# Automatic Object Detection
# ----YOLO & SSD

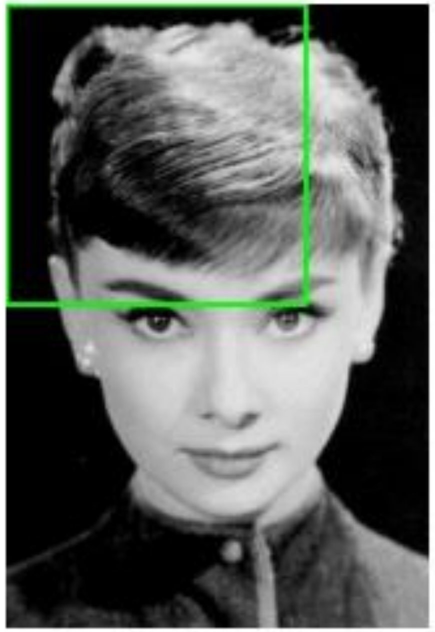

Jianping Fan
Dept of Computer Science
UNC-Charlotte


**Course Website:**
**http://webpages.uncc.edu/jfan/itcs5152.html**

# Before Deep Learning

Sliding windows.

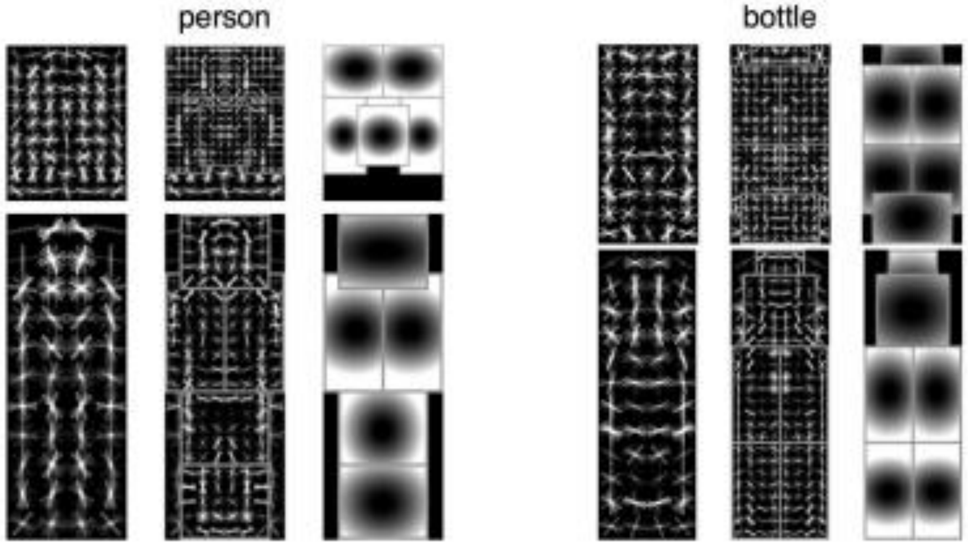
- Score every subwindow.



http://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/

Deformable part models (DPM)

- Uses HOG features
- Very fast




https://cs.brown.edu/~pff/papers/lsvm-pami.pdf

# Techniques for Key Components

## Selective Search

http://www.huppelen.nl/publications/selectiveSearchDraft.pdf

Uijlings, J. R., van de Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition

## (a) Two-stage approaches

## (b) One-stage approaches

## Hard Negative Mining

Imbalance between positive and negative examples.

Use negative examples with higher confidence score.

## Non Maximum Suppression

If output boxes overlap, only consider the most confident.

## Bounding Box Regression

Regression used to find bounding box parameters

Applied in one of two ways

- Bounding box refinement
- Complete object detection

# You Only Look Once:
# Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†

University of Washington*, Allen Institute for AI†, Facebook AI Research¶

http://pjreddie.com/yolo/



## Abstract

*We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.*

*Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.*


**Figure 1: The YOLO Detection System.** Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to $448 \times 448$, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

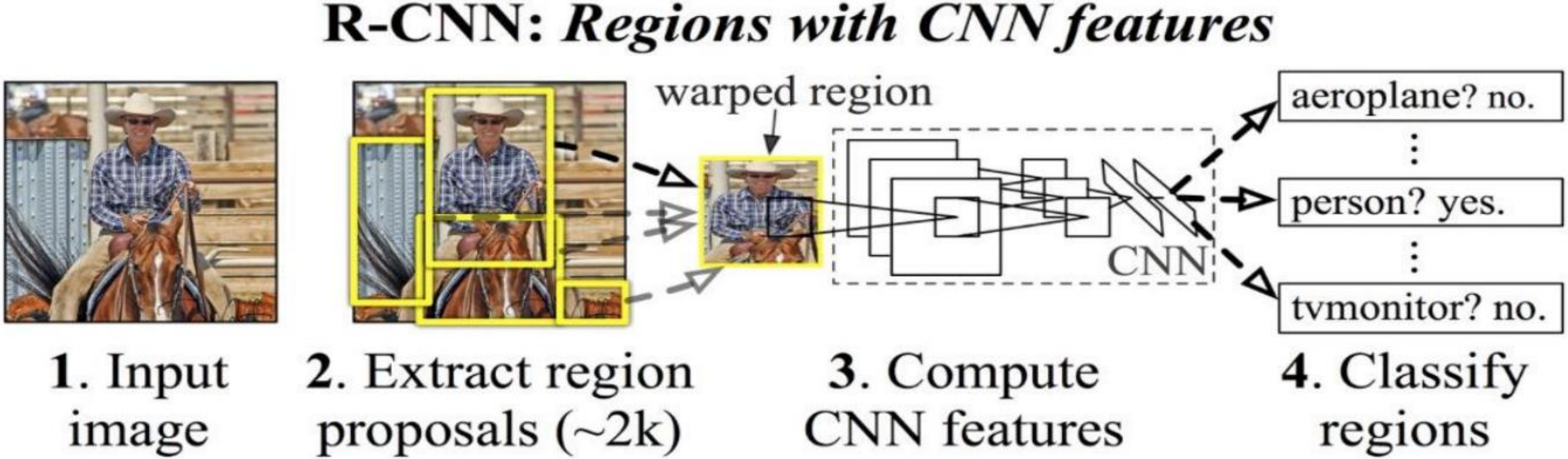
methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only

# Shortcoming:

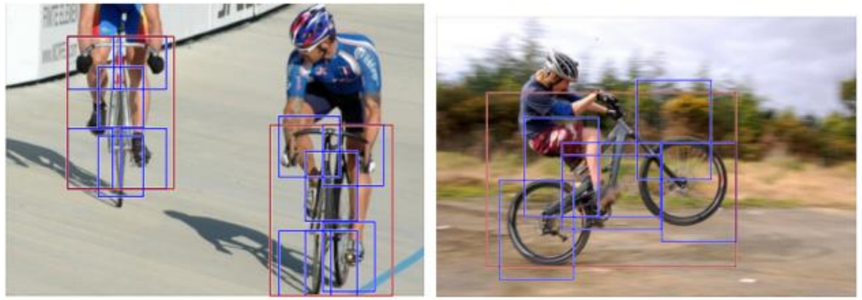
1. Slow, impossible for real-time detection

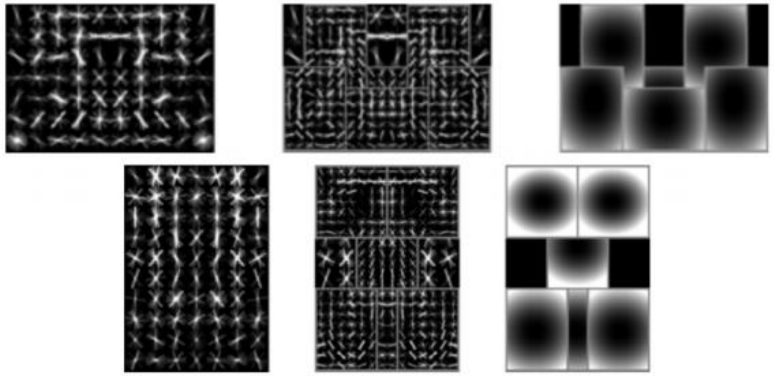
2. Hard to optimize

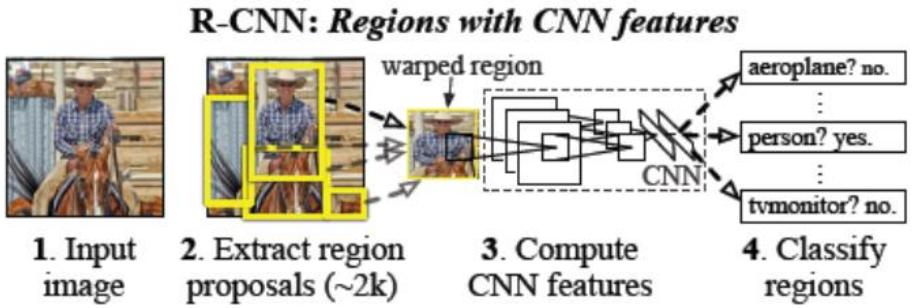


# Object Detection as Regression Problem

- **Previous:** Repurpose classifiers to perform detection
  - <u>**Deformable Parts Models (DPM)**</u>
    - Sliding window
  - <u>**R-CNN based methods**</u>
    - 1) generate potential bounding boxes.
    - 2) run classifiers on these proposed boxes
    - 3) post-processing (refinement, elimination, rescore)







# What's new? Regression

## YOLO Features :

1. Extremely fast (45 frames per second)

2. Reason Globally on the Entire Image

3. Learn Generalizable Representations

[web.cs.ucdavis.edu › ~yjlee › teaching › YOLO,](web.cs.ucdavis.edu) Liyang Zhong Quan Zou

# Object Detection as Regression Problem

- **YOLO:** Single Regression Problem
  - Image → bounding box coordinate and class probability.
  - Extremely Fast
  - Global reasoning
  - Generalizable representation

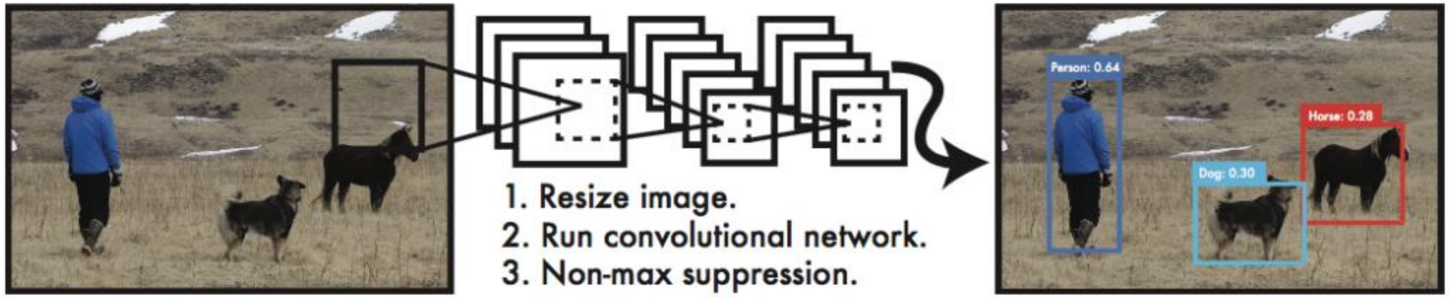



**Figure 1: The YOLO Detection System.** Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to $448 \times 448$, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

# YOLO
# You Only Look Once: Unified, Real-Time Object Detection

? Tool?-- A single neural network, unified architecture

? Framework?– Darknet

? Technology background?--related methods are slow , not real-time and devoid of generalization ability

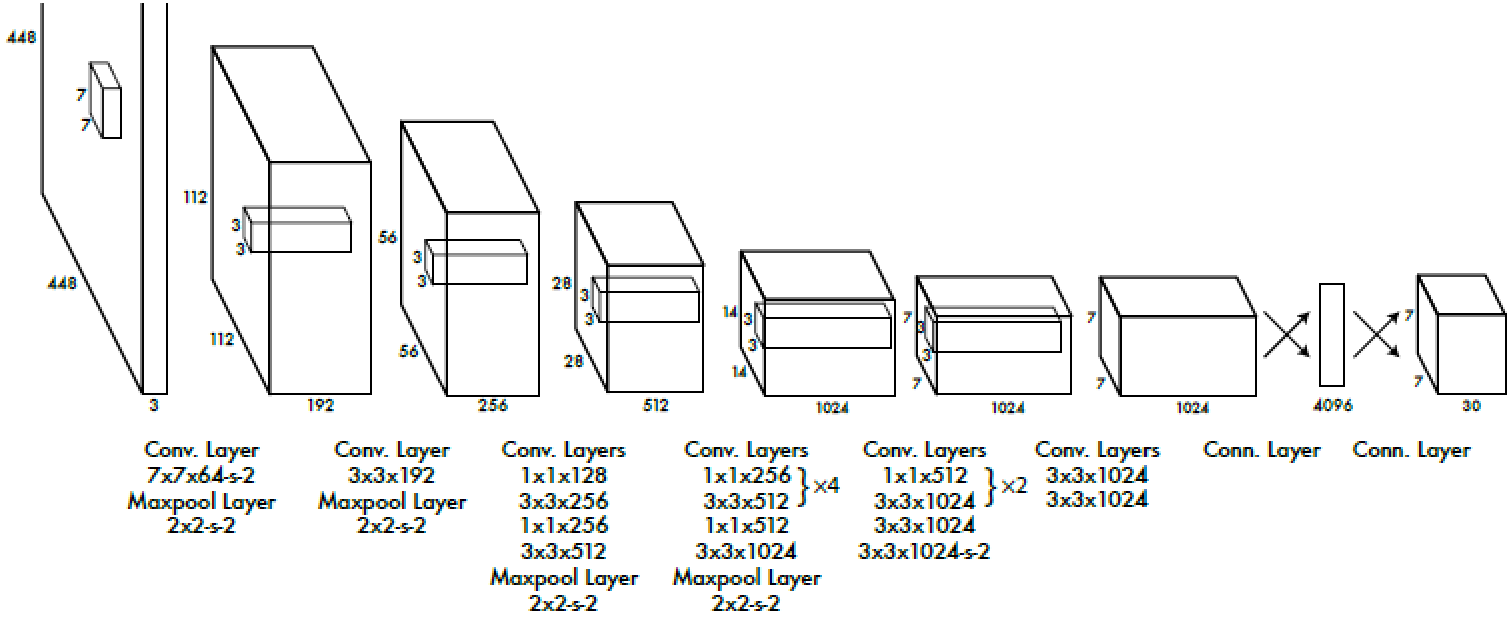
- Solution and Advantages:

@Simpler structure of network

@much more faster, even with real-time property: 150fps: able to process streaming video in real-time with less than 25 milliseconds of latency

@Maintaining  a proper accuracy range

# How YOLO gets its goal?




- For speed: Also! structure advantage!

➢ No bounding box proposals and subsequent pixel or feature resampling stage

✓ A neural network predicts bounding boxes and class probabilities directly from full images in one evaluation

# Unified Detection

- All BBox, All classes

1) Image → **S x S** grids

2) grid cell

→ **B**: BBoxes and Confidence score

x, y, w, h, confidence → $\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$

→ **C**: class probabilities w.r.t #classes

$\Pr(\text{Class}_i | \text{Object})$

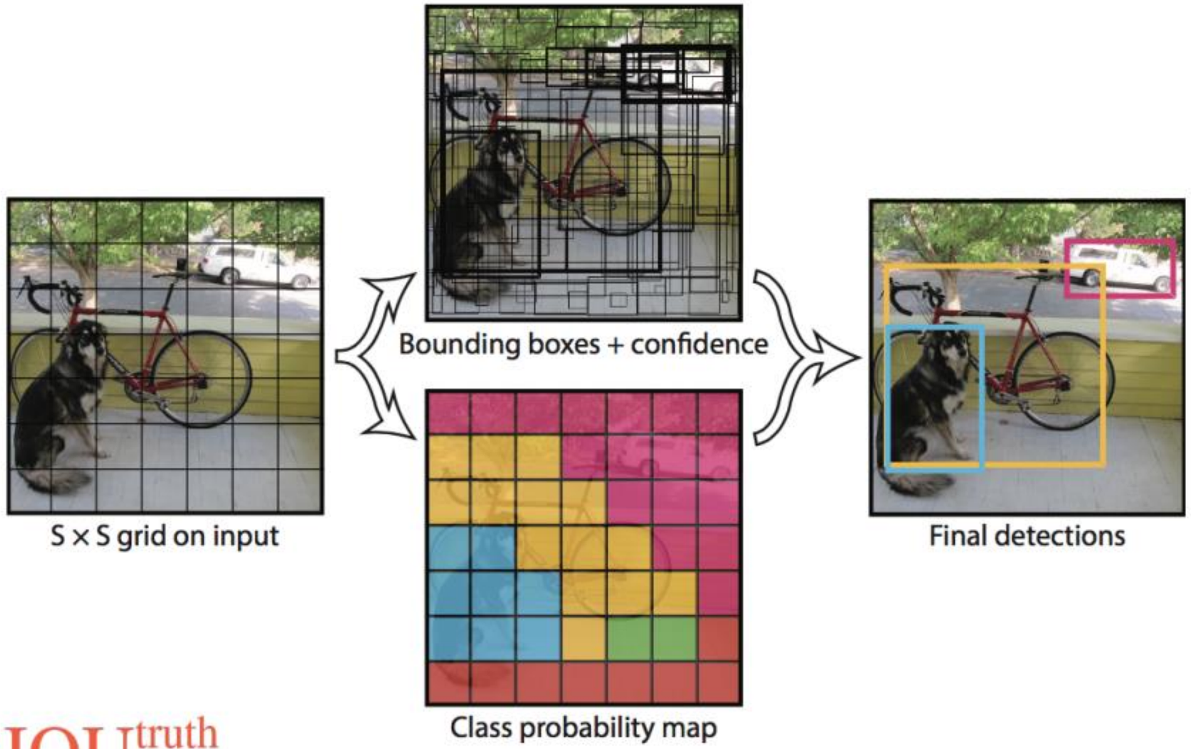



**Figure 2: The Model.** Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts $B$ bounding boxes, confidence for those boxes, and $C$ class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

# Unified Detection

- Predict one set of class probabilities per grid cell, regardless of the number of boxes B.
- At test time, individual box confidence prediction

$$\Pr(\mathrm{Class}_i|\mathrm{Object}) * \Pr(\mathrm{Object}) * \mathrm{IOU}_{\mathrm{pred}}^{\mathrm{truth}}$$
$$= \Pr(\mathrm{Class}_i) * \mathrm{IOU}_{\mathrm{pred}}^{\mathrm{truth}}$$

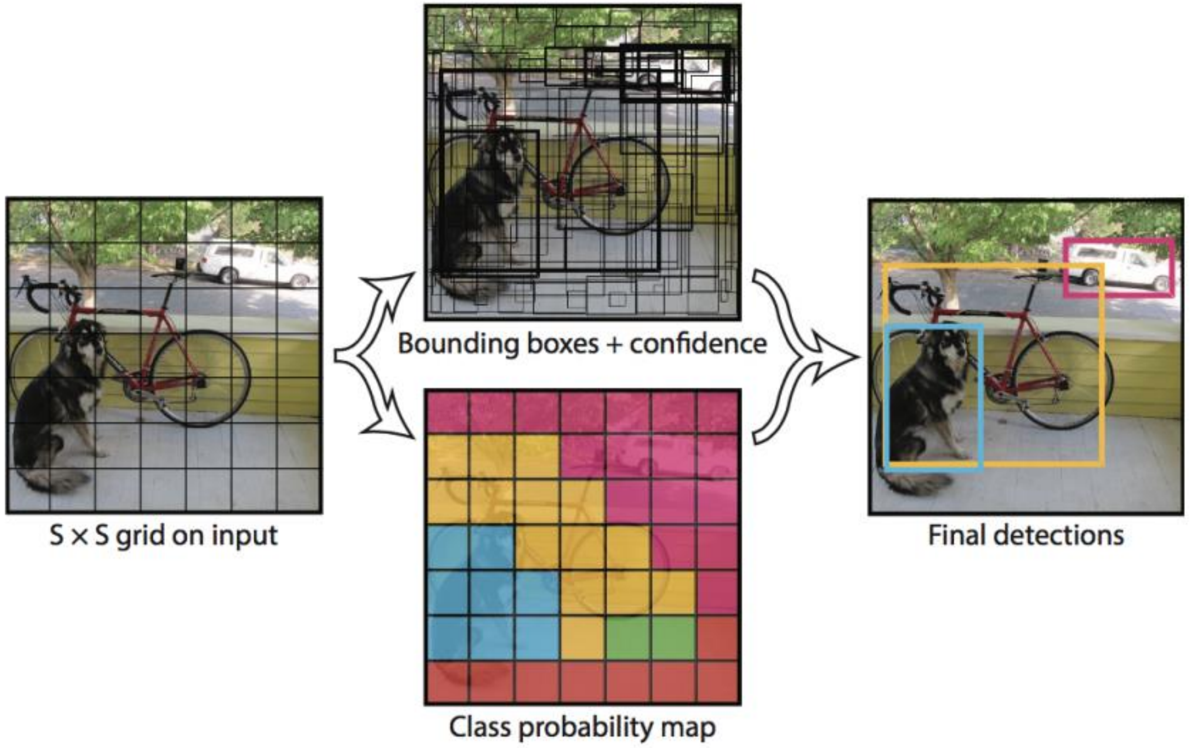
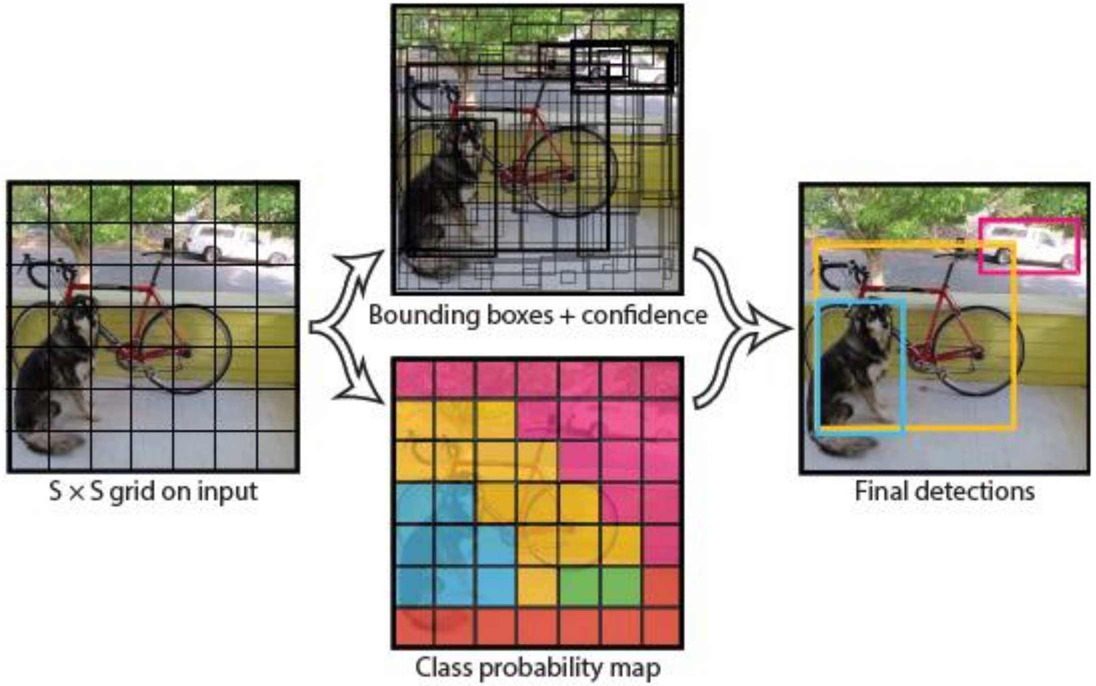


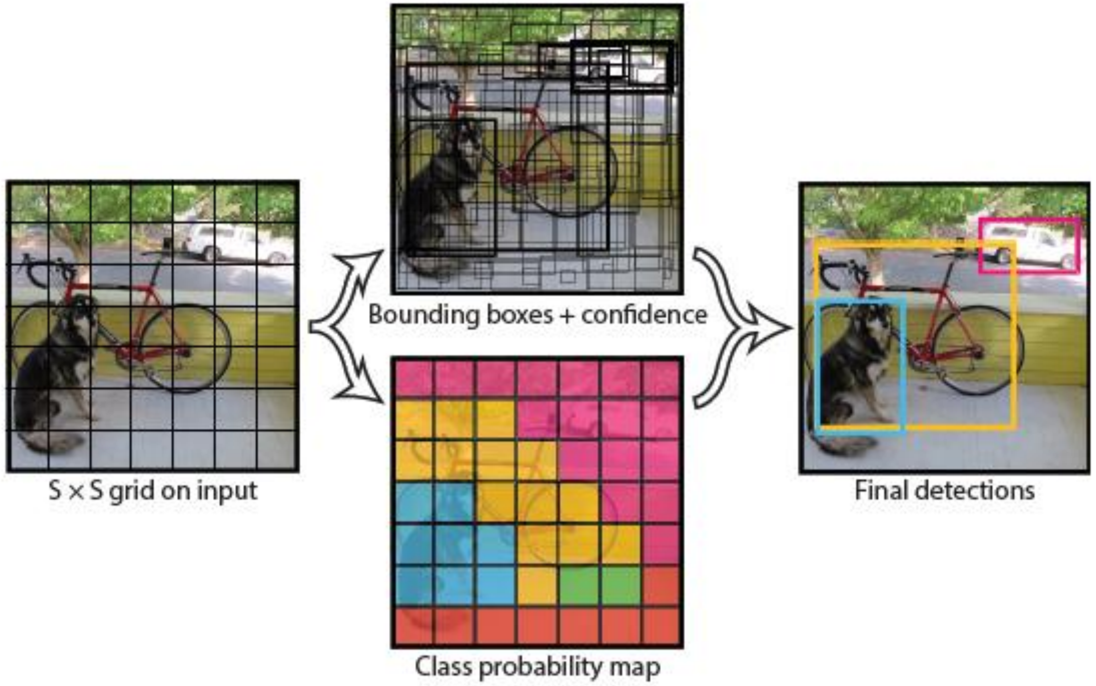

**Figure 2: The Model.** Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts $B$ bounding boxes, confidence for those boxes, and $C$ class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

# How unified detection works?




- uses features from the entire image to predict each bounding box
- predicts all bounding boxes across all classes for an image simultaneously?
- ➢ divides the input image into an s*s grid. If the center of an object falls into a grid cell, the cell is responsible for detecting that object.
- ➢ each grid cell predicts B bounding boxes and confidence scores for those boxes
- ➢ Each grid also predicts C conditional(conditioned on the grid cell containing an object) class probabilities

# How unified detection works?



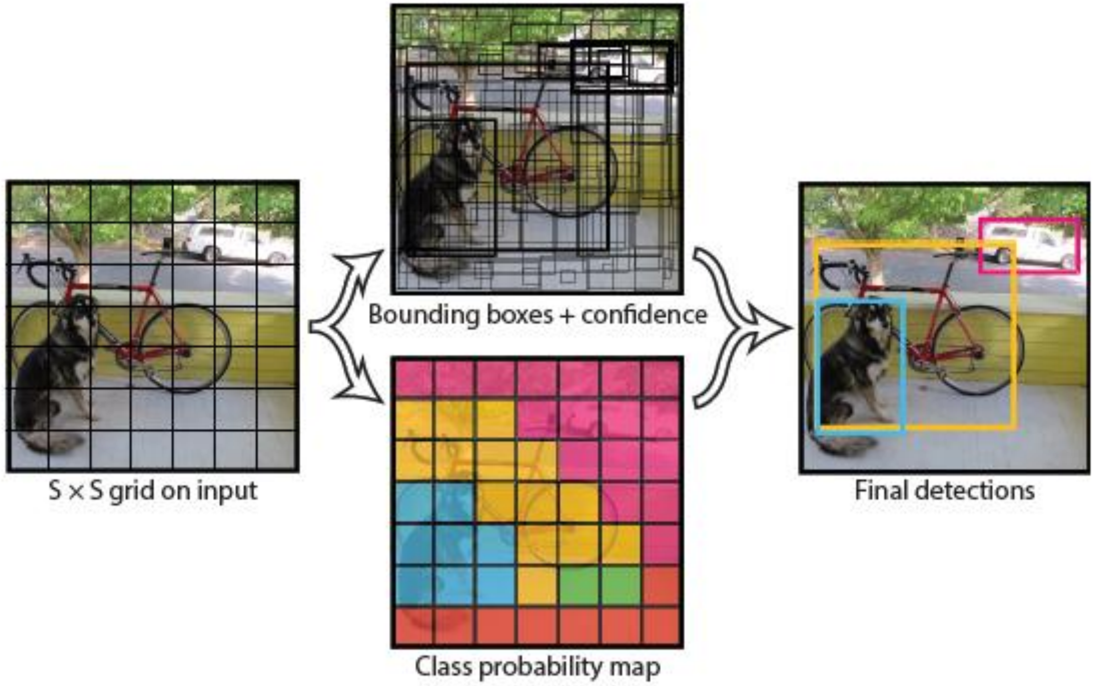

**<u>confidence scores</u>**: reflect how confident is that the box contains an object+how accurate the box is .

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

**conditional class probabilities:** conditioned on the grid cell containing an object

$$\Pr(\text{Class}_i|\text{Object})$$

# How unified detection works?

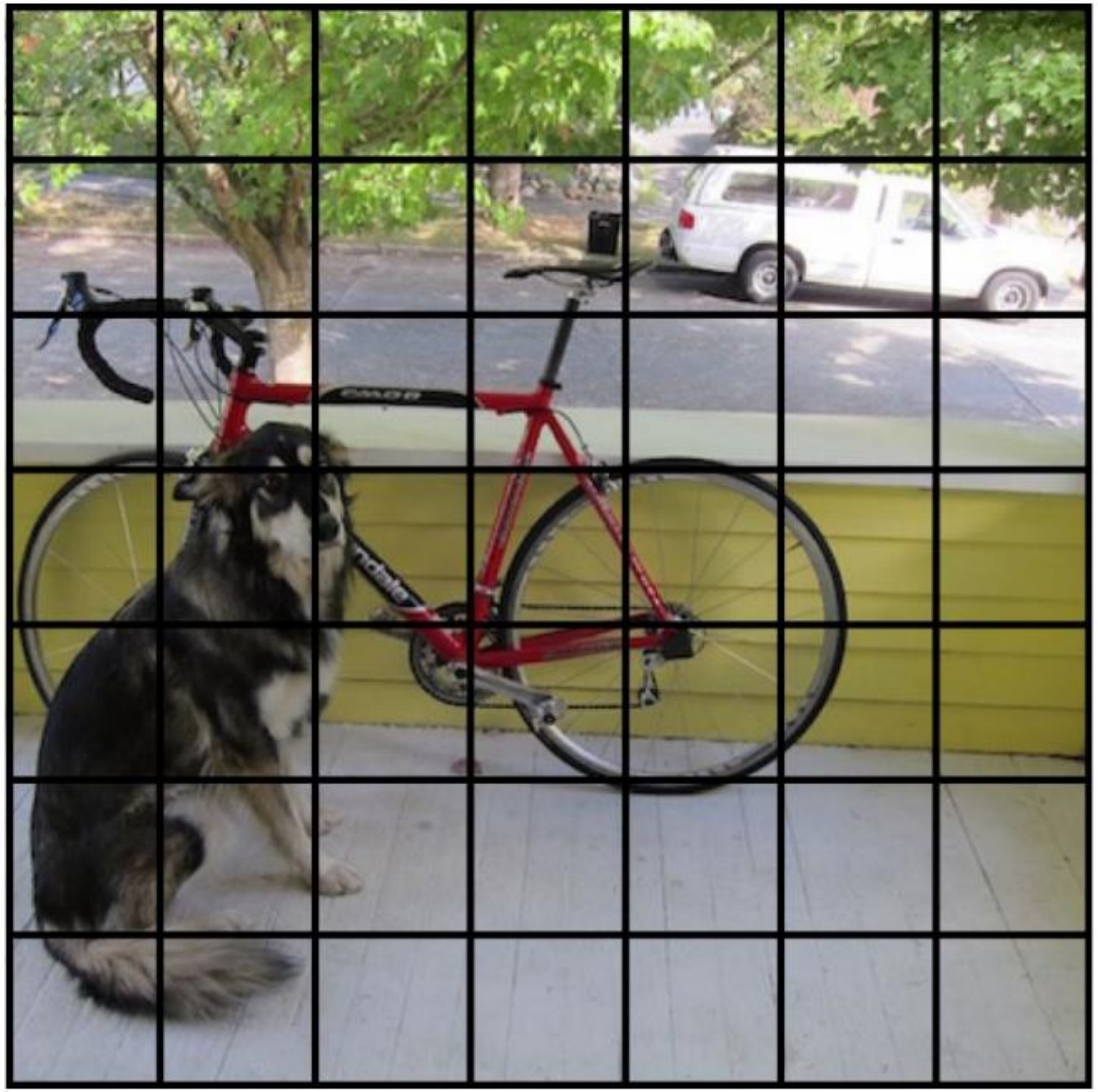



$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

- At test time,  multiply the conditional class probabilities and the individual box confidence predictions
- giving class-specific confidence scores for each box
- Showing  both the probability of that class appearing in the box and how well the predicted box fits the object

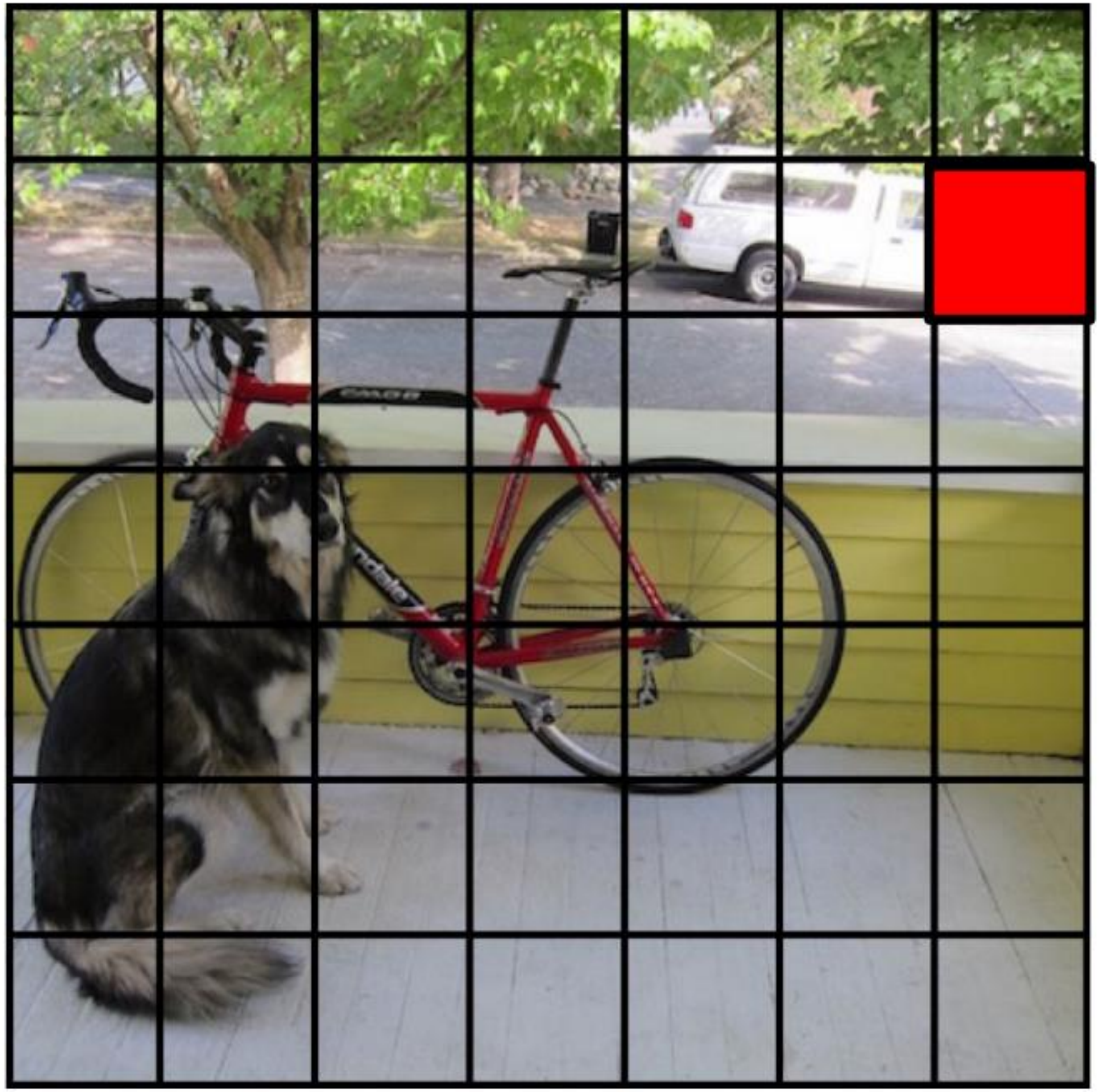
# Automatic Object Detection from Image

# We split the image into an S*S grid



**7*7 grids**

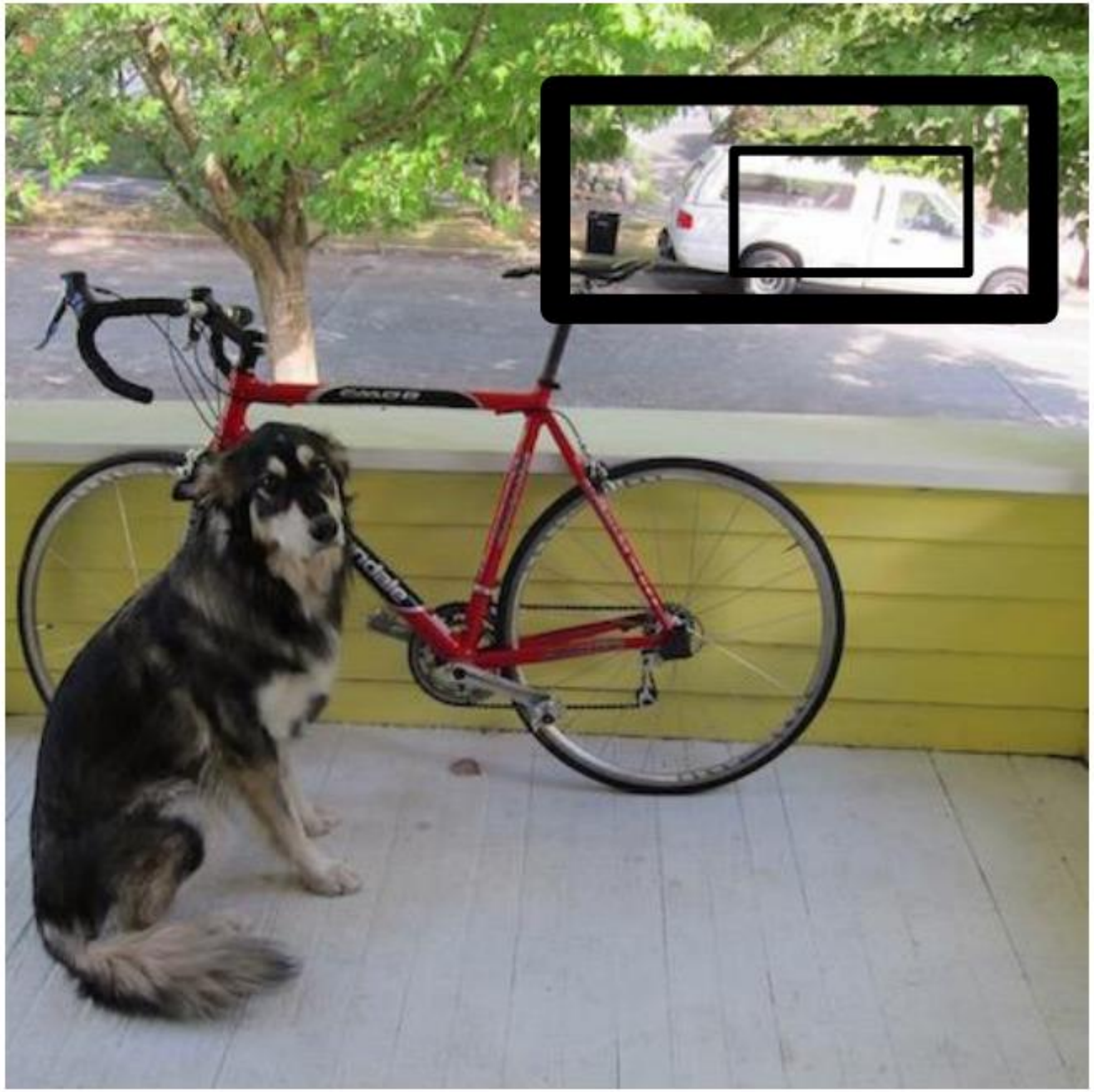
Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)



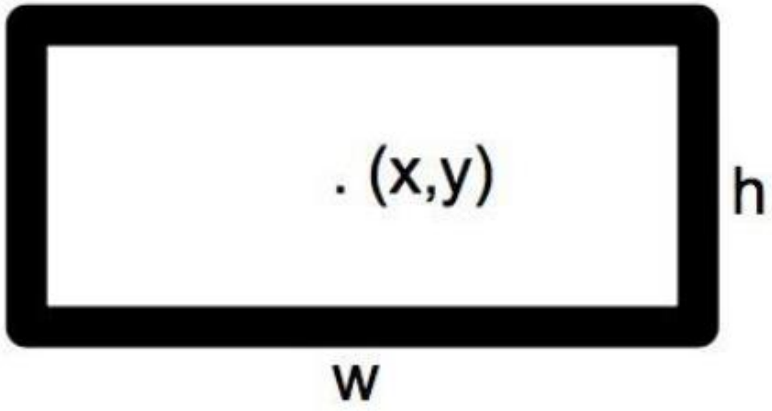
Each cell predicts B boxes(x,y,w,h) and confidences of each box: P(Object)

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

B = 2

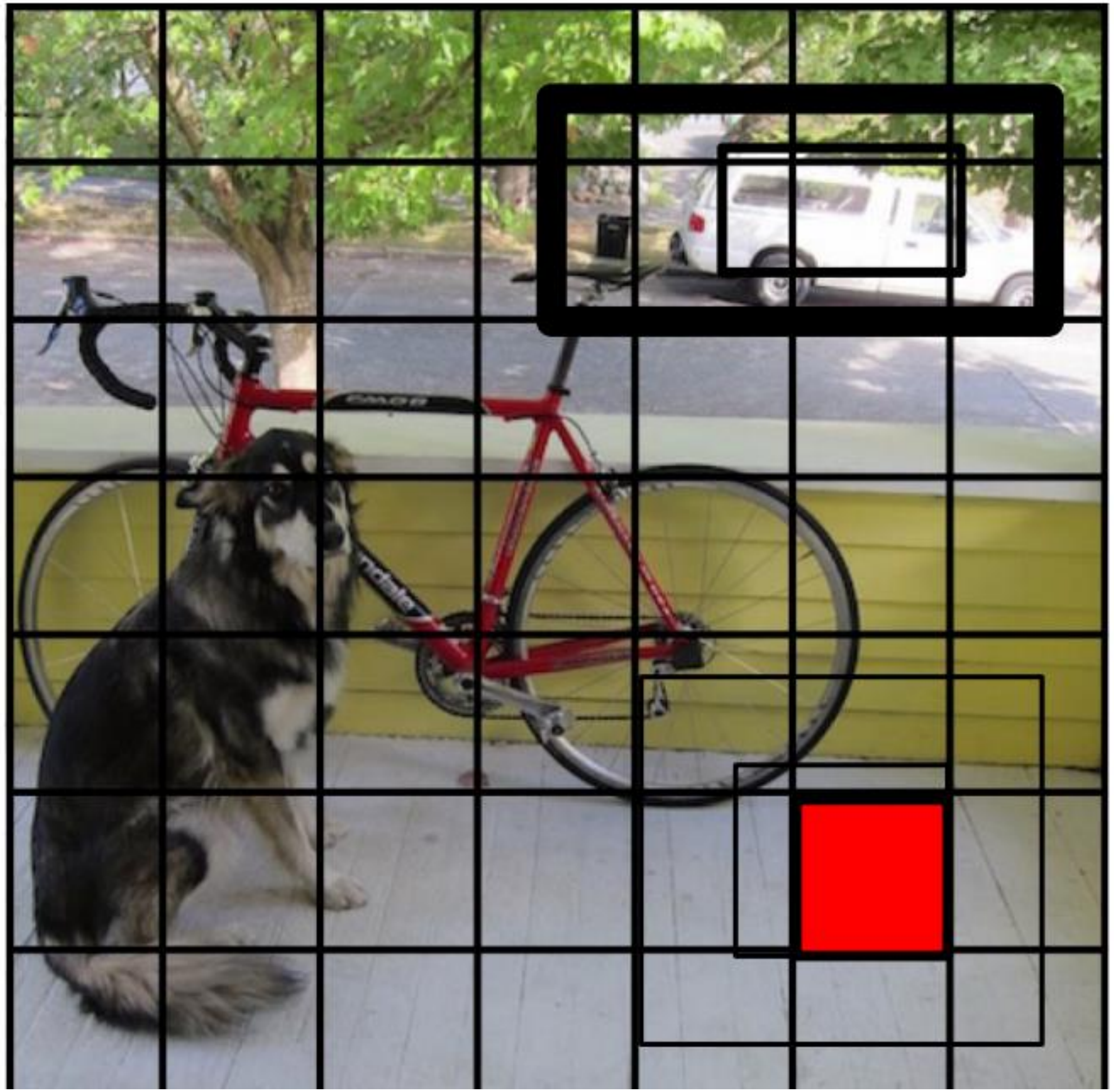


each box predict:



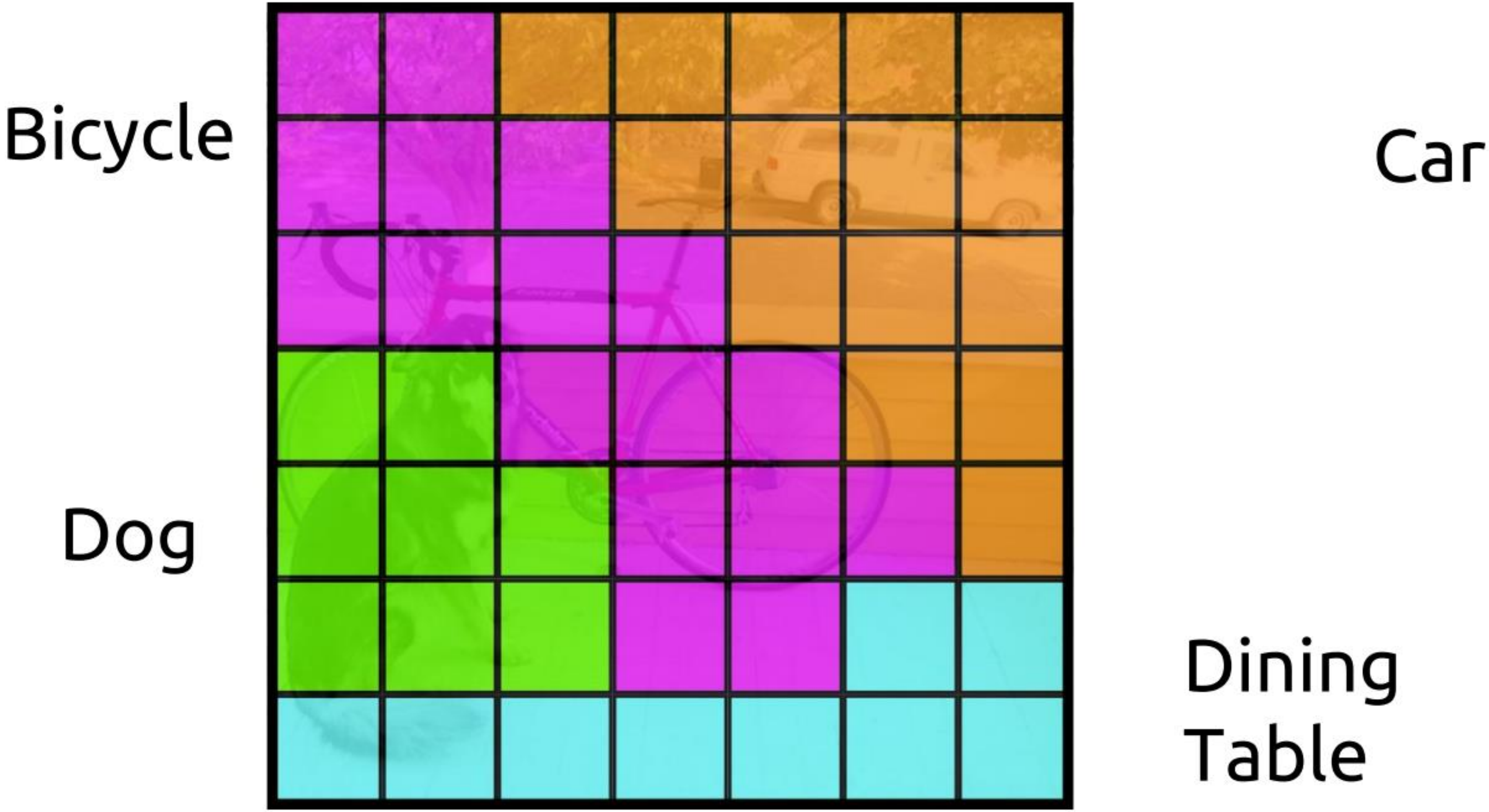

P(Object): probability that the box contains an object

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



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

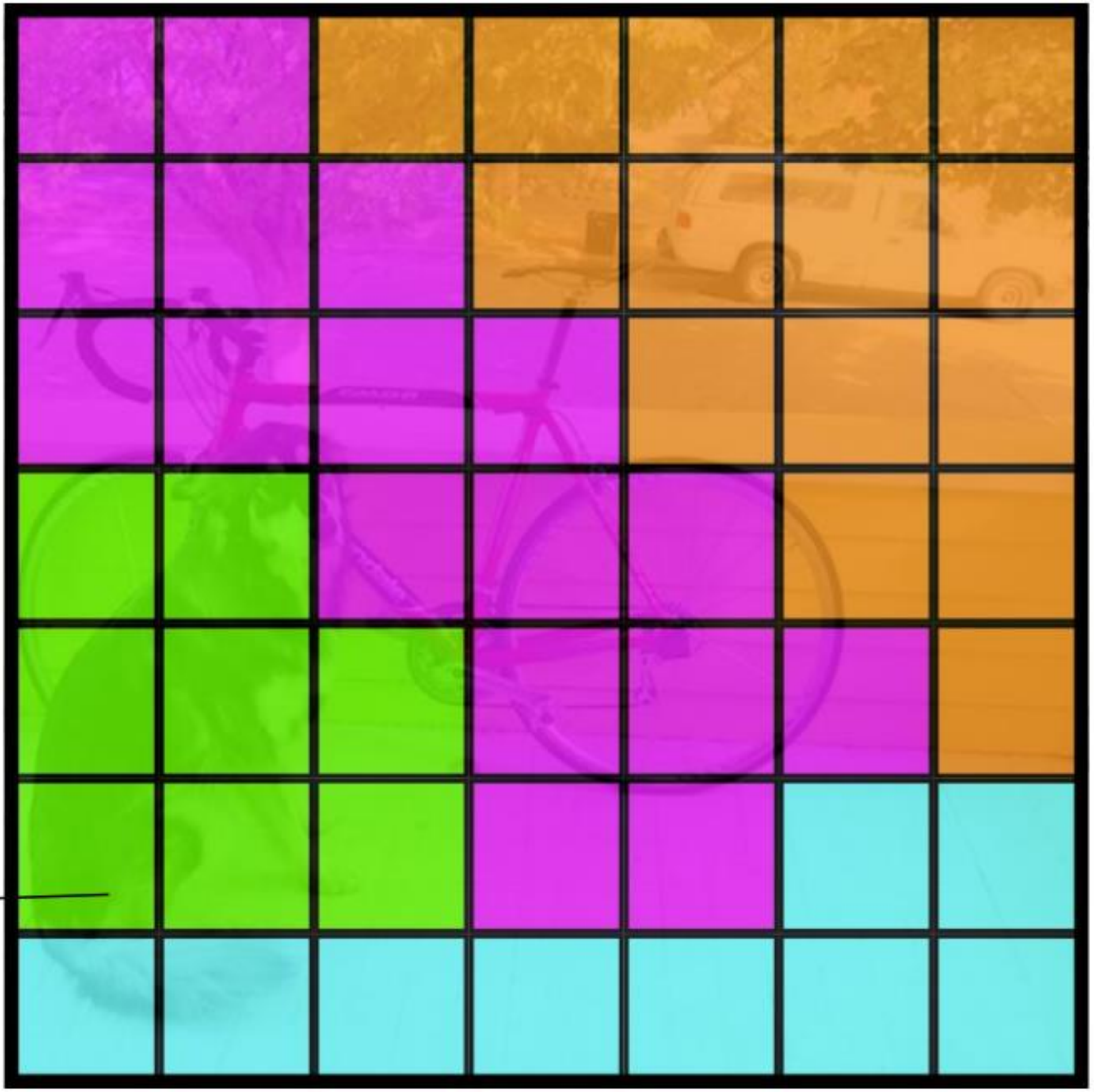
# Each cell also predicts a class probability.



# Conditioned on object: P(Car | Object)

Bicycle

Car

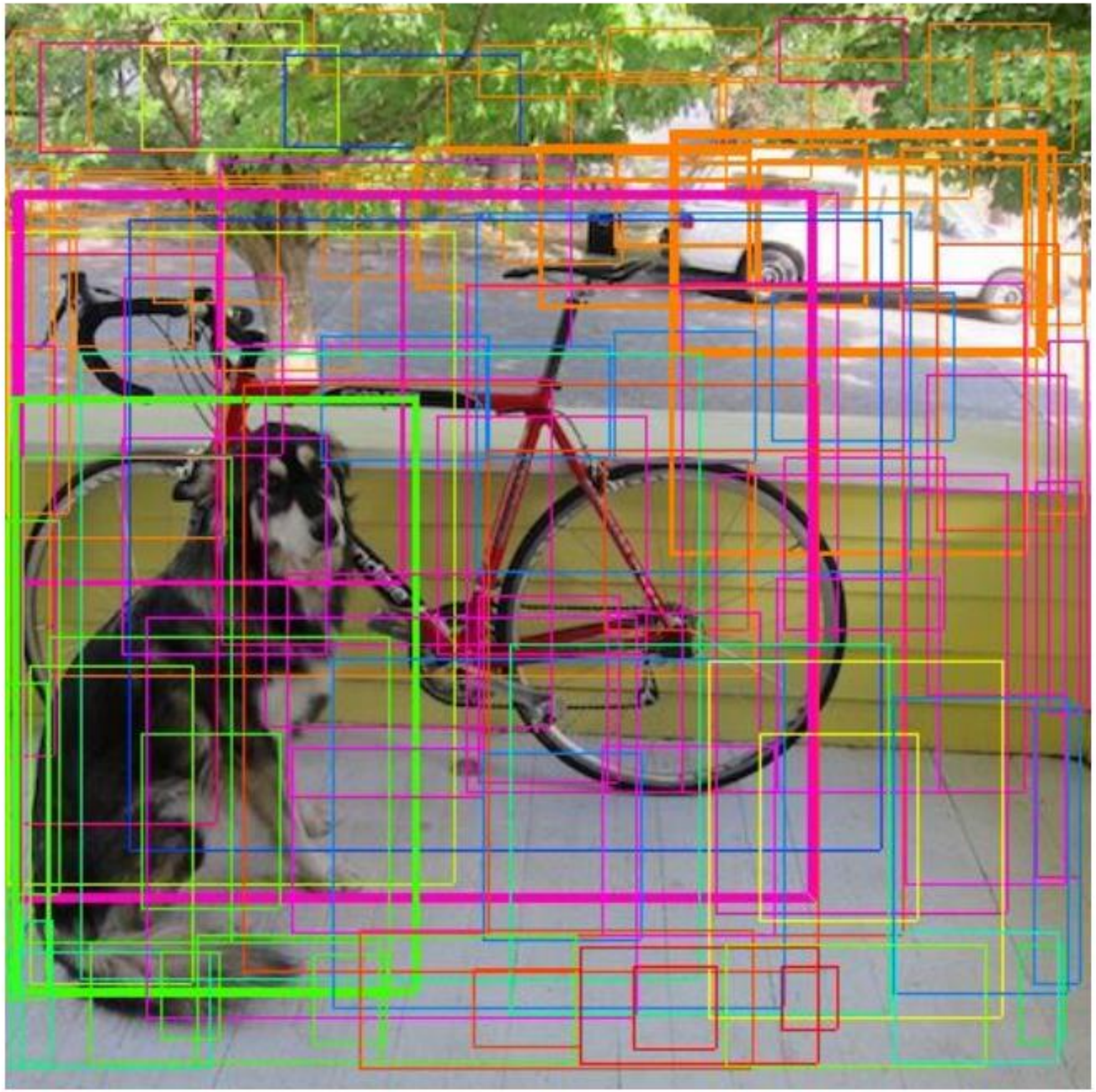


Dog

**Eg.**
**Dog = 0.8**
Cat = 0
Bike = 0

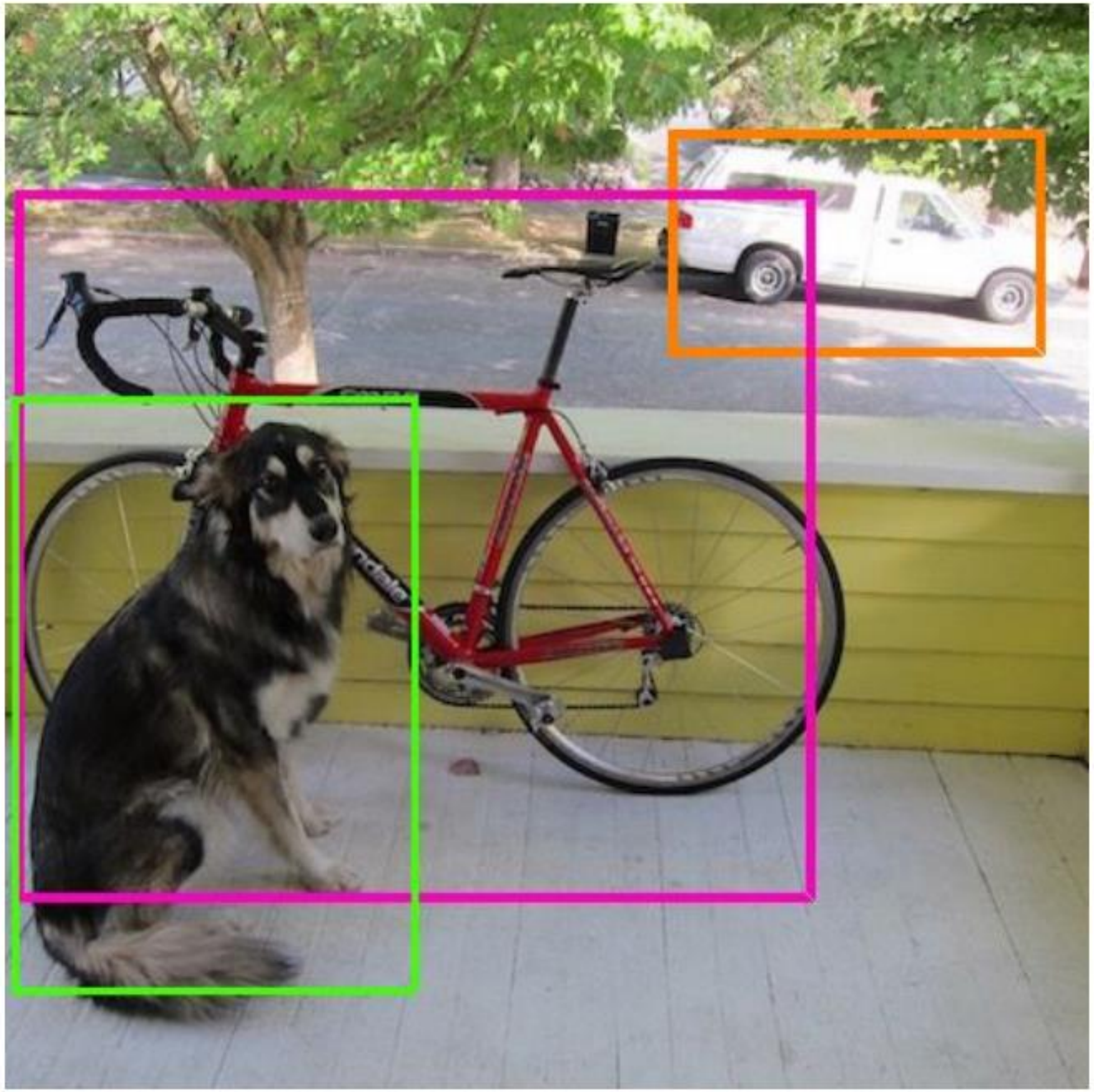
Dining Table

# Then we combine the box and class predictions.



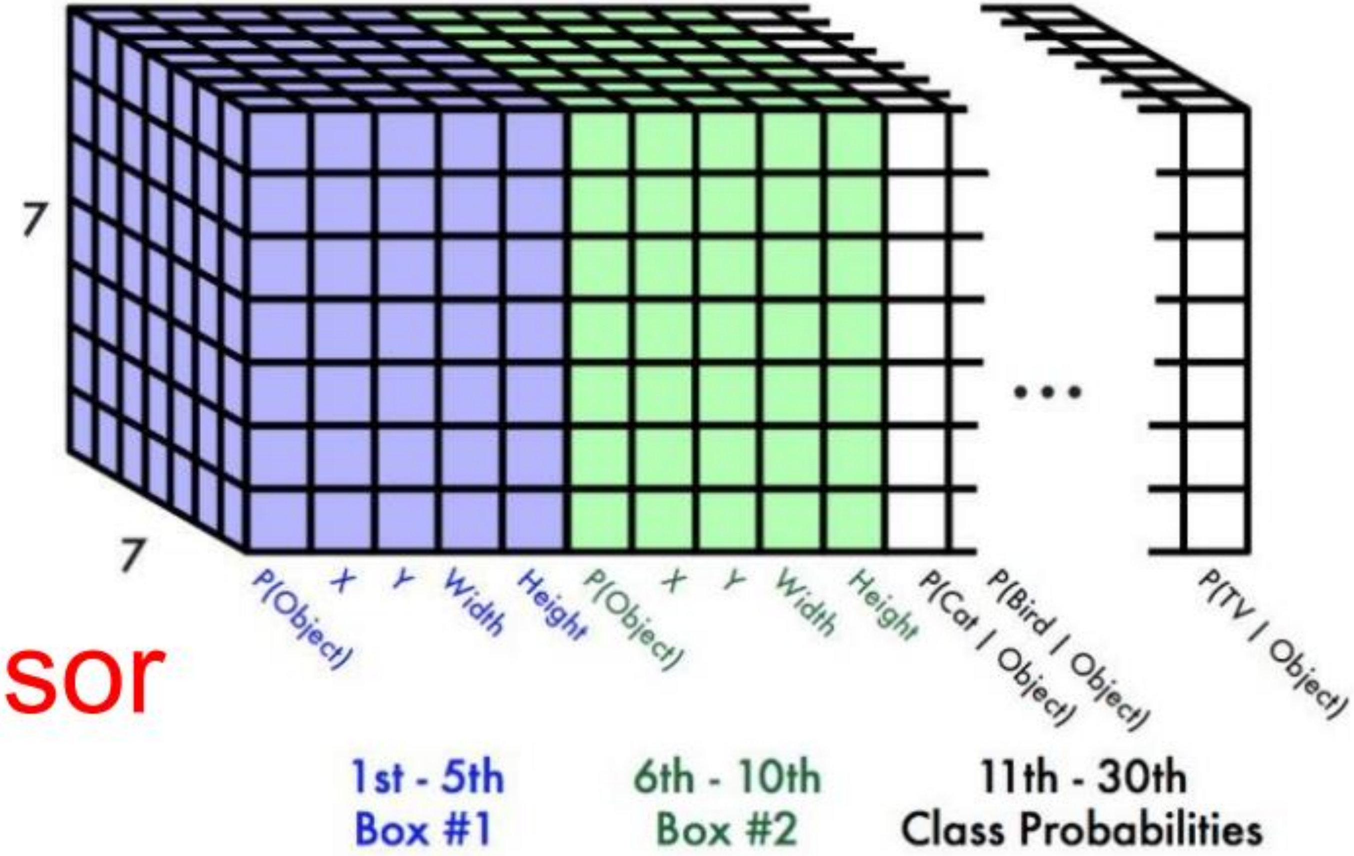
$$P(\text{class}|\text{Object}) * P(\text{Object}) = P(\text{class})$$

# Finally we do threshold detections and NMS



Each cell predicts:

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

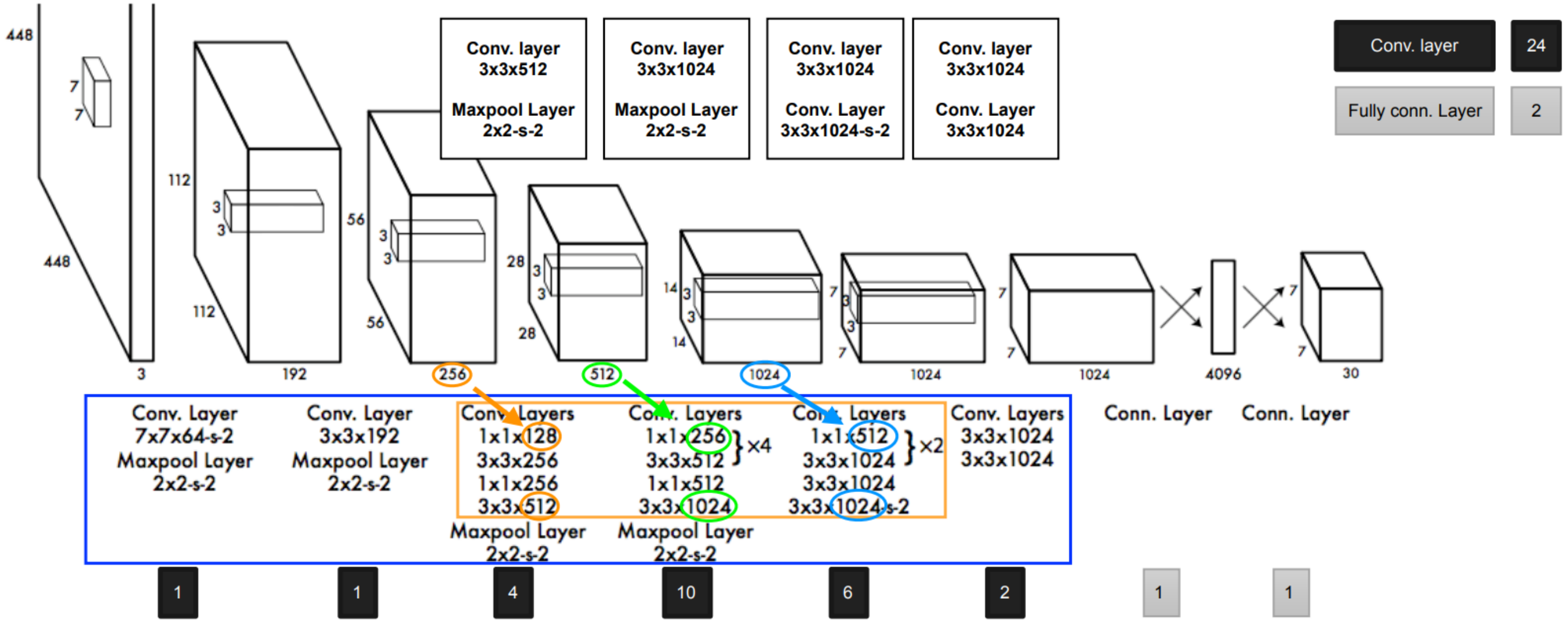



S * S * (B * 5 + C) tensor

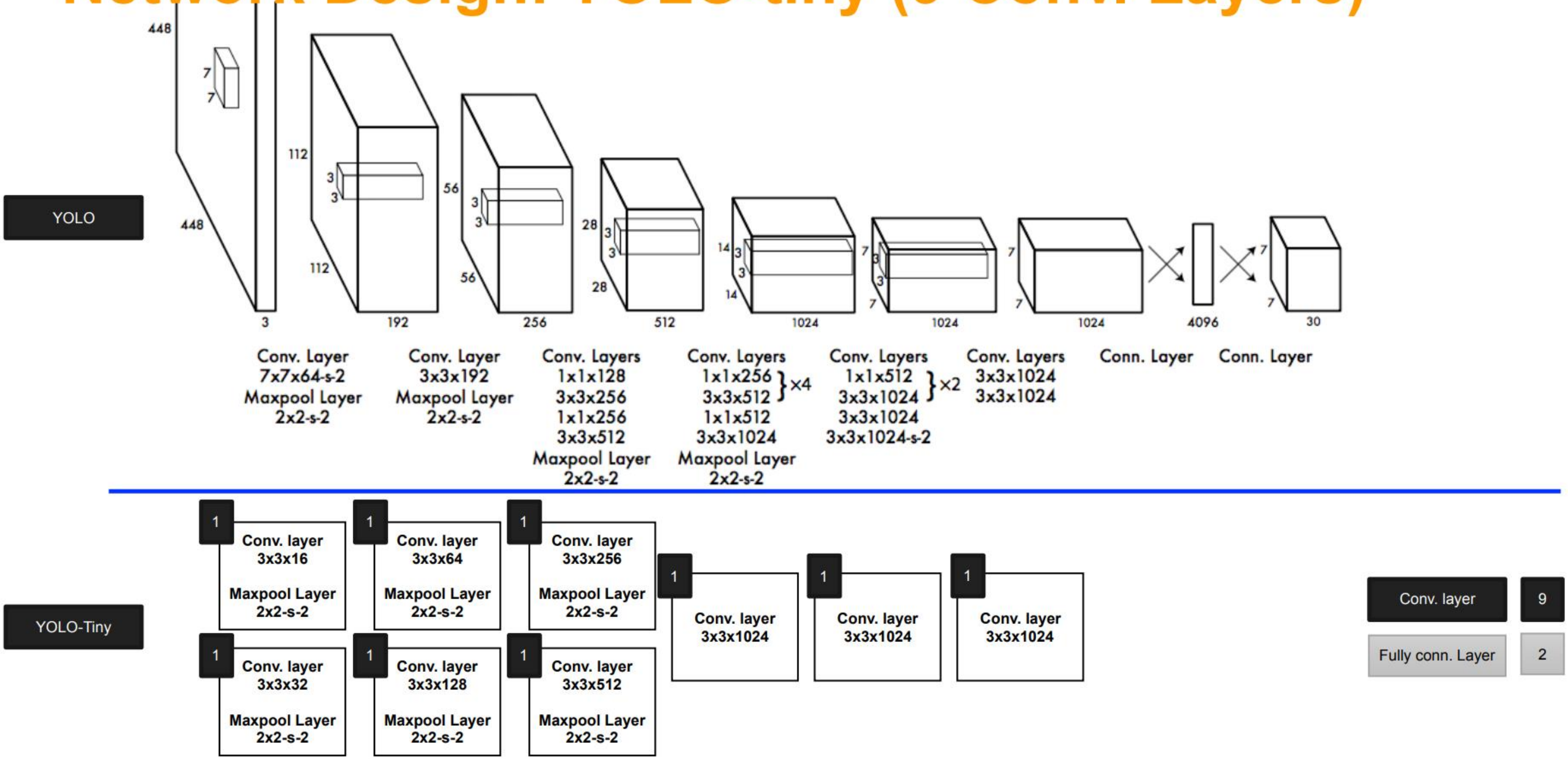
# Network Design: YOLO

- Modified GoogLeNet
- 1x1 reduction layer ("Network in Network")

Our network architecture is inspired by the GoogLeNet model for image classification [34]. Our network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, we simply use $1 \times 1$ reduction layers followed by $3 \times 3$ convolutional layers, similar to Lin et al [22]. The full network is shown in Figure 3.



# Network Design: YOLO-tiny (9 Conv. Layers)

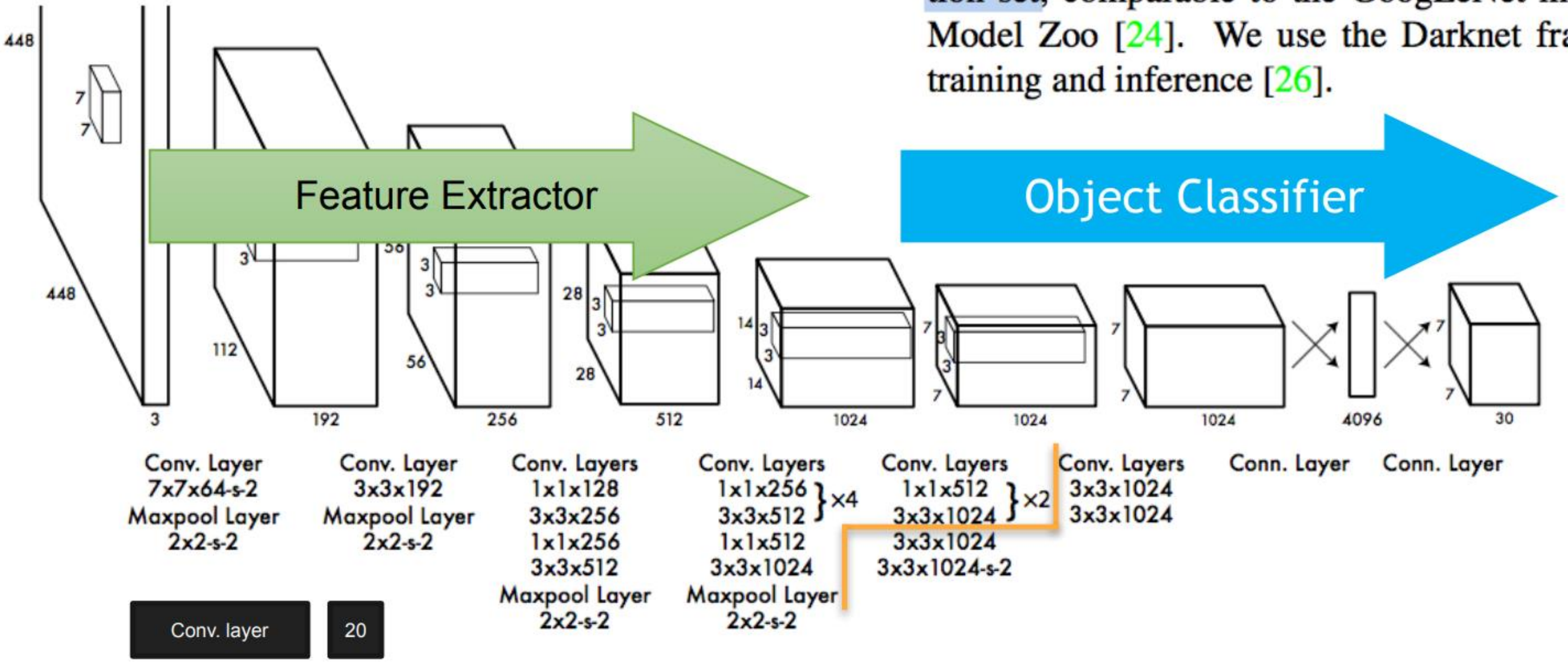



Slide credit to Taegyun Jeon

# Training

1) Pretrain with ImageNet 1000-class competition dataset

We pretrain our convolutional layers on the ImageNet 1000-class competition dataset [30]. For pretraining we use the first 20 convolutional layers from Figure 3 followed by a average-pooling layer and a fully connected layer. We train this network for approximately a week and achieve a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set, comparable to the GoogLeNet models in Caffe's Model Zoo [24]. We use the Darknet framework for all training and inference [26].



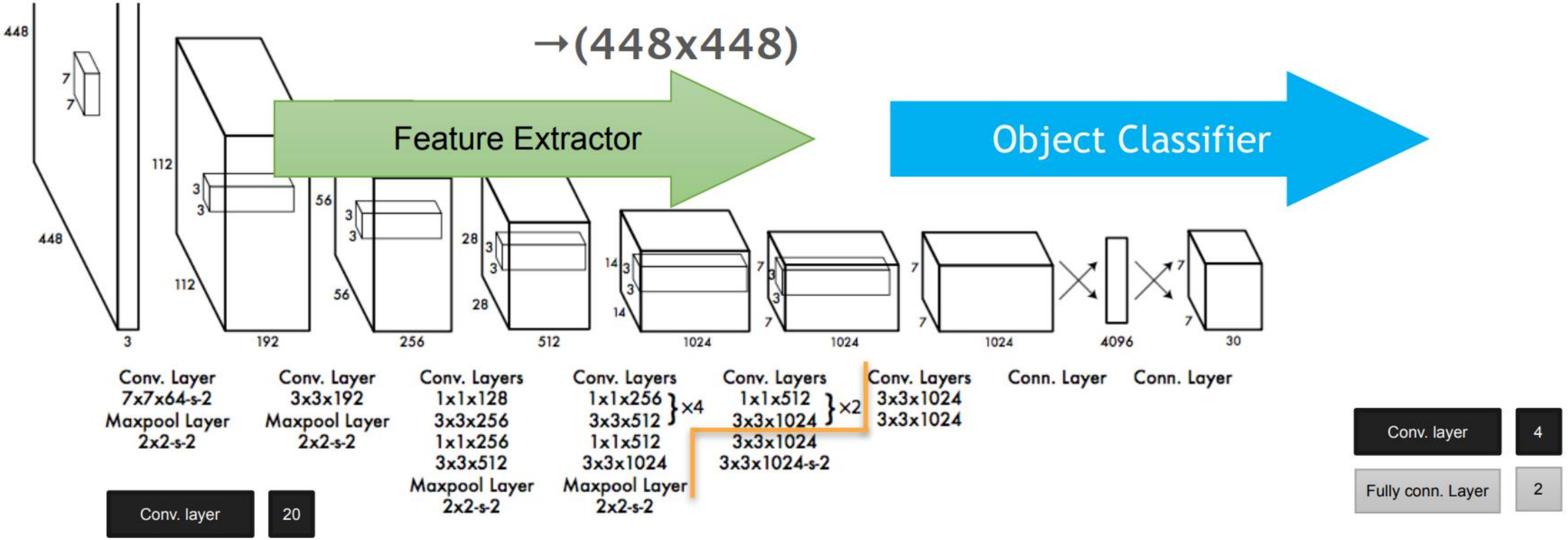

Slide credit to Taegyun Jeon

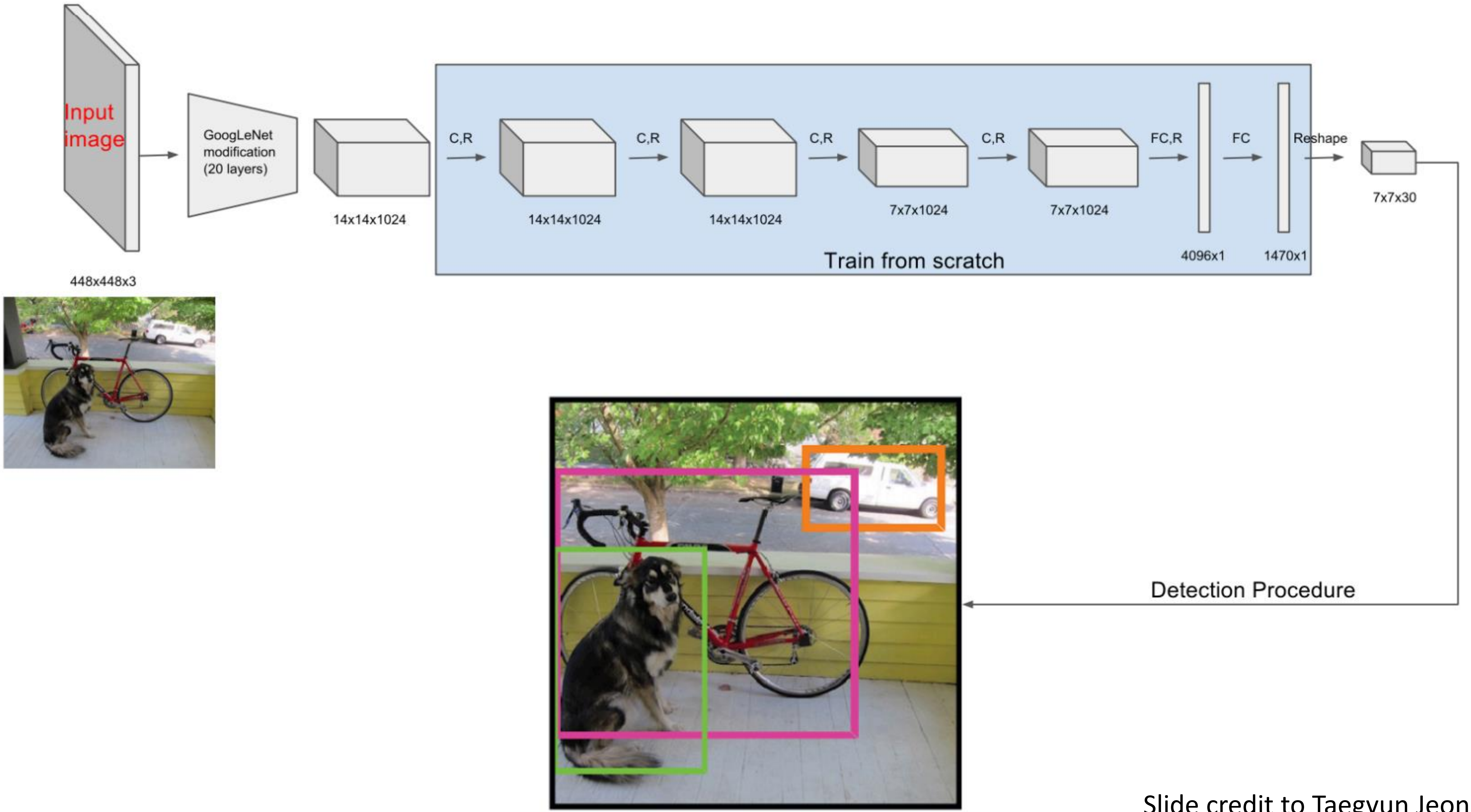
# Training

2) "Network on Convolutional Feature Maps"

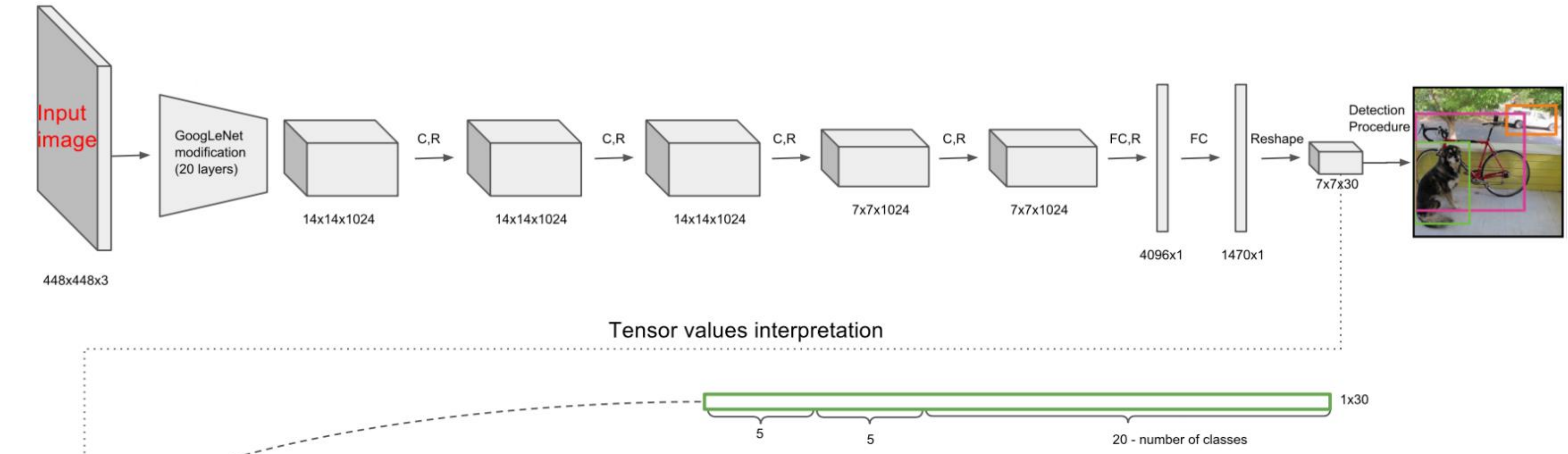
Increased input resolution (224x224) →(448x448)

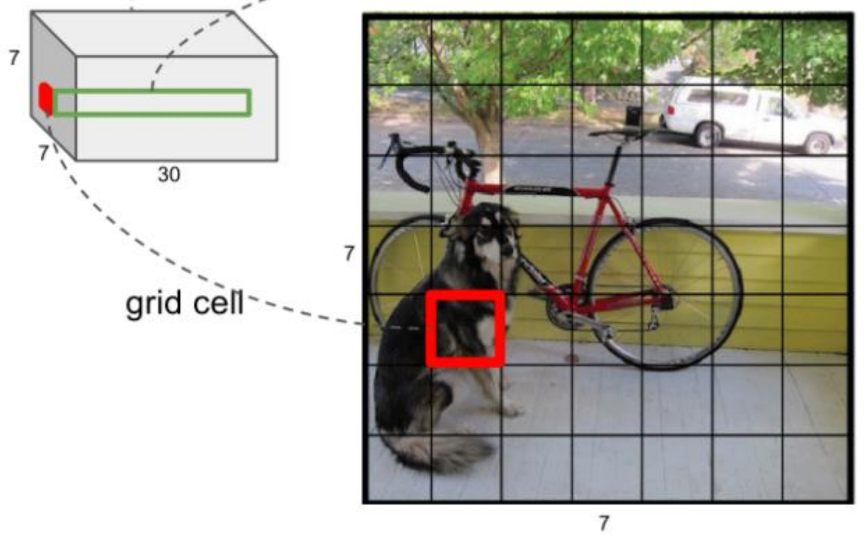
We then convert the model to perform detection. Ren et al. show that adding both convolutional and connected layers to pretrained networks can improve performance [29]. Following their example, we add four convolutional layers and two fully connected layers with randomly initialized weights. Detection often requires fine-grained visual information so we increase the input resolution of the network from 224 × 224 to 448 × 448.



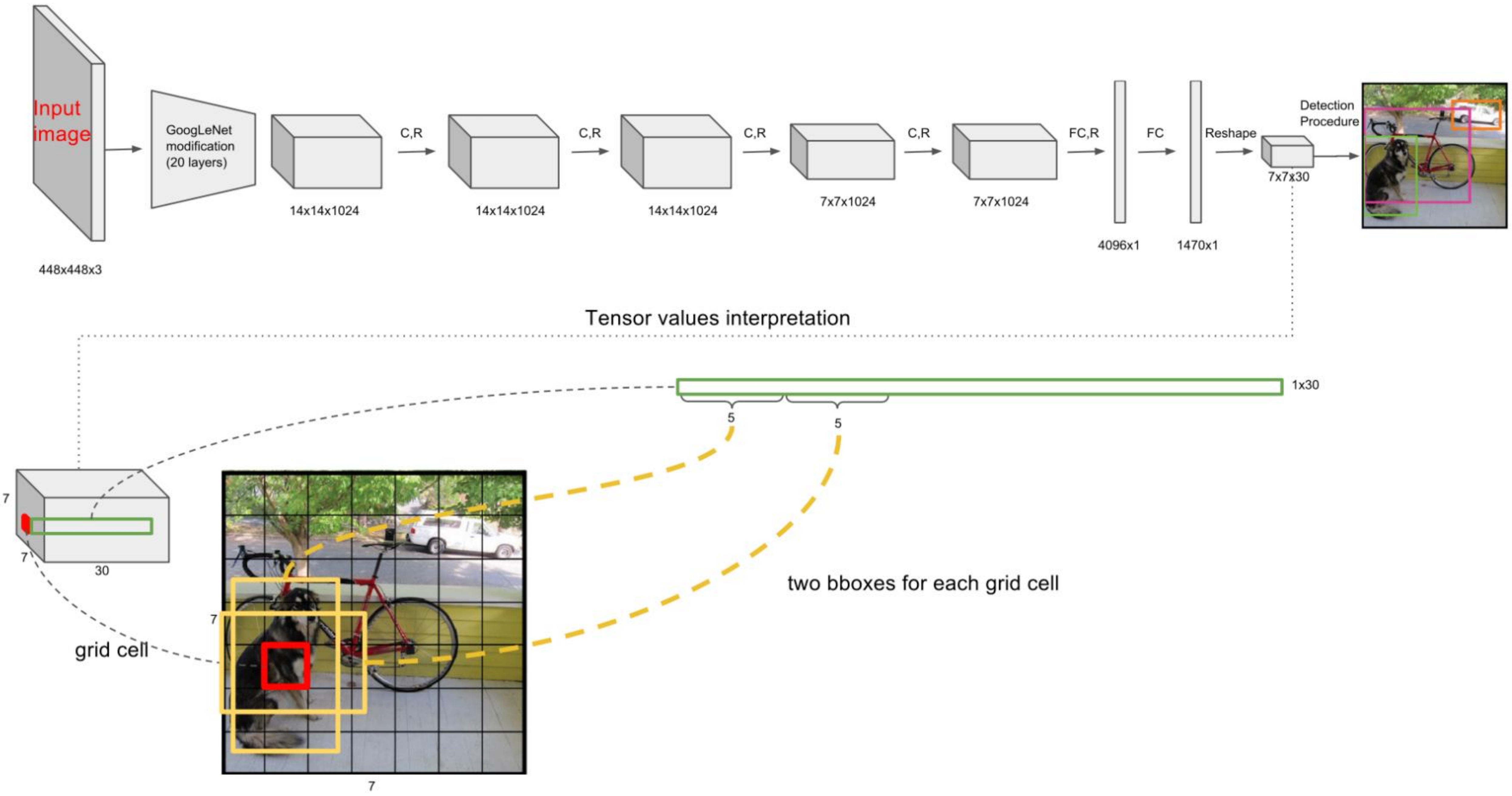



Slide credit to Taegyun Jeon

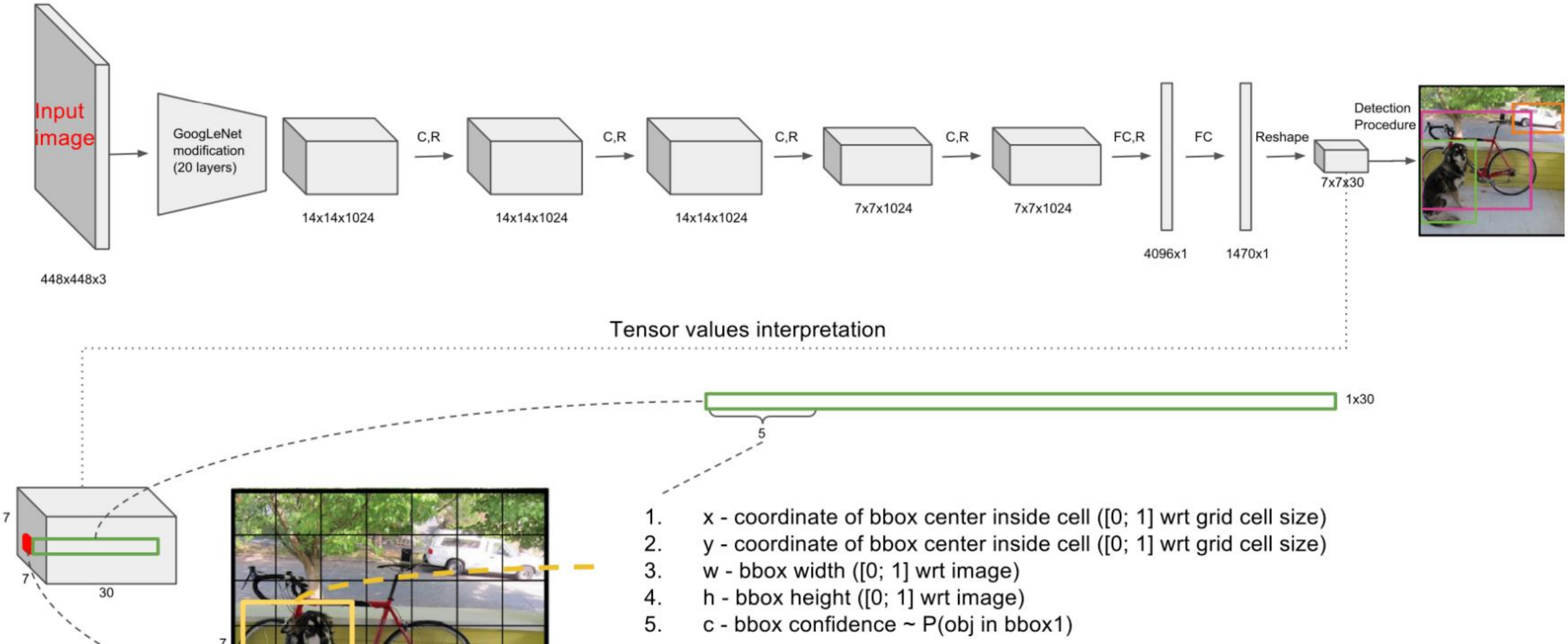


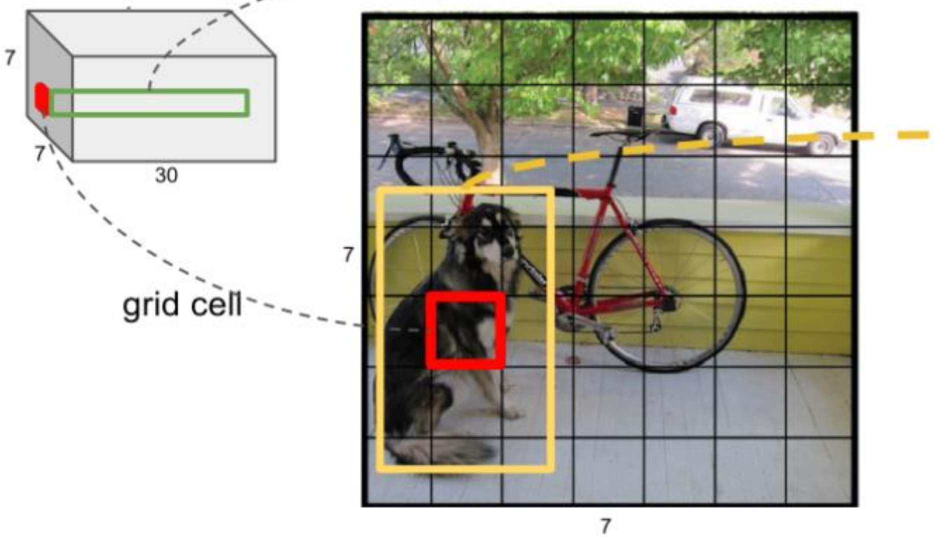




Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parametrize the bounding box $x$ and $y$ coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.



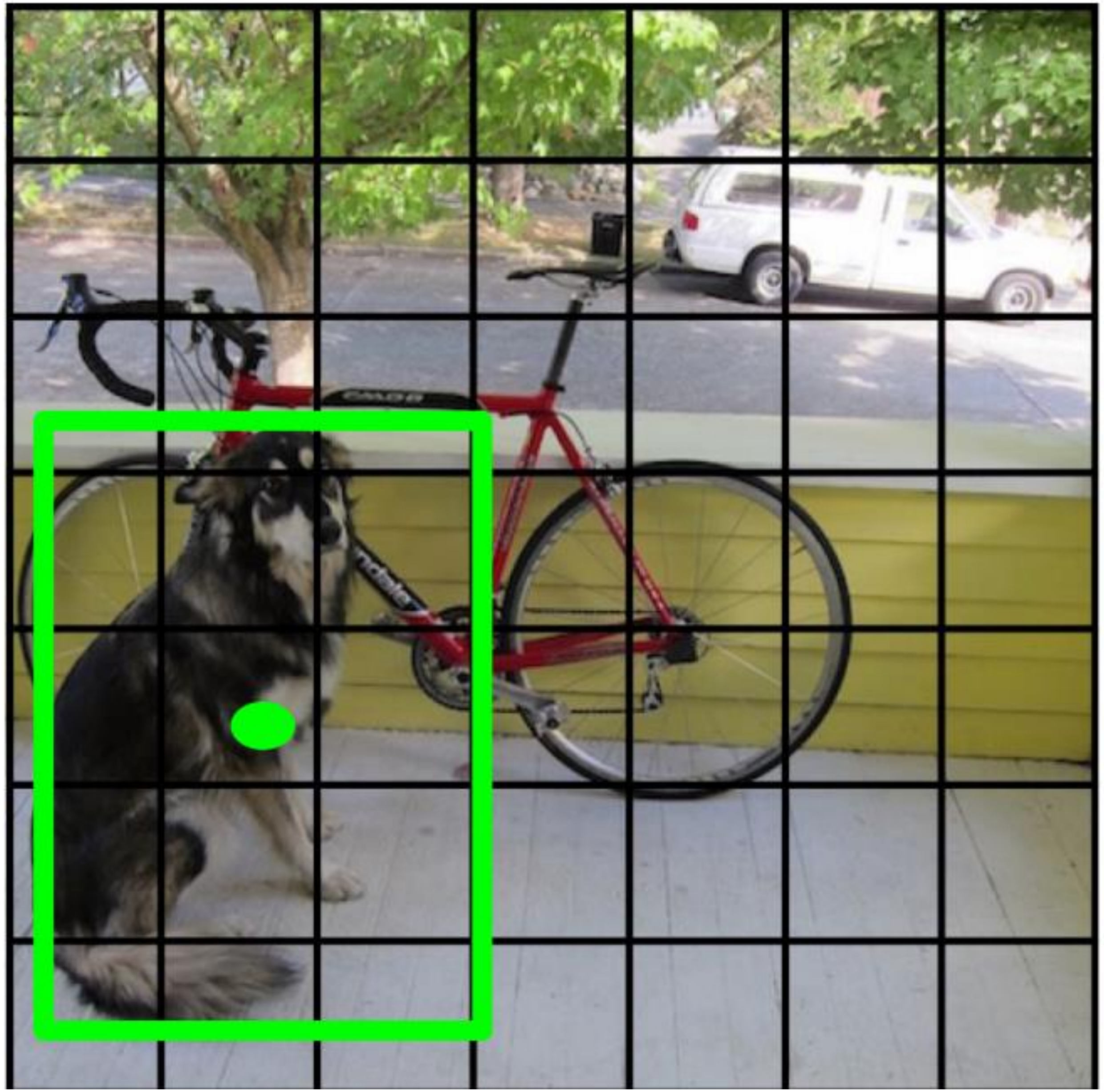

Slide credit to Taegyun Jeon



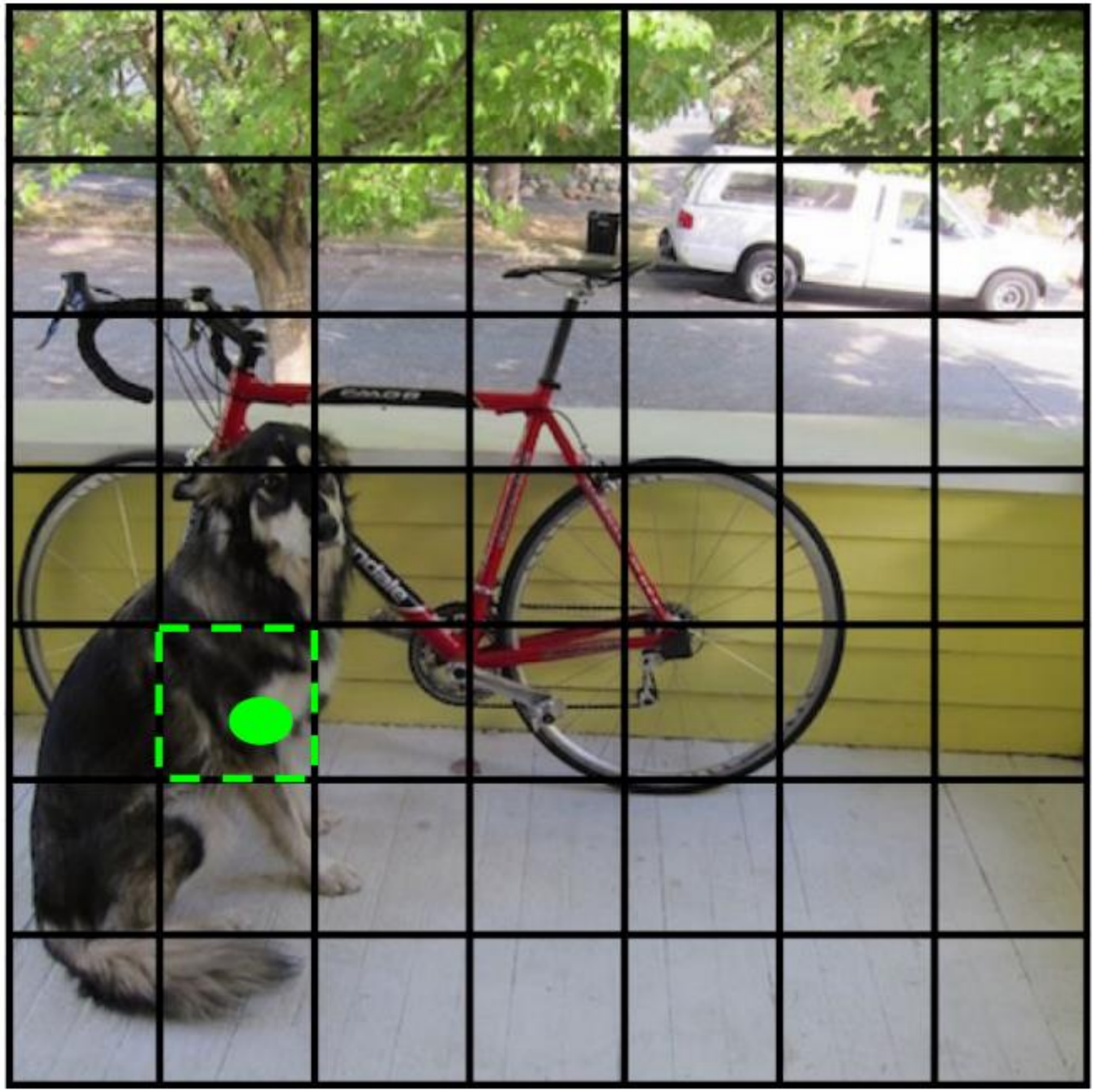




Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parametrize the bounding box $x$ and $y$ coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.

# During training, match example to the right cell

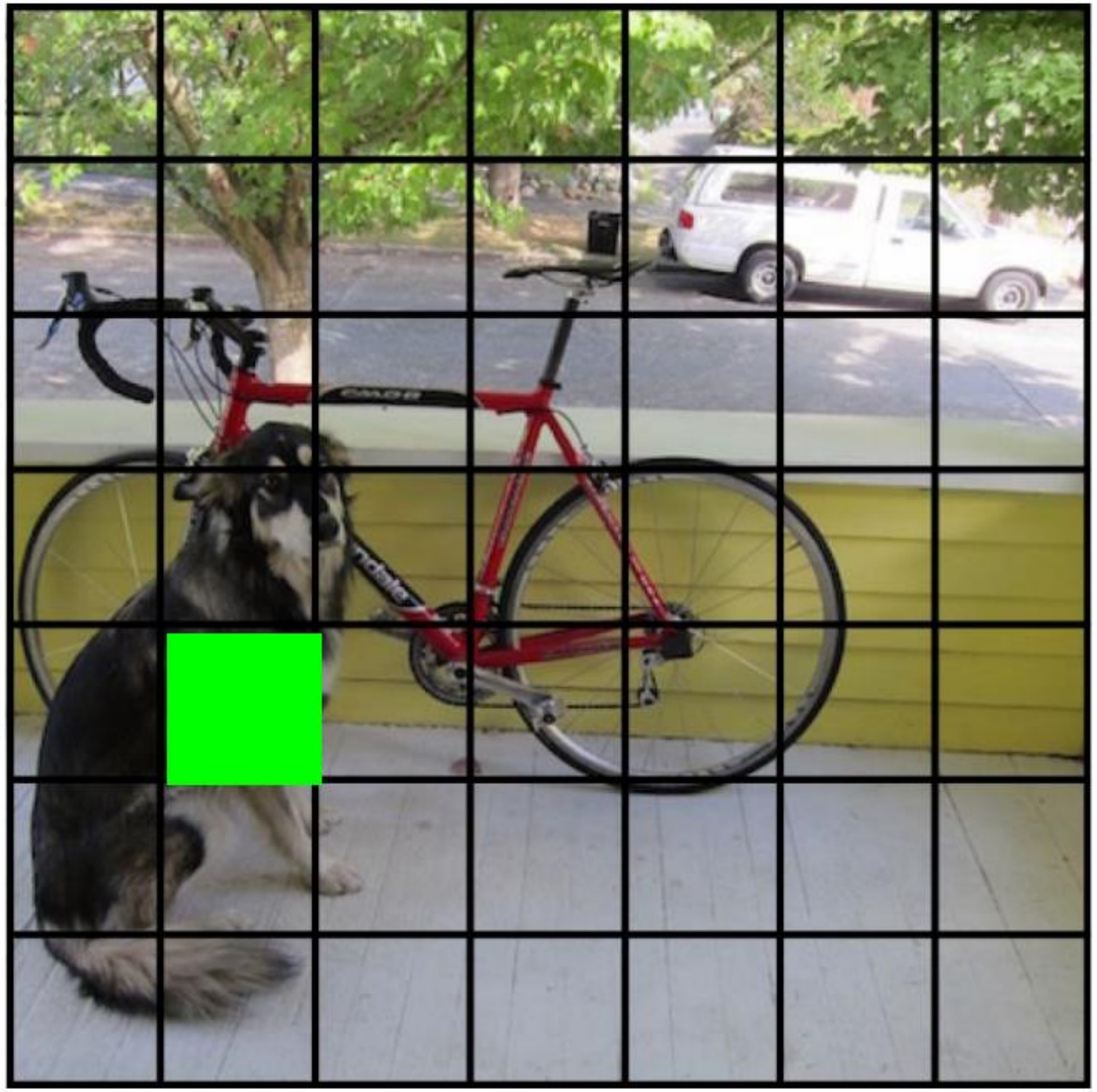


# During training, match example to the right cell

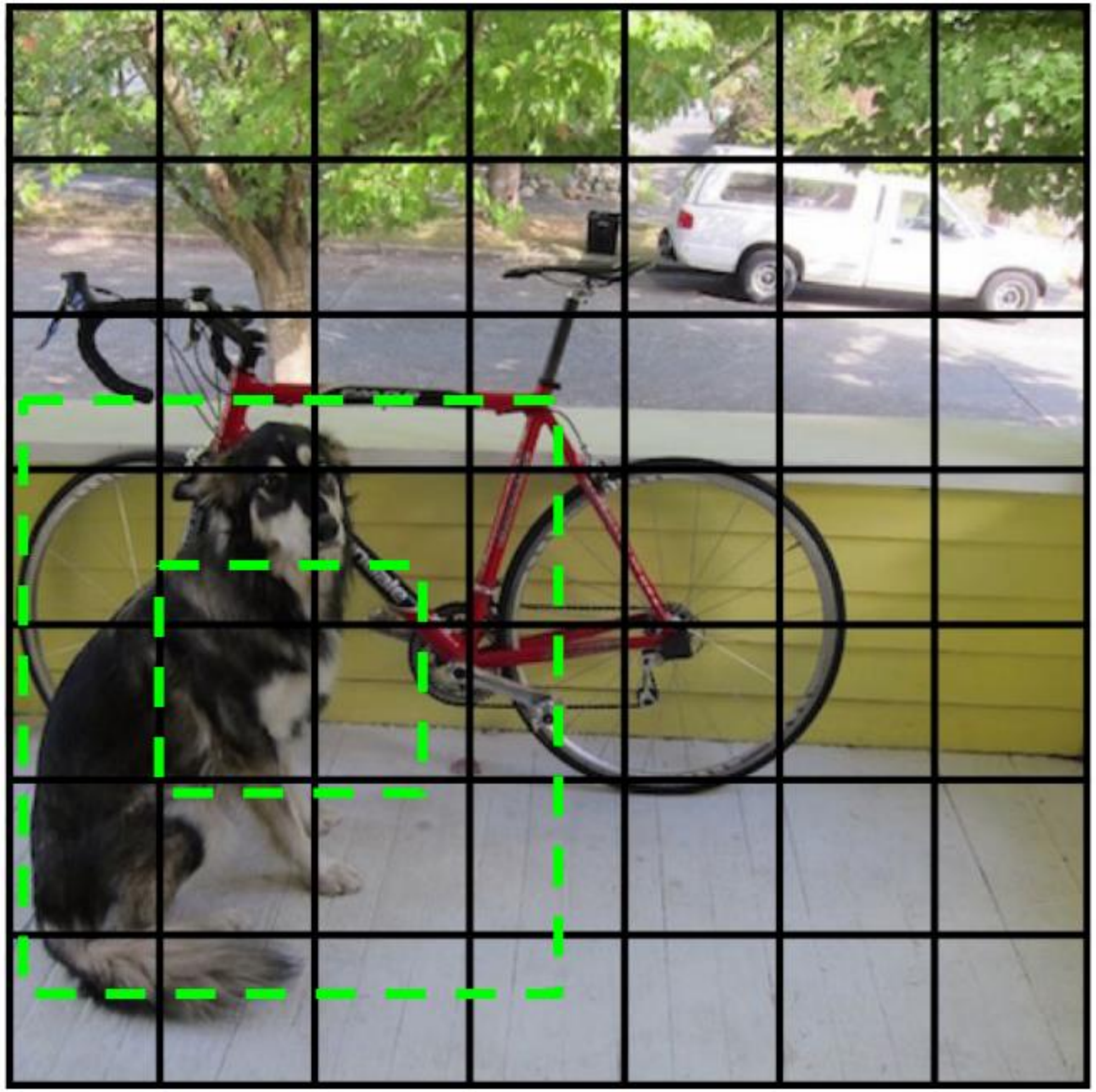


# Adjust that cell's class prediction

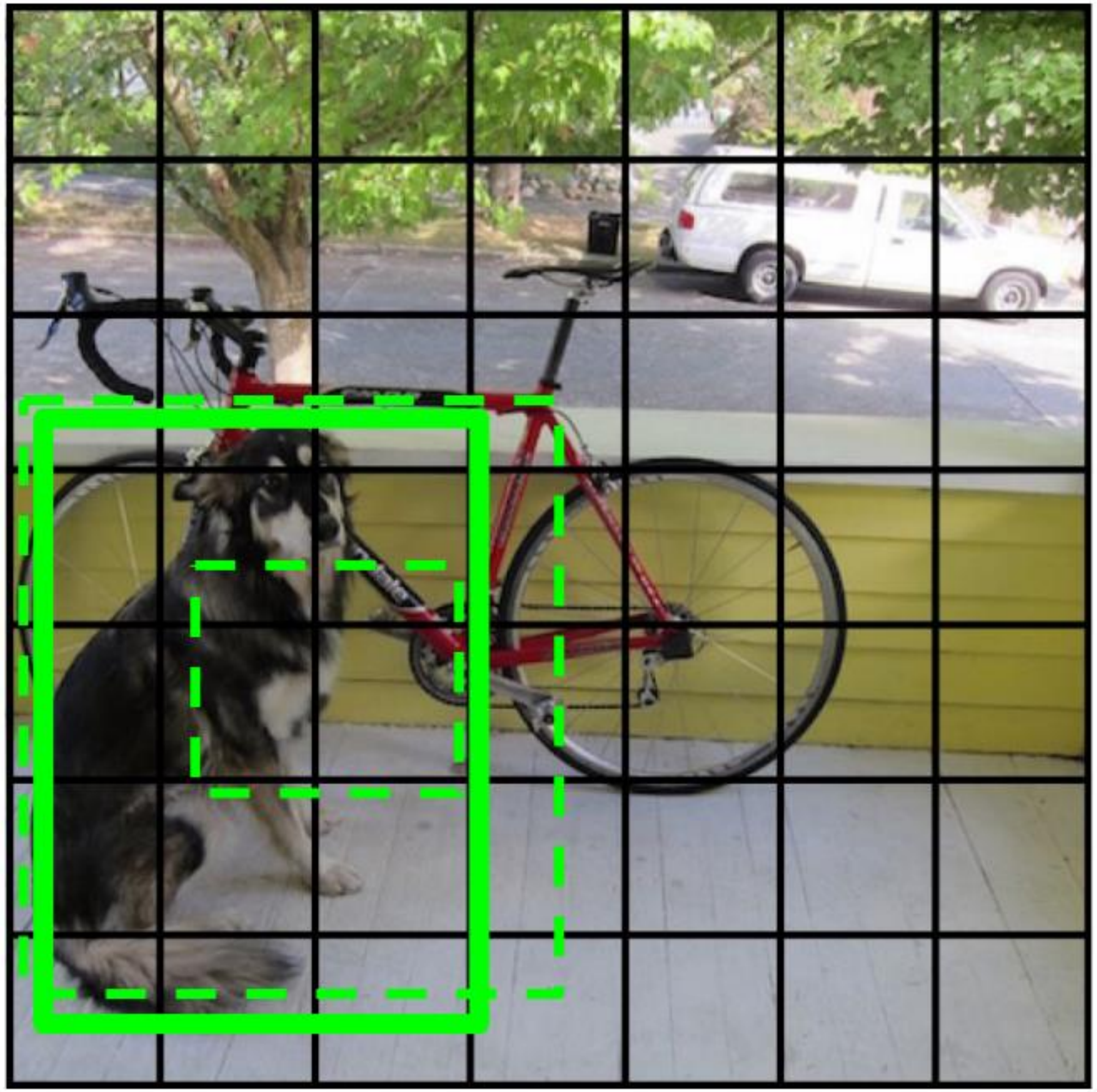
**Dog = 1**
Cat = 0
Bike = 0
...



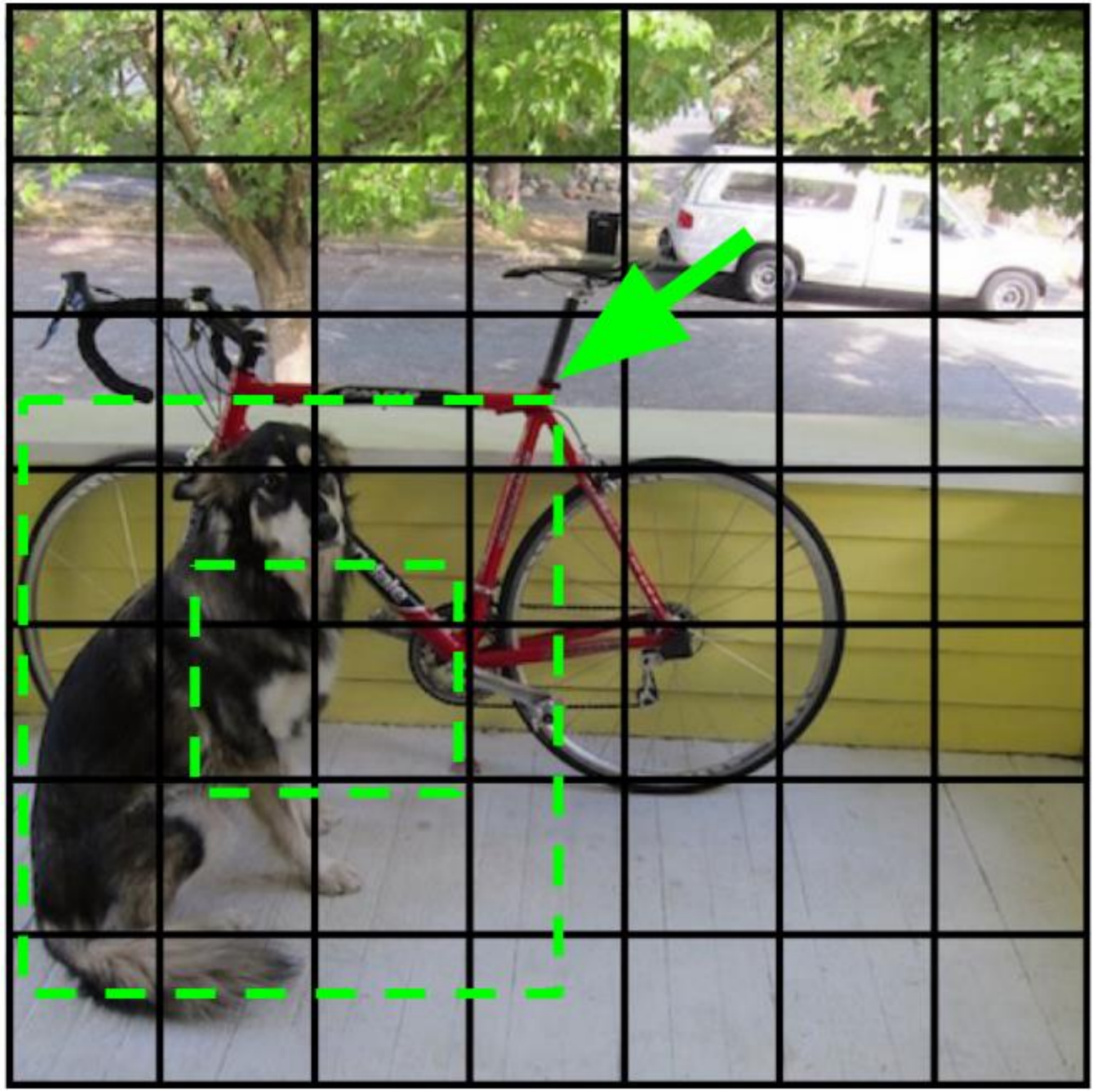
# Look at that cell's predicted boxes



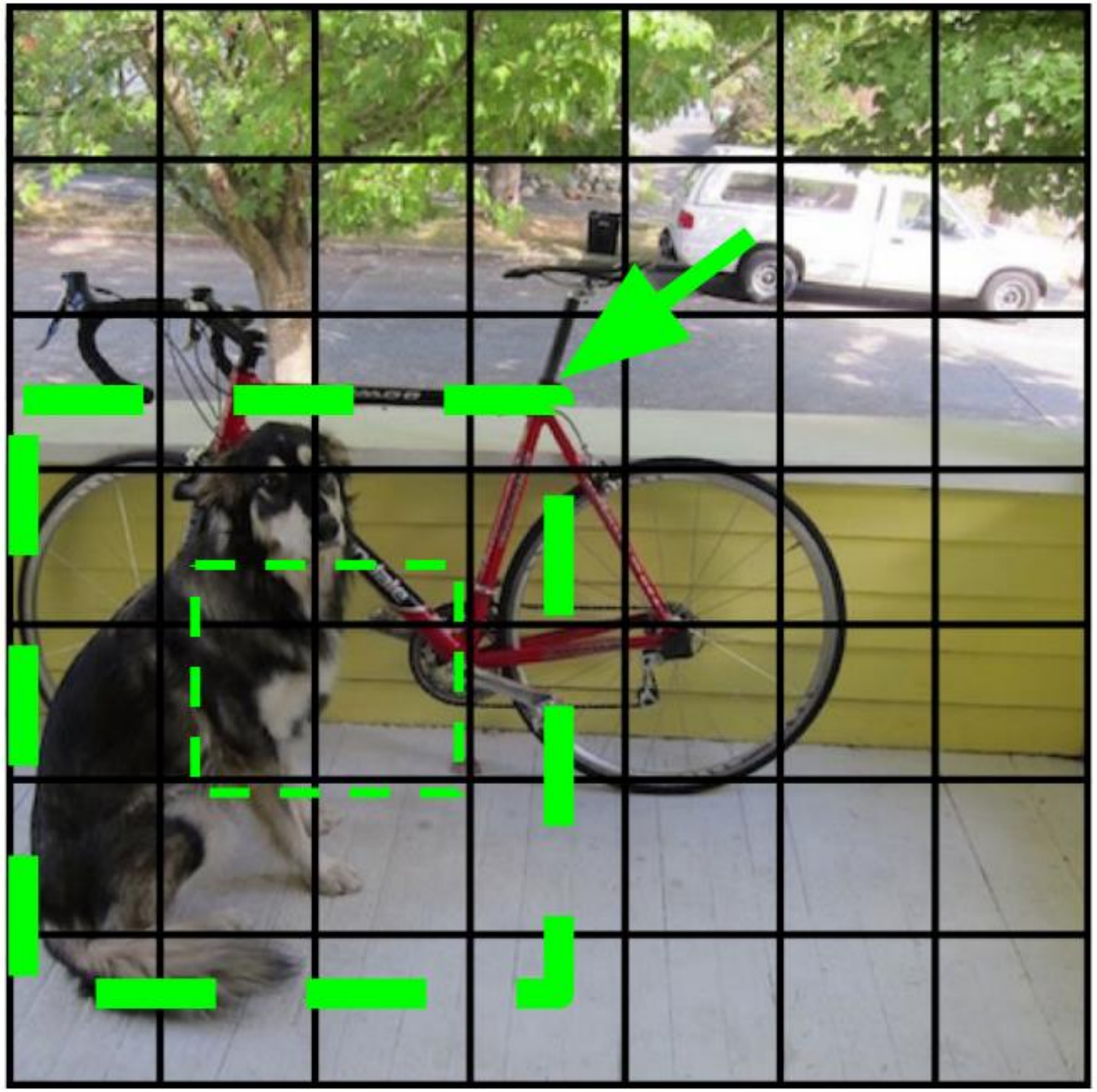
# Find the best one, adjust it, increase the confidence



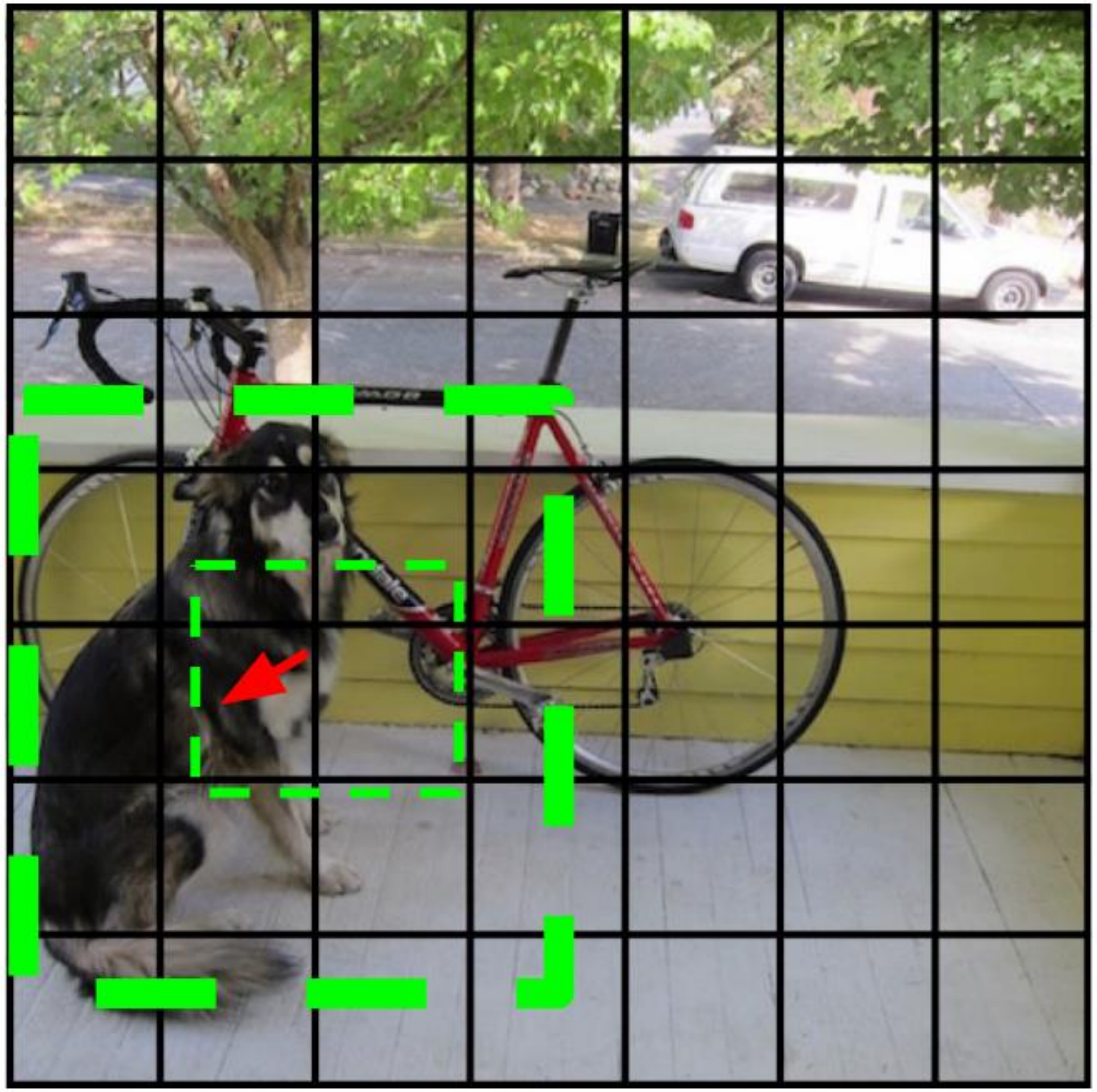
# Find the best one, adjust it, increase the confidence



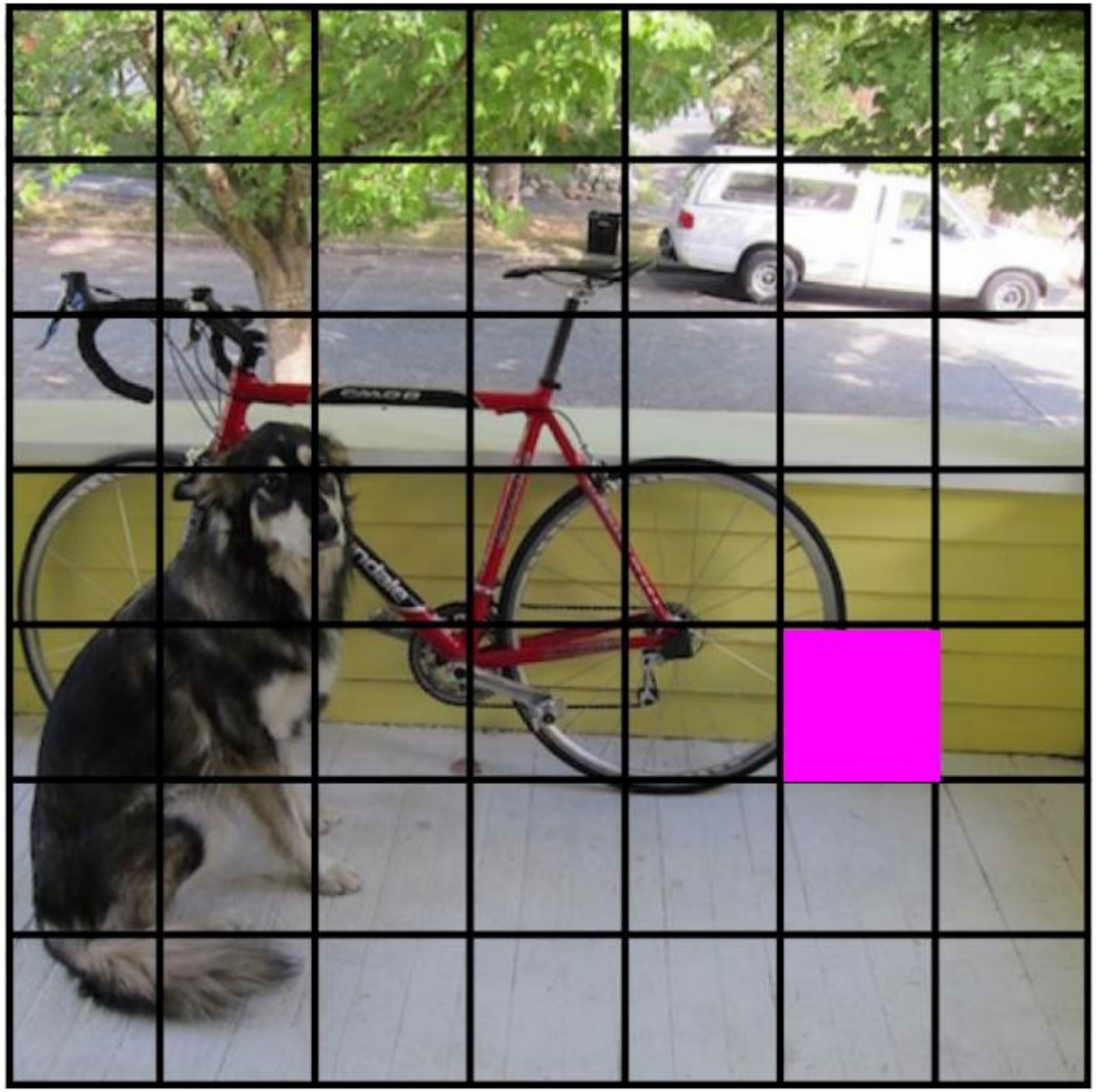
# Find the best one, adjust it, increase the confidence



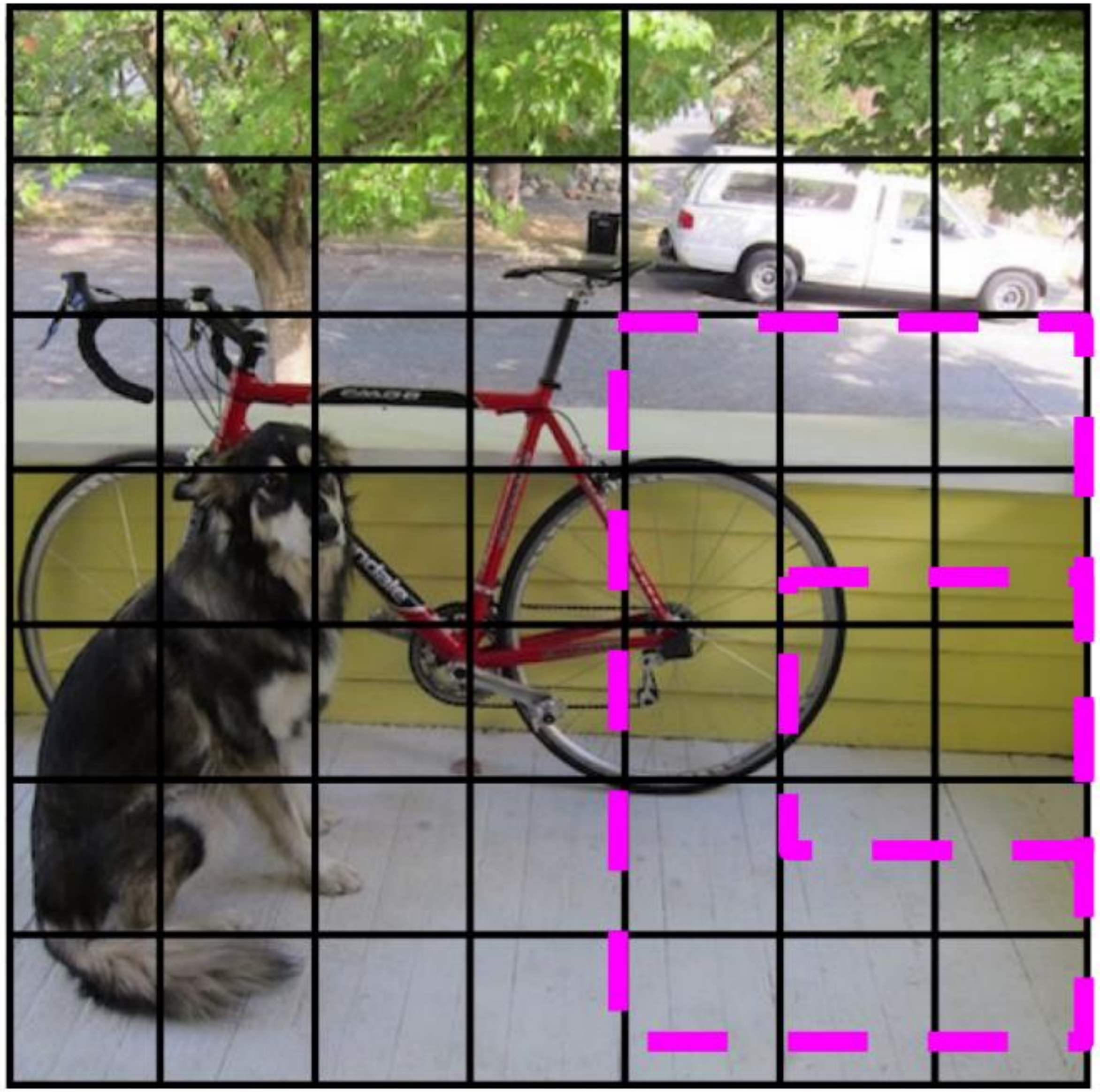
# Decrease the confidence of the other box



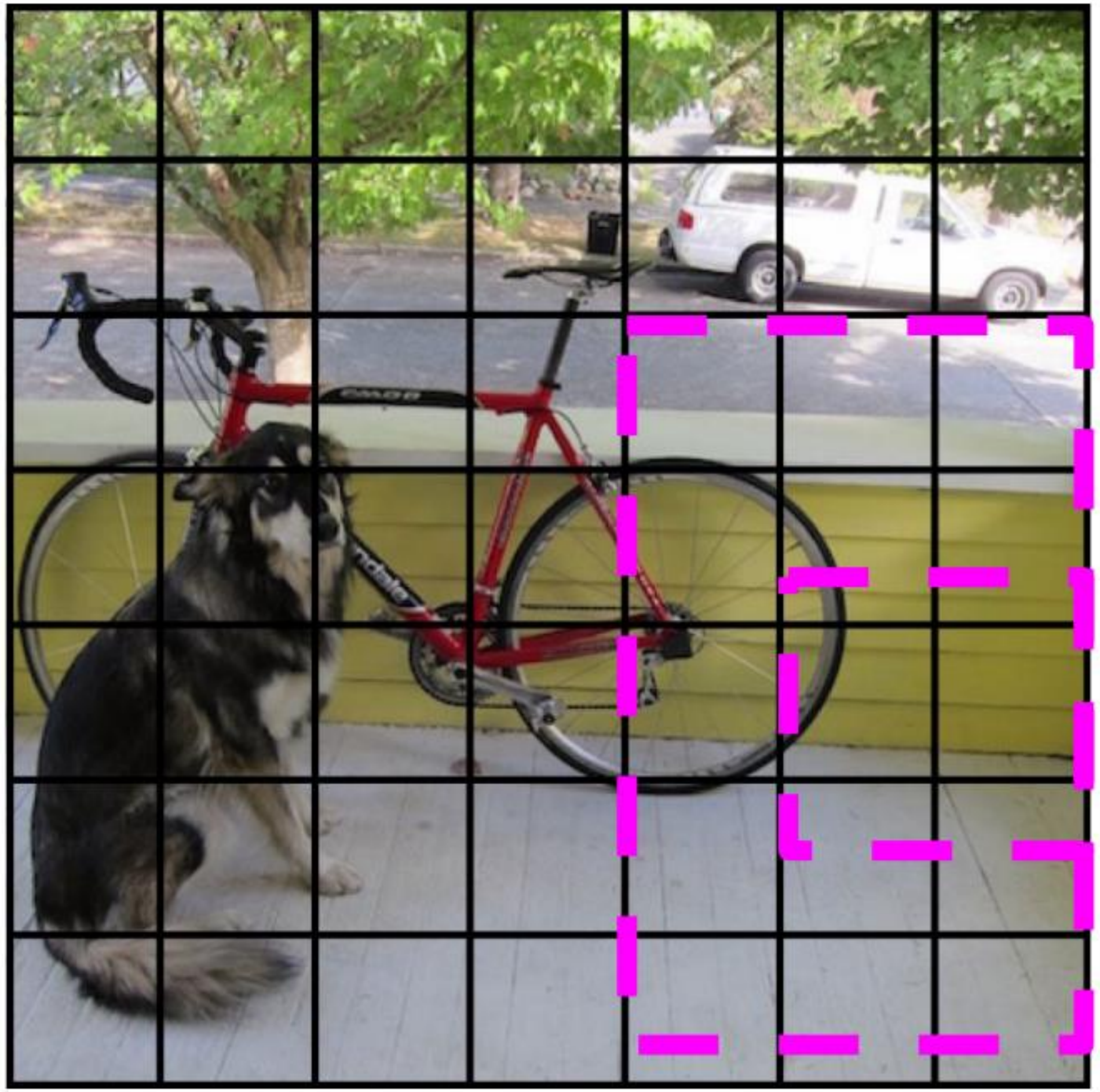
# Some cells don’t have any ground truth detections!



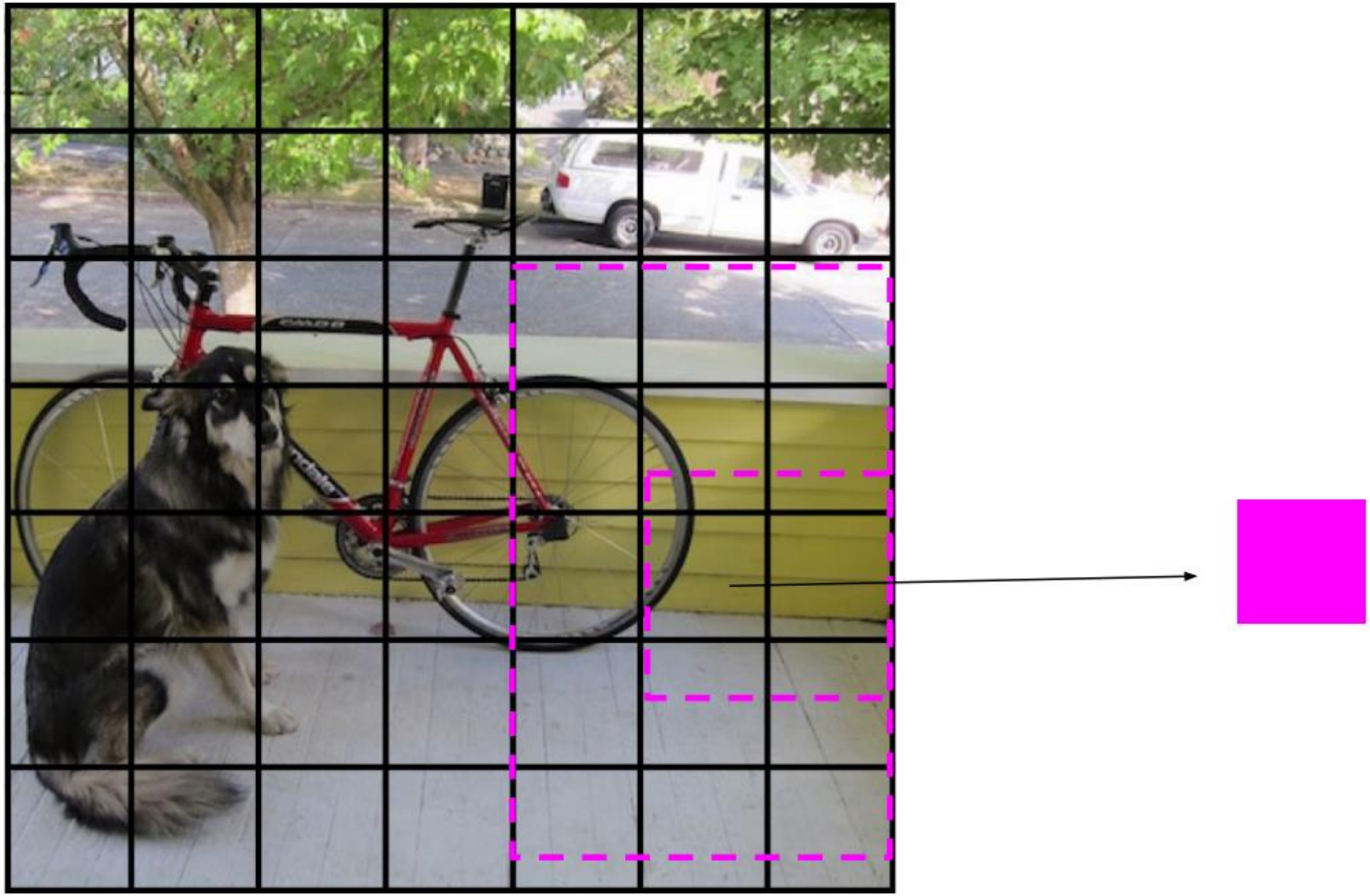
# Some cells don’t have any ground truth detections!



# Decrease the confidence of boxes boxes



# Don’t adjust the class probabilities or coordinates



# Loss Function (sum-squared error)

loss function:

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

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, $\lambda_{\text{coord}}$ and $\lambda_{\text{noobj}}$ to accomplish this. We set $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{noobj}} = .5$.

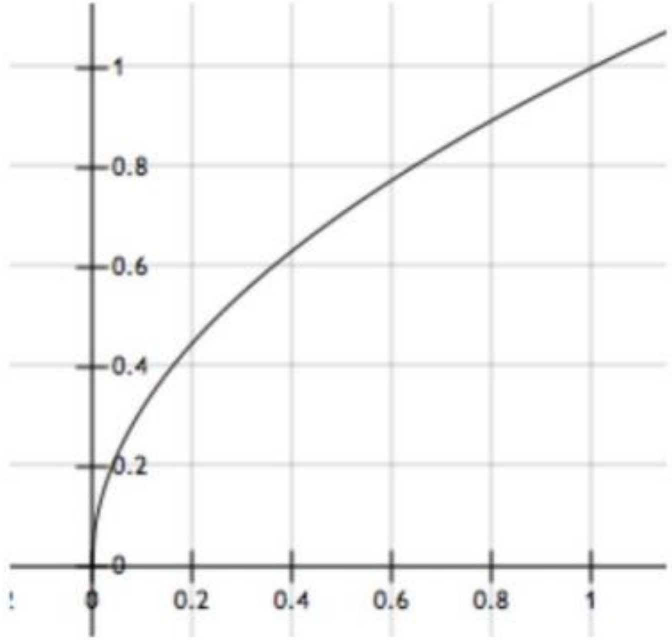
$$\lambda_{\text{coord}}=5, \ \lambda_{\text{noobj}}=0.5$$

# Loss Function (sum-squared error)

loss function:

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

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.



# Loss Function (sum-squared error)

loss function:

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

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

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

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

$$+ \sum_{i=0}^{S^2} \boxed{\mathbb{1}_{i}^{\text{obj}}} \sum_{c \in \text{classes}} \left( p_i(c) - \hat{p}_i(c) \right)^2 \quad (3)$$

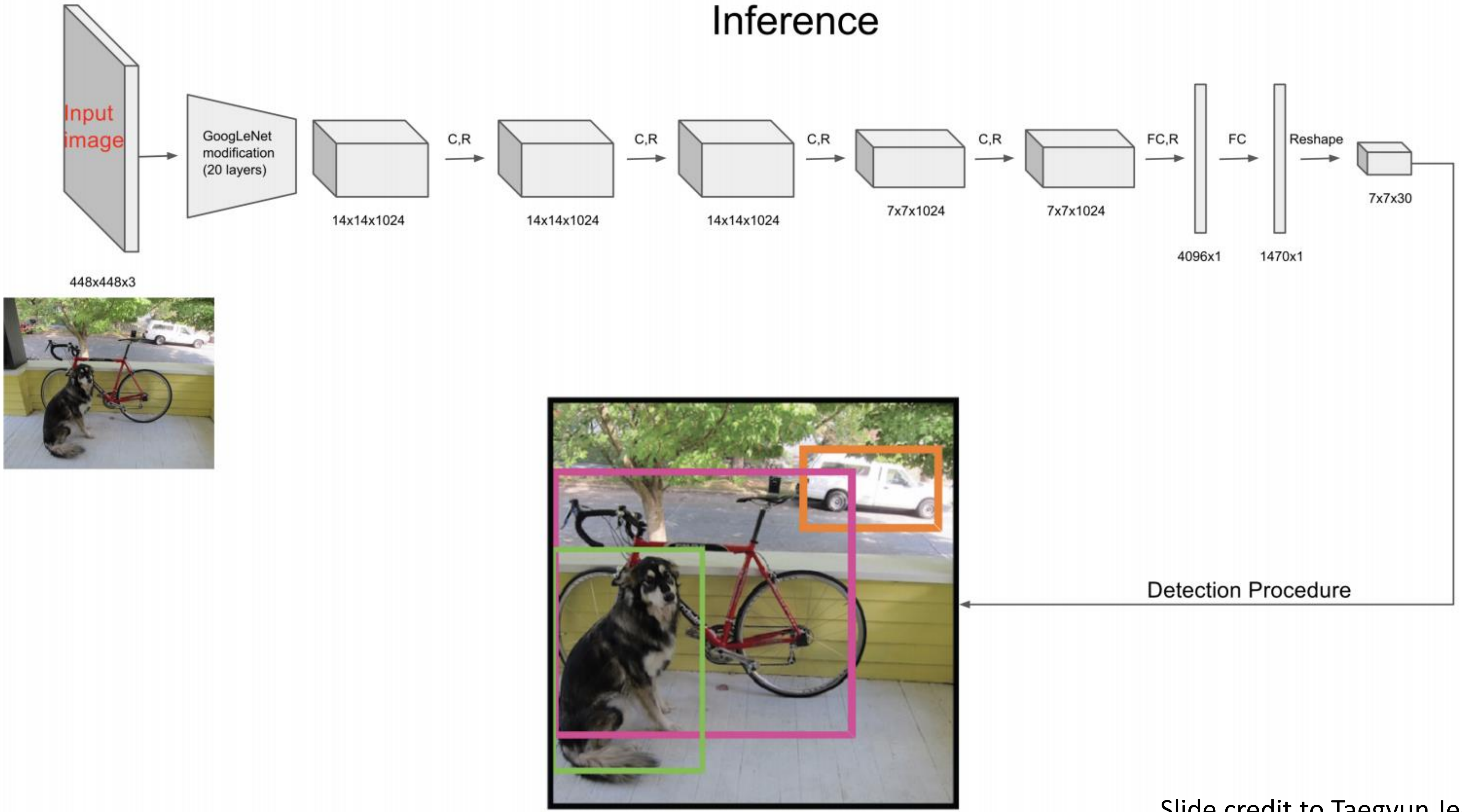
$\boxed{\mathbb{1}_{ij}^{obj}}$ **The *j*th bbox predictor** in ***cell i*** is "responsible" for that prediction

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

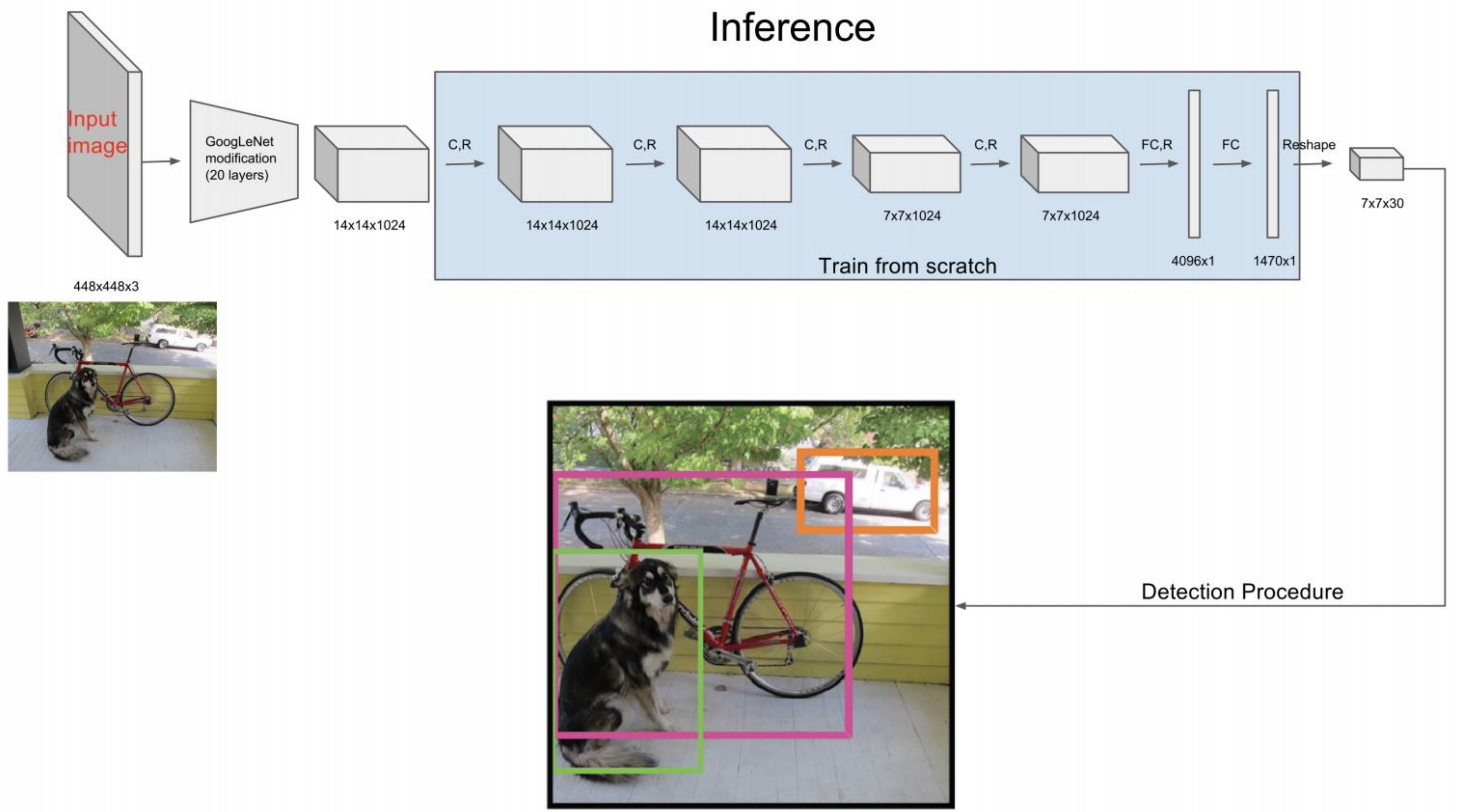
$\boxed{\mathbb{1}_{i}^{obj}}$ If object appears in ***cell i***

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).

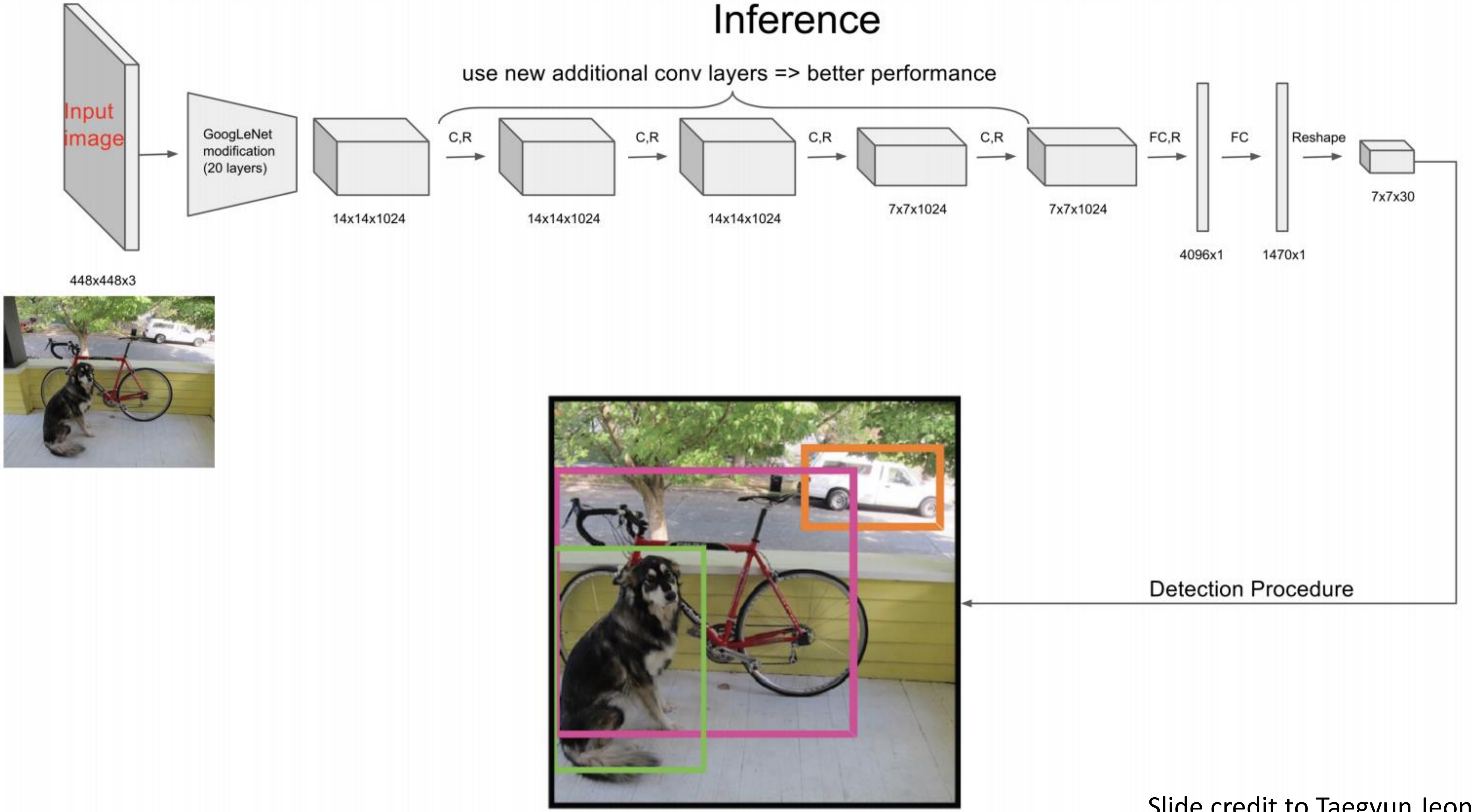
# Inference




Slide credit to Taegyun Jeon

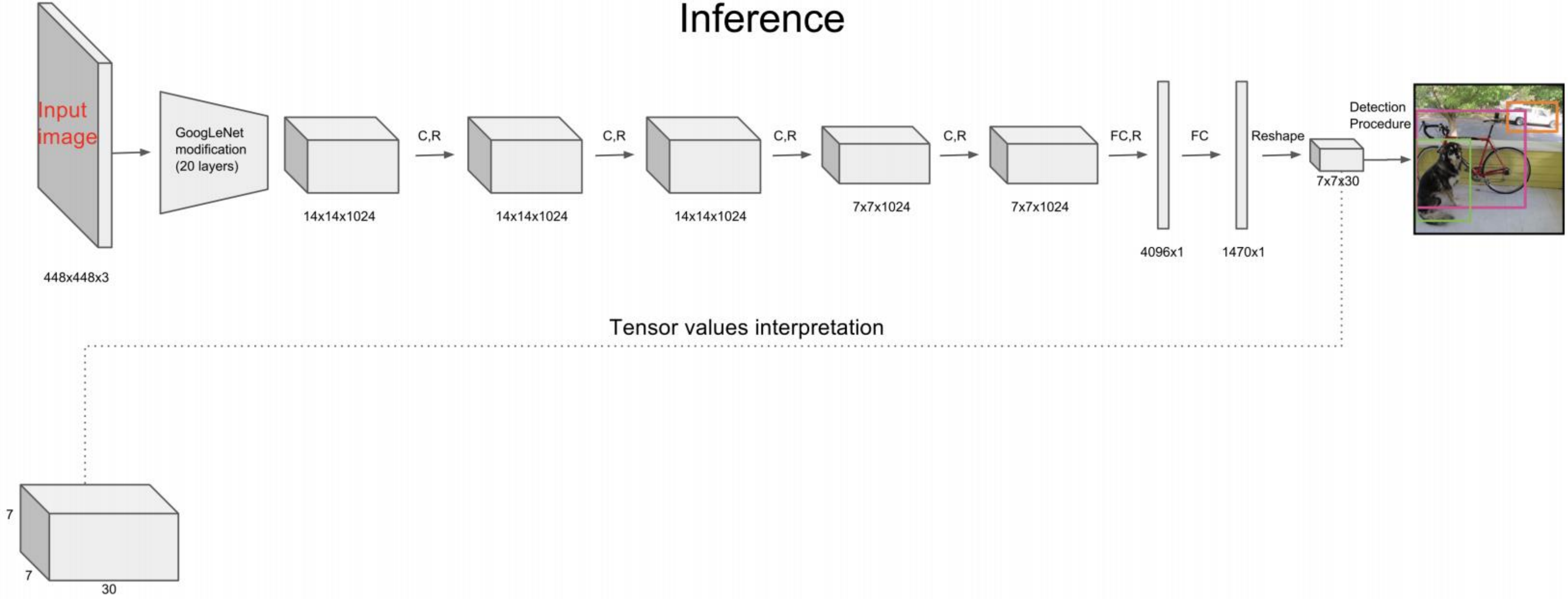

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Train from scratch
Detection Procedure

# Inference

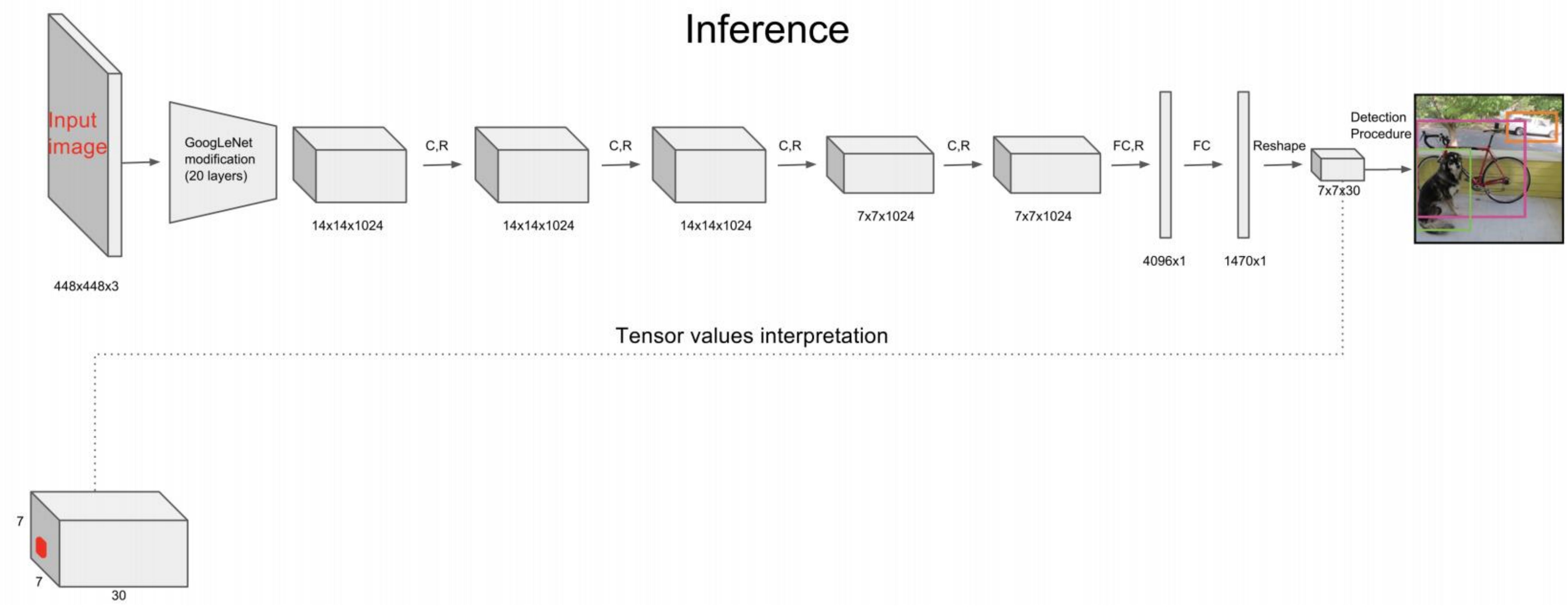


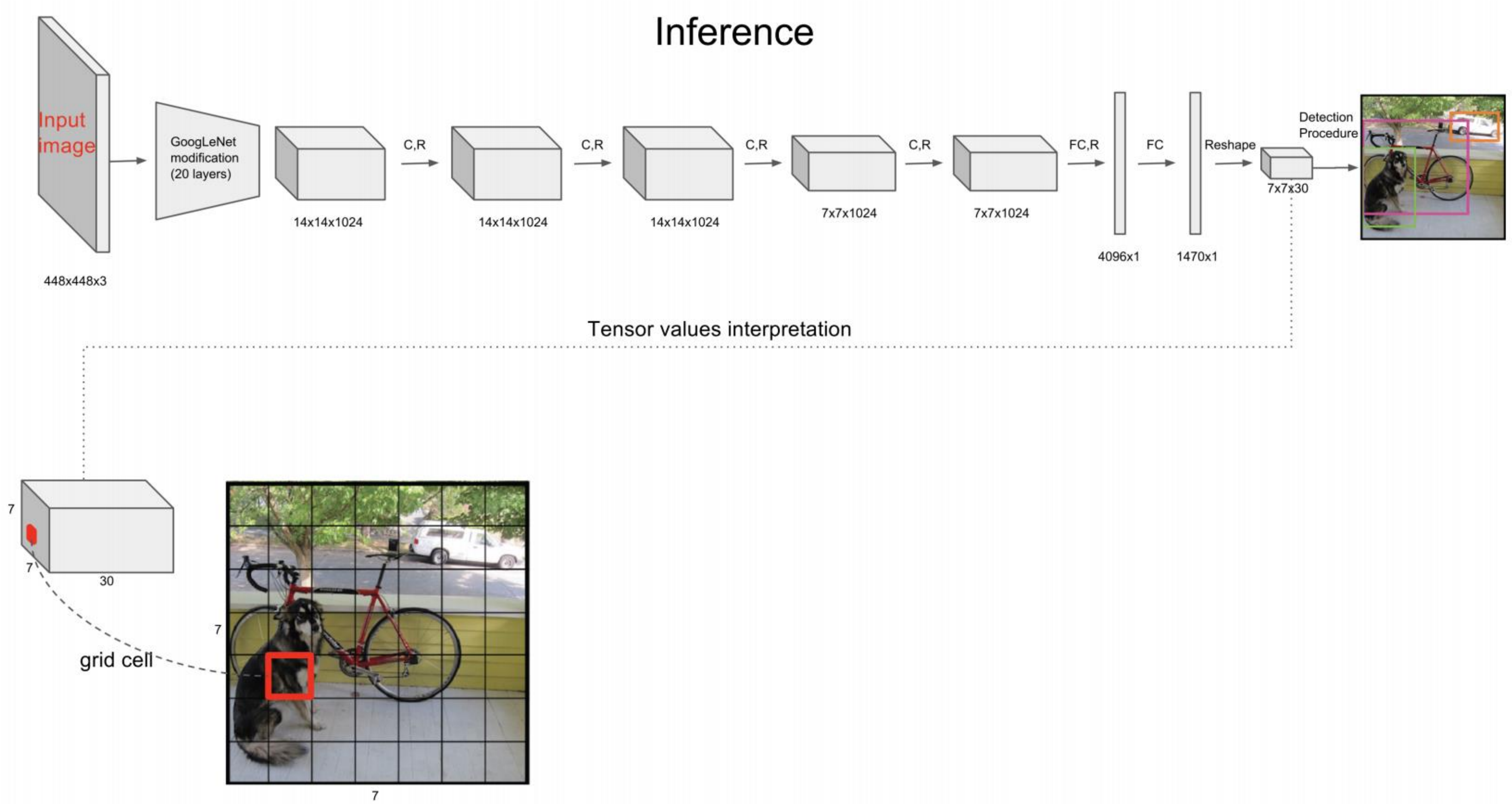

Slide credit to Taegyun Jeon

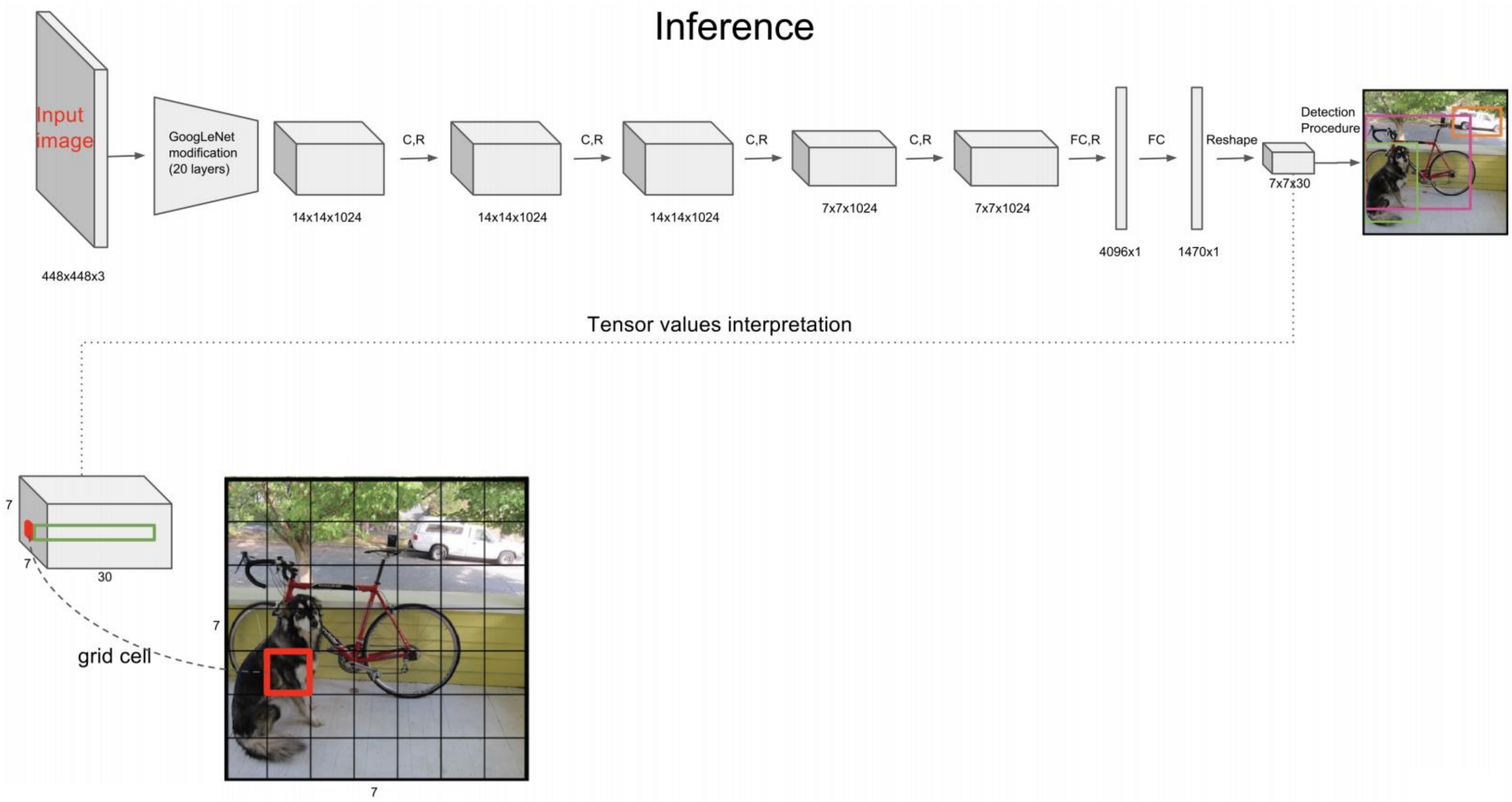
# Inference

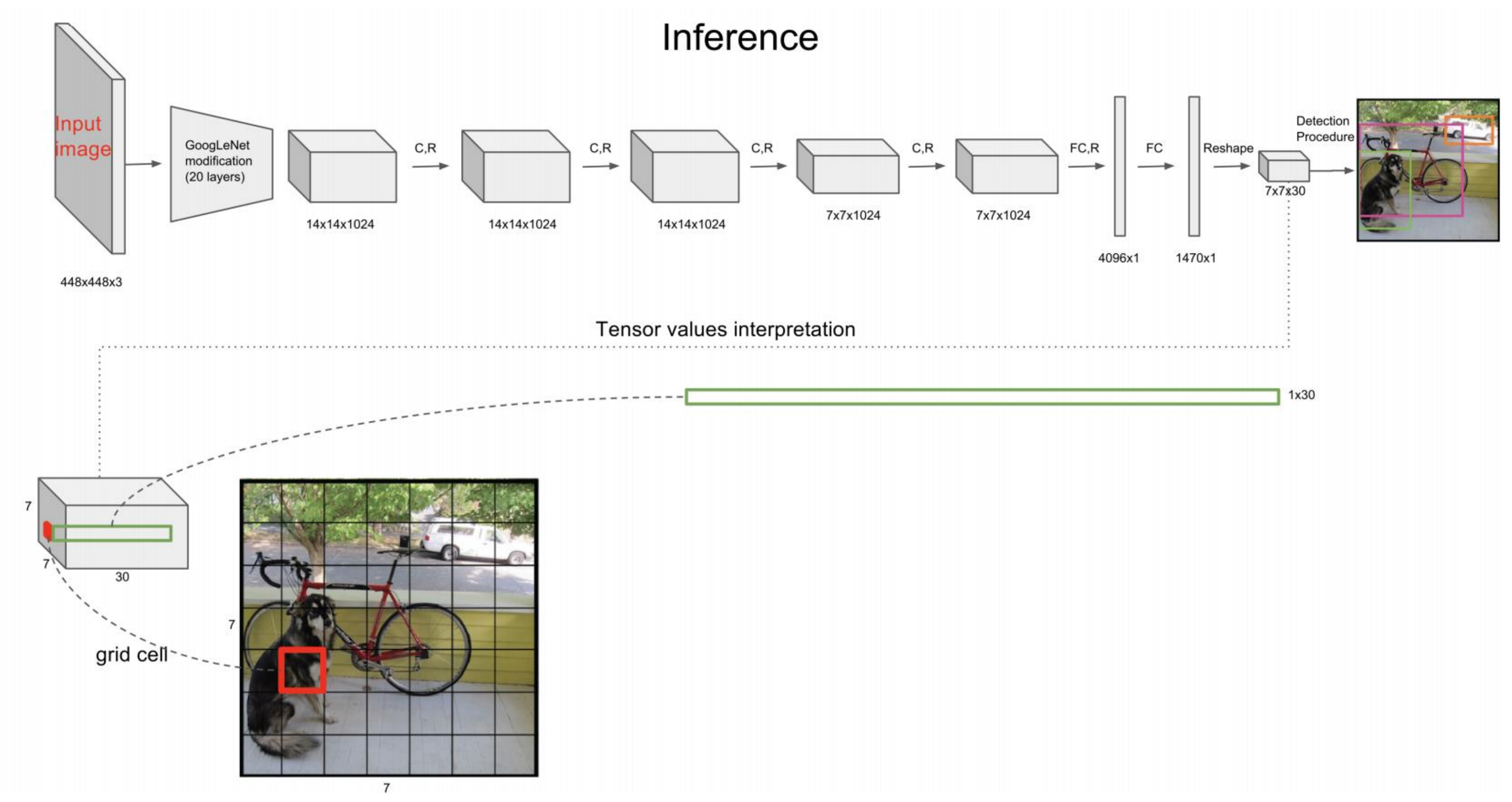



Slide credit to Taegyun Jeon

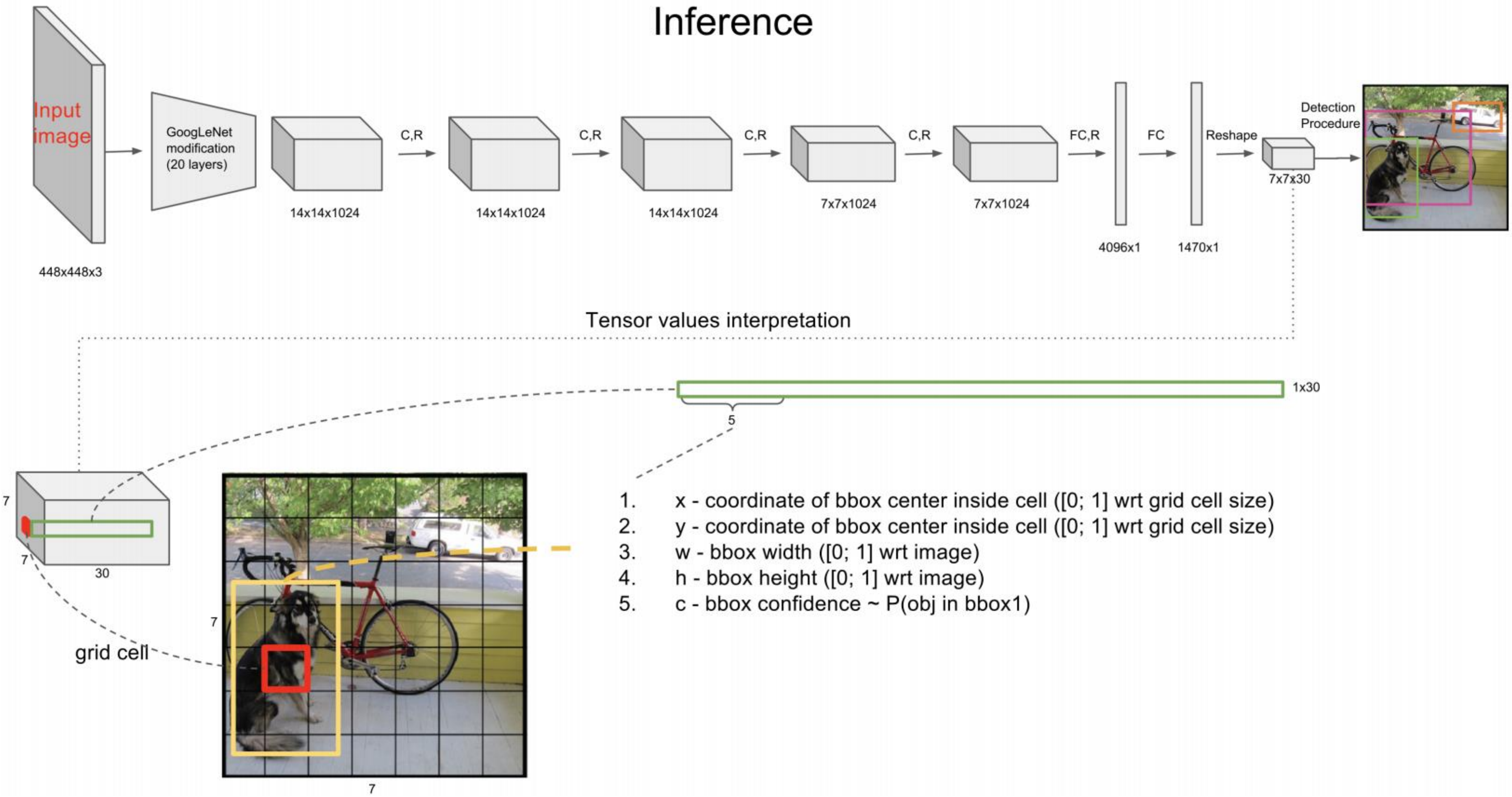

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
7
7
30


Inference
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
Tensor values interpretation
7
7
30
grid cell
7
7

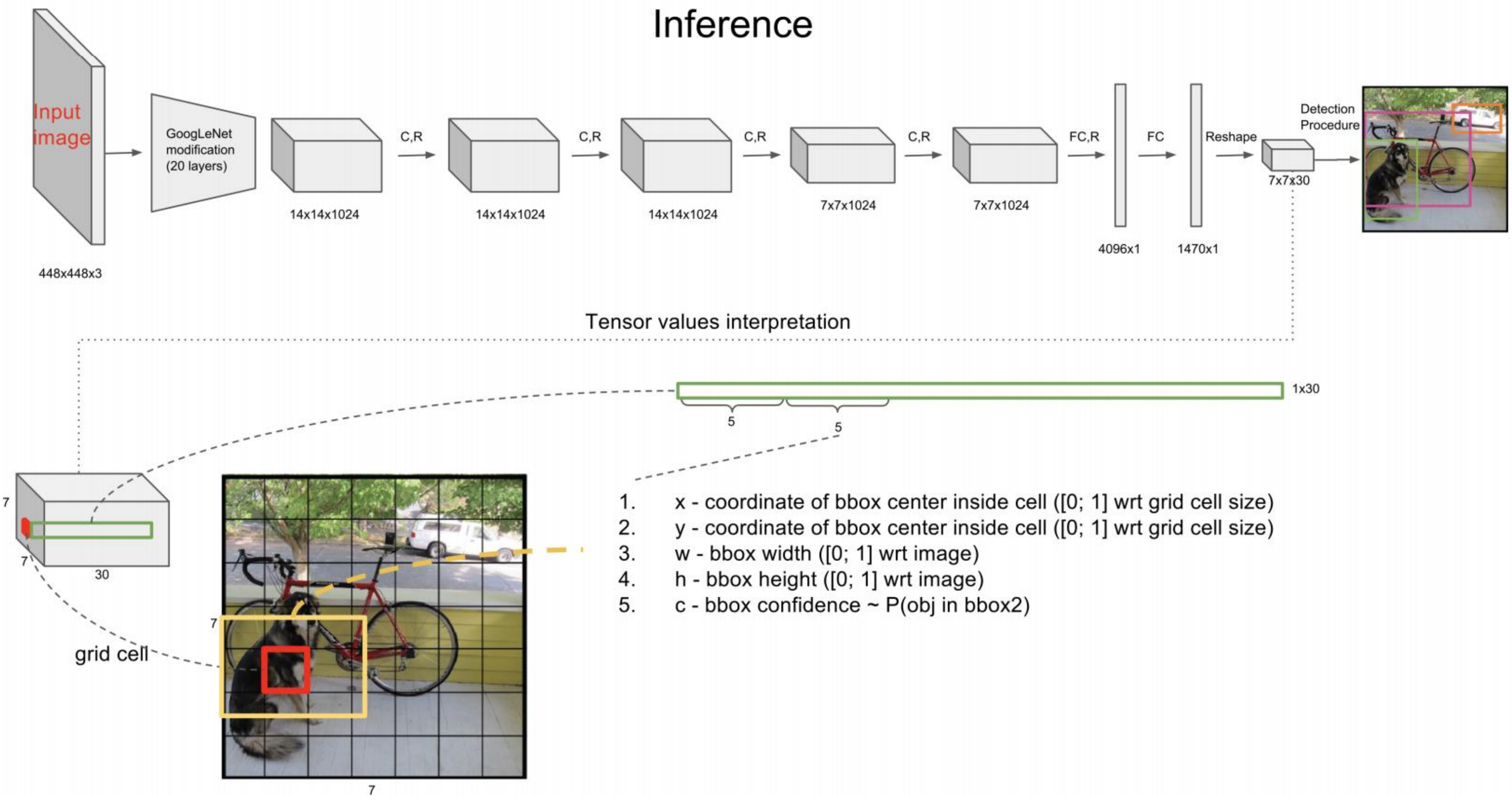

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
7
7
30
grid cell
7
7



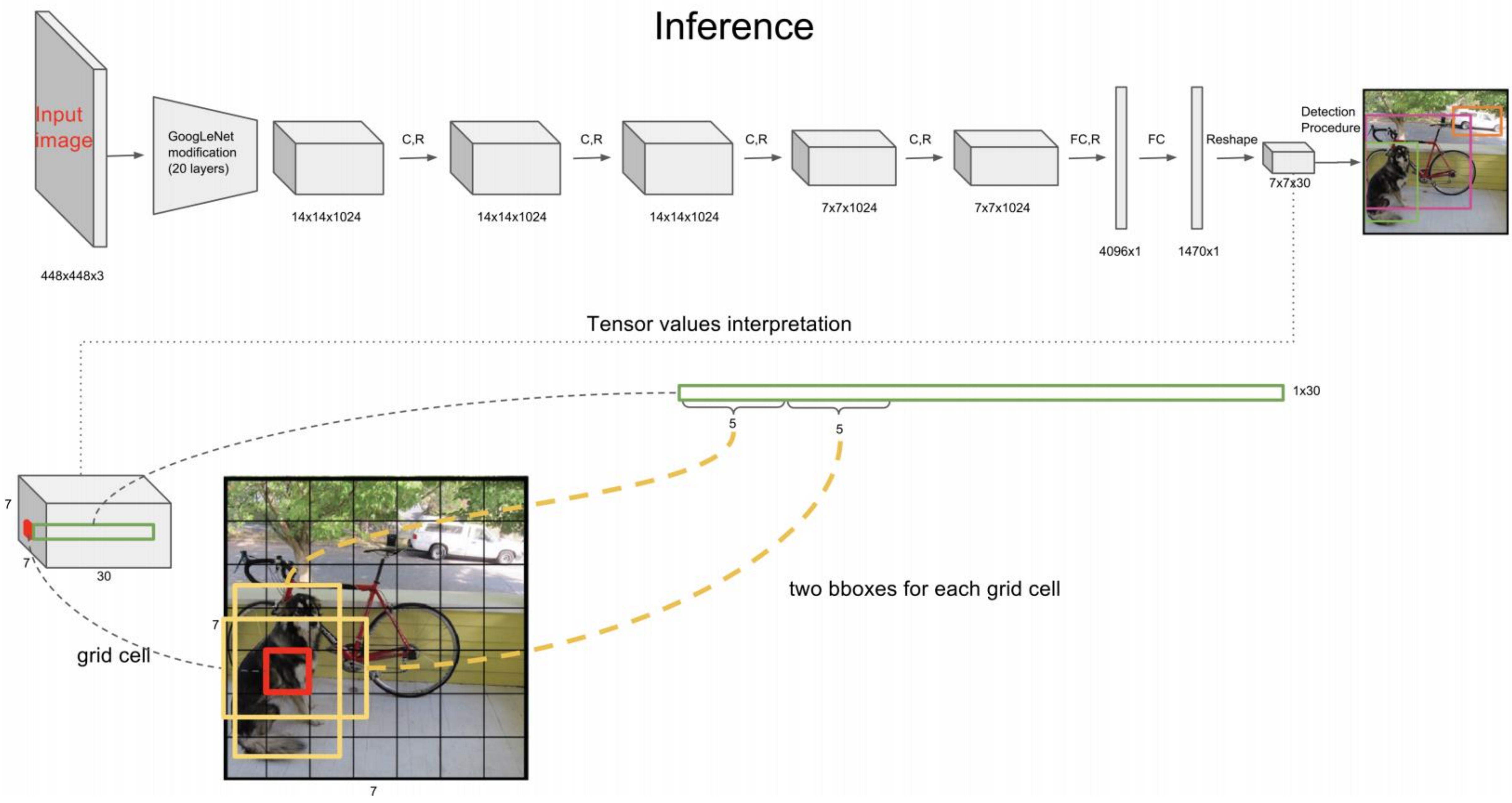
Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
1x30
7
7
30
grid cell
7
7

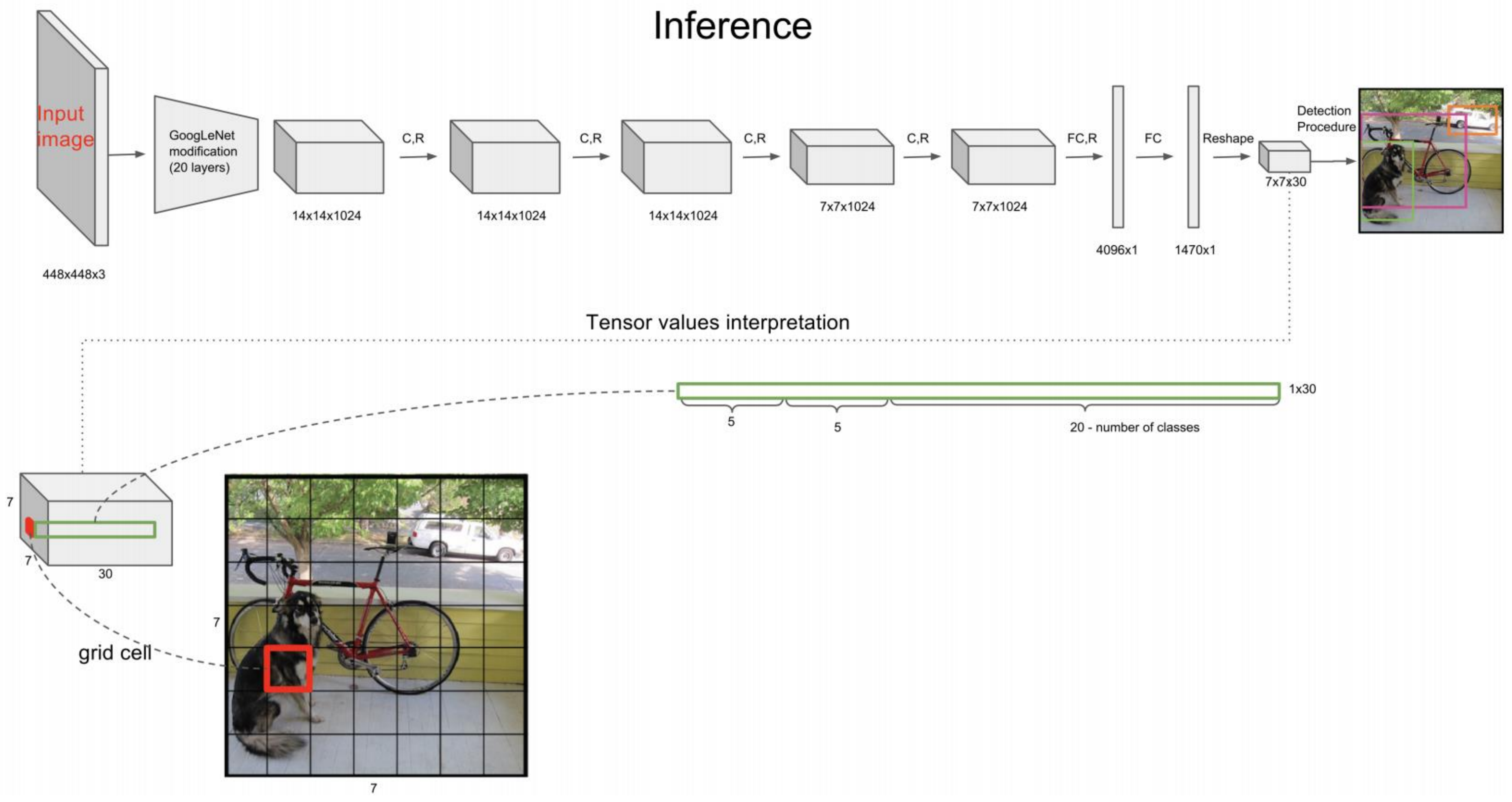


Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
1x30
5
7
7
30
grid cell
7
7
1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox1)

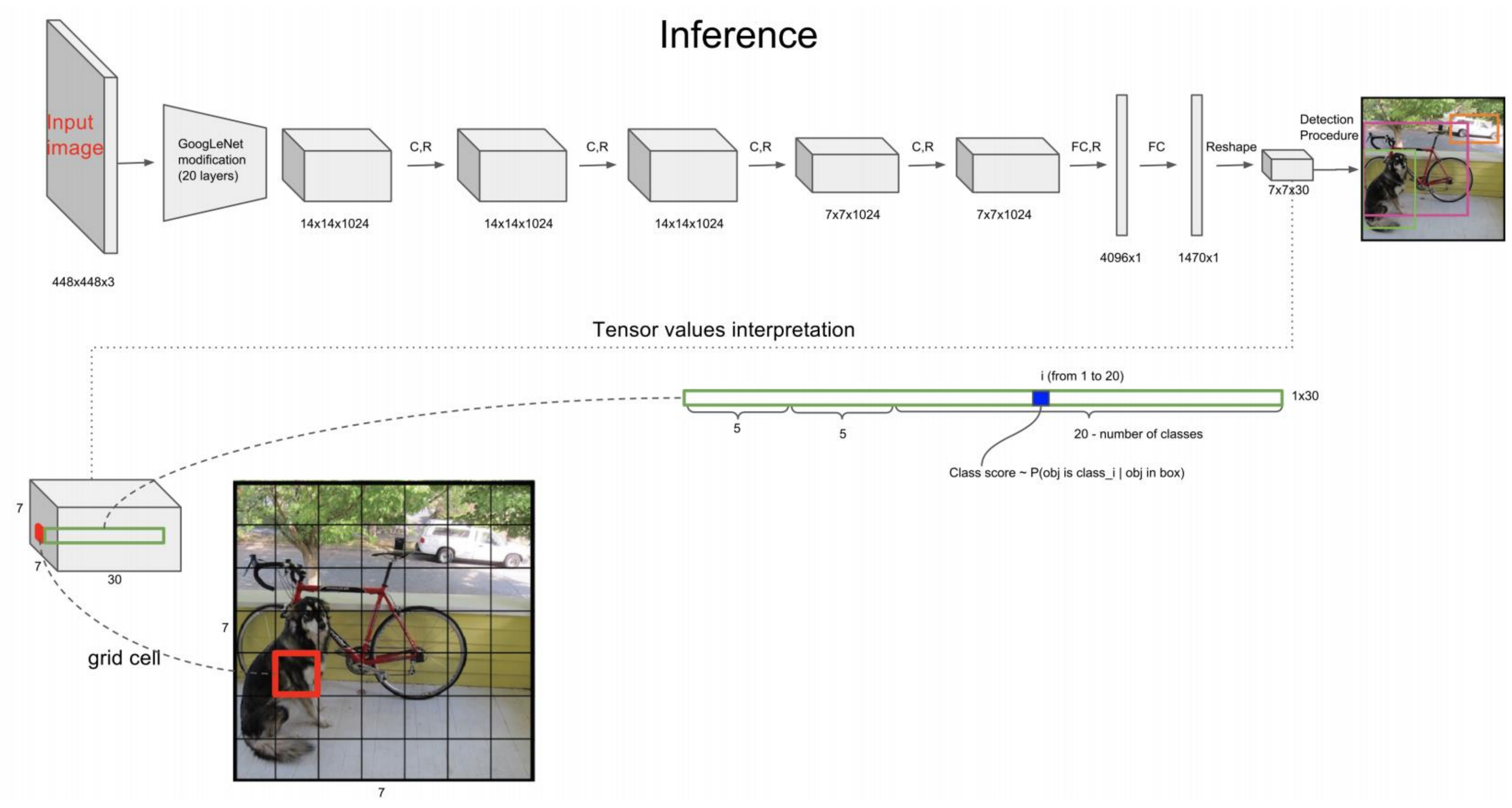


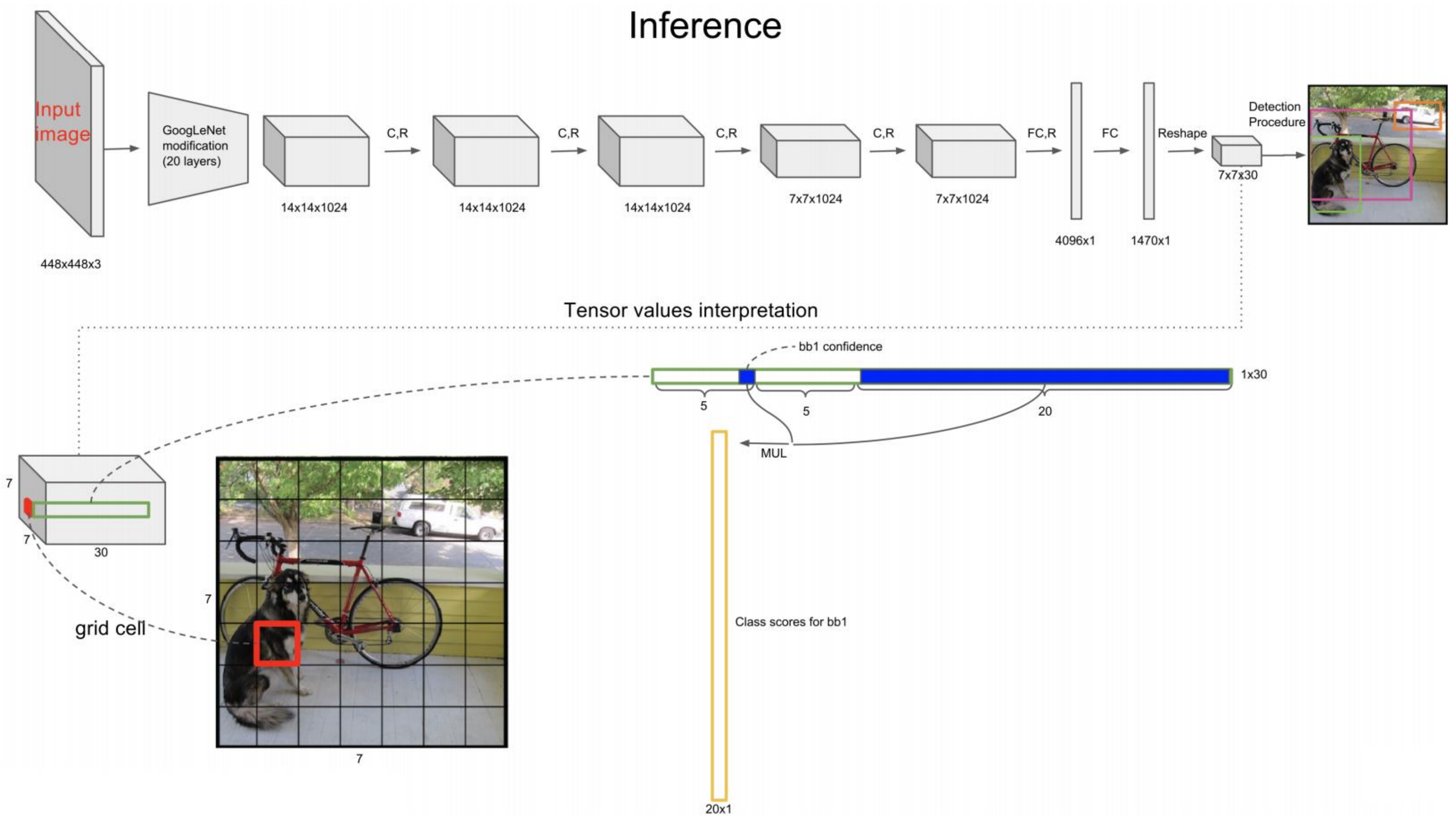
Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
1x30
5
5
7
7
30
grid cell
7
7
1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox2)

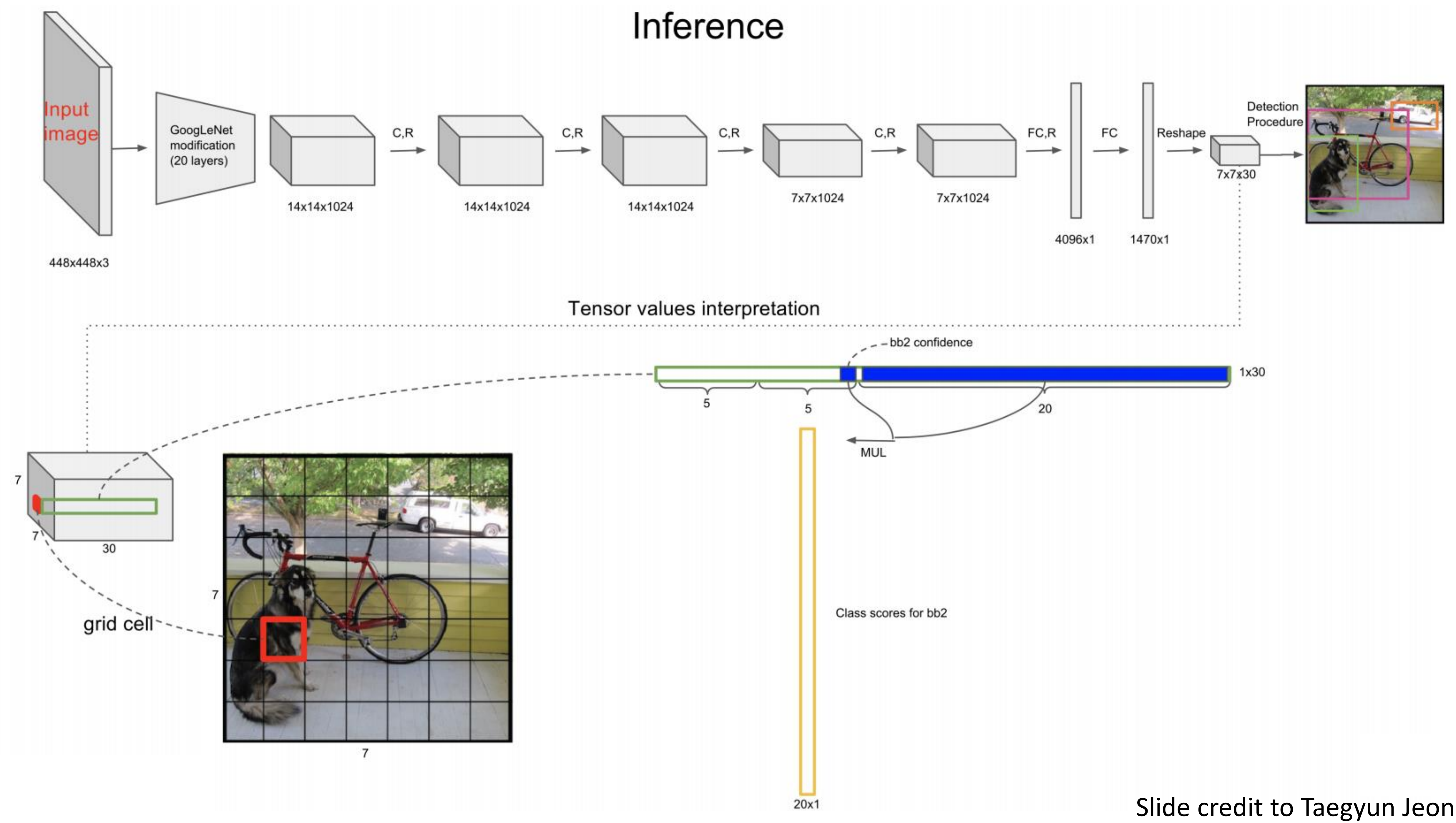

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
1x30
5
5
7
7
30
grid cell
7
7
two bboxes for each grid cell



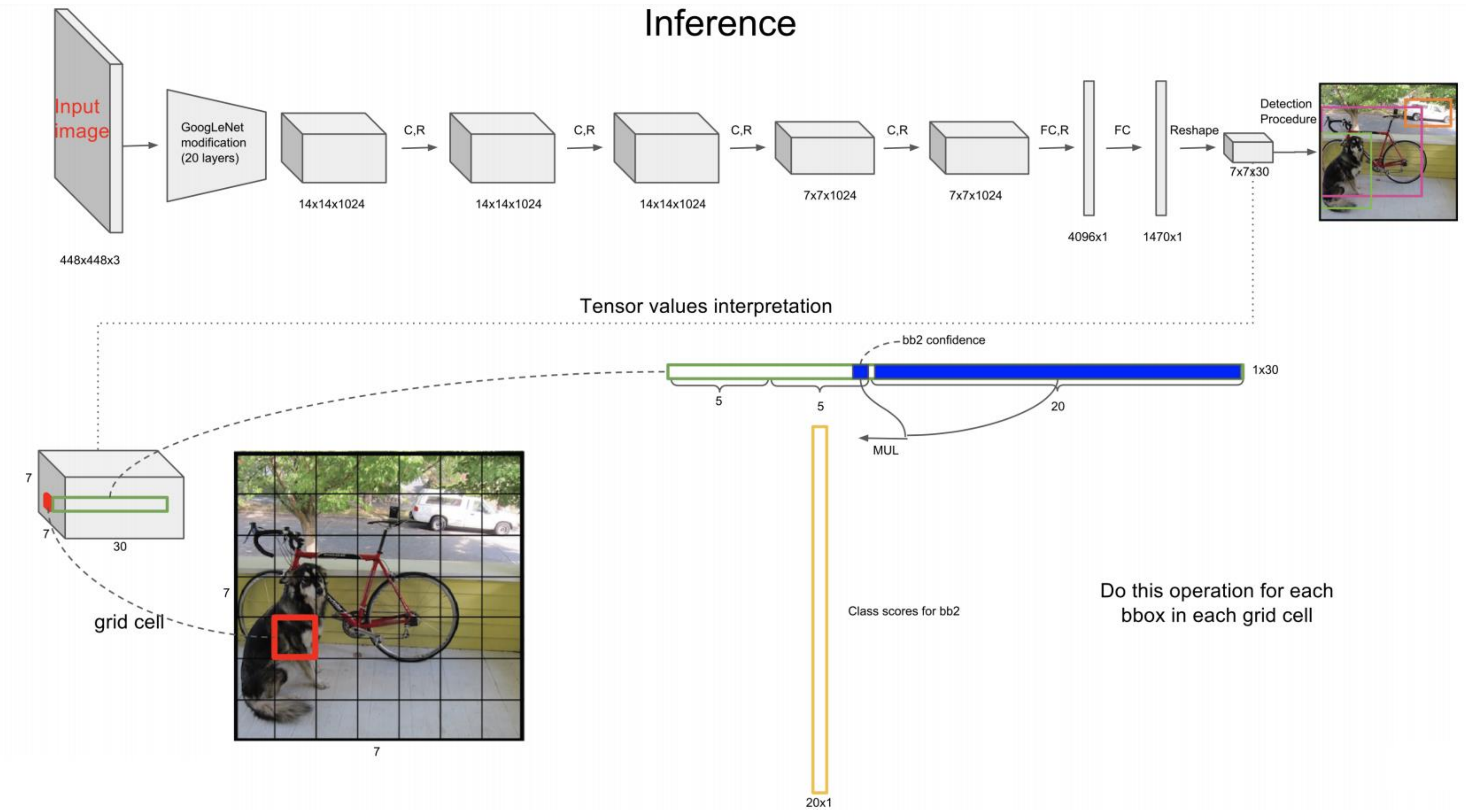
Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
1x30
5
5
20 - number of classes
7
7
30
grid cell
7
7


Inference
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
Tensor values interpretation
i (from 1 to 20)
1x30
5
5
20 - number of classes
Class score ~ P(obj is class_i | obj in box)
7
7
30
grid cell
7
7

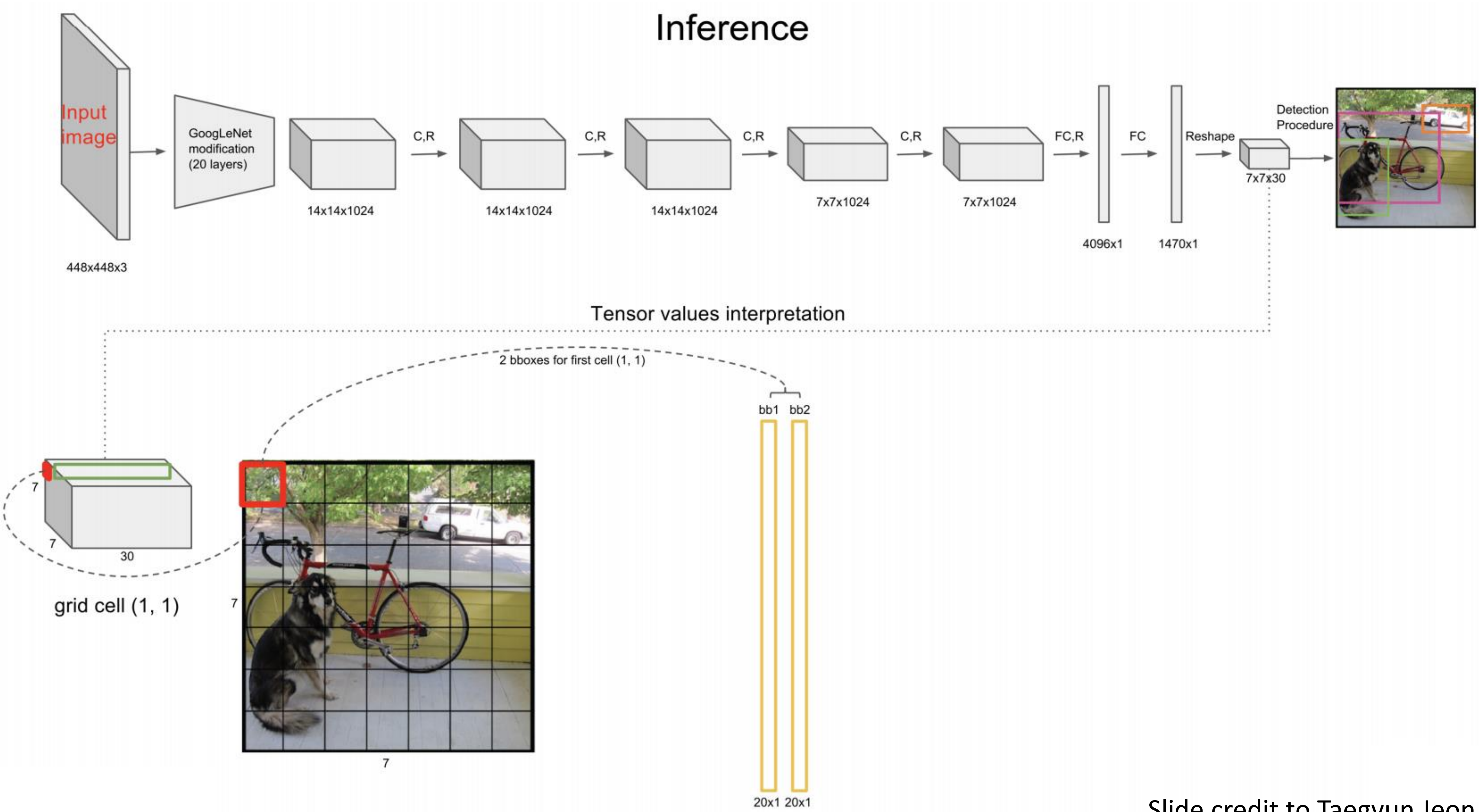

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
bb1 confidence
1x30
5
5
20
MUL
Class scores for bb1
20x1
7
7
30
grid cell
7
7



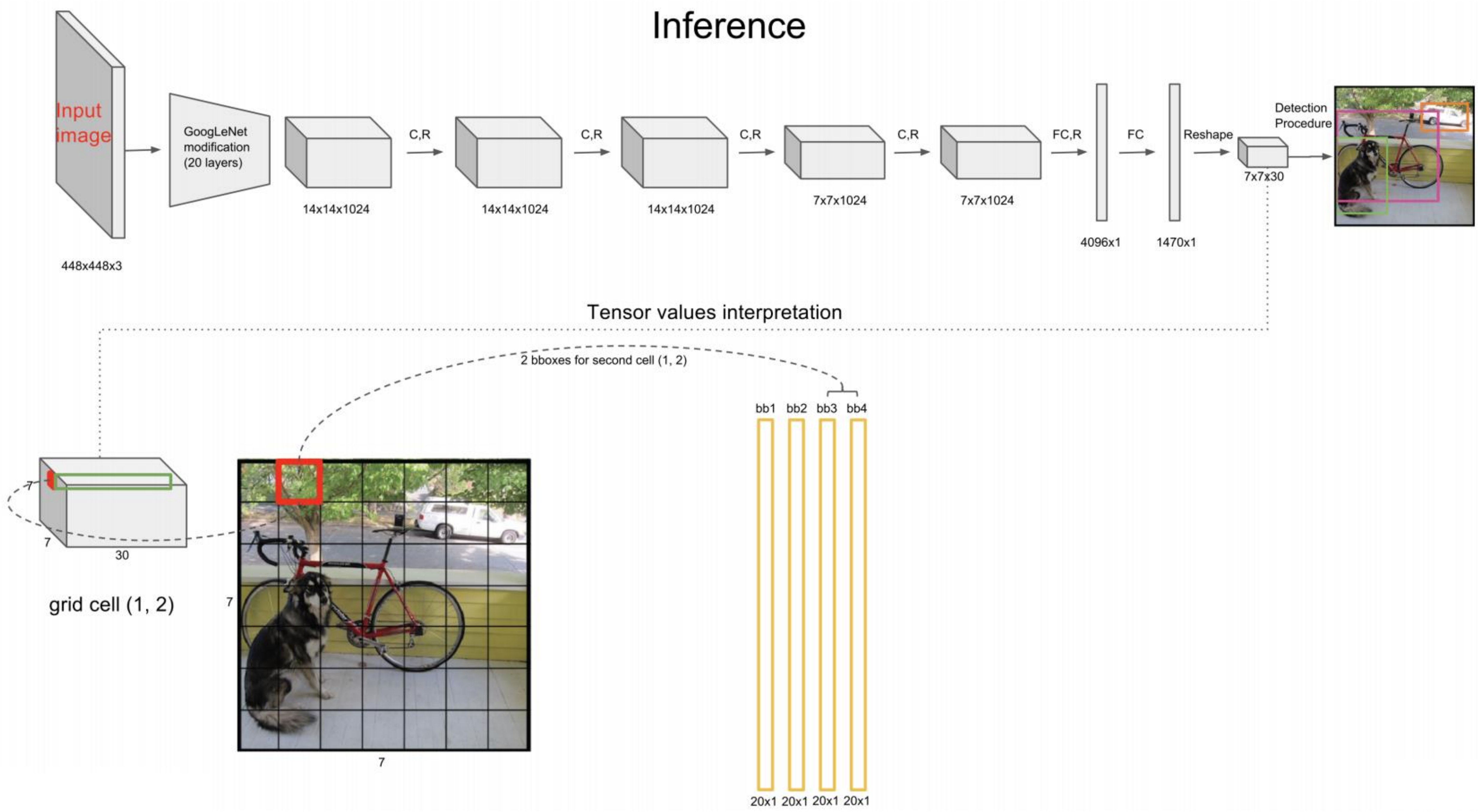

Slide credit to Taegyun Jeon

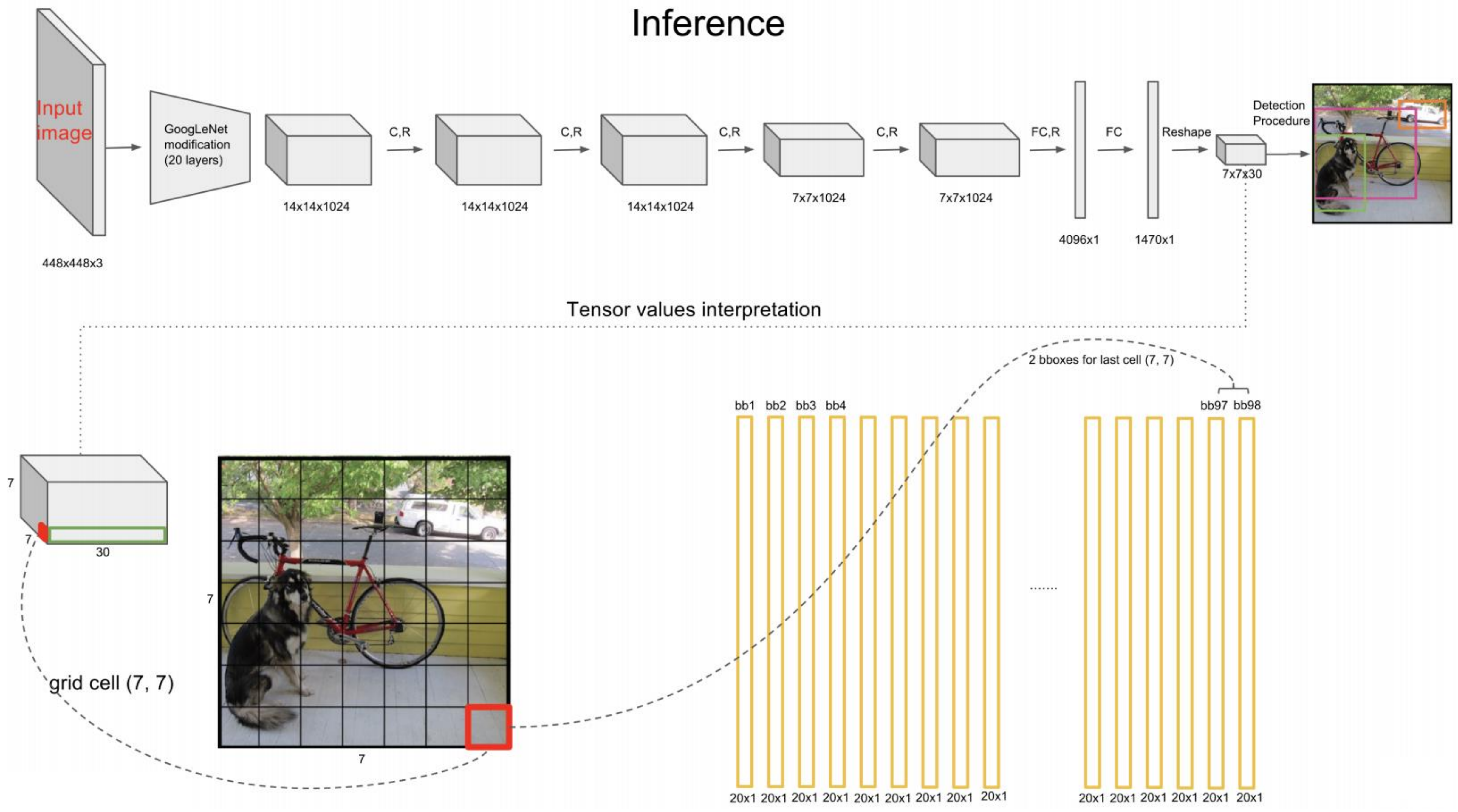


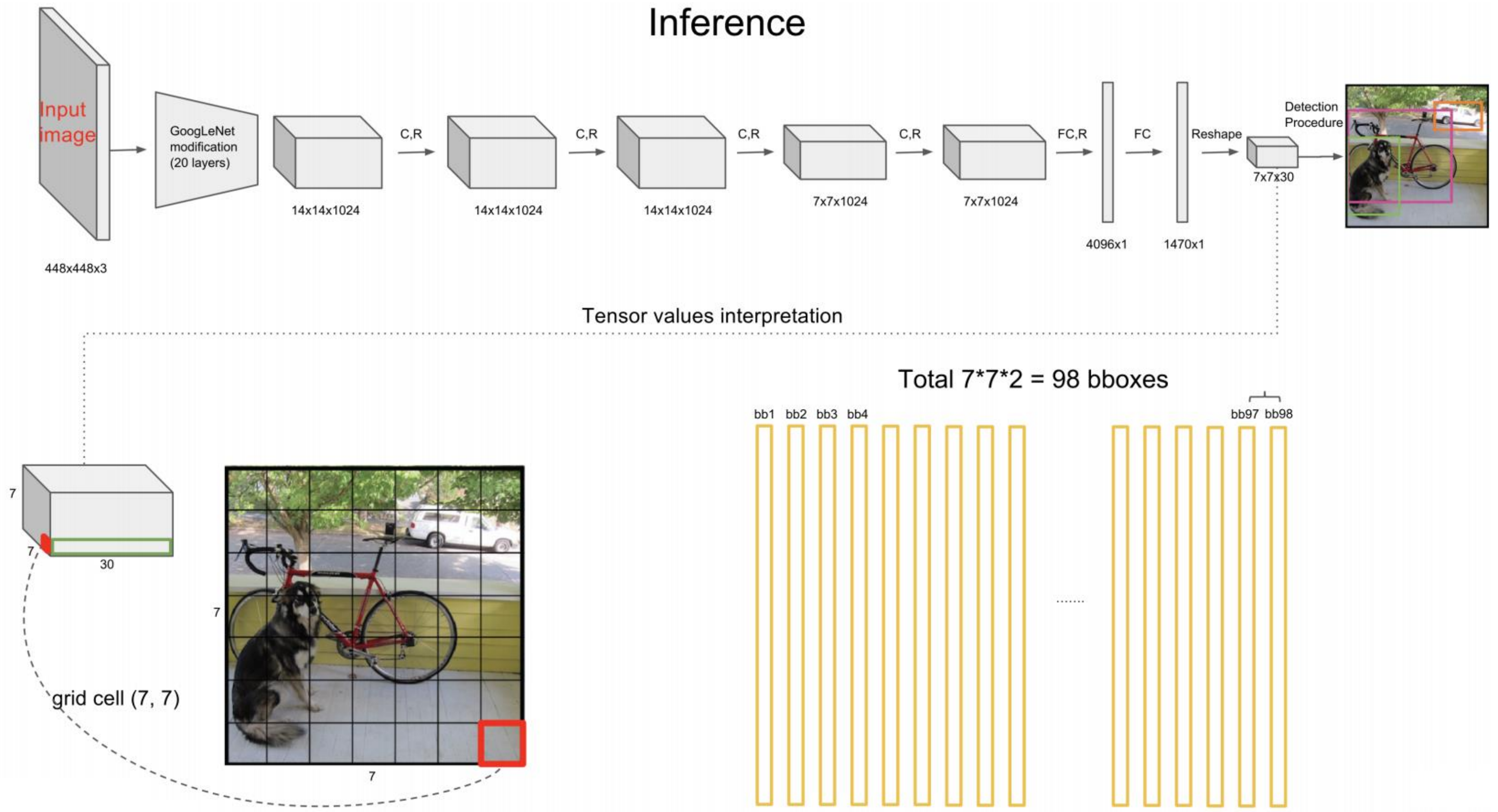
Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
bb2 confidence
1x30
5
5
20
MUL
7
7
30
grid cell
7
7
Class scores for bb2
20x1
Do this operation for each bbox in each grid cell




Slide credit to Taegyun Jeon

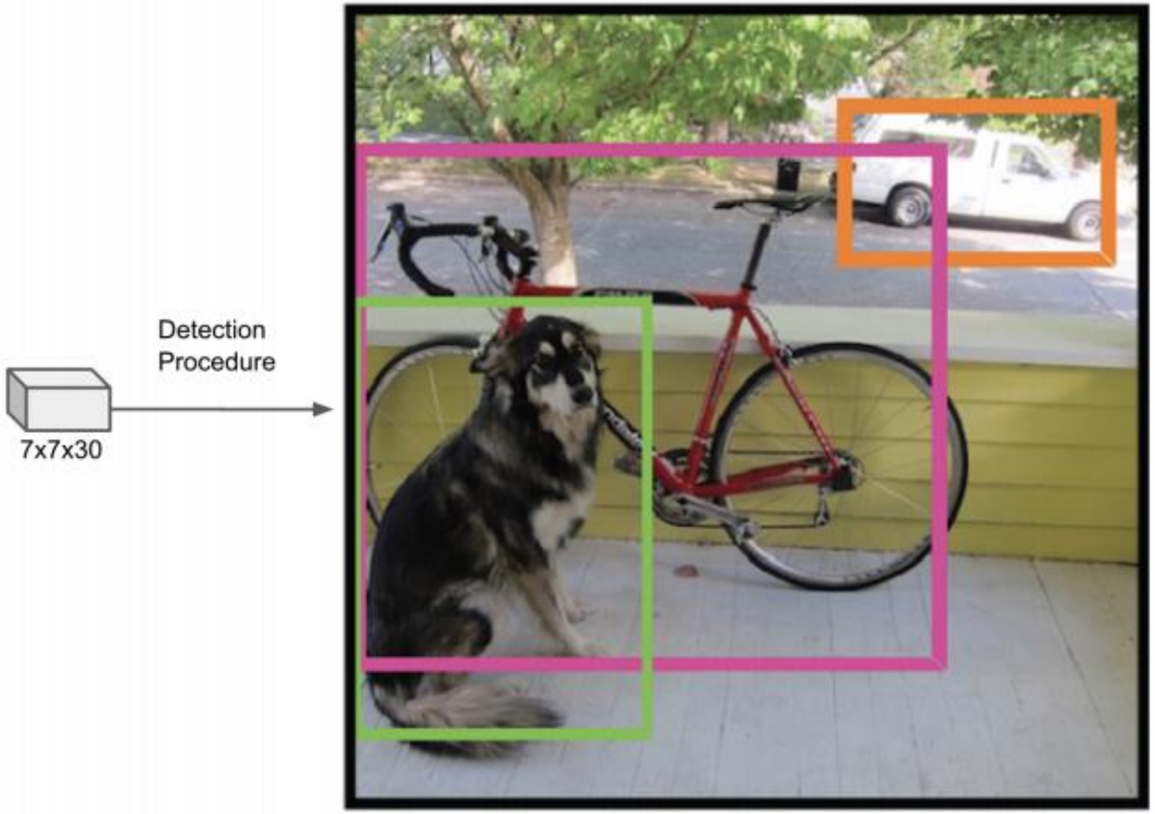

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
2 bboxes for second cell (1, 2)
bb1 bb2 bb3 bb4
20x1 20x1 20x1 20x1
7
7
30
grid cell (1, 2)
7
7

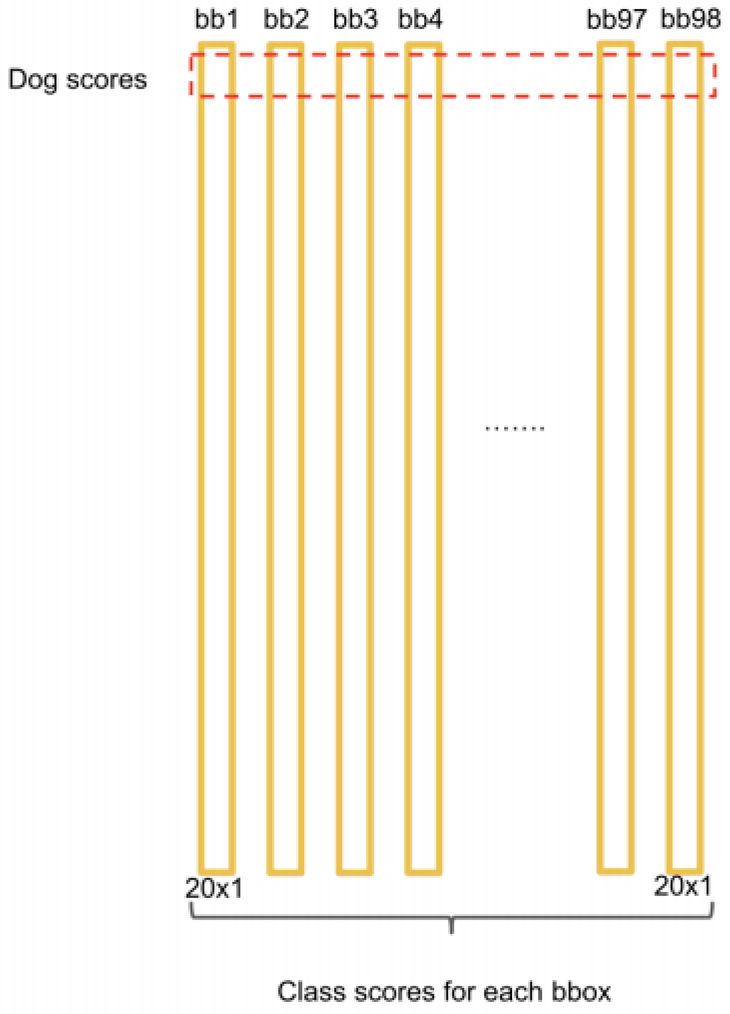

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
7
7
30
grid cell (7, 7)
7
7
2 bboxes for last cell (7, 7)
bb1 bb2 bb3 bb4
bb97 bb98
.......
20x1 20x1 20x1 20x1 20x1 20x1 20x1 20x1 20x1
20x1 20x1 20x1 20x1 20x1 20x1



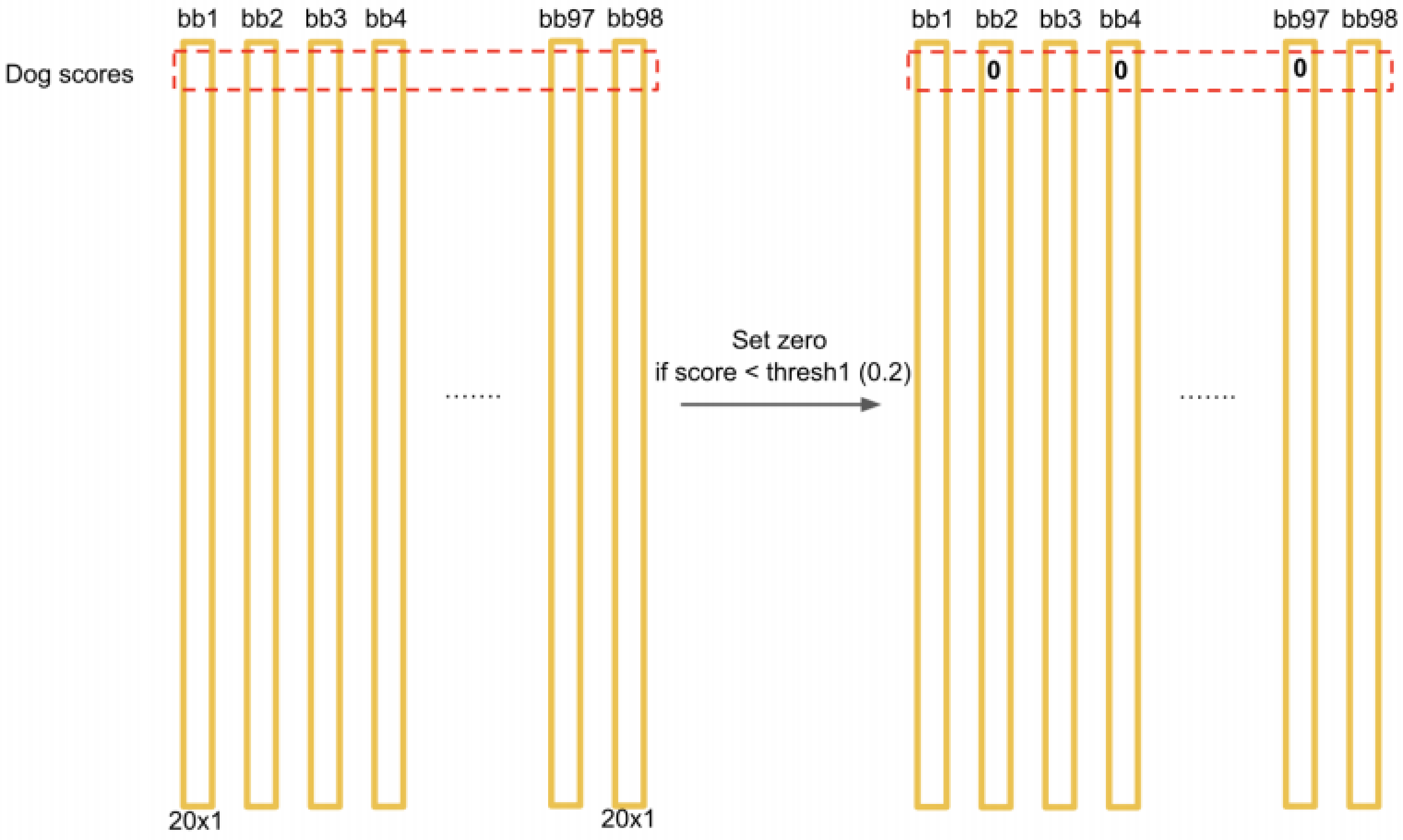
Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
7
7
30
7
7
grid cell (7, 7)
Total 7*7*2 = 98 bboxes
bb1 bb2 bb3 bb4
bb97 bb98
.......

# Look at detection procedure

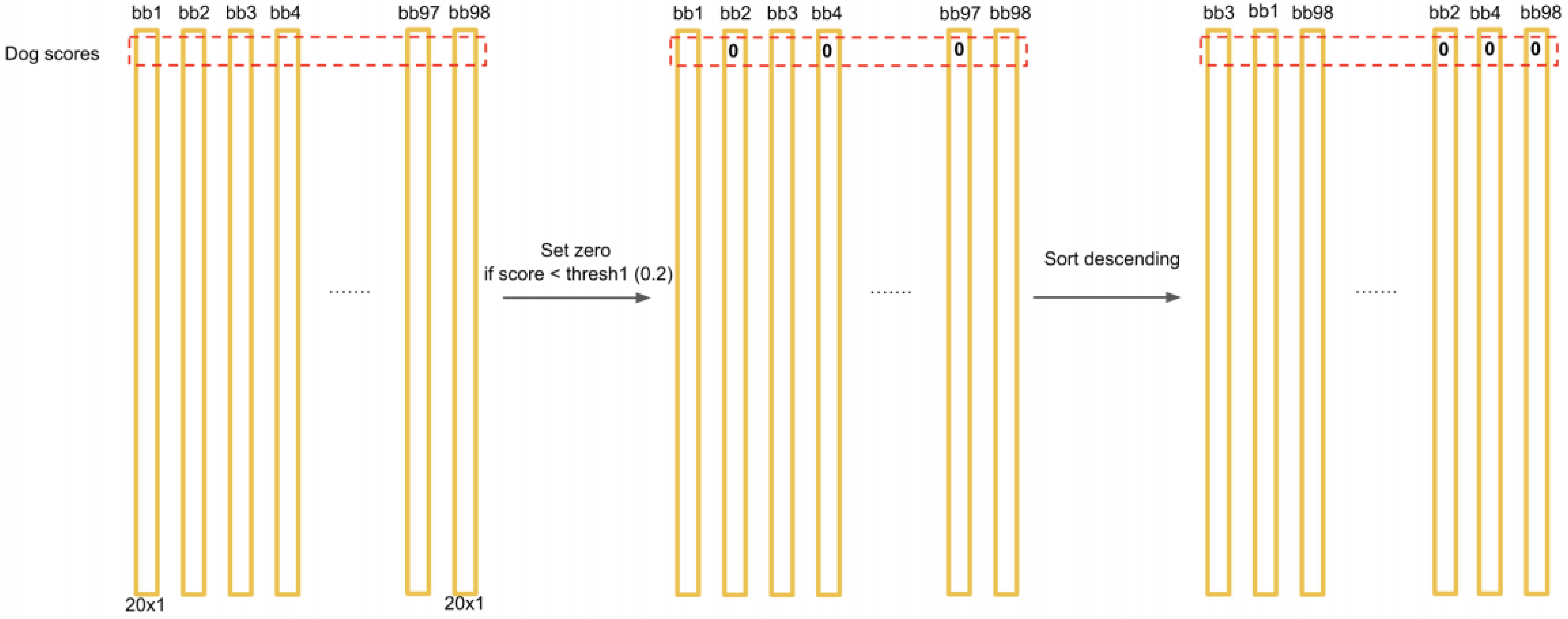




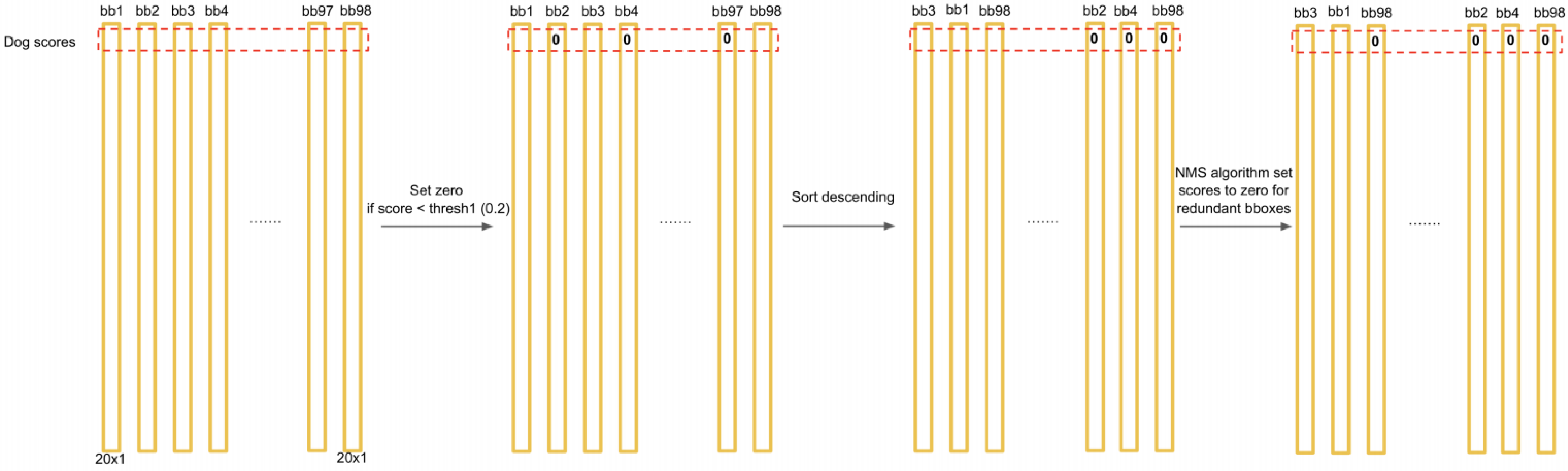

Get first class scores for each bbox



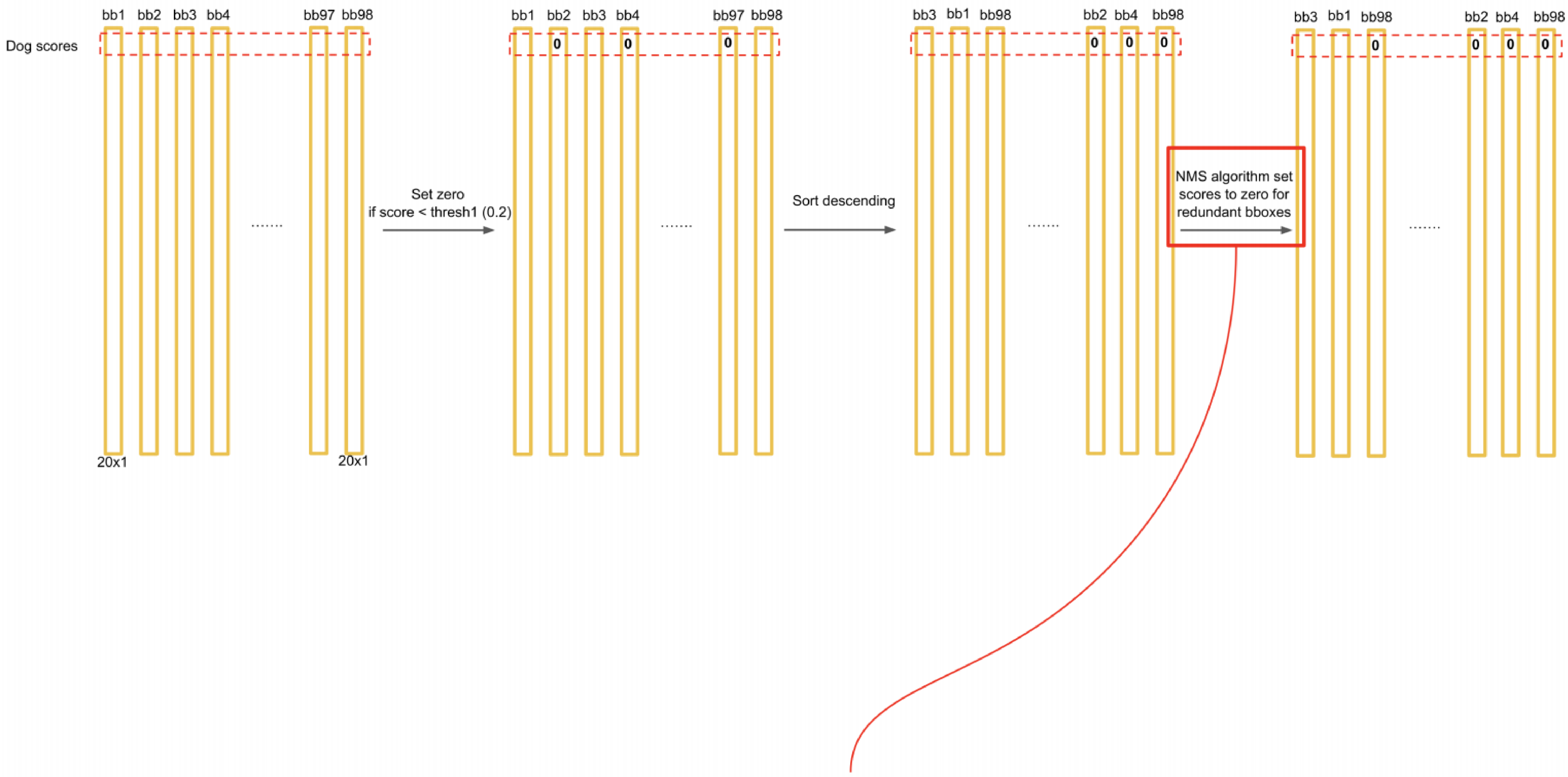

Slide credit to Taegyun Jeon



Dog scores
bb1
bb2
bb3
bb4
bb97
bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1
bb2
bb3
bb4
bb97
bb98
0
0
0
Sort descending
bb3
bb1
bb98
bb2
bb4
bb98
0
0
0



Dog scores
bb1
bb2
bb3
bb4
bb97
bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1
bb2
bb3
bb4
bb97
bb98
0
0
0
Sort descending
bb3
bb1
bb98
bb2
bb4
bb98
0
0
0
NMS algorithm set
scores to zero for
redundant bboxes
bb3
bb1
bb98
bb2
bb4
bb98
0
0
0
0

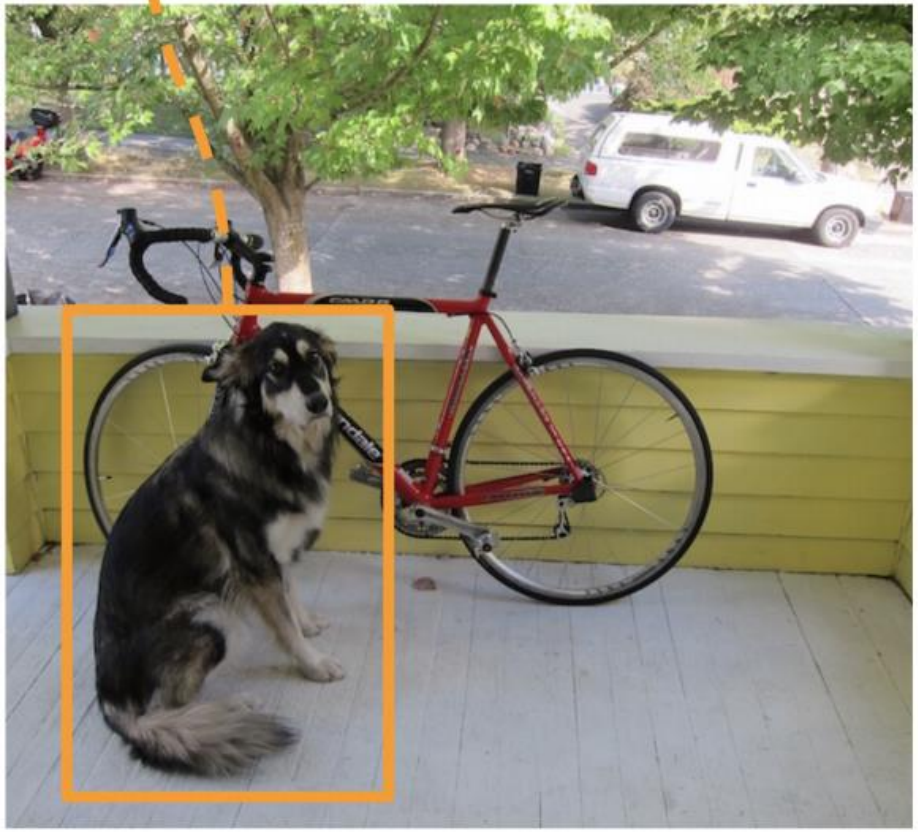

Dog scores
bb1 bb2 bb3 bb4 bb97 bb98
20x1
20x1
.......
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 bb97 bb98
0 0 0
.......
Sort descending
bb3 bb1 bb98 bb2 bb4 bb98
0 0 0
.......
NMS algorithm set
scores to zero for
redundant bboxes
bb3 bb1 bb98 bb2 bb4 bb98
0 0 0 0
.......

How it works

# Non-Maximum Suppression: intuition

class (dog) scores for each bbox

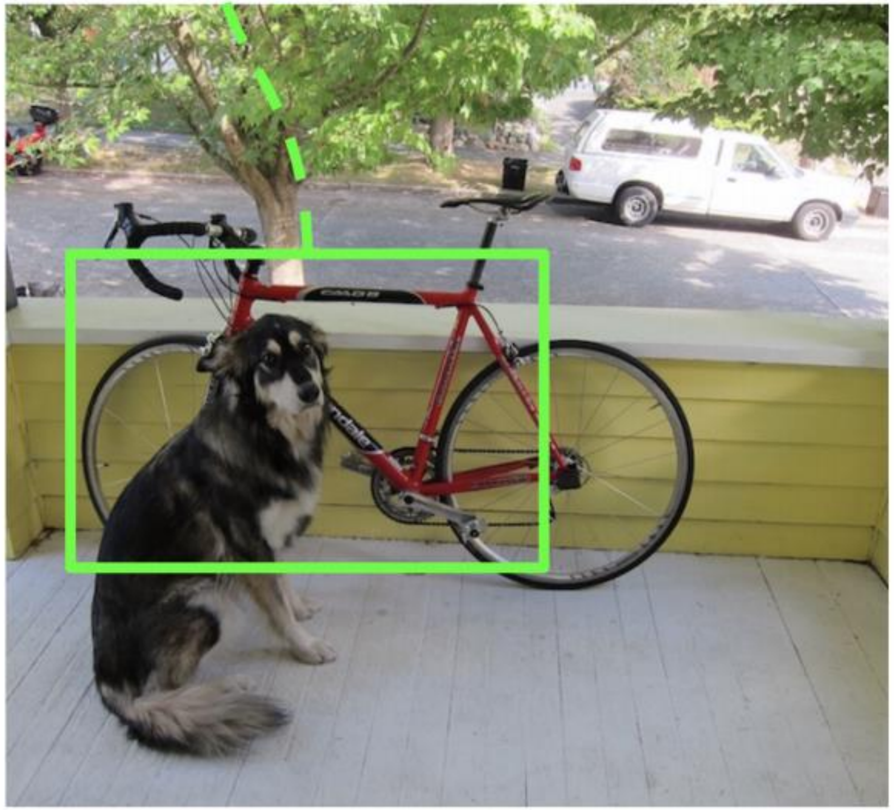
| | bb47 | bb20 | bb15 | bb7 | | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | 0.3 | 0.2 | 0.1 | | | | | | | 0 | 0 | 0 | 0 | 1x98 |



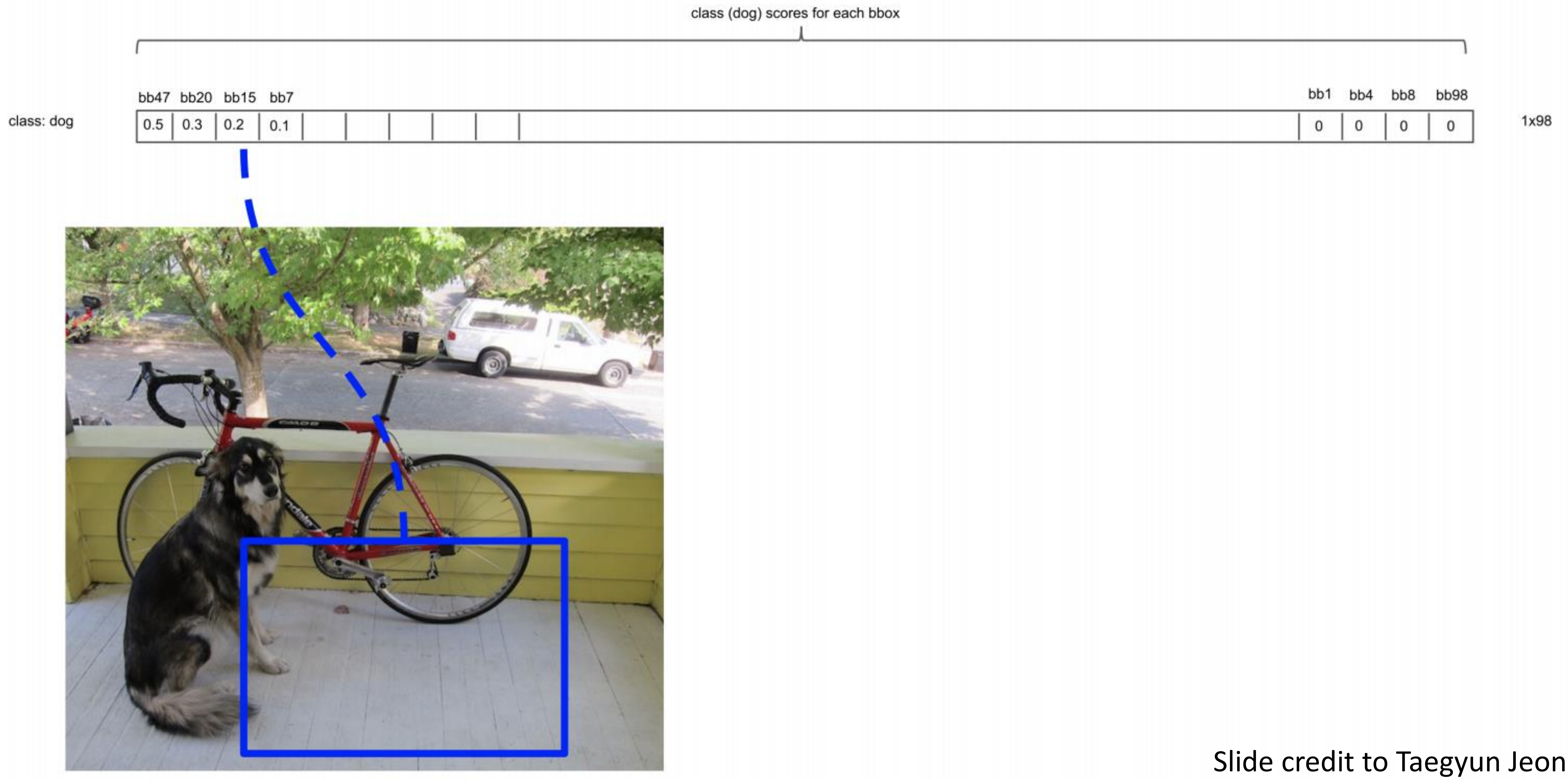
# Non-Maximum Suppression: intuition

class (dog) scores for each bbox

| | bb47 | bb20 | bb15 | bb7 | | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | 0.3 | 0.2 | 0.1 | | | | | | | 0 | 0 | 0 | 0 | 1x98 |



# Non-Maximum Suppression: intuition

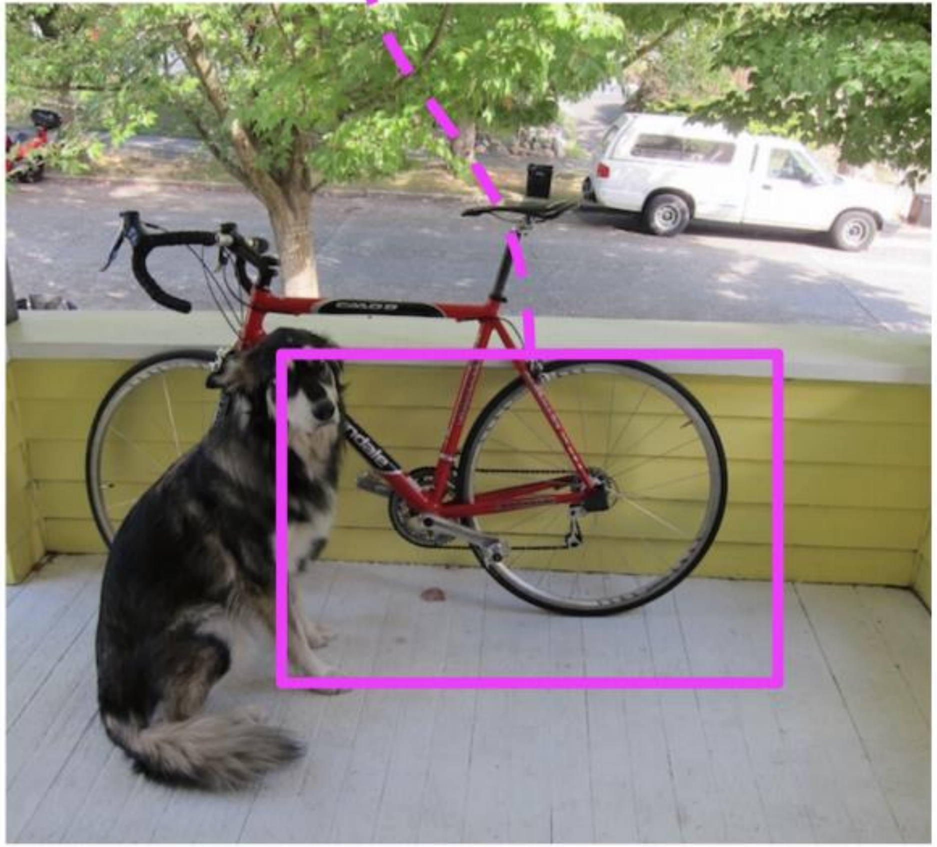



Slide credit to Taegyun Jeon

# Non-Maximum Suppression: intuition

class (dog) scores for each bbox

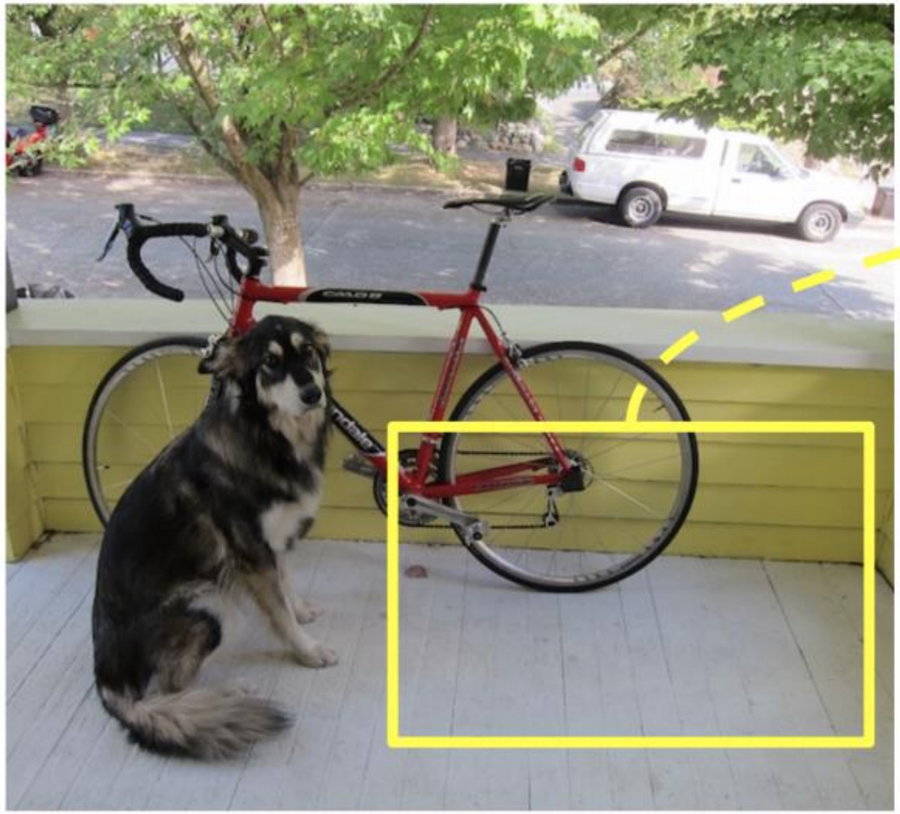
| | bb47 | bb20 | bb15 | bb7 | | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | 0.3 | 0.2 | 0.1 | | | | | | | 0 | 0 | 0 | 0 | 1x98 |



# Non-Maximum Suppression: intuition

class (dog) scores for each bbox

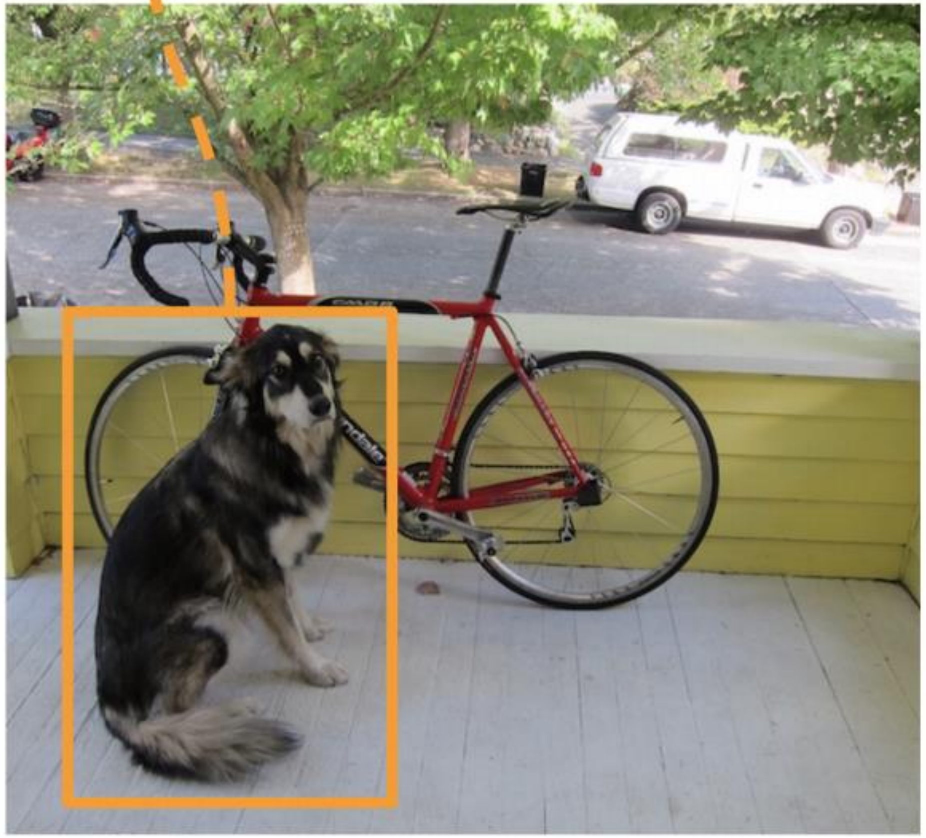
| | bb47 | bb20 | bb15 | bb7 | | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | 0.3 | 0.2 | 0.1 | | | | | | | 0 | 0 | 0 | 0 | 1x98 |



# Non-Maximum Suppression: intuition

class (dog) scores for each bbox

| | bb47 | bb20 | bb15 | bb7 | | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | 0.3 | 0.2 | 0.1 | | | | | | | 0 | 0 | 0 | 0 | 1x98 |

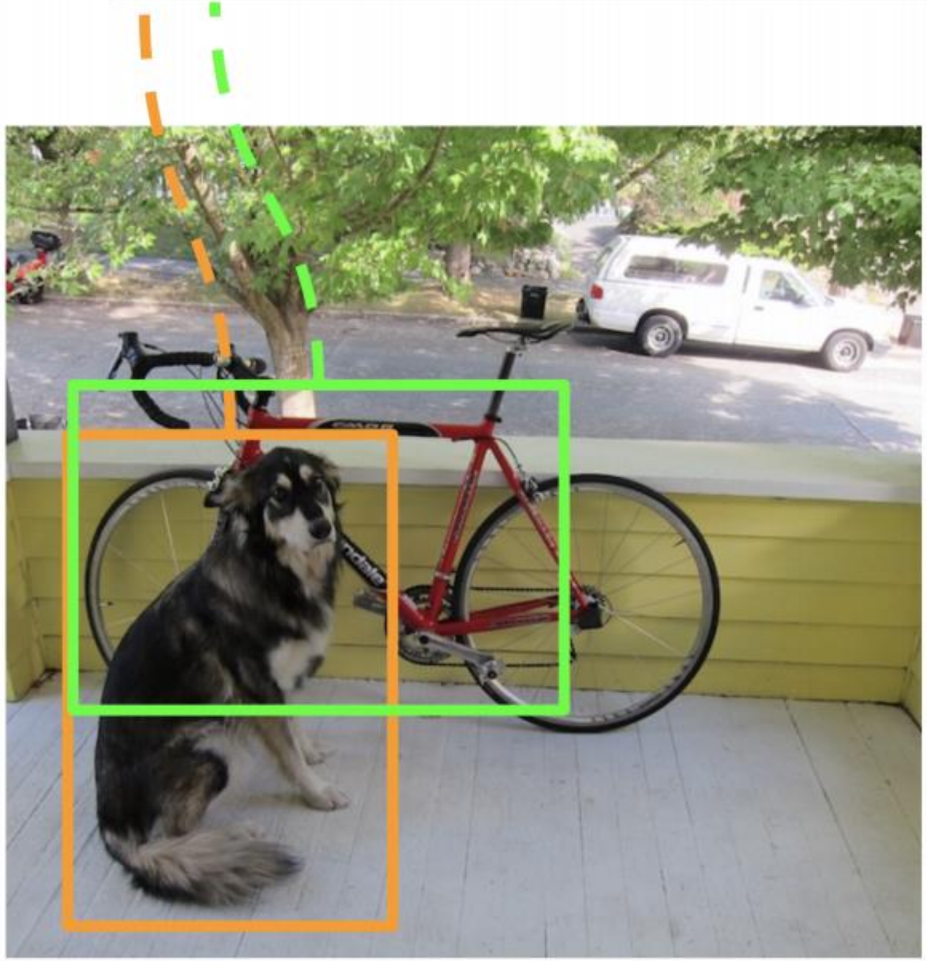


Get ~~bbox with max~~ score. Let’s denote it “bbox_max”

# Non-Maximum Suppression: intuition

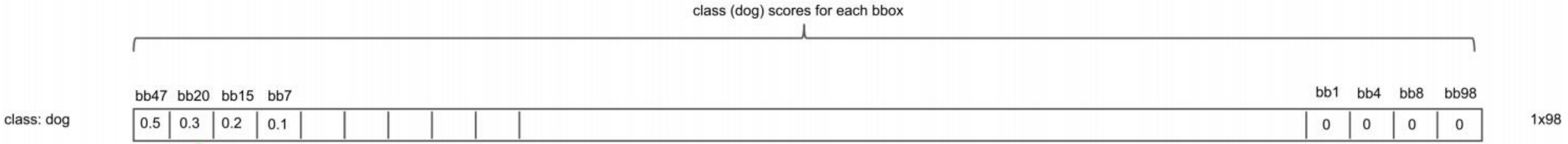
class (dog) scores for each bbox

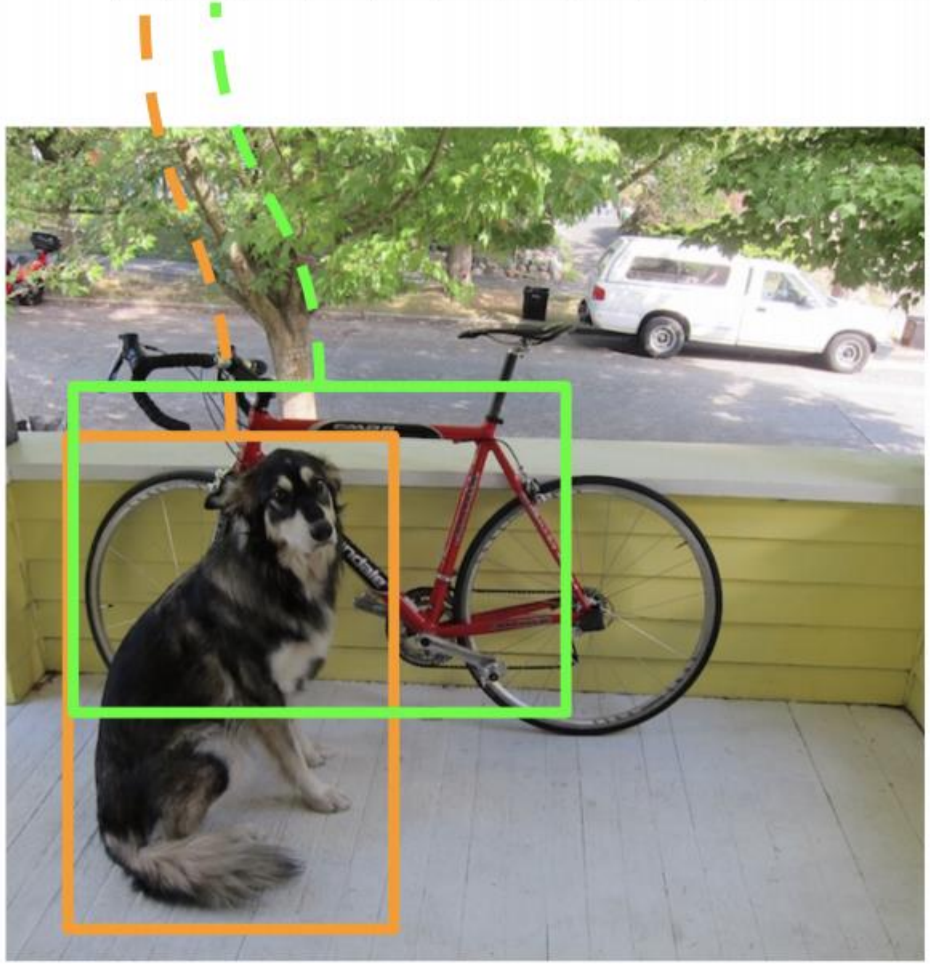
| | bb47 | bb20 | bb15 | bb7 | | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | 0.3 | 0.2 | 0.1 | | | | | | | 0 | 0 | 0 | 0 | 1x98 |



Compare "bbox_max" with others less score (non-zero!) bboxes. Let's denote it "bbox_cur"

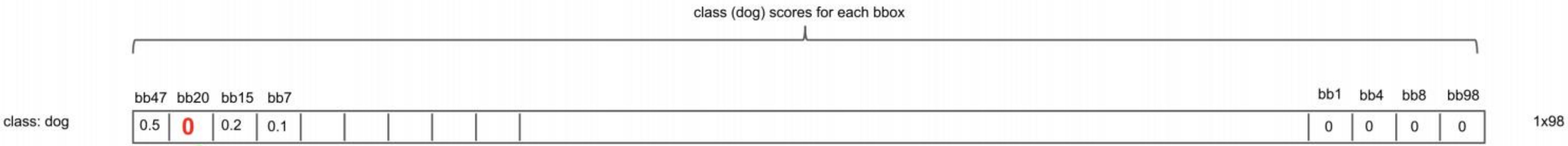
# Non-Maximum Suppression: intuition

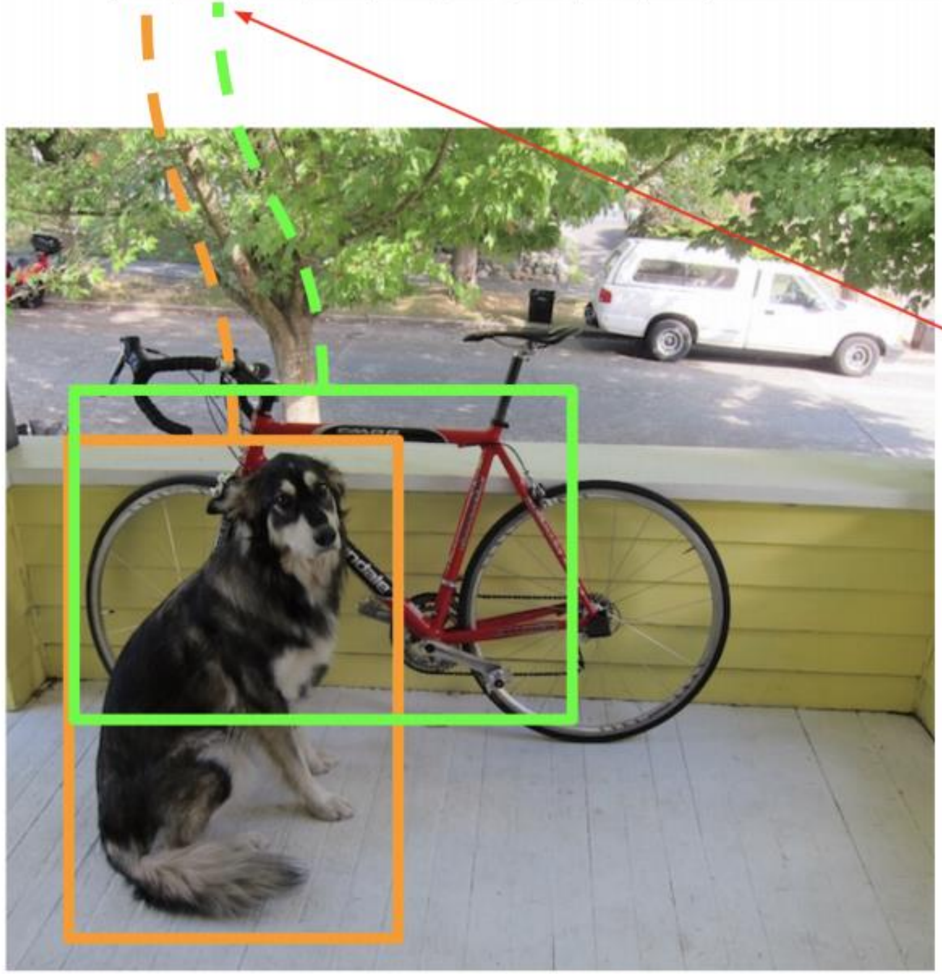


If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

# Non-Maximum Suppression: intuition



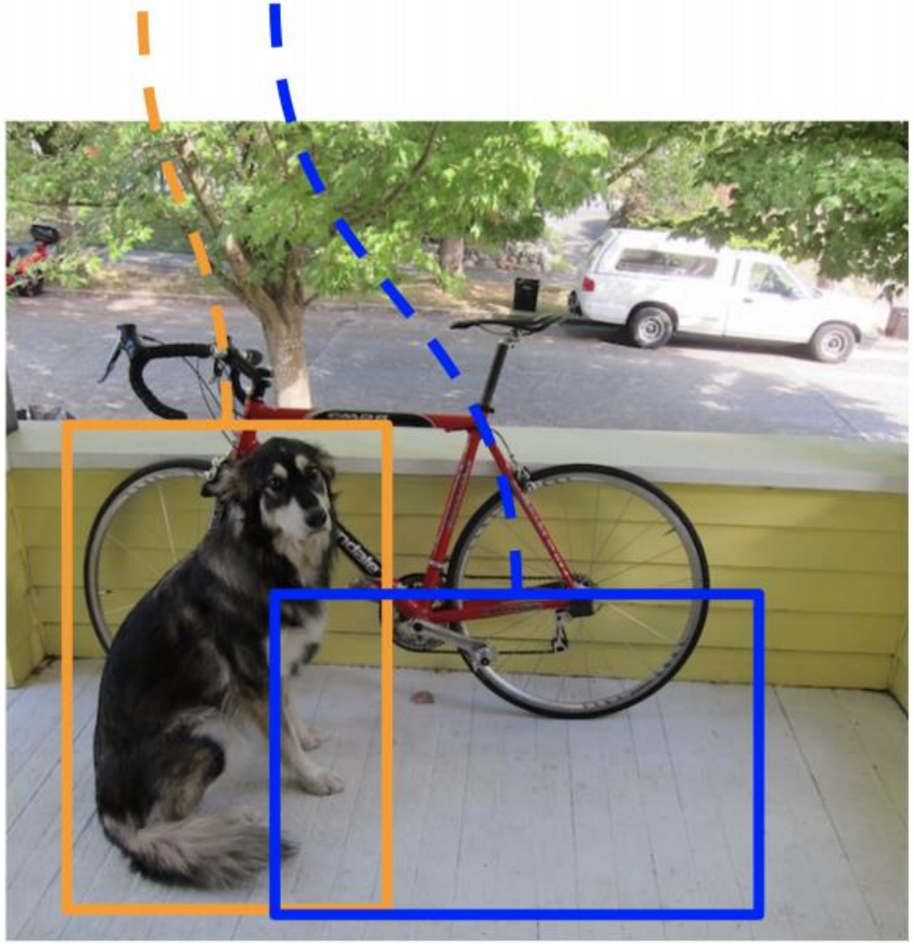
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: set to 0.

# Non-Maximum Suppression: intuition

class (dog) scores for each bbox

| | bb47 | bb20 | bb15 | bb7 | | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | 0 | 0.2 | 0.1 | | | | | | | 0 | 0 | 0 | 0 | 1x98 |

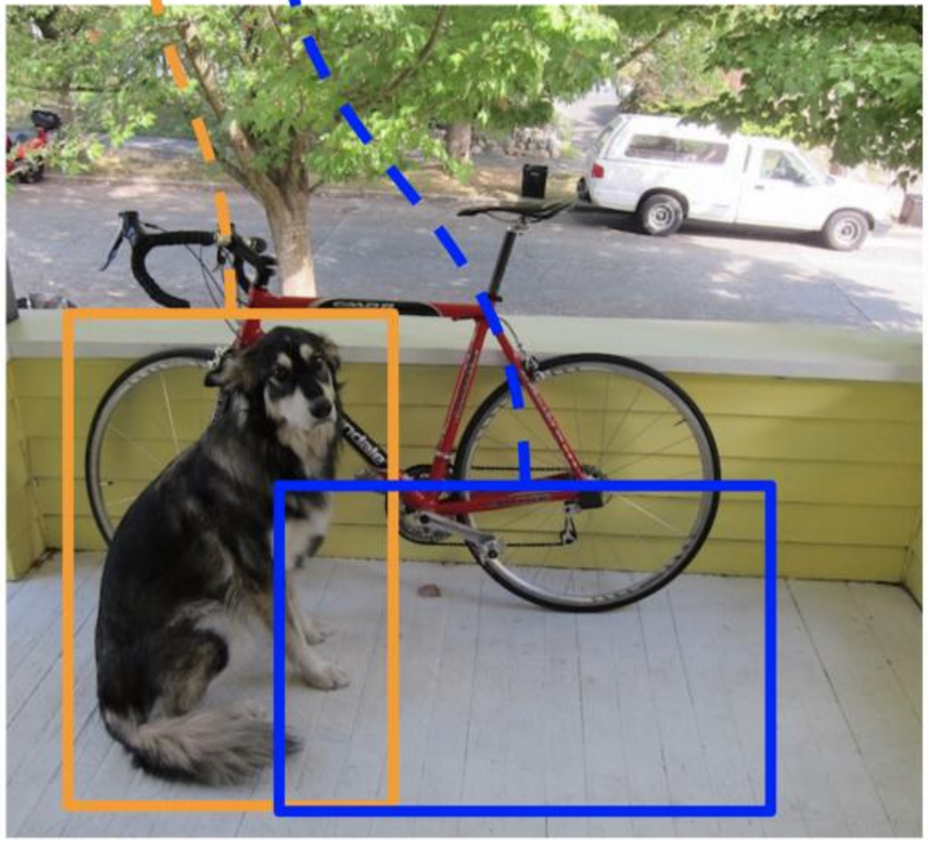


Go to next bbox_cur.

# Non-Maximum Suppression: intuition

class (dog) scores for each bbox

| | bb47 | bb20 | bb15 | bb7 | | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | **0** | 0.2 | 0.1 | | | | | | | 0 | 0 | 0 | 0 | 1x98 |

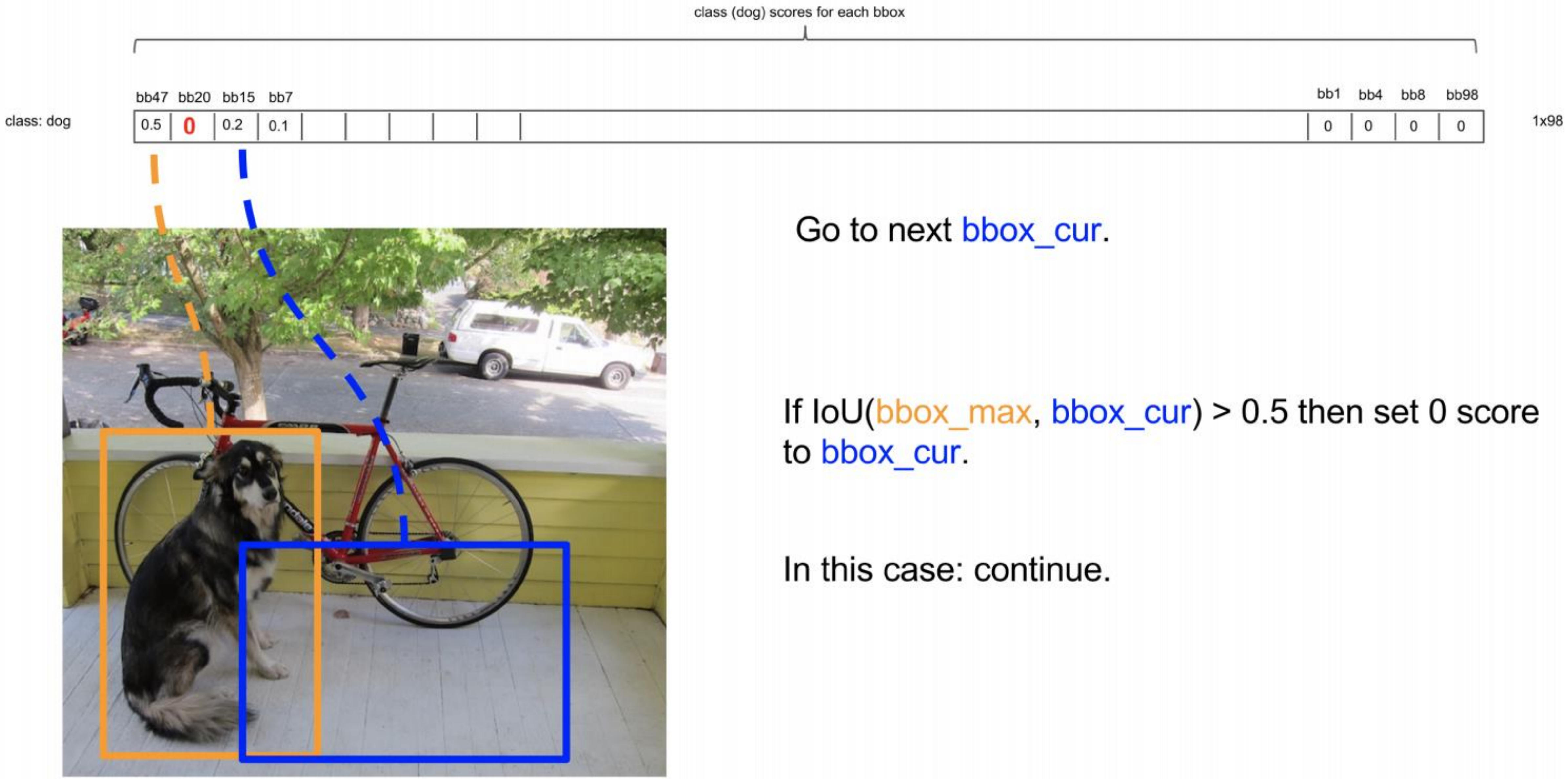


Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

Slide credit to Taegyun Jeon

# Non-Maximum Suppression: intuition




Go to next bbox_cur.

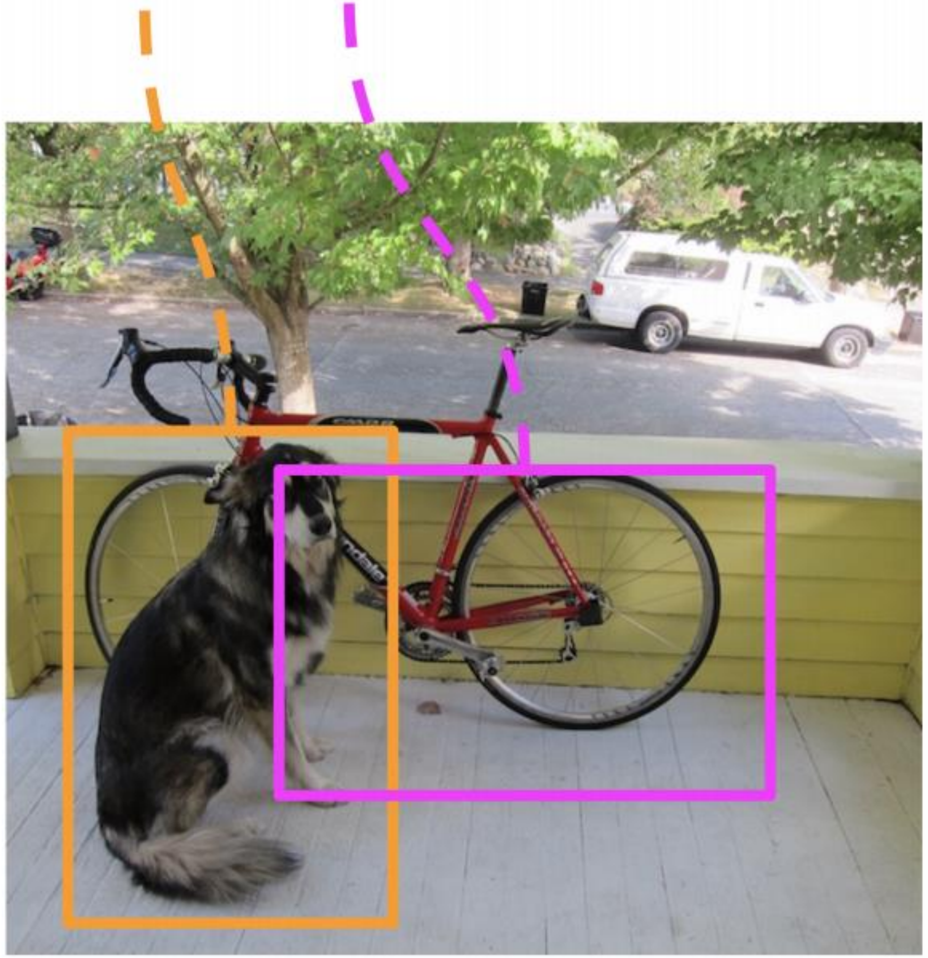
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: continue.

# Non-Maximum Suppression: intuition

class (dog) scores for each bbox

| | bb47 | bb20 | bb15 | bb7 | | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | 0 | 0.2 | 0.1 | | | | | | | 0 | 0 | 0 | 0 | 1x98 |

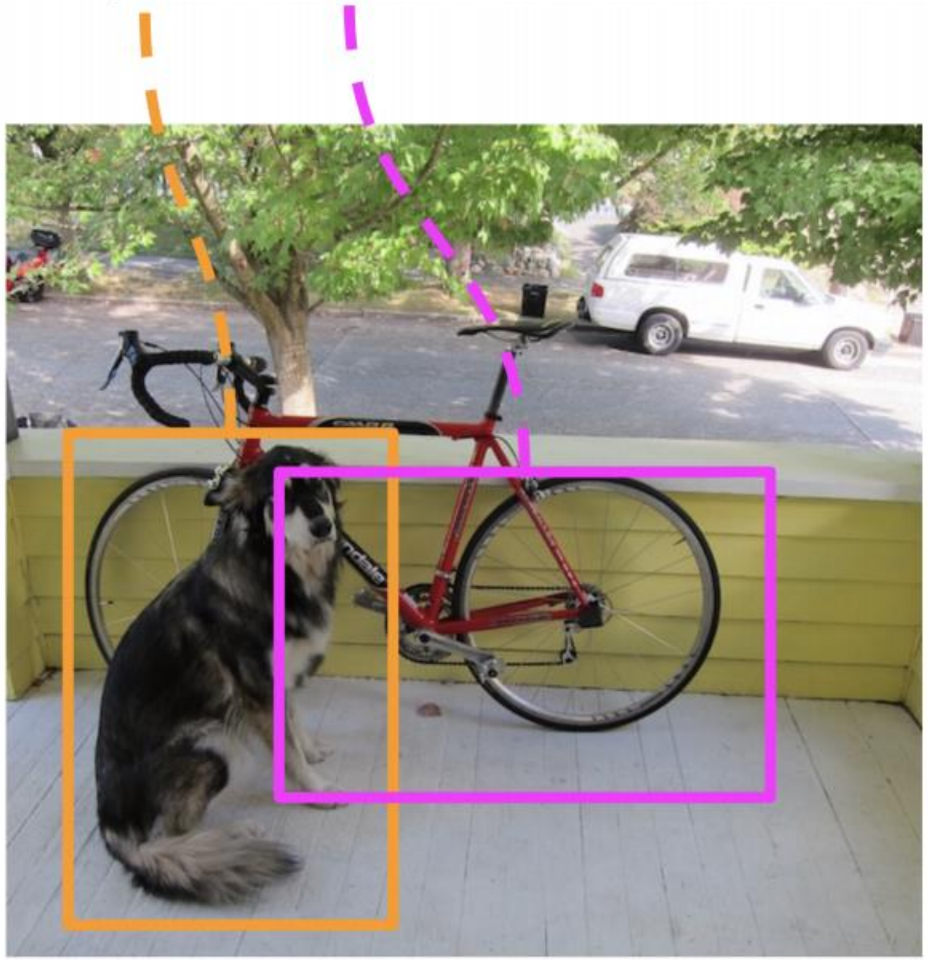


Go to next bbox_cur.

# Non-Maximum Suppression: intuition

class (dog) scores for each bbox

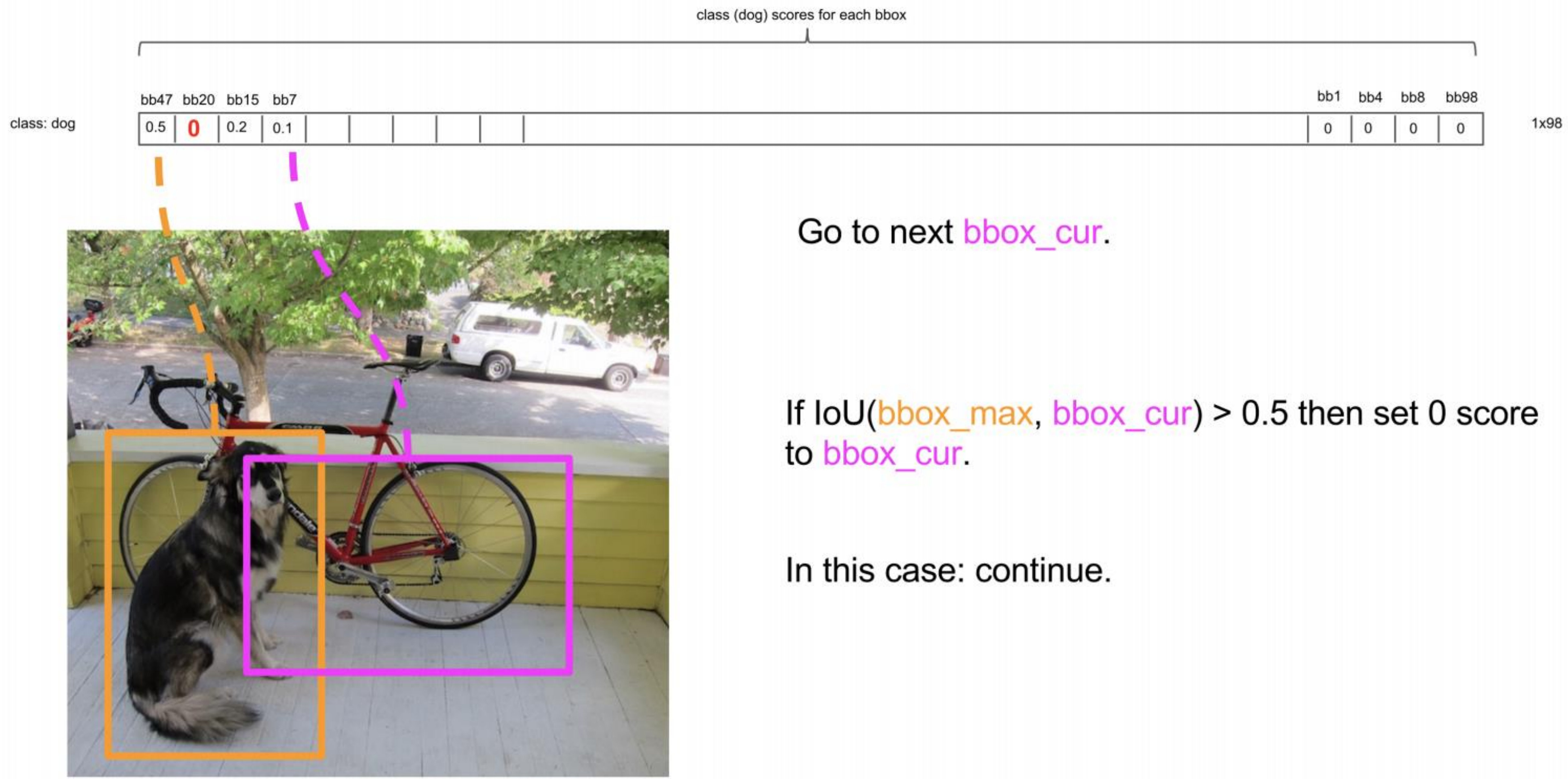
| | bb47 | bb20 | bb15 | bb7 | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | 0 | 0.2 | 0.1 | | | | | | 0 | 0 | 0 | 0 | 1x98 |



Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

# Non-Maximum Suppression: intuition




Go to next bbox_cur.

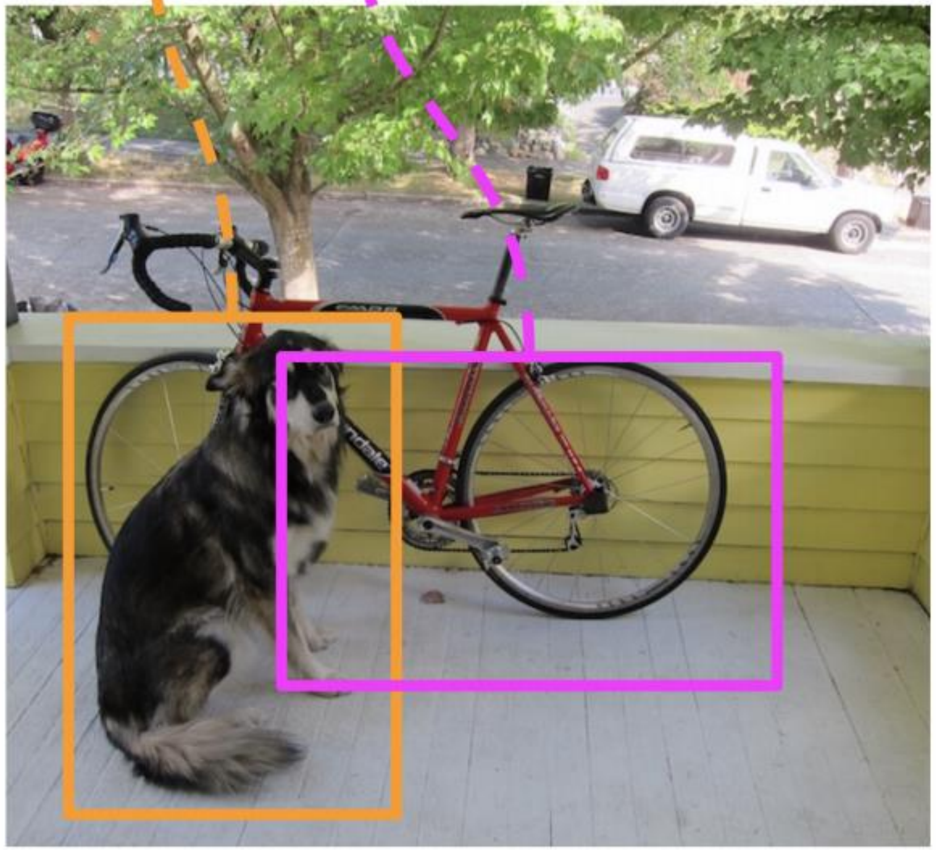
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: continue.

# Non-Maximum Suppression: intuition

class (dog) scores for each bbox

| | bb47 | bb20 | bb15 | bb7 | | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | 0 | 0.2 | 0.1 | | | | | | | 0 | 0 | 0 | 0 | 1x98 |



Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

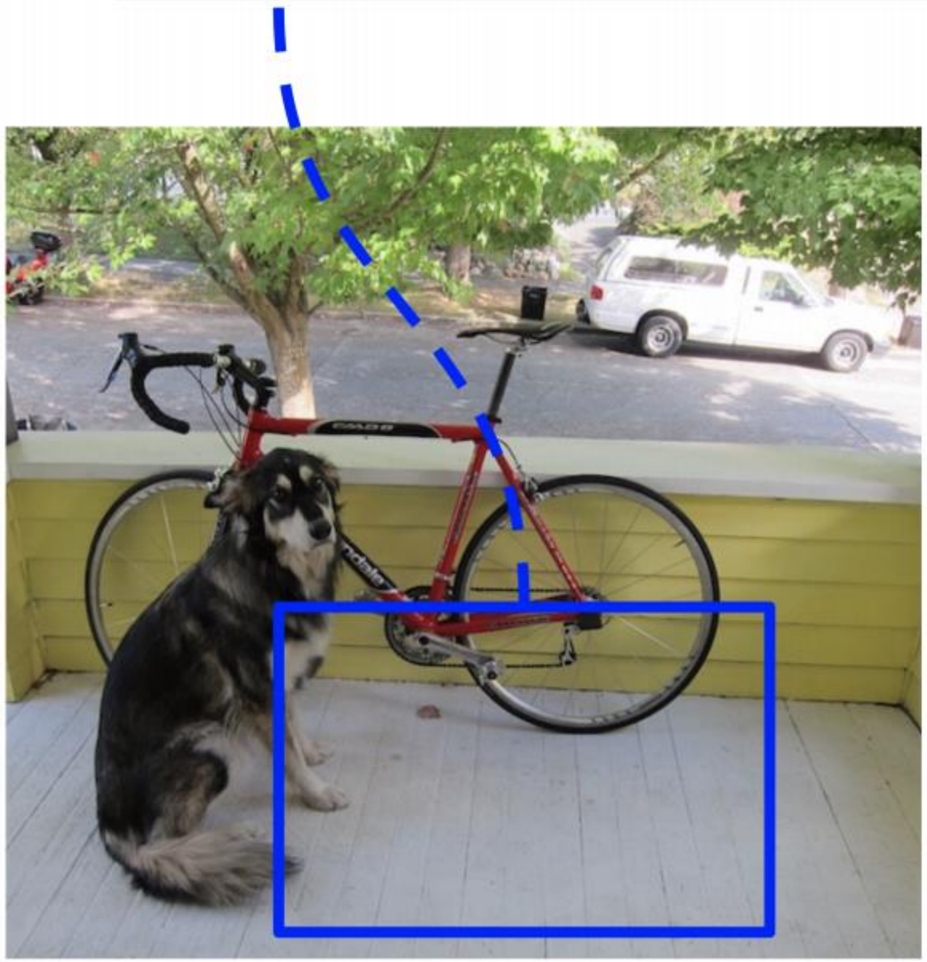
In this case: continue.

Do this procedure for other “bbox_cur”. After that ...

# Non-Maximum Suppression: intuition

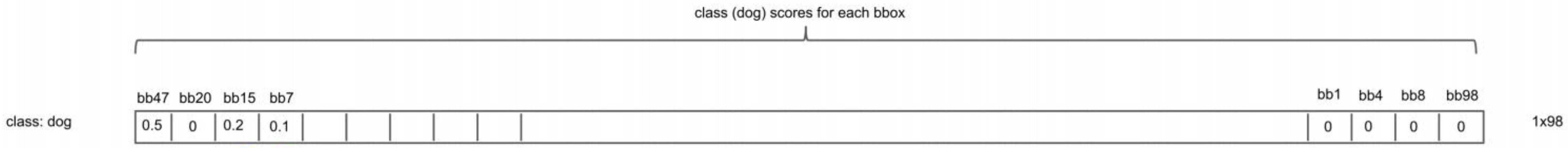
class (dog) scores for each bbox

| | bb47 | bb20 | bb15 | bb7 | | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | 0 | 0.2 | 0.1 | | | | | | | 0 | 0 | 0 | 0 | 1x98 |

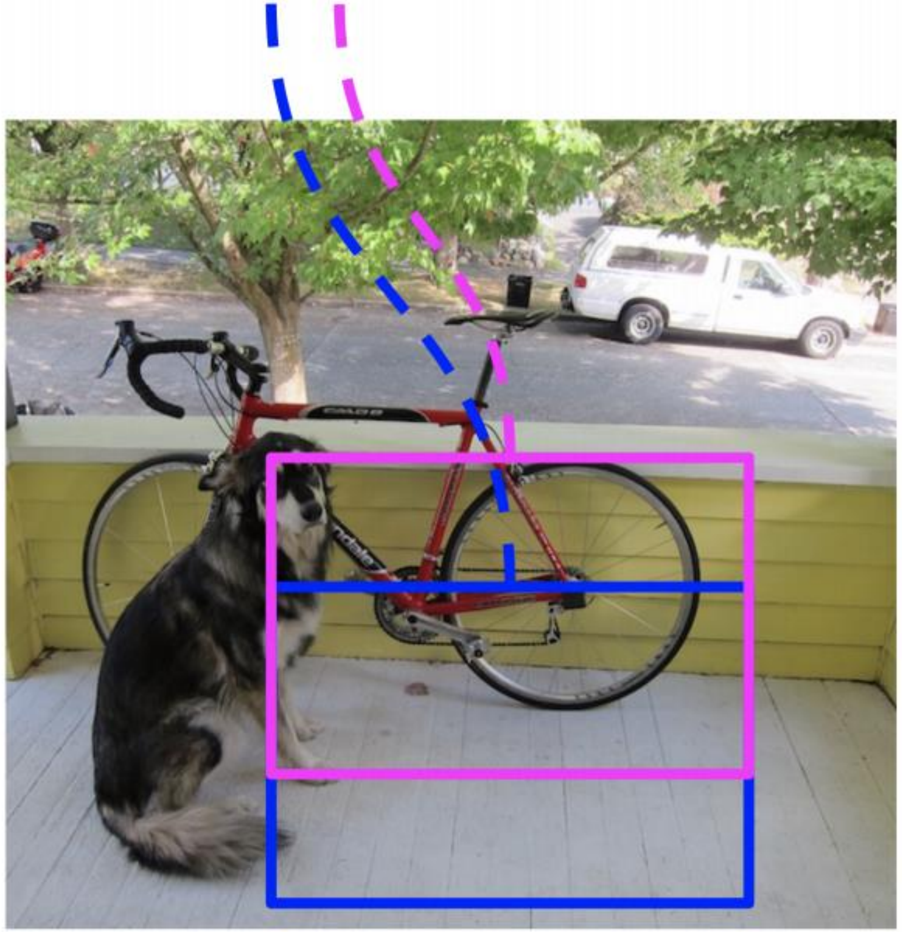


Go to next bbox with big score.
Let's denote it "bbox_max"

# Non-Maximum Suppression: intuition

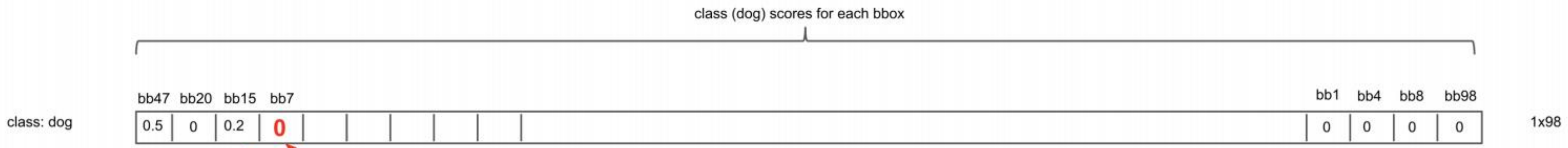


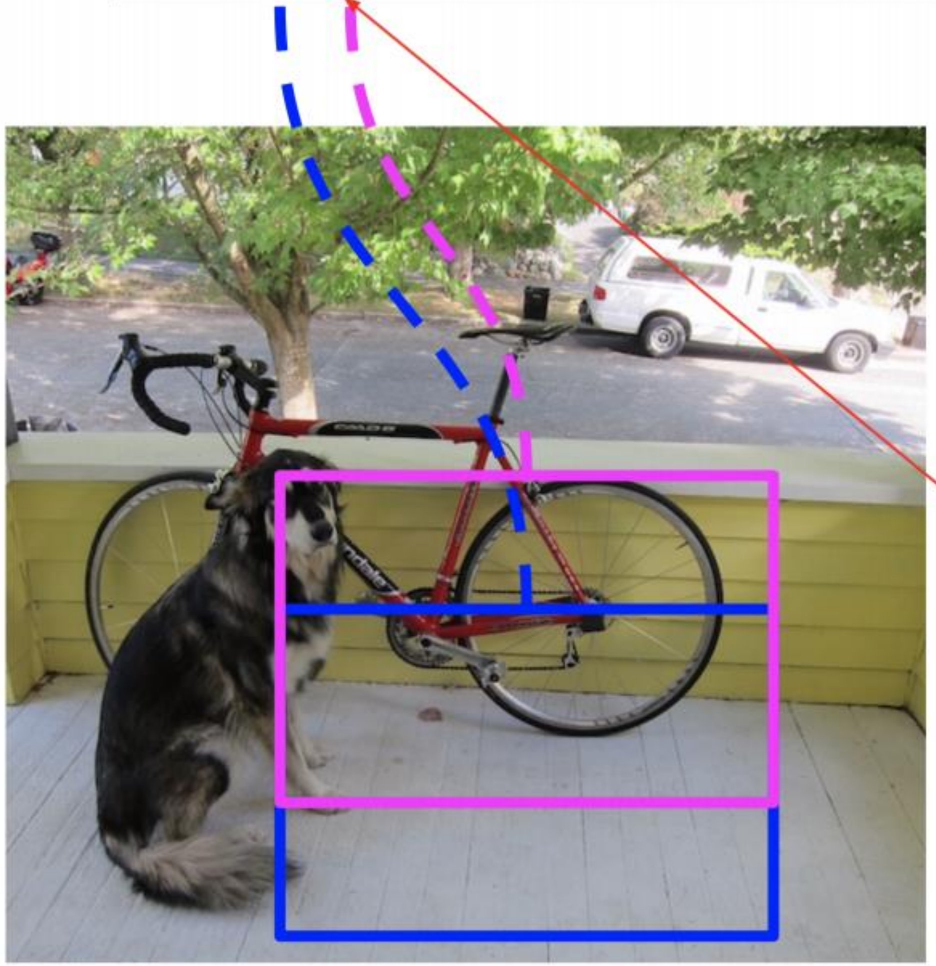

Go to next bbox_cur.



Slide credit to Taegyun Jeon

# Non-Maximum Suppression: intuition



Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

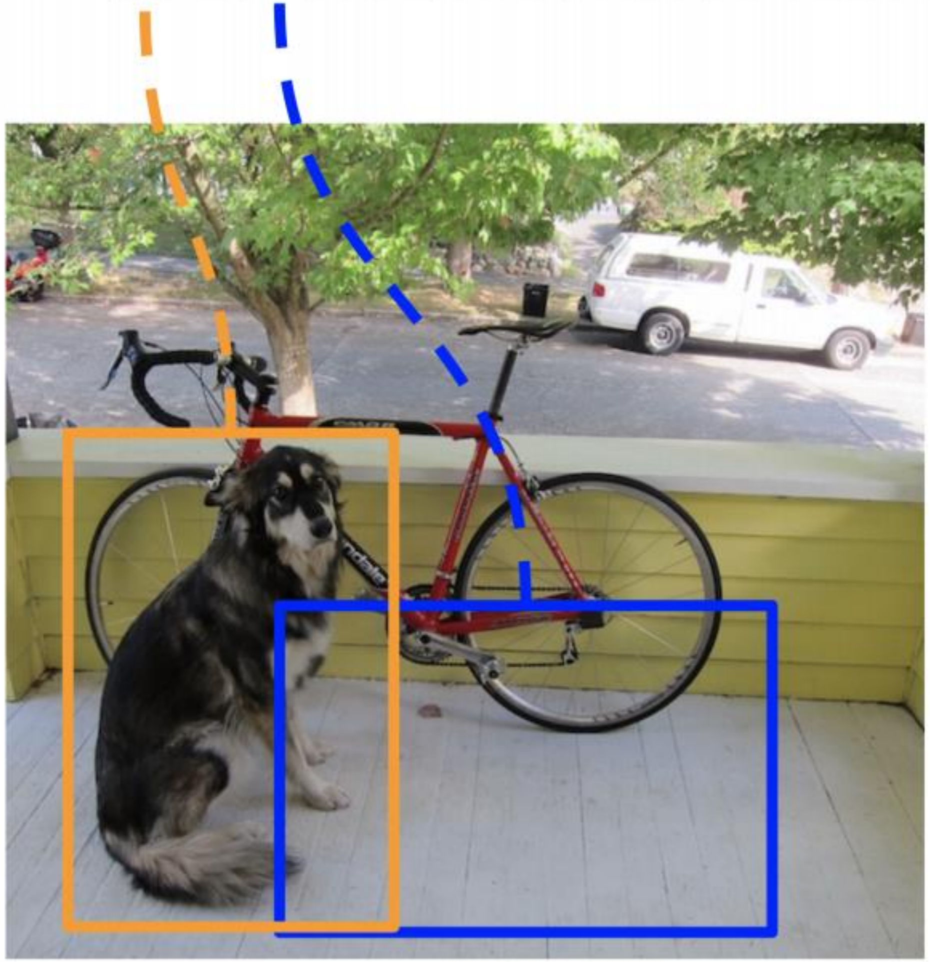
In this case: set to 0.

Do this procedure for other “bbox_max” and for other corresponding “bbox_cur”.

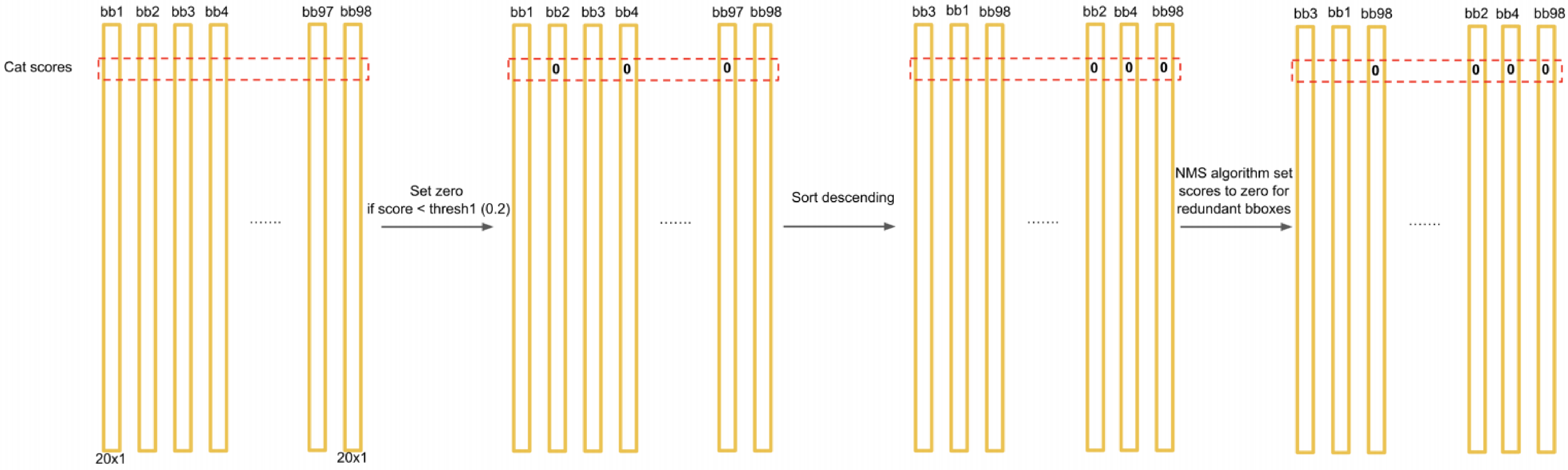
# Non-Maximum Suppression: intuition

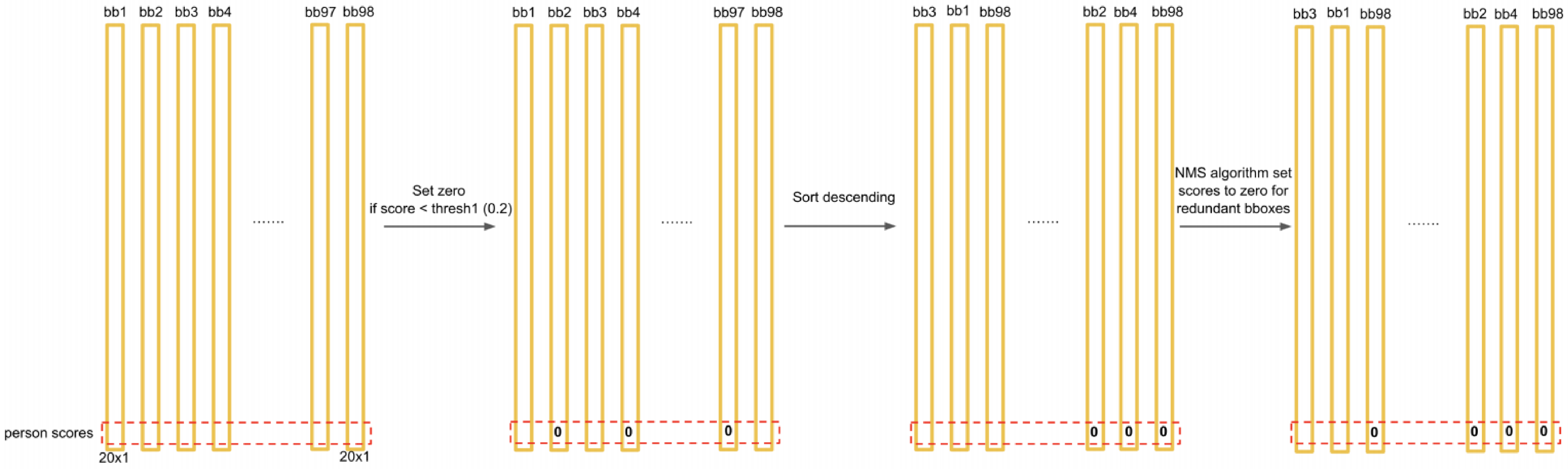
class (dog) scores for each bbox

| | bb47 | bb20 | bb15 | bb7 | | | | | | | bb1 | bb4 | bb8 | bb98 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| class: dog | 0.5 | 0 | 0.2 | 0 | | | | | | | 0 | 0 | 0 | 0 | 1x98 |

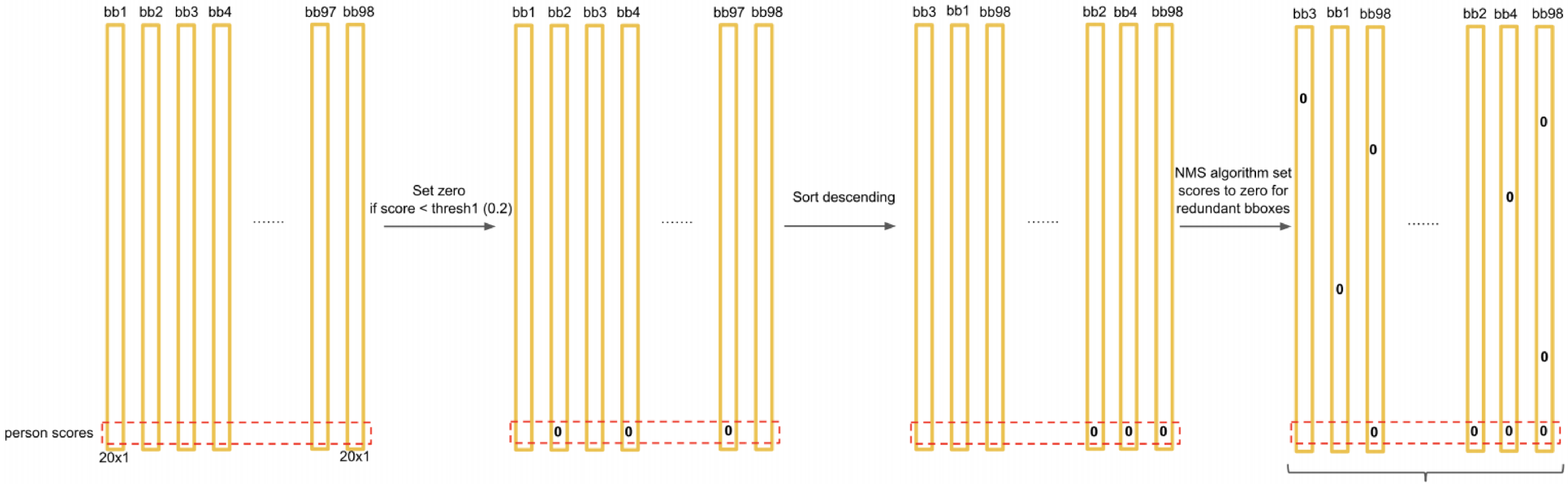


After comparison ~~almost all~~ pairs of bboxes the only two bboxes left with non-zero class score value.

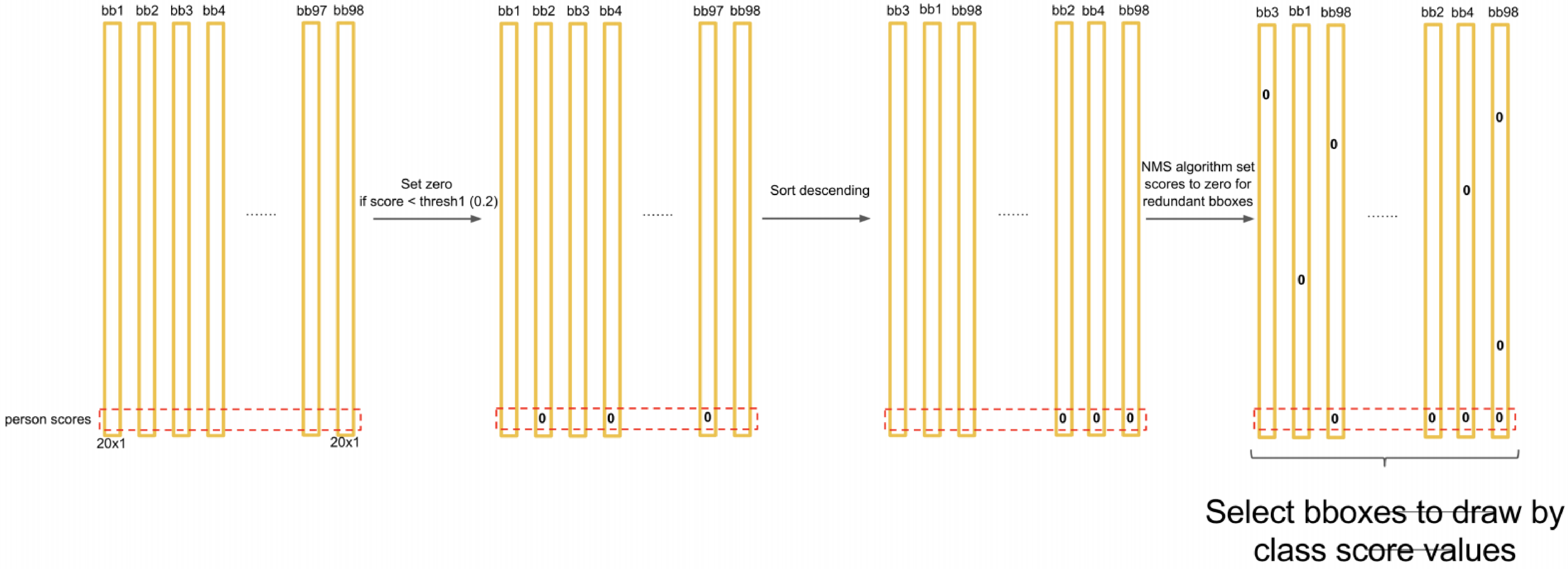

bb1 bb2 bb3 bb4 bb97 bb98
Cat scores
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 bb97 bb98
0 0 0
Sort descending
bb3 bb1 bb98 bb2 bb4 bb98
0 0 0
NMS algorithm set
scores to zero for
redundant bboxes
bb3 bb1 bb98 bb2 bb4 bb98
0 0 0 0

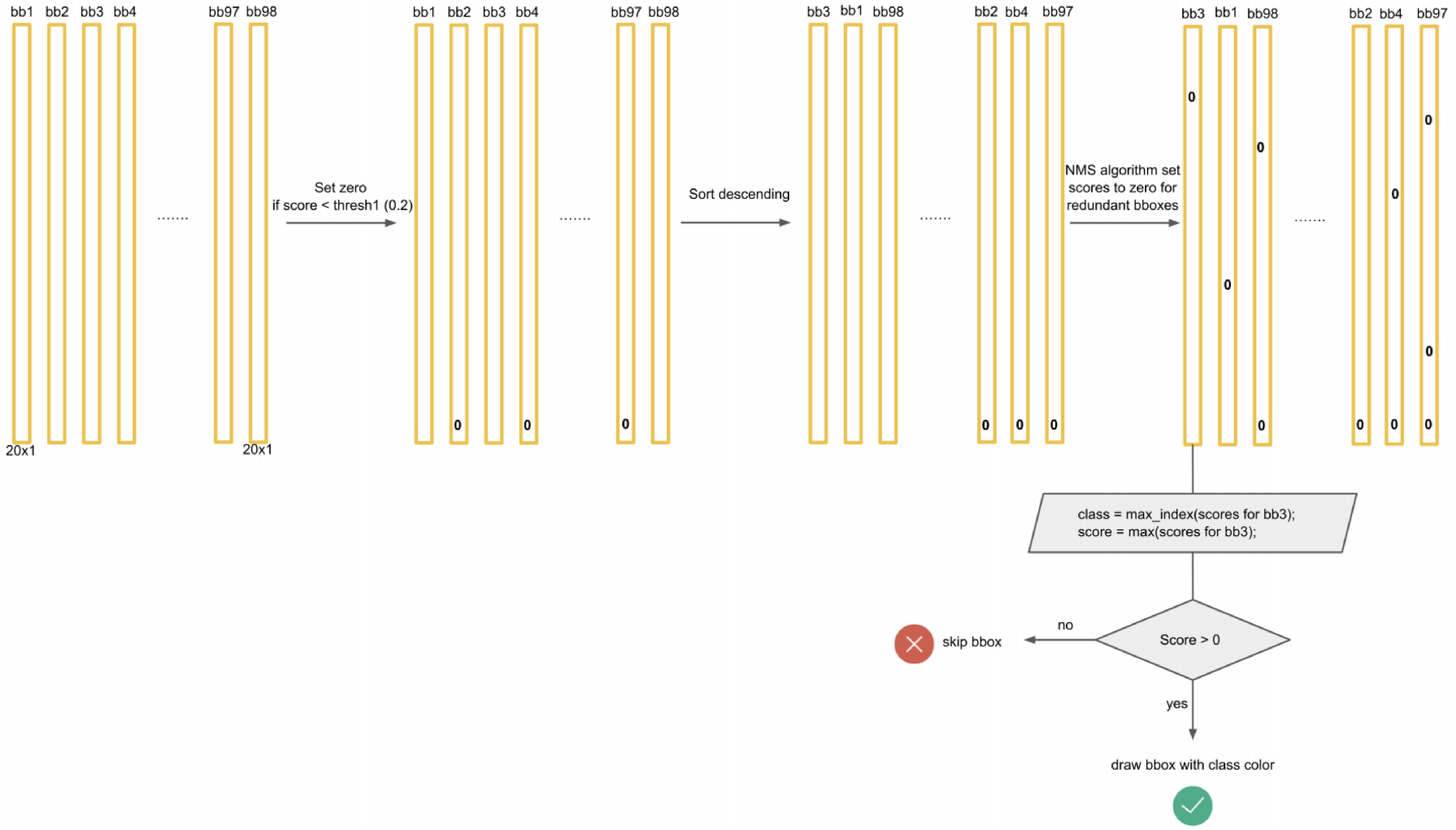



Do this procedure for all classes

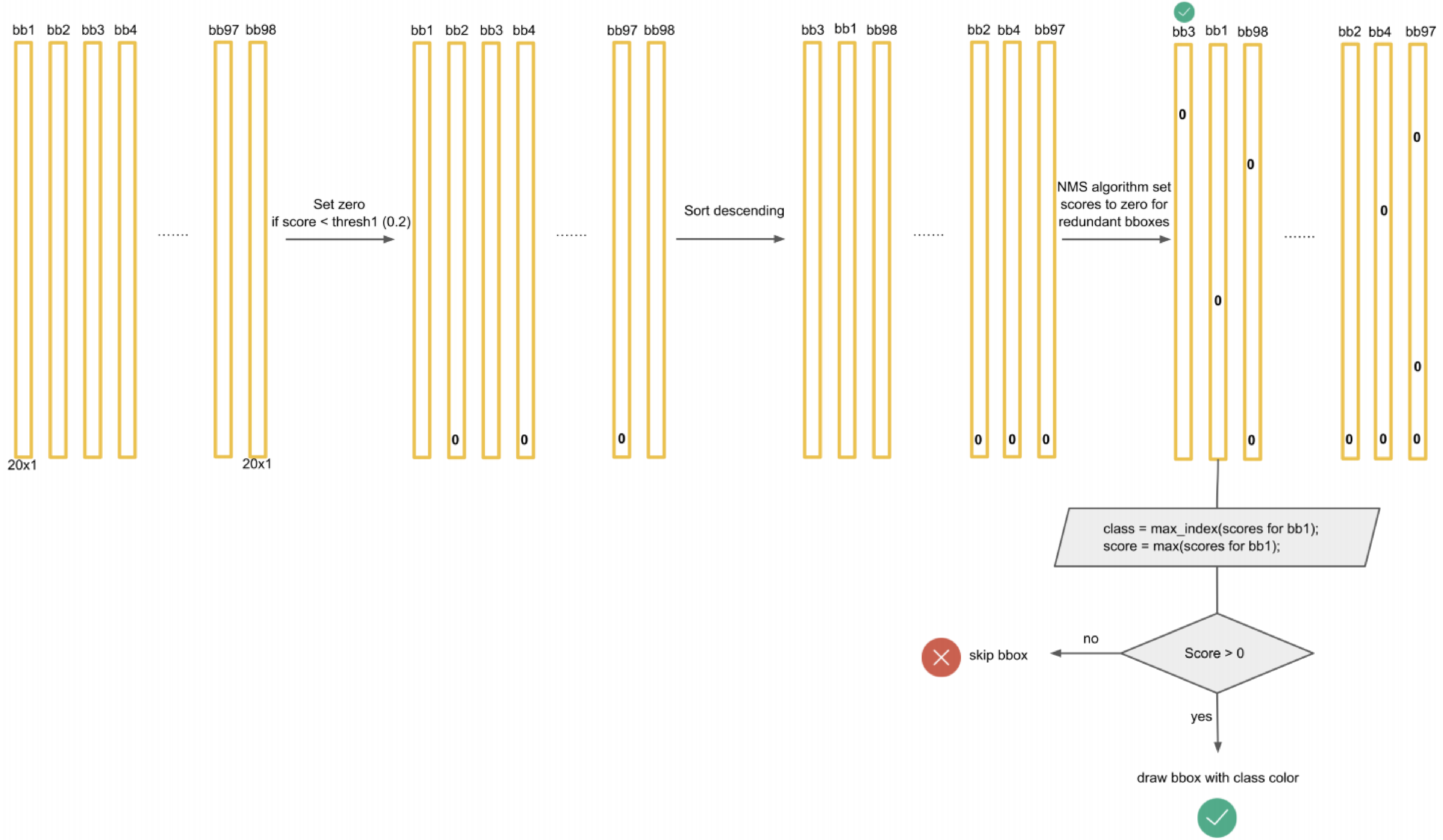



After this procedure - a lot of zeros

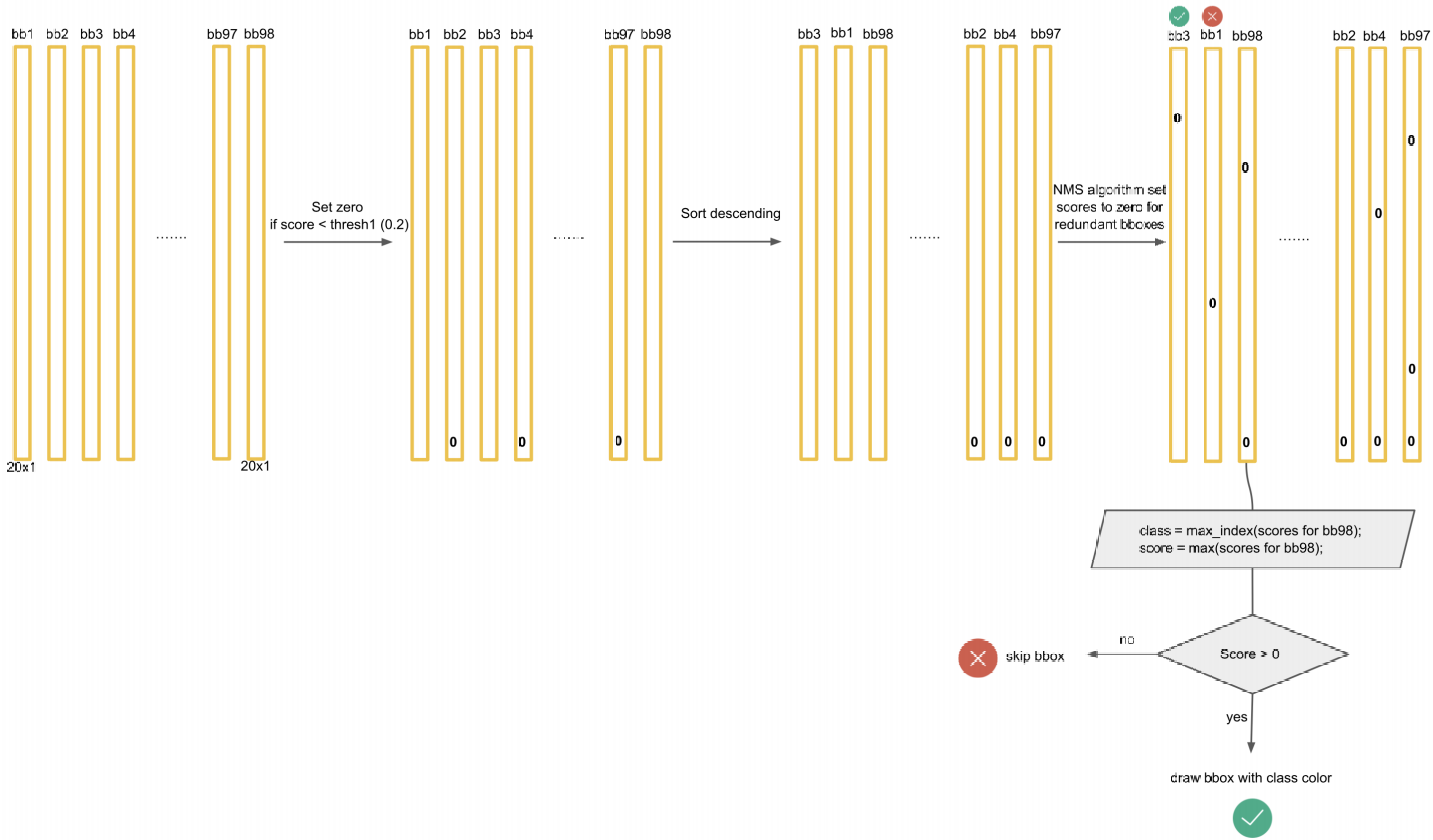

bb1 bb2 bb3 bb4 bb97 bb98
person scores
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 bb97 bb98
0 0 0
Sort descending
bb3 bb1 bb98 bb2 bb4 bb98
0 0 0
NMS algorithm set
scores to zero for
redundant bboxes
bb3 bb1 bb98 bb2 bb4 bb98
0 0 0 0 0 0 0 0 0 0 0
Select bboxes to draw by
class score values

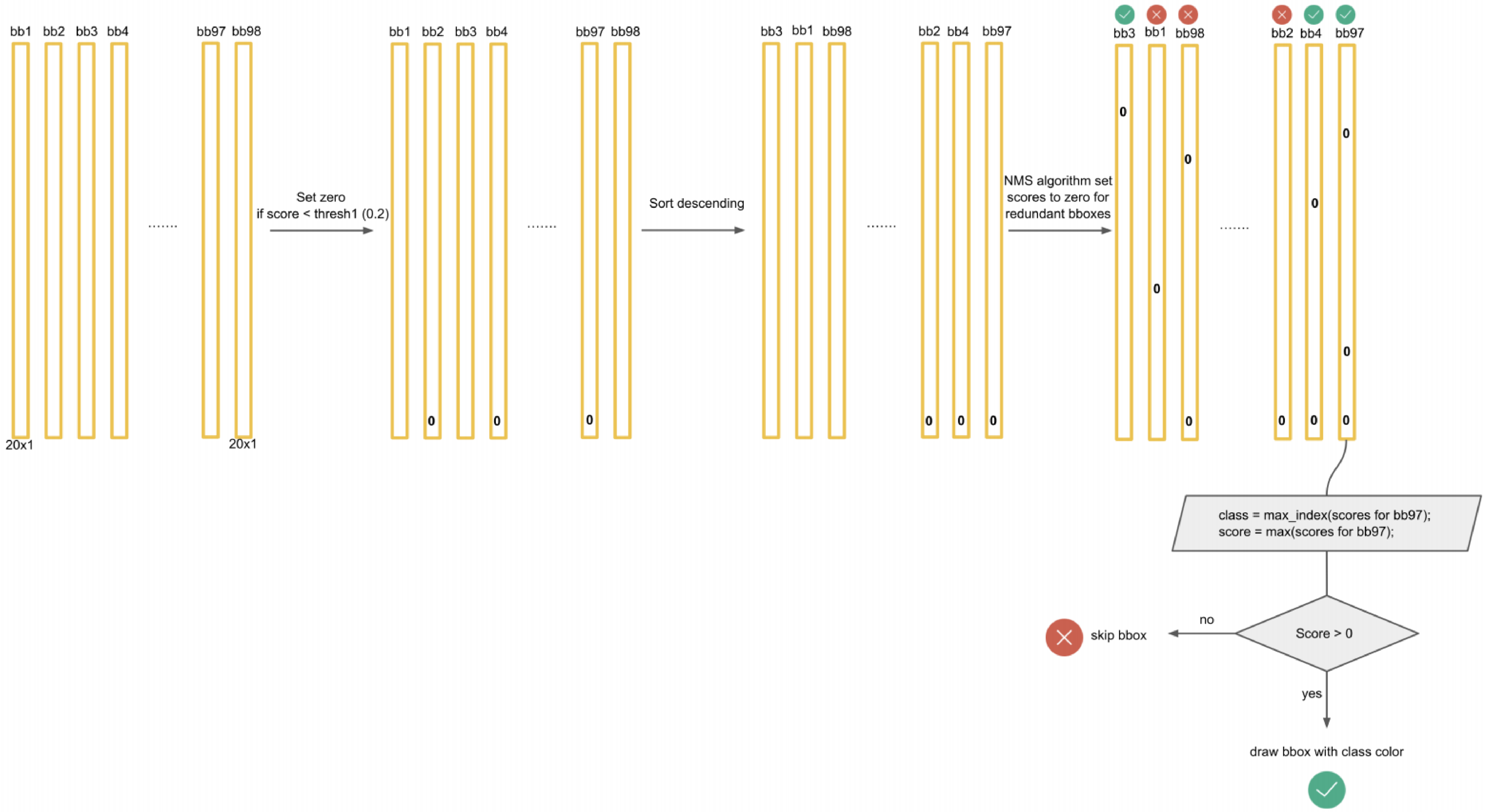


bb1 bb2 bb3 bb4 bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 bb97 bb98
Sort descending
bb3 bb1 bb98 bb2 bb4 bb97
NMS algorithm set scores to zero for redundant bboxes
bb3 bb1 bb98 bb2 bb4 bb97
class = max_index(scores for bb3);
score = max(scores for bb3);
Score > 0
no
skip bbox
yes
draw bbox with class color

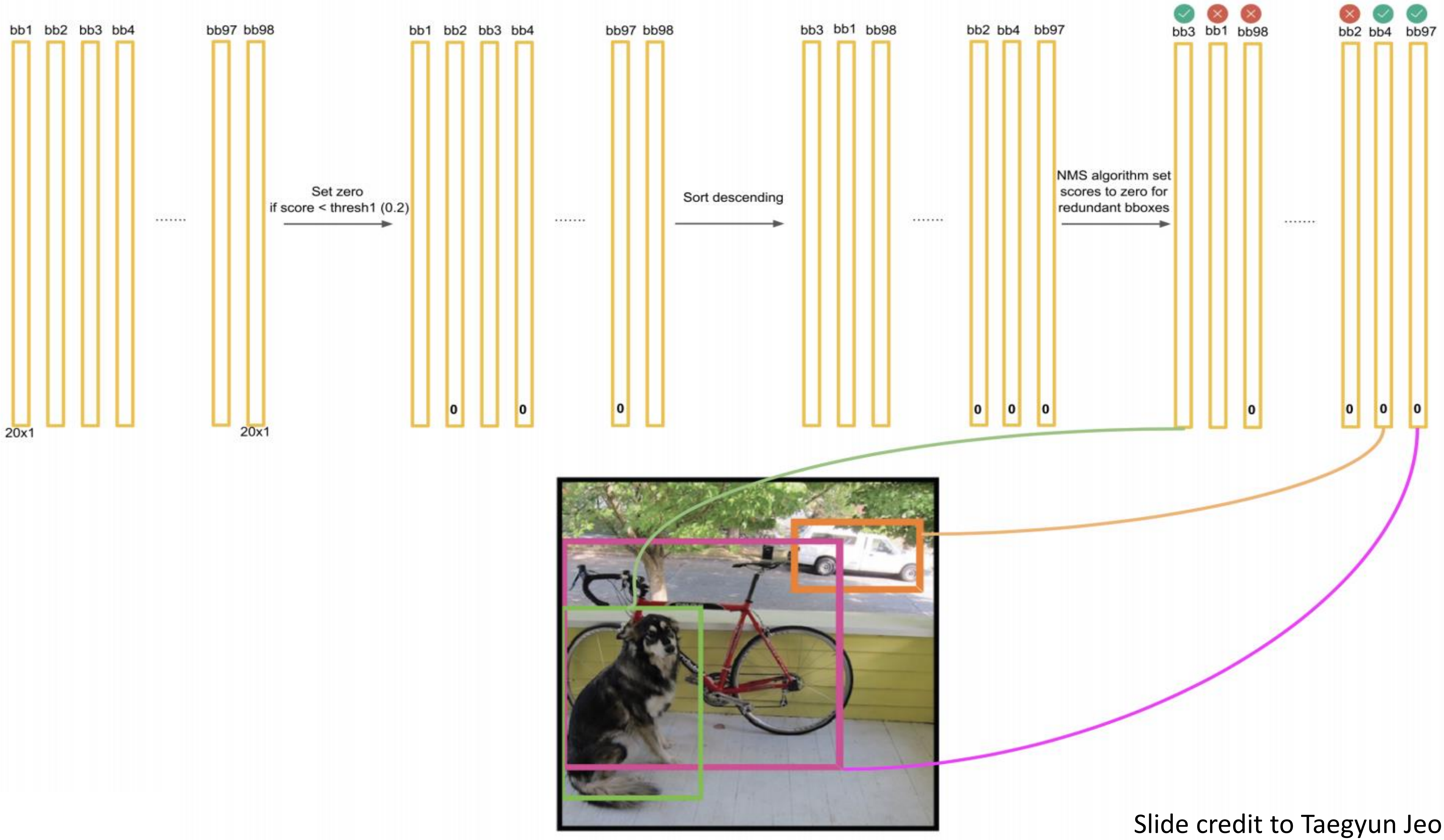


bb1 bb2 bb3 bb4 ....... bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 ....... bb97 bb98
0 0 0
Sort descending
bb3 bb1 bb98 ....... bb2 bb4 bb97
0 0 0
NMS algorithm set
scores to zero for
redundant bboxes
bb3 bb1 bb98 ....... bb2 bb4 bb97
0 0 0 0 0 0 0 0 0 0 0
class = max_index(scores for bb1);
score = max(scores for bb1);
Score > 0
no
skip bbox
yes
draw bbox with class color

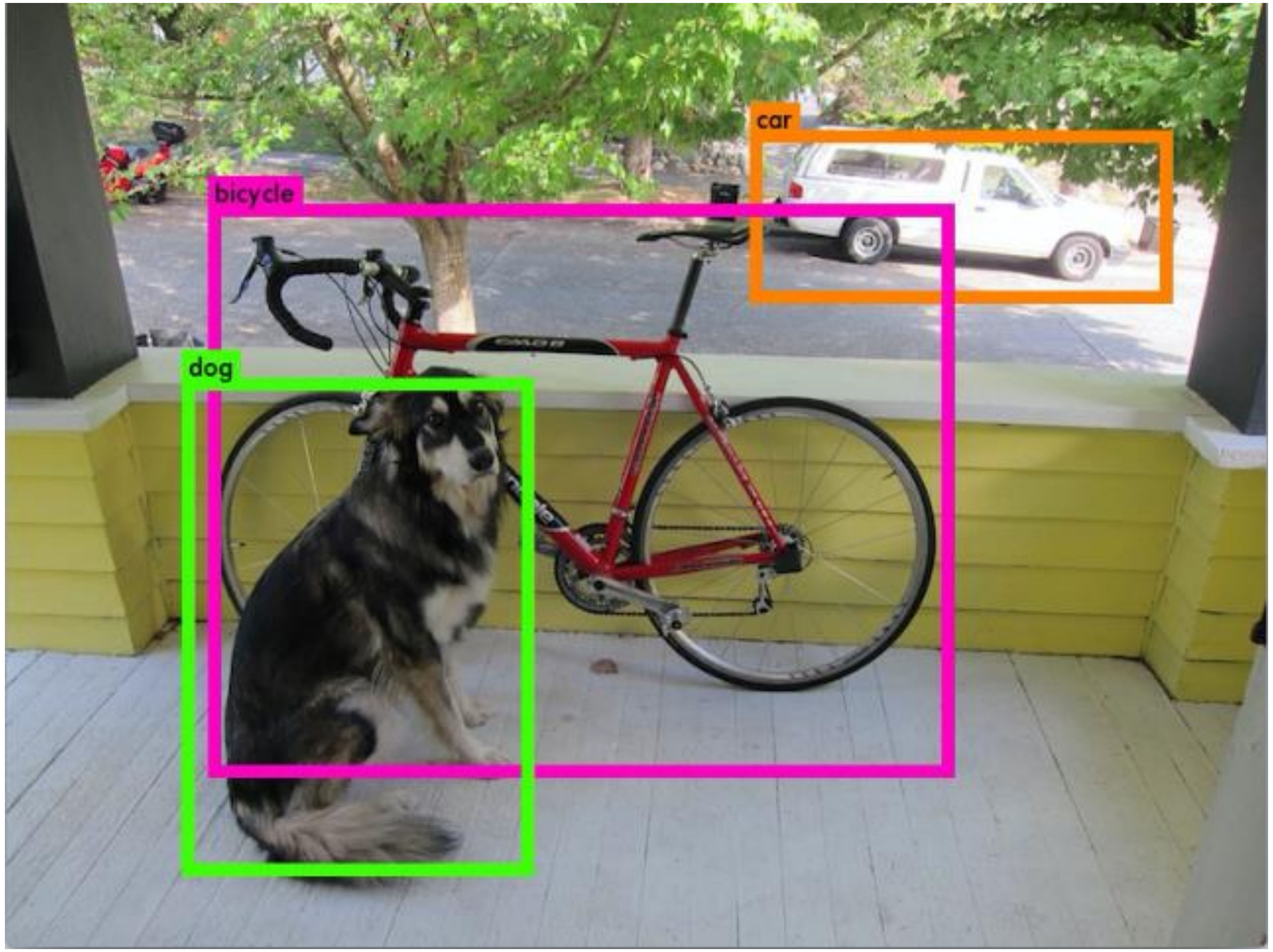

bb1
bb2
bb3
bb4
bb97
bb98
20x1
20x1
.......
Set zero
if score < thresh1 (0.2)
bb1
bb2
bb3
bb4
bb97
bb98
0
0
0
Sort descending
bb3
bb1
bb98
bb2
bb4
bb97
0
0
0
NMS algorithm set
scores to zero for
redundant bboxes
bb3
bb1
bb98
bb2
bb4
bb97
0
0
0
0
0
0
0
0
0
0
class = max_index(scores for bb98);
score = max(scores for bb98);
Score > 0
no
skip bbox
yes
draw bbox with class color

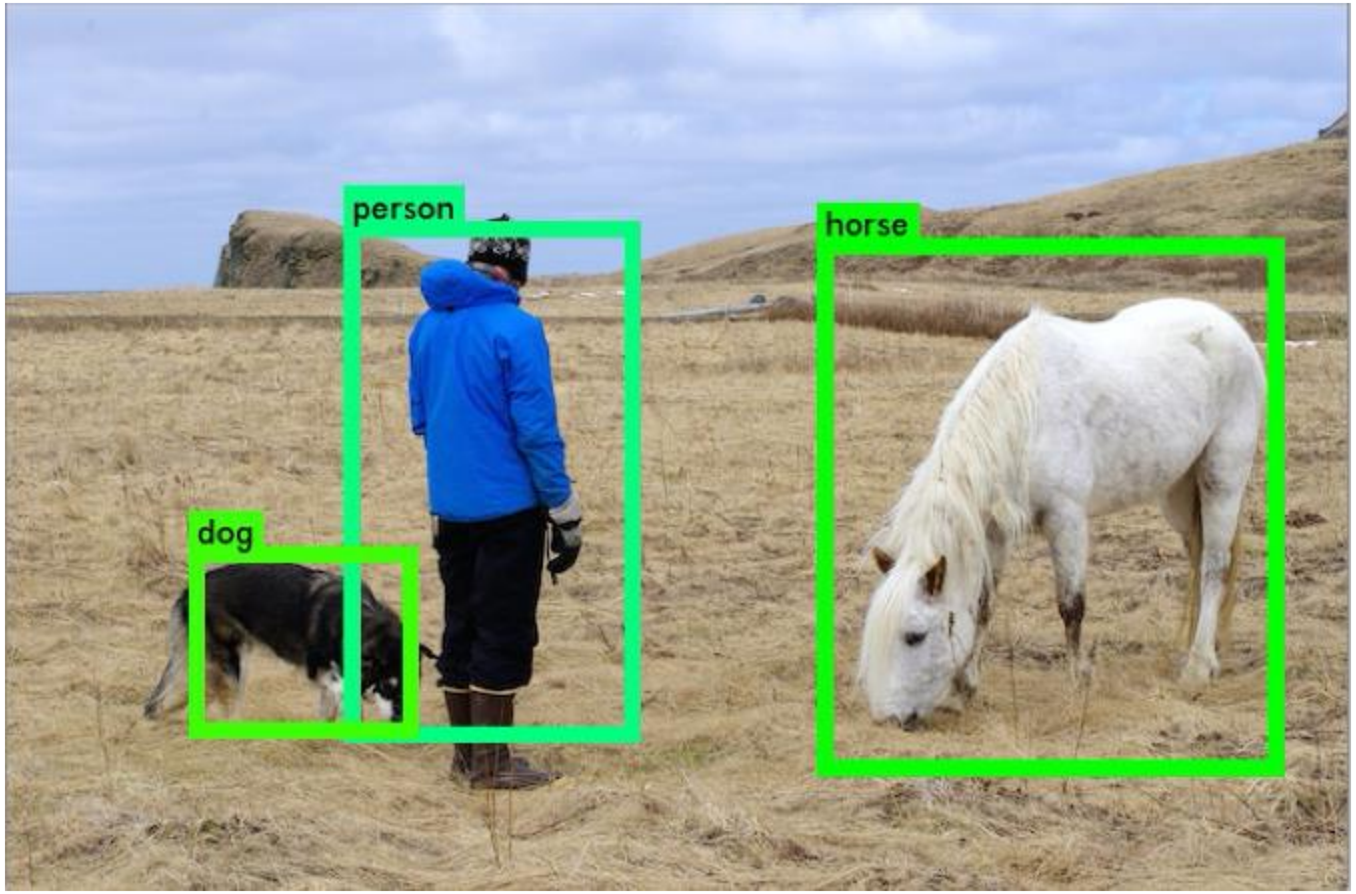


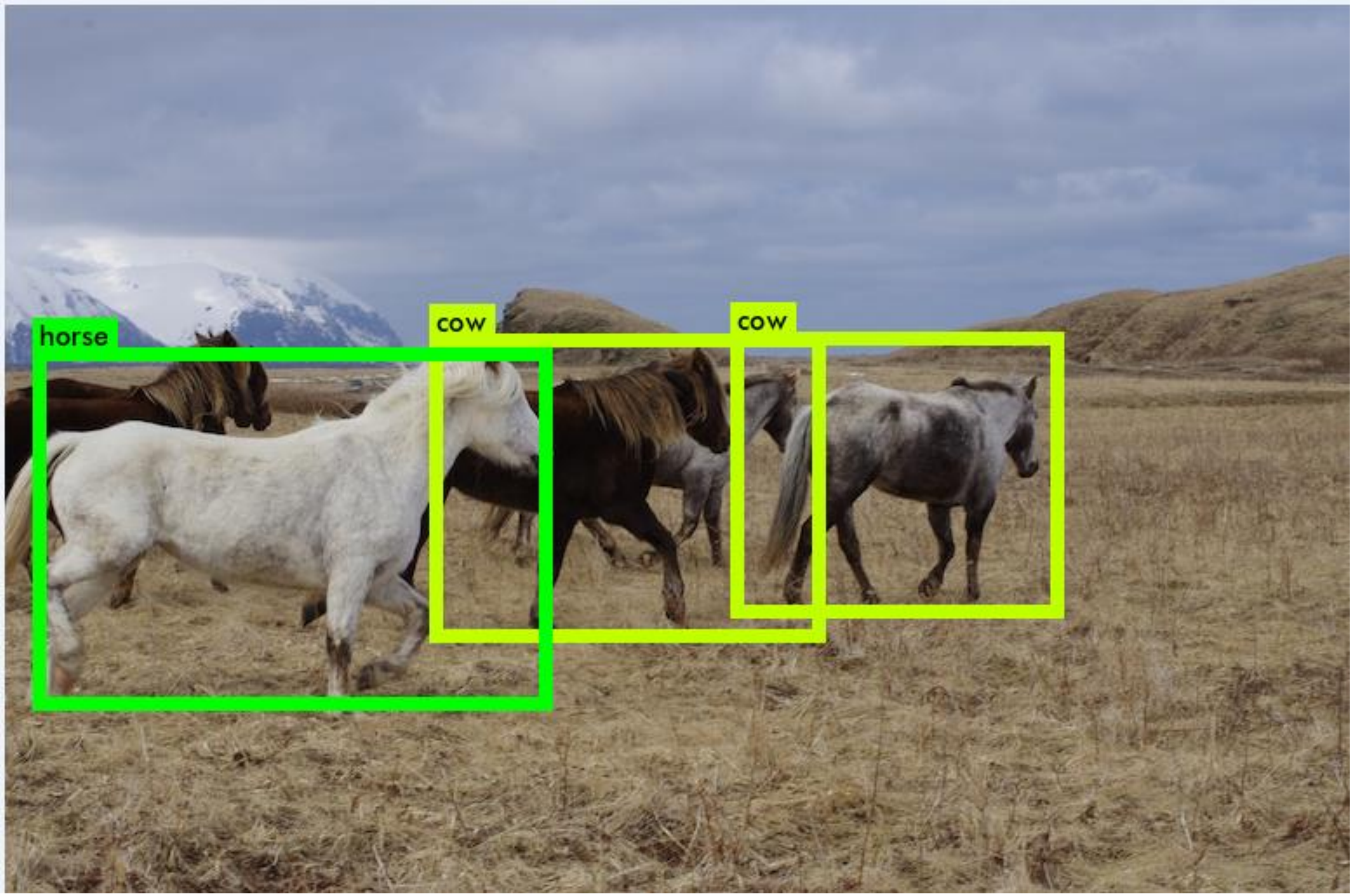
bb1 bb2 bb3 bb4 ....... bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 ....... bb97 bb98
0 0 0
Sort descending
bb3 bb1 bb98 ....... bb2 bb4 bb97
0 0 0
NMS algorithm set scores to zero for redundant bboxes
bb3 bb1 bb98 ....... bb2 bb4 bb97
0
0
0
0
0
0
0 0 0
class = max_index(scores for bb97);
score = max(scores for bb97);
Score > 0
no
skip bbox
yes
draw bbox with class color




Slide credit to Taegyun Jeon


car
bicycle
dog


person
horse
dog


horse
cow
cow

# Strengths and Weaknesses

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

# Limitation of YOLO

- Group of small objects
- Unusual aspect ratios
- Coarse feature
- Localization error of bounding box

### 2.4. Limitations of YOLO

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. Our model struggles with small objects that appear in groups, such as flocks of birds.

Since our model learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations. Our model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple downsampling layers from the input image.

Finally, while we train on a loss function that approximates detection performance, our loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU. Our main source of error is incorrect localizations.

# 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 $288 \times 288$ | 2007+2012 | 69.0 | 91 |
| YOLOv2 $352 \times 352$ | 2007+2012 | 73.7 | 81 |
| YOLOv2 $416 \times 416$ | 2007+2012 | 76.8 | 67 |
| YOLOv2 $480 \times 480$ | 2007+2012 | 77.8 | 59 |
| YOLOv2 $544 \times 544$ | 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

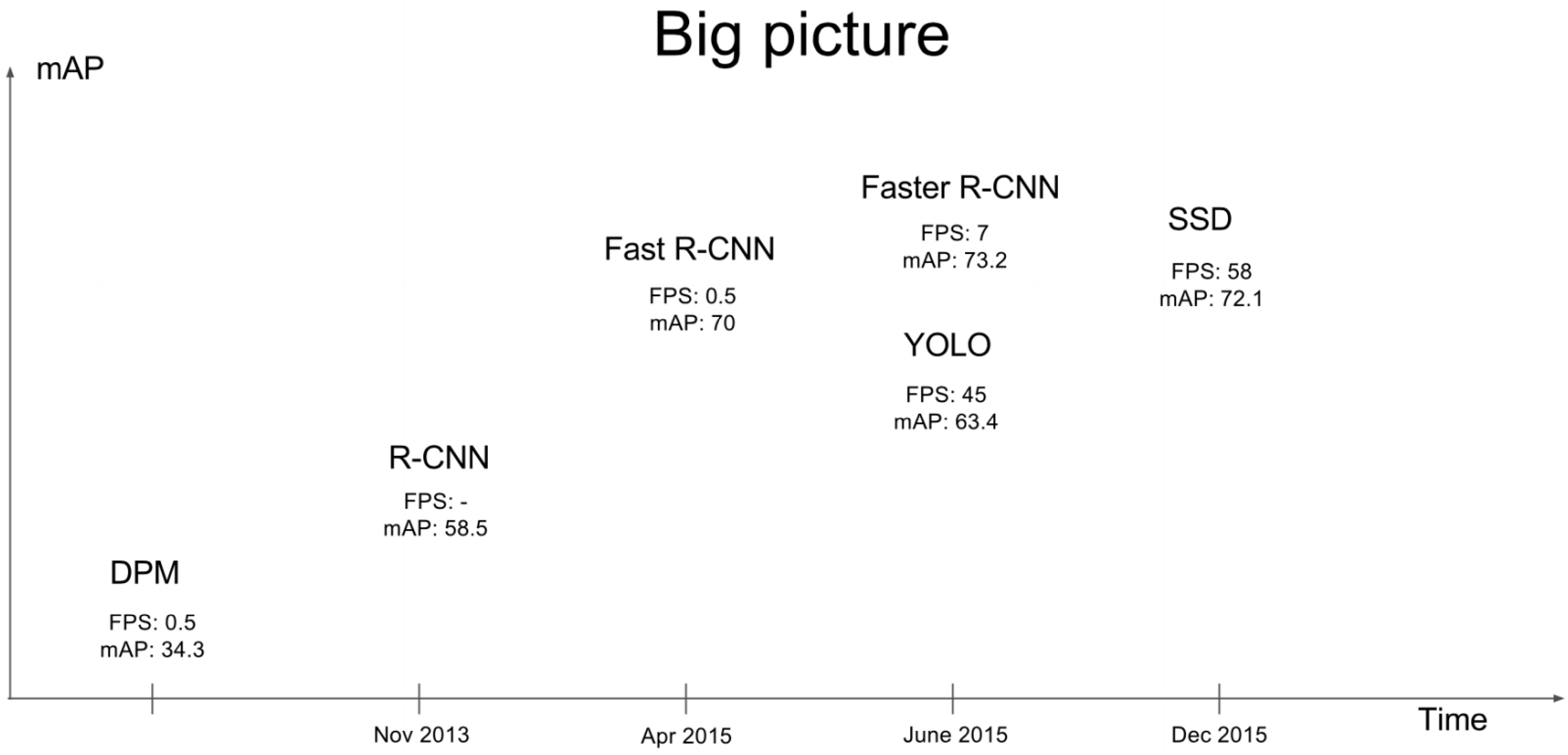
# Improvement

- **YOLO**: (1) make each cell predict more bounding boxes
  (2) also put the idea of multi-scale into it in order to process small objects

Features

- **SSD**: simplify its structure and speed up

# Comparison to Other Detection System



# Appendix | Implementation

- YOLO (darknet): https://pjreddie.com/darknet/yolov1/
- YOLOv2 (darknet): https://pjreddie.com/darknet/yolo/
- YOLO (caffe): https://github.com/xingwangsfu/caffe-yolo
- YOLO (TensorFlow: Train+Test): https://github.com/thtrieu/darkflow
- YOLO (TensorFlow: Test): https://github.com/gliese581gg/YOLO_tensorflow

# SSD: Single Shot MultiBox Detector

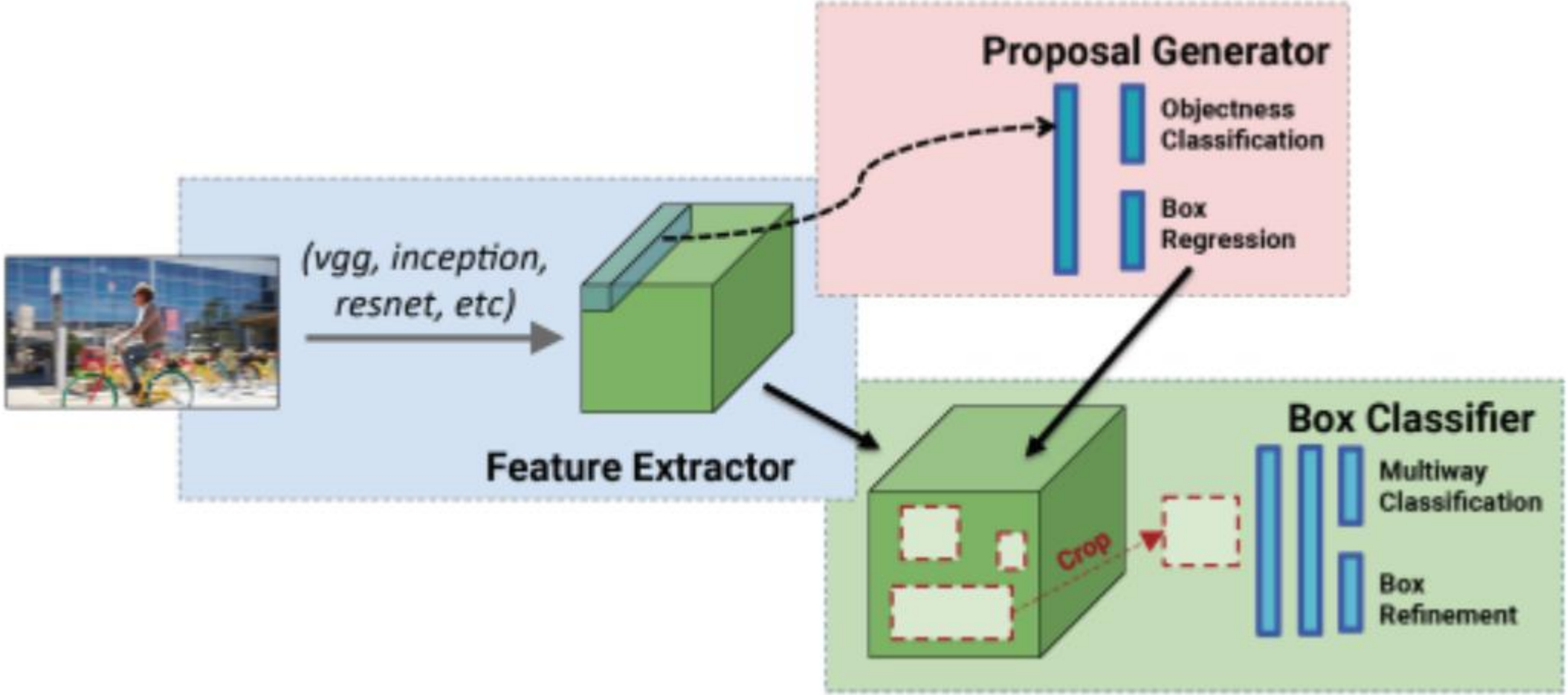
# SSD: Single Shot MultiBox Detector

Wei Liu[1], Dragomir Anguelov[2], Dumitru Erhan[3], Christian Szegedy[3], Scott Reed[4], Cheng-Yang Fu[1], Alexander C. Berg[1]

[1]UNC Chapel Hill [2]Zoox Inc. [3]Google Inc. [4]University of Michigan, Ann-Arbor
[1]wliu@cs.unc.edu, [2]drago@zoox.com, [3]{dumitru,szegedy}@google.com,
[4]reedscot@umich.edu, [1]{cyfu,aberg}@cs.unc.edu

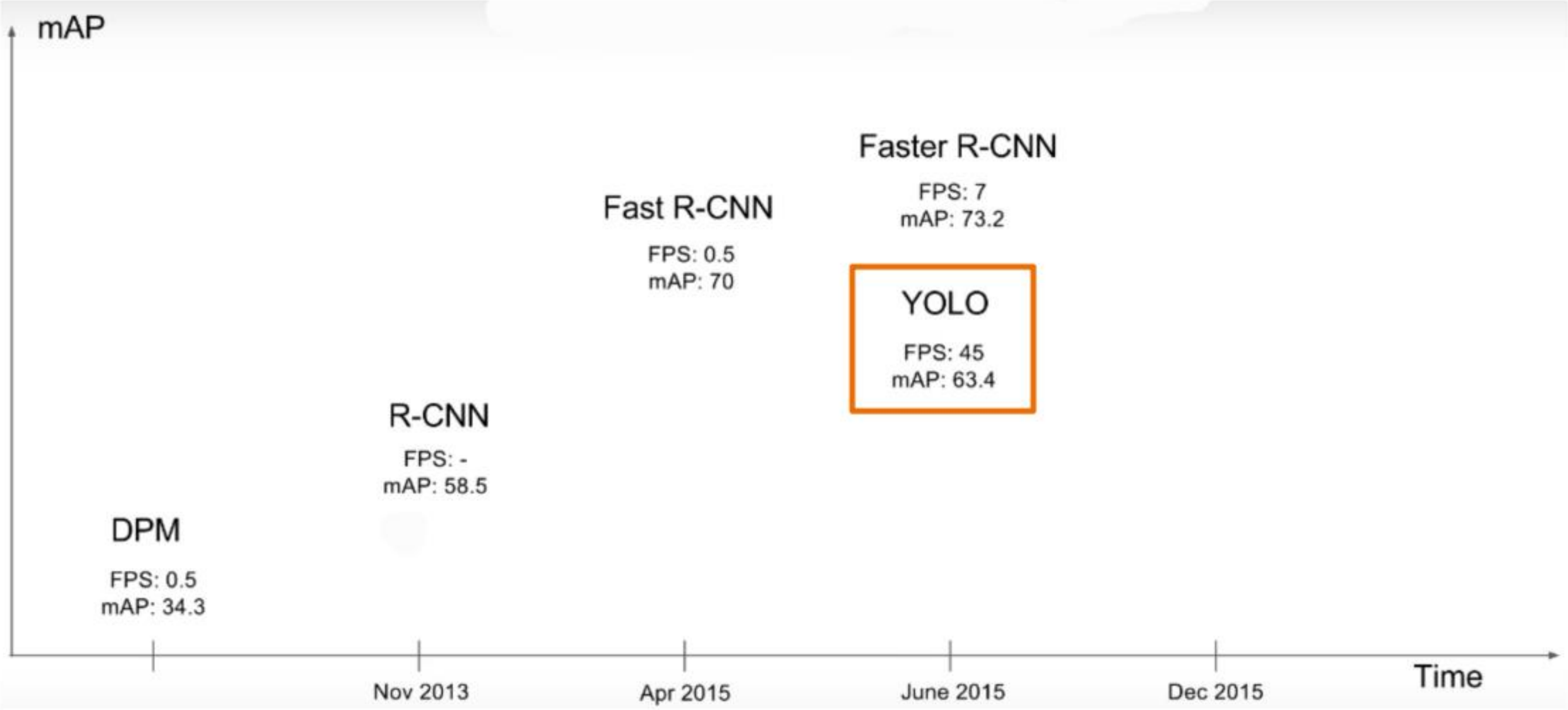
**Abstract.** We present a method for detecting objects in images using a single deep neural network. Our approach, named SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. Our SSD model is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stage and encapsulates all computation in a single network. This makes SSD easy to train and straightforward to integrate into systems that require a detection component. Experimental results on the PASCAL VOC, MS COCO, and ILSVRC datasets confirm that SSD has comparable accuracy to methods that utilize an additional object proposal step and is much faster, while providing a unified framework for both training and inference. Compared to other single stage methods, SSD has much better accuracy, even with a smaller input image size. For $300 \times 300$ input, SSD achieves 72.1% mAP on VOC2007 `test` at 58 FPS on a Nvidia Titan X and for $500 \times 500$ input, SSD achieves 75.1% mAP, outperforming a comparable state of the art Faster R-CNN model. Code is available at https://github.com/weiliu89/caffe/tree/ssd.

Faster R-CNN: Box Classification and Regression are being done 2 times.




Two stage object detection is time-consuming. Faster R-CNN is faster but not fast enough.

# YOLO: Fast but not accurate enough.

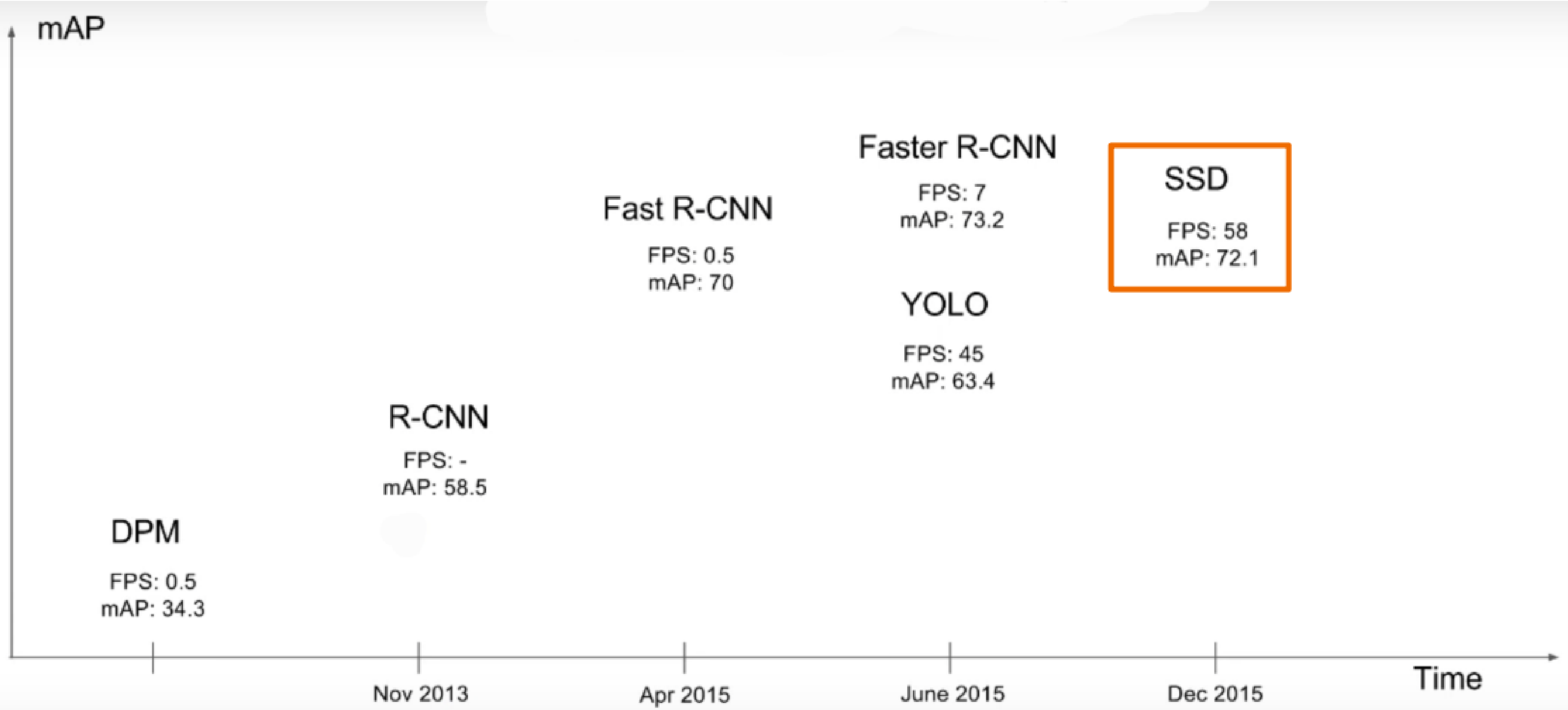



YOLO: Fast but not accurate enough.

Object detection needs a good tradeoff between accuracy and speed.

Slide courtesy of DeepSystem.io “YOLO: You only look once Review”

# SSD: Single Shot MultiBox Detection (ECCV 2016)




1. Achieves competitive mAP (72.1) as faster R-CNN (73.2).
2. Much faster (58 fps) than faster R-CNN (7fps) and YOLO (45fps), making accurate real-time detection possible.
3. Makes predictions on multiple feature maps with different resolutions to handle objects of different sizes.

# SSD: Single Shot MultiBox Detector

? Tool?-- A single deep neural network

? Framework?– Caffe

? Technology background?--related methods are structure-complicated and hard to bring high speed and good accuracy

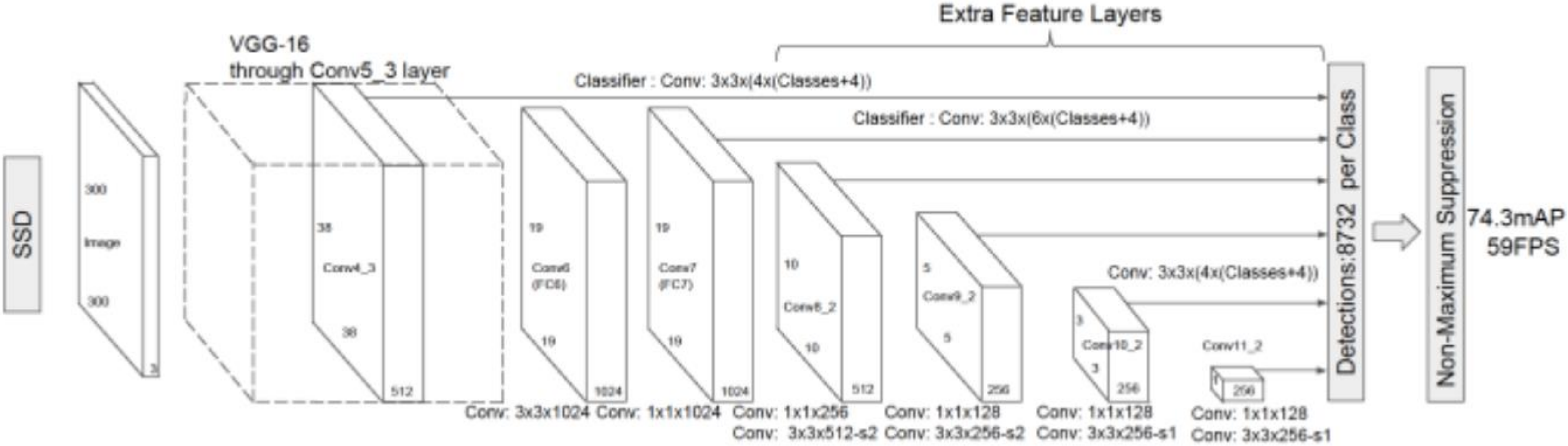
- Solution and Advantages:

@Providing a unified framework

@much faster

@better accuracy, even with a smaller input image size

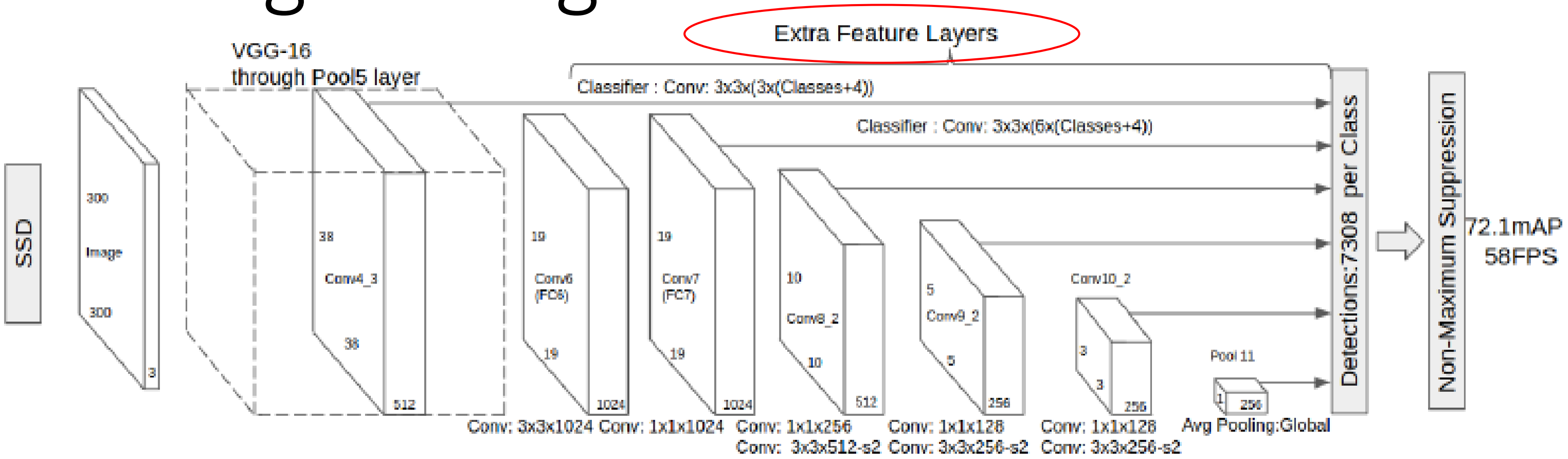
# Network Architecture




VGG network - extract features

Detectors on multiple feature maps

# How SSD gets its goal?

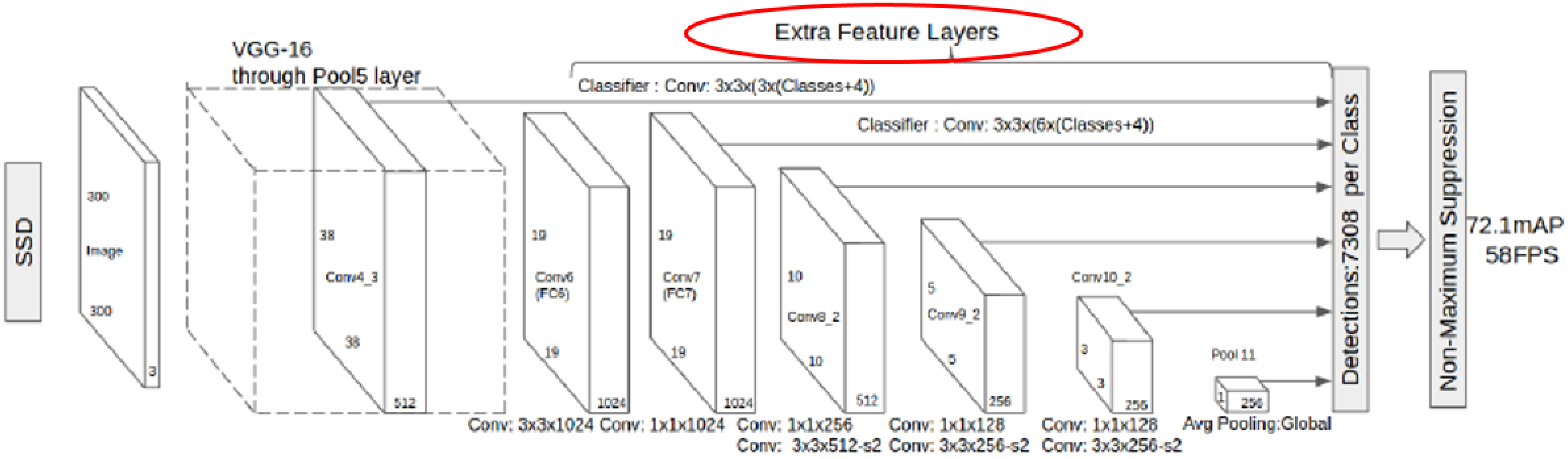



- For speed: structure advantage!

➢ Eliminating bounding box proposals and subsequent pixel or feature resampling stage

✓ Adding convolution feature layers to the end of the tructed base network to predict detections at multiple scalaes

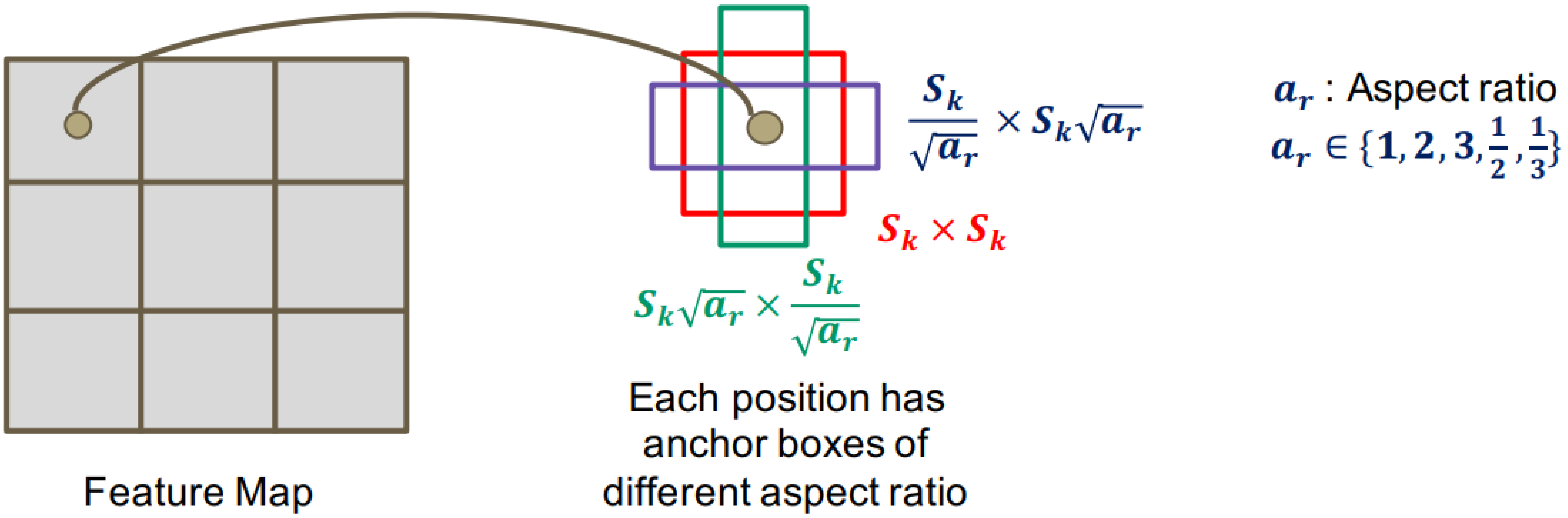
# How SSD gets its goal?




- For speed: structure advantage!
- Eliminating bounding box proposals and subsequent pixel or feature resampling stage
- Adding convolution feature layers to the end of the tructed base network to predict detections at multiple scalaes

# Default (Anchor) Boxes

- Similar to faster R-CNN, SSD also uses anchor boxes.
- At each feature map position, anchor boxes have different aspect ratios.

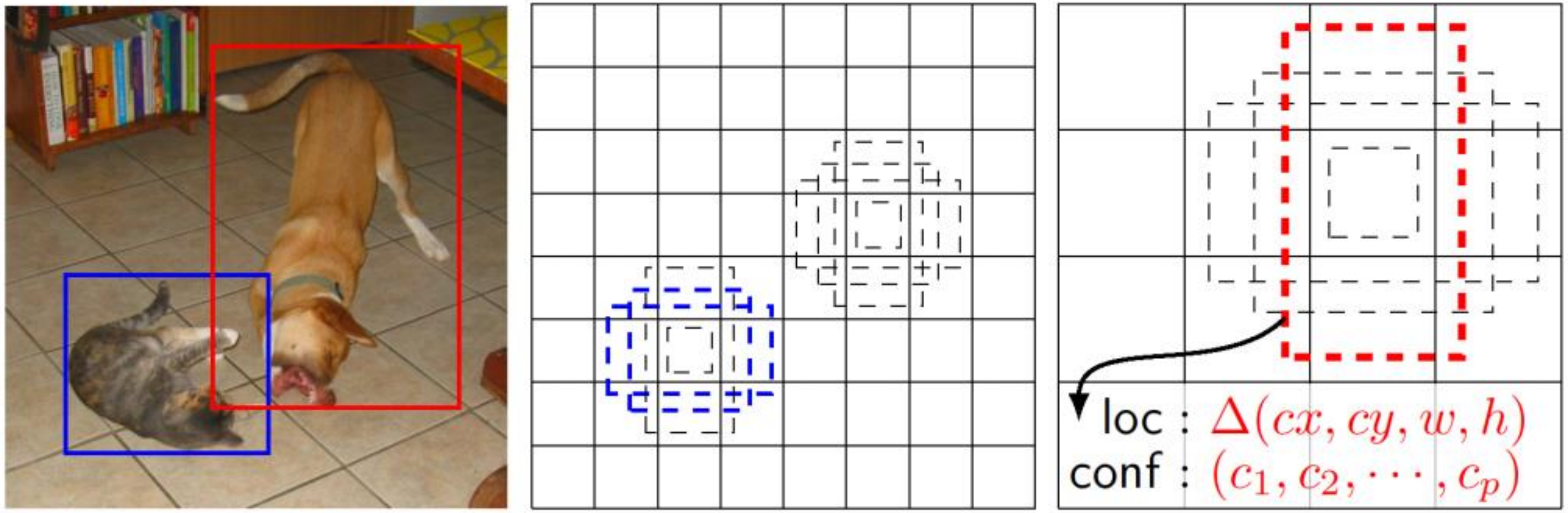


# Default (Anchor) Boxes

- Anchor boxes of different feature maps have unique scales. So different feature map are responsible for objects of different sizes.

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m - 1}(k - 1), \quad k \in [1, m]$$

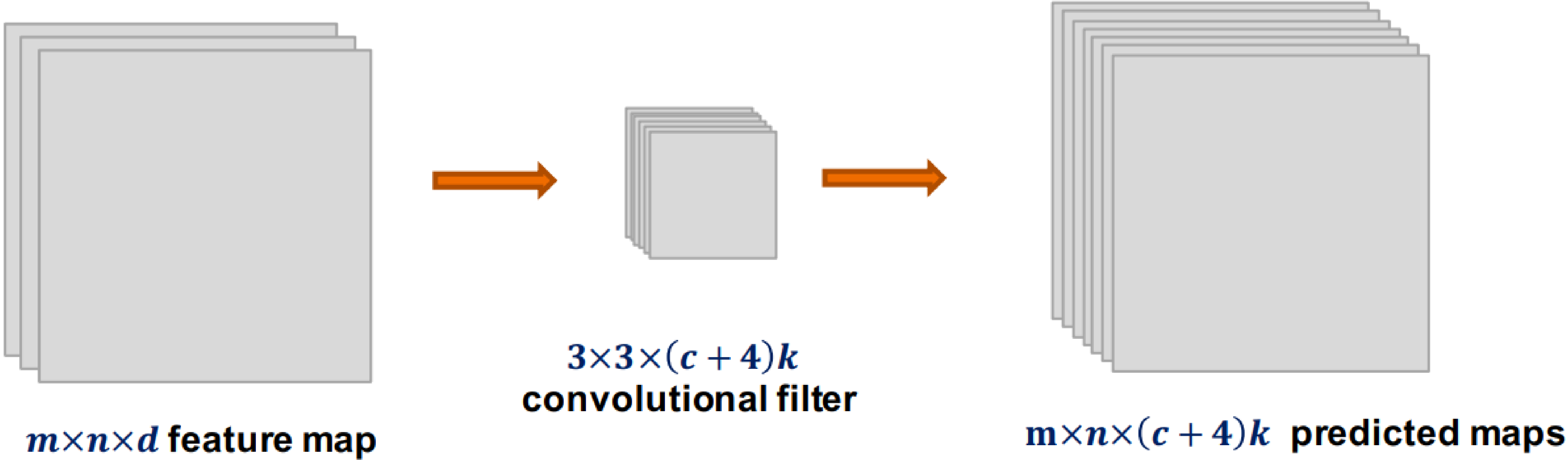
- $\boldsymbol{S_k}$ denotes the scale of of the k-th feature map. $m$ is the number of feature maps for prediction. $\boldsymbol{S_{min}} = 0.2$, $\boldsymbol{S_{max}} = 0.9$.



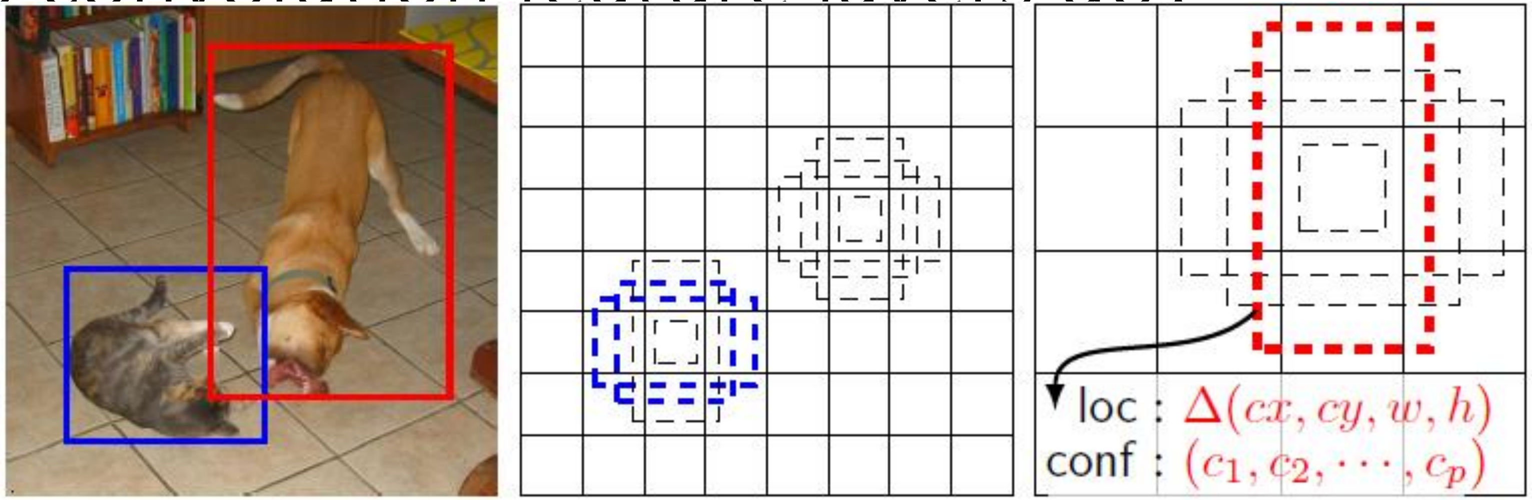

(a) Image with GT boxes (b) 8 × 8 feature map (c) 4 × 4 feature map

# Convolutional Filters for Prediction

- On each feature map, two types of convolutional filters will be applied:
    - C filters for category prediction, where C is the number of object categories.
    - 4 filters for bounding box regression, 4 for the coordinates x, y, w, h.
- Together there will be $(c+4)k$ filters for each feature map, $k$ is the number of anchor box types.
- The output for a $m \times n$ feature map will be a map of $m \times n \times (c+4)k$, indicating the category and coordinates for each bounding box.



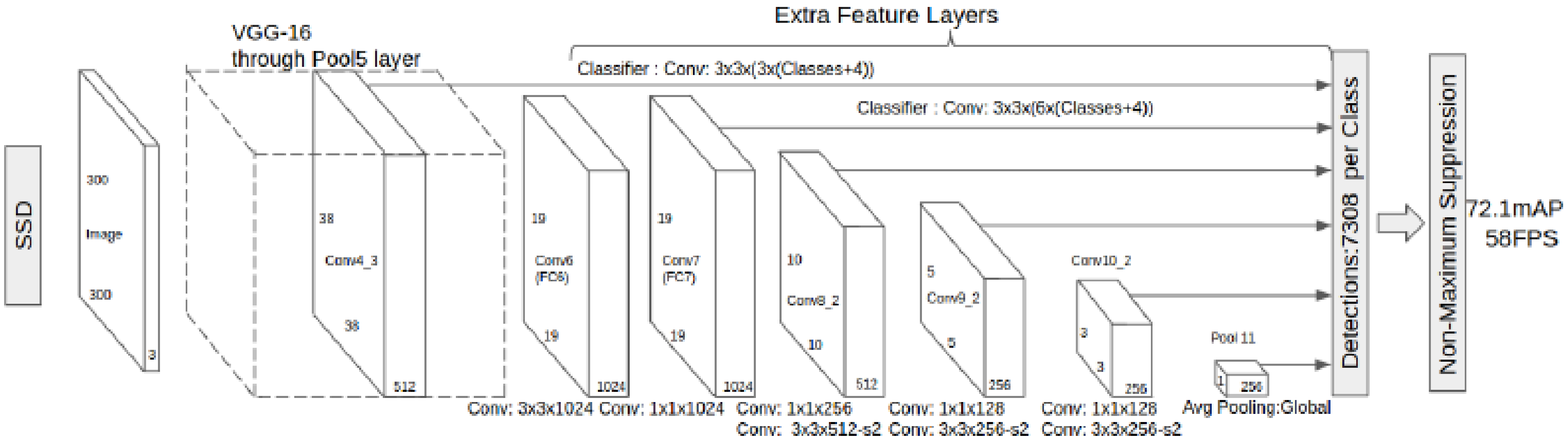
# What do convolution feature layers do?

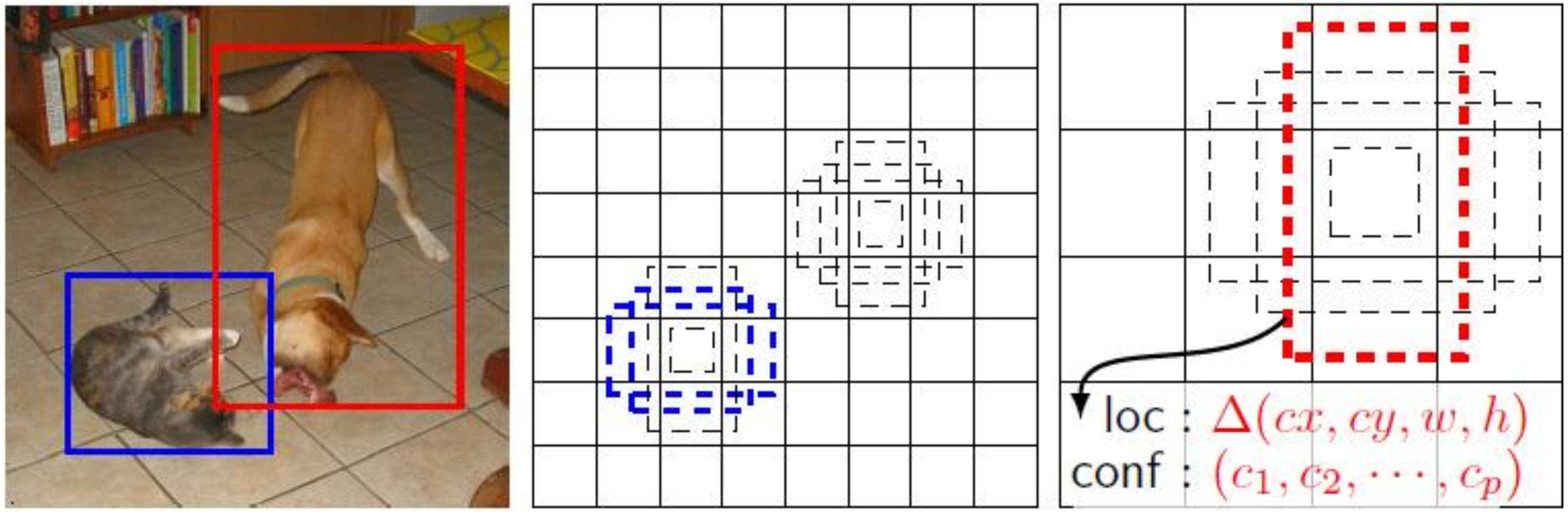



(a) Image with GT boxes (b) $8 \times 8$ feature map (c) $4 \times 4$ feature map

- Needs an input image and ground truth boxes for each object during training
- Evaluate a small set(e.g.4)of default boxes of different aspect ratios at each locations in several feature maps with different scales.
- Filters match these default boxes to the ground truth boxes and predict both the shape offsets and confidence for all object categories for each default box
- The feed-forward convolutional network produces a fixed-size collection of bounding boxes and scores for the presence of object class in those boxes

# SSD overcoming the influence of resolution difference



# Training objective

- The loss used in SSD is a combination of confidence loss and localization loss.

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$$

- Confidence loss is the softmax loss over multiple classes confidences.

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0)$$

- Localization loss is the same smooth L1 loss as faster R-CNN.

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

# Matching Strategy

- During training, we need to determine which anchor boxes are corresponding to ground truth boxes.
    1. Match each ground truth box to the anchor box with the best jaccard overlap.
    2. Match default box to any ground truth with jaccard overlap higher than 0.5.
- Question : Why not using predicted box (anchor box after regression) for matching?
- Question : Negative anchor boxes are much more than positive anchor boxes, how to deal with this imbalance?

# Hard Negative Mining

- Number of negative anchor boxes >> number of positive anchor boxes
- Calculate the **confidence loss** of each negative anchor boxes.
- Select anchor boxes that have **highest** loss as negative training samples.
- Keep the ratio between negatives and positives **3:1**.
- This leads to faster convergence and stable training.

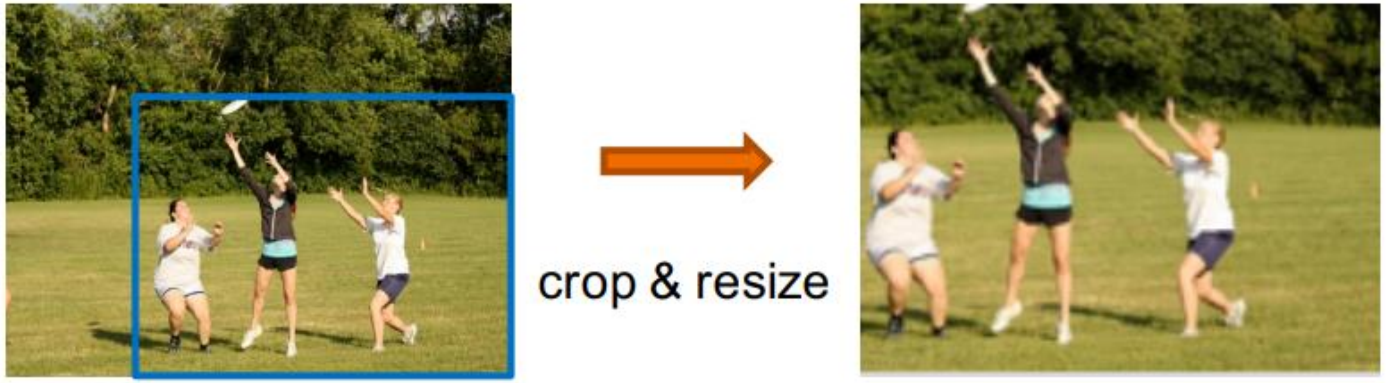
- Question: Is this approach sufficient for training negative samples?
    - **No**. A recent paper show that negative samples need more elegant loss calculation.
    - (ICCV 2017 best student paper award: *Focal Loss for Dense Object Detection*, Lin et al.)

- Base network: VGG16, pretrained on ILSVRC dataset.
- Add layers on top of VGG Conv5 layer.
- Use dilated convolution in Conv6 layer.

# Results – small objects

- SSD does not perform well on small objects.
- No feature resampling step in SSD.
- Relatively low-level feature maps are responsible for detecting small objects. These feature maps do not have sufficient high-level semantic information.

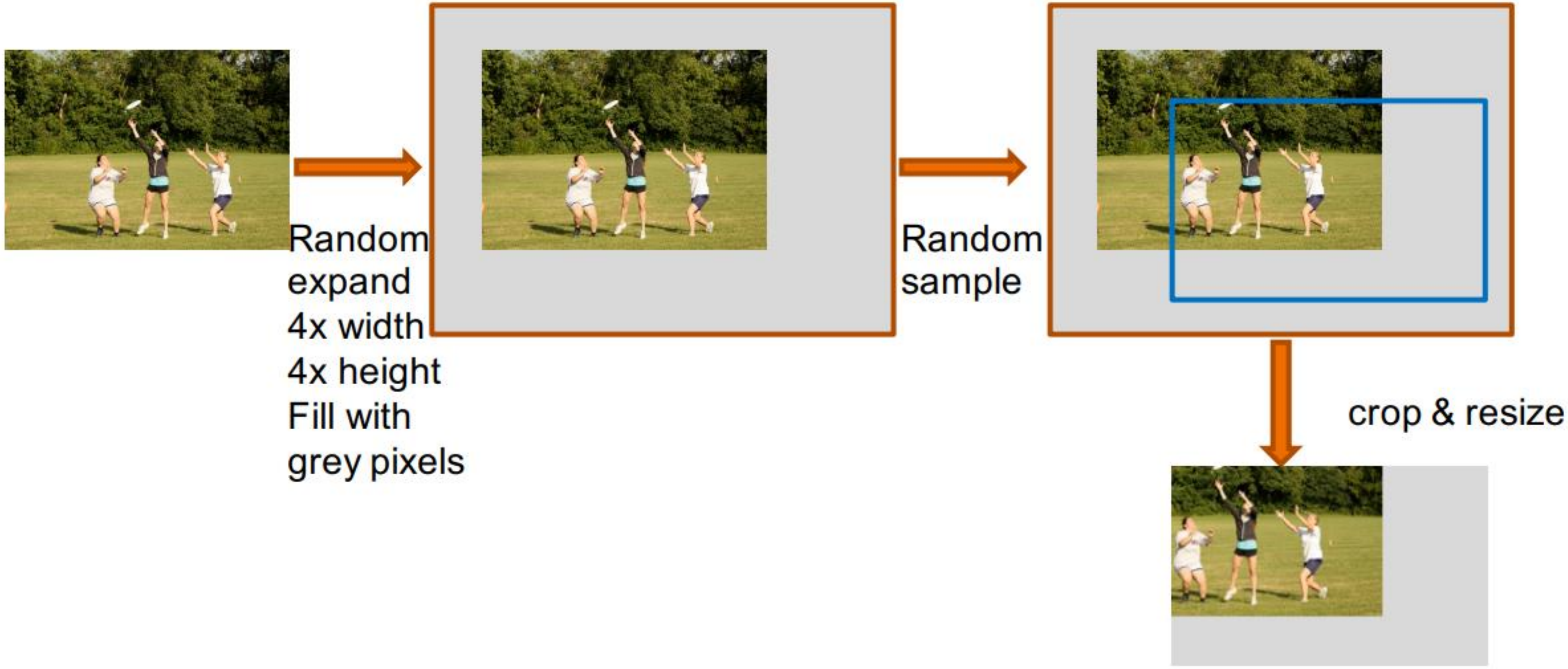
# Data Augmentation




- Photo-metric distortions.
- Random crop:
  - Use the entire original input image.
  - Crop a patch so that the minimum jaccard overlap with the objects is 0.1, 0.3, 0.5, 0.7 or 0.9.
  - Randomly crop a patch.
- Question: Will data augmentation also benefit this much to Faster R-CNN?
  - No, because faster R-CNN uses a feature pooling step which is relatively robust to object translation.

# More data augmentation for small objects

- Keep small objects small.



# Quantity choice

- Imposing different aspects ratios for the default boxes, and denote them as :

$$a_r = \left\lfloor 1, 2, 3, \frac{1}{2}, \frac{1}{3} \right\rfloor$$

- Instead of using all the negative examples, SSD sorts them using the highest confidence for each default box and pick the top ones so that the ratio between the negatives and positives is at most 3:1——leading to faster optimization and more stable training

# Quantity choice

$$L(x,c,l,g)=\frac{1}{N}(L_{conf}(x,c)+\alpha L_{loc}(x,l,g))$$

--The overall objective loss function is a weighted sum of the localization loss and the confidence loss(conf)
--N: the number of matched default boxes
--l: predicted boxes
--g: the ground truth box
--x=1 denotes some certain default box is matched to a ground truth box

# Choosing scales and aspect ratios for default boxes:

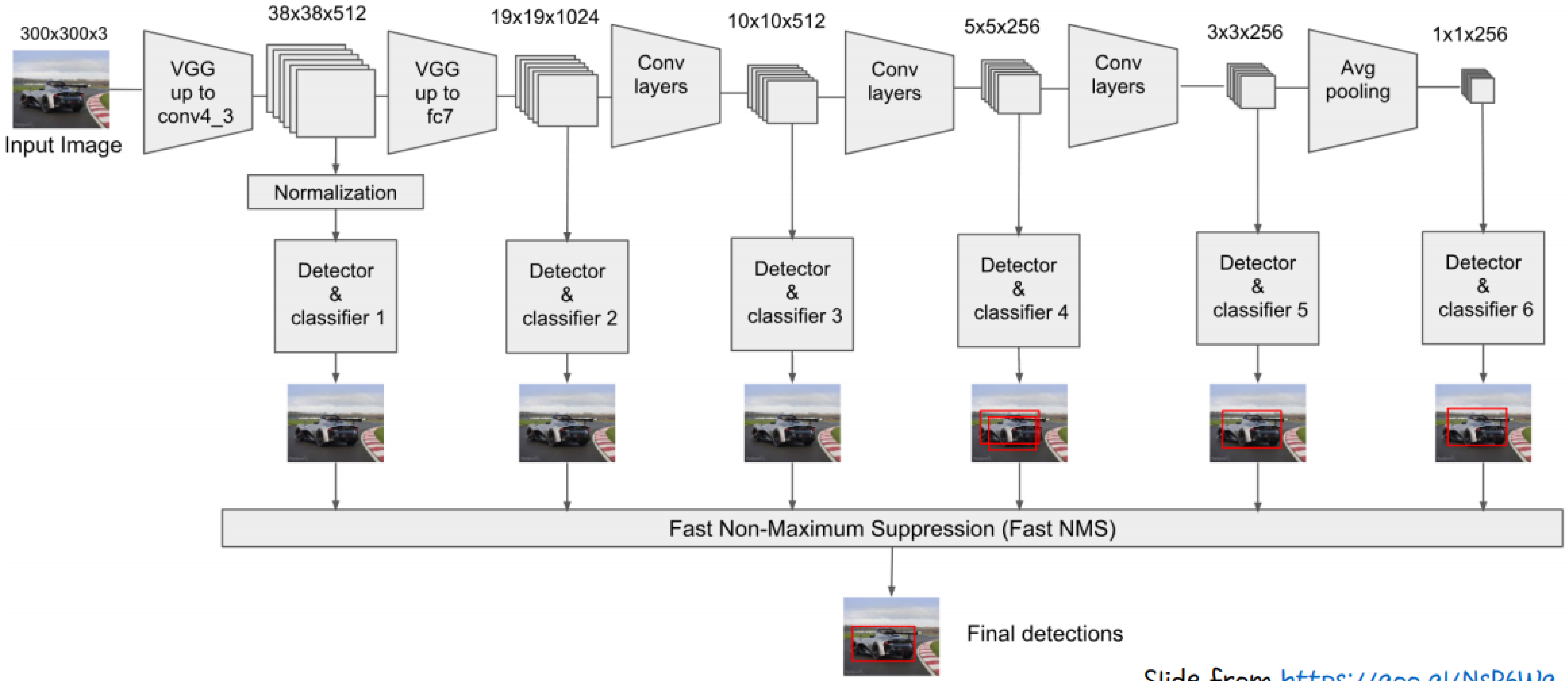
- If *m* feature maps are used for prediction, the **scale** of the default boxes for each feature map is:

$$s_k = s_{\text{min}} + \frac{s_{\text{max}} - s_{\text{min}}}{m - 1}(k - 1), \quad k \in [1, m]$$

$s_{\text{min}} =$ 0.1

$s_{\text{max}} =$ 0.95

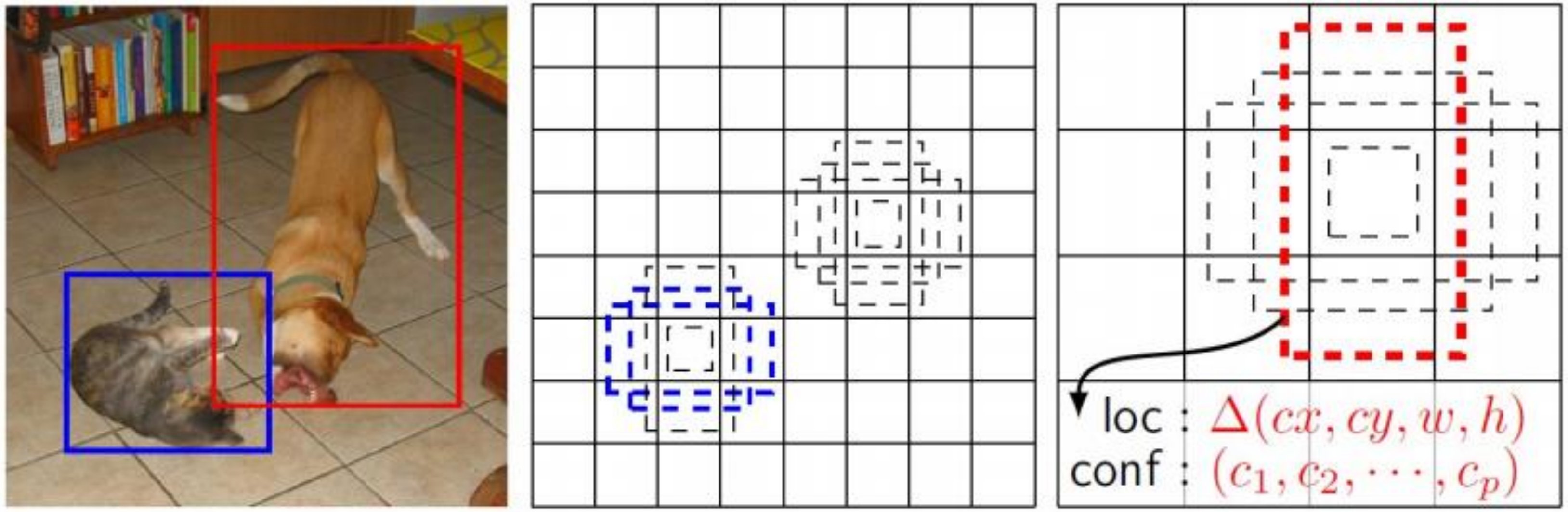
# Choosing Scales and Aspect Ratios for Default Boxes




Slide from https://goo.gl/NsP6Wg

# Choosing Scales and Aspect Ratios for Default Boxes

- At each scale, different aspect ratios are considered




(a) Image with GT boxes (b) $8 \times 8$ feature map (c) $4 \times 4$ feature map

$$a_r \in \{1, 2, 3, \tfrac{1}{2}, \tfrac{1}{3}\}$$

$$w_k^a = s_k\sqrt{a_r}$$

$$h_k^a = s_k/\sqrt{a_r}$$

For the aspect ratio of 1, one default box is added whose scale is

$$s'_k = \sqrt{s_k s_{k+1}}$$

# Hard Negative Mining

- Significant imbalance between positive and negative training examples.
  - After the matching step, most of the default boxes are negatives, especially when the number of possible default boxes is large.

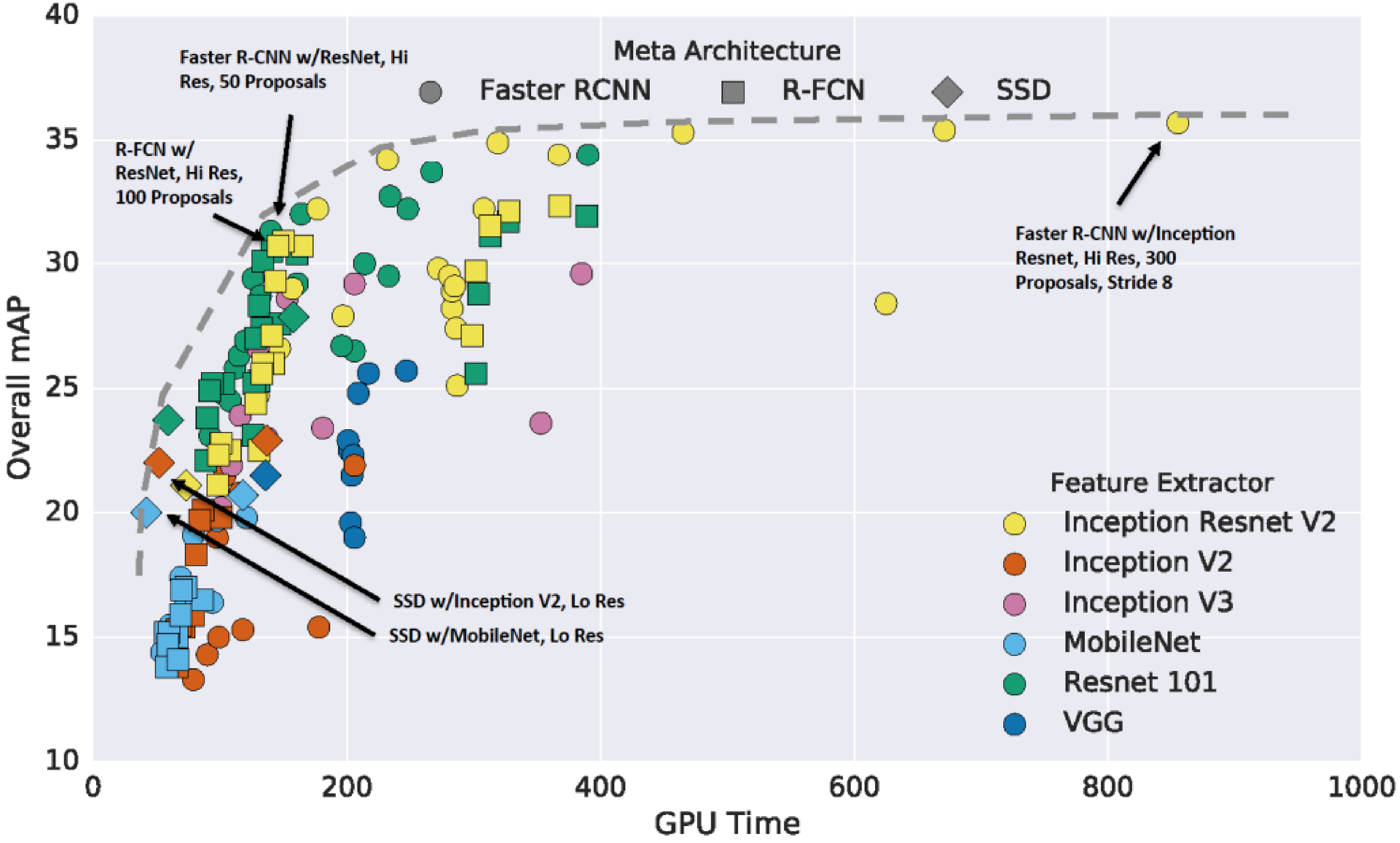
- Sorting them using the highest confidence loss for each default box.
- Pick the top ones so that the ratio between the negatives and positives is at most 3:1

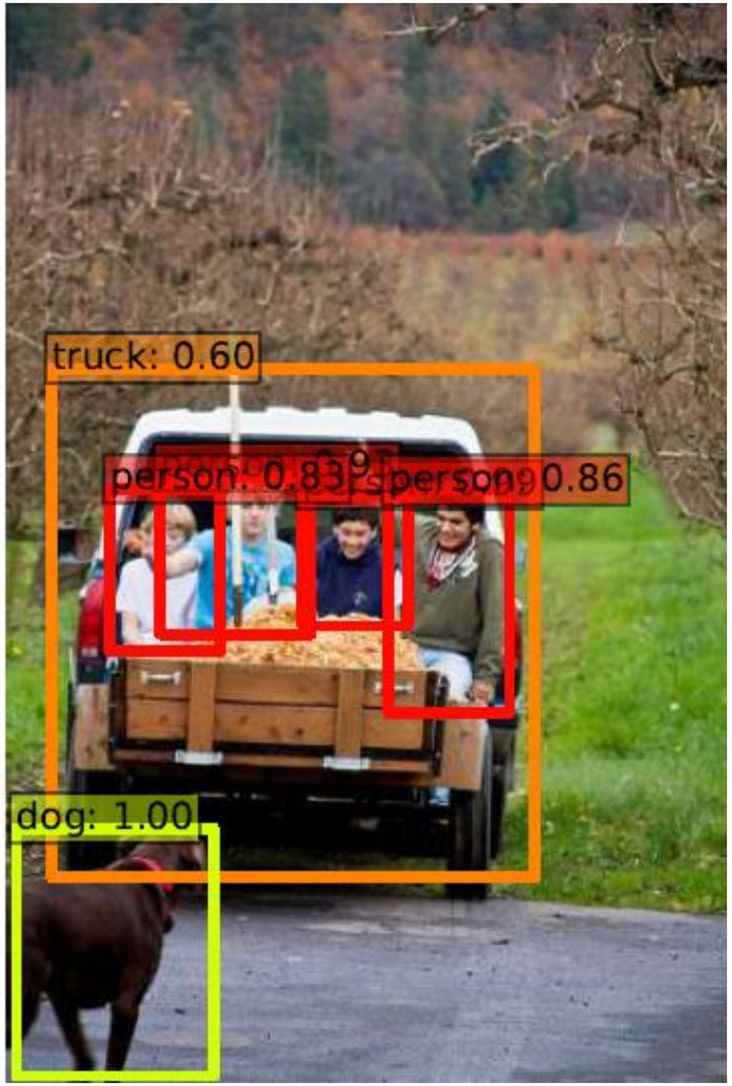
# Data Augmentation

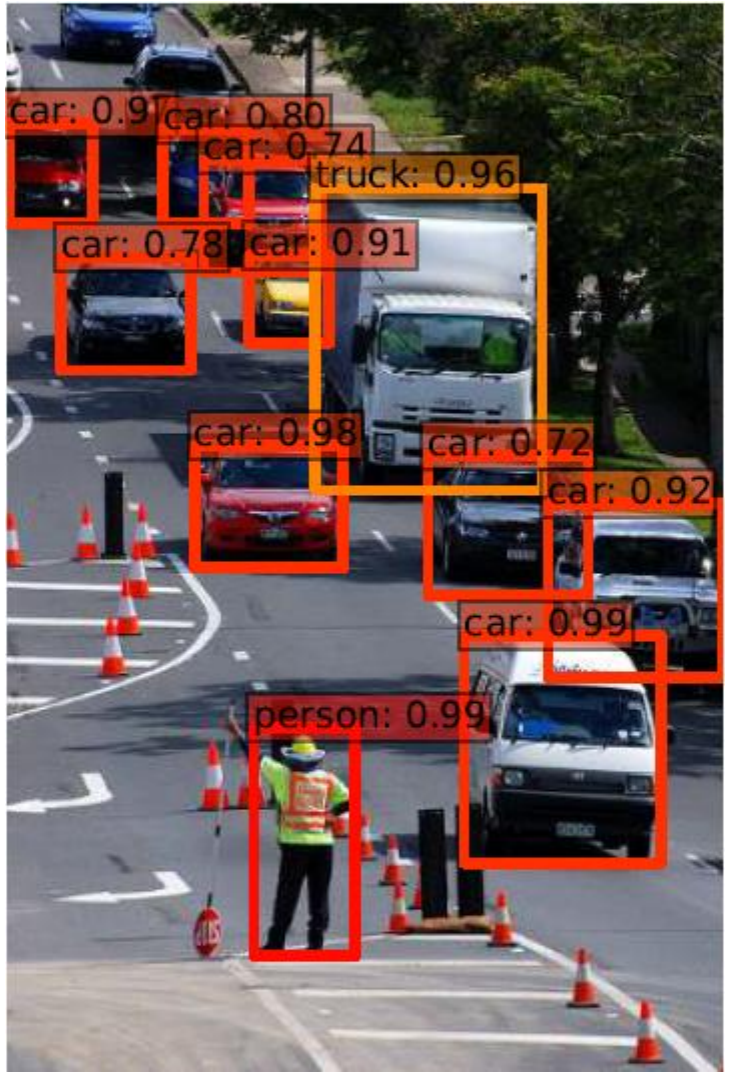
- Use the entire original input image.
- Sample a patch so that the minimum jaccard overlap with the objects is 0.1, 0.3, 0.5, 0.7 or 0.9.
- Randomly sample a patch.
- The size of each sampled patch is [0,1, 1] of the original image size
- Aspect ratio is between ½ and 2.
- Horizontally flipped with probability of 0.5

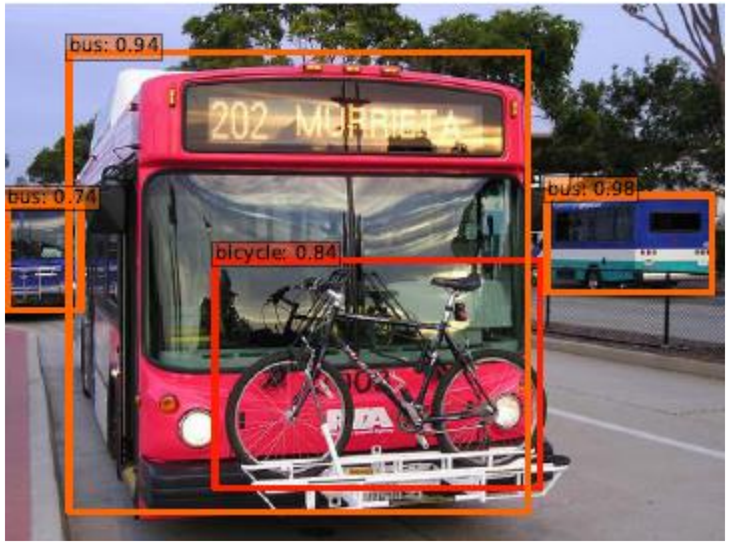
# Results - comparison with other methods

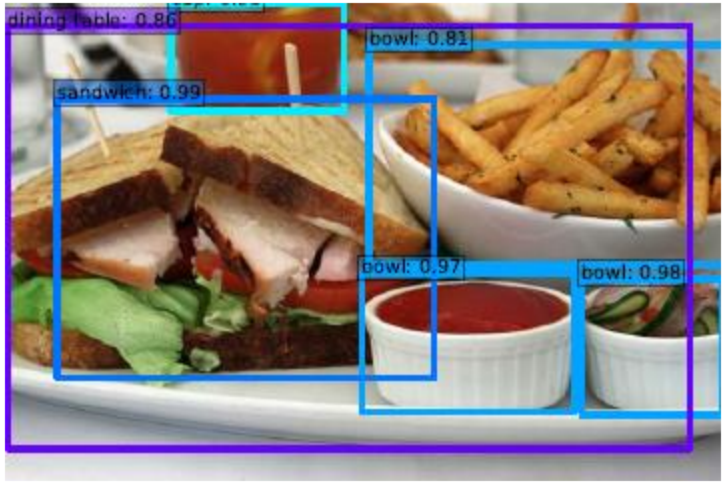
- SSD leads in detection speed; it is good at speed / accuracy tradeoff.

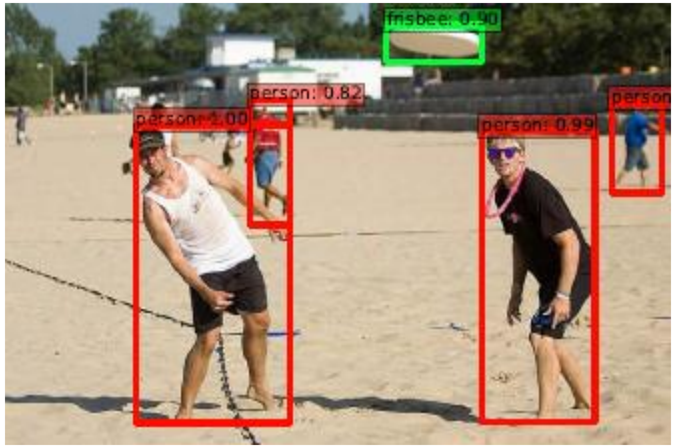



truck: 0.60
person: 0.83
person: 0.86
dog: 1.00


car: 0.80
car: 0.74
truck: 0.96
car: 0.78
car: 0.91
car: 0.98
car: 0.72
car: 0.92
car: 0.99
person: 0.99


bus: 0.94
202 MURRIETA
bus: 0.74
bus: 0.98
bicycle: 0.84


dining table: 0.86
bowl: 0.81
sandwich: 0.99
bowl: 0.97
bowl: 0.98


frisbee: 0.90
person: 0.82
person: 1.00
person: 0.99

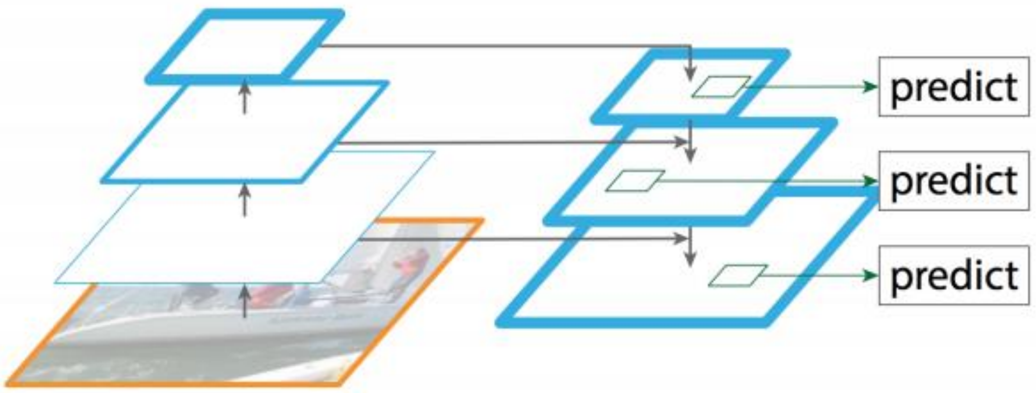

person: 1.00
backpack: 0.82
motorcycle: 0.99

# Conclusion

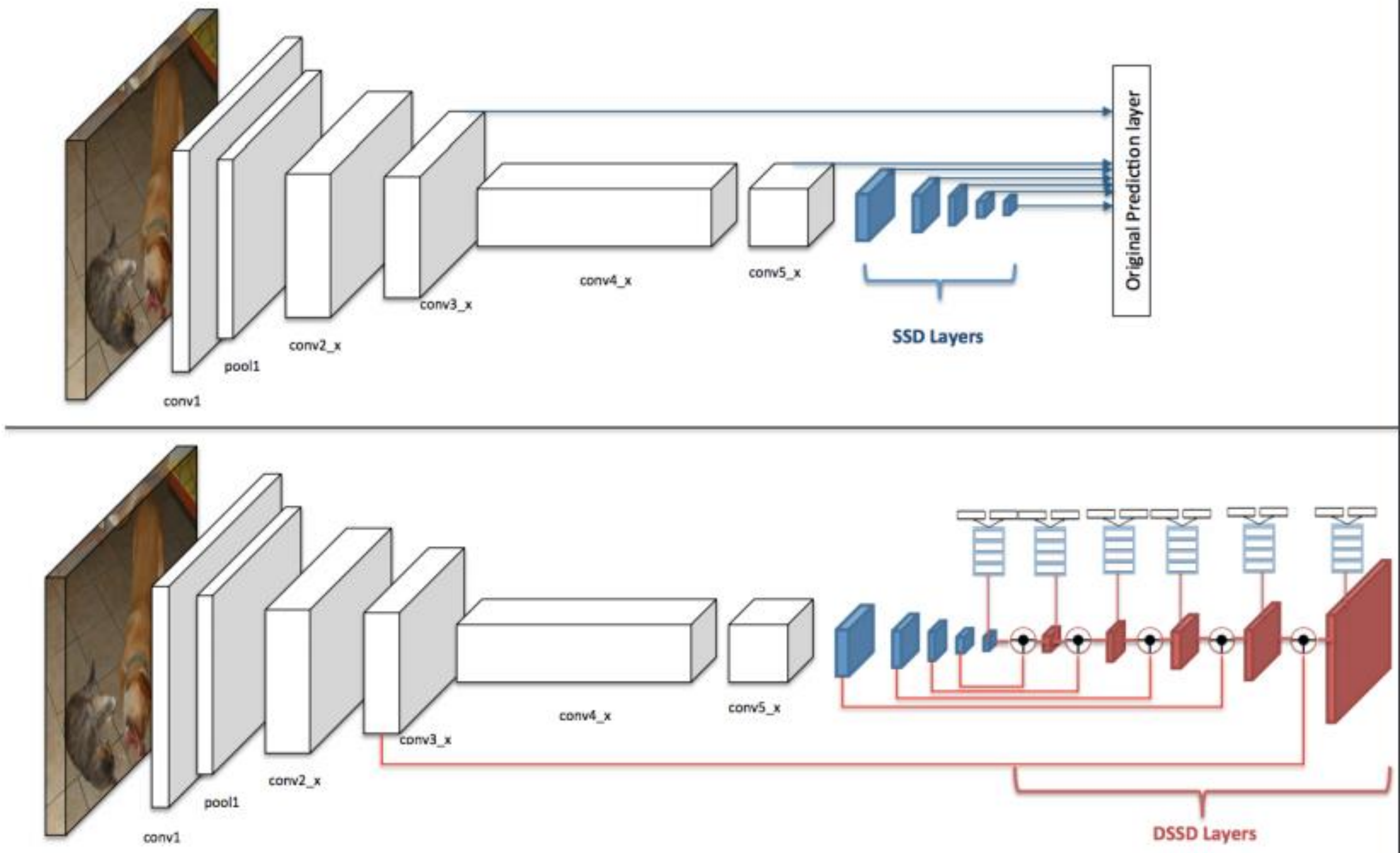
- Introduces a single-stage detector for object detection.
- Uses convolutional predictor on multiple feature maps, each responsible for a unique scale of objects.
- Achieves competitive accuracy and faster speed on various datasets.

# Extensions?

- Use skip connections and deconvolution to integrate low-level location information and high-level semantic information.




*Feature Pyramid Networks for Object Detection (CVPR 2017)*




*DSSD : Deconvolutional Single Shot Detector (2017)*