Outline

1. Review: R-CNN

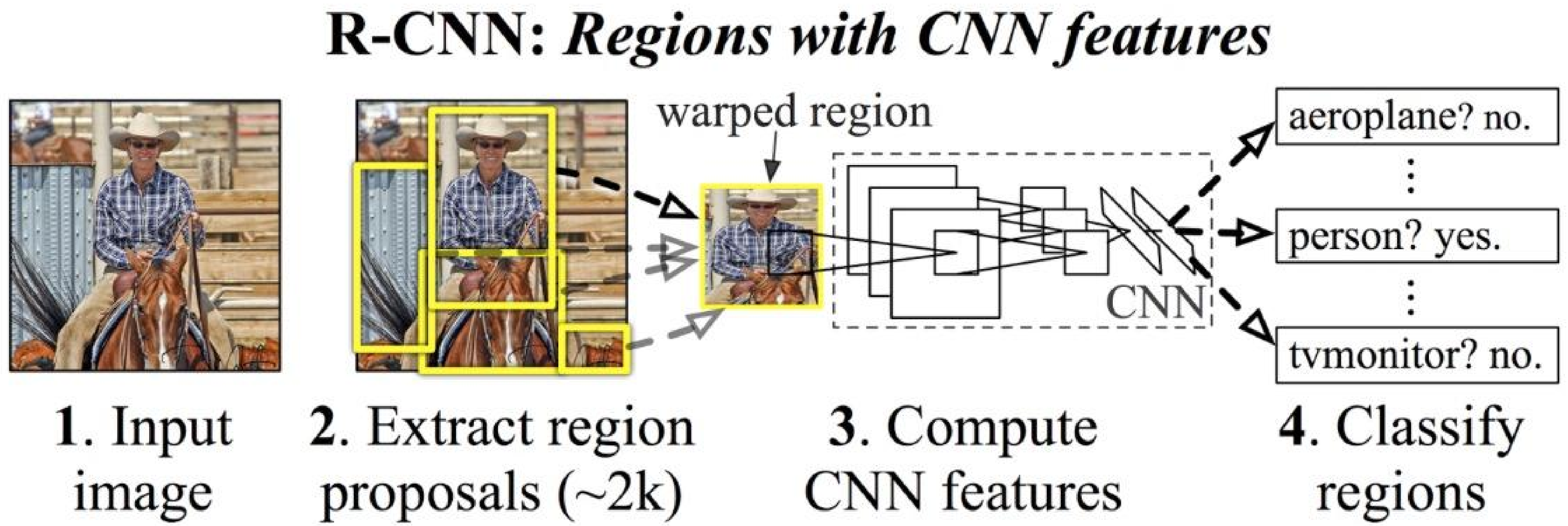
2. YOLO: -- Detection Procedure

   -- Network Design

   -- Training Part

   -- Experiments

Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation

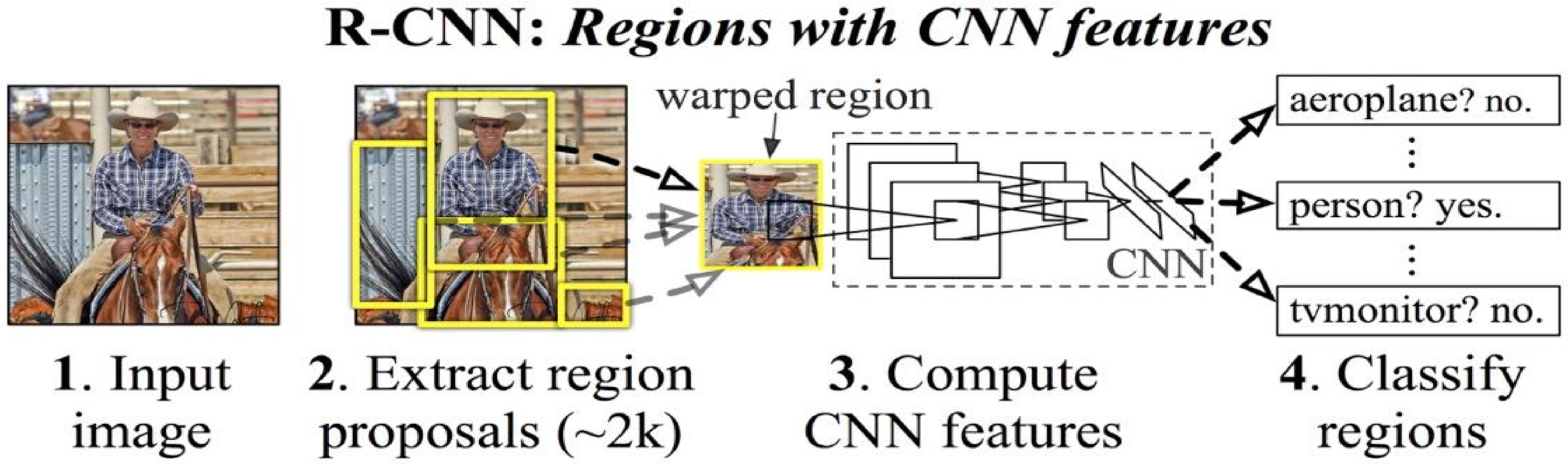

R-CNN: *Regions with CNN features*
warped region
aeroplane? no.
person? yes.
tvmonitor? no.
CNN
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

Proposal + Classification

Shortcoming:

1. Slow, impossible for real-time detection

2. Hard to optimize



Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation

WHAT’S NEW

Regression

YOLO Features：

1. Extremely fast (45 frames per second)

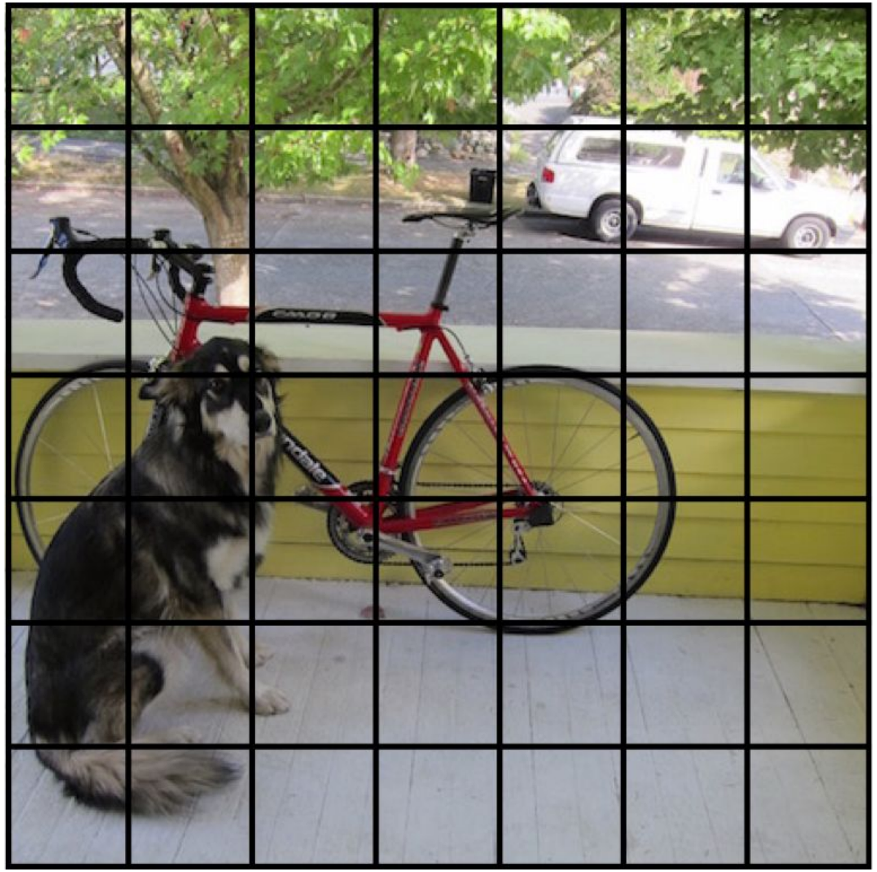
2. Reason Globally on the Entire Image

3. Learn Generalizable Representations

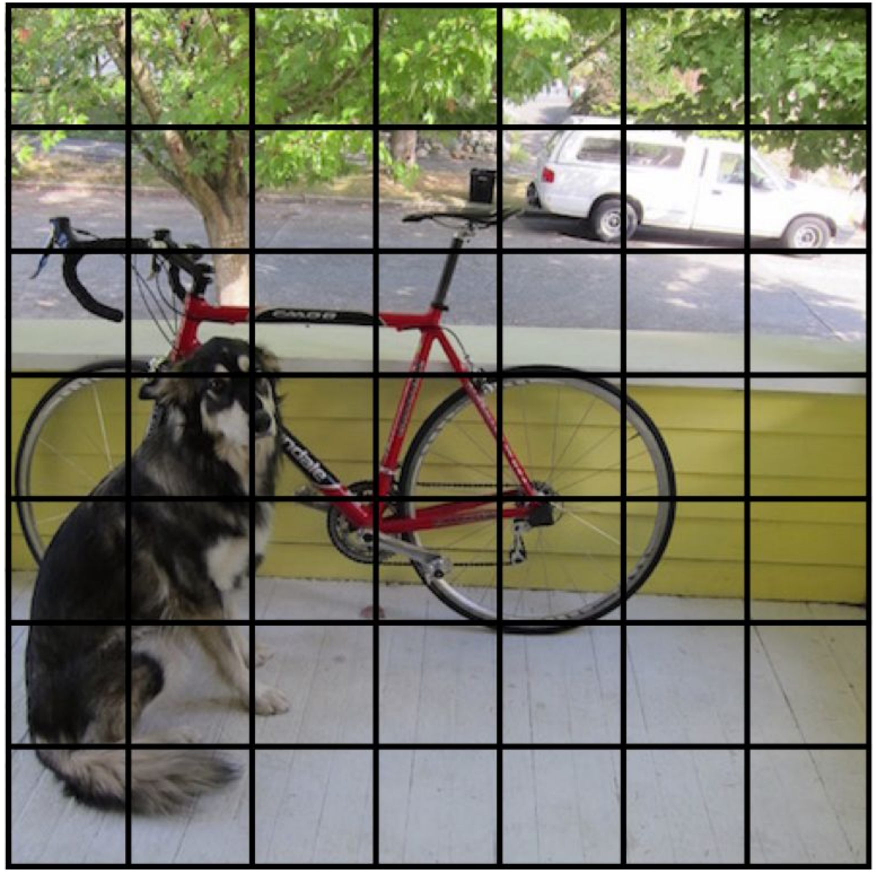
Detection Procedure

https://docs.google.com/presentation/d/1kAa7NOamBt4calBU9iHgT8a86RRHz
9Yz2oh4-GTdX6M/edit#slide=id.g151008b386_0_44

We split the image into an S*S grid

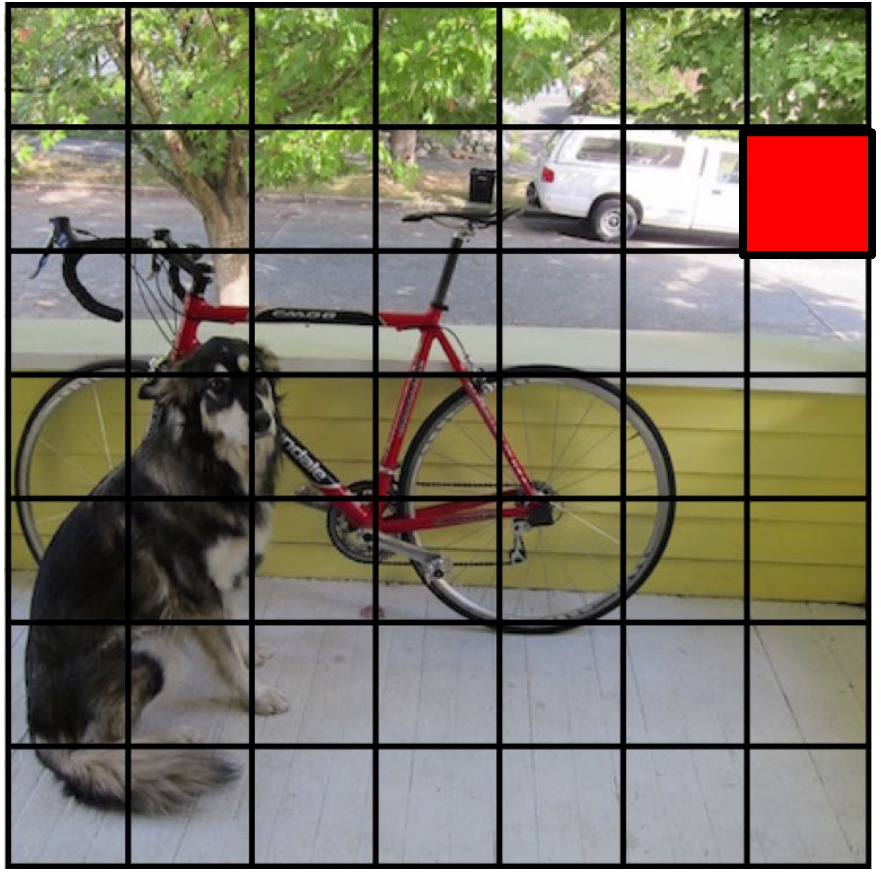


We split the image into an S*S grid

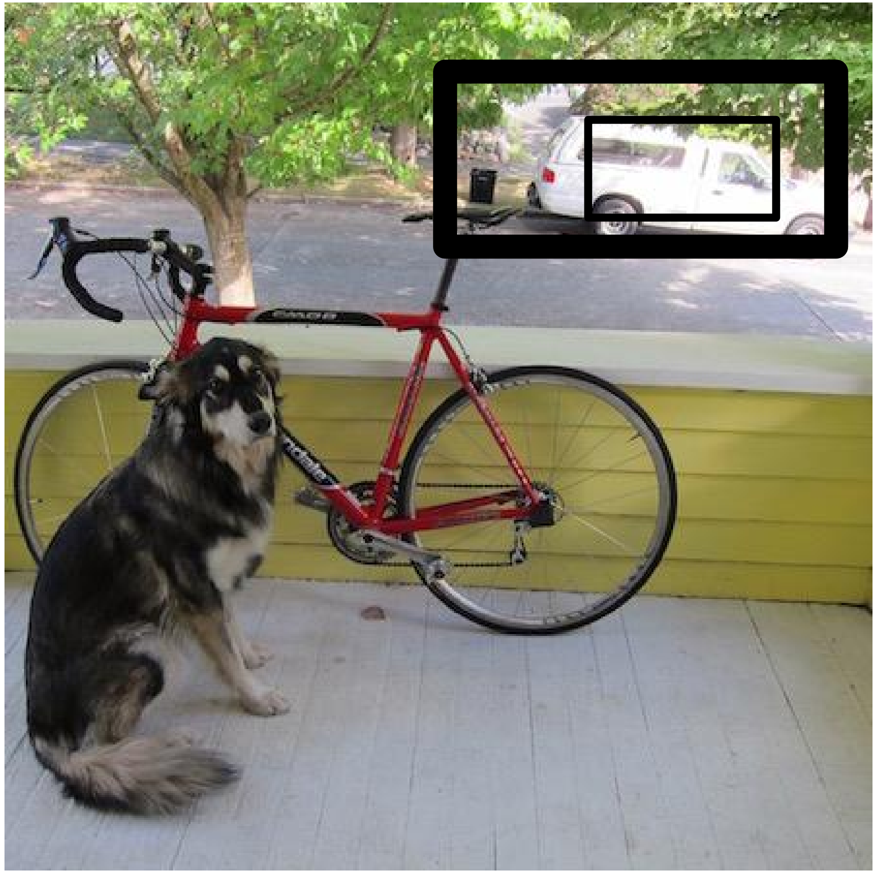


7*7 grid

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



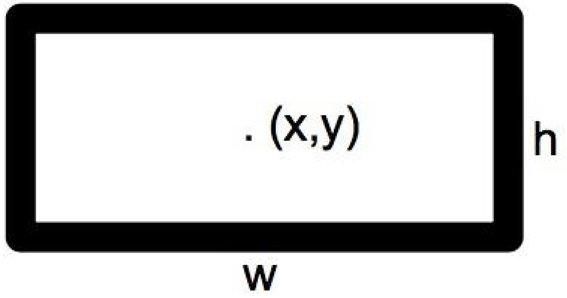
Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)



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

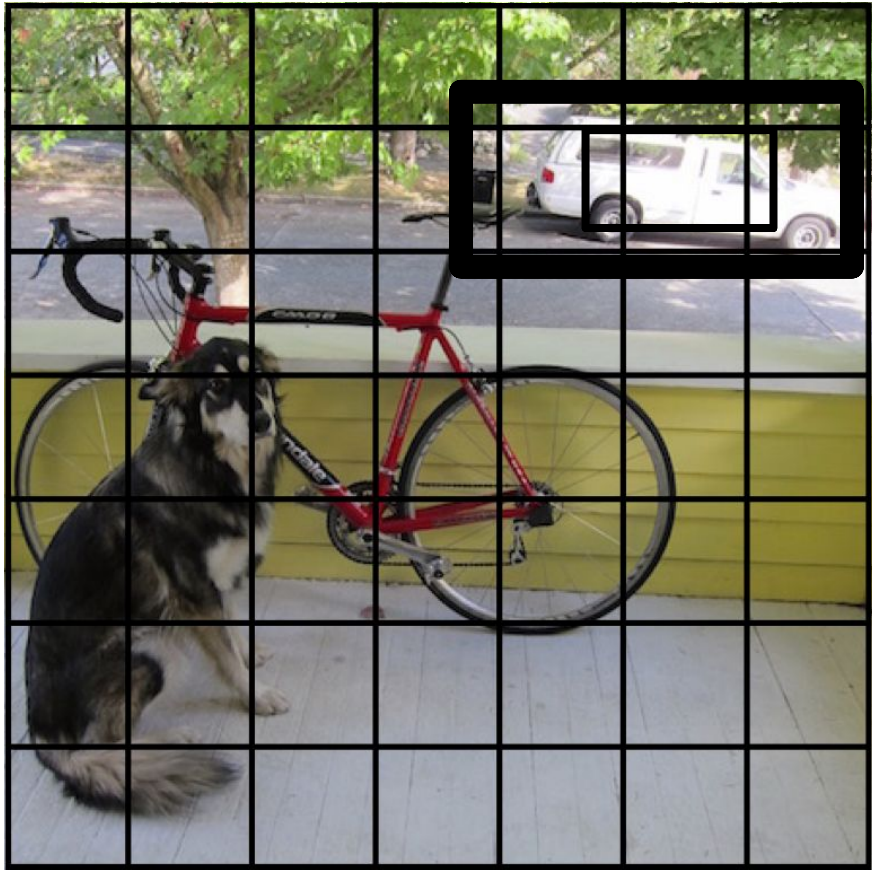
B = 2

each box predict:

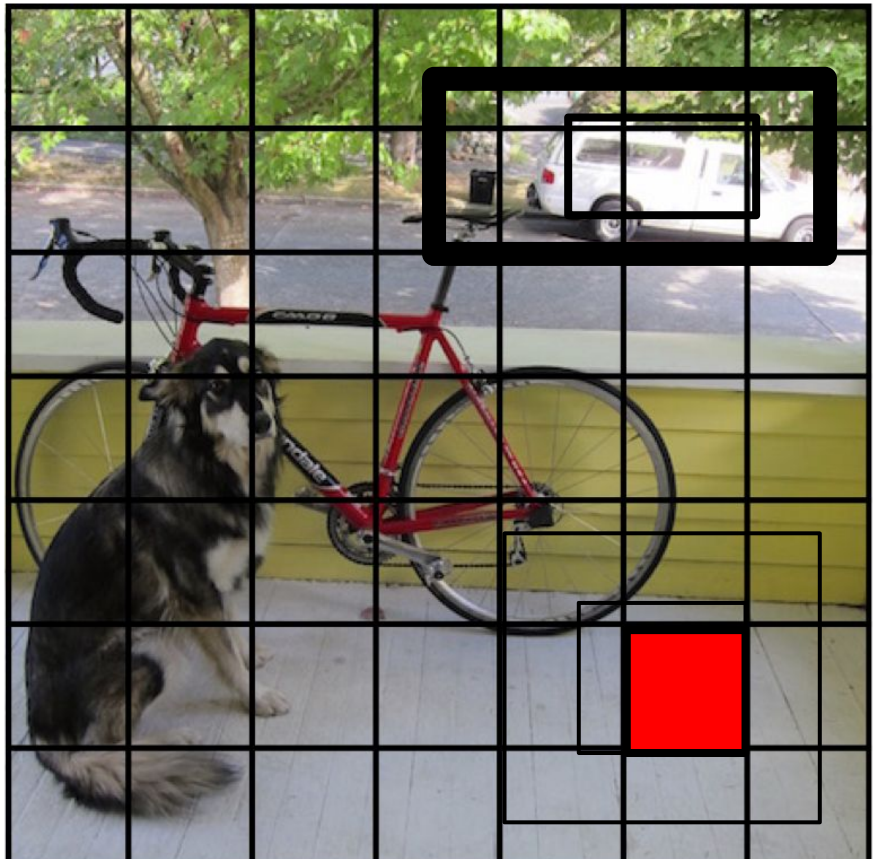


P(Object): probability that the box contains an object

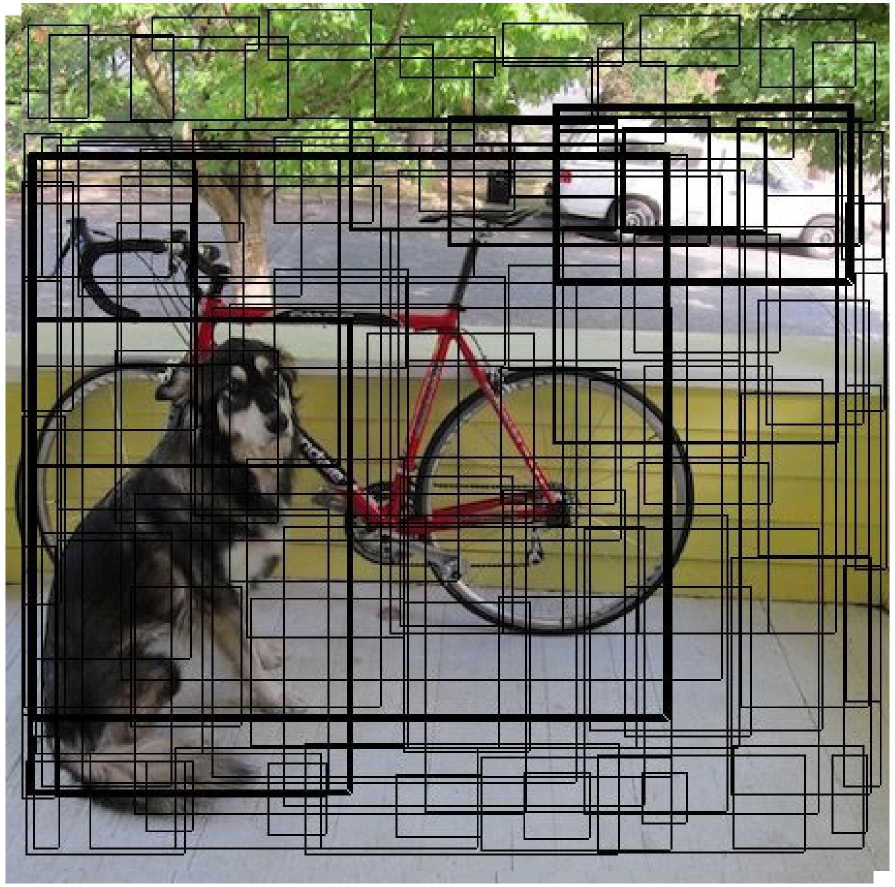
Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)



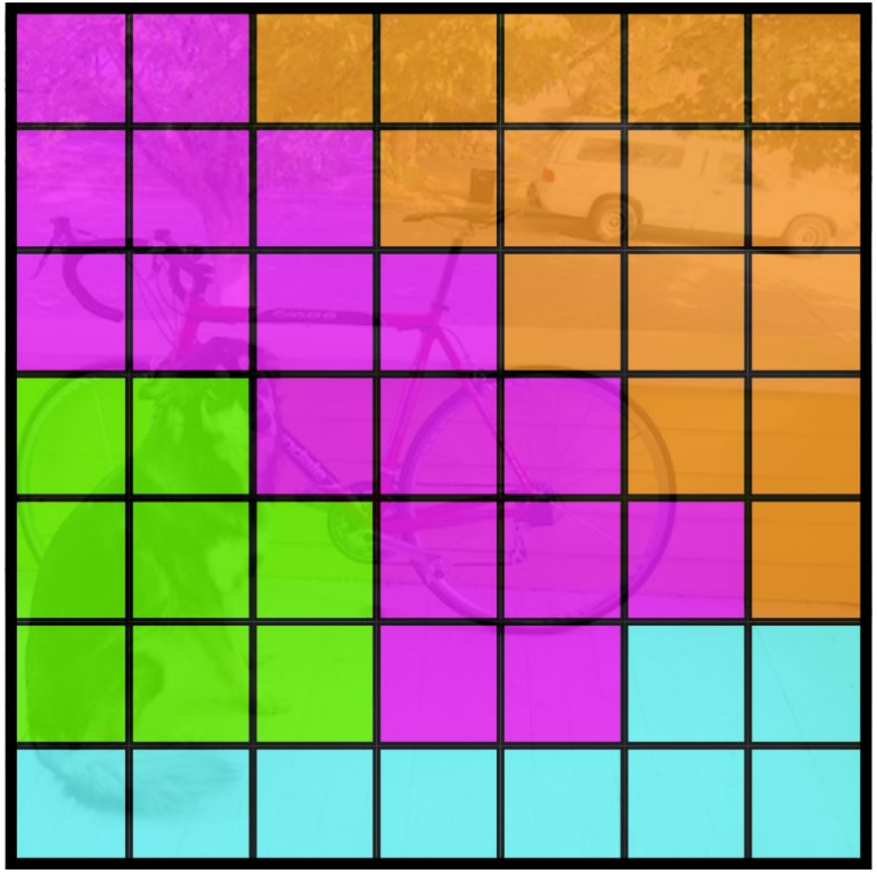
Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)



Each cell predicts boxes and confidences: P(Object)



Each cell also predicts a class probability.



Conditioned on object: P(Car | Object)

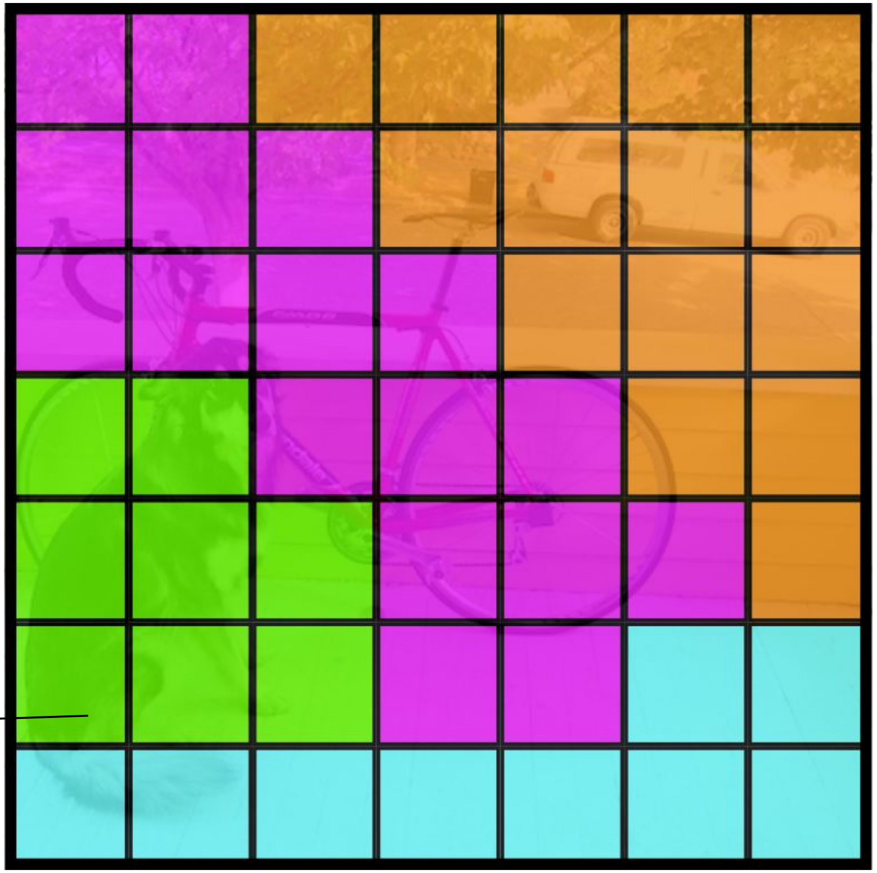
Bicycle

Car

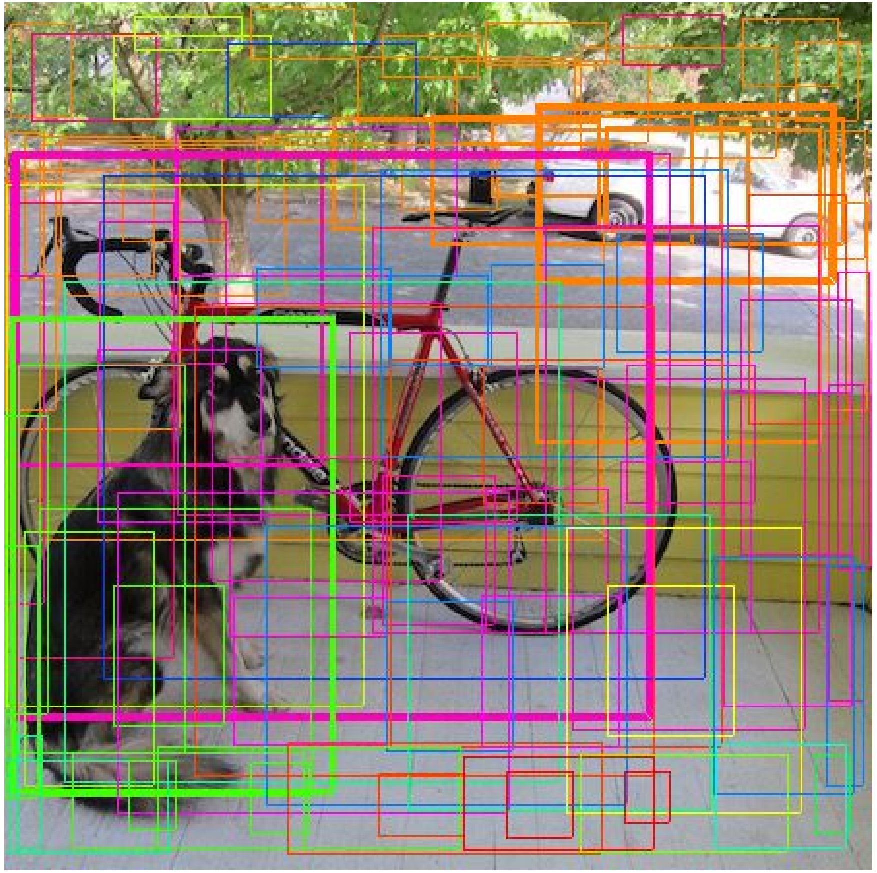
Dog

Dining Table

Eg.
Dog = 0.8
Cat = 0
Bike = 0

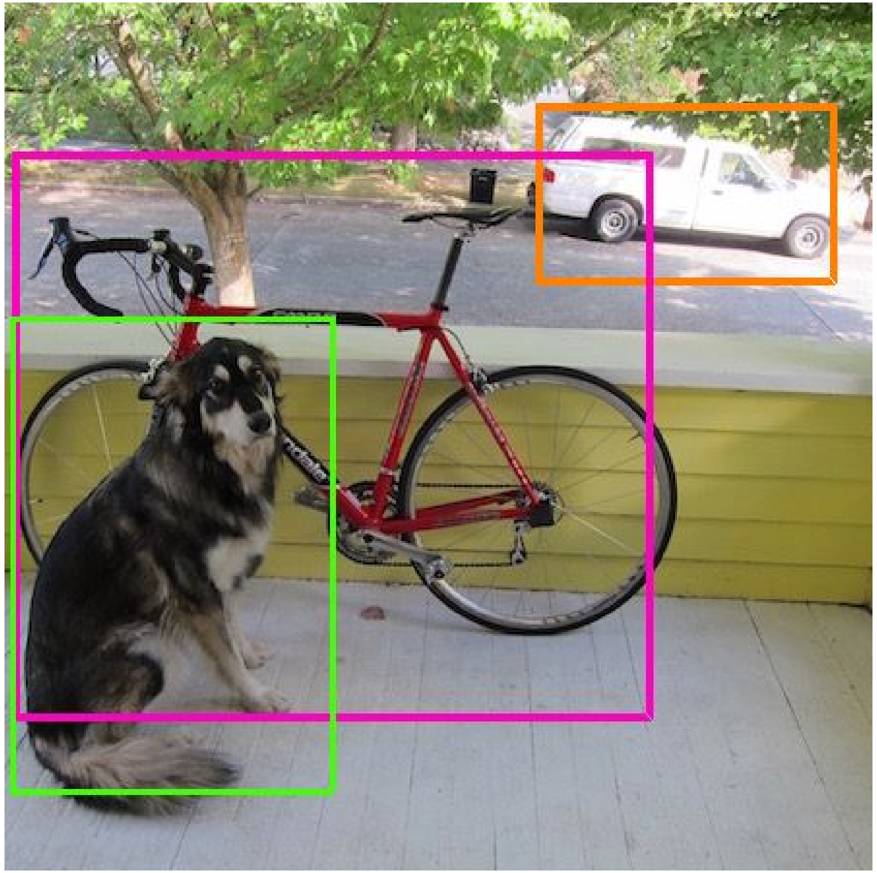


Then we combine the box and class predictions.



$P(\text{class}|\text{Object}) * P(\text{Object})$
$=P(\text{class})$

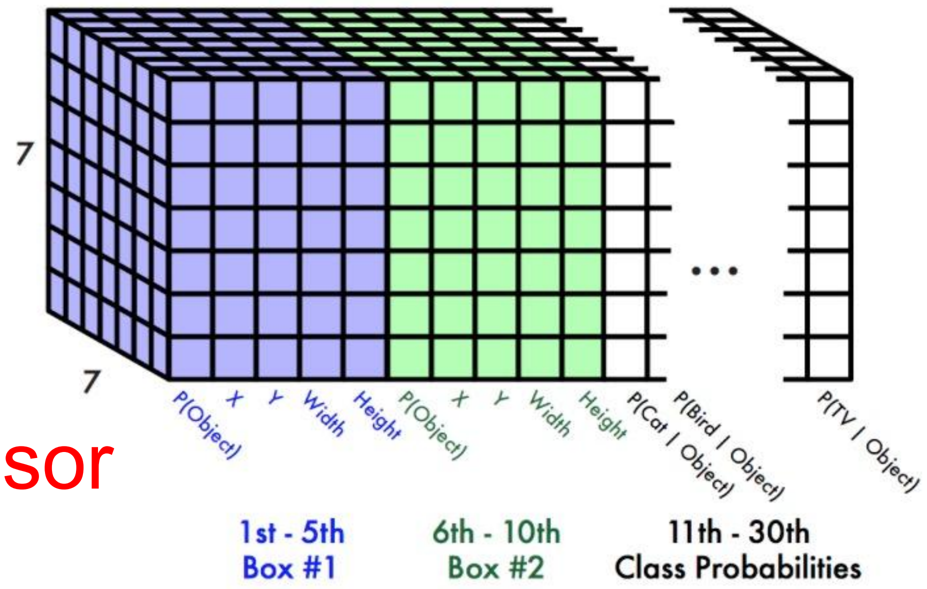
Finally we do threshold detections and NMS



https://docs.google.com/presentation/d/1kAa7NOamBt4caIBU9iHgT8a86RRHz9Yz2oh4-GTdX6M/edit#slide=id.g151008b386_0_44

Each cell predicts:

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

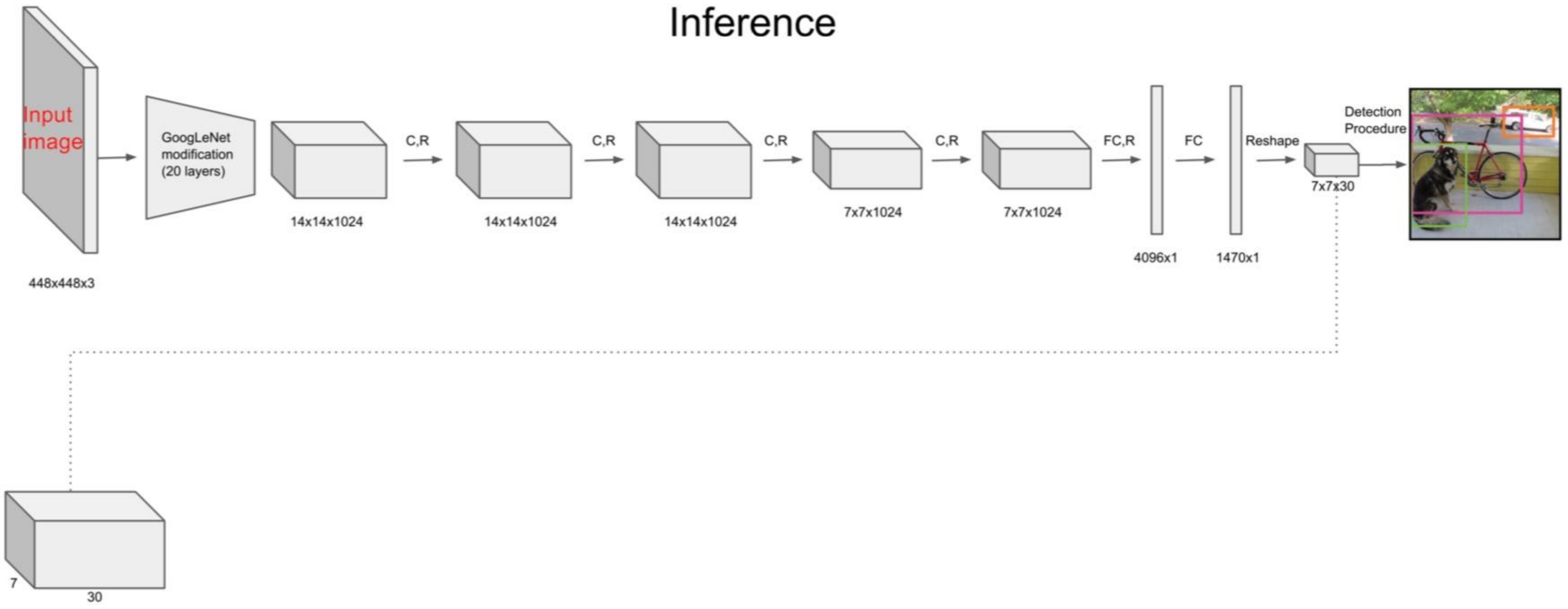


S * S * (B * 5 + C) tensor

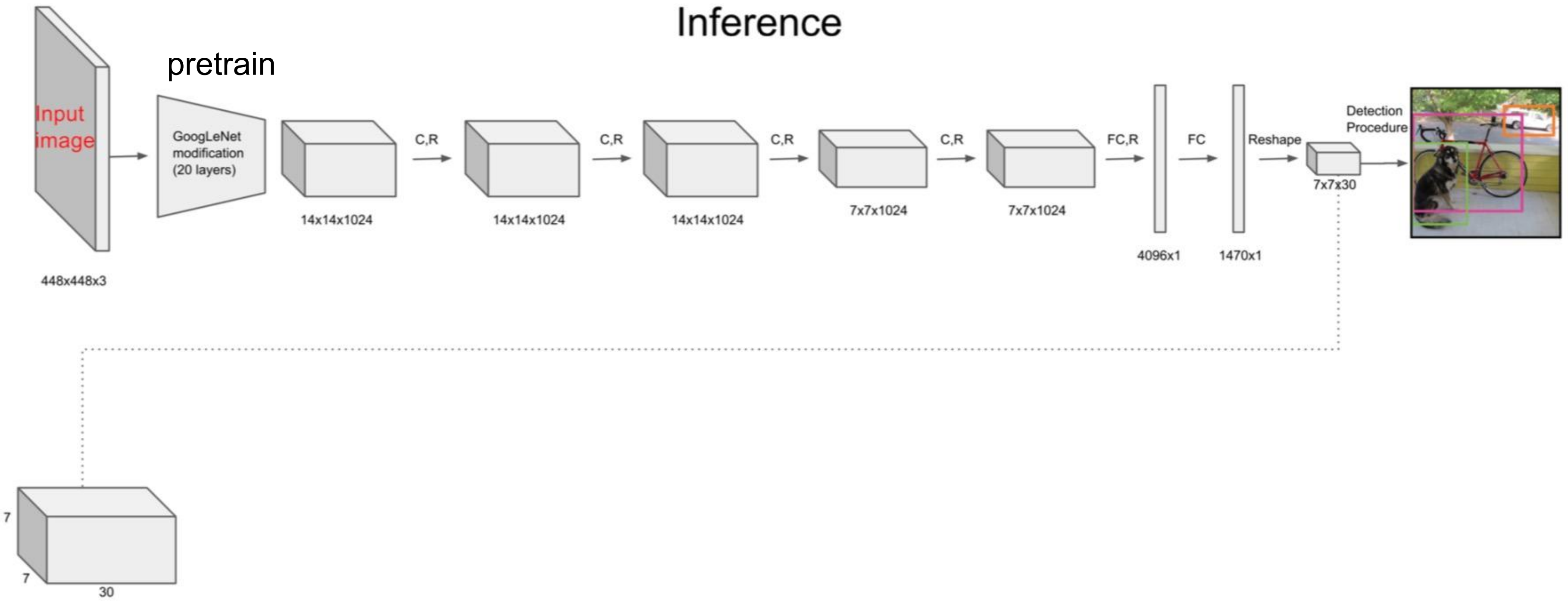
Network

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

Inference

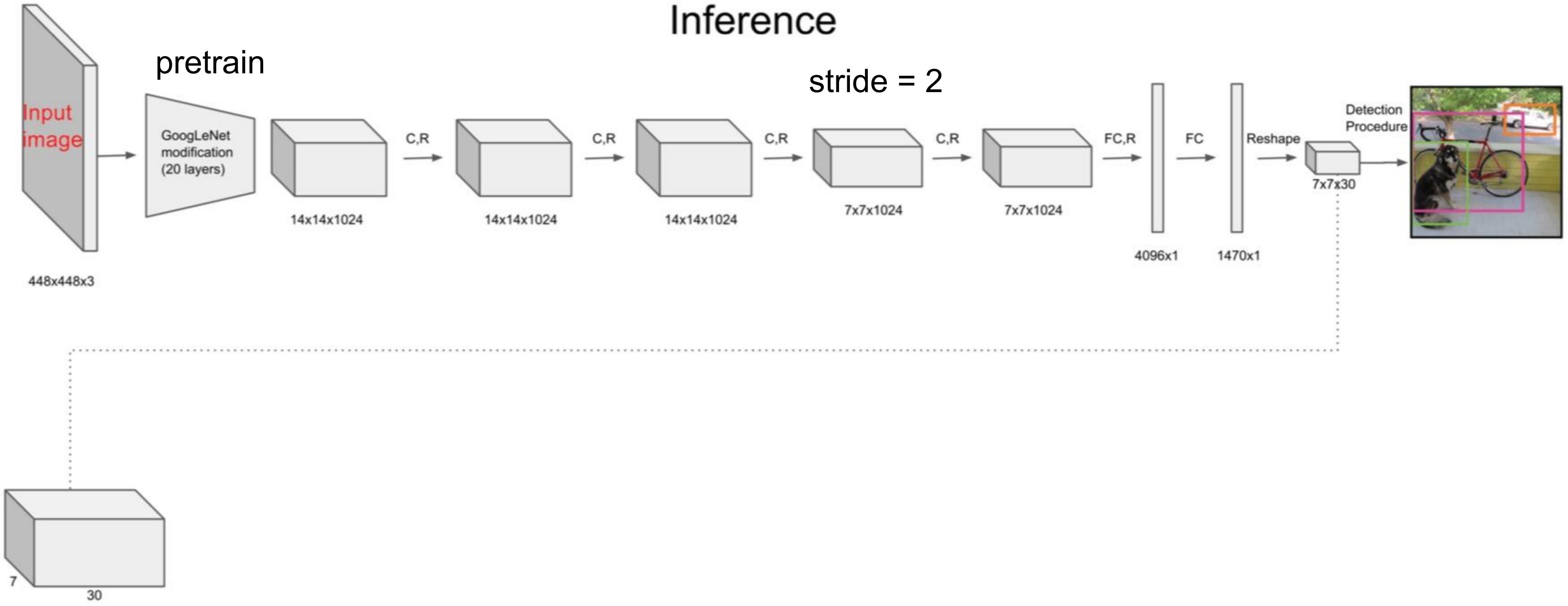


Inference


pretrain
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
7
7
30

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

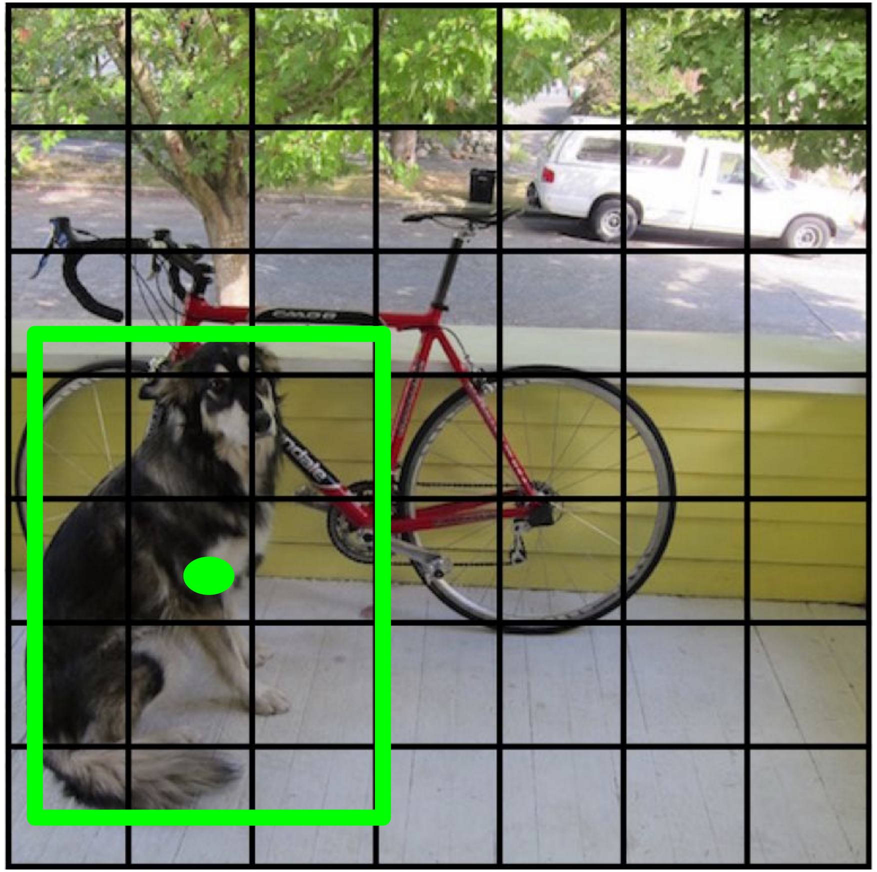
Inference



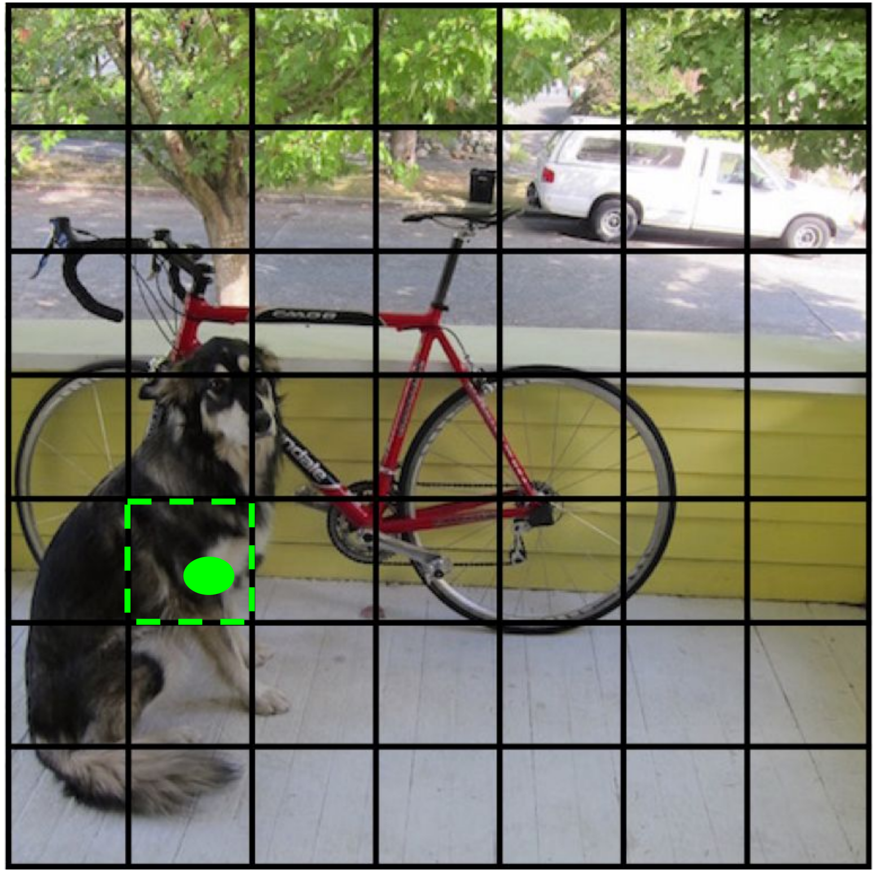
Train

https://docs.google.com/presentation/d/1kAa7NOamBt4calBU9iHgT8a86RRHz9Yz2oh4-GTdX6M/edit#slide=id.g151008b386_0_44

During training, match example to the right cell

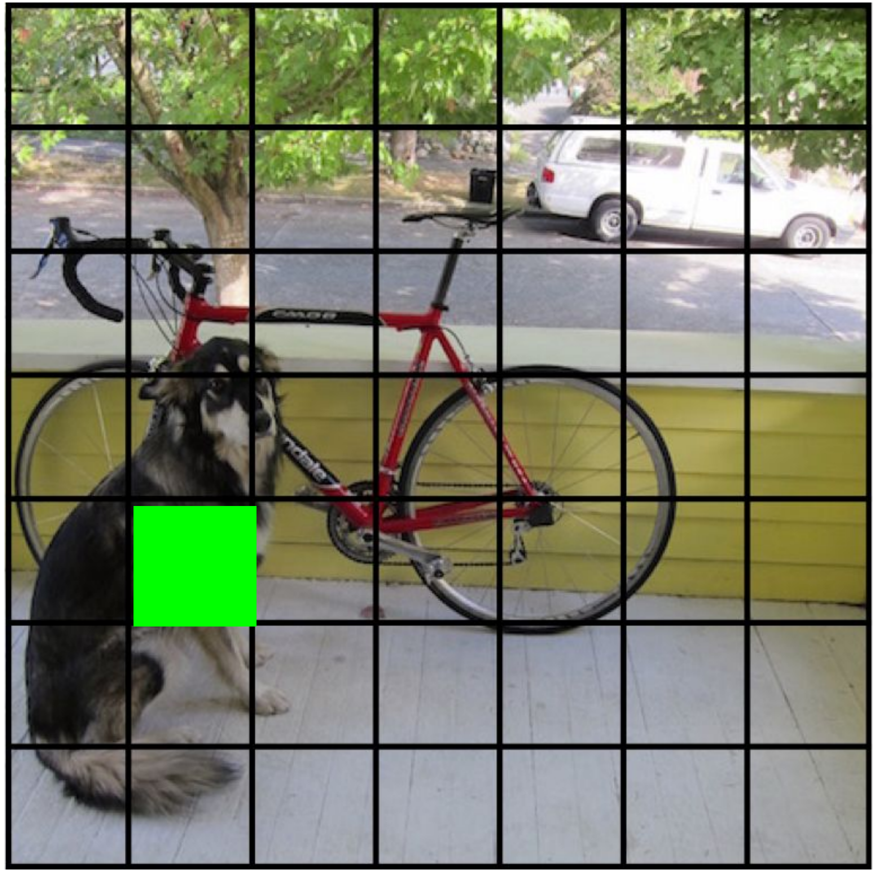


During training, match example to the right cell

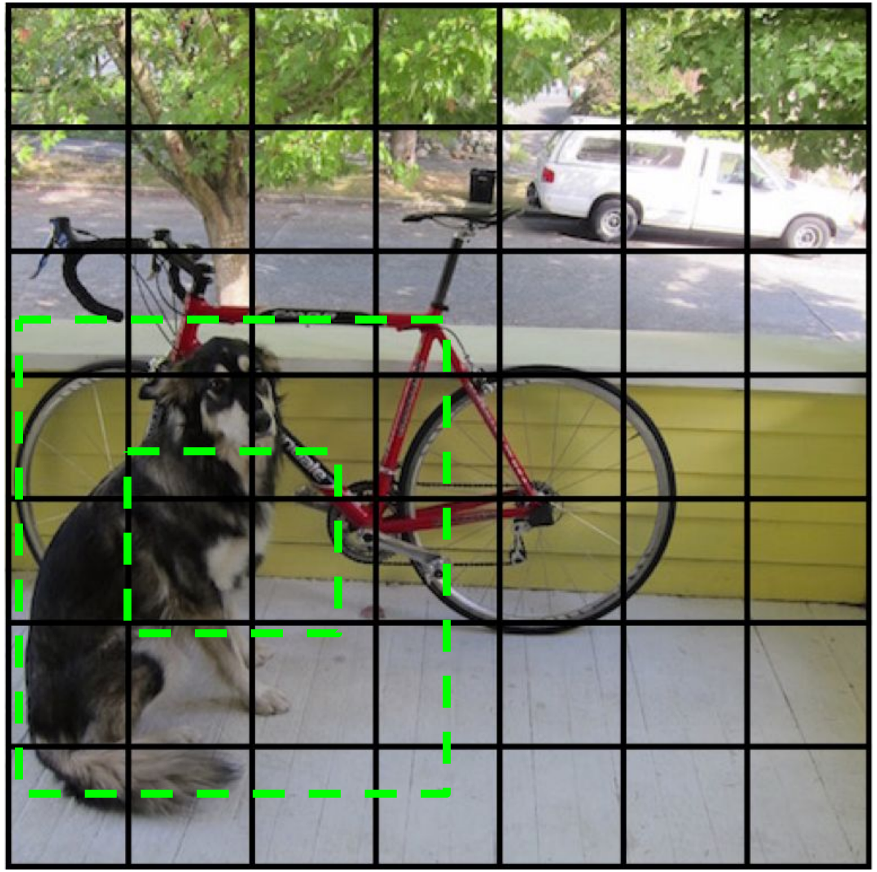


Adjust that cell's class prediction

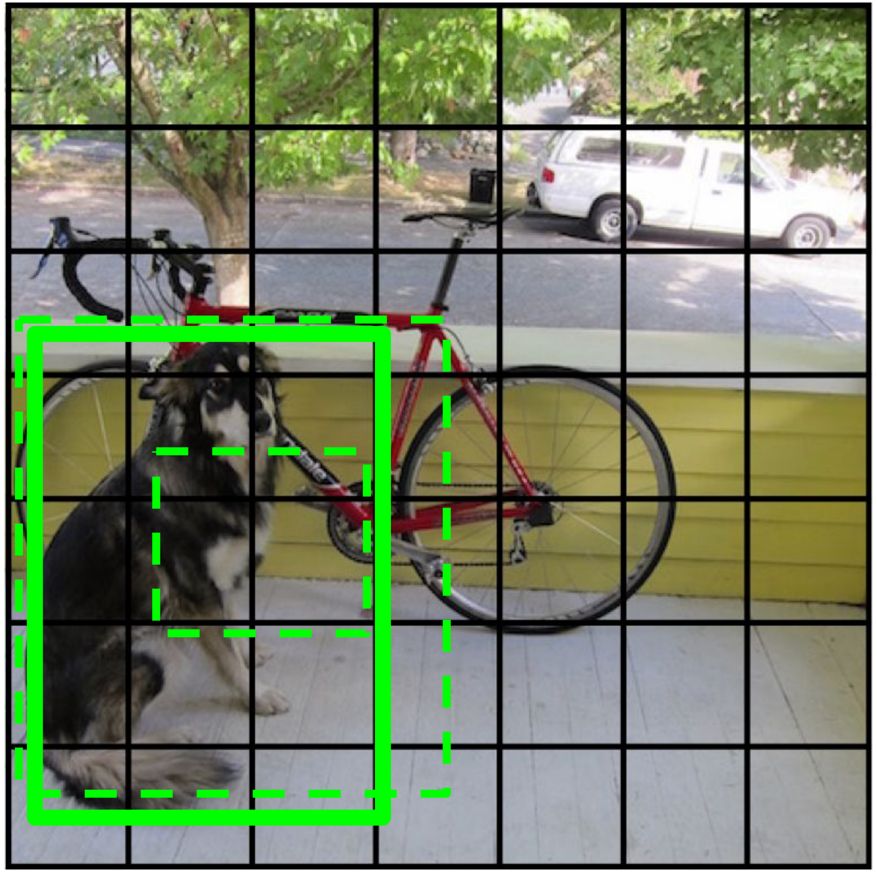
Dog = 1
Cat = 0
Bike = 0
...



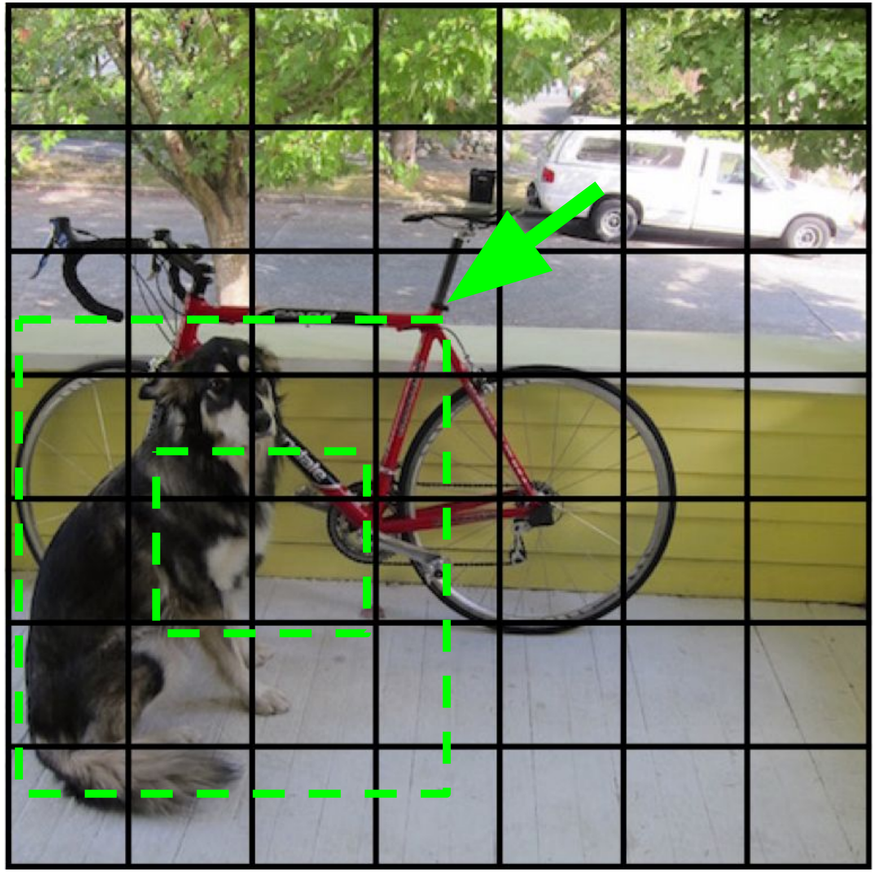
Look at that cell’s predicted boxes



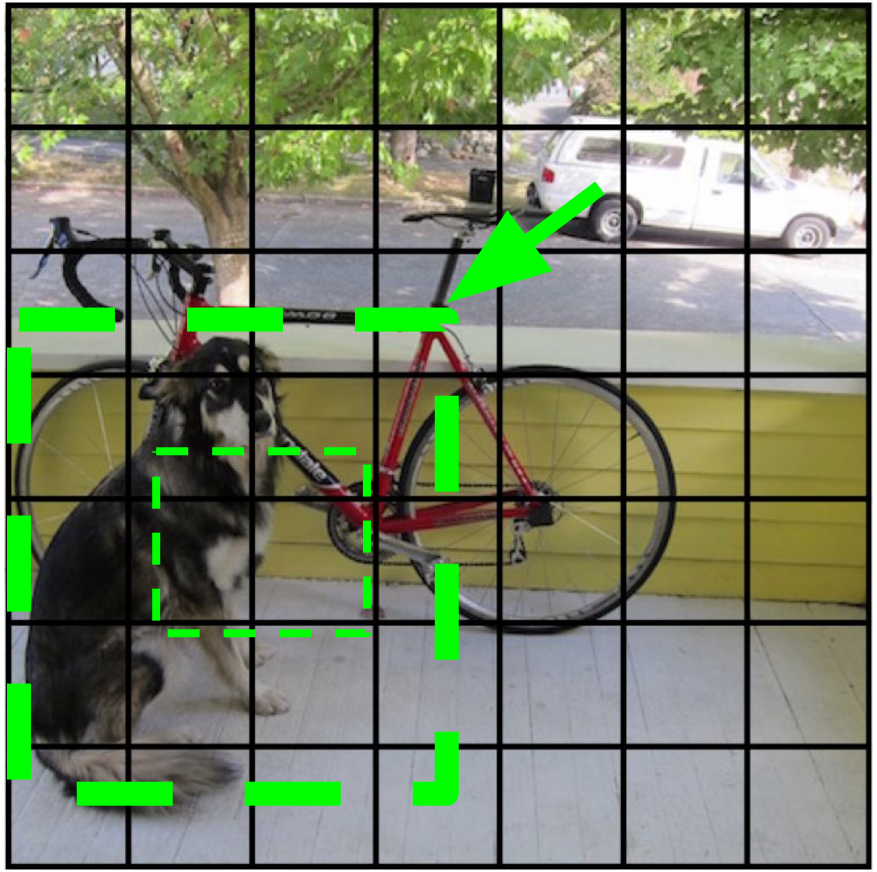
Find the best one, adjust it, increase the confidence



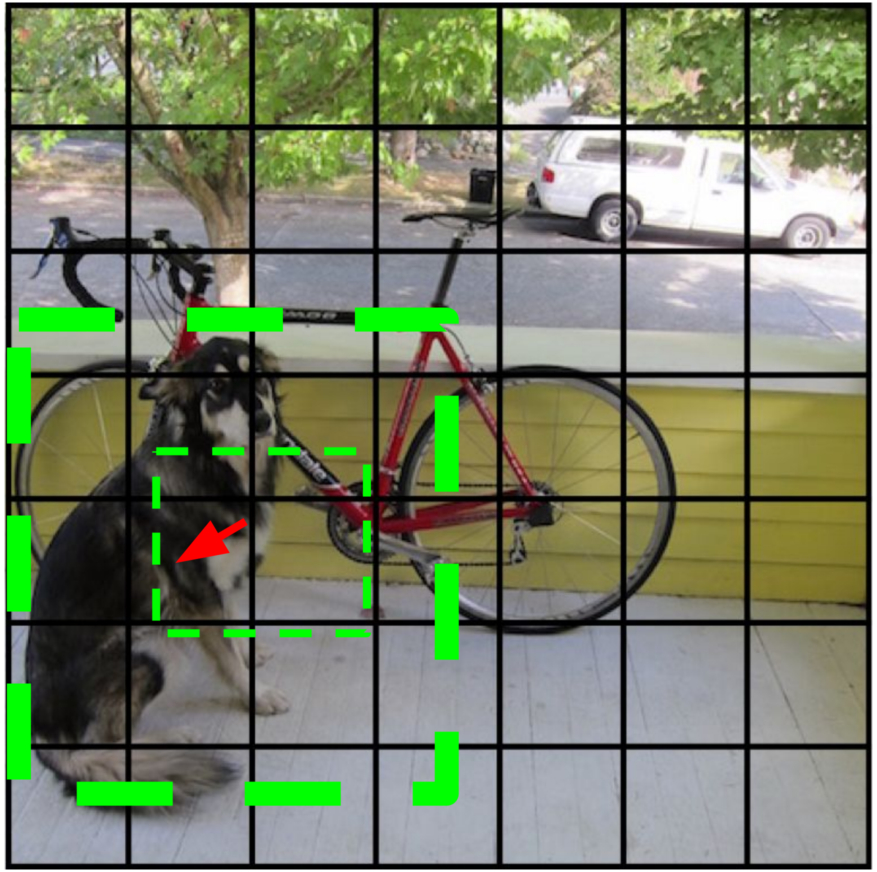
Find the best one, adjust it, increase the confidence



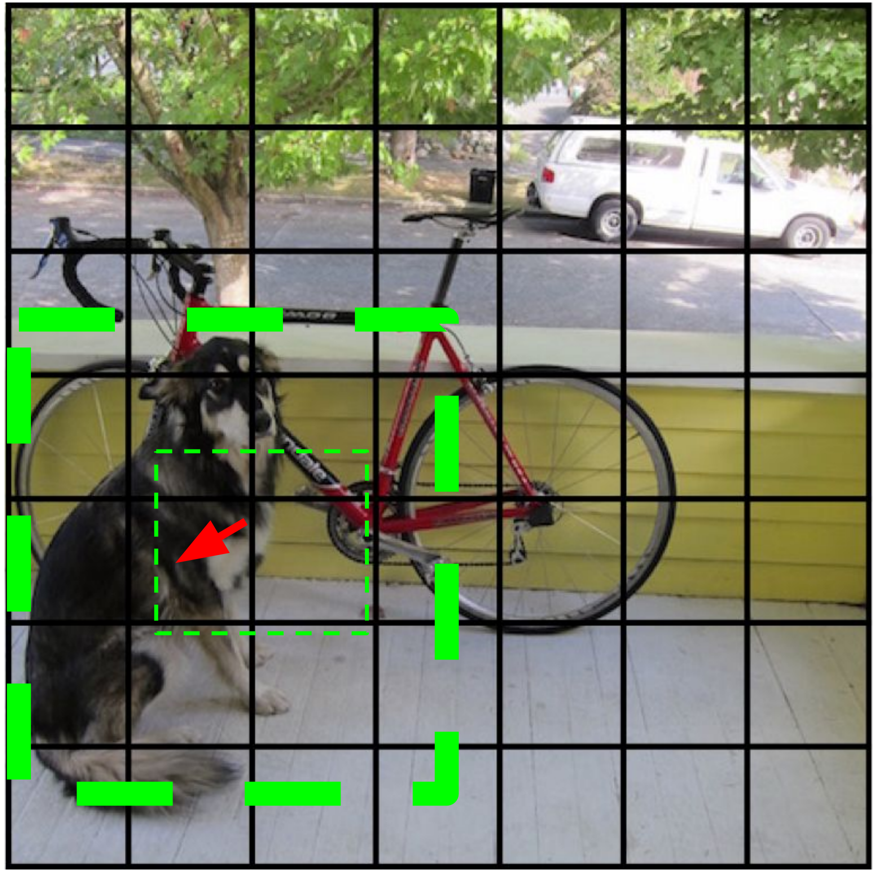
Find the best one, adjust it, increase the confidence



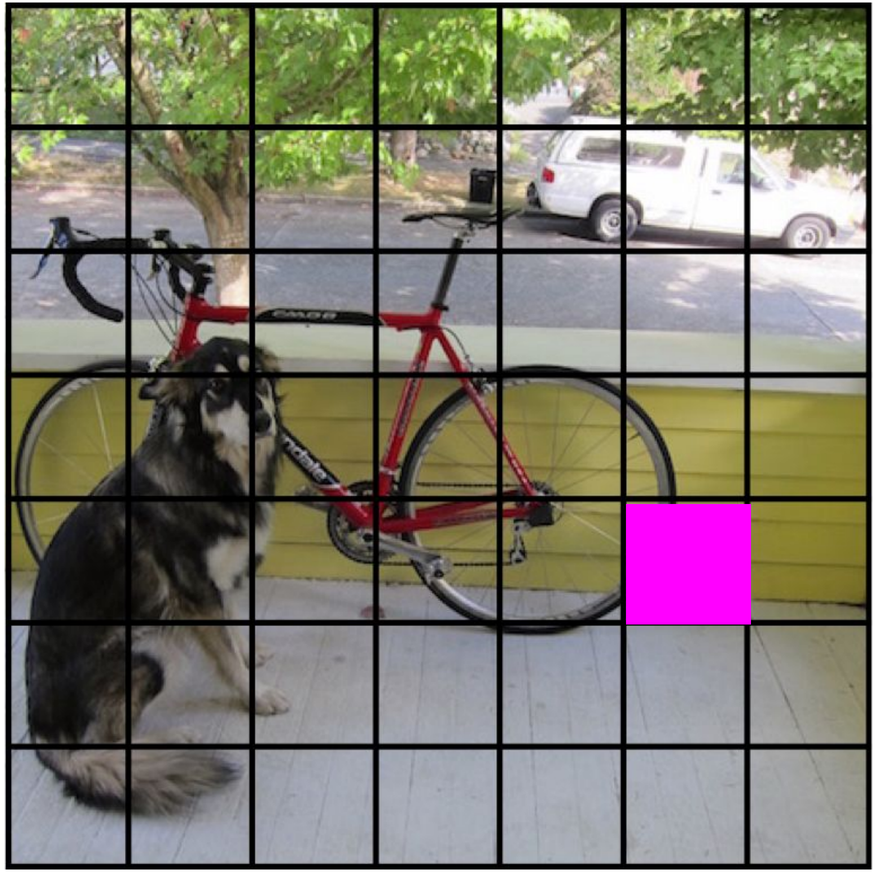
Decrease the confidence of the other box



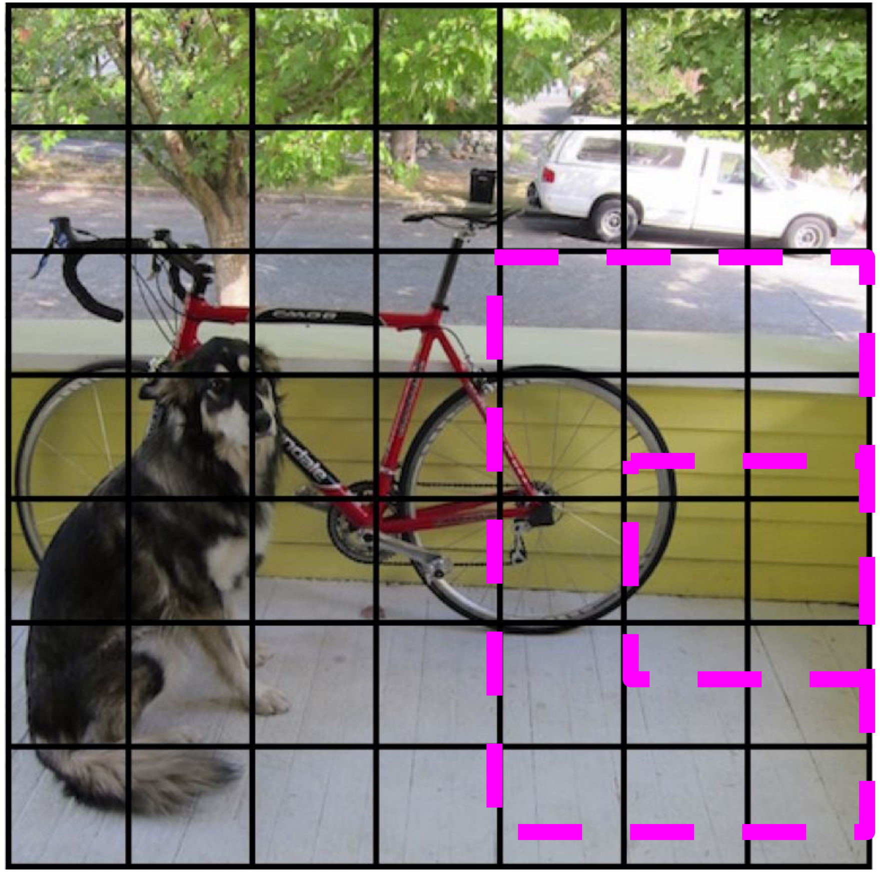
Decrease the confidence of the other box



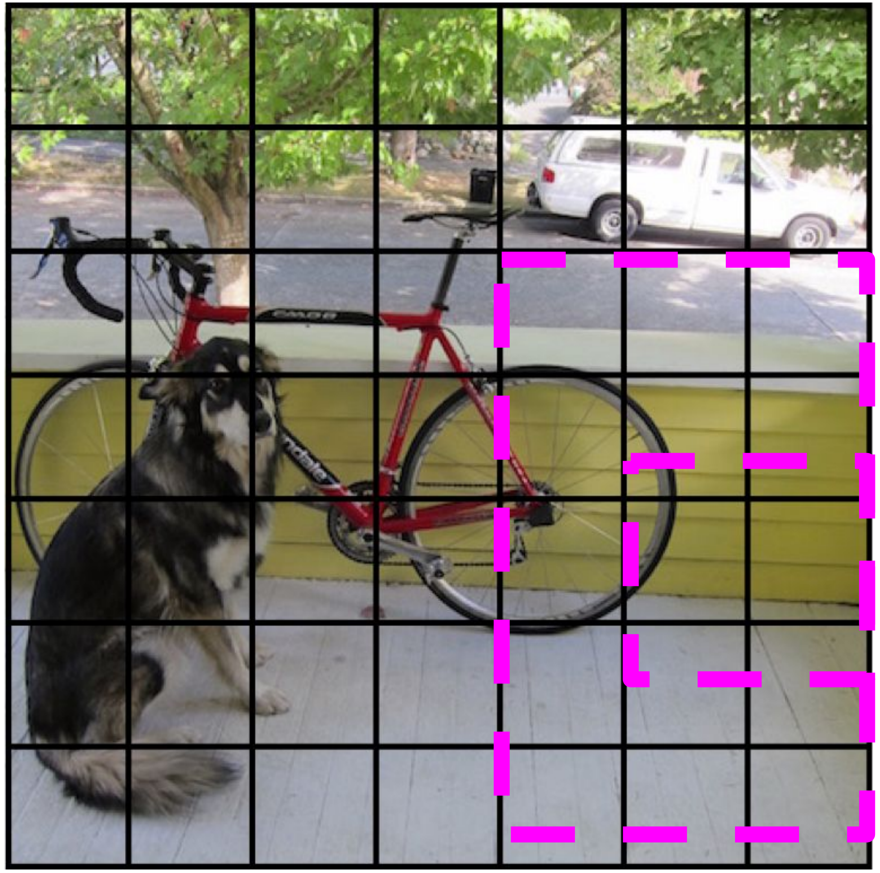
Some cells don’t have any ground truth detections!



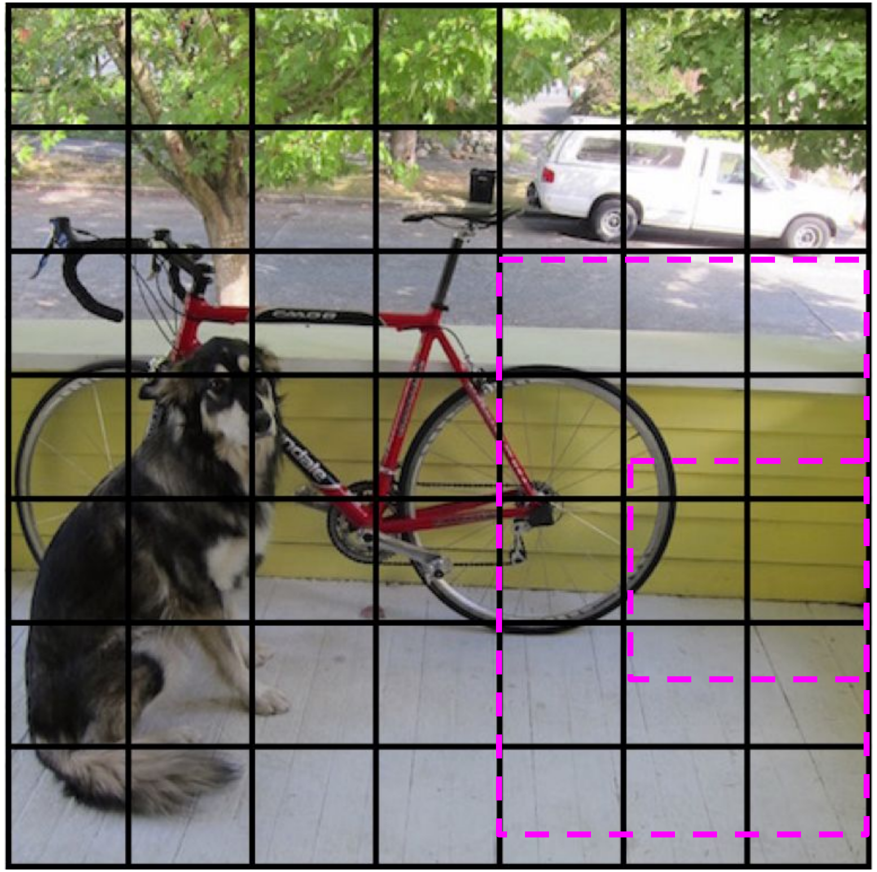
Some cells don’t have any ground truth detections!



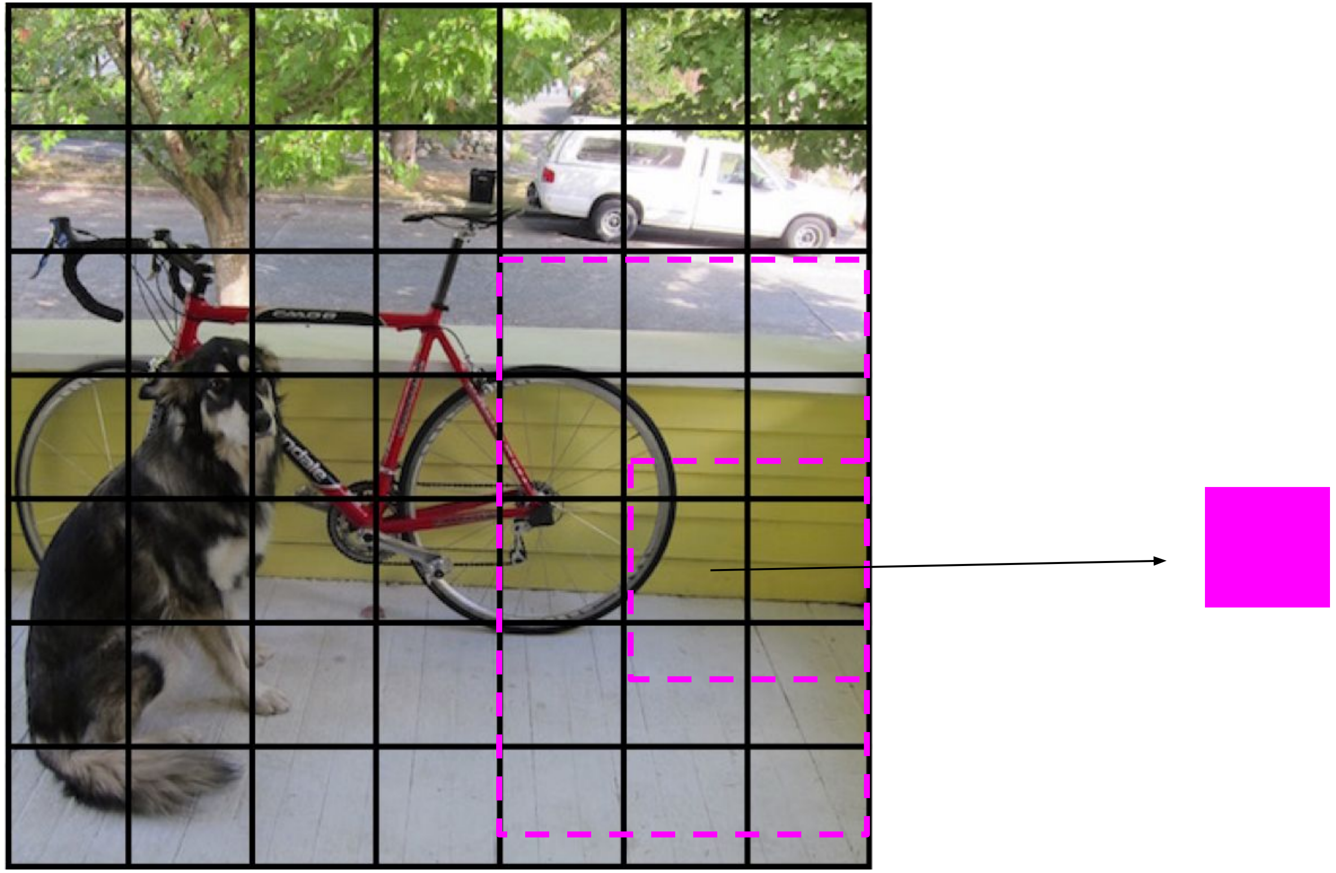
Decrease the confidence of boxes boxes



Decrease the confidence of these boxes



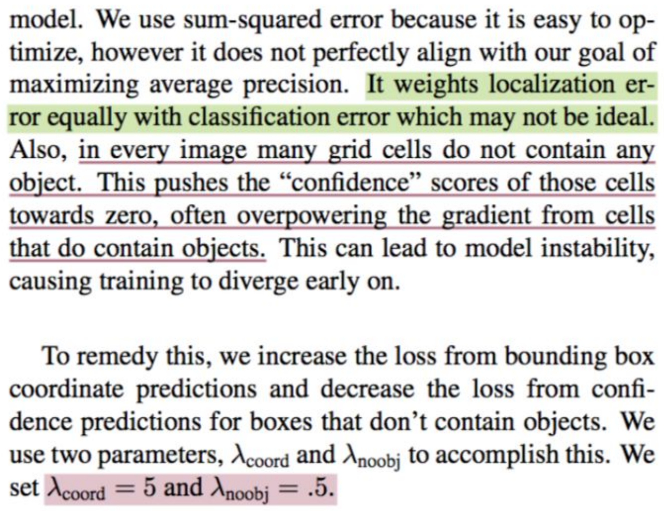
Don’t adjust the class probabilities or coordinates



Loss Function (sum-squared error)

loss function:

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)
\end{aligned}
$$



model. We use sum-squared error because it is easy to optimize, however it does not perfectly align with our goal of maximizing average precision. It weights localization error equally with classification error which may not be ideal. Also, in every image many grid cells do not contain any object. This pushes the "confidence" scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model instability, causing training to diverge early on.

To remedy this, we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects. We use two parameters, λ_{coord} and λ_{noobj} to accomplish this. We set $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{noobj}} = .5$.

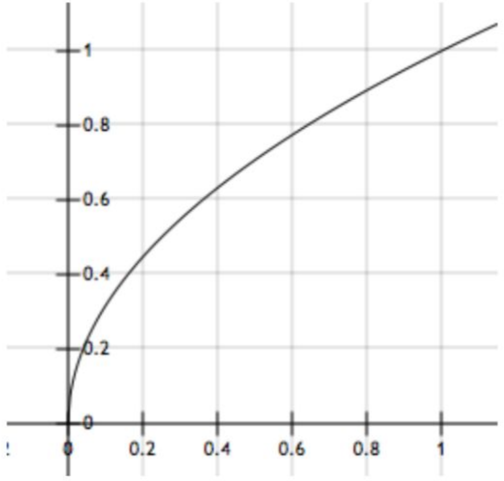
$$\lambda_{\text{coord}}=5, \ \ \lambda_{\text{noobj}}=0.5$$

Loss Function (sum-squared error)

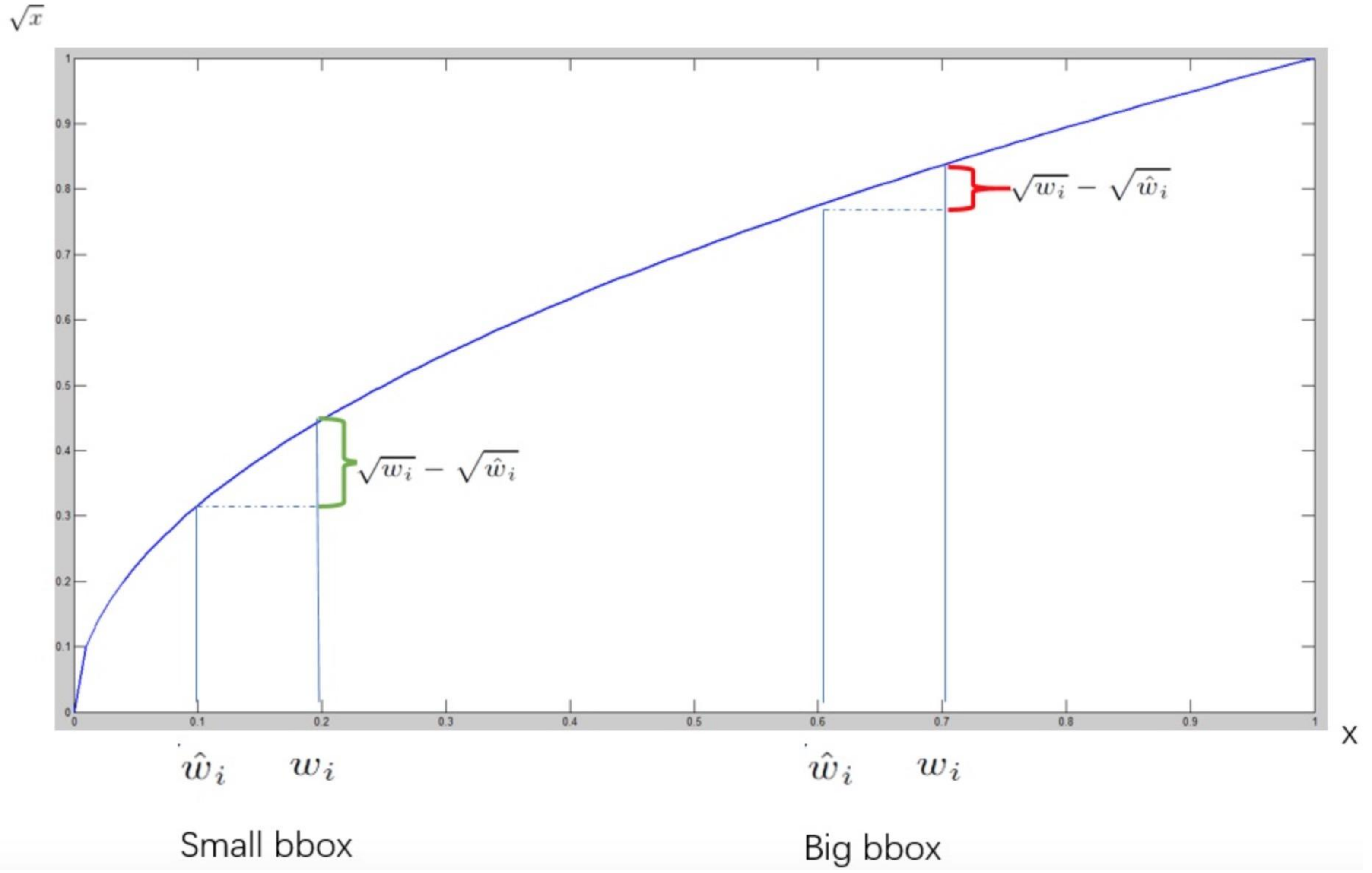
loss function:

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \boxed{\left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]}$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.



https://www.slideshare.net/TaegyunJeon1/pr12-you-only-look-once-yolo-unified-realtime-object-detection?from_action=save



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

Loss Function (sum-squared error)

loss function:

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \boxed{\mathbb{1}_{ij}^{\text{obj}}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \boxed{\mathbb{1}_{ij}^{\text{obj}}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \boxed{\mathbb{1}_{ij}^{\text{obj}}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \boxed{\mathbb{1}_{ij}^{\text{noobj}}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \boxed{\mathbb{1}_{i}^{\text{obj}}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)
\end{aligned}
$$

$\boxed{\mathbb{1}_{ij}^{obj}}$ **The *j*th bbox predictor** in ***cell i*** is "responsible" for that prediction

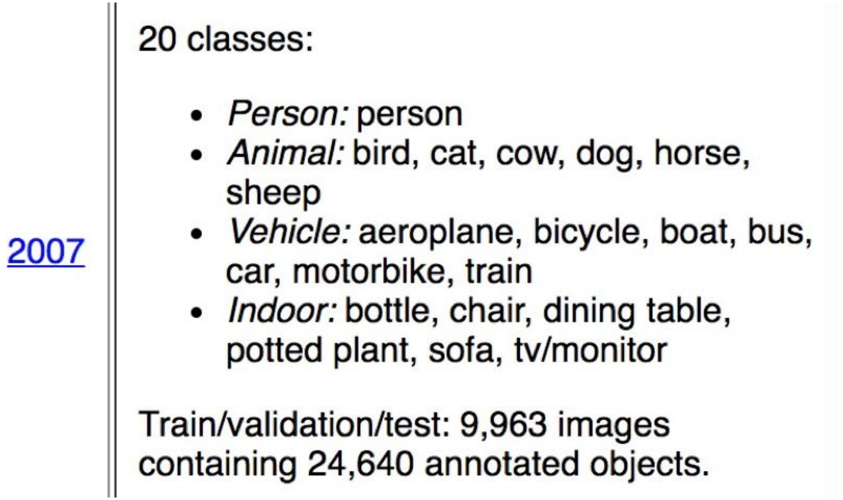
$\boxed{\mathbb{1}_{ij}^{noobj}}$

$\boxed{\mathbb{1}_{i}^{obj}}$ If object appears in ***cell i***

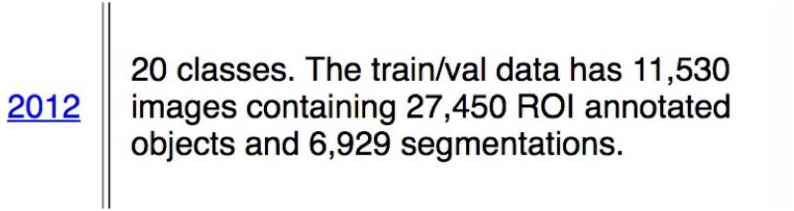
Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

Experiments

- Datasets
- PASCAL VOC 2007 & VOC 2012



2007	20 classes: • *Person:* person • *Animal:* bird, cat, cow, dog, horse, sheep • *Vehicle:* aeroplane, bicycle, boat, bus, car, motorbike, train • *Indoor:* bottle, chair, dining table, potted plant, sofa, tv/monitor Train/validation/test: 9,963 images containing 24,640 annotated objects.

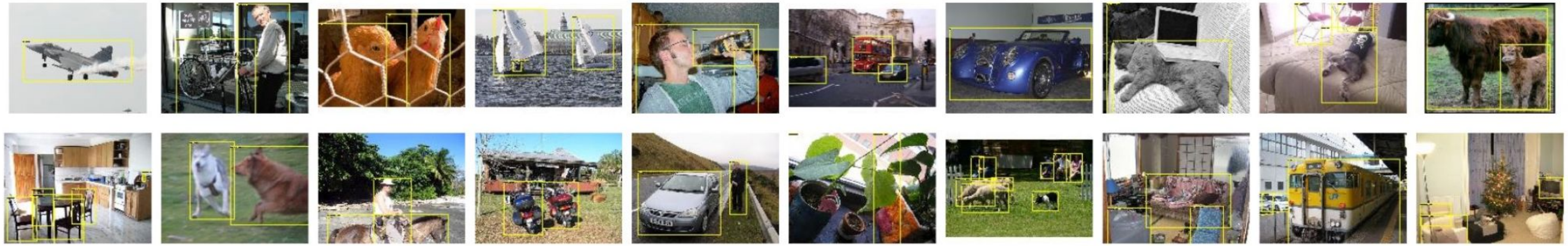


2012	20 classes. The train/val data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations.

Experiments

- Datasets

20 classes



Accurate object detection is slow!

	Pascal 2007 mAP	**Speed**	
DPM v5	33.7	.07 FPS	14 s/img

Ref: https://pjreddie.com/publications/

Accurate object detection is slow!

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img

Ref: https://pjreddie.com/publications/

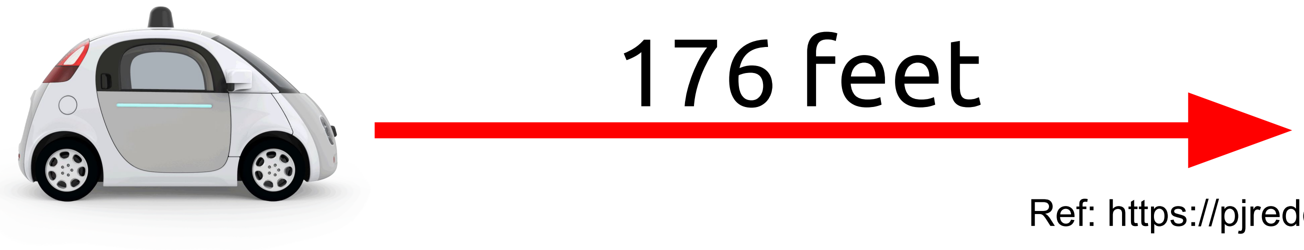
Accurate object detection is slow!

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img

Ref: https://pjreddie.com/publications/

Accurate object detection is slow!

	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



Ref: https://pjreddie.com/publications/

Accurate object detection is slow!

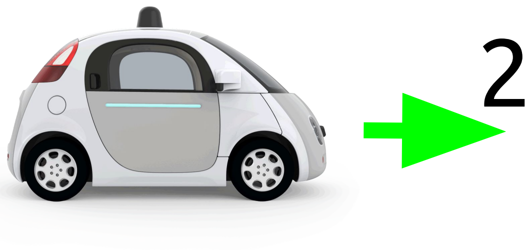
	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img

8 feet
12 feet

Ref: https://pjreddie.com/publications/

Accurate object detection is slow!

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



2 feet

Ref: https://pjreddie.com/publications/

Error Analysis

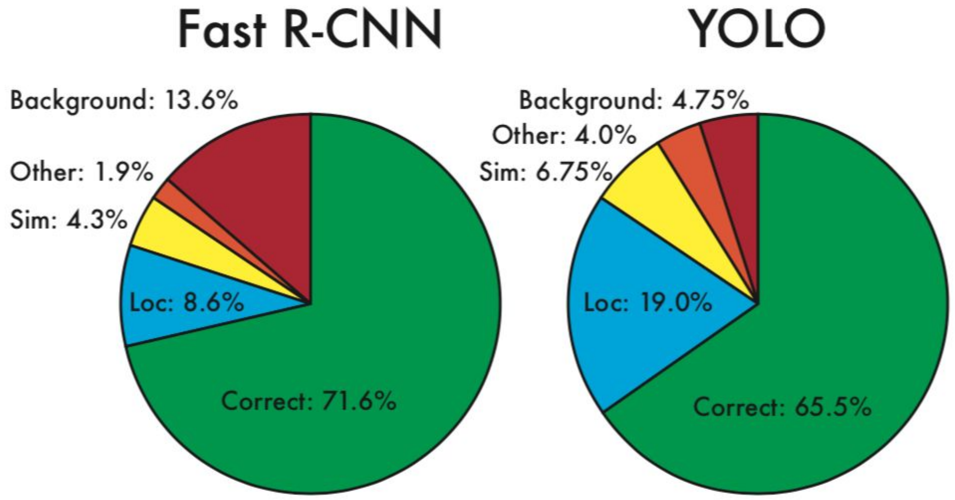


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

Loc: Localization Error

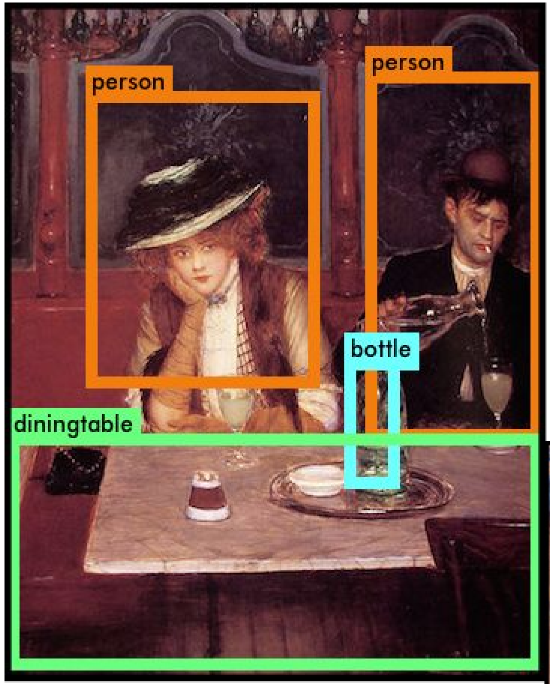
Correct class,

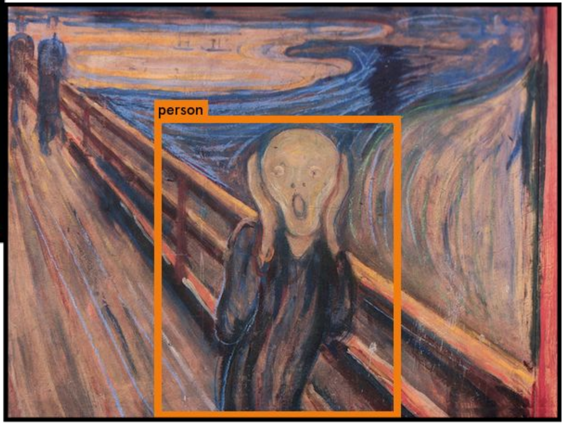
.1<IOU<.5

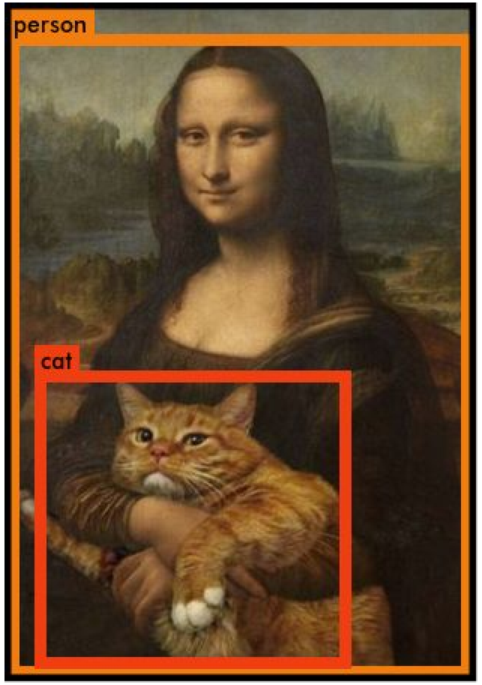
Background:

IOU<0.1

YOLO generalizes well to new domains (like art)

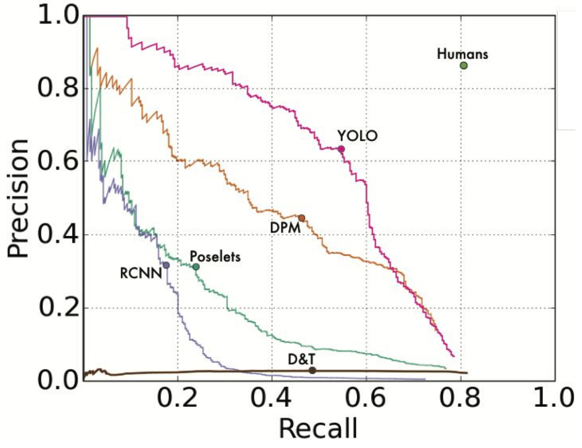






Ref: https://pjreddie.com/publications/

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

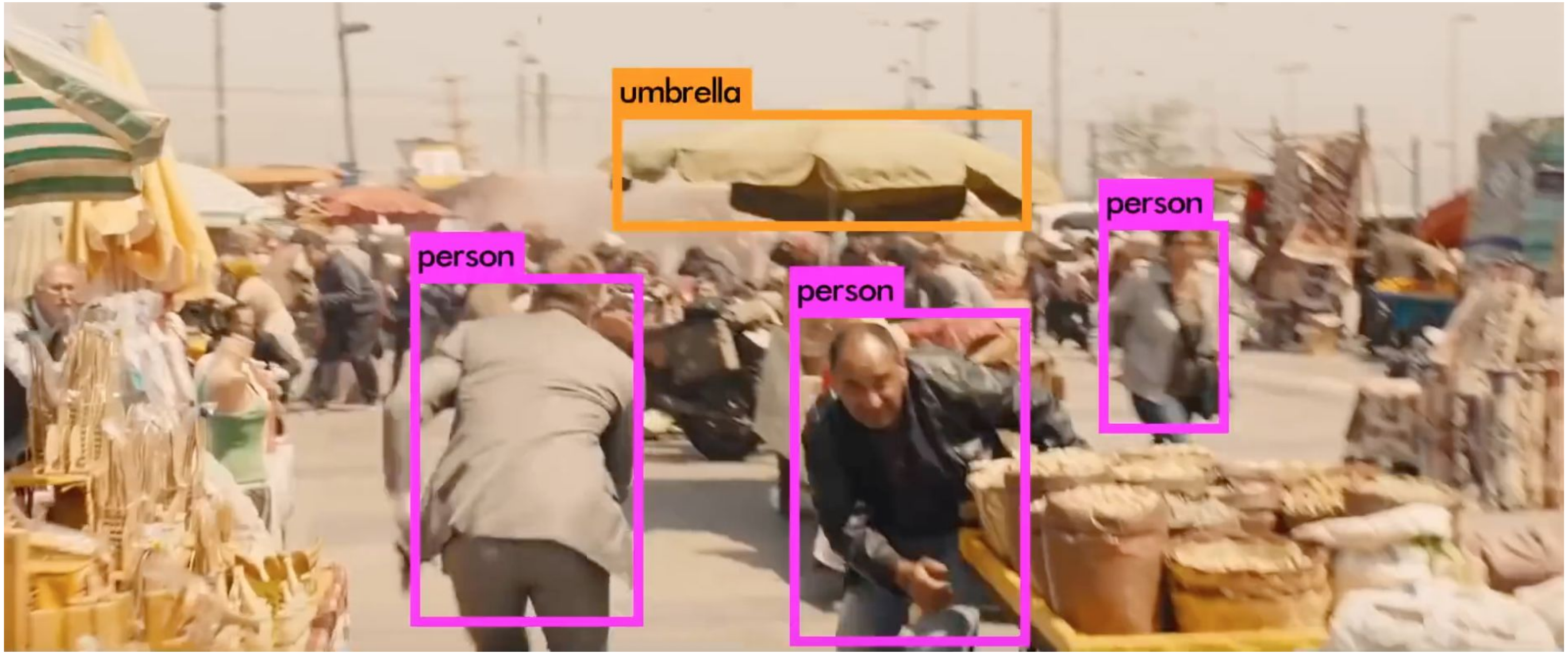


	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32

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.

Demo



Strengths and Weaknesses

- Strengths:
  - Fast: 45fps, smaller version 155fps
  - End2end training
  - Background error is low

Strengths and Weaknesses

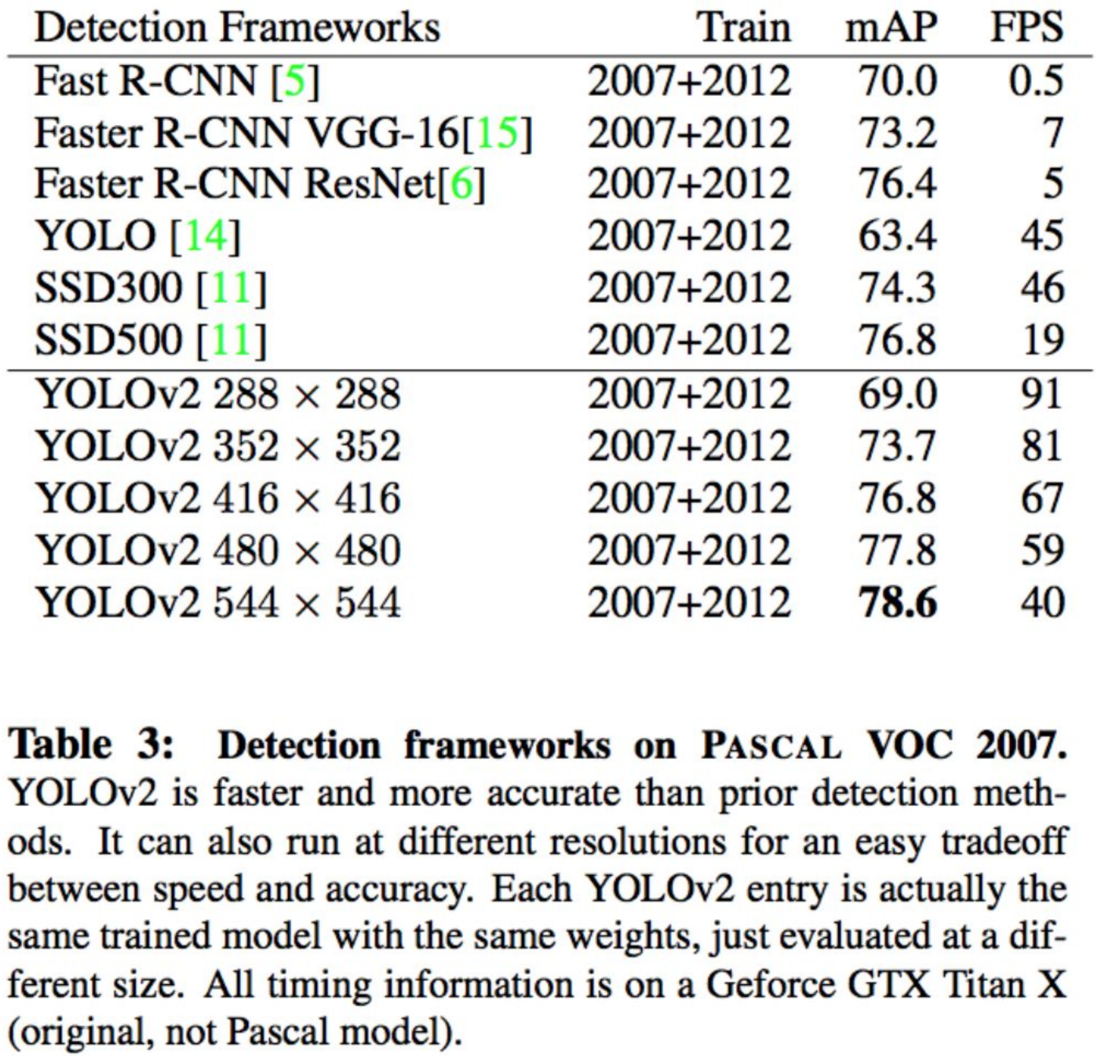
- Weaknesses:
  - Performance is lower than state-of-art
  - Makes more localization errors

Open Questions

- How to determine the number of cell, bounding box and the size of the box
- Why normalization x,y,w,h even all the input images have the same resolution?
-

Extension Part

YOLOv2!



Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288×288	2007+2012	69.0	91
YOLOv2 352×352	2007+2012	73.7	81
YOLOv2 416×416	2007+2012	76.8	67
YOLOv2 480×480	2007+2012	77.8	59
YOLOv2 544×544	2007+2012	**78.6**	40

Table 3: Detection frameworks on PASCAL VOC 2007. YOLOv2 is faster and more accurate than prior detection methods. It can also run at different resolutions for an easy tradeoff between speed and accuracy. Each YOLOv2 entry is actually the same trained model with the same weights, just evaluated at a different size. All timing information is on a Geforce GTX Titan X (original, not Pascal model).

arXiv:1612.08242