

Object Recognition and Face Recognition

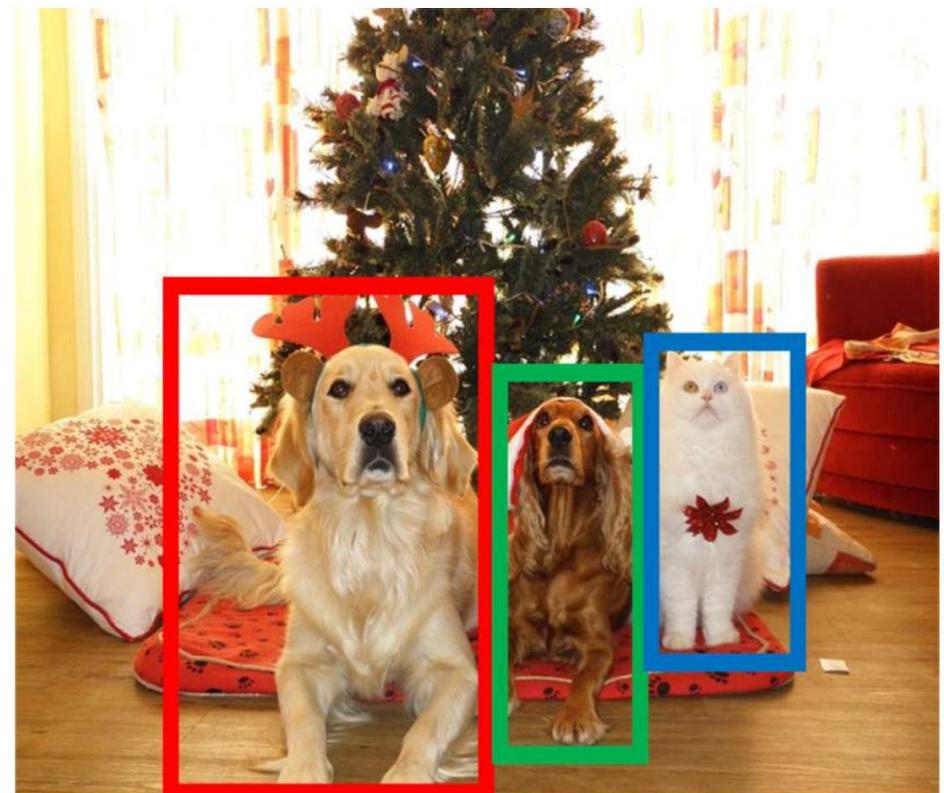
Object Detection: Task Definition

Input: Single RGB Image

Output: A set of detected objects;

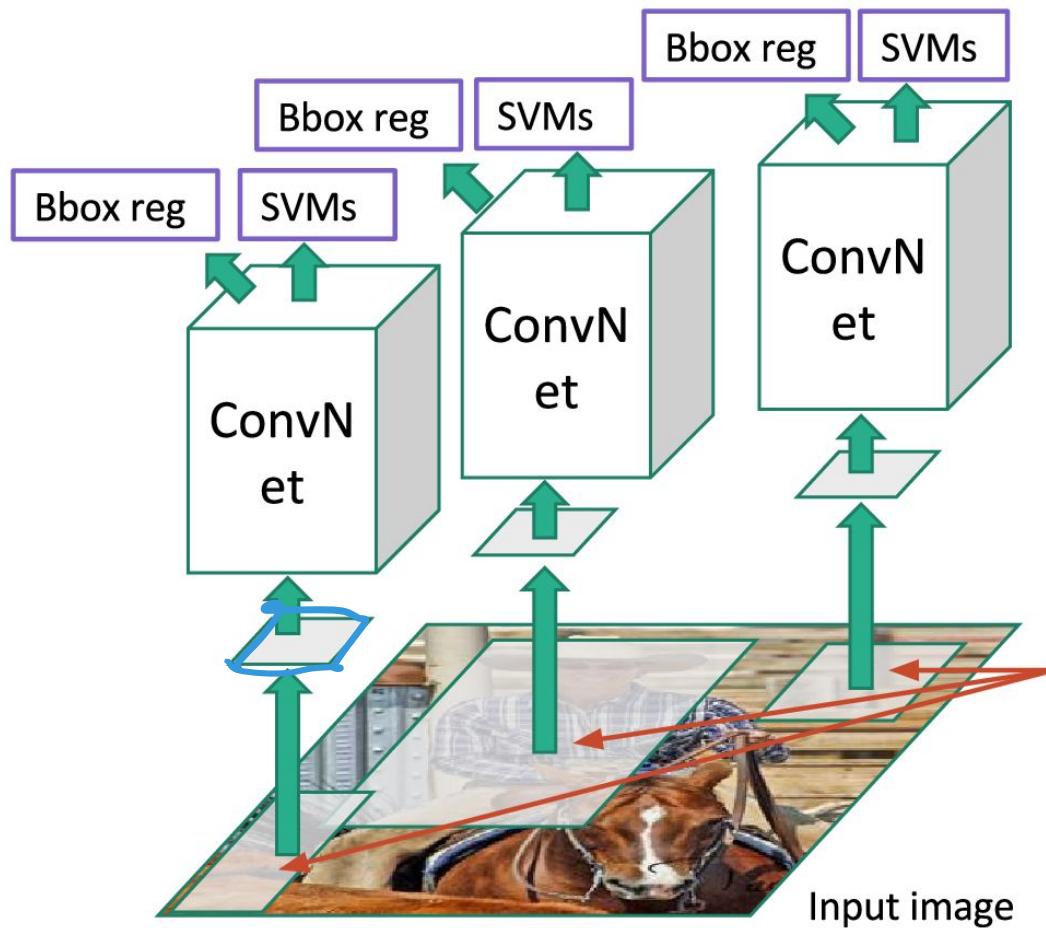
For each object predict:

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



Rcnn

R-CNN



Problem: Training is slow (84h),
takes a lot of disk space



Accurate object detection is slow!

	Pascal 2007 mAP	Speed
R-CNN	66.0	.05 FPS 20 s/img



Accurate object detection is slow!

	Pascal 2007 mAP	Speed
R-CNN	66.0	.05 FPS 20 s/img

60 miles/H (96.5 Km/H)

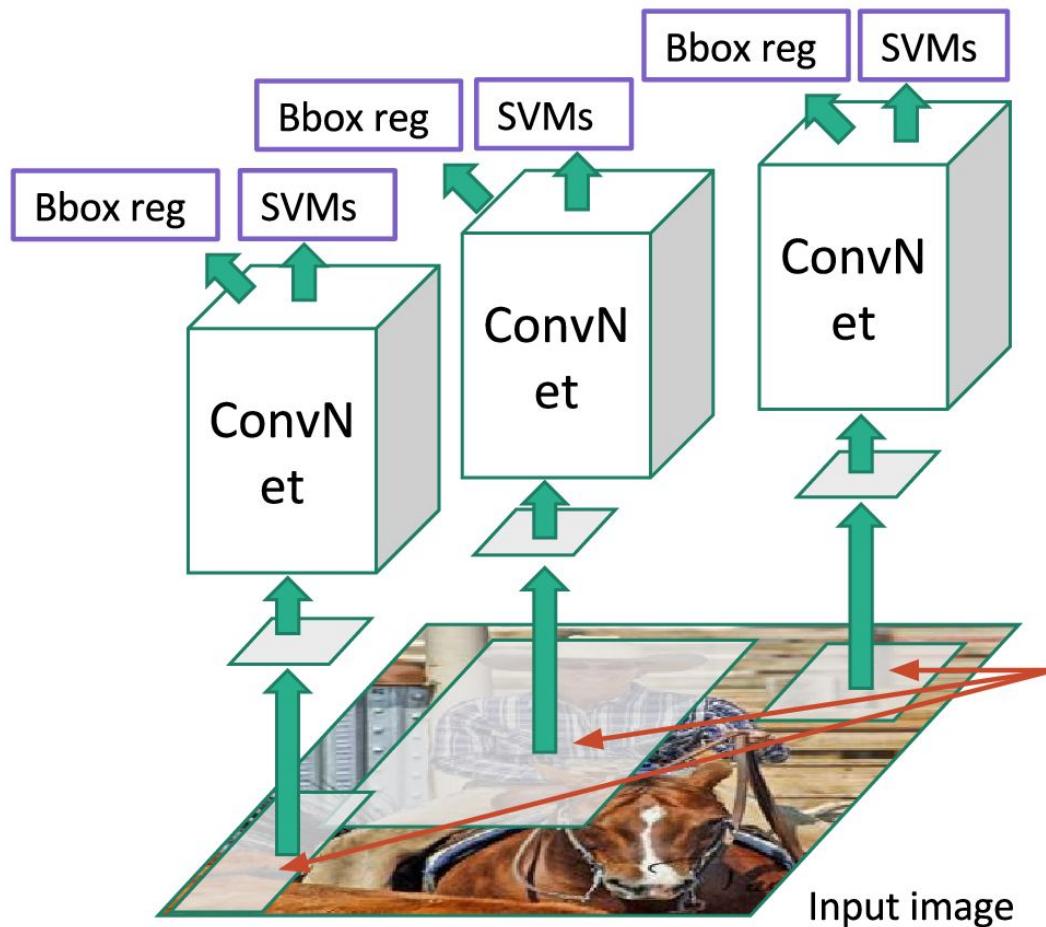


$\frac{1}{3}$ Mile, 1760 feet

20 sec



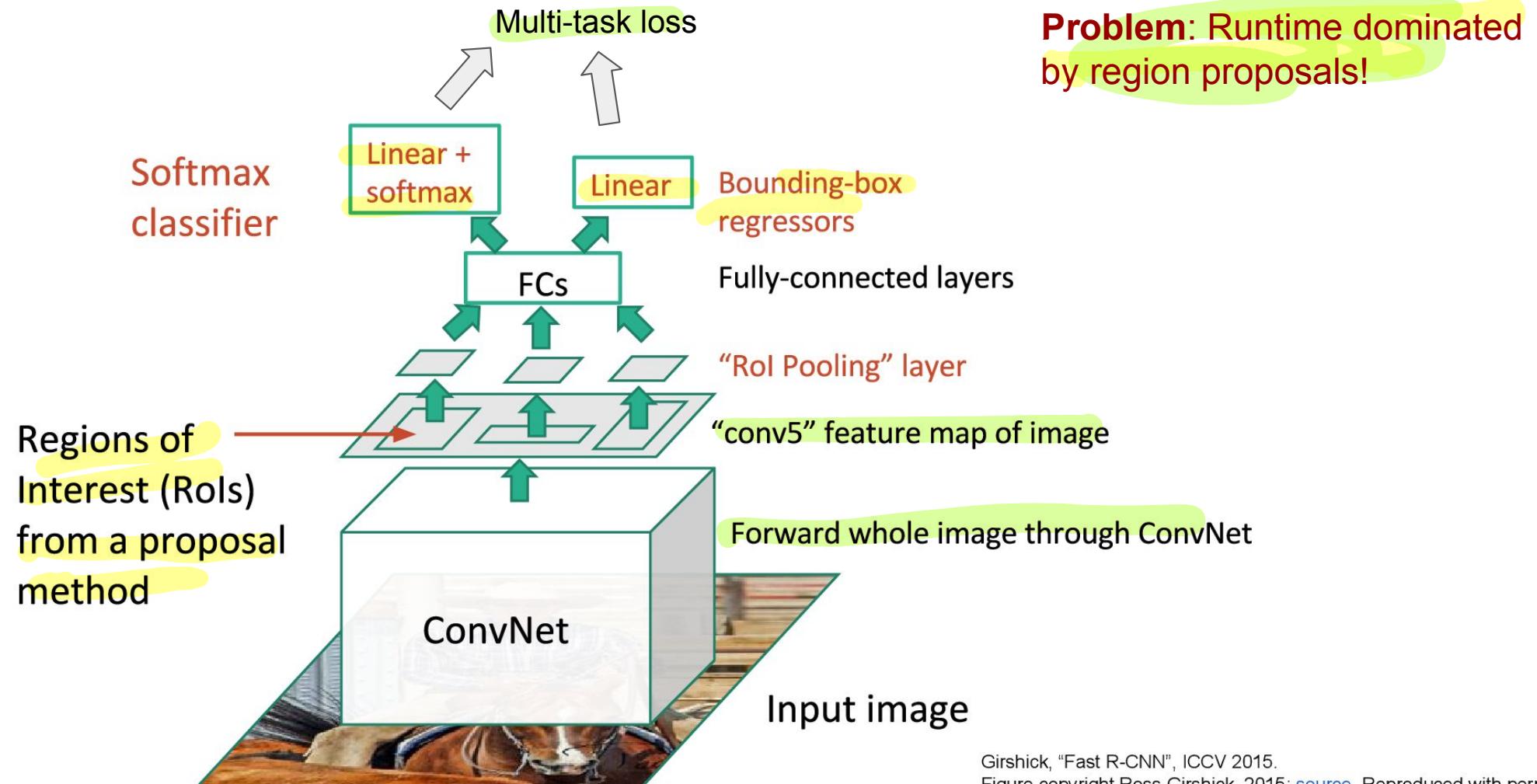
R-CNN



Problem: Training is **slow** (84h), takes a lot of disk space

Solution: Pass the image through convnet before cropping! Crop the conv feature instead!

Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015.

Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Accurate object detection is slow!

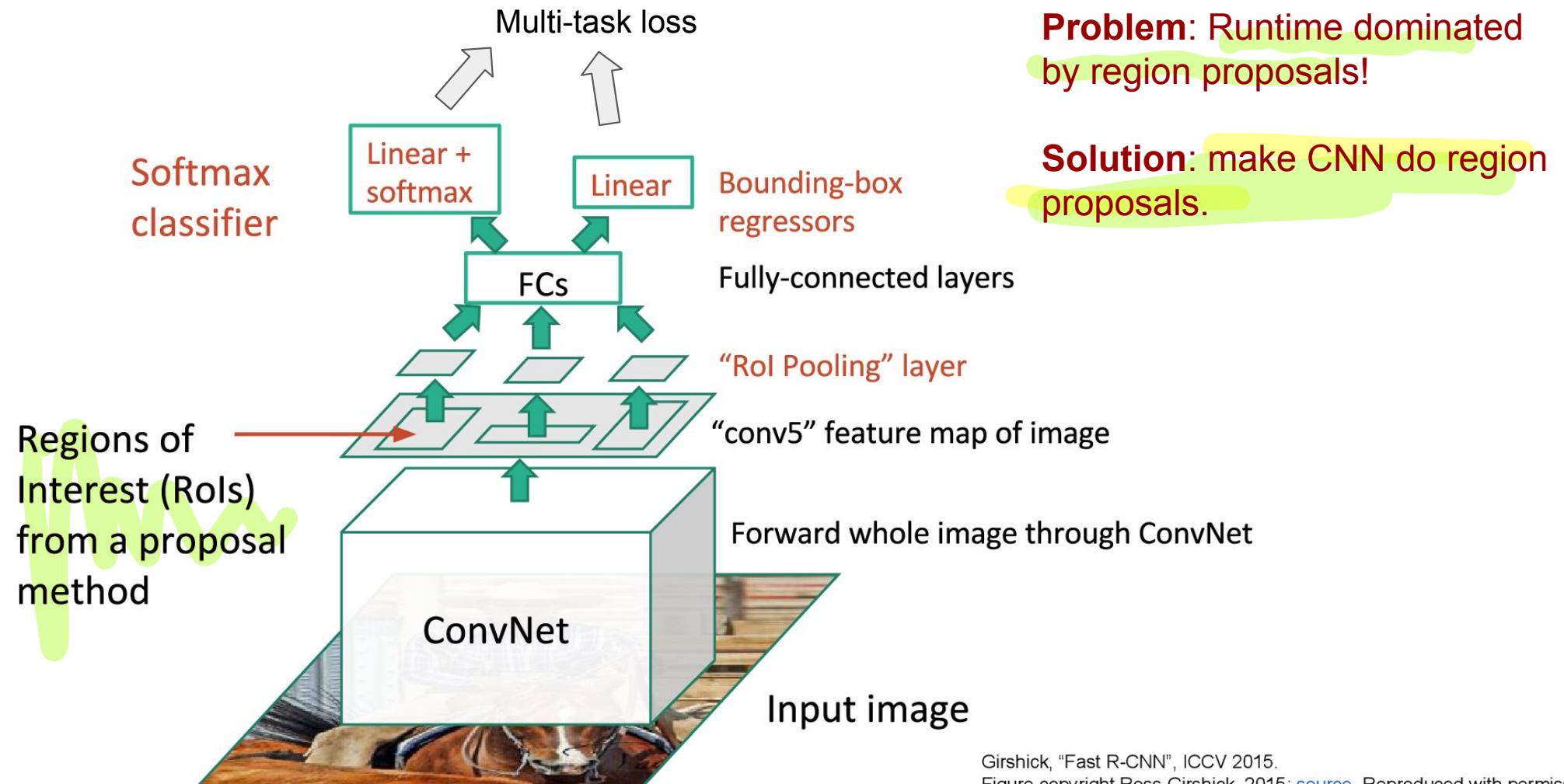
	Pascal 2007 mAP	Speed	
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img



176 feet



Fast R-CNN

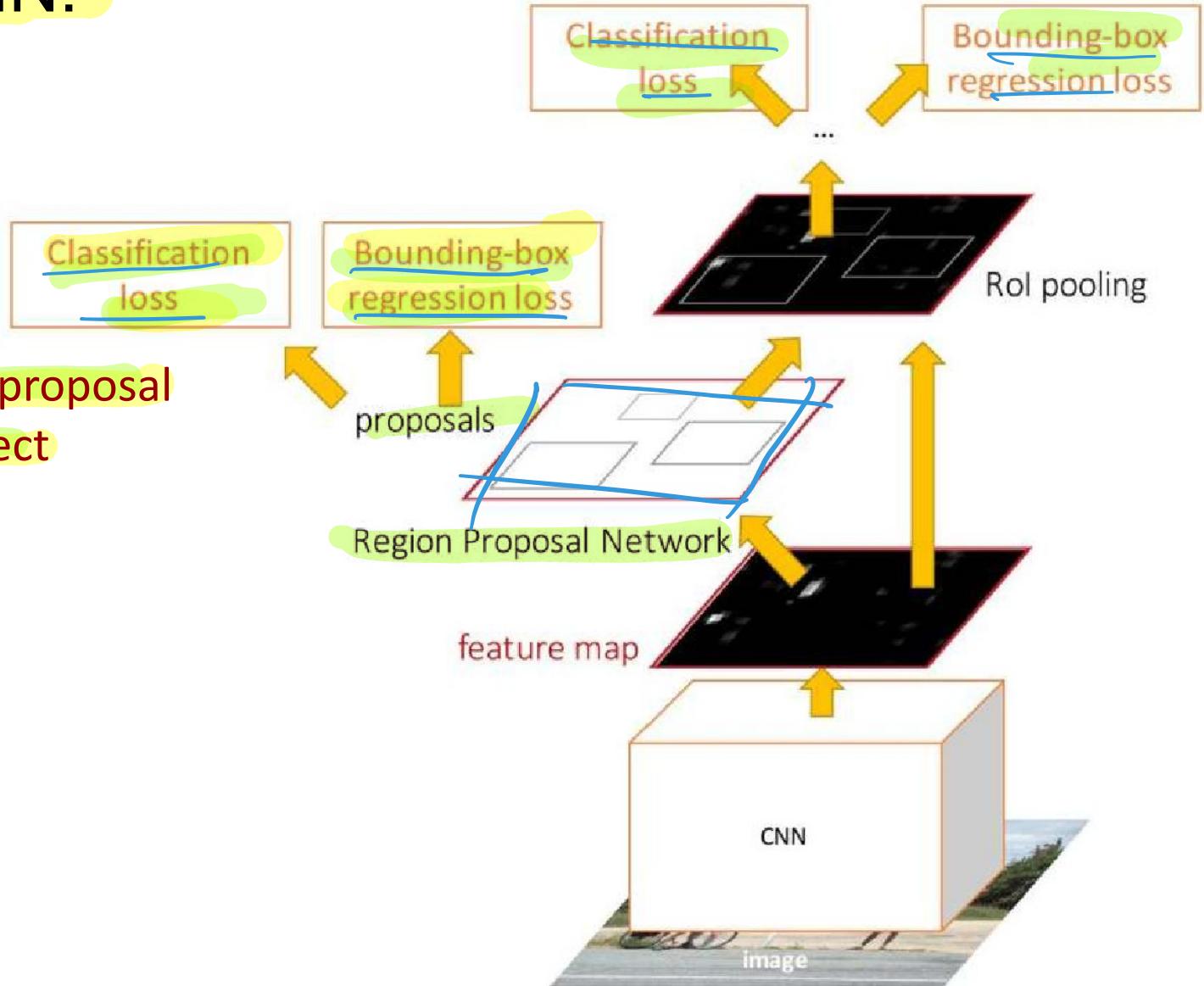


Girshick, "Fast R-CNN", ICCV 2015.

Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

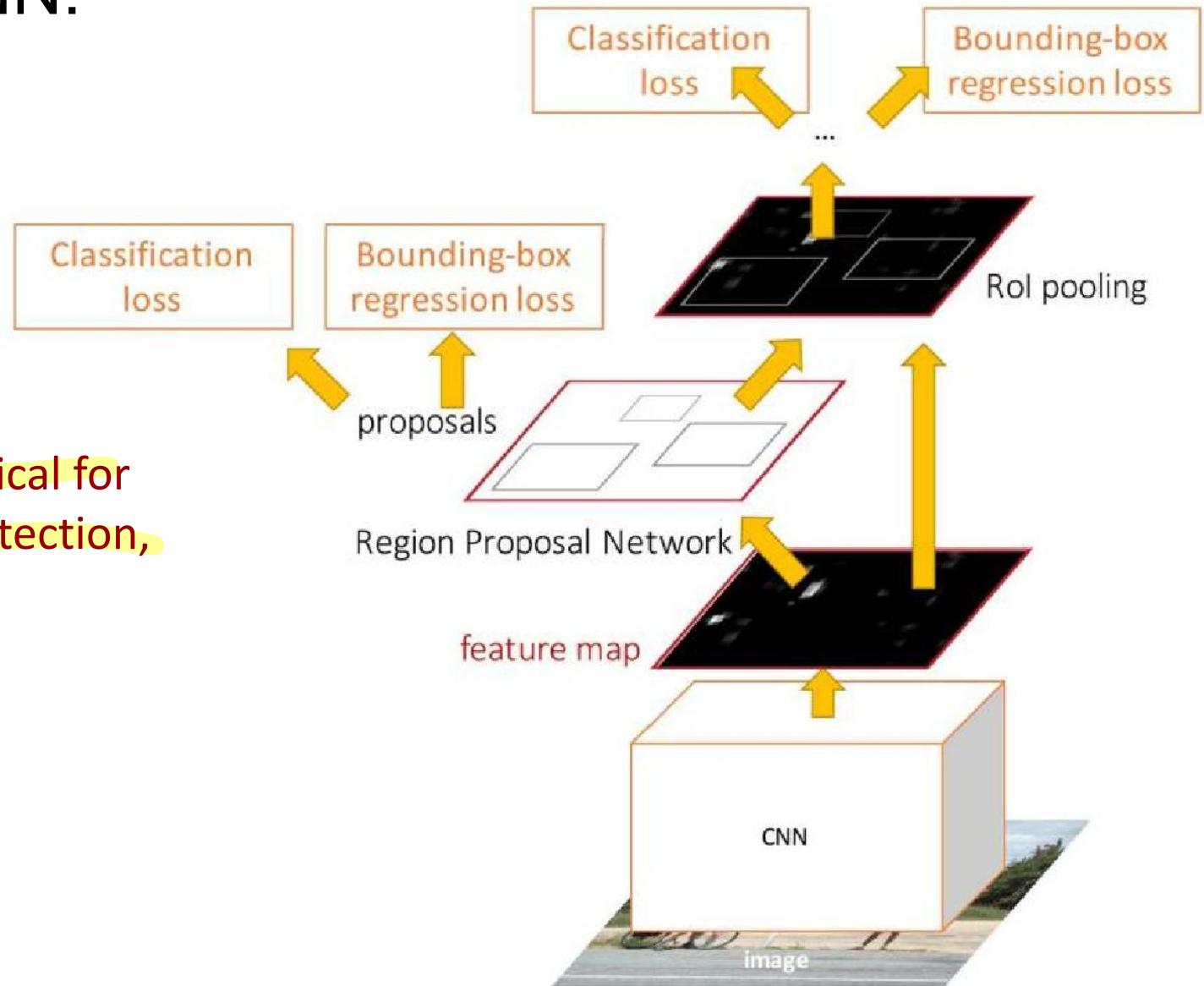
Faster R-CNN:

Two stages: region proposal generation and object classification



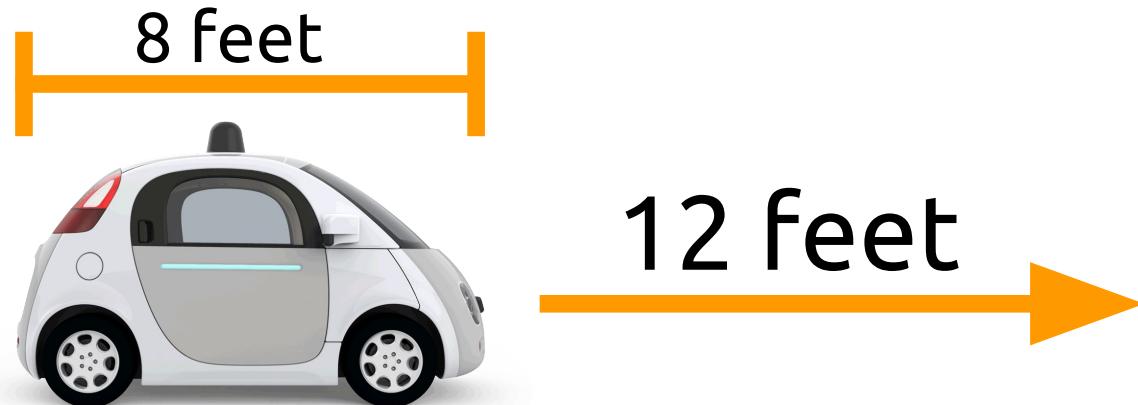
Faster R-CNN:

Problem: Not practical for real time Object Detection, Inference is slow



Accurate object detection is slow!

	Pascal 2007 mAP	Speed	
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



Accurate object detection is slow!

	Pascal 2007 mAP	Speed	
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

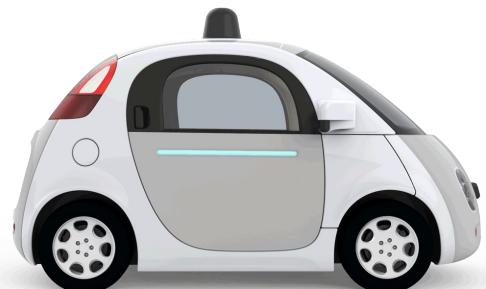


2 feet
→

A large green arrow points from the text "2 feet" towards the bottom right of the car icon.

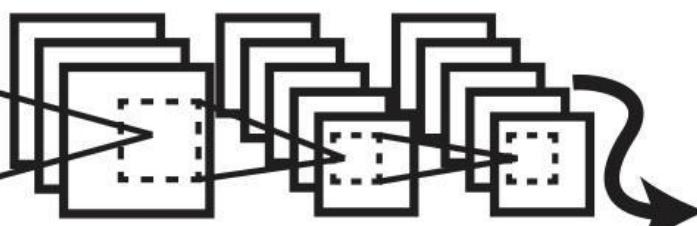
Accurate object detection is slow!

	Pascal 2007 mAP	Speed	
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 69.0	45 FPS	22 ms/img

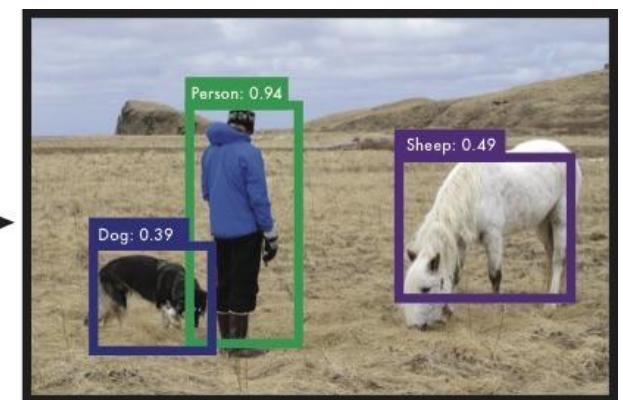


2 feet
→

A large green arrow points from the text "2 feet" towards the bottom right of the car icon.



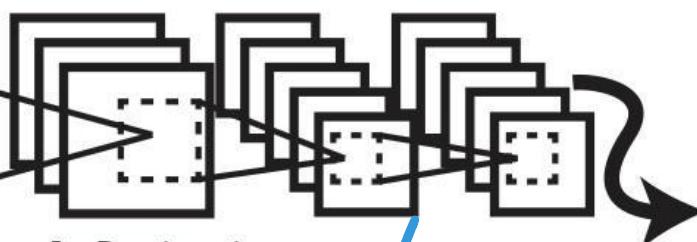
1. Resize image. ✓
2. Run convolutional network. ✓
3. Threshold detections.



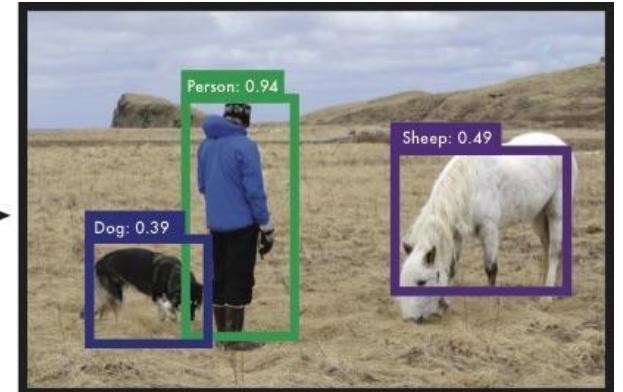
• Resize
• Convolve
• Threshold.

With YOLO, you only look once at an image to perform detection

YOLO: *You Only Look Once*



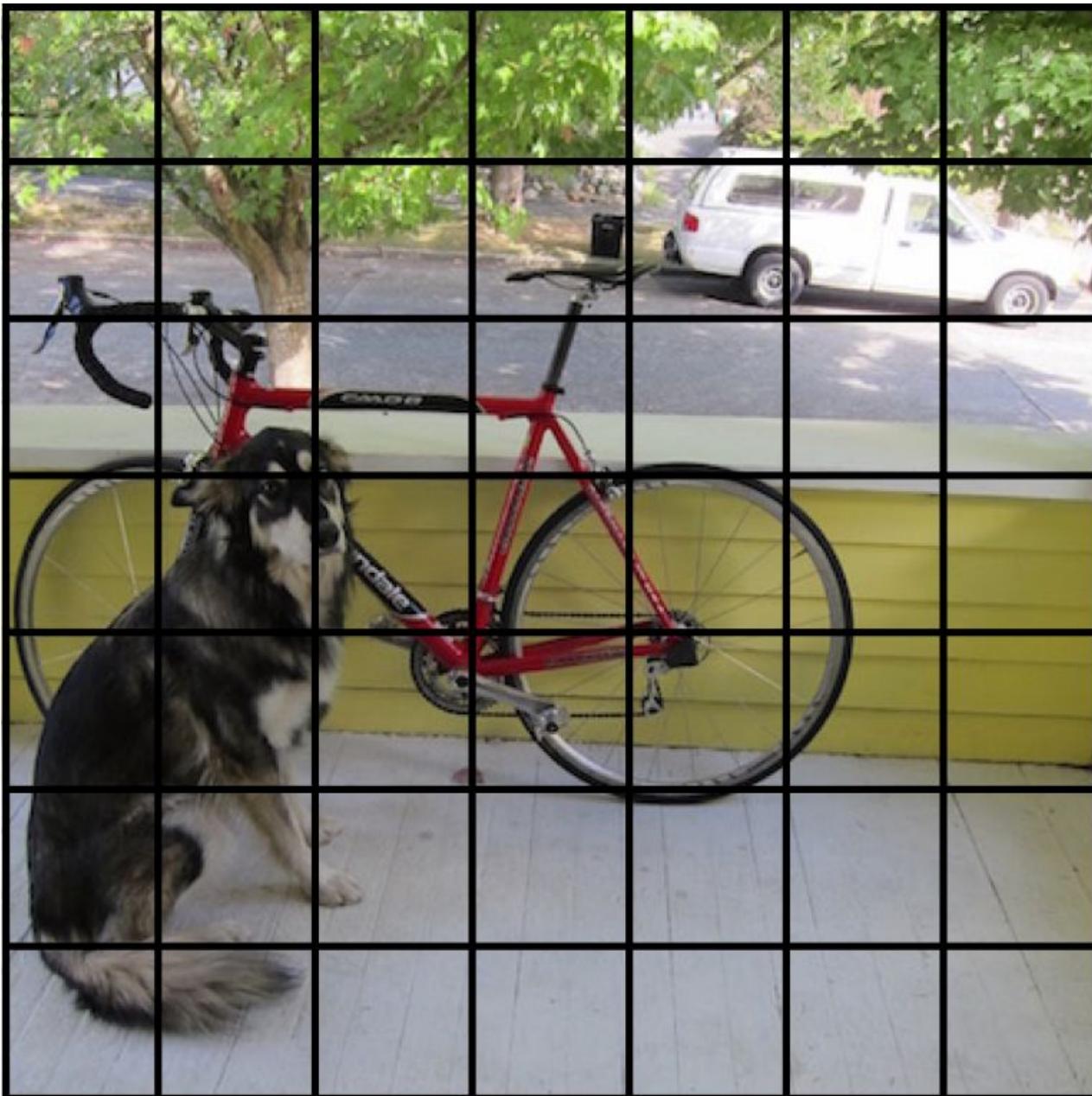
1. Resize image.
2. Run convolutional network.
3. Threshold detections.



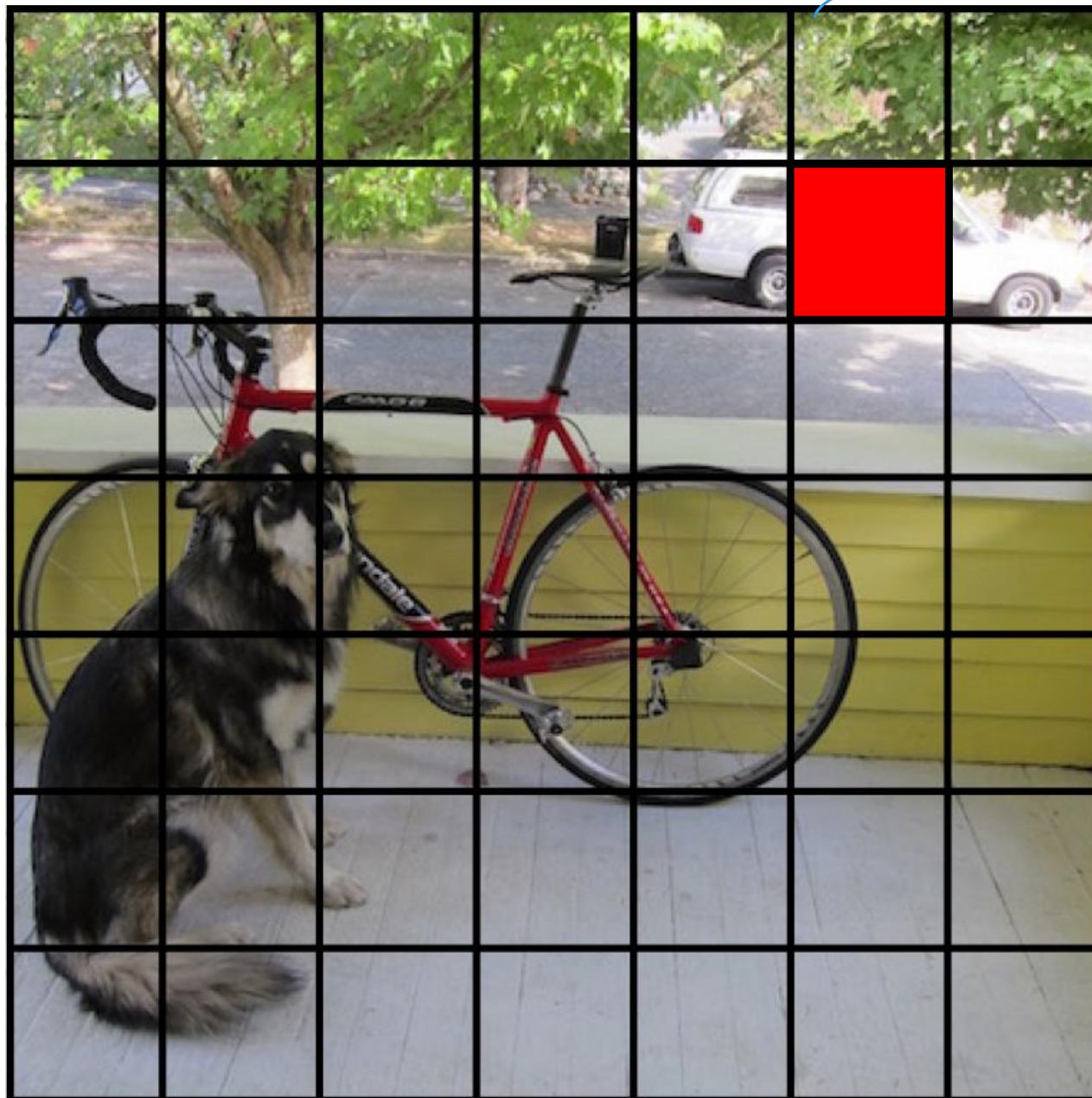


Box of Confidence.

We split the image into a grid



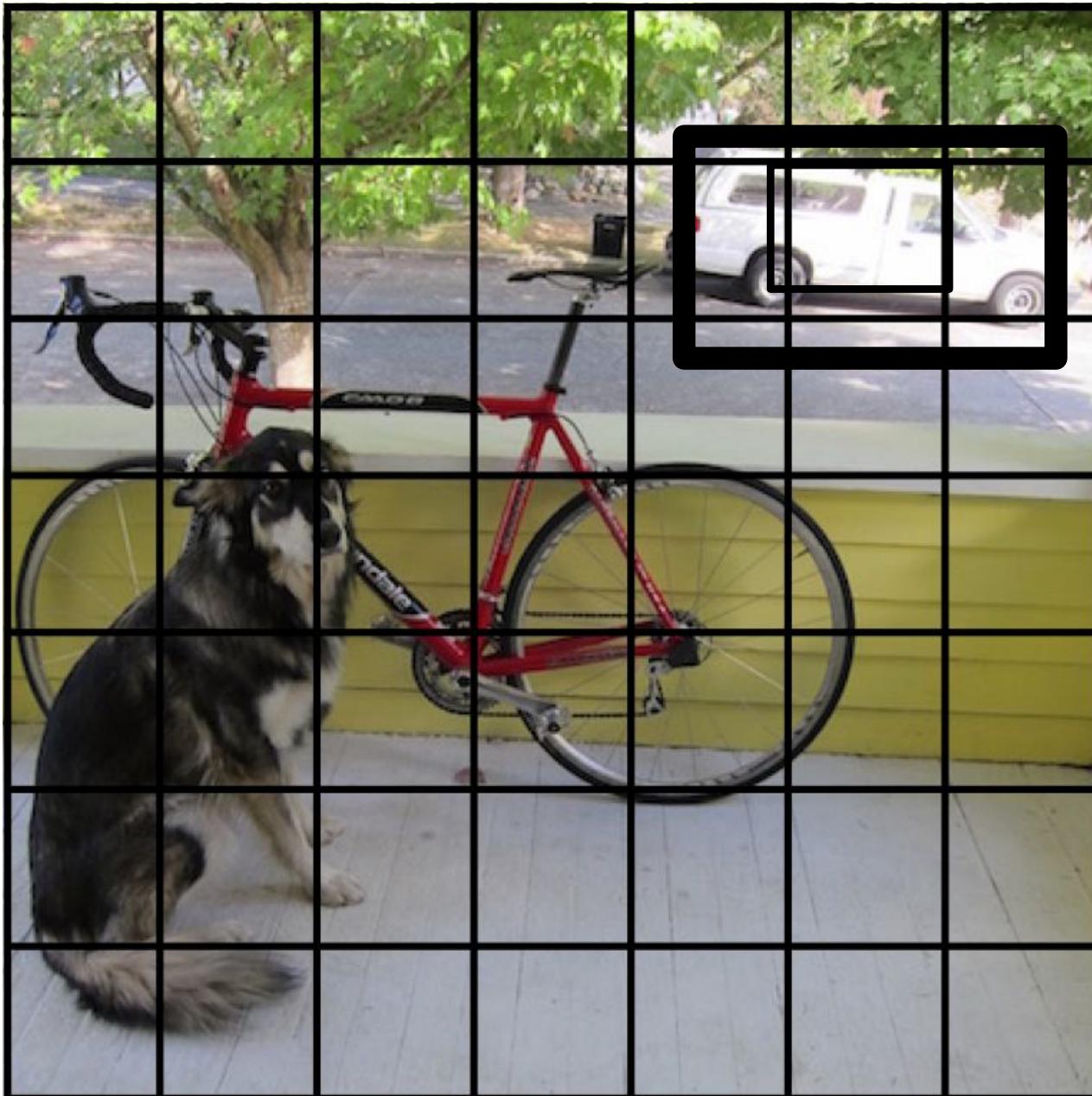
Each cell predicts boxes and confidences: $P(\text{Object})$



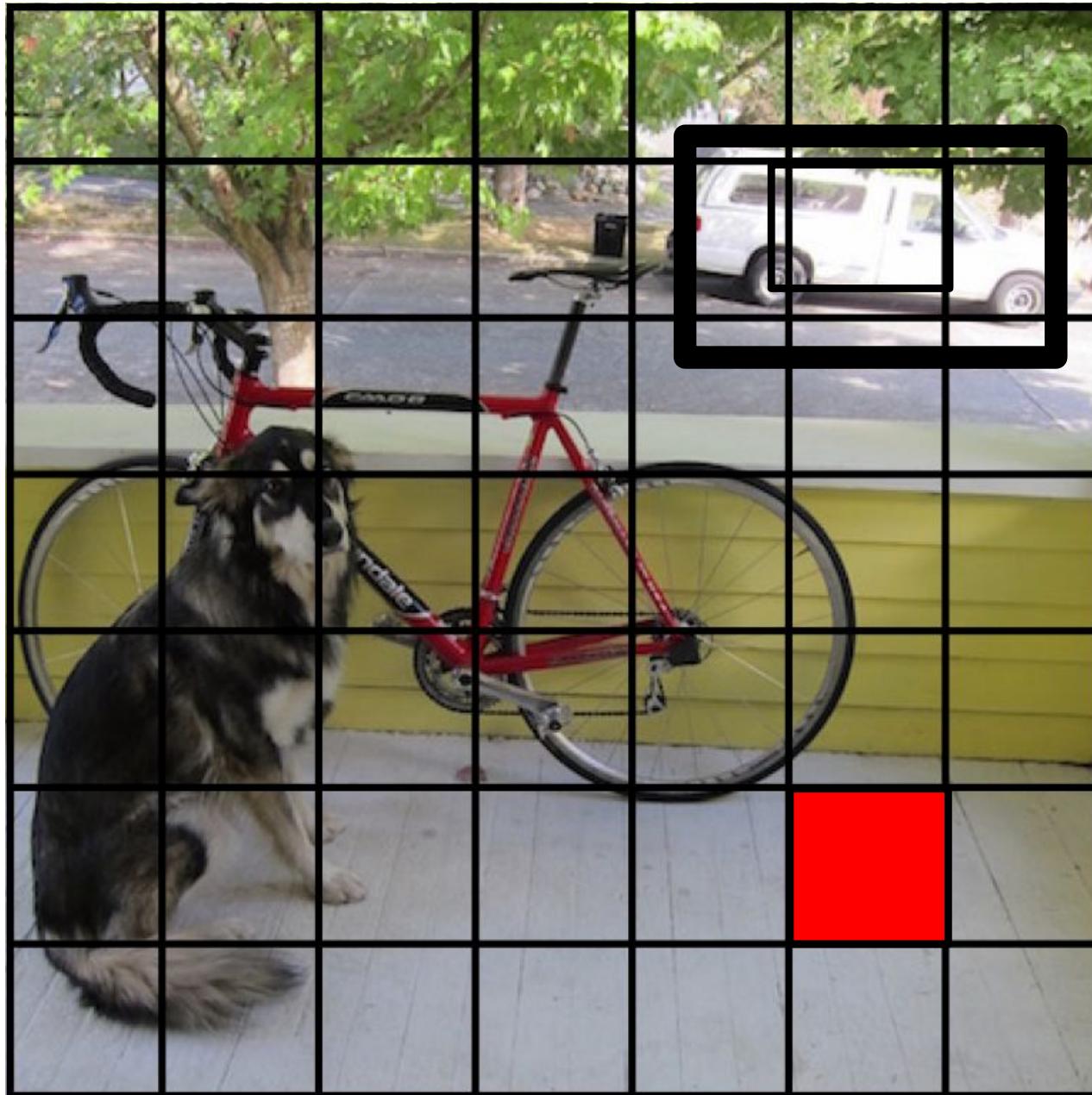
Each cell predicts boxes and confidences: $P(\text{Object})$



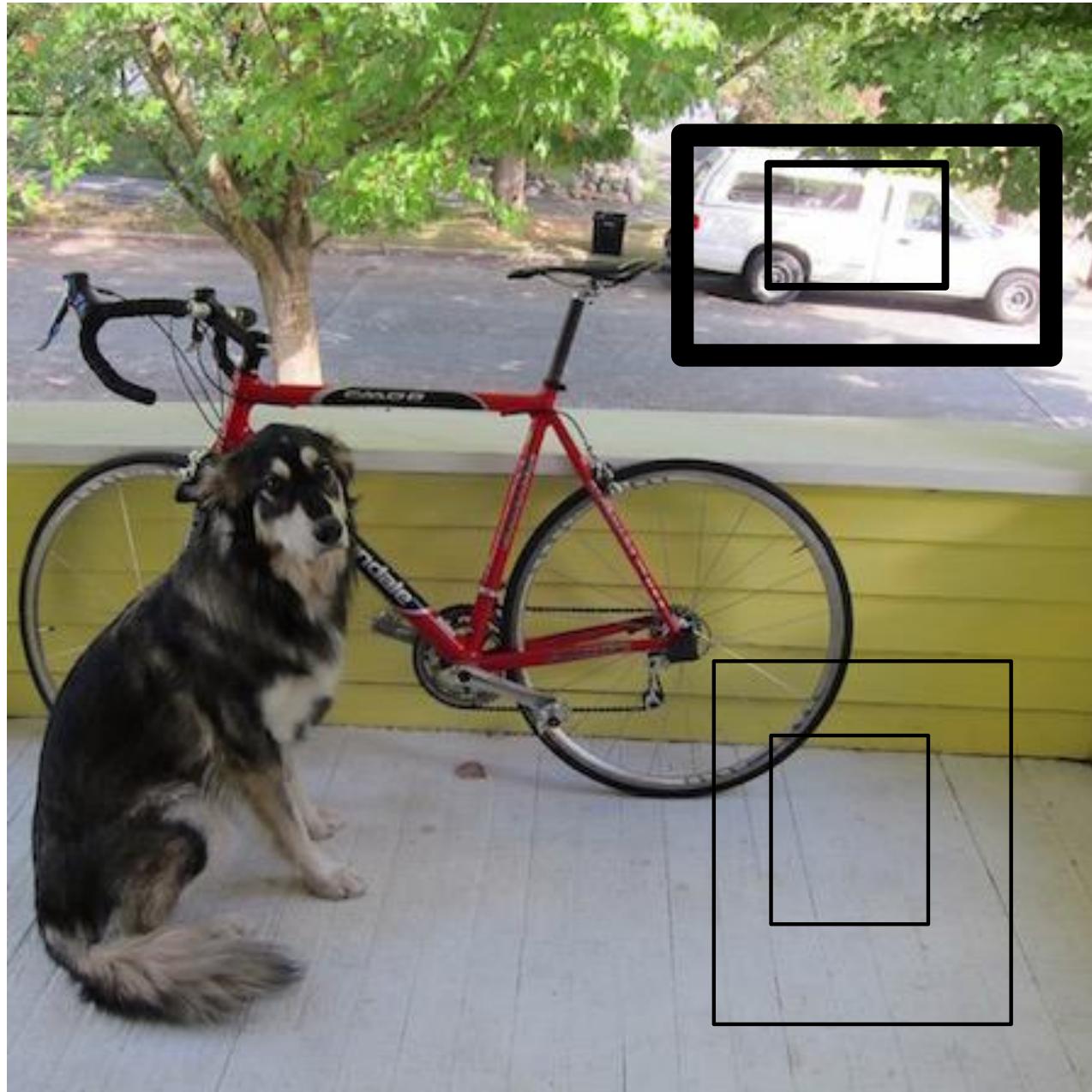
Each cell predicts boxes and confidences: $P(\text{Object})$



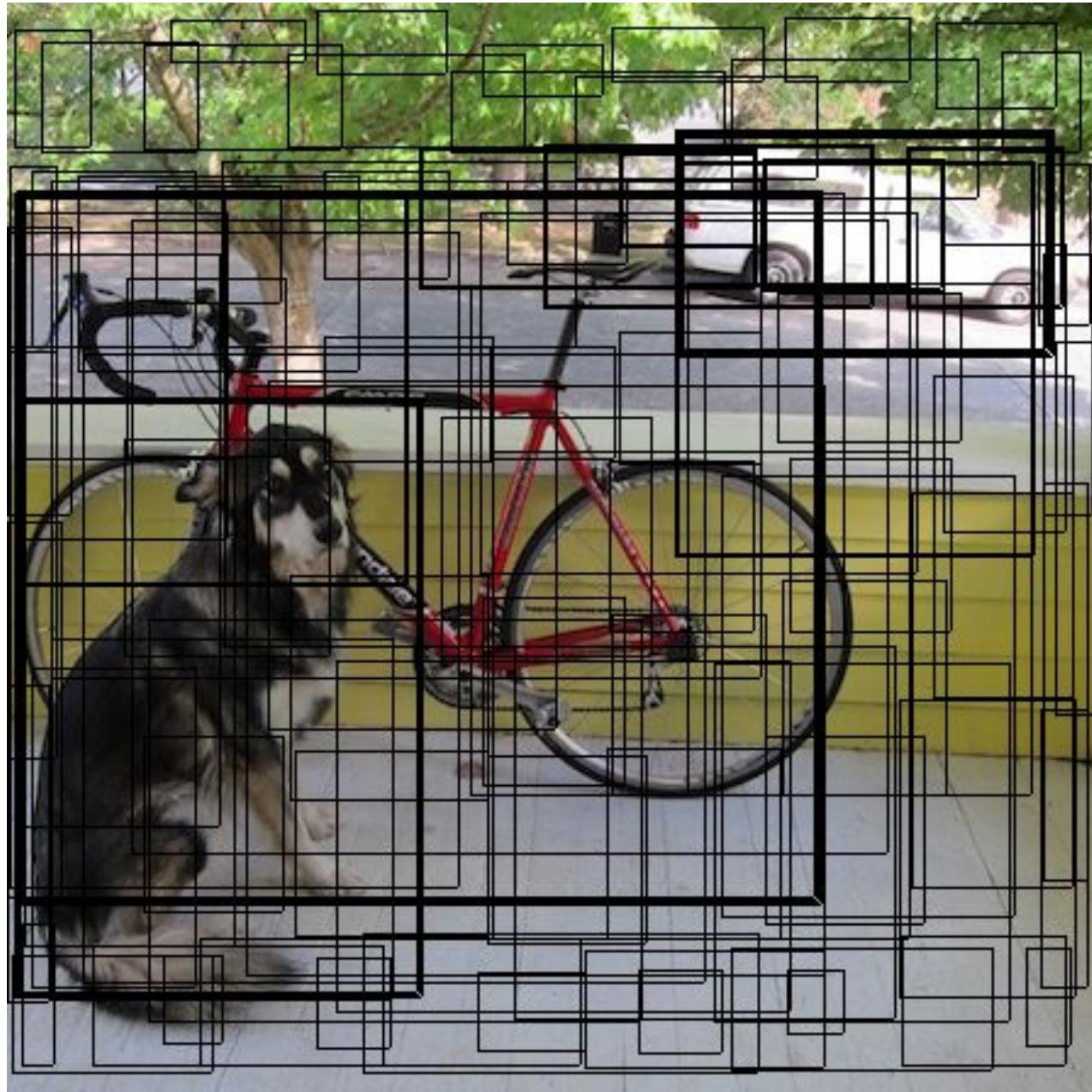
Each cell predicts boxes and confidences: $P(\text{Object})$



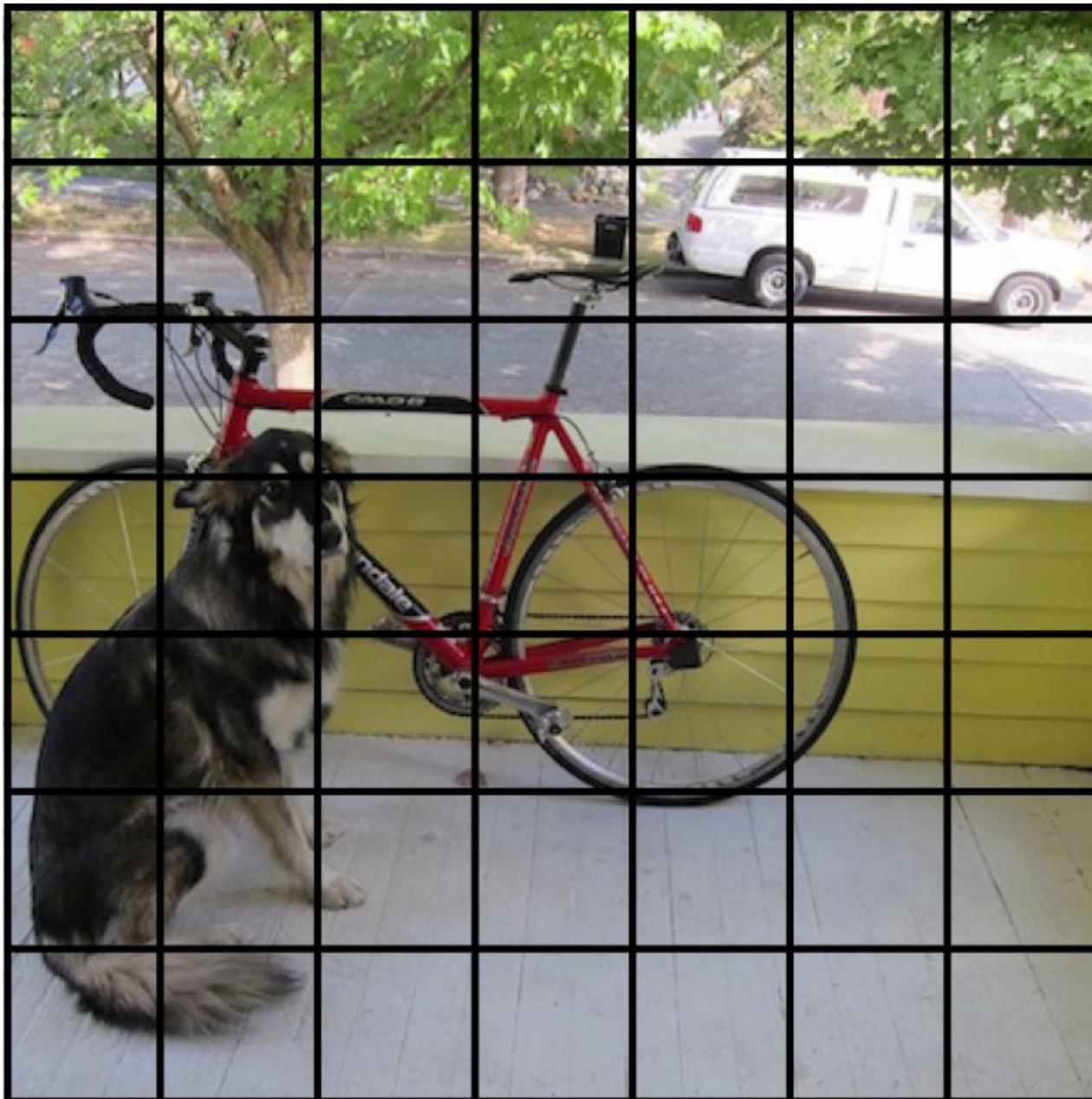
Each cell predicts boxes and confidences: $P(\text{Object})$



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



Each cell also predicts a class probability.



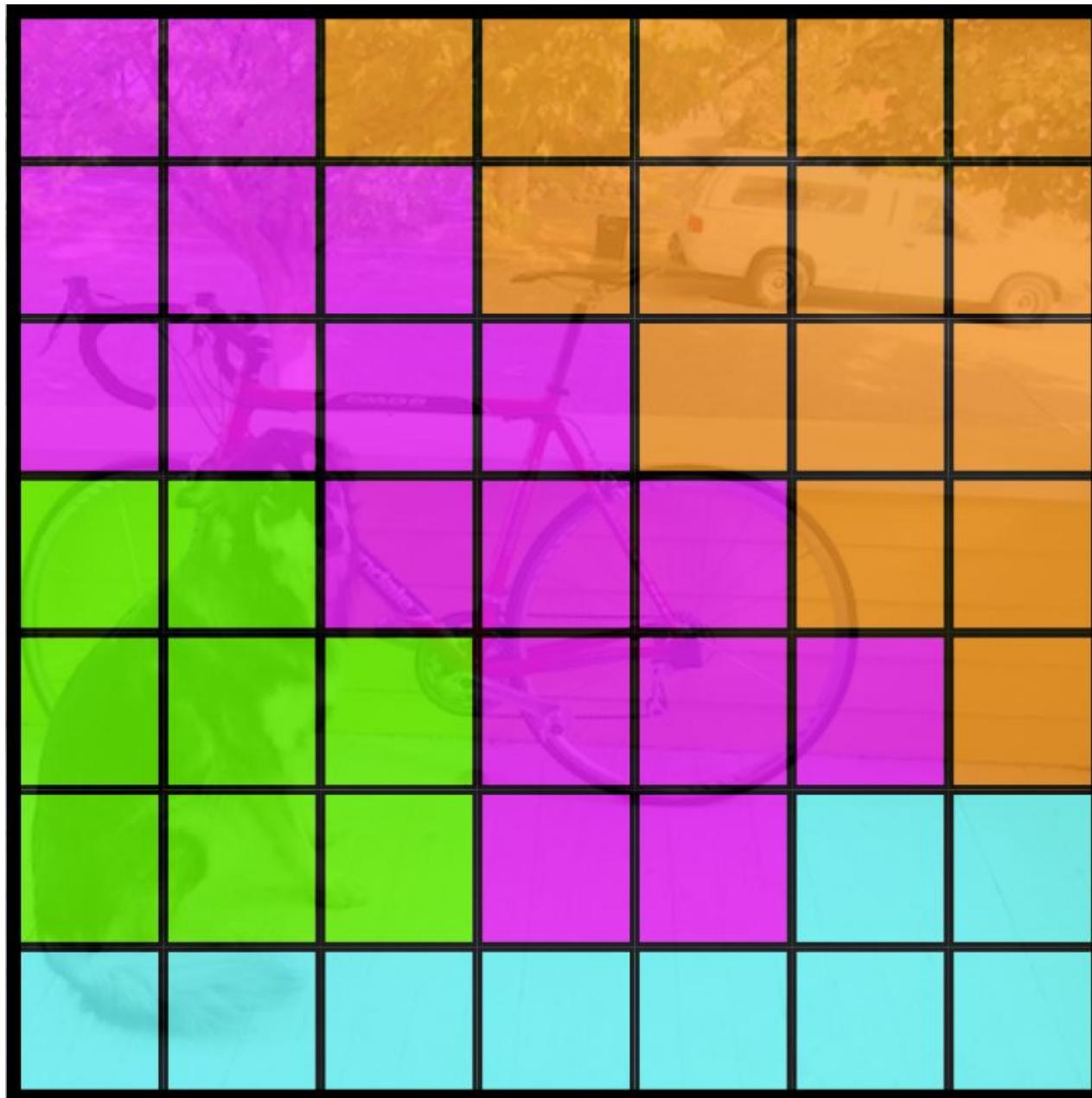
Each cell also predicts a class probability.

Bicycle

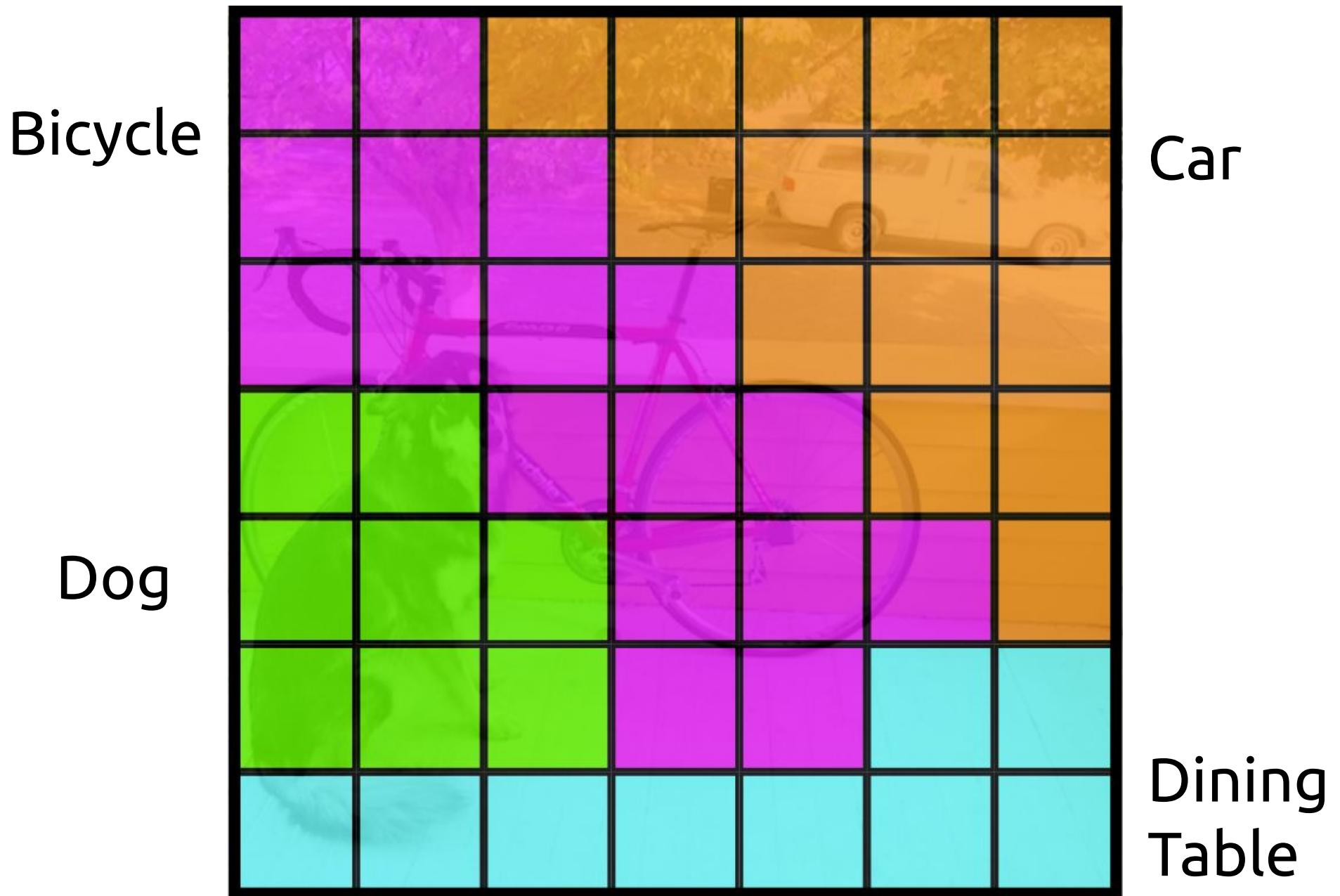
Car

Dog

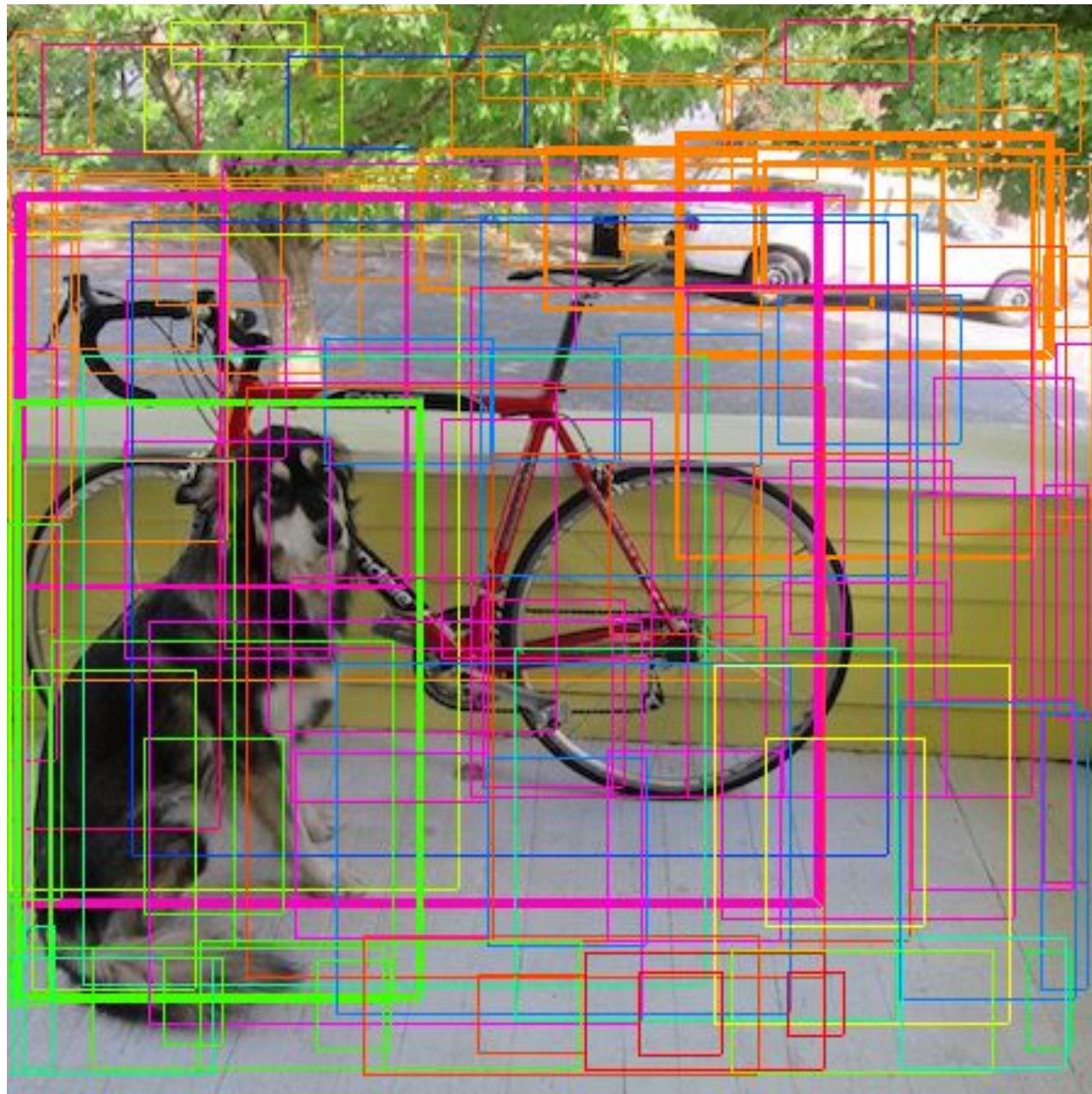
Dining
Table



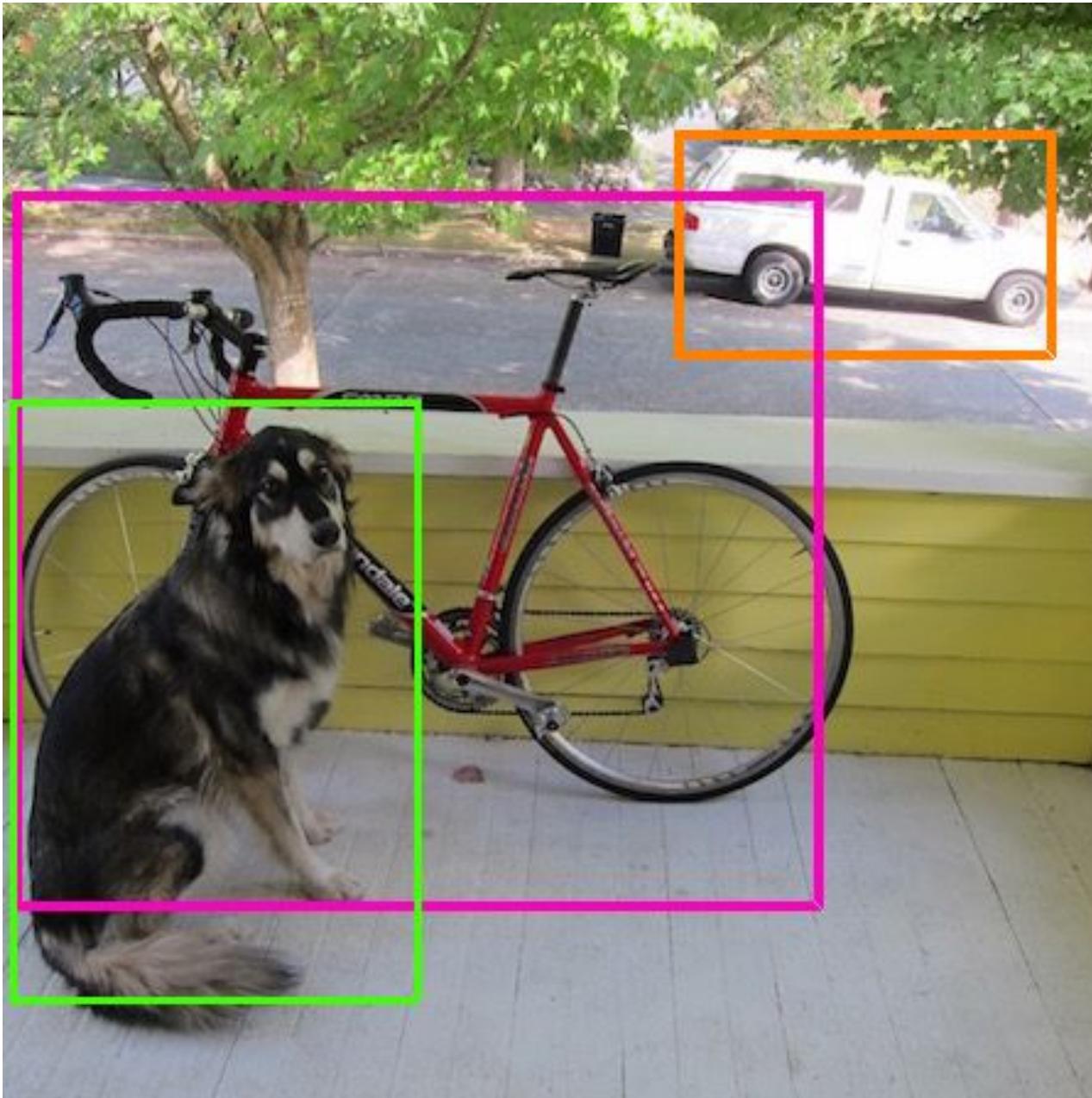
Conditioned on object: $P(\text{Car} \mid \text{Object})$



Then we combine the box and class predictions.



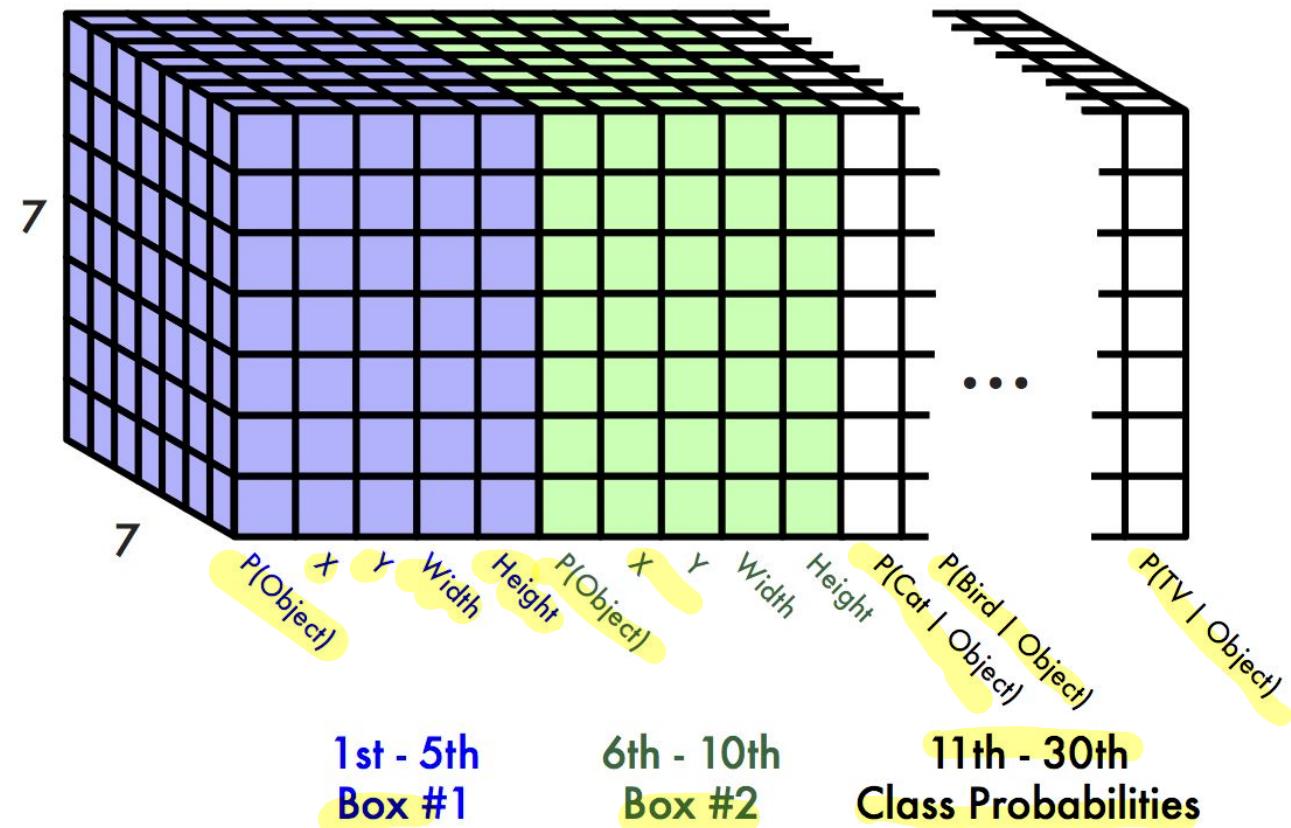
Finally, set a threshold (Non-maximal suppression) to fix multiple detections



This parameterization fixes the output size

Each cell predicts:

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



This parameterization fixes the output size

Each cell predicts:

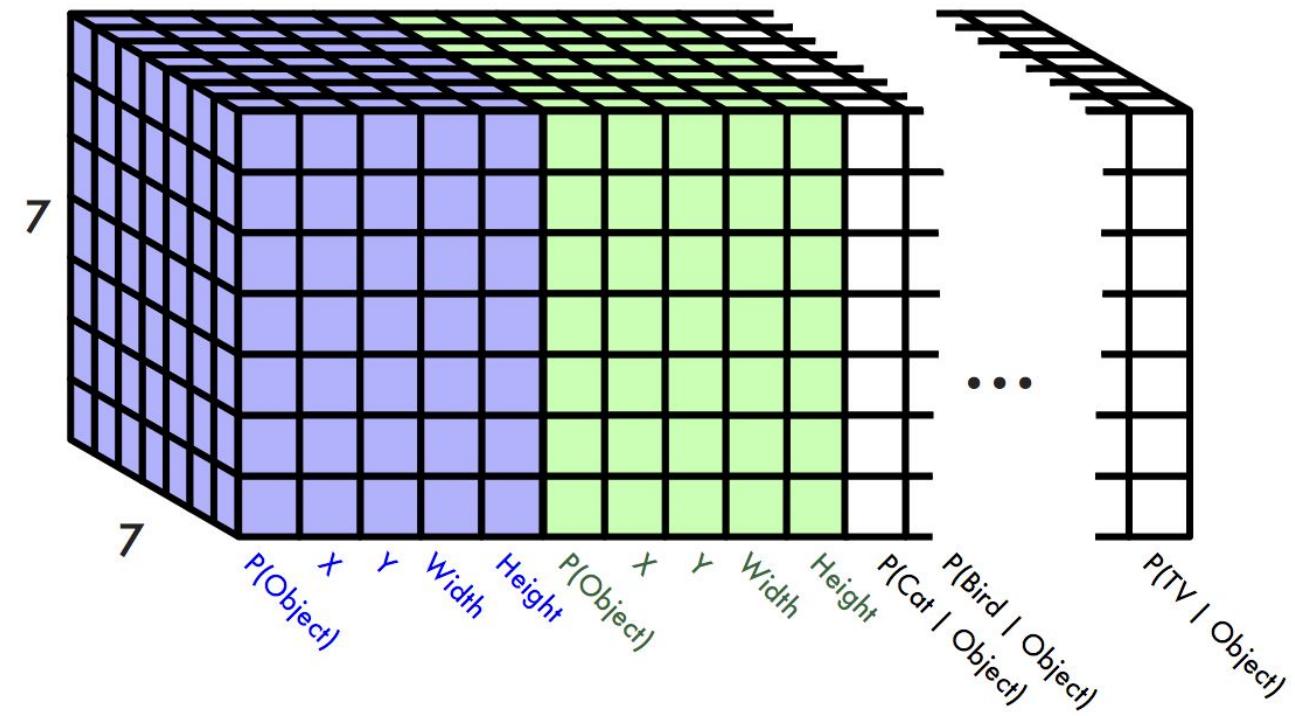
- For each bounding box:
 - 4 coordinates (x, y, w, h)
 - 1 confidence value
- Some number of class probabilities

7x7 grid

Total Predictions

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

$$7 \times 7 \times (2 \times 5 + 20) = 7 \times 7 \times 30 \text{ tensor} = 1470 \text{ outputs}$$



1st - 5th
Box #1

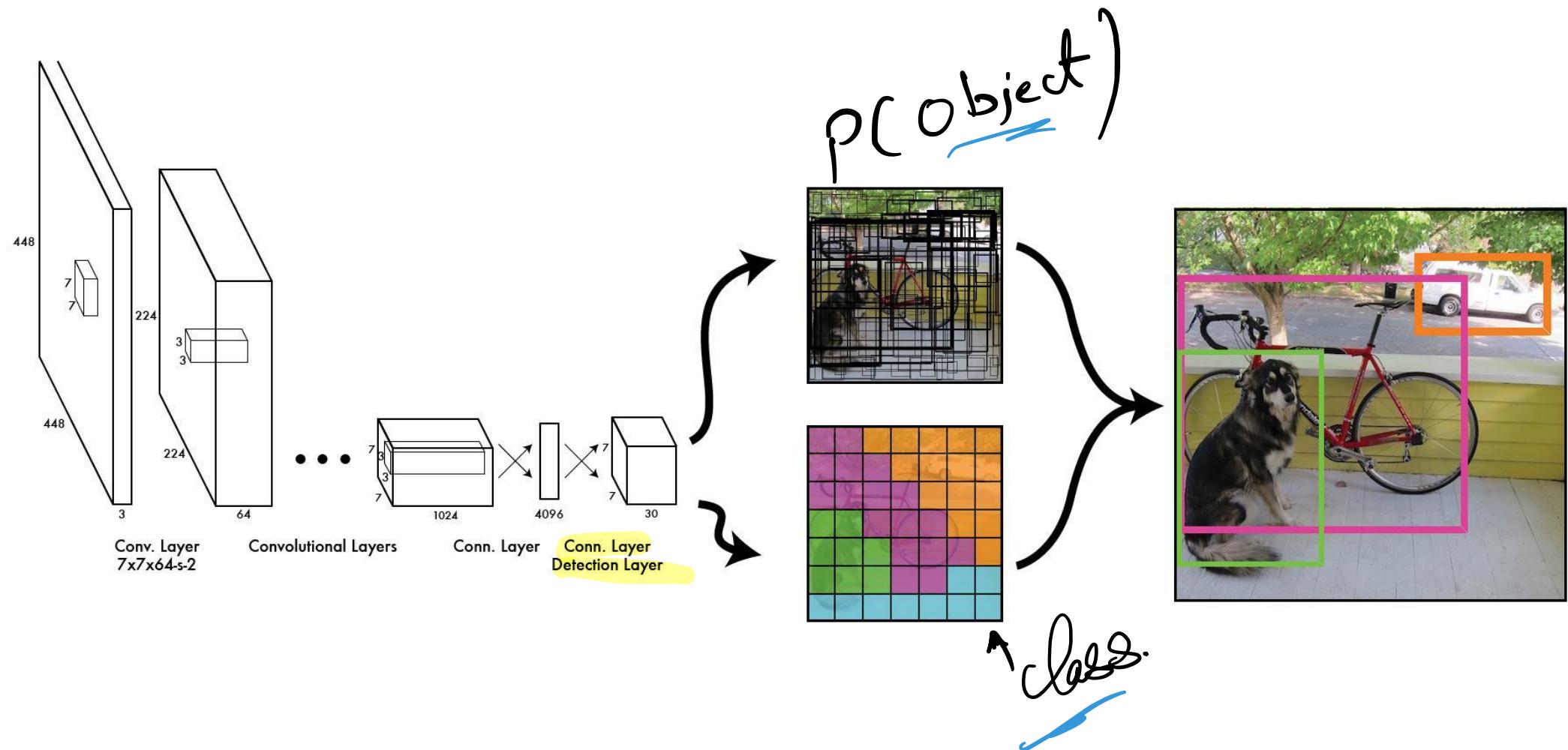
6th - 10th
Box #2

11th - 30th
Class Probabilities

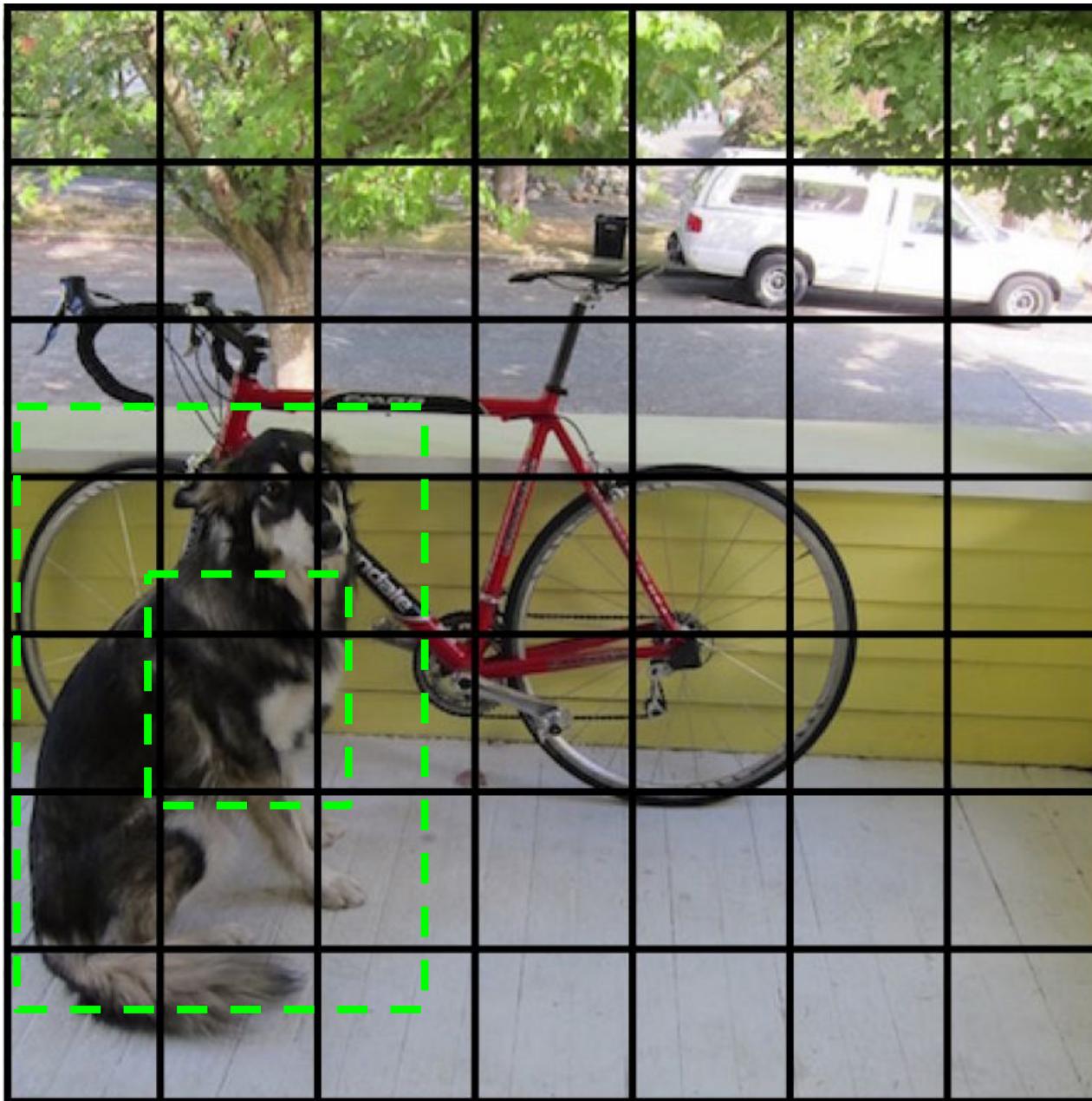
① Grid
② Predict boxes
and confidence
(or conditional
prob.)

Thus we can train one neural network to be a whole detection pipeline

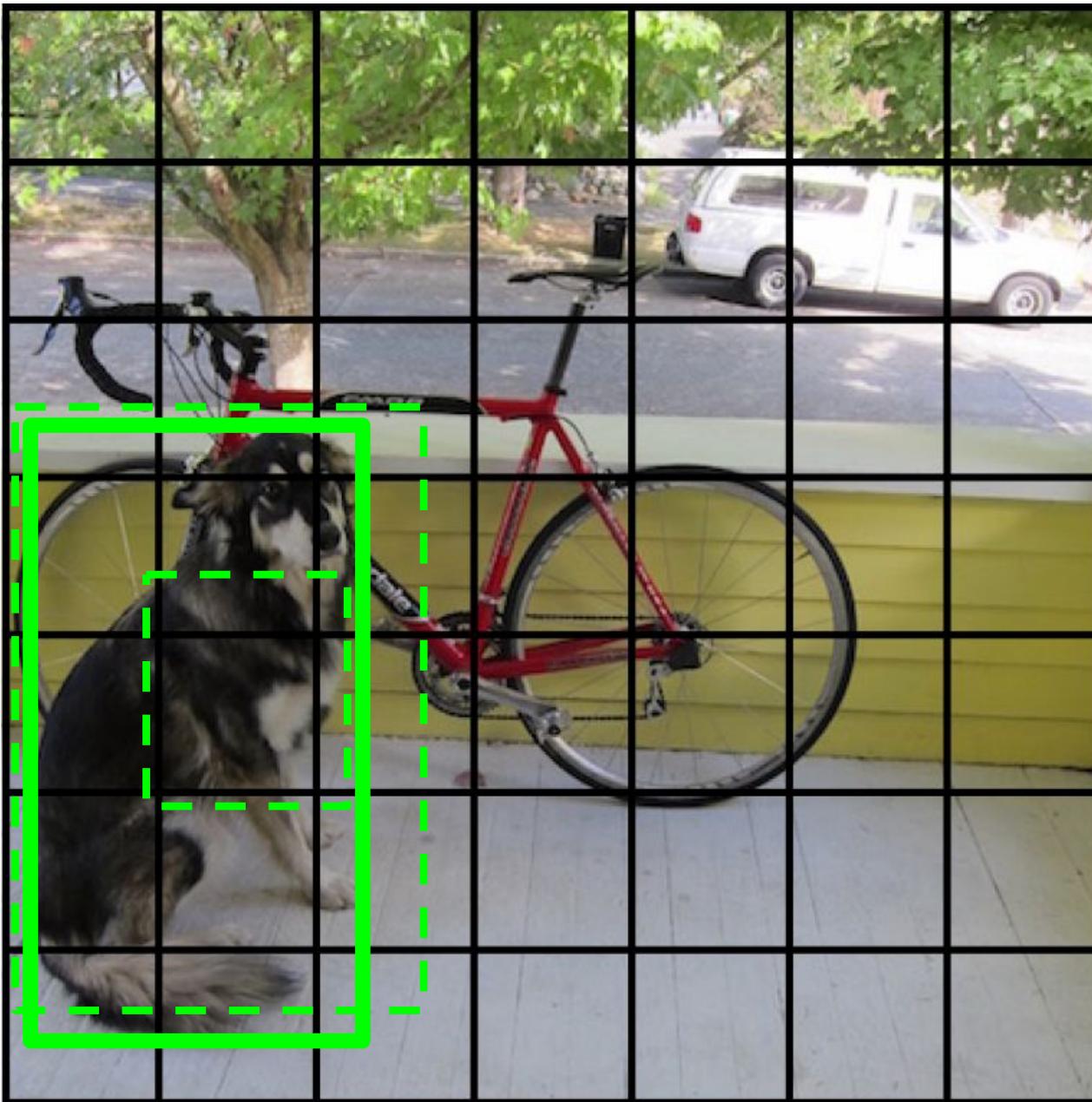
③ Class $\rightarrow p_{\text{obj}}$



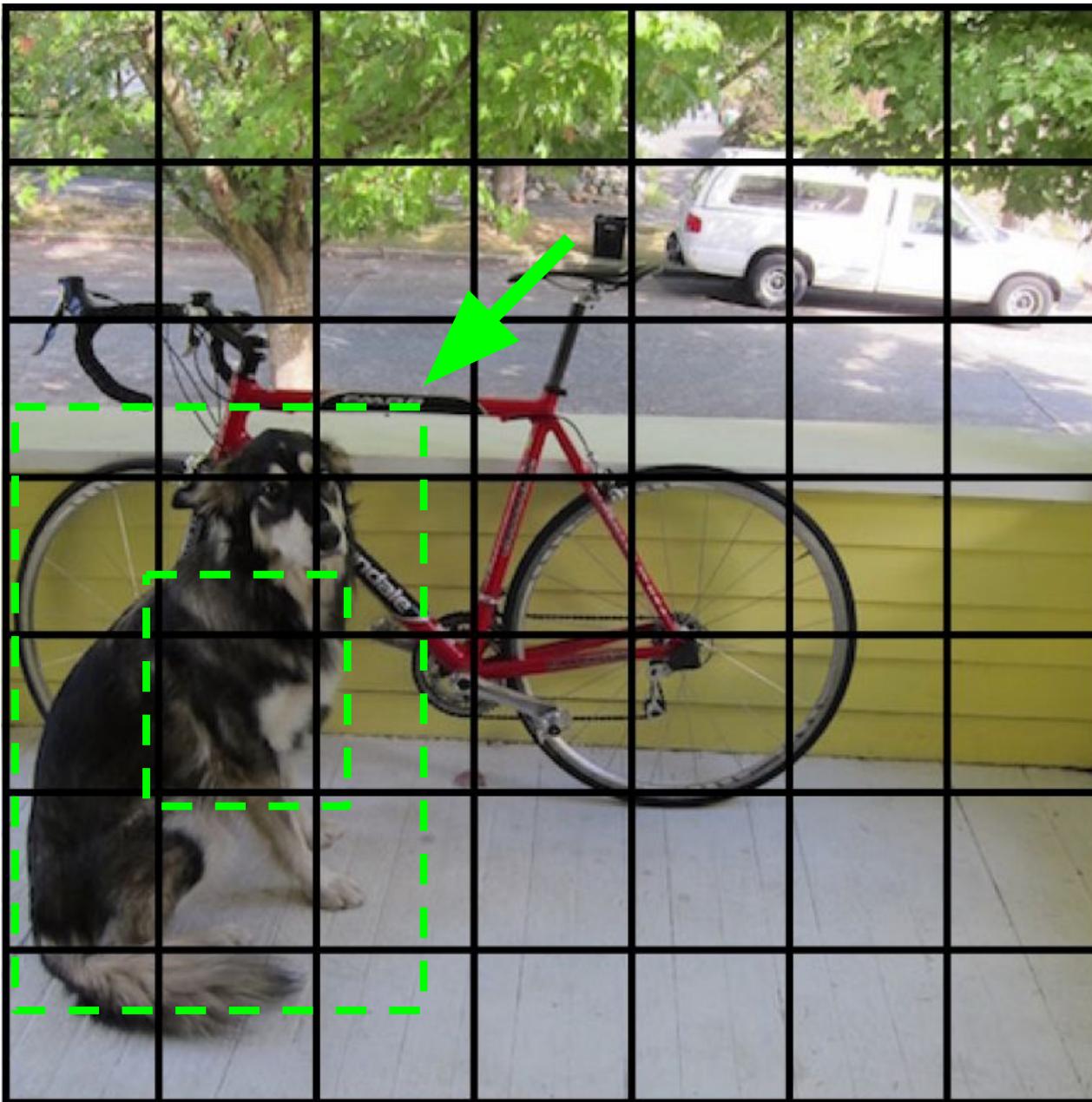
Look at that cell's predicted boxes



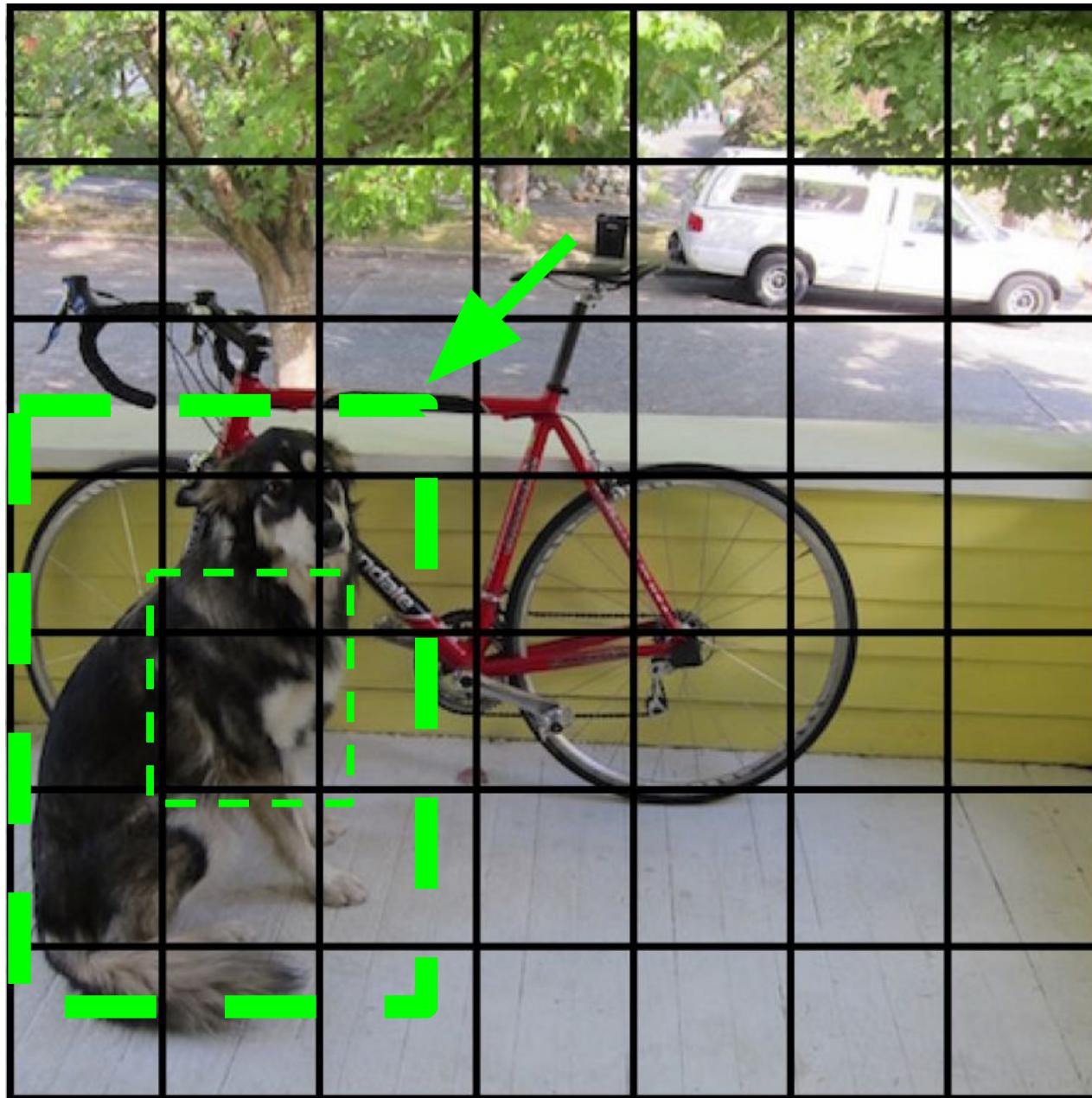
Find the best one, adjust it, increase the confidence



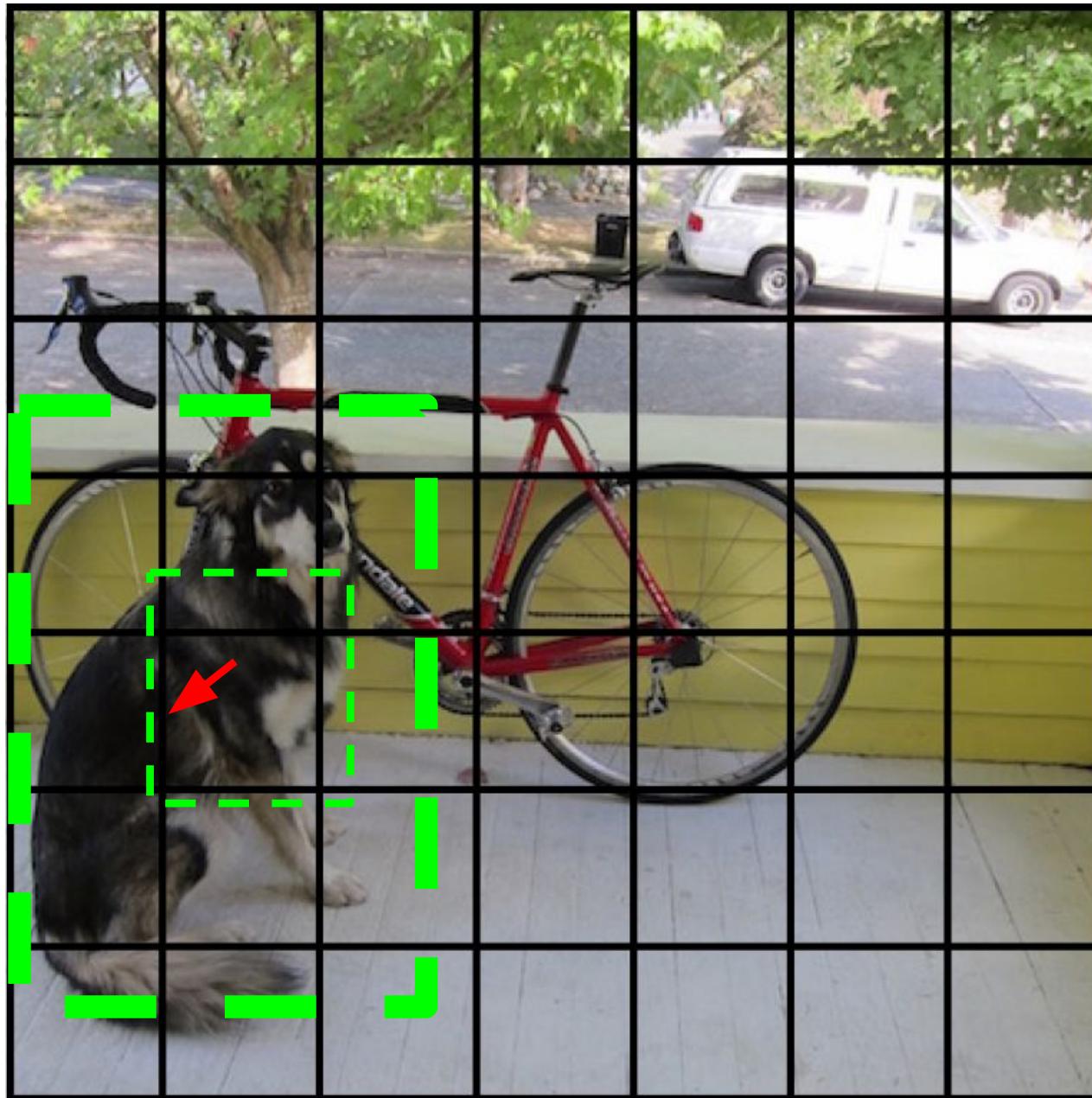
Find the best one, adjust it, increase the confidence



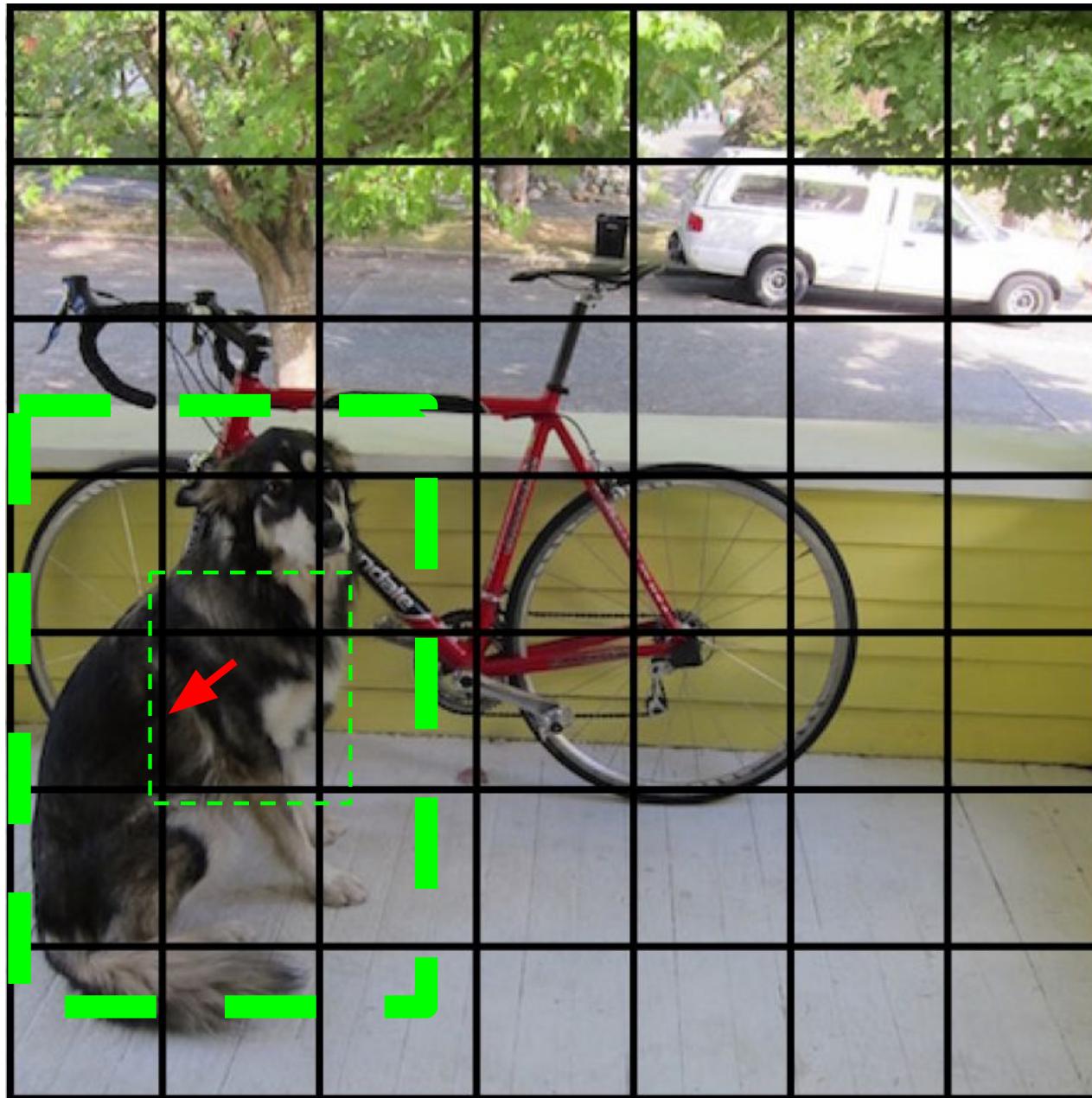
Find the best one, adjust it, increase the confidence



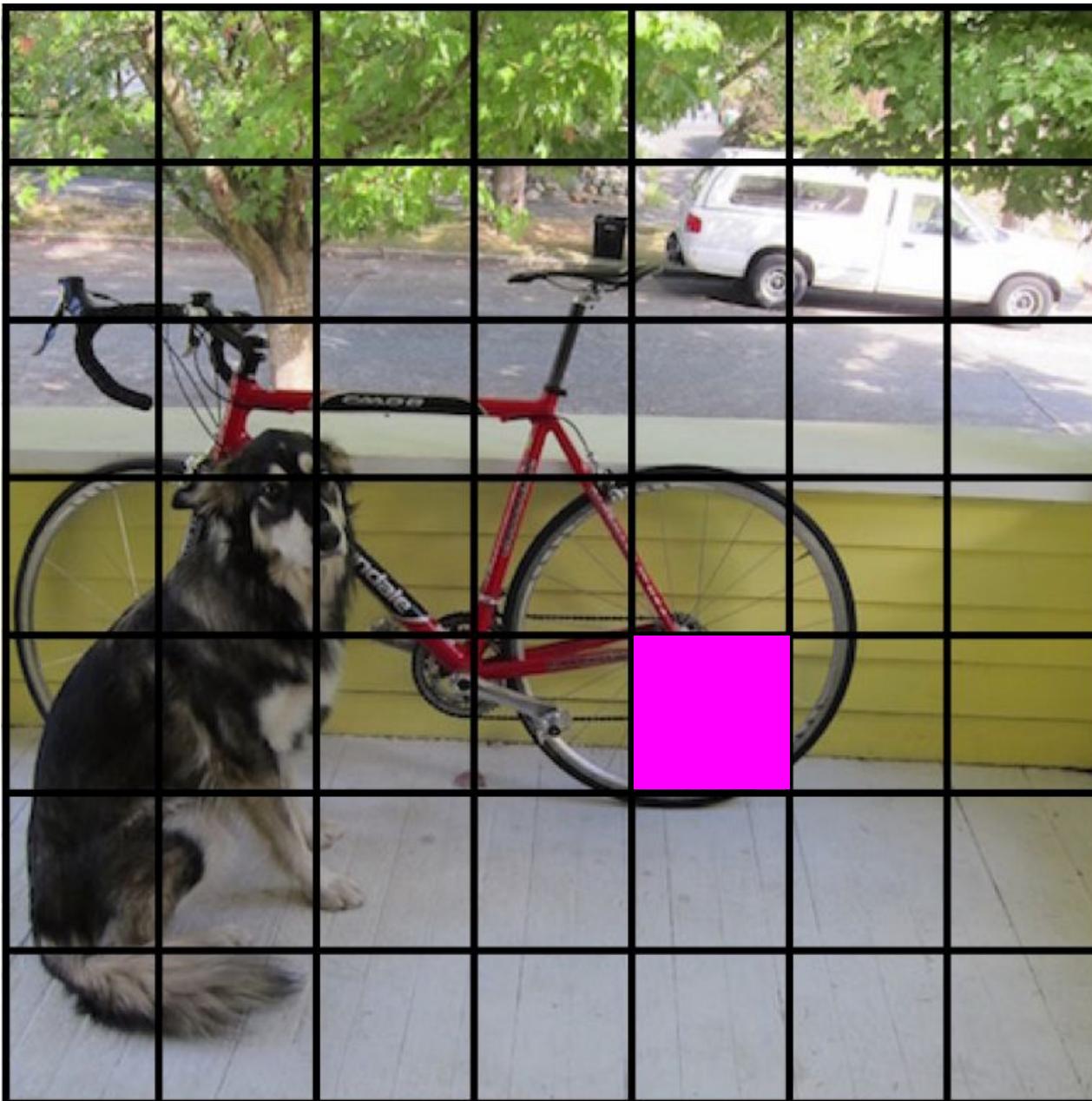
Decrease the confidence of other boxes



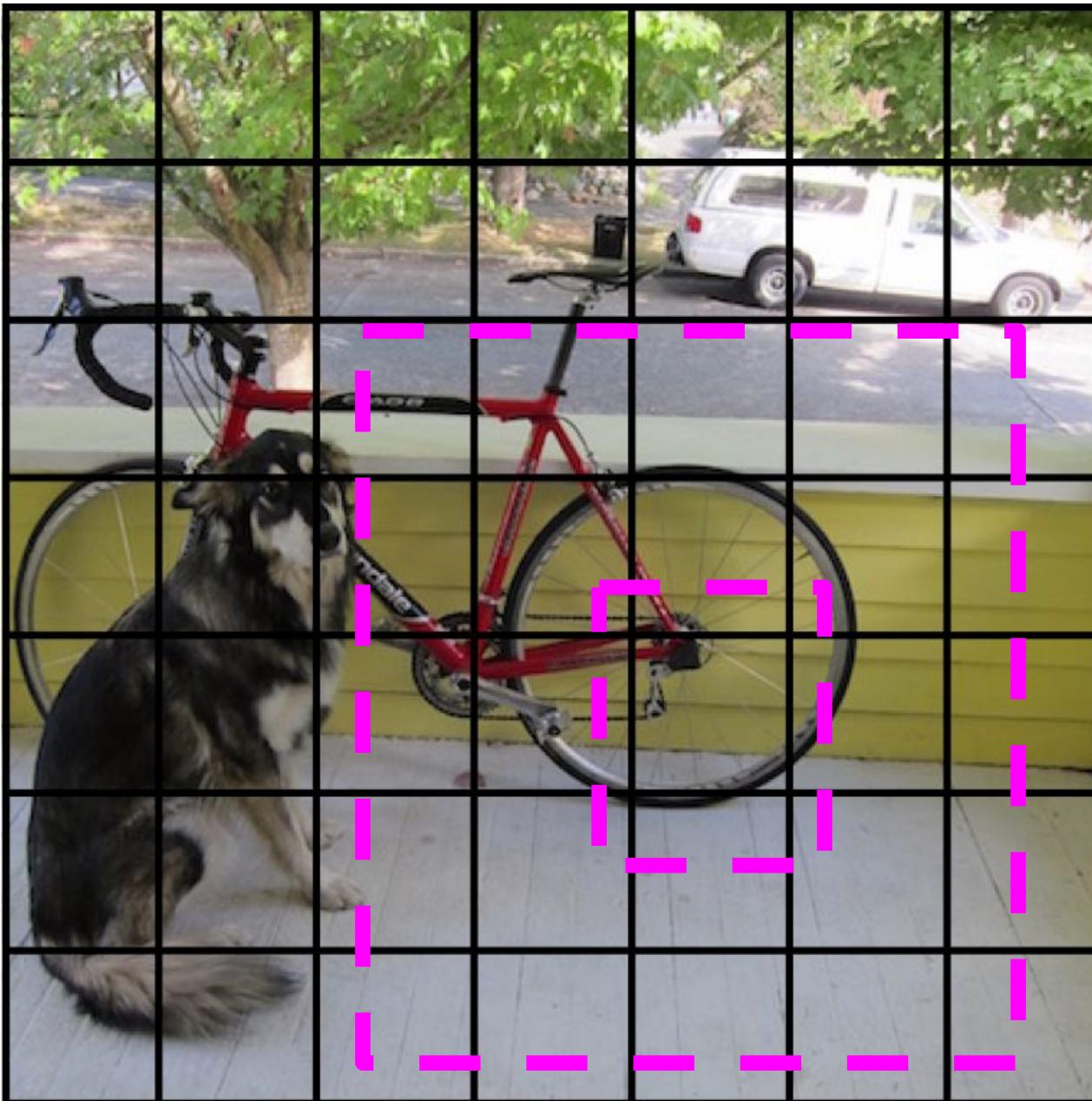
Decrease the confidence of other boxes



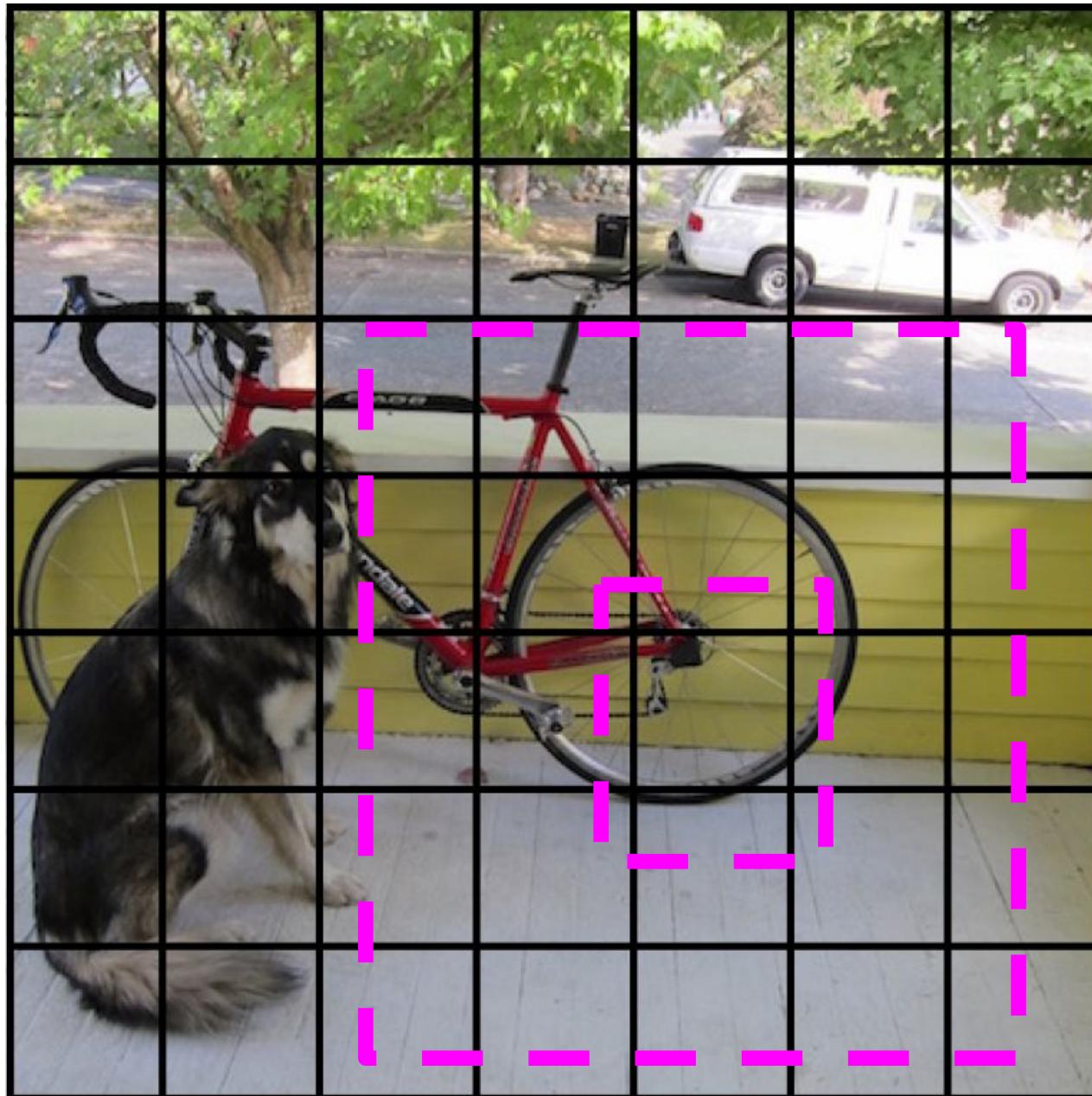
Some cells don't have any ground truth detections!



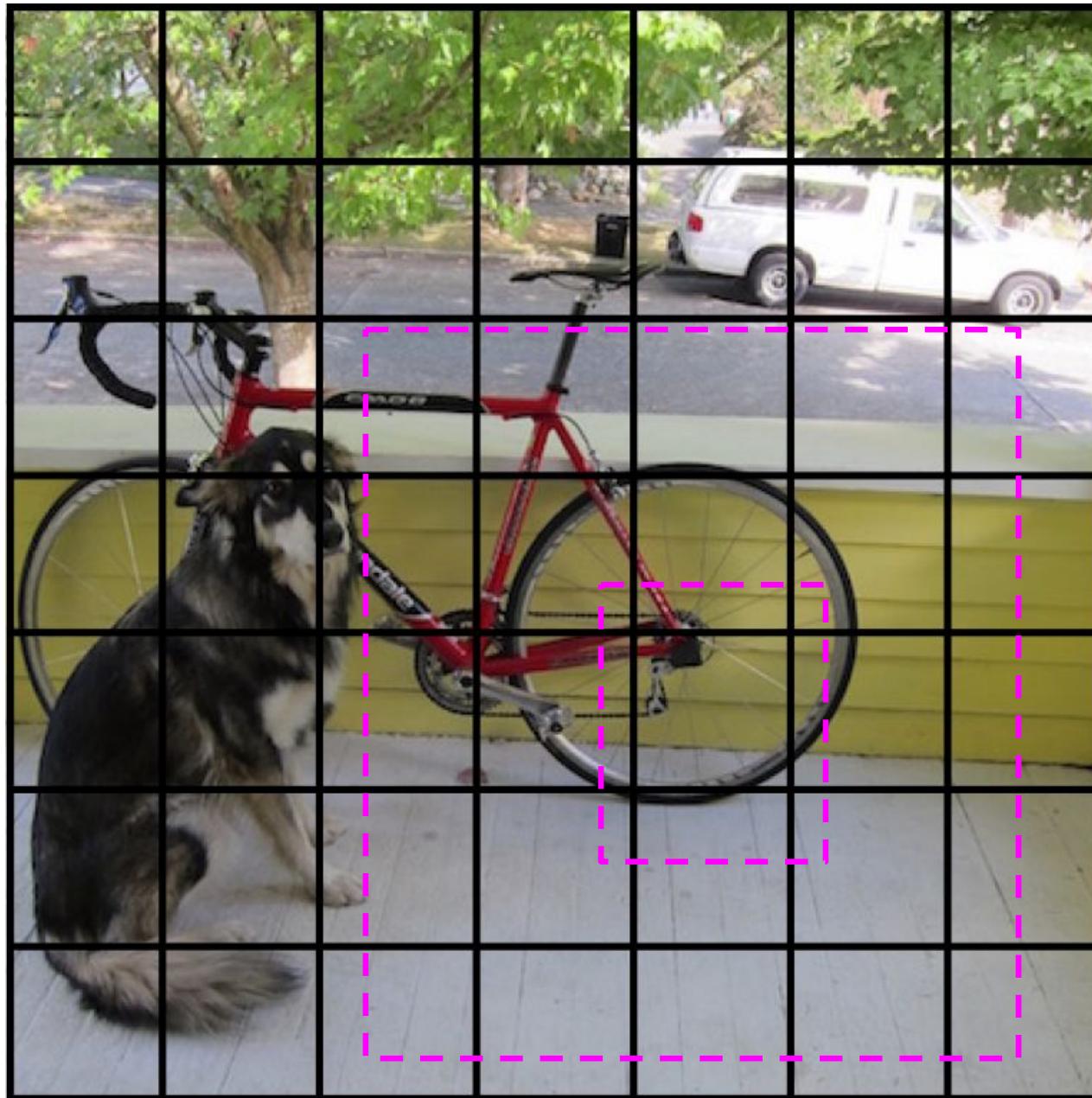
Some cells don't have any ground truth detections!



Decrease the confidence of these boxes

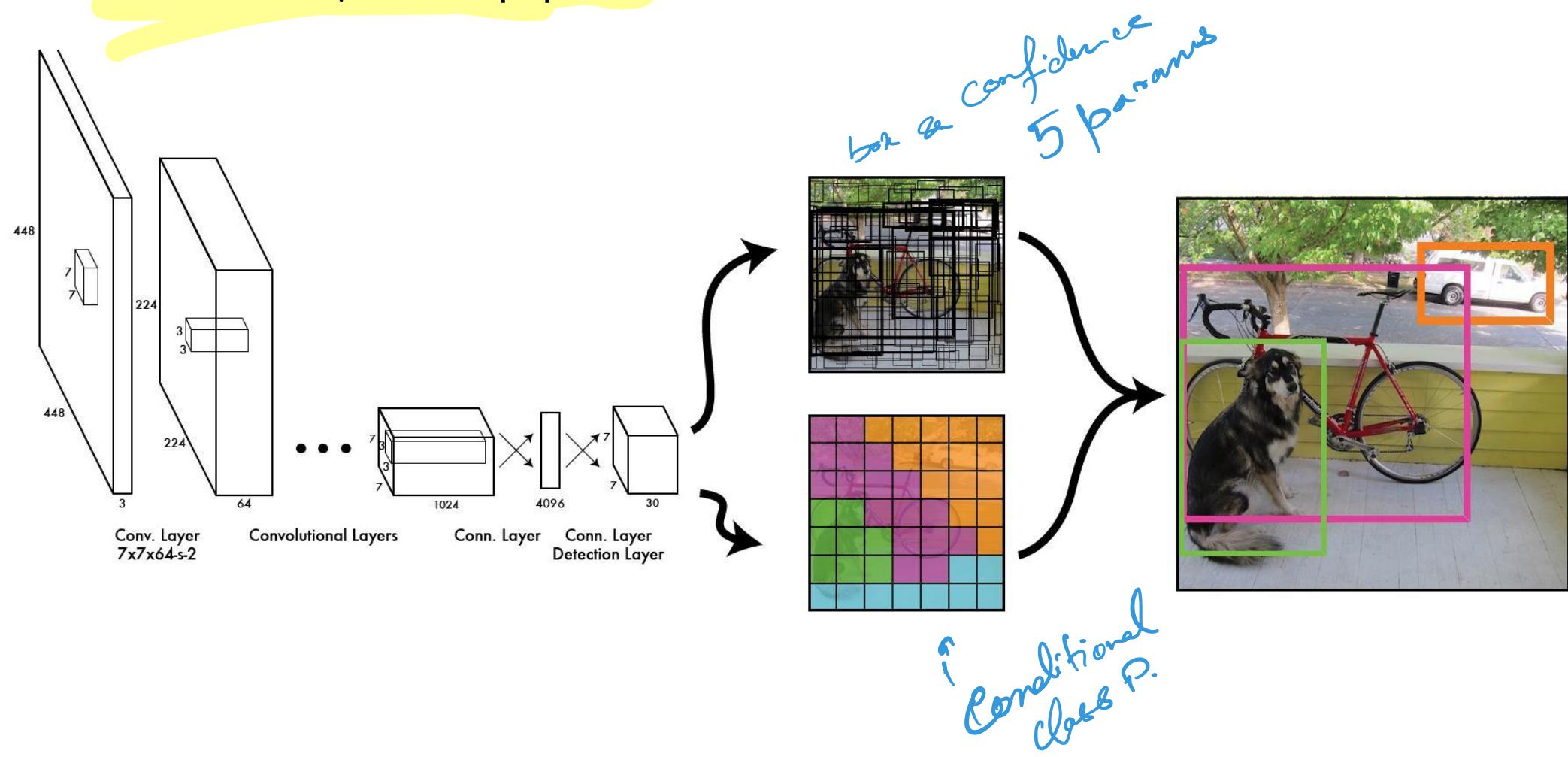


Decrease the confidence of these boxes

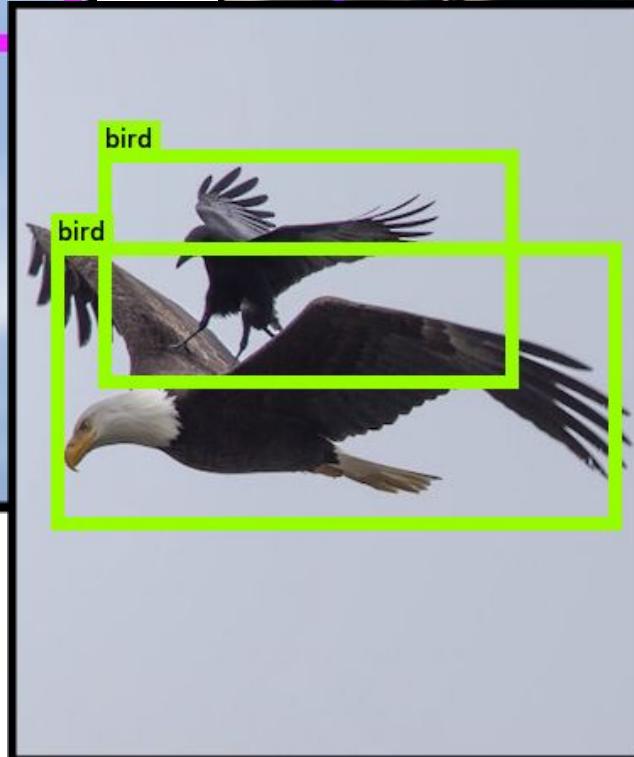
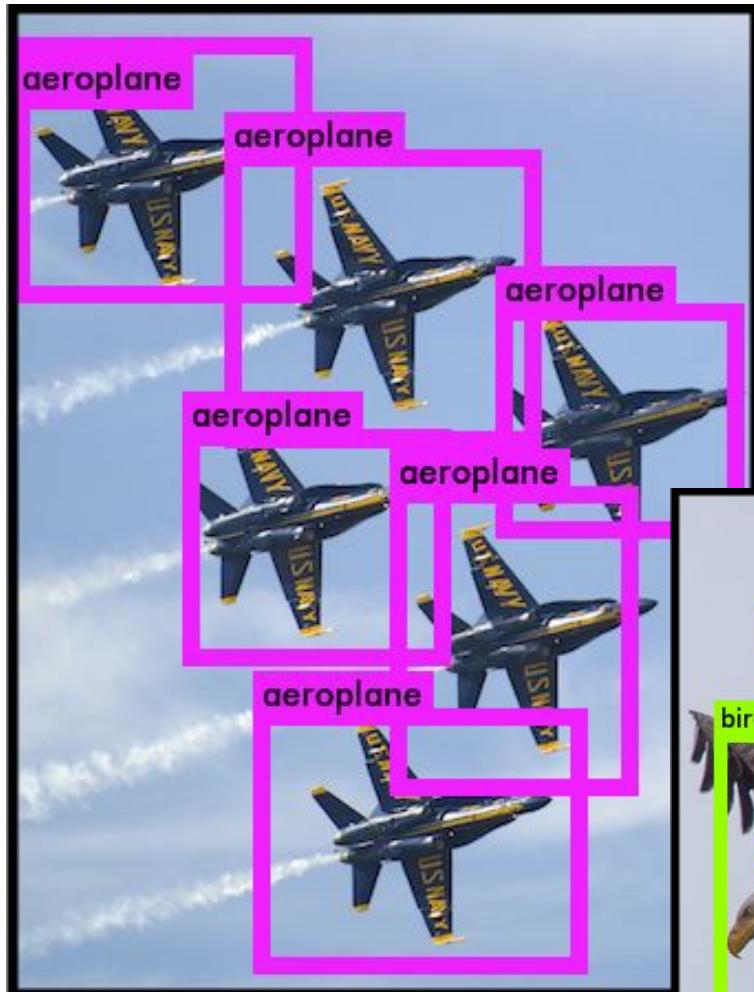


We train with standard tricks:

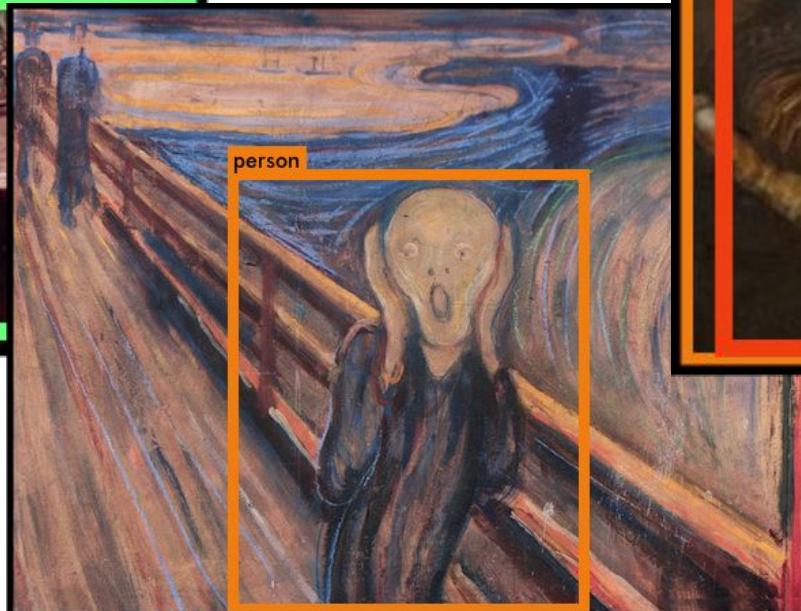
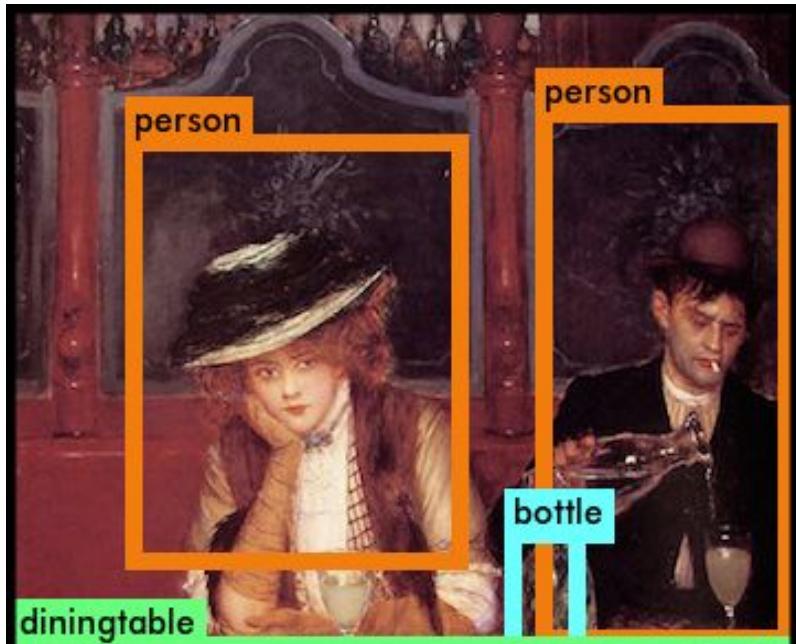
- Pretraining on Imagenet
- Extensive data augmentation
- For details, see the paper



YOLO works across a variety of natural images

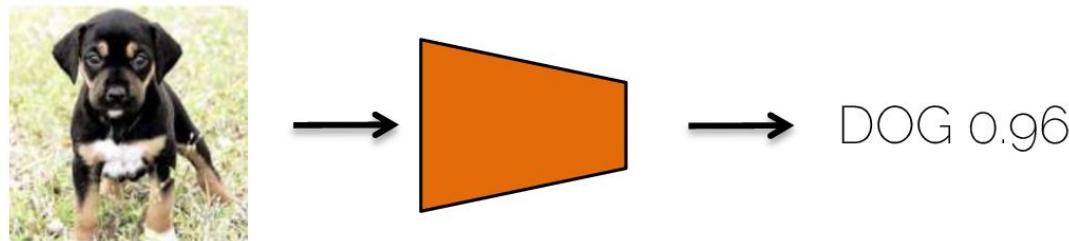


It also generalizes well to new domains (like art)



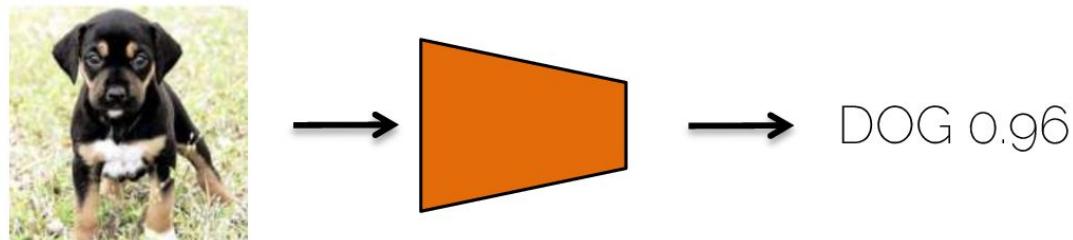
Visualizing
importance

The occlusion experiment



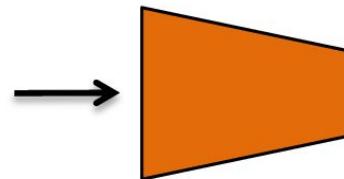
The occlusion experiment

Block different parts of the image and see how the classification score changes

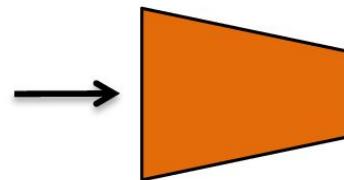


The occlusion experiment

Block different parts of the image and see how the classification score changes



→ DOG 0.95



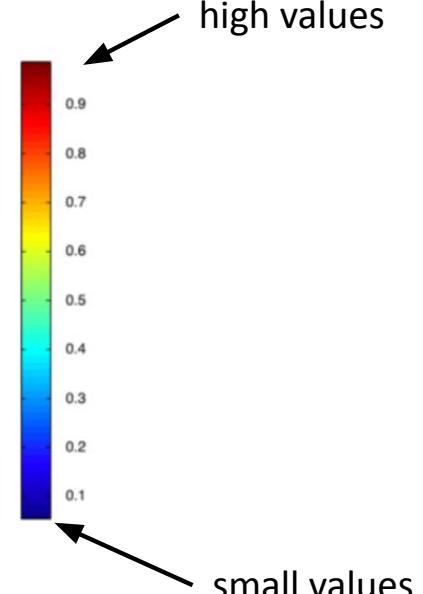
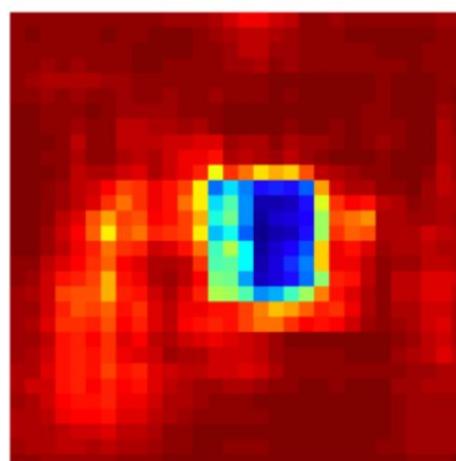
→ DOG 0.35

The face of the dog is more important for correct classification

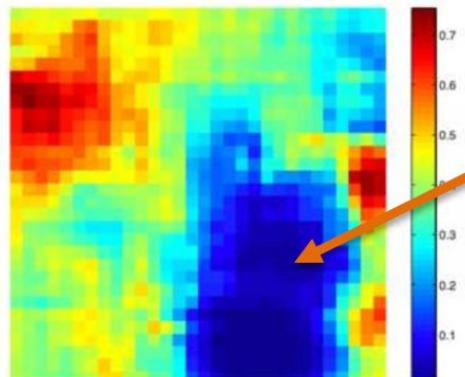
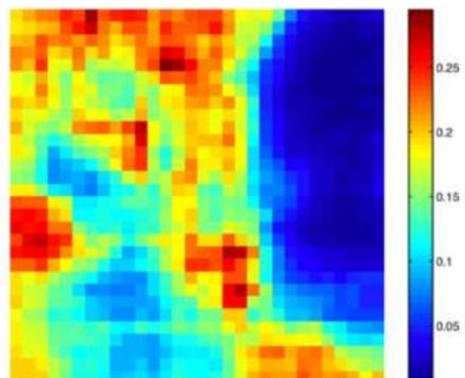
The occlusion experiment

↳ *closing/blocking*

Create a map, where each pixel represents the classification probability if an occlusion square is placed in that region



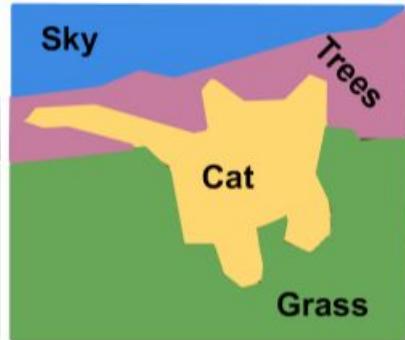
The occlusion experiment



Most important pixels for classification

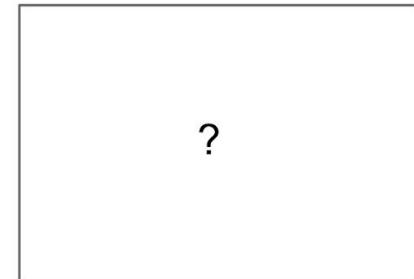
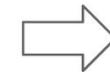
Semantic Segmentation

Semantic Segmentation



**GRASS, CAT,
TREE, SKY, ...**

Paired training data: for each training image, each pixel is labeled with a semantic category.



At test time, classify each pixel of a new image.

Semantic Segmentation Idea: Sliding Window

Full image



Semantic Segmentation Idea: Sliding Window

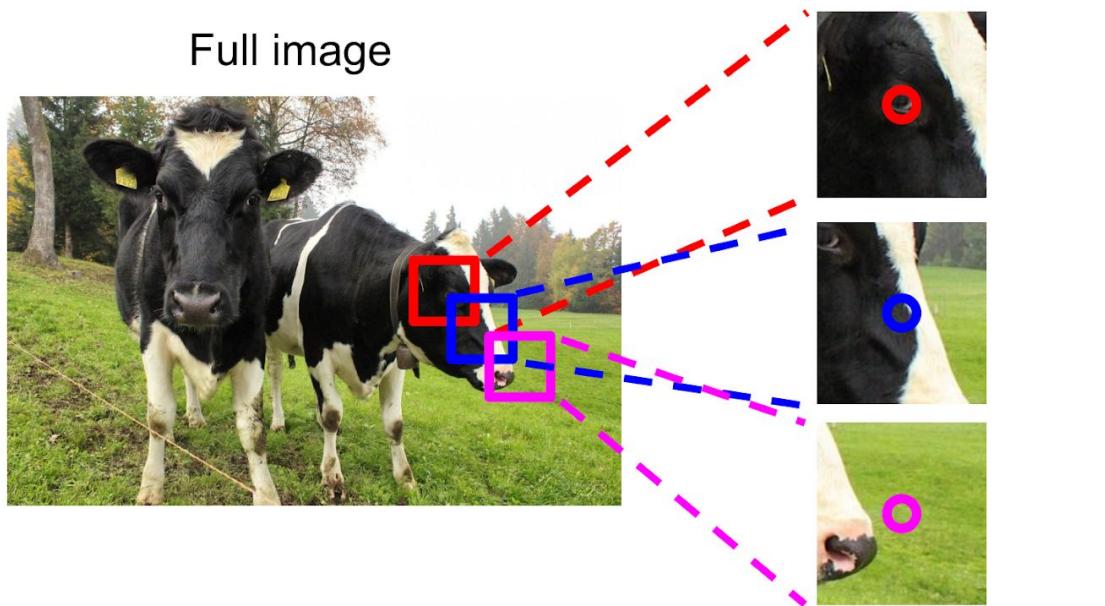
Full image



Impossible to classify without context

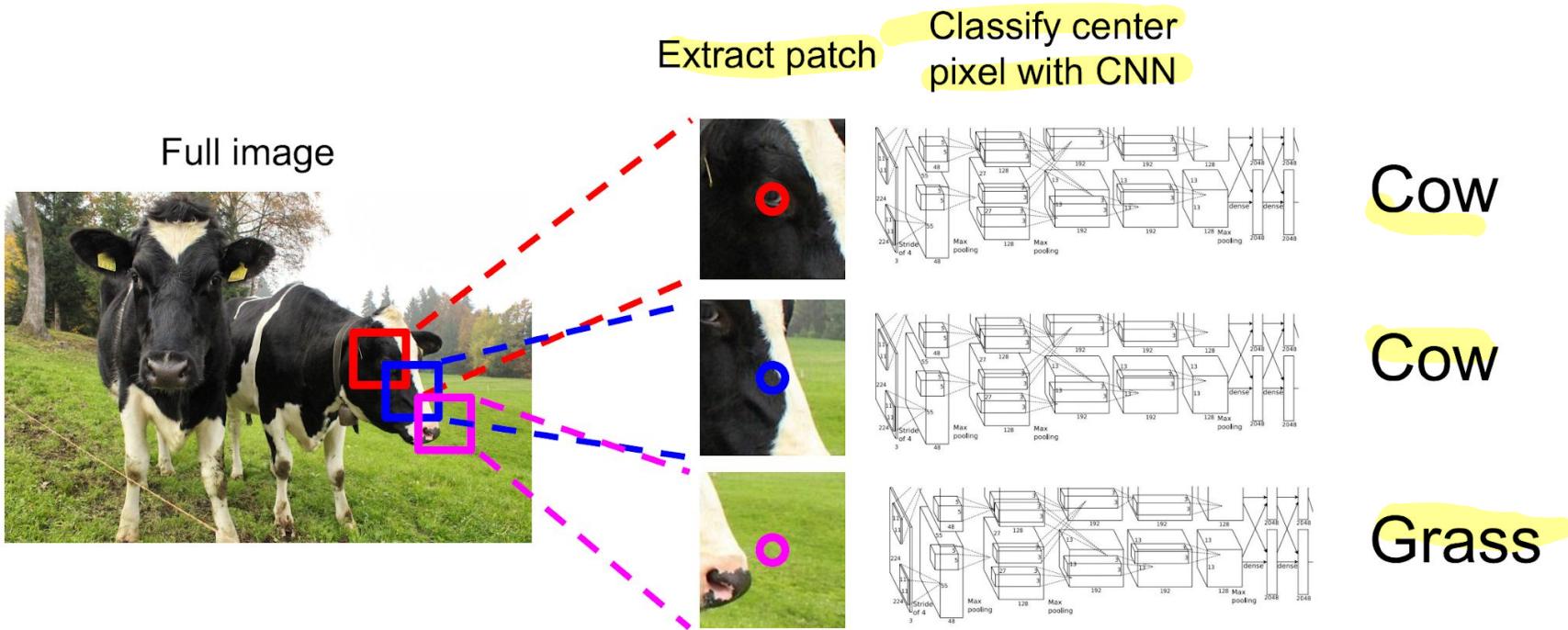
Q: how do we include context?

Semantic Segmentation Idea: Sliding Window



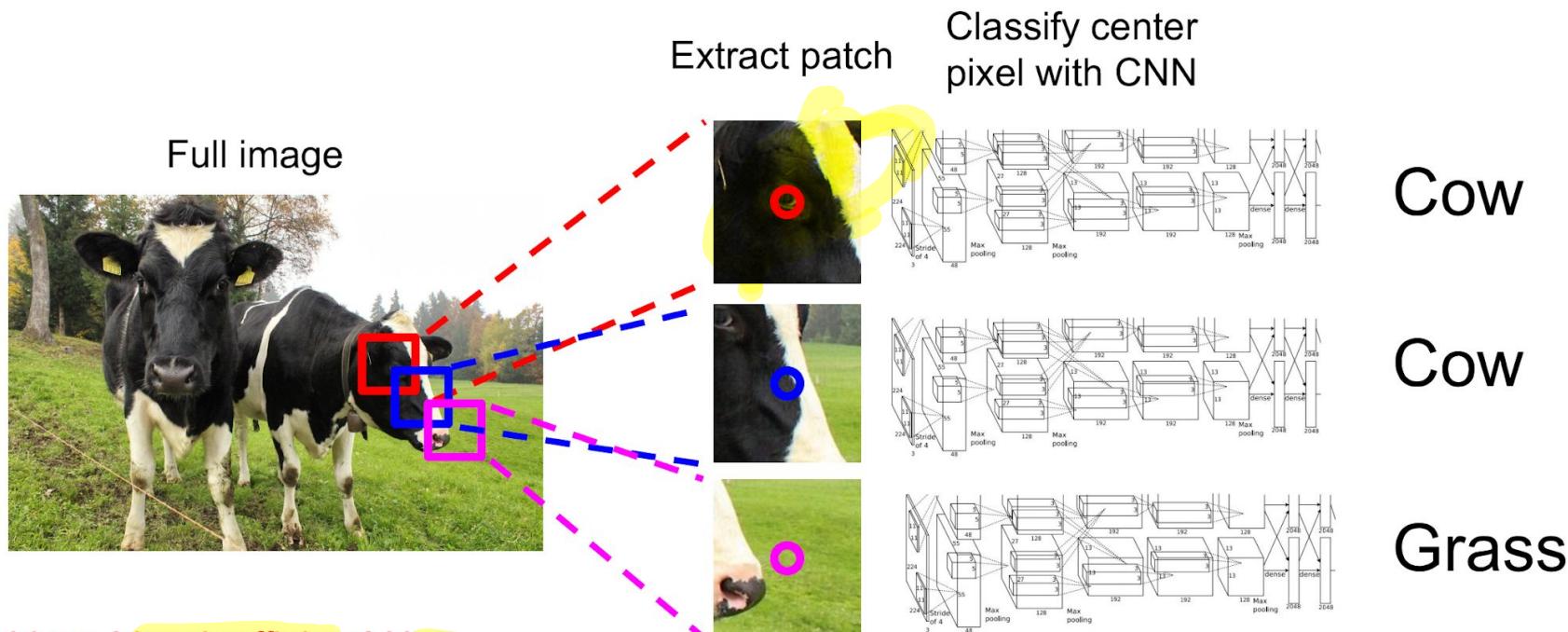
Q: how do we model this?

Semantic Segmentation Idea: Sliding Window



Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation Idea: Sliding Window

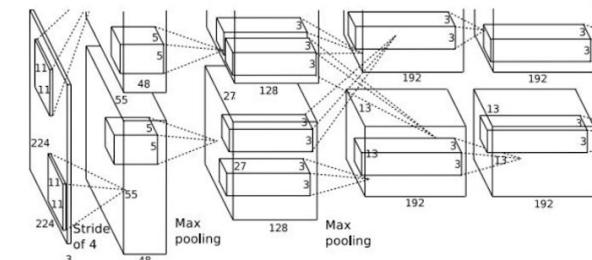


Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation idea: CNN

Full image



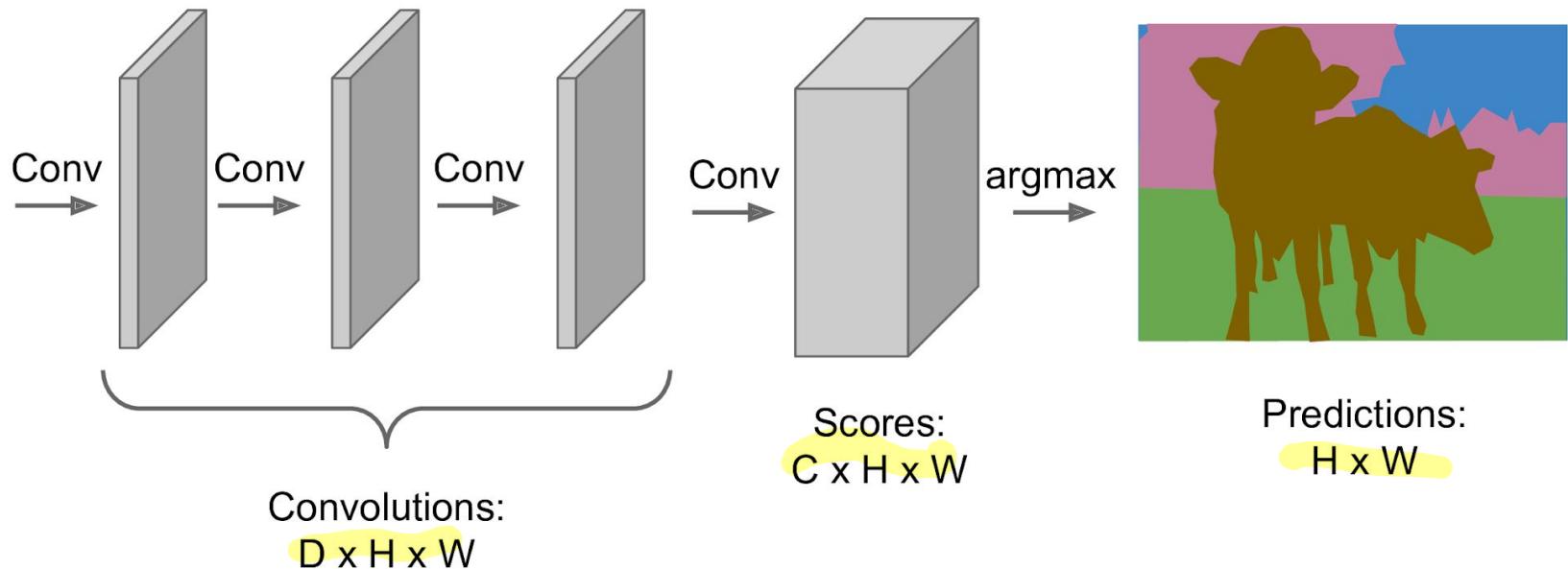
Challenge: output size is of the
order of input size

Semantic Segmentation idea: CNN

Design a network with only convolutional layers
without downsampling operators to make predictions
for pixels all at once!

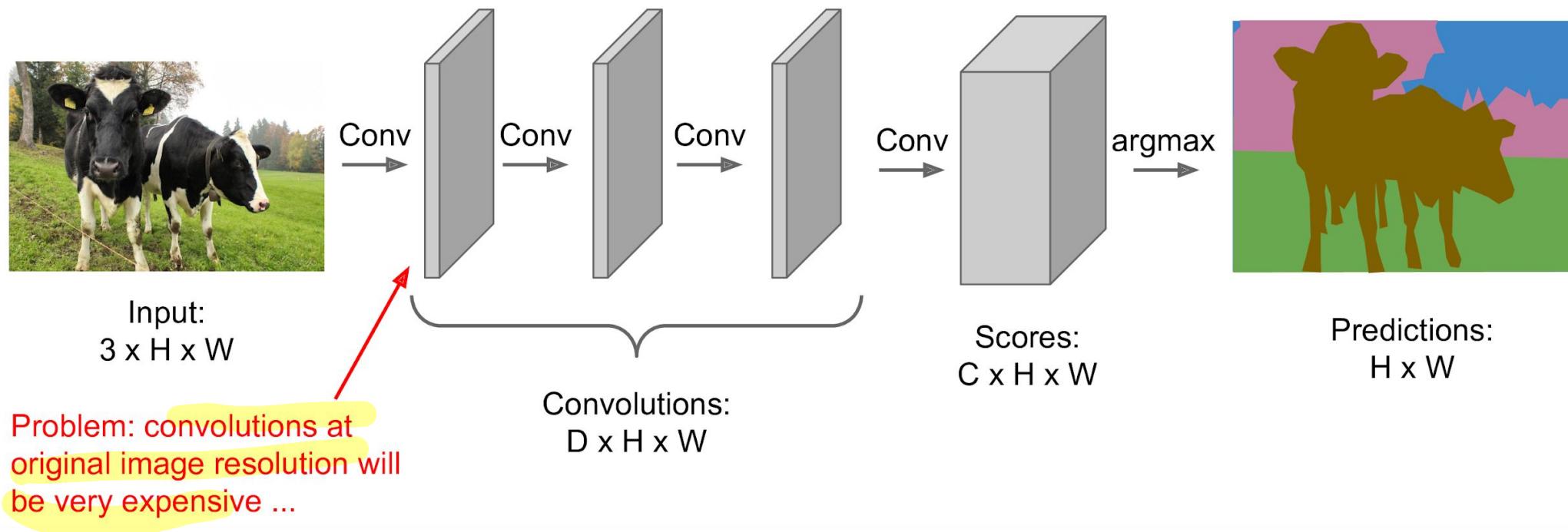


Input:
 $3 \times H \times W$



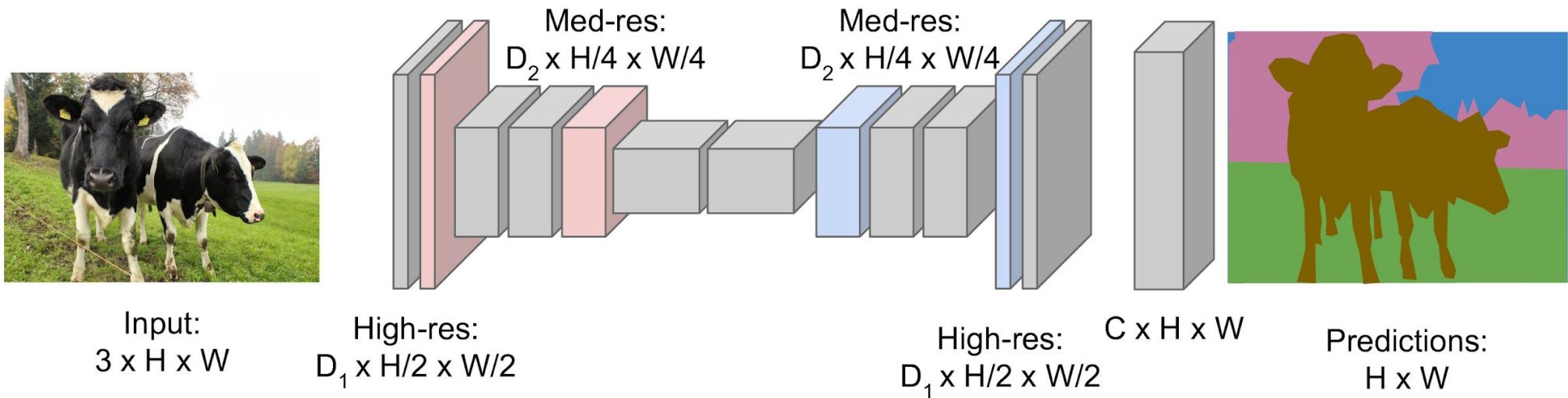
Semantic Segmentation idea: CNN

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



Semantic Segmentation idea: CNN

Design network as a bunch of convolutional layers, with
downsampling and **upsampling** inside the network!



Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

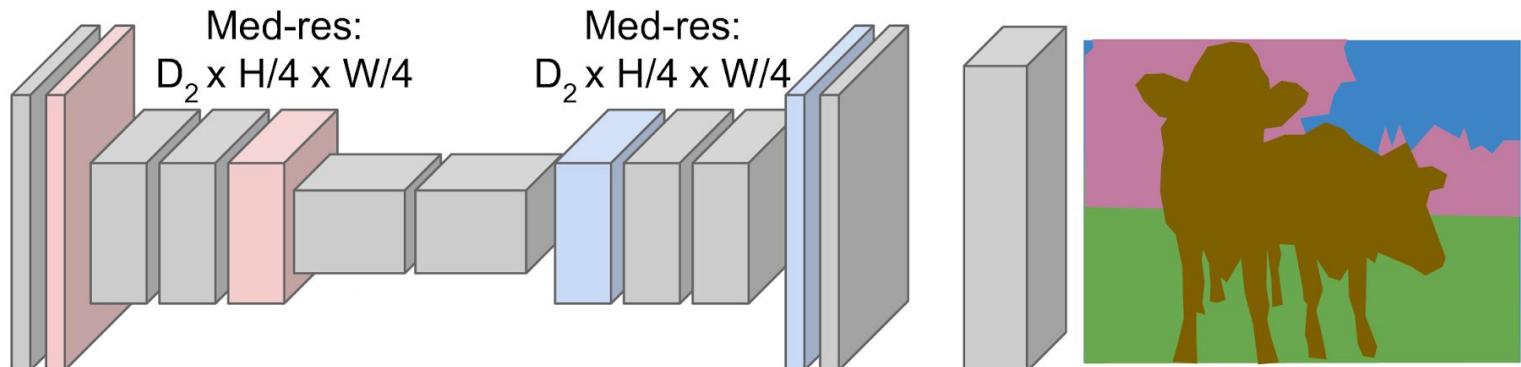
Semantic Segmentation idea: CNN

Downsampling:
Pooling, strided
convolution



Input:
 $3 \times H \times W$

Design network as a bunch of convolutional layers, with
downsampling and **upsampling** inside the network!



High-res:
 $D_1 \times H/2 \times W/2$

High-res:
 $D_1 \times H/2 \times W/2$

Upsampling:
???

Predictions:
 $H \times W$

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

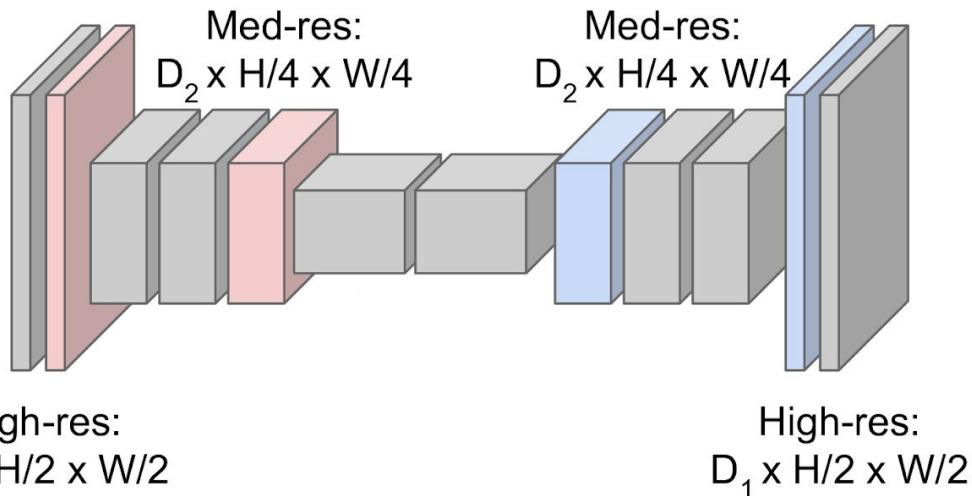
Semantic Segmentation idea: CNN

Downsampling:
Pooling, strided convolution



Input:
 $3 \times H \times W$

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



Upsampling:
Unpooling or
transposed convolution



Predictions:
 $H \times W$

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Semantic Segmentation Limitation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels



[This image is CC0 public domain](#)

