165B
Machine Learning
Object Detection

Lei Li (leili@cs)
UCSB

Acknowledgement: Slides borrowed from Bhiksha Raj's 11485 and Mu Li & Alex Smola's 157 courses on Deep Learning, with modification

Recap

- Gradient descent can be sped up by incremental updates
  - Convergence is guaranteed under most conditions
    - Learning rate must shrink with time for convergence
  - Stochastic gradient descent: update after each observation. Can be much faster than batch learning
  - Mini-batch updates: update after batches. Can be more efficient than SGD

- Convergence can be improved using smoothed updates
  - AdaGrad, RMSprop, Adam and more advanced techniques

Stochastic Gradient Descent

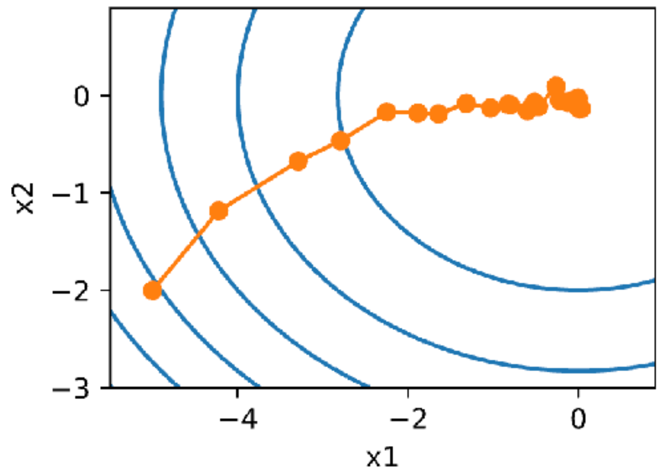
- Instead of compute the full gradient, at each step, randomly select a sample t_i

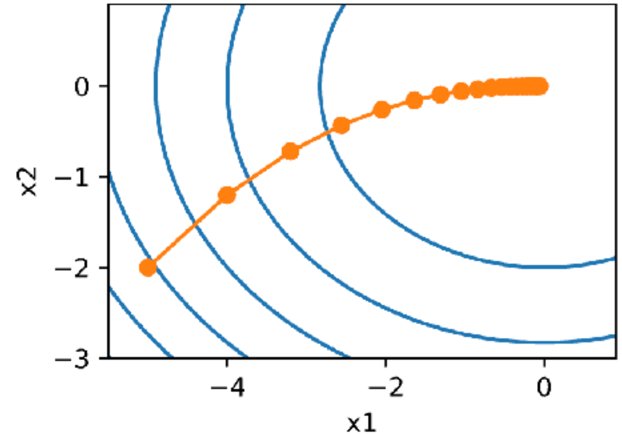
$$\mathbf{x}_t = \mathbf{x}_{t-1} - \eta_t \nabla \ell_{t_i}(\mathbf{x}_{t-1})$$

- Compare to gradient descent

$$\mathbf{x}_t = \mathbf{x}_{t-1} - \eta \nabla f(\mathbf{x}_{t-1})$$

$$f(\mathbf{x}) = \frac{1}{n} \sum_{i=0}^{n} \ell_i(\mathbf{x})$$



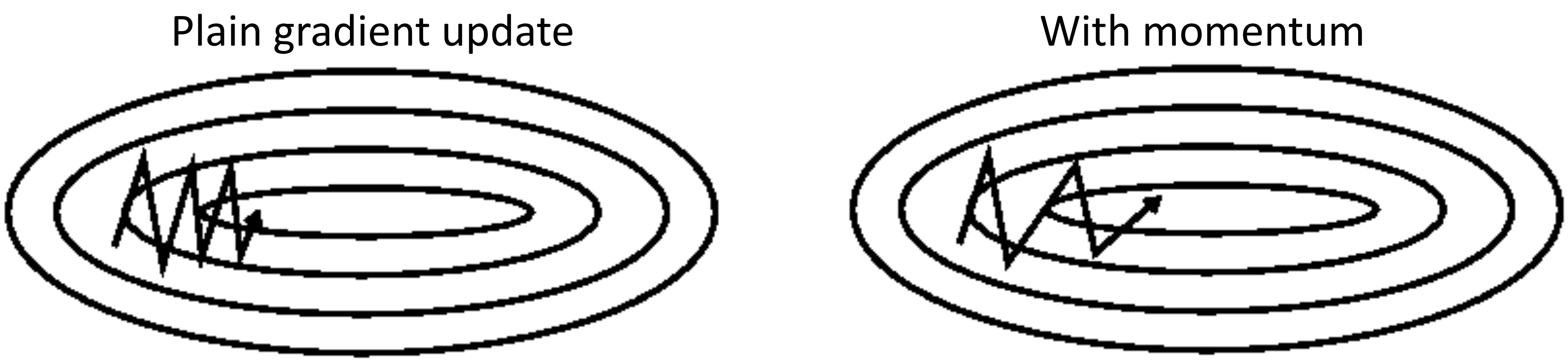


Minibatch Stochastic Gradient Descent

- Instead of full gradient, evaluate and update on random minibatch of data samples B_t

$$x_{t+1} = x_t - \frac{\eta}{|B_t|} \sum_{t_n \in B_t} \nabla \ell_{t_n}(x_t)$$

Momentum Method



- The momentum method maintains a running average of all gradients until the *current* step

$$v_{t+1} = \beta v_t - \eta \nabla \ell(x_t)$$
$$x_{t+1} = x_t + v_t$$

  - Typical β value is 0.9
- The running average steps
  - Get longer in directions where gradient retains the same sign
  - Become shorter in directions where the sign keeps flipping

AdaGrad

- AdaGrad (Duchi, Hazan, and Singer 2010) very popular adaptive method.

element-wise

$$G_{t+1} = G_t + \nabla \ell(x_t)^2$$

$$x_{t+1} = x_t - \eta \frac{1}{\sqrt{G_{t+1} + \epsilon}} \nabla \ell(x_t)$$

- Benefits:
  - AdaGrad does not require tuning learning rate η
  - Actual learning rate will decrease
  - Can drastically improve over SGD

RMSProp

- Similar to AdaGrad, accumulate the squared gradients, but with running average
  - Adagrad denominator monotonically increase ==> diminishing updates for parameters
  - why not decay the denominator

$$G_{t+1} = \beta G_t + (1 - \beta) \nabla \ell(x_t)^2$$

element-wise

$$x_{t+1} = x_t - \eta \frac{1}{\sqrt{G_{t+1} + \epsilon}} \nabla \ell(x_t)$$

-

ADAM: RMSprop + Momentum

- RMS prop only considers a second-moment normalized version of the current gradient
- ADAM utilizes a smoothed version of the *momentum-augmented* gradient
  - Considers both first and second moments

$$m_{t+1} = \beta_1 m_t - (1-\beta_1)\nabla \ell(x_t)$$

$$v_{t+1} = \beta_2 v_t + (1-\beta_2)(\nabla \ell(x_t))^2$$

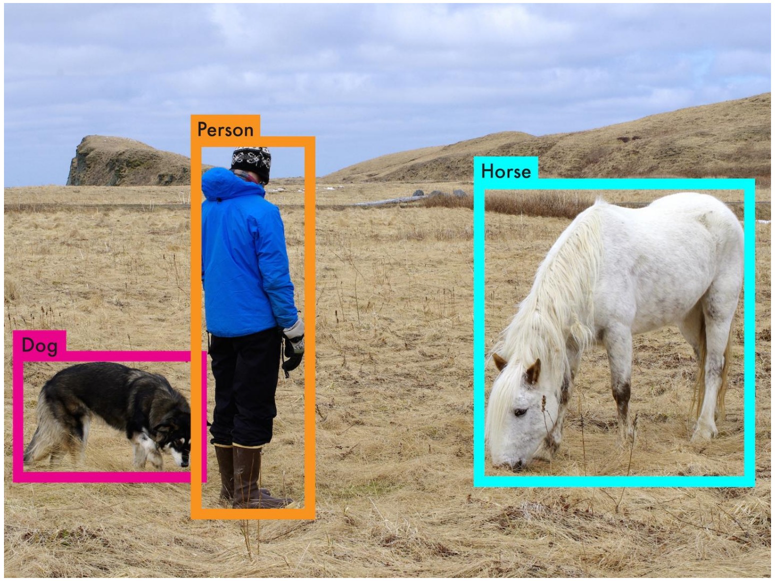
$$\hat{m}_{t+1} = \frac{m_{t+1}}{1-\beta_1^{t+1}}$$

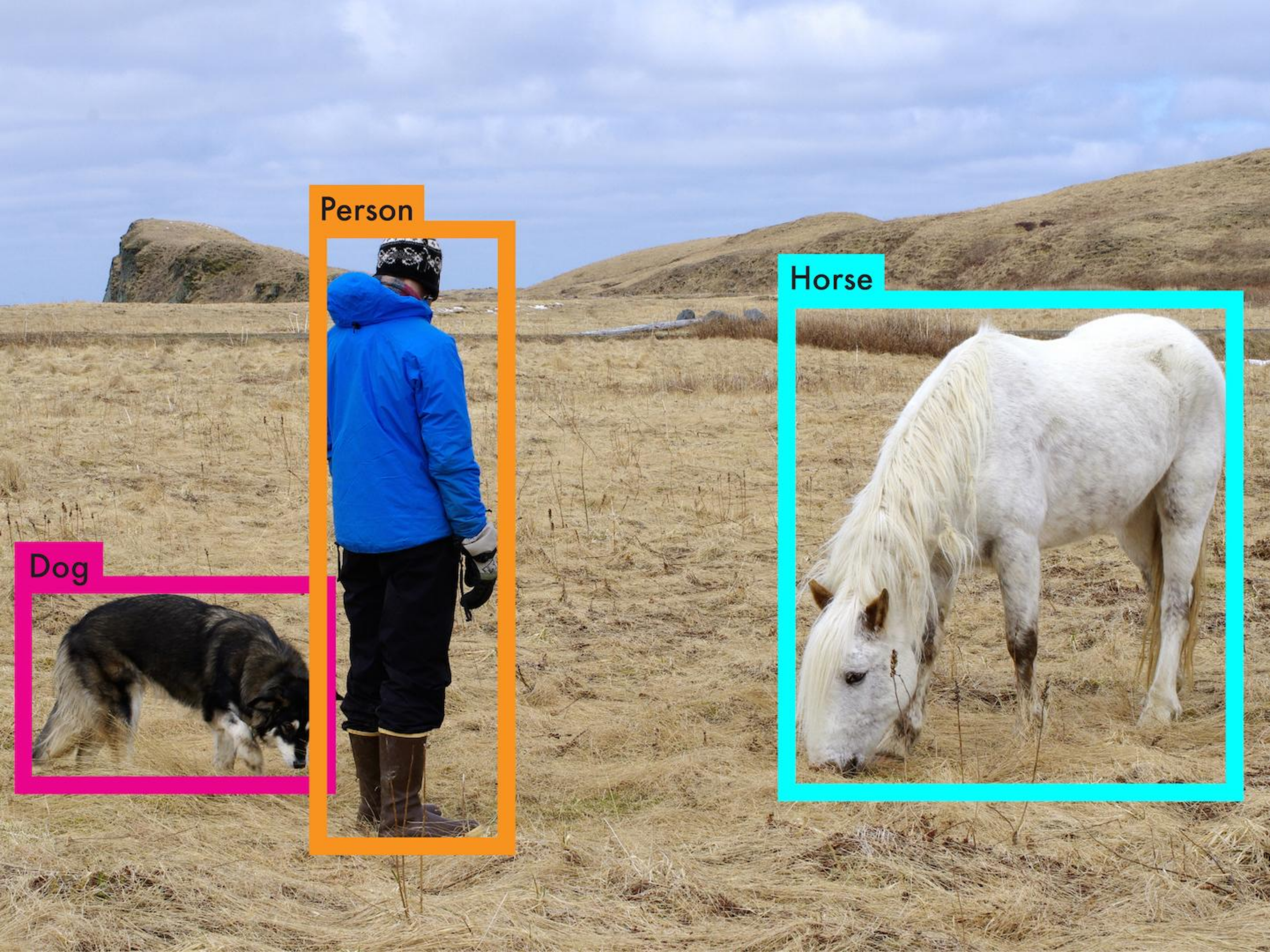
$$\hat{v}_{t+1} = \frac{v_{t+1}}{1-\beta_2^{t+1}}$$

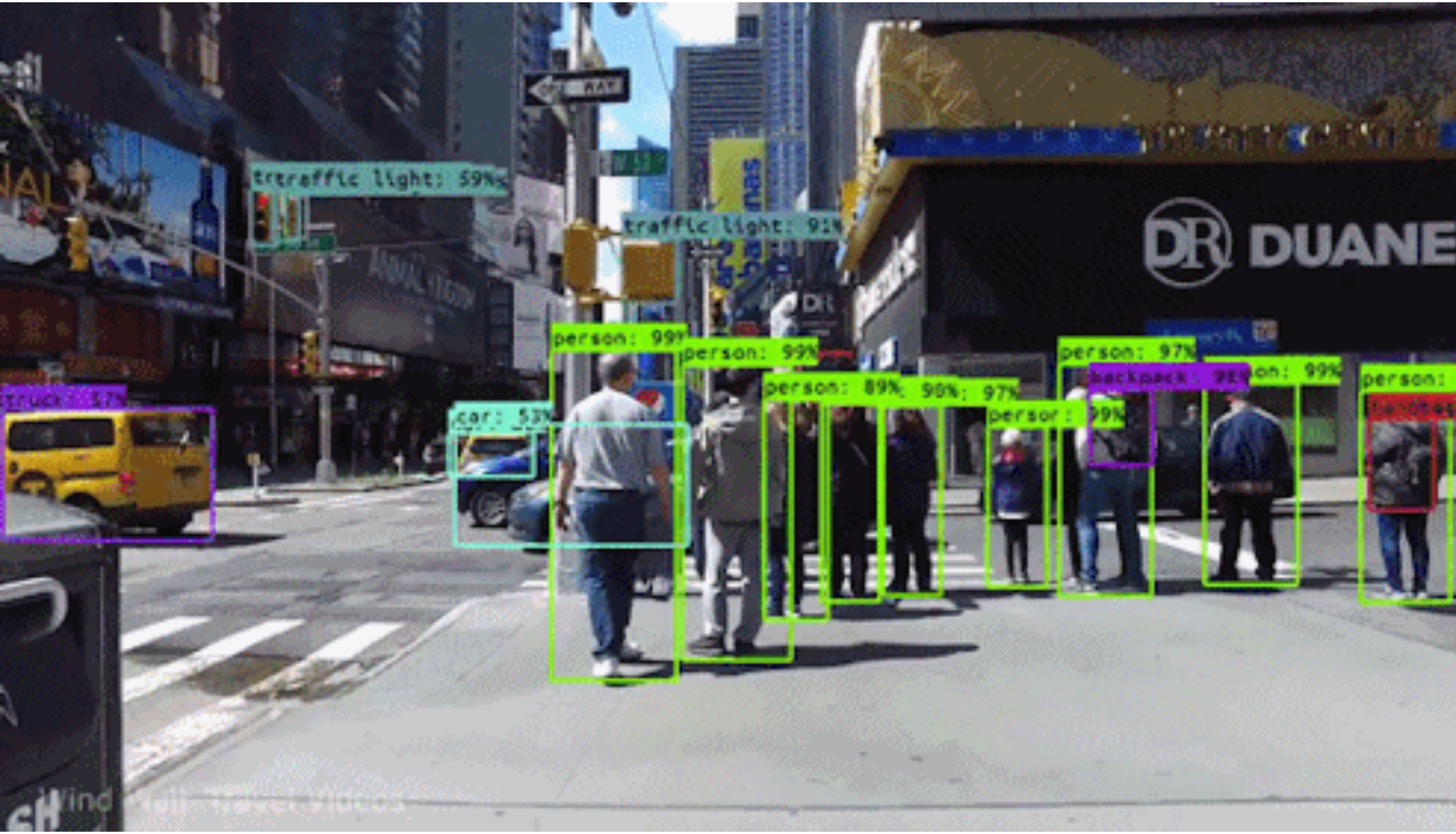
$$x_{t+1} = x_t - \frac{\eta}{\sqrt{\hat{v}_{t+1}}+\epsilon}\hat{m}_{t+1}$$

Image classification

Object Detection

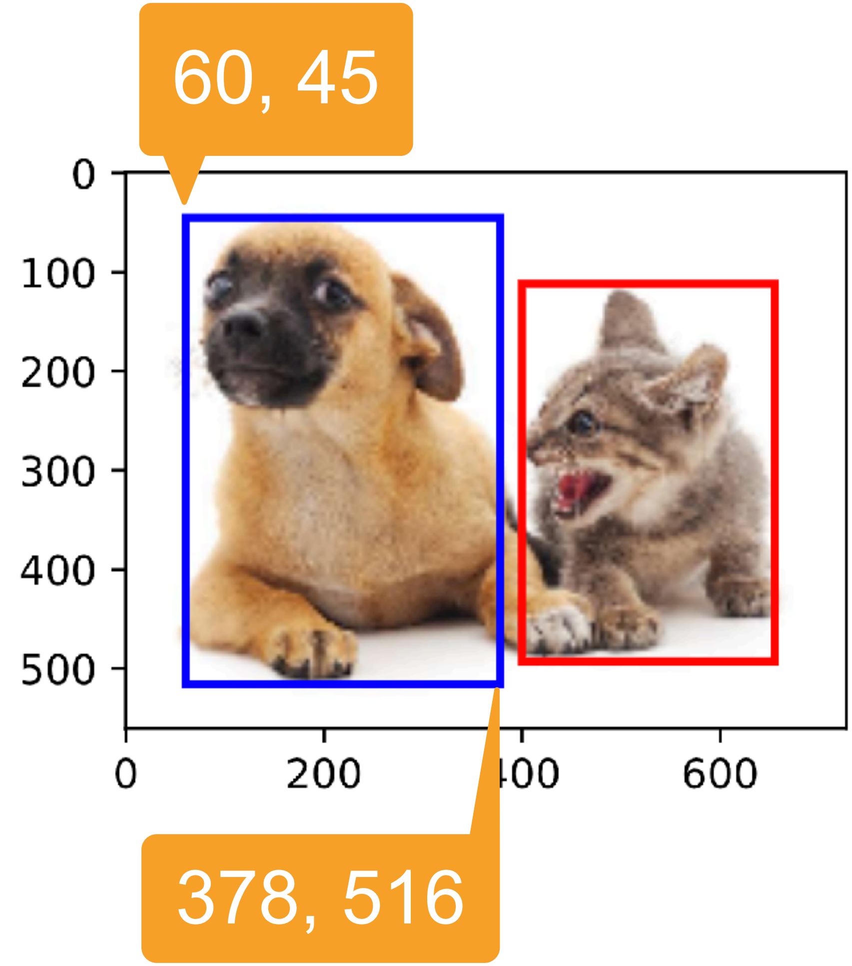



Person
Horse
Dog


traffic light: 59%
traffic light: 91%
person: 99%
person: 99%
person: 89%
person: 97%
backpack
car: 53%
DR DUANE

Locating the Object: Bounding Box

- A bounding box can be defined by 4 numbers,
  - (top-left x, top-left y, bottom-right x, bottom-right y)
  - (top-left x, top-right y, width, height)

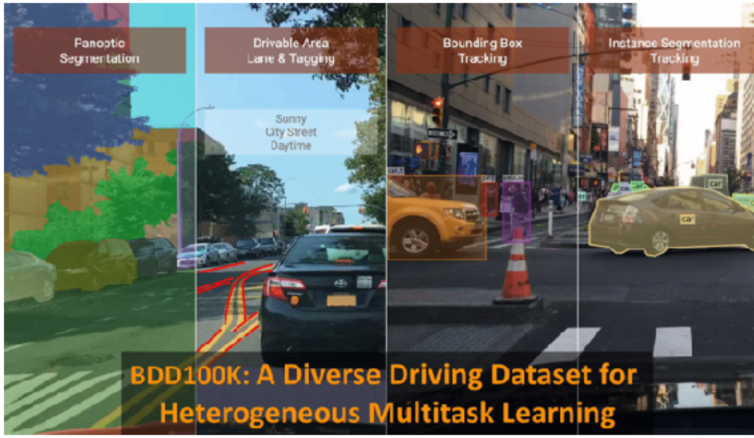


Object Detection Dataset

- Each row present an object
  - Image_filename, object_category, bounding box
- PASCAL VOC
  - 11.53k images, 27.45k objects, 20 classes
- COCO (cocodataset.org)
  - 80 object classes
  - 330K images
  - 1.5M objects

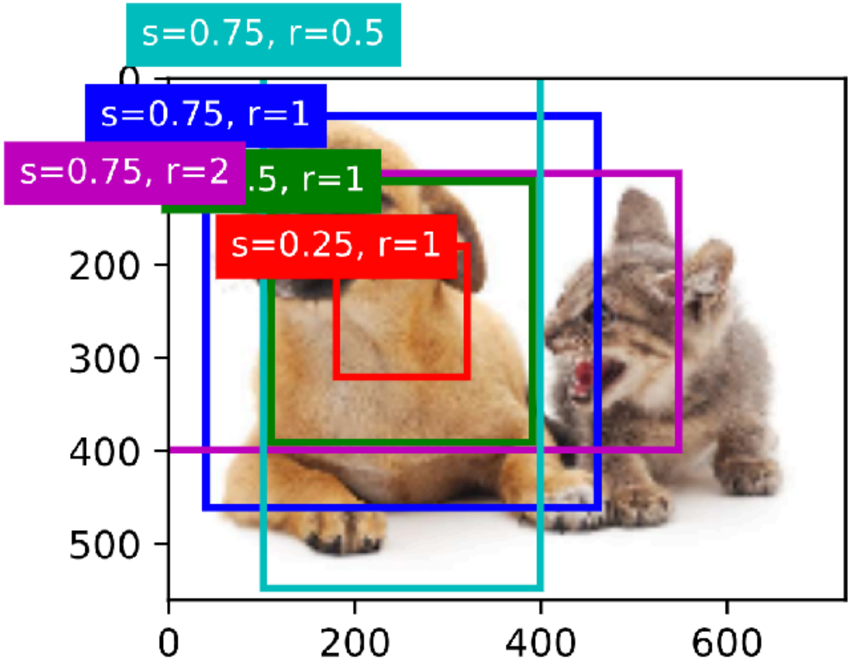
Object Detection Dataset

- Open Image (v6):
  - 9M images,
  - 1.9M images with 16M bounding boxes, 600 classes
  - Includes 3.3M visual relations (of 1466 types)
  - https://storage.googleapis.com/openimages/web/factsfigures.html
- BDD100k
  - 100k videos in driving scenario
  - https://github.com/bdd100k/bdd100k



Anchor Boxes

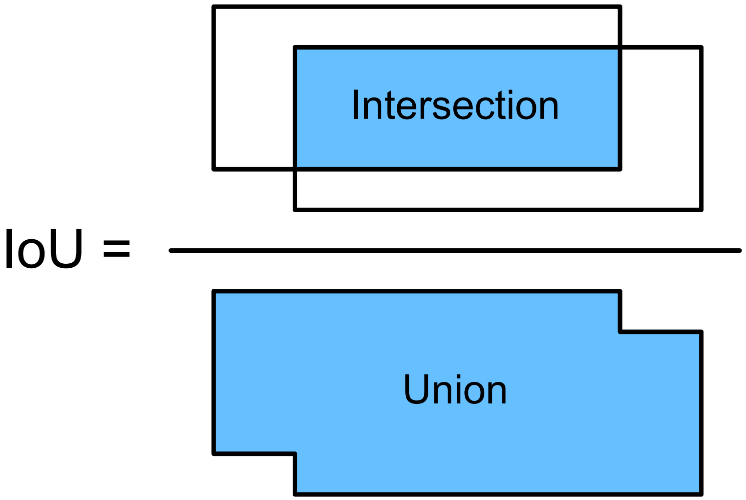
- A detection algorithm often
  - Proposes multiple regions, called anchor boxes
  - Predict if an anchor box contains an object
  - If yes, predict the offset from the anchor box to the ground truth bounding box



IoU - Intersection over Union

- IoU measures the similarity between two boxes
  - 0 means no-overlapping
  - 1 means identical
- It's an especial case of Jacquard index
  - Given sets *A* and *B*

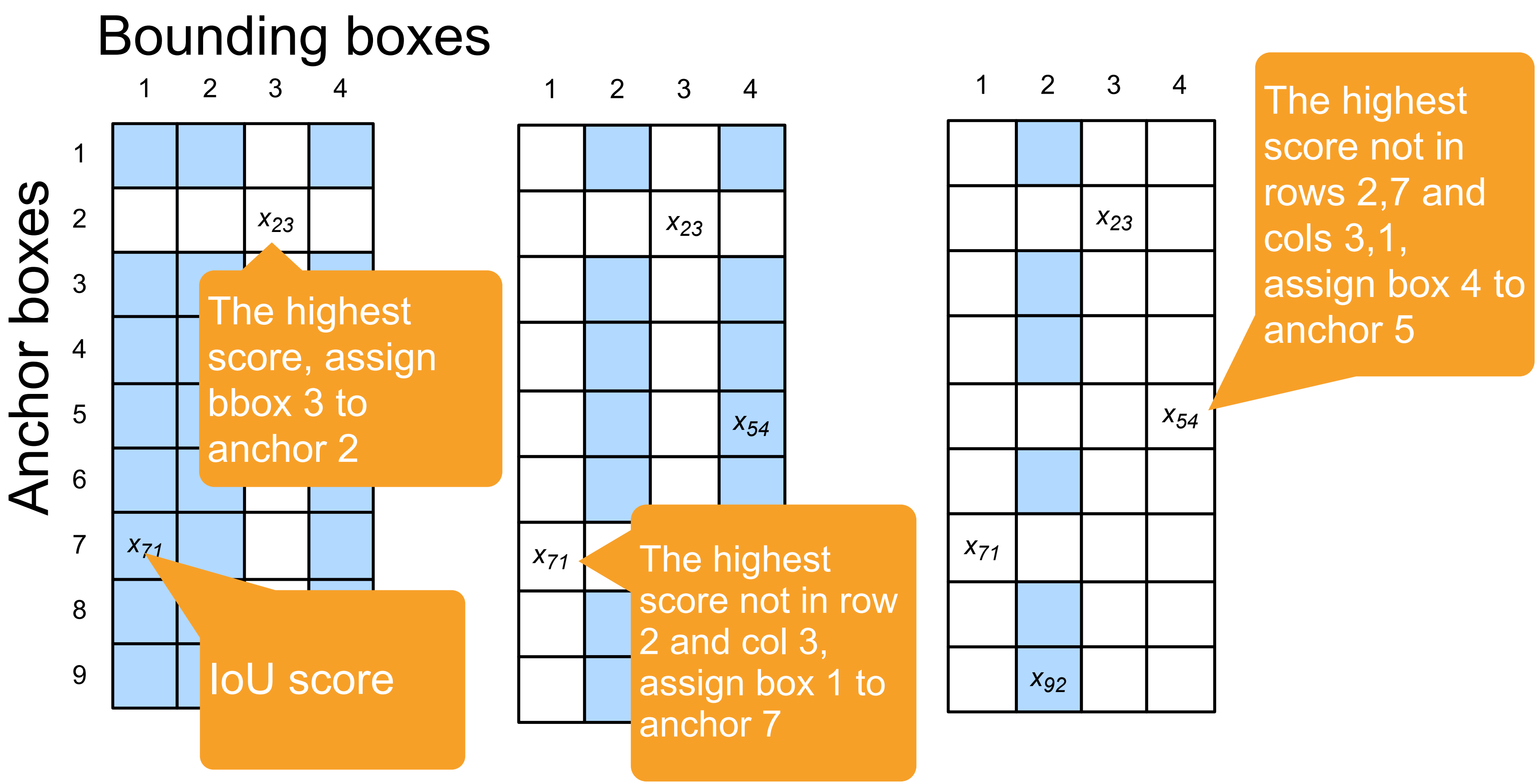
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$



Assign Labels to Anchor Boxes

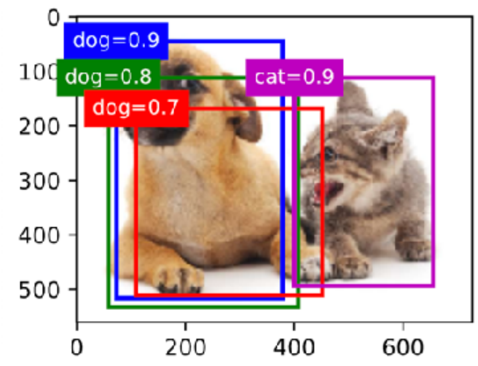
- Each anchor box is a training example
- Label each anchor box with
  - Background
  - Associate with a bounding box
- We may generate a large amount of anchor boxes
  - Leads to a large portion of negative examples

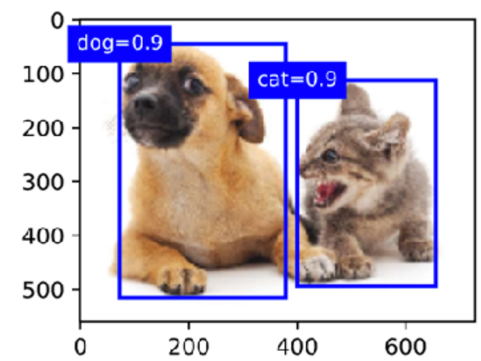
Assign Labels to Anchor Boxes



Output with non-maximum suppression (NMS)

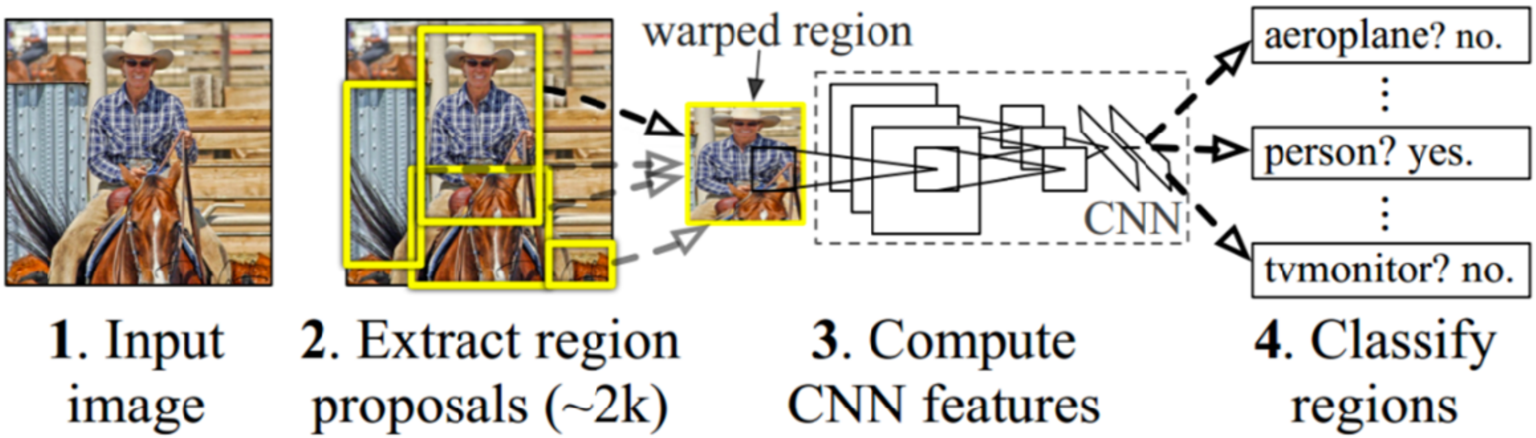
- Each anchor box generates one bounding box prediction
- Select the one with the highest score (not background)
- Remove all other predictions with IoU > θ compared to the selected one
- Repeat until all are selected or removed





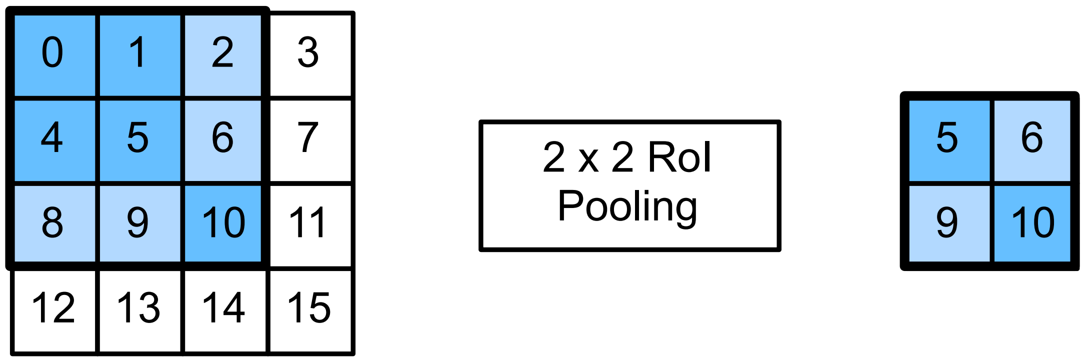
Region-based CNNs

R-CNN



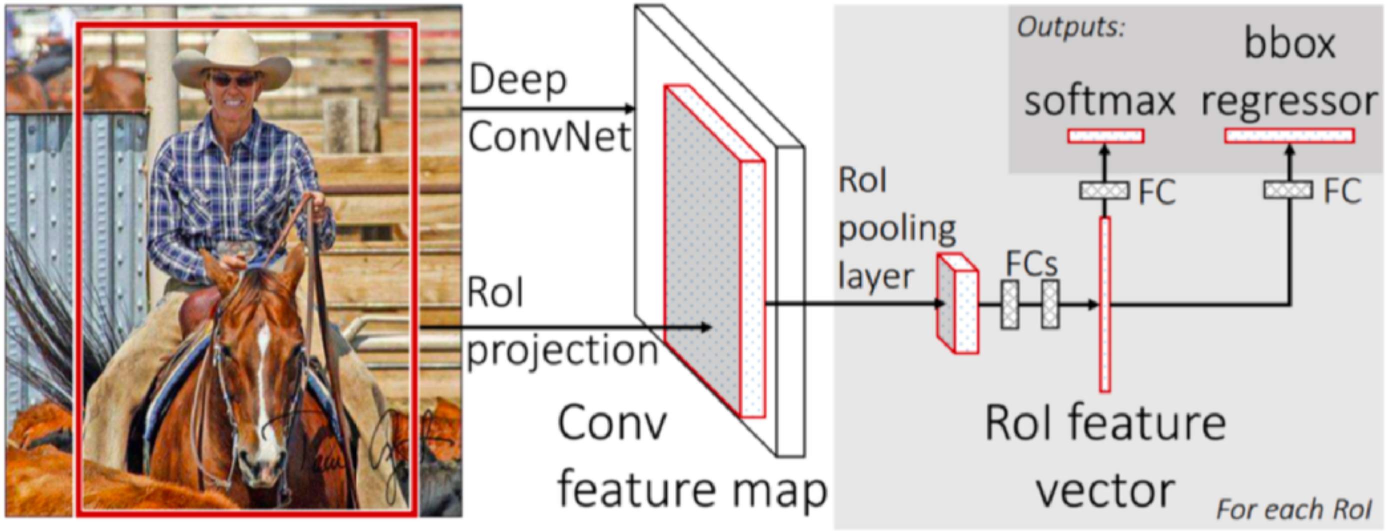
- Select anchor boxes with a heuristic algorithm
- Use a pre-trained networks to extract features for each anchor box
  - Adding classifier layer
  - and regression layer to predict bounding boxes

Region of Interest (RoI) Pooling



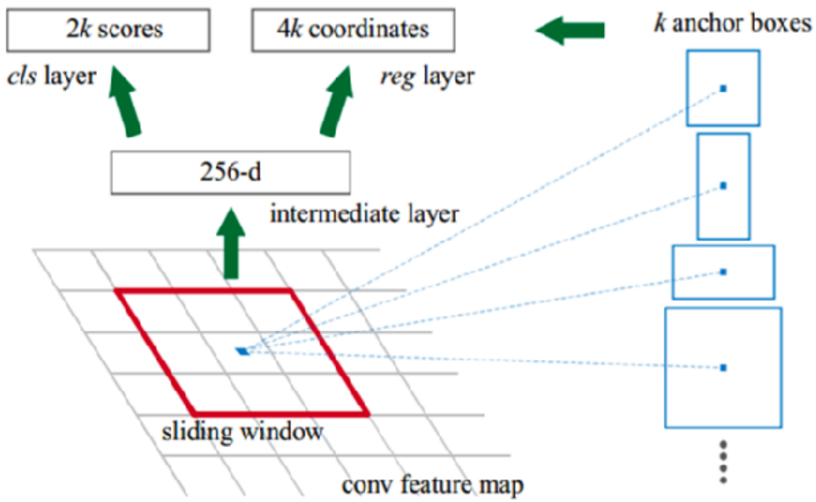
- Given an anchor box, uniformly cuts it into *n x m* blocks, output the maximal value in each block
- Returns *nm* values for each anchor box
- A special case of maxpooling

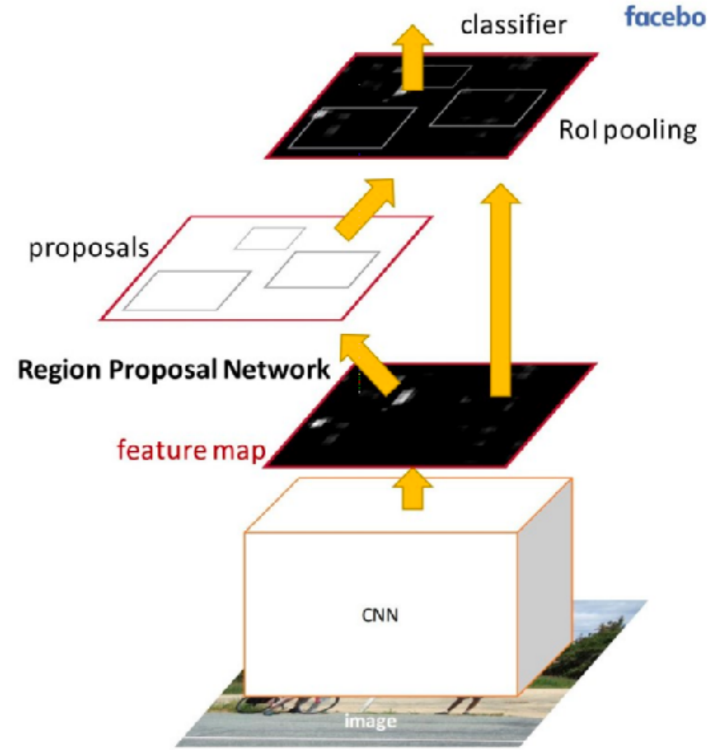
Fast RCNN



- A CNN to extract features
- Siding windows on the feature maps
- RoI pooling returns fixed length feature for each anchor box

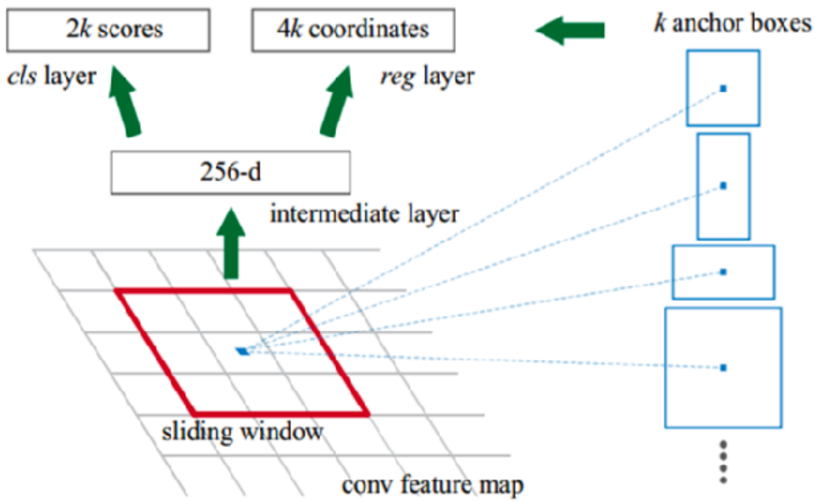
Faster R-CNN

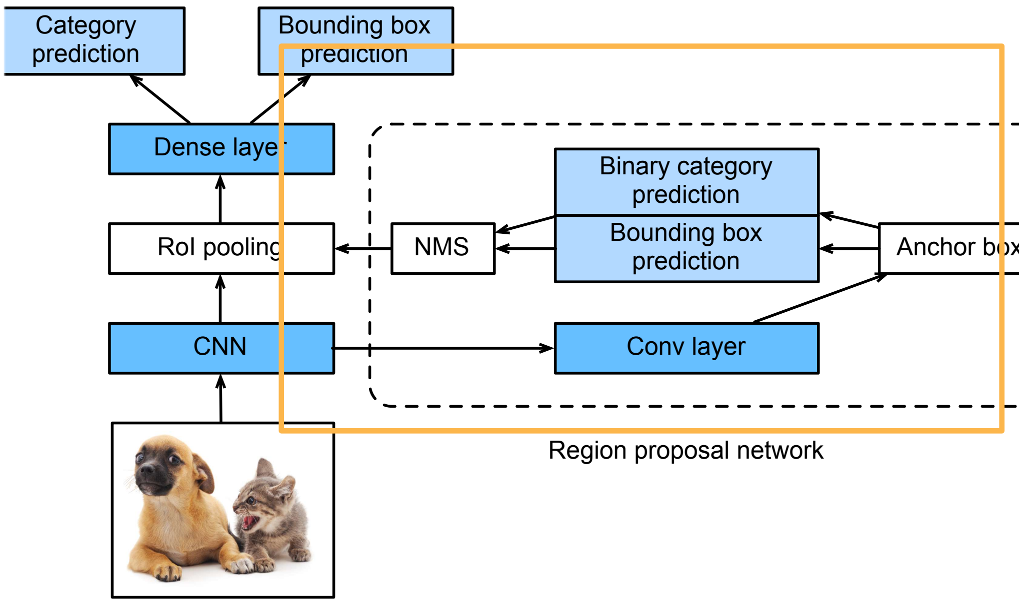




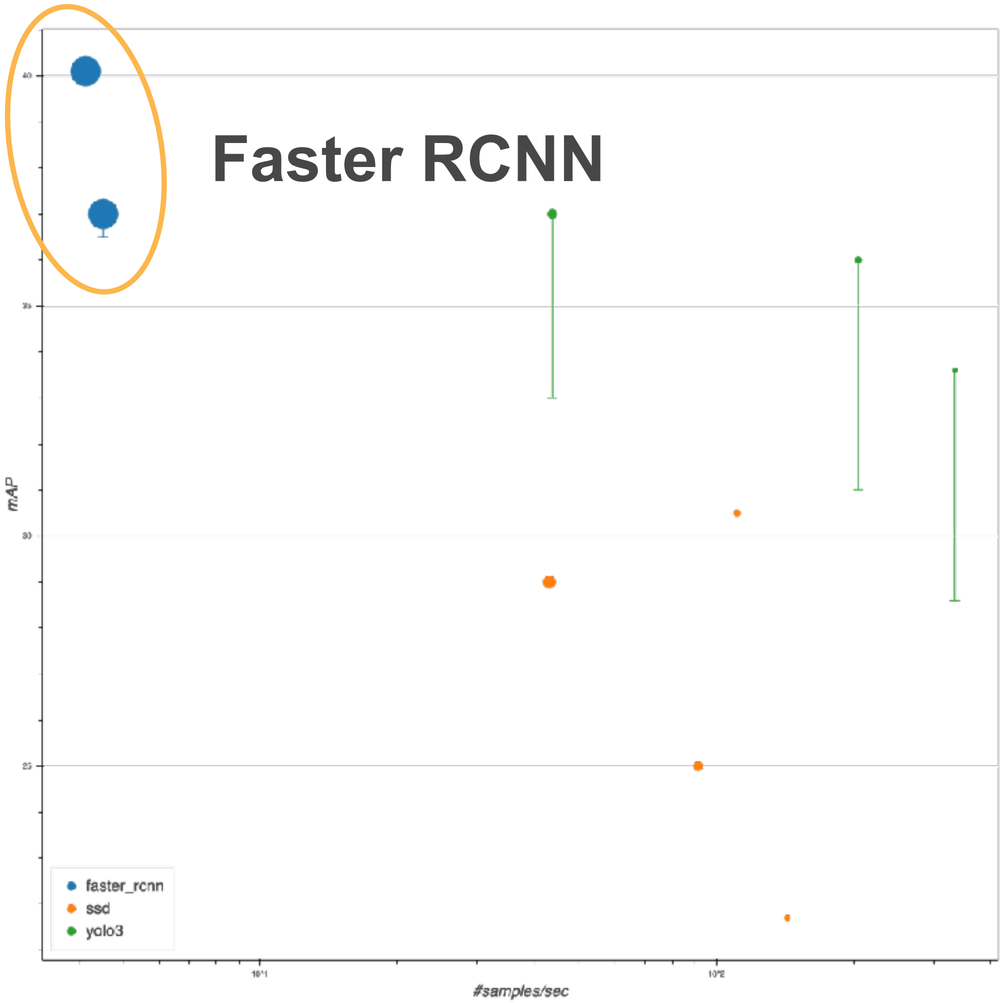
- Use a region proposal network to replace select search for high quality anchor boxes

Faster R-CNN





- Use a region proposal network to replace select search for high quality anchor boxes

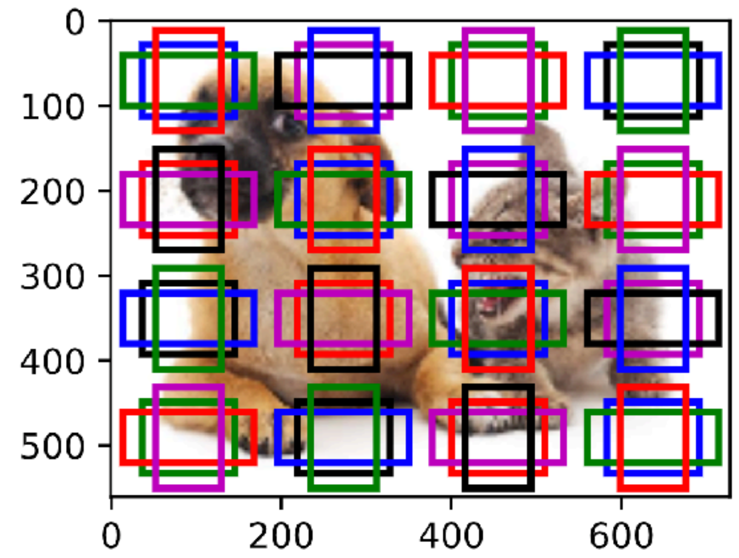


https://gluon-cv.mxnet.io/model_zoo/detection.html

Single Shot Multibox Detection (SSD)

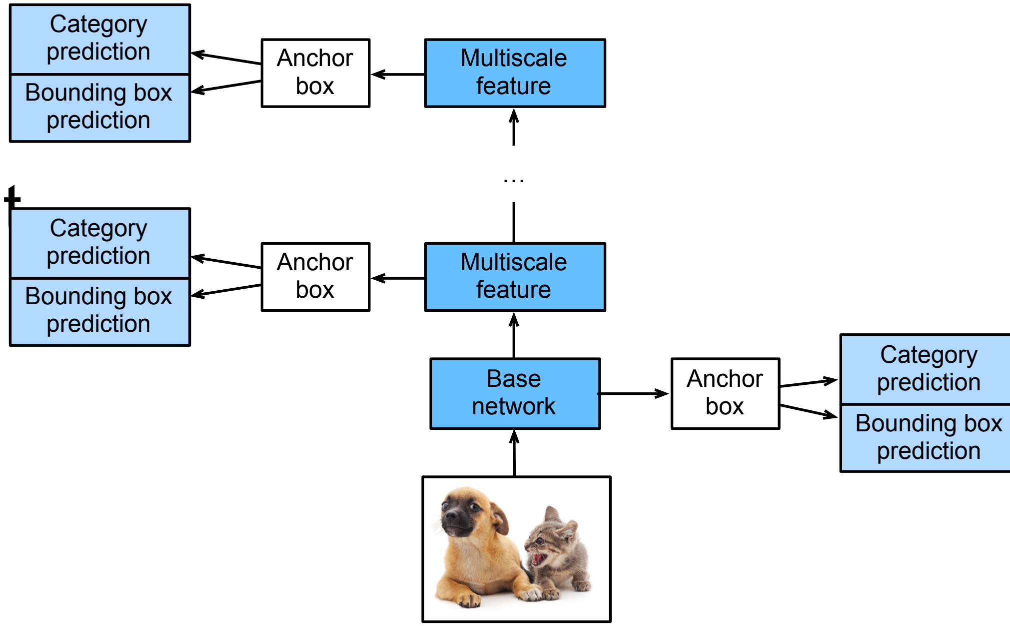
Generate Anchor Boxes

- For each pixel, generate multiple anchor boxes centered at this pixel
- Given *n* sizes $s_1, \ldots, s_n$ and *m* ratios $r_1, \ldots, r_m$, generate *n+m-1* anchor boxes $(s_1, r_1), (s_2, r_1), \ldots, (s_n, r_1), (s_1, r_2), \ldots, (s_1, r_m)$



SSD Model

- A base network to extract feature, followed by conv-blocks to halve width and height
- Generate anchor boxes at each sale
  - Bottom for small objects and
    top for large objects
- Predict class and bounding box for each anchor box



You Only Look Once (YOLO)

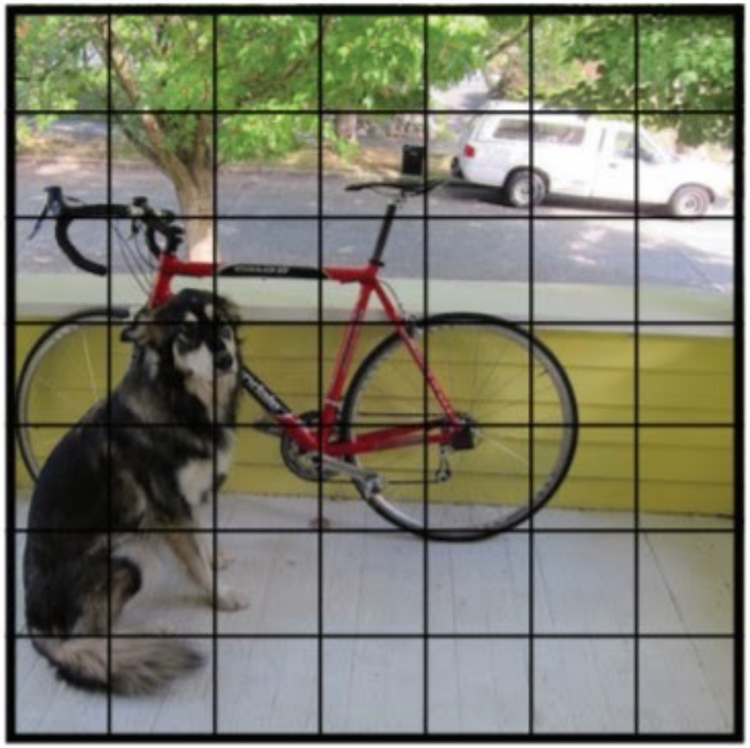
Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi

CVPR 2016

by J Redmon · 2016 · Cited by 21627

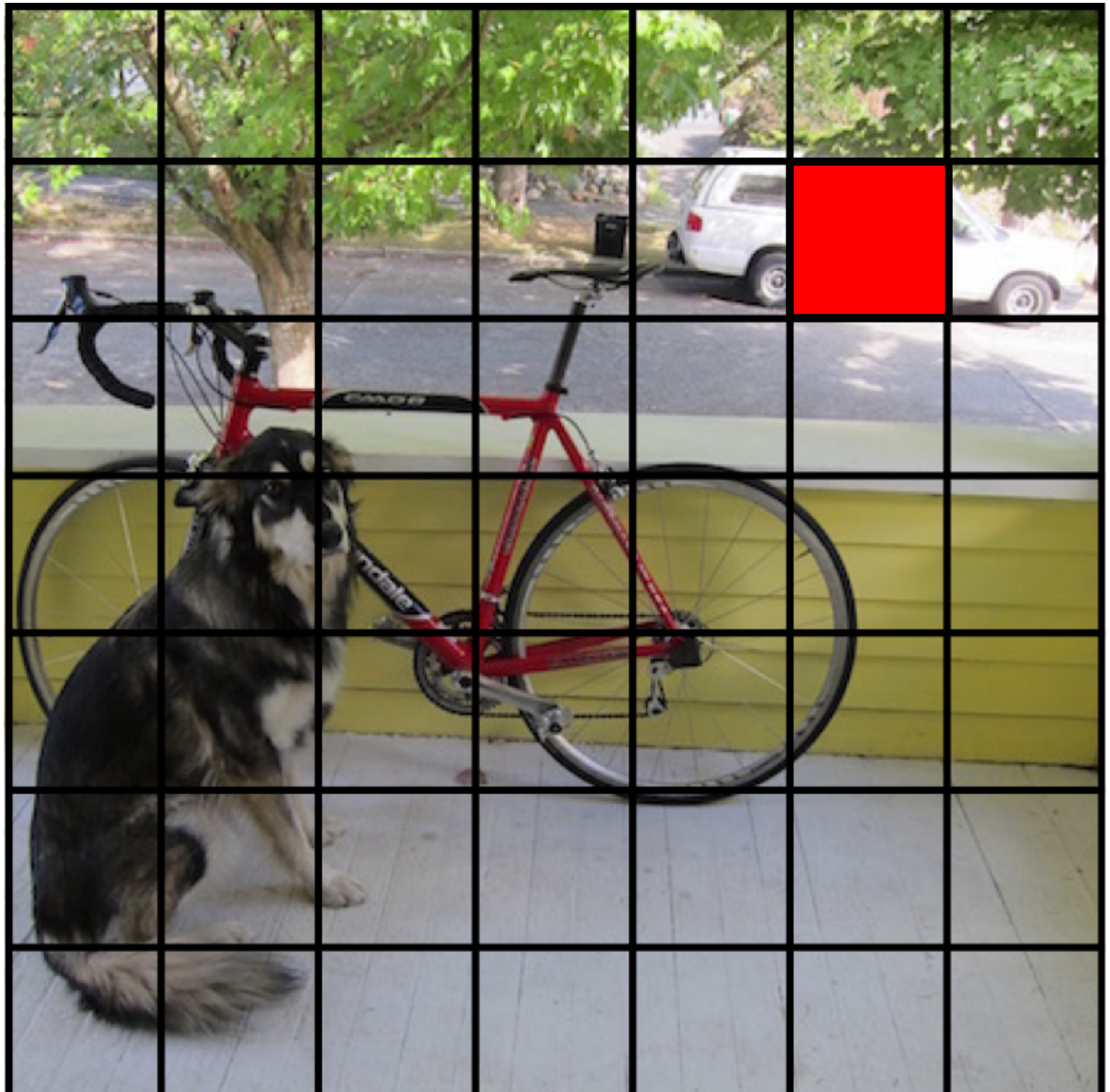
YOLO

- Anchor boxes are highly overlapped in SSD
- YOLO cuts the input image uniformly into *S* x *S* anchor boxes
- Each anchor box predicts *B* bounding boxes



S × S grid on input

Each cell predicts boxes and confidences: P(Object)



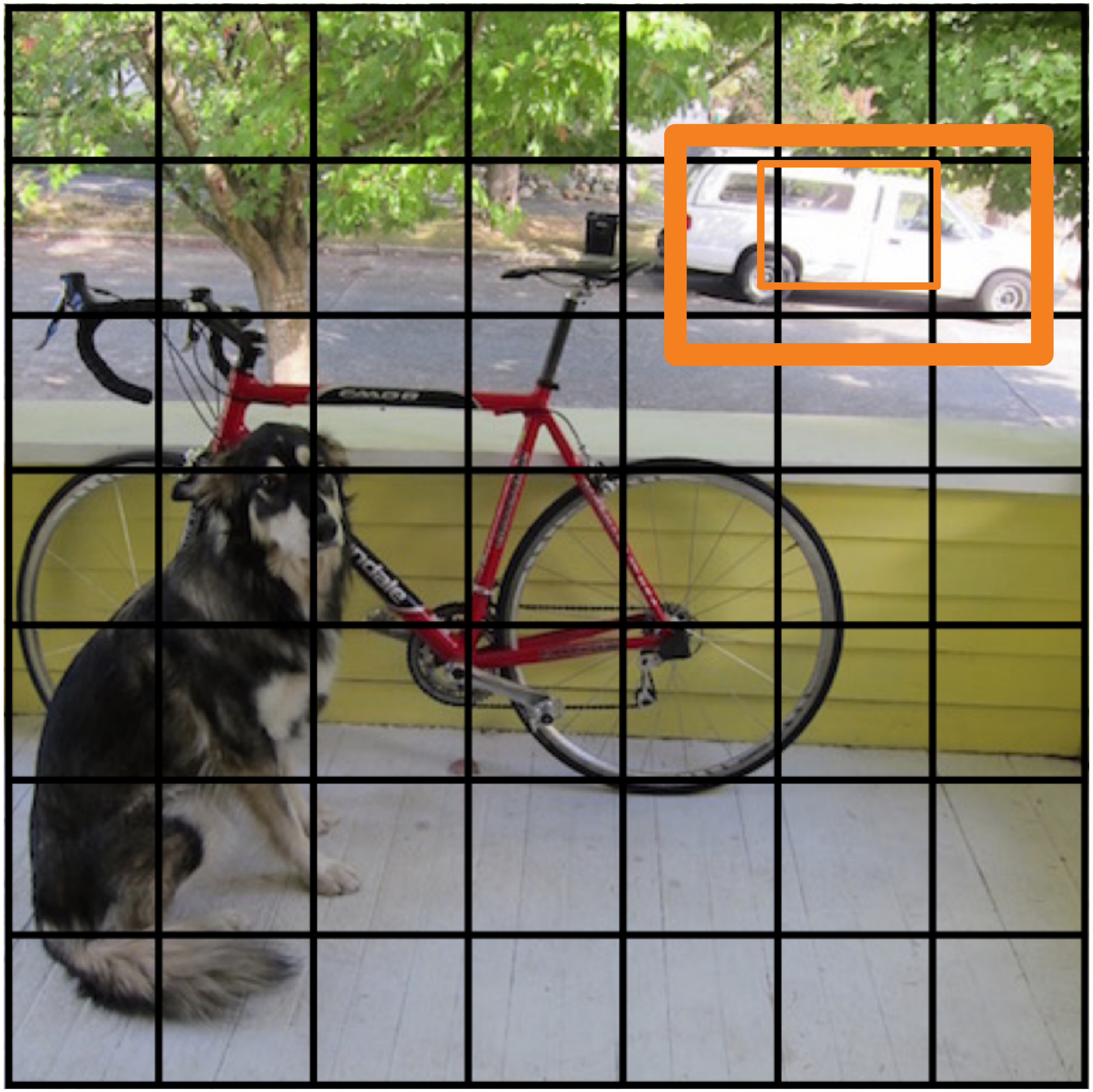
Redmon. et al. 2016

Each cell predicts boxes and confidences: P(Object)

normalize
(x,y,w,h)
P

Redmon. et al.
2016

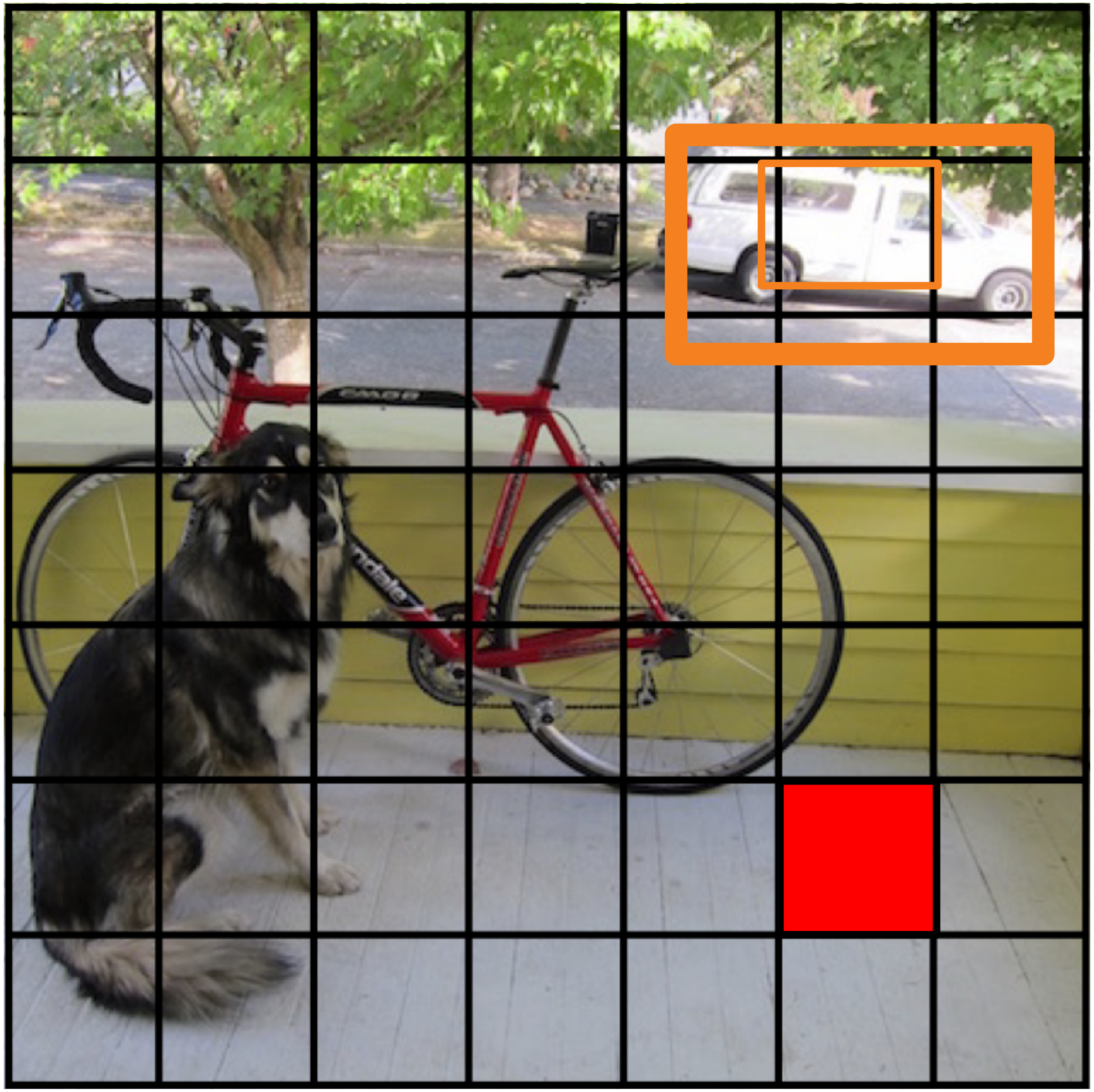
Each cell predicts boxes and confidences: P(Object)



(x,y,w,h)
P

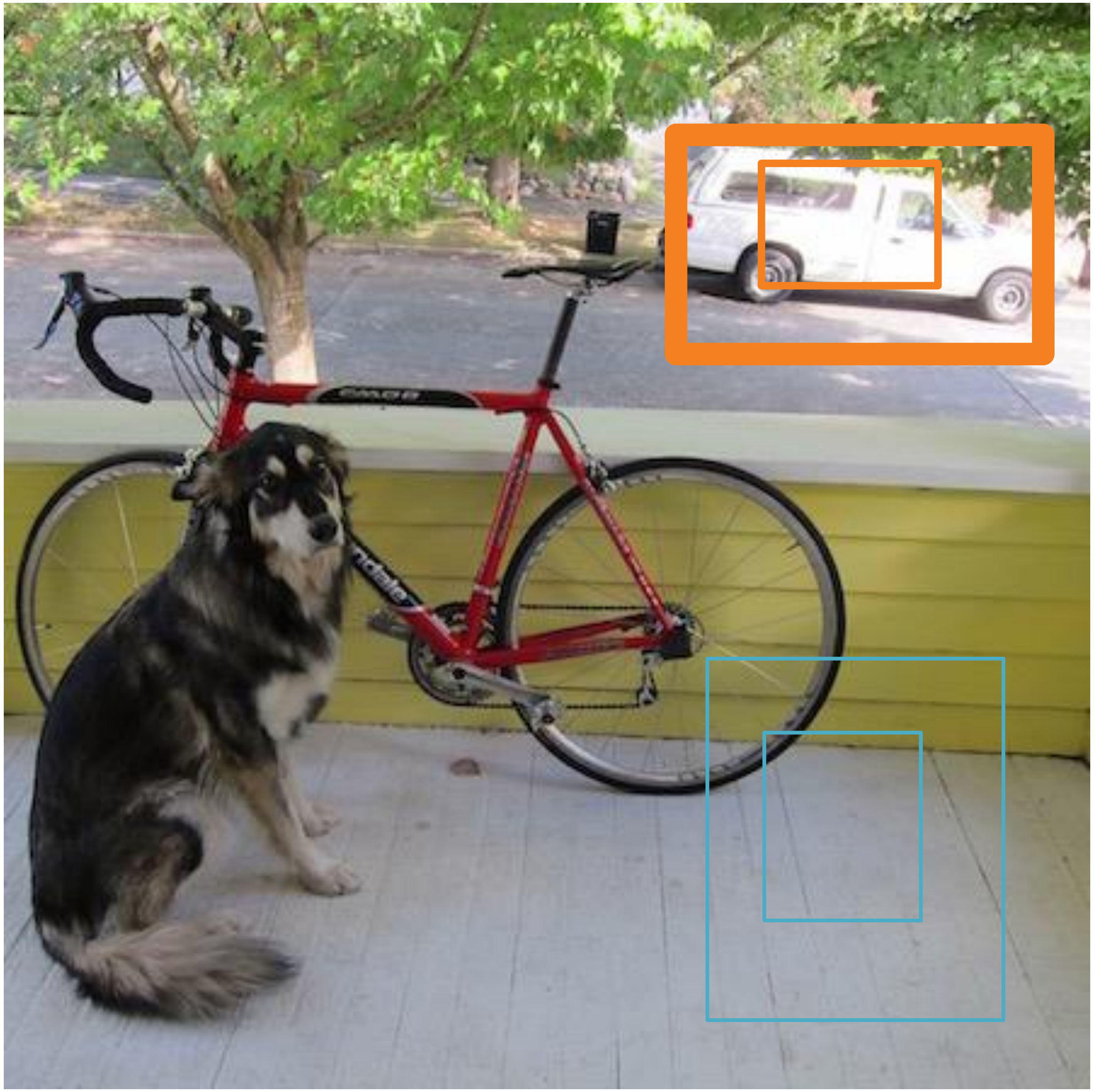
Redmon. et al. 2016

Each cell predicts boxes and confidences: P(Object)



Redmon. et al.
2016

Each cell predicts boxes and confidences: P(Object)



(x,y,w,h)
P

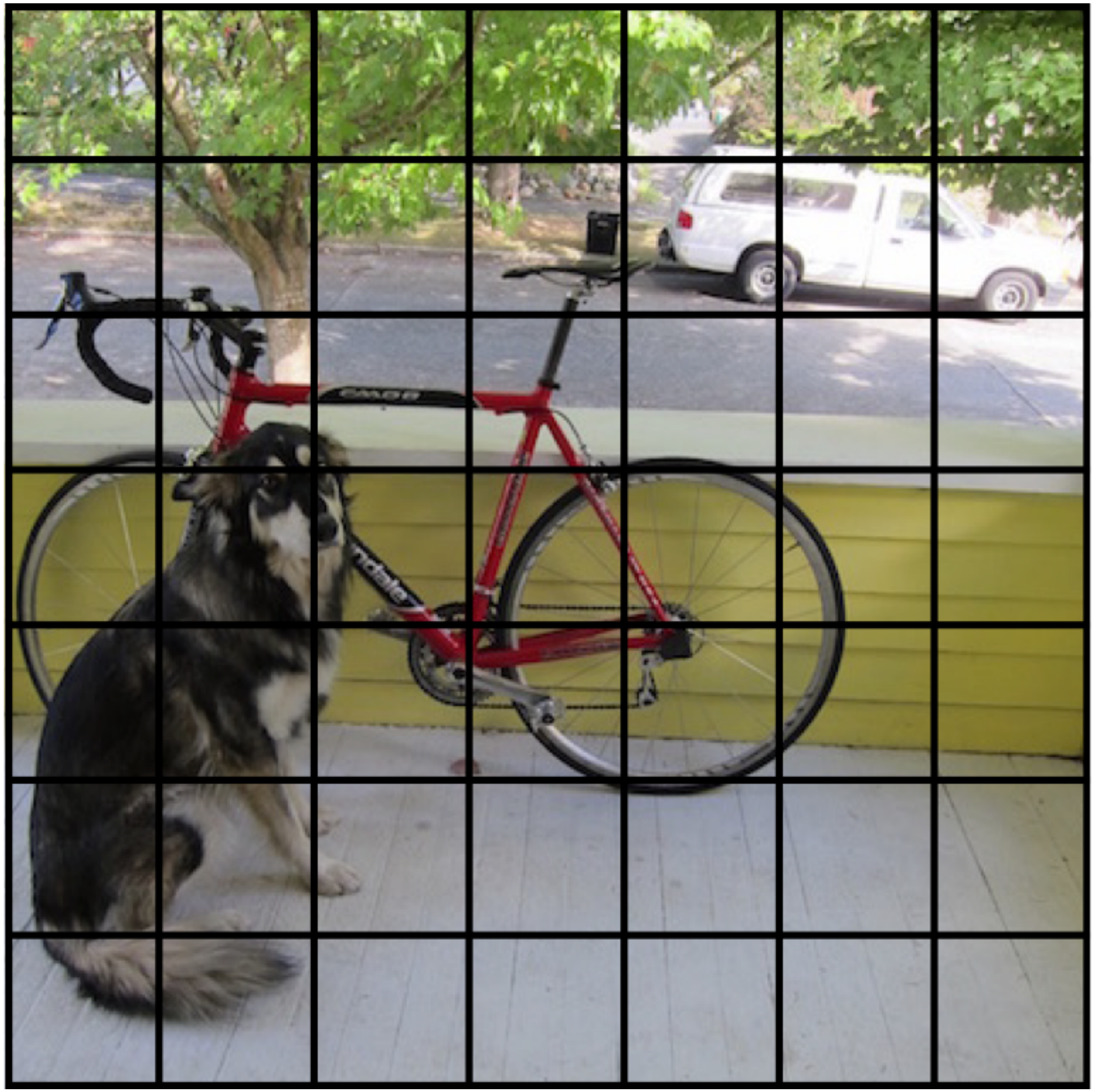
(x,y,w,h)
P

Redmon. et al.
2016

Each cell predicts boxes and confidences: P(Object)

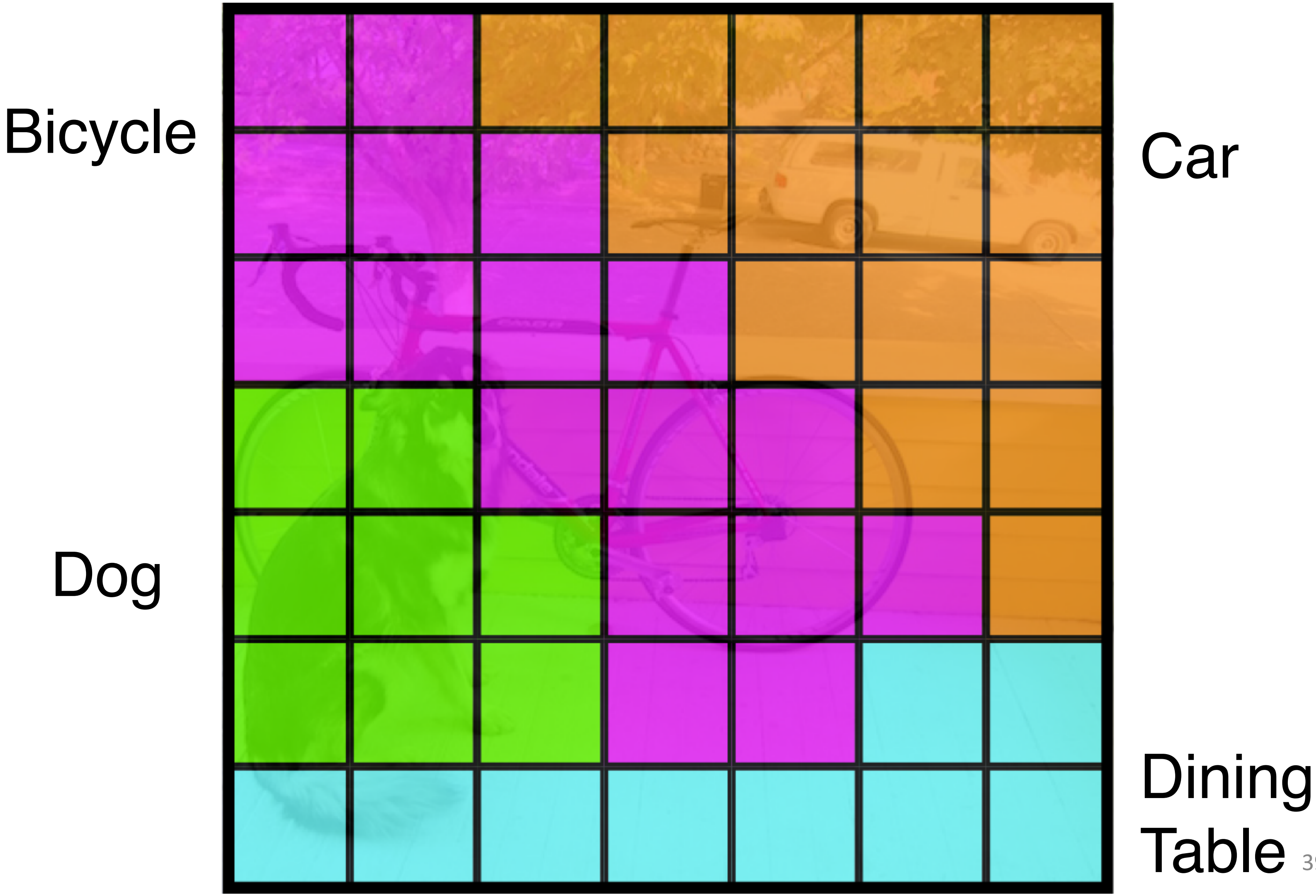
Redmon. et al.
2016

Each cell also predicts a class probability.

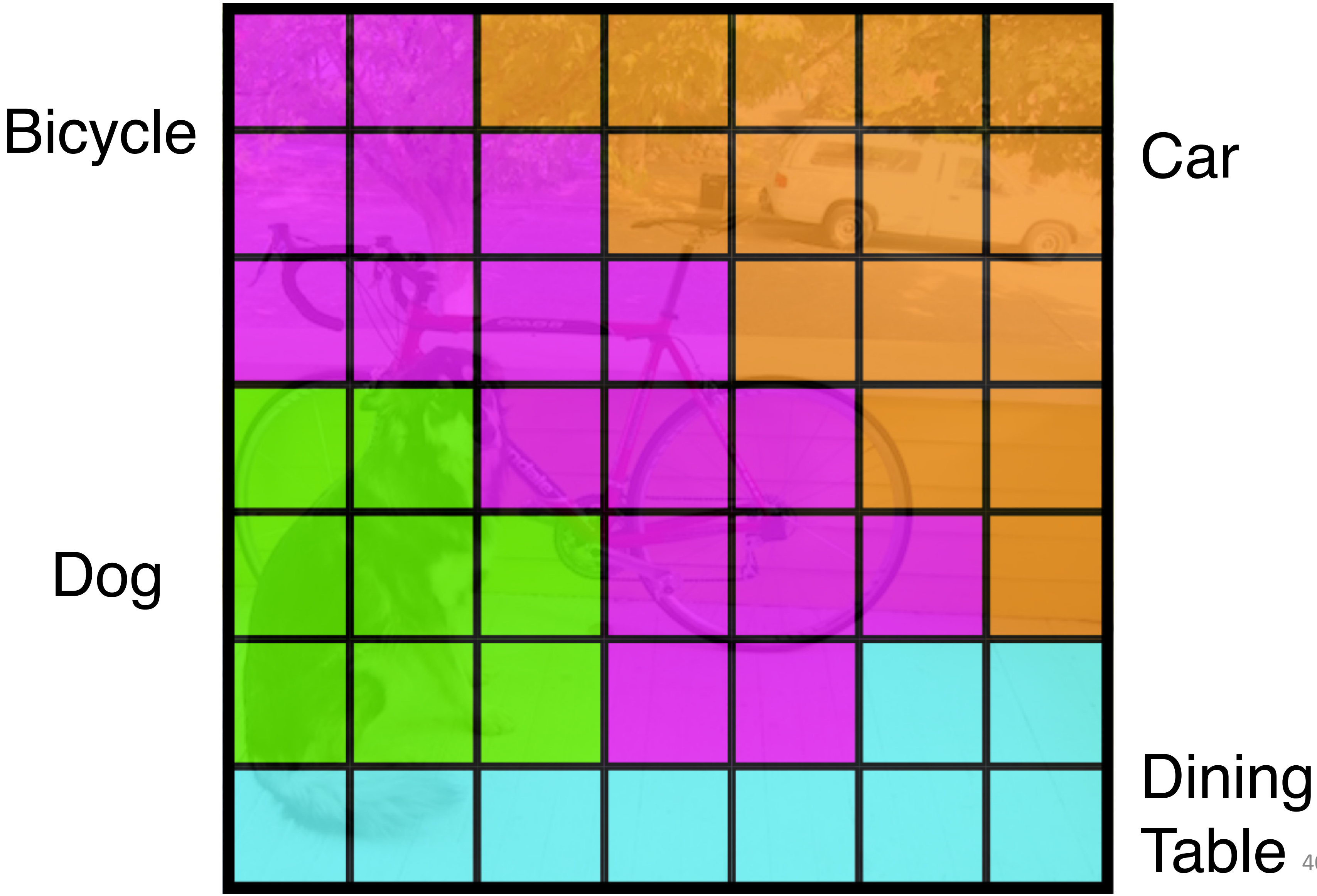


Redmon. et al. 2016

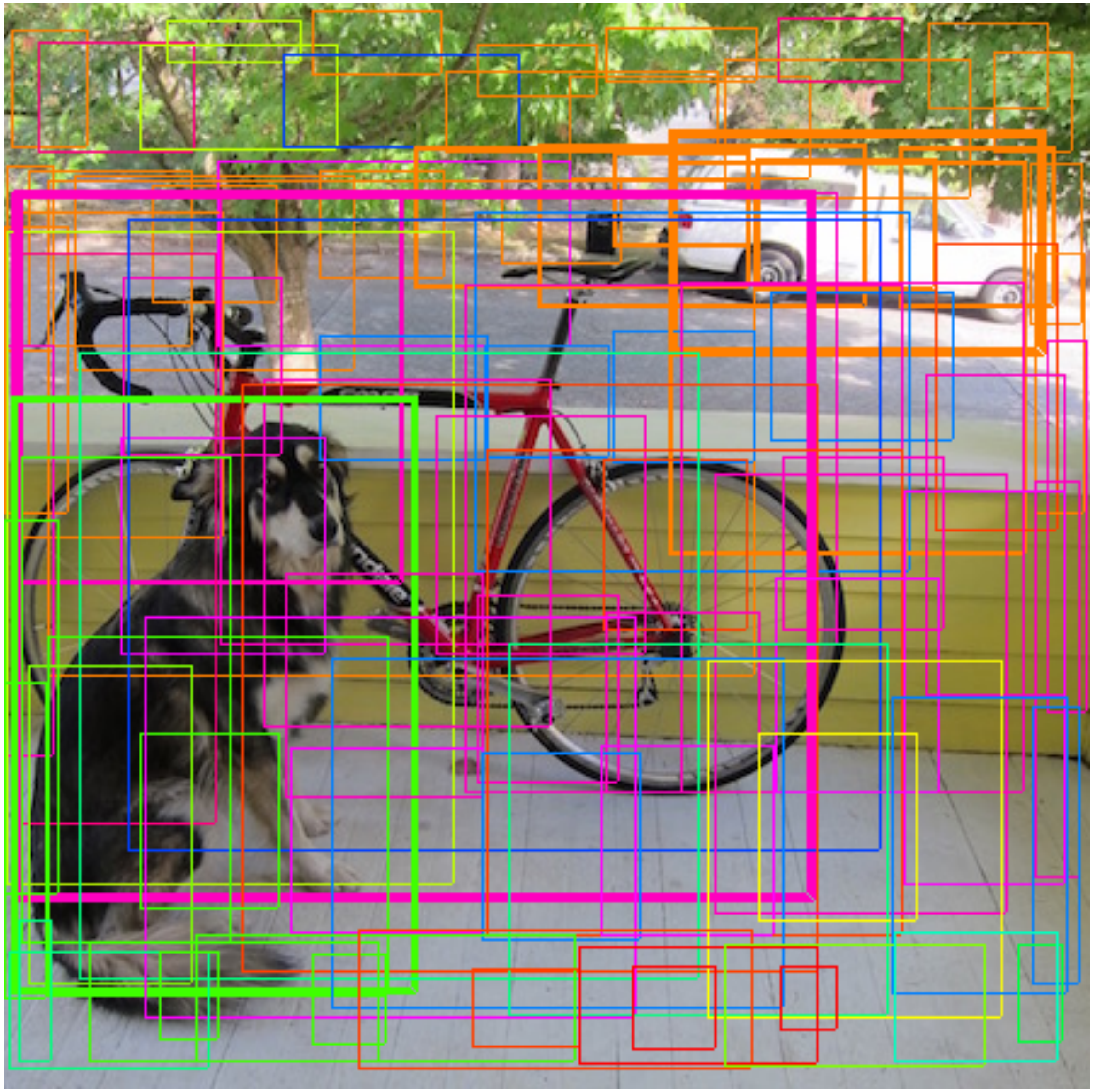
Each cell also predicts a class probability.



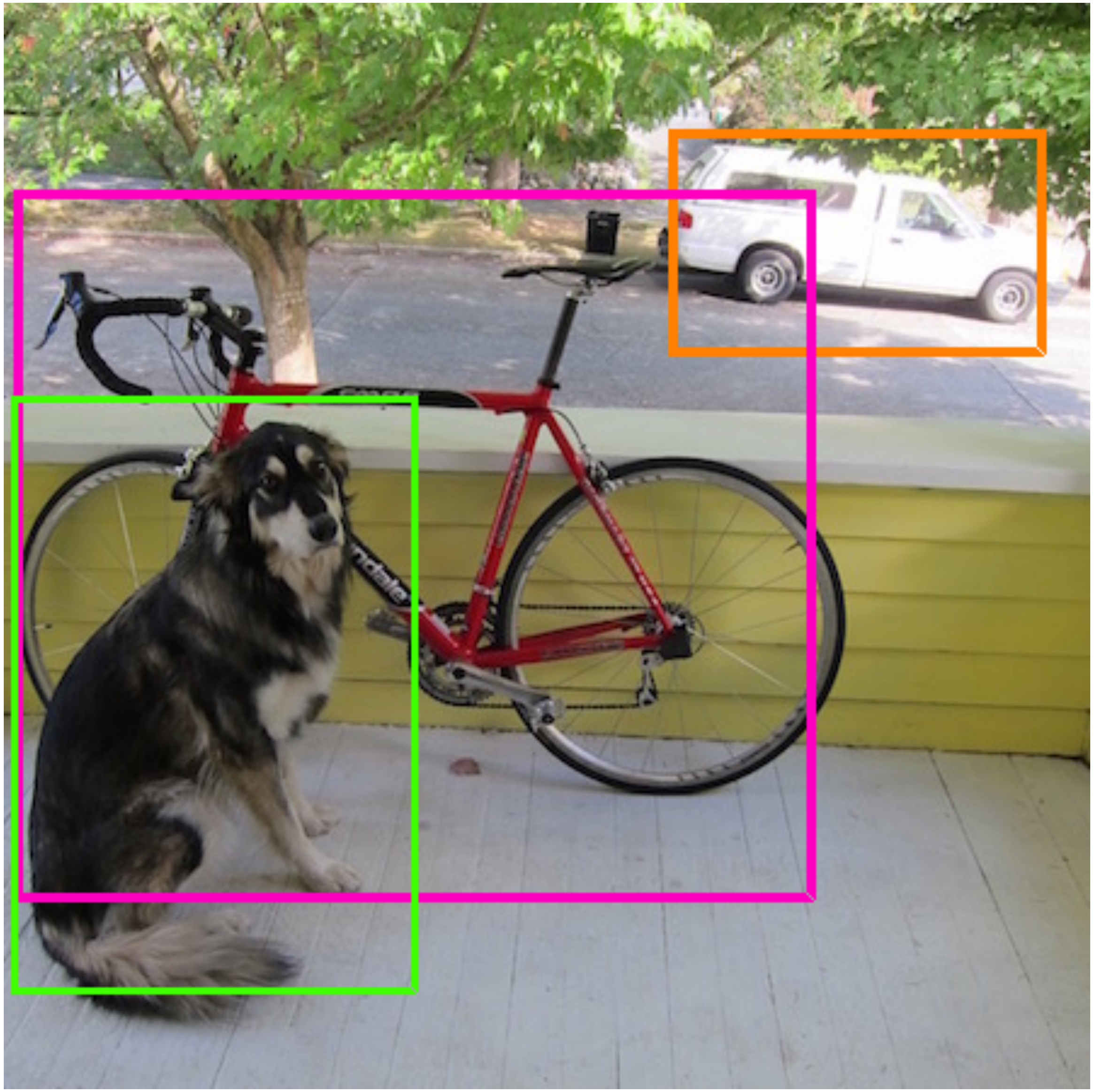
Conditioned on object: P(Car | Object)



Then we combine the box and class predictions.



Finally we do NMS and threshold detections



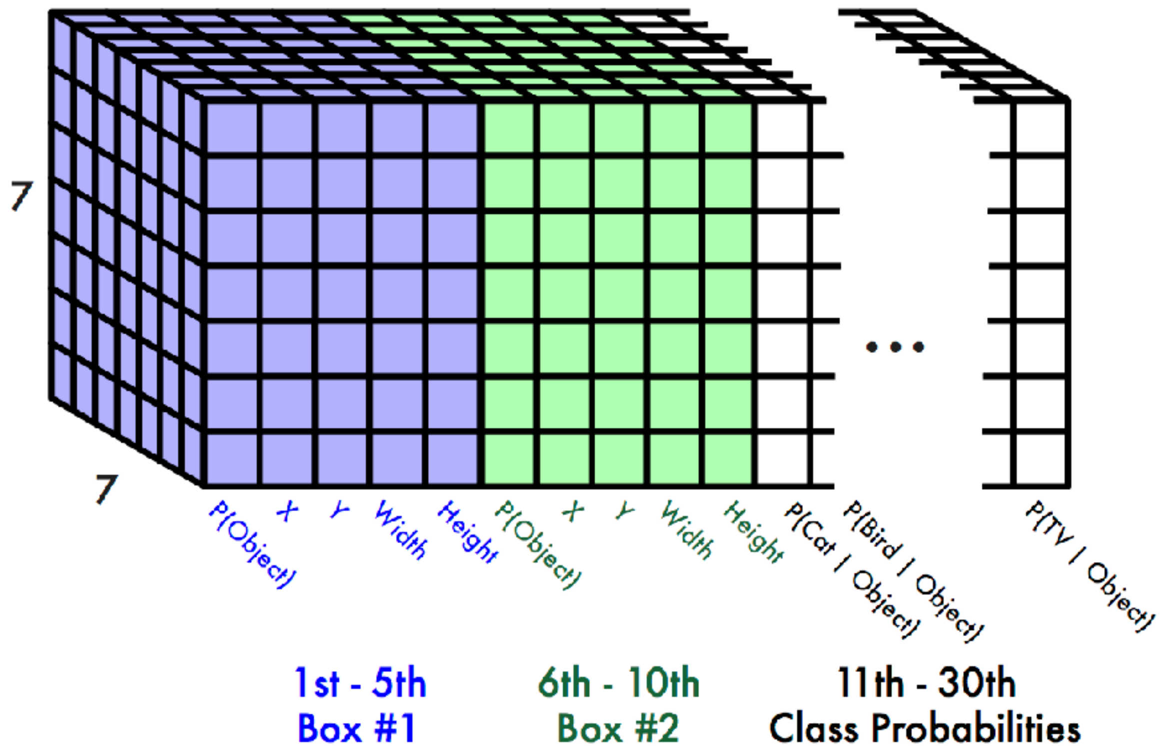
The output

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities

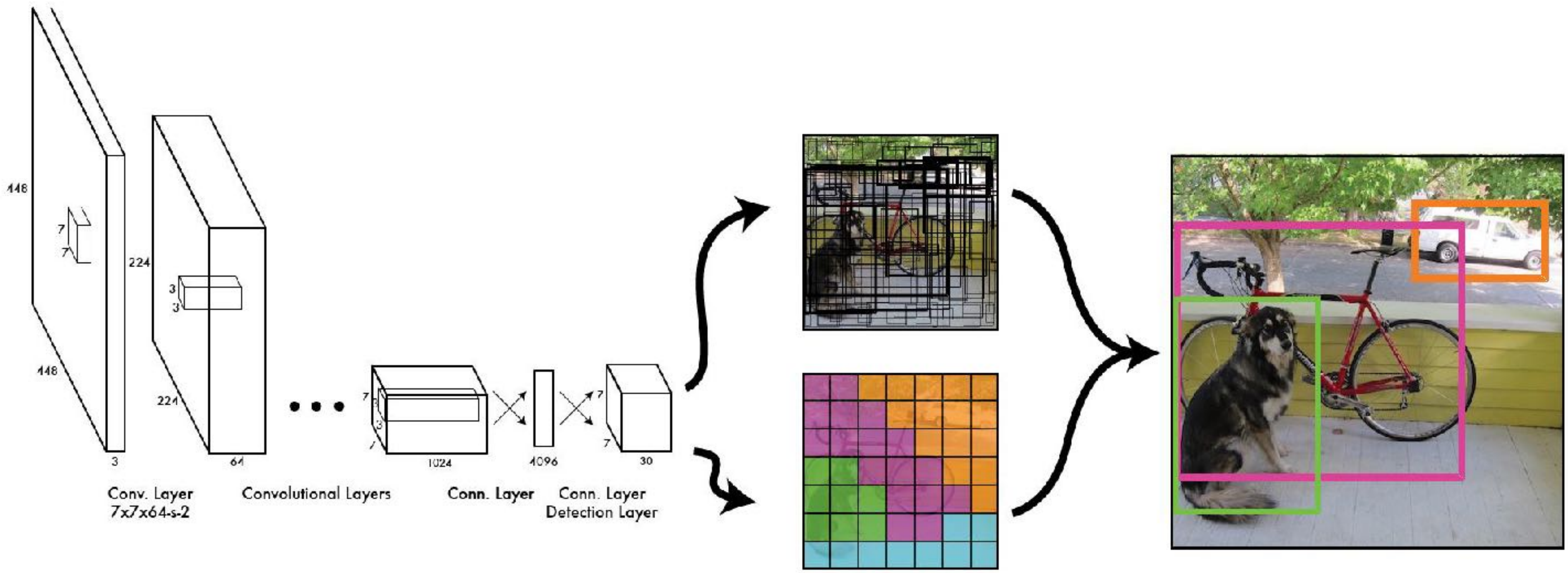
For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes



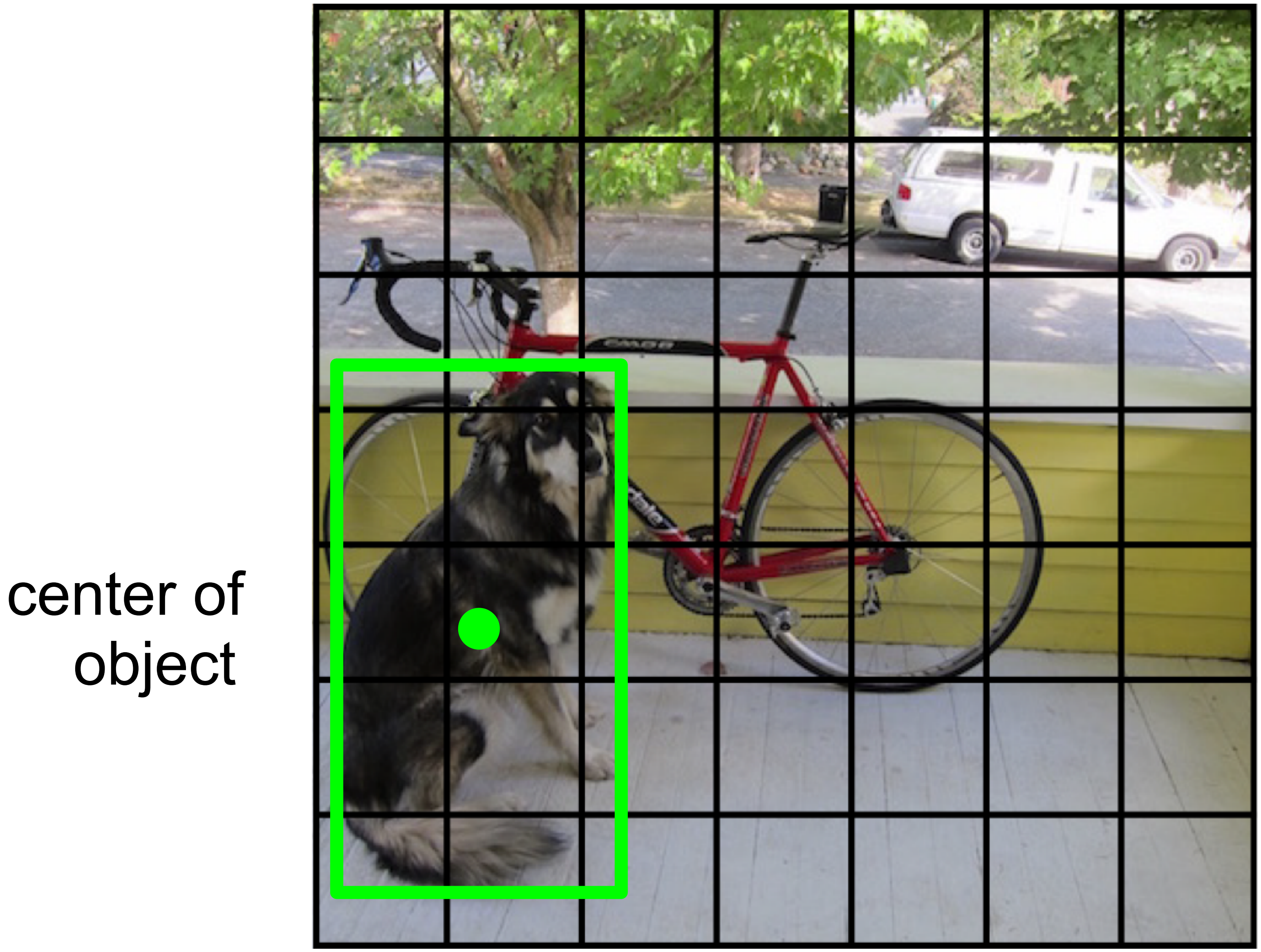
7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**

A single pipeline for detection

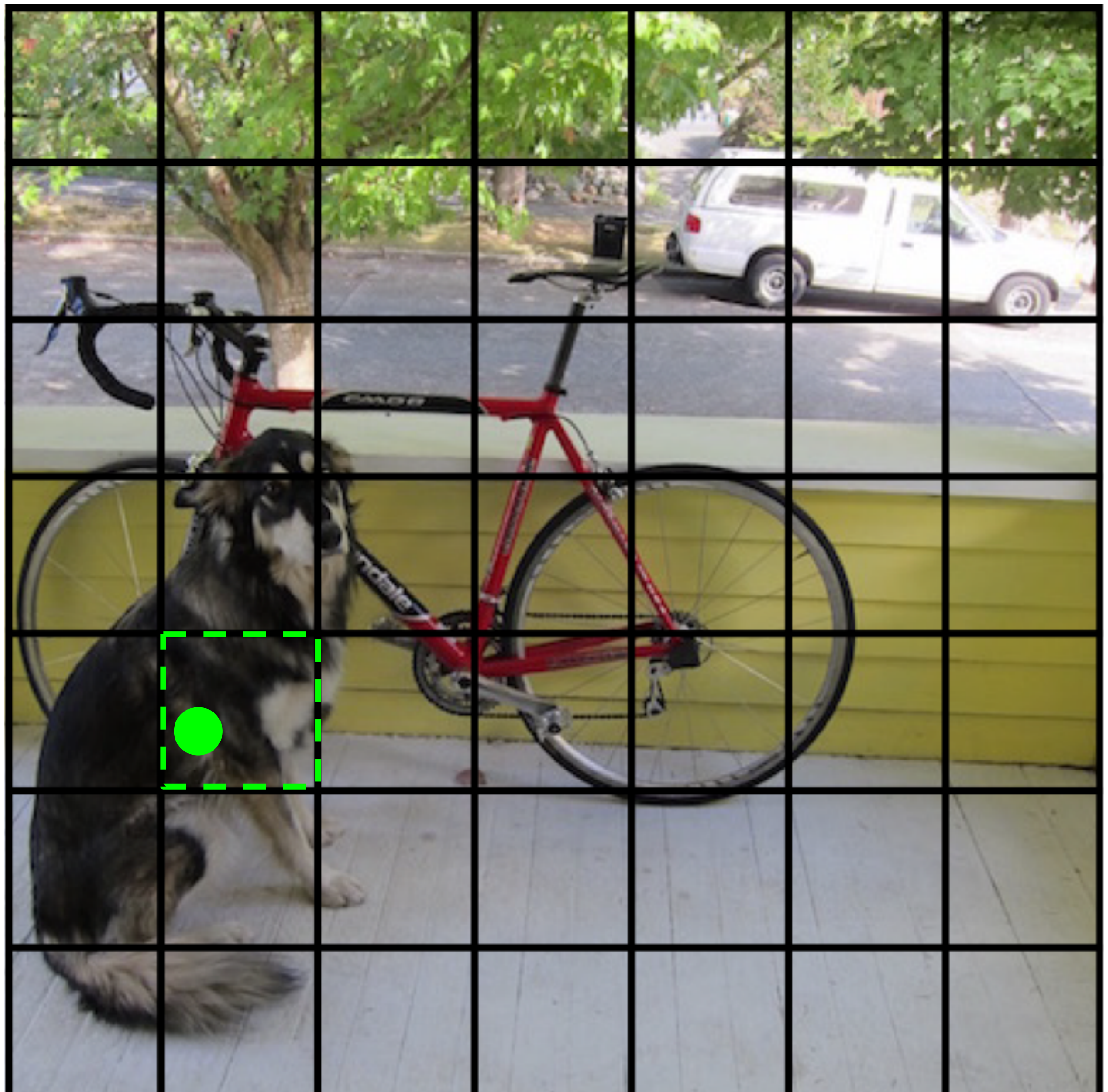


backbone network: VGG16, ResNet101, …

During training, match example to the right cell



During training, match example to the right cell



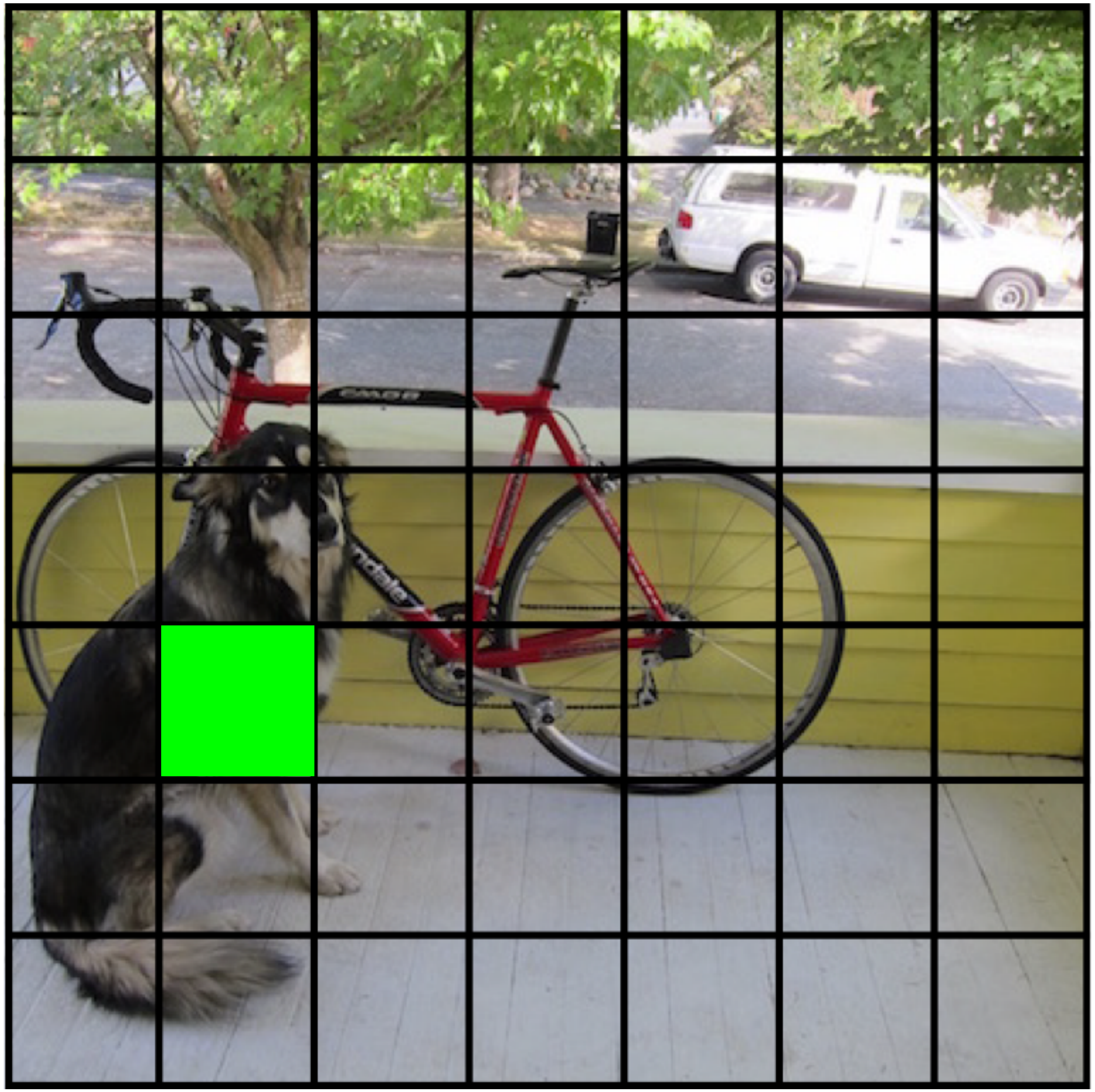
Adjust that cell's class prediction

Dog = 1

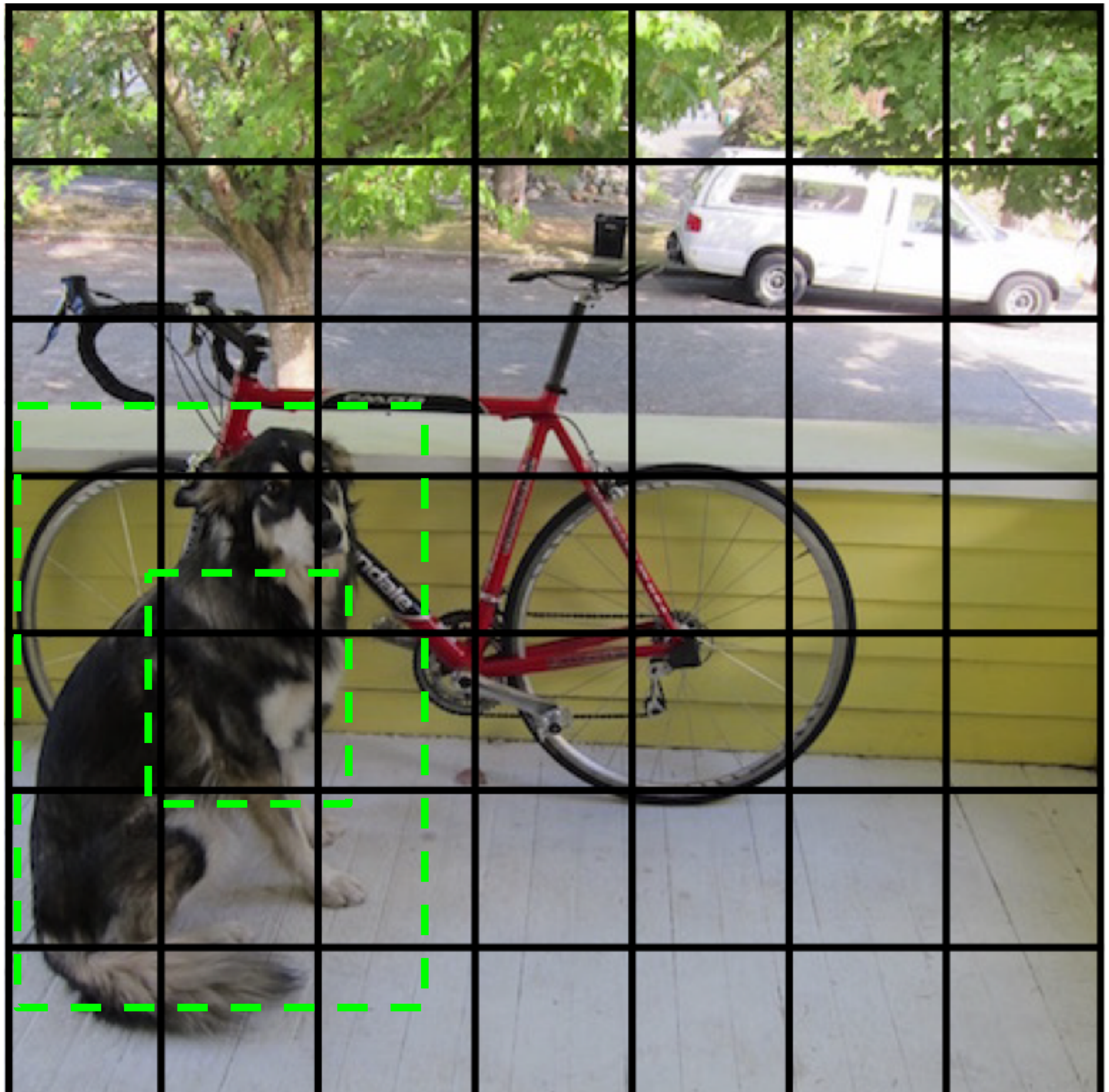
Cat = 0

Bike = 0

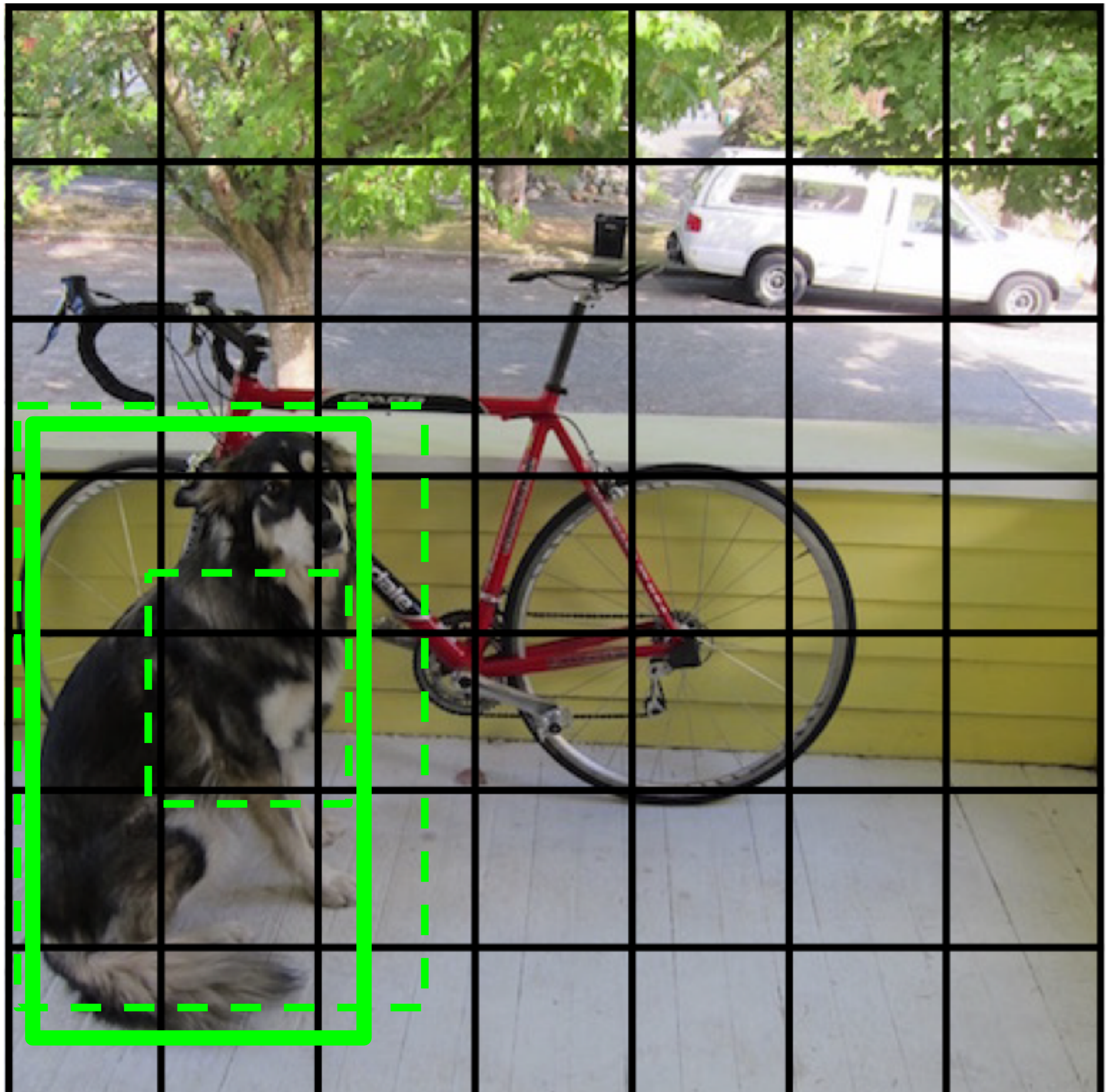
...



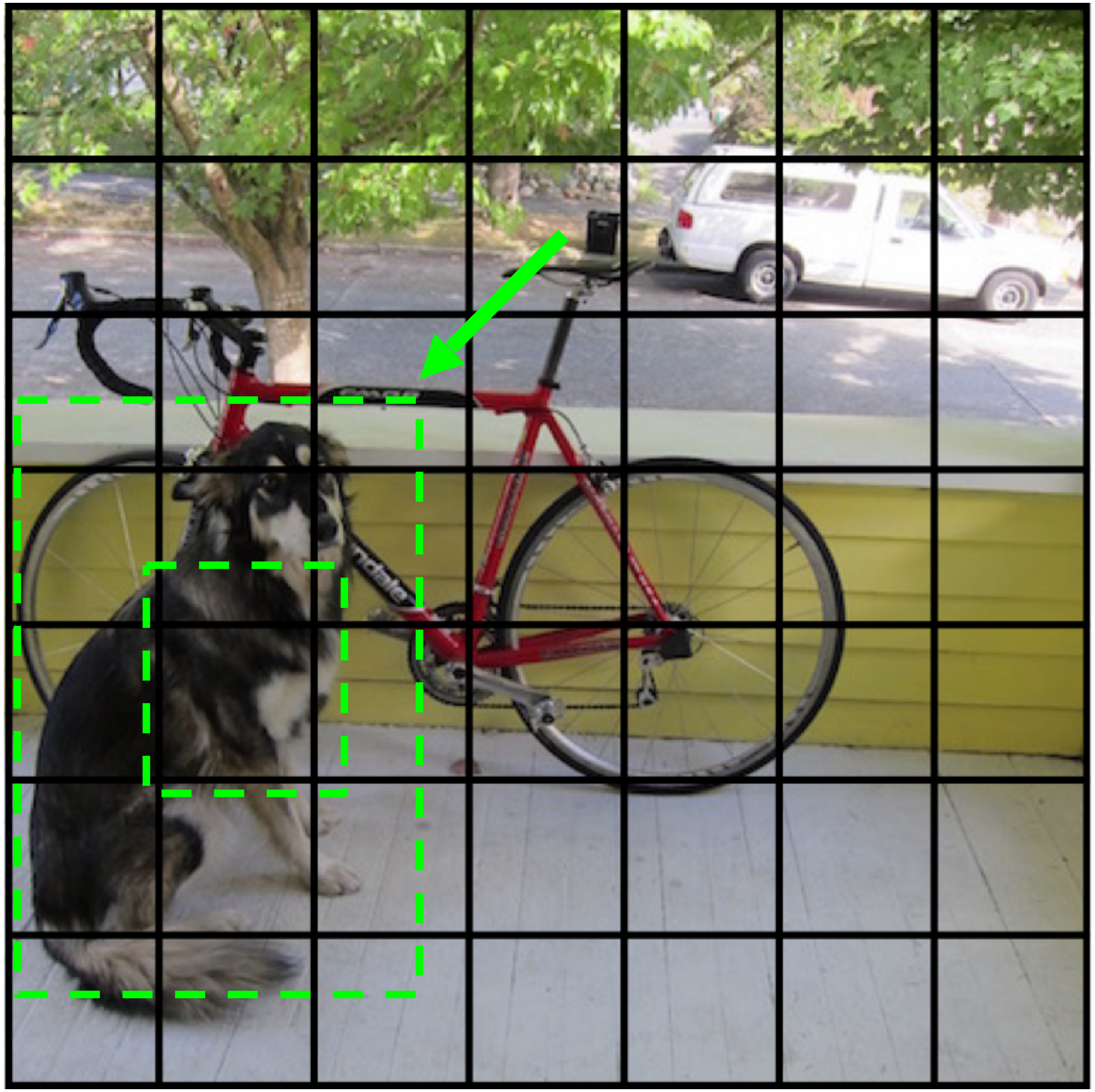
Look at that cell's predicted boxes



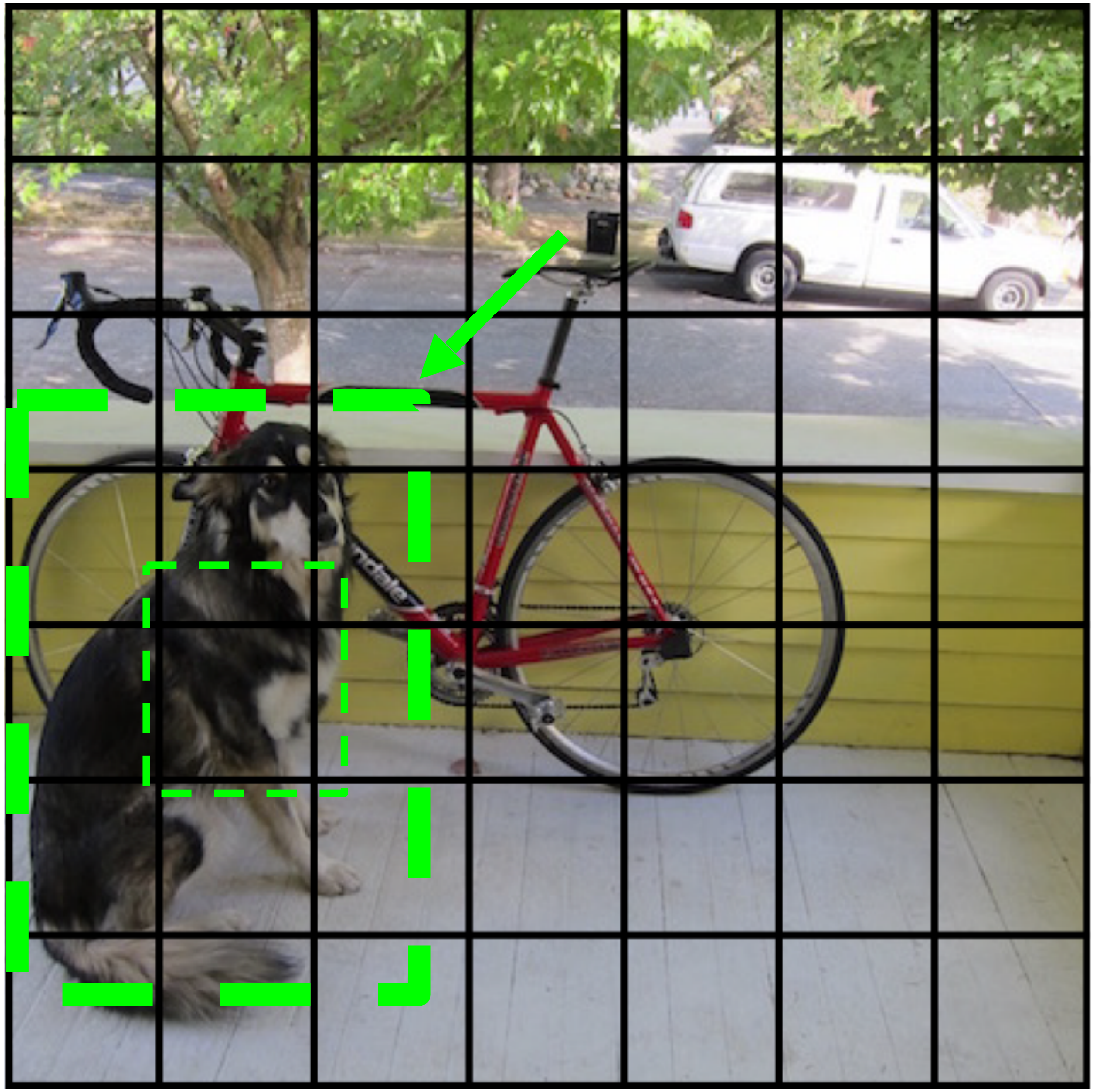
Find the best one, adjust it, increase the confidence



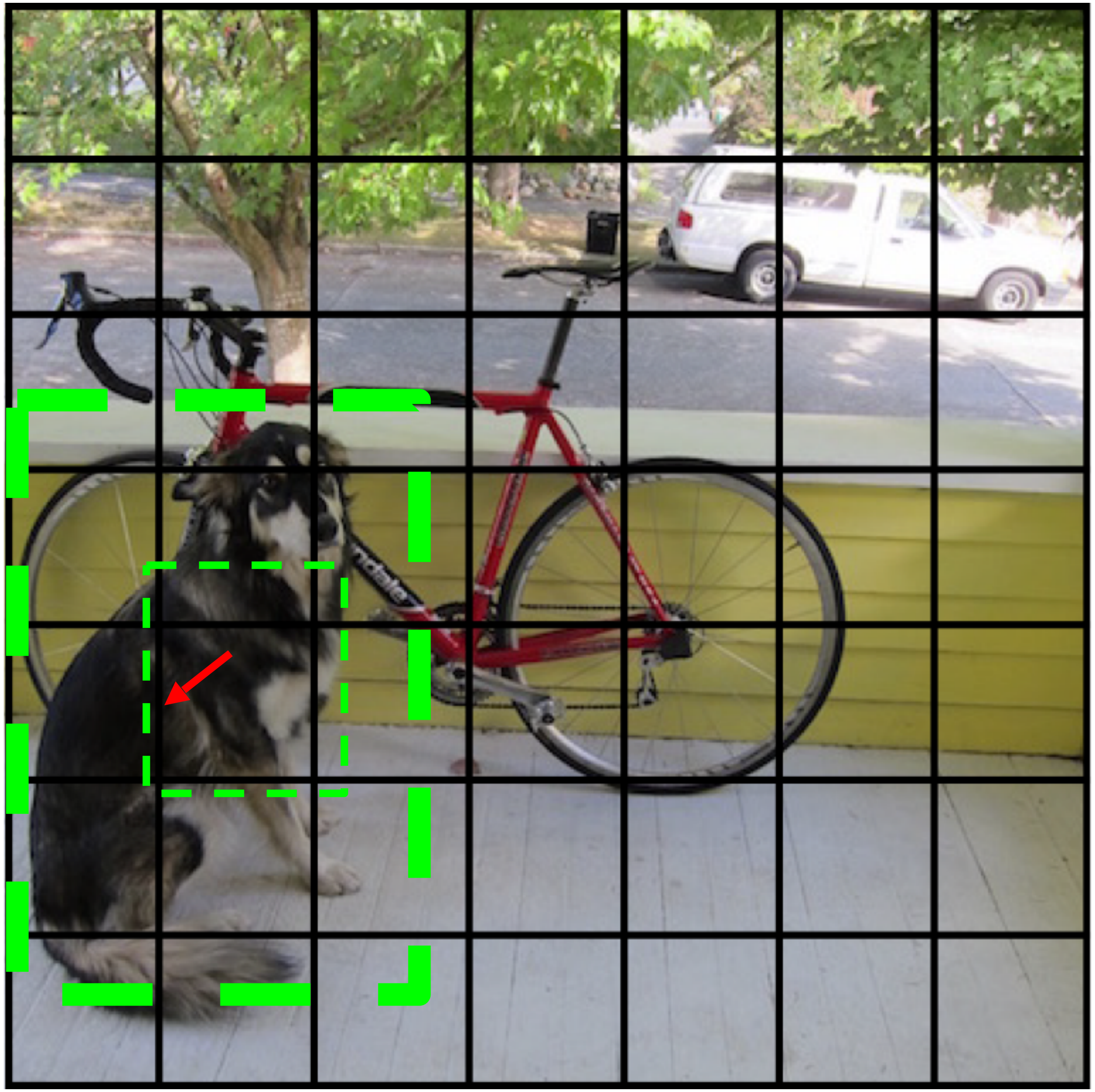
Find the best one, adjust it, increase the confidence



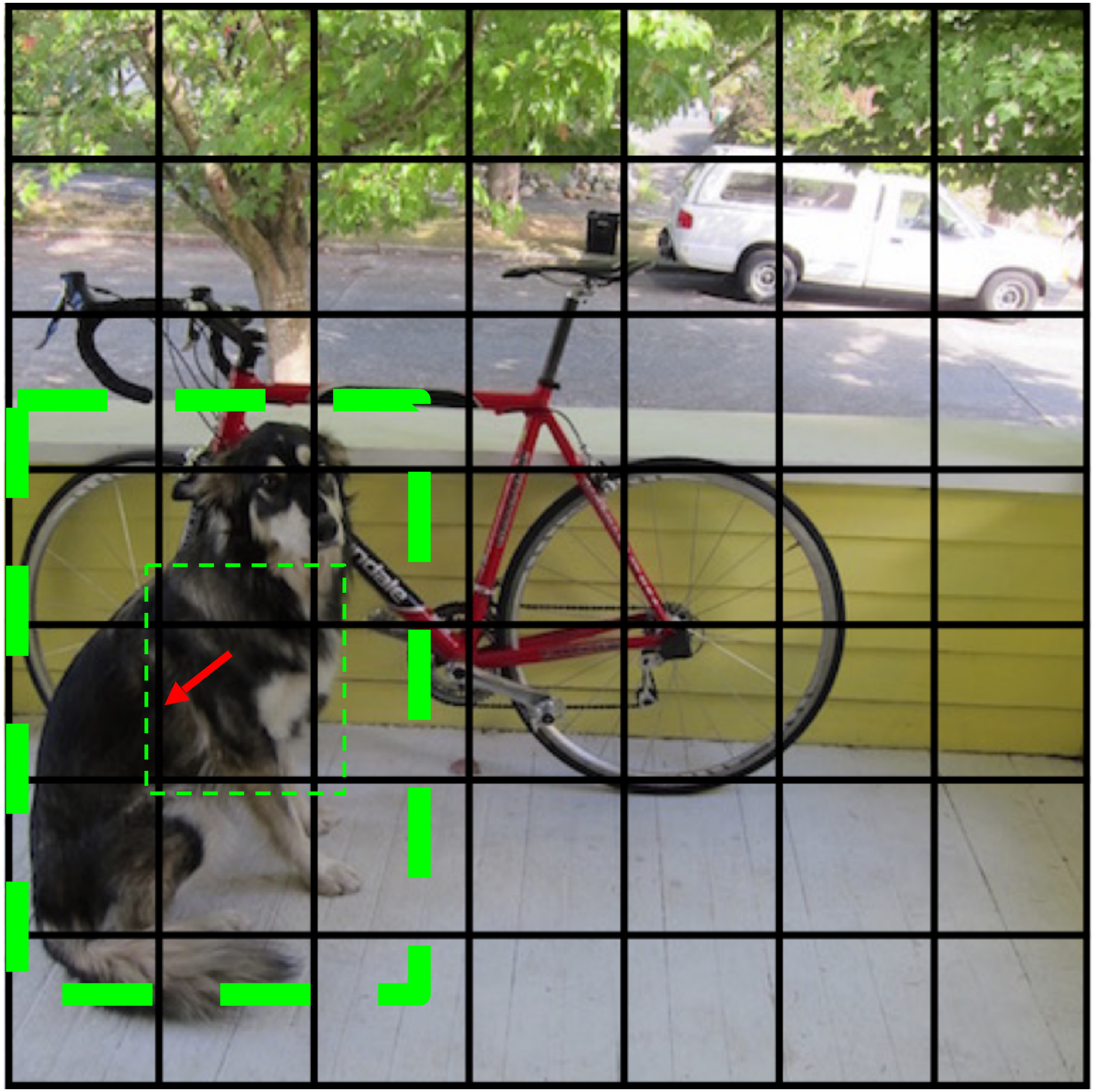
Find the best one, adjust it, increase the confidence



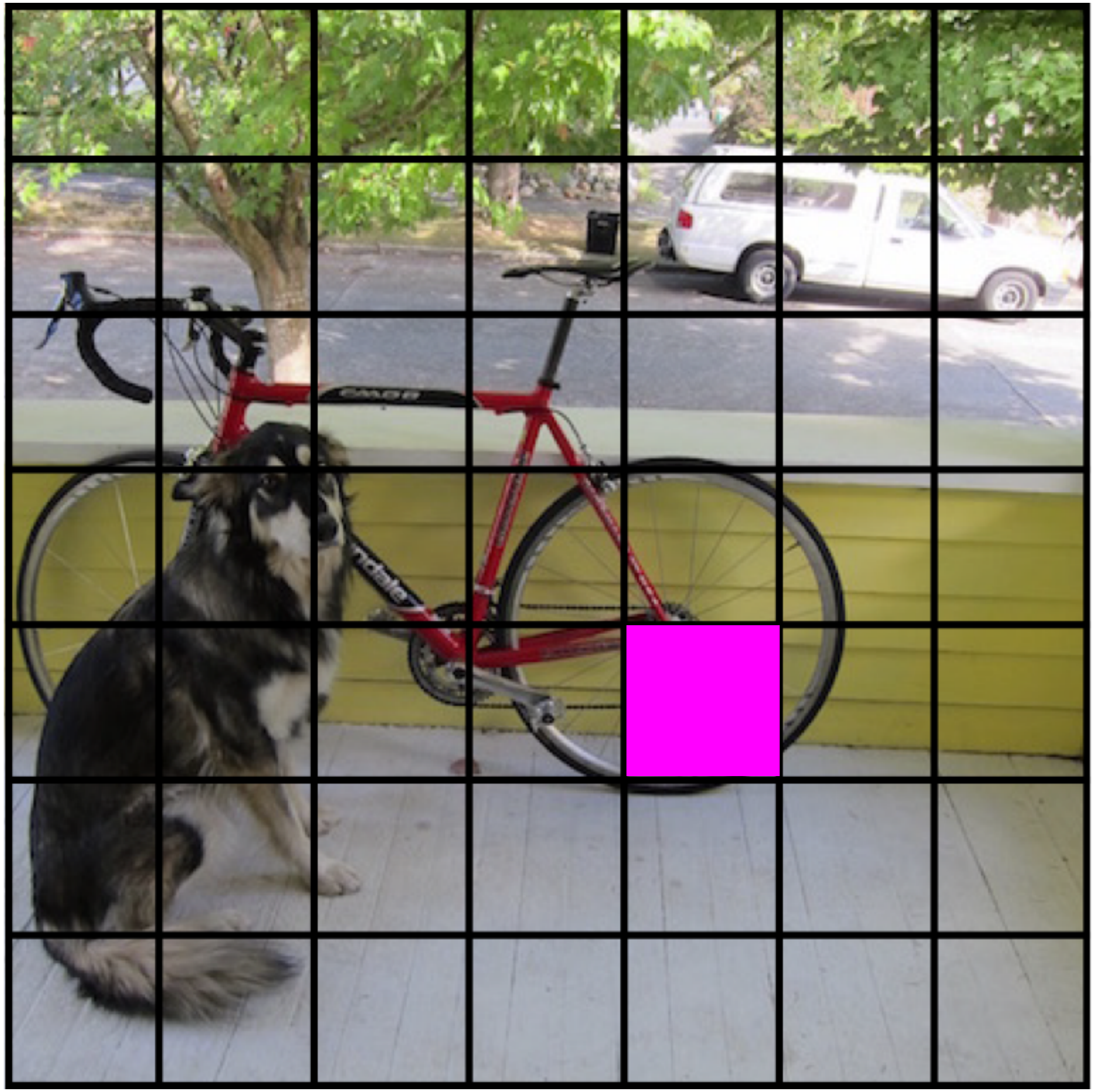
Decrease the confidence of other boxes



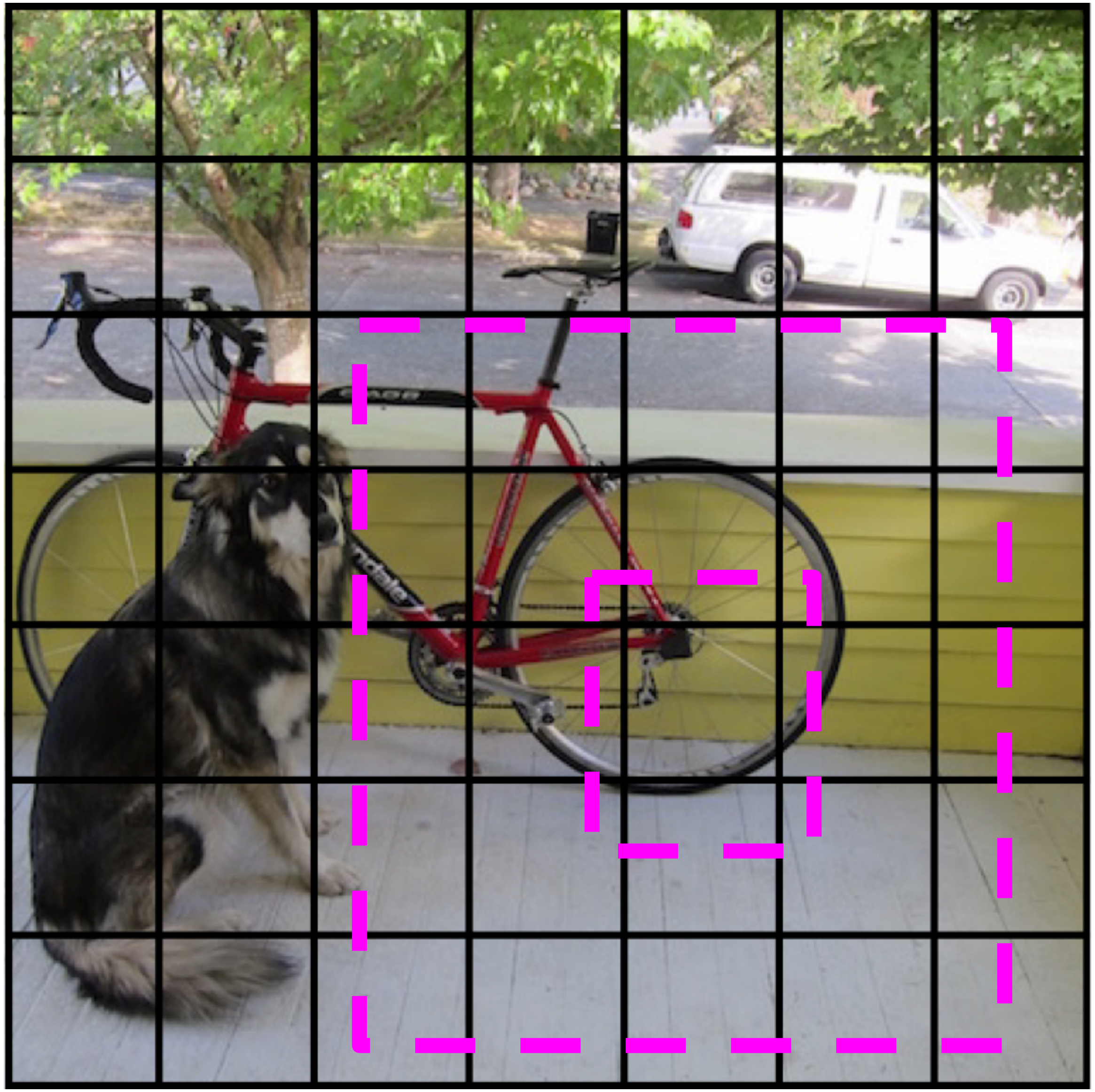
Decrease the confidence of other boxes



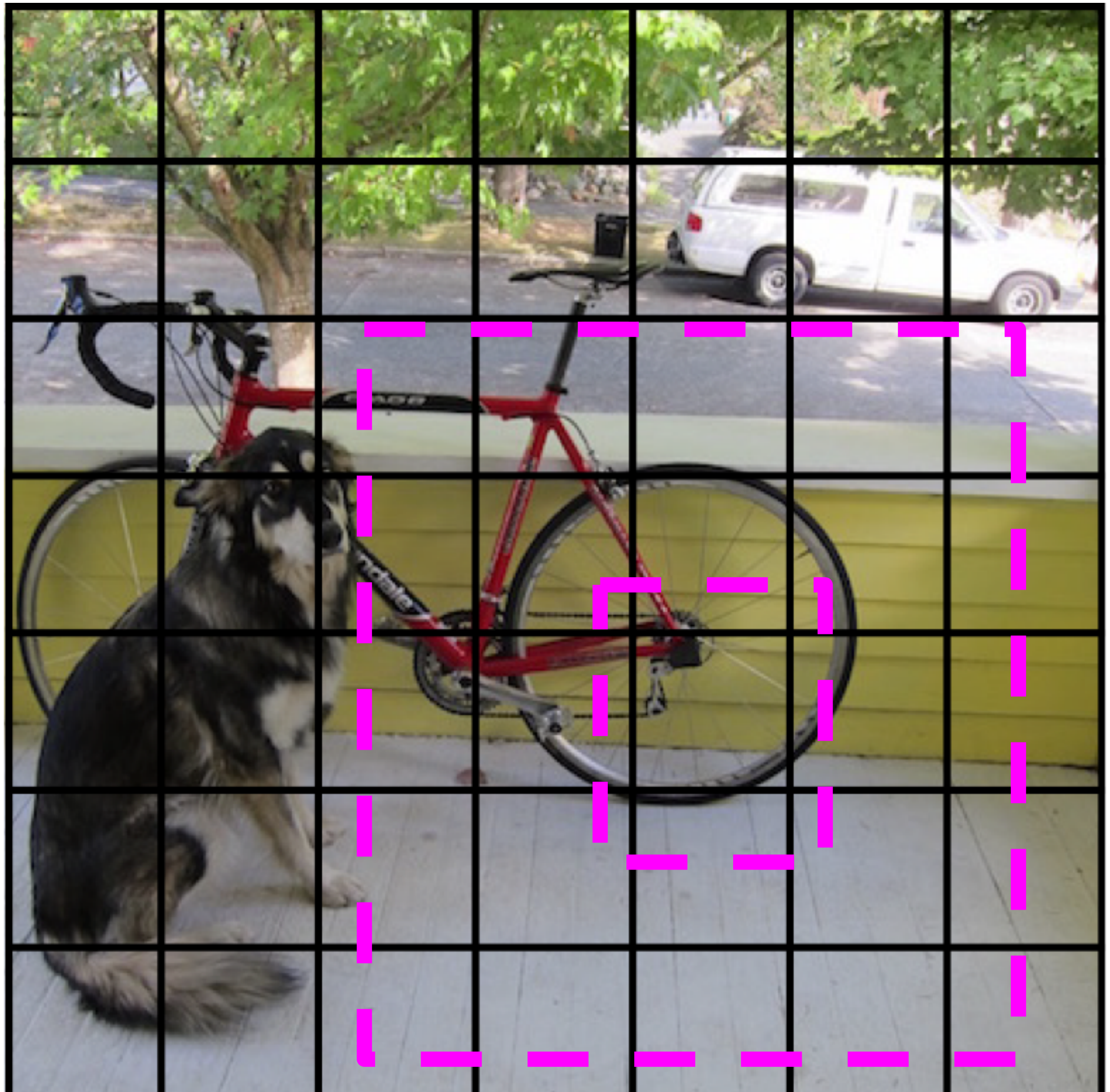
Some cells don't have any ground truth detections!



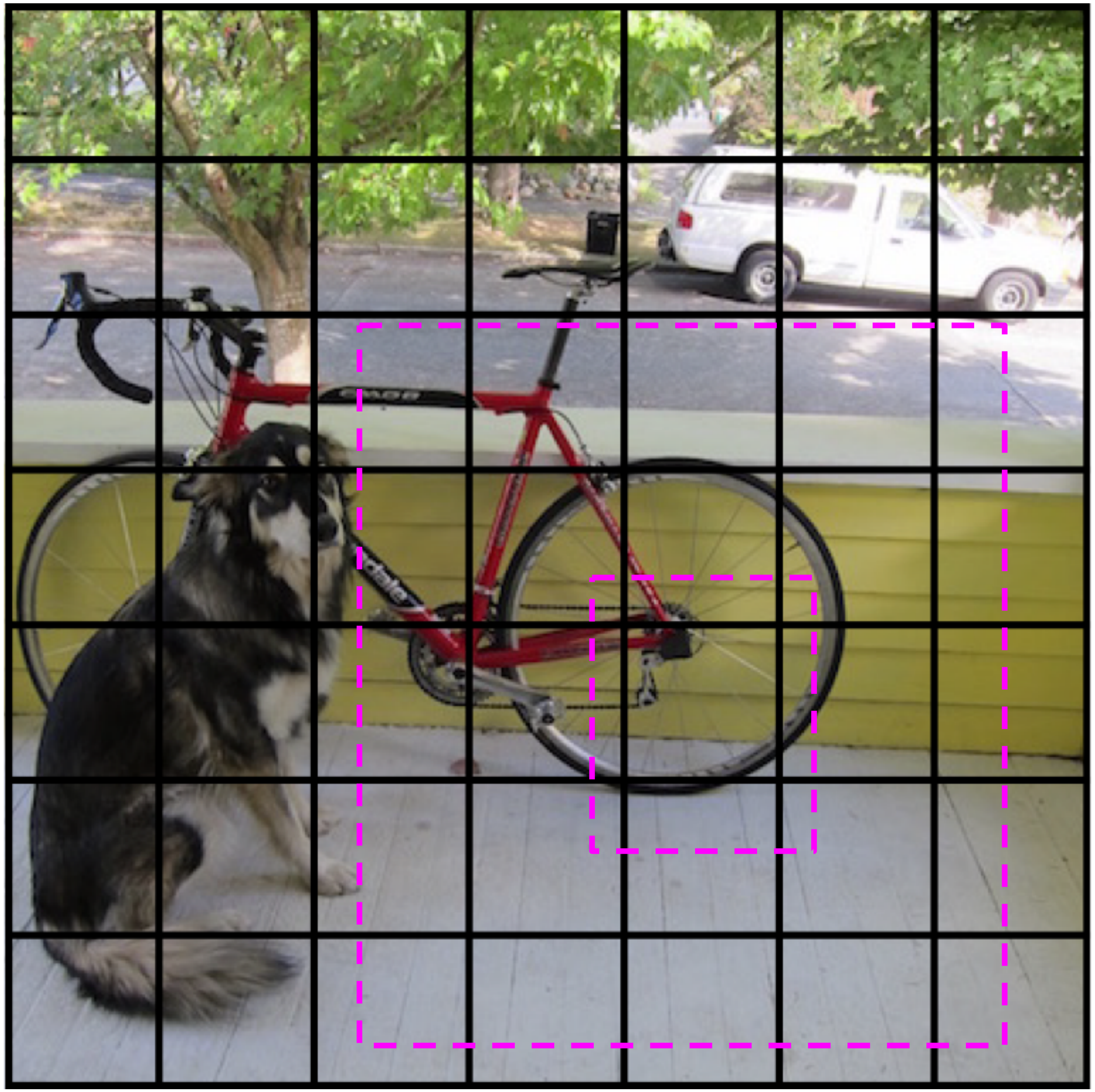
Some cells don’t have any ground truth detections!



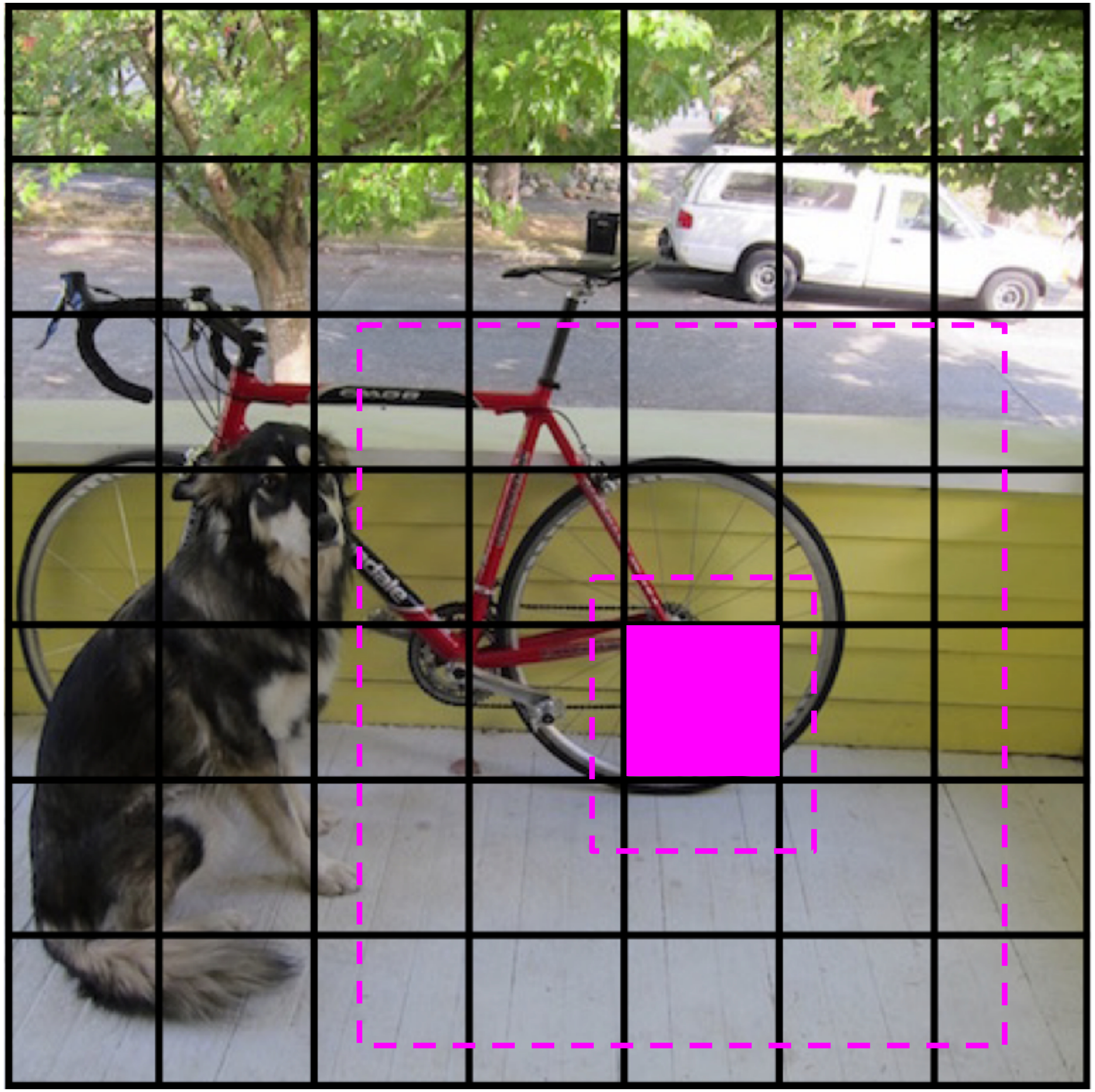
Decrease the confidence of these boxes

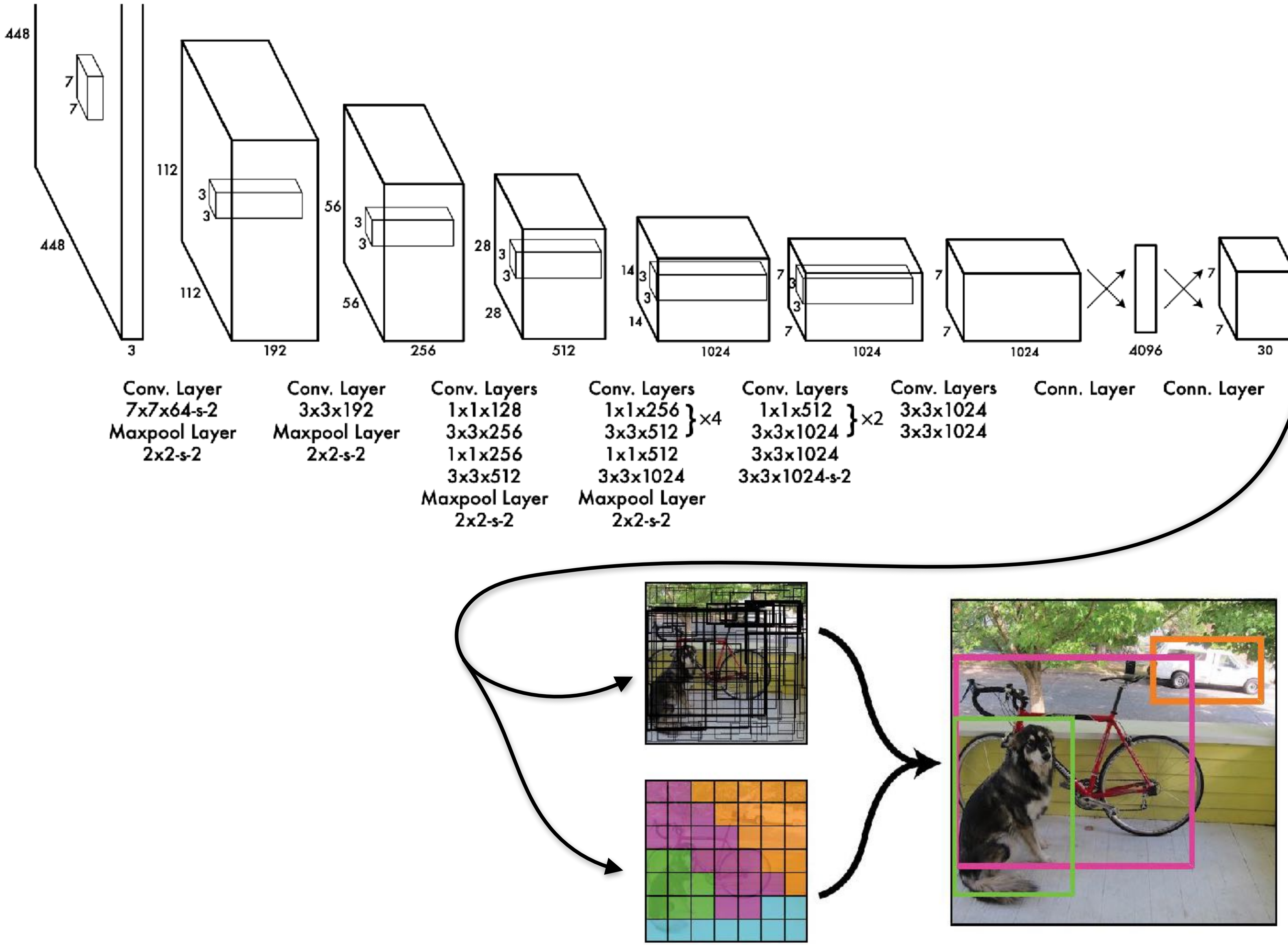


Decrease the confidence of these boxes



Don't adjust the class probabilities or coordinates




448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2
Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x256
3x3x512
×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x512
3x3x1024
×2
3x3x1024
3x3x1024-s-2
Conv. Layers
3x3x1024
3x3x1024
Conn. Layer
Conn. Layer

Training YOLO

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

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

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

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

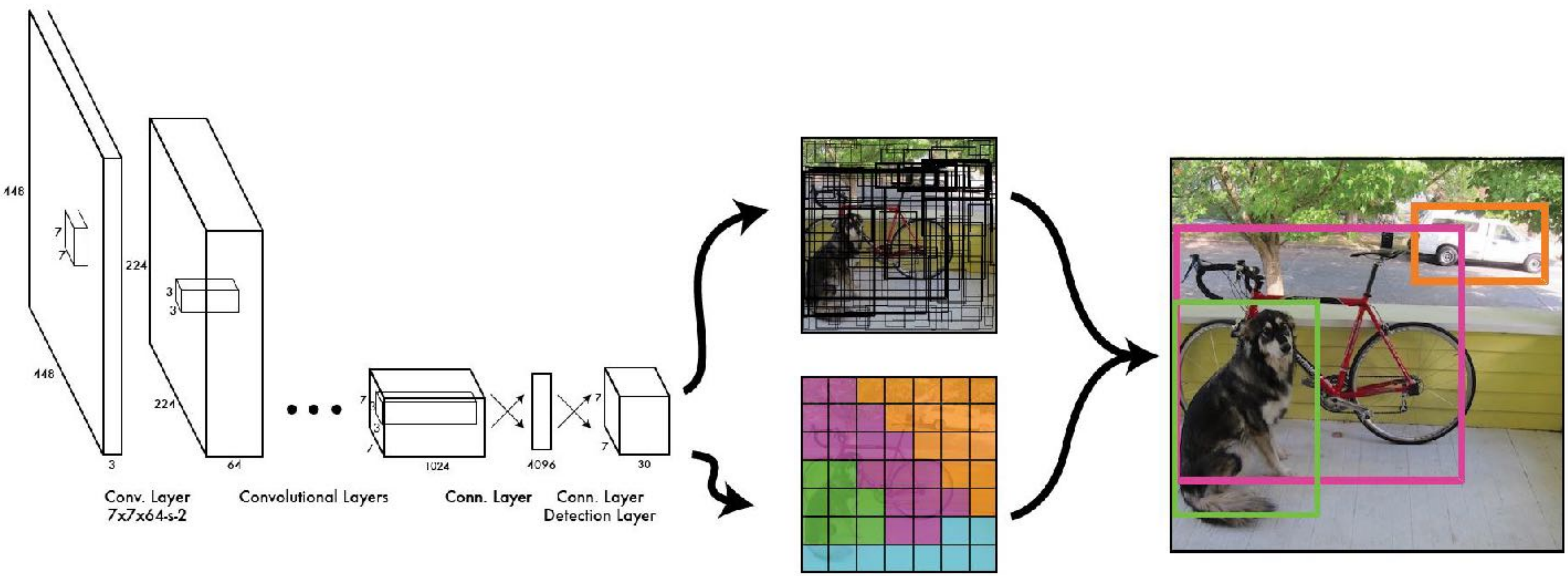
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

if i-th cell contain object and j-th box has max IoU

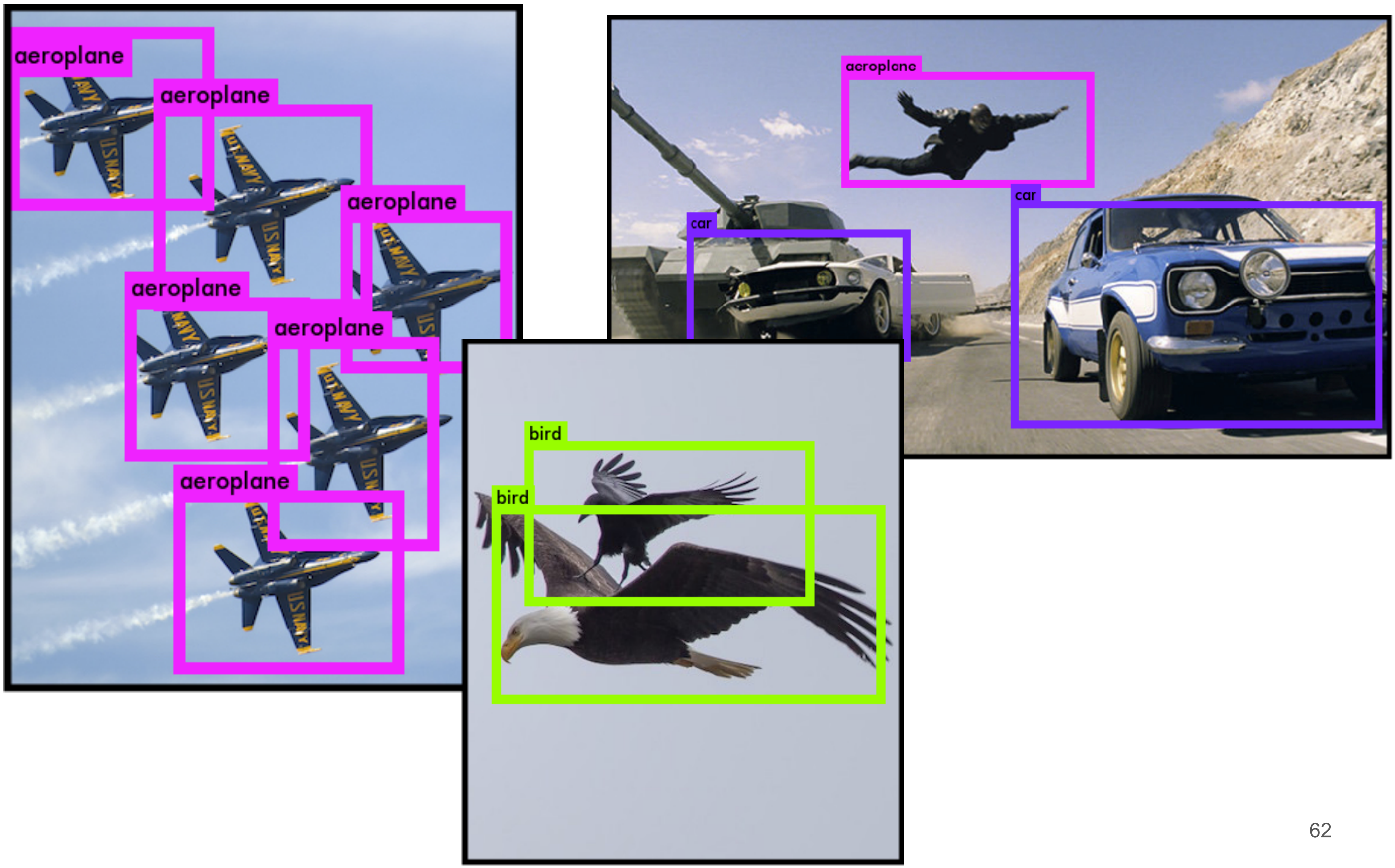
if i-th cell contain object and j-th box has max IoU

Other tricks

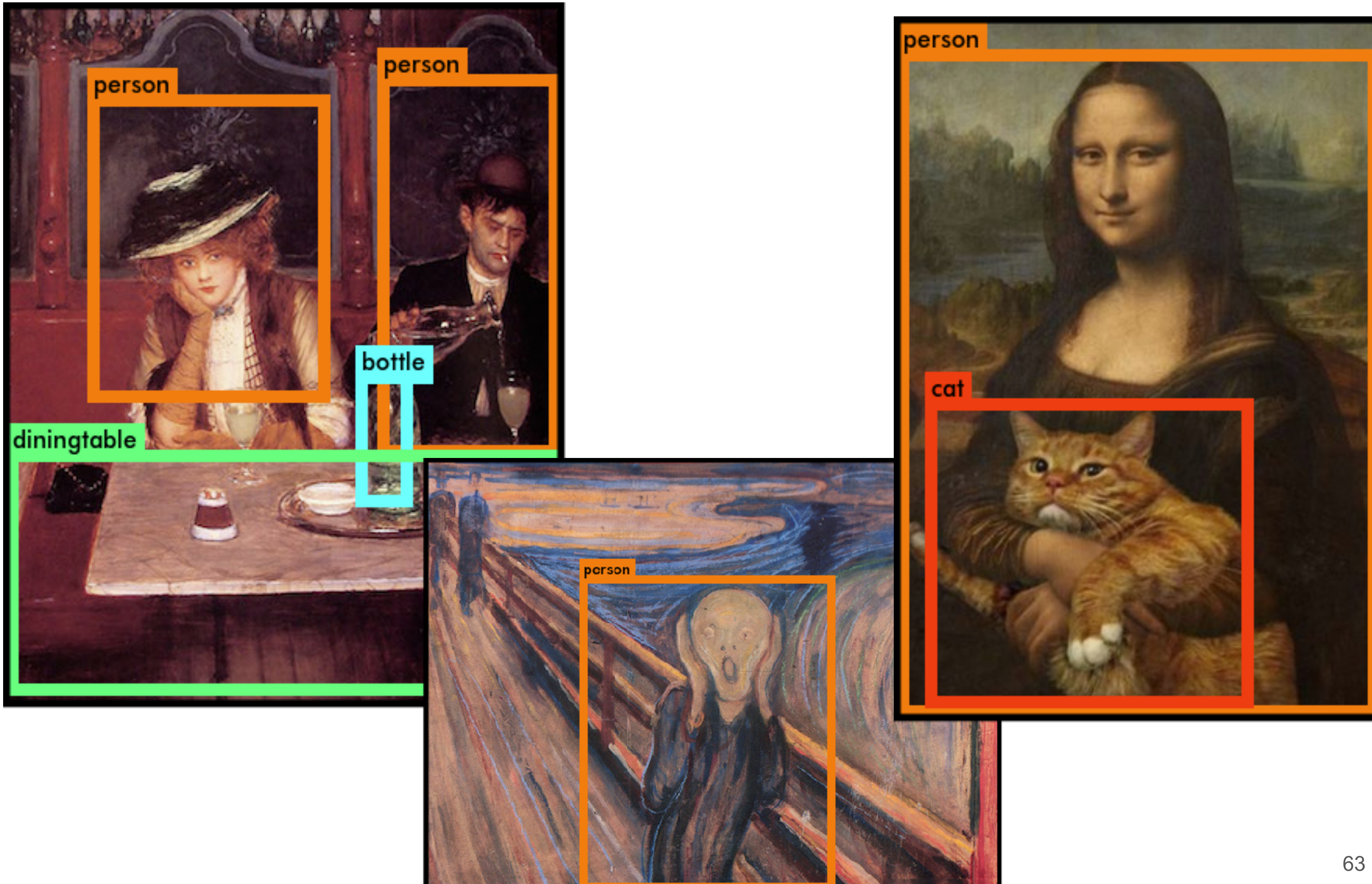
- Pretraining on Imagenet
- SGD with decreasing learning rate
- Extensive data augmentation



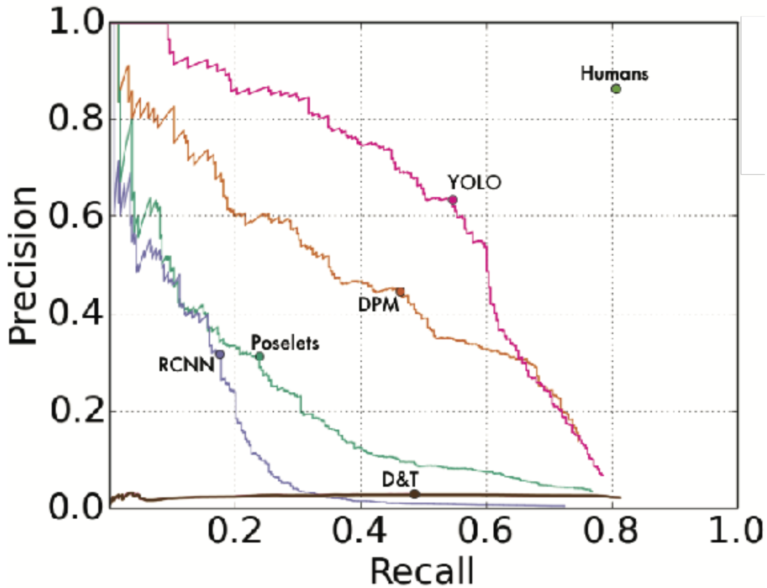
YOLO works across a variety of natural images



It also generalizes well to new domains (like art)



YOLO outperforms methods like DPM and R-CNN when generalizing to person detection in artwork



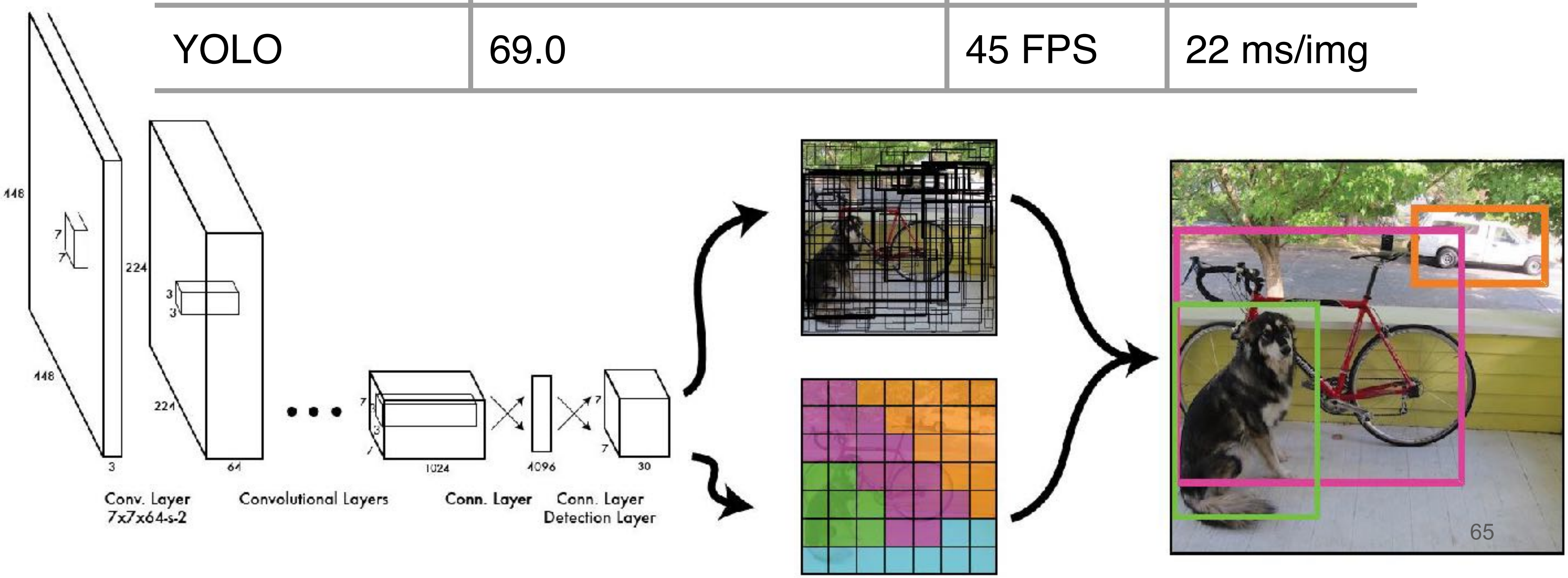
	VOC 2007 AP	Picasso AP	Picasso Best F_1	People-Art AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32

S. Ginosar, D. Haas, T. Brown, and J. Malik. Detecting people in cubist art. In Computer Vision-ECCV 2014 Workshops, pages 101–116. Springer, 2014.

H. Cai, Q. Wu, T. Corradi, and P. Hall. The cross-depiction problem: Computer vision algorithms for recognising objects in artwork and in photographs.

Results: Performance vs Speed

	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	69.0	45 FPS	22 ms/img

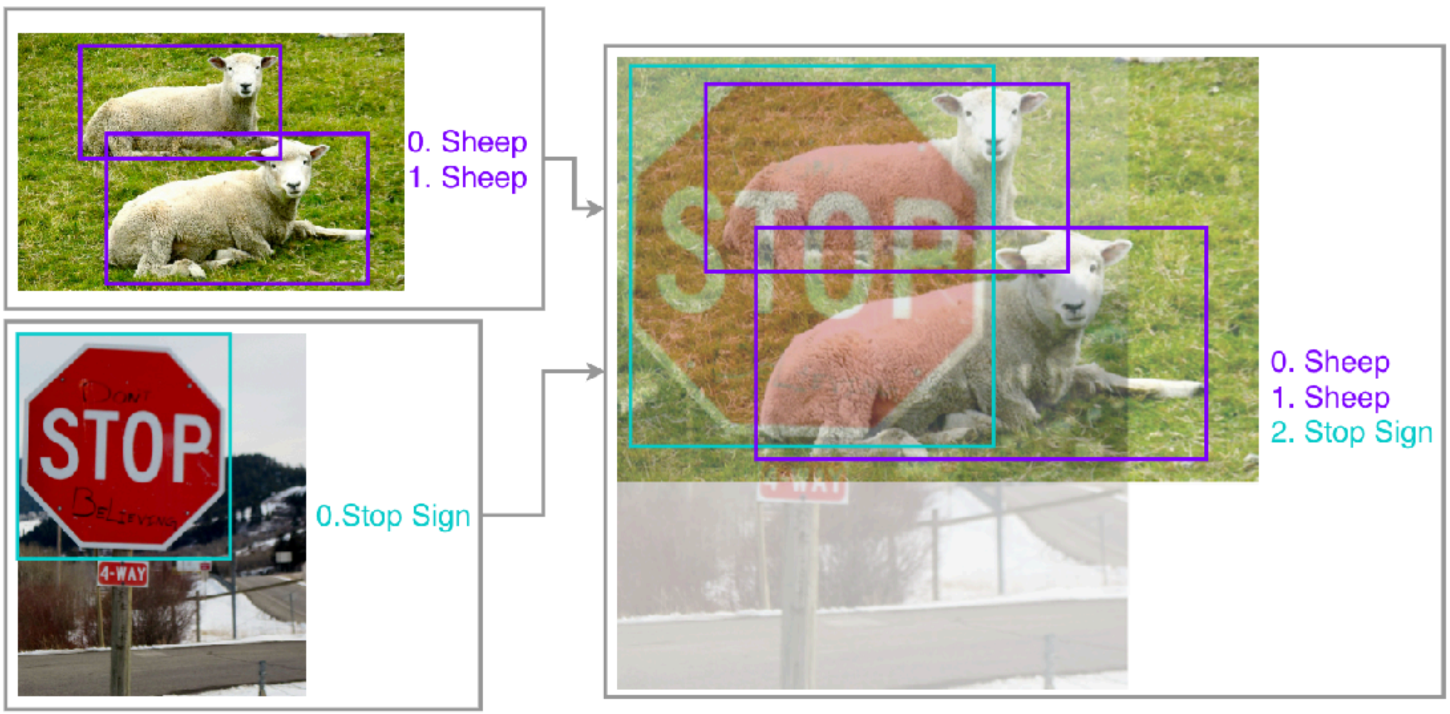


YOLO Series

- YOLO
- YOLOv2 improves the detection of small objects in groups and the localization accuracy.
  - and adding batch norm
- YOLOv3,
  - 106 layer resnet
  - multi-scale detection (three scales)
- YOLOv4, …

Additional Tricks: Mixup

- Apply to object detection as well



Results for YOLOv3

Incremental Tricks	mAP	Δ	Cumu Δ
- data augmentation	64.26	-15.99	-15.99
baseline	80.25	0	0
+ synchronize BN	80.81	+0.56	+0.56
+ random training shapes	81.23	+0.42	+0.98
+ cosine lr schedule	81.69	+0.46	+1.44
+ class label smoothing	82.14	+0.45	+1.89
+ mixup	**83.68**	**+1.54**	**+3.43**

Zhi et al, *Bag of Freebies for Training Object Detection Neural Networks*

Summary

- Object Detection
  - RCNN
  - YOLO: single pipeline model (e2e) for object detection

TA evaluation form

https://forms.gle/QWgfehMBDasvRozu7

Next Up

- Recurrent neural networks