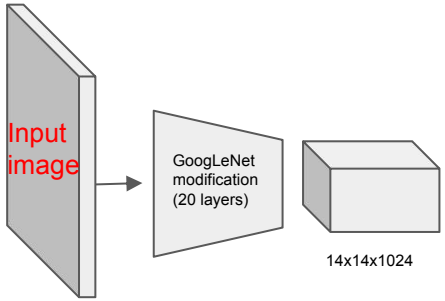
YOLO

You Only Look Once: Unified, Real-Time Object Detection
Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi
YOLO9000: Better, Faster, Stronger
Joseph Redmon, Ali Farhadi
Reporter: Jian.Yin

Inference

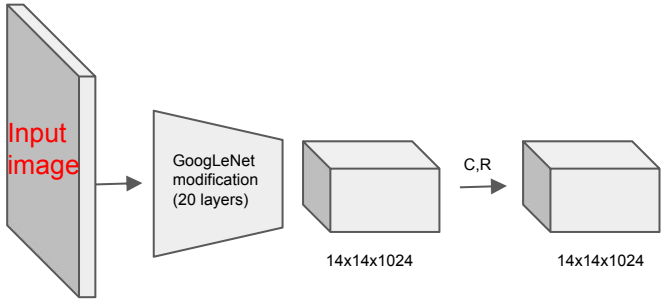
448x448x3

Inference



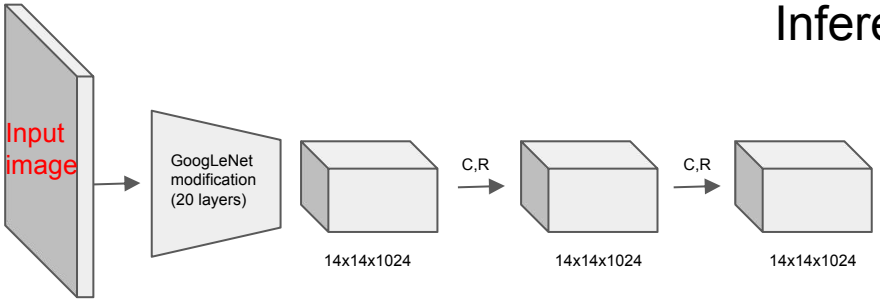
448x448x3

Inference



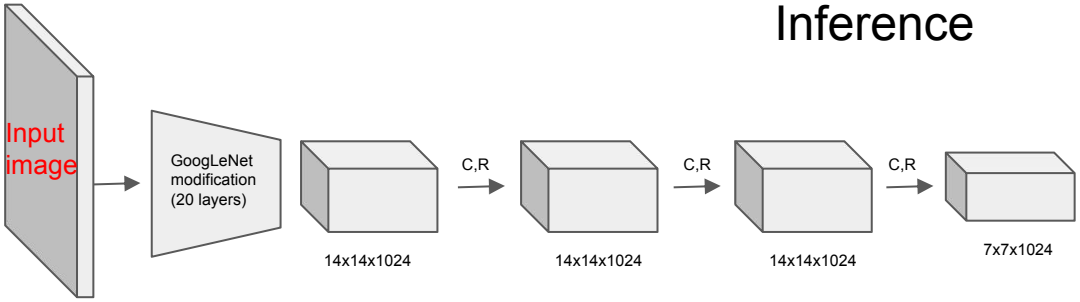
448x448x3

Inference



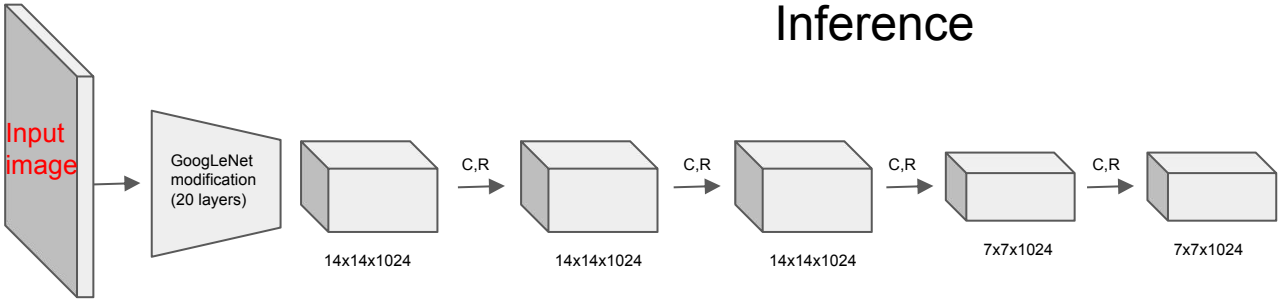
448x448x3

Inference



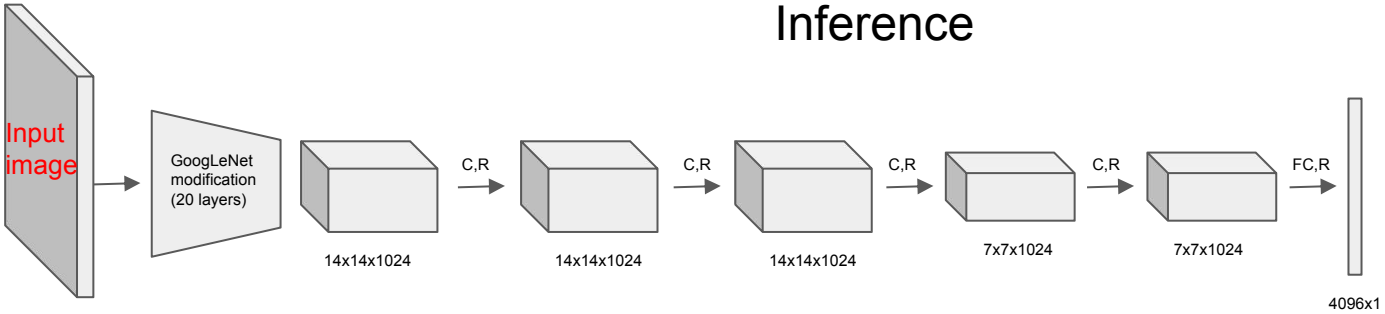
448x448x3

Inference



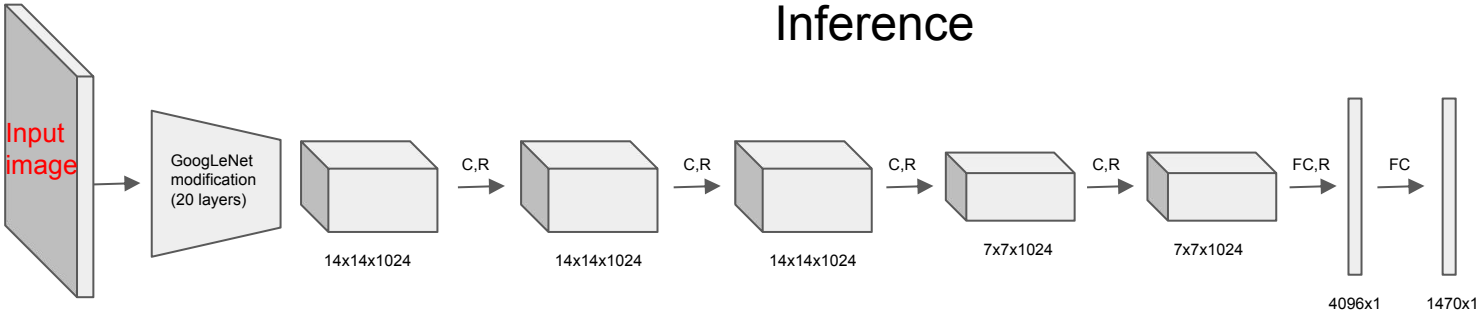
448x448x3

Inference



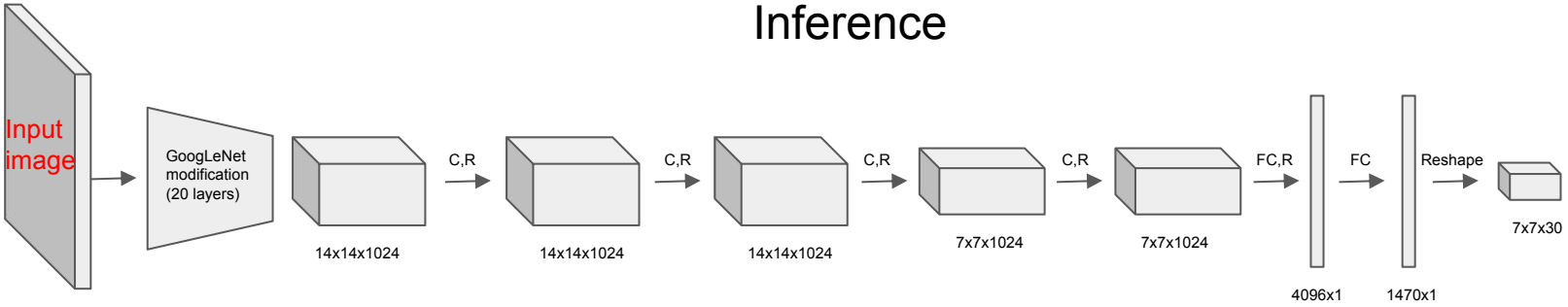
448x448x3

Inference

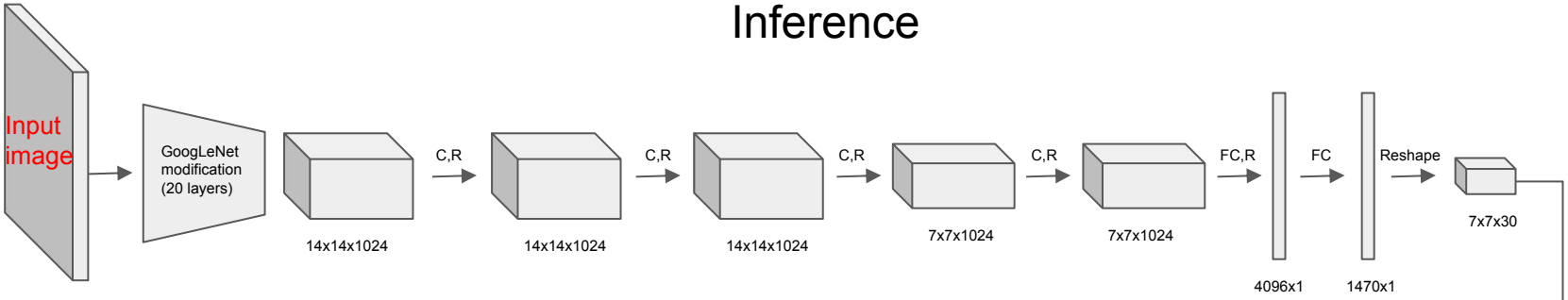


448x448x3

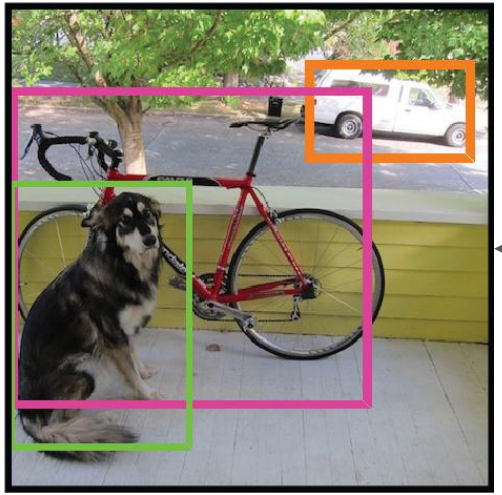
Inference


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

Inference

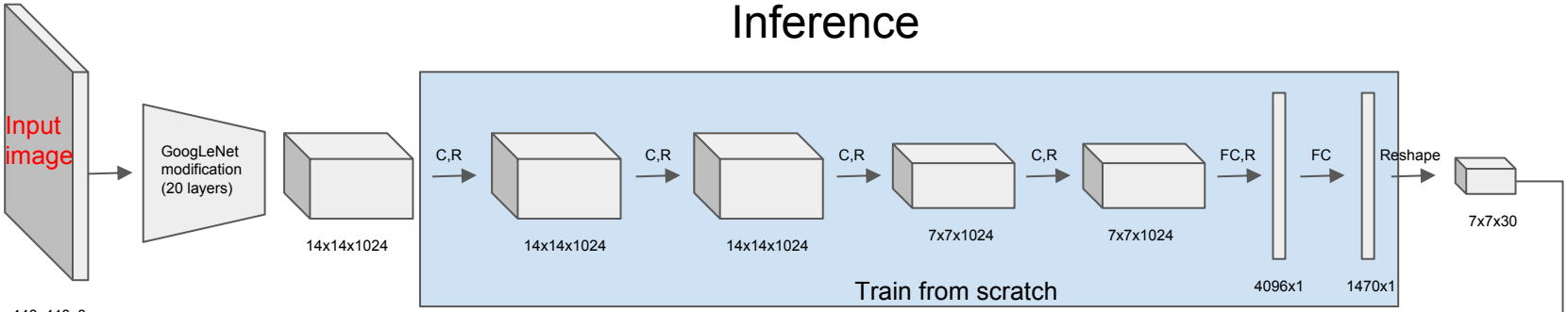


448x448x3

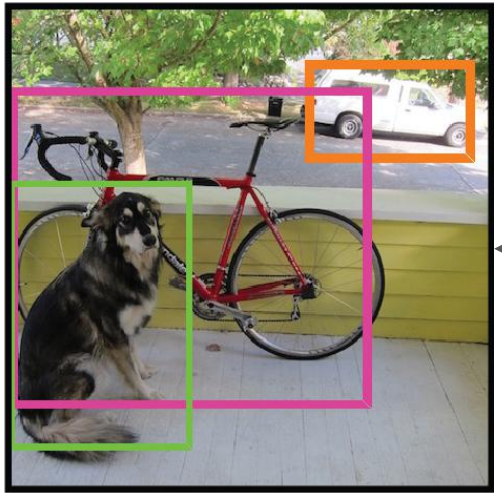


Detection Procedure

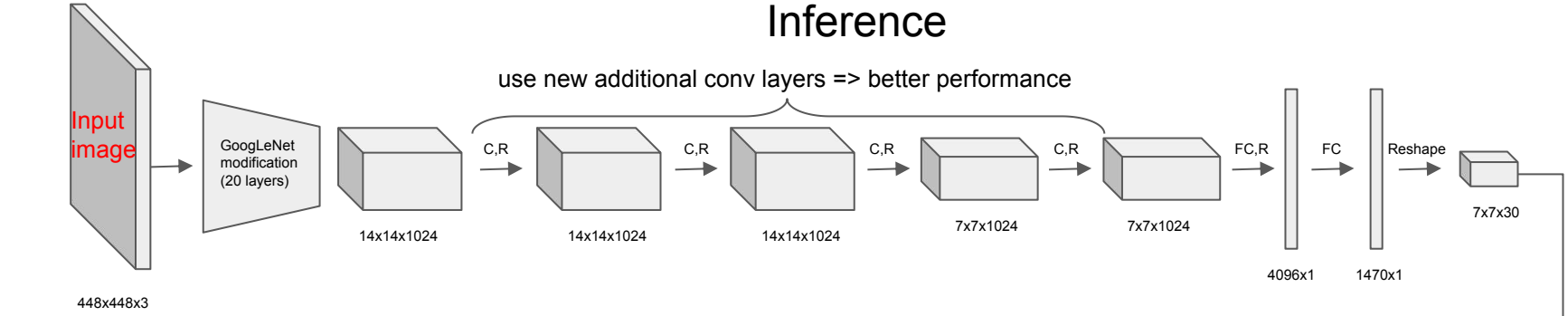
Inference

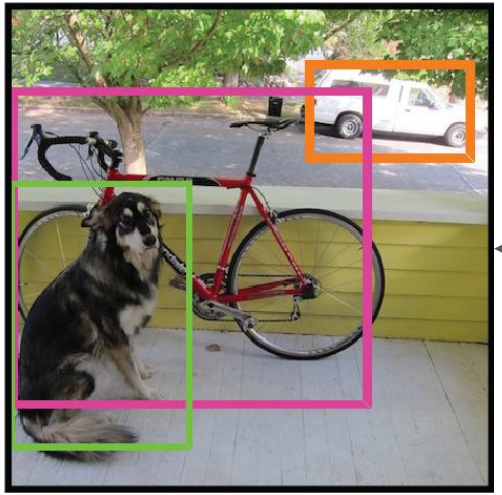


Detection Procedure



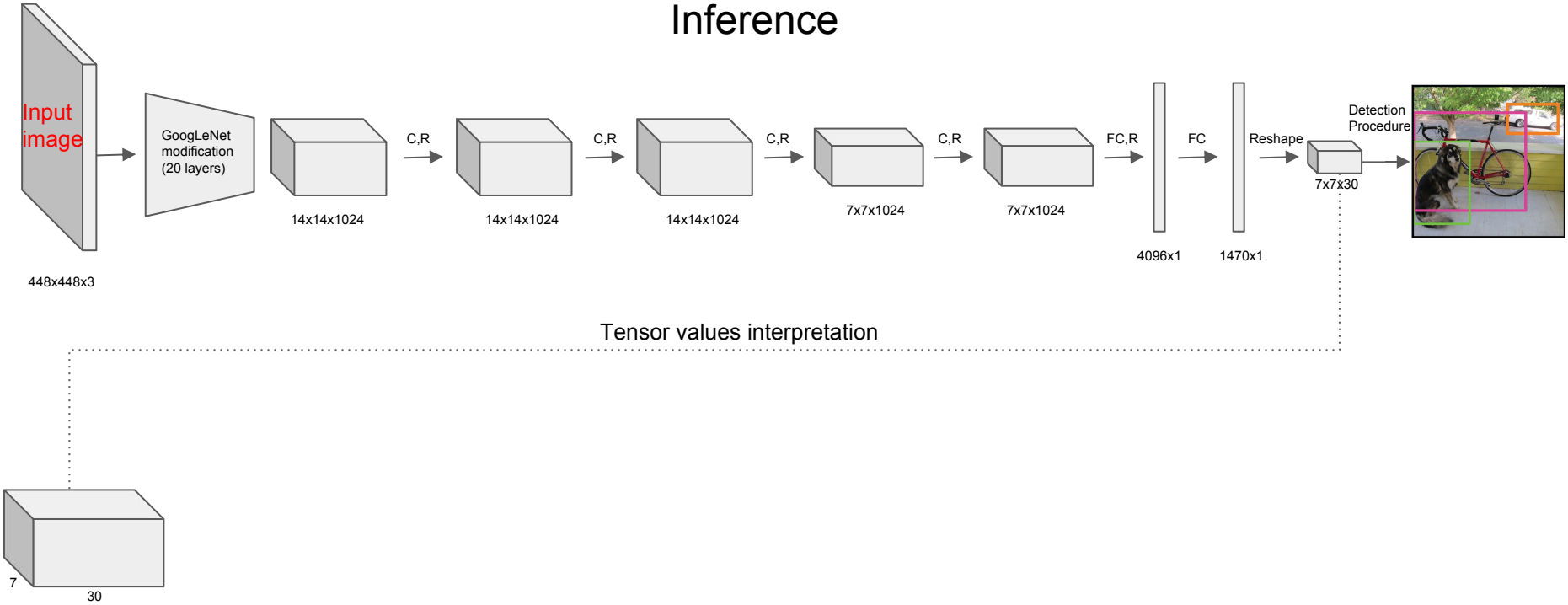
Inference



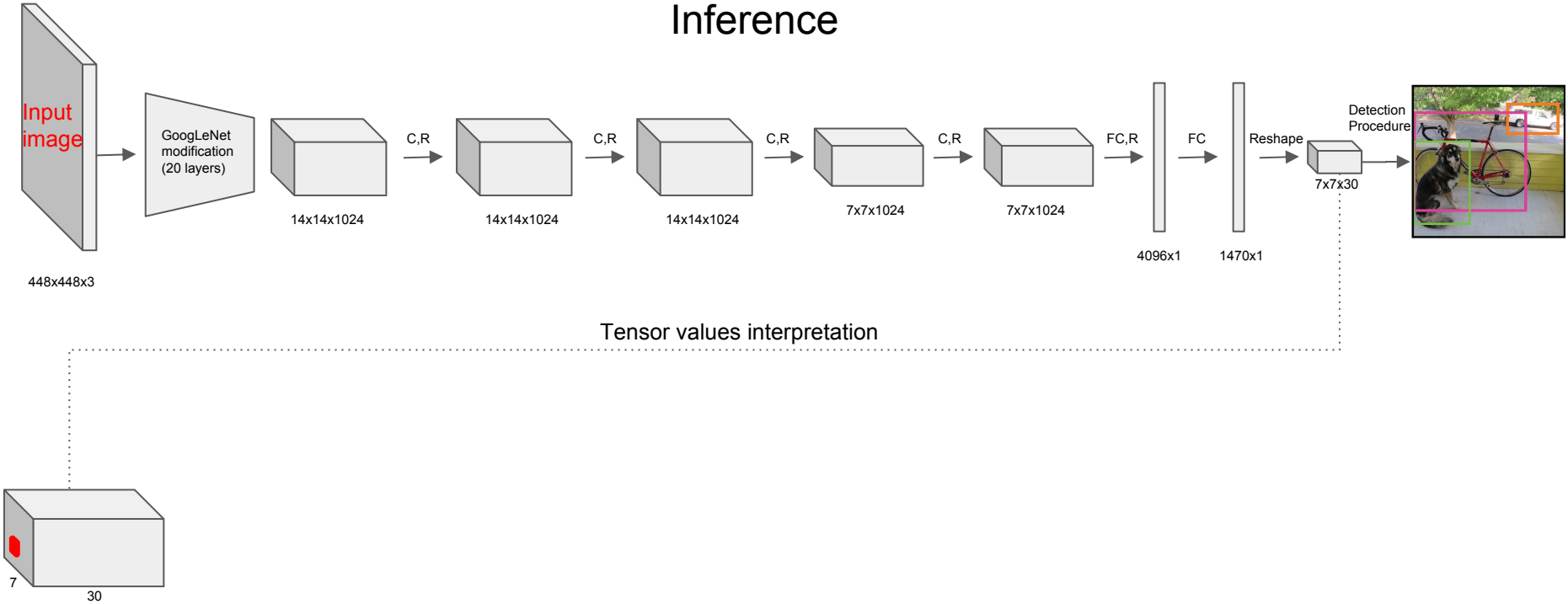


Detection Procedure

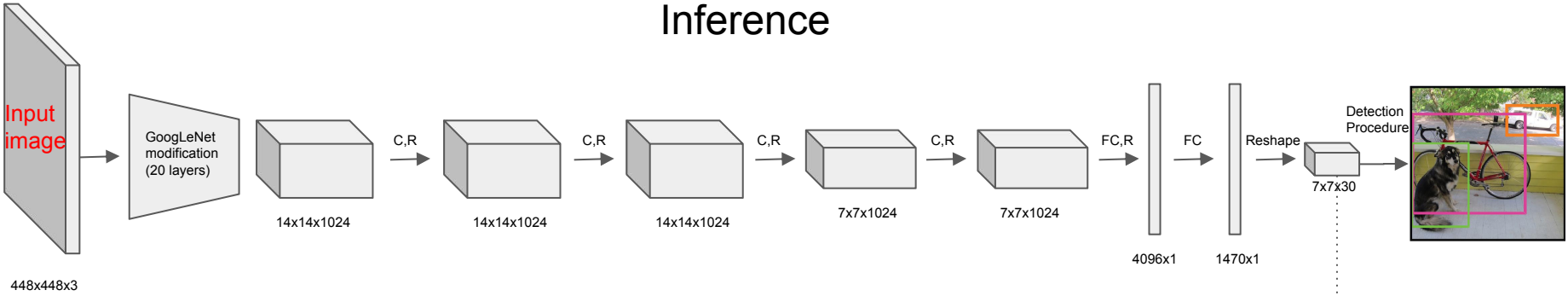
Inference


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

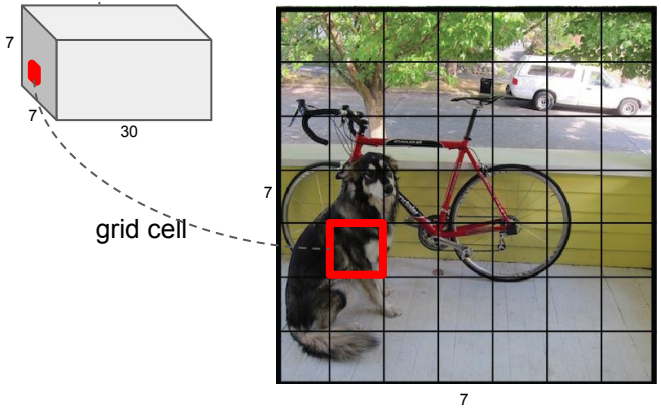
Inference


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

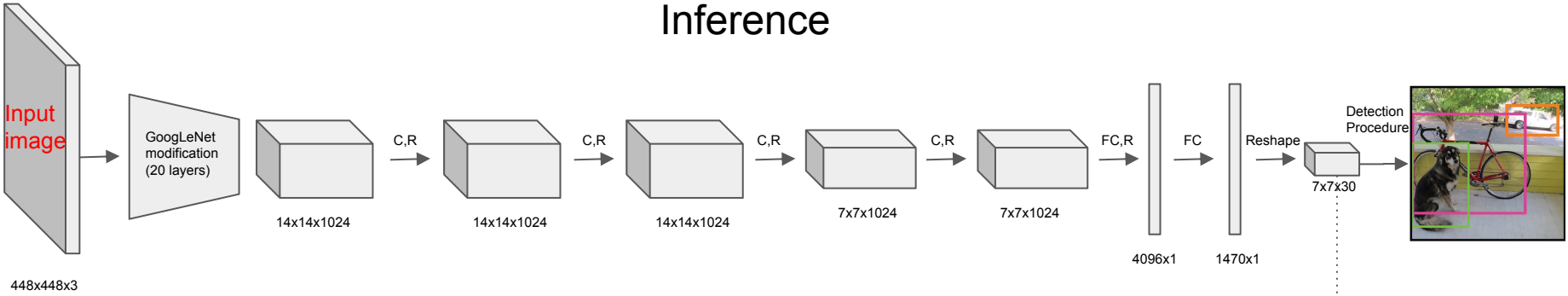
Inference



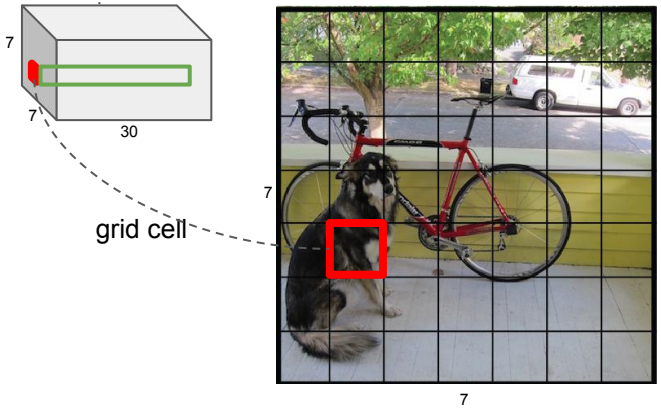
Tensor values interpretation



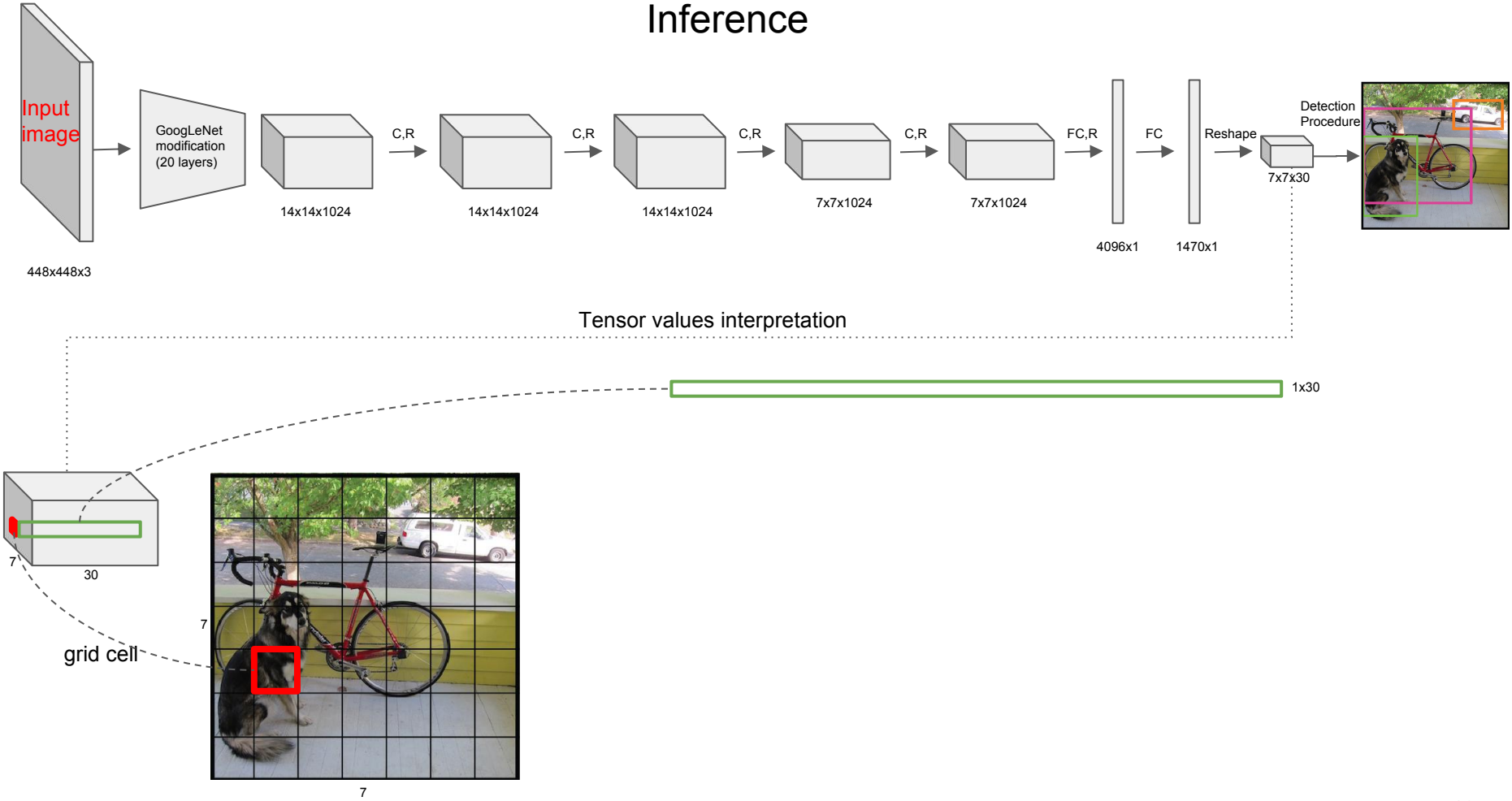
Inference



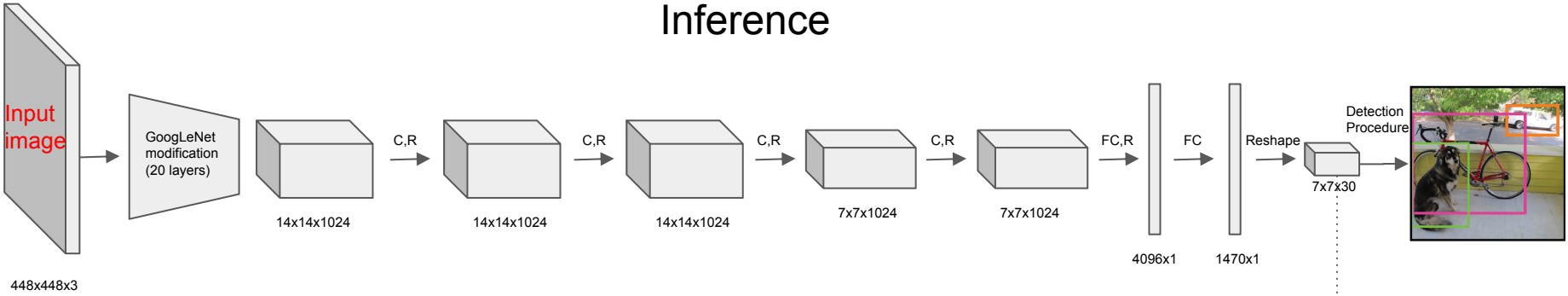
Tensor values interpretation



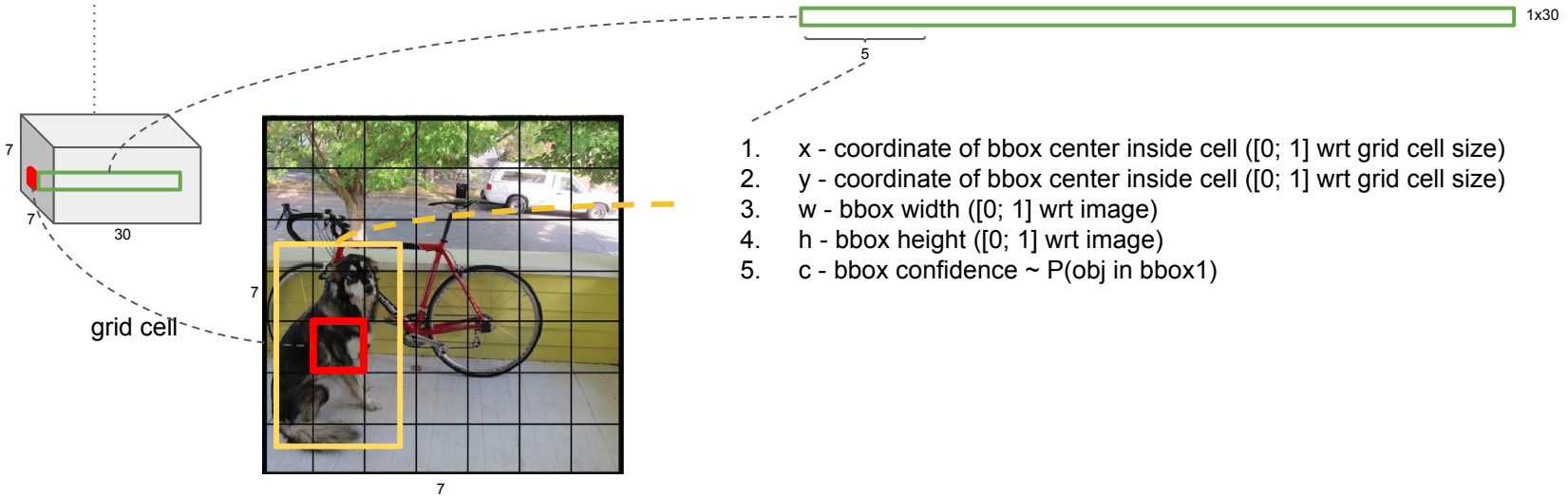
Inference


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

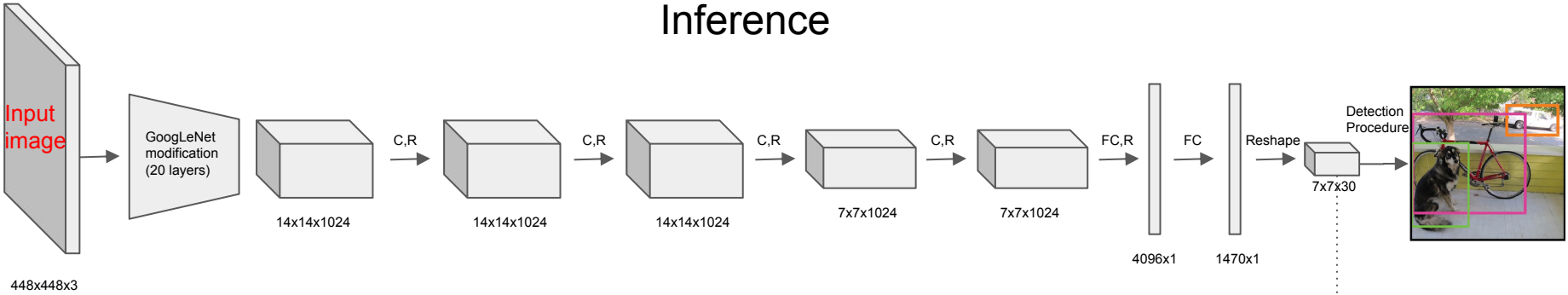


Tensor values interpretation

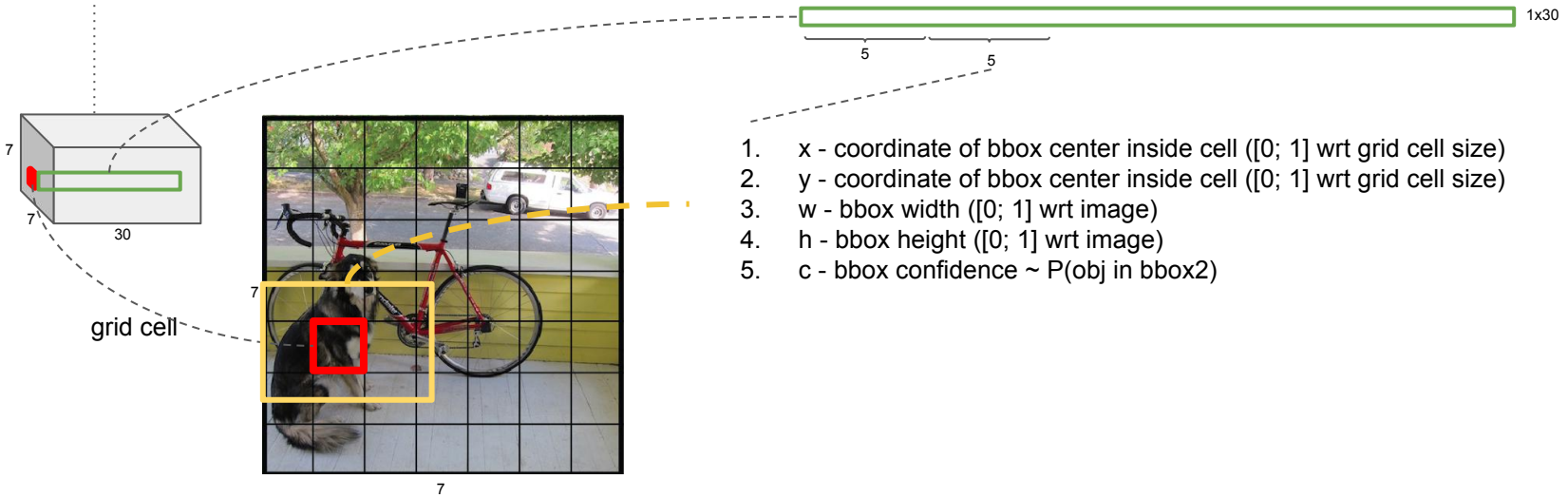


1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox1)

Inference

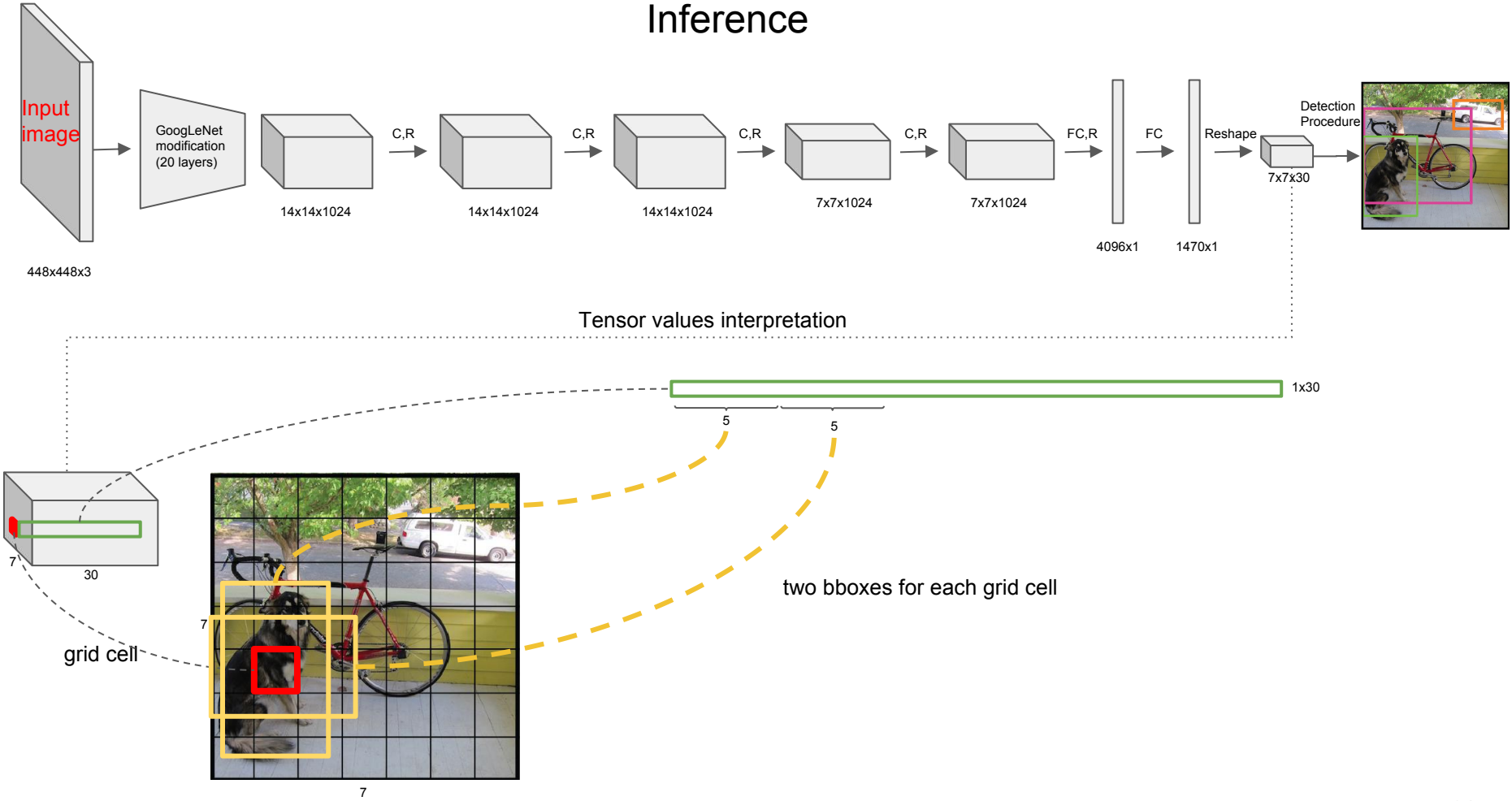


Tensor values interpretation

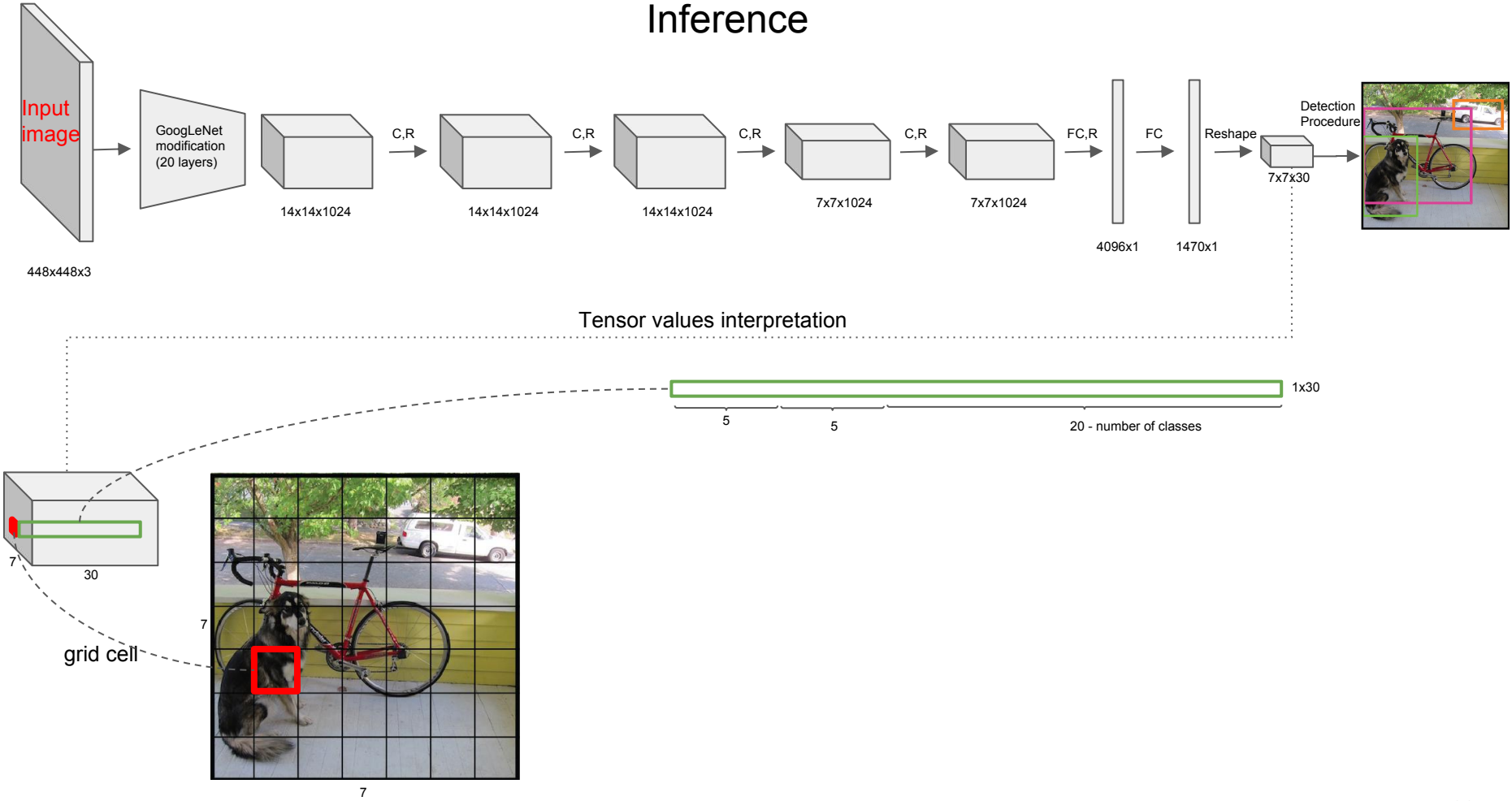


1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox2)

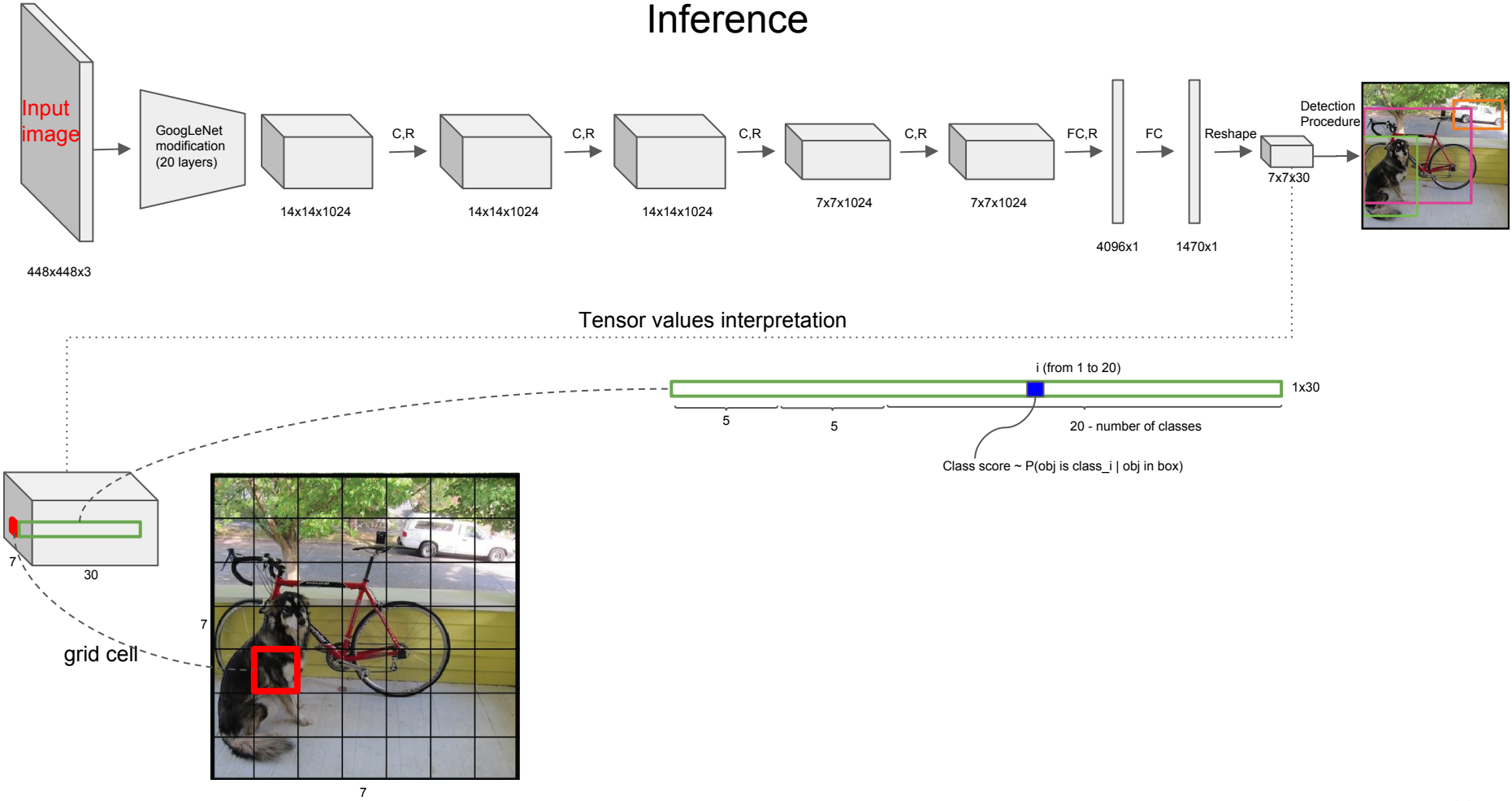
Inference


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

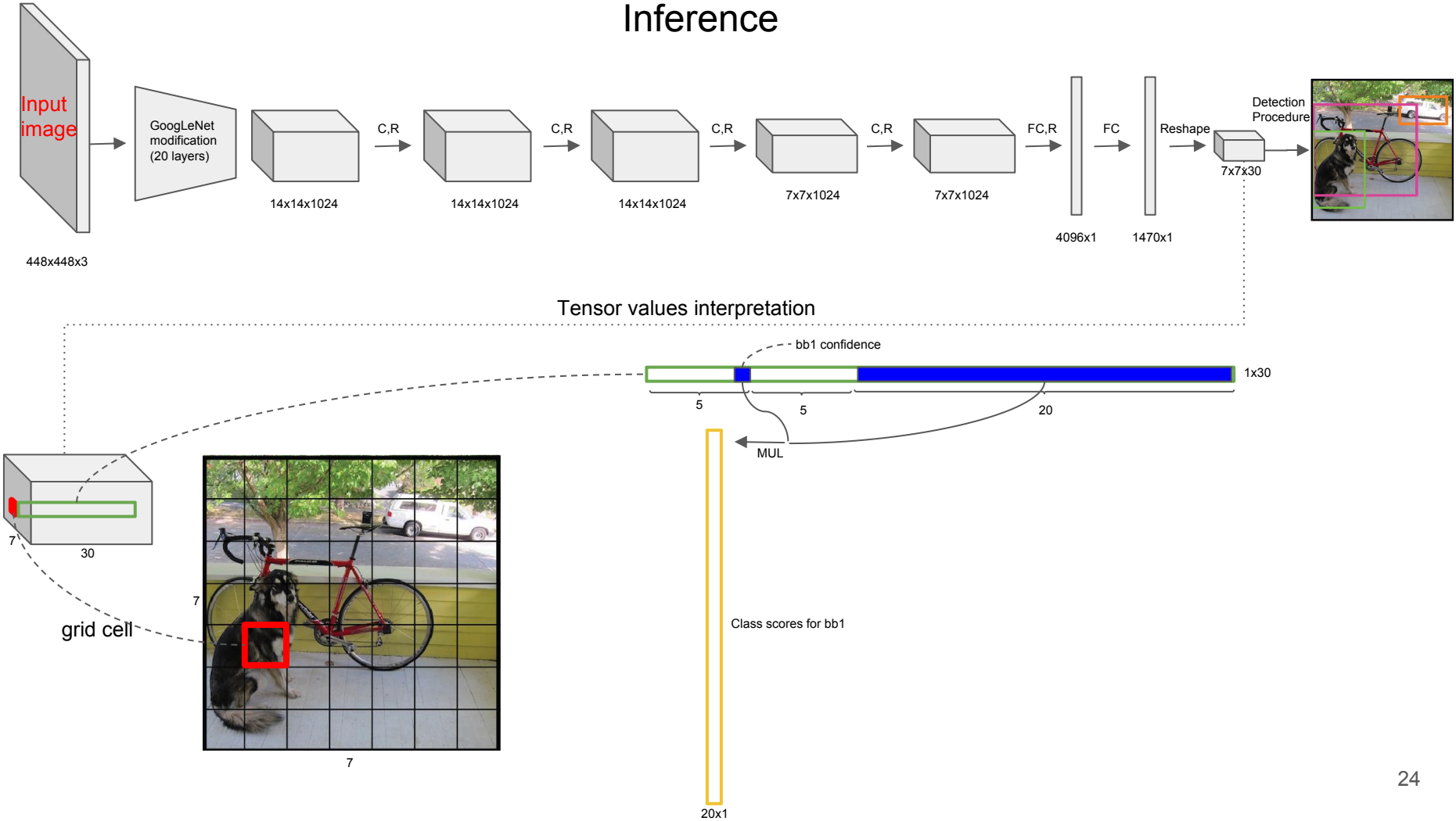
Inference


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

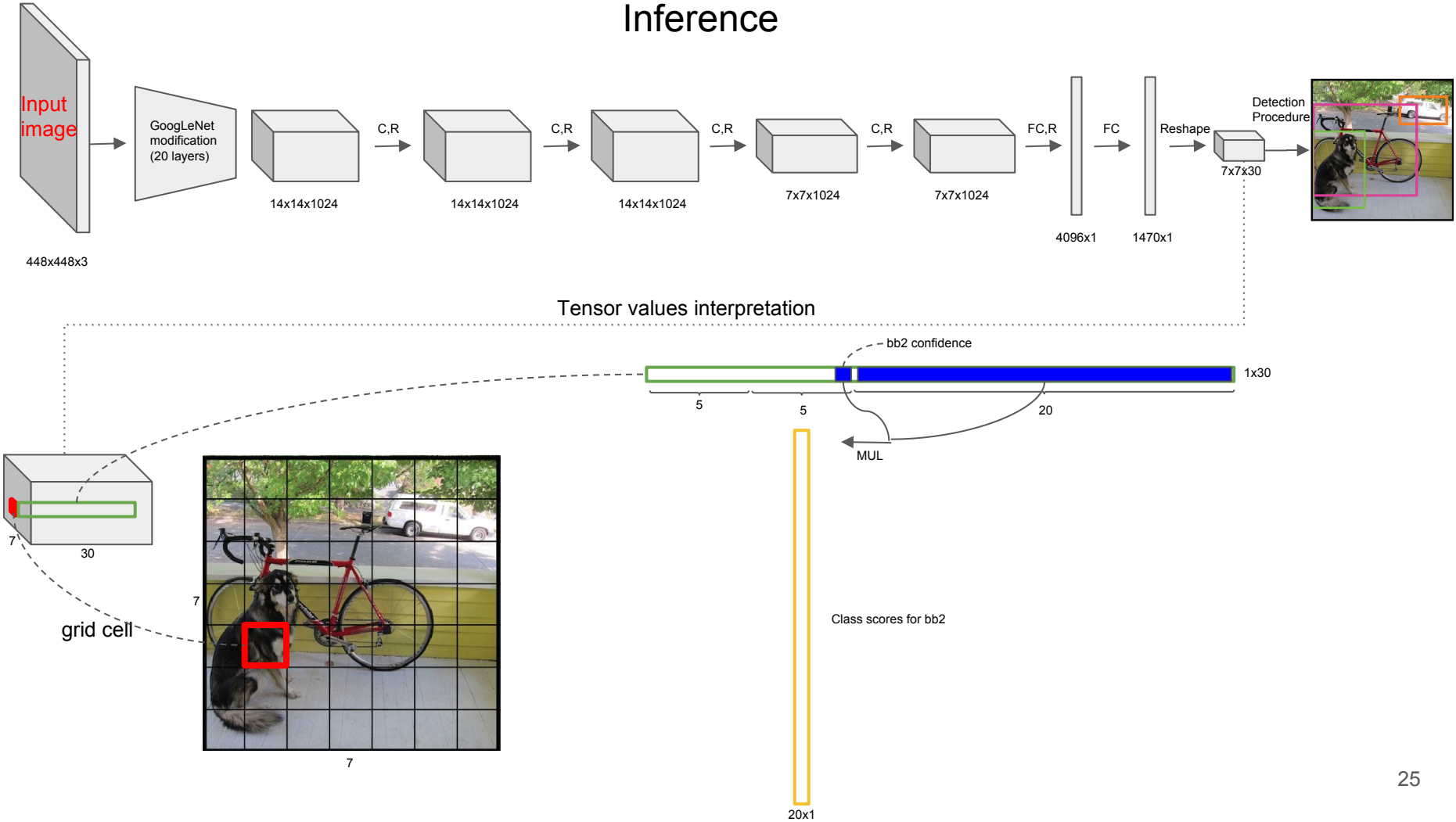
Inference


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

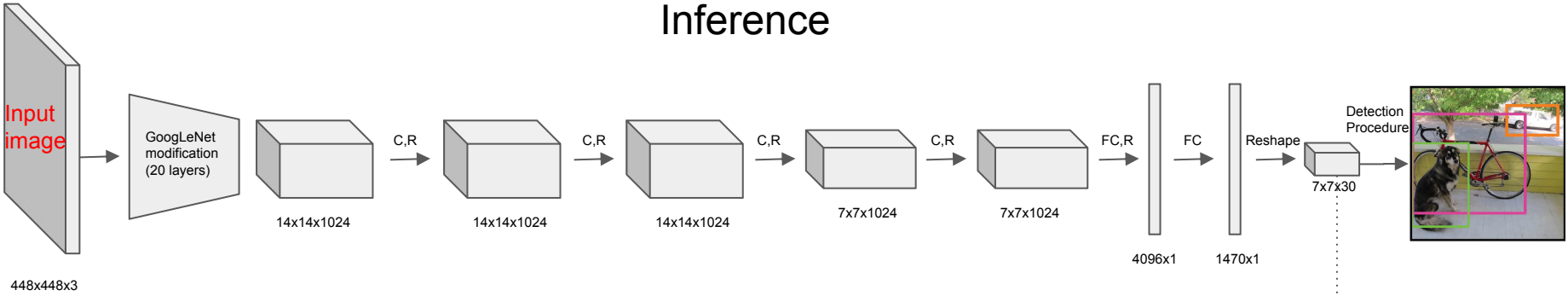
Inference


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
7
7
30
grid cell
7
7
Class scores for bb1
20x1

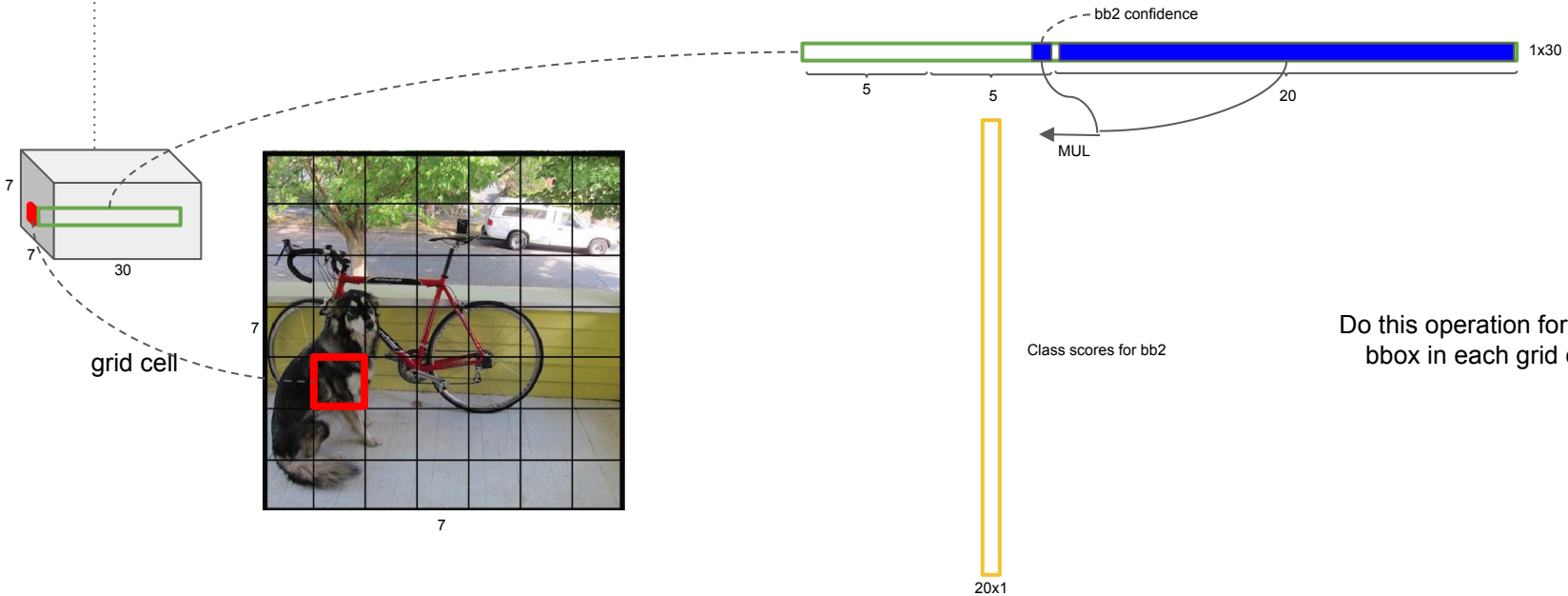
Inference


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
bb2 confidence
1x30
5
5
20
MUL
7
7
30
grid cell
7
7
Class scores for bb2
20x1

Inference

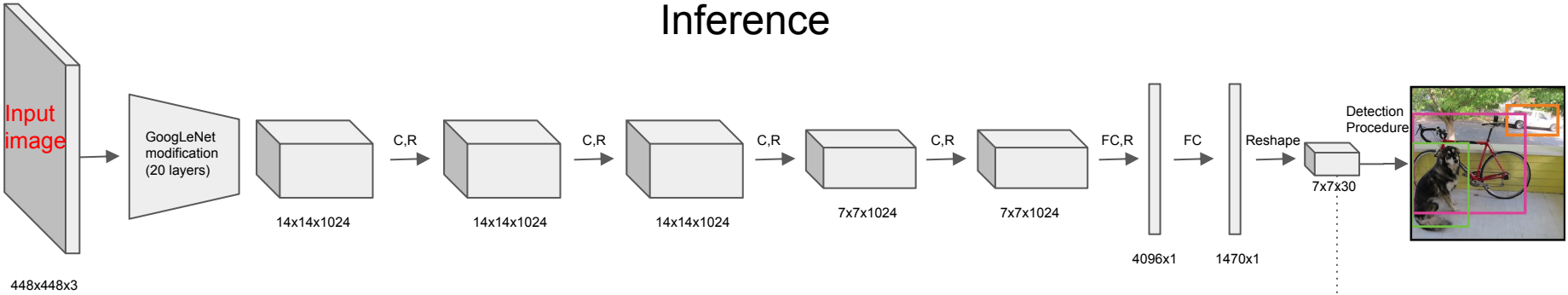


Tensor values interpretation

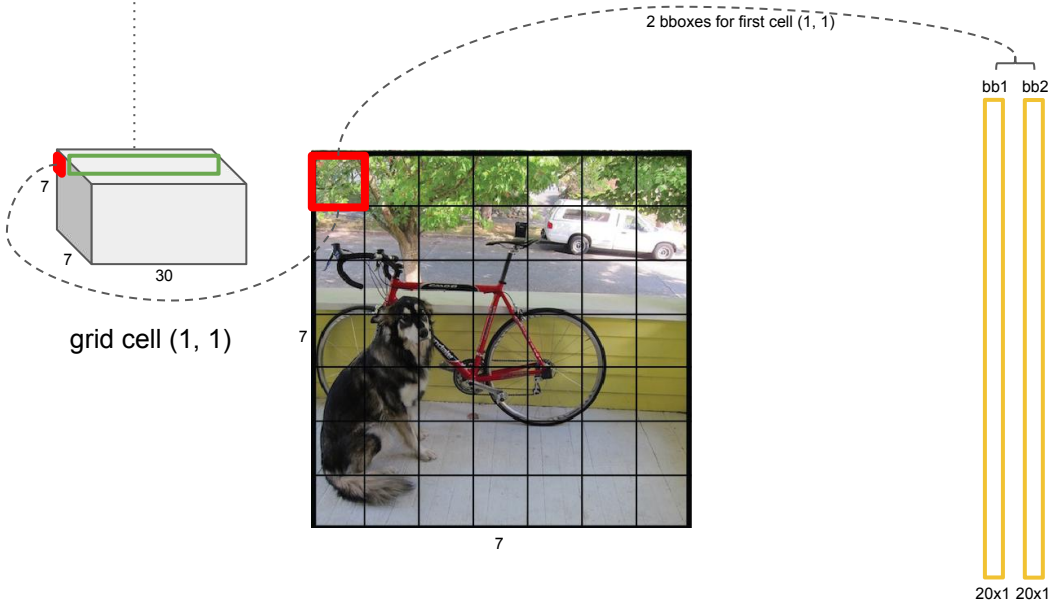


Do this operation for each bbox in each grid cell

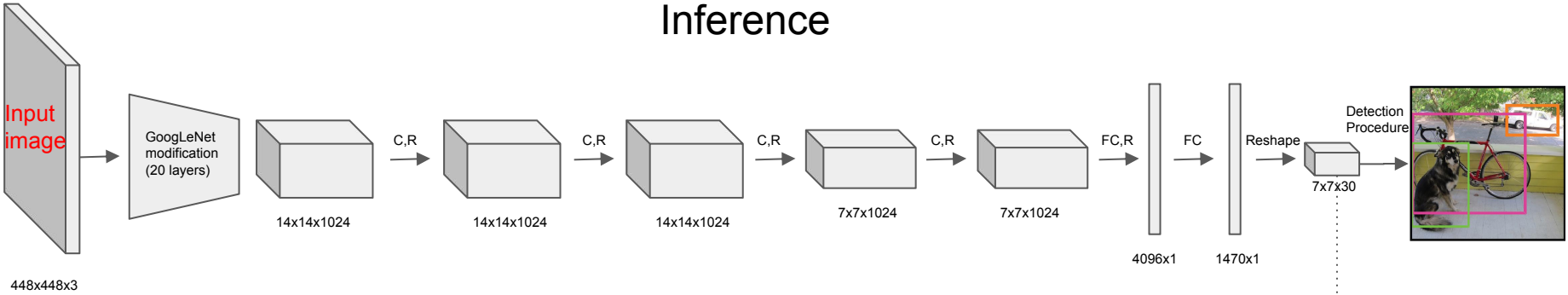
Inference



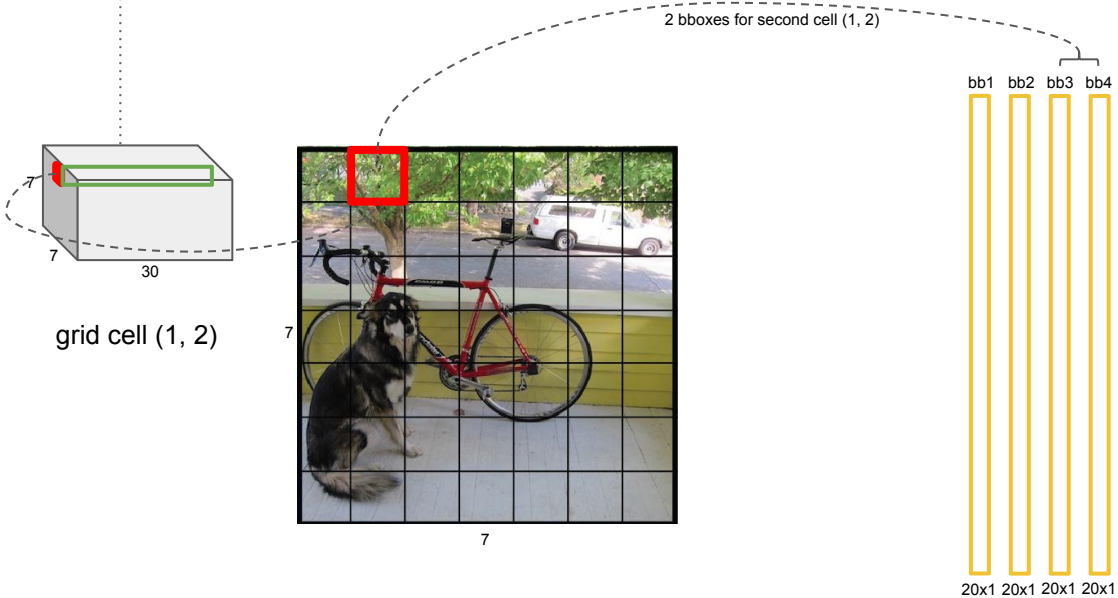
Tensor values interpretation



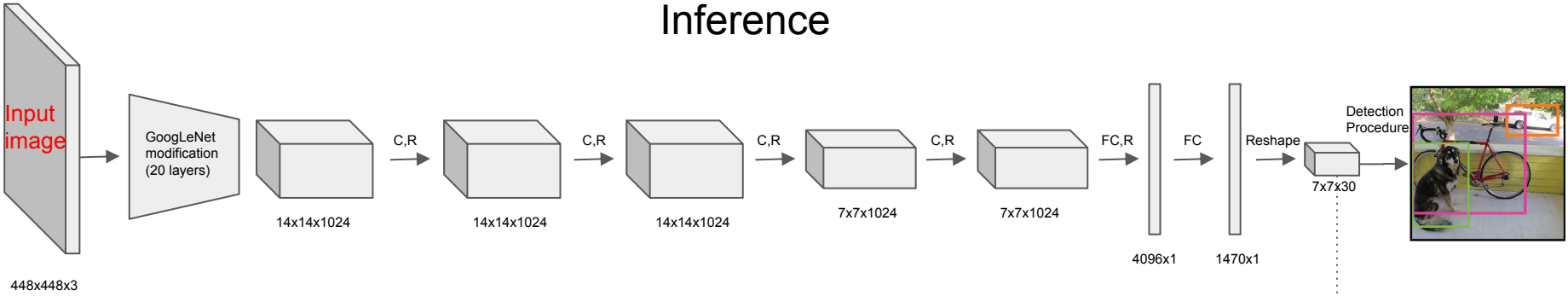
Inference



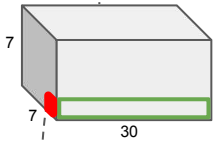
Tensor values interpretation



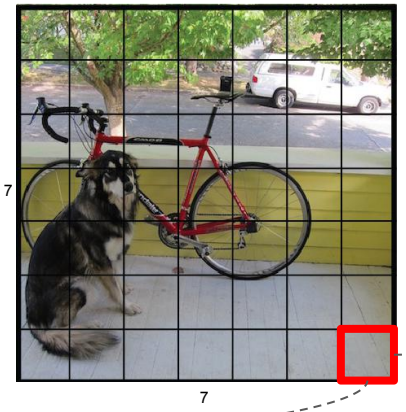
Inference

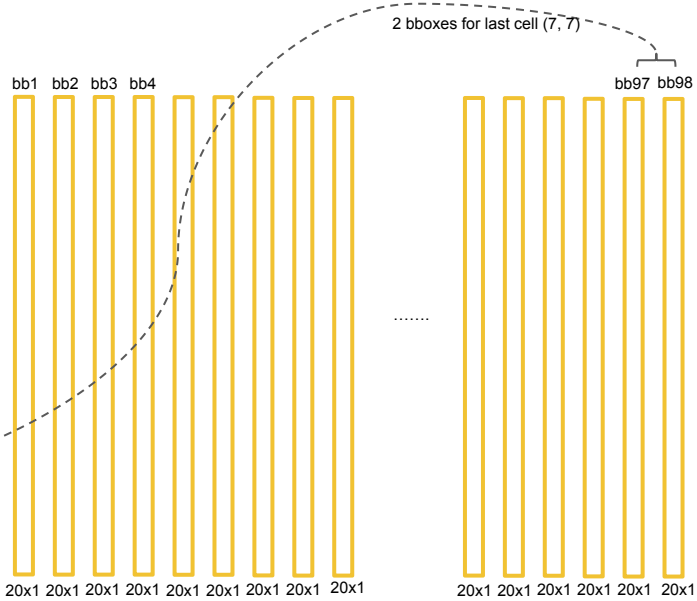


Tensor values interpretation

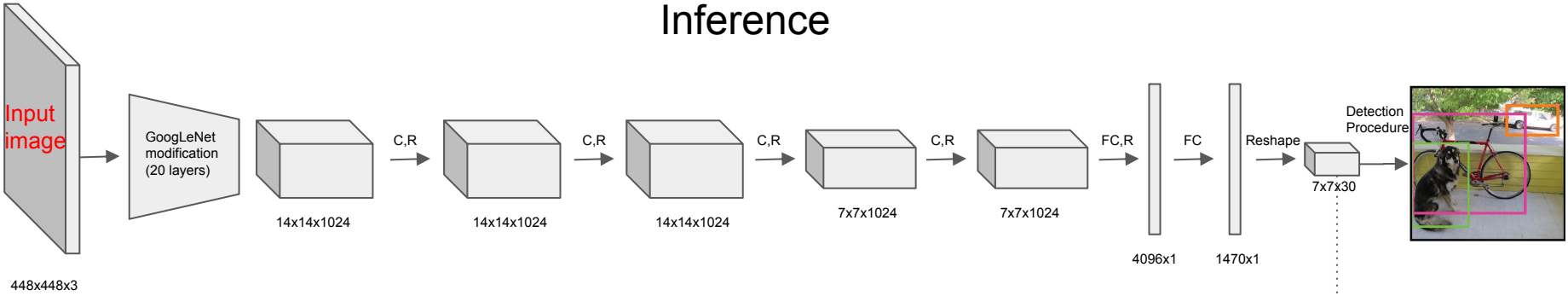


grid cell (7, 7)

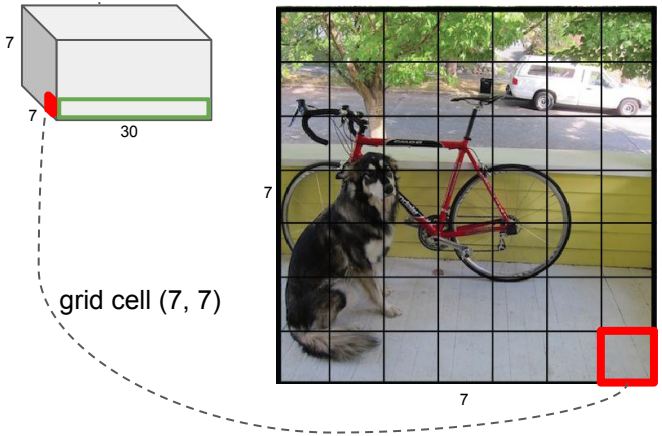




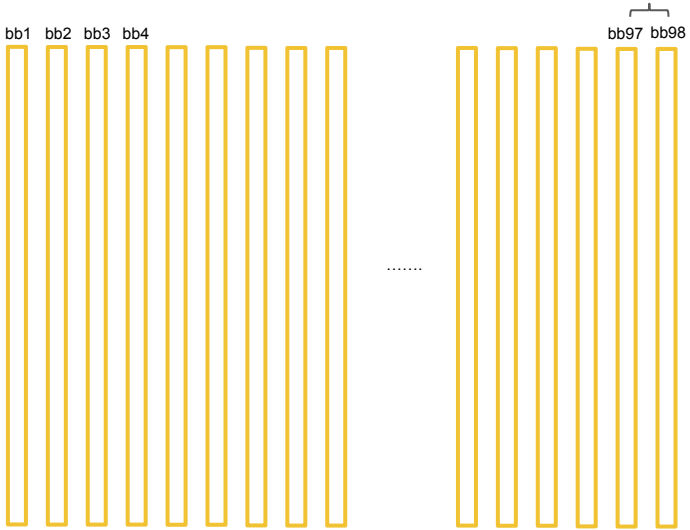
Inference



Tensor values interpretation



Total 7*7*2 = 98 bboxes



Loss function

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

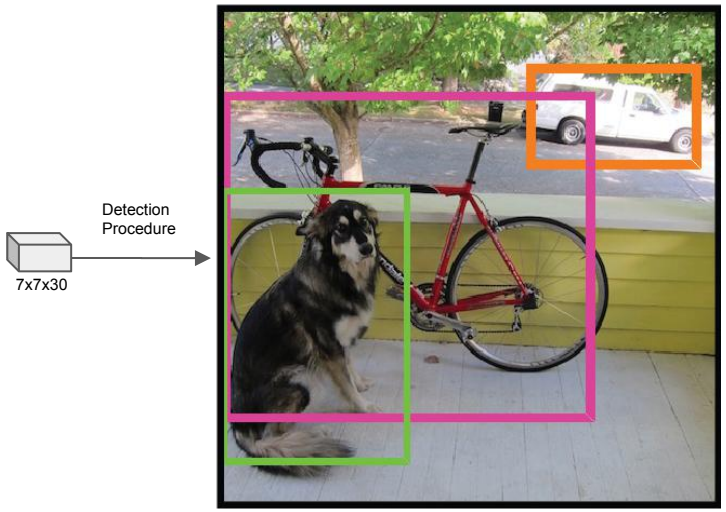
$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

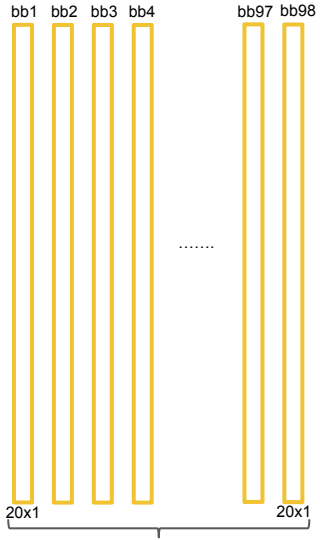
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

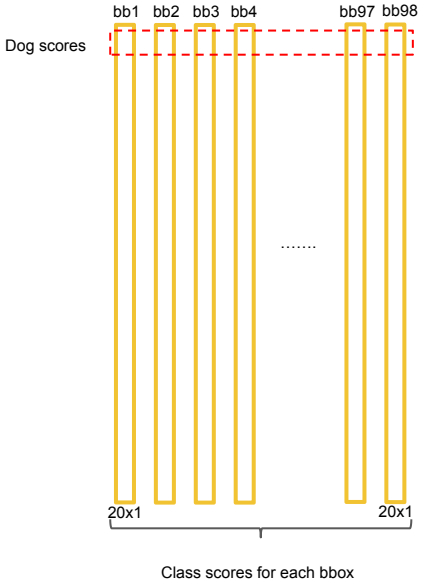
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

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

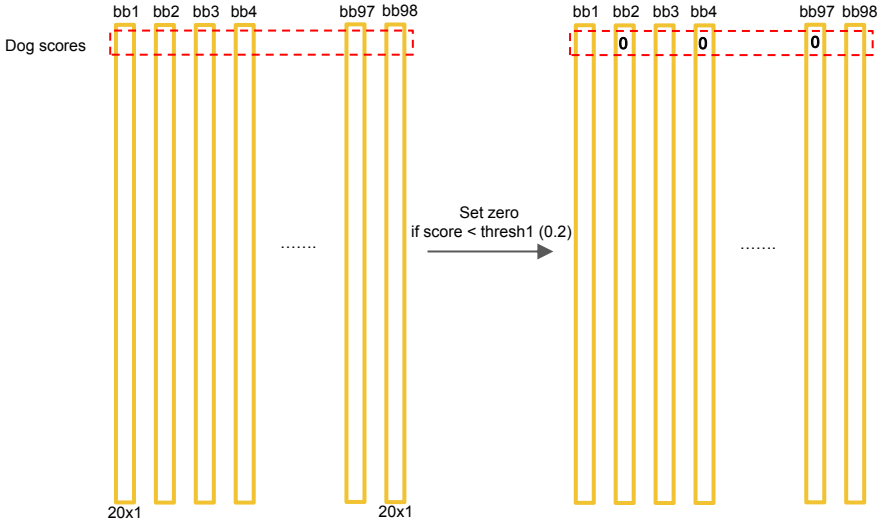
Look at detection procedure

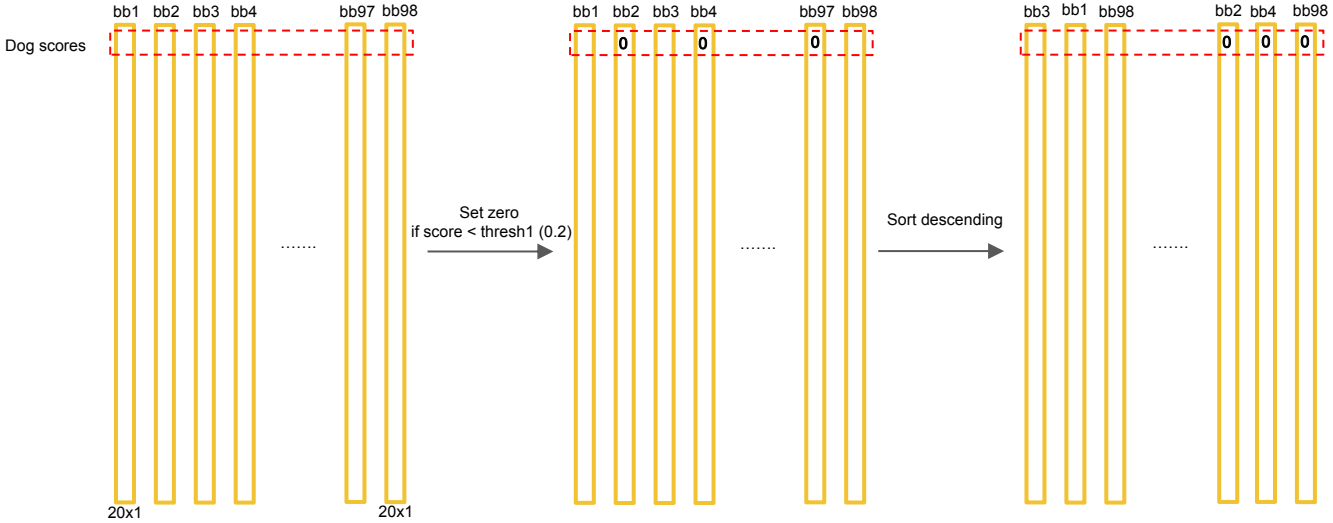


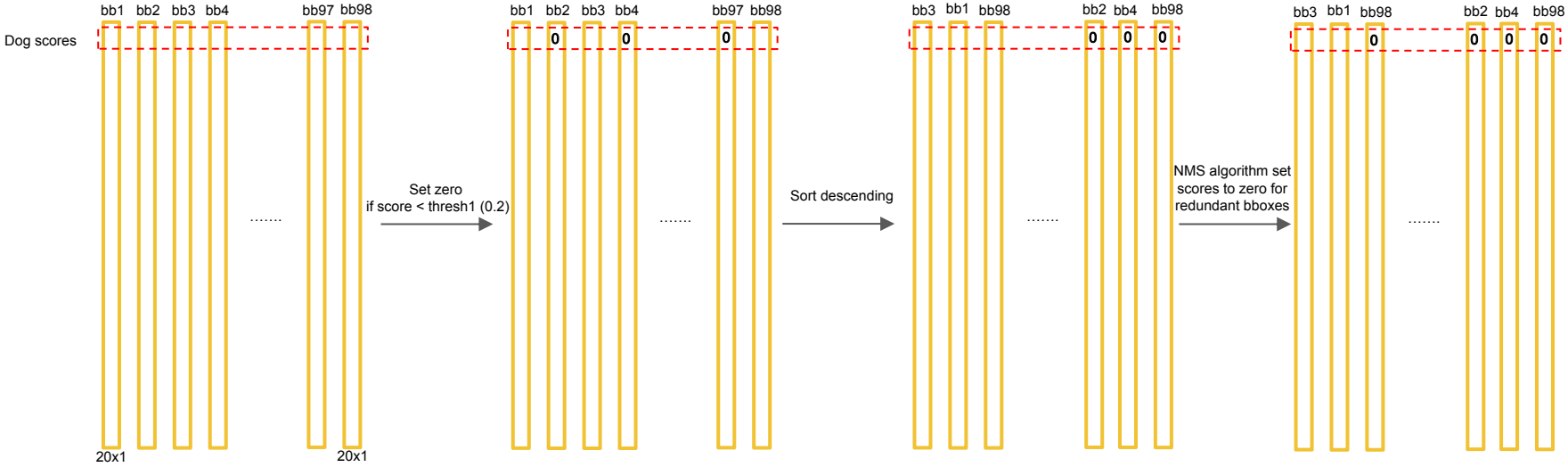

bb1
bb2
bb3
bb4
.....
bb97
bb98
20x1
20x1
Class scores for each bbox

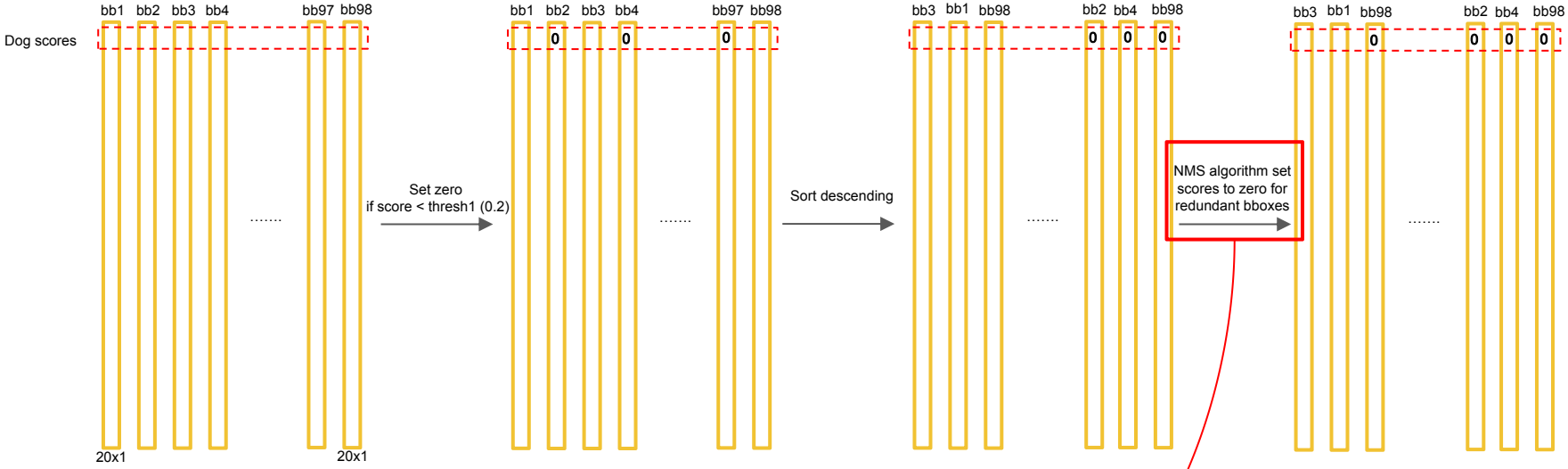


Get first class scores for each bbox


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
.....


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

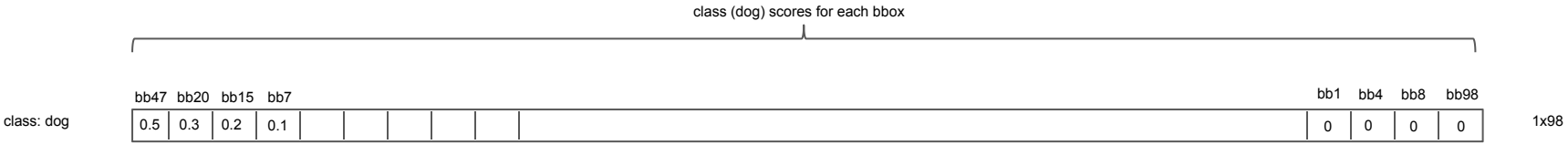

Dog scores
bb1 bb2 bb3 bb4 ..... bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 ..... bb97 bb98
0 0 0
Sort descending
bb3 bb1 ..... bb98 bb2 bb4 bb98
NMS algorithm set
scores to zero for
redundant bboxes
bb3 bb1 bb98 ..... bb2 bb4 bb98
0 0 0 0 0 0


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

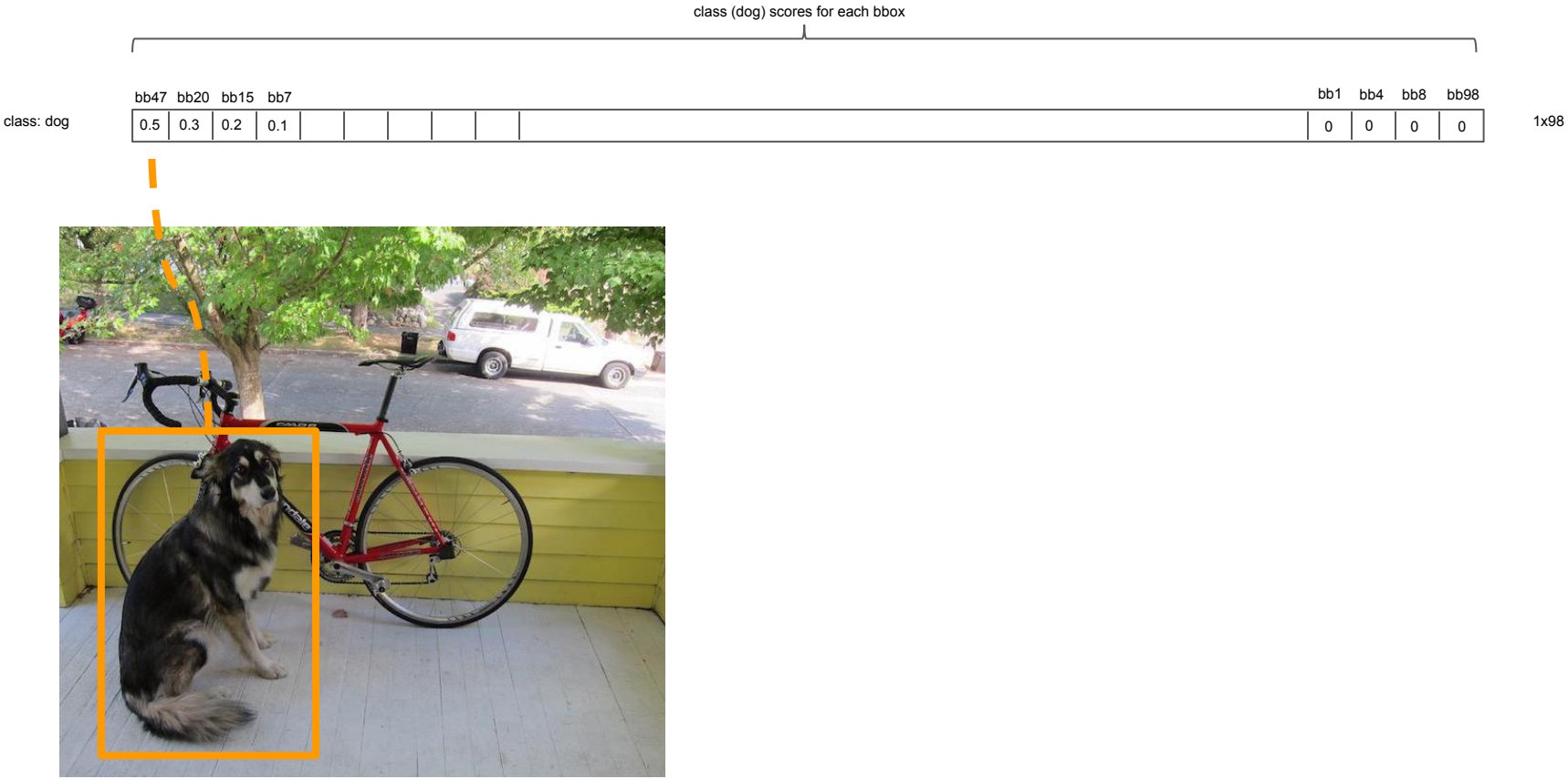
How it works

Non-Maximum Suppression: intuition

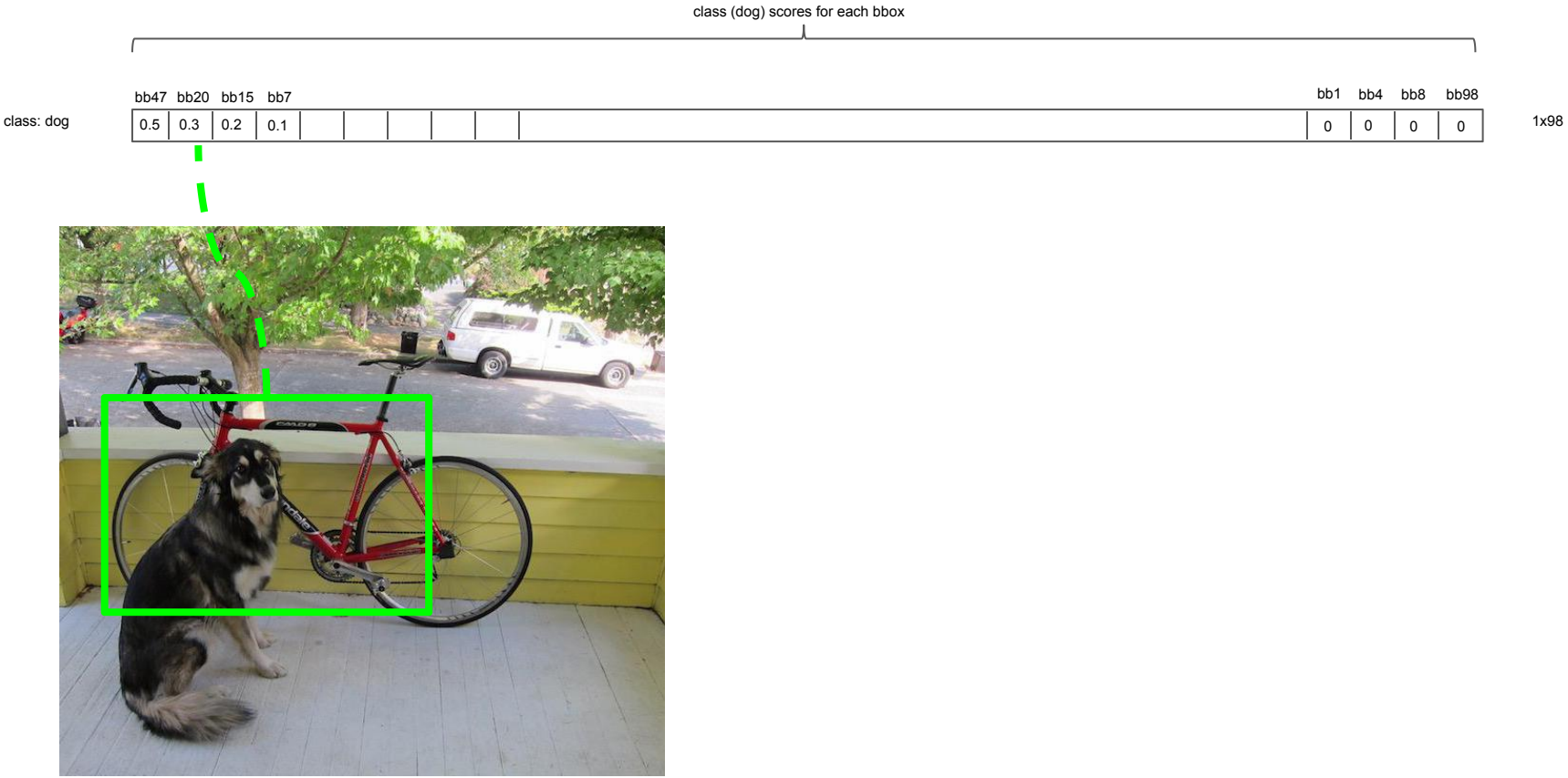
Non-Maximum Suppression: intuition



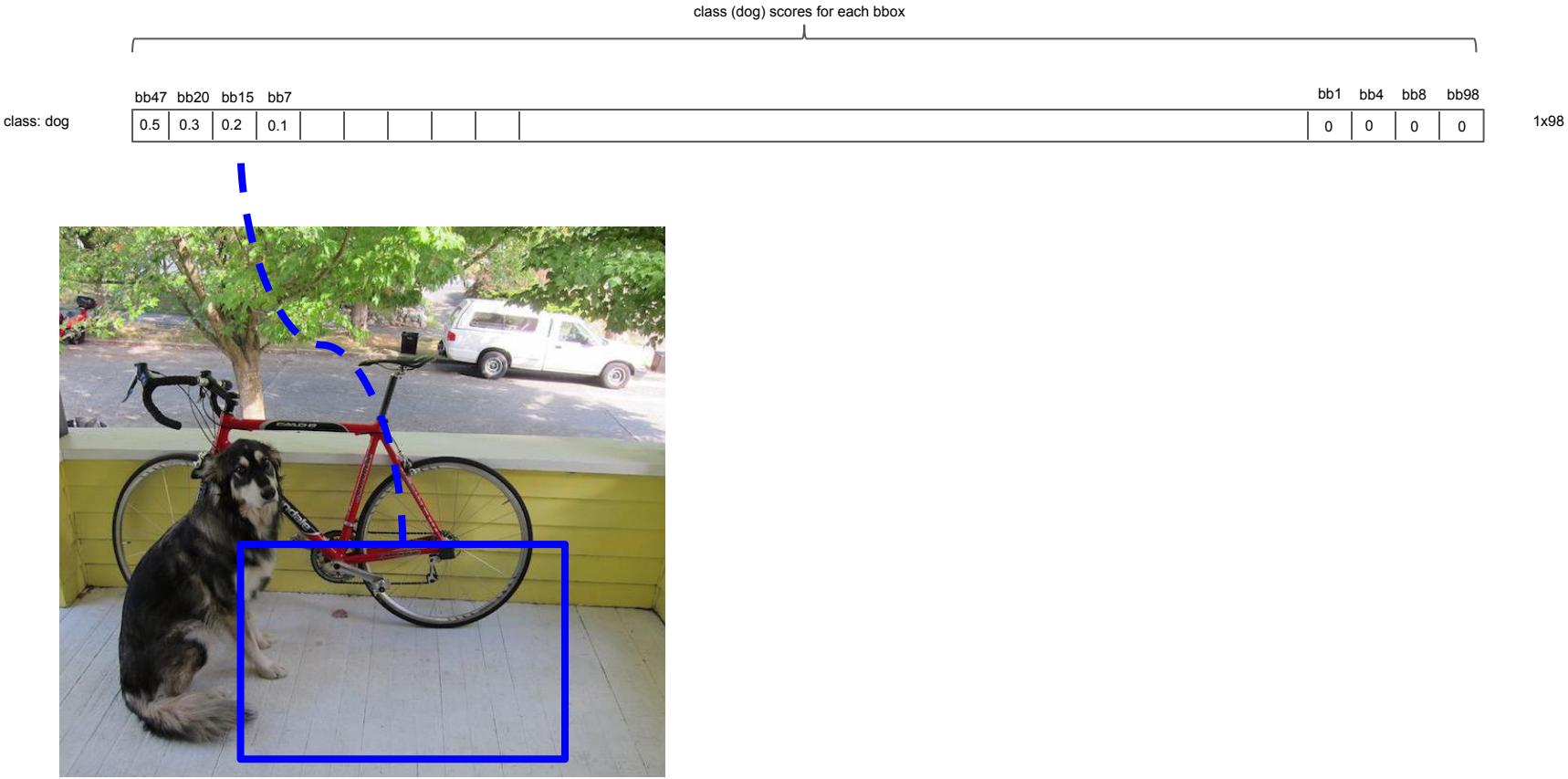
Non-Maximum Suppression: intuition



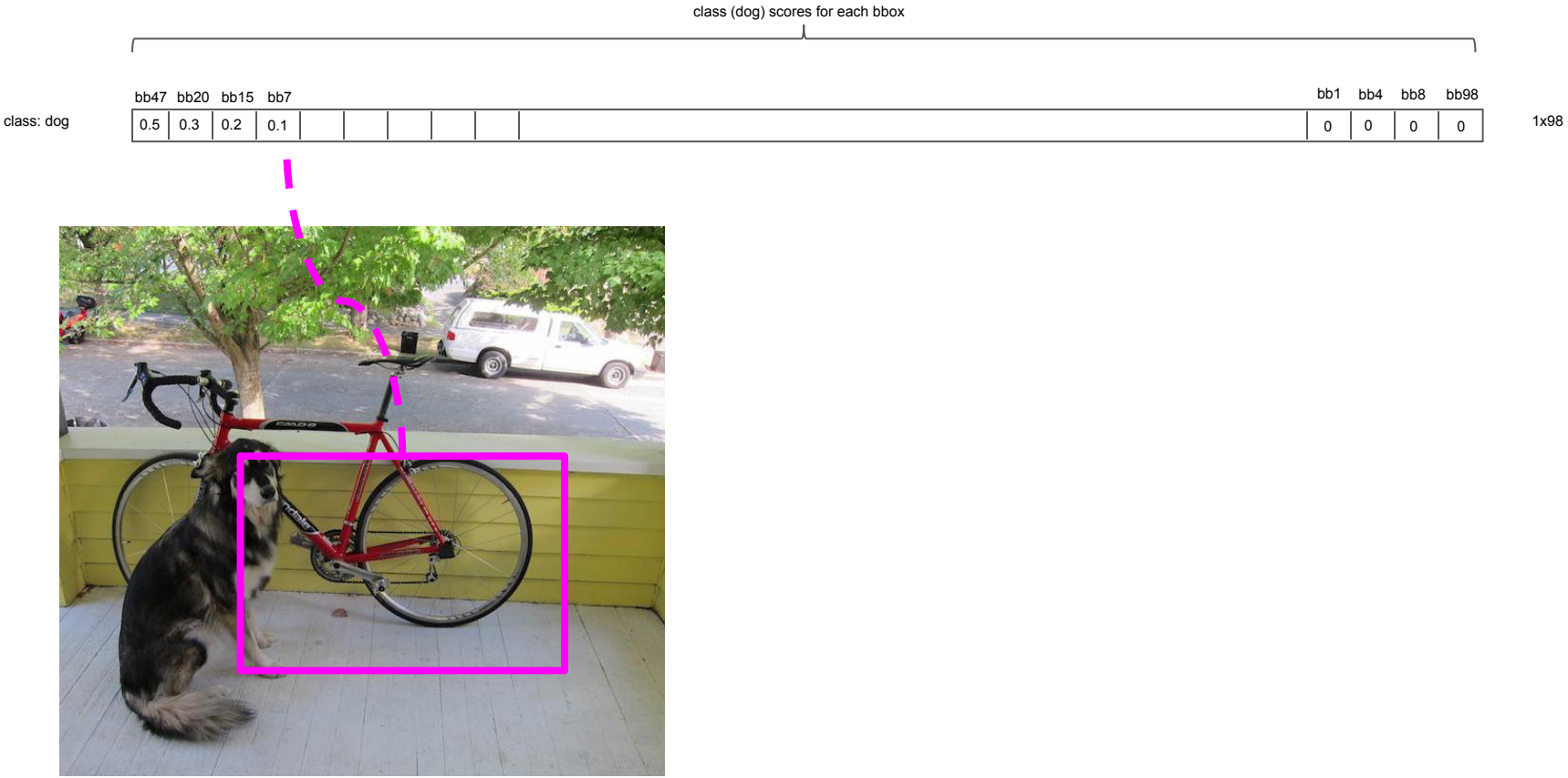
Non-Maximum Suppression: intuition



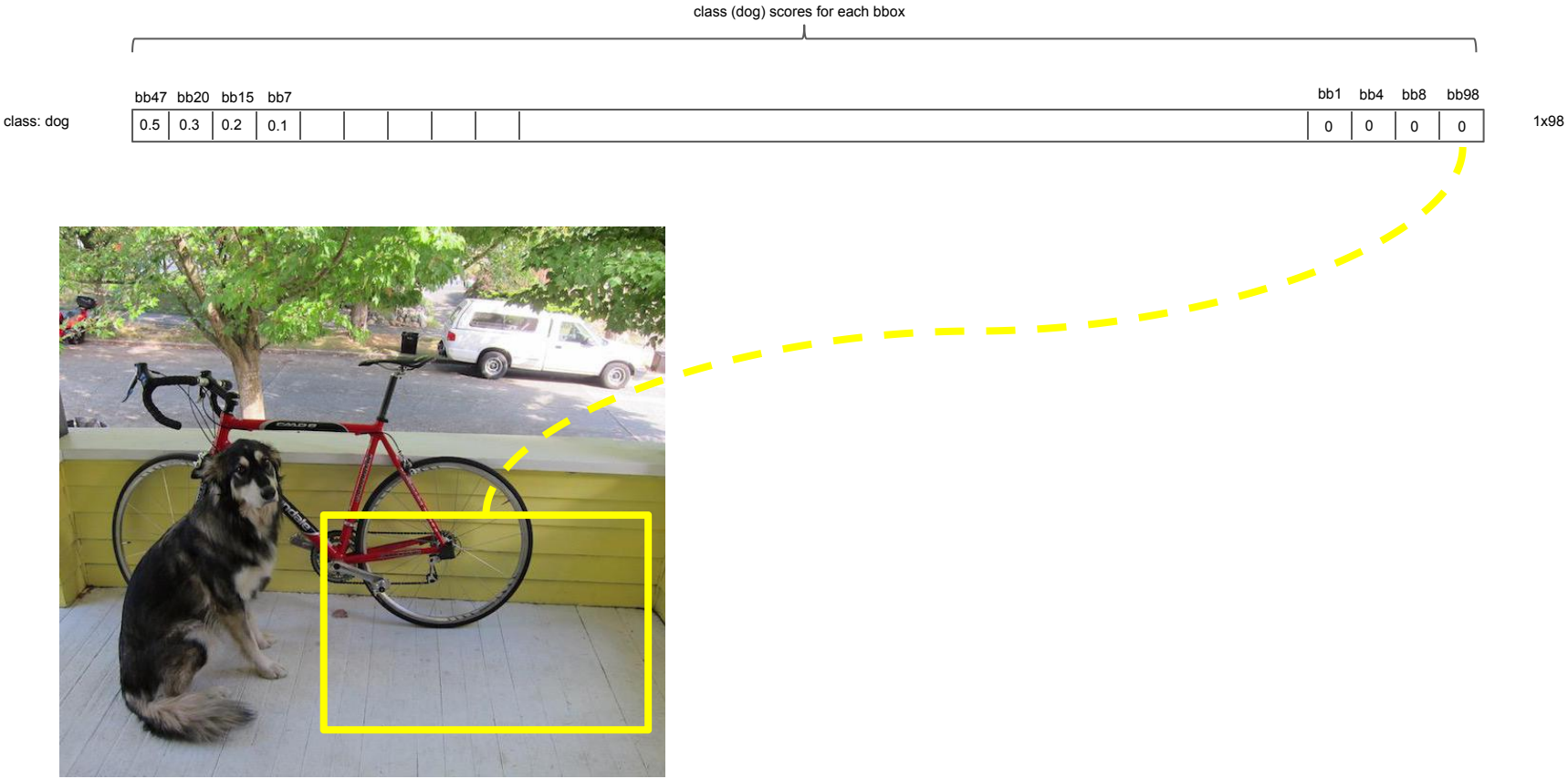
Non-Maximum Suppression: intuition



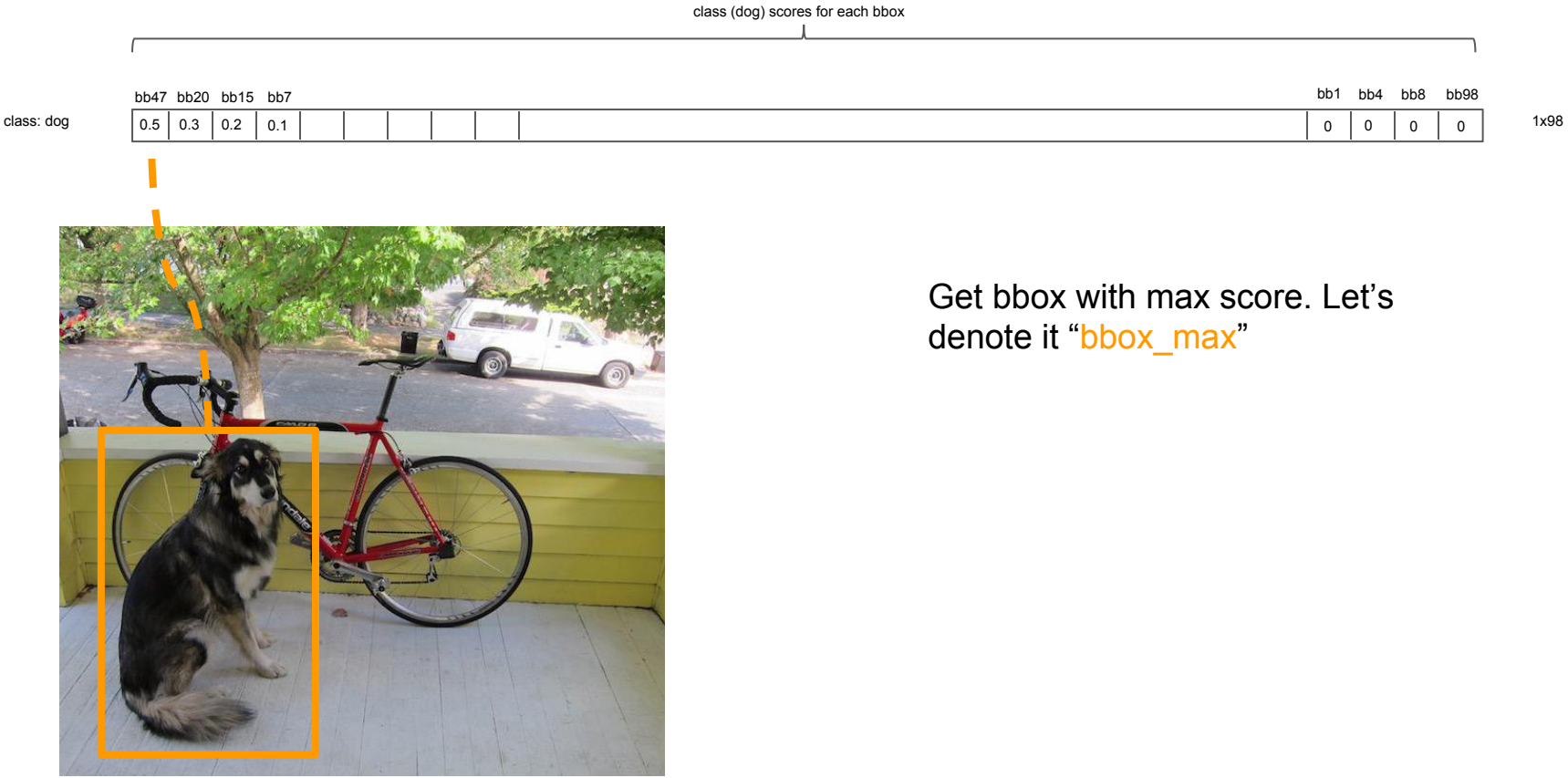
Non-Maximum Suppression: intuition



Non-Maximum Suppression: intuition

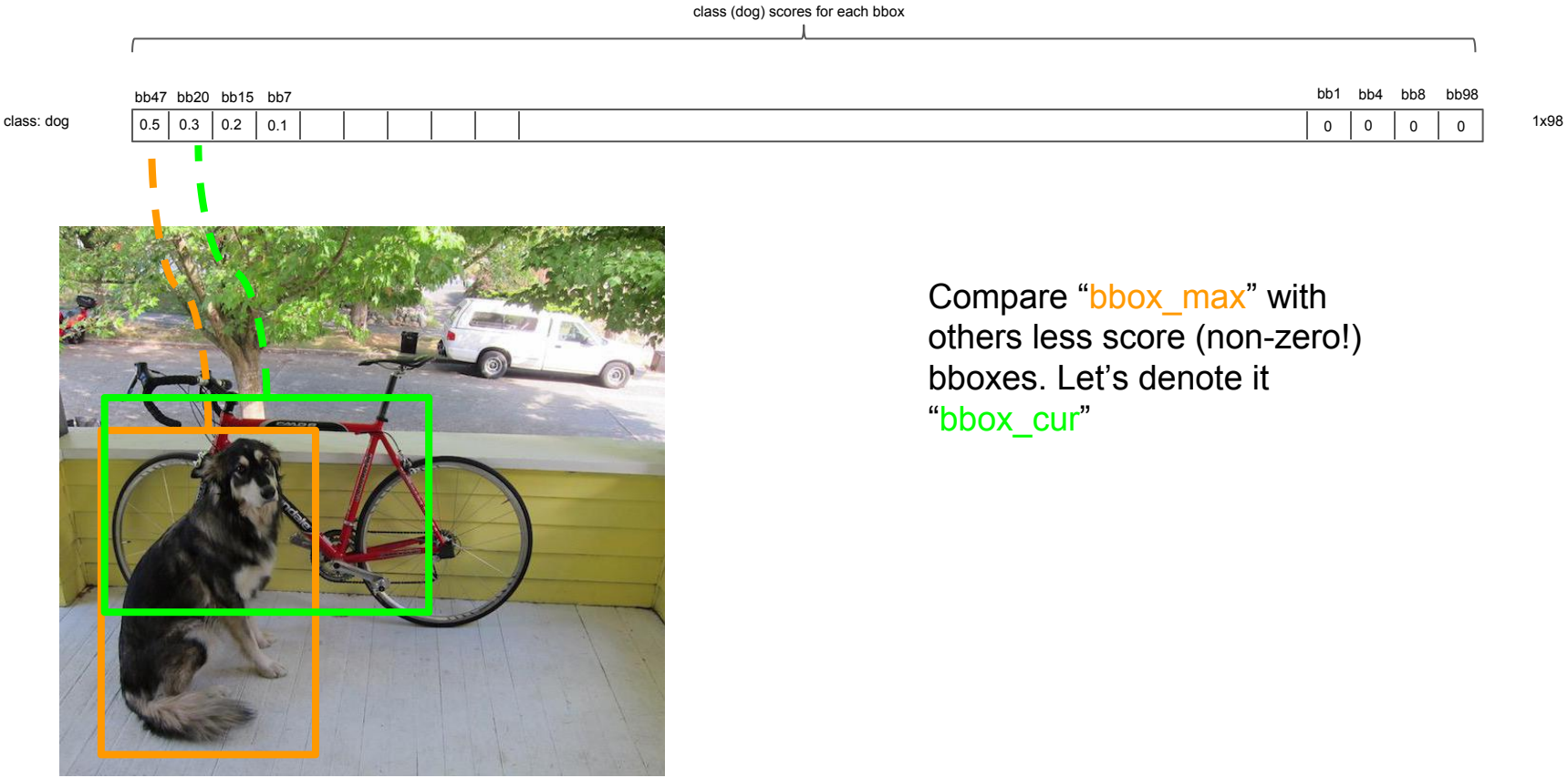


Non-Maximum Suppression: intuition



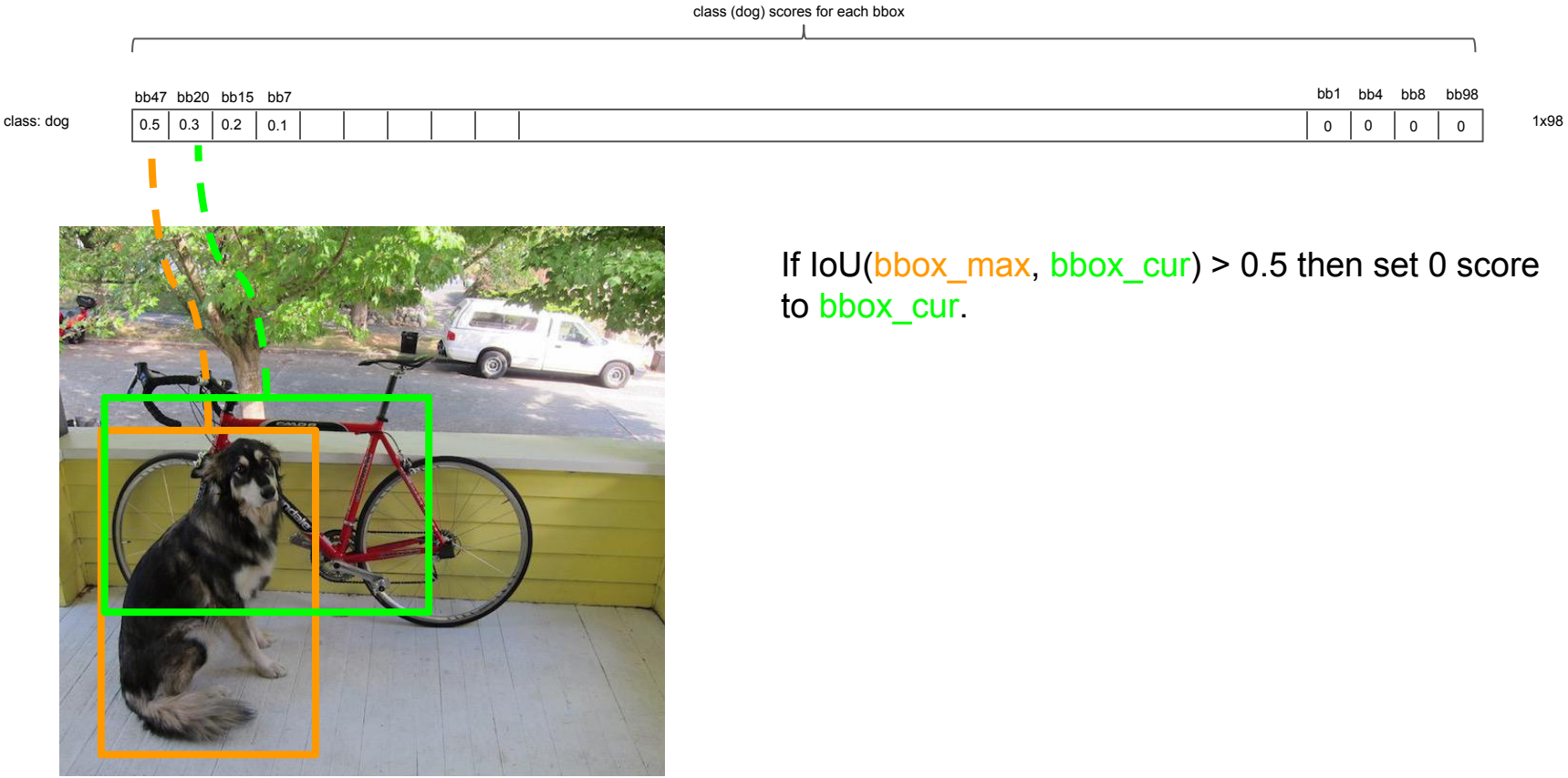
Get bbox with max score. Let's denote it "bbox_max"

Non-Maximum Suppression: intuition



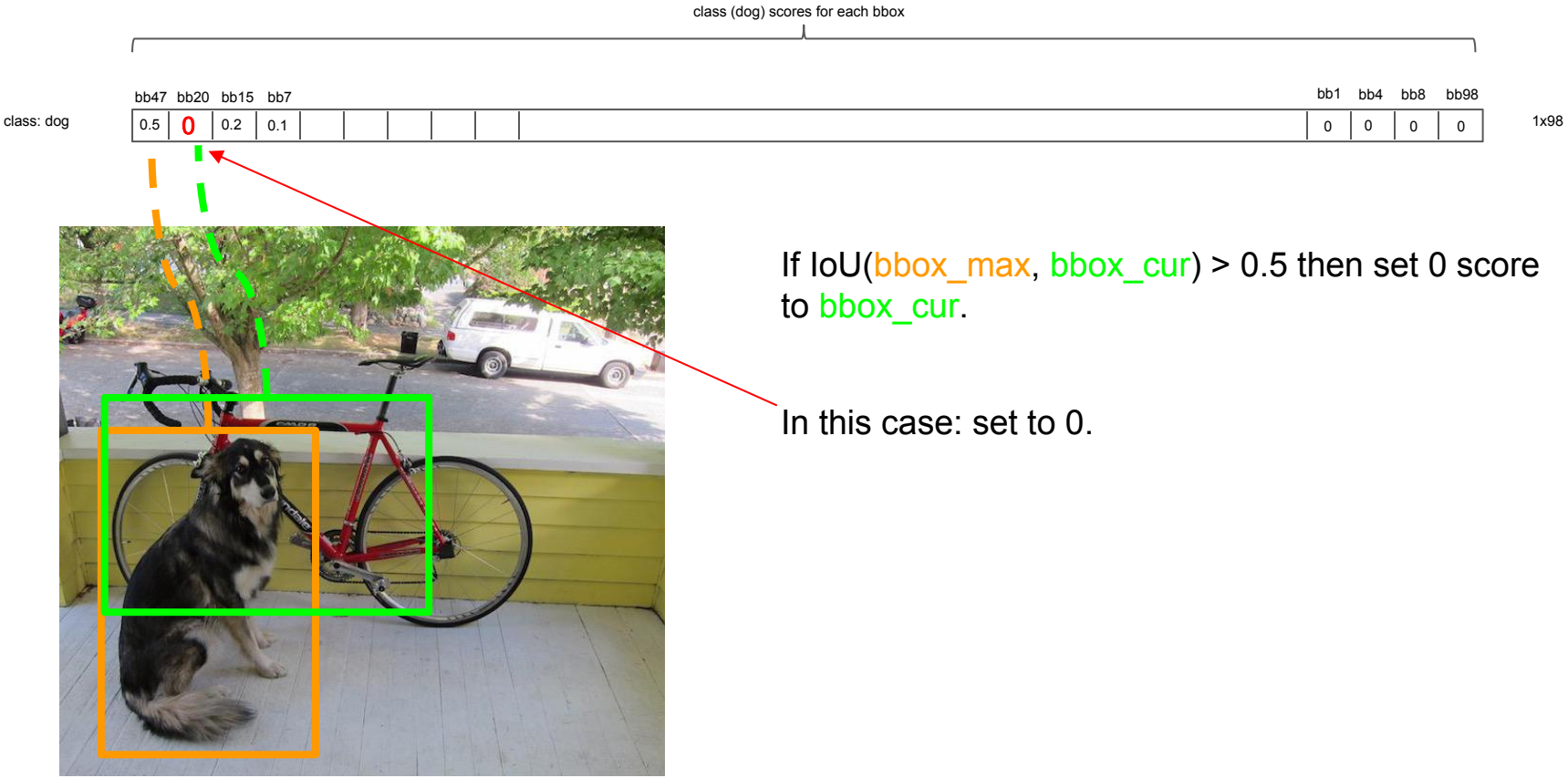
Compare “bbox_max” with others less score (non-zero!) bboxes. Let’s denote it “bbox_cur”

Non-Maximum Suppression: intuition



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

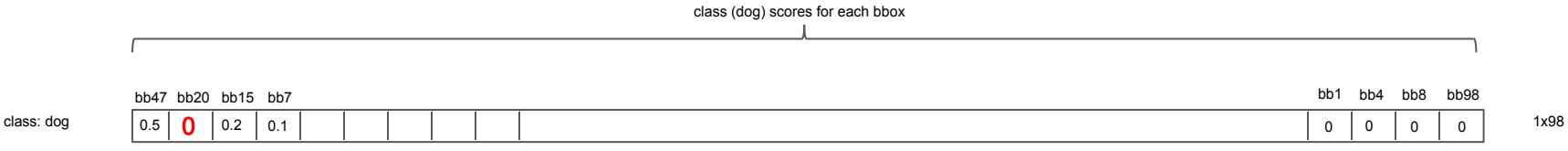
Non-Maximum Suppression: intuition

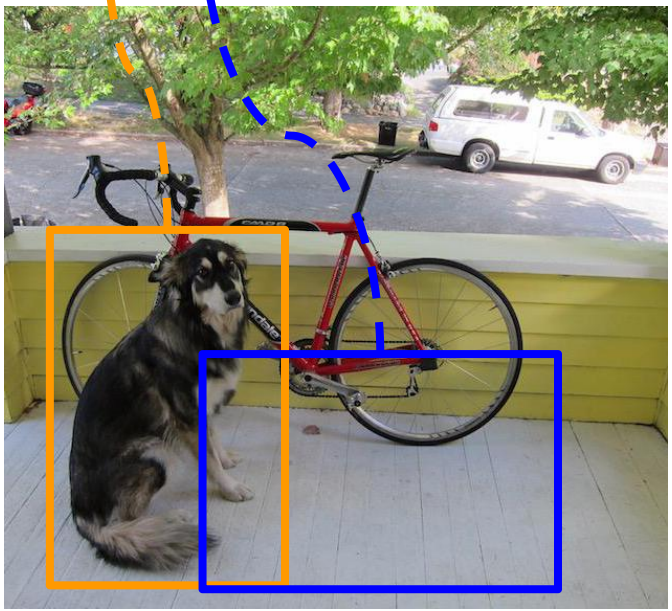


If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: set to 0.

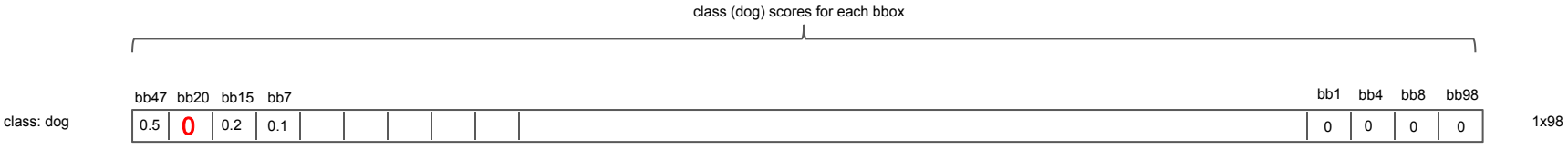
Non-Maximum Suppression: intuition

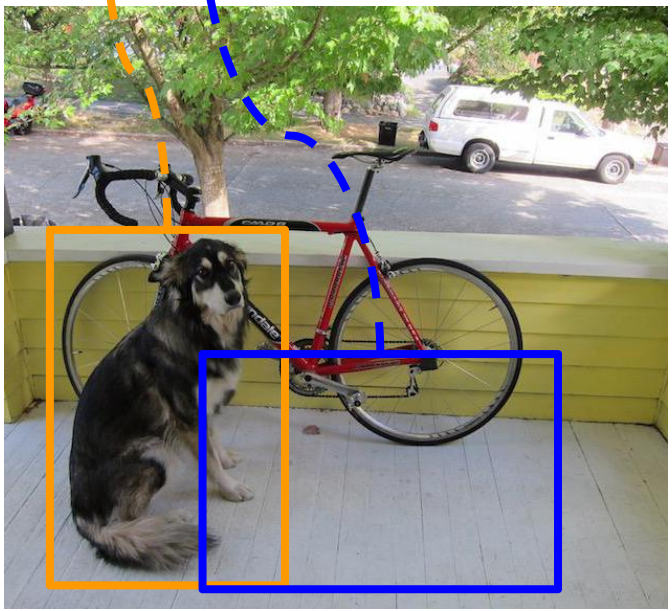




Go to next bbox_cur.

Non-Maximum Suppression: intuition

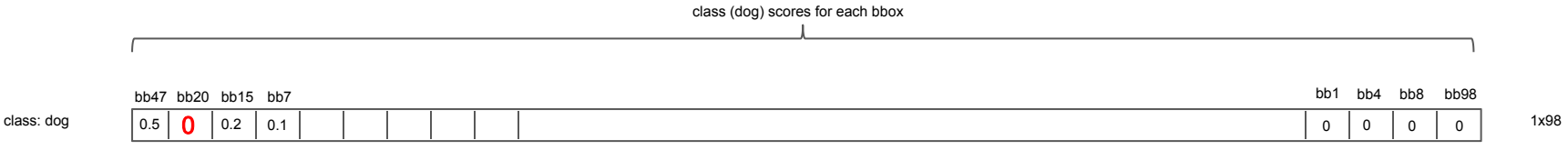


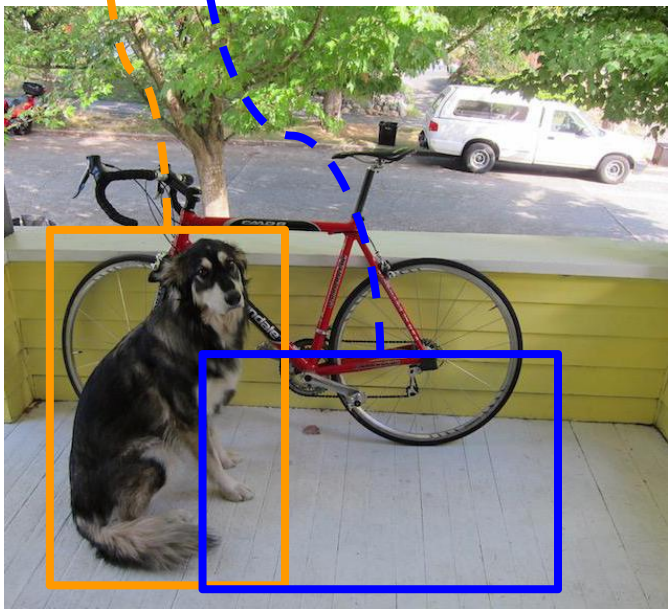


Go to next bbox_cur.

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

Non-Maximum Suppression: intuition



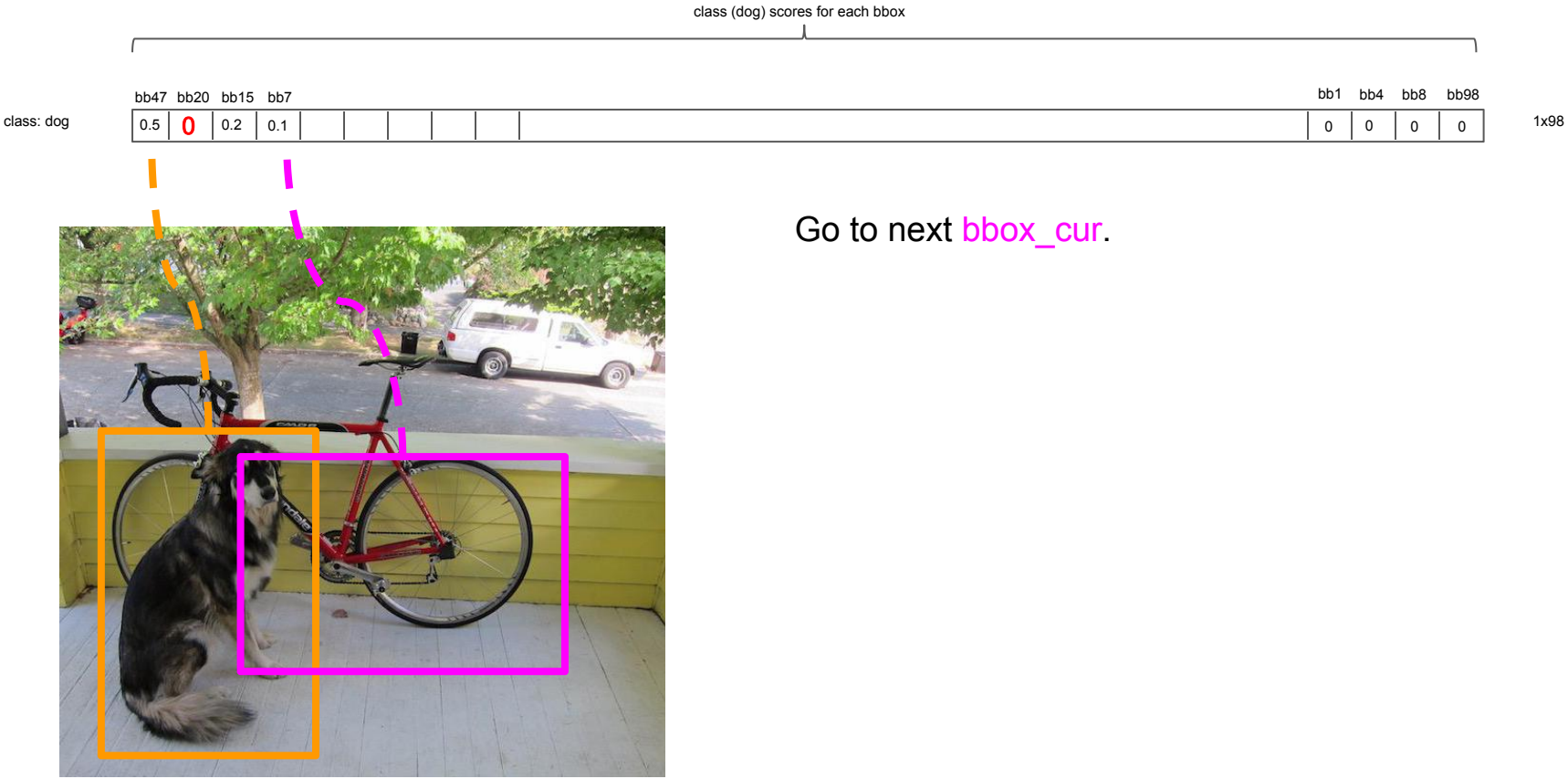


Go to next bbox_cur.

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

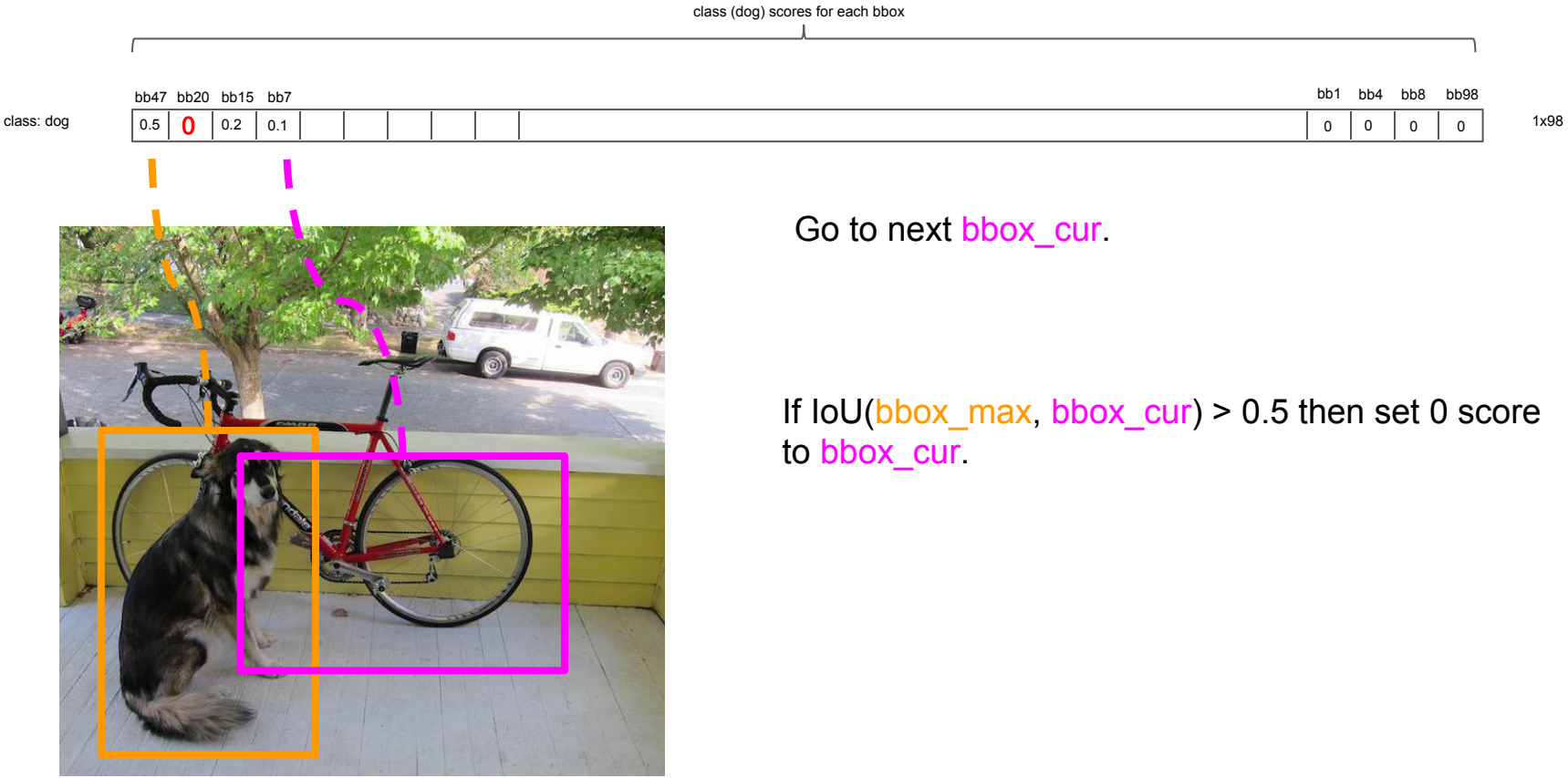
In this case: continue.

Non-Maximum Suppression: intuition



Go to next bbox_cur.

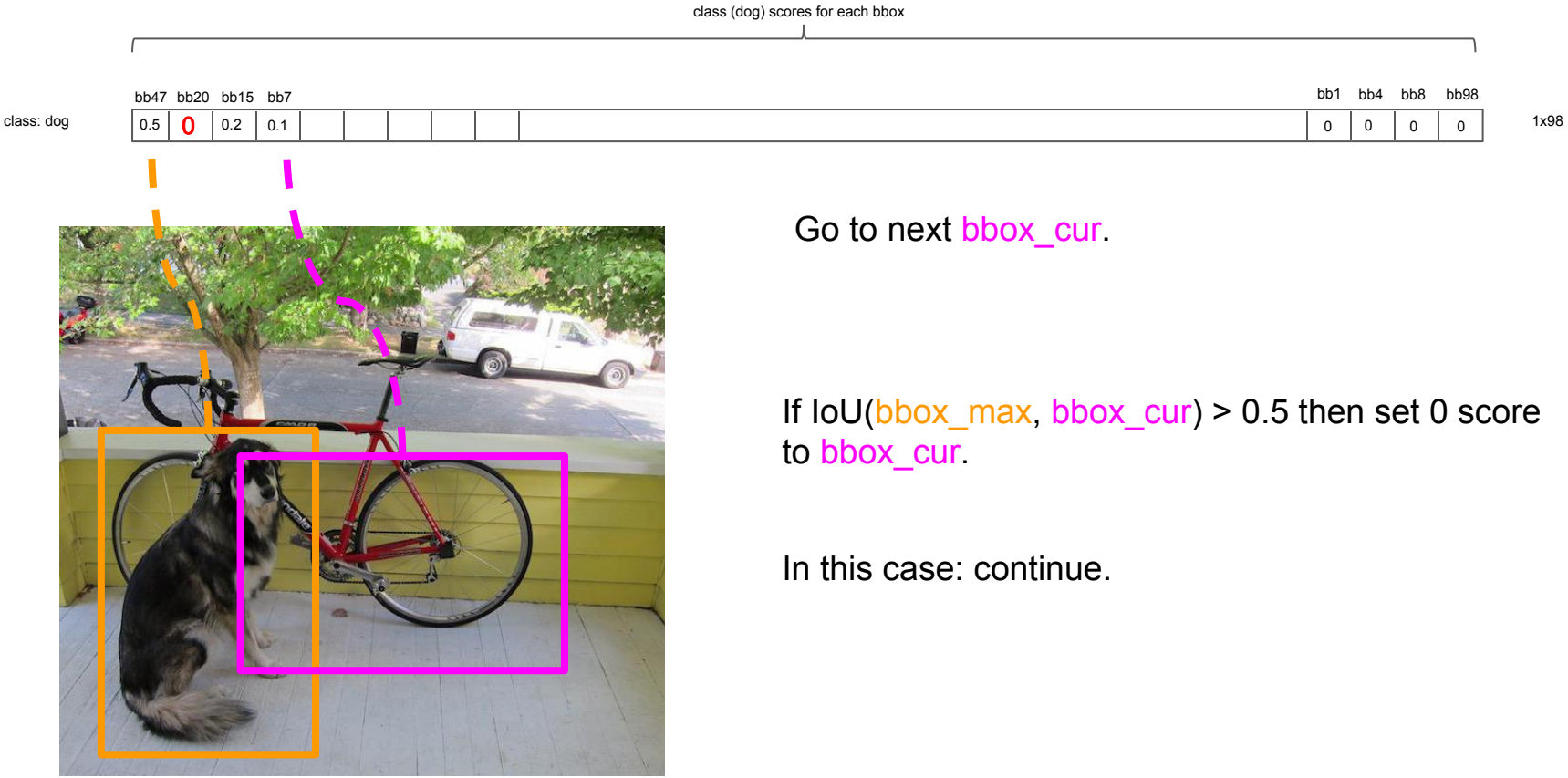
Non-Maximum Suppression: intuition



Go to next bbox_cur.

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

Non-Maximum Suppression: intuition

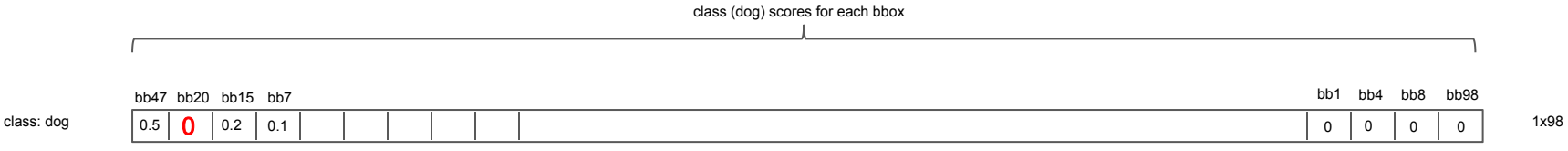


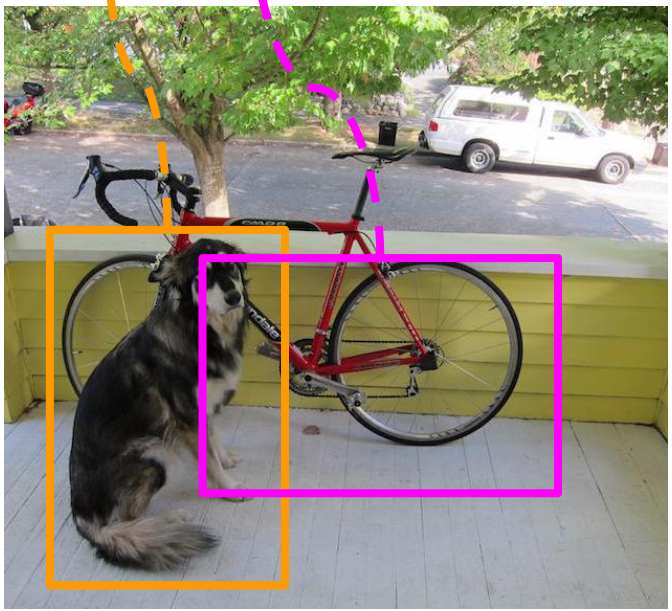
Go to next bbox_cur.

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

In this case: continue.

Non-Maximum Suppression: intuition





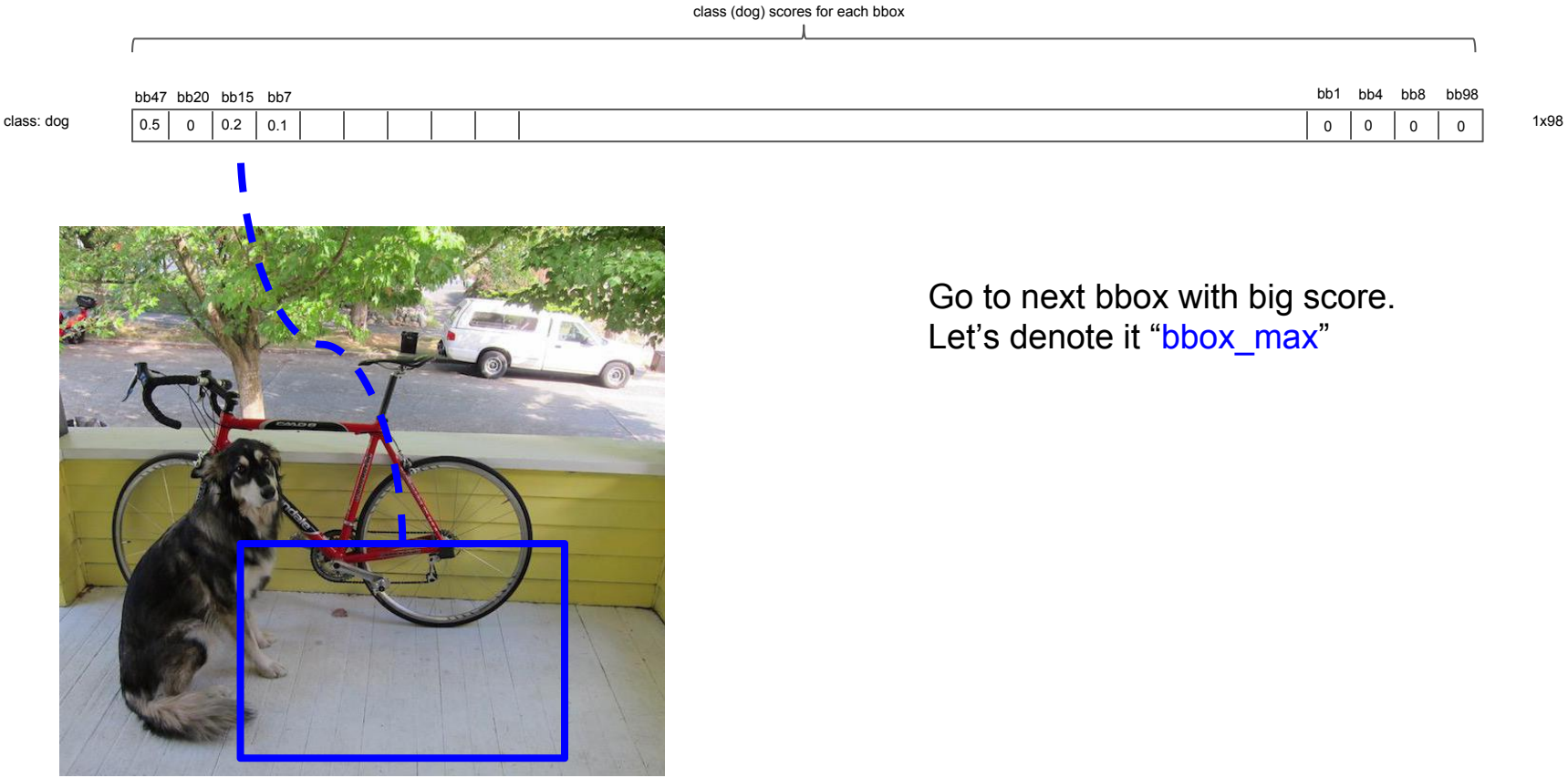
Go to next bbox_cur.

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

In this case: continue.

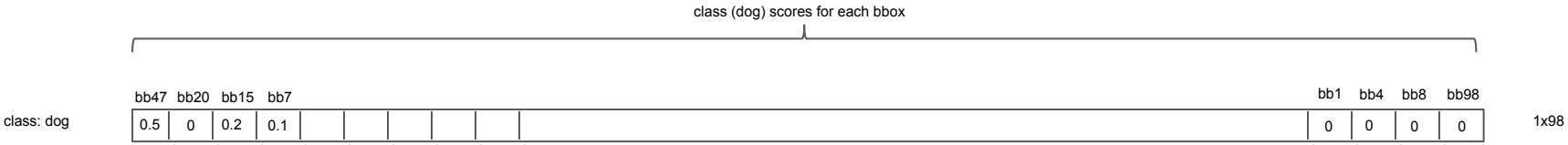
Do this procedure for other “bbox_cur”. After that ...

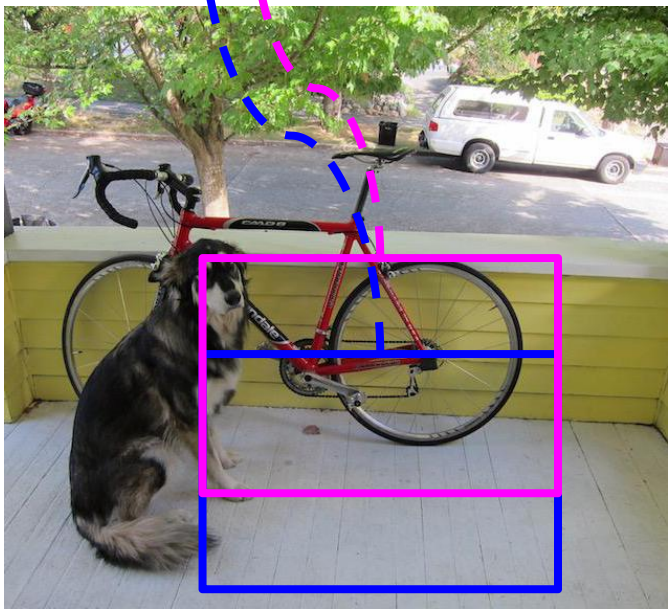
Non-Maximum Suppression: intuition



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

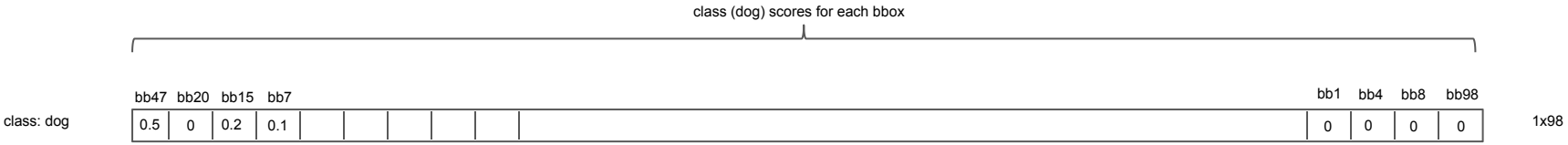
Non-Maximum Suppression: intuition

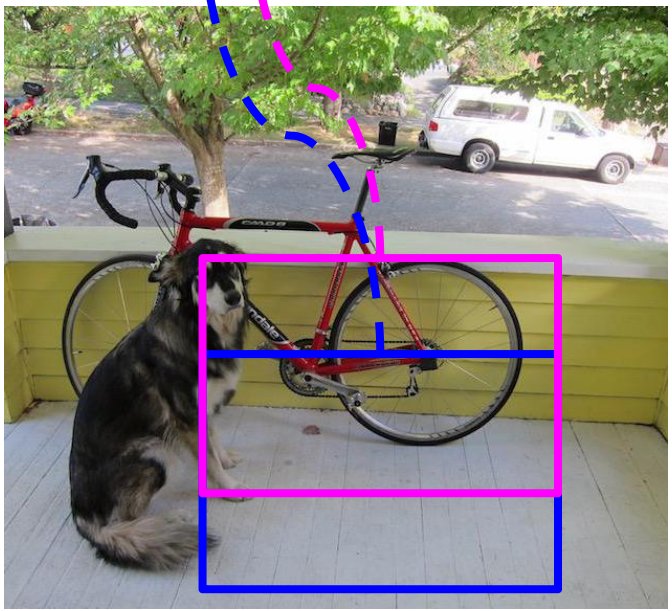




Go to next bbox_cur.

Non-Maximum Suppression: intuition

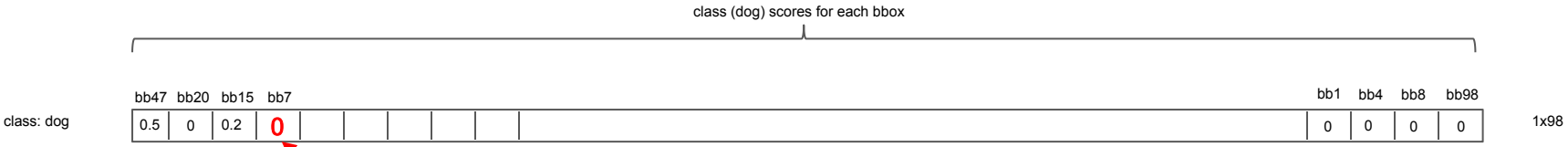


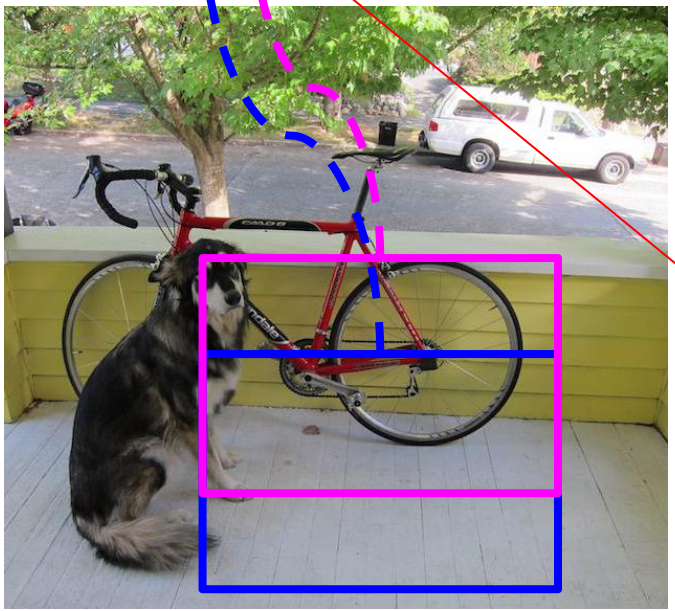


Go to next bbox_cur.

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

Non-Maximum Suppression: intuition





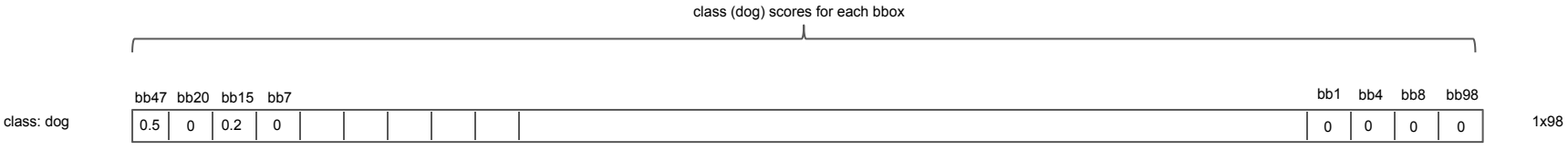
Go to next bbox_cur.

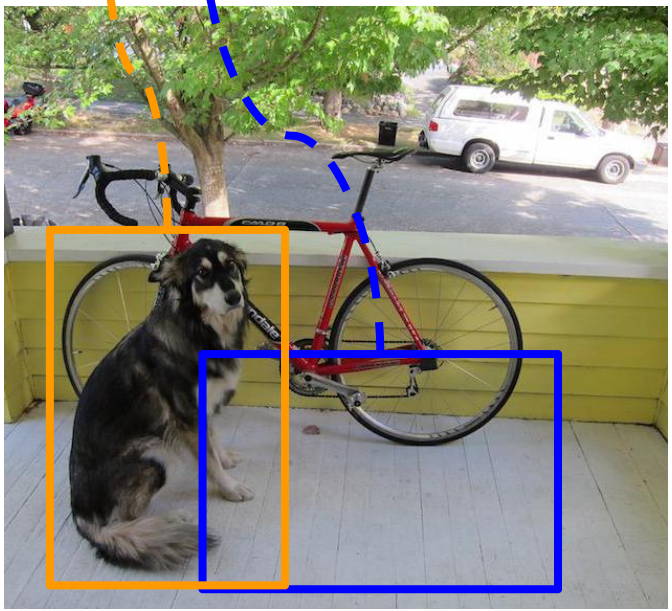
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: set to 0.

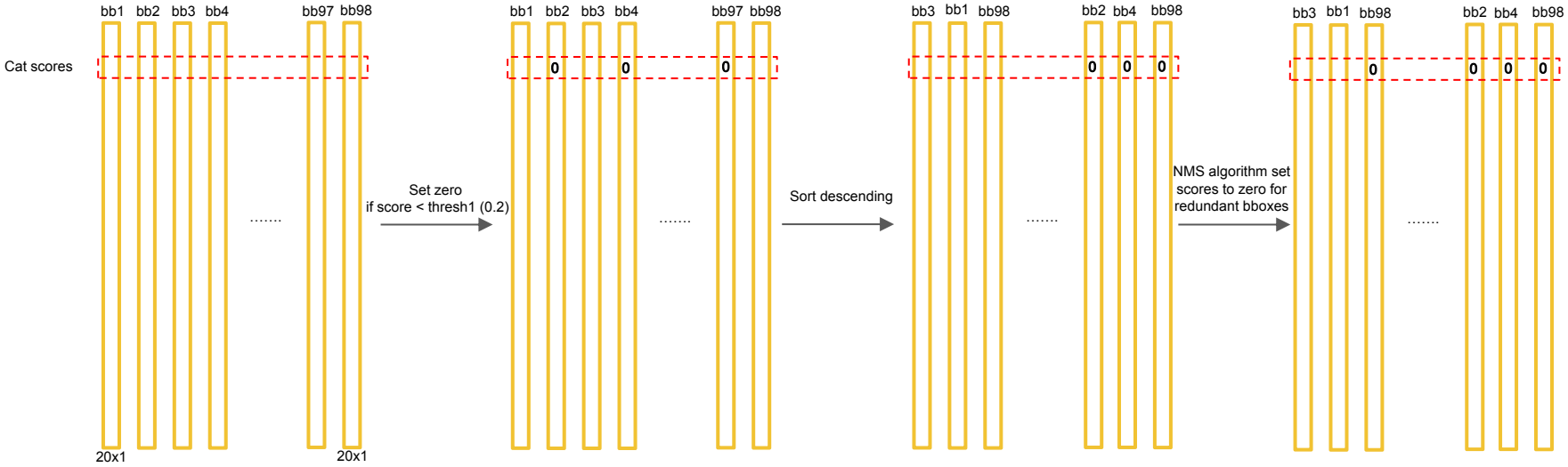
Do this procedure for other “bbox_max” and for other corresponding “bbox_cur”.

Non-Maximum Suppression: intuition

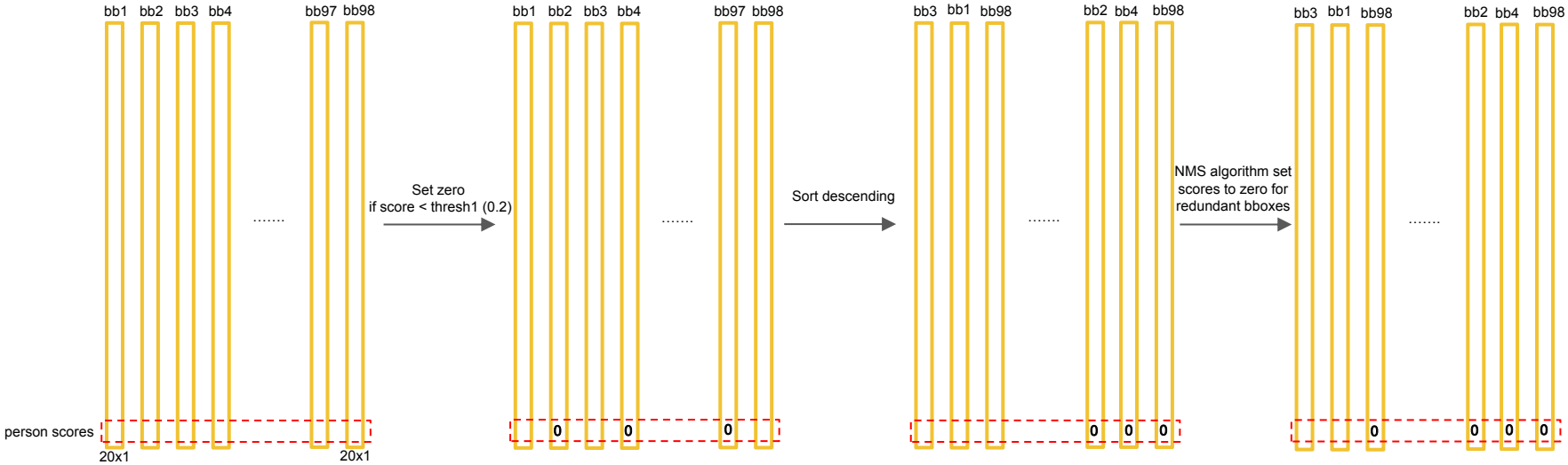




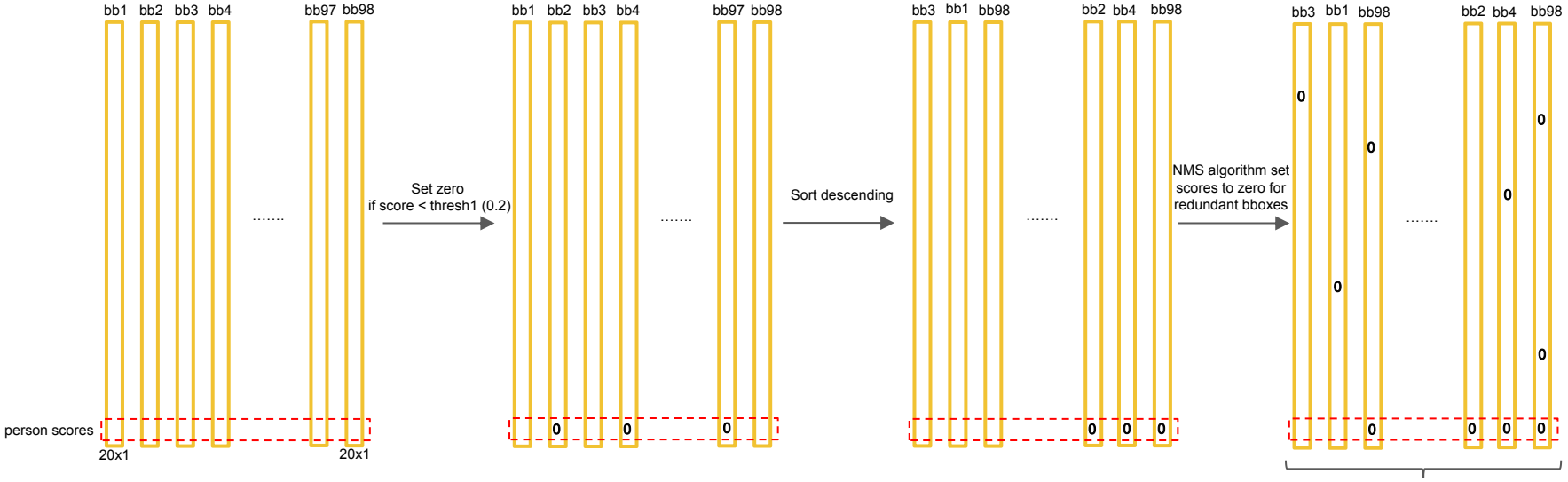
After comparison almost all pairs of bboxes the only two bboxes left with non-zero class score value.



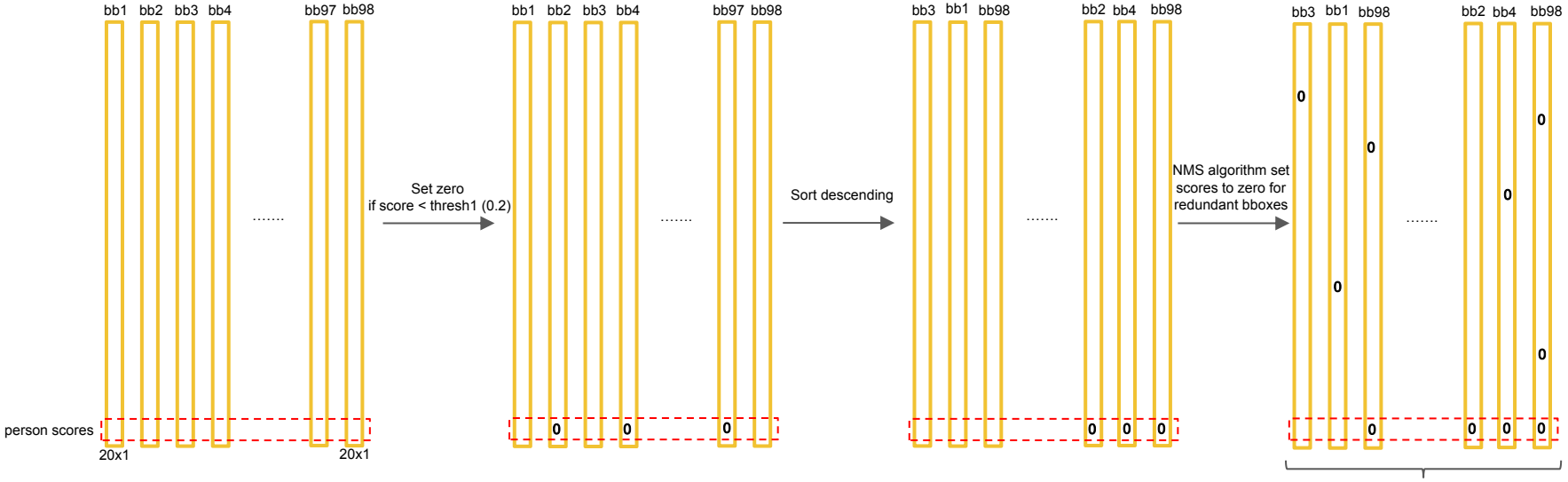
Do this procedure for next class



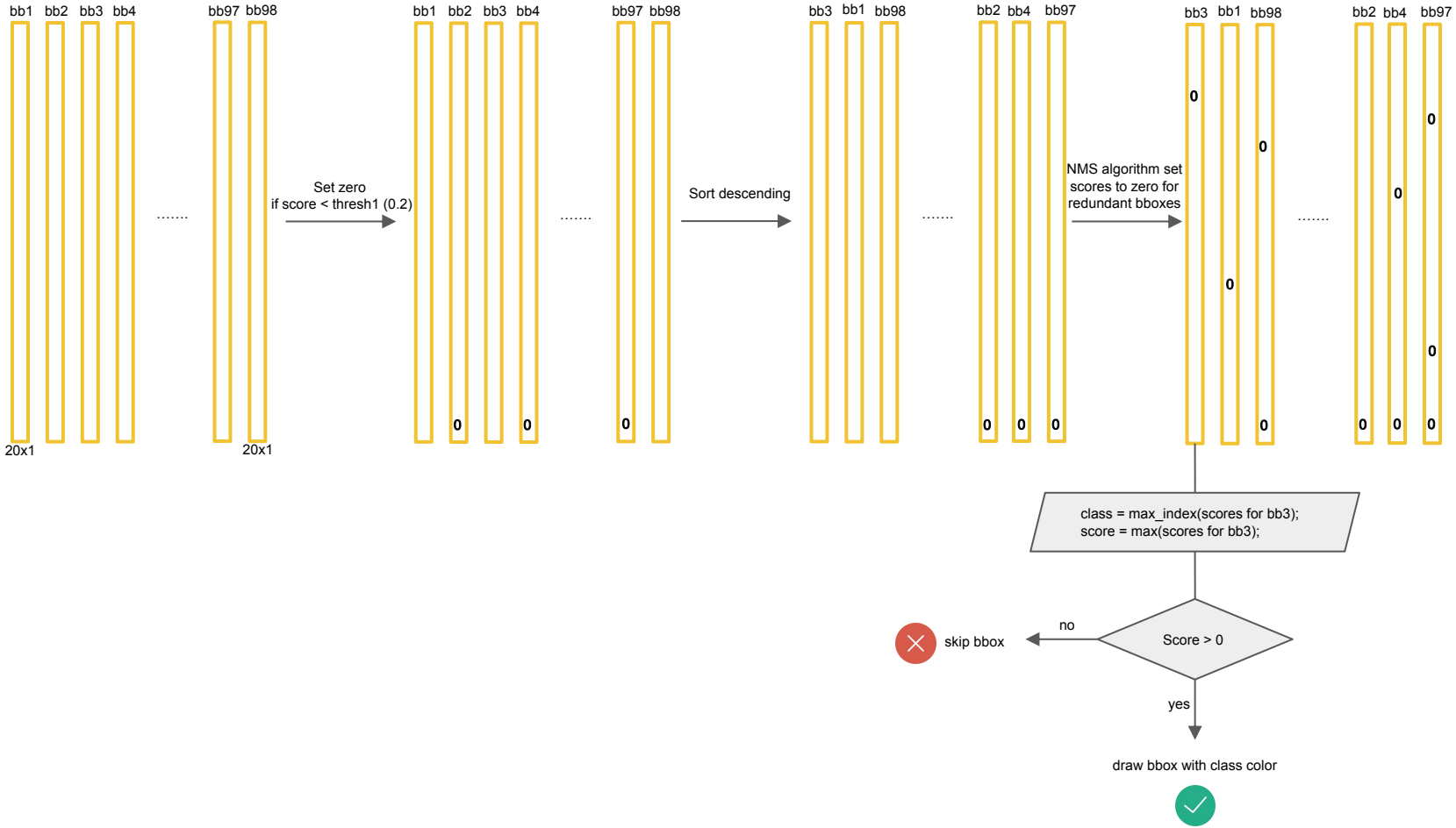
Do this procedure for all classes

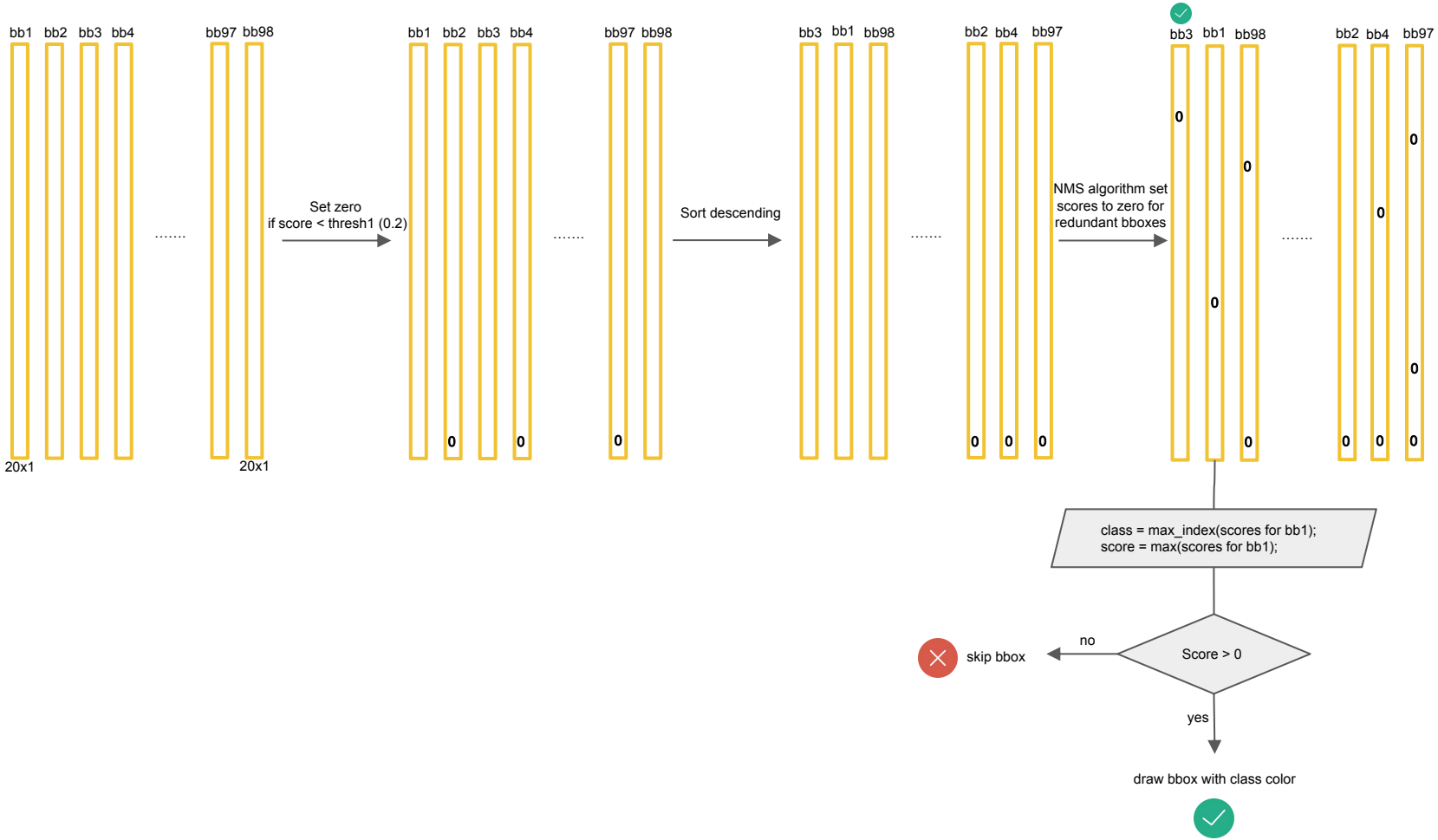


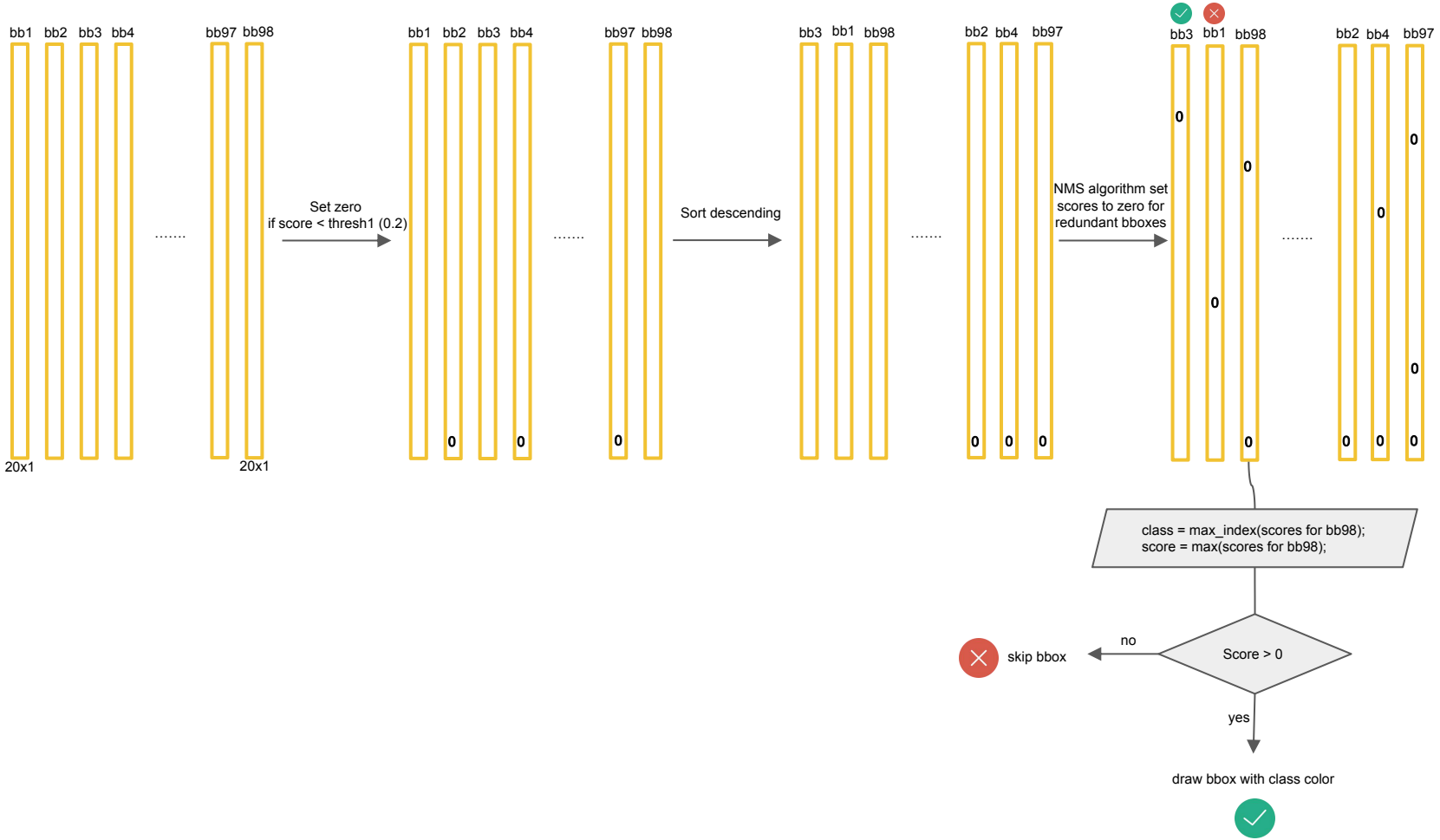
After this procedure -
a lot of zeros

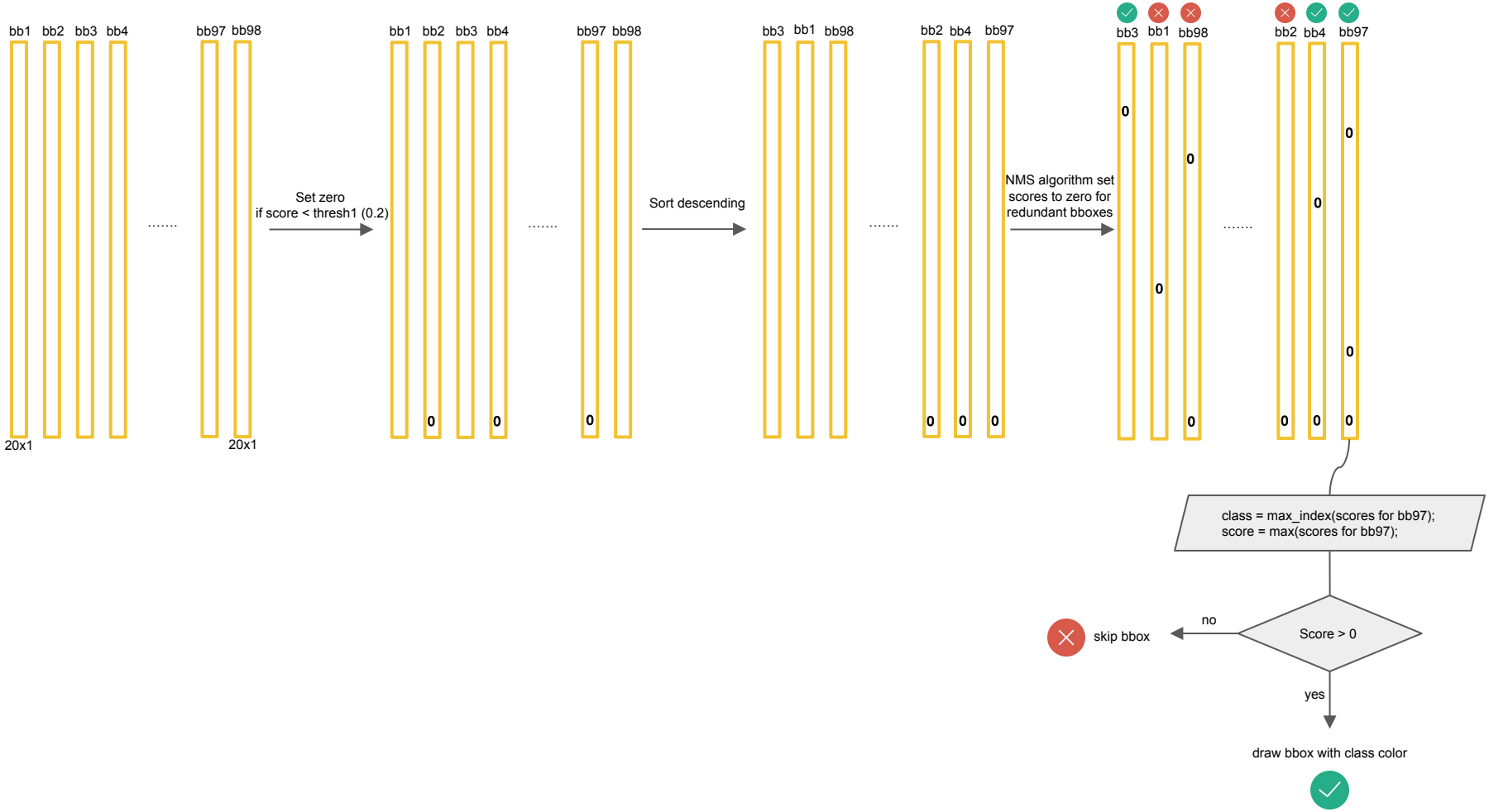


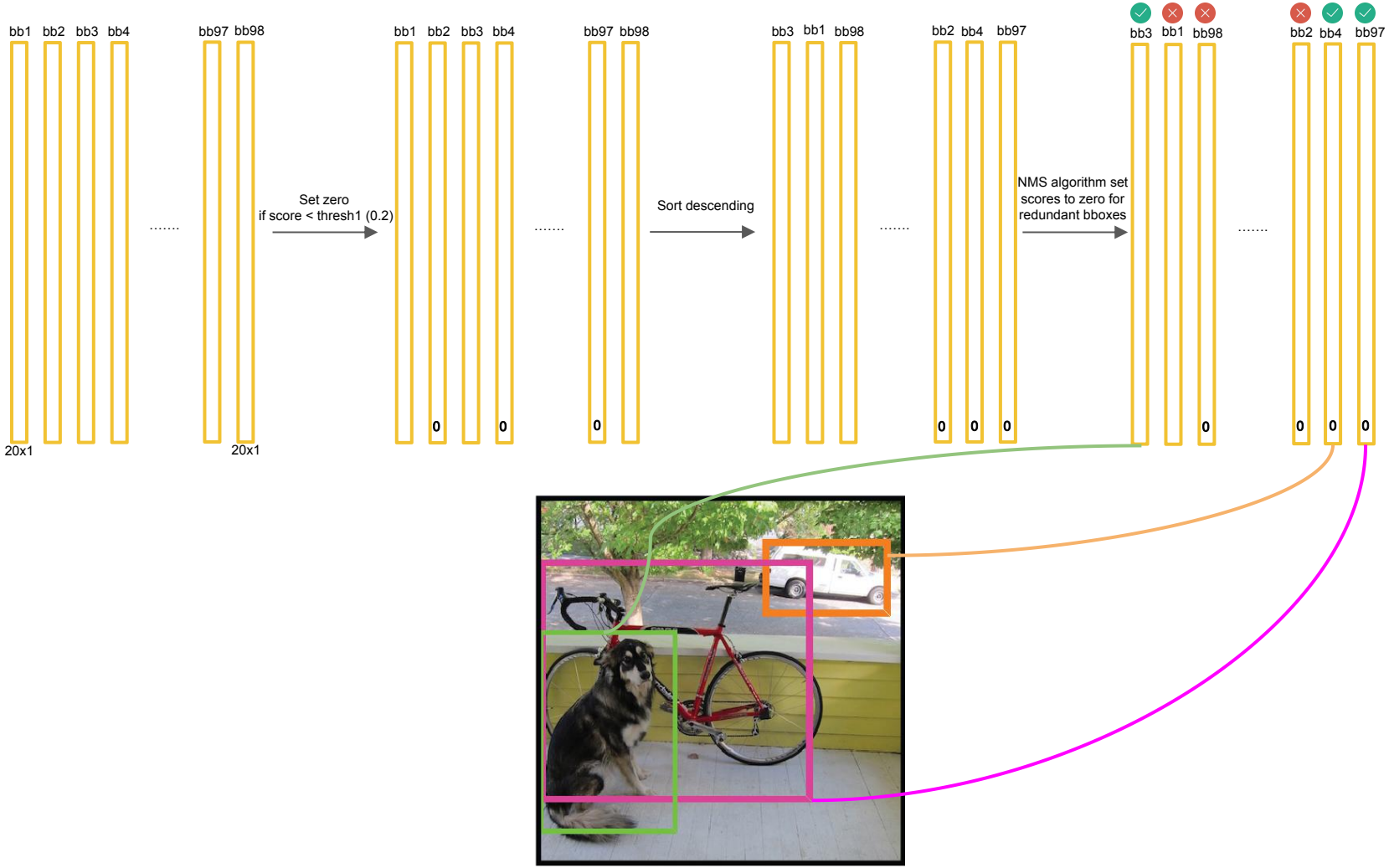
Select bboxes to draw by class score values


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


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

Ten Tricks

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?							✓	✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Table from *YOLO9000: Better, Faster, Stronger* (https://arxiv.org/abs/1612.08242)

Thank you!