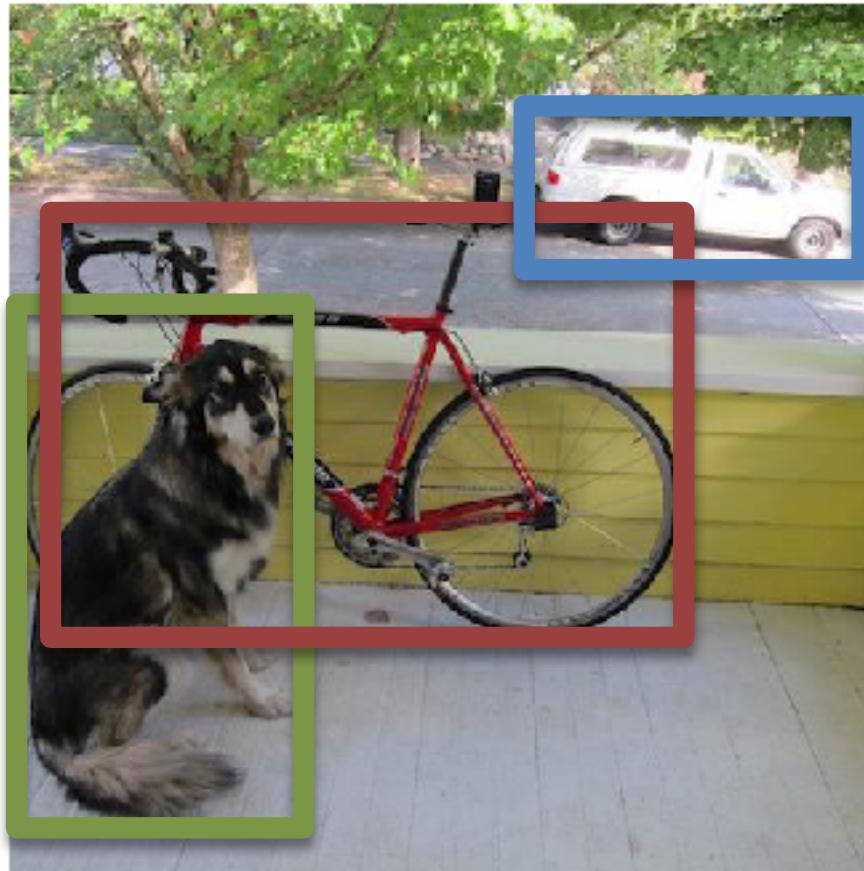


YOLO: You Only Look Once



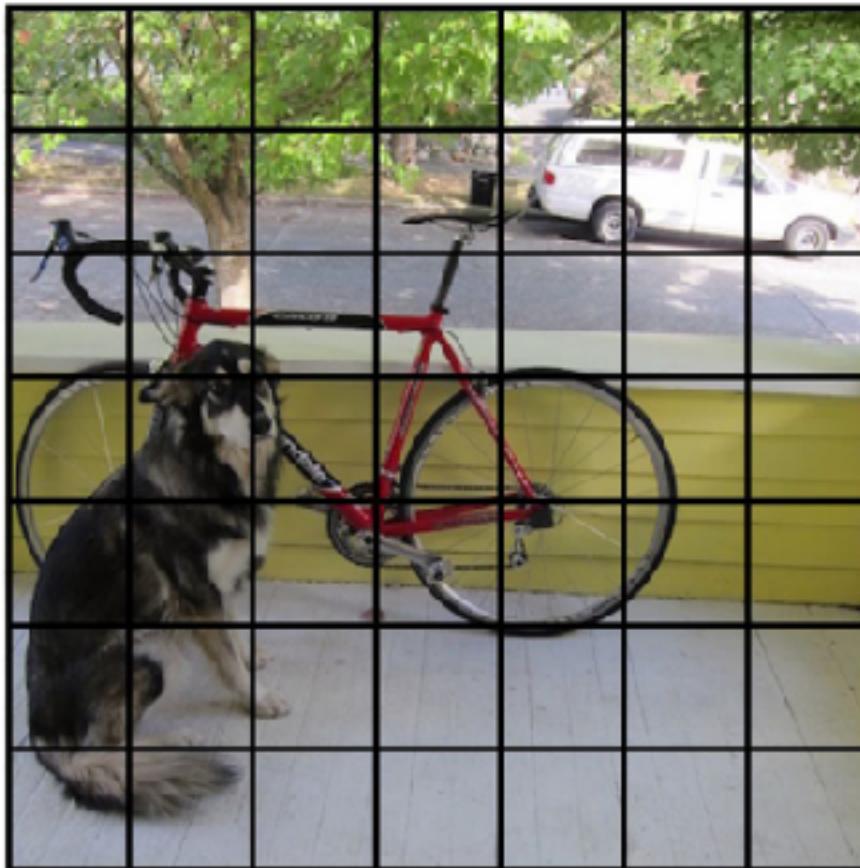
YOLO: You Only Look Once

▶ Detection Procedure



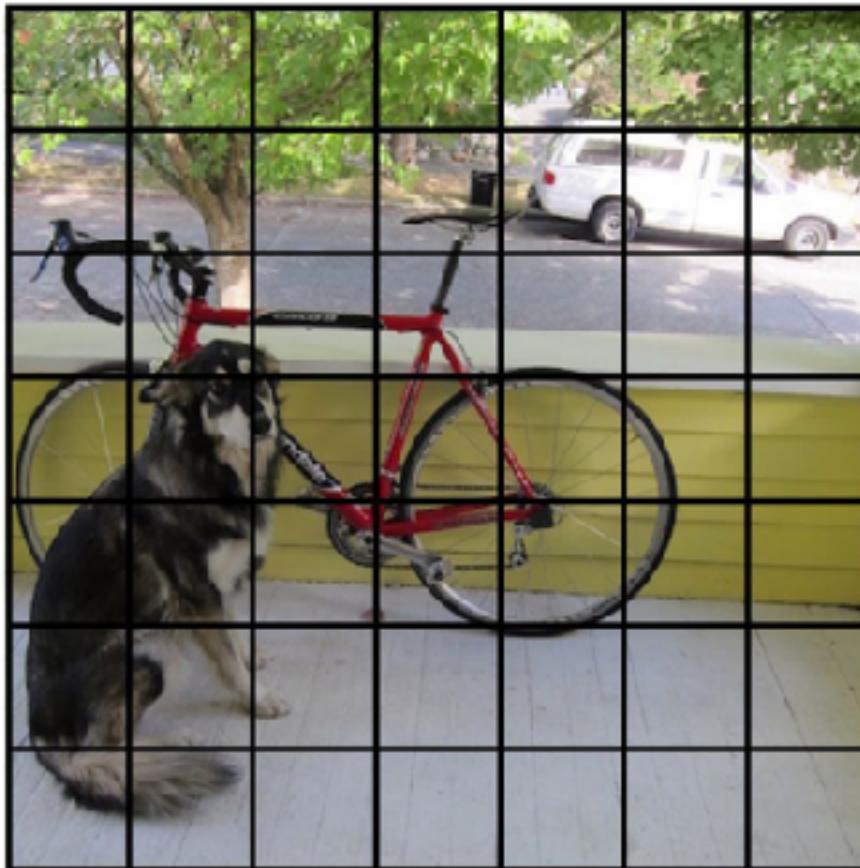
YOLO: You Only Look Once

We split the image into an $S \times S$ grid



YOLO: You Only Look Once

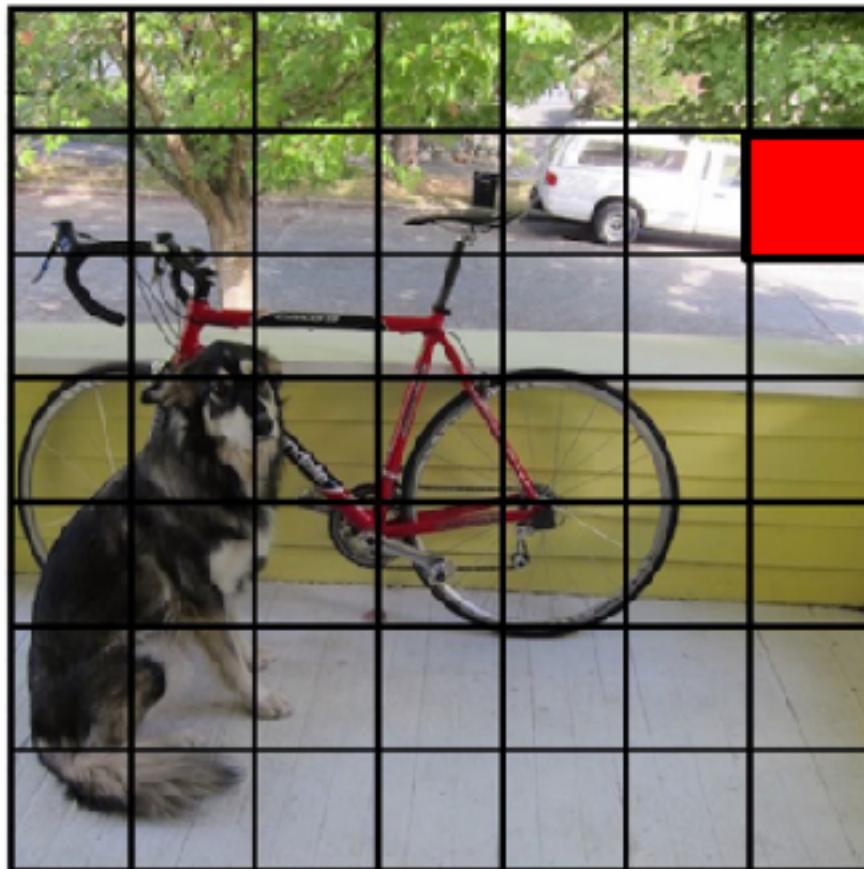
We split the image into an $S \times S$ grid



7*7 grid

YOLO: You Only Look Once

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



YOLO: You Only Look Once

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



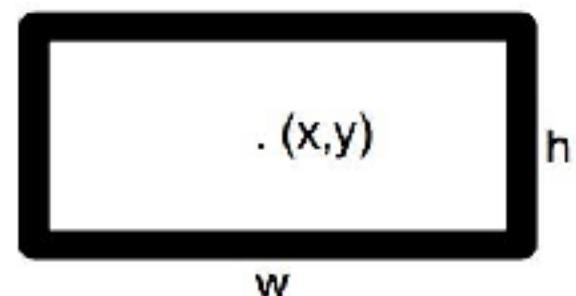
YOLO: You Only Look Once

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

$$B = 2$$



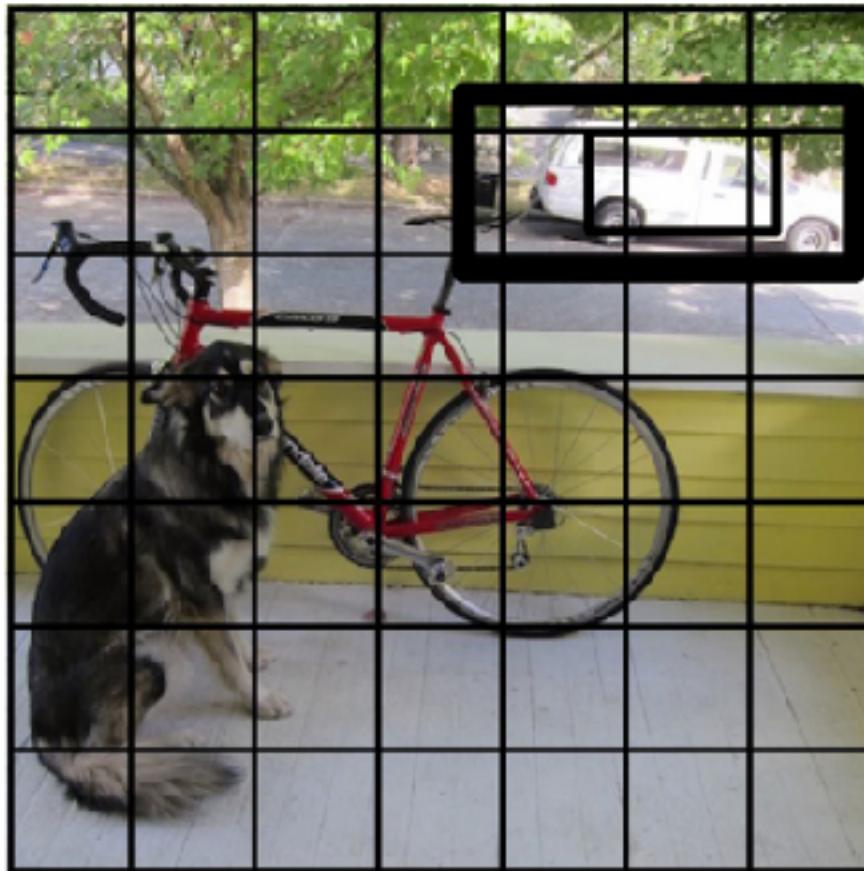
each box predict:



$P(\text{Object})$: probability that
the box contains an object

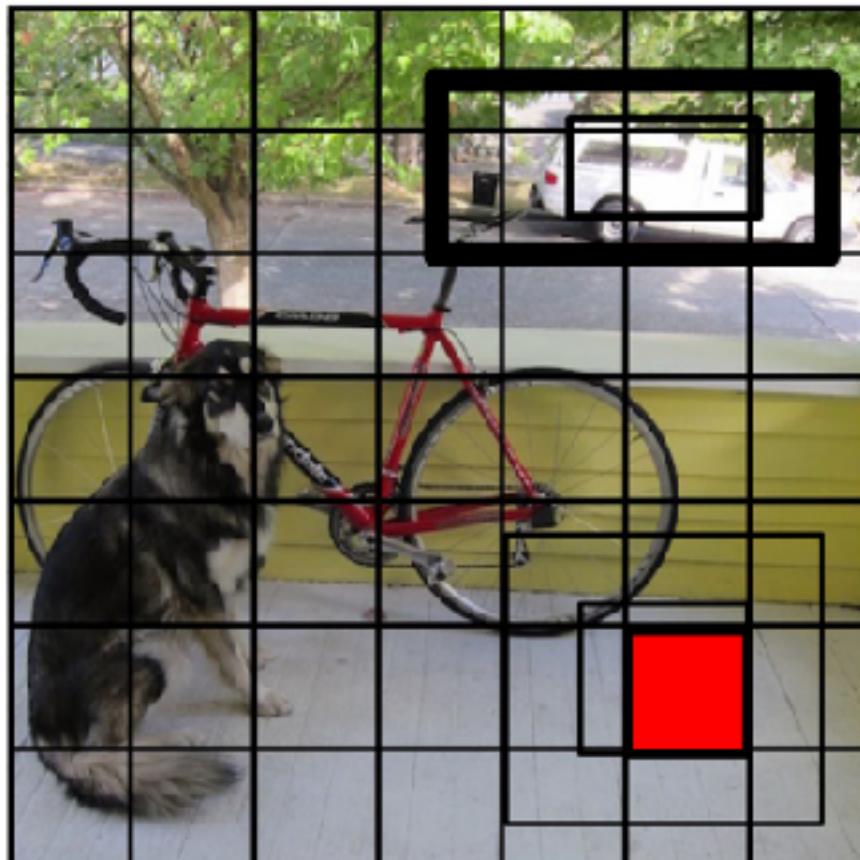
YOLO: You Only Look Once

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



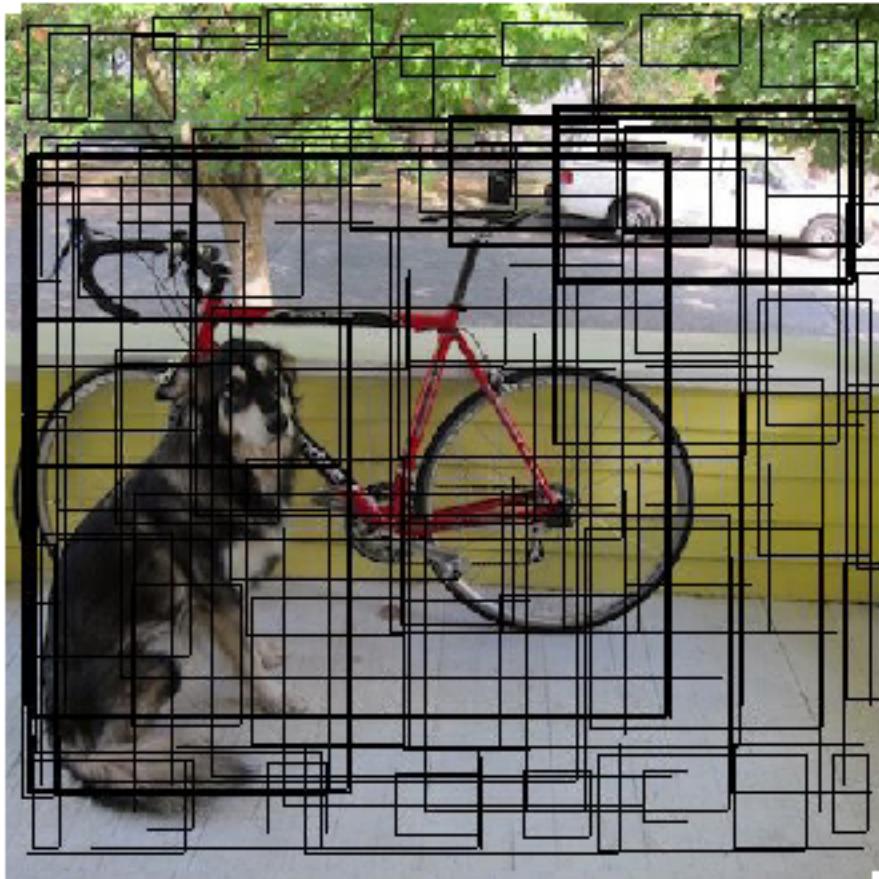
YOLO: You Only Look Once

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



YOLO: You Only Look Once

Each cell predicts boxes and confidences: $P(\text{Object})$



YOLO: You Only Look Once

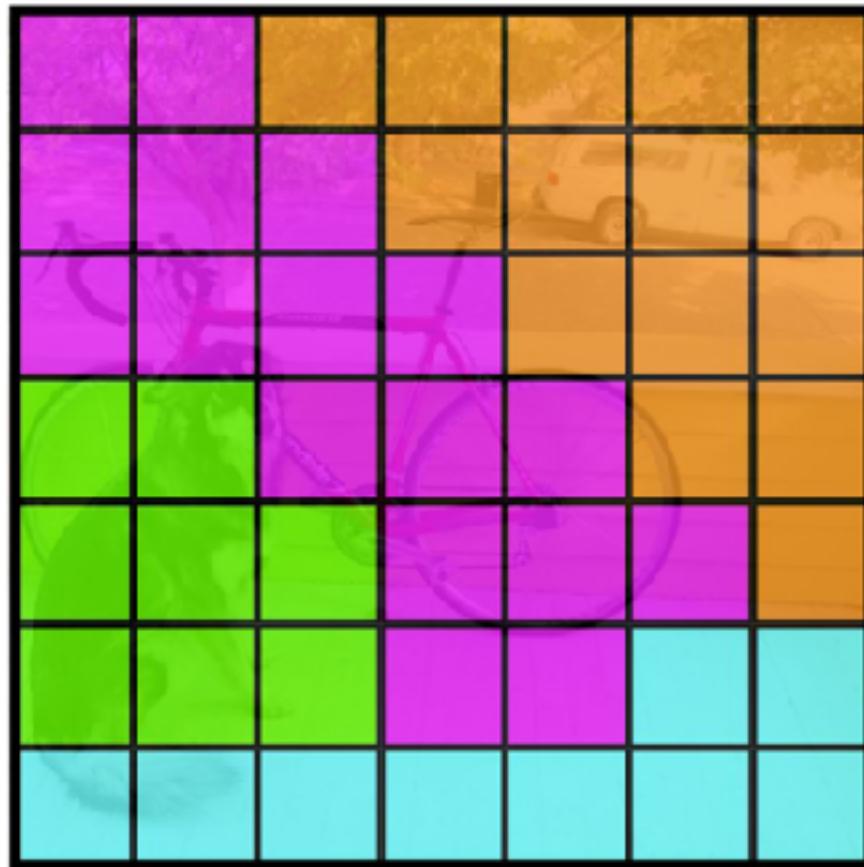
Each cell also predicts a class probability.

Bicycle

Car

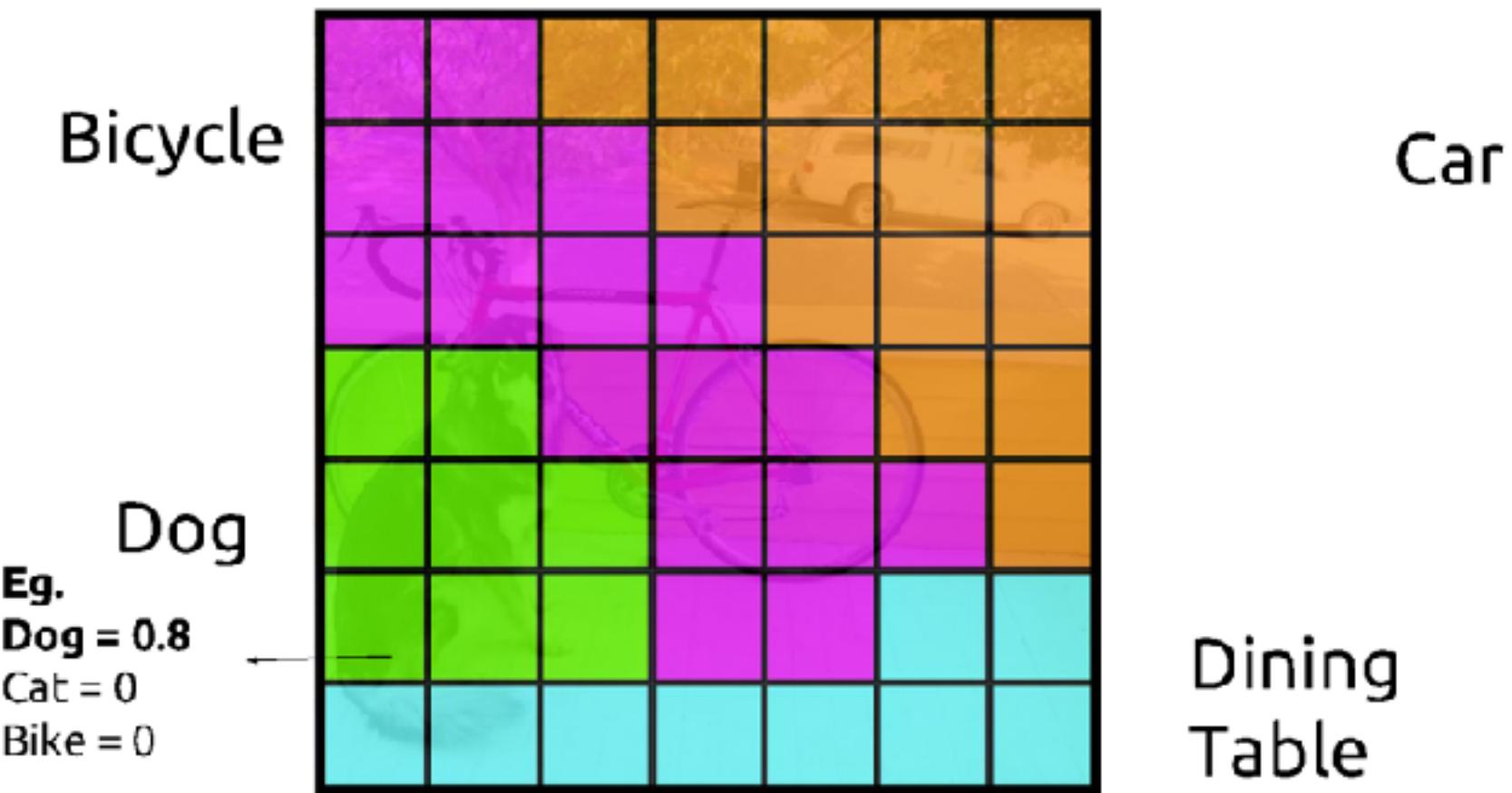
Dog

Dining
Table



YOLO: You Only Look Once

Conditioned on object: $P(\text{Car} \mid \text{Object})$



YOLO: You Only Look Once

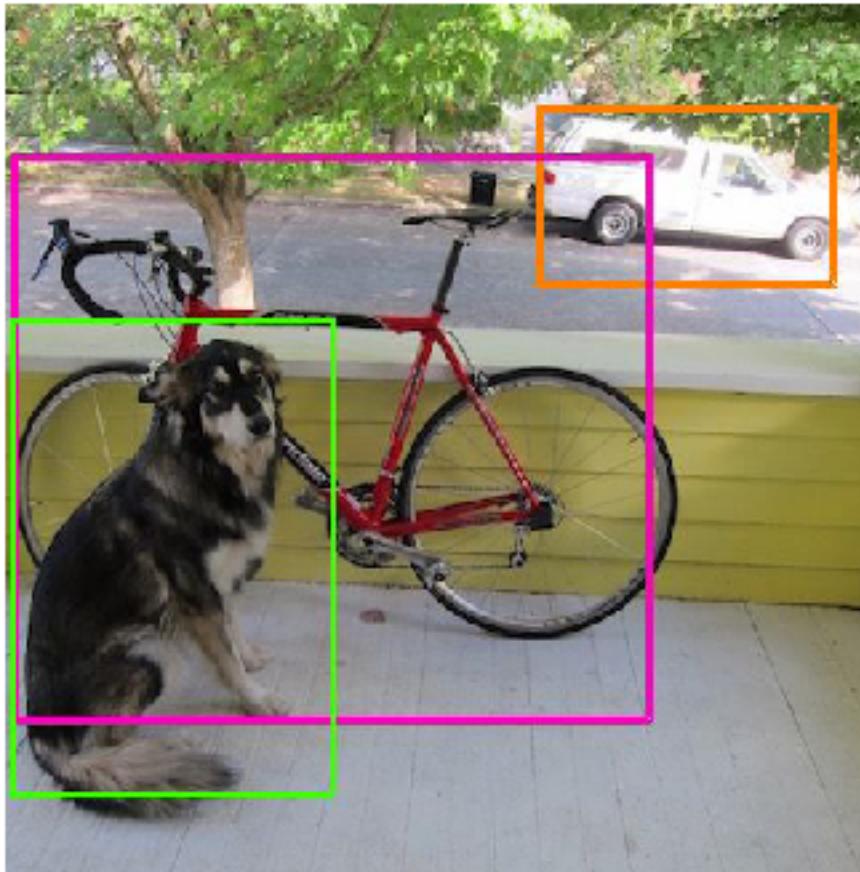
Then we combine the box and class predictions.



$$\begin{aligned} & P(\text{class}|\text{Object}) * P(\text{Object}) \\ & = P(\text{class}) \end{aligned}$$

YOLO: You Only Look Once

Finally we do threshold detections and NMS

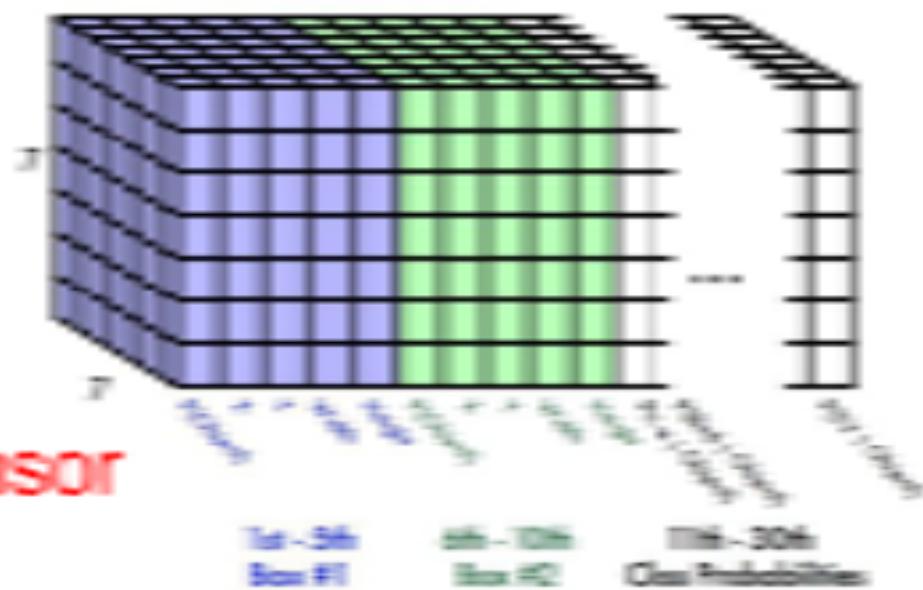


YOLO: You Only Look Once

<https://arxiv.org/abs/1506.02640>

Each cell predicts:

- For each bounding box:
 - Coordinates (x, y, w, h)
 - Confidence value
- Some number of class probabilities



$S \times S \times (B + E + C)$ tensor

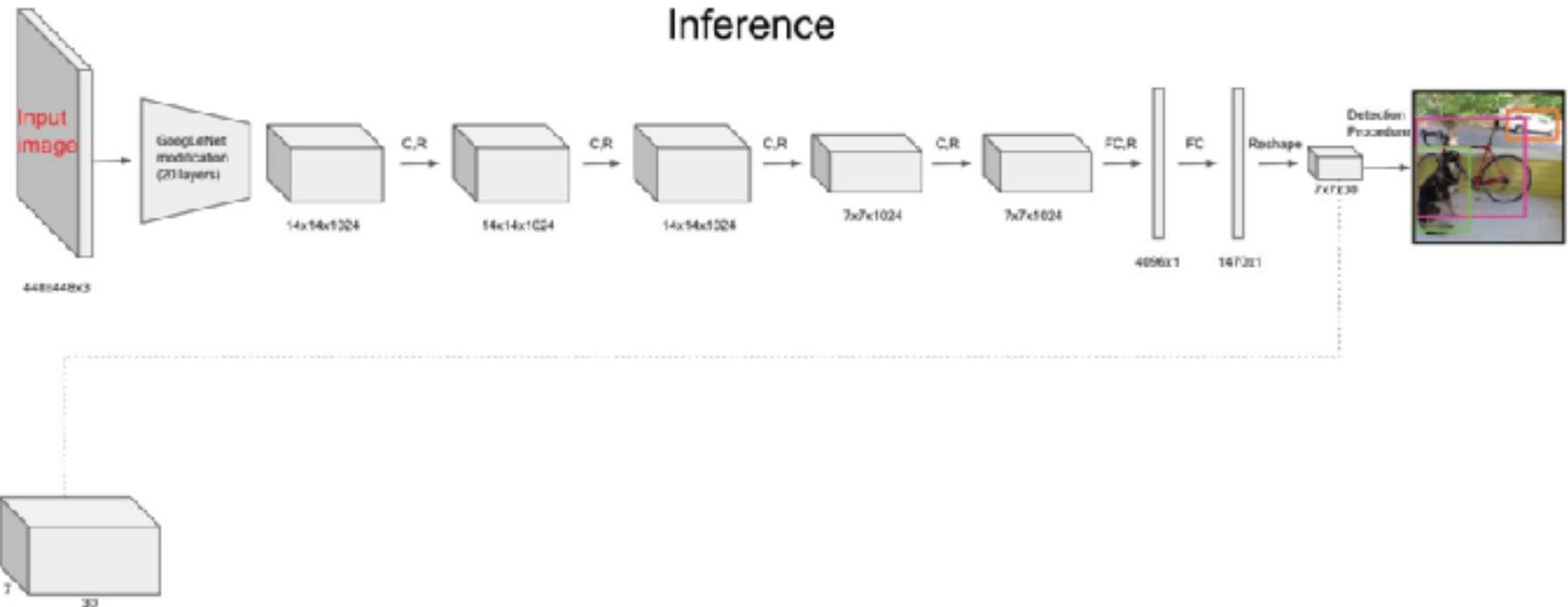
YOLO: You Only Look Once

Network

YOLO: You Only Look Once

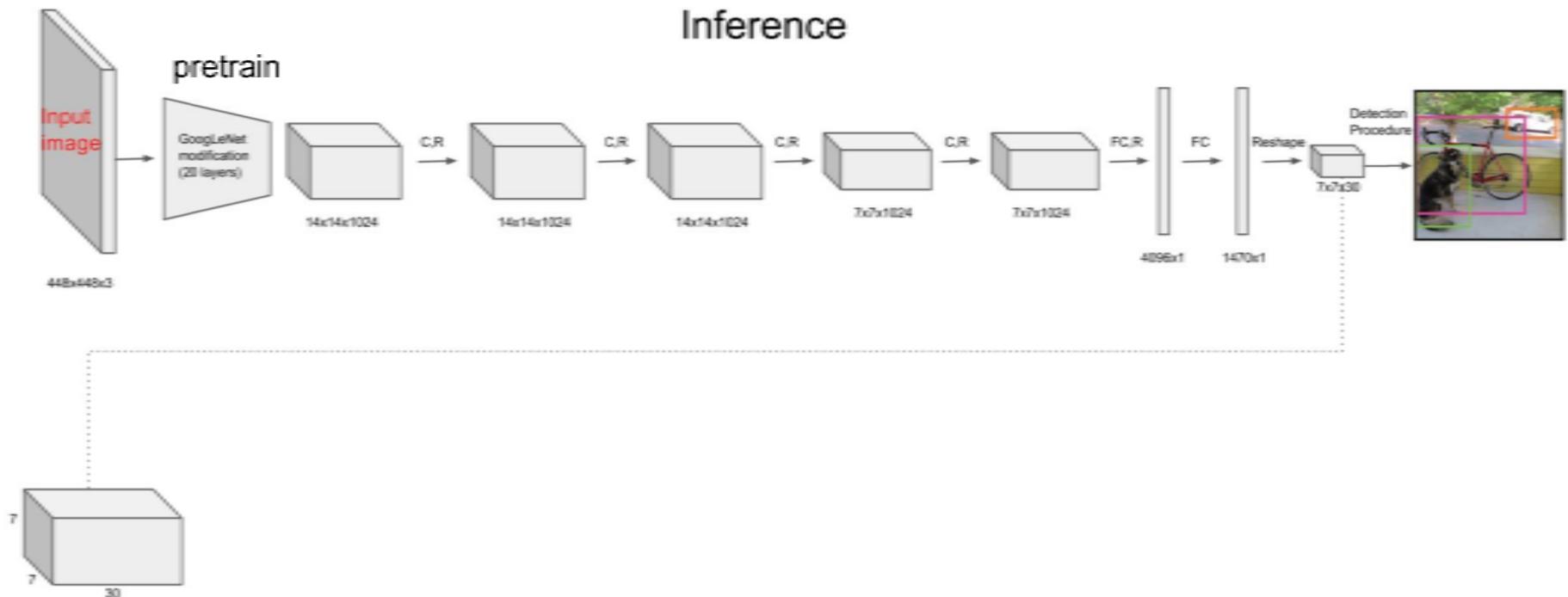
<https://zhuanlan.zhihu.com/p/24916786?refer=xiaoleiminote>

Inference



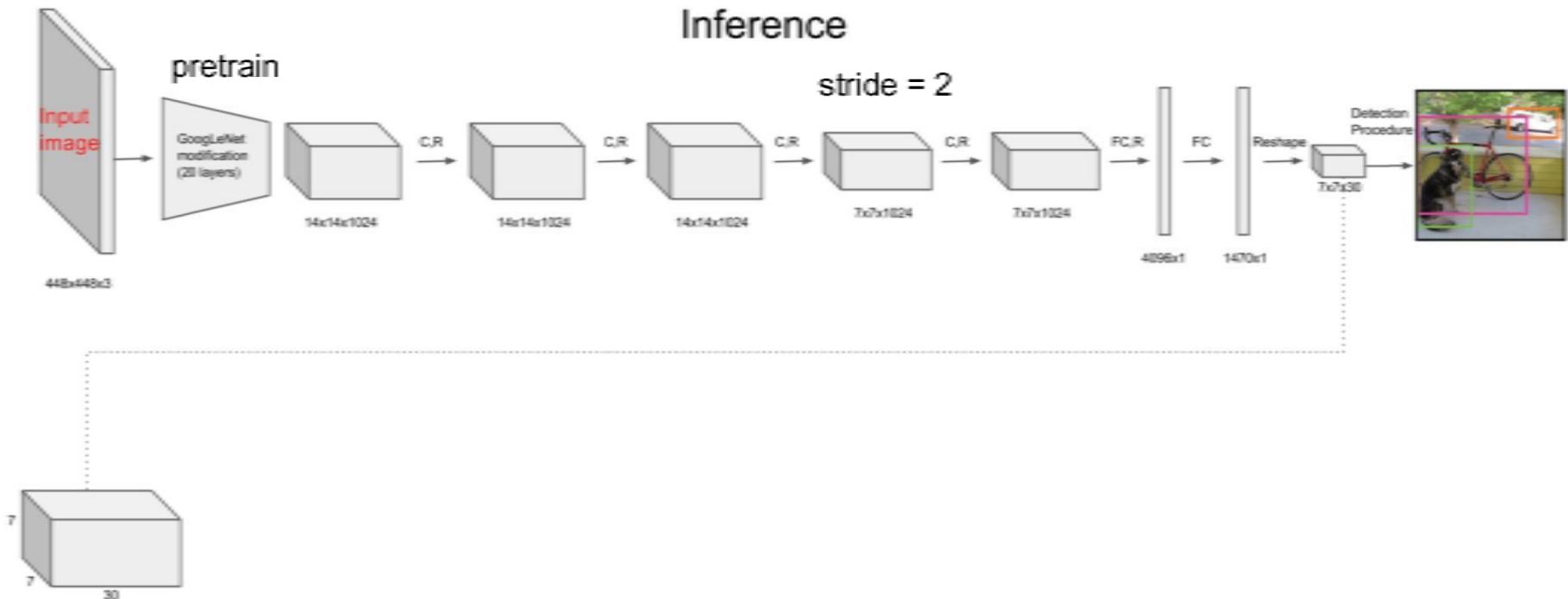
YOLO: You Only Look Once

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



YOLO: You Only Look Once

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

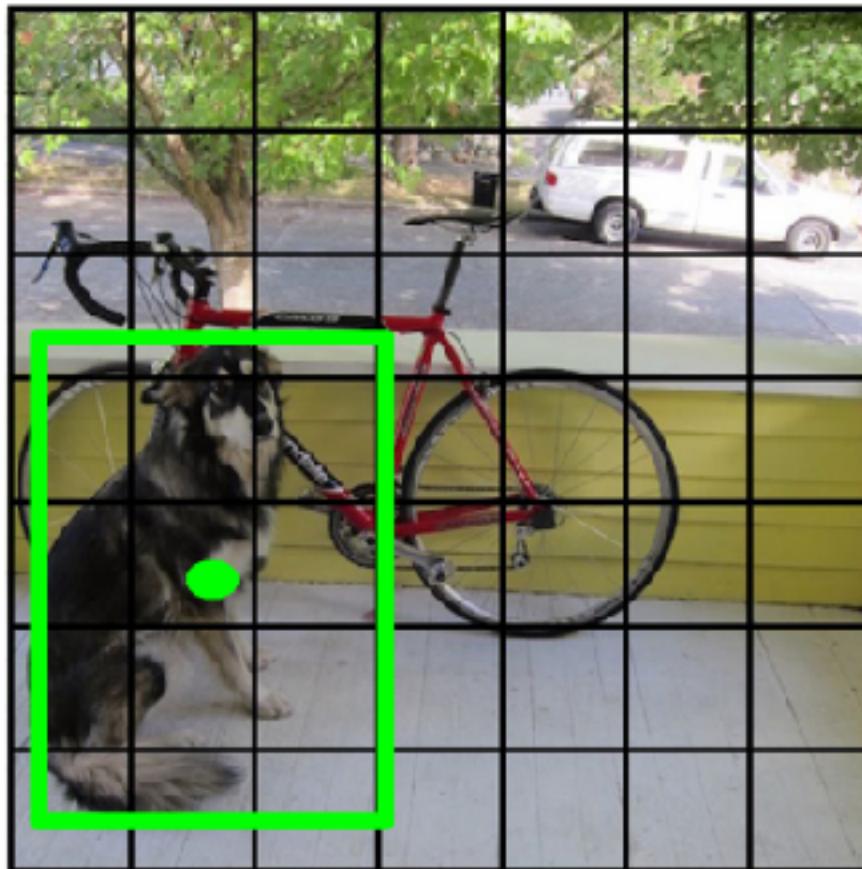


YOLO: You Only Look Once

Train

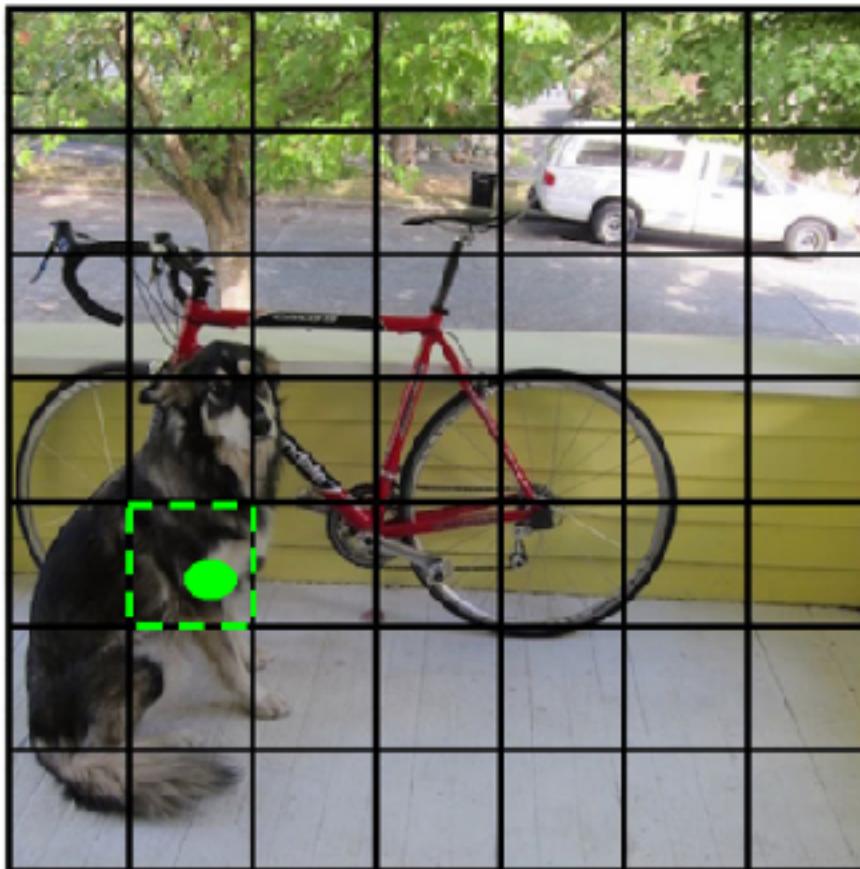
YOLO: You Only Look Once

During training, match example to the right cell



YOLO: You Only Look Once

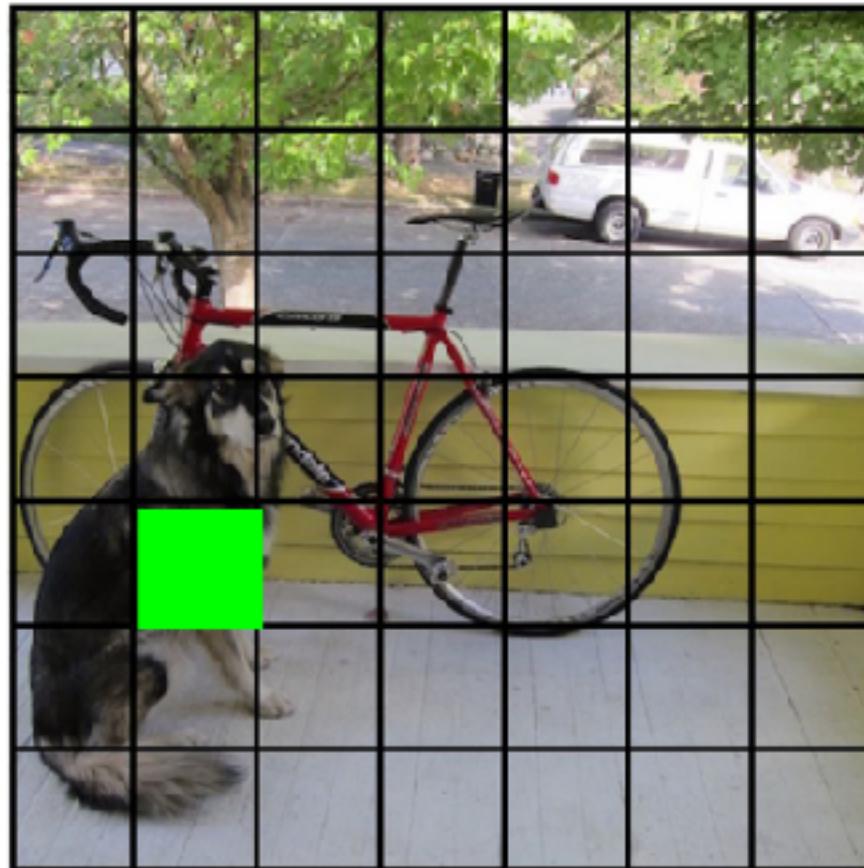
During training, match example to the right cell



YOLO: You Only Look Once

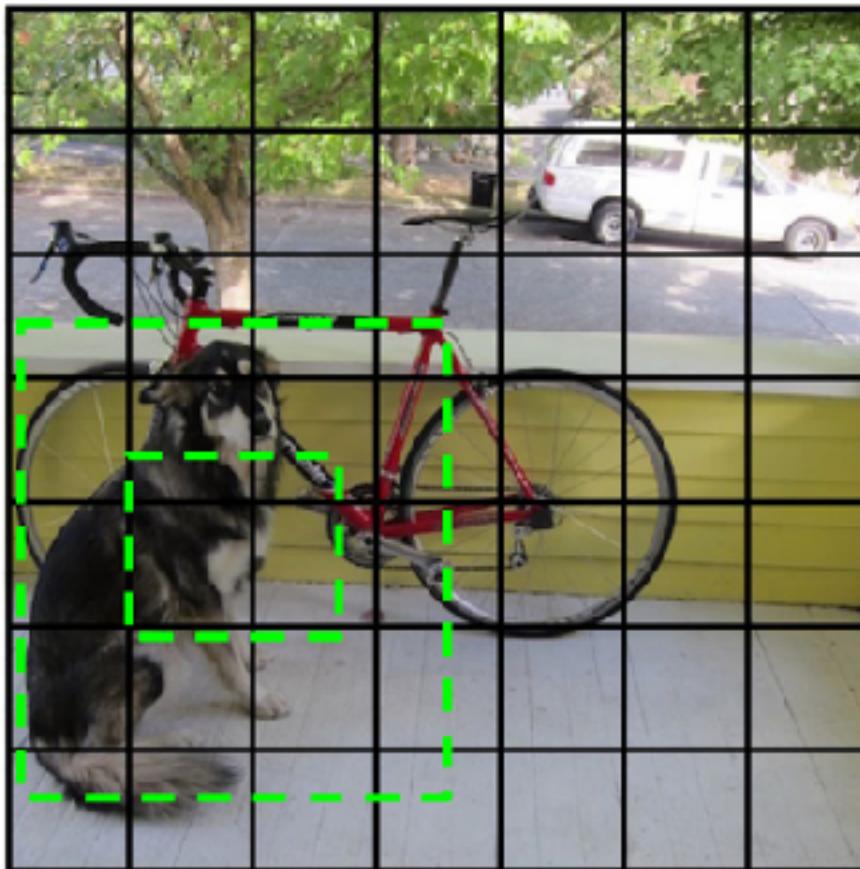
Adjust that cell's class prediction

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



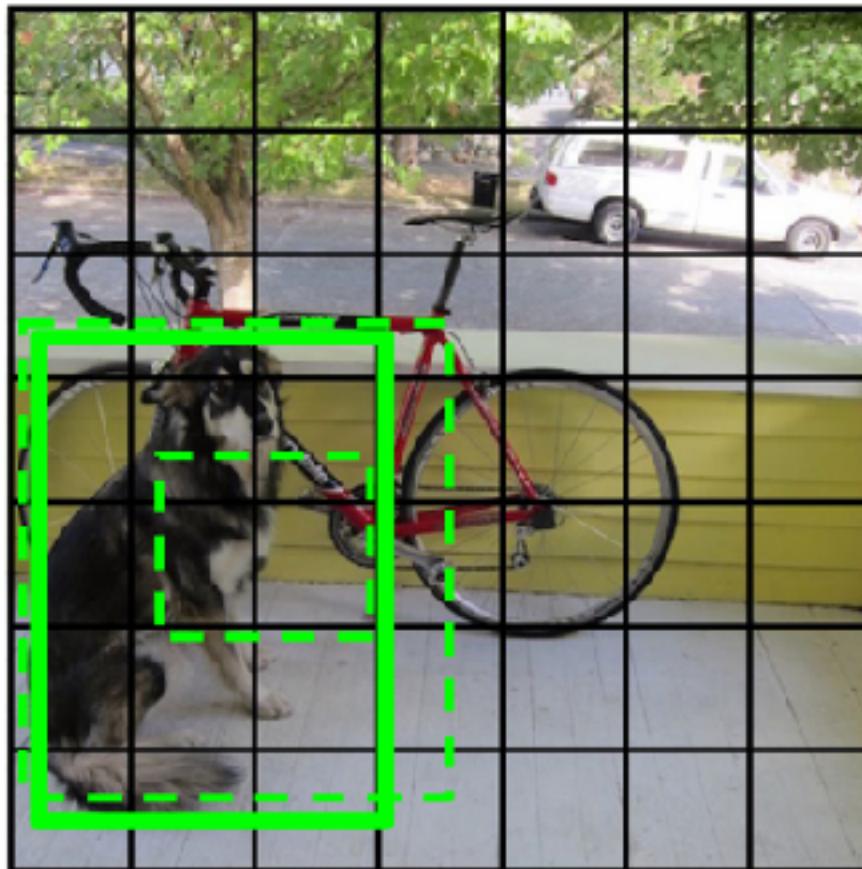
YOLO: You Only Look Once

Look at that cell's predicted boxes



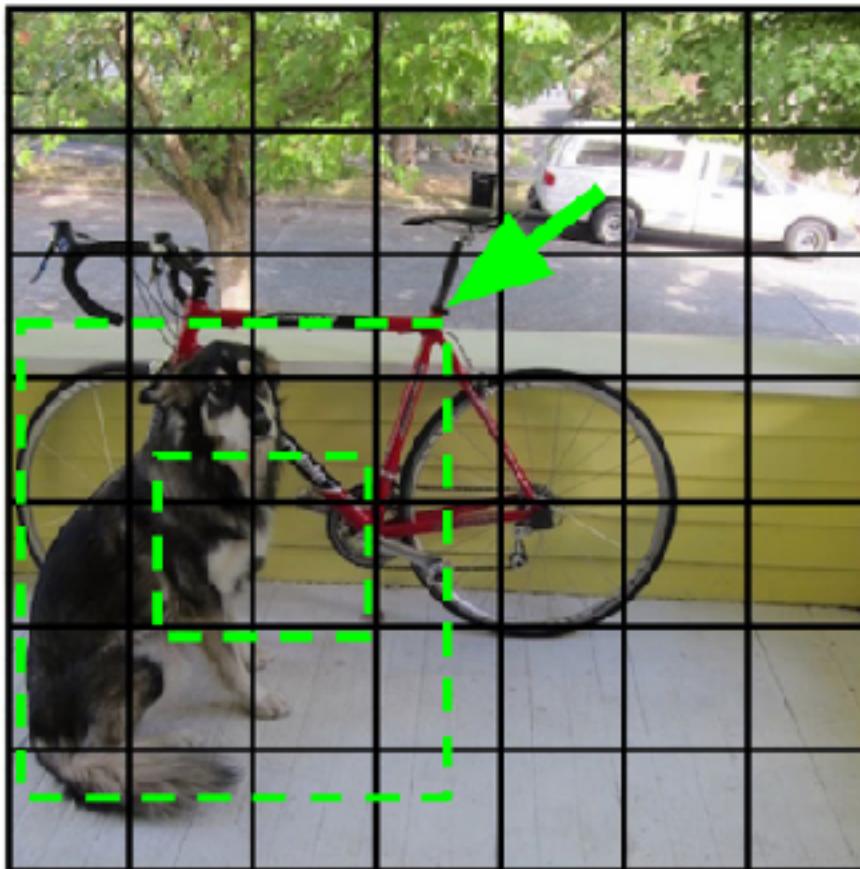
YOLO: You Only Look Once

Find the best one, adjust it, increase the confidence



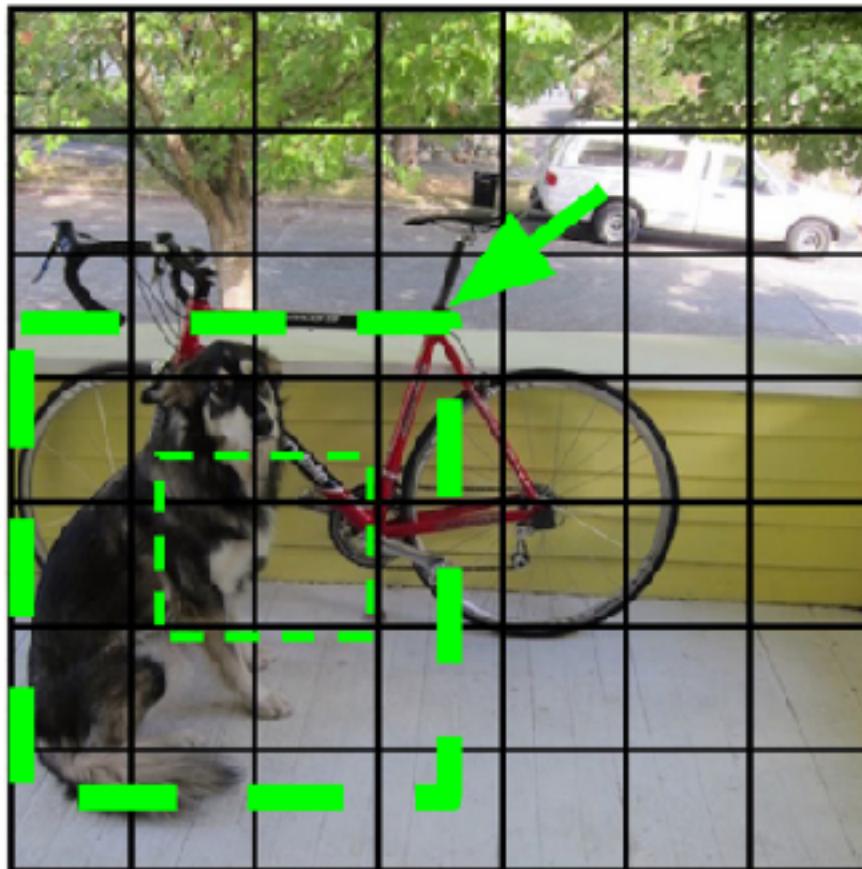
YOLO: You Only Look Once

Find the best one, adjust it, increase the confidence



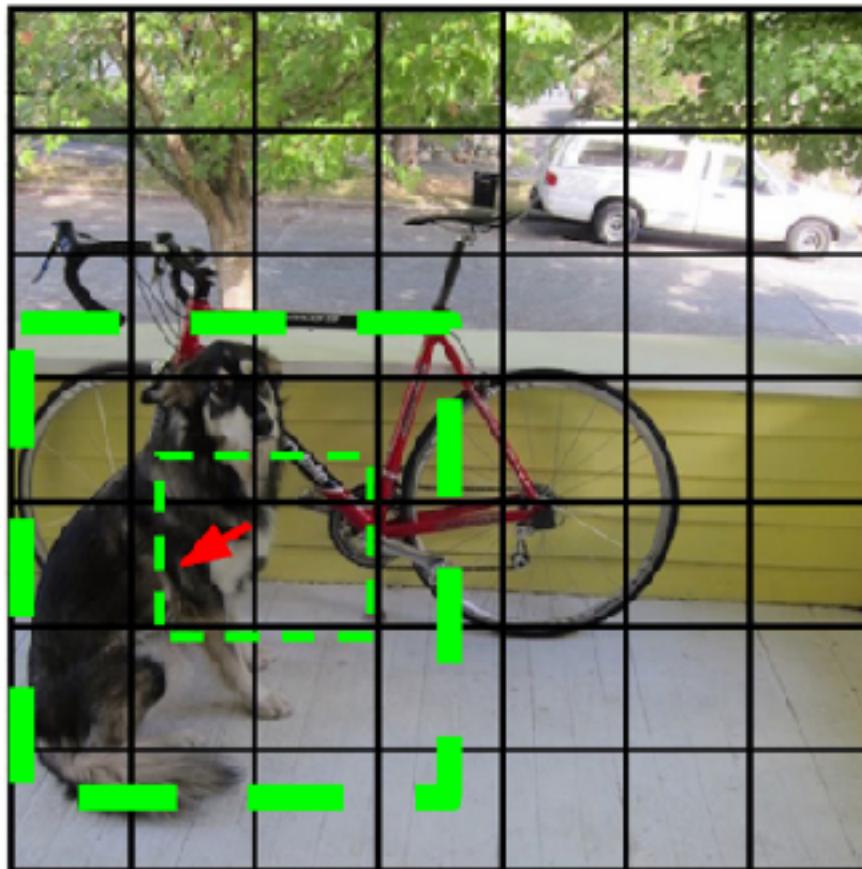
YOLO: You Only Look Once

Find the best one, adjust it, increase the confidence



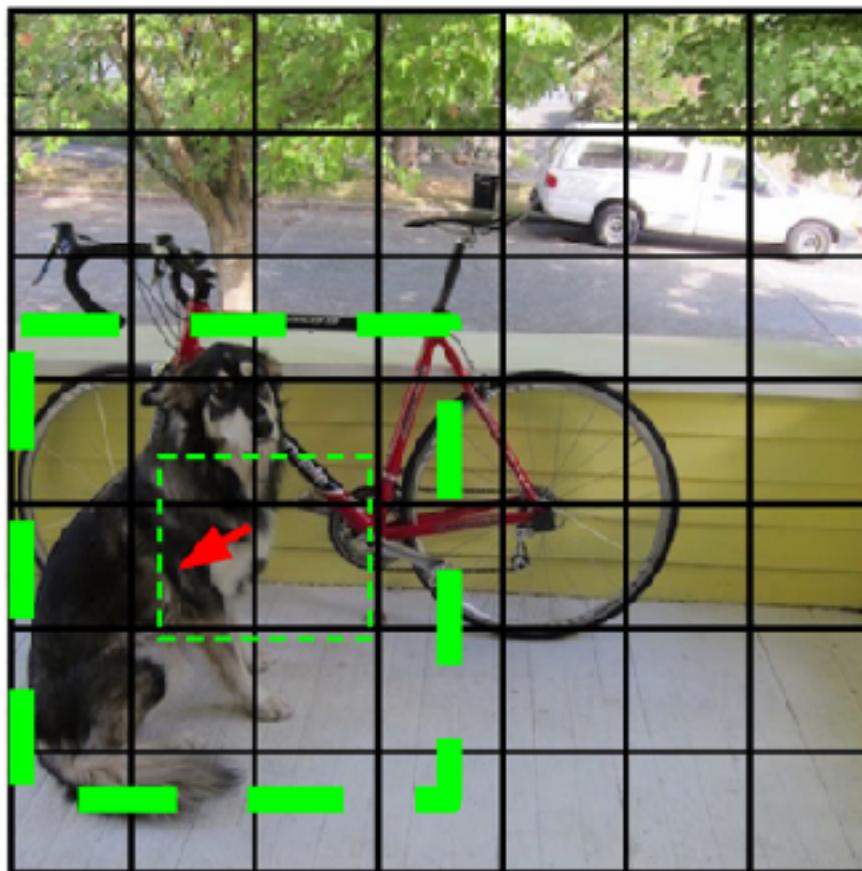
YOLO: You Only Look Once

Decrease the confidence of the other box



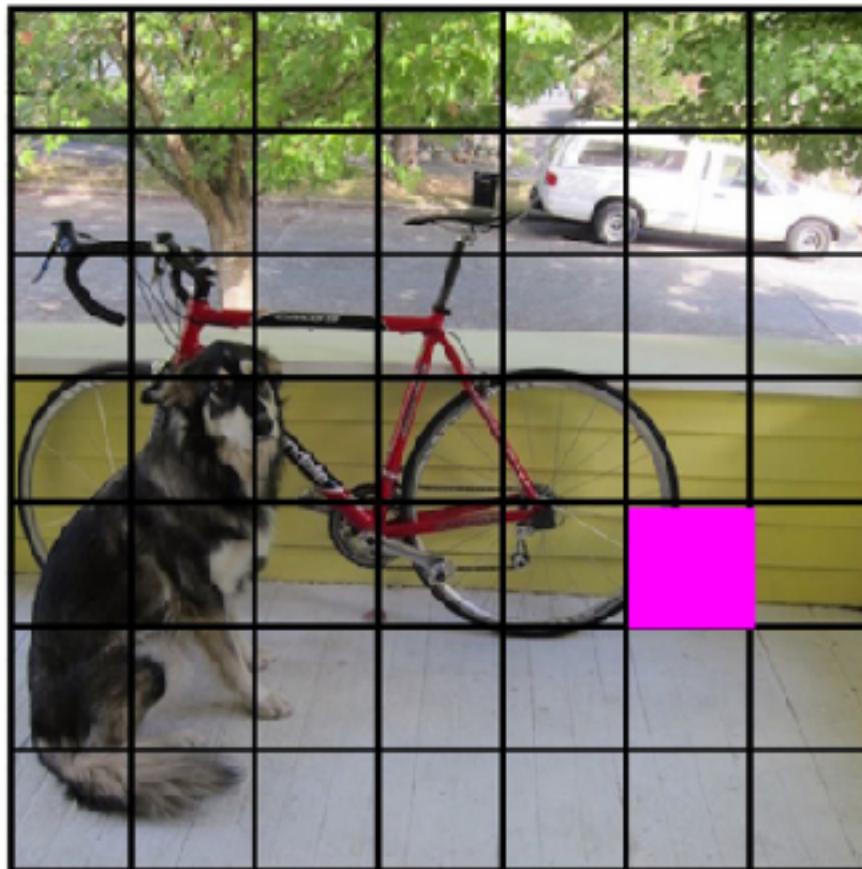
YOLO: You Only Look Once

Decrease the confidence of the other box



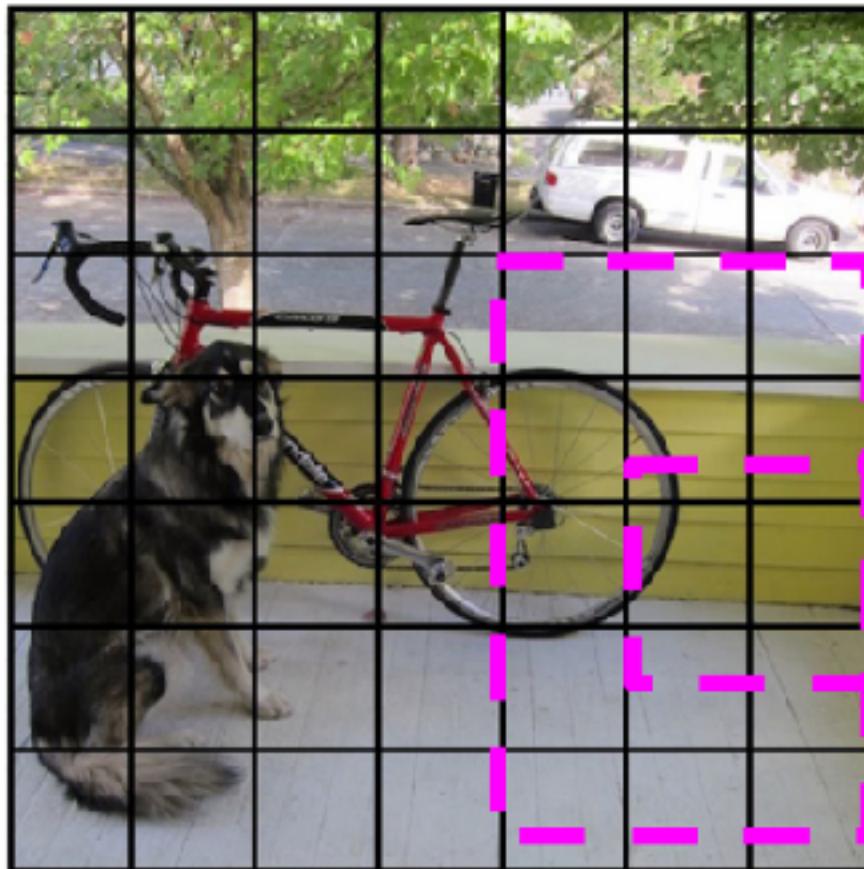
YOLO: You Only Look Once

Some cells don't have any ground truth detections!



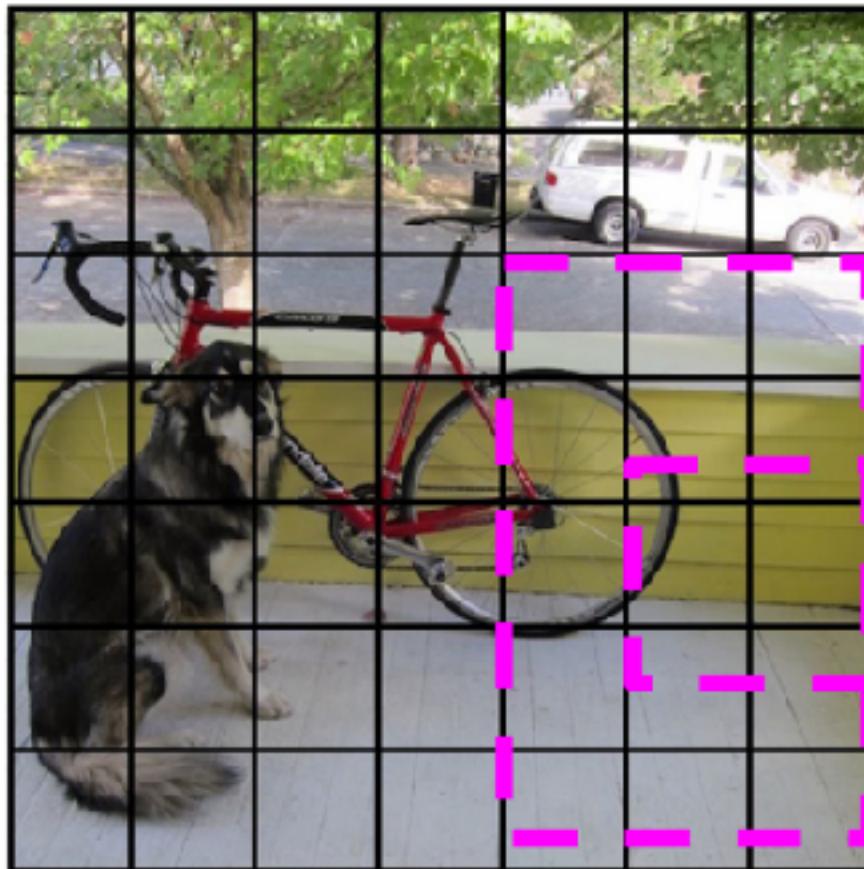
YOLO: You Only Look Once

Some cells don't have any ground truth detections!



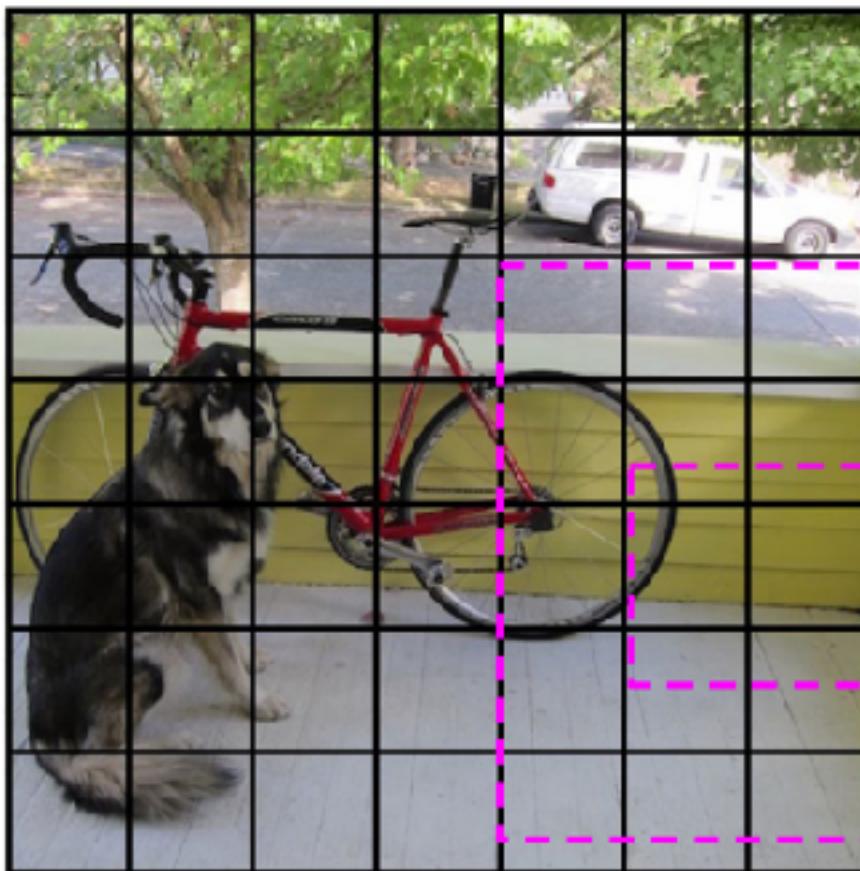
YOLO: You Only Look Once

Decrease the confidence of boxes boxes



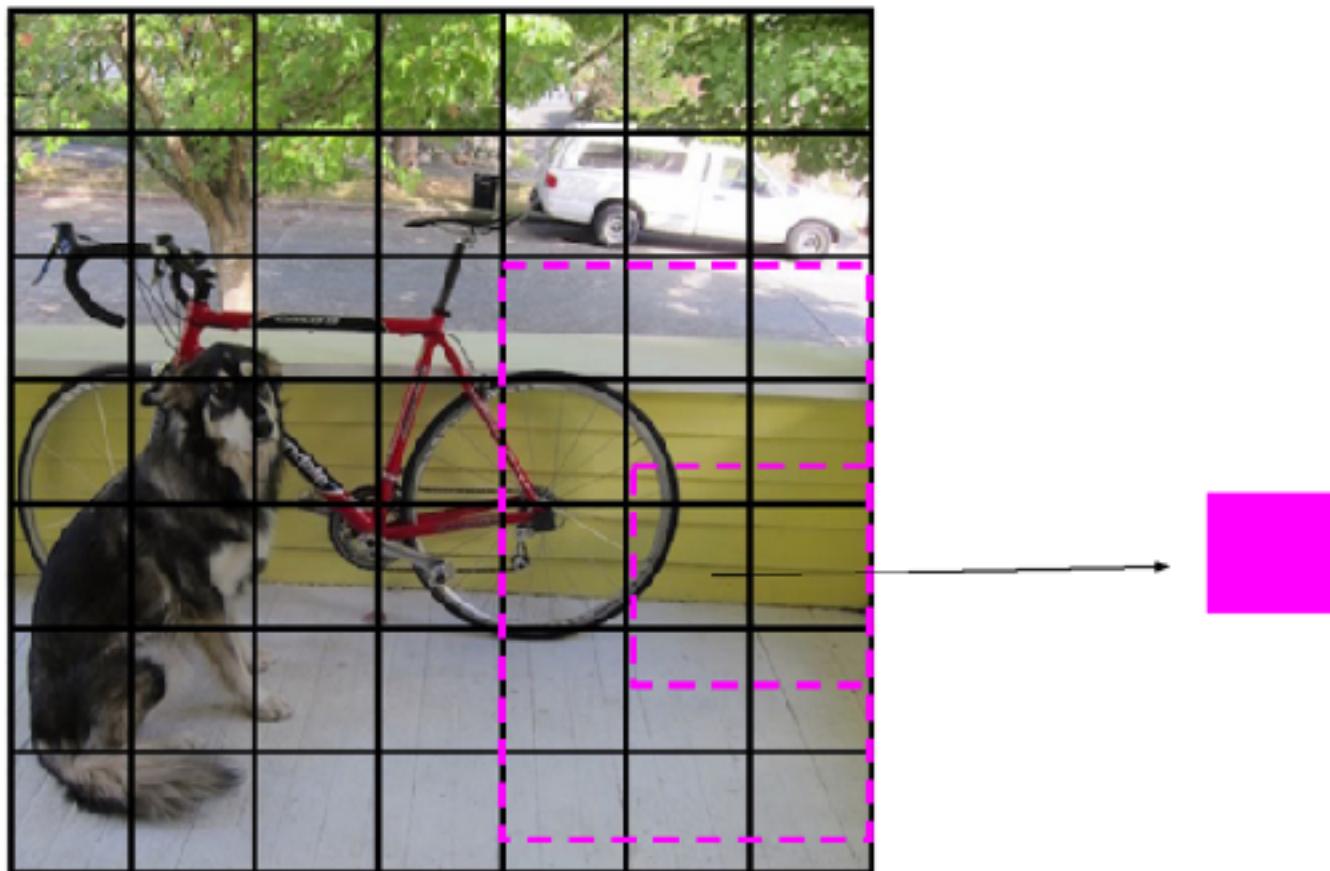
YOLO: You Only Look Once

Decrease the confidence of these boxes



YOLO: You Only Look Once

Don't adjust the class probabilities or coordinates



YOLO: Loss Function

- 1_i^{obj} is 1 if there is an object in cell i and 0 otherwise,
- $1_{i,j}^{obj}$ 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 $1_i^{obj} = 1$, 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

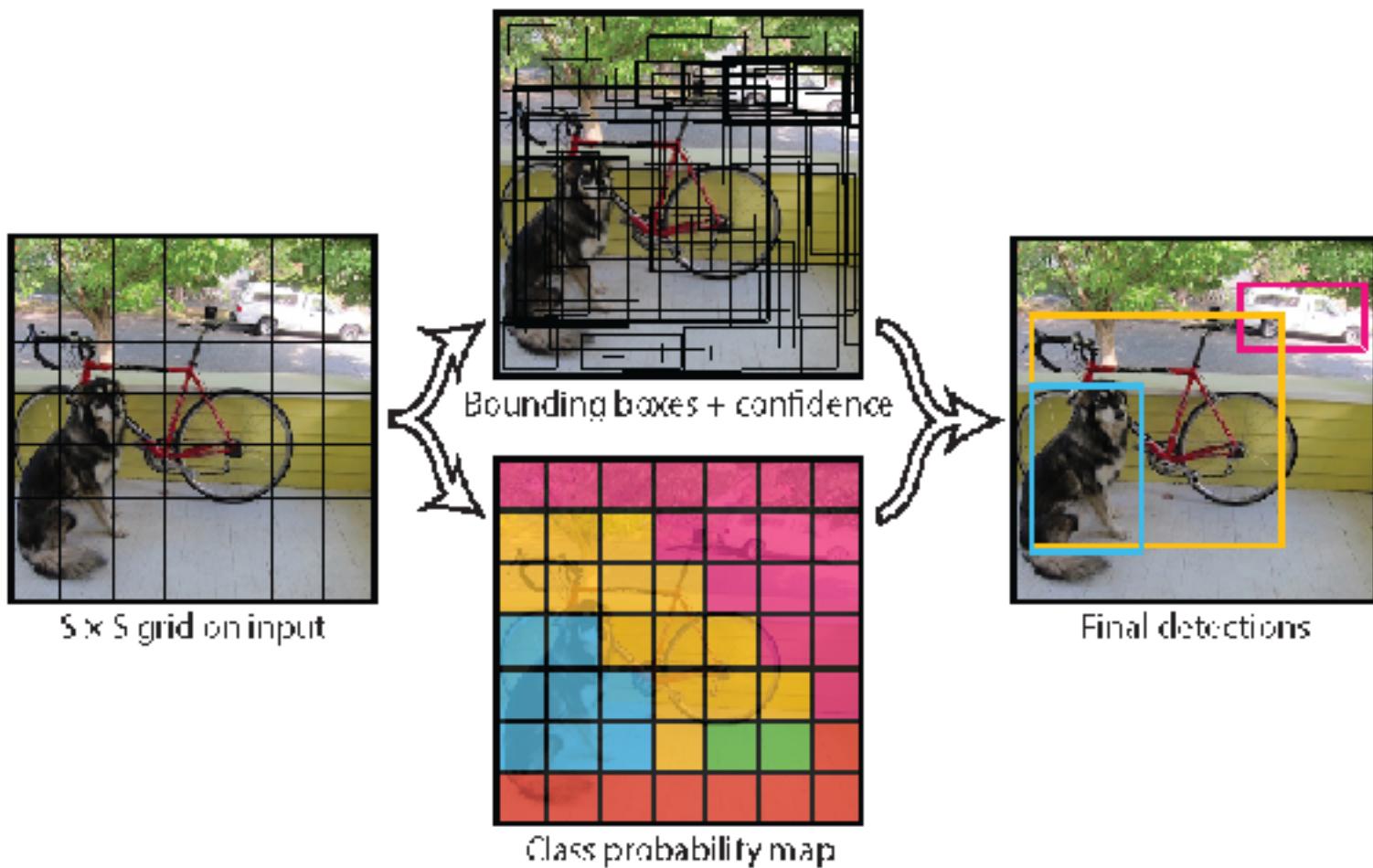
The training procedure first computes on each image the value of the $\mathbf{1}_{i,j}^{obj}$'s and $c_{i,j}$, and then does one step to minimize

$$\begin{aligned}& \lambda_{coord} \sum_{i=1}^S \sum_{j=1}^B \mathbf{1}_{i,j}^{obj} \left((x_i - \hat{x}_{i,j})^2 + (y_i - \hat{y}_{i,j})^2 + (\sqrt{w_i} - \sqrt{\hat{w}_{i,j}})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_{i,j}})^2 \right) \\& + \lambda_{obj} \sum_{i=1}^S \sum_{j=1}^B \mathbf{1}_{i,j}^{obj} (c_{i,j} - \hat{c}_{i,j})^2 + \lambda_{noobj} \sum_{i=1}^S \sum_{j=1}^B (1 - \mathbf{1}_{i,j}^{obj}) \hat{c}_{i,j}^2 \\& + \lambda_{classes} \sum_{i=1}^S \mathbf{1}_i^{obj} \sum_{c=1}^C (p_{i,c} - \hat{p}_{i,c})^2.\end{aligned}$$

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)

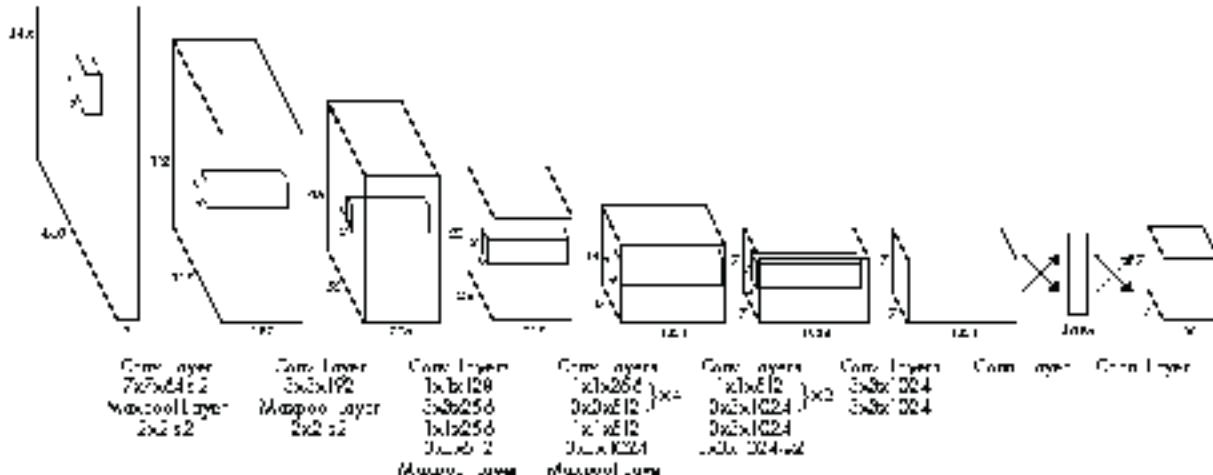
YOLO: You Only Look Once



(Redmon et al., 2015)

YOLO: You Only Look Once

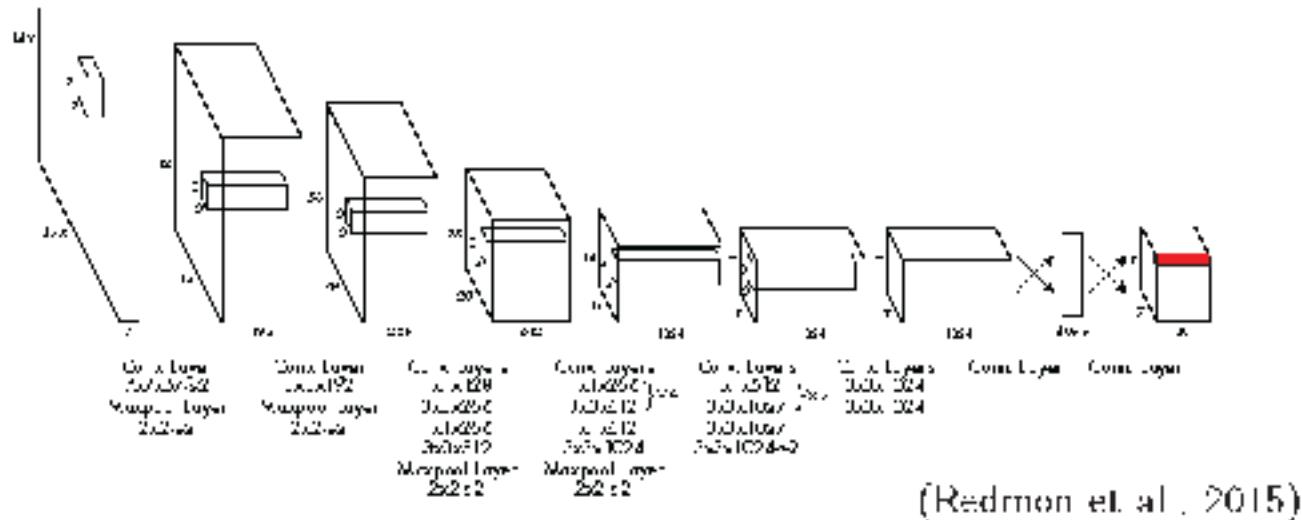
The output corresponds to splitting the image into a regular $S \times S$ grid, with $S = 7$.



(Redmon et al., 2015)

YOLO: You Only Look Once

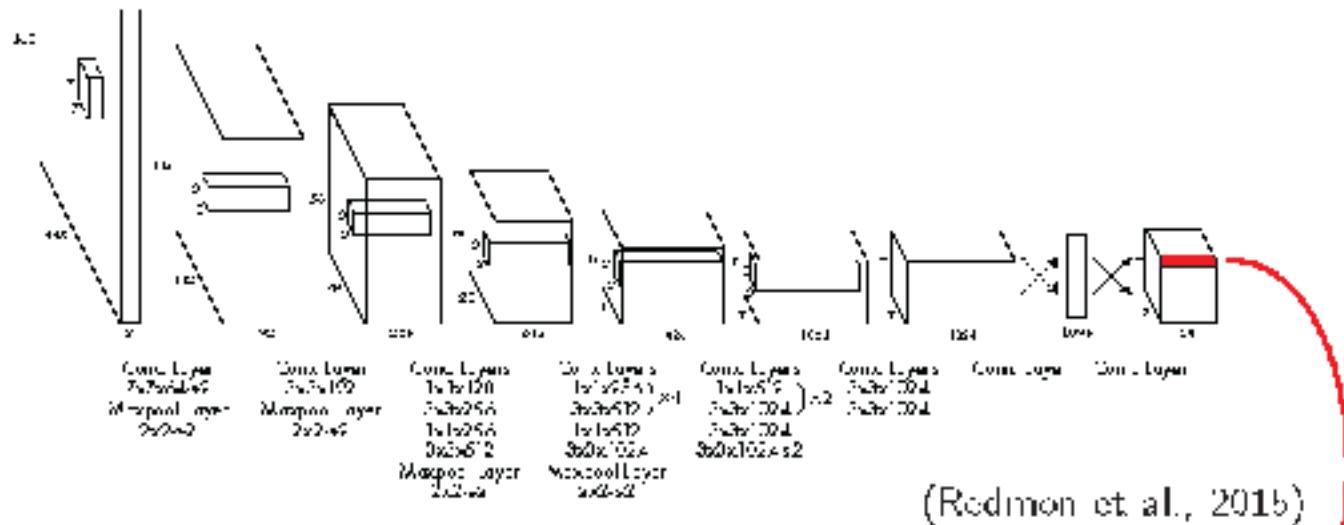
The output corresponds to splitting the image into a regular $S \times S$ grid, with $S = 7$, and for each cell, to predict a 30d vector.



YOLO: You Only Look Once

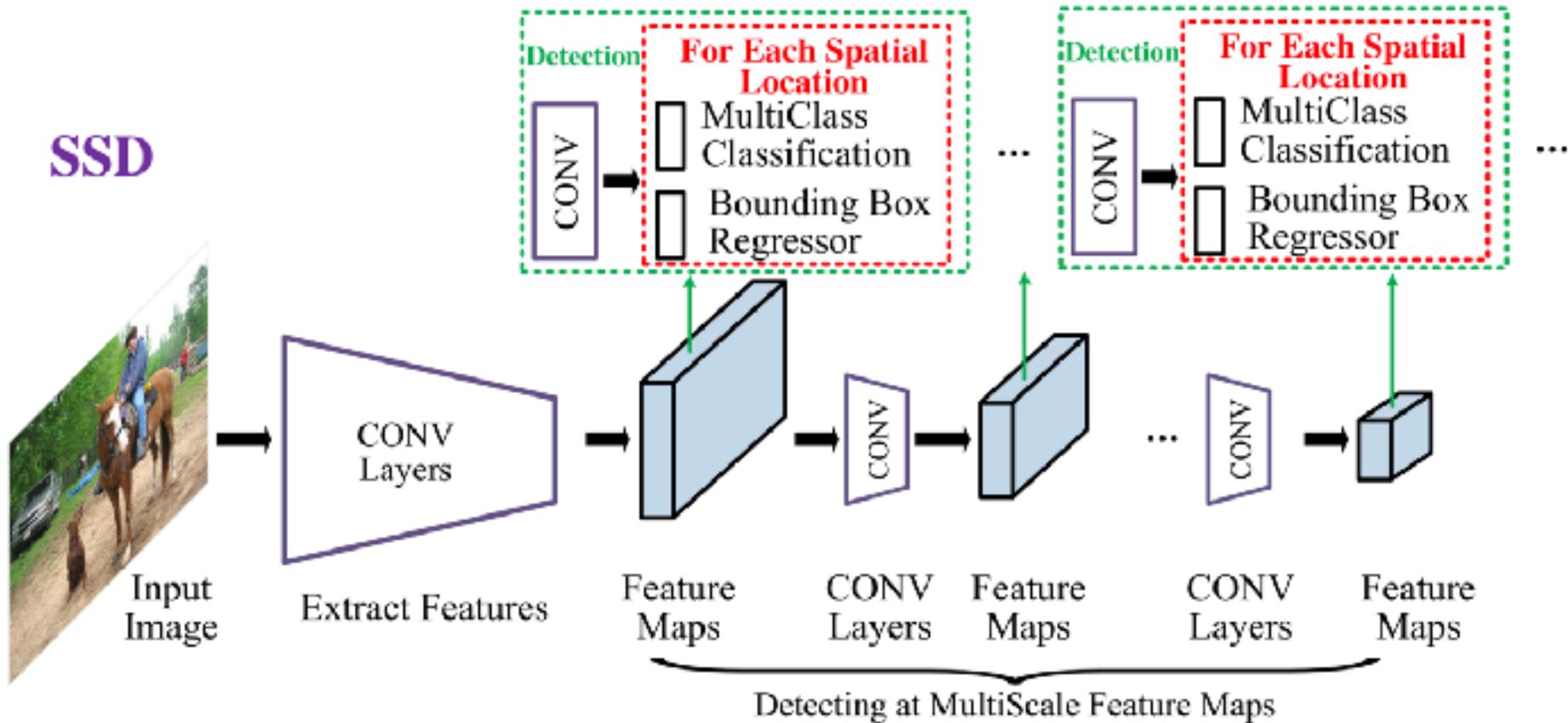
The output corresponds to splitting the image into a regular $S \times S$ grid, with $S = 7$, and for each cell, to predict a 30d vector:

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

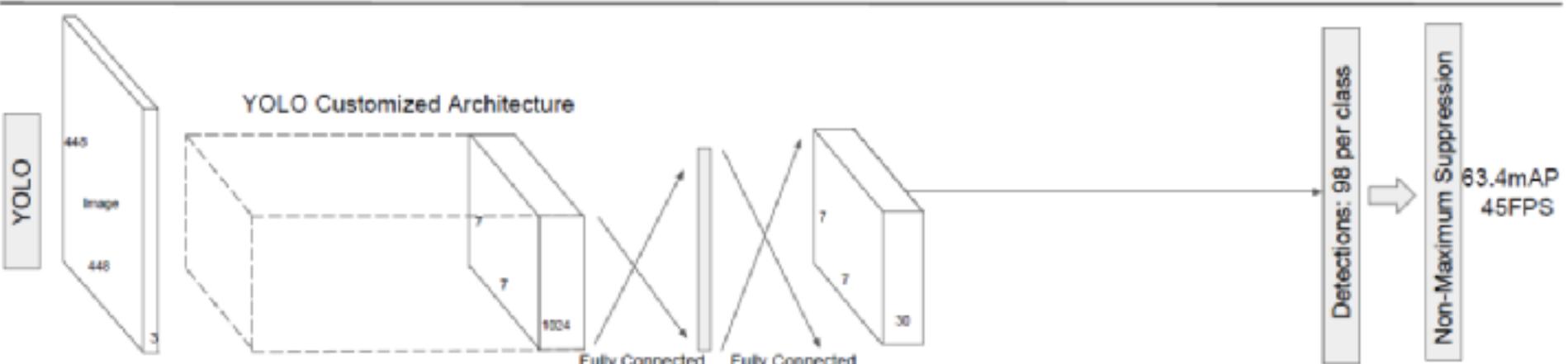


SSD: Single Shot Multibox Detector

SSD



SSD: Single Shot Multibox Detector



SSD: Single Shot Multibox Detector

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

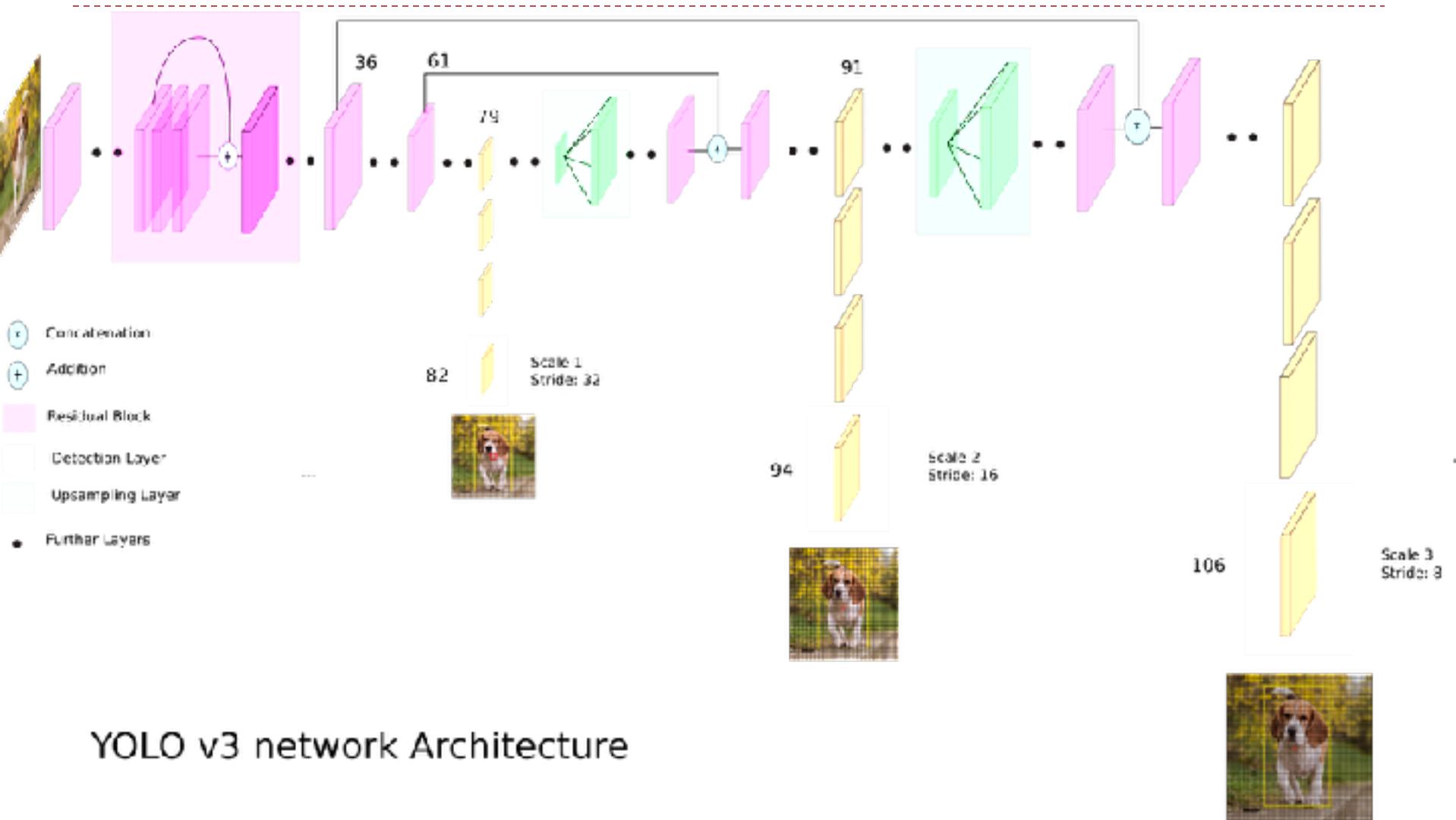
SSD: Single Shot Multibox Detector

- ▶ At **Conv4_3**,
 - ▶ size $38 \times 38 \times 512$
 - ▶ 4 bounding boxes (classes + 4) outputs.
 - ▶ the output is $38 \times 38 \times 4 \times (c+4)$.
 - ▶ For 20 object classes plus one background class
 - ▶ $38 \times 38 \times 4 \times (21+4) = 144,400$.
 - ▶ there are $38 \times 38 \times 4 = 5776$ bounding boxes.
- ▶ Similarly for other conv layers:
 - ▶ Conv7: $19 \times 19 \times 6 = 2166$ boxes (6 boxes for each location)
 - ▶ Conv8_2: $10 \times 10 \times 6 = 600$ boxes (6 boxes for each location)
 - ▶ Conv9_2: $5 \times 5 \times 6 = 150$ boxes (6 boxes for each location)
 - ▶ Conv10_2: $3 \times 3 \times 4 = 36$ boxes (4 boxes for each location)
 - ▶ Conv11_2: $1 \times 1 \times 4 = 4$ boxes (4 boxes for each location)

$5776 + 2166 + 600 + 150 + 36 + 4 = 8732$ boxes in total.

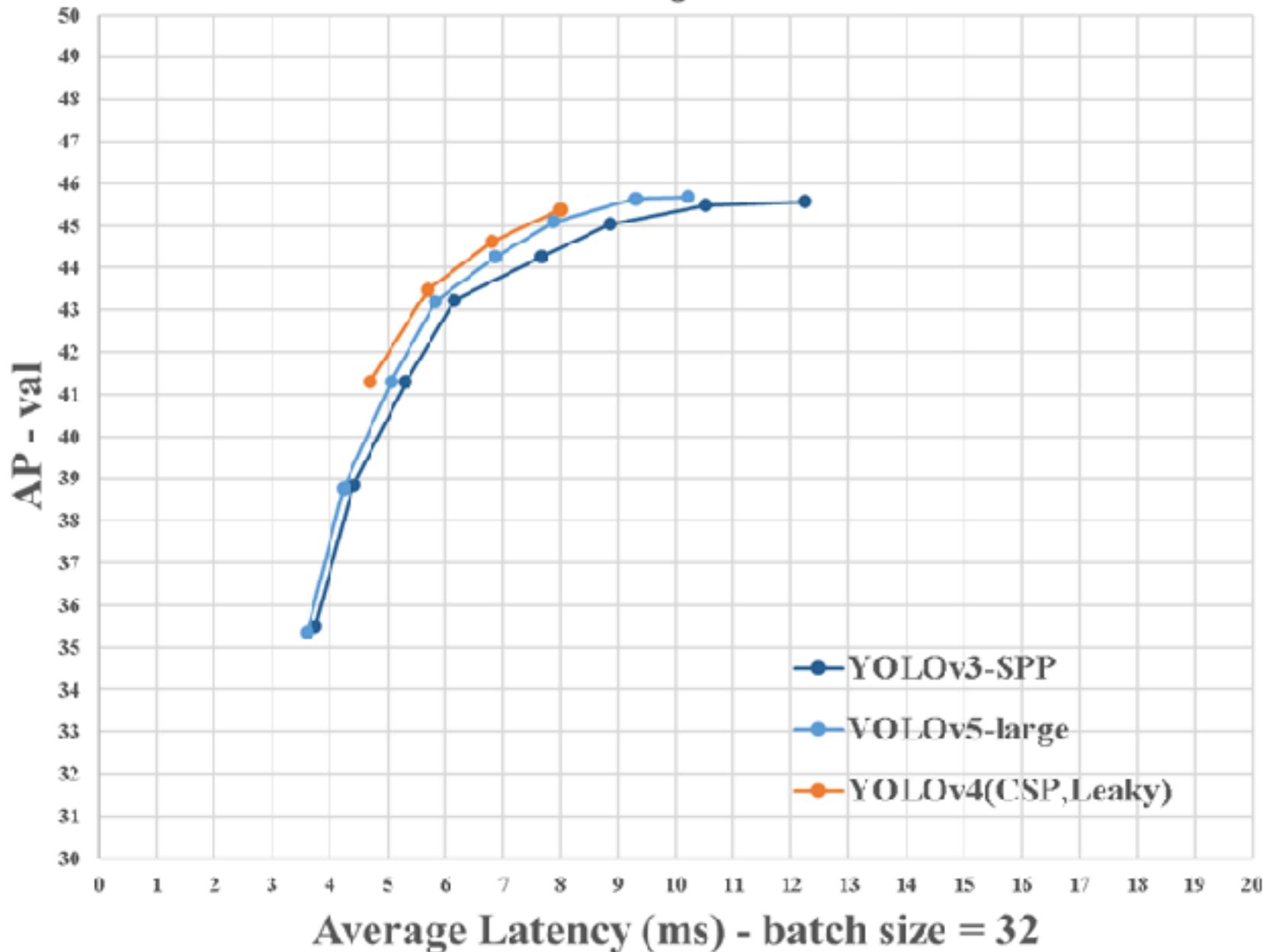
YOLO only got $7 \times 7 \times 2 = 98$ boxes.

YoloV3



YOLO v3 network Architecture

MS COCO Object Detection



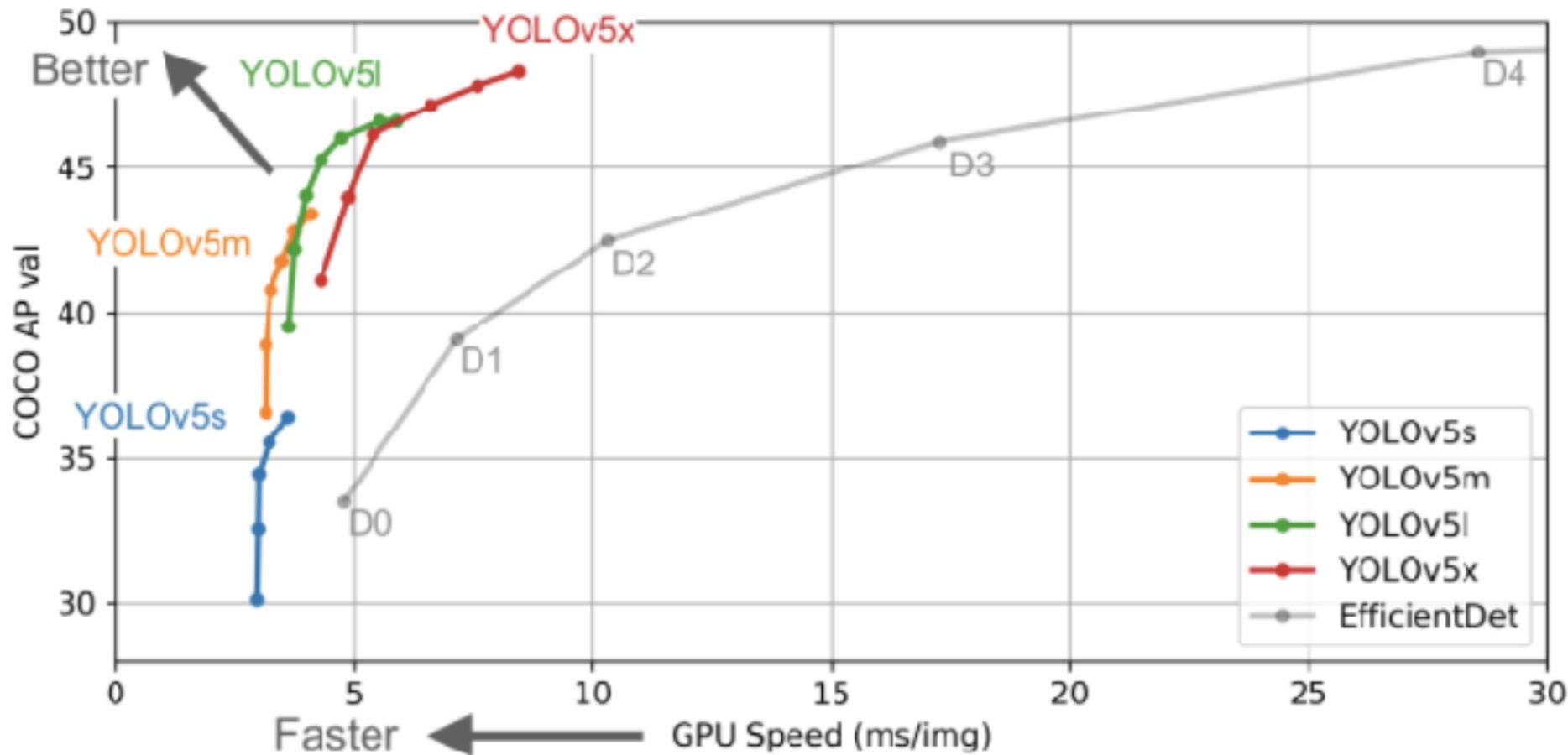
YOLOv5

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).

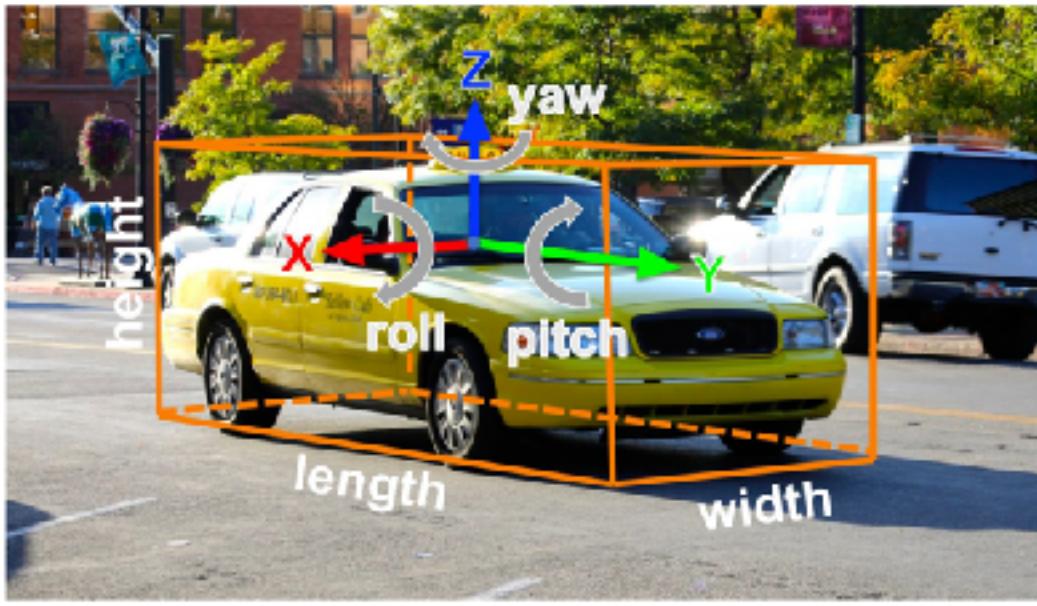
Yolov5



YoloV3Keras.ipynb

3D Object Detection

3D Object Detection



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

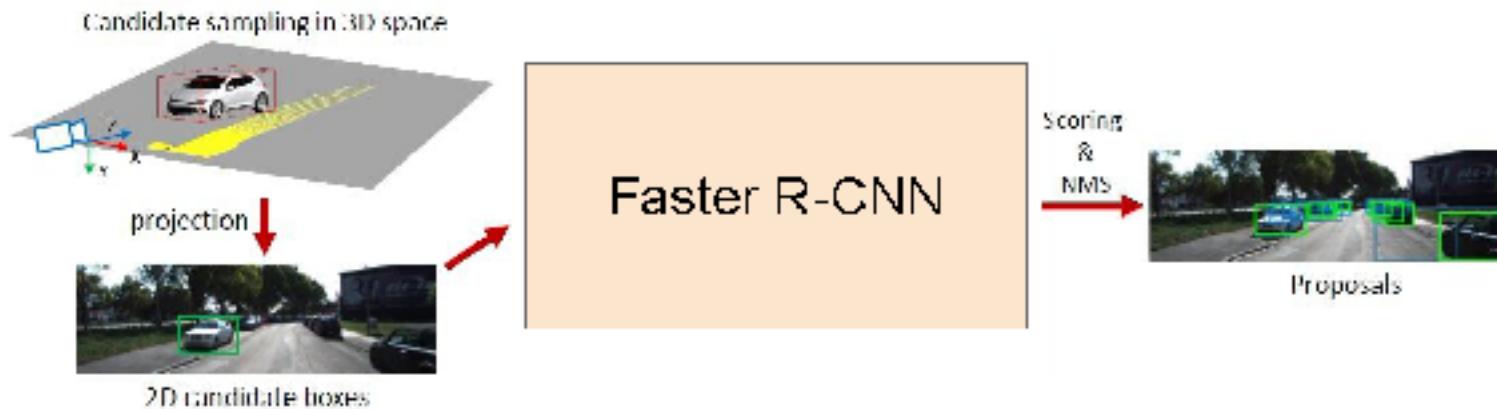
3D Object Detection:
3D oriented bounding box
($x, y, z, w, h, l, r, p, \gamma$)

Simplified bbox: no roll & pitch

Much harder problem than 2D object detection!

This image is CC0 public domain

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

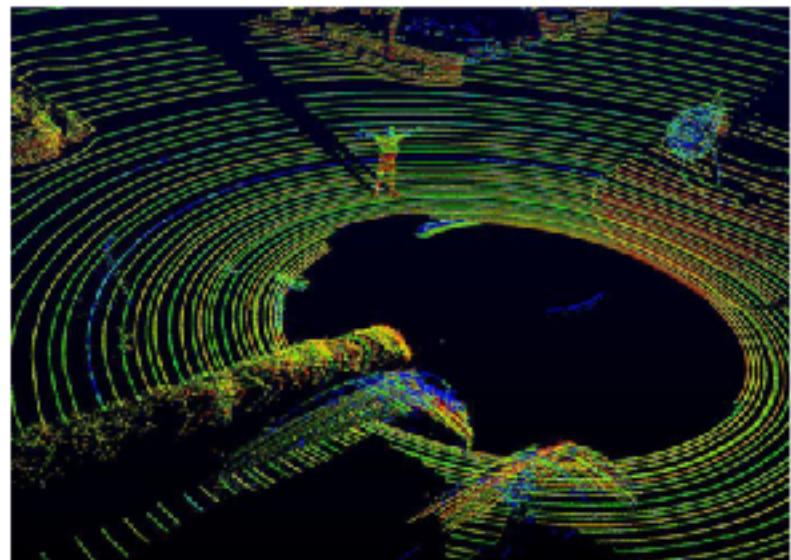
Chen, Xiaozhi, Keuster Kuntz, Ziqi Zhang, Huijin Ma, Sam Gaidar, and Lapjuee Olszewski. "Monocular 3D object detection for autonomous driving." CVPR 2018.

3D Object Detection: Camera + LiDAR

[This image is CC0 public domain](#)

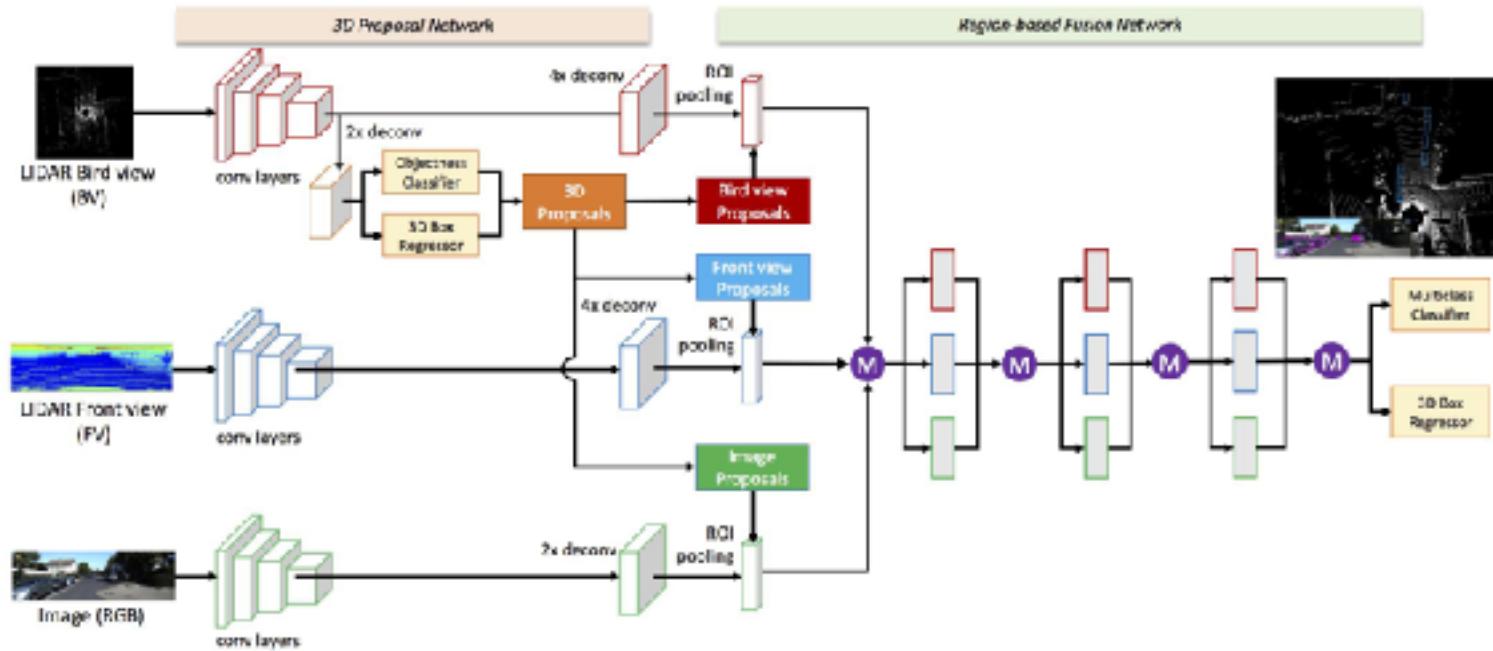


Velodyne (HDL-64e)



3D Point Cloud

3D Object Detection: Camera + LiDAR



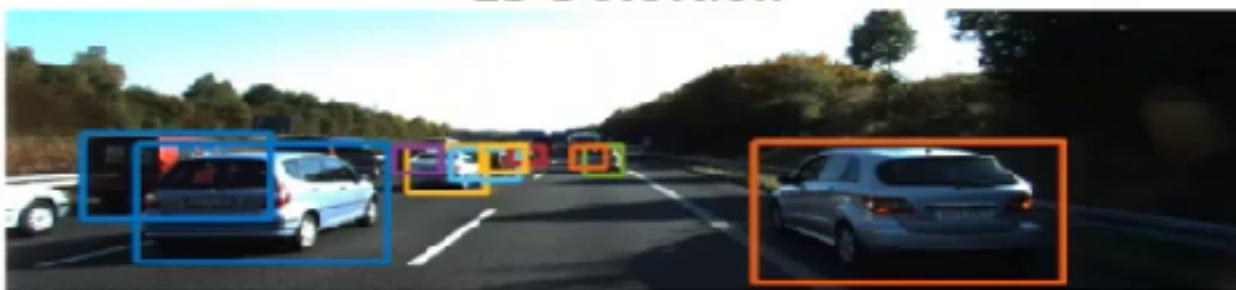
- Combine 3D proposals from multiple views & sensors
- regress 3D box parameters + class score

Chen Xiaozihi, 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

Object Detection for Autonomous Driving

2D Detection

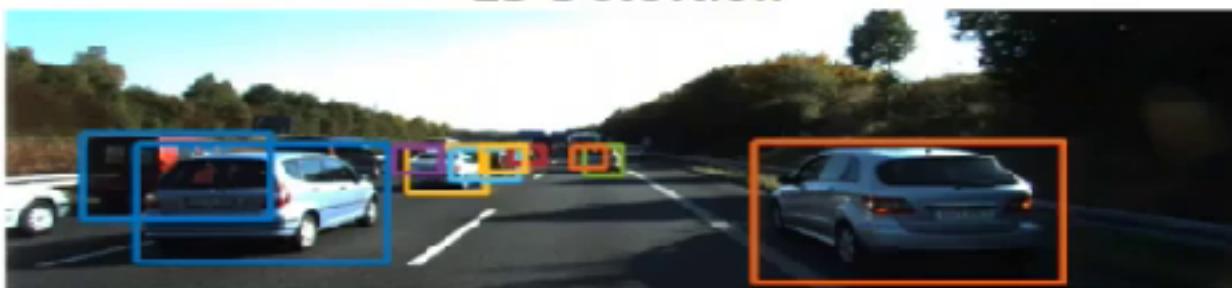


Semantics

3D Object Detection: Camera + LiDAR

Object Detection for Autonomous Driving

2D Detection



Semantics

3D Detection

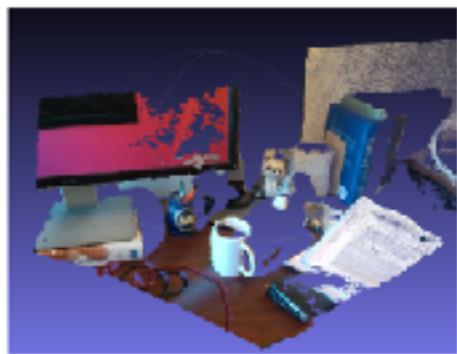


Semantics

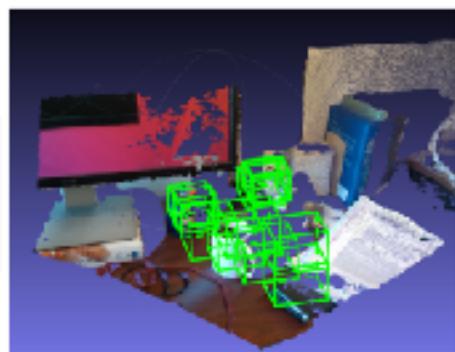
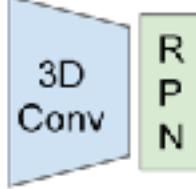
+

3D/Depth

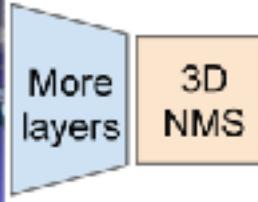
3D Object Detection: RGB-Depth Camera



Voxelized RGB-D point cloud



3D region proposals



Object categories +
3D bounding boxes

“Faster RCNN in 3D”

S. Song, and J. Xiao, Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images, CVPR 2015

Beyond Object Detection

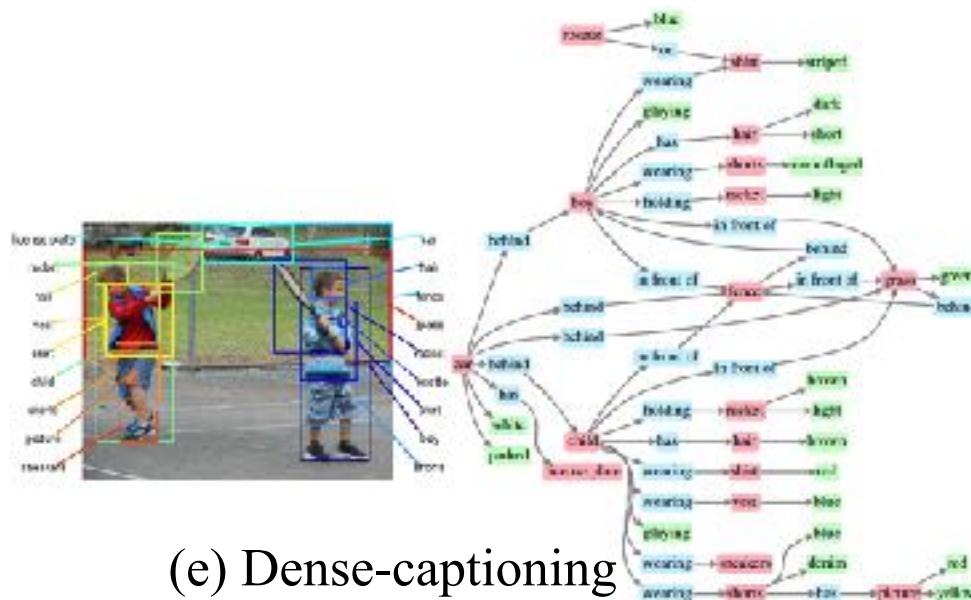
A person riding a motorcycle on a dirt road



Two dogs play in the grass



(e) Auto-captioning



(e) Dense-captioning