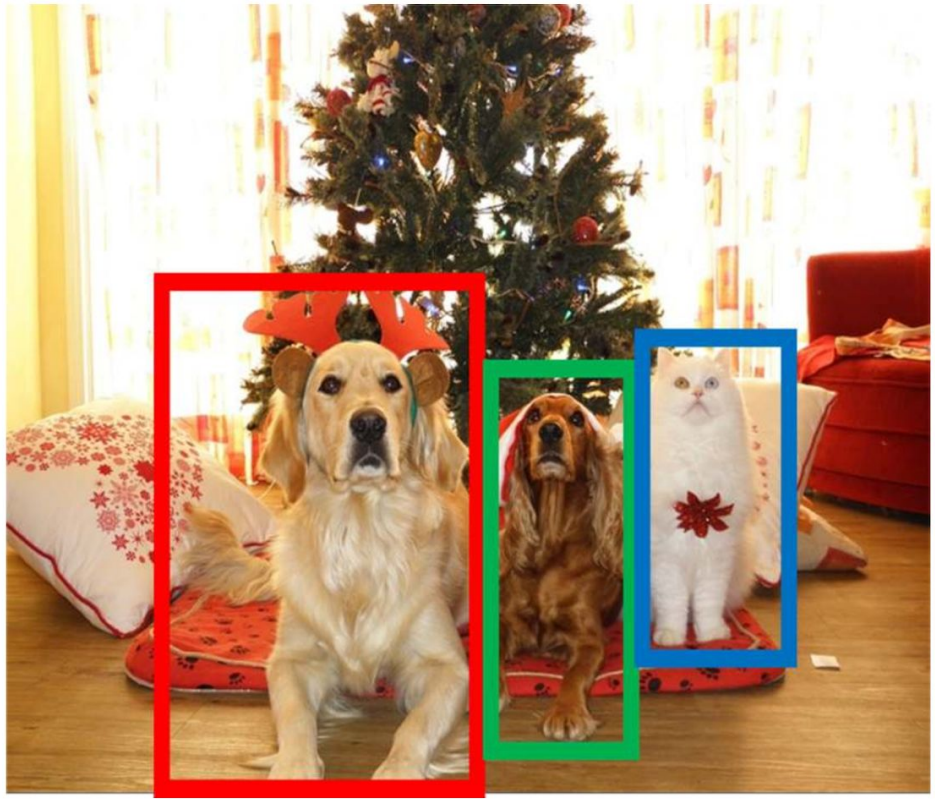
# Object Recognition and Face Recognition

# Object Detection: Task Definition

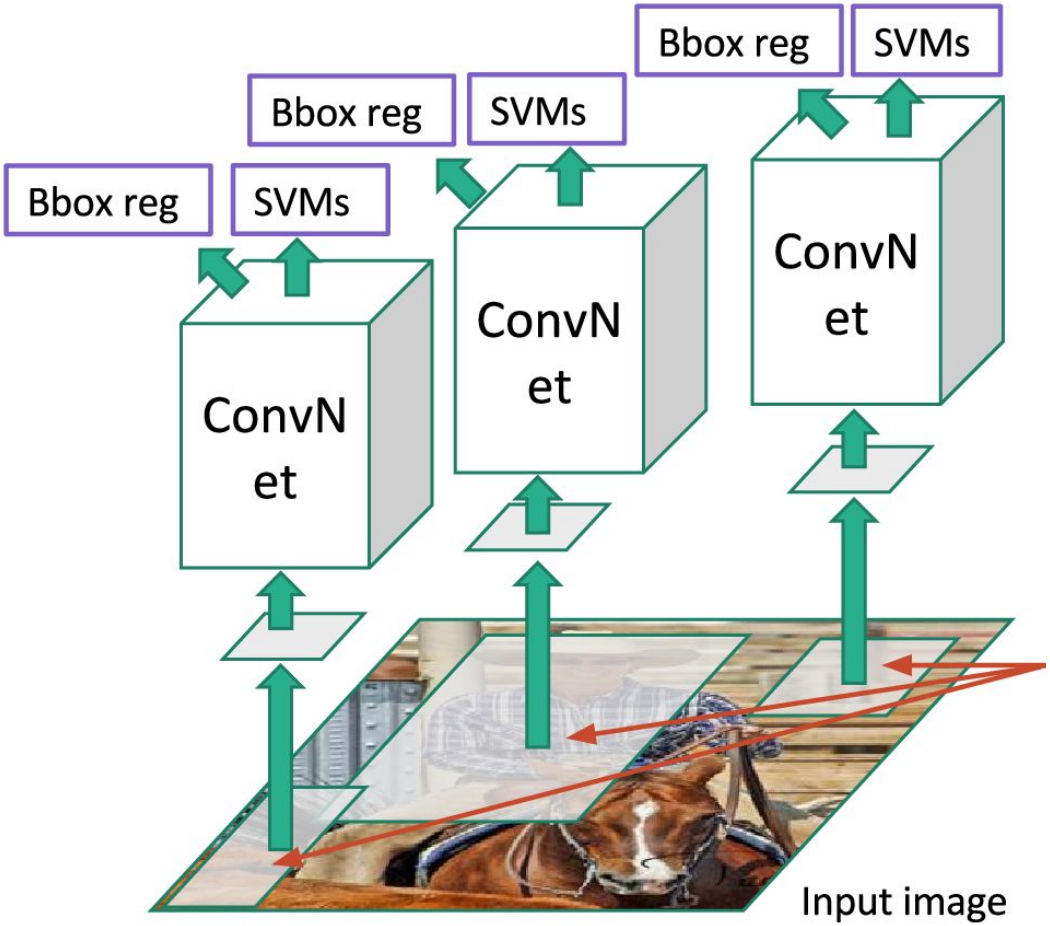
**Input**: Single RGB Image

**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)



# R-CNN




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

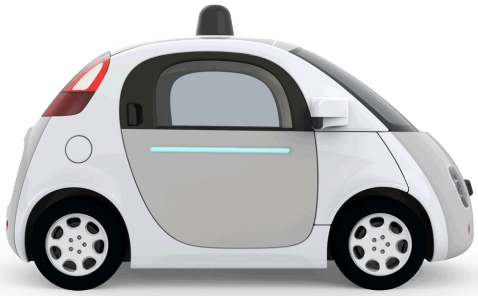
# Accurate object detection is slow!

| | **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 |

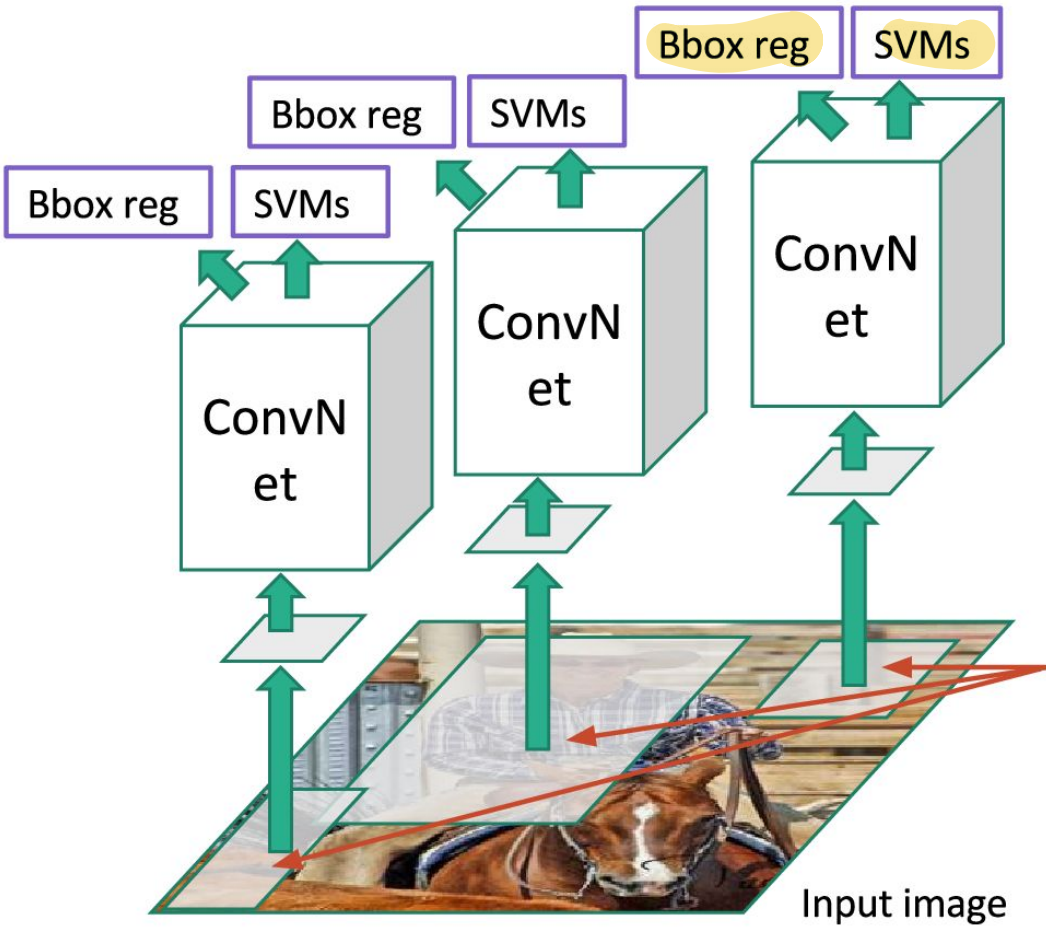
**60 miles/H (96.5 Km/H)**



⅓ Mile, 1760 feet

**20 sec**

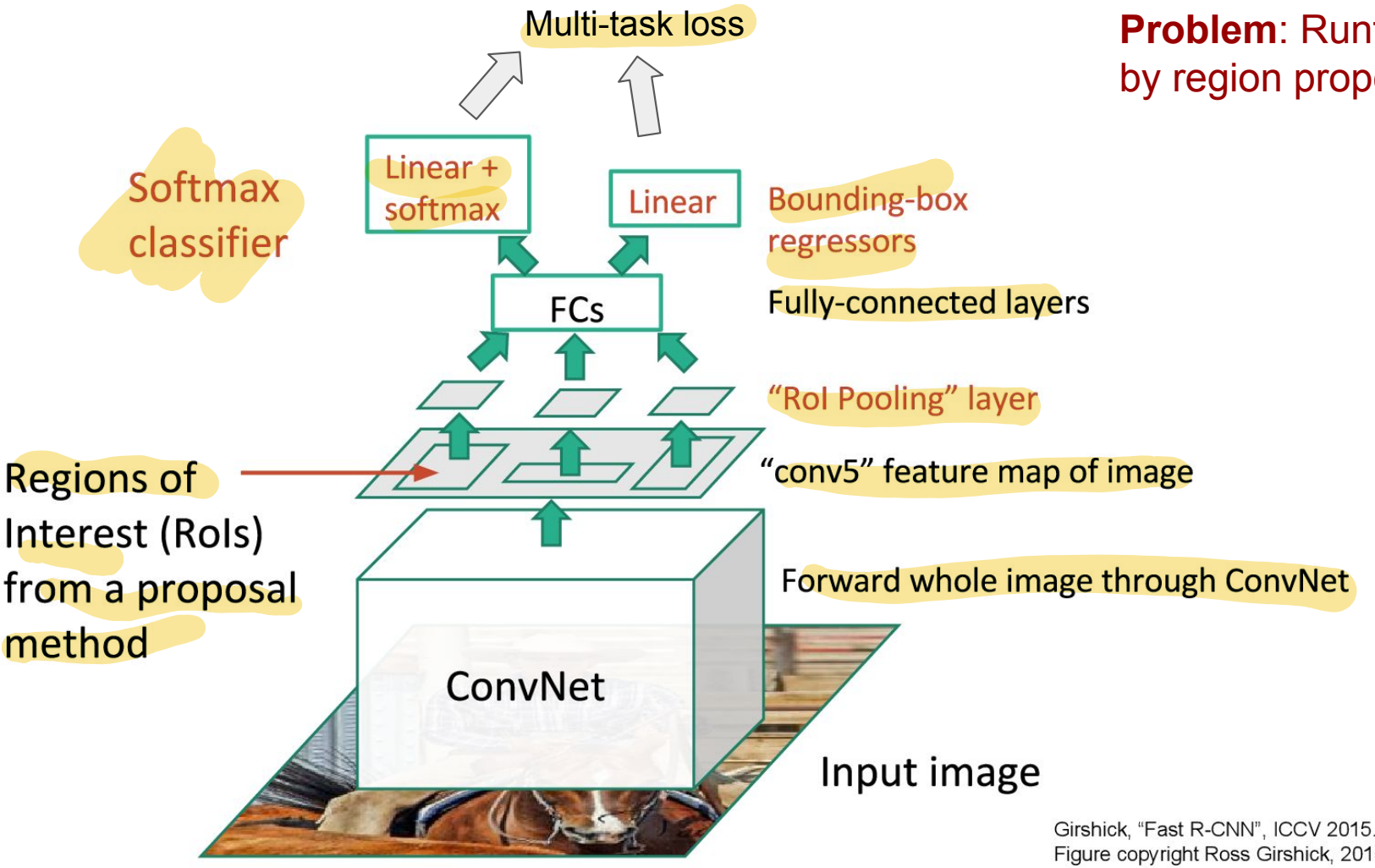
# R-CNN




**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



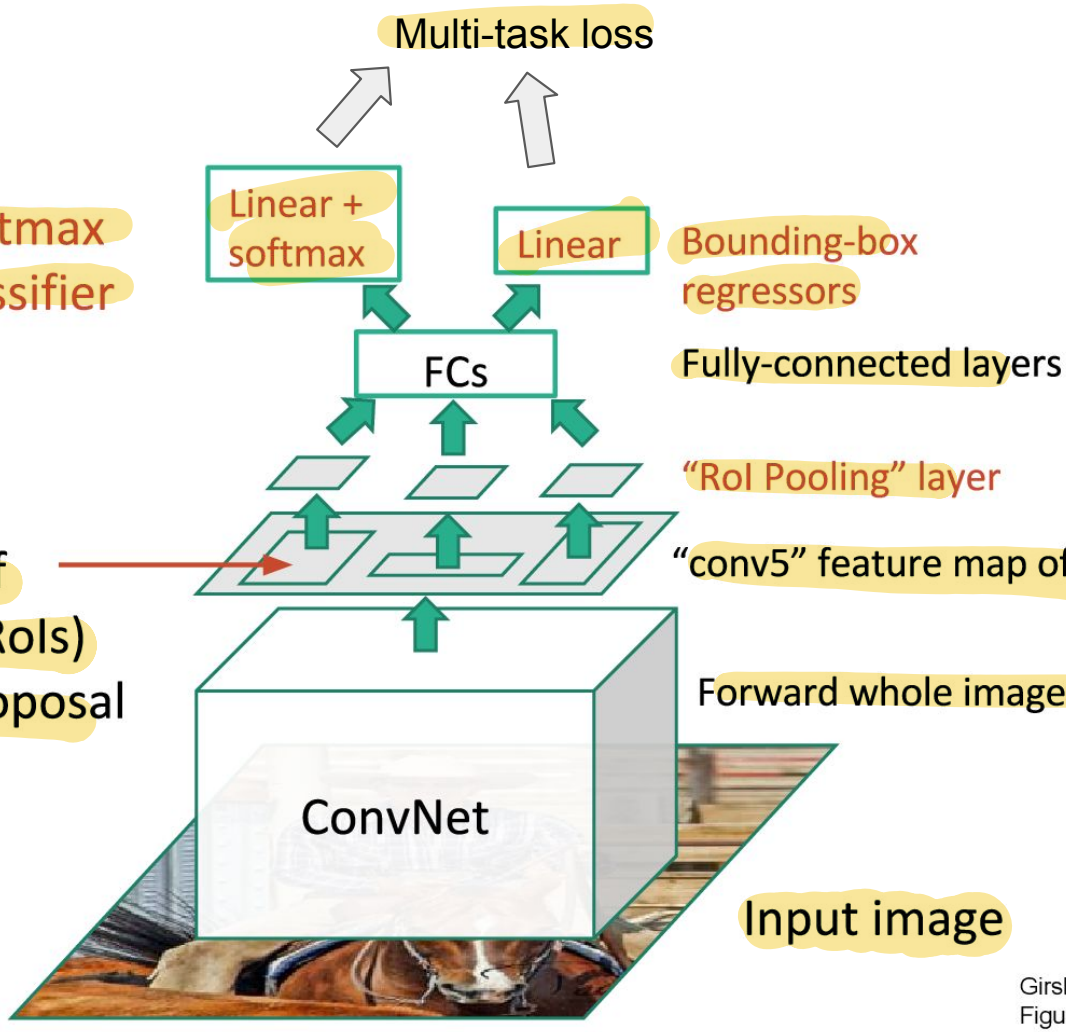

**Problem**: Runtime dominated by region proposals!

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 |

# Fast R-CNN



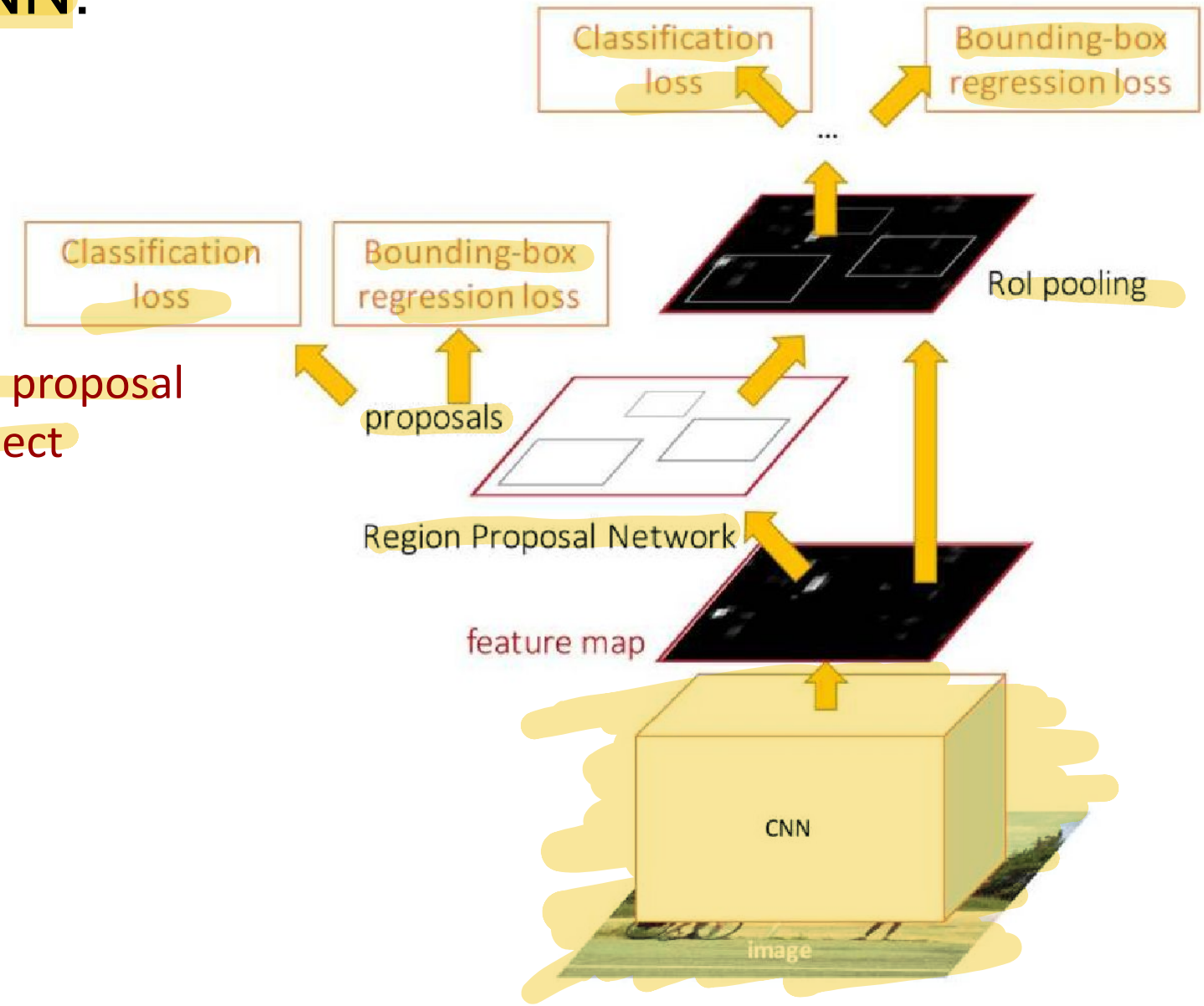

**Problem**: Runtime dominated by region proposals!

**Solution**: make CNN do region proposals.

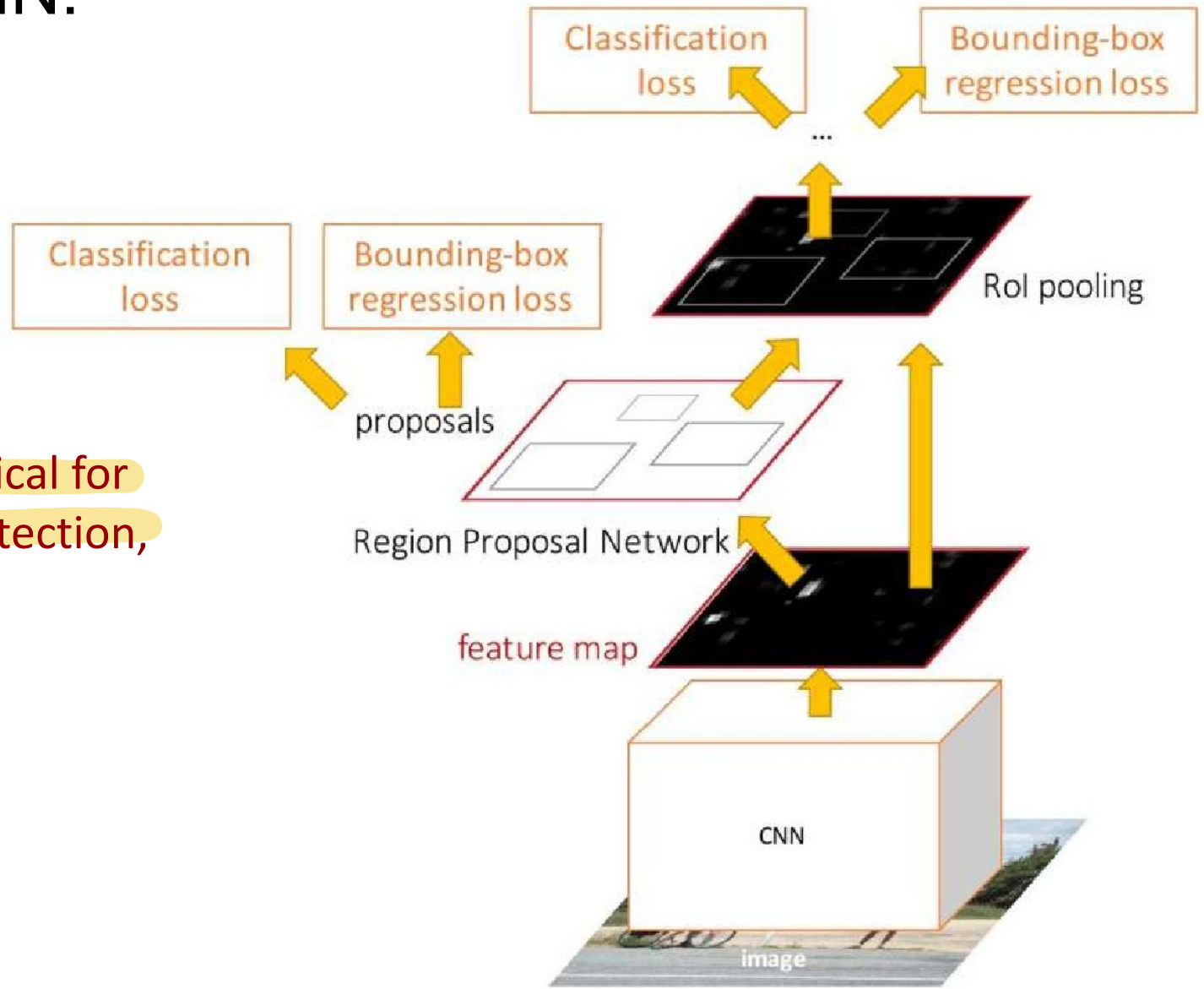
Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

# Faster R-CNN:

**Two stages**: region proposal generation and object classification



# Faster R-CNN:



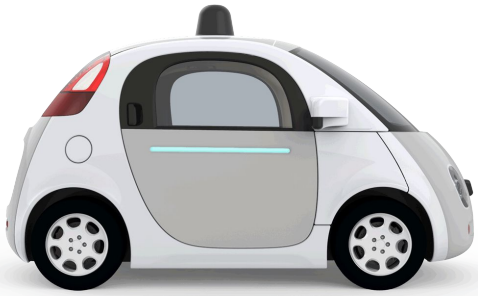

**Problem**: Not practical for real time Object Detection, Inference is slow

# 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!

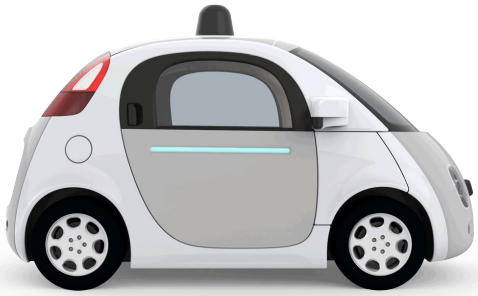
| | **Pascal 2007 mAP** | **Speed** | |
|---|---|---|---|
| R-CNN | 66.0 | .05 FPS | 20 s/img |
| Fast R-CNN | 70.0 | .5 FPS | 2 s/img |
| Faster R-CNN | 73.2 | 7 FPS | 140 ms/img |
| YOLO | 63.4 | 45 FPS | 22 ms/img |



2 feet

# 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 |



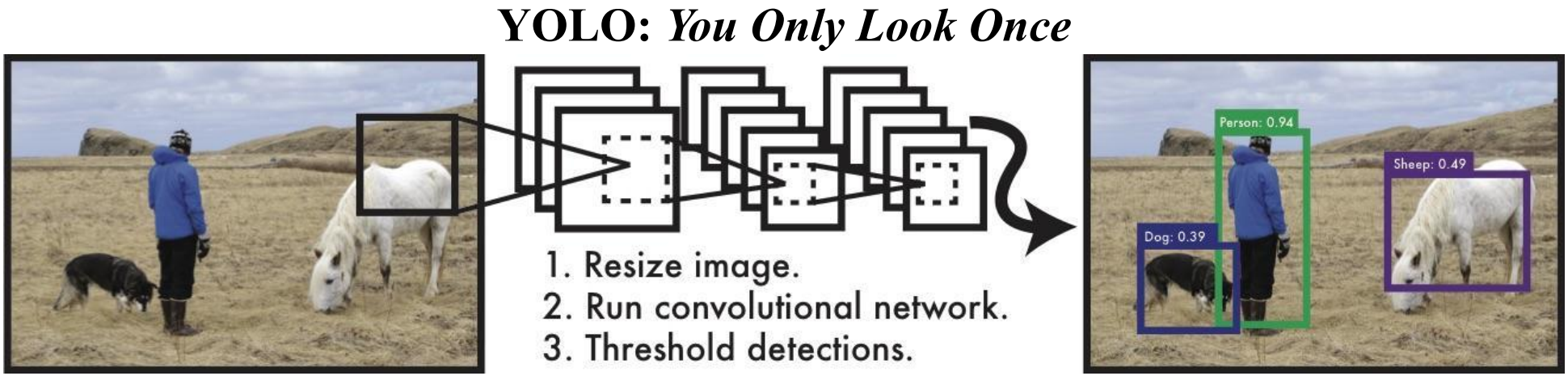
2 feet

YOLO
↳ resize image
• run convolution network
• Threshold detections.


resize
Convolve

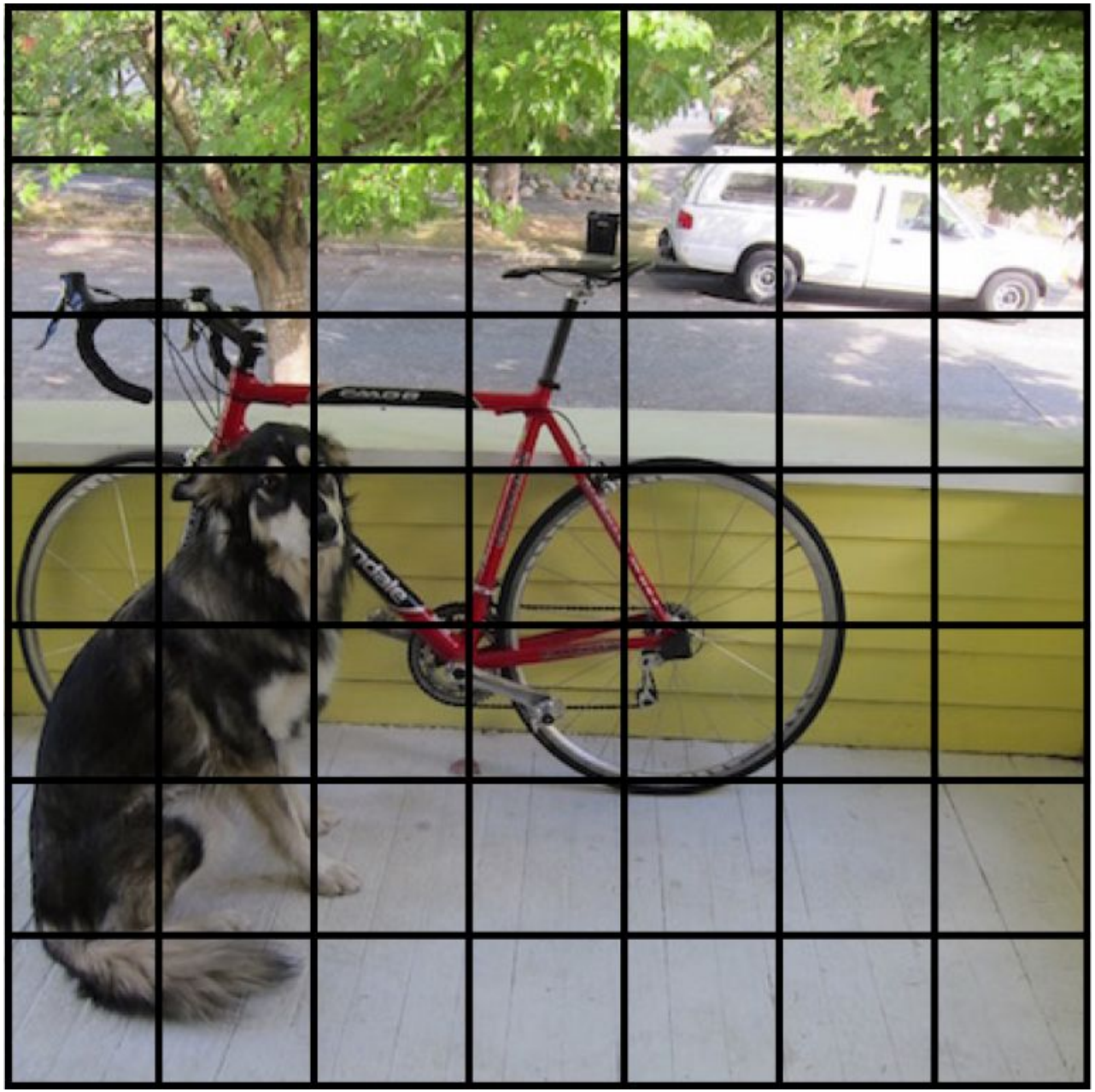
Threshold.

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




EAGLE
ridley

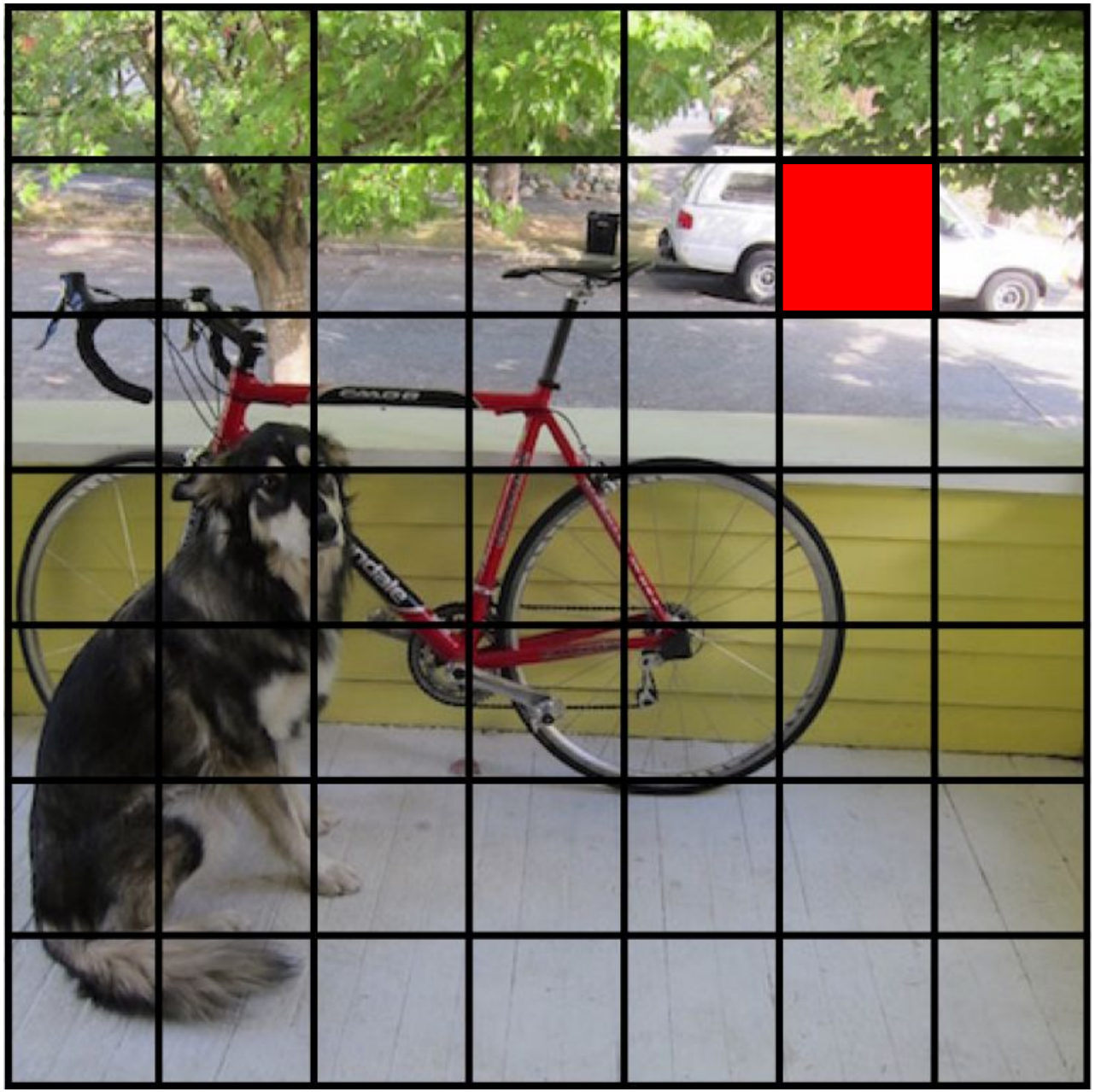
We split the image into a grid



→ detects boxes and the pc object)
→ the class probability

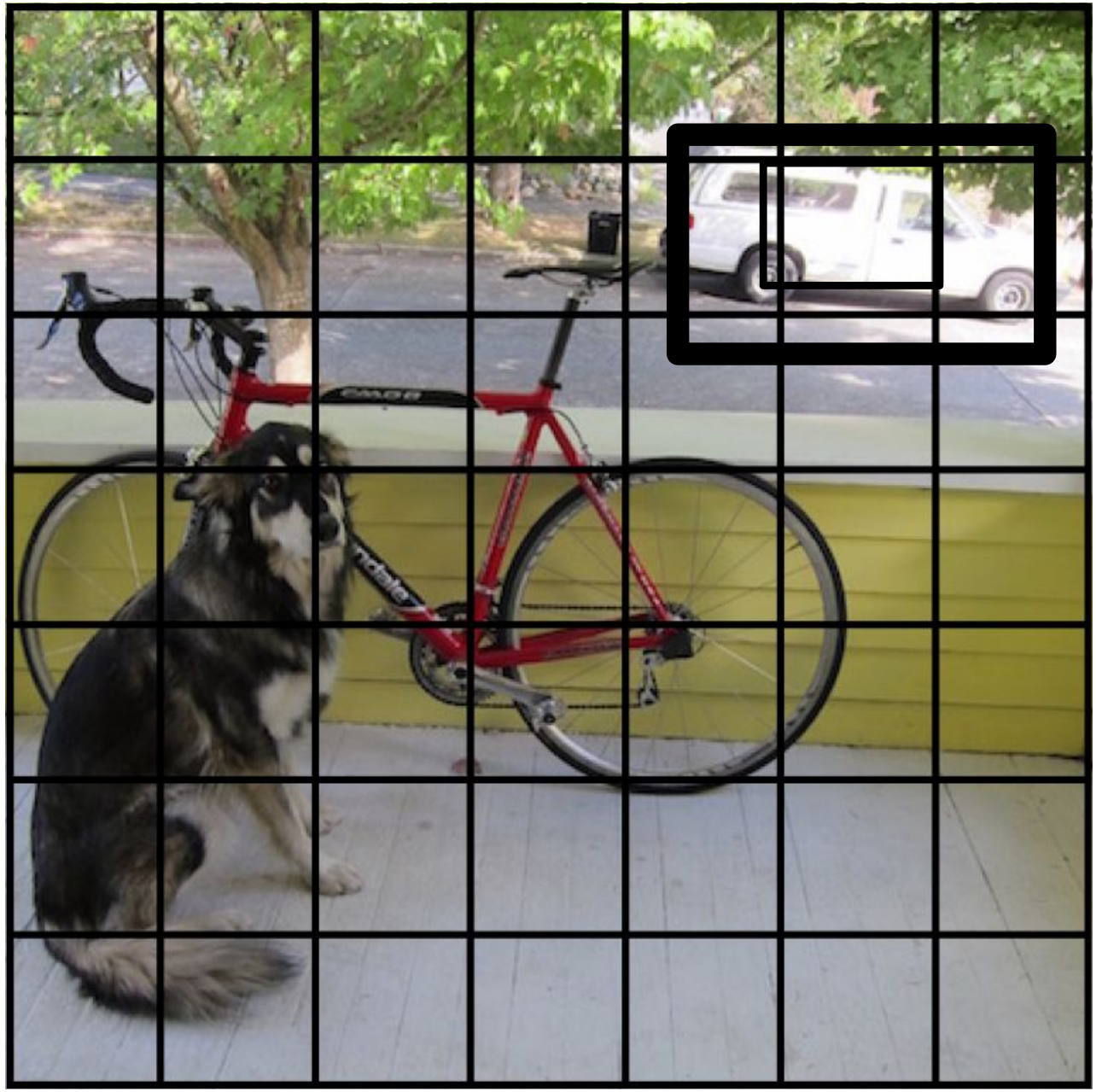
P(object) & P(dog | object)

# Each cell predicts boxes and confidences: P(Object)

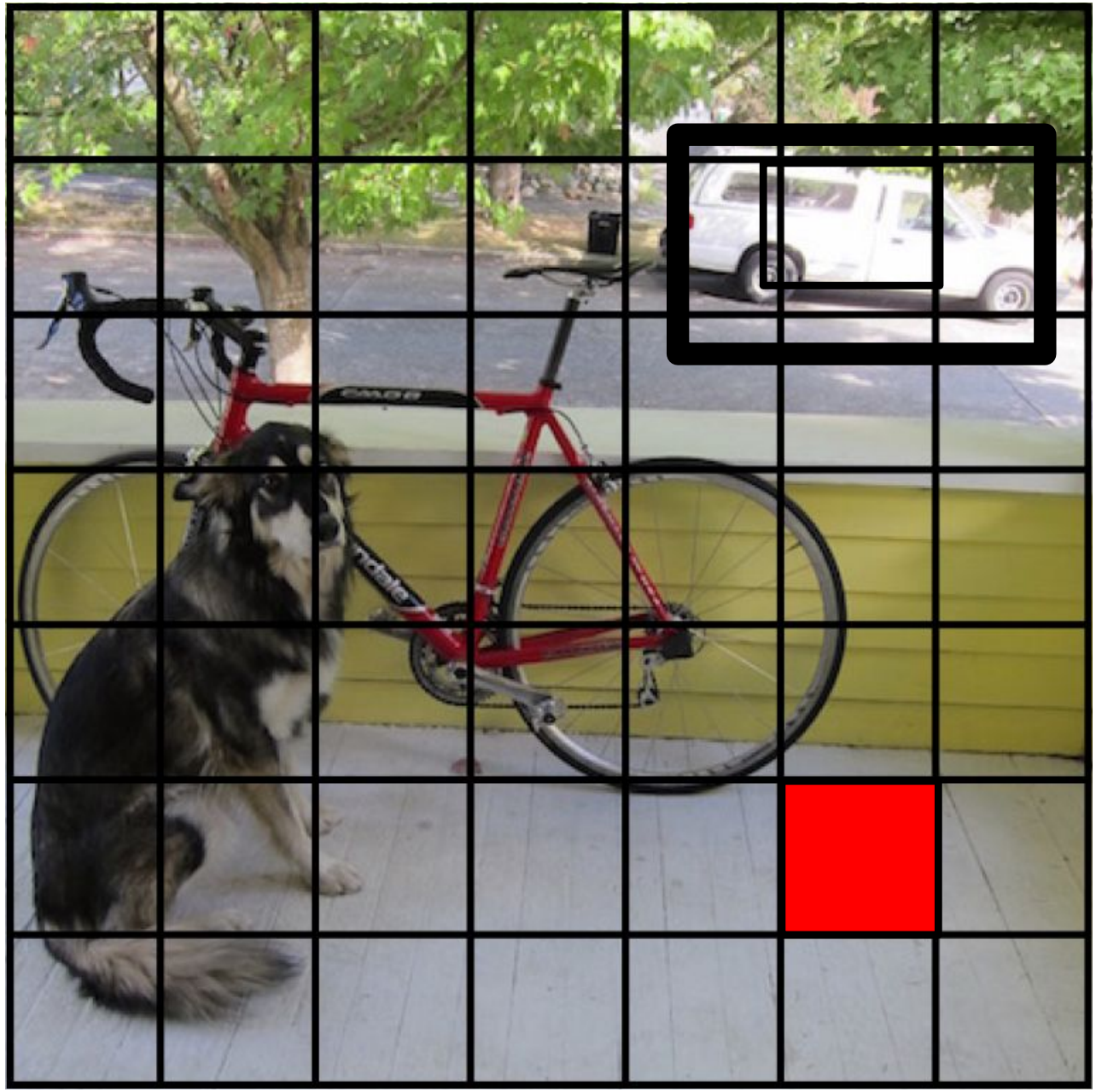


# Each cell predicts boxes and confidences: P(Object)

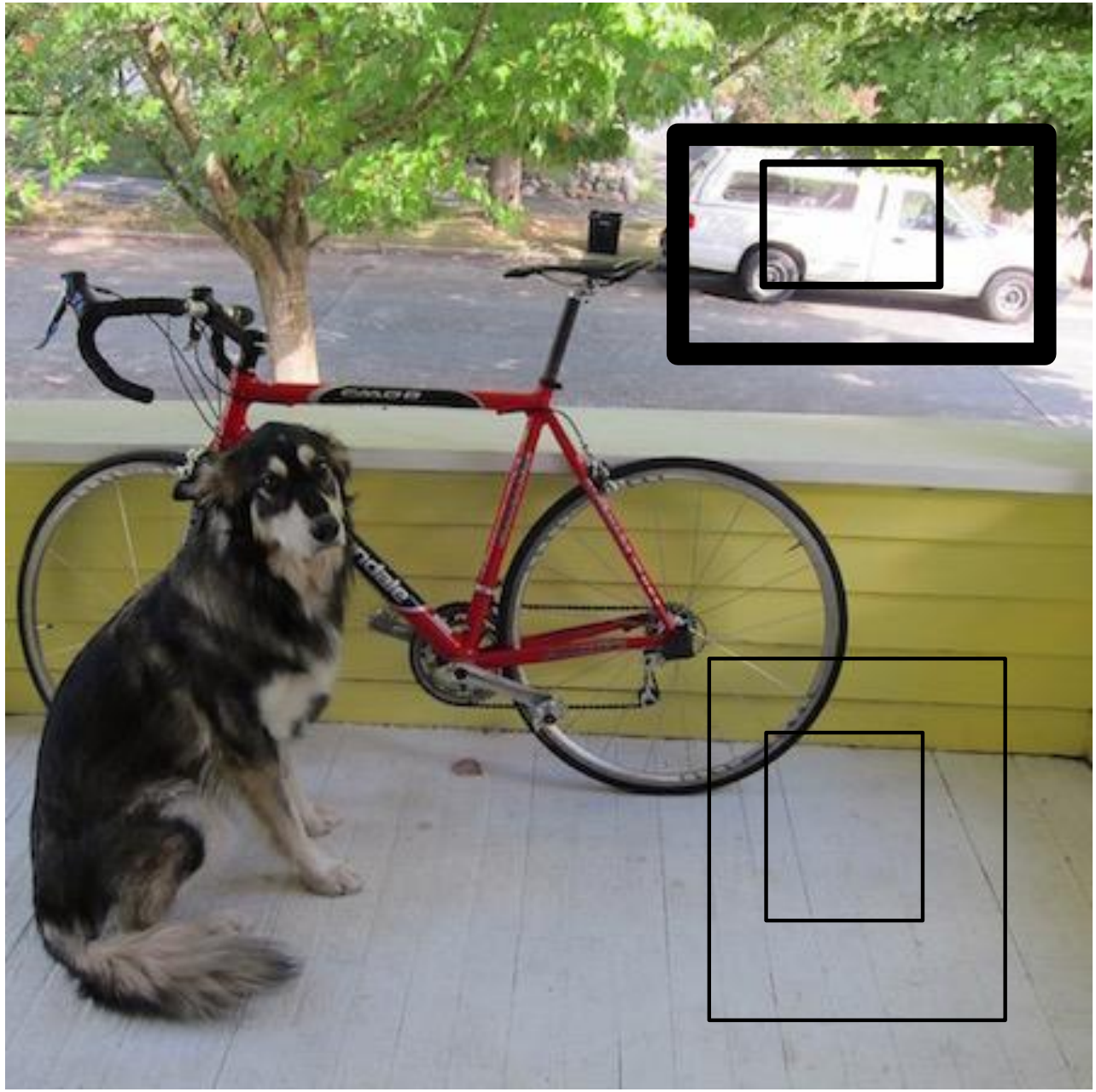
Each cell predicts boxes and confidences: P(Object)



# Each cell predicts boxes and confidences: P(Object)



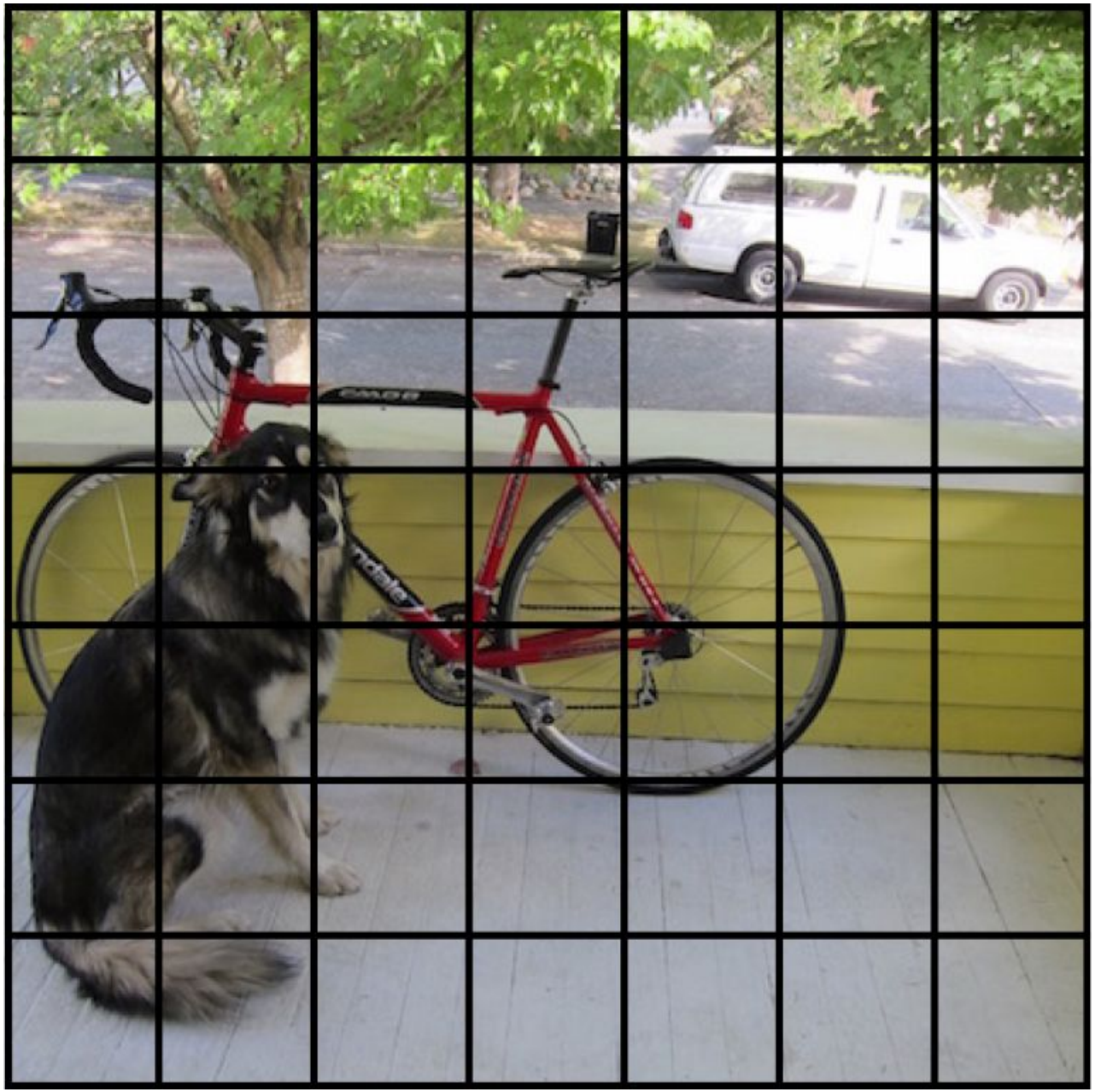
# Each cell predicts boxes and confidences: P(Object)



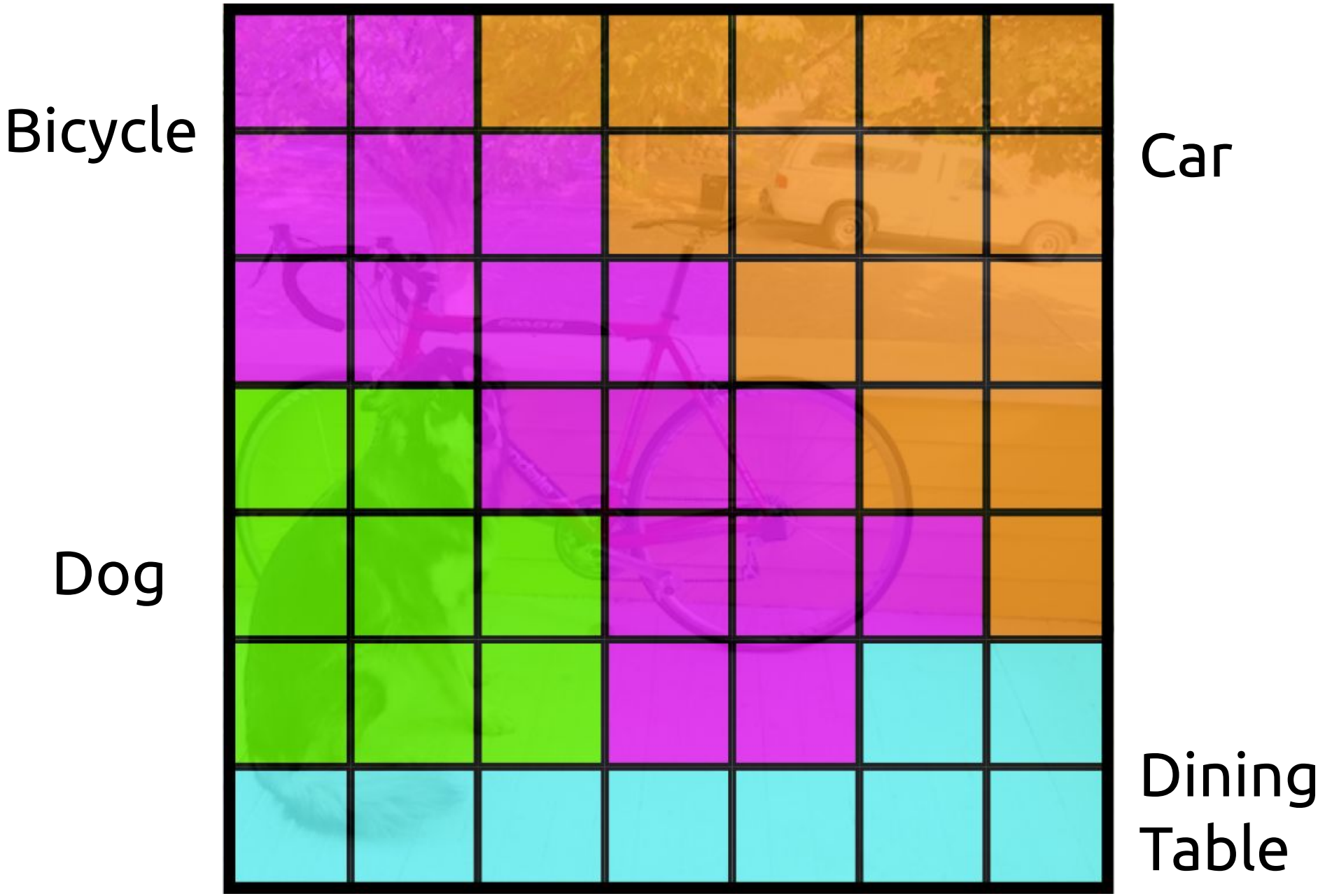
# Each cell predicts boxes and confidences: P(Object)

Each cell → predicts bboxes & confidences
Each cell → A class probability

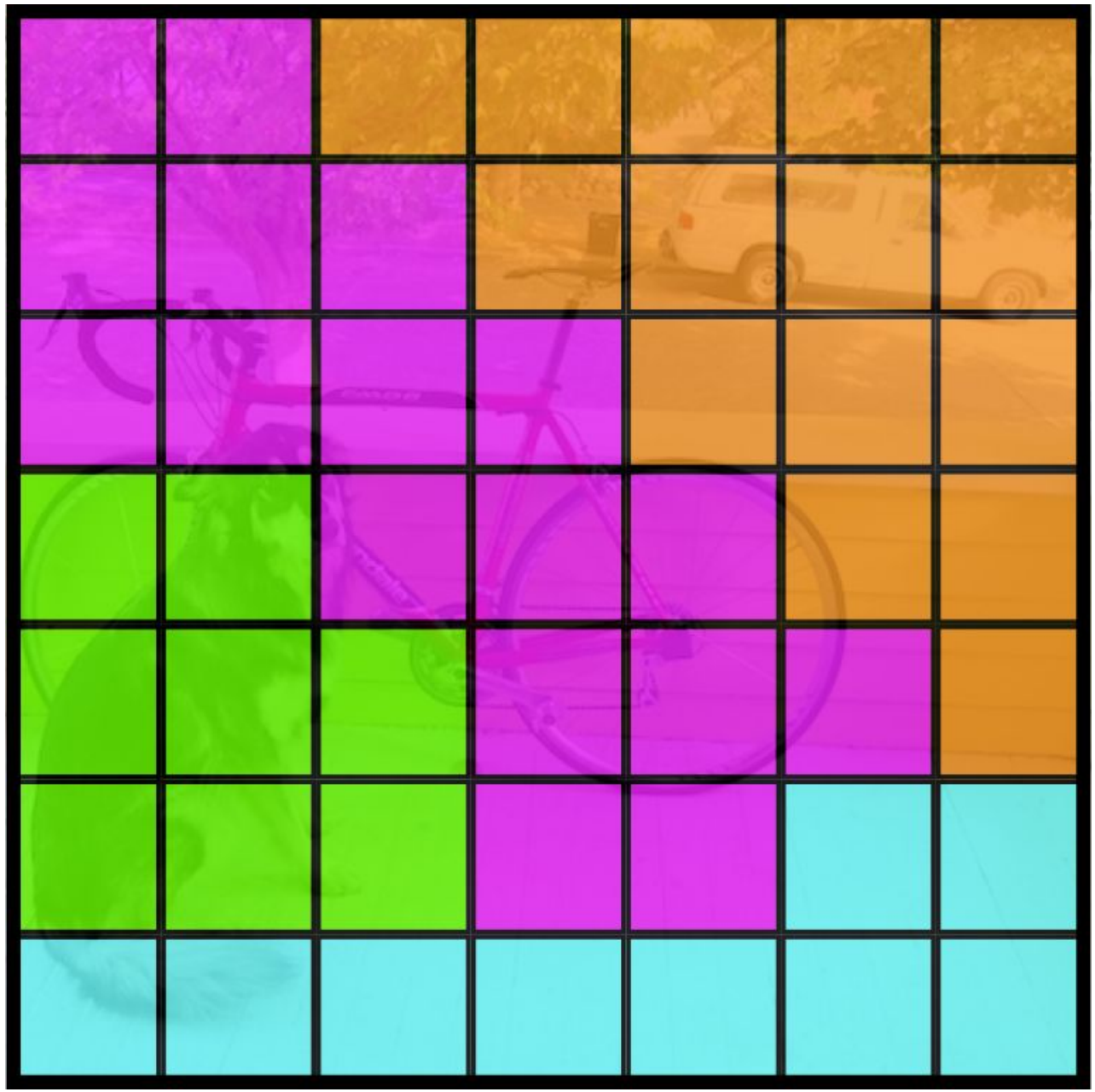
Each cell also predicts a class probability.



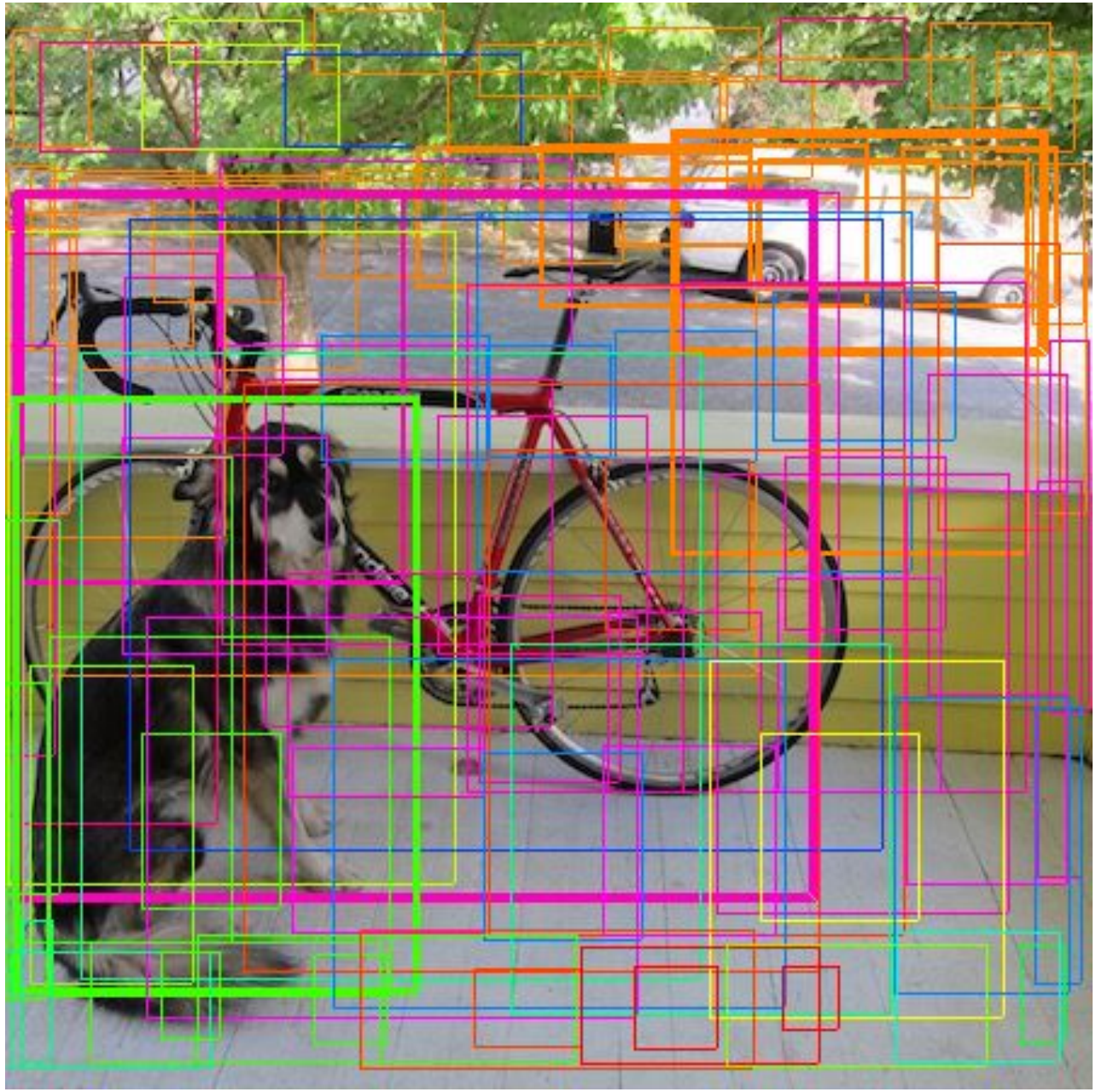
Each cell also predicts a class probability.



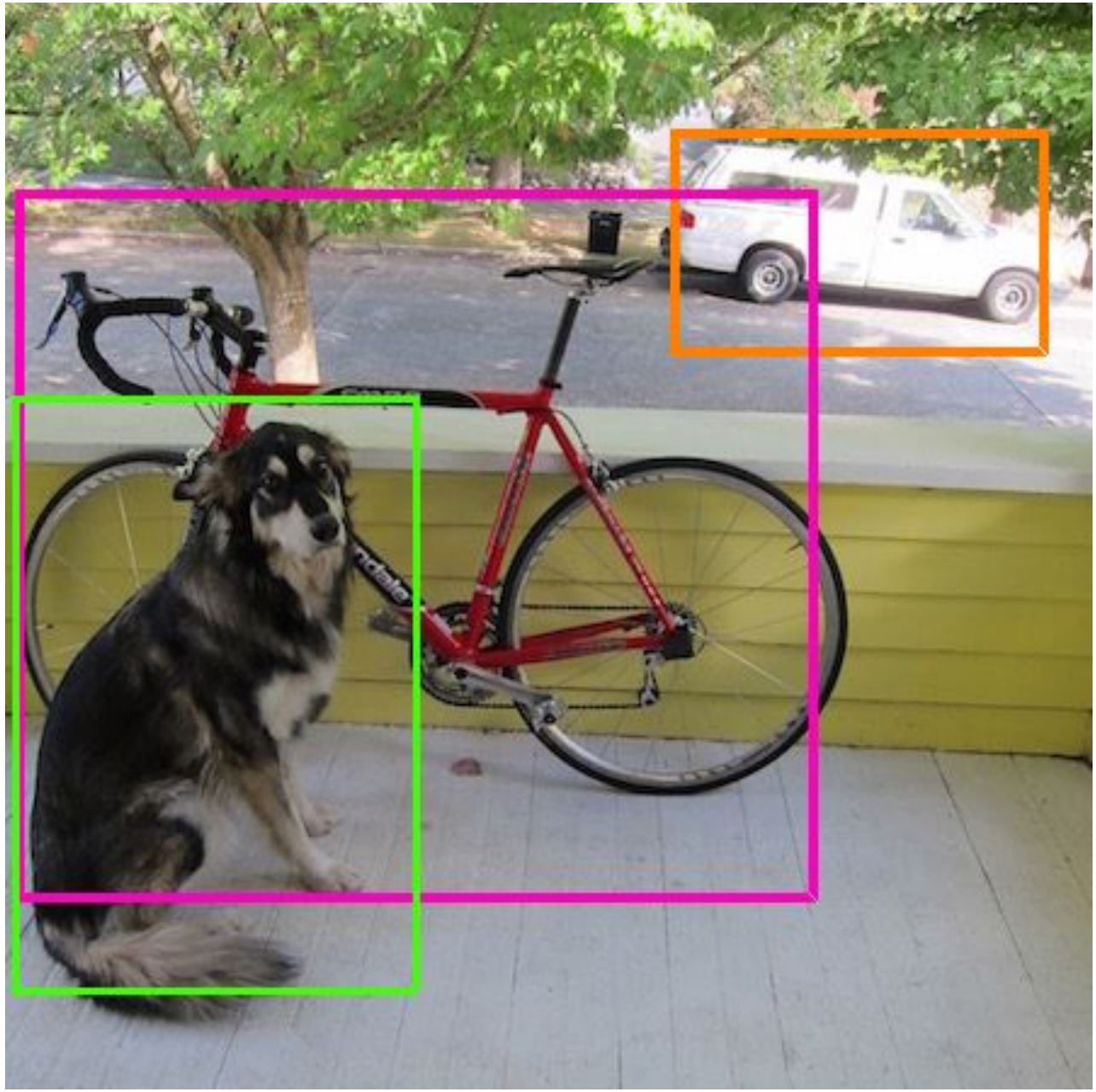
Conditioned on object: P(Car | Object)



Then we combine the box and class predictions.



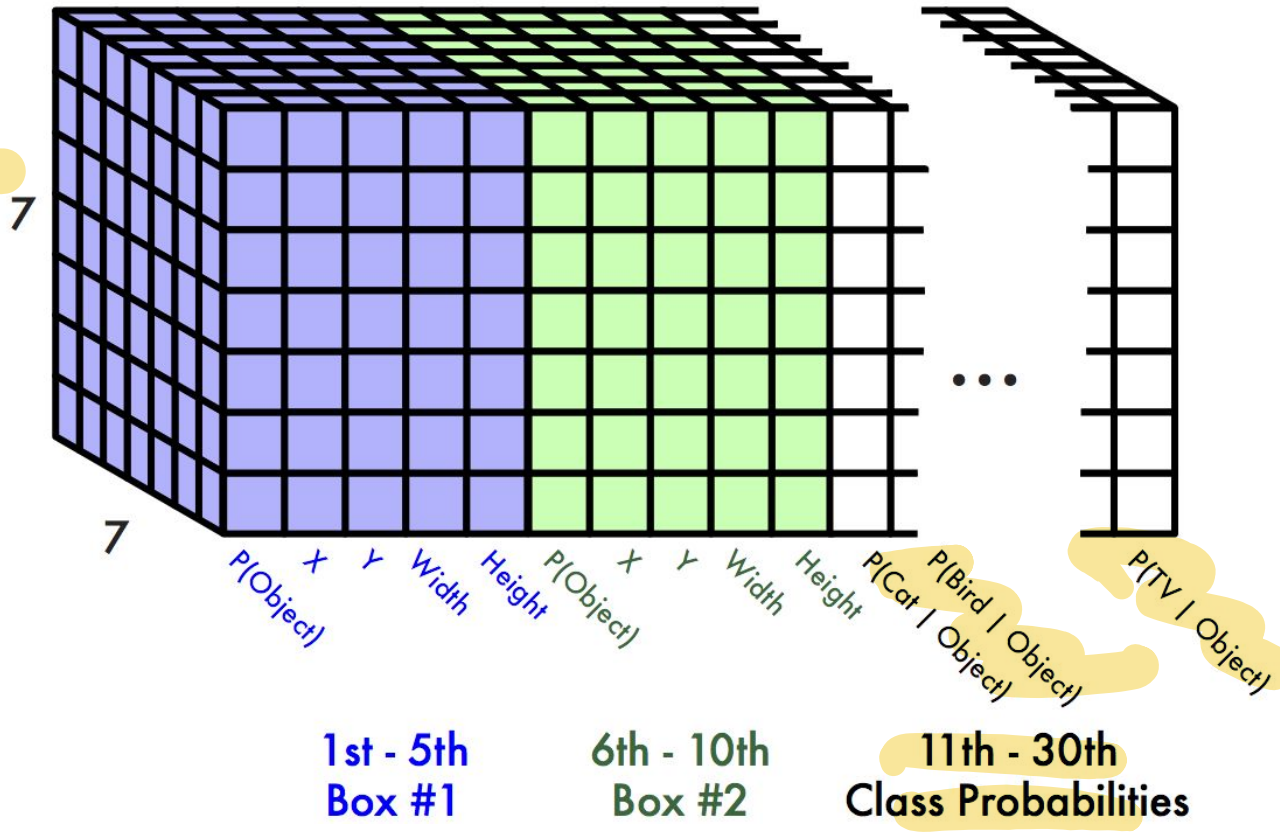
Finally, set a threshold (Non-maximal suppression) to fix multiple detections



# This parameterization fixes the output size

Each cell predicts:

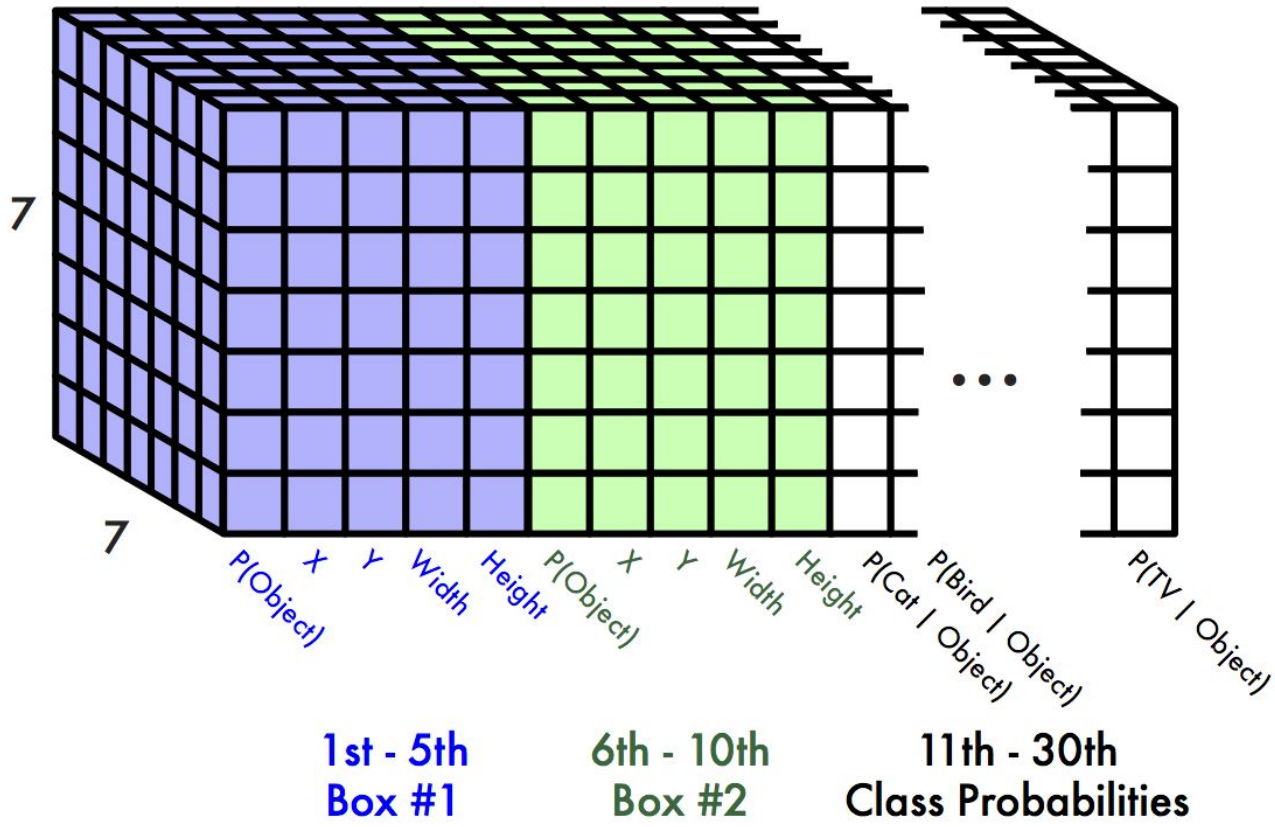
- For each bounding box:
  - 4 coordinates (x, y, w,
  - 1 confidence value
- Some number of class probabilities



# This parameterization fixes the output size

Each cell predicts:

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

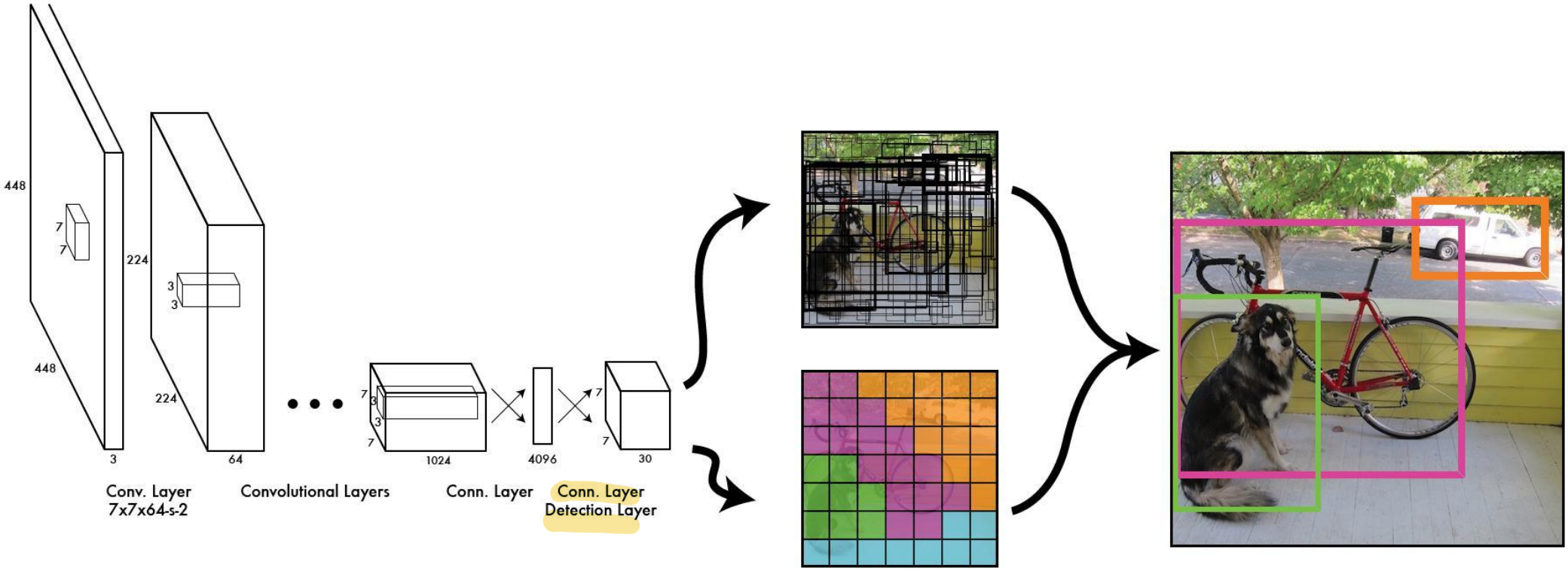



Total Predictions

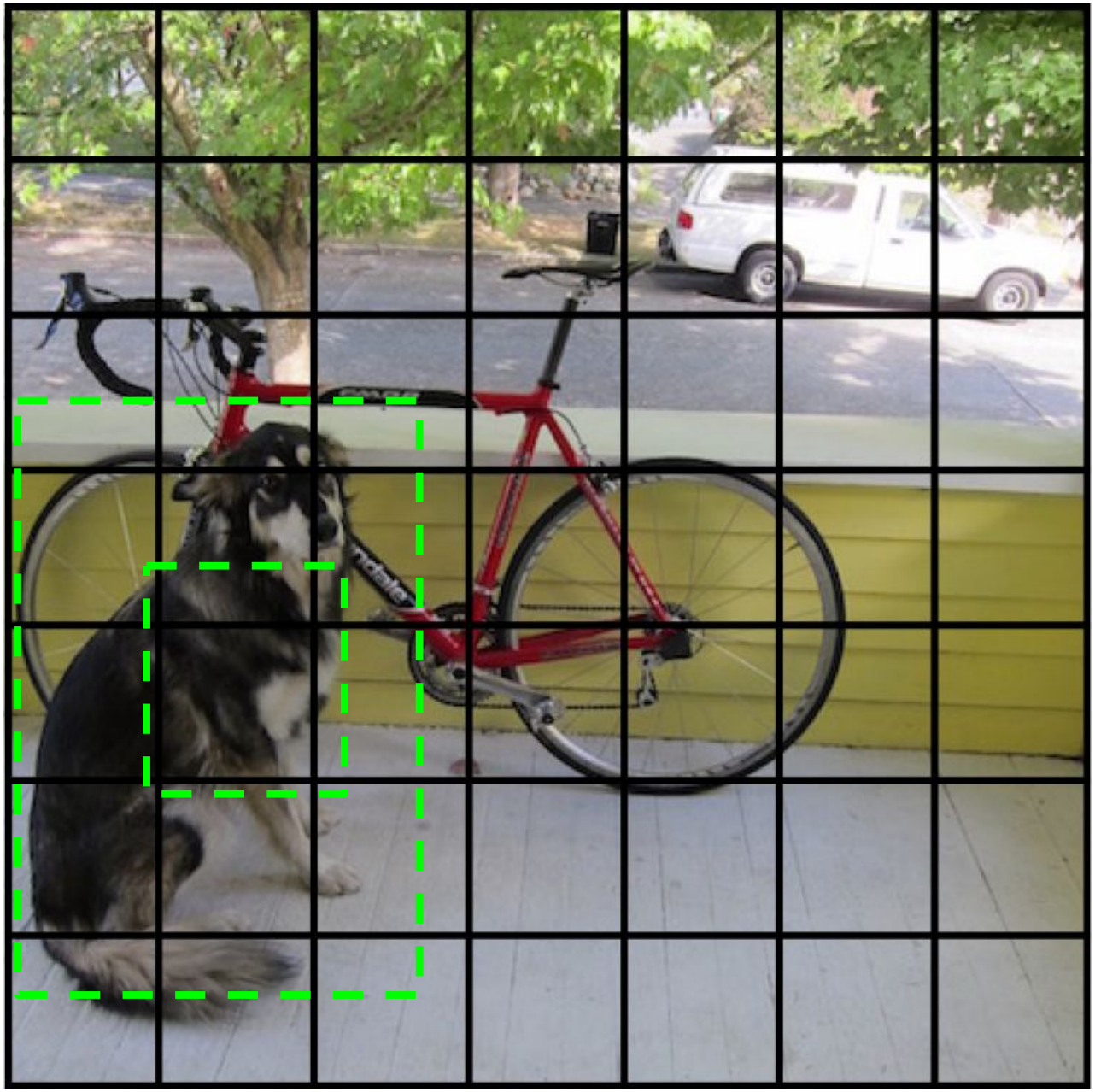
- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

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

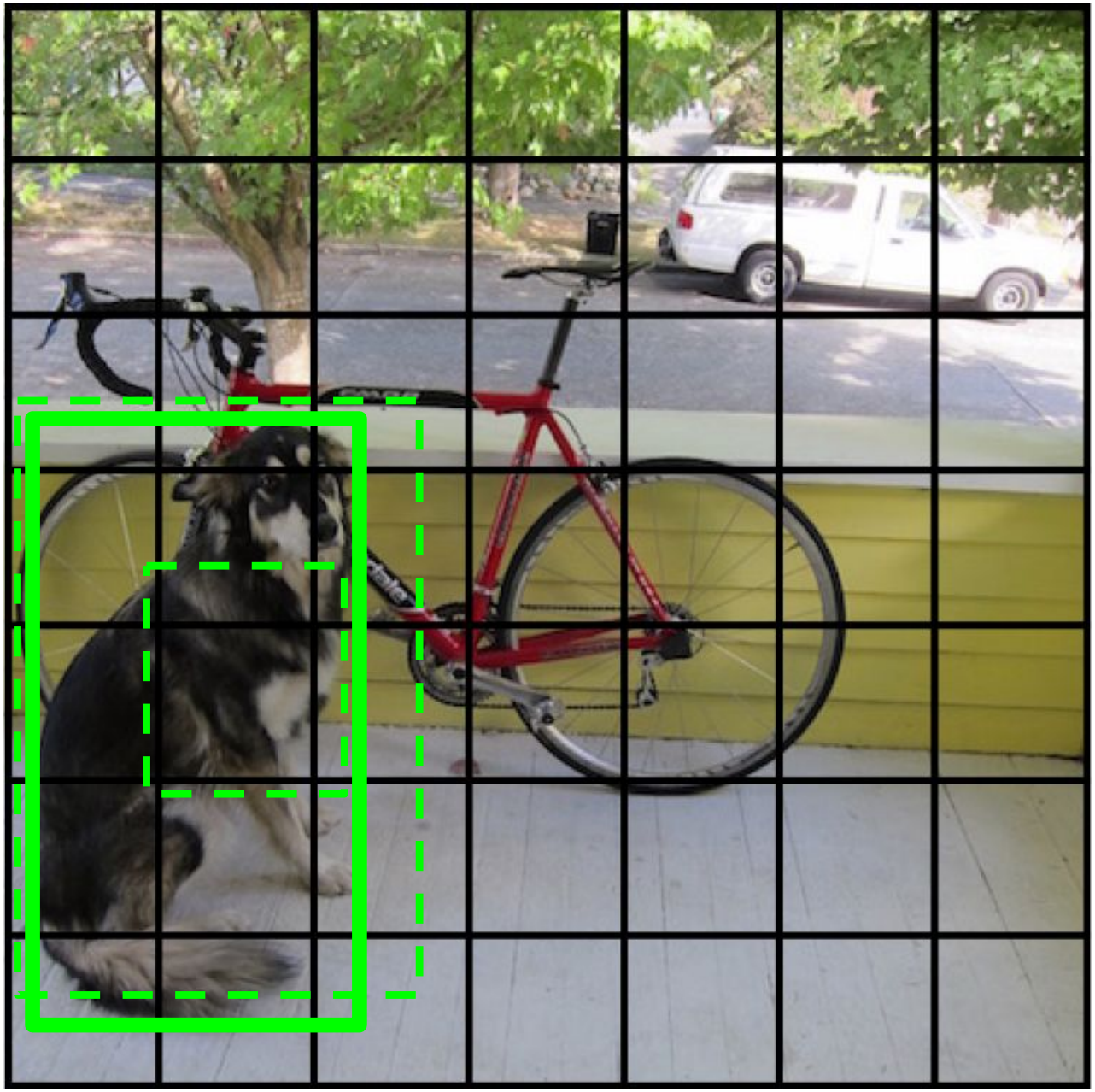
# Thus we can train one neural network to be a whole detection pipeline



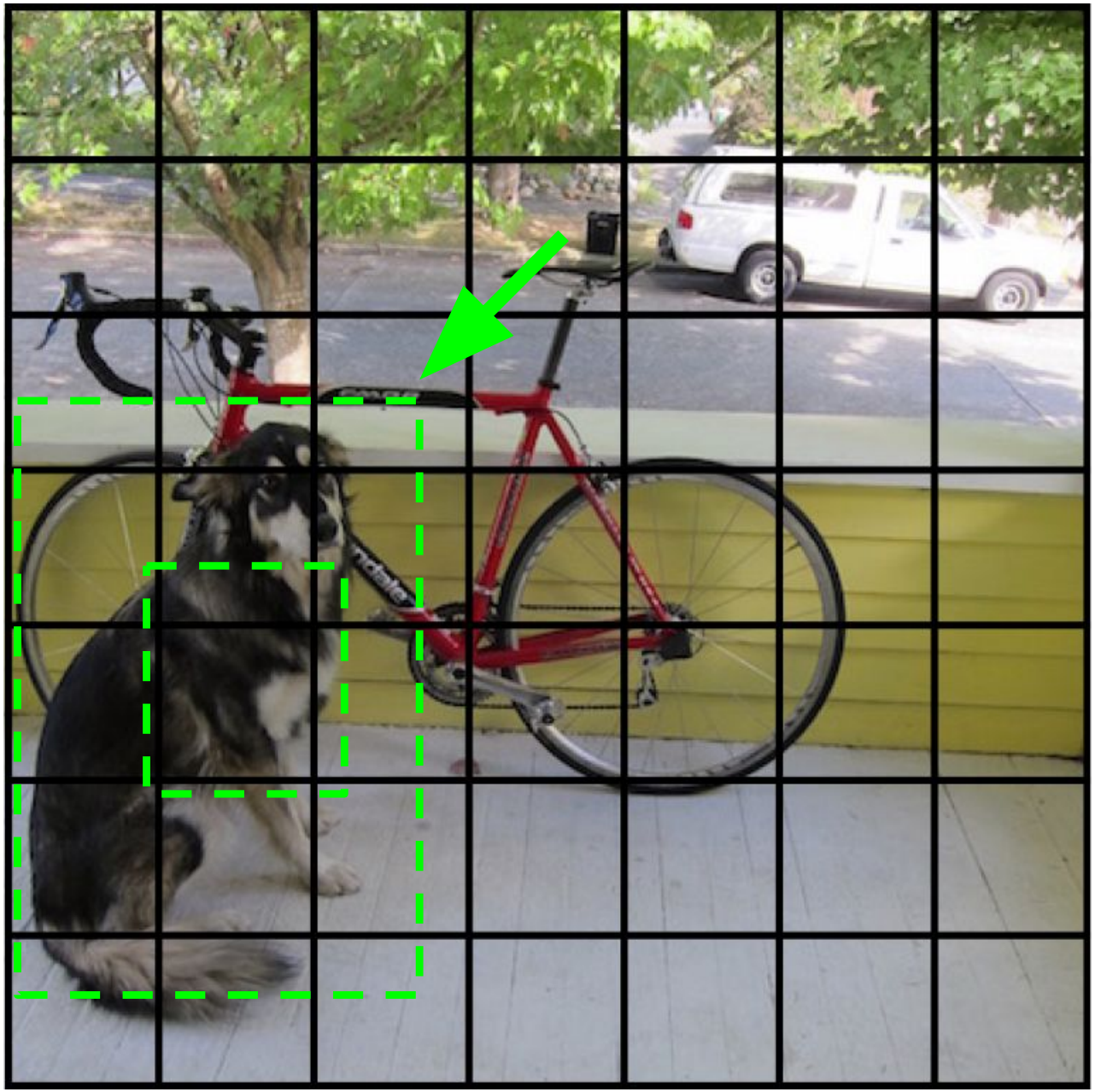
# Look at that cell's predicted boxes



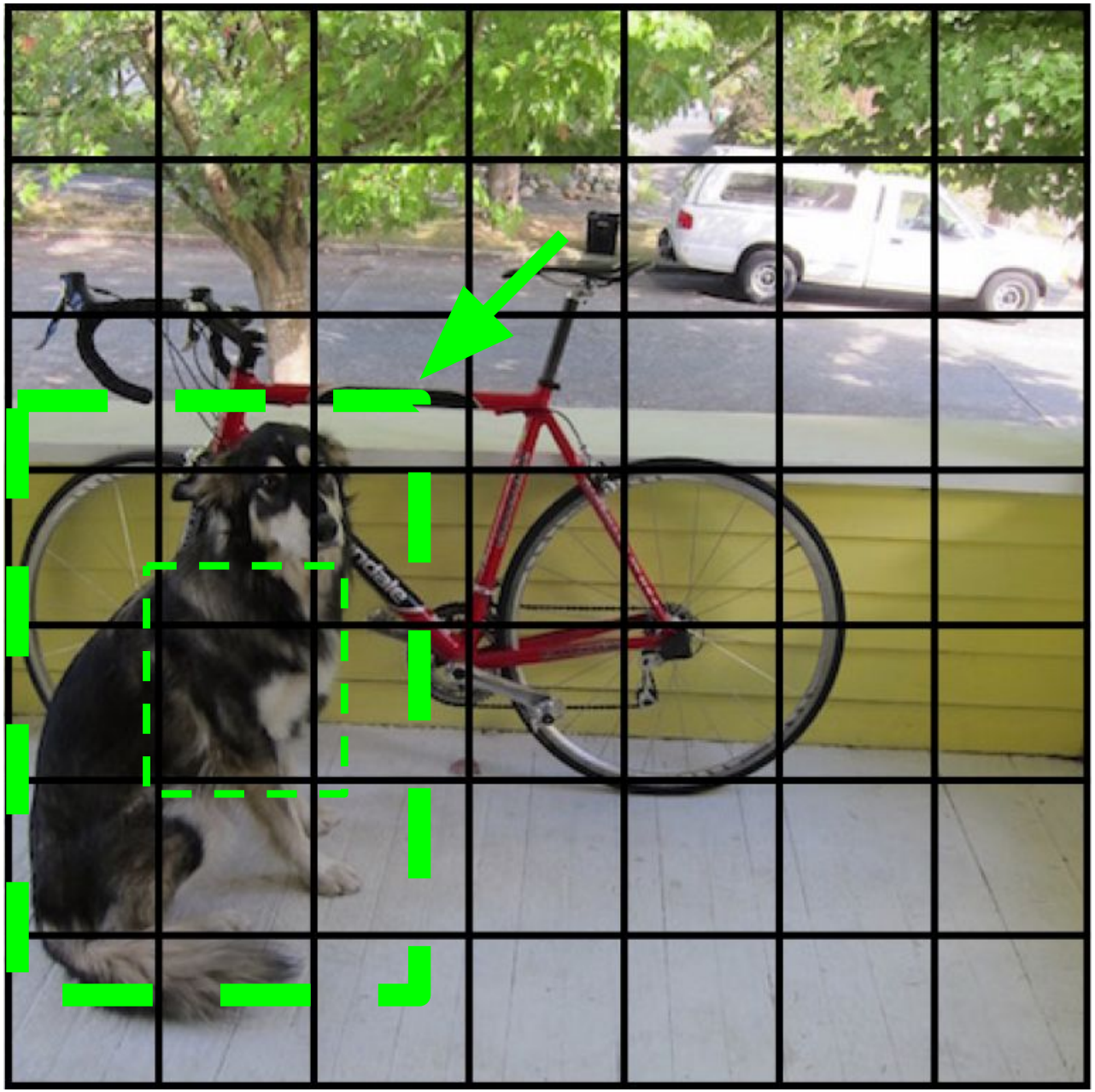
Find the best one, adjust it, increase the confidence



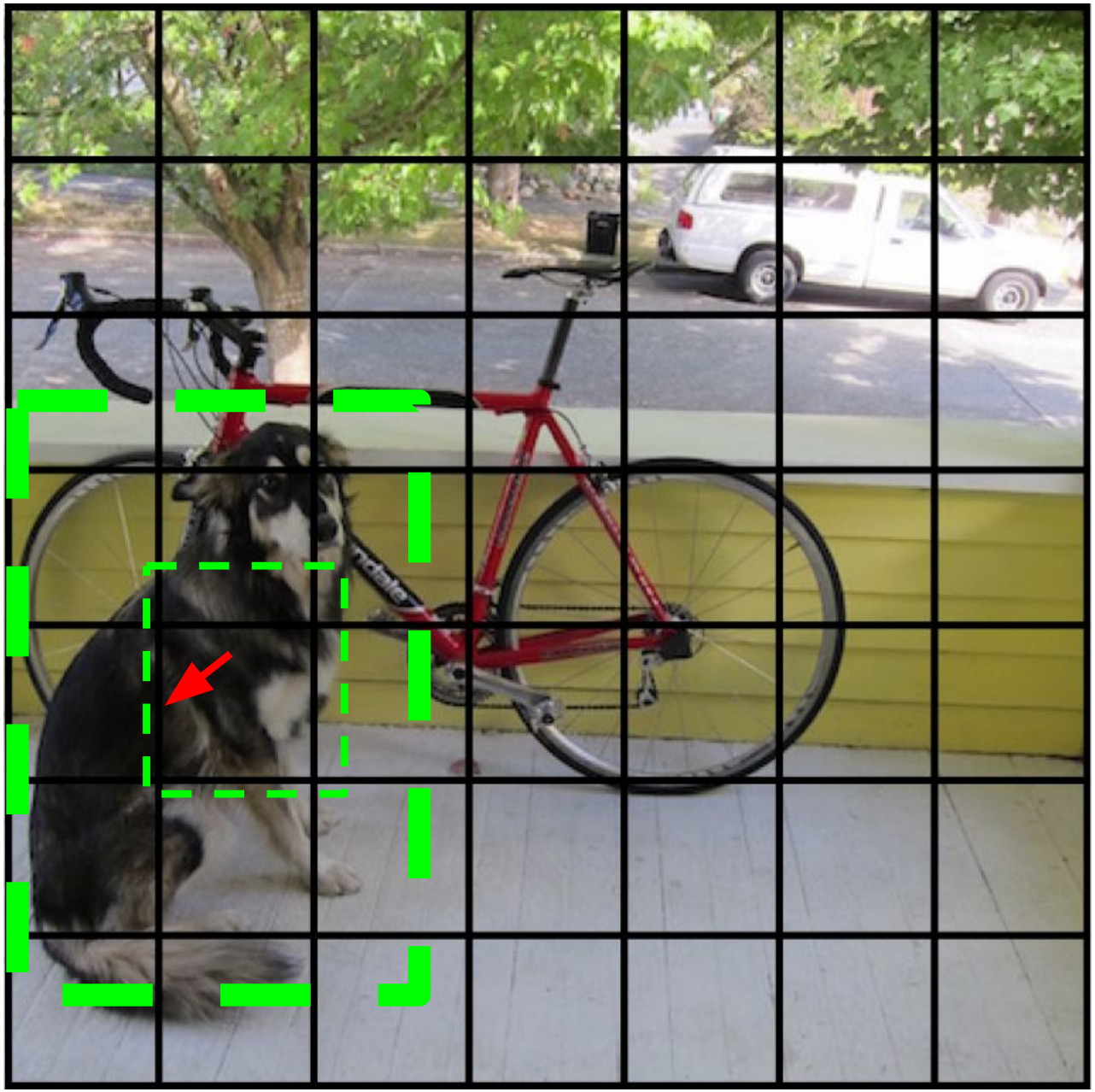
Find the best one, adjust it, increase the confidence



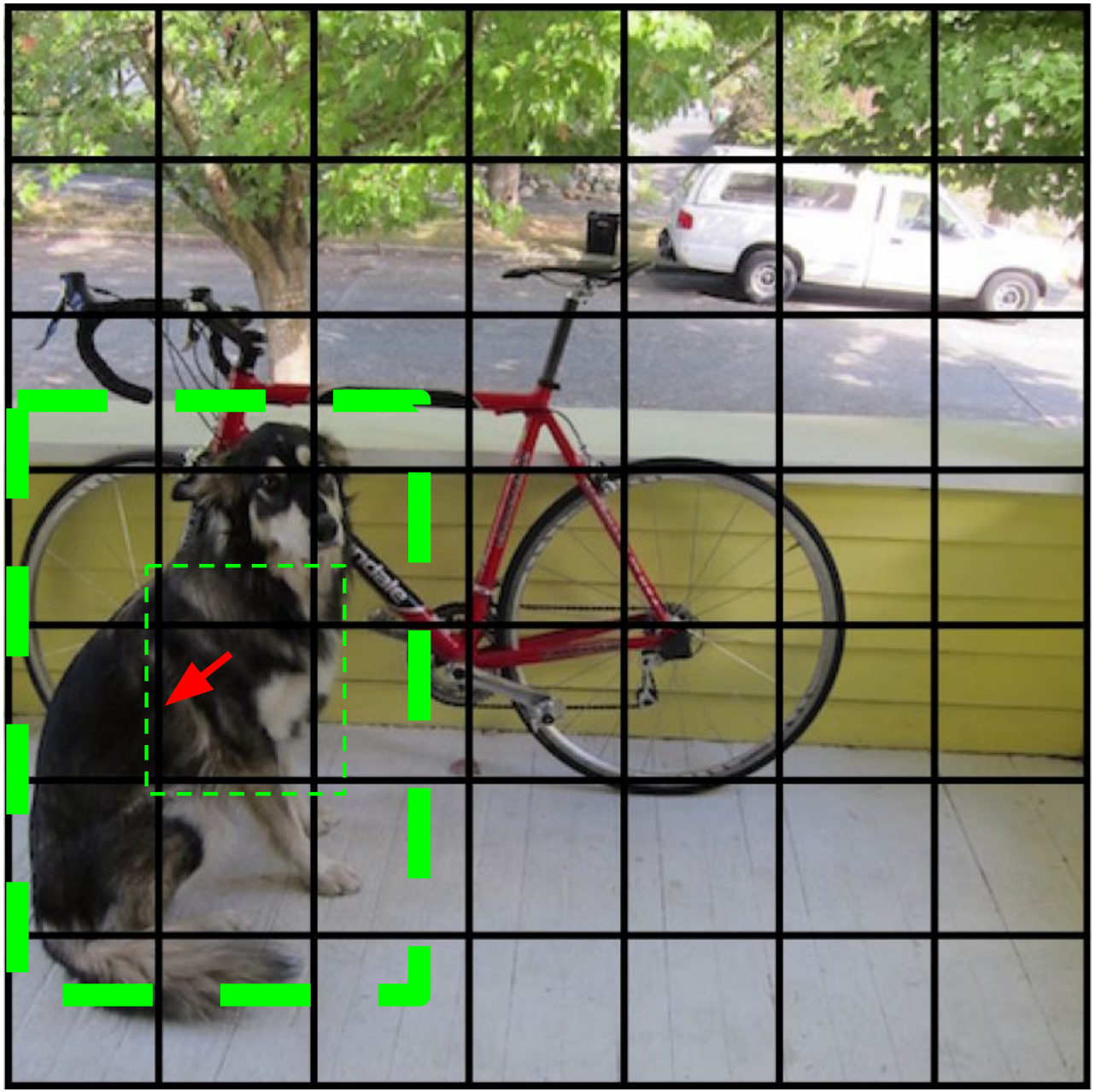
Find the best one, adjust it, increase the confidence



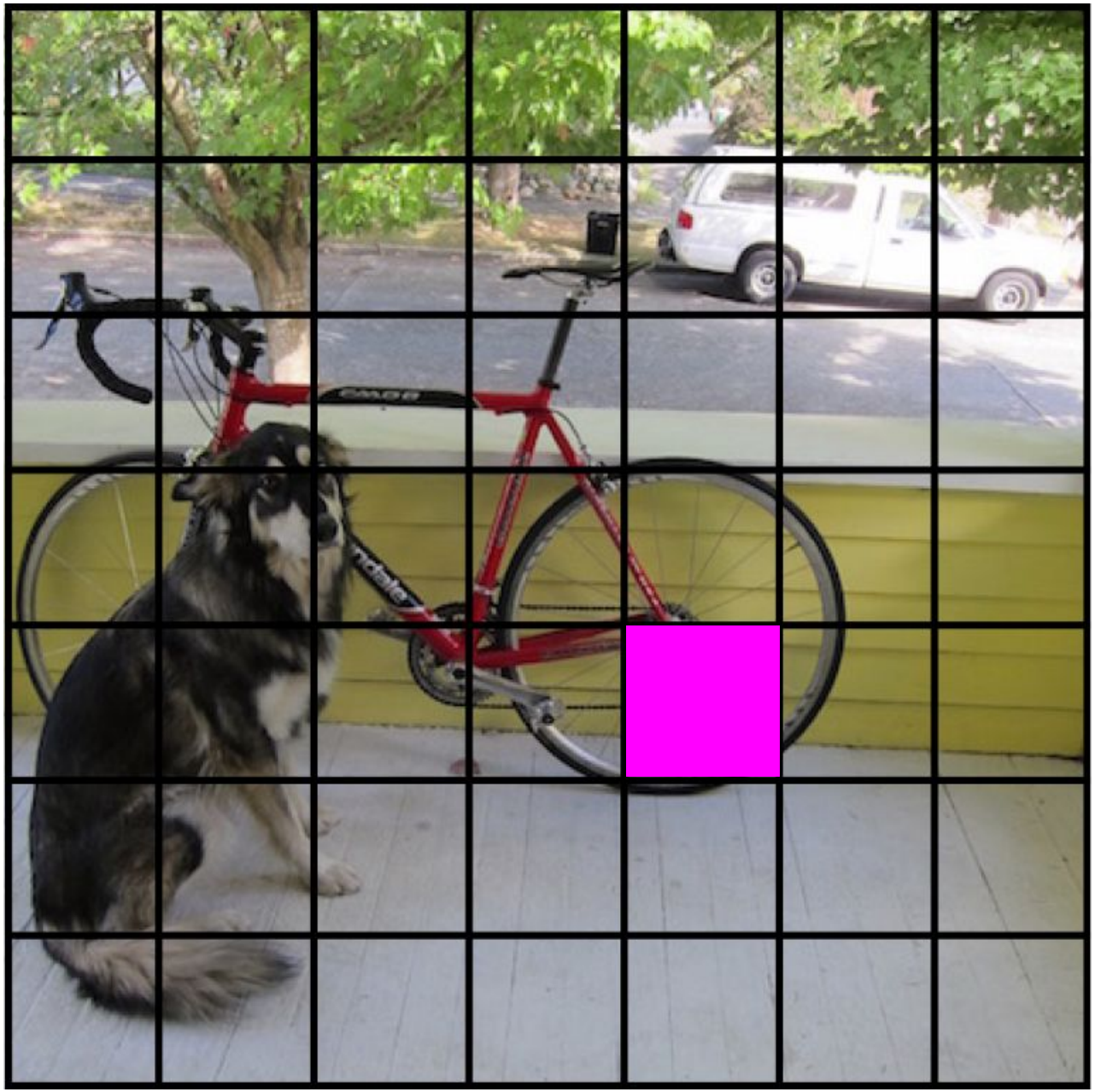
# Decrease the confidence of other boxes



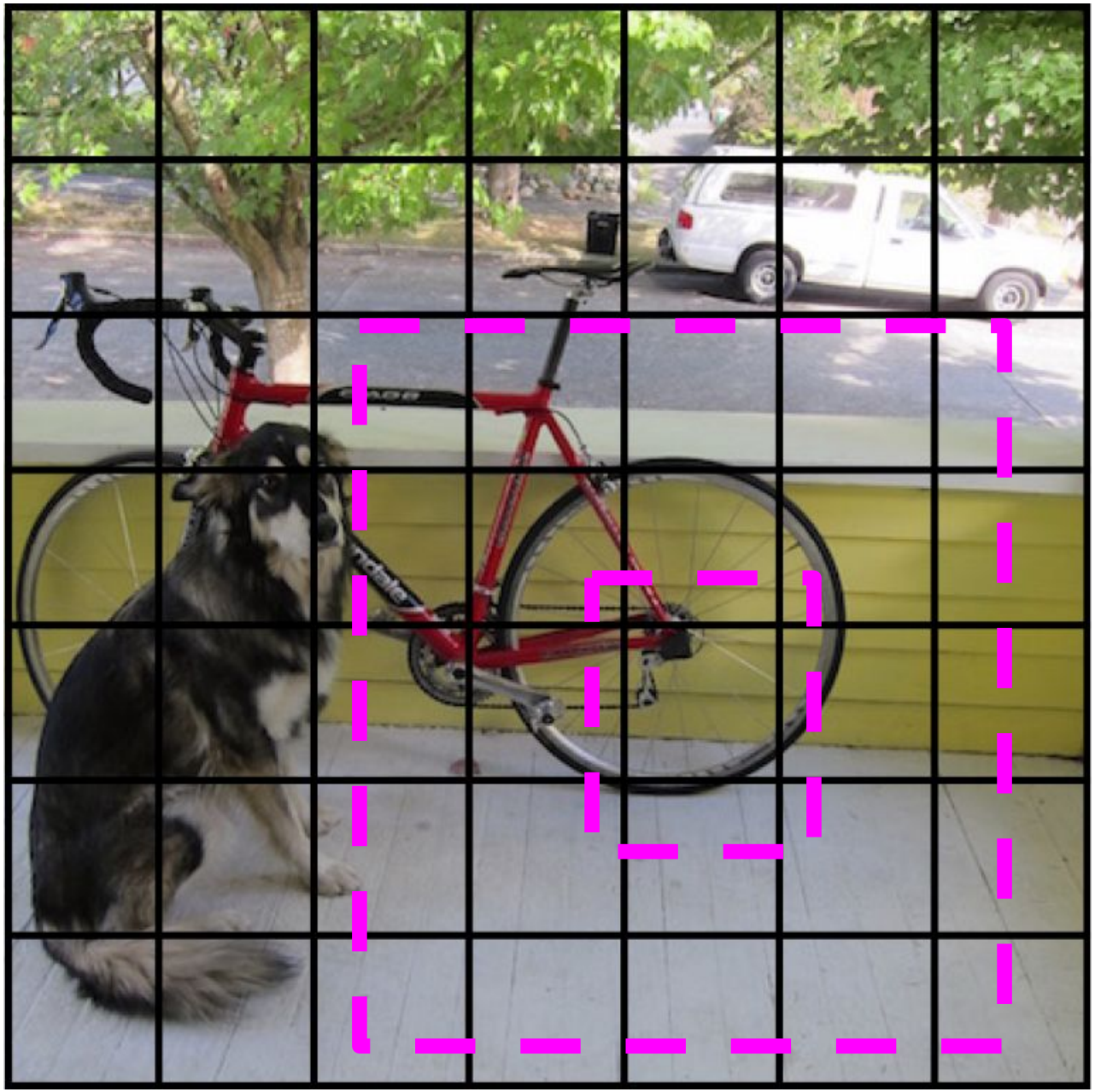
# Decrease the confidence of other boxes



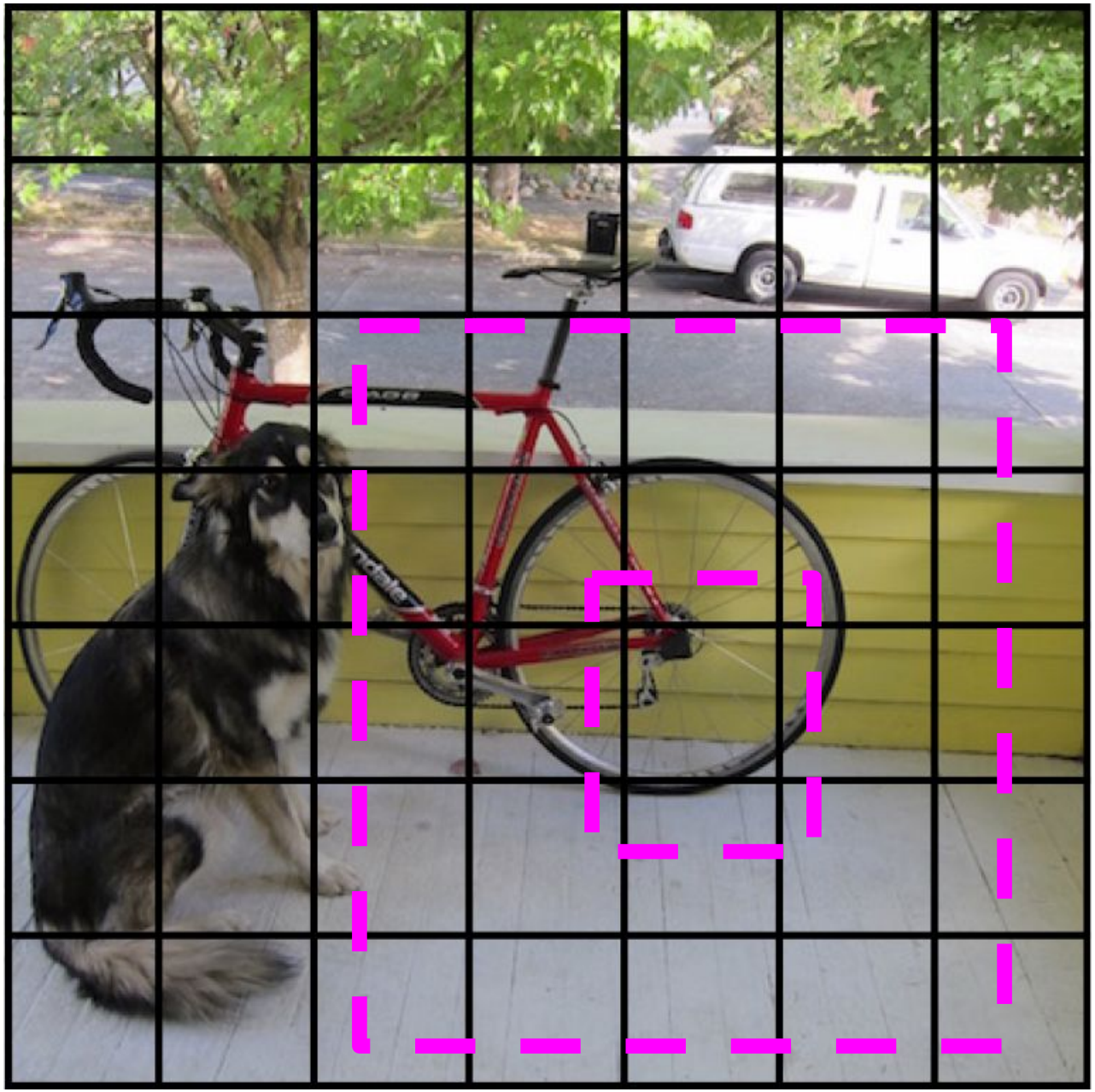
Some cells don’t have any ground truth detections!



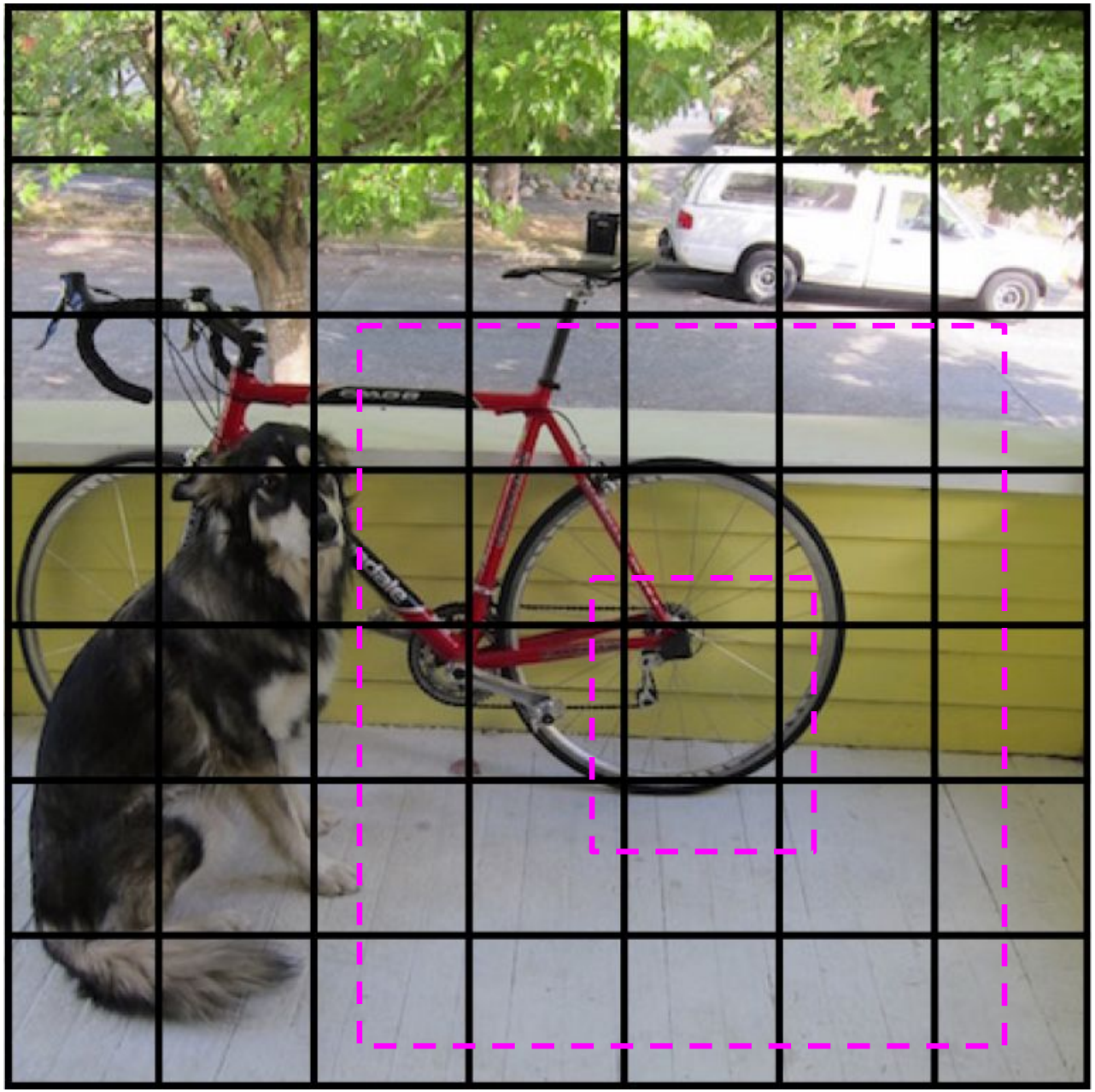
Some cells don’t have any ground truth detections!



# Decrease the confidence of these boxes

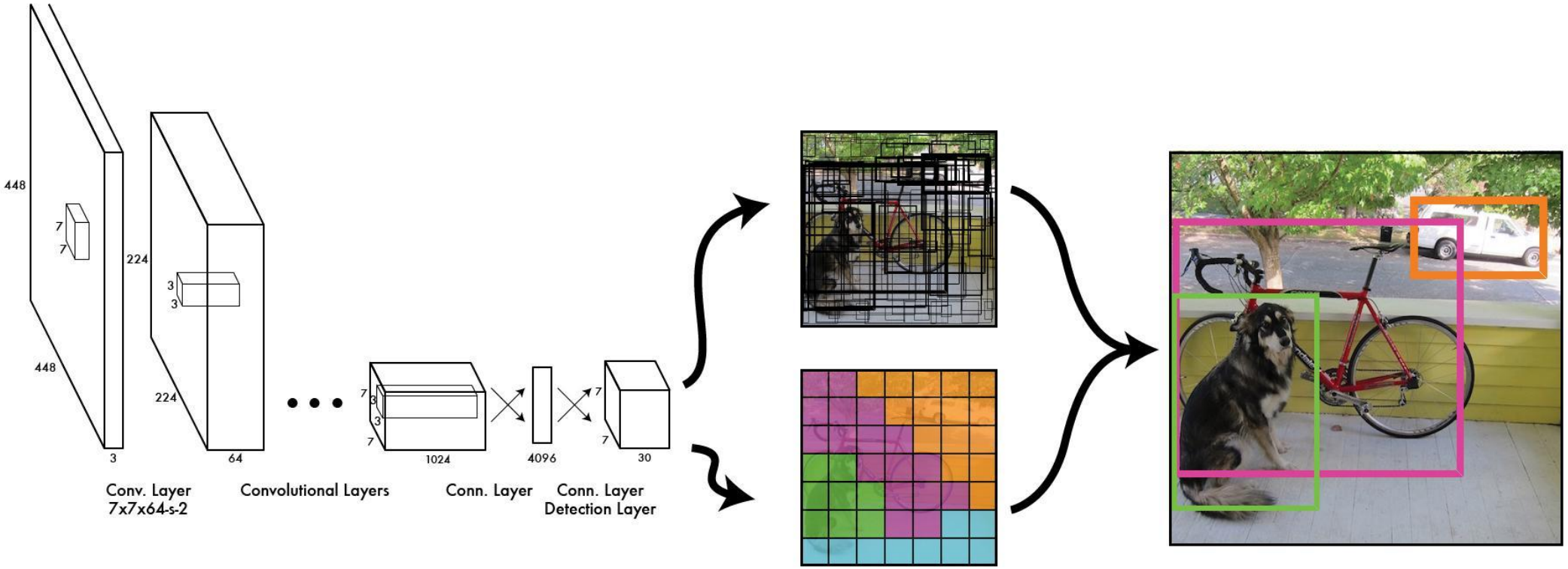


# Decrease the confidence of these boxes

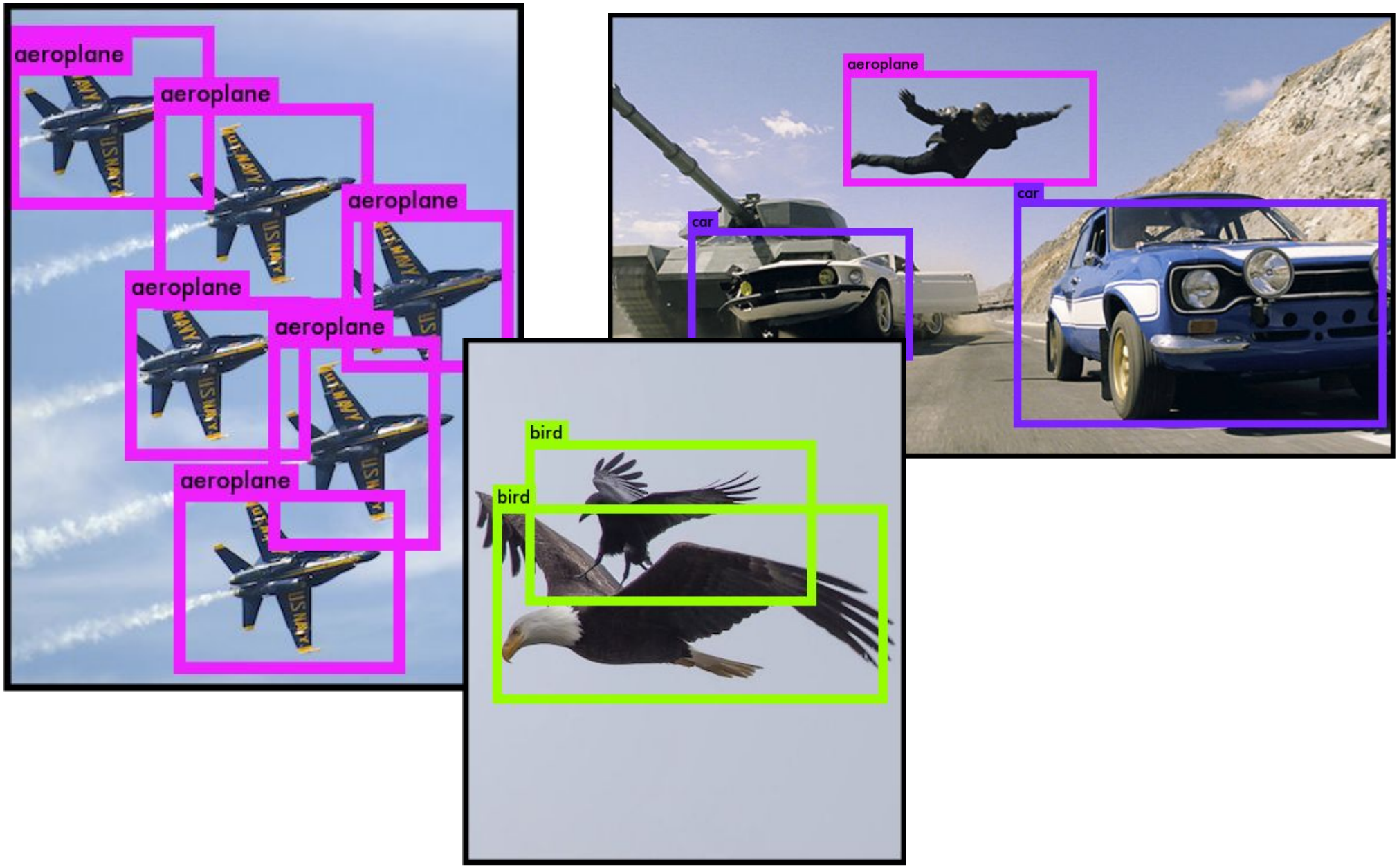


# We train with standard tricks:

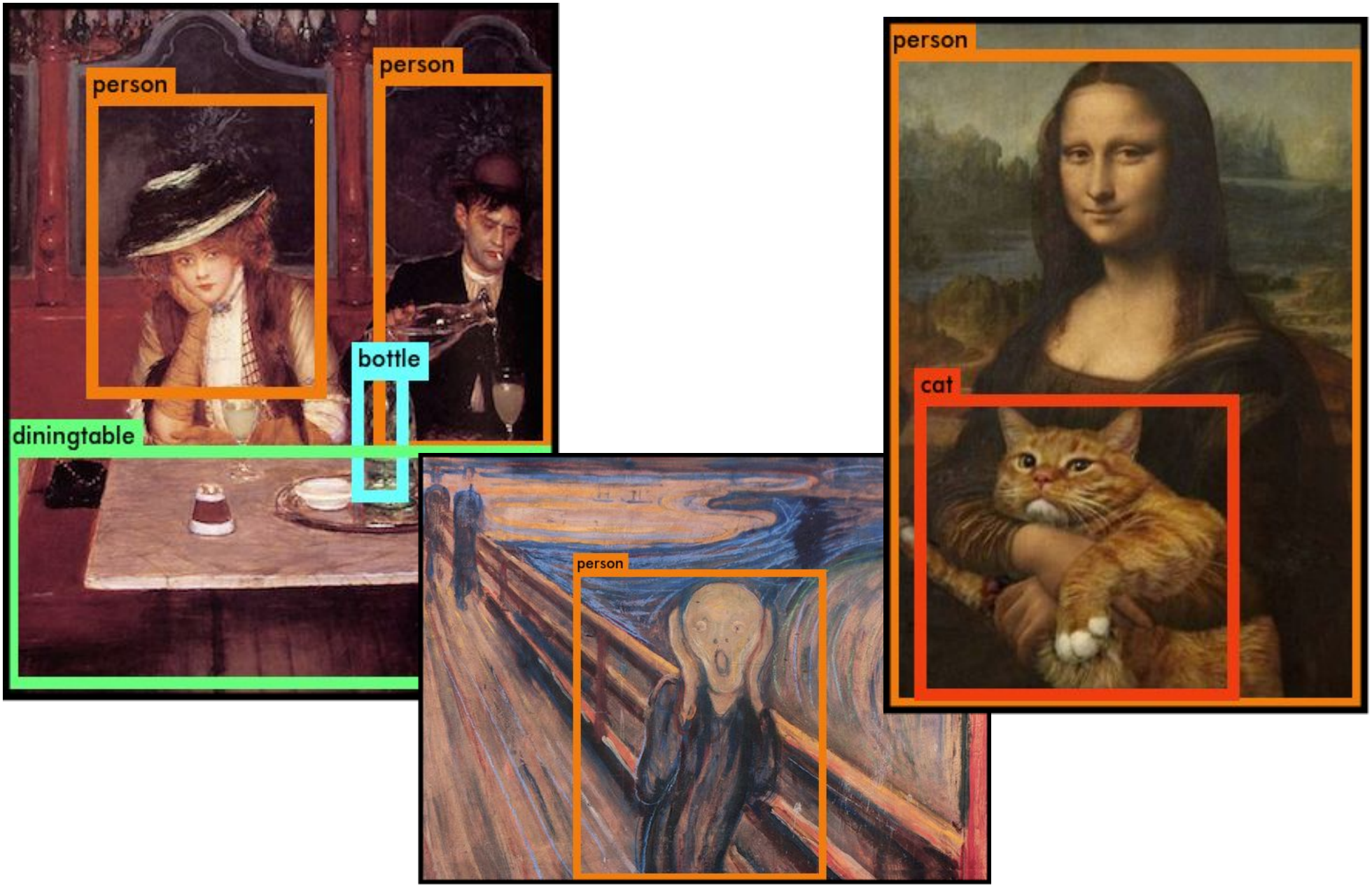
- Pretraining on Imagenet
- Extensive data augmentation
- For details, see the paper



# YOLO works across a variety of natural images

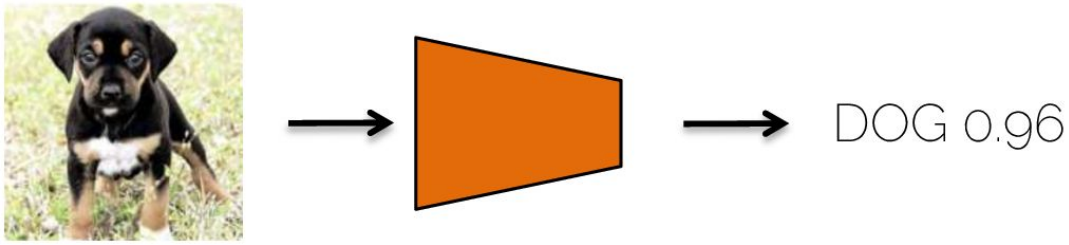


# It also generalizes well to new domains (like art)



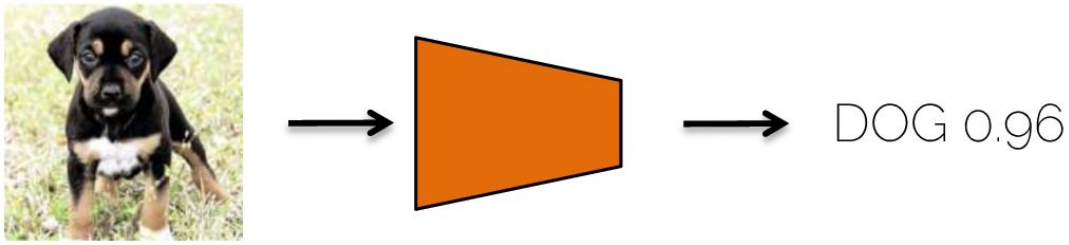
# Visualizing importance

# The occlusion experiment



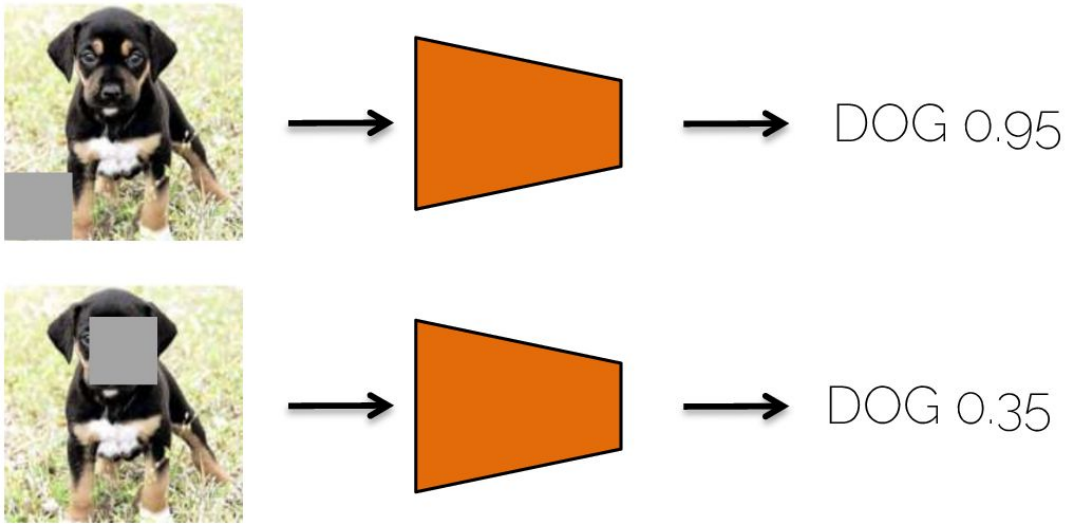
# The occlusion experiment

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



# The occlusion experiment

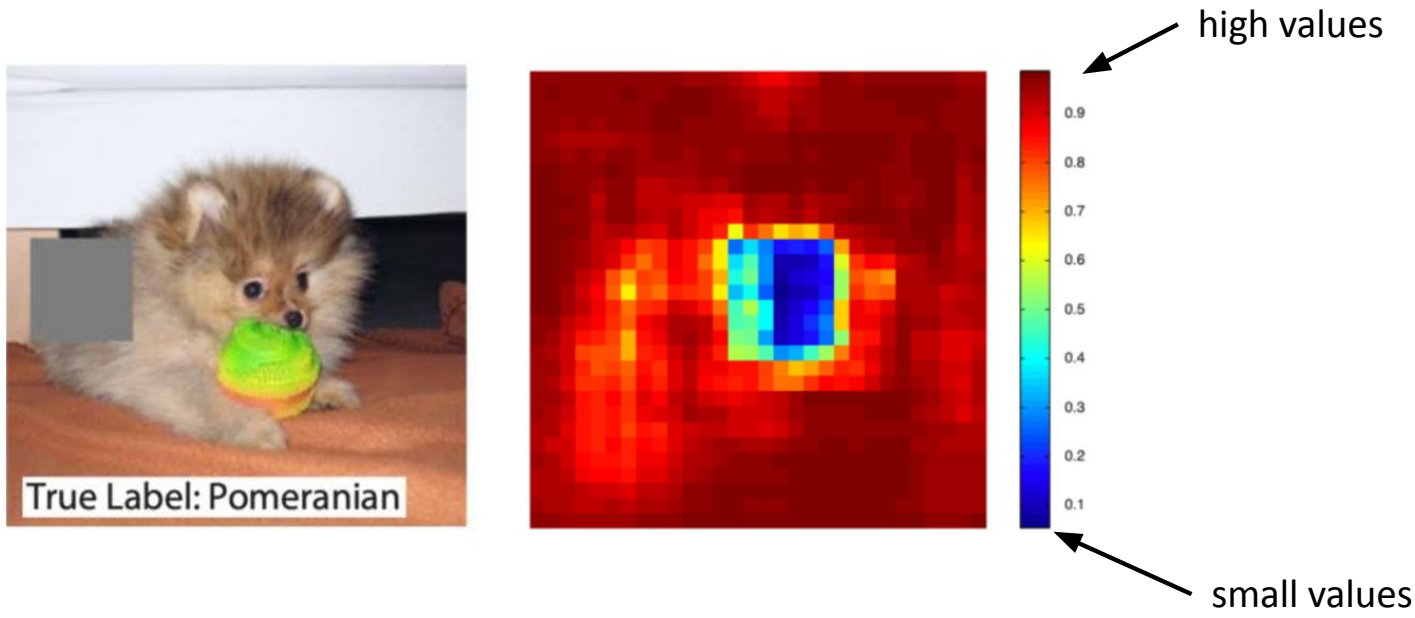
Block different parts of the image and see how the classification score changes



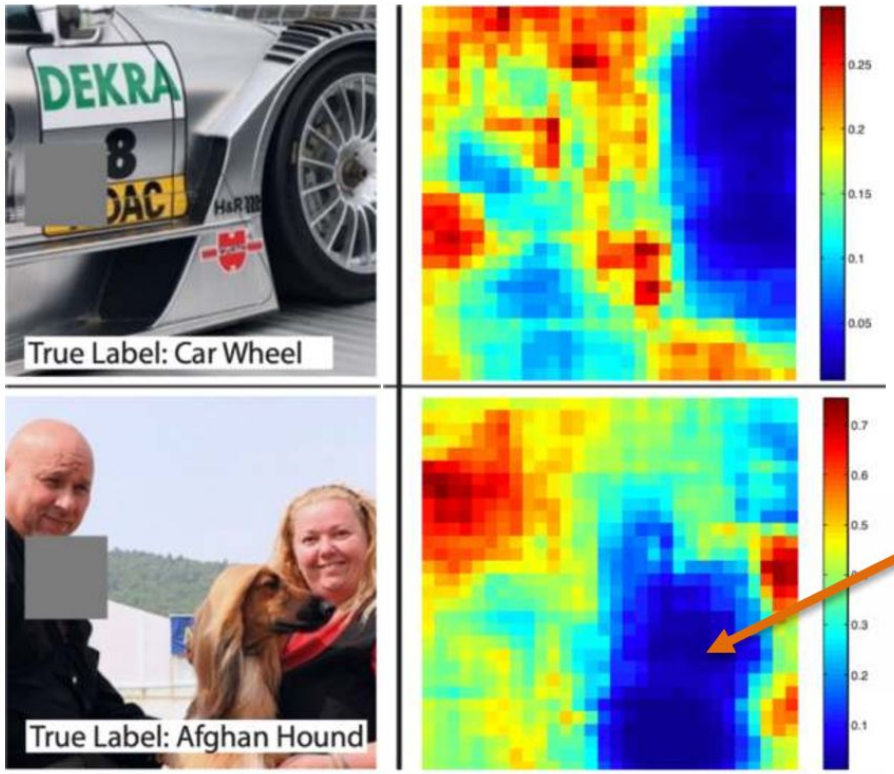

The face of the dog is more important for correct classification

# The occlusion experiment

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



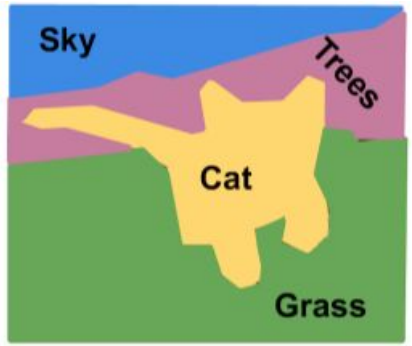
# The occlusion experiment




Most important pixels for classification

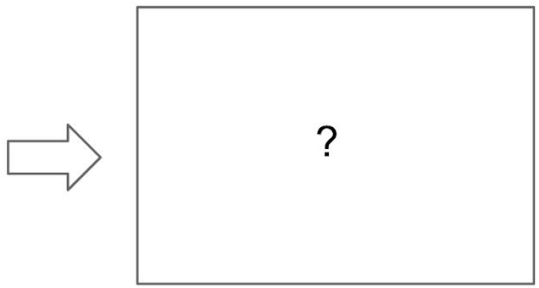
# Semantic Segmentation

# Semantic Segmentation



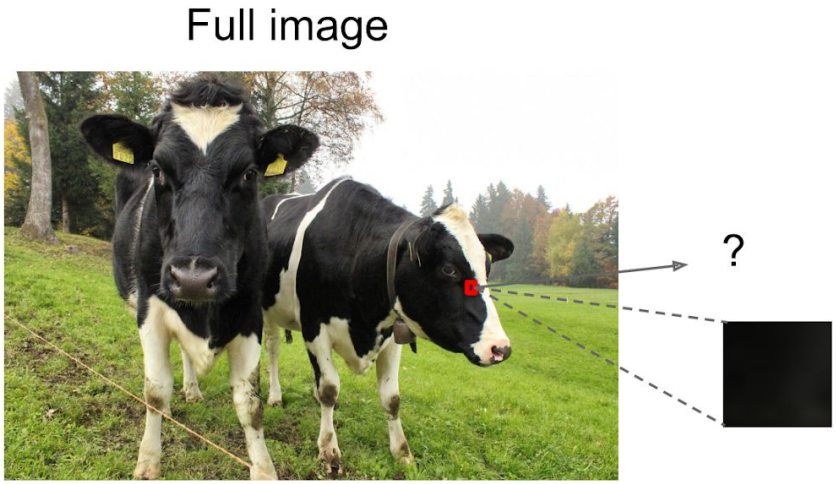

**GRASS**, **CAT**,
**TREE**, **SKY**, **...**

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



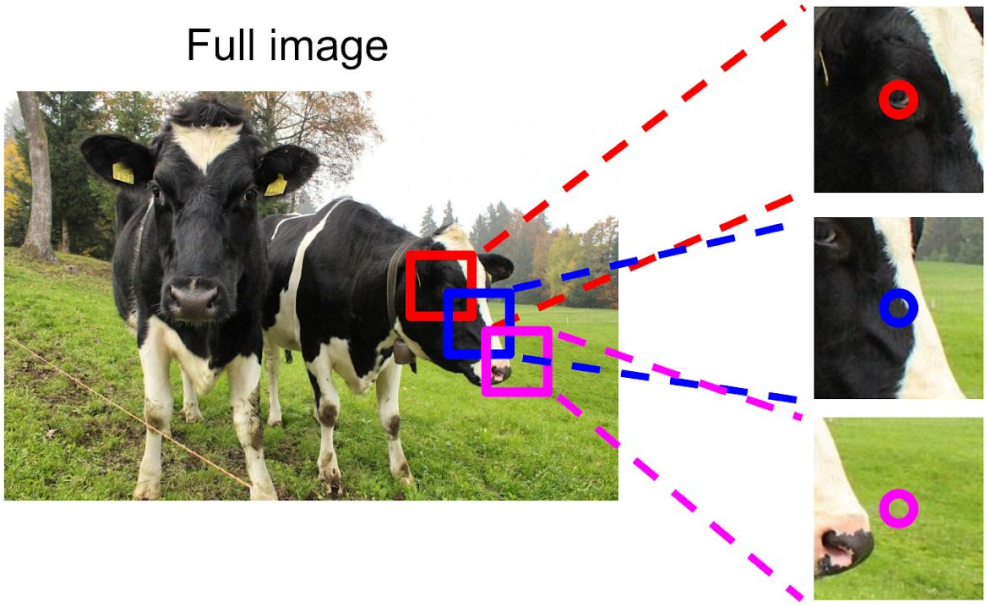
# Semantic Segmentation Idea: Sliding Window

Full image


Impossible to classify without context

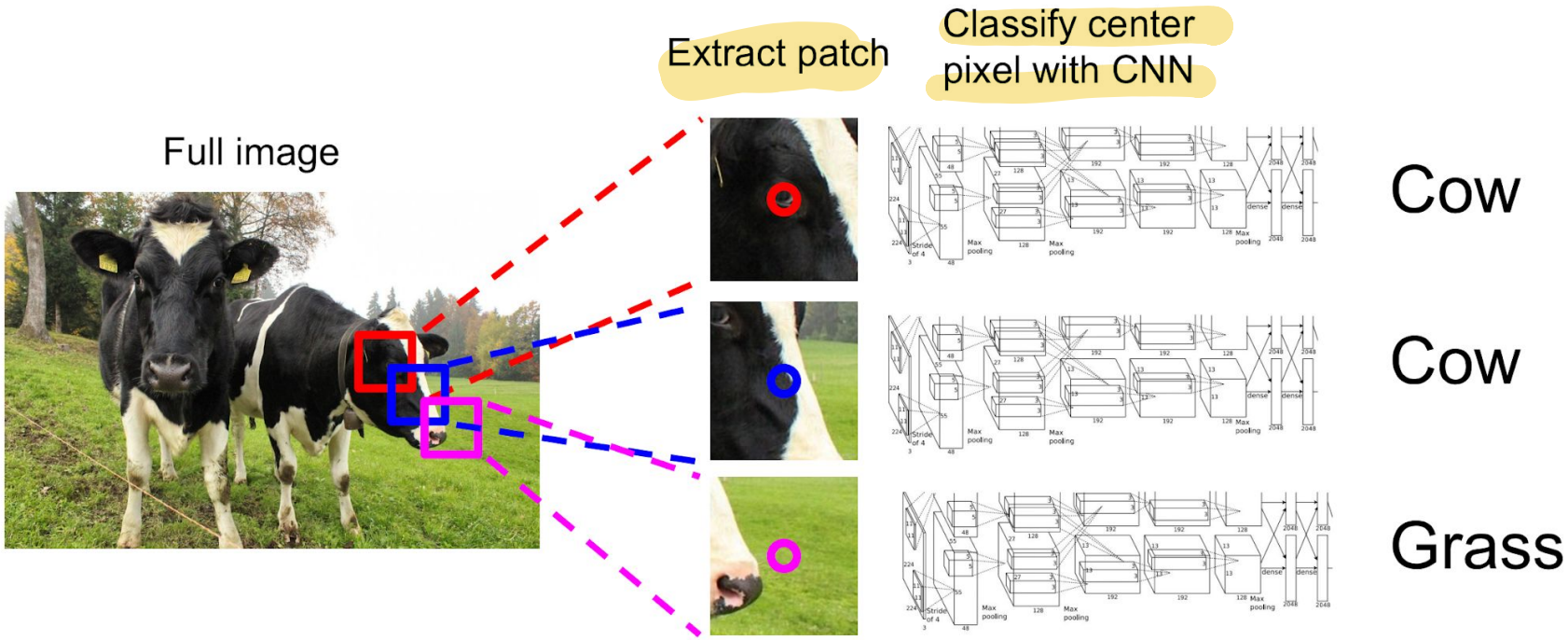
Q: how do we include context?

# Semantic Segmentation Idea: Sliding Window



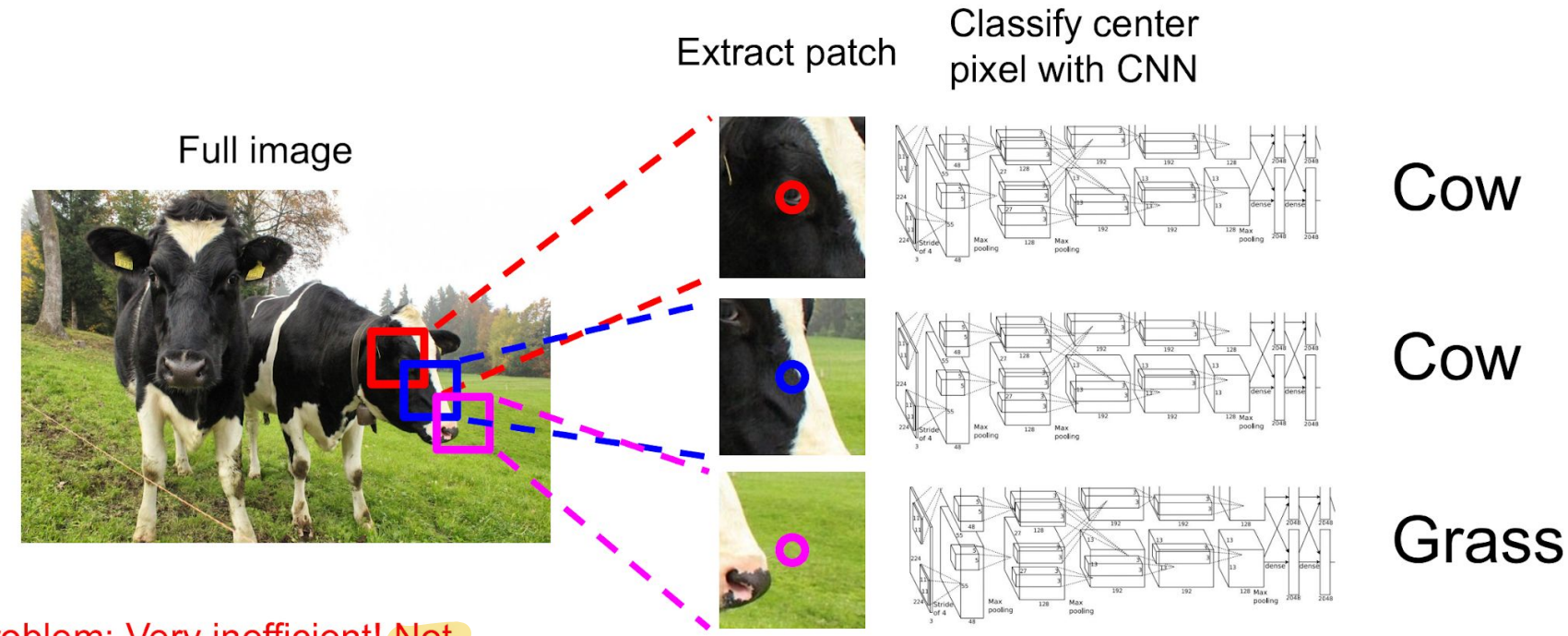

Q: how do we model this?

# 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

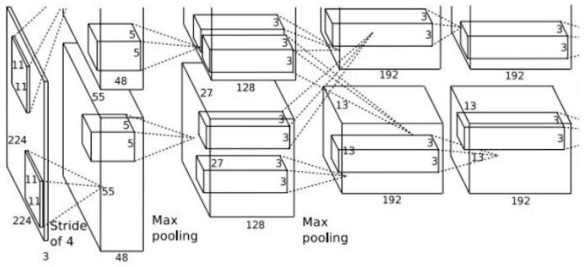



Problem: Very inefficient! Not reusing shared features between overlapping patches

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

Patch wali approach impractical coz not using shared features among patches.

# Semantic Segmentation idea: CNN

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

Input:
3 x H x W

Conv → Conv → Conv → Conv → argmax

Convolutions:
D x H x W

Scores:
C x H x W

Predictions:
H x W

# Semantic Segmentation idea: CNN

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

Input:
3 x H x W

Conv

Conv

Conv

Convolutions:
D x H x W

Conv

Scores:
C x H x W

argmax

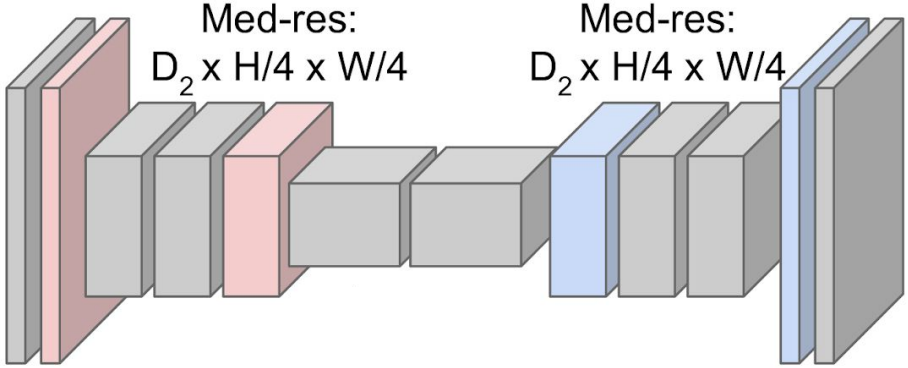
Predictions:
H x W

Problem: convolutions at original image resolution will be very expensive ...

# Semantic Segmentation idea: CNN

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

Input:
3 x H x W




High-res:
$D_1$ x H/2 x W/2

High-res:
$D_1$ x H/2 x W/2

C x H x W

Predictions:
H x W

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

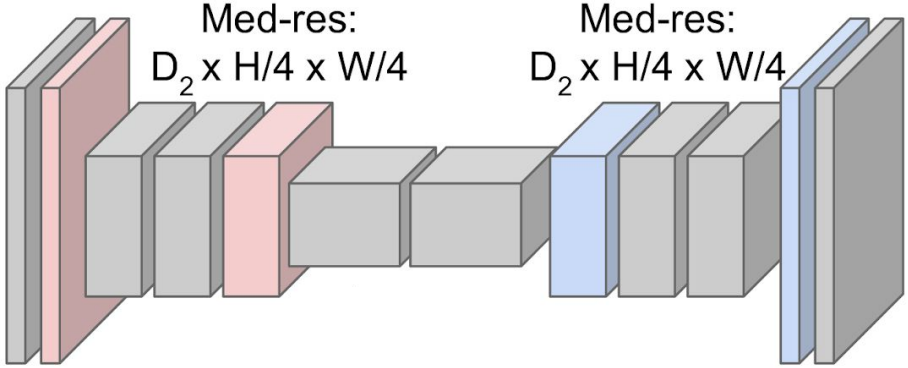
# Semantic Segmentation idea: CNN

**Downsampling**:
Pooling, strided convolution

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

**Upsampling**:
???

Input:
3 x H x W




High-res:
$D_1$ x H/2 x W/2

High-res:
$D_1$ x H/2 x W/2

C x H x W

Predictions:
H x W


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

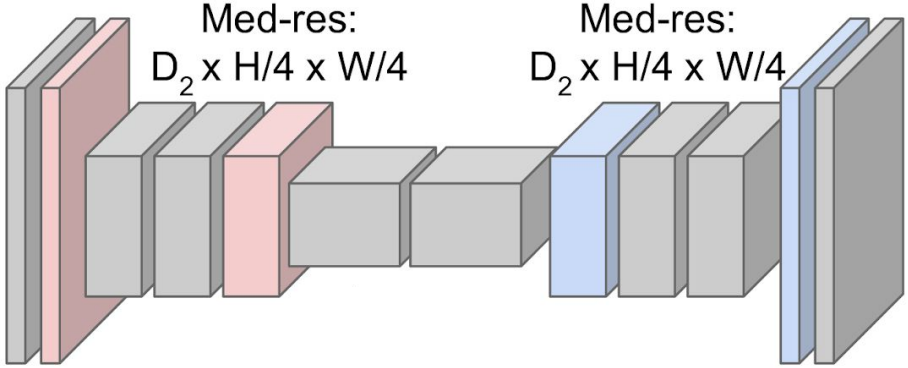
# Semantic Segmentation idea: CNN

**Downsampling**:
Pooling, strided convolution

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

**Upsampling**:
Unpooling or transposed convolution

Input:
3 x H x W




High-res:
$D_1$ x H/2 x W/2

High-res:
$D_1$ x H/2 x W/2

Predictions:
H x 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