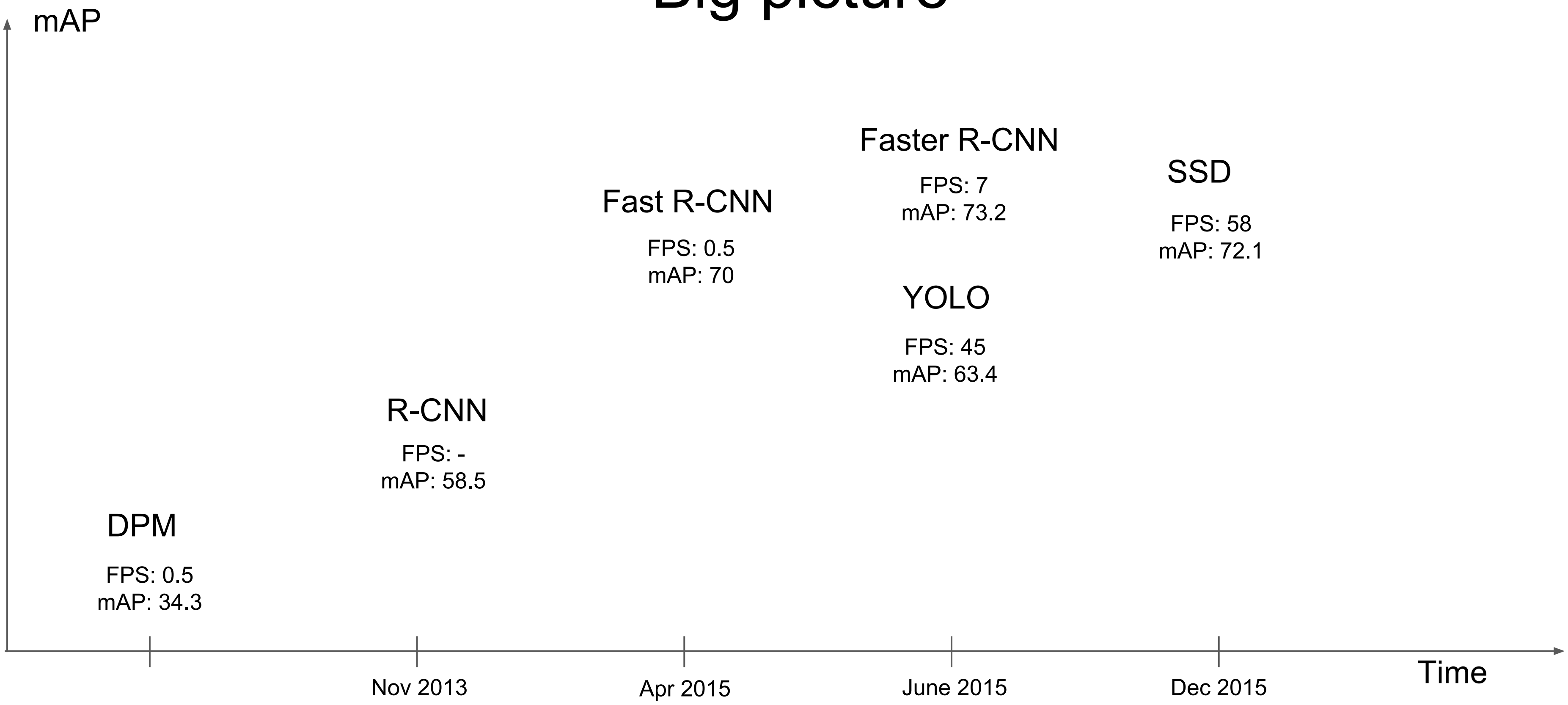
Обзор от deepsystems.io

YOLO

You Only Look Once: Unified, Real-Time Object Detection
Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

Big picture



mAP

DPM

FPS: 0.5
mAP: 34.3

R-CNN

FPS: -
mAP: 58.5

Fast R-CNN

FPS: 0.5
mAP: 70

Faster R-CNN

FPS: 7
mAP: 73.2

YOLO

FPS: 45
mAP: 63.4

SSD

FPS: 58
mAP: 72.1

Nov 2013

Apr 2015

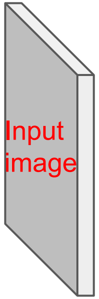
June 2015

Dec 2015

Time

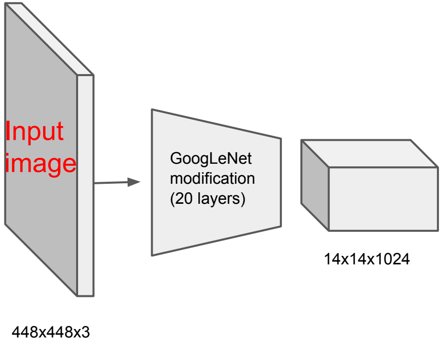
Результаты на тестовой выборки Pascal VOC 2007. Обучение на trainval sets 2007+2012

Inference

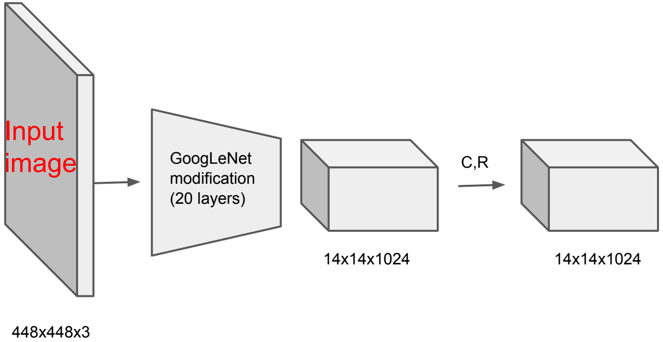


448x448x3

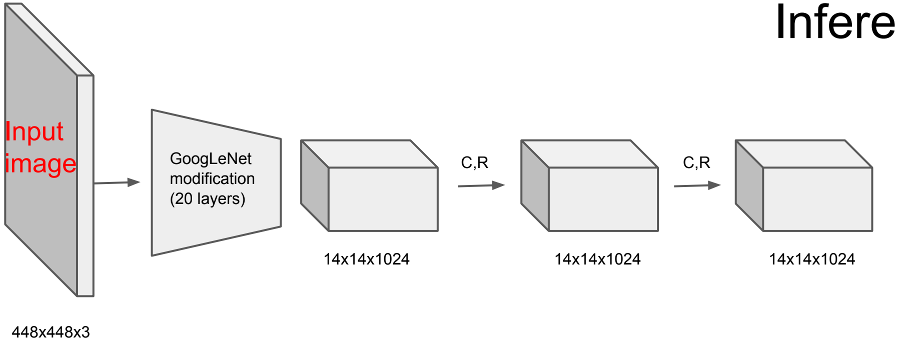
Inference


Input image
GoogLeNet modification (20 layers)
14x14x1024
448x448x3

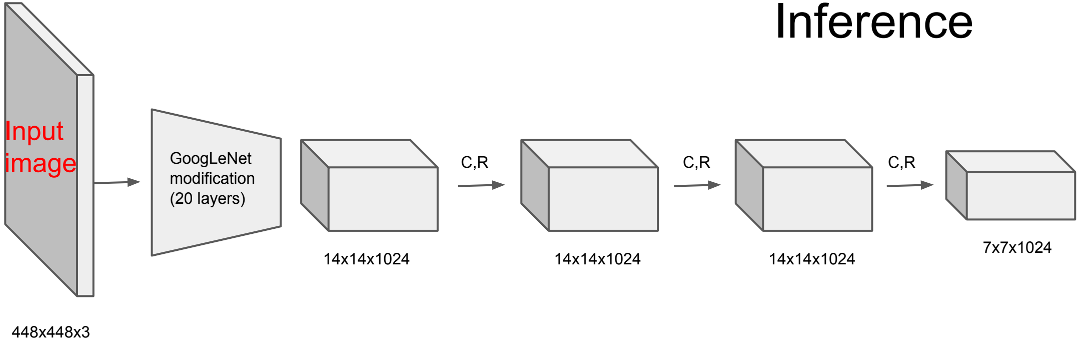
Inference


Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024

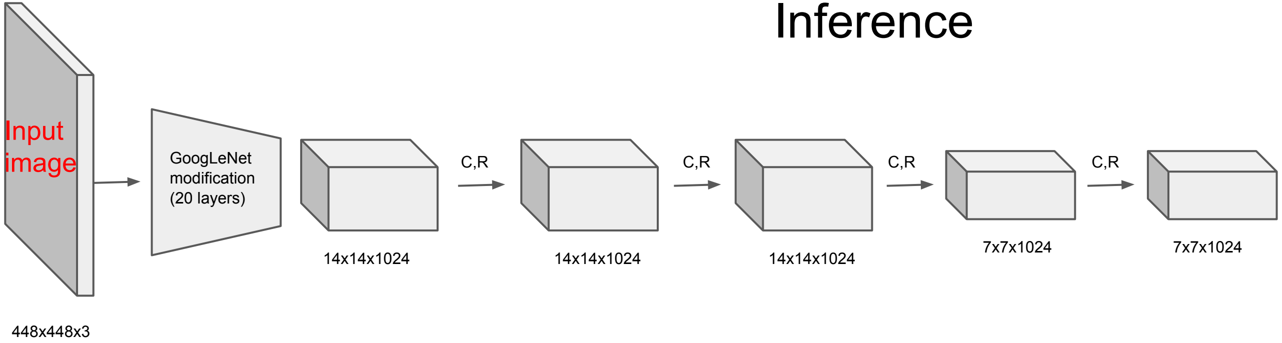
Inference


Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024

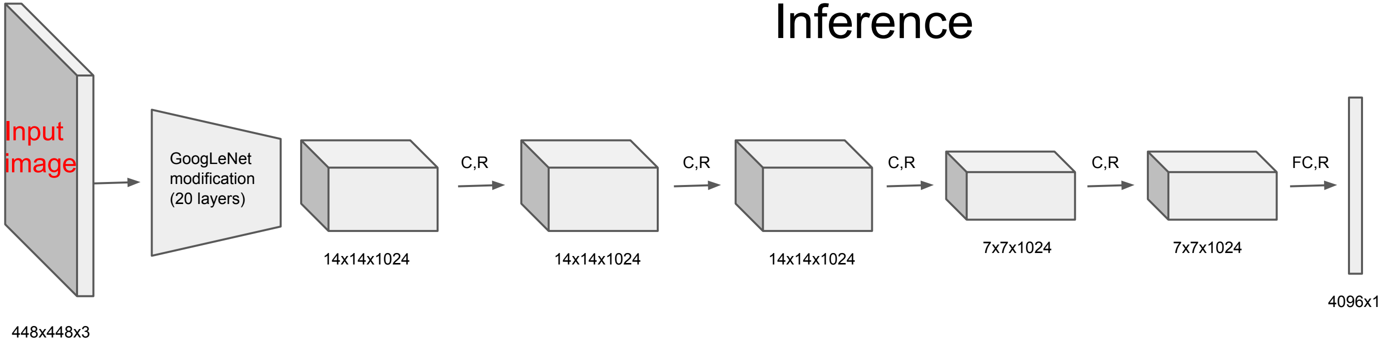
Inference


Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024

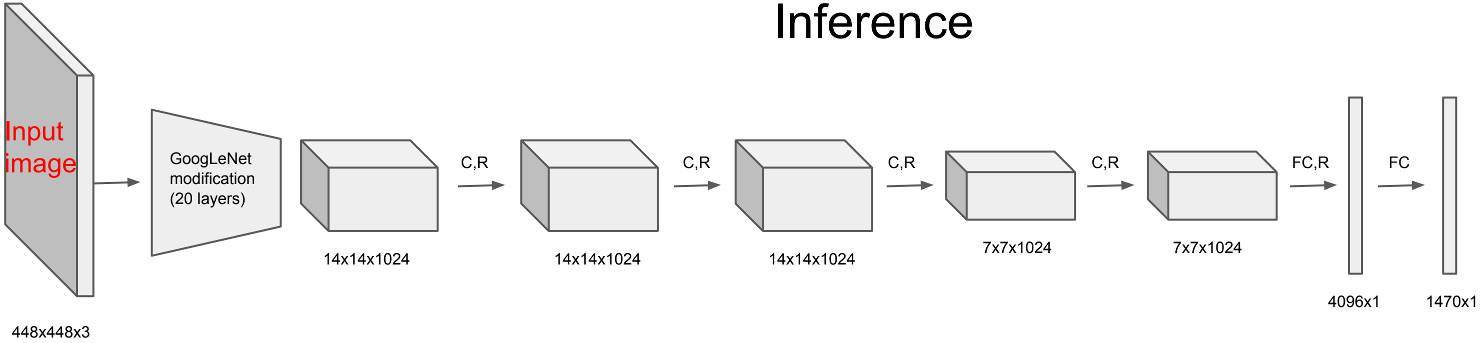
Inference


Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024

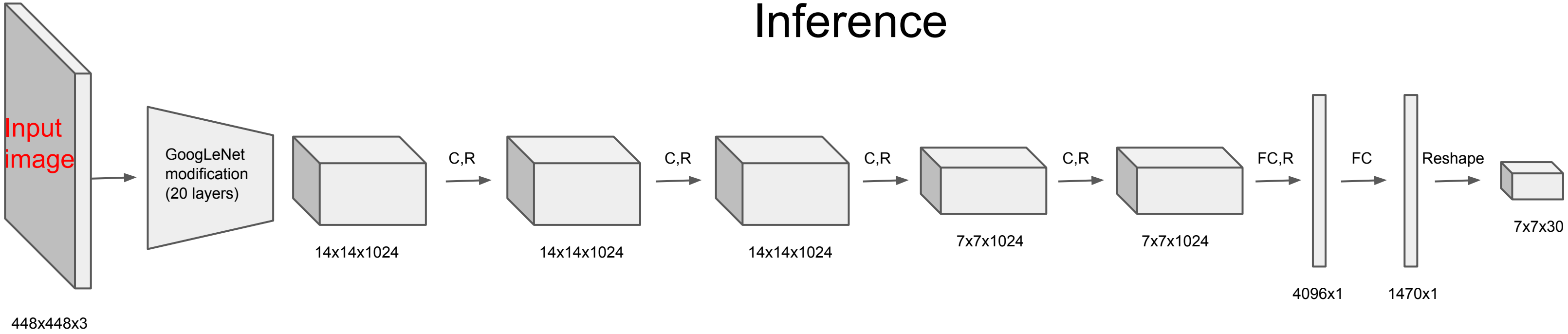
Inference


Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1

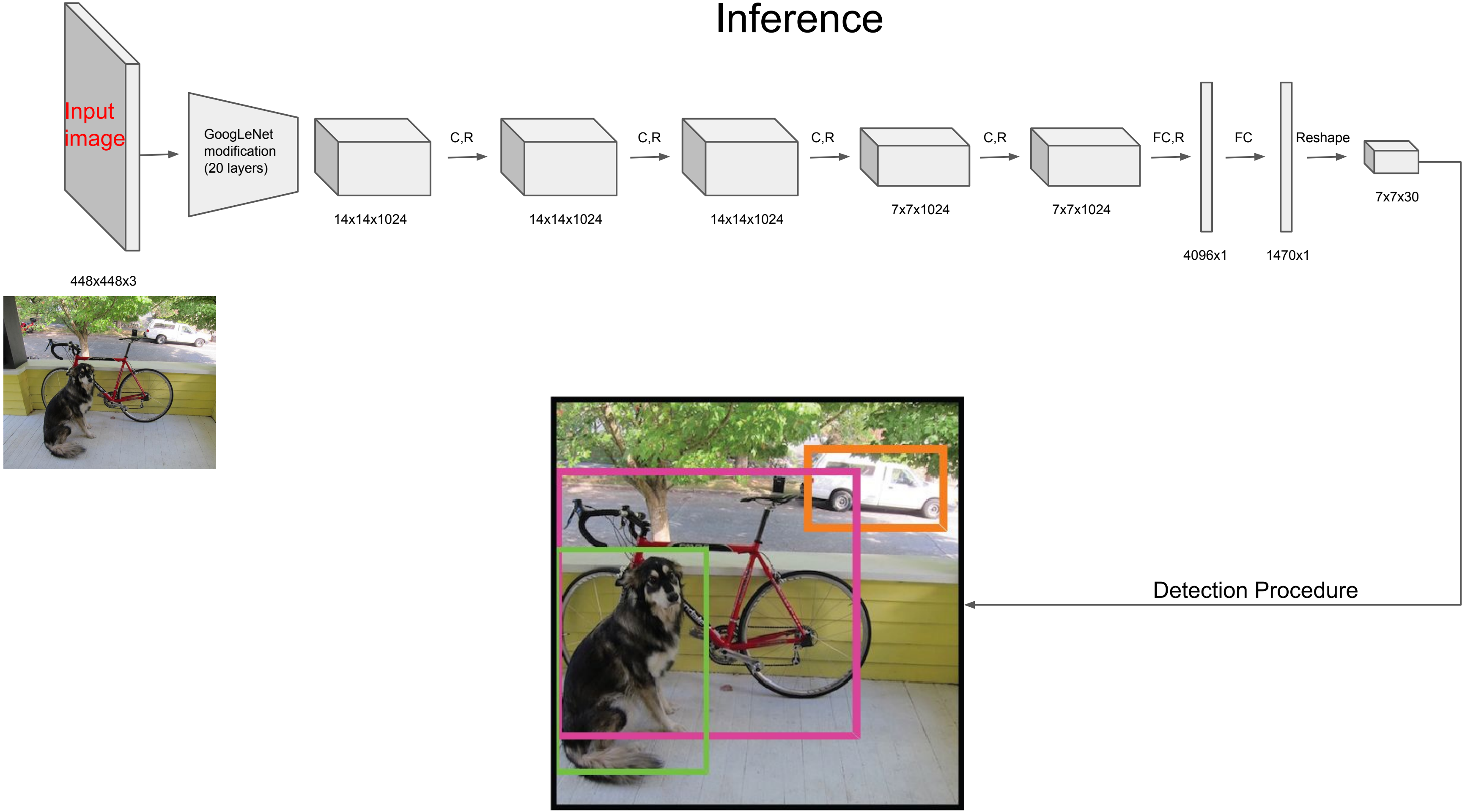
Inference


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

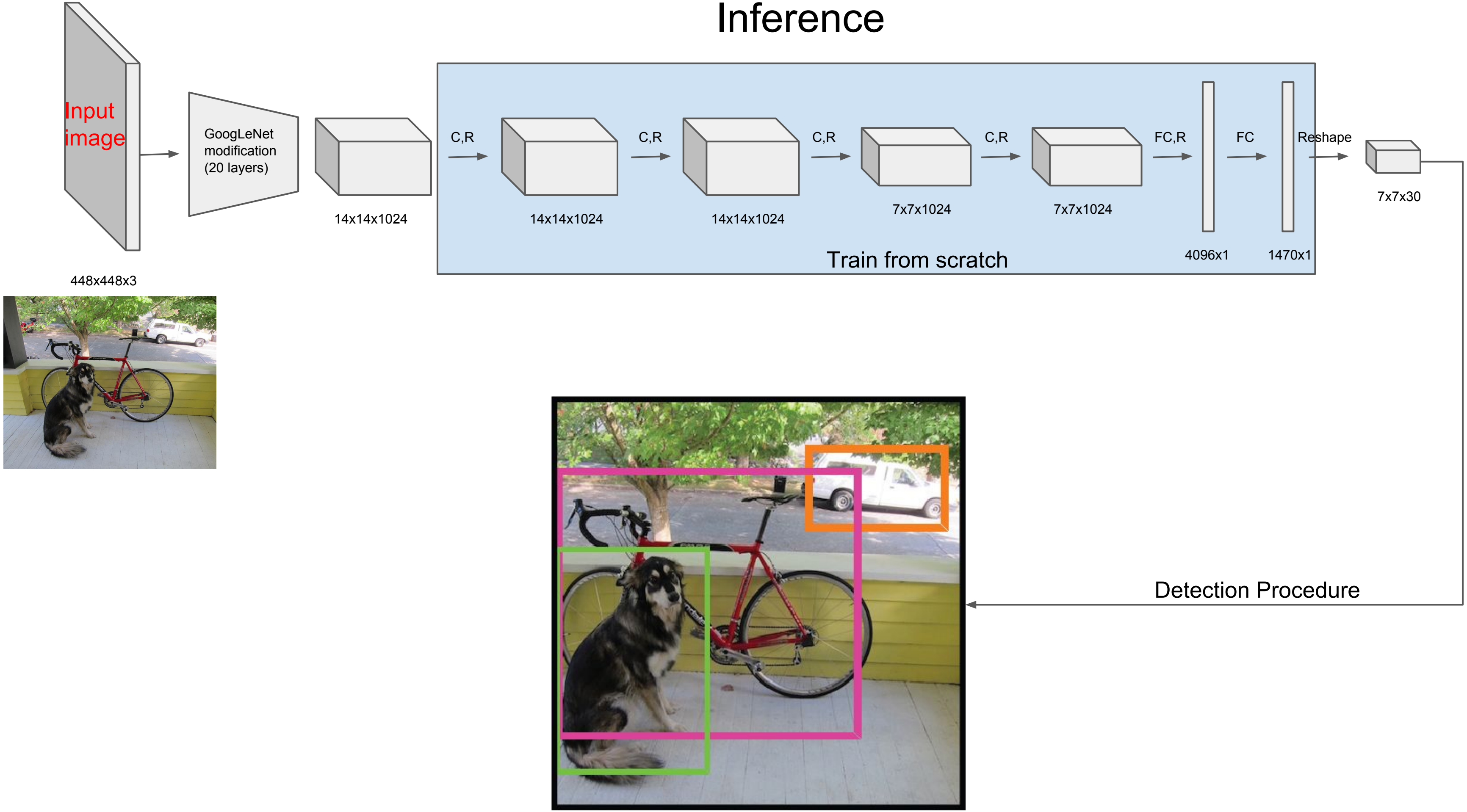
Inference


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

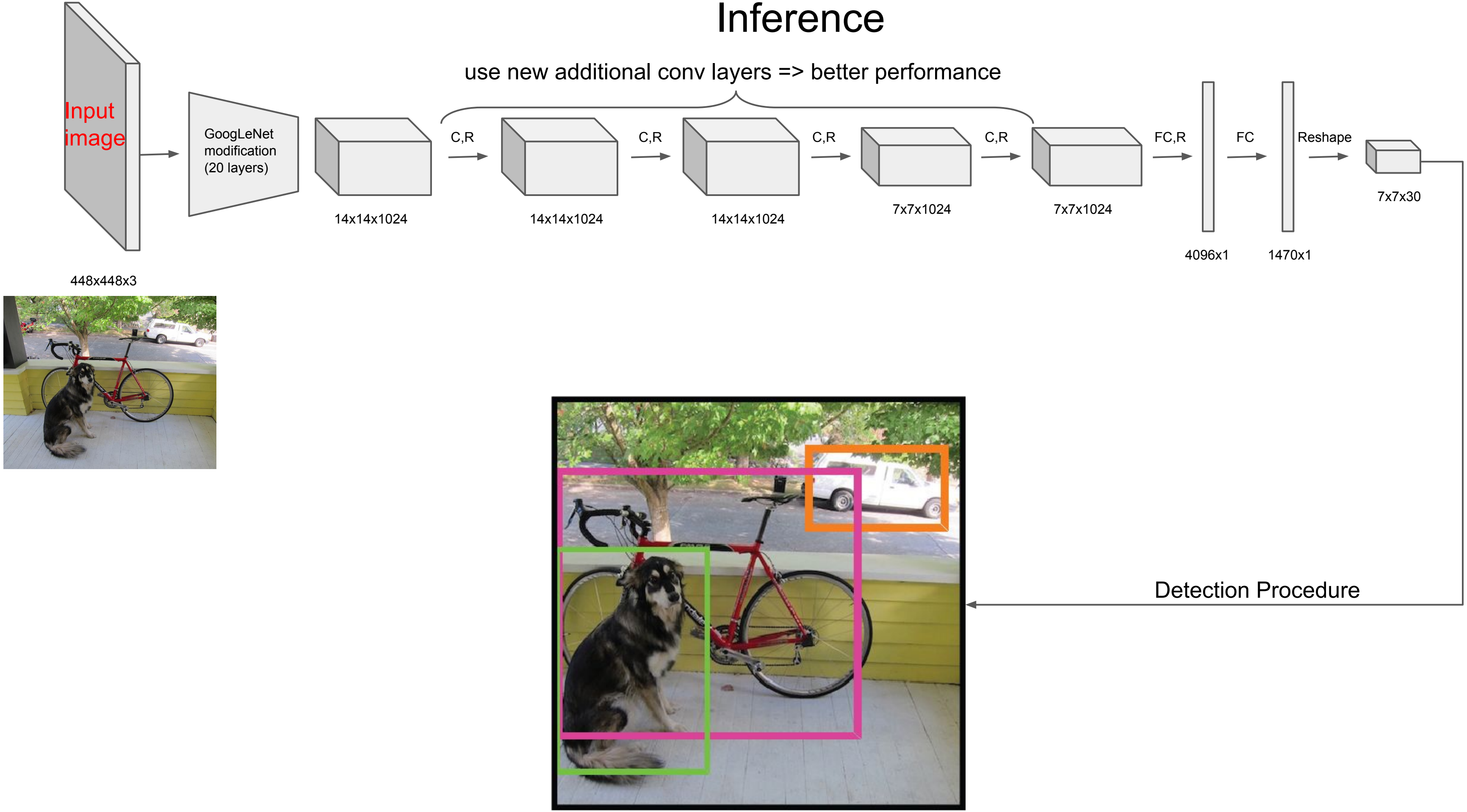
Inference


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

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



use new additional conv layers => better performance

Input image

448x448x3

GoogLeNet modification (20 layers)

14x14x1024

C,R

14x14x1024

C,R

14x14x1024

C,R

7x7x1024

C,R

7x7x1024

FC,R

4096x1

FC

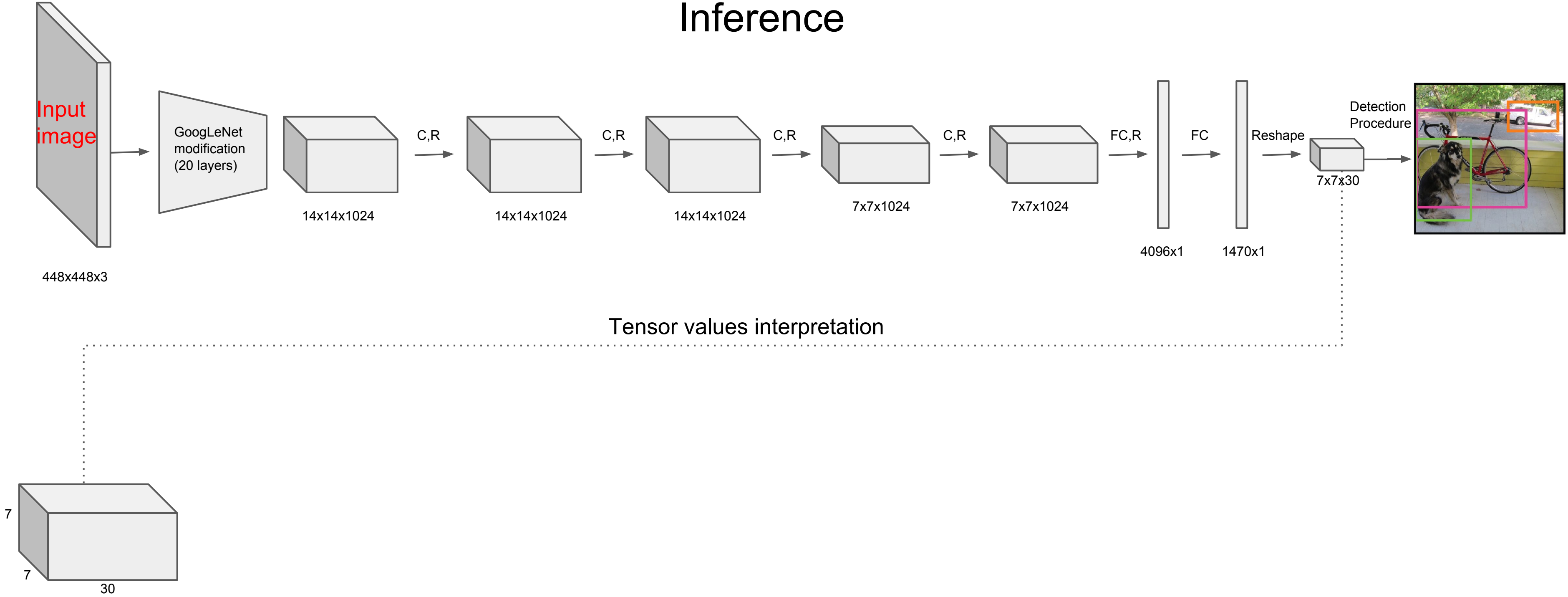
1470x1

Reshape

7x7x30

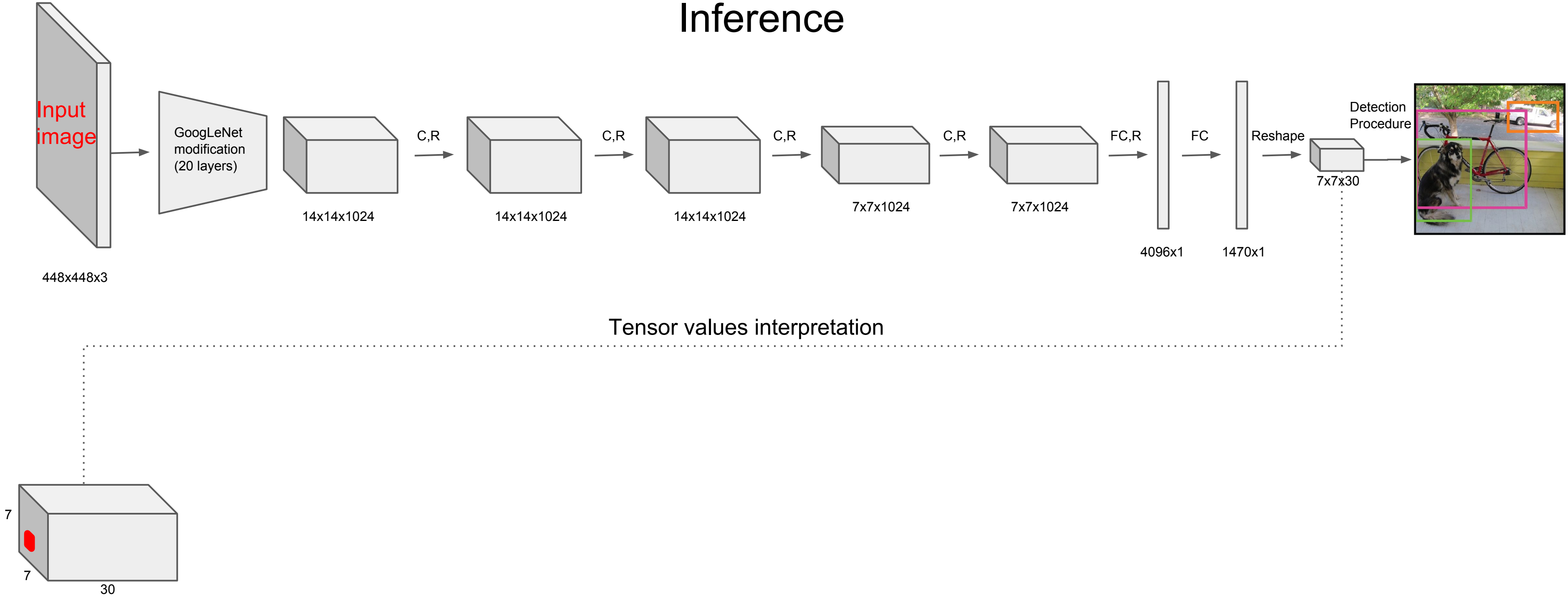
Detection Procedure

Inference

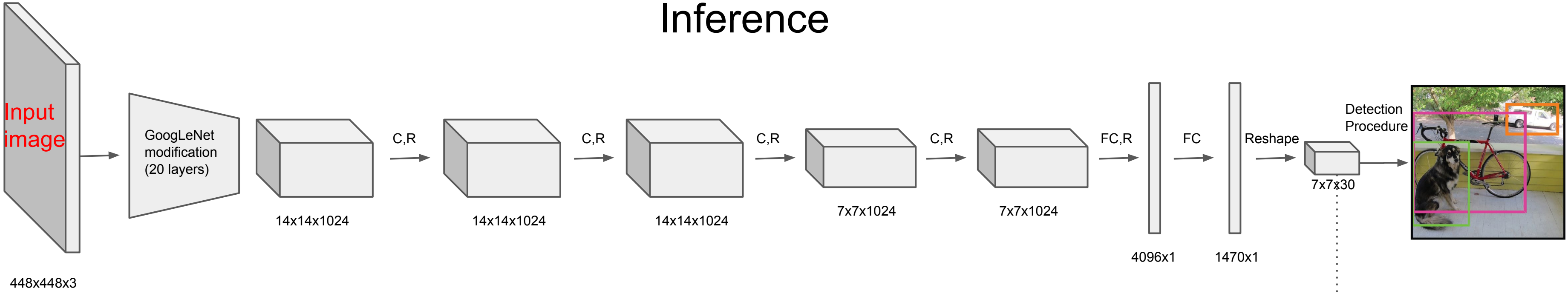


Tensor values interpretation

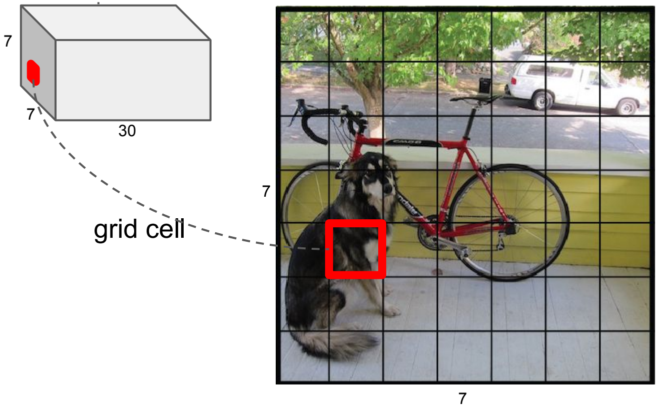
Inference


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
7
7
30

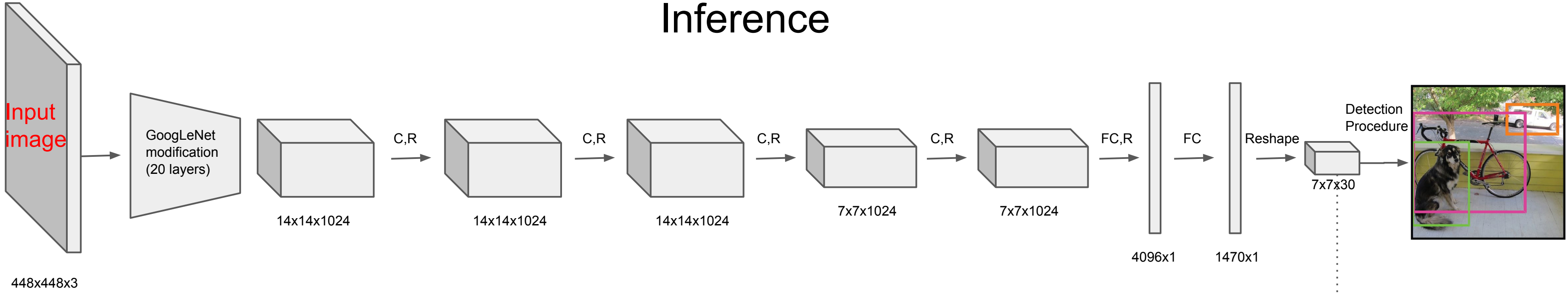
Inference



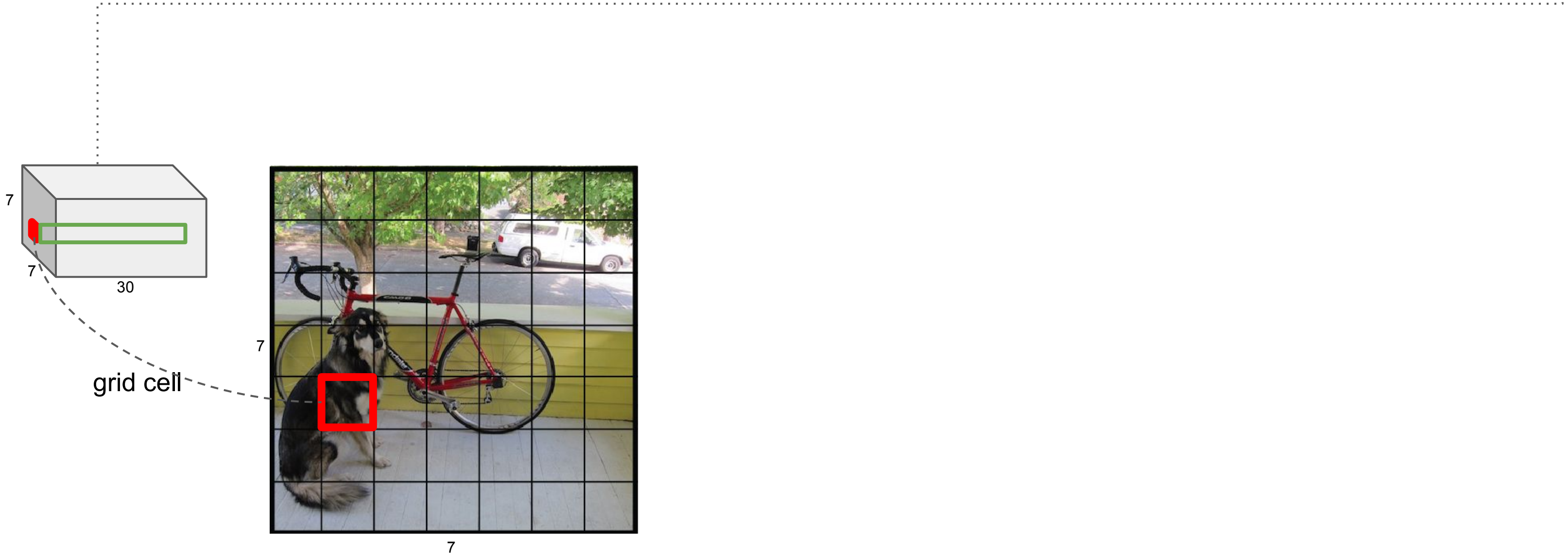
Tensor values interpretation



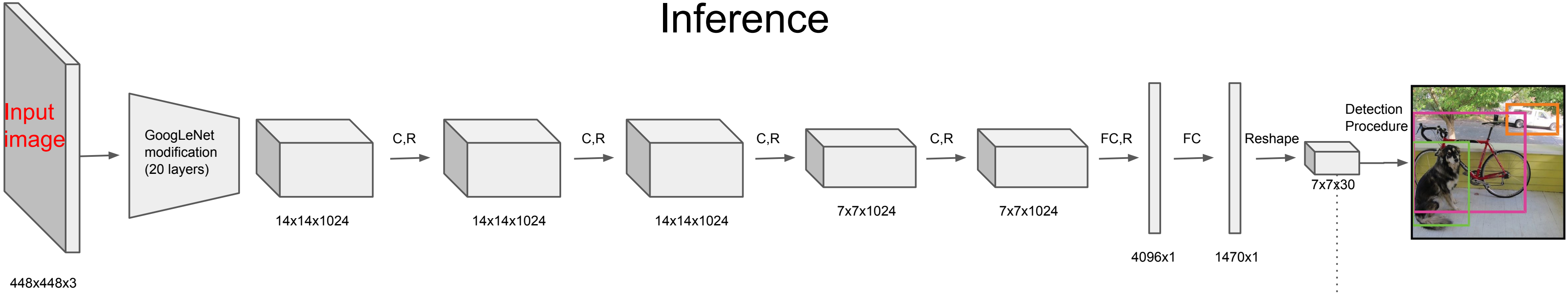
Inference



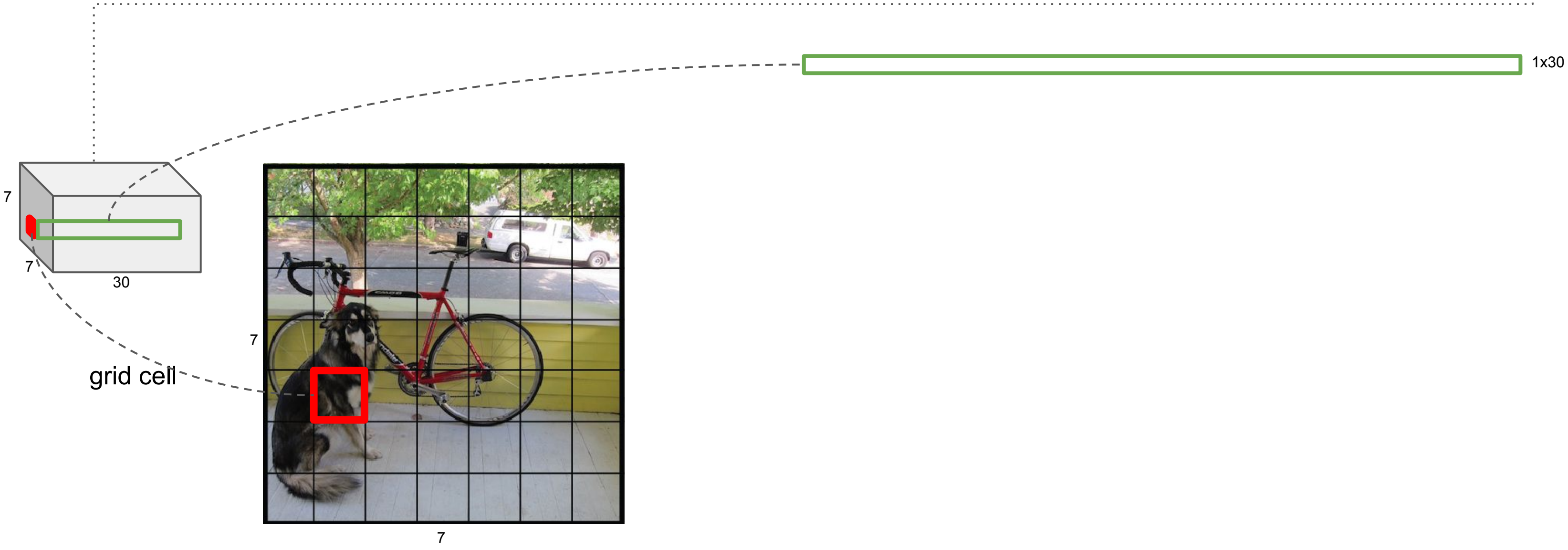
Tensor values interpretation



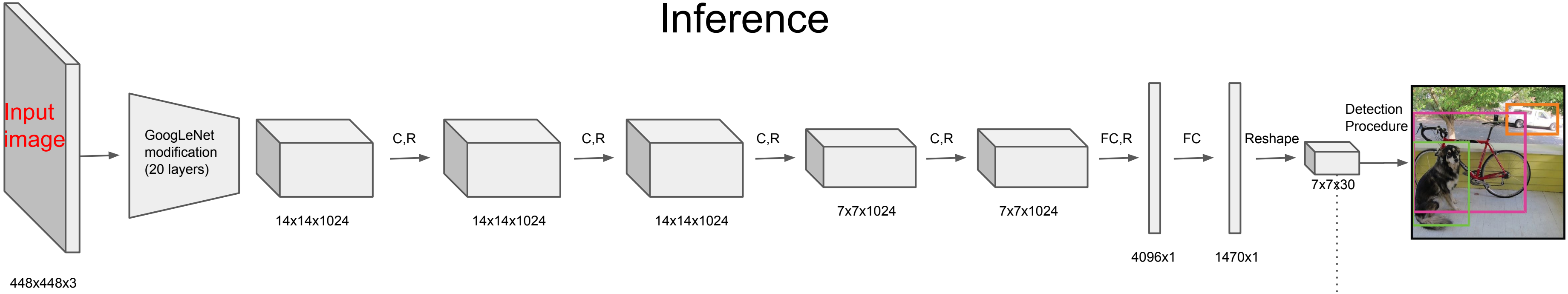
Inference



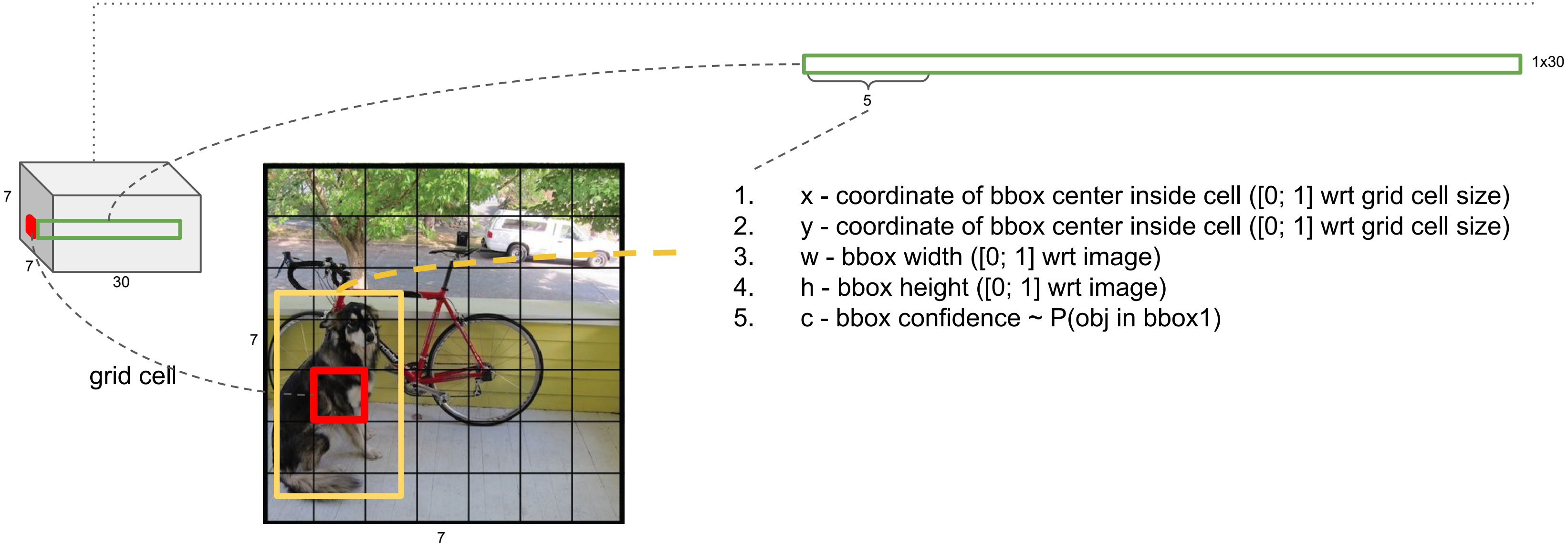
Tensor values interpretation



Inference

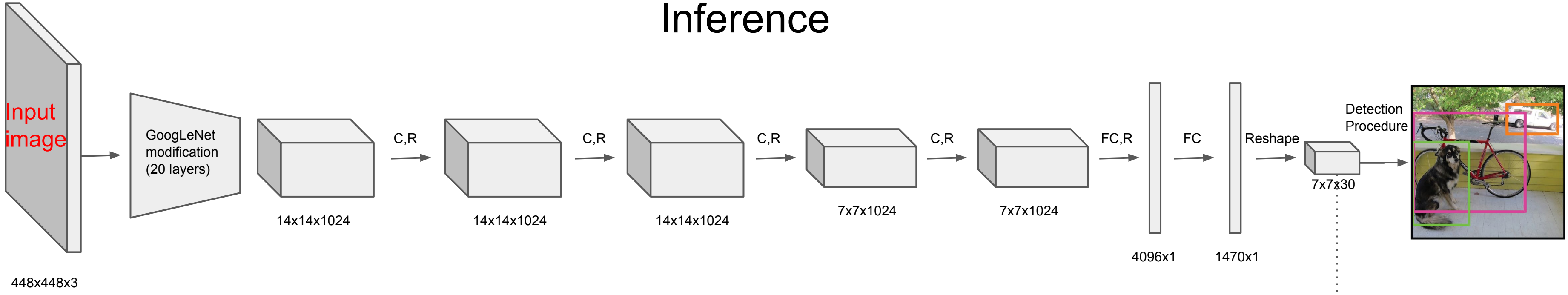


Tensor values interpretation

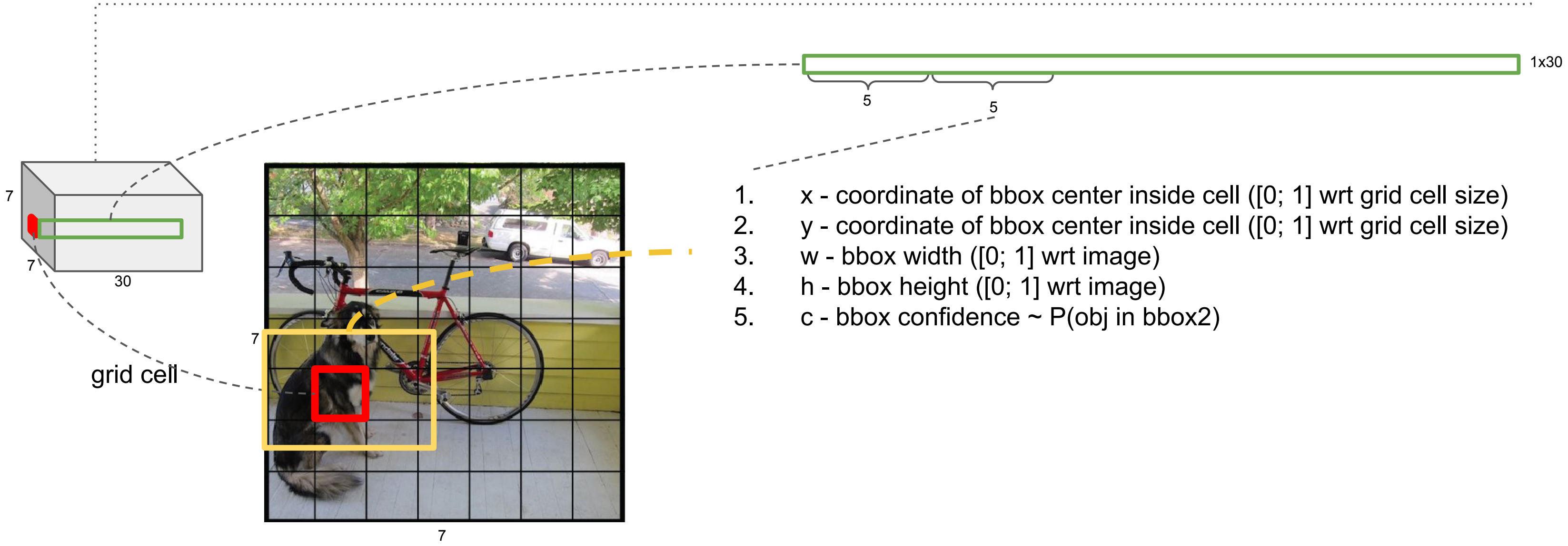


1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox1)

Inference

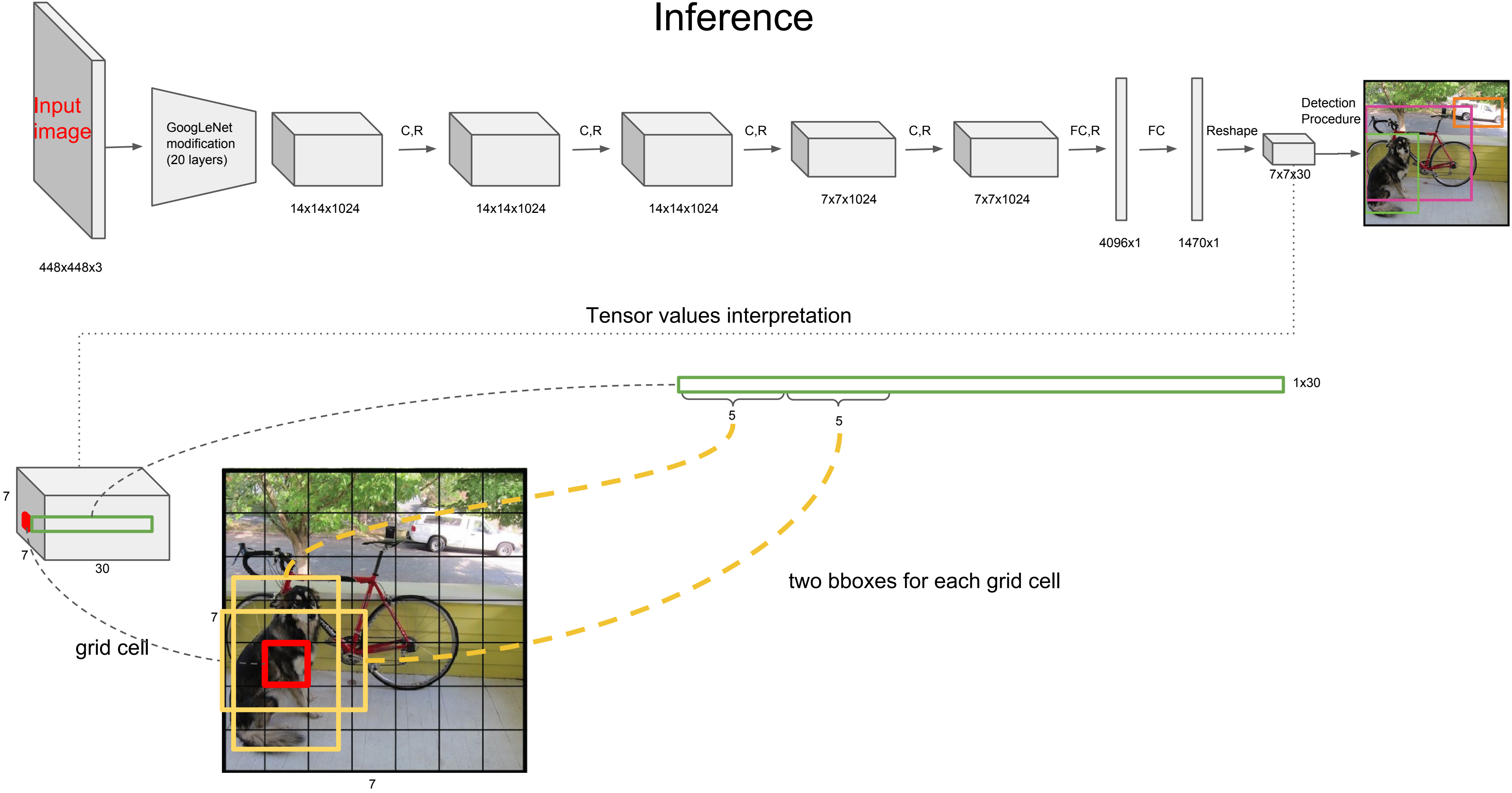


Tensor values interpretation



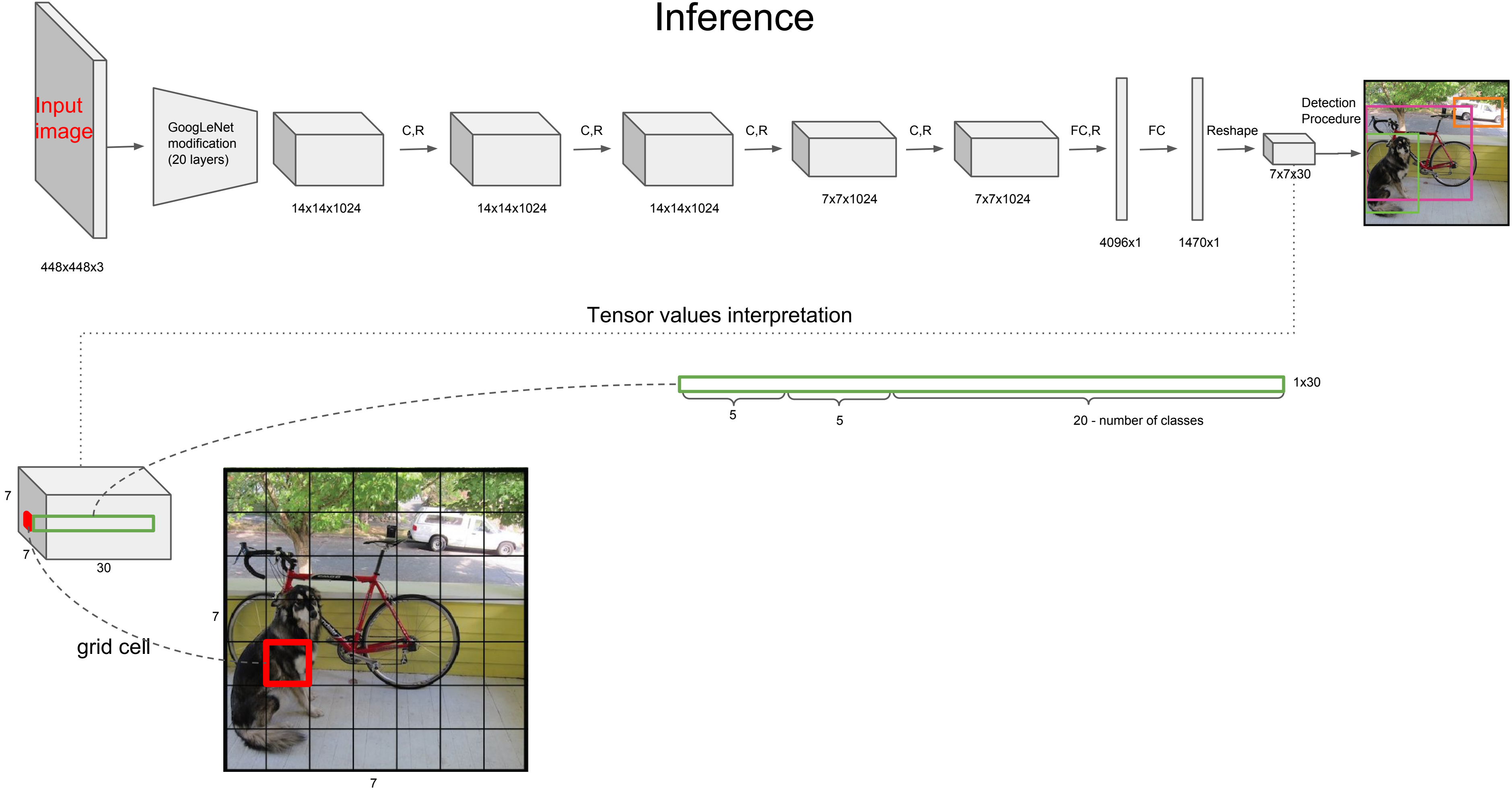
1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox2)

Inference

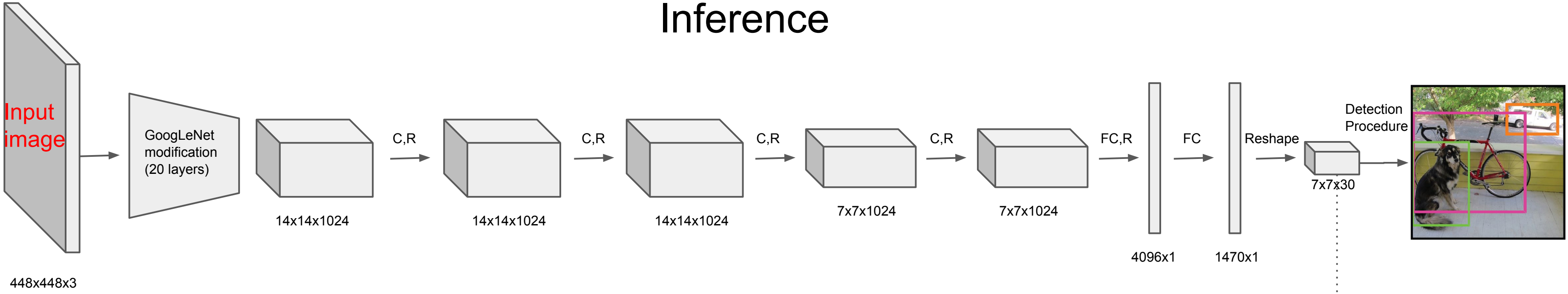


Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
1x30
5
5
7
7
30
grid cell
7
7
two bboxes for each grid cell

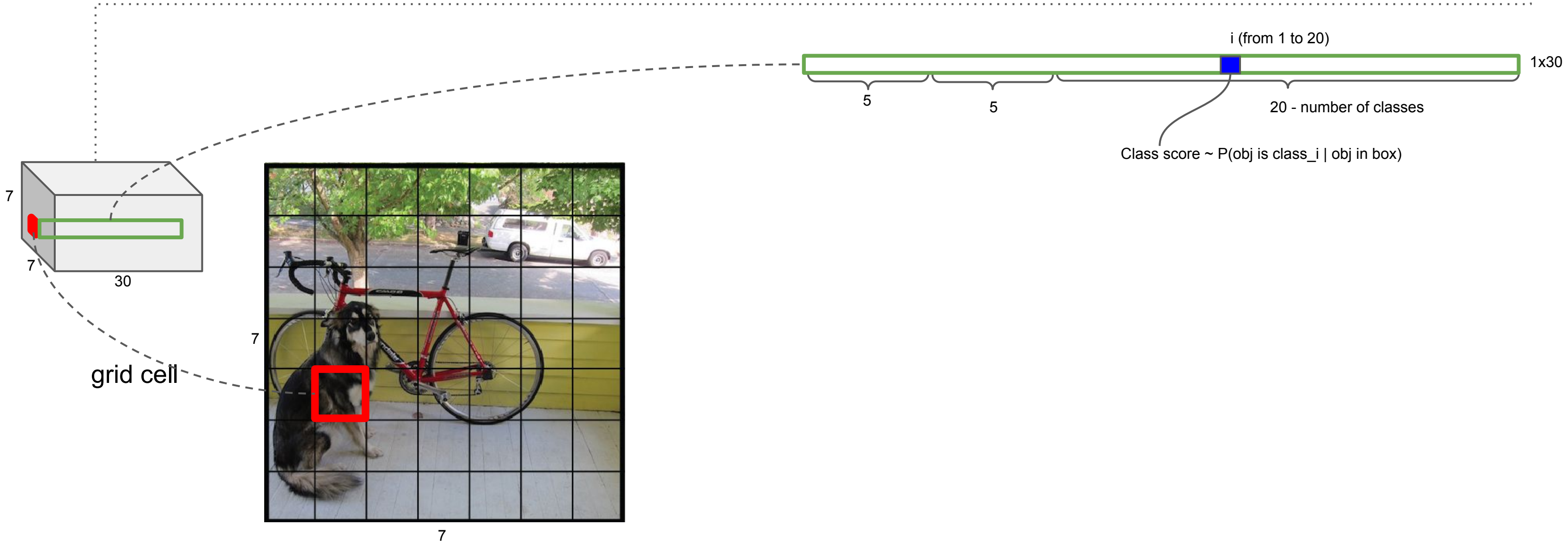
Inference


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

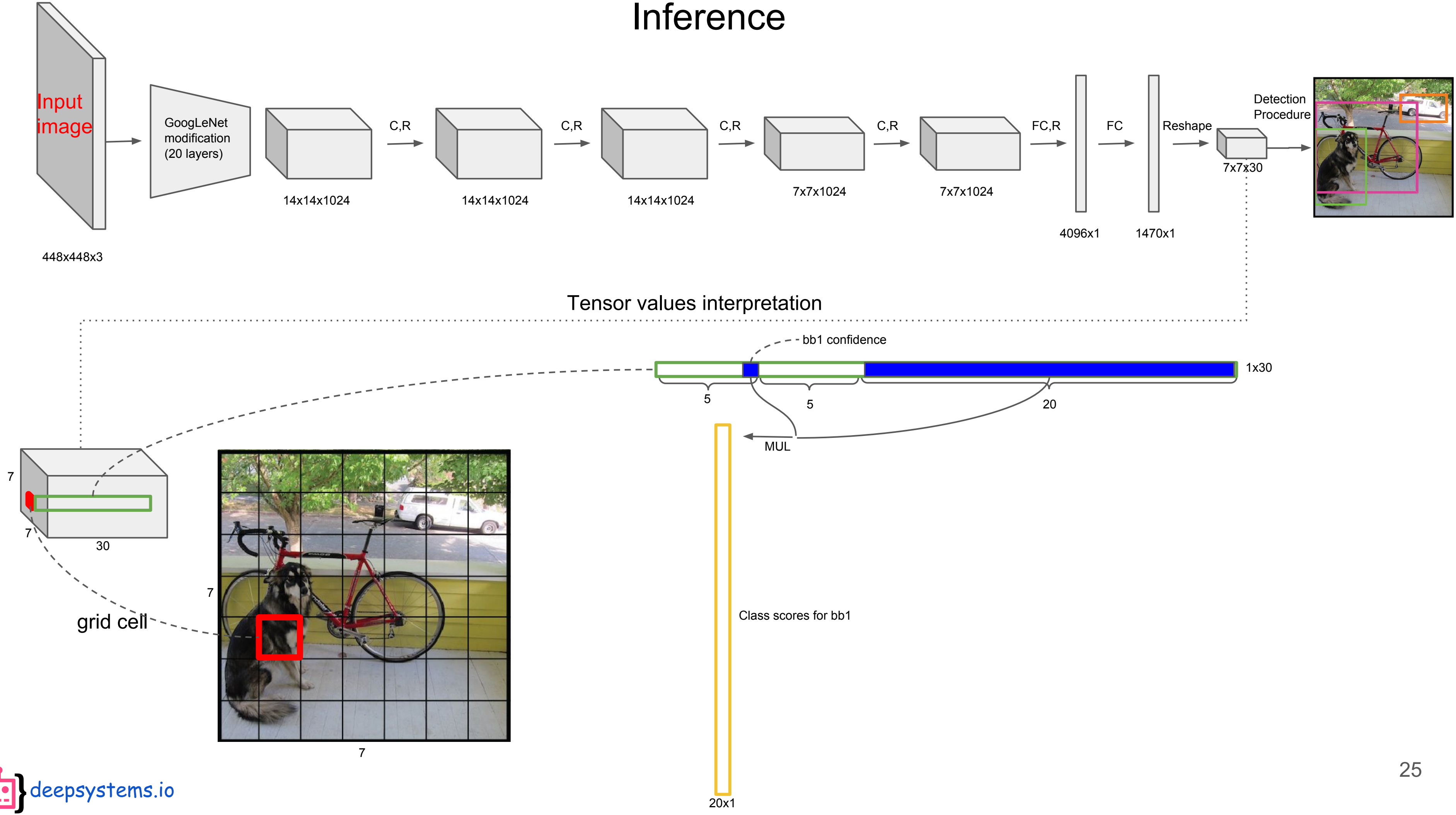
Inference



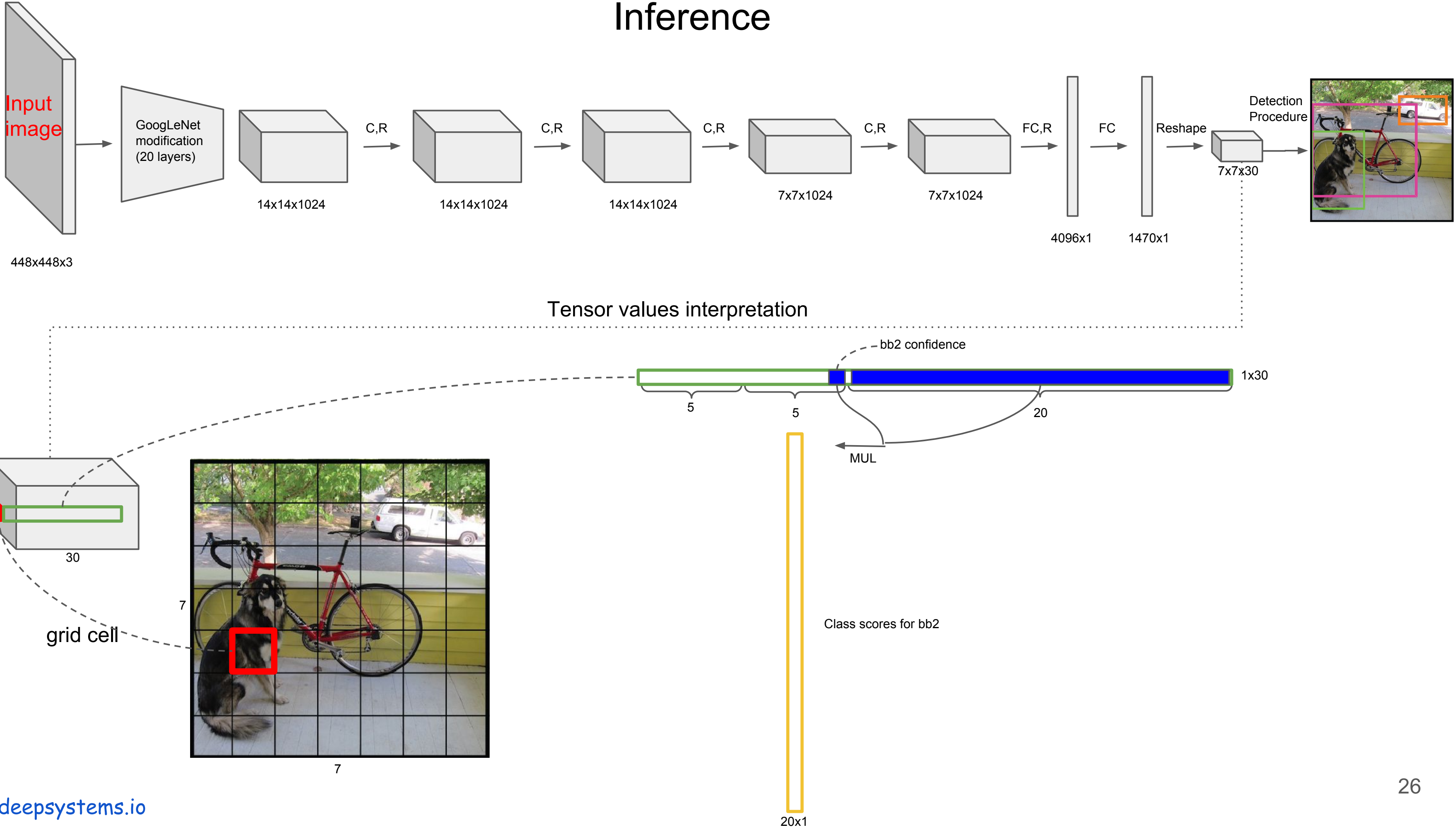
Tensor values interpretation



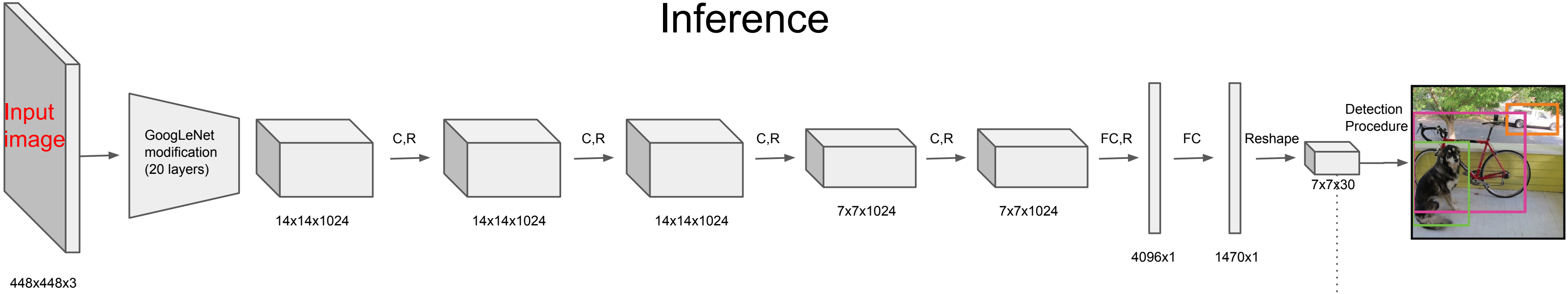
Inference


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
bb1 confidence
1x30
5
5
20
MUL
Class scores for bb1
20x1
7
7
30
grid cell
7
7

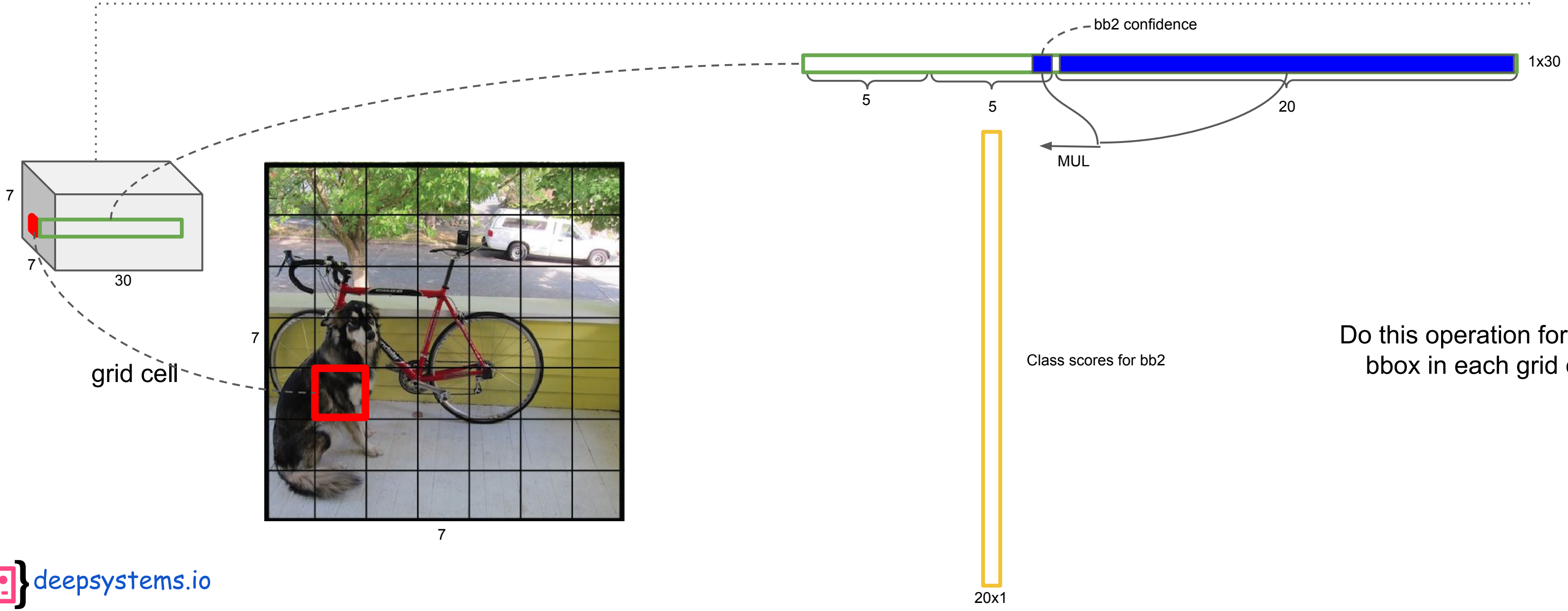
Inference


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

Inference

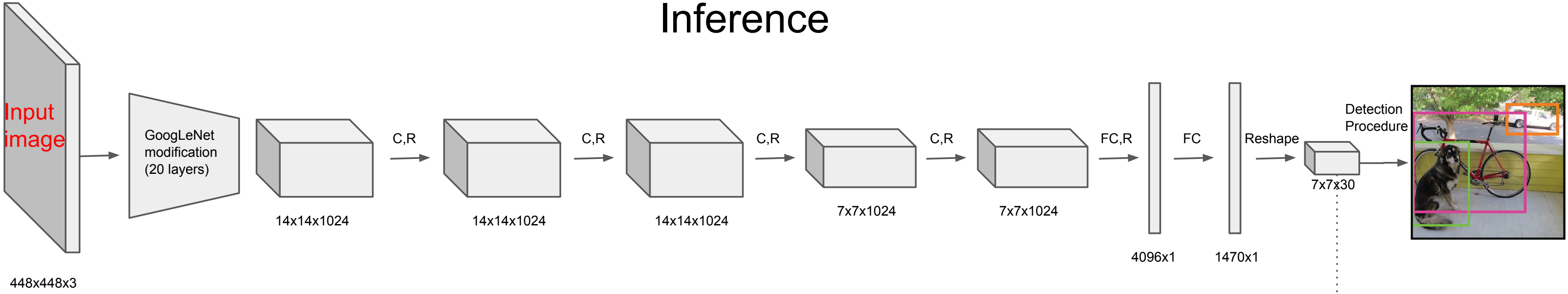


Tensor values interpretation

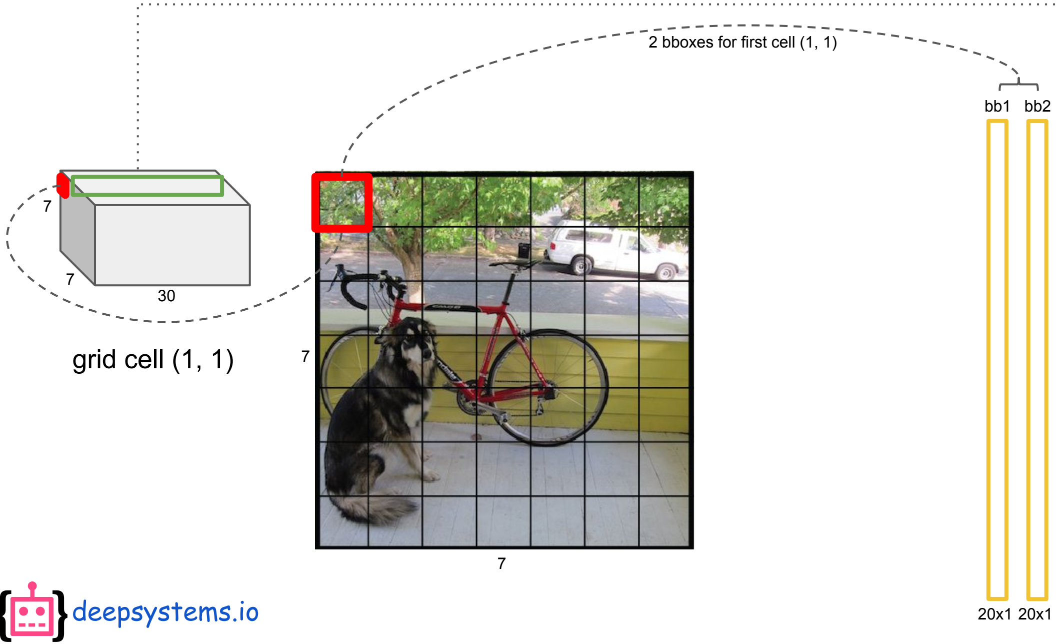


Do this operation for each bbox in each grid cell

Inference

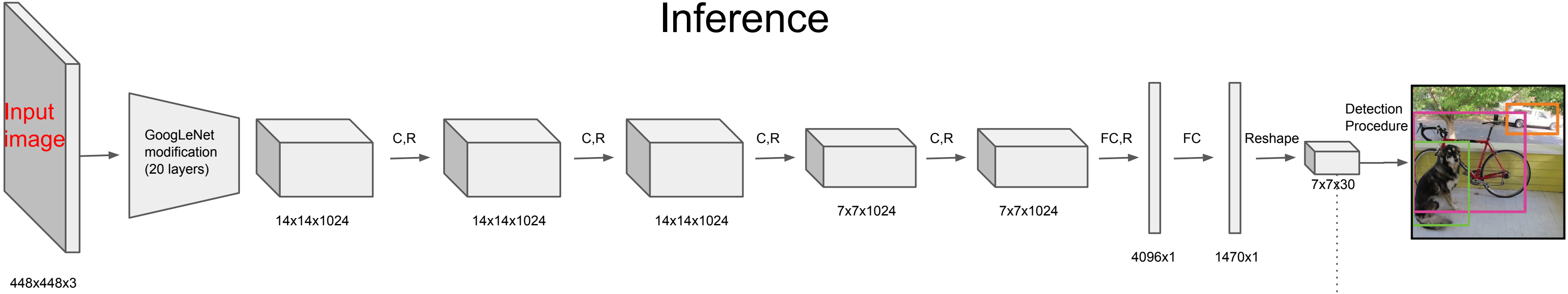


Tensor values interpretation

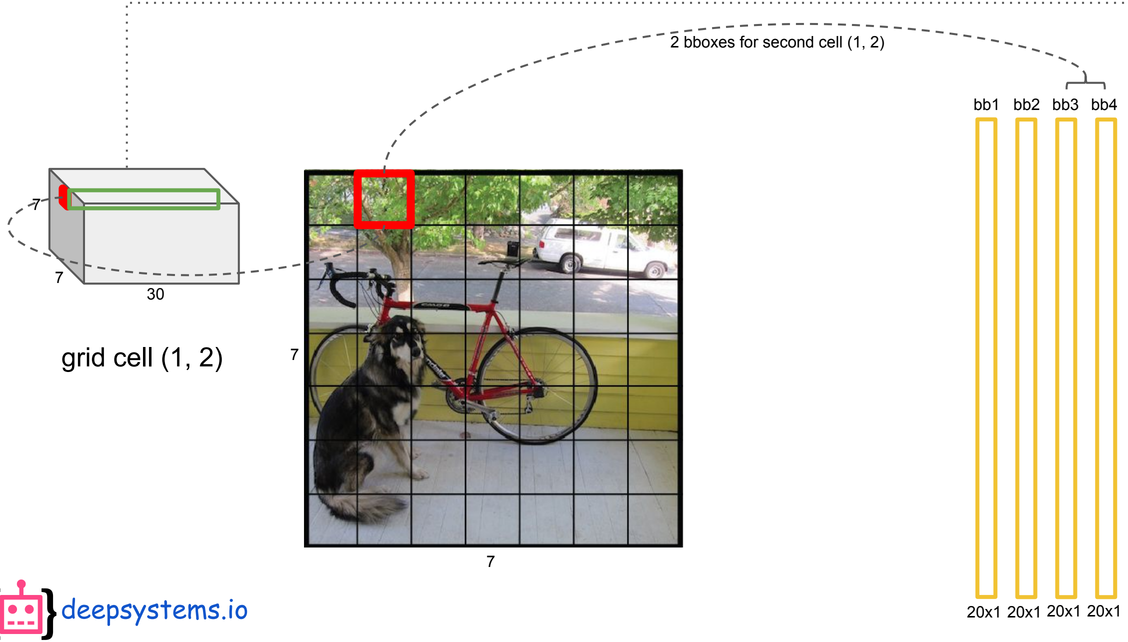


grid cell (1, 1)

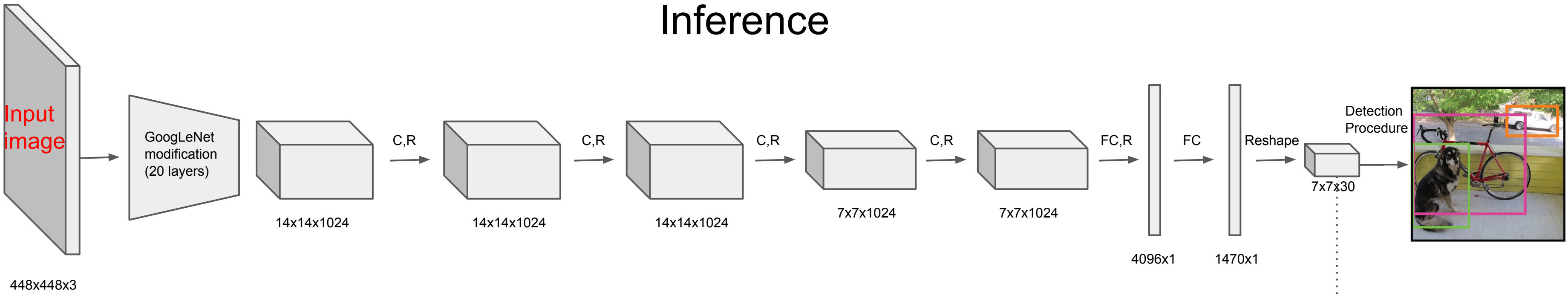
Inference



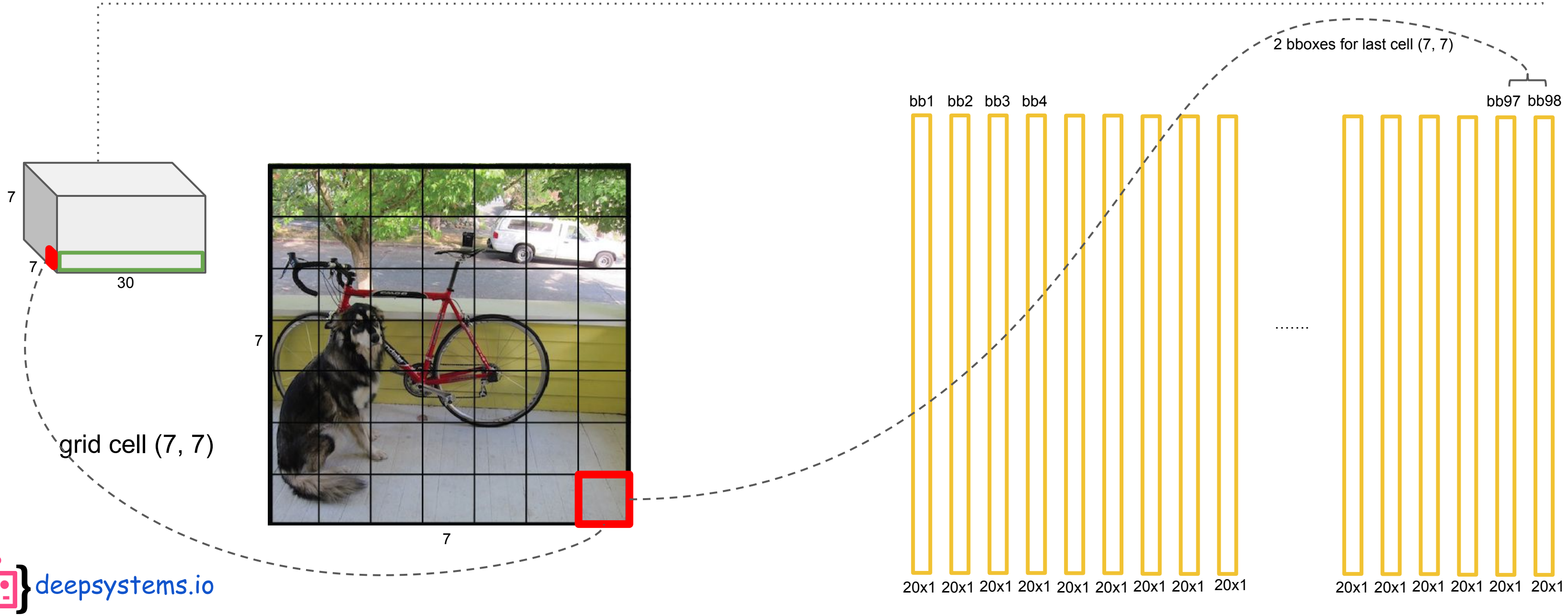
Tensor values interpretation



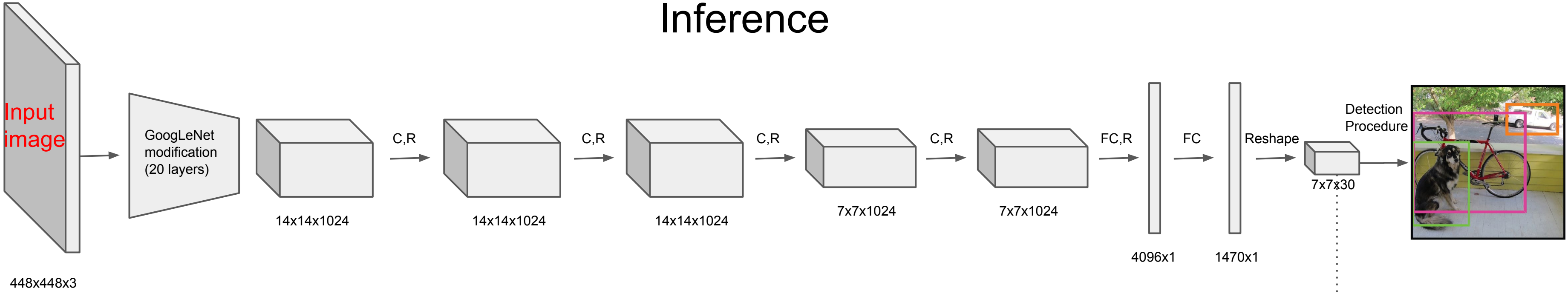
Inference



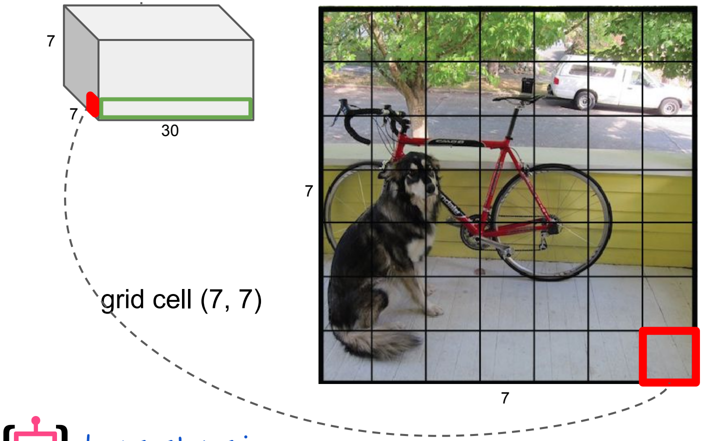
Tensor values interpretation



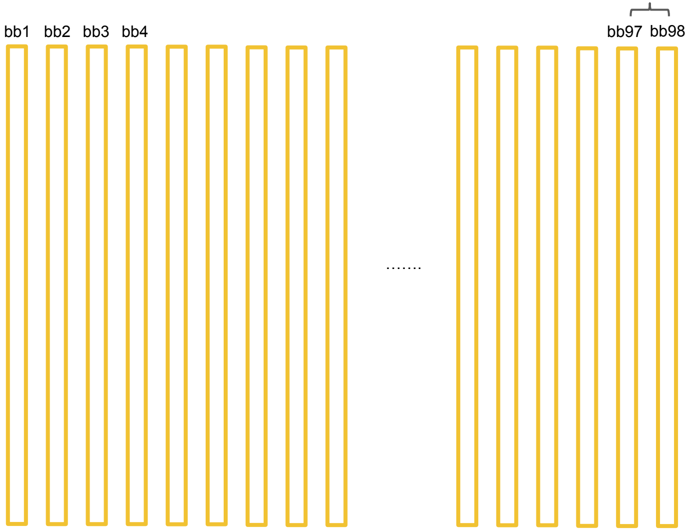
Inference



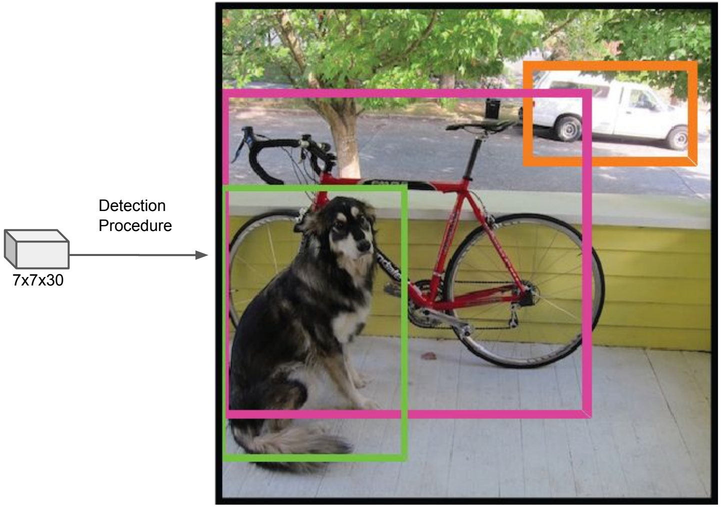
Tensor values interpretation



Total 7*7*2 = 98 bboxes

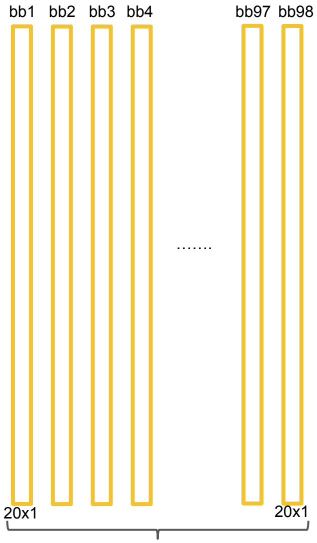


Look at detection procedure



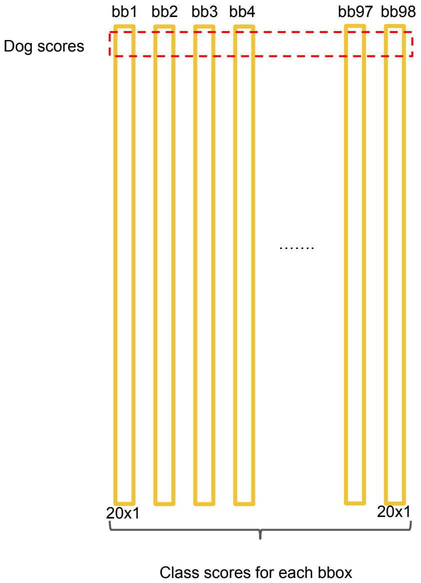
Detection Procedure

7x7x30

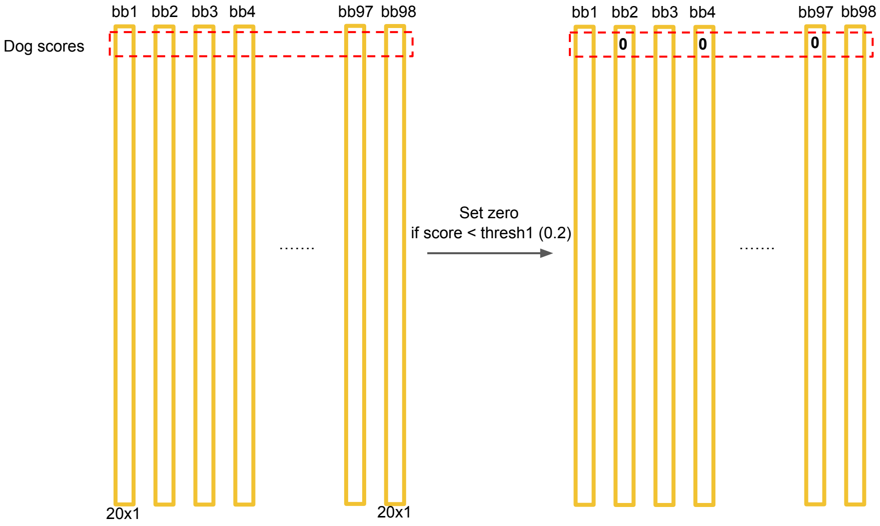


Class scores for each bbox

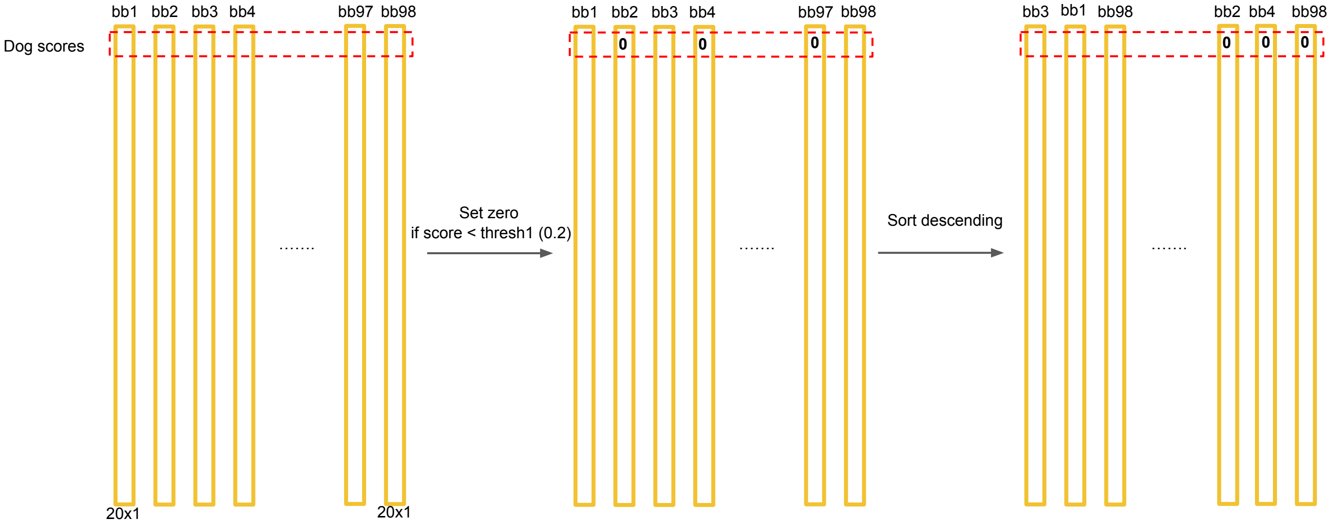
deepsystems.io

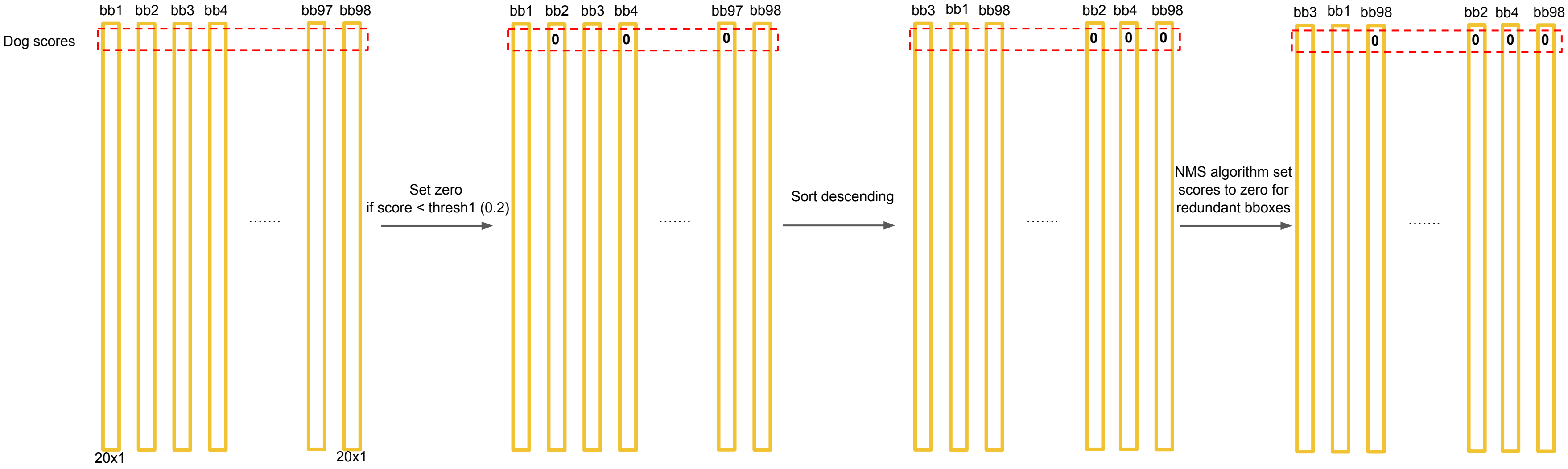


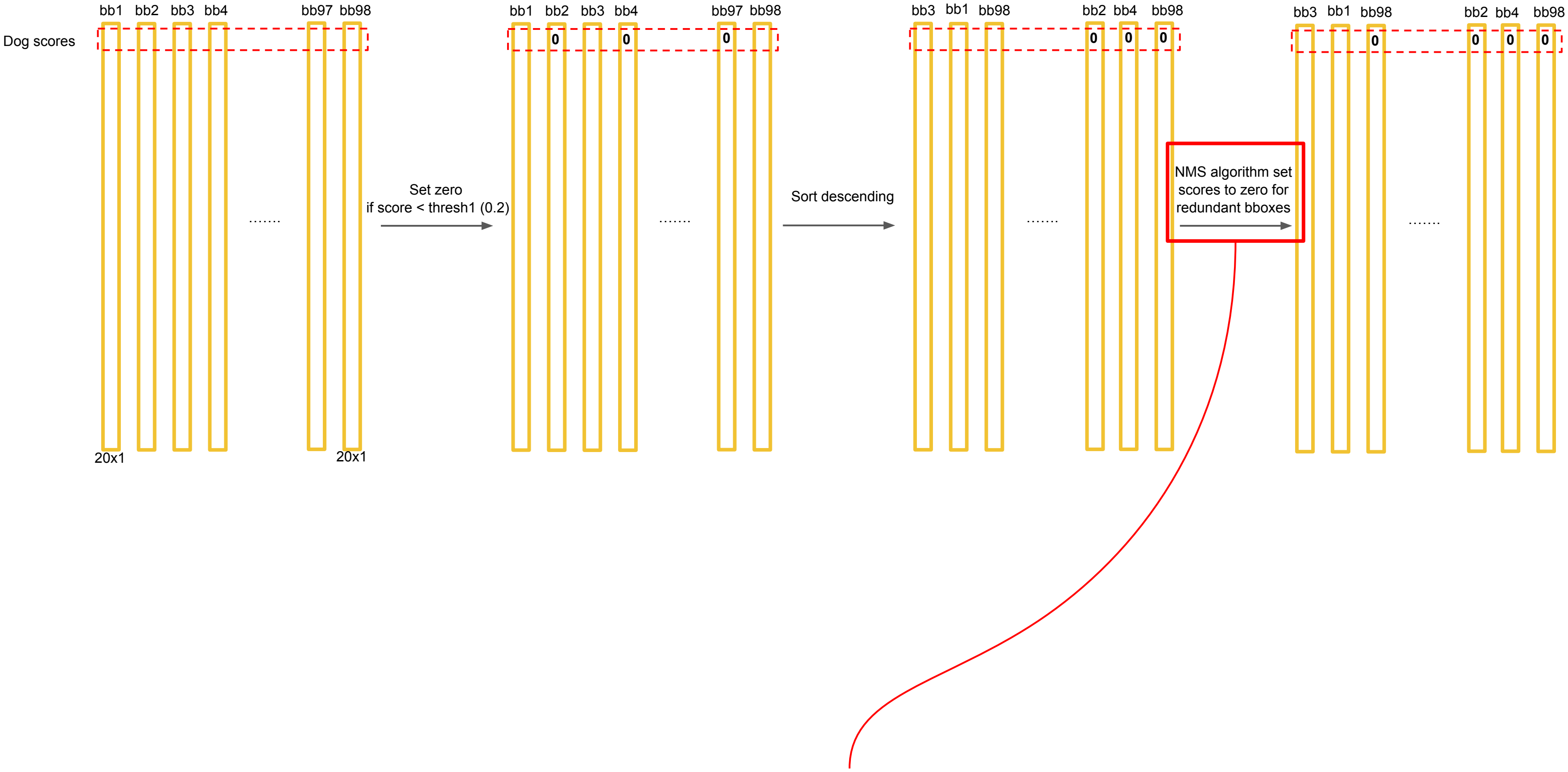
Get first class scores for each bbox



deepsystems.io


bb1
bb2
bb3
bb4
bb97
bb98
Dog 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
....


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

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

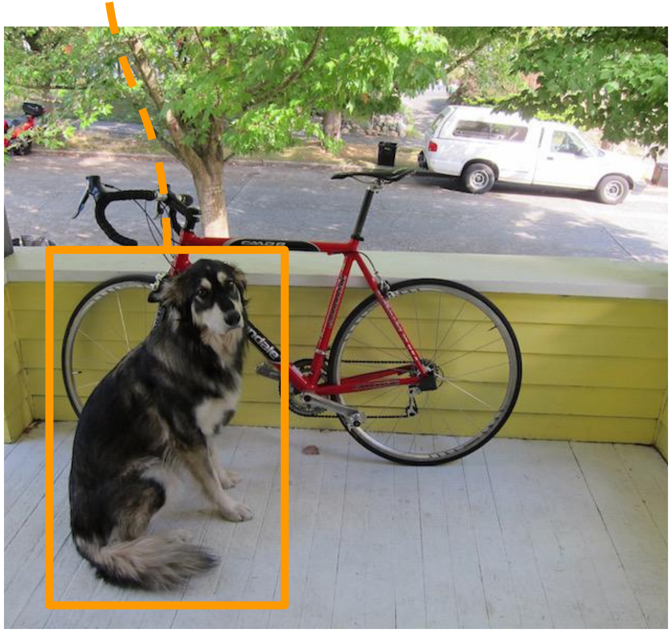
	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98
class: dog	0.5	0.3	0.2	0.1							0	0	0	0

1x98

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

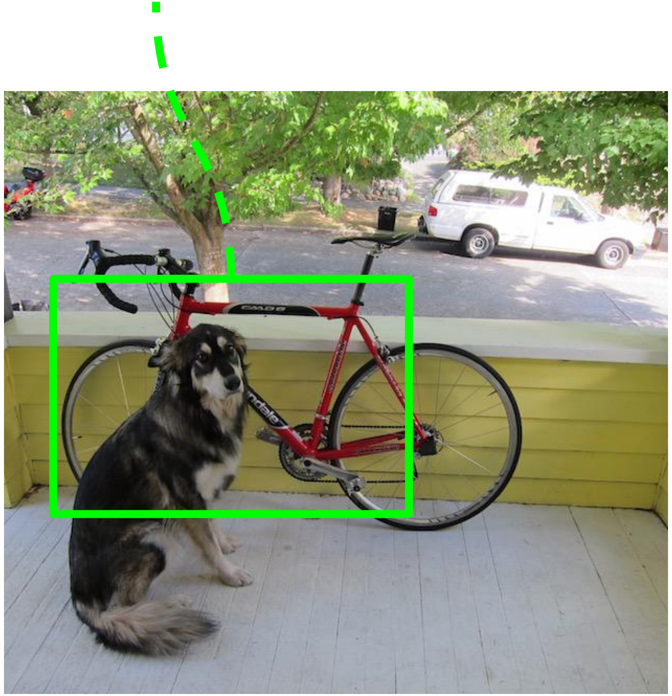
	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0.3	0.2	0.1							0	0	0	0	1x98



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

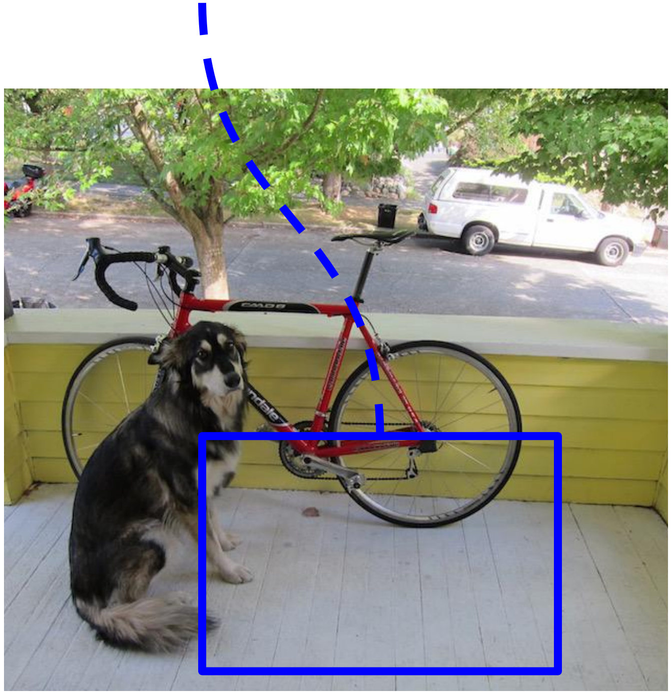


Non-Maximum Suppression: intuition

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

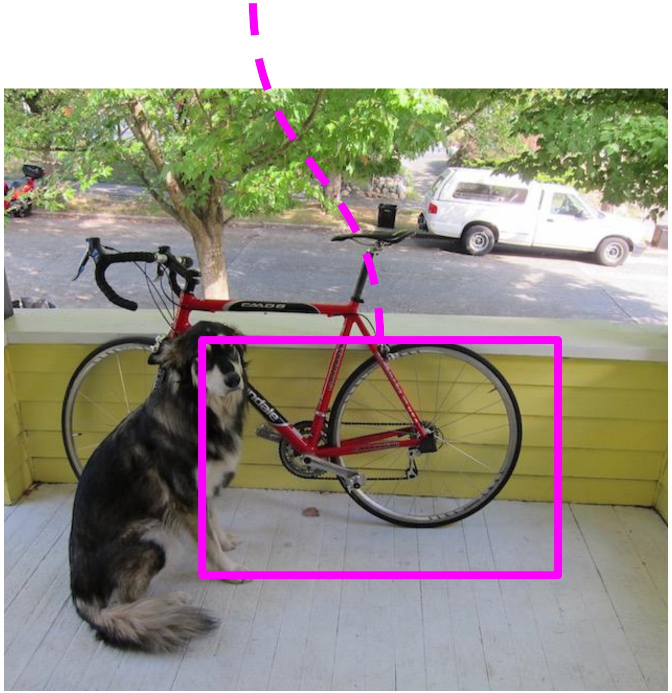
1x98



Non-Maximum Suppression: intuition

class (dog) scores for each bbox

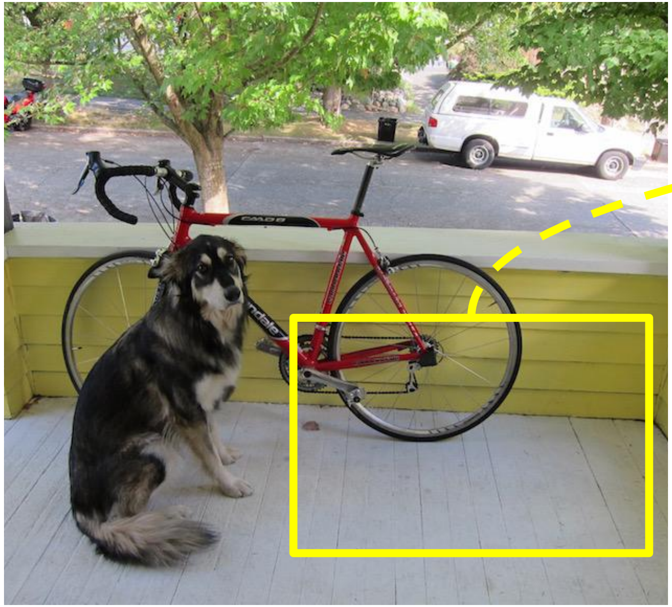
	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0.3	0.2	0.1							0	0	0	0	1x98



Non-Maximum Suppression: intuition

class (dog) scores for each bbox

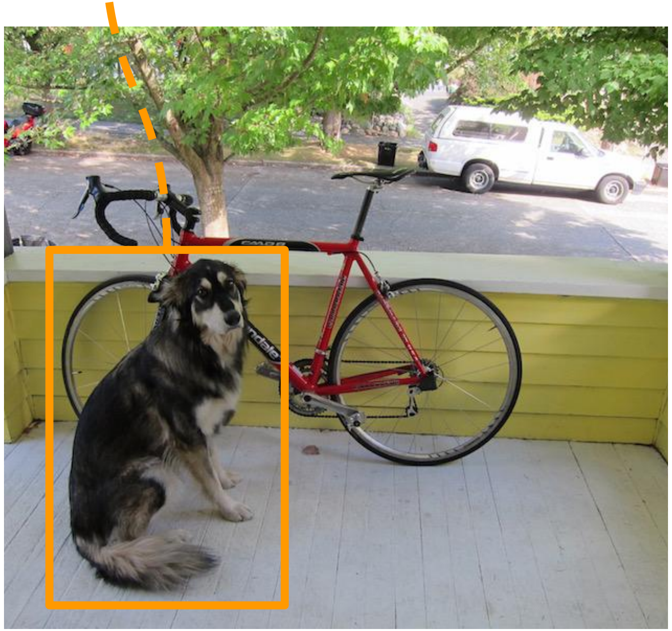
	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0.3	0.2	0.1							0	0	0	0	1x98



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

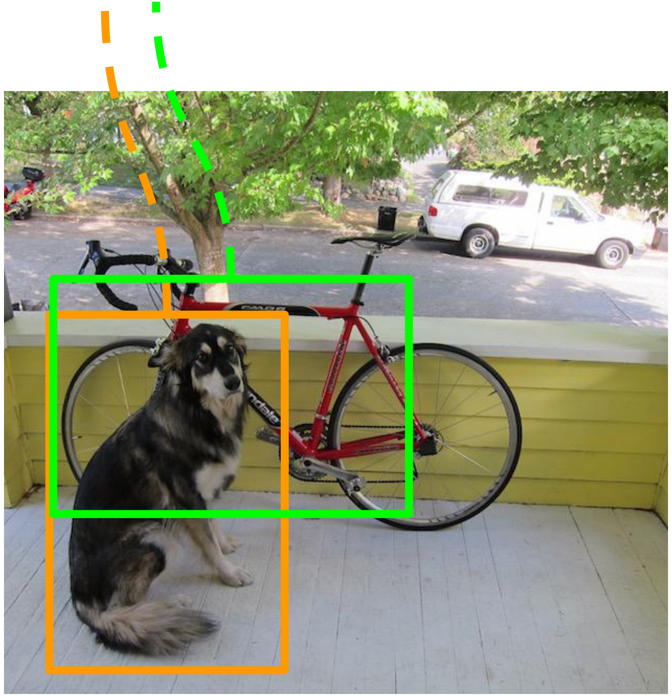


Get bbox with max score. Let’s denote it “bbox_max”

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

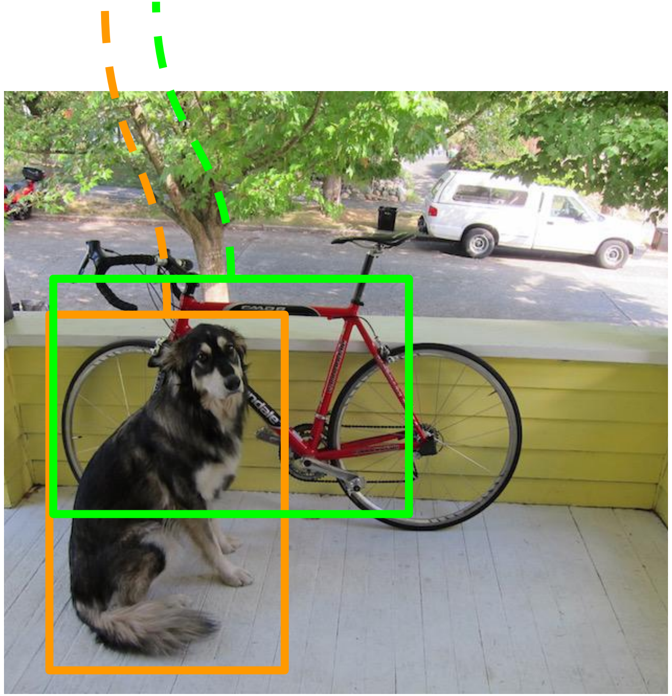


Compare “bbox_max” with others less score (non-zero!) bboxes. Let’s denote it “bbox_cur”

Non-Maximum Suppression: intuition

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

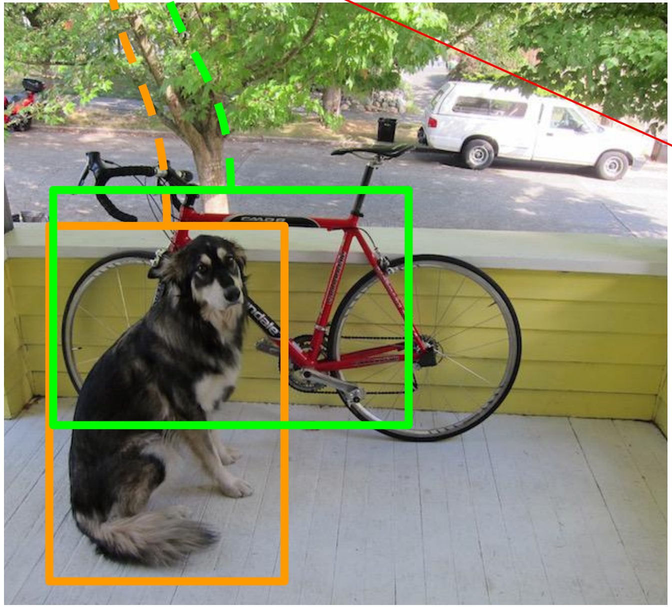


If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

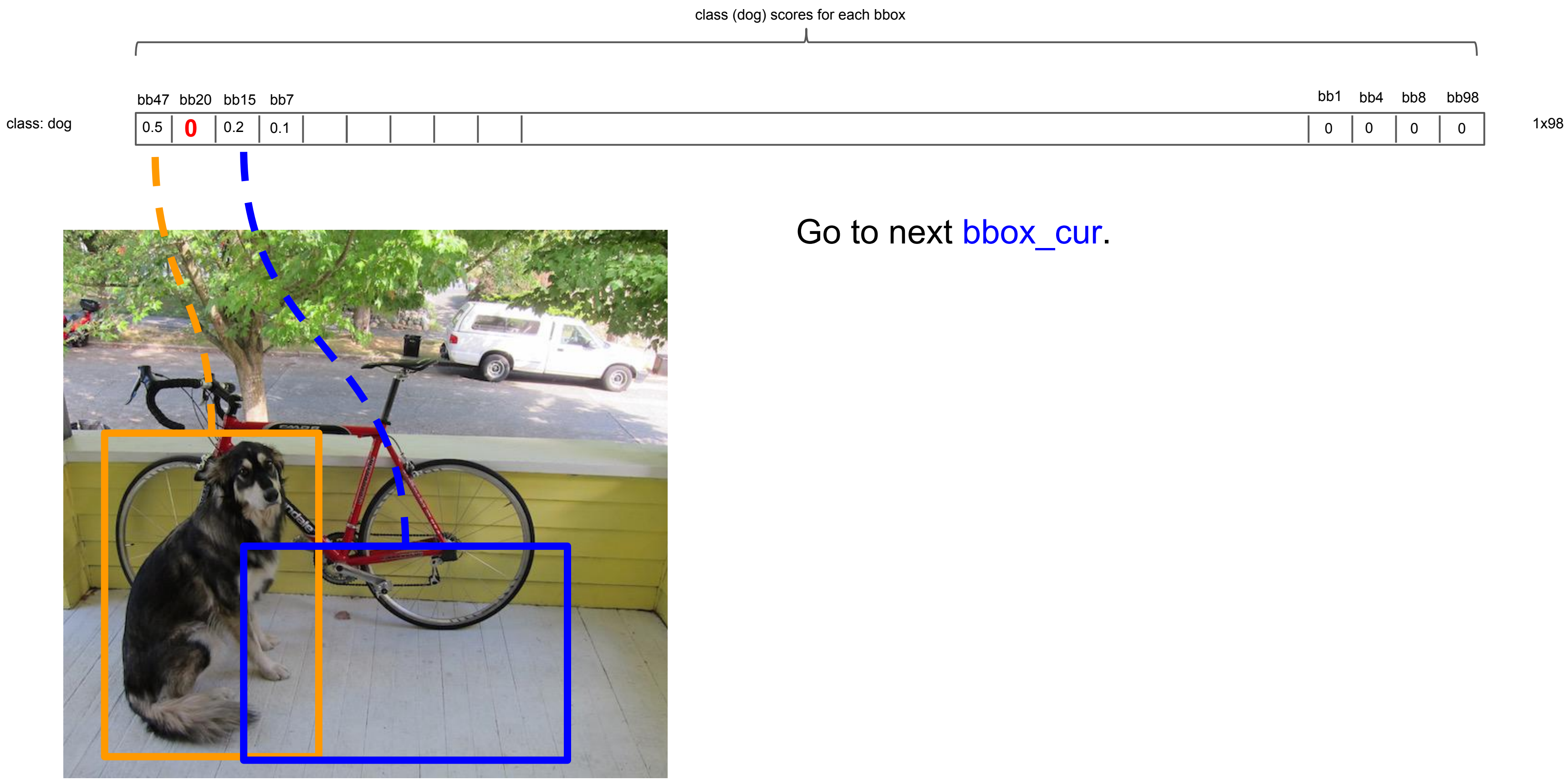
	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	**0**	0.2	0.1							0	0	0	0	1x98



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

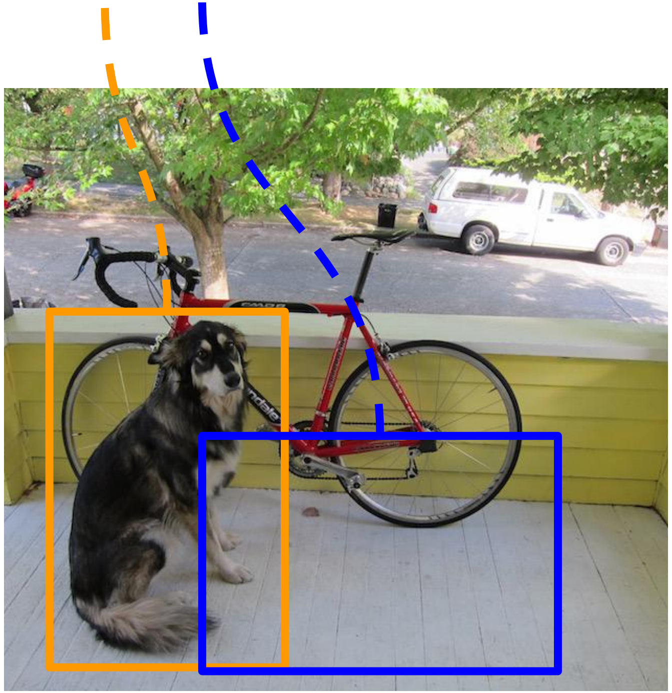
	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0.1							0	0	0	0	1x98

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



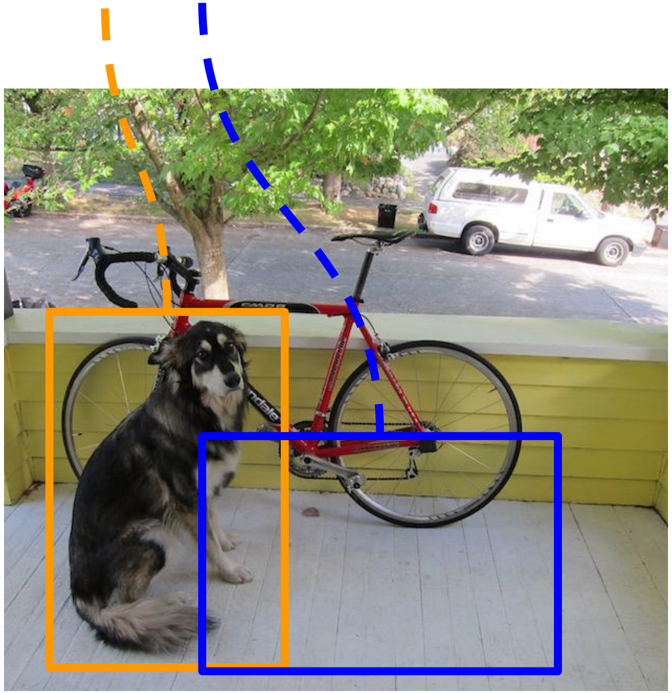
Go to next bbox_cur.

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

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.

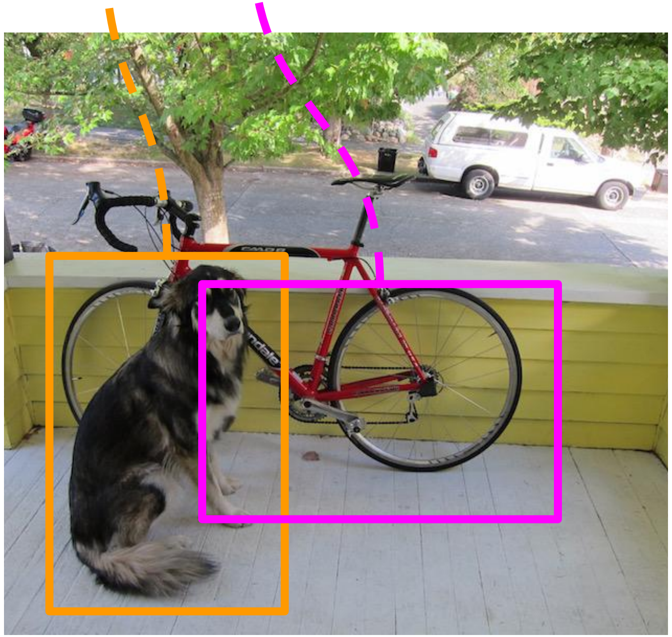
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: continue.

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

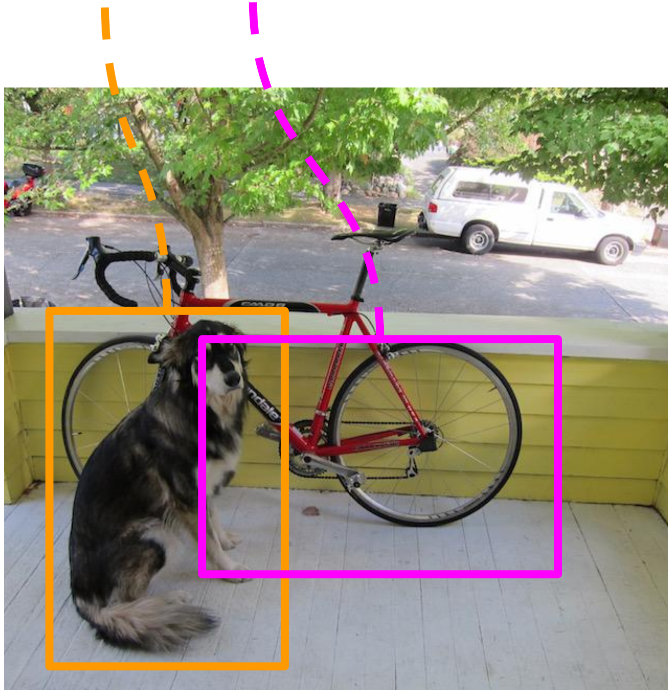


Go to next bbox_cur.

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



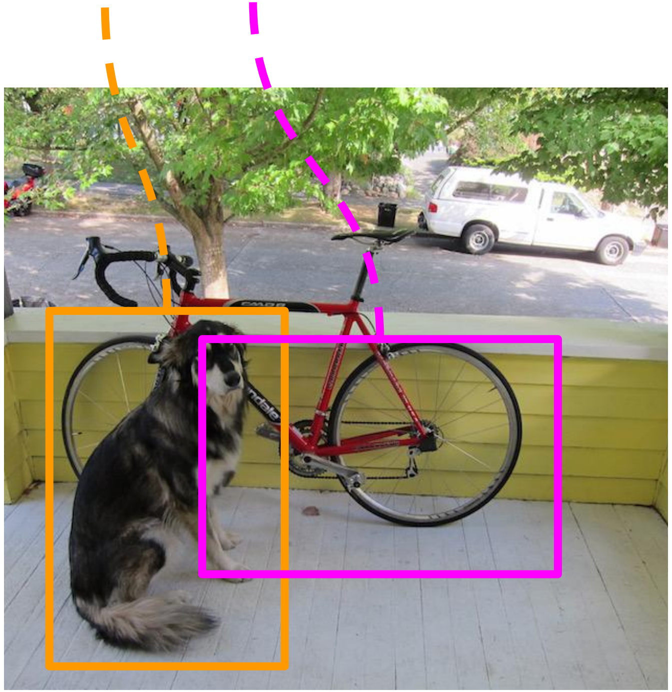
Go to next bbox_cur.

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

Non-Maximum Suppression: intuition

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.

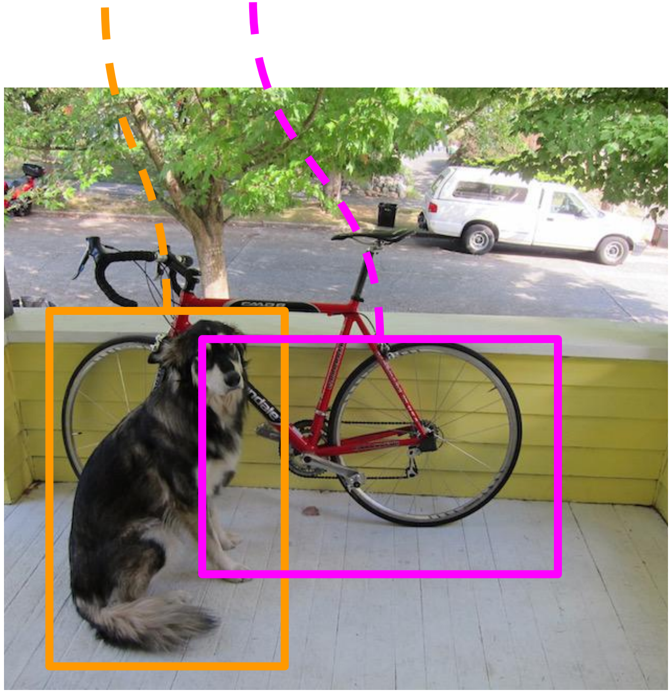
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: continue.

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.

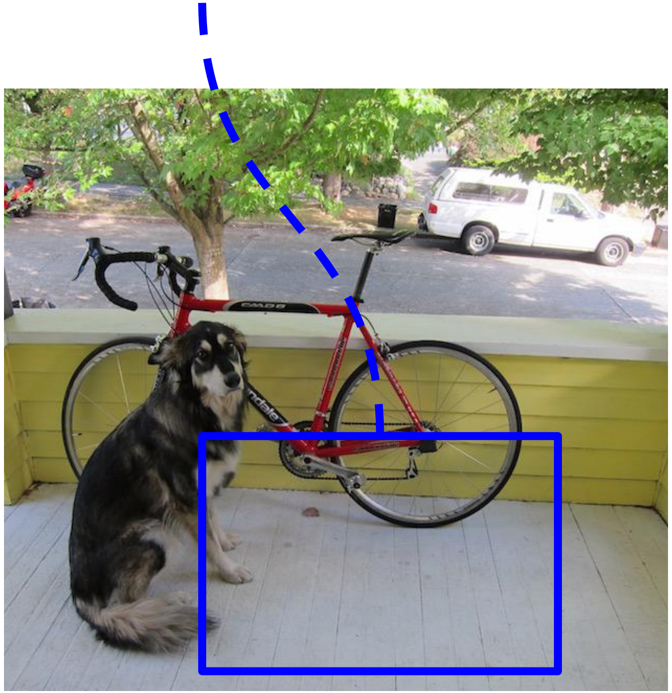
In this case: continue.

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

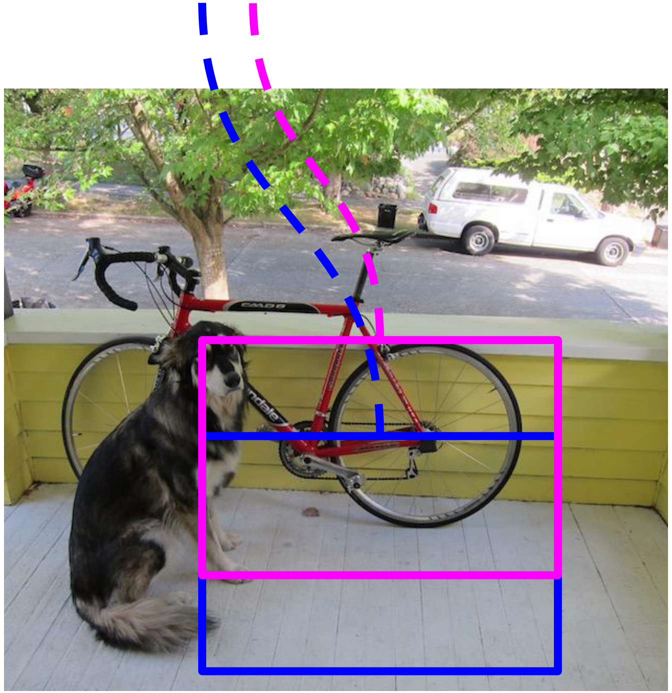


Go to next bbox with big score.
Let’s denote it “bbox_max”

Non-Maximum Suppression: intuition

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

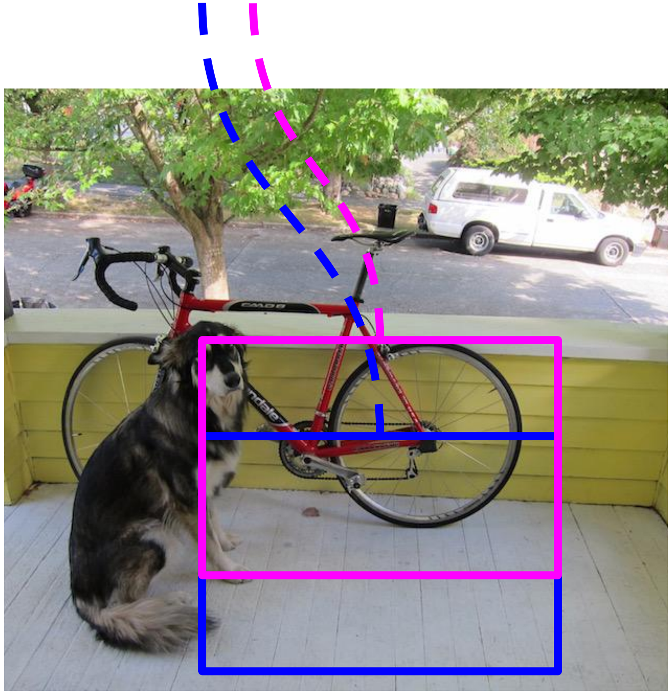


Go to next bbox_cur.

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.

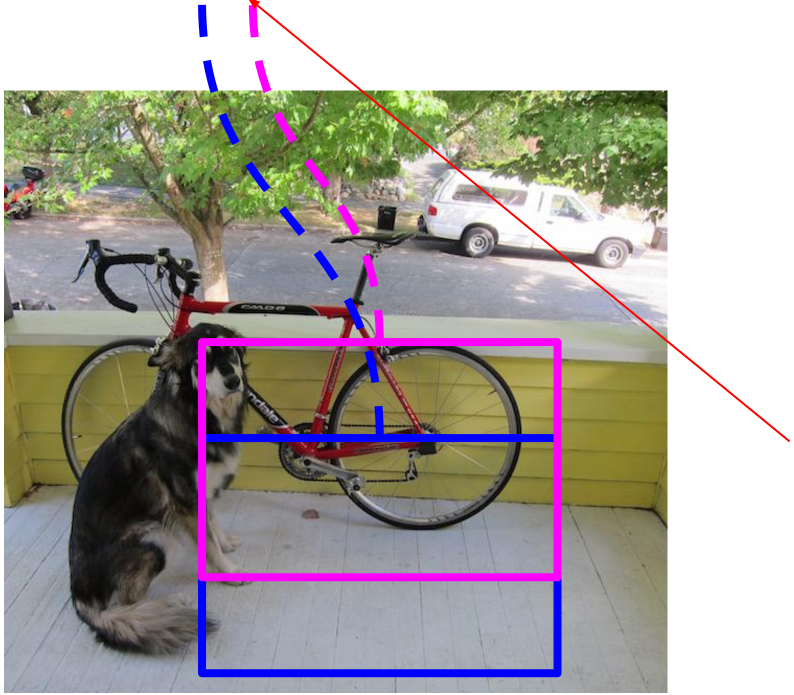
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

deepsystems.io

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



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.

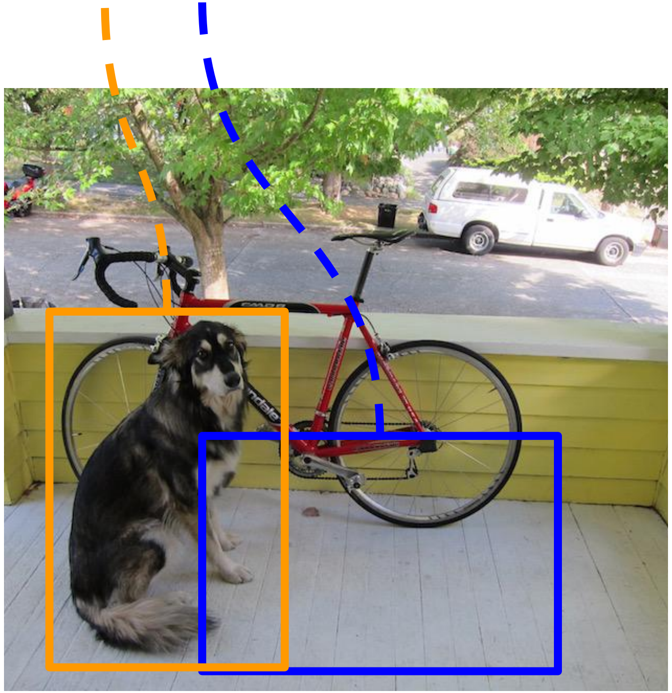
Do this procedure for other “bbox_max” and for other corresponding “bbox_cur”.

deepsystems.io

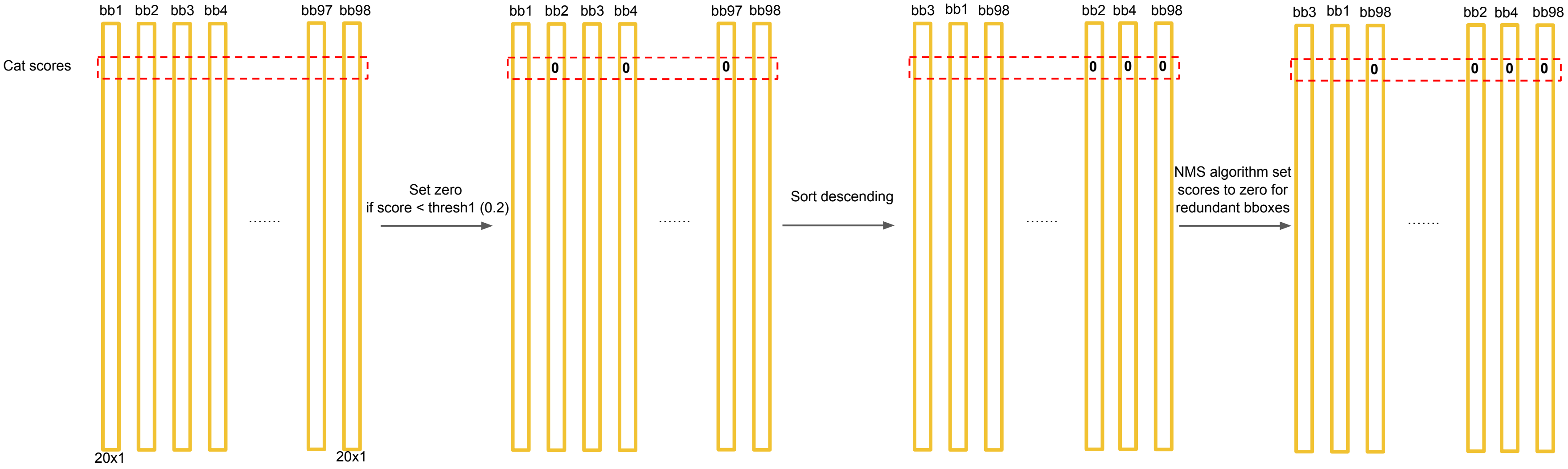
Non-Maximum Suppression: intuition

class (dog) scores for each bbox

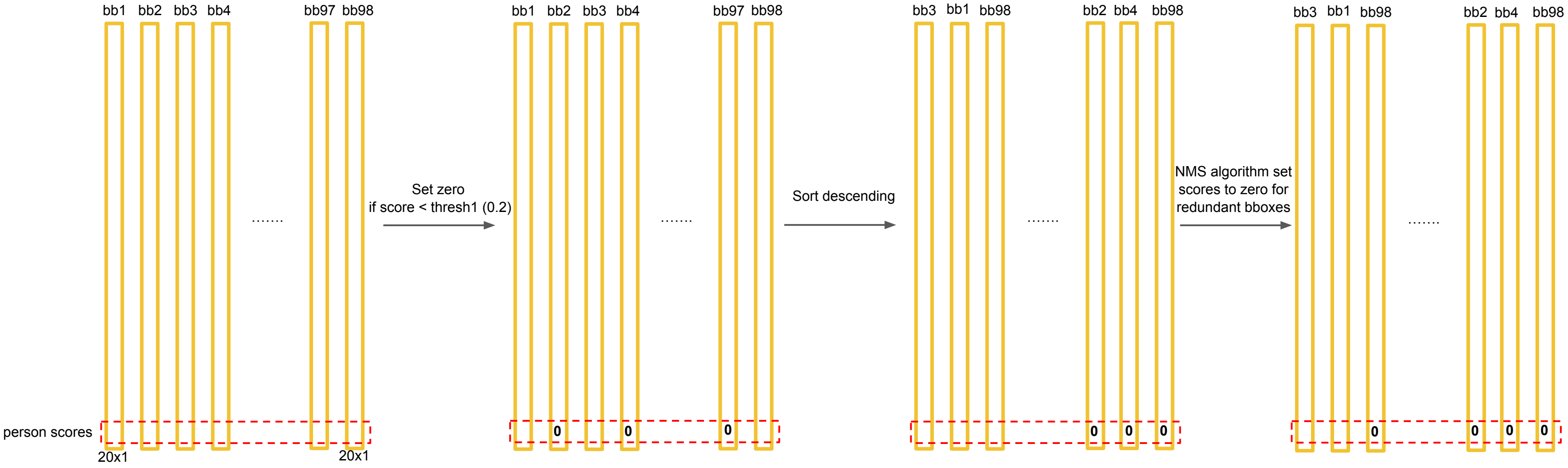
	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0							0	0	0	0	1x98



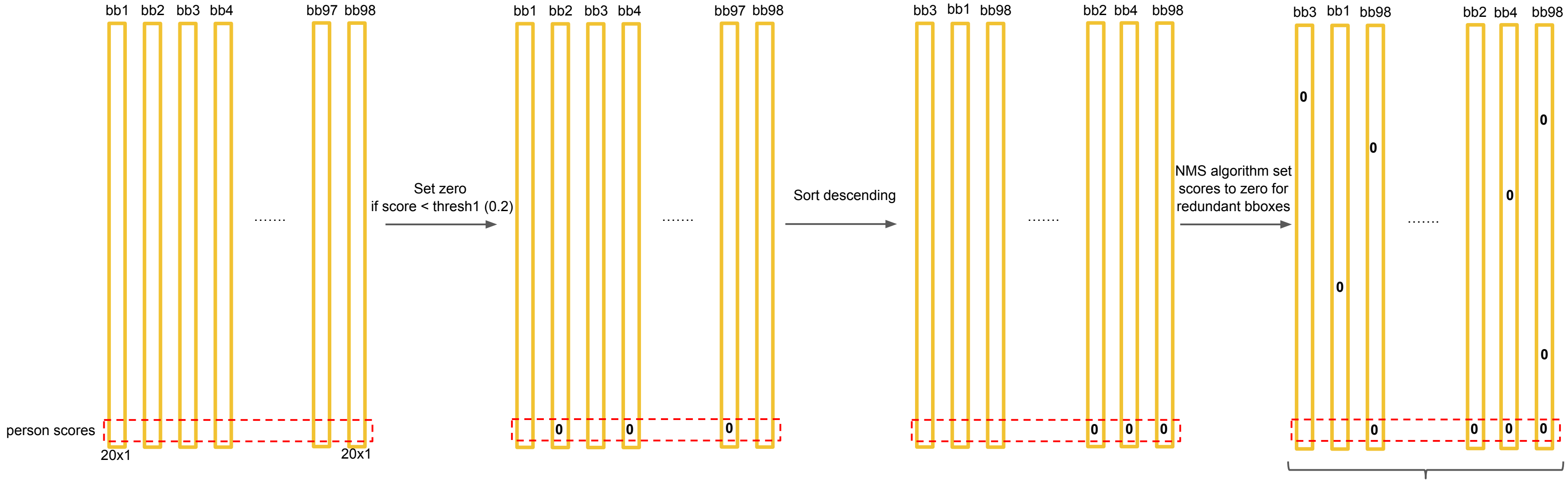
After comparison almost all pairs of bboxes the only two bboxes left with non-zero class score value.



Do this procedure for next class

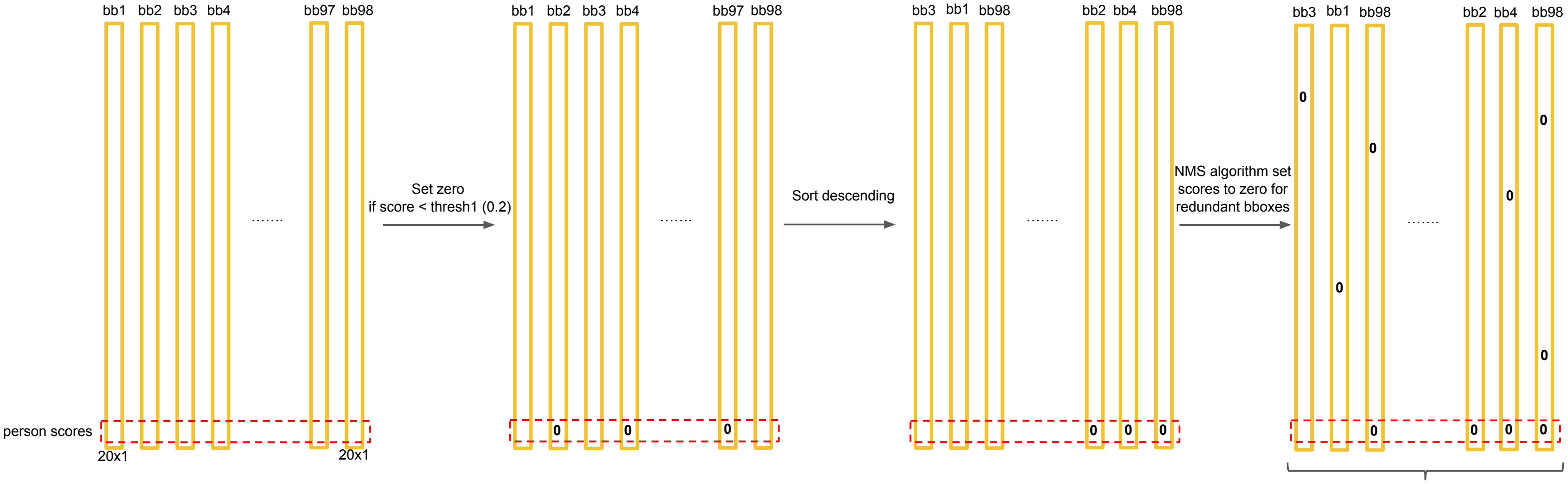


Do this procedure for all classes


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

After this procedure -
a lot of zeros

deepsystems.io

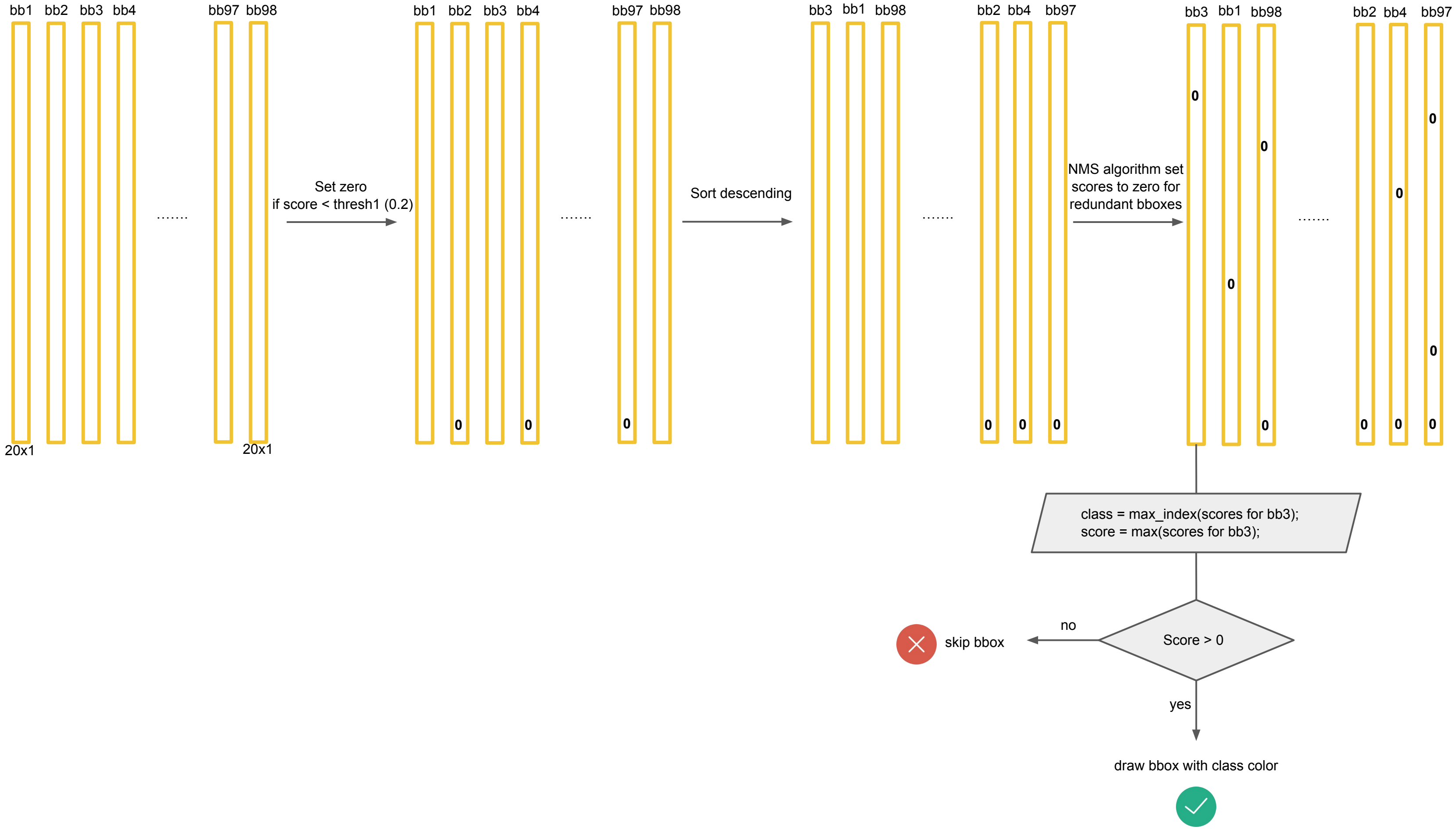

bb1 bb2 bb3 bb4 .... bb97 bb98
person scores
20x1
20x1
bb1 bb2 bb3 bb4 .... bb97 bb98
0 0 0
bb3 bb1 bb98 .... bb2 bb4 bb98
0 0 0
bb3 bb1 bb98 .... bb2 bb4 bb98
0 0 0 0 0 0 0 0 0 0 0 0

Set zero
if score < thresh1 (0.2)

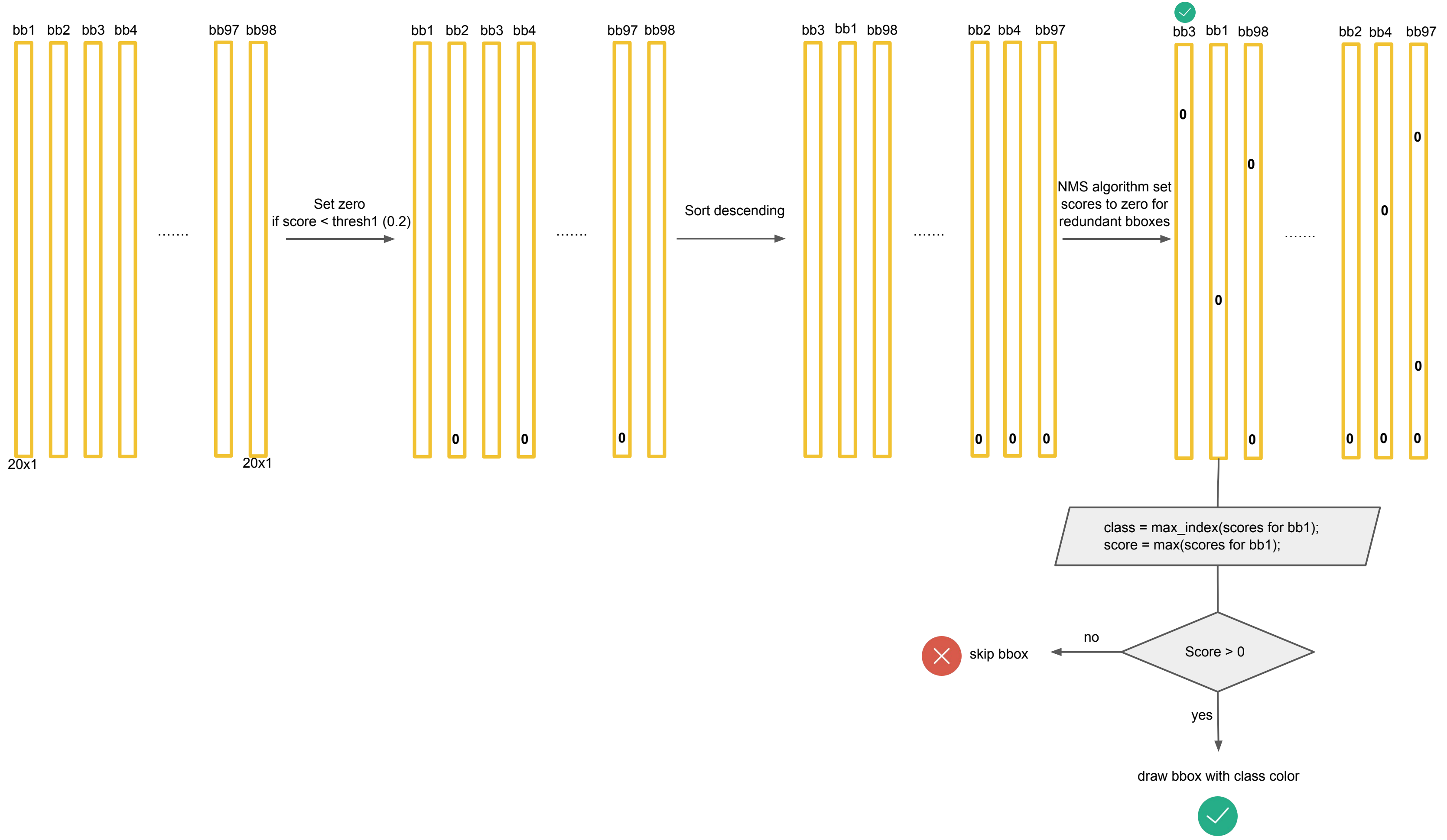
Sort descending

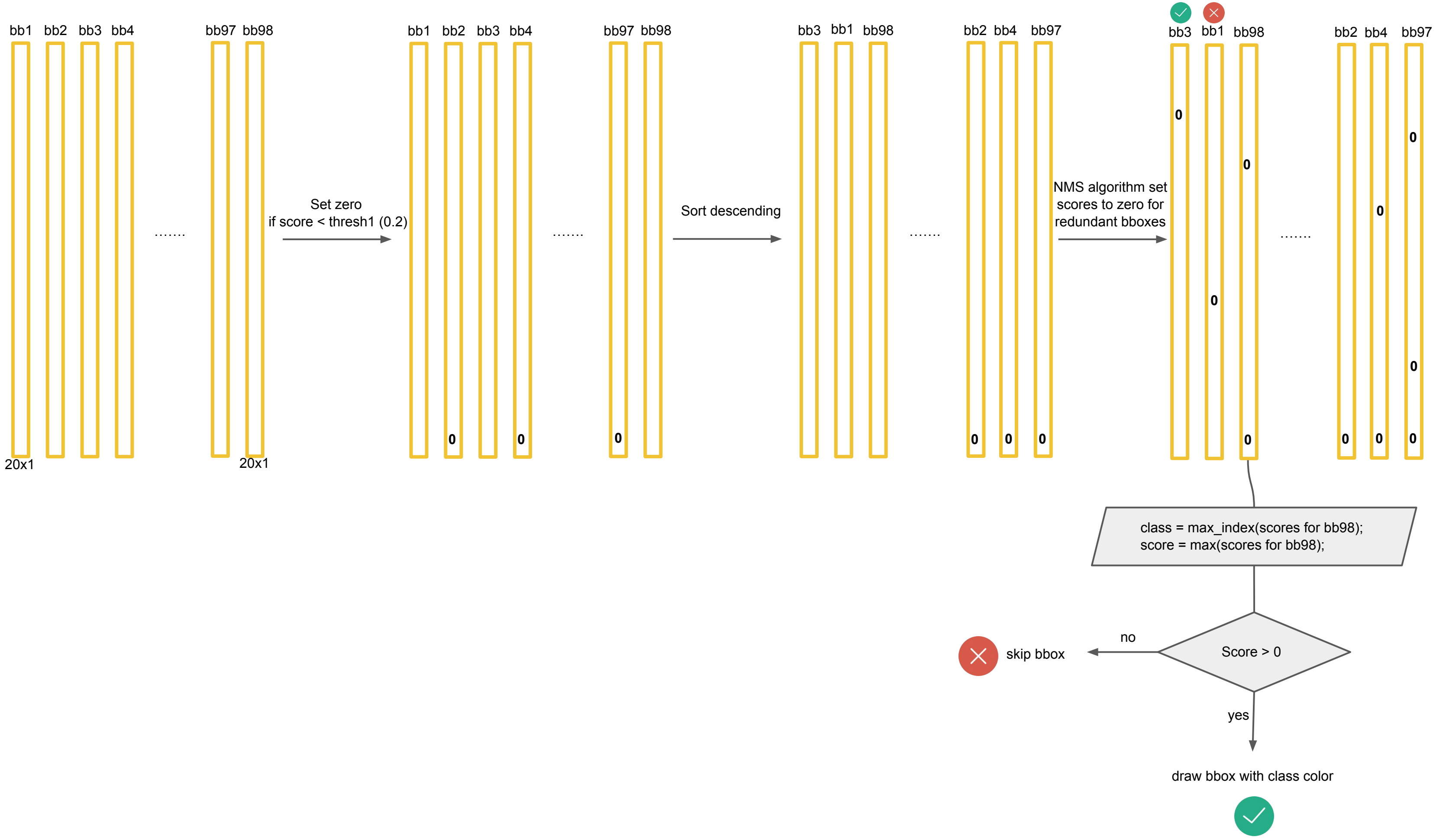
NMS algorithm set
scores to zero for
redundant bboxes

Select bboxes to draw by
class score values



deepsystems.io


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



Set zero
if score < thresh1 (0.2)

Sort descending

NMS algorithm set
scores to zero for
redundant bboxes

class = max_index(scores for bb98);
score = max(scores for bb98);

Score > 0

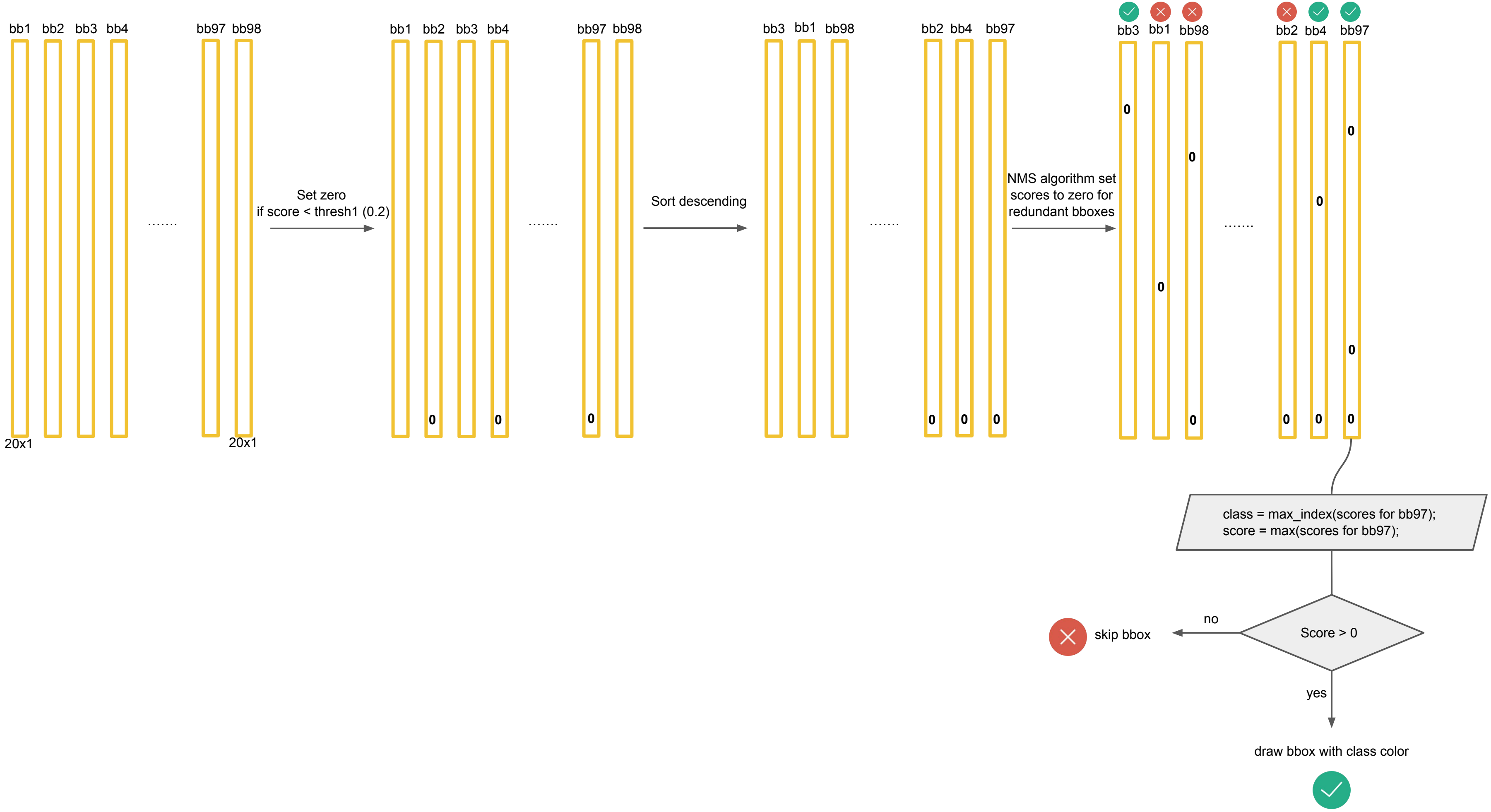
no

skip bbox

yes

draw bbox with class color

deepsystems.io


bb1 bb2 bb3 bb4 .... bb97 bb98
20x1
20x1
bb1 bb2 bb3 bb4 .... bb97 bb98
0
0
0
bb3 bb1 bb98 .... bb2 bb4 bb97
0
0
0
bb3 bb1 bb98 .... bb2 bb4 bb97
0
0
0
0
0
0
0
0
0
0
no
yes

Set zero
if score < thresh1 (0.2)

Sort descending

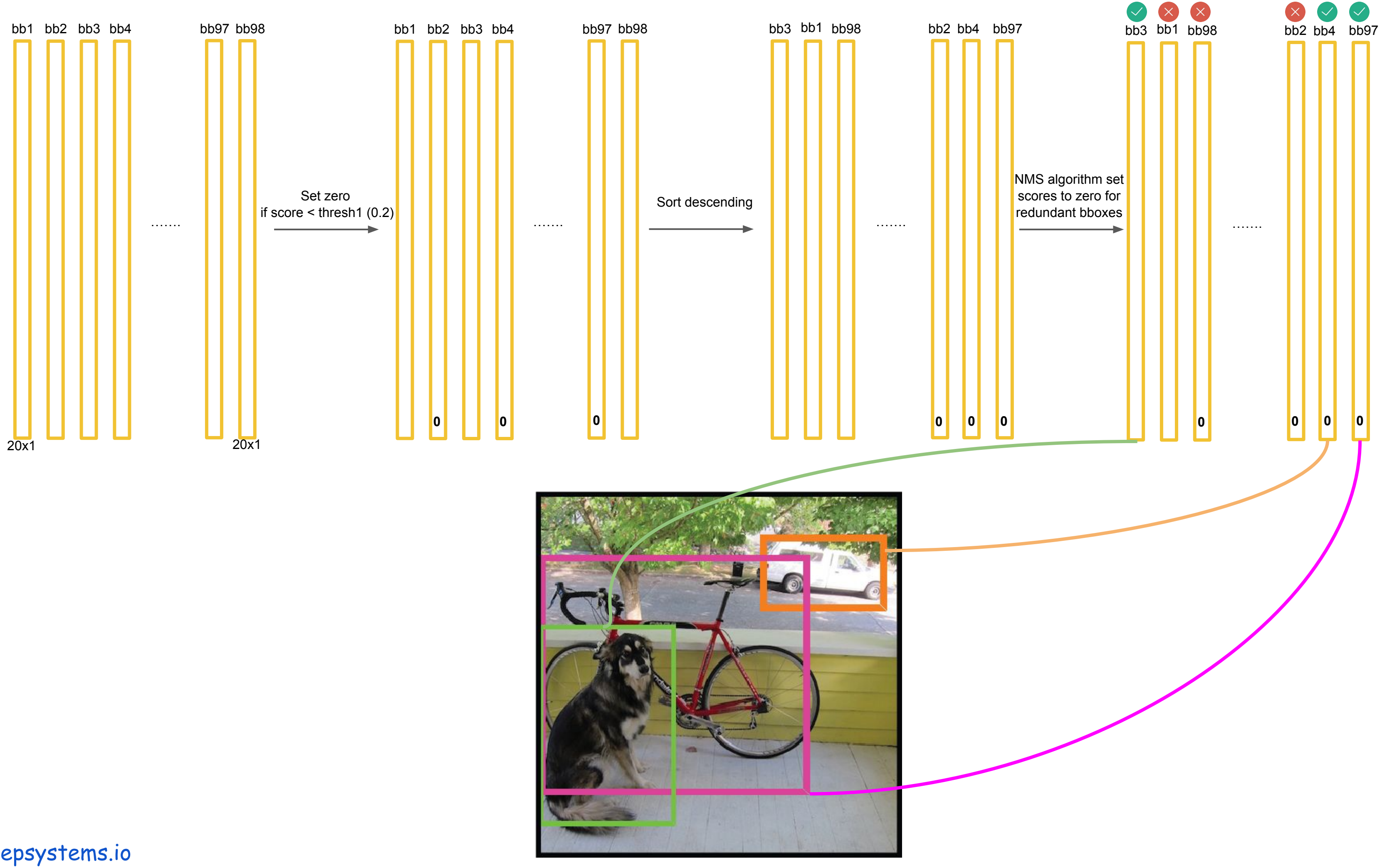
NMS algorithm set
scores to zero for
redundant bboxes

class = max_index(scores for bb97);
score = max(scores for bb97);

Score > 0

skip bbox

draw bbox with class color



Set zero if score < thresh1 (0.2)

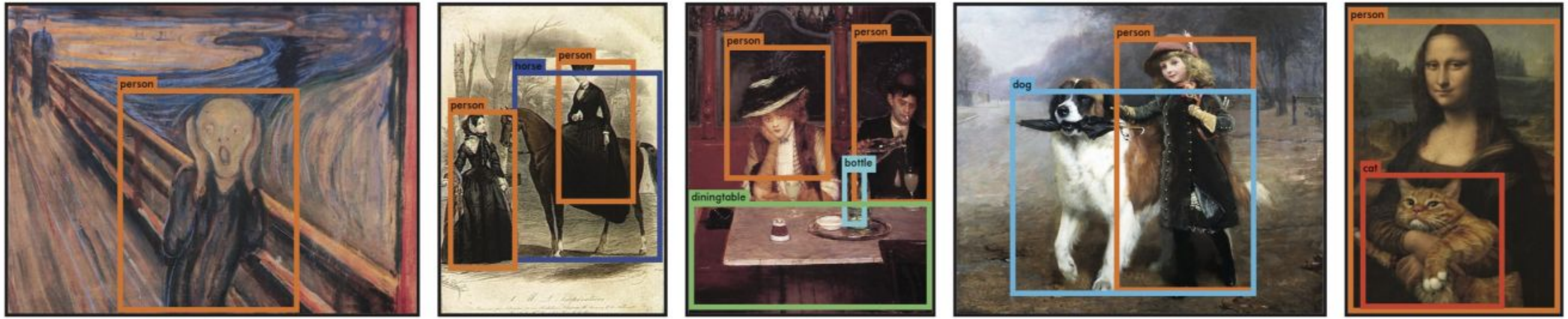
Sort descending

NMS algorithm set scores to zero for redundant bboxes

bb1 bb2 bb3 bb4 .... bb97 bb98
20x1
20x1
bb1 bb2 bb3 bb4 .... bb97 bb98
0 0 0
bb3 bb1 bb98 .... bb2 bb4 bb97
0 0 0
✓ ✗ ✗
bb3 bb1 bb98
0
....
✗ ✓ ✓
bb2 bb4 bb97
0 0 0

Key Points

1. Fast: YOLO - 45 fps, YOLO-tiny - 155 fps.
2. End-to-end training.
3. Makes more localization errors but is less likely to predict false positives on background
4. Performance is lower than the current state of the art.
5. Combined Fast R-CNN + YOLO model is one of the highest performing detection methods.
6. Learns very general representations of objects: it outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains



Links

- Arxiv: https://arxiv.org/abs/1506.02640
- Blog: http://pjreddie.com/publications/yolo/
- Darknet: https://github.com/pjreddie/darknet
- Caffe: https://github.com/xingwangsfu/caffe-yolo
- Tensorflow:
  - Test+train: https://github.com/thtrieu/yolotf
  - Test: https://github.com/gliese581gg/YOLO_tensorflow

Thank you!

Our website: deepsystems.io

Our team is looking for business partners to make exciting deep learning solutions.