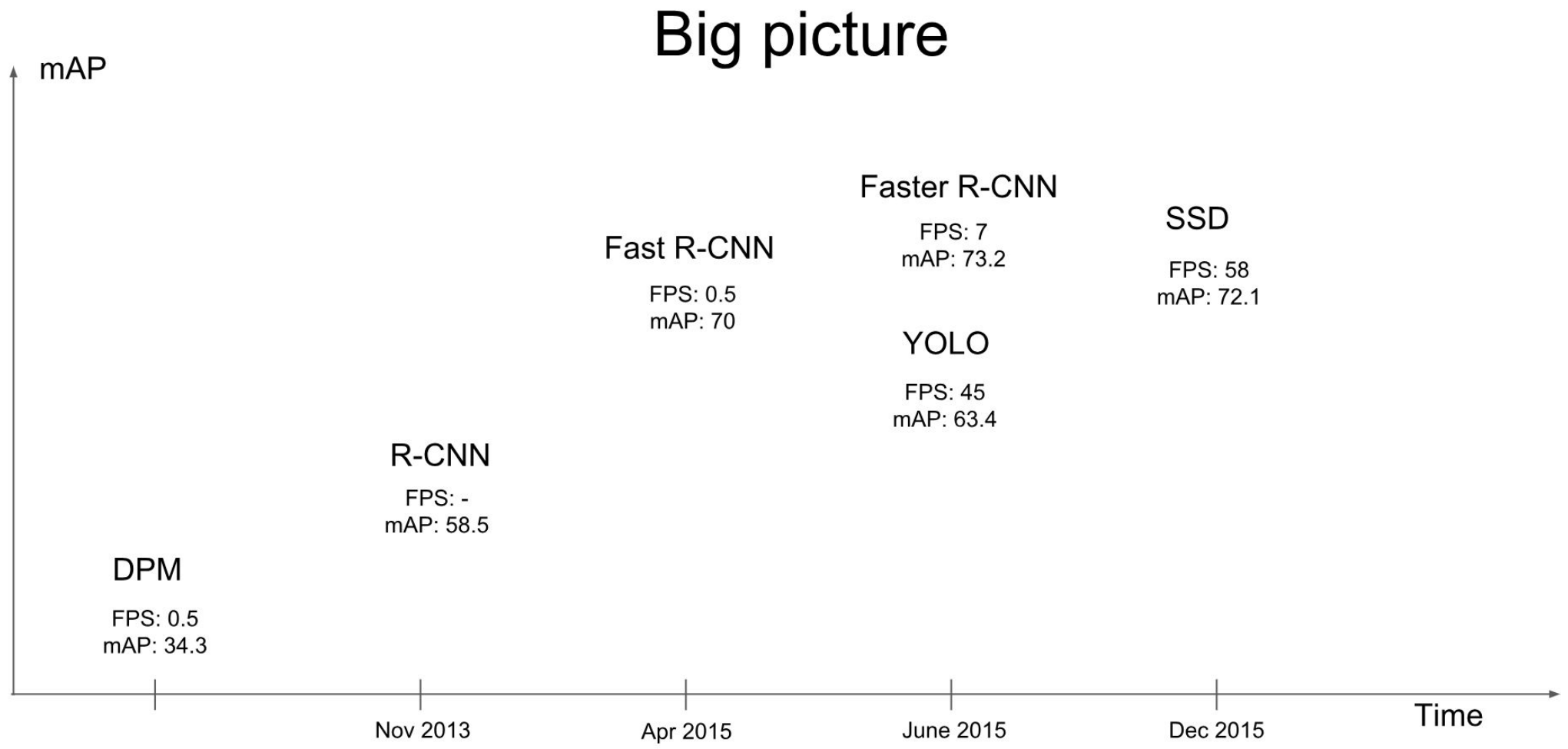
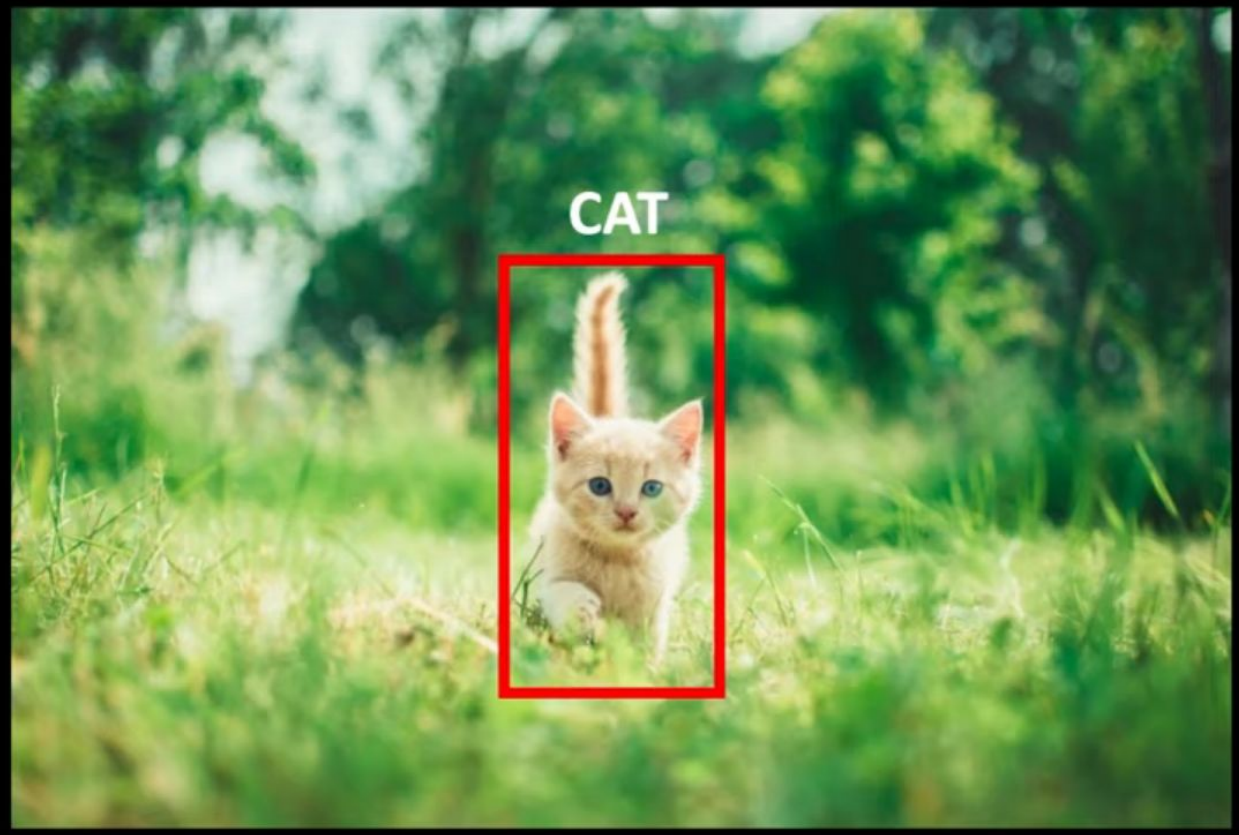
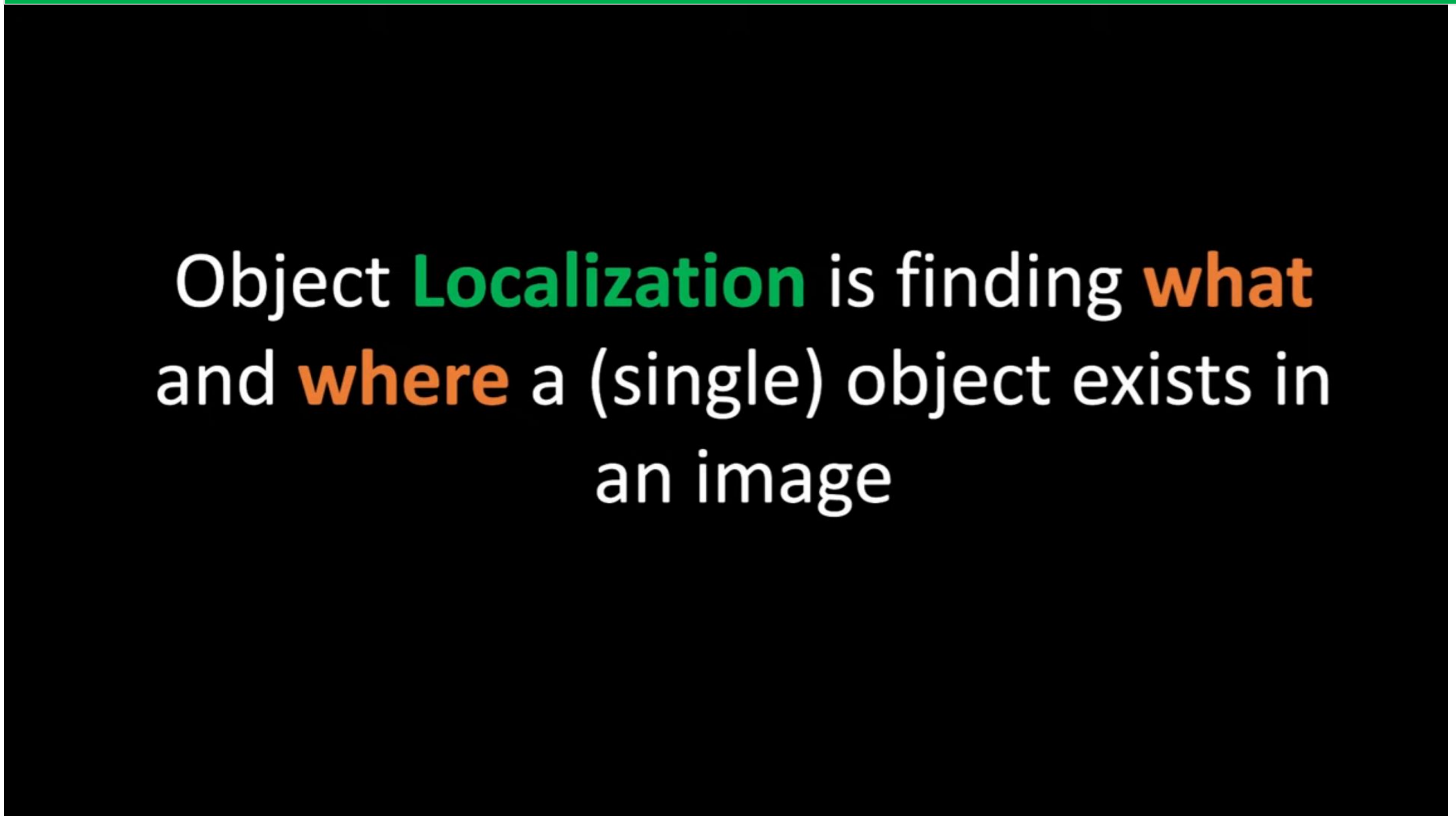


Результаты на тестовой выборки Pascal VOC 2007. Обучение на trainval sets 2007+2012

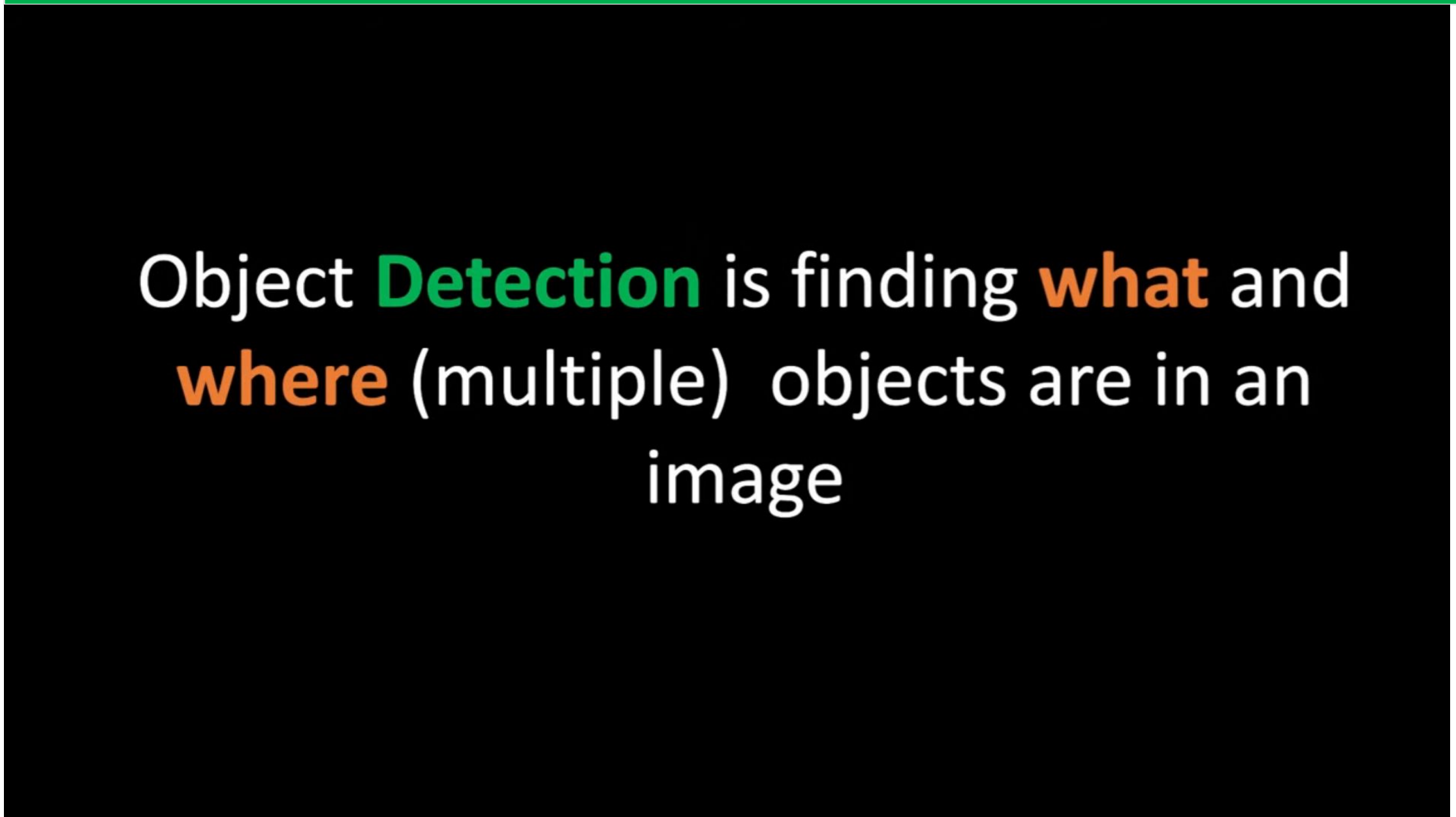
Starting with Localization



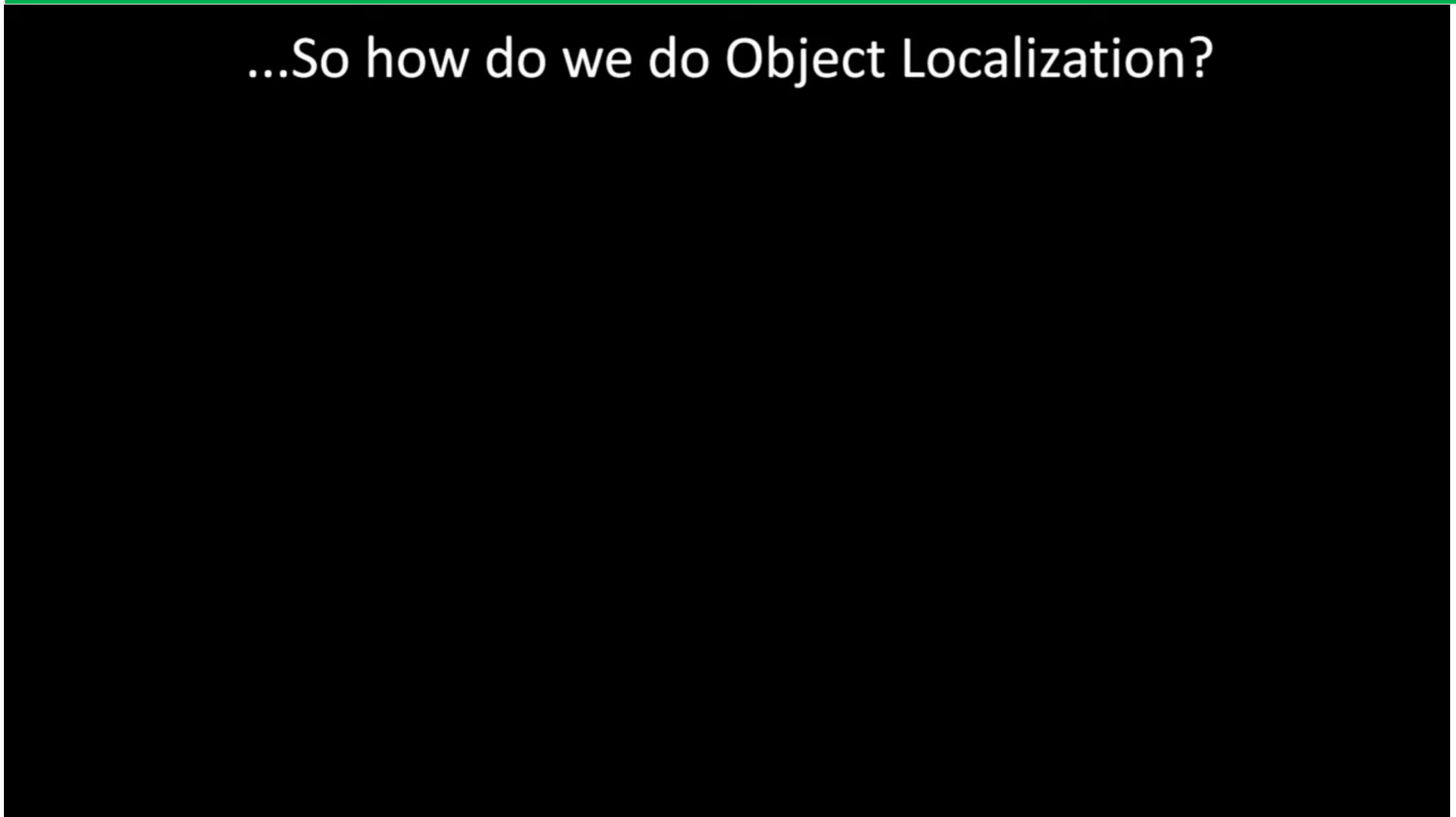
Object Detection



Object Detection



Object Detection



...So how do we do Object Localization?

CNN

CAT

DOG

x1

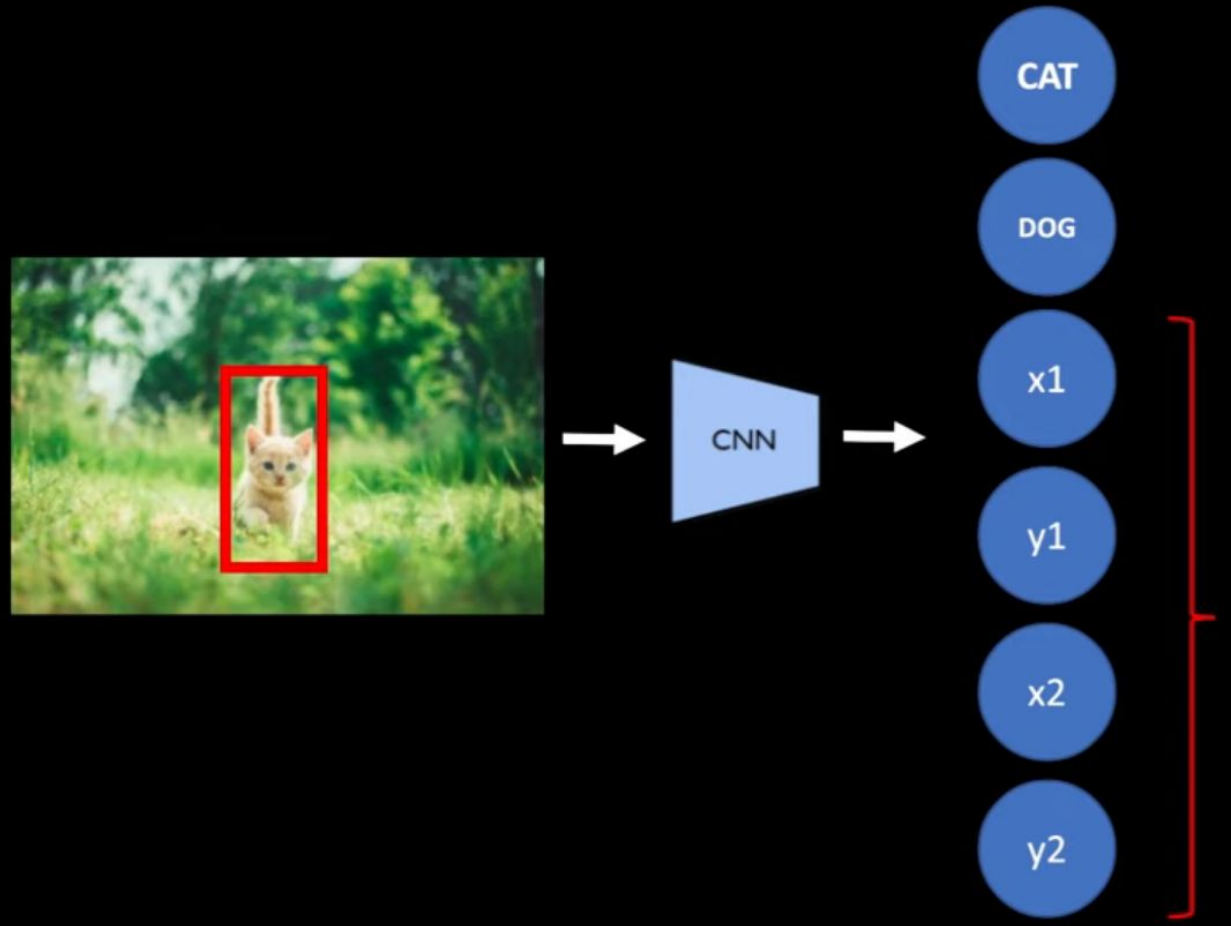
y1

x2

y2

Need at least 4 points to define a bounding box!

...So how do we do Object Localization?



Two Common ways to define BBOXES:

1. (x1, y1) is upper left corner point, (x2,y2) is bottom right corner point

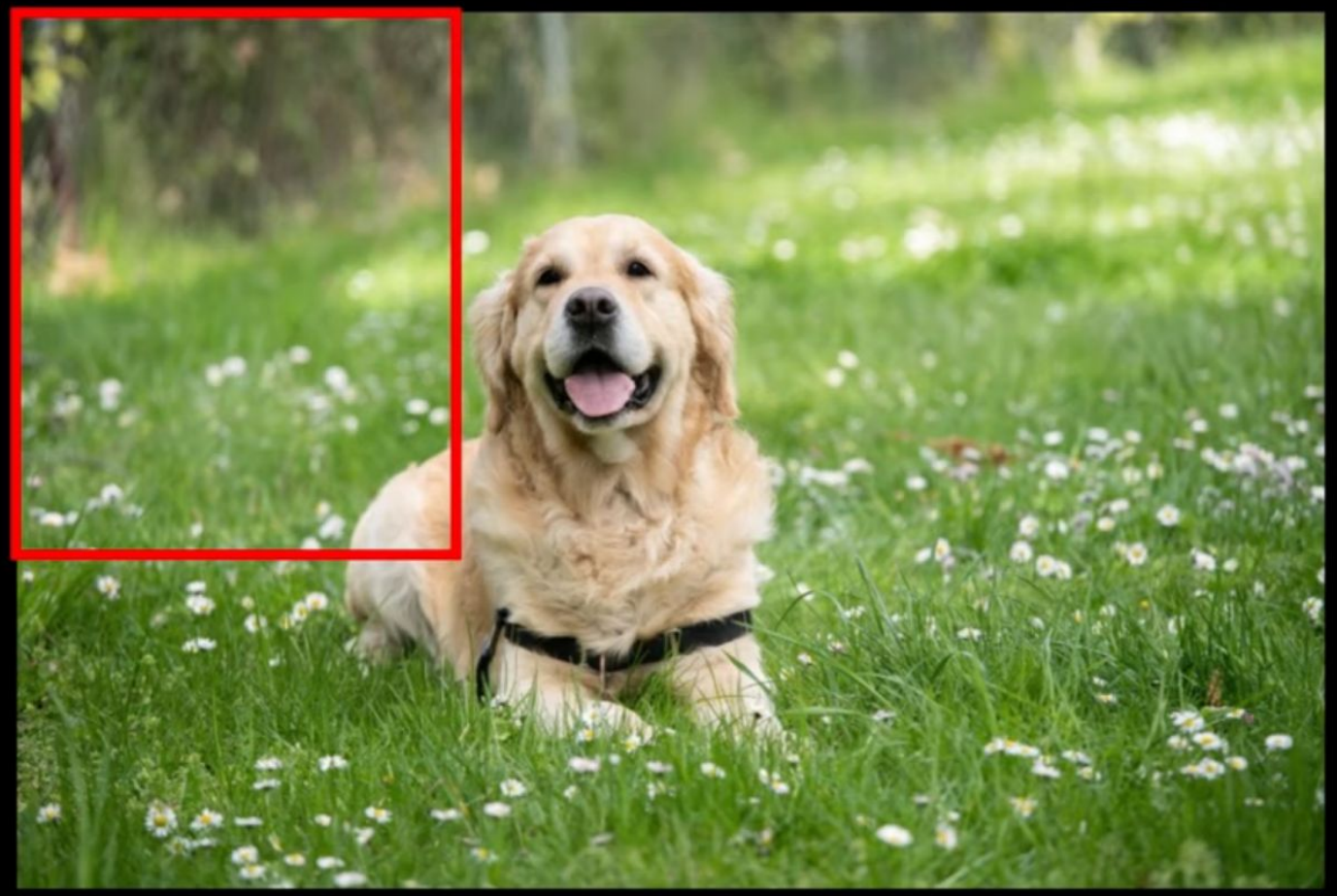
2. Two points define a corner point, and two points to define height and width

Alright, so localization is no biggie!

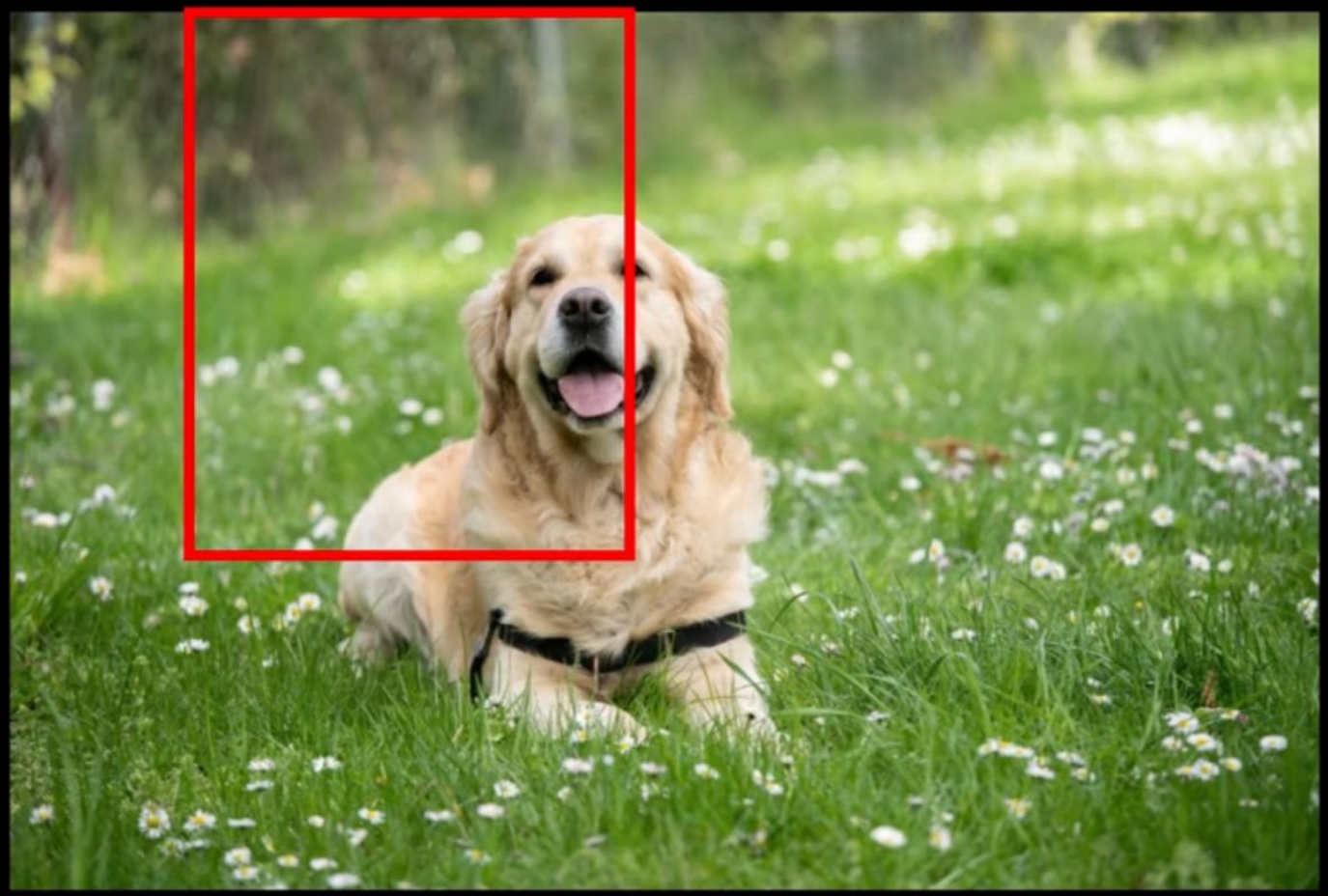
But... how do we **generalize** this for multiple objects in an image?

There are many approaches to solving object detection and each is in many ways unique!

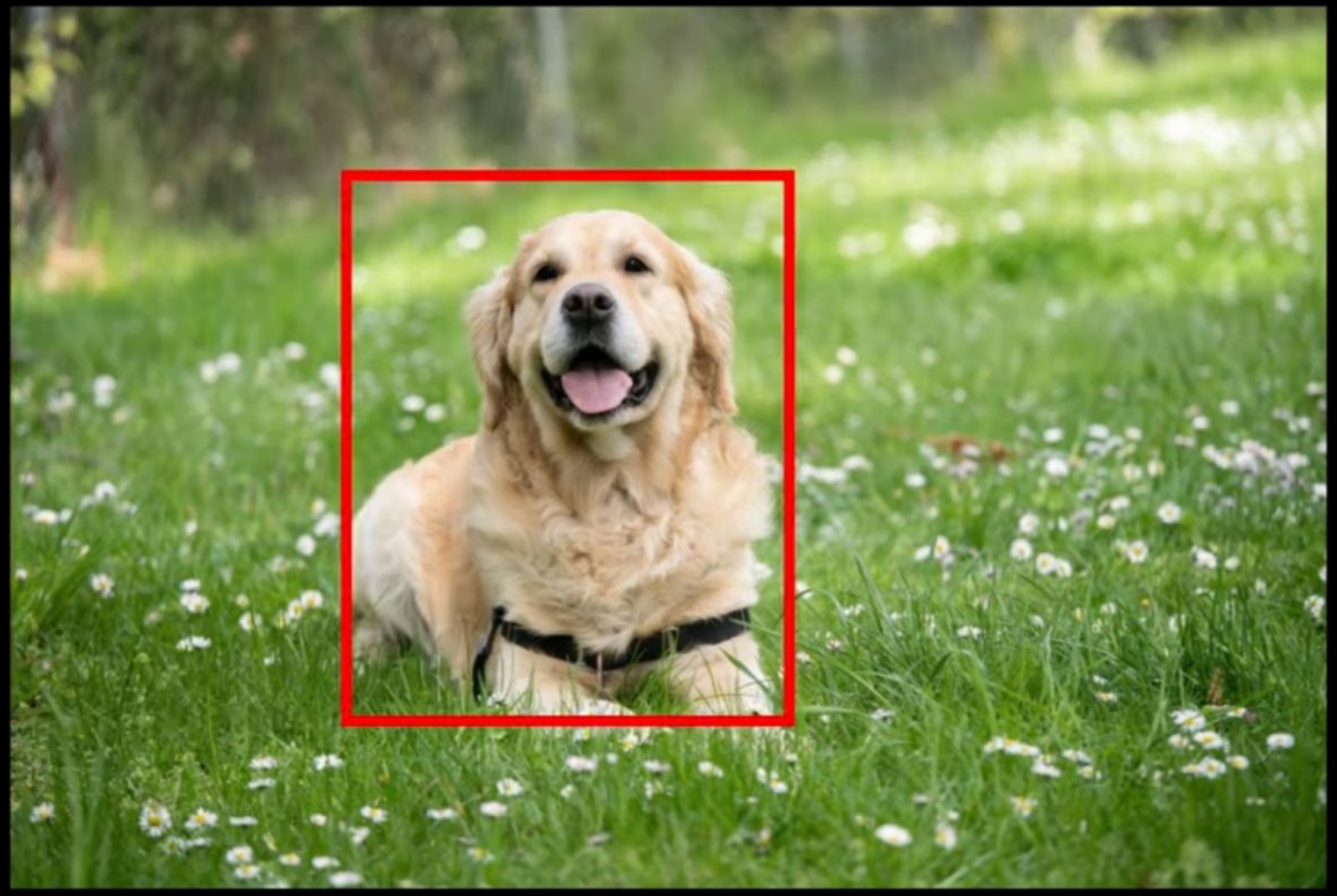
Sliding Windows: The "natural" extension



Sliding Windows: The "natural" extension



Sliding Windows: The "natural" extension



Sliding Windows: The "natural" extension

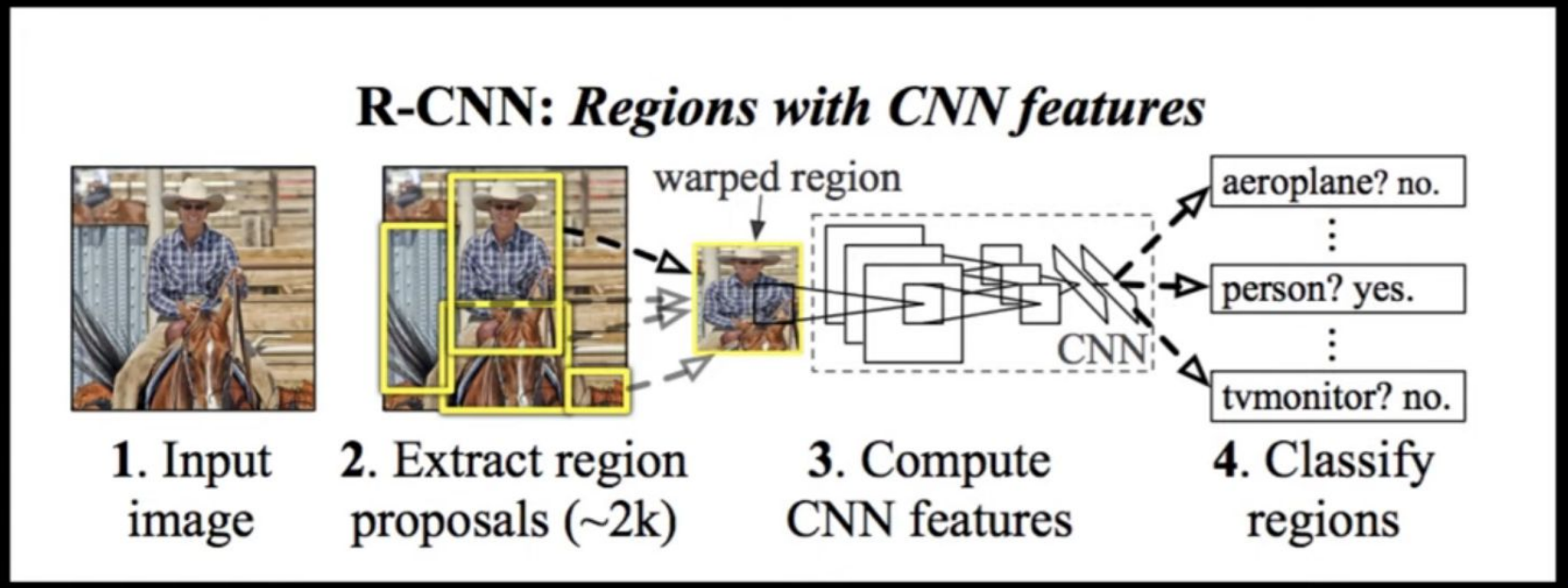
Note:
The sliding window is usually a square

Potential Problems?

1. **A LOT** of computation!

2. **Many** bounding boxes for same object

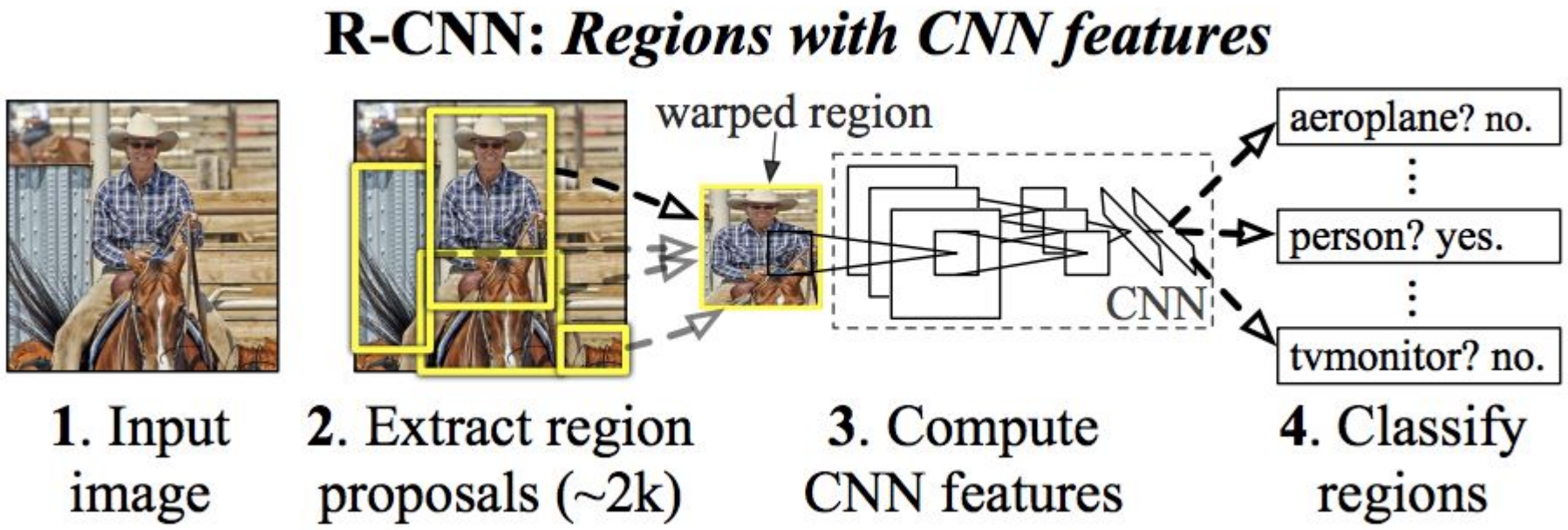
Regional based networks



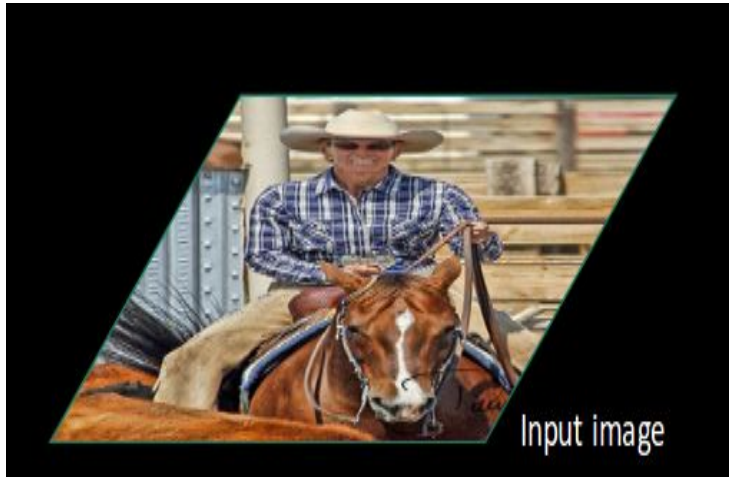
3 Papers:
R-CNN
Fast rCNN
Faster rCNN

Note: These networks are very tricky to implement!

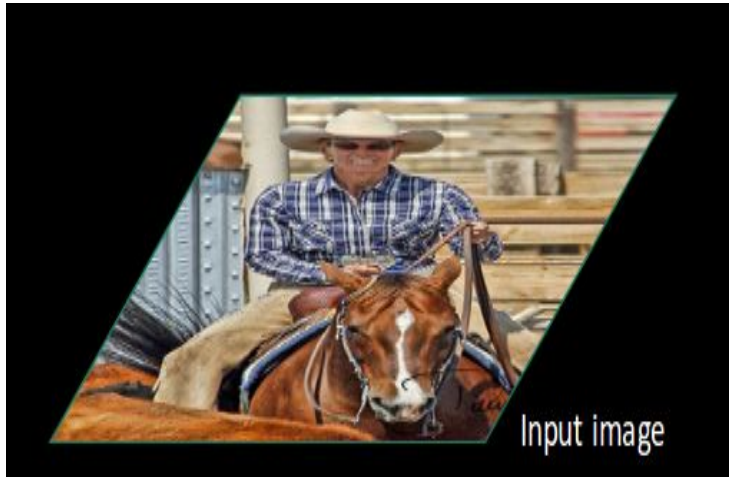
R-CNN


R-CNN: Regions with CNN features
warped region
CNN
aeroplane? no.
person? yes.
tvmonitor? no.
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

R-CNN

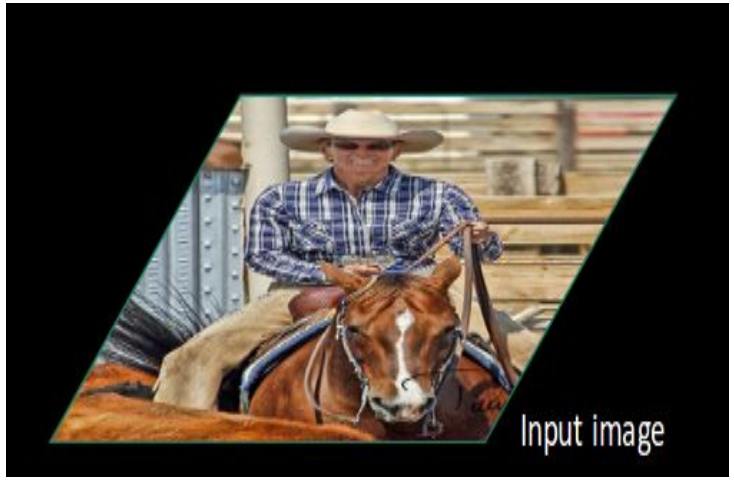


R-CNN

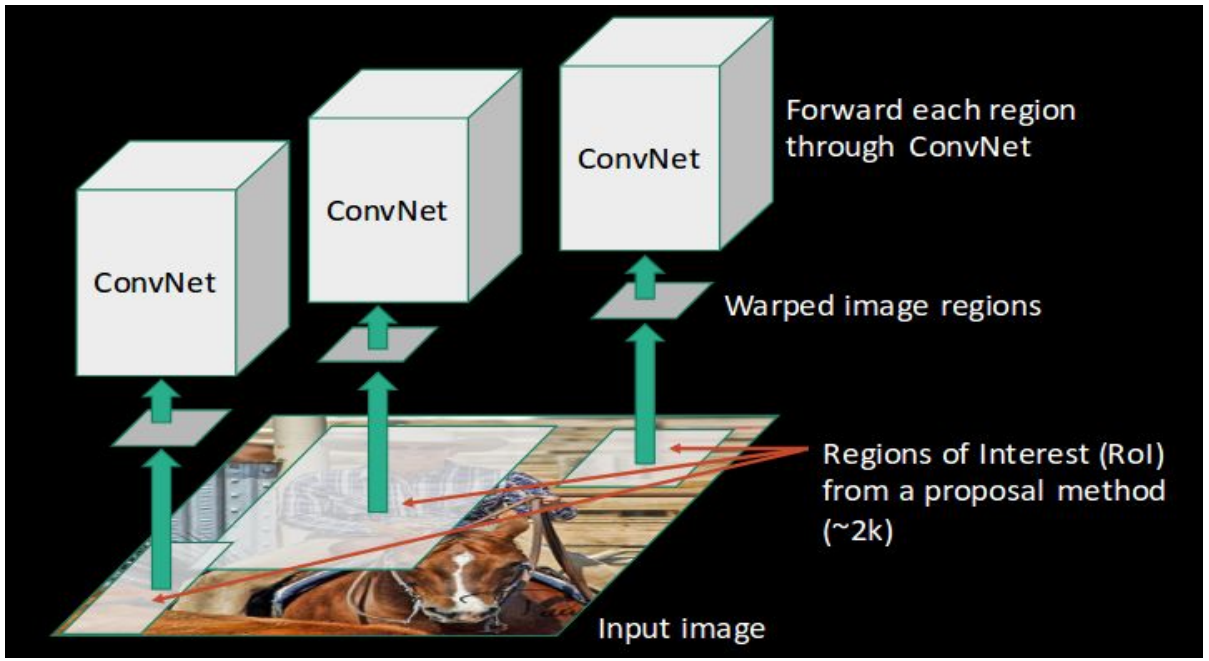

Input image


Regions of Interest (RoI)
from a proposal method
(~2k)
Input image

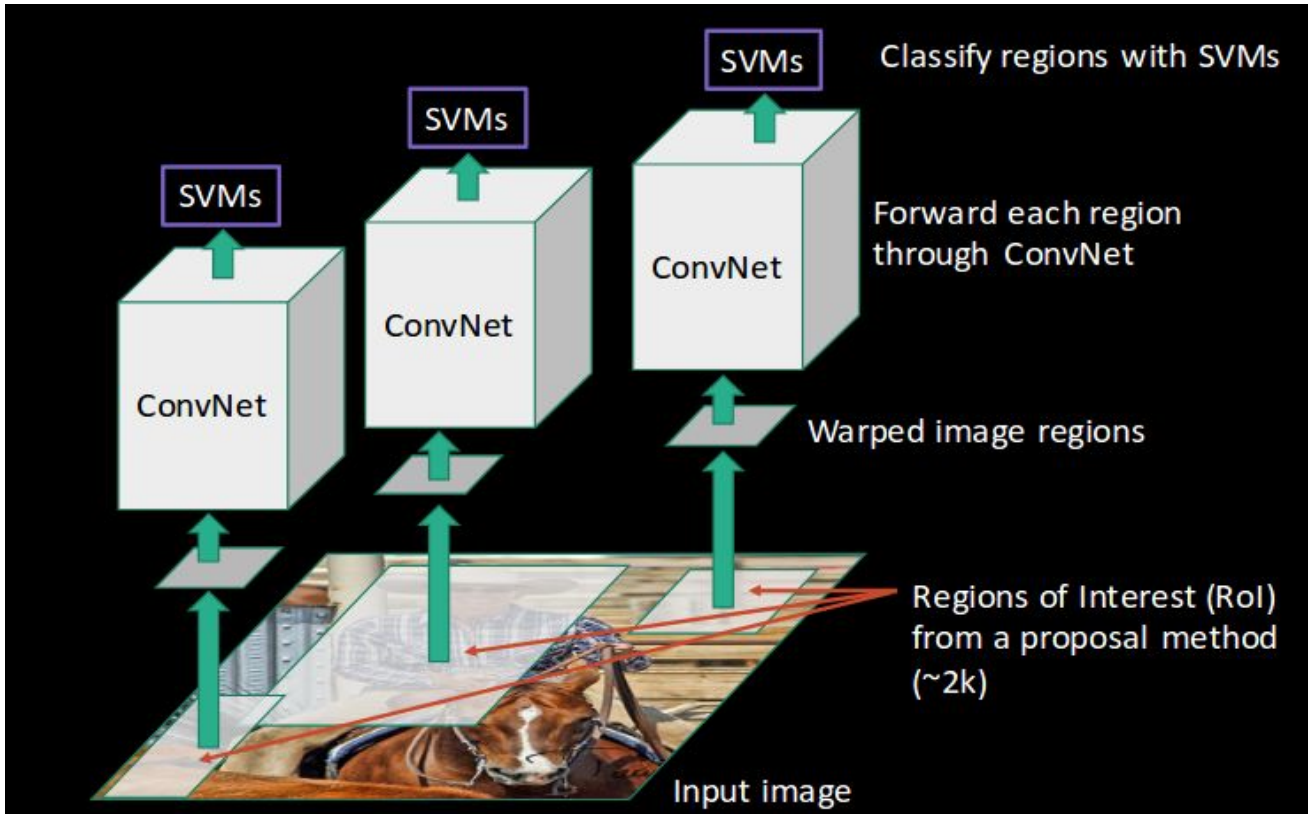
R-CNN


Input image

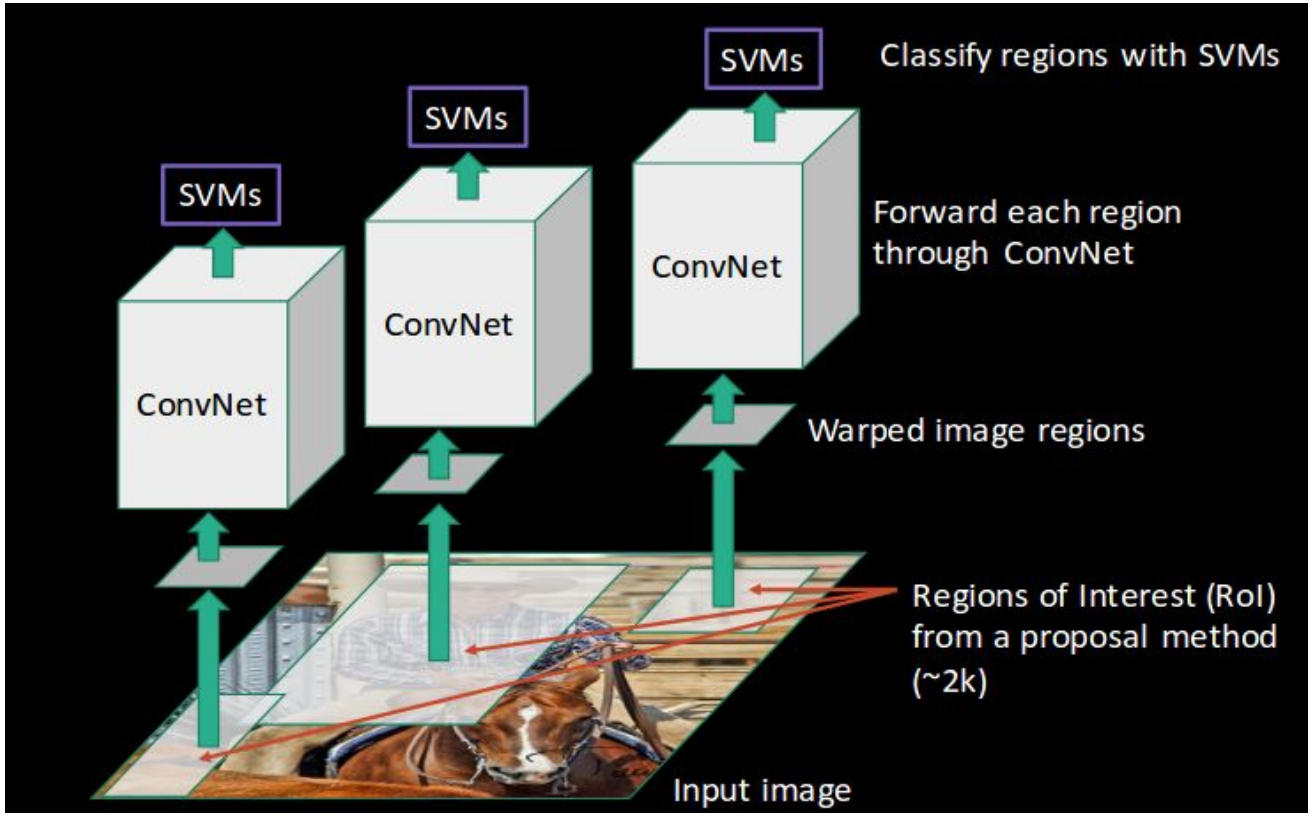

Regions of Interest (RoI)
from a proposal method
(~2k)
Input image

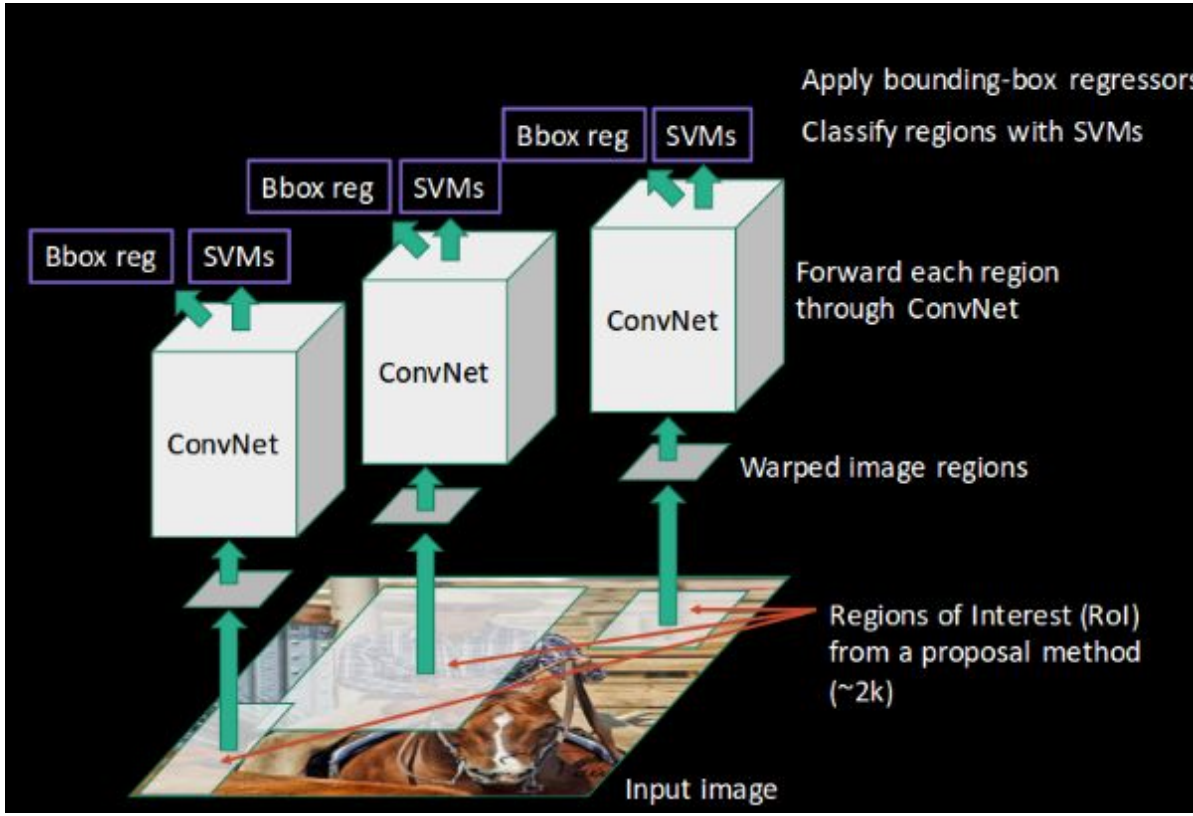

ConvNet
ConvNet
ConvNet
Forward each region
through ConvNet
Warped image regions
Regions of Interest (RoI)
from a proposal method
(~2k)
Input image

R-CNN

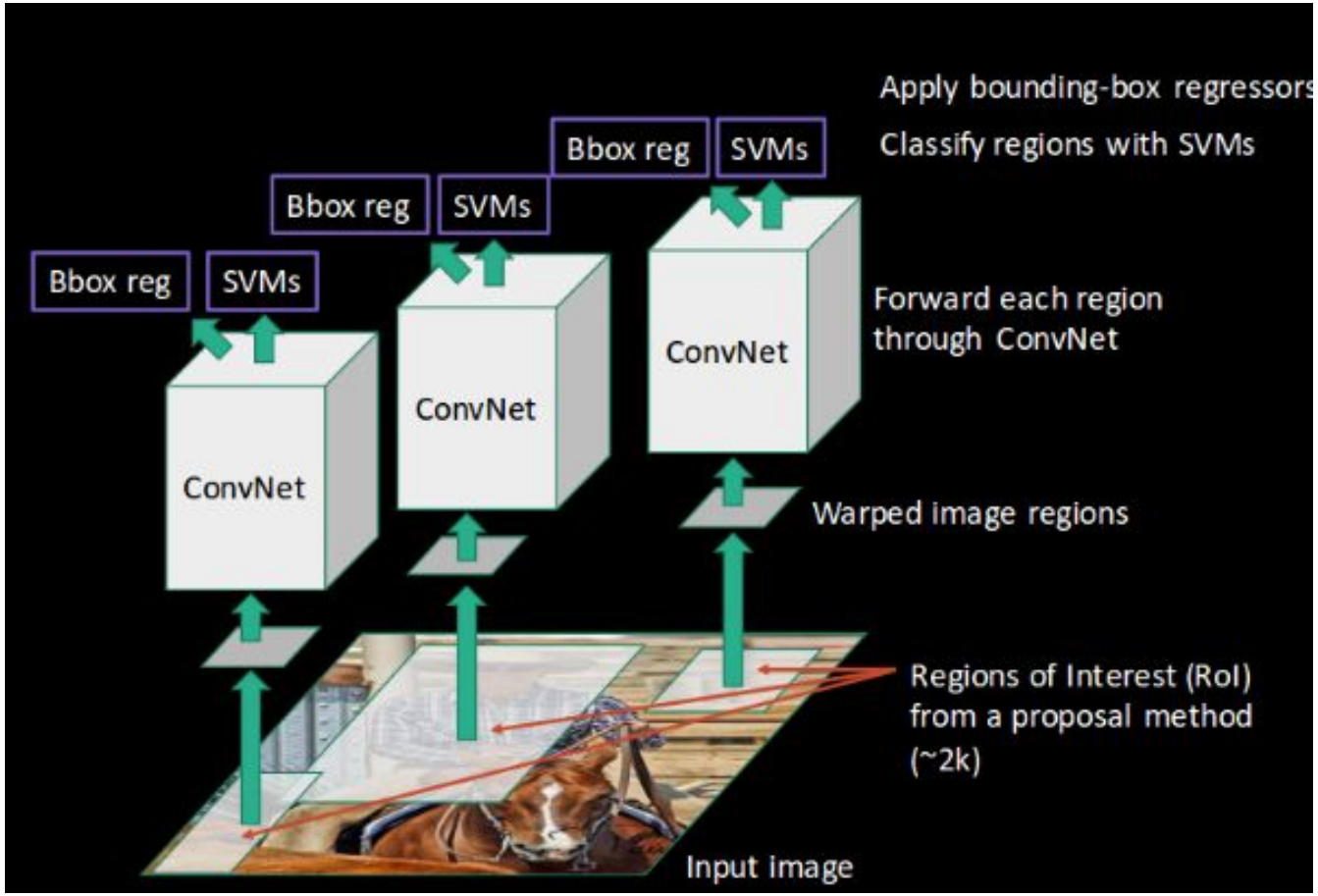

SVMs
SVMs
SVMs
Classify regions with SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

R-CNN

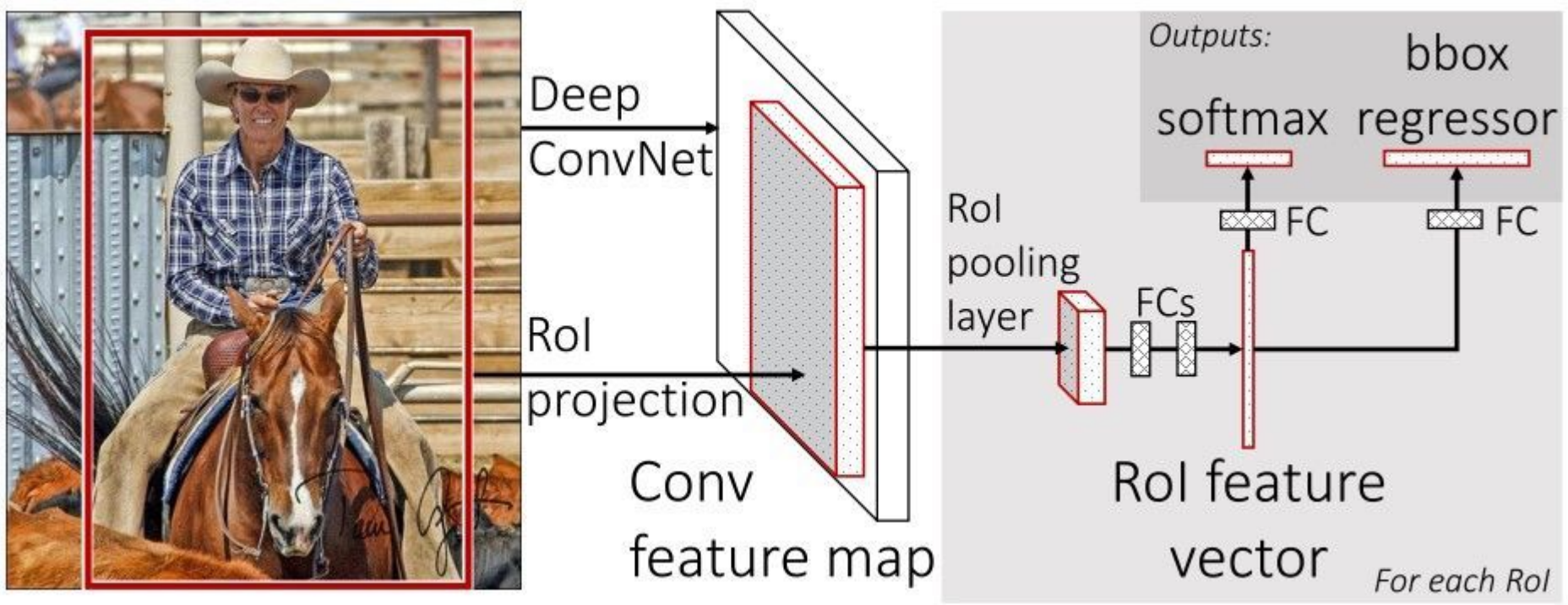

SVMs
SVMs
SVMs
Classify regions with SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image


Apply bounding-box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

R-CNN


Apply bounding-box regressors
Classify regions with SVMs
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image
Bbox reg
SVMs
ConvNet

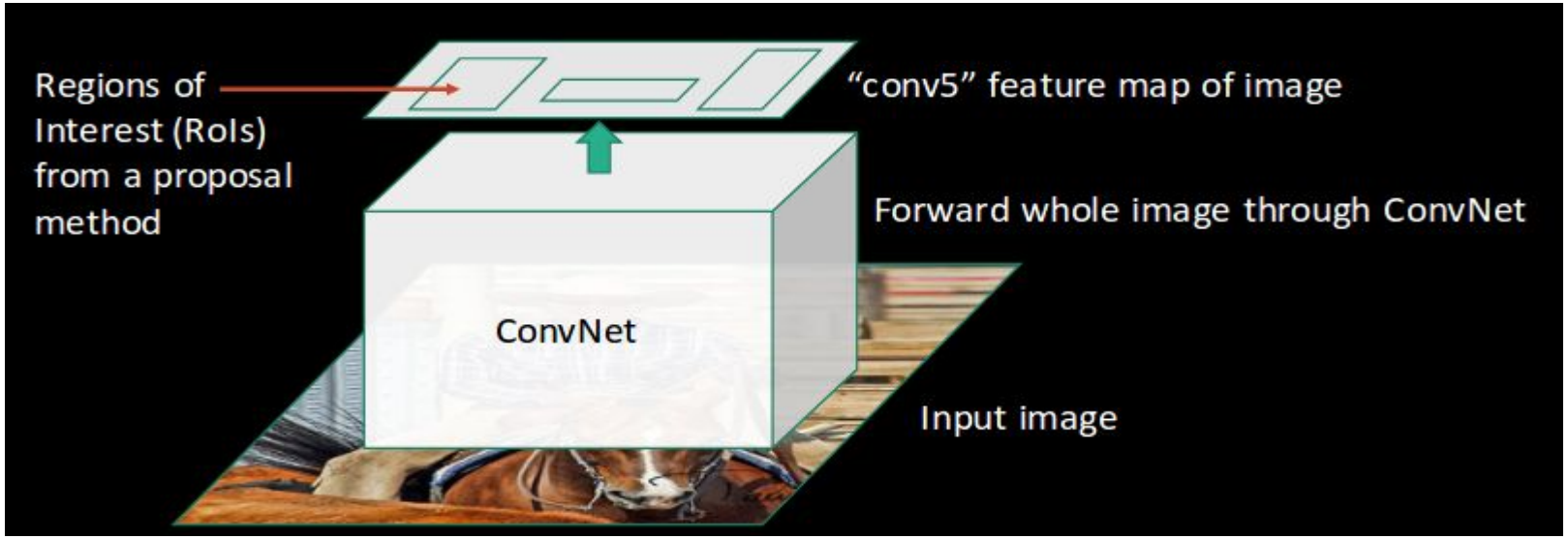
Fast R-CNN


Deep
ConvNet
RoI
projection
Conv
feature map
RoI
pooling
layer
FCs
Outputs:
softmax
bbox
regressor
FC
FC
RoI feature
vector
For each RoI

Fast R-CNN

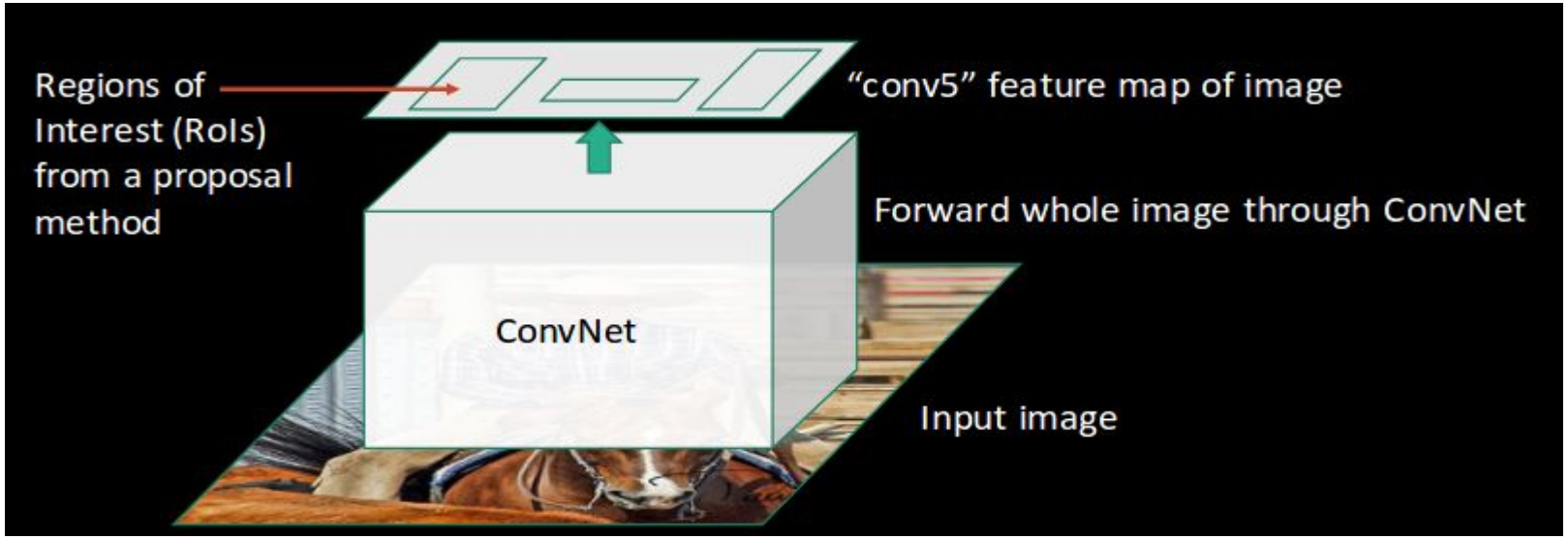
Fast R-CNN

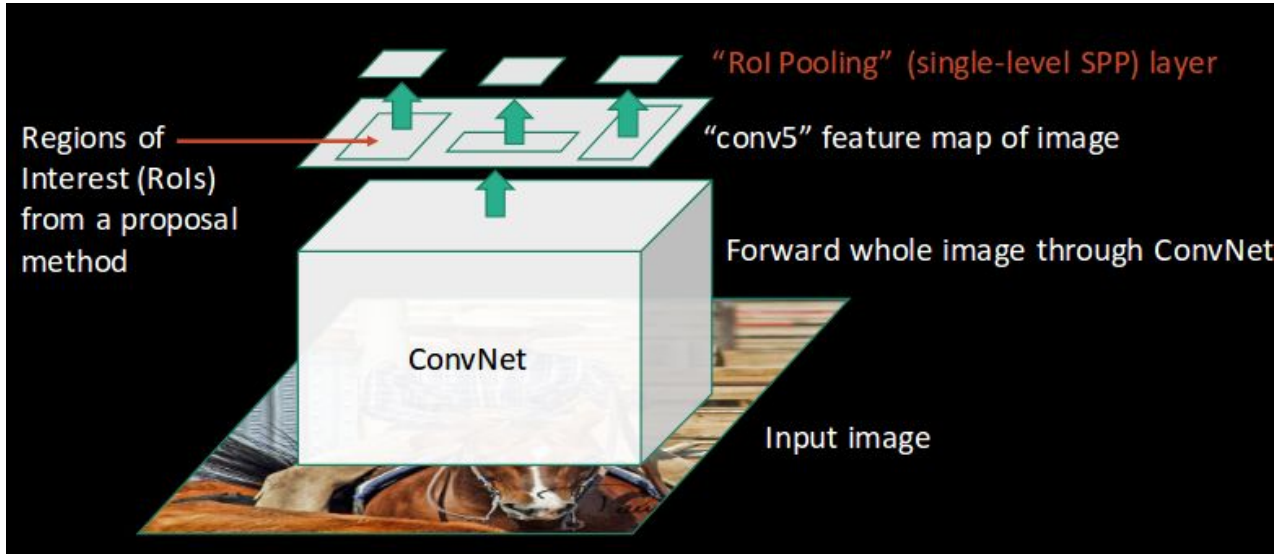

Input image


Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image

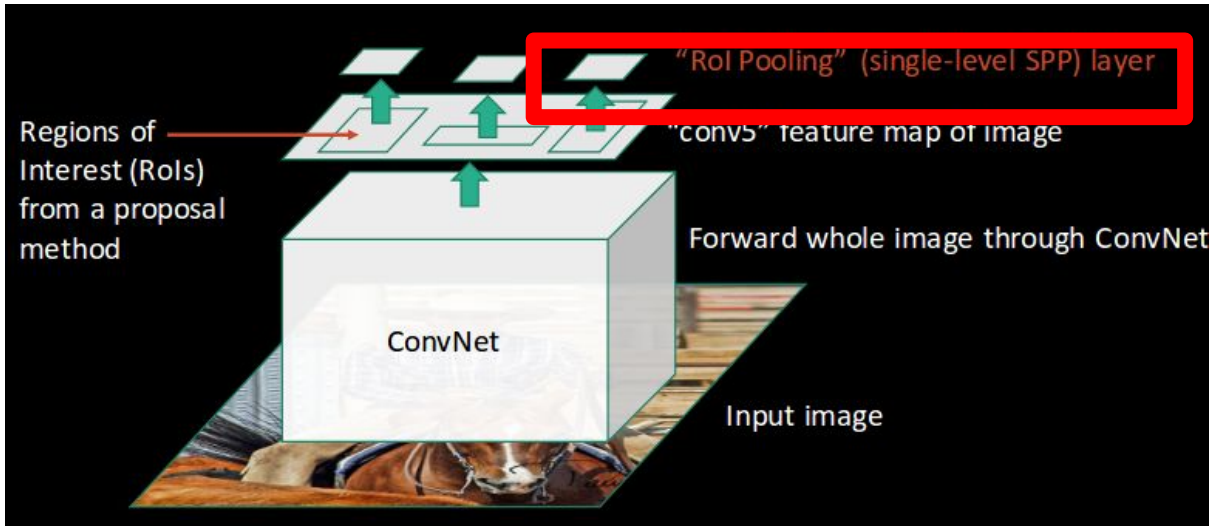
Fast R-CNN

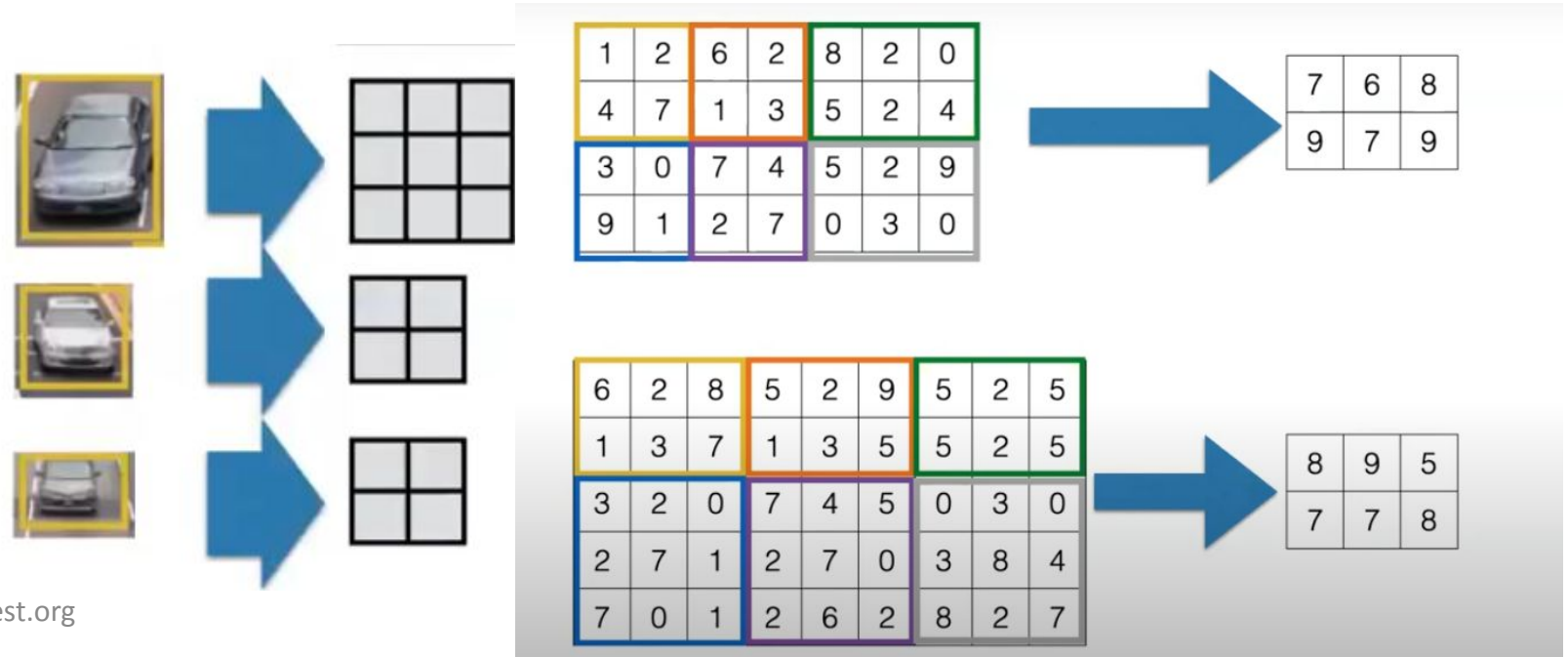

Input image


Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image


"RoI Pooling" (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image

ROI Pooling


"RoI Pooling" (single-level SPP) layer
"conv5" feature map of image
Regions of Interest (RoIs) from a proposal method
Forward whole image through ConvNet
ConvNet
Input image


1 2 6 2 8 2 0
4 7 1 3 5 2 4
3 0 7 4 5 2 9
9 1 2 7 0 3 0
7 6 8
9 7 9
6 2 8 5 2 9 5 2 5
1 3 7 1 3 5 5 2 5
3 2 0 7 4 5 0 3 0
2 7 1 2 7 0 3 8 4
7 0 1 2 6 2 8 2 7
8 9 5
7 7 8

ROI Pooling

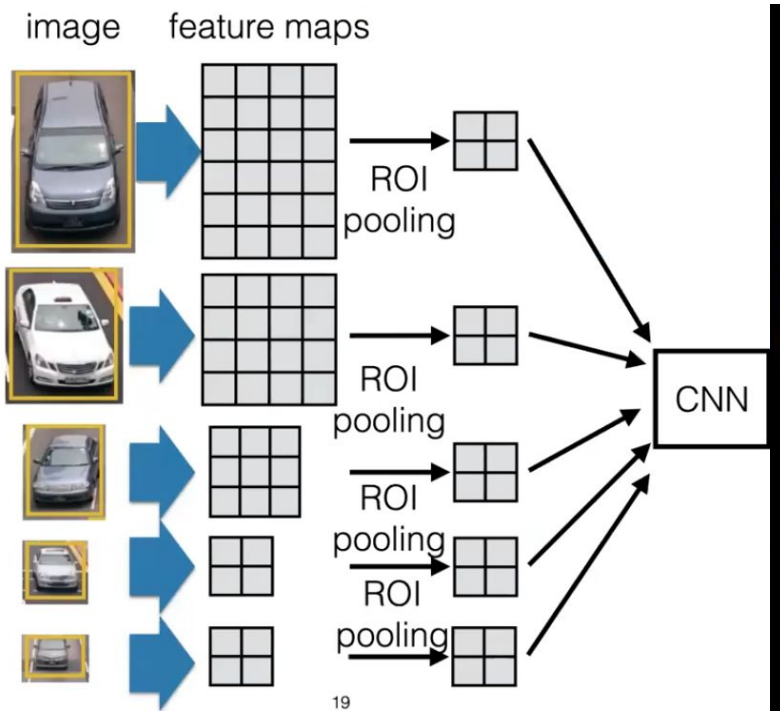
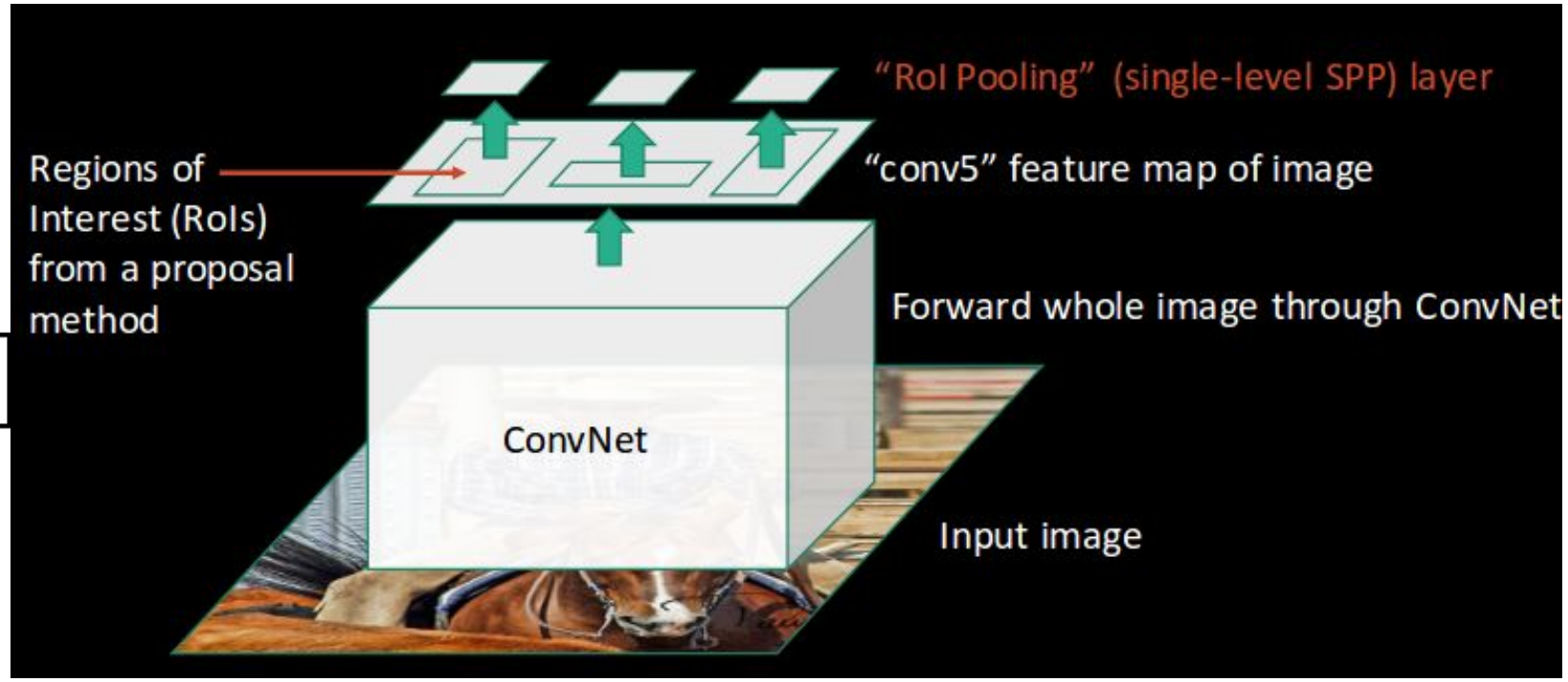
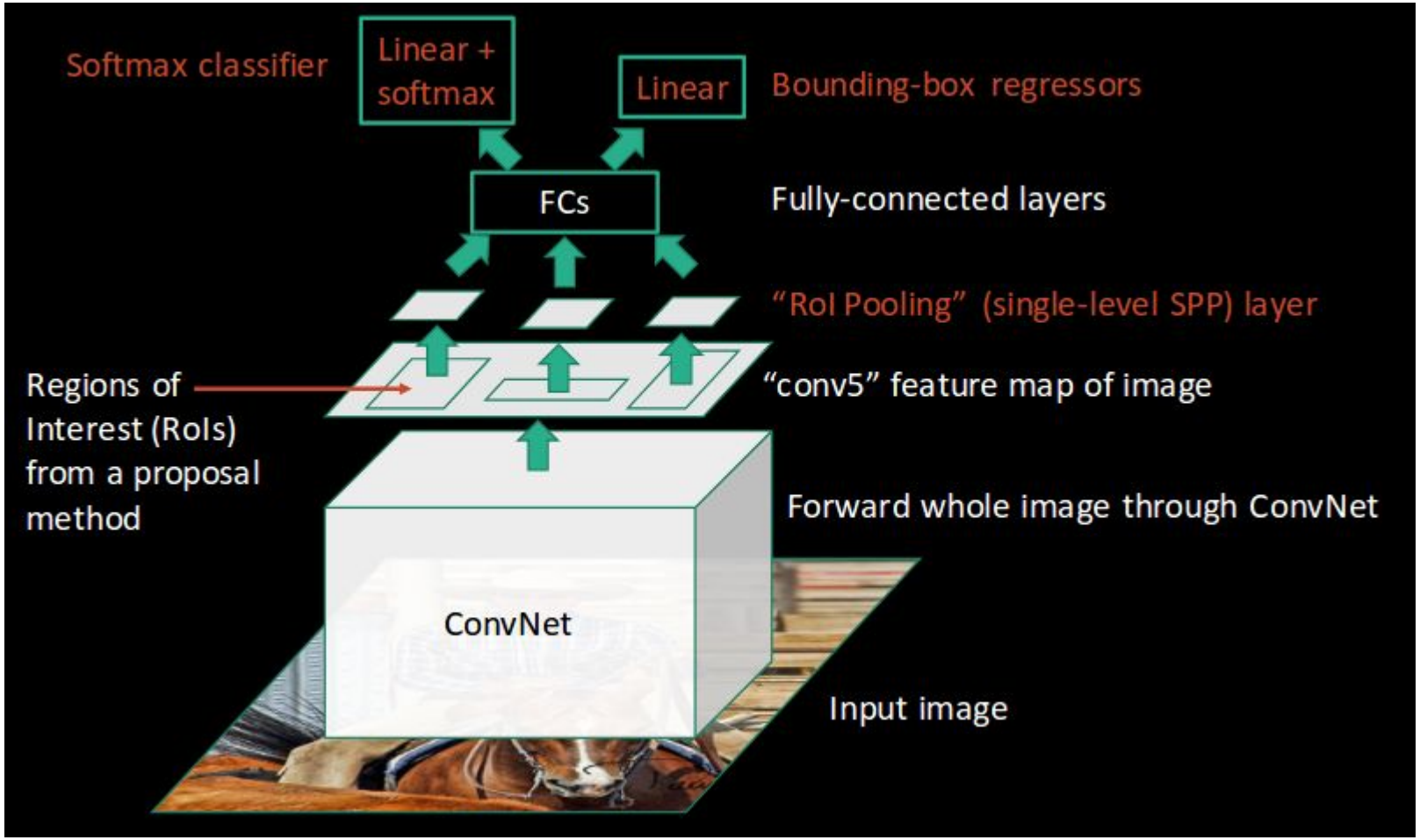


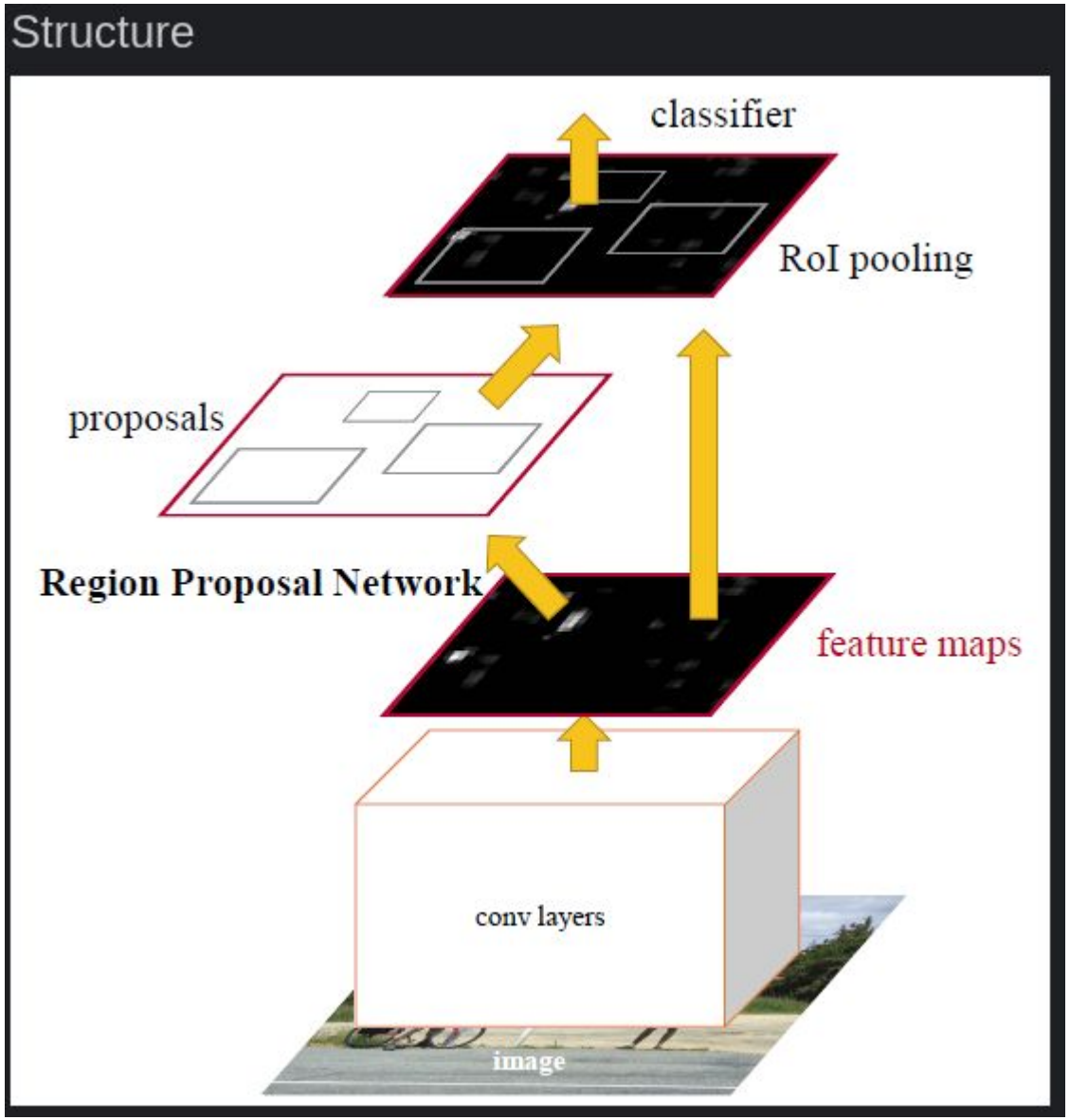
image
feature maps
ROI pooling
ROI pooling
ROI pooling
ROI pooling
CNN
19


"RoI Pooling" (single-level SPP) layer
"conv5" feature map of image
Regions of Interest (RoIs) from a proposal method
Forward whole image through ConvNet
ConvNet
Input image

Fast R-CNN


Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image

Faster R-CNN


Structure
classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image

Overview

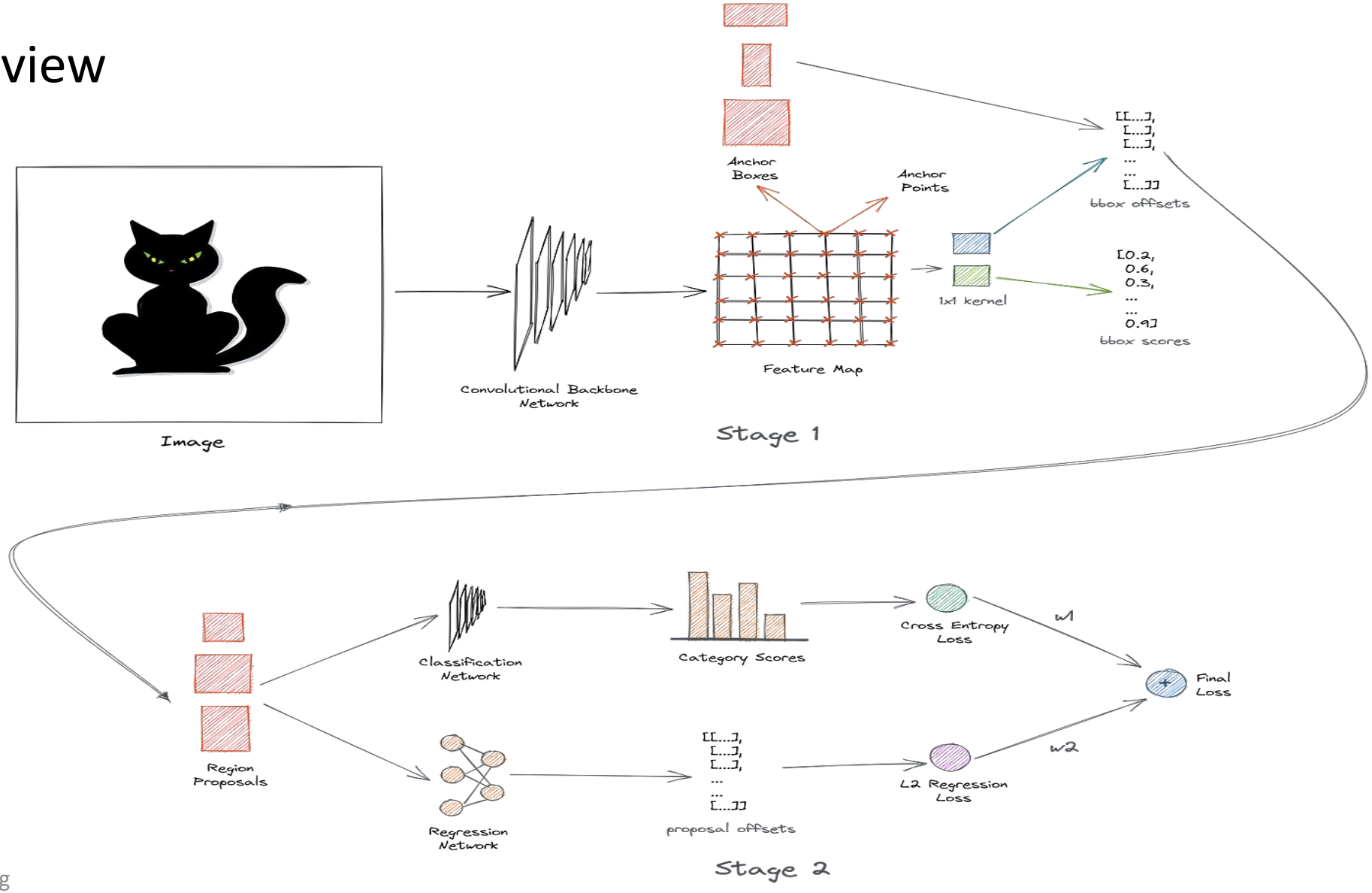
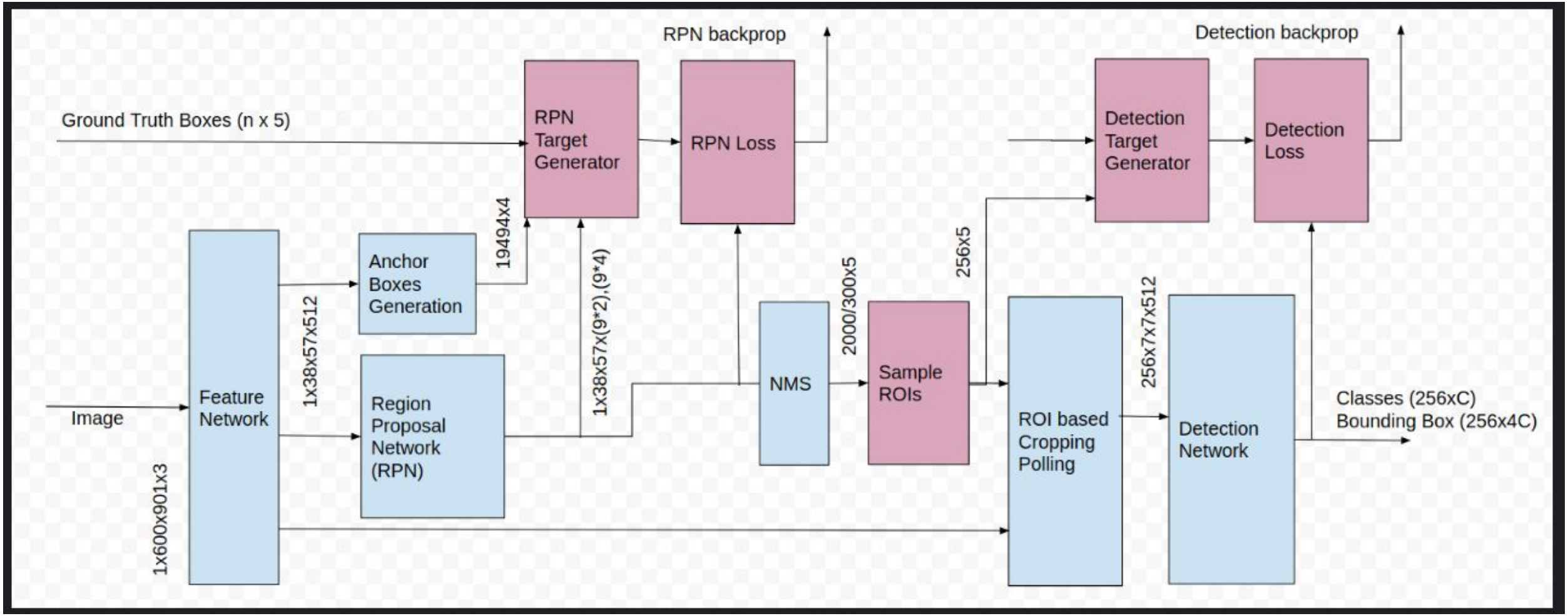


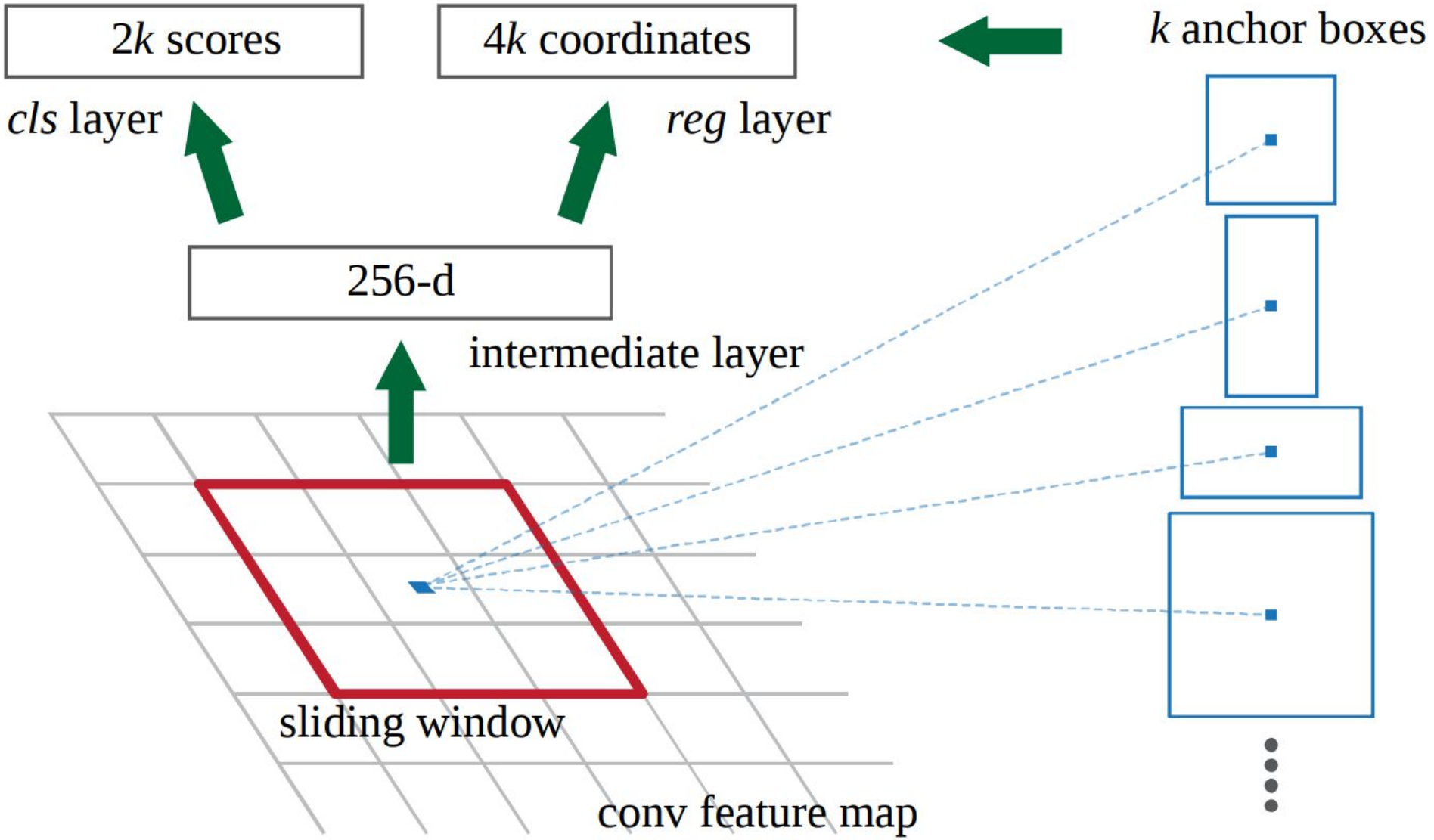
Image
Convolutional Backbone Network
Anchor Boxes
Anchor Points
Feature Map
1x1 kernel
[[...],
[...],
[...],
...
...
[...]]
bbox offsets
[0.2,
0.6,
0.3,
...
...
0.9]
bbox scores
Stage 1
Region Proposals
Classification Network
Category Scores
Cross Entropy Loss
w1
Final Loss
Regression Network
[[...],
[...],
[...],
...
...
[...]]
proposal offsets
L2 Regression Loss
w2
Stage 2

Architecture



Faster R-CNN

RPN



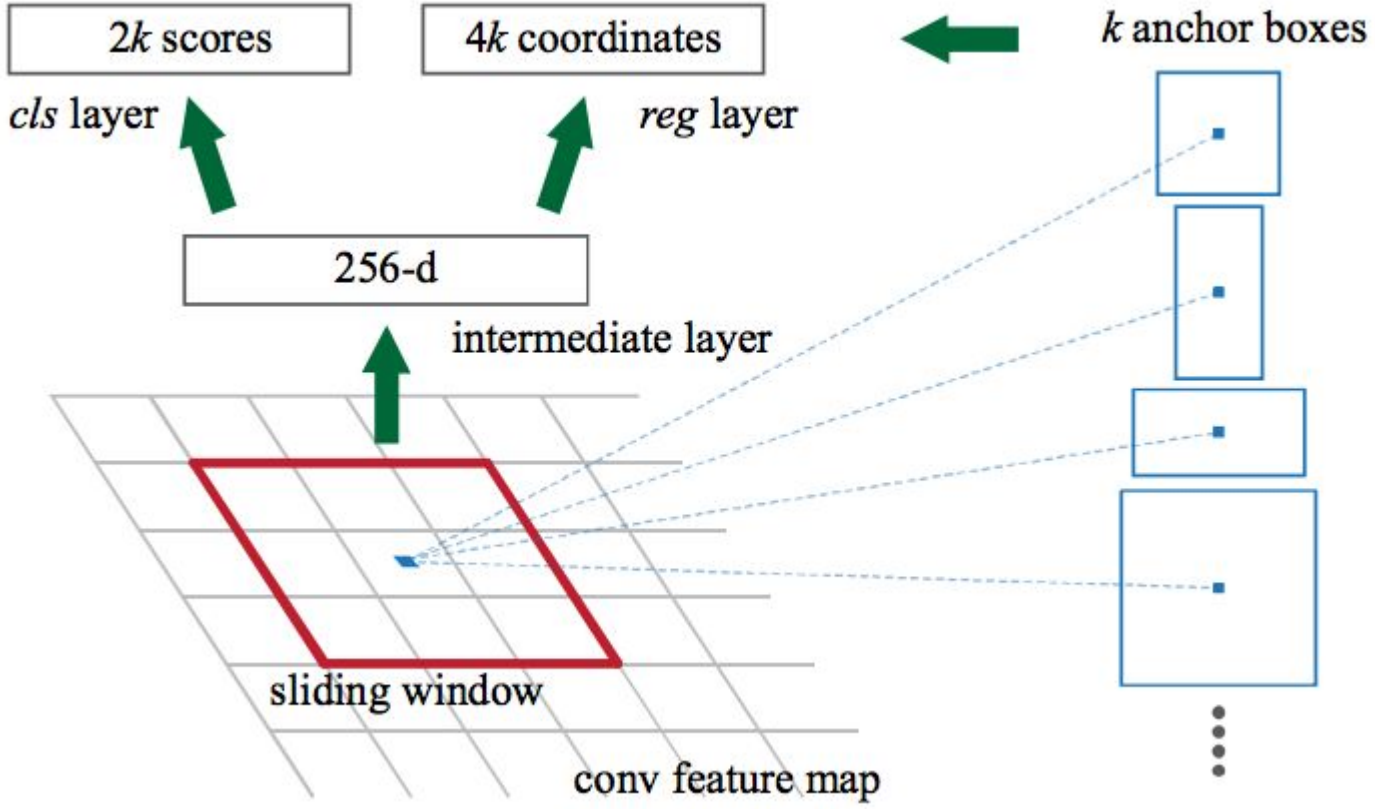
Faster R-CNN

To begin with, Faster RCNN takes the feature maps from CNN and passes them on to the Region Proposal Network. RPN uses a sliding window over these feature maps, and at each window, it generates *k* Anchor boxes of different shapes and sizes:

Anchor boxes are fixed-sized boundary boxes that are placed throughout the image and have different shapes and sizes. For each anchor, RPN predicts two things:

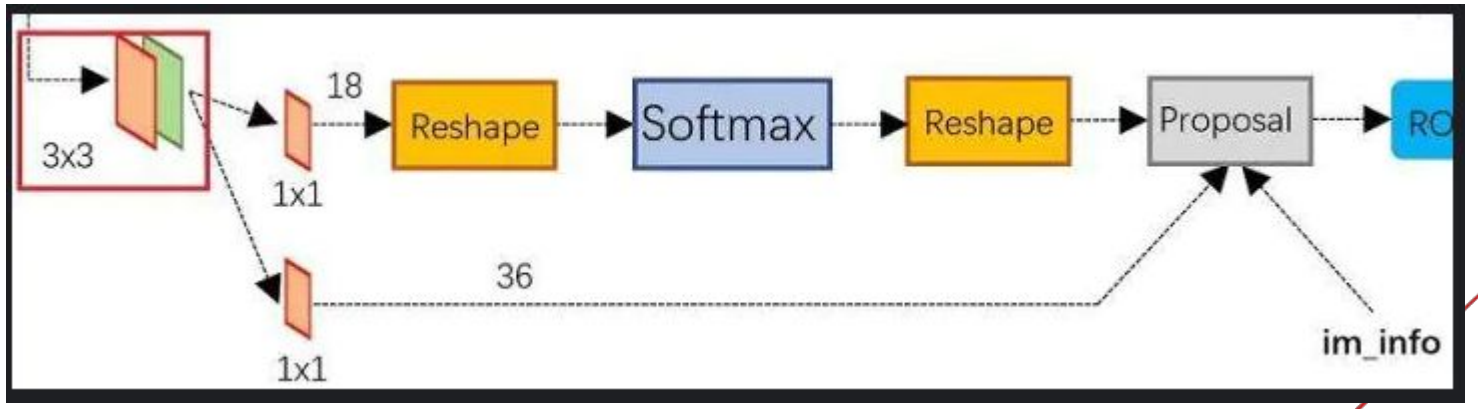
- The first is the probability that an anchor is an object (it does not consider which class the object belongs to)
- Second is the bounding box regressor for adjusting the anchors to better fit the object

RPN



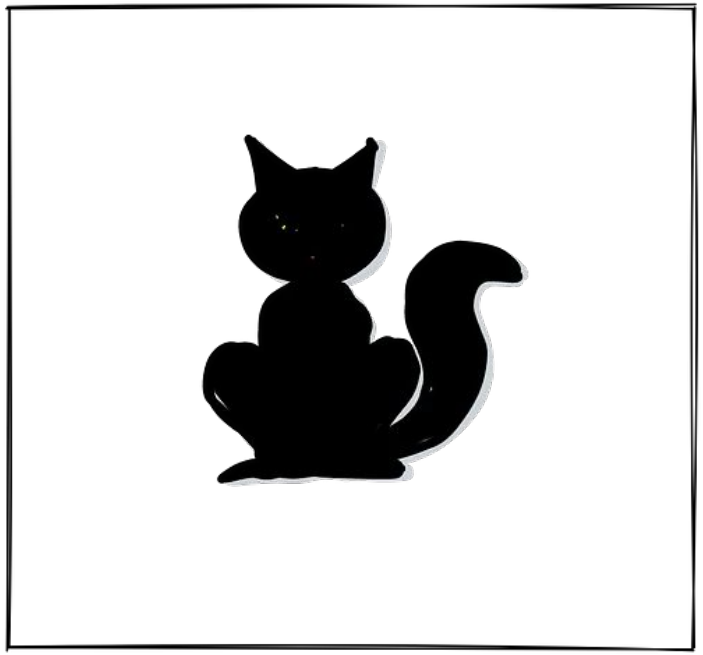
Faster R-CNN

RPN



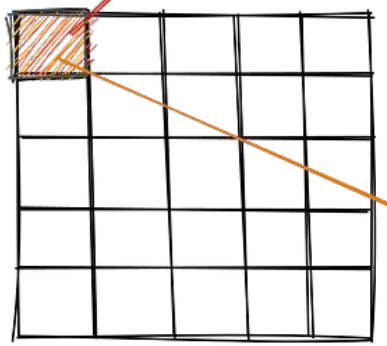
[[...],
[...],
[...],
...
...
[...]]

bbox offsets



Image

1x1 kernel

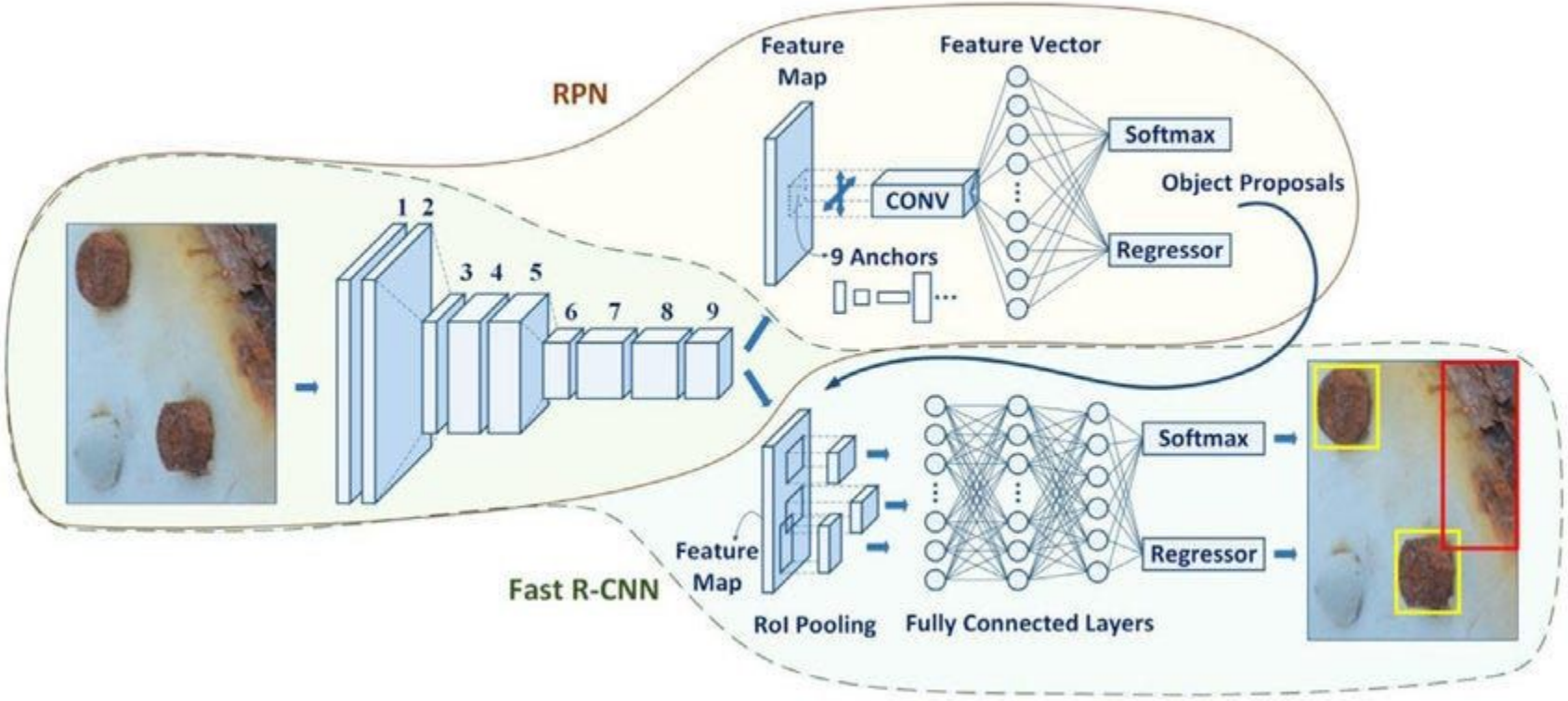


Feature Map

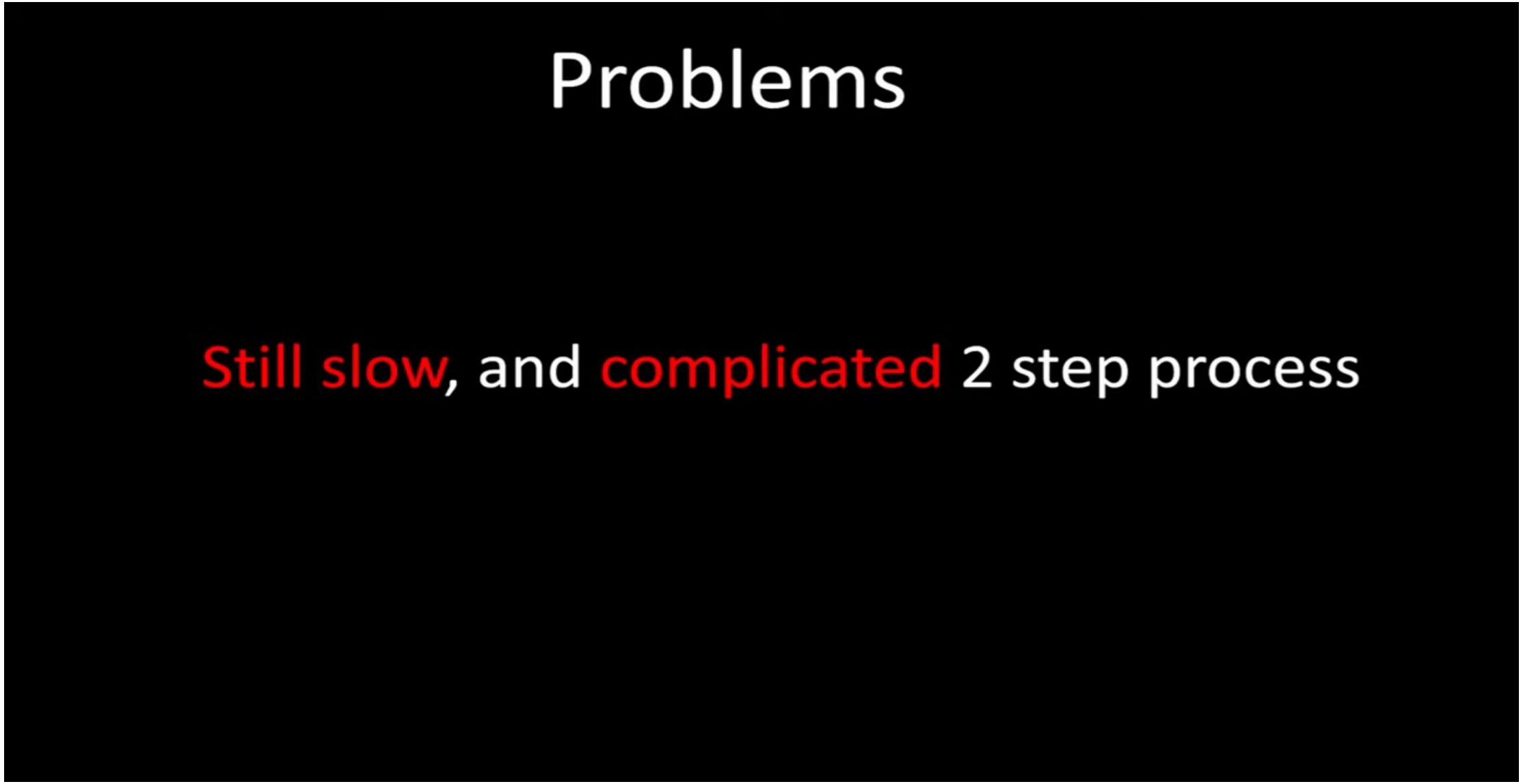
[0.2,
0.6,
0.3,
...
...
0.9]

bbox scores

Faster R-CNN


RPN
Feature Map
Feature Vector
CONV
Softmax
Object Proposals
Regressor
9 Anchors
1 2
3 4 5
6 7 8 9
Fast R-CNN
Feature Map
RoI Pooling
Fully Connected Layers
Softmax
Regressor

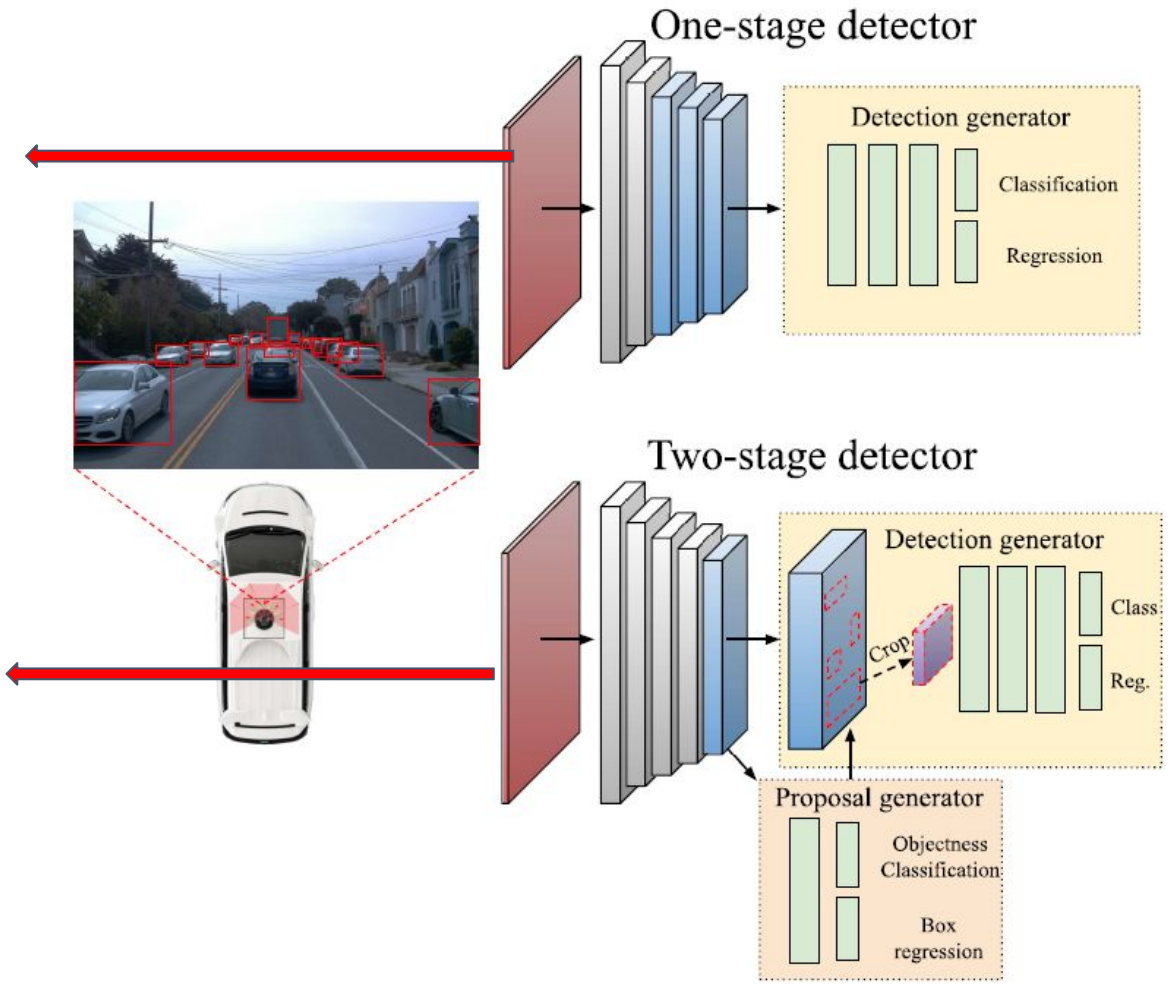
Regional Based Detection Model Problems

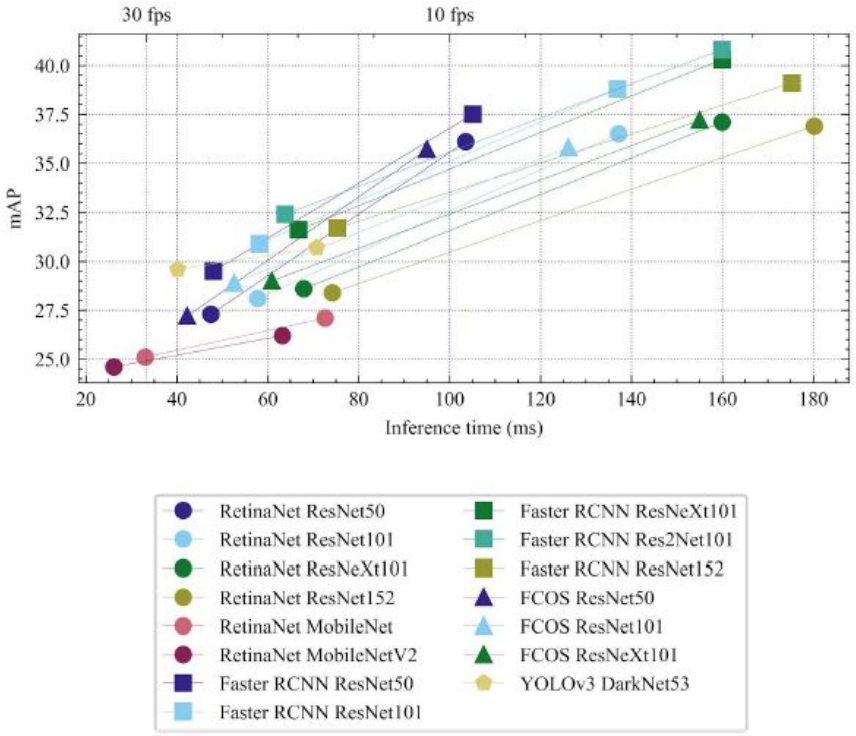


Object Detection types

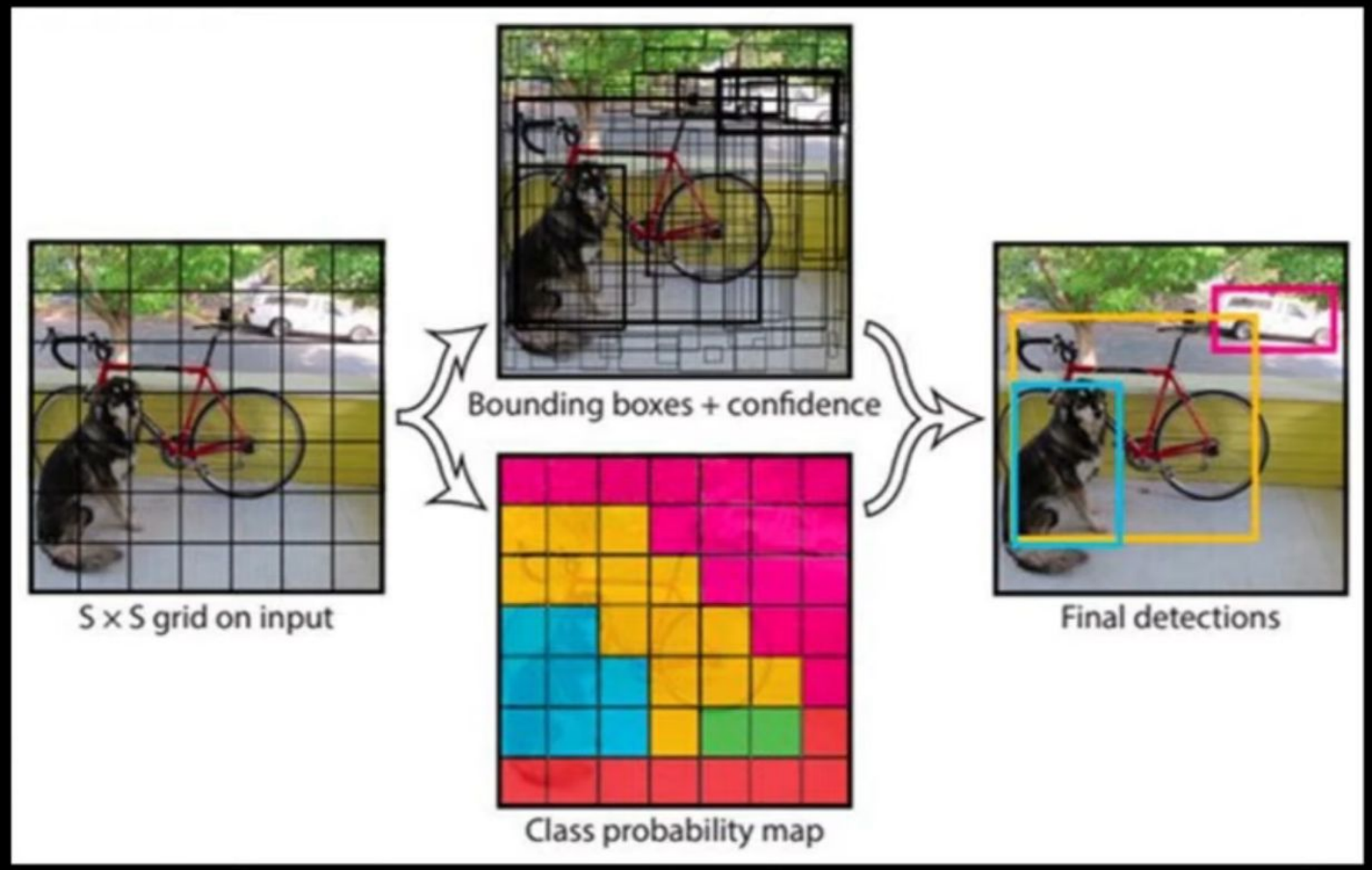
Yolo, SSD etc.

R-CNN,Fast R-CNN etc.



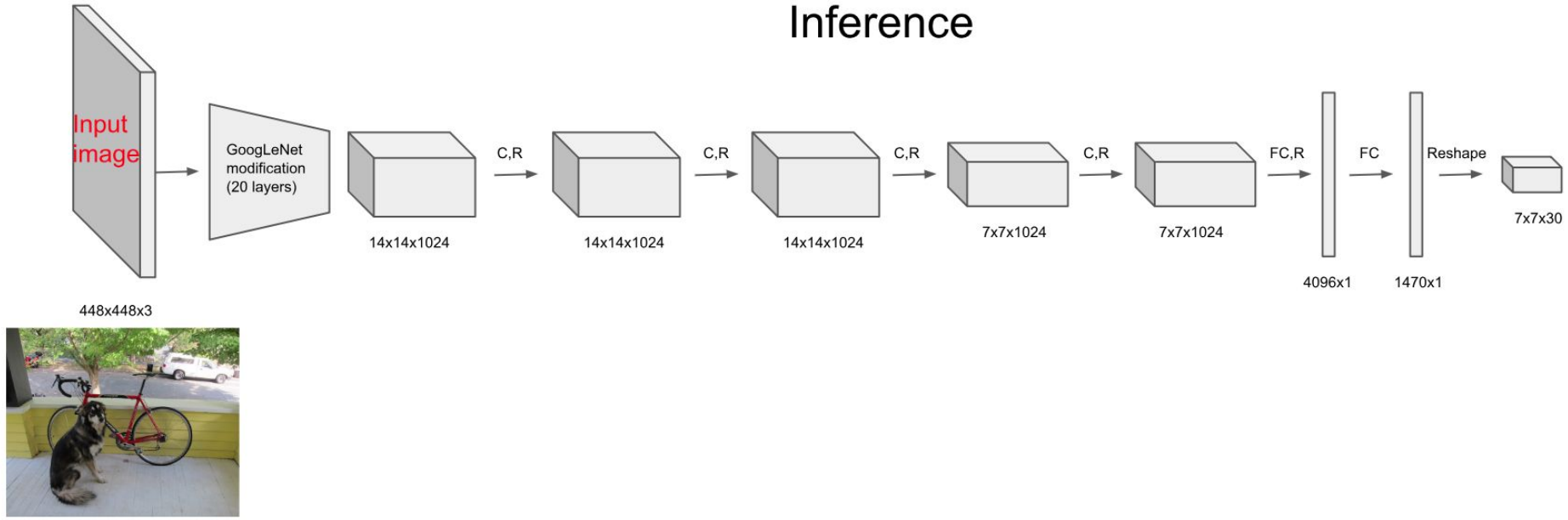


Yolo – You Only Look Once

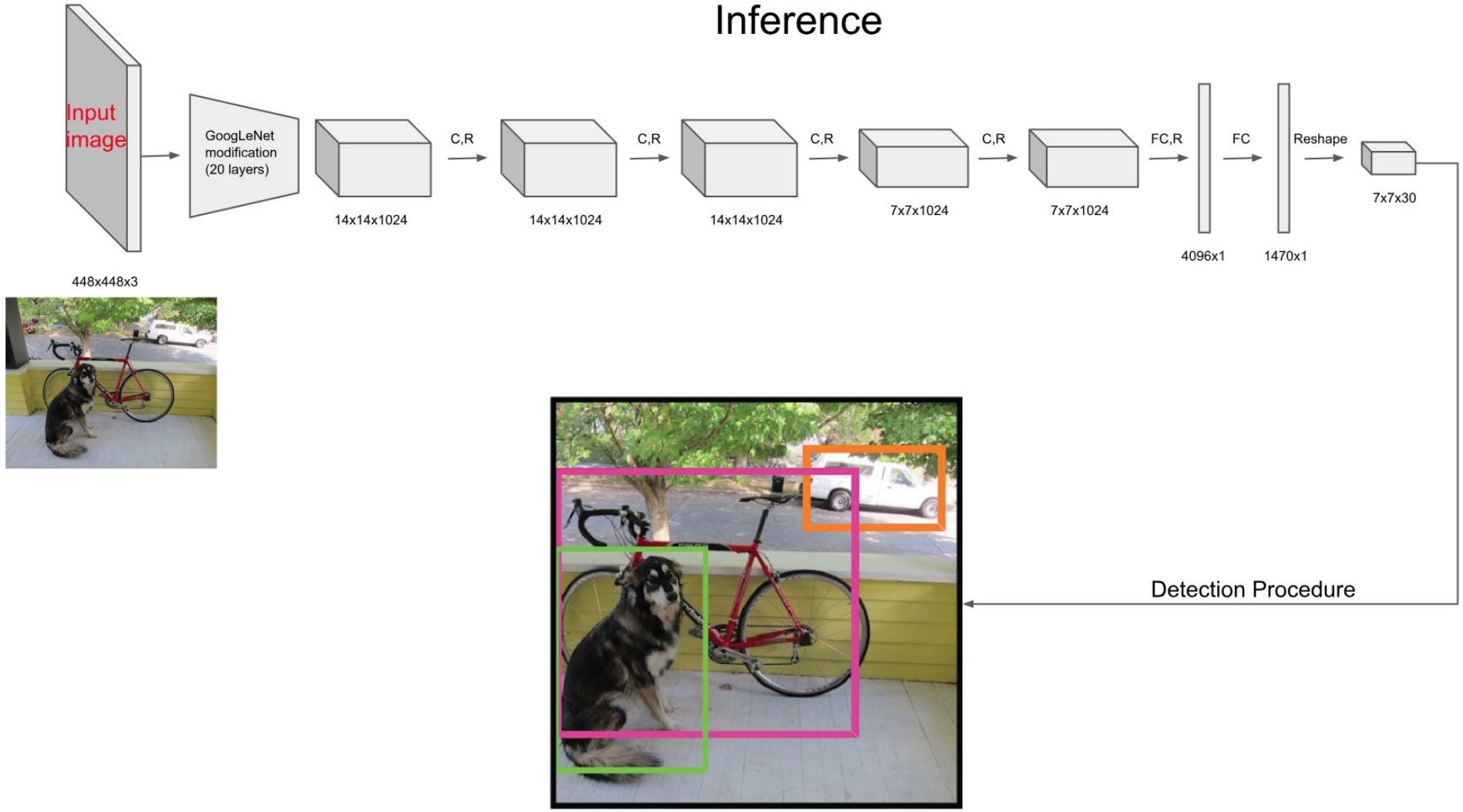


4 Papers:
Yolo (v1)
Yolo (v2)
Yolo (v3)
Yolo (v4)

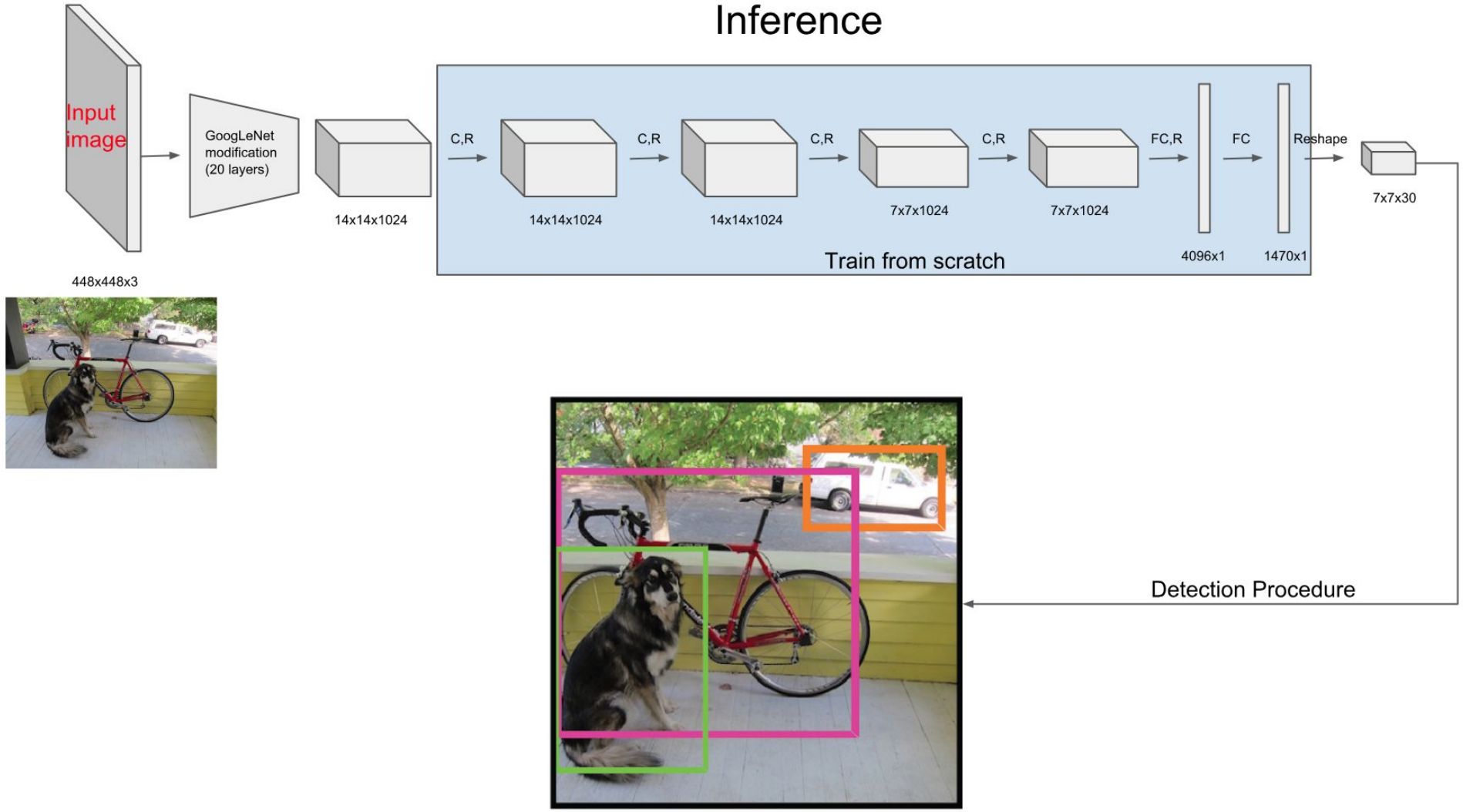
YOLO


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

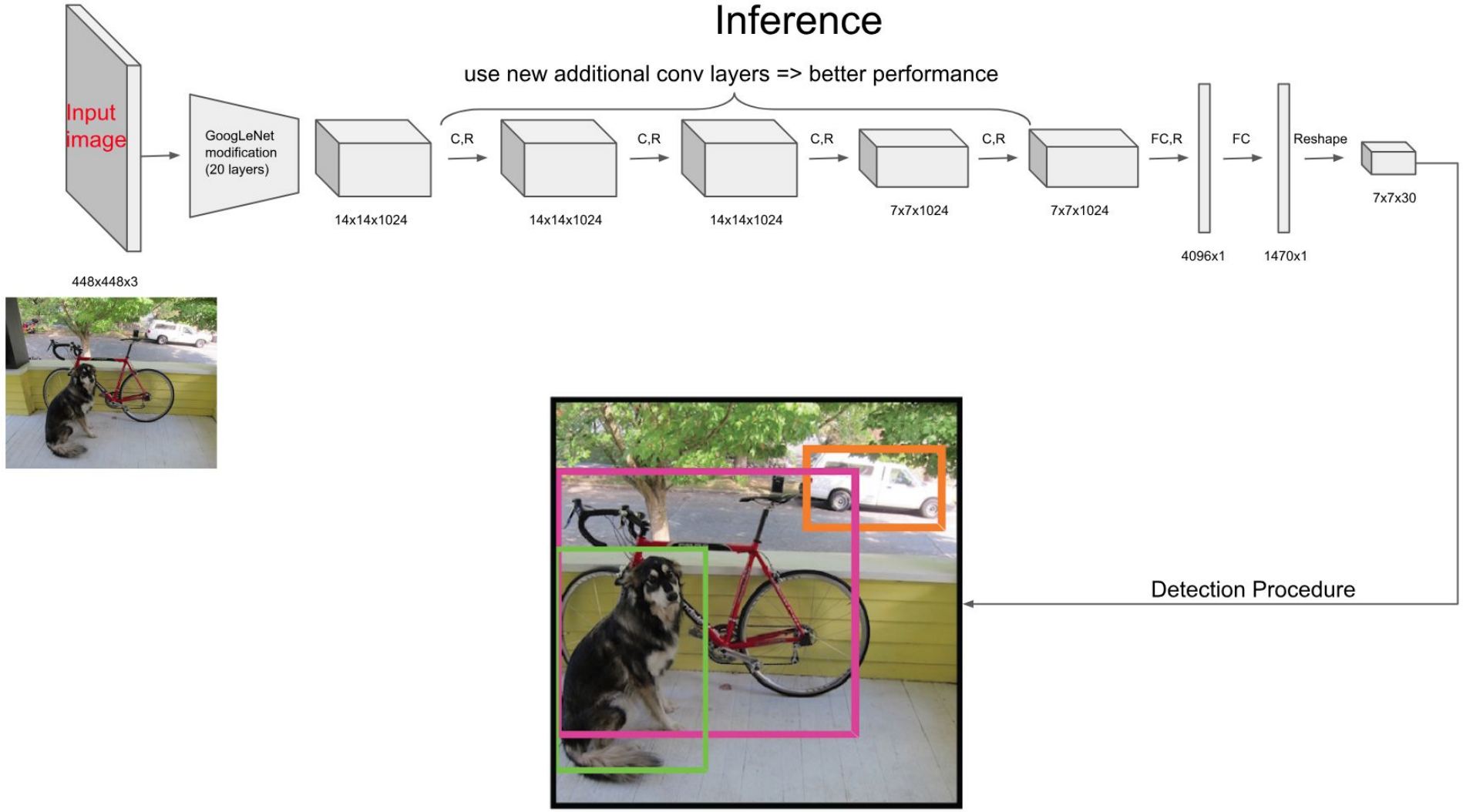
YOLO


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

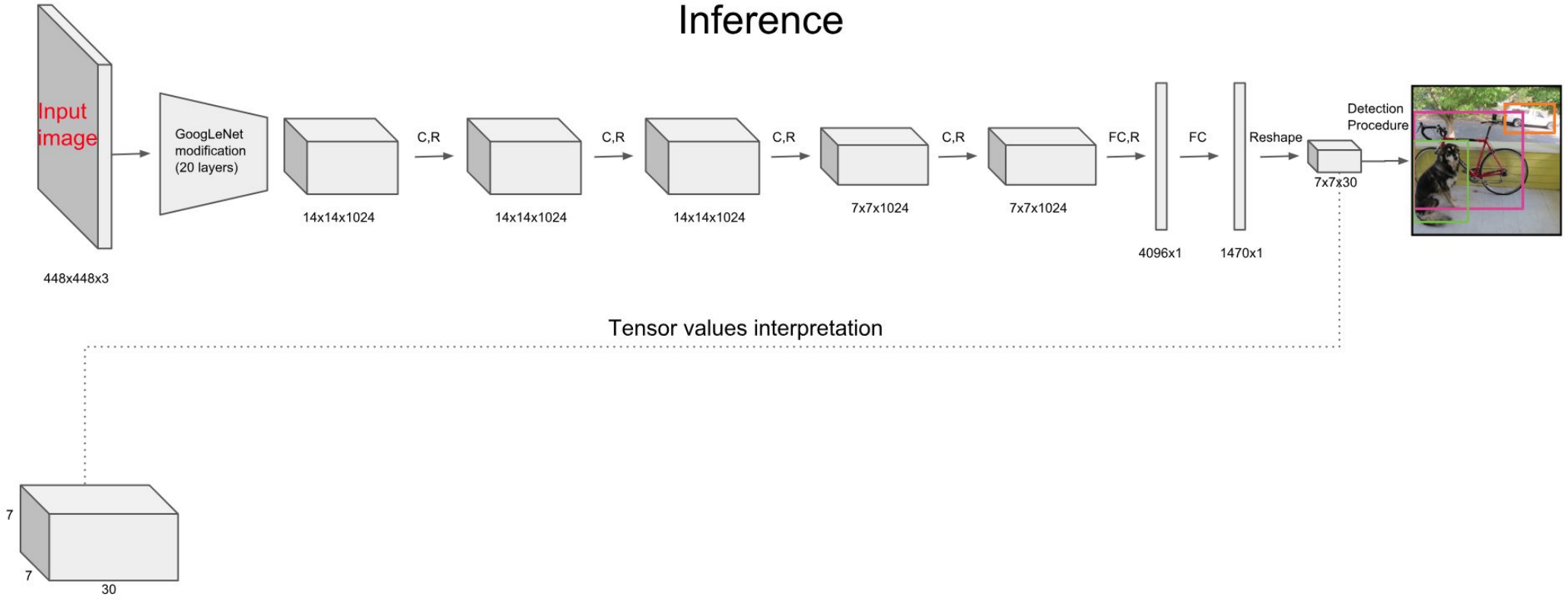
YOLO


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

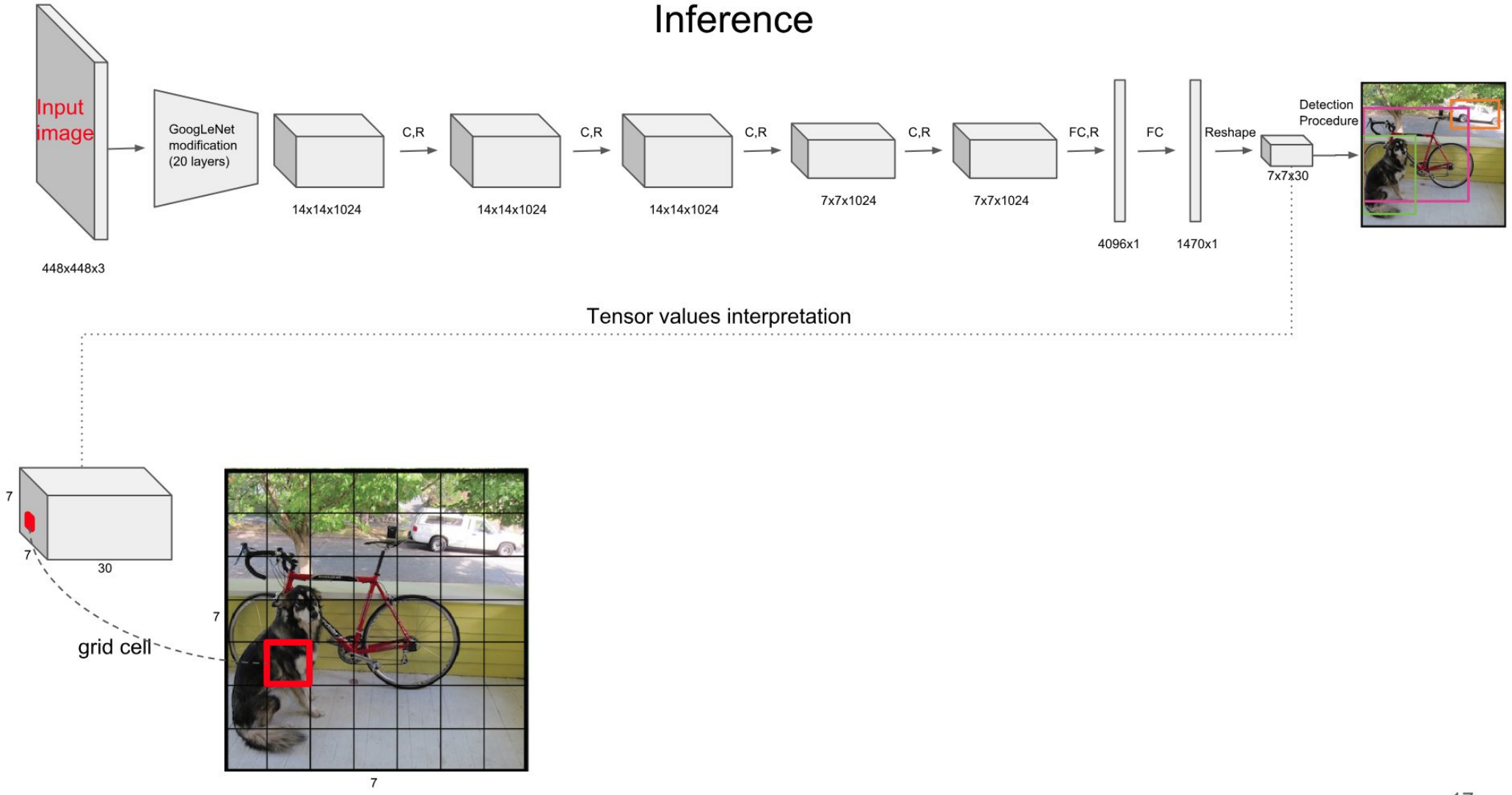
YOLO


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

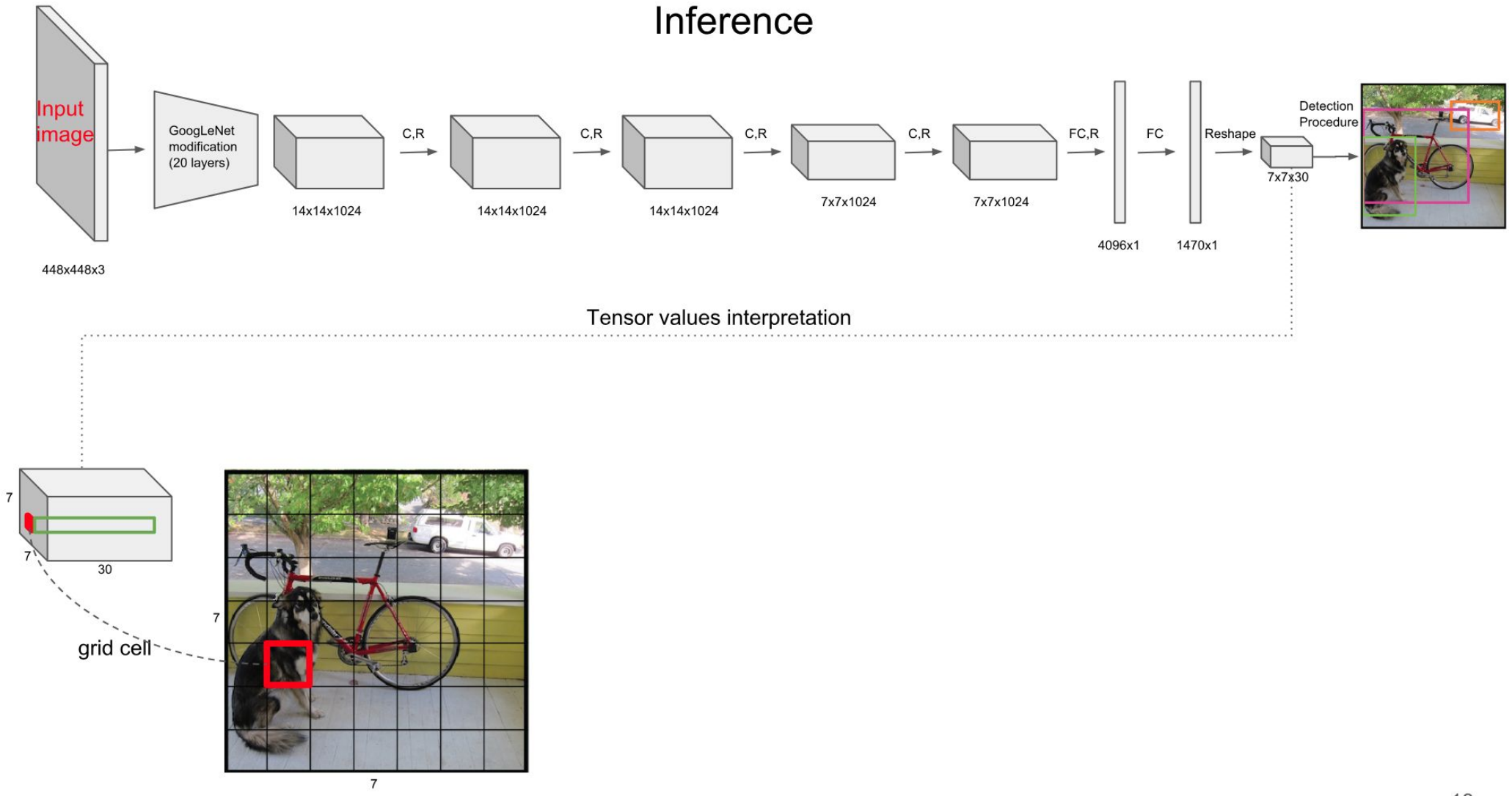
YOLO


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

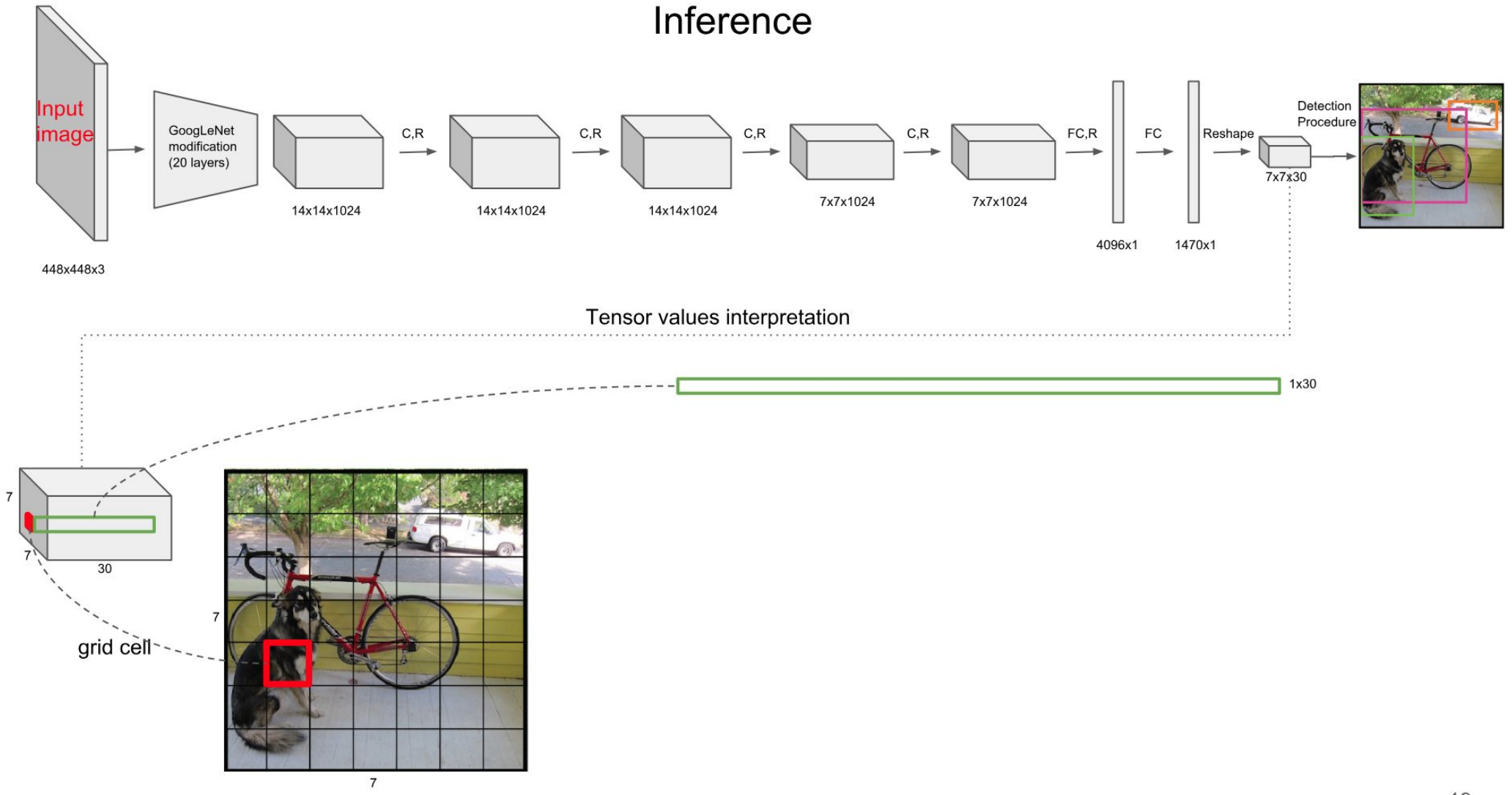
YOLO

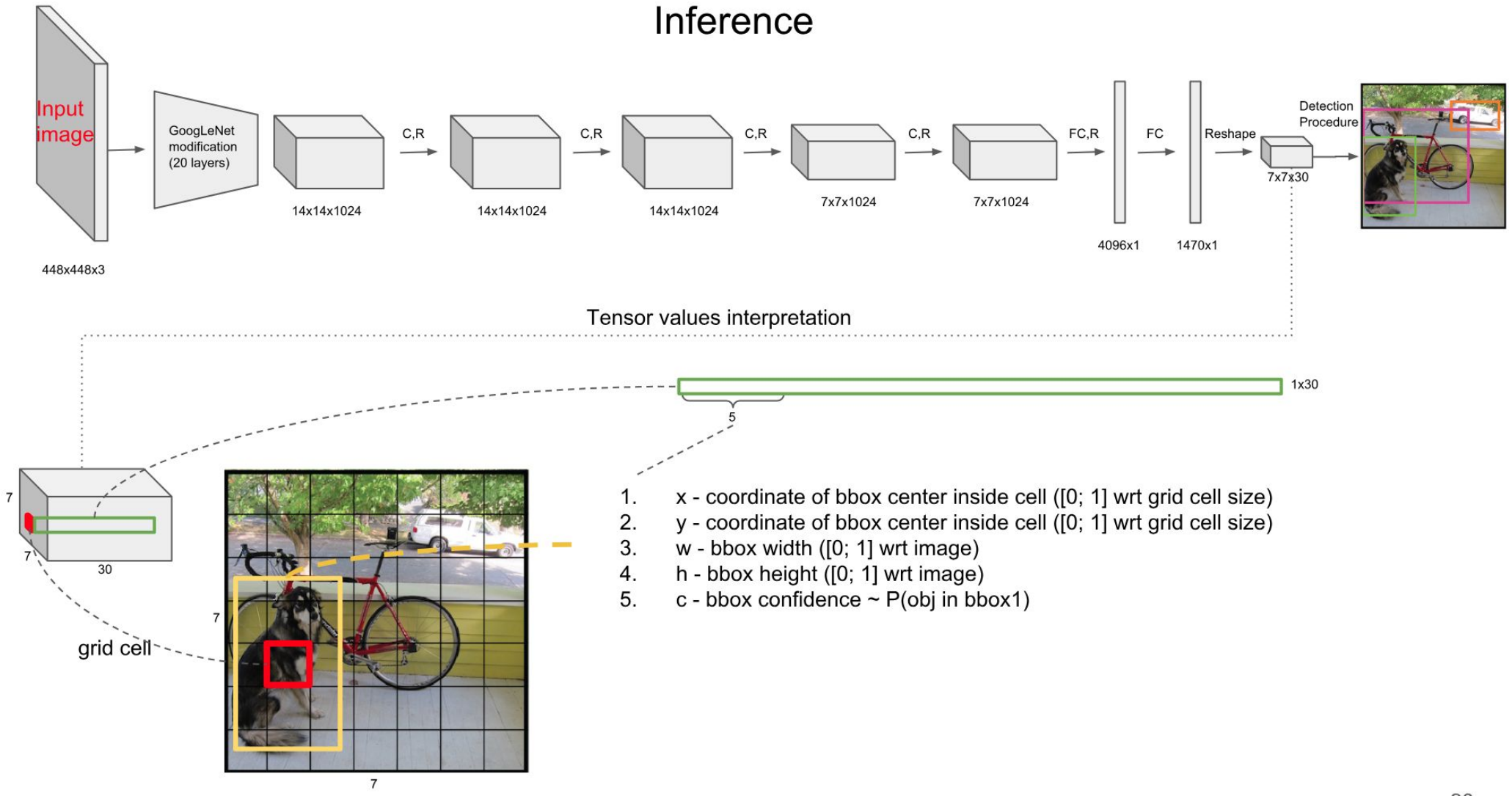

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
7
7
30
grid cell
7
7

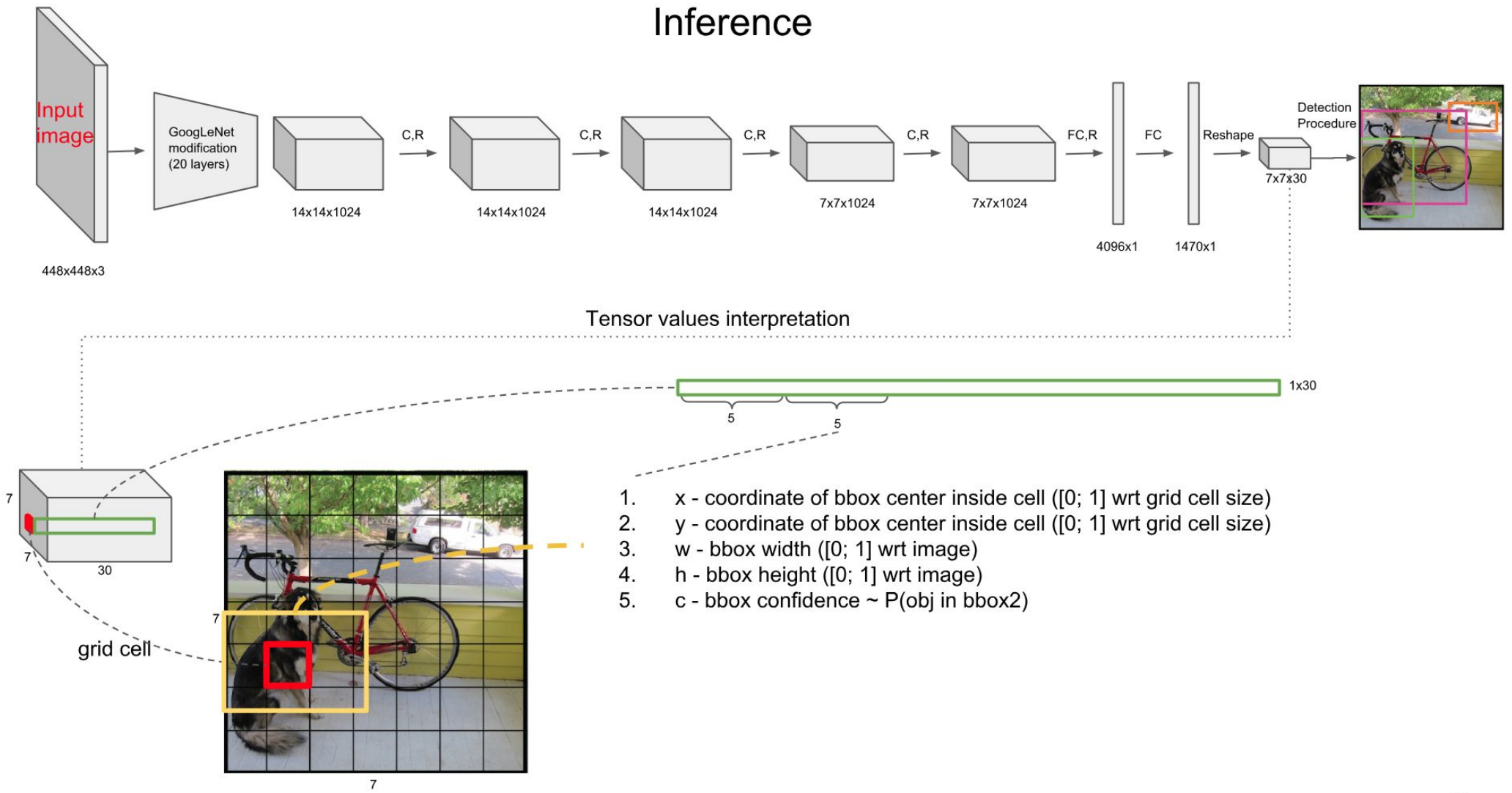
YOLO


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

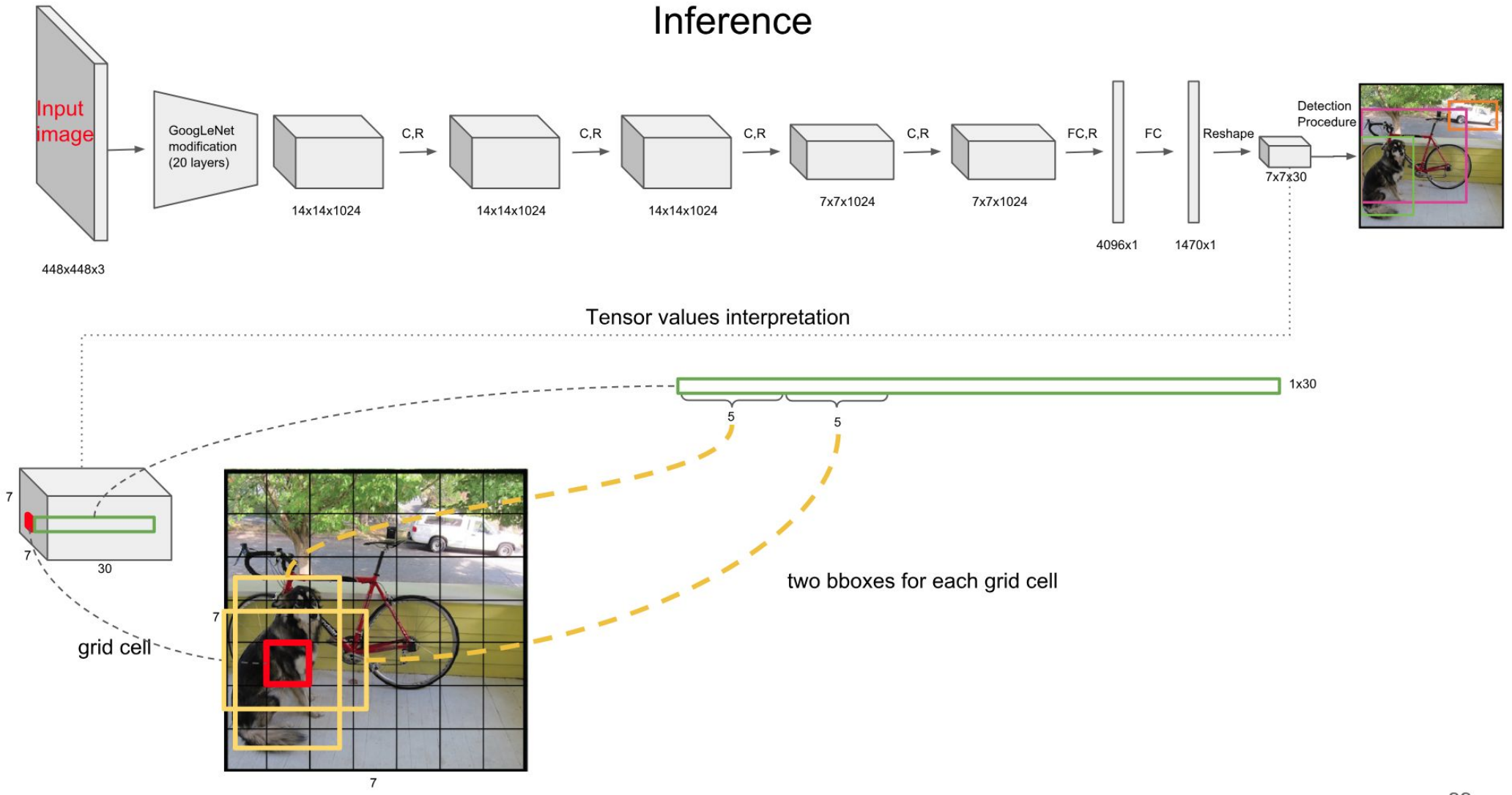
YOLO


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

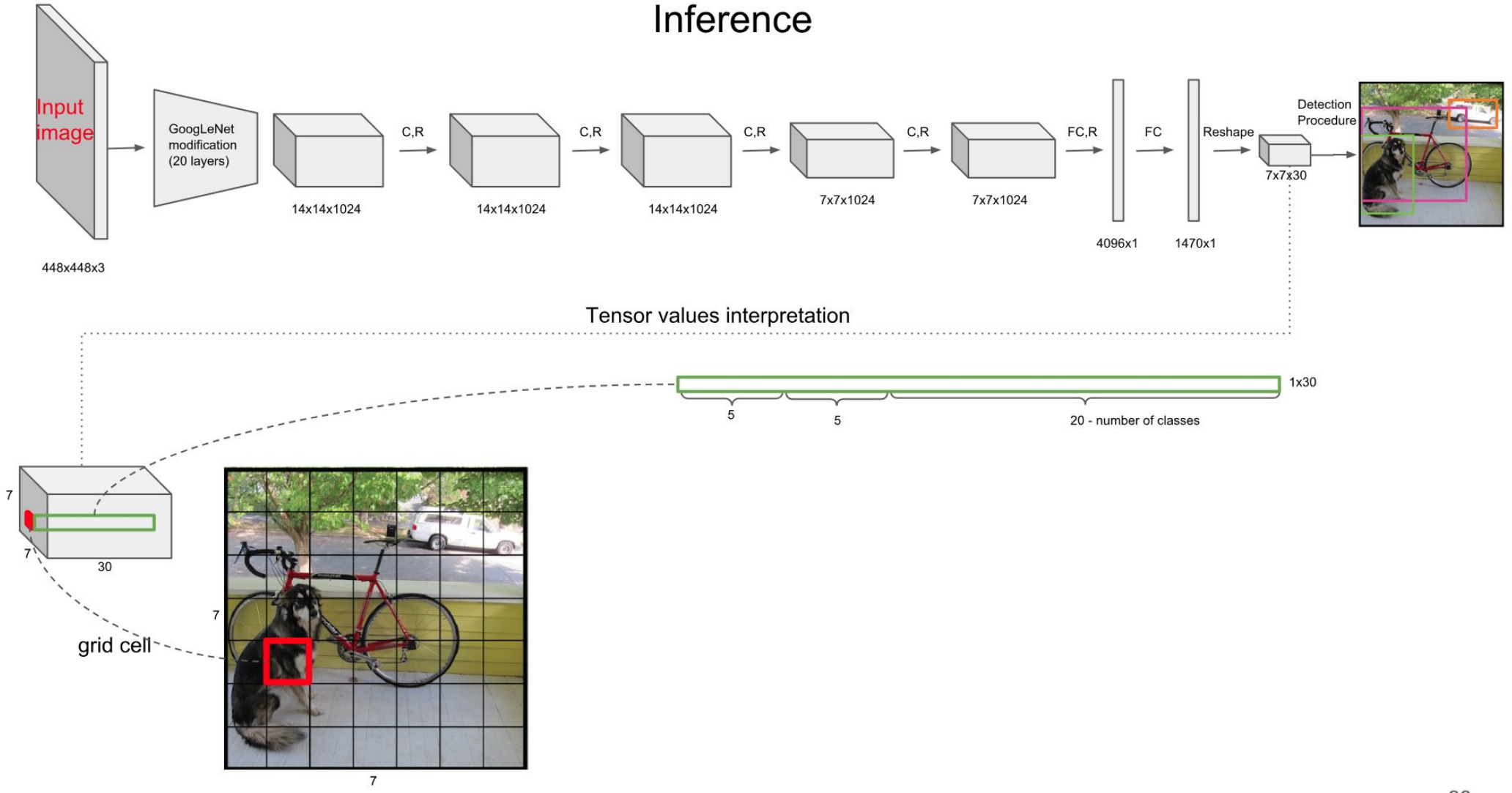

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
1x30
5
7
7
30
grid cell
7
7
1. x - coordinate of bbox center inside cell ([0; 1] 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)

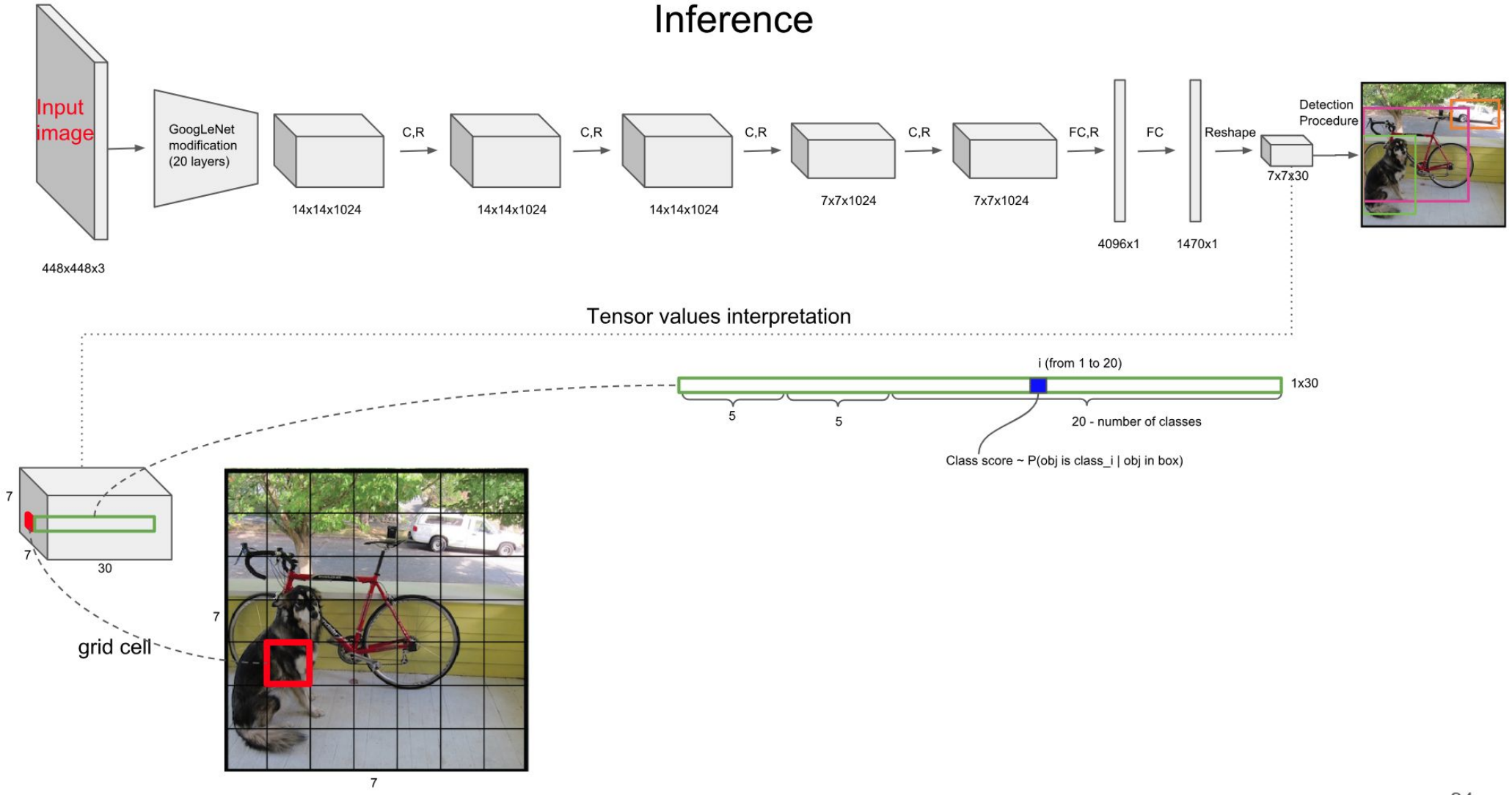
YOLO


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

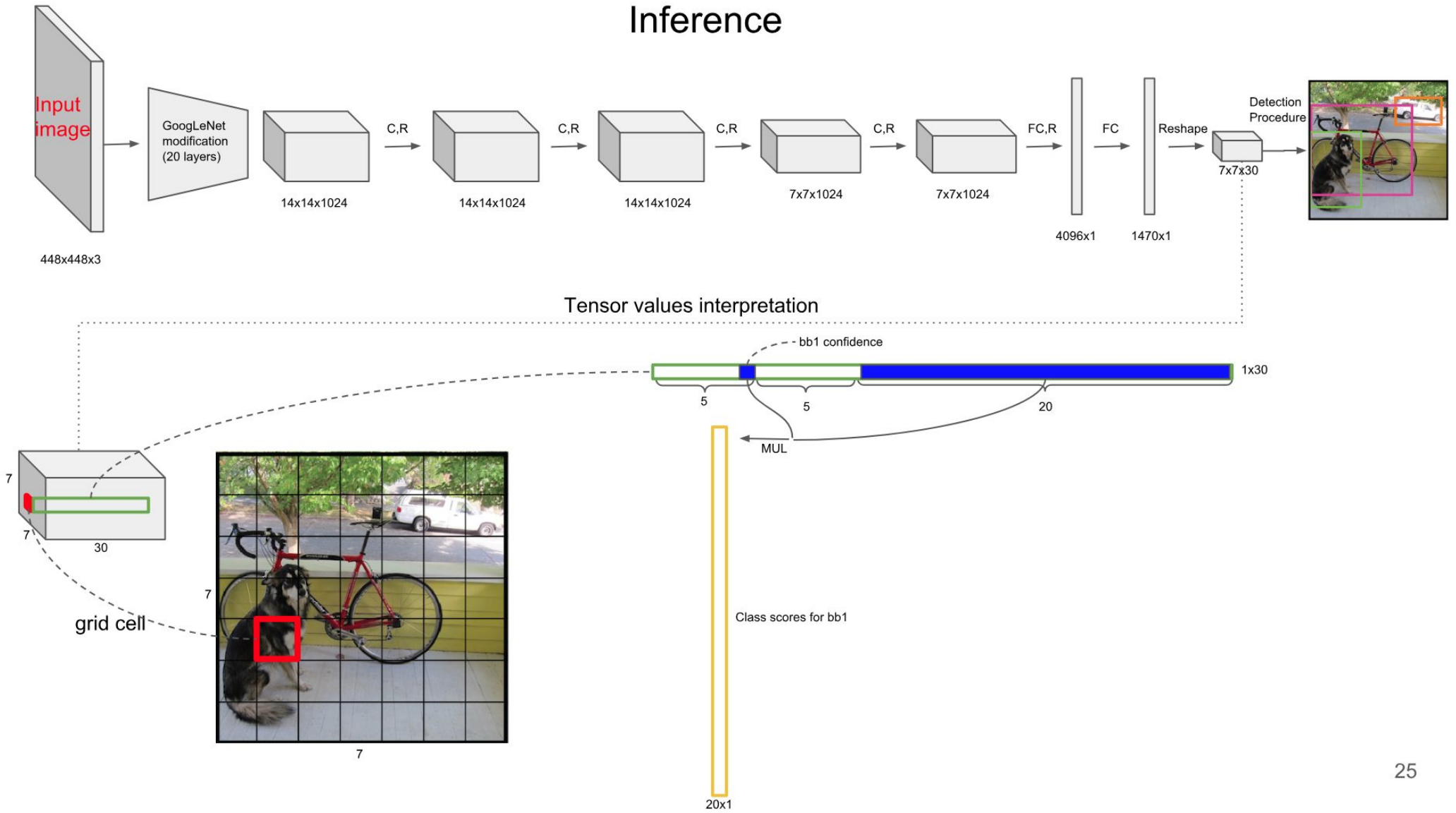
YOLO


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

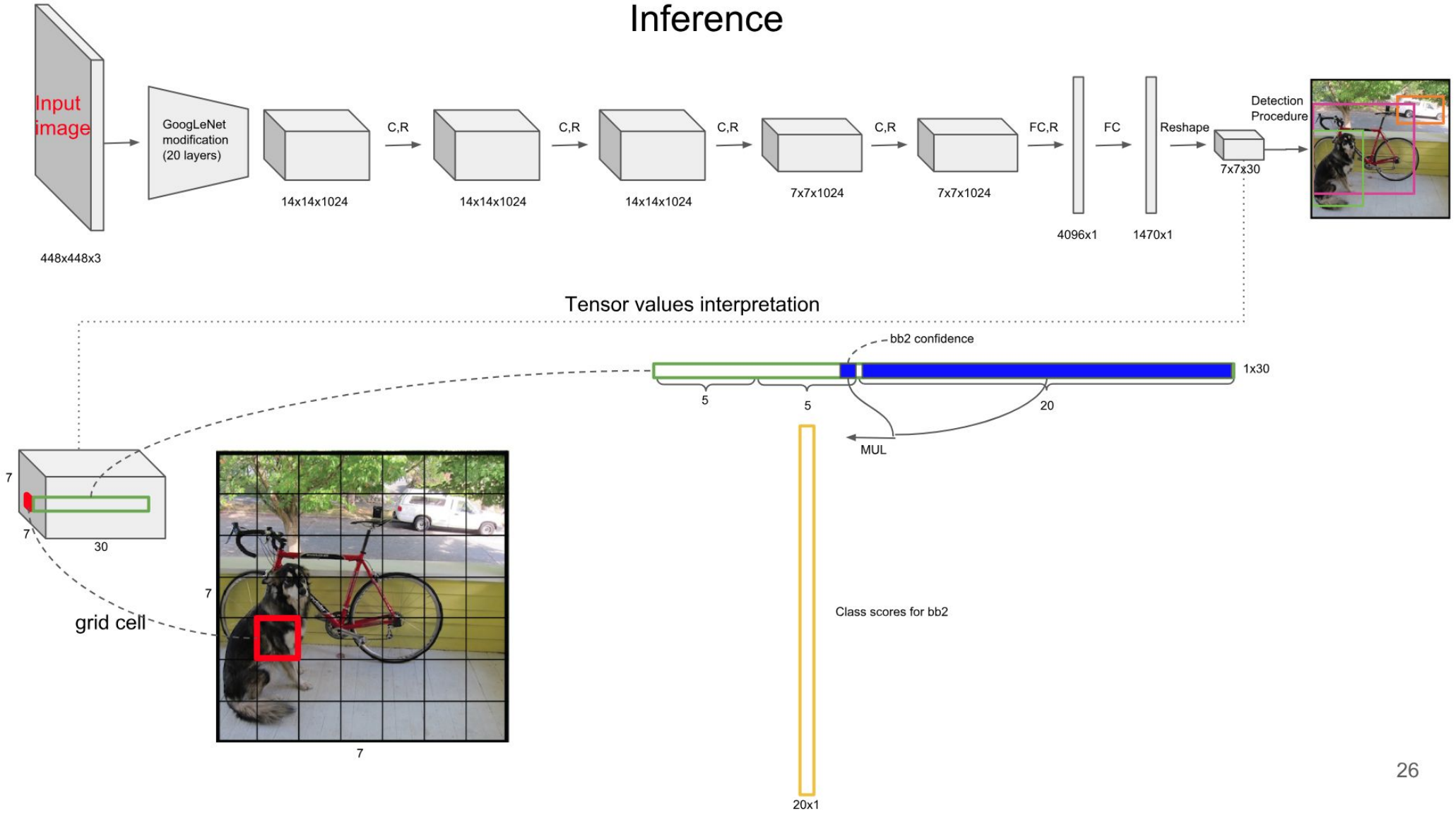
YOLO

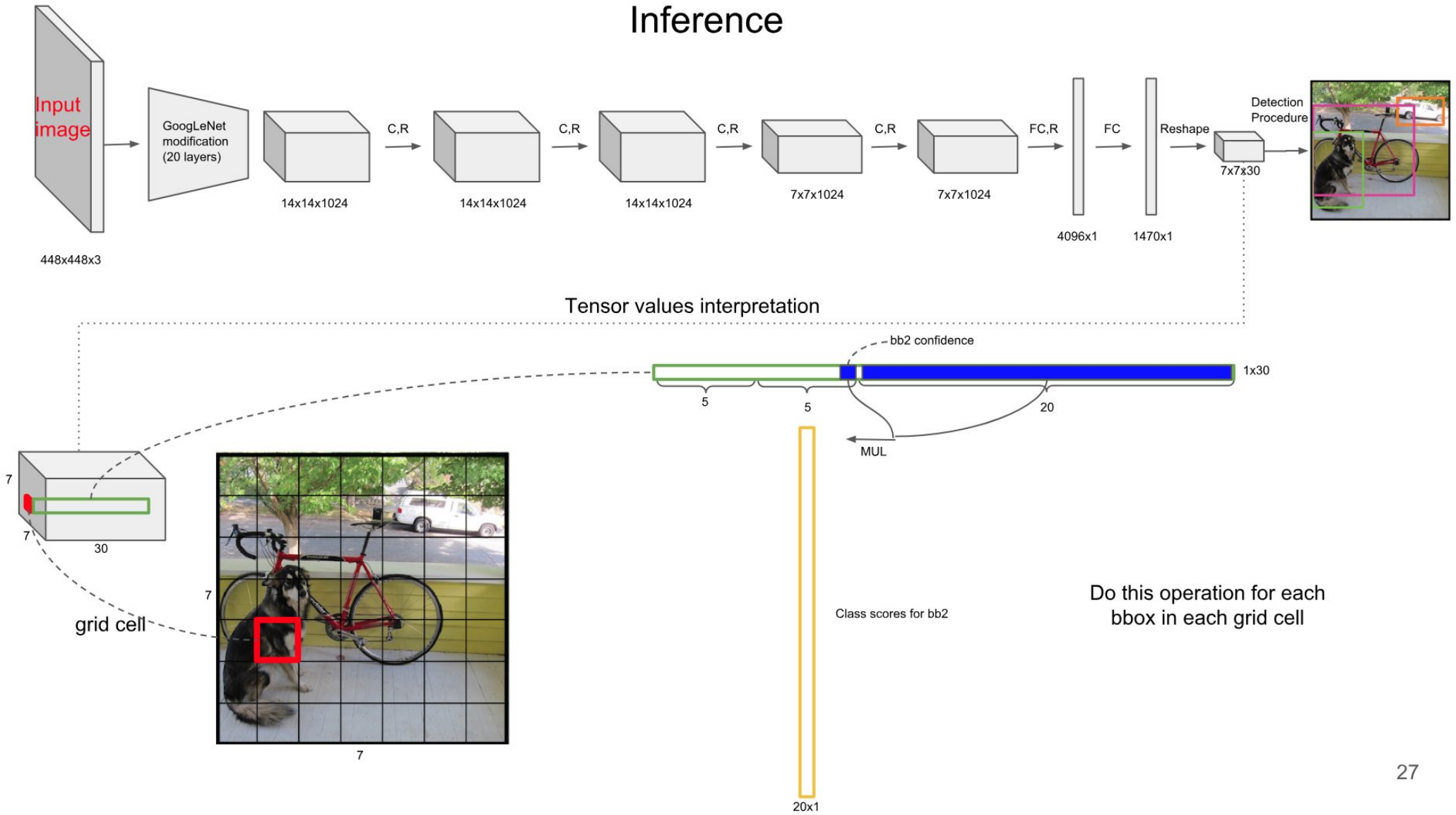

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
i (from 1 to 20)
1x30
5
5
20 - number of classes
Class score ~ P(obj is class_i | obj in box)
7
7
30
grid cell
7
7

YOLO

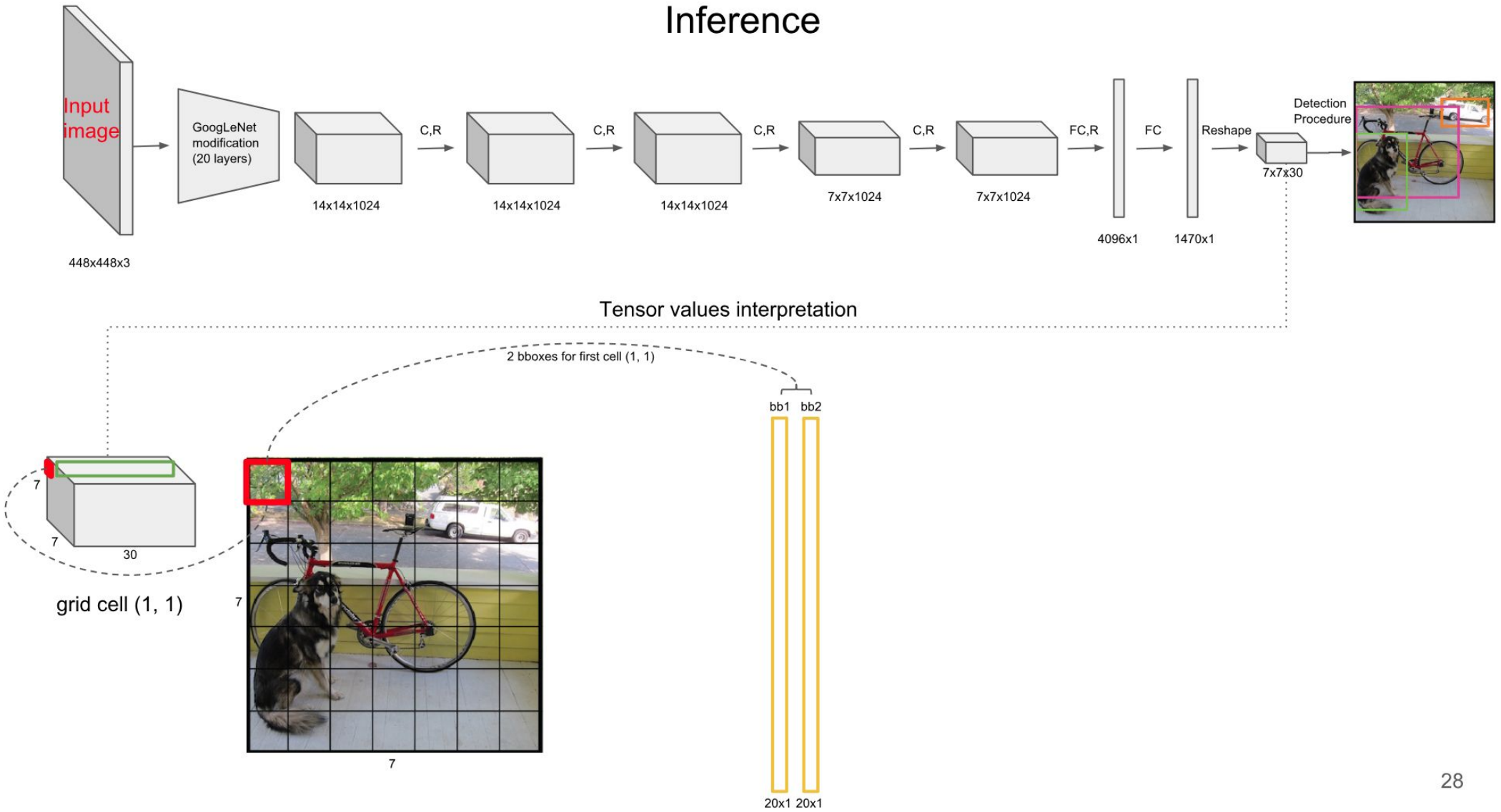

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
bb1 confidence
1x30
5
5
20
MUL
7
7
30
grid cell
7
7
Class scores for bb1
20x1

YOLO

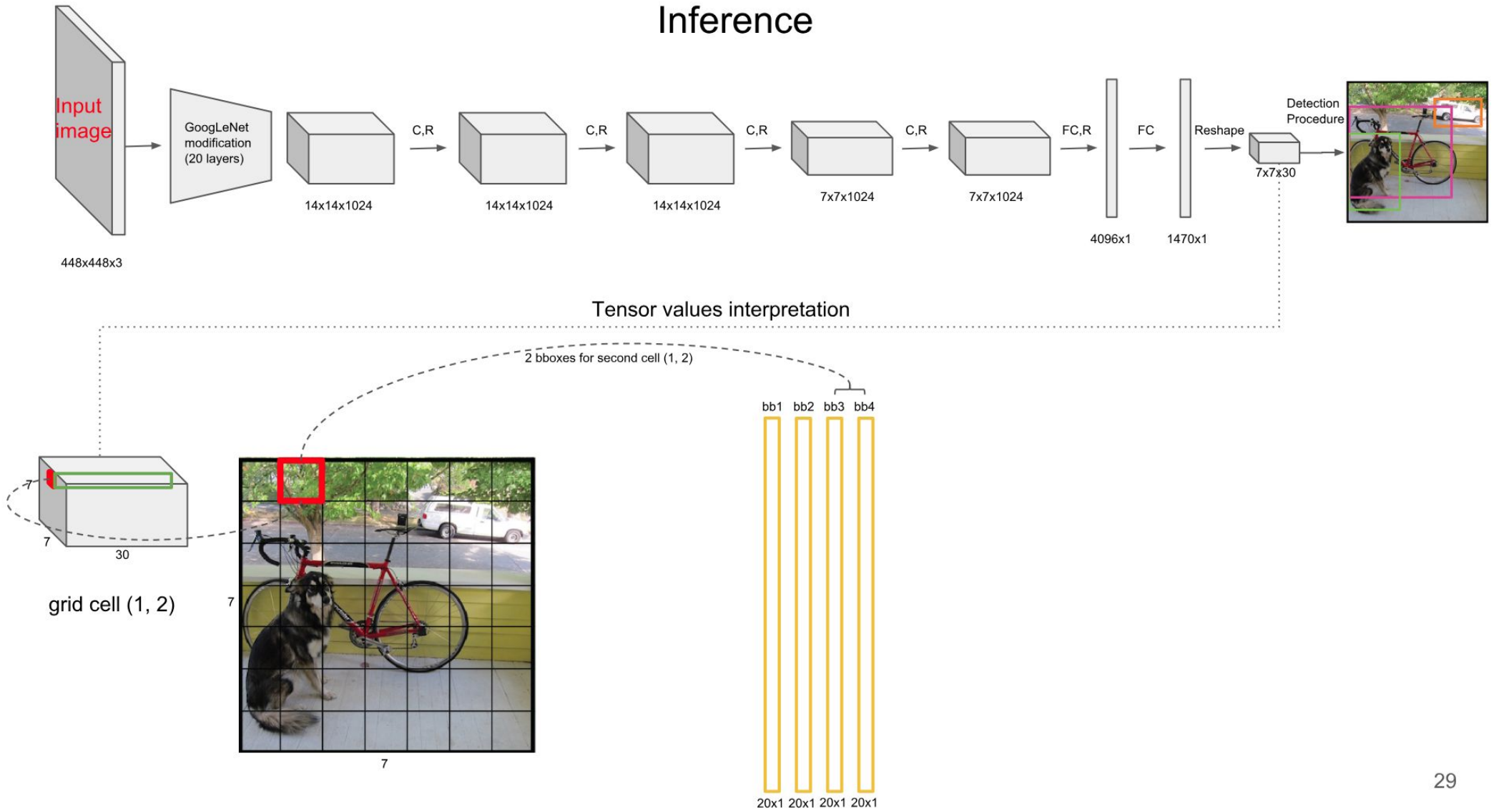

Inference
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
bb2 confidence
1x30
5
5
20
MUL
7
7
30
grid cell
7
7
Class scores for bb2
20x1


Inference
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

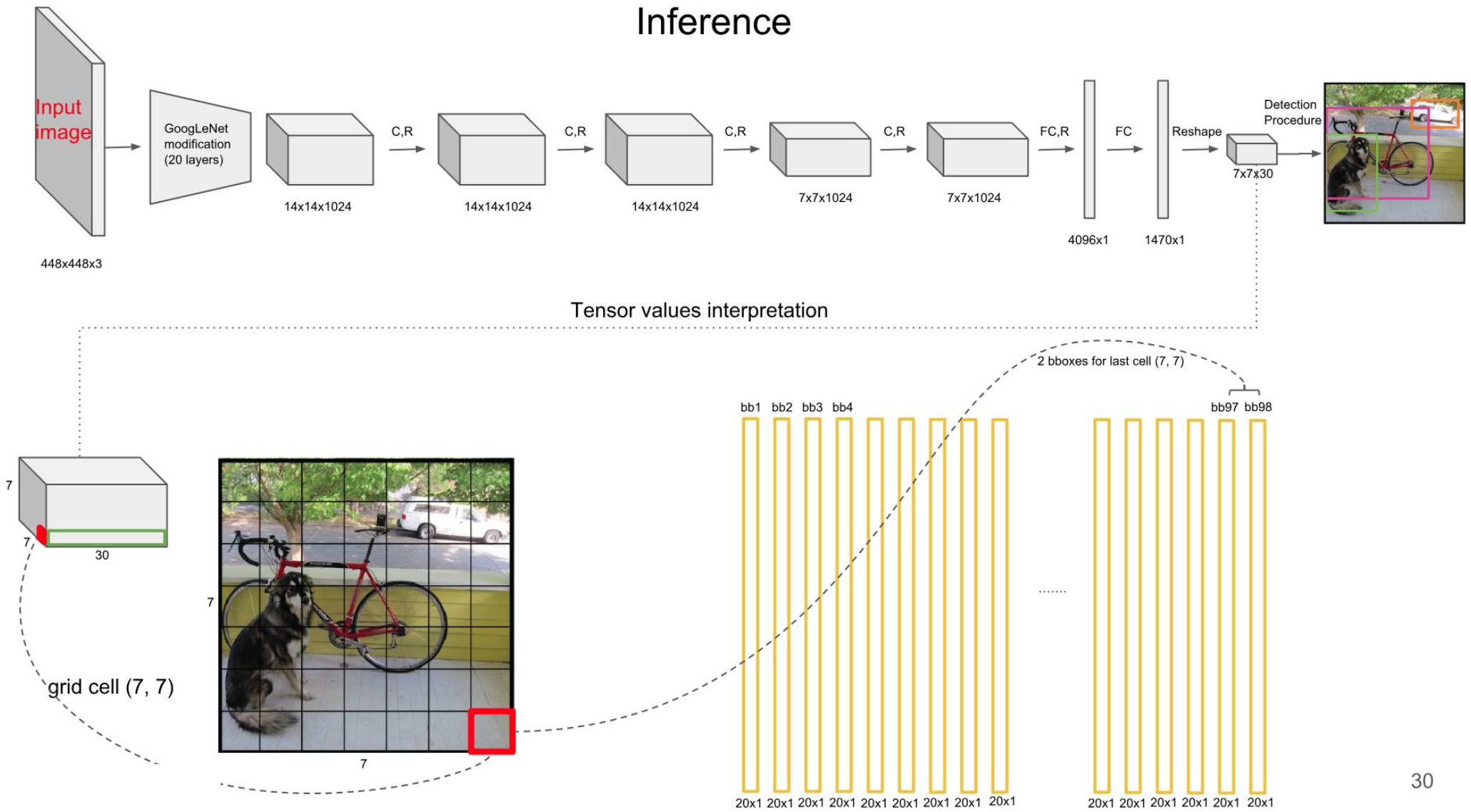
YOLO


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 first cell (1, 1)
bb1 bb2
7
7
30
grid cell (1, 1)
7
7
20x1 20x1

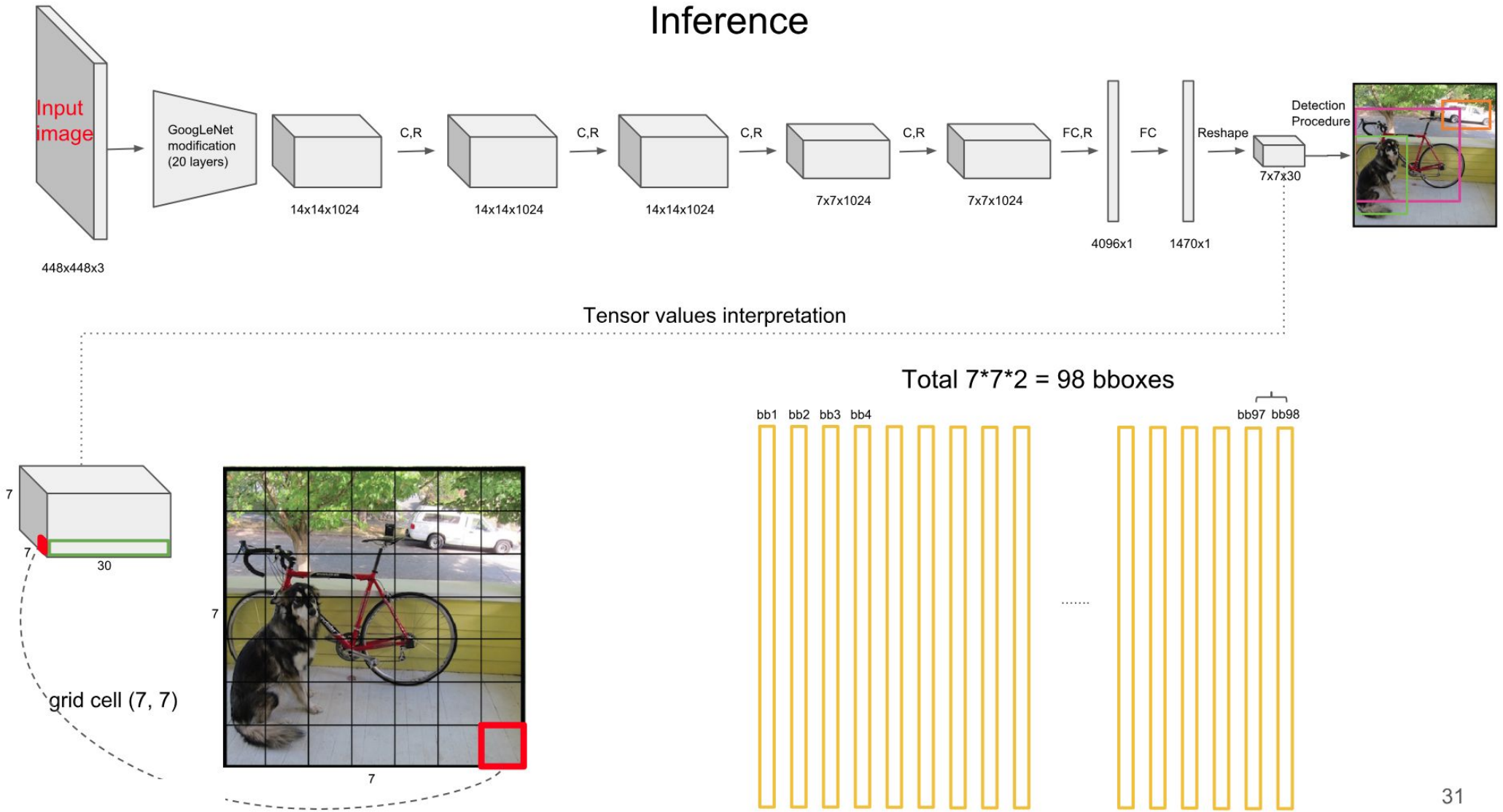
YOLO


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
7
7
30
grid cell (1, 2)
7
7
20x1 20x1 20x1 20x1

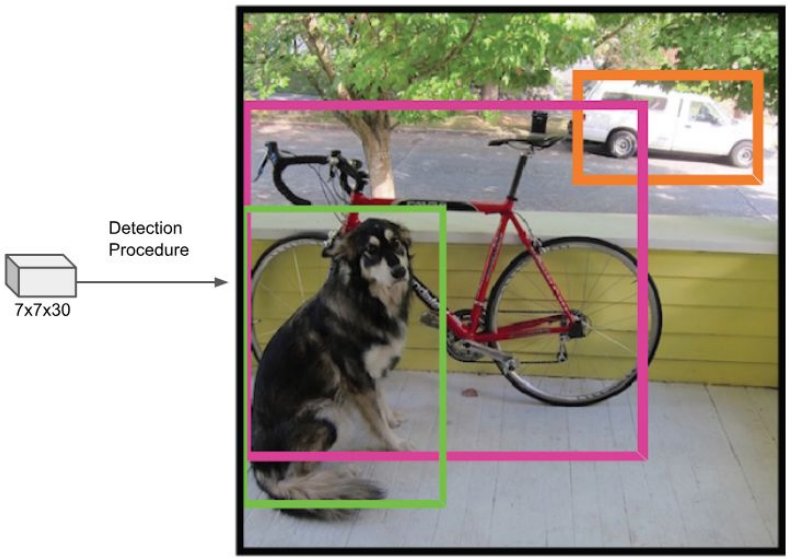
YOLO


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 last cell (7, 7)
bb1 bb2 bb3 bb4
bb97 bb98
7
7
30
grid cell (7, 7)
7
7
20x1 20x1 20x1 20x1 20x1 20x1 20x1 20x1 20x1
20x1 20x1 20x1 20x1 20x1 20x1

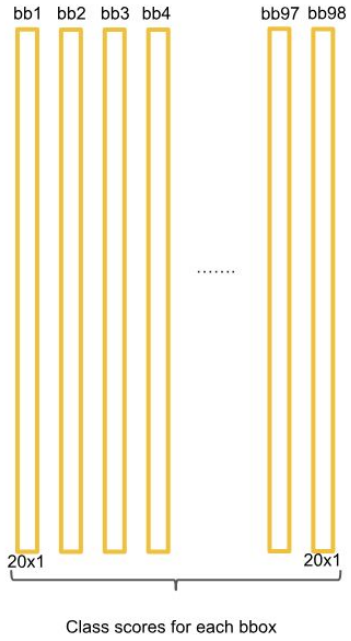
YOLO


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
Total 7*7*2 = 98 bboxes
bb1 bb2 bb3 bb4
bb97 bb98

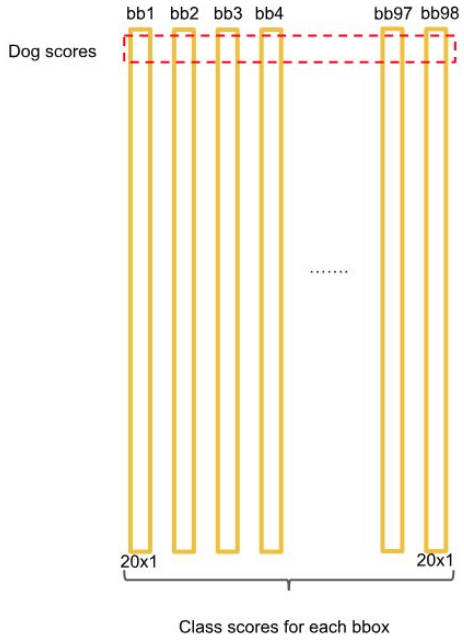
Look at detection procedure



YOLO

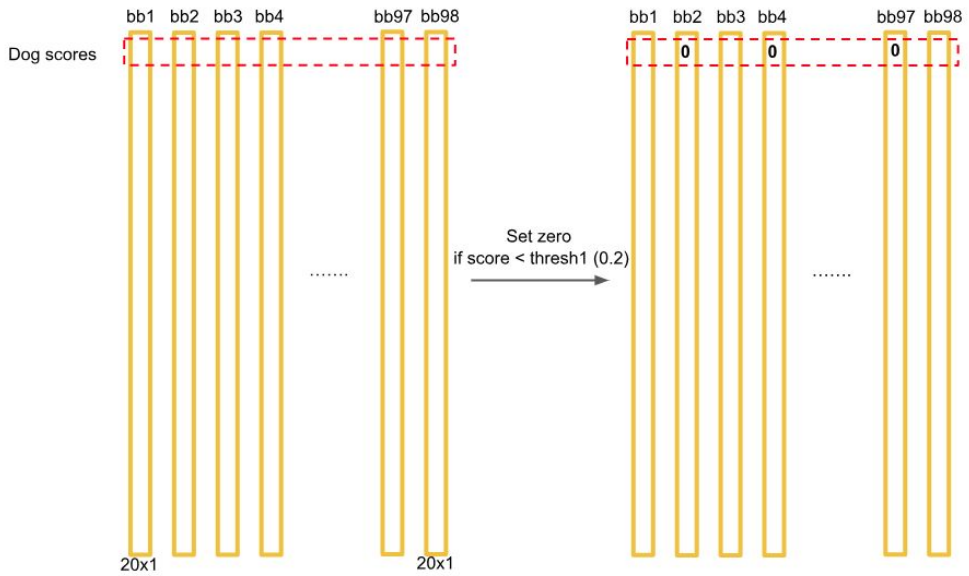

bb1
bb2
bb3
bb4
bb97
bb98
.....
20x1
20x1
Class scores for each bbox

YOLO

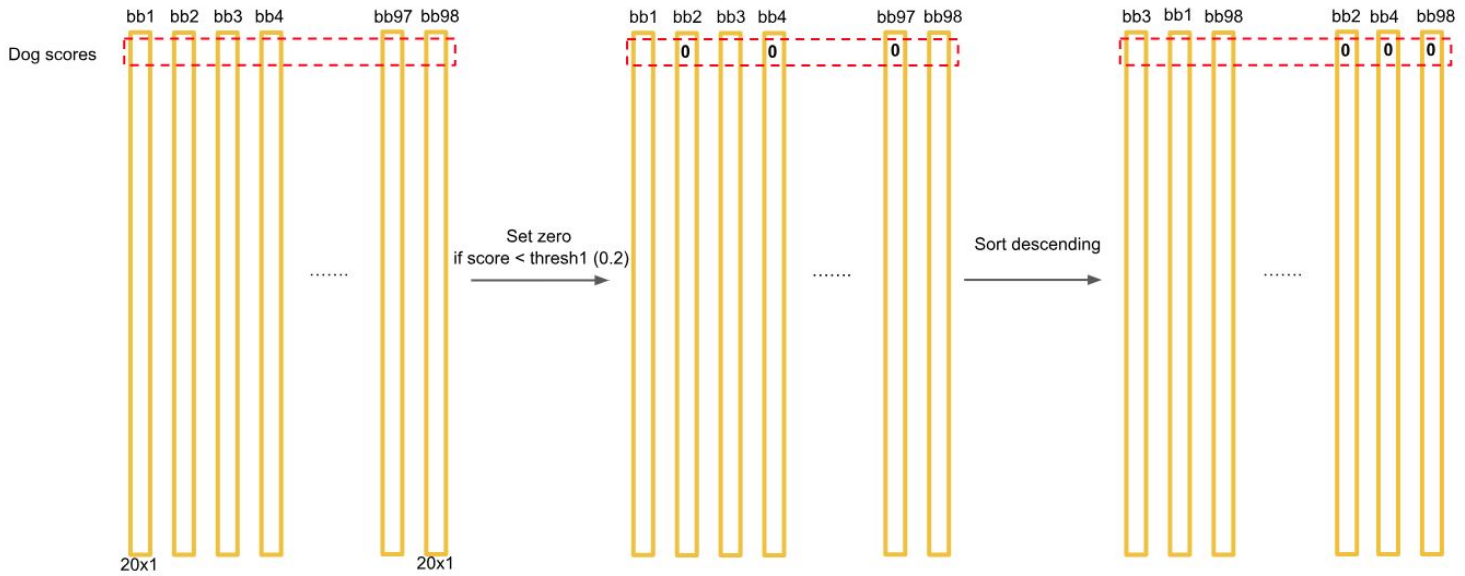


Get first class scores for each bbox

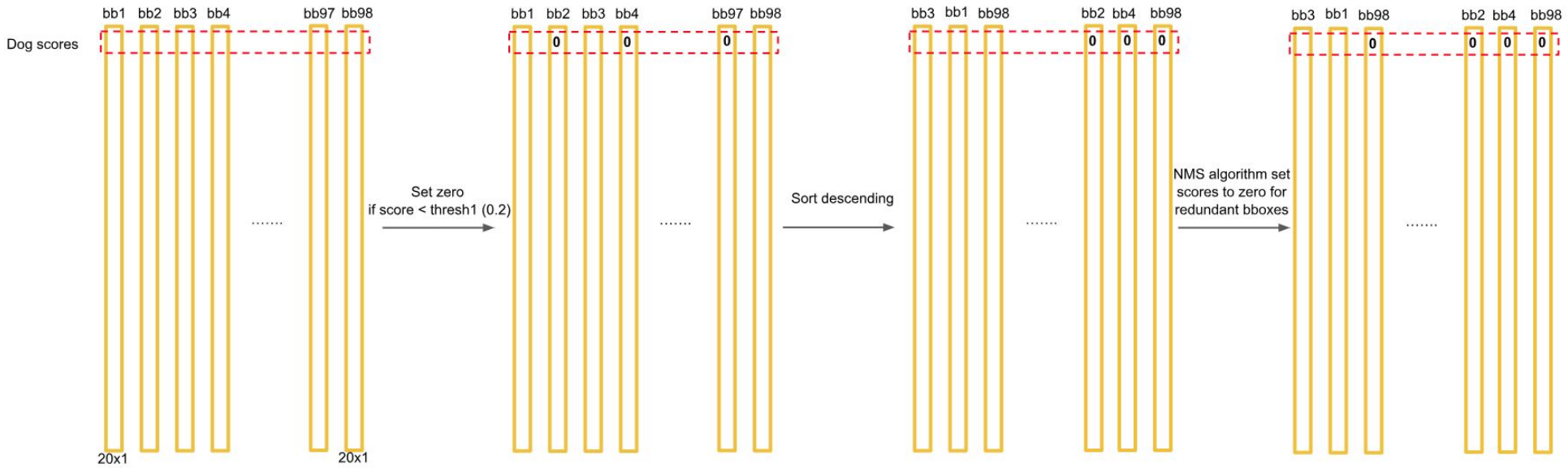
YOLO


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

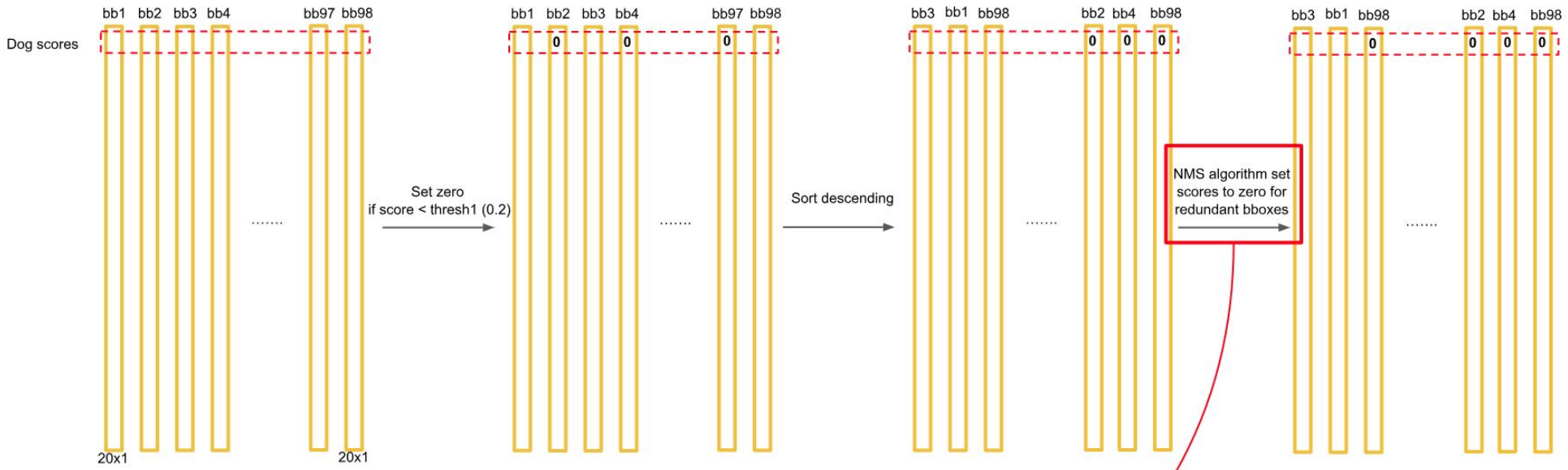
YOLO


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

YOLO


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

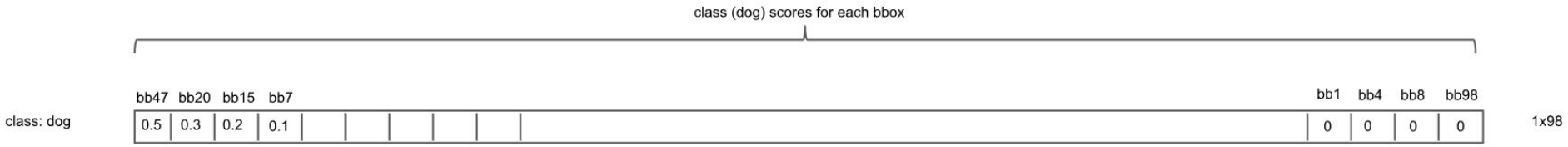
YOLO



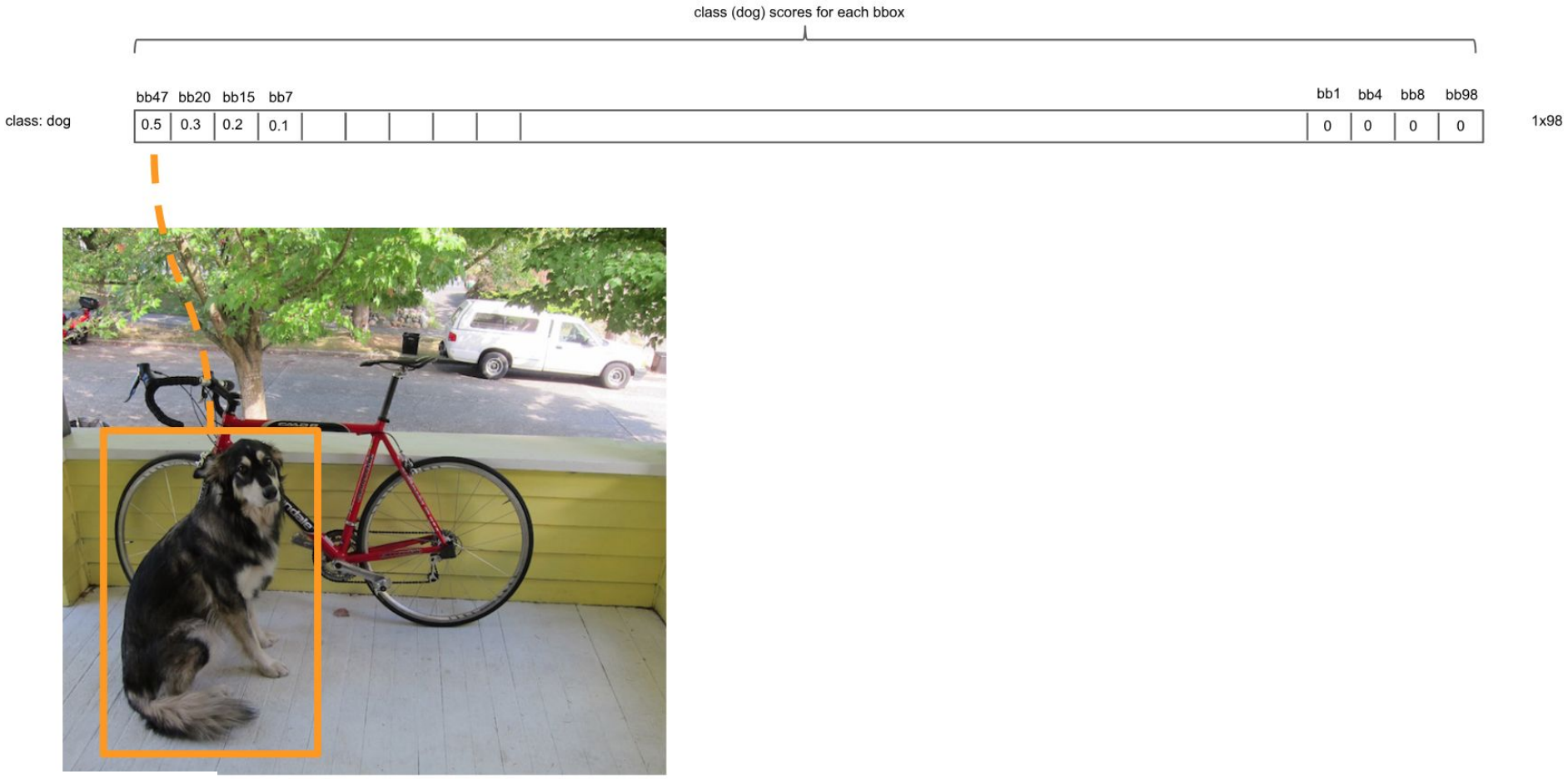
How it works

Non-Maximum Suppression: intuition

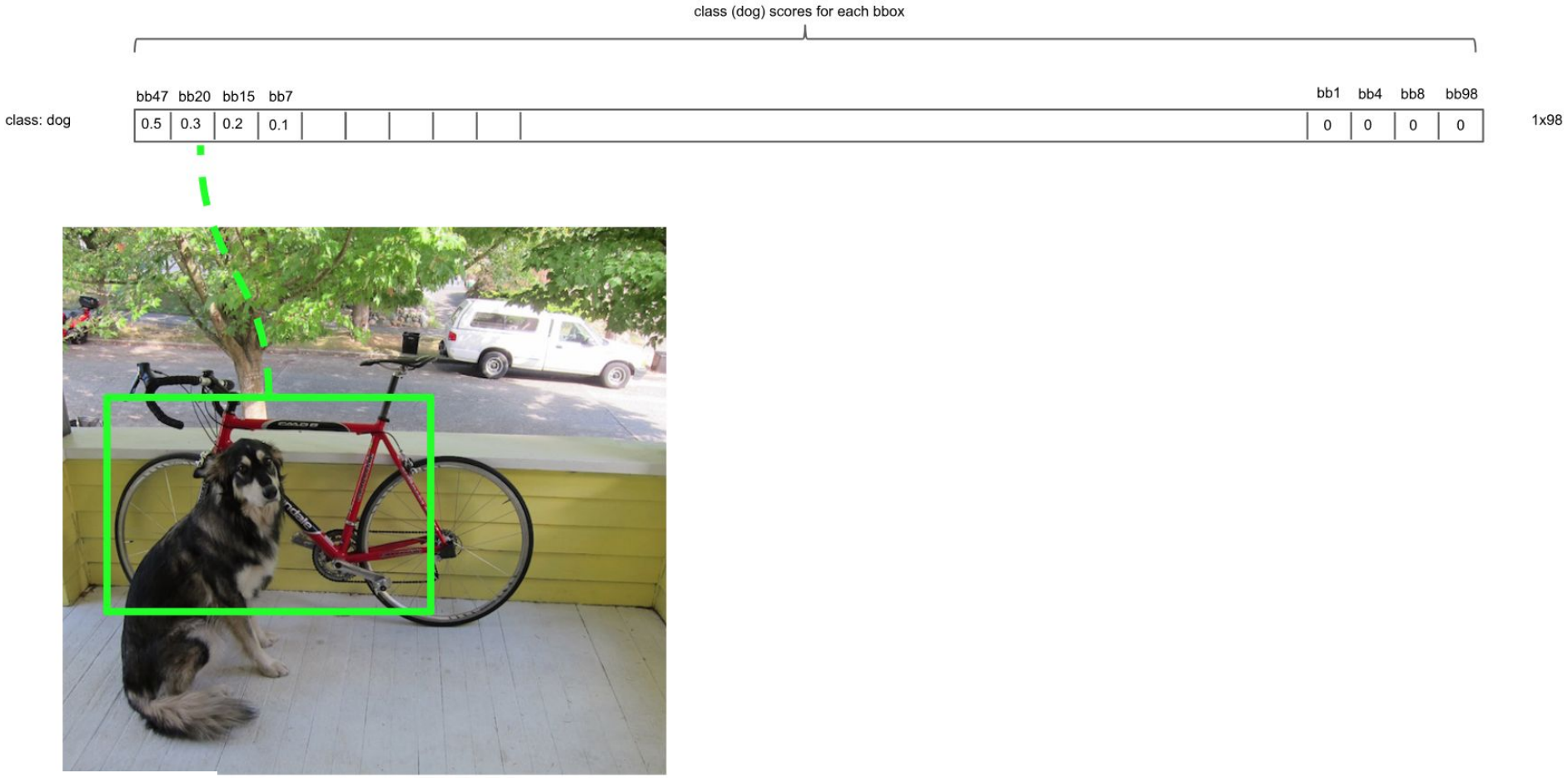
Non-Maximum Suppression: intuition



Non-Maximum Suppression: intuition



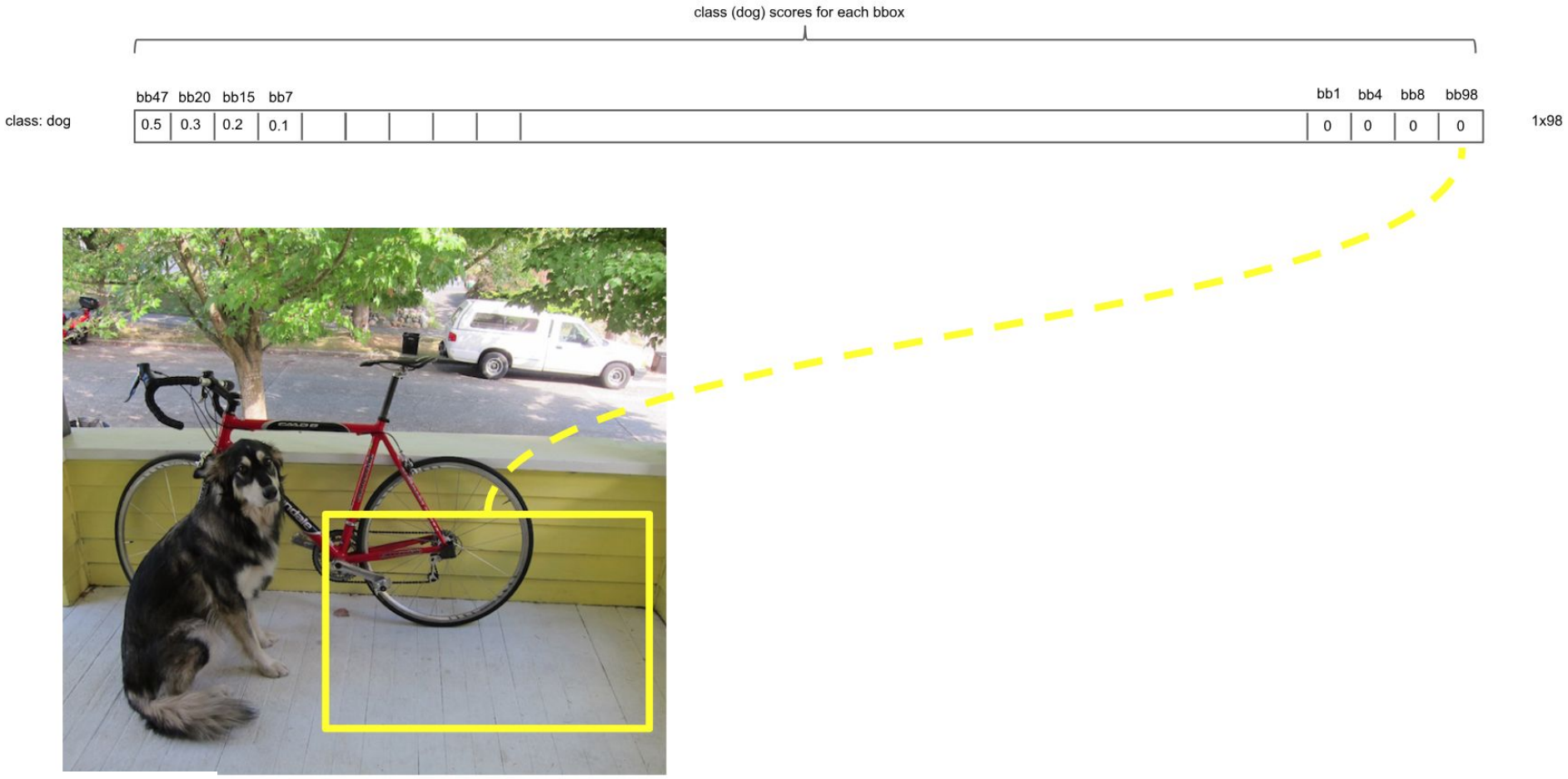
Non-Maximum Suppression: intuition



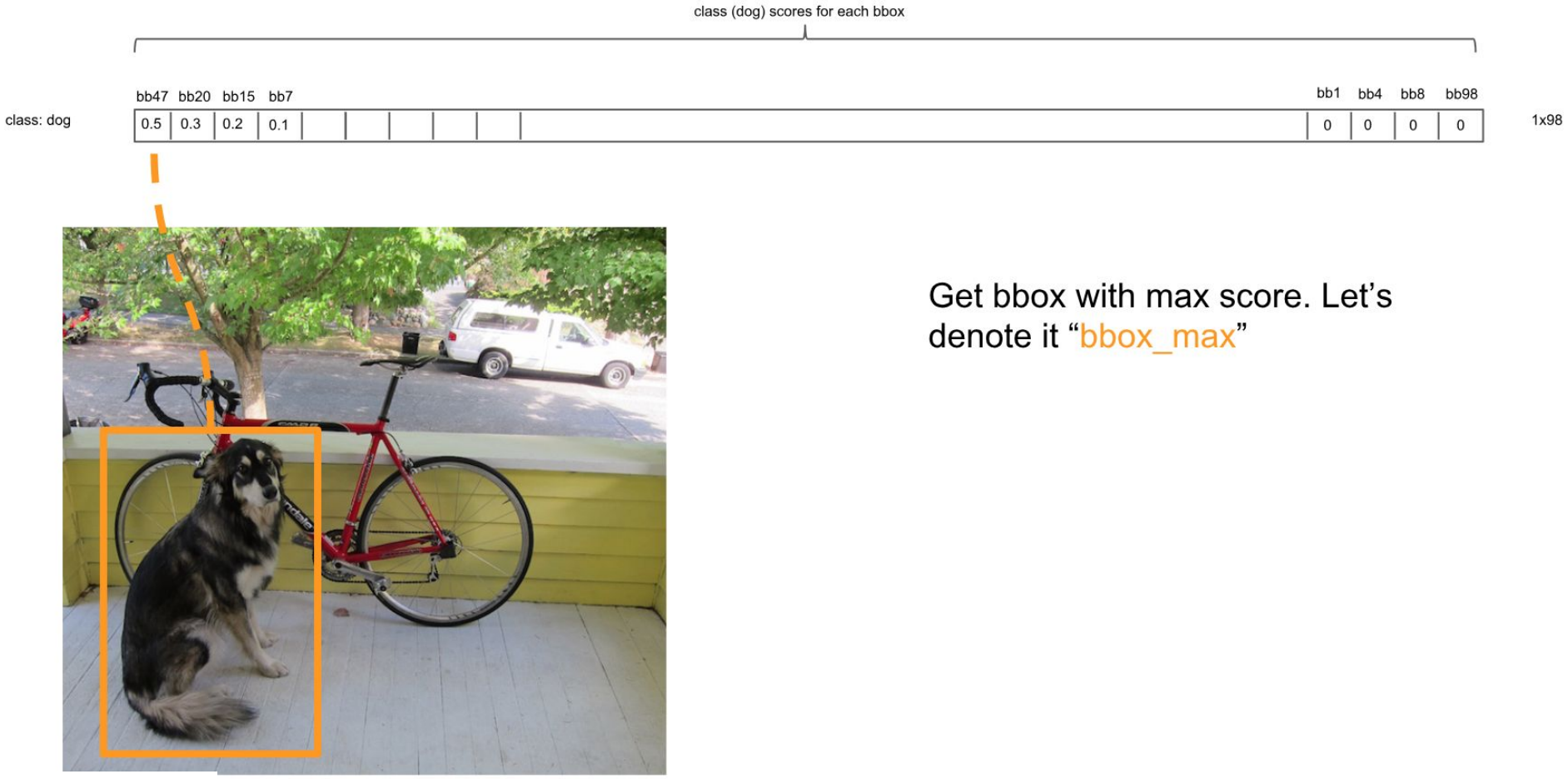
Non-Maximum Suppression: intuition

Non-Maximum Suppression: intuition

Non-Maximum Suppression: intuition

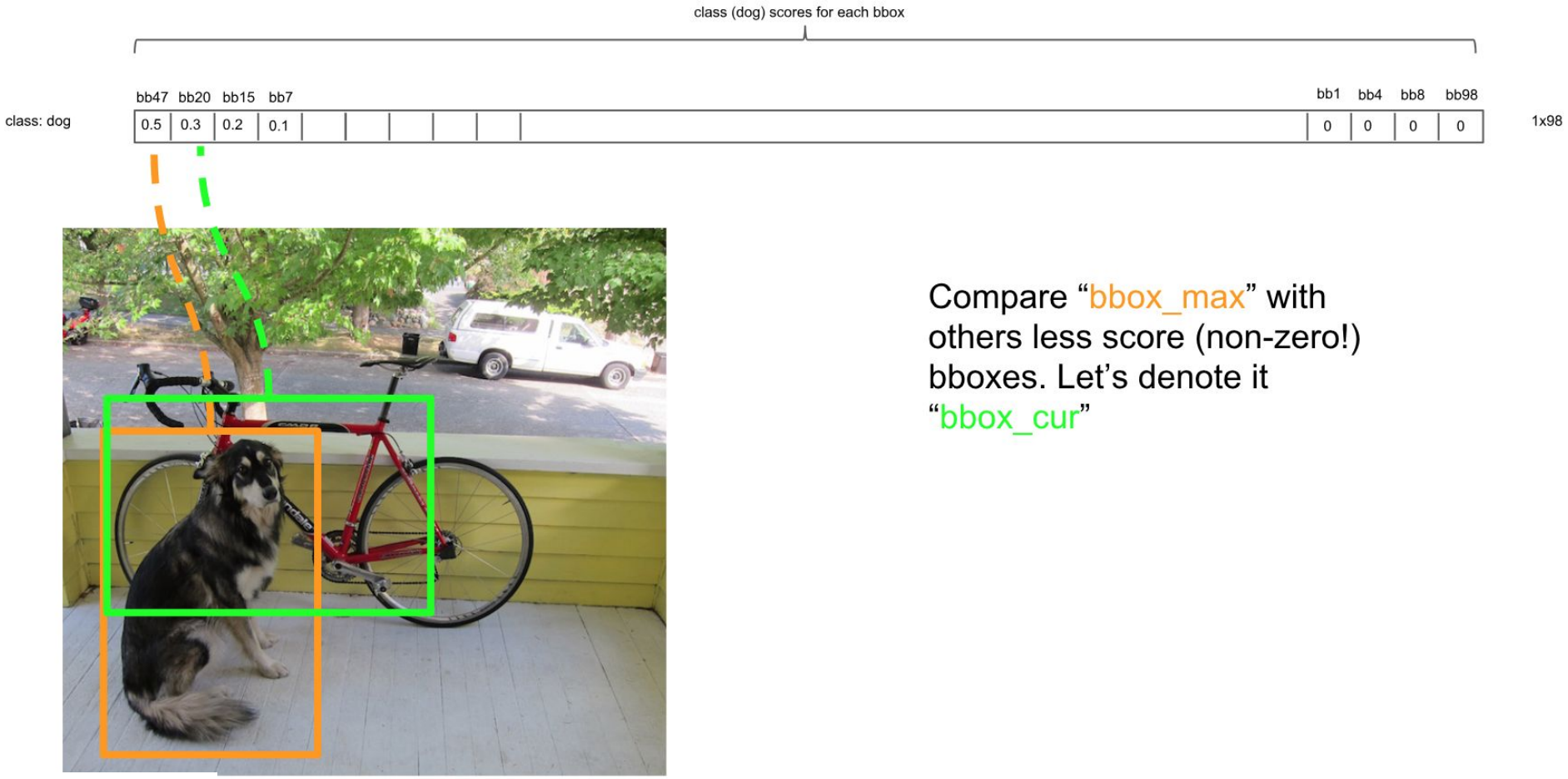


Non-Maximum Suppression: intuition



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

Non-Maximum Suppression: intuition

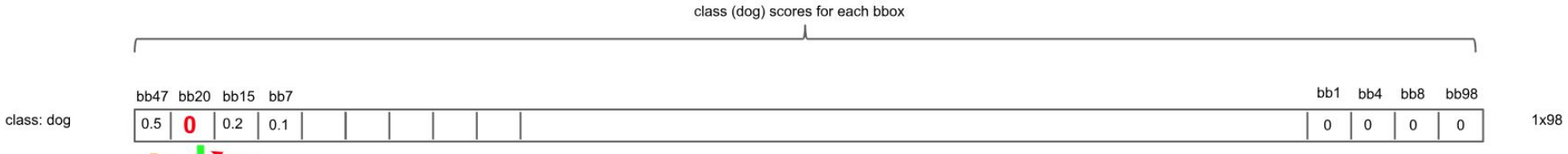


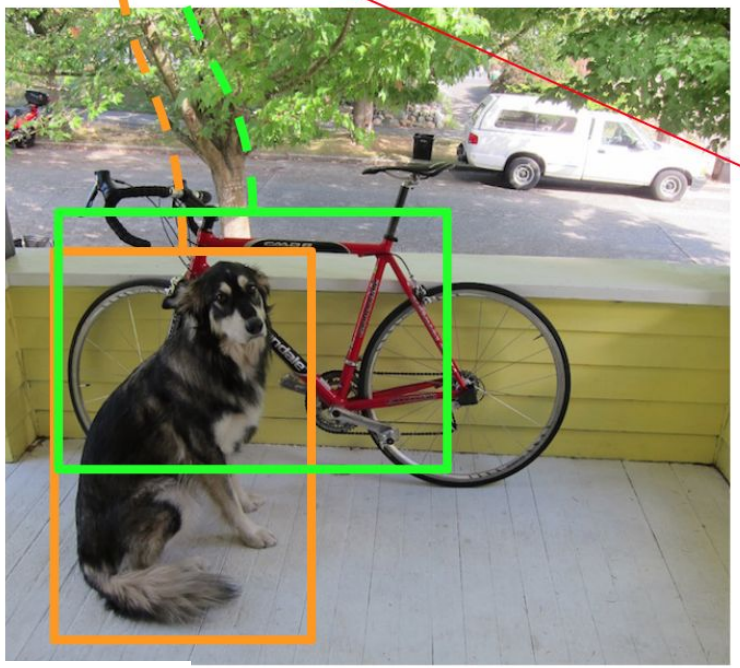
Compare “bbox_max” with others less score (non-zero!) bboxes. Let’s denote it “bbox_cur”

Non-Maximum Suppression: intuition

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

Non-Maximum Suppression: intuition

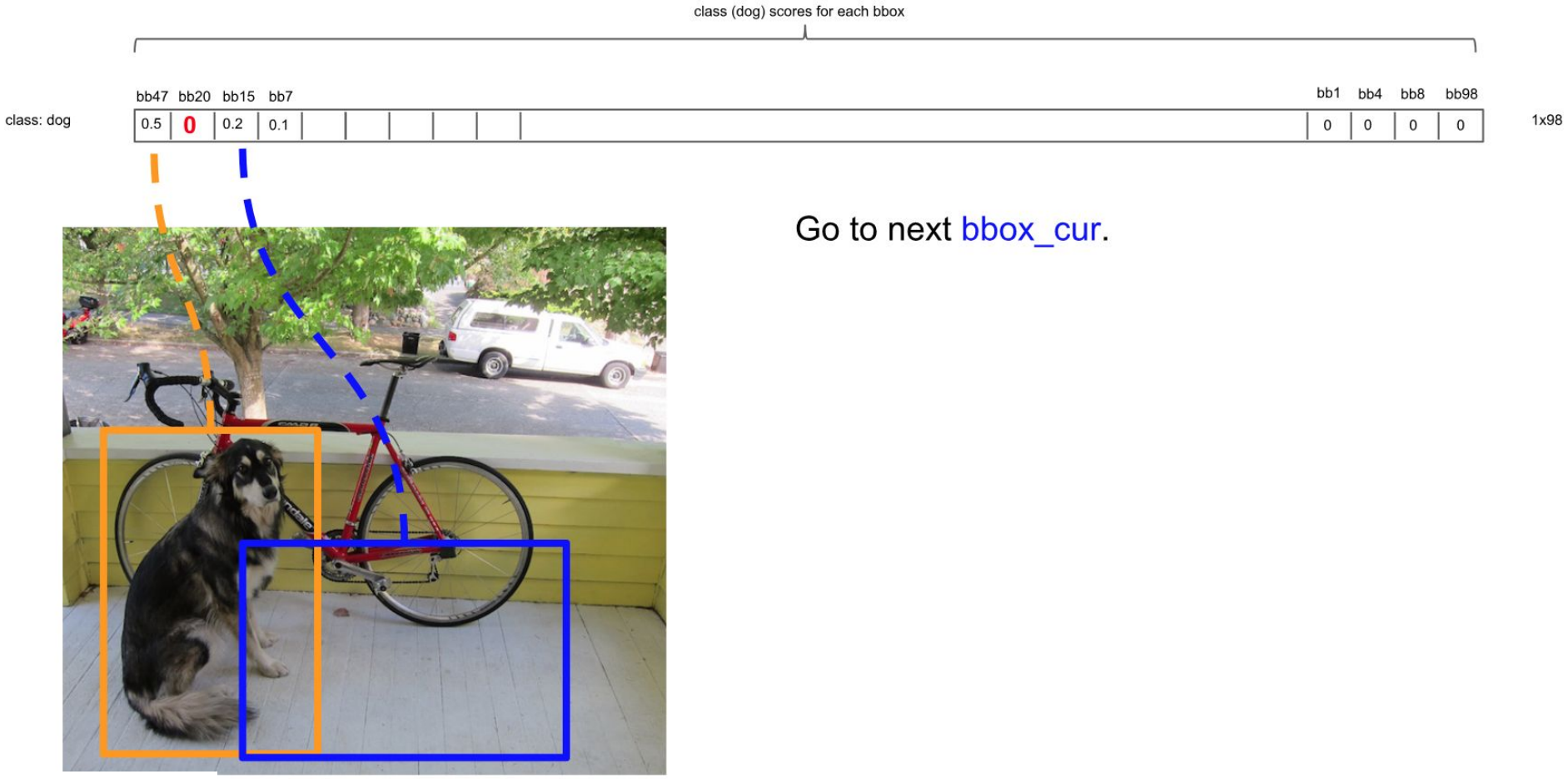




If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

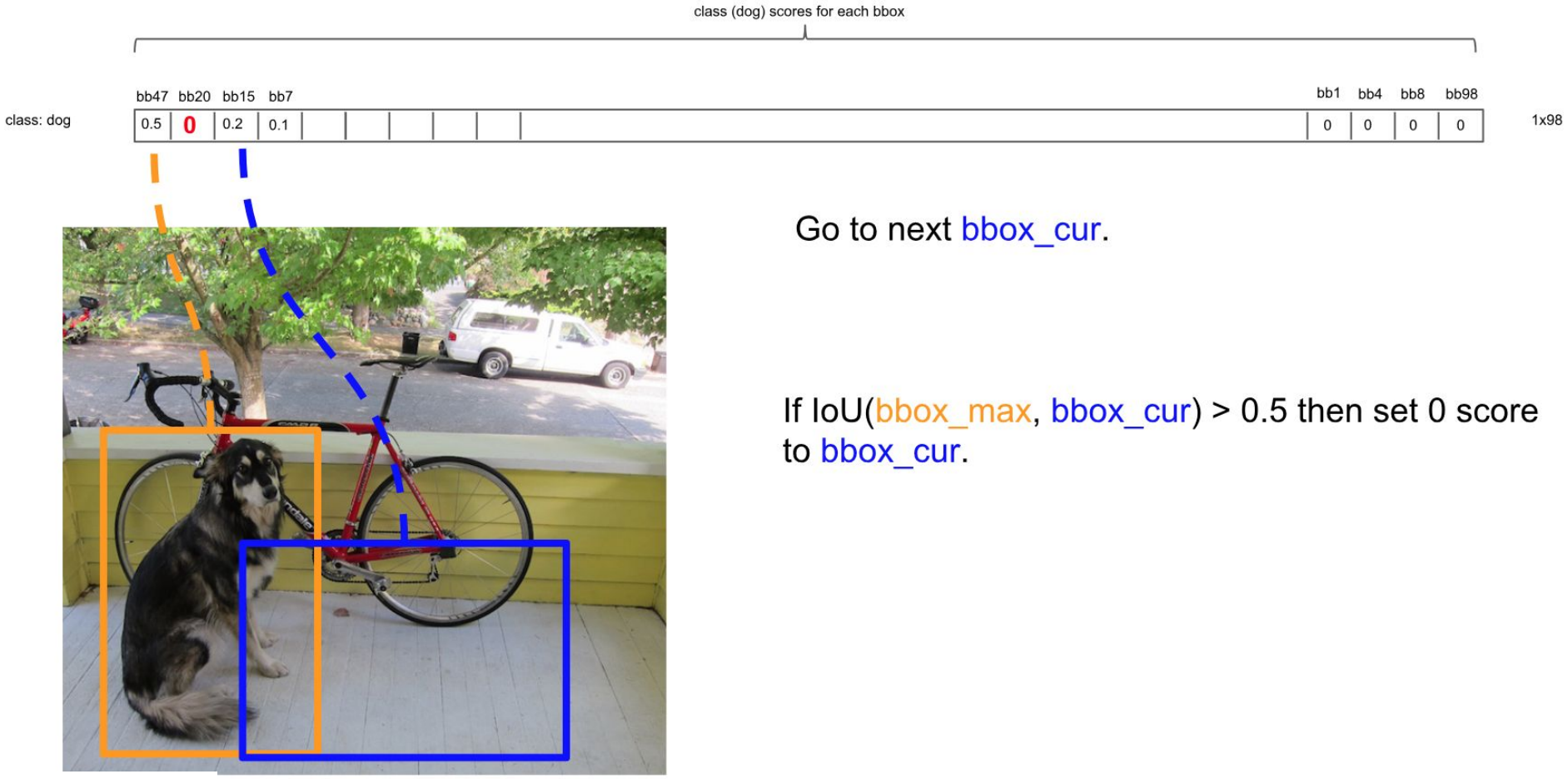
In this case: set to 0.

Non-Maximum Suppression: intuition



Go to next bbox_cur.

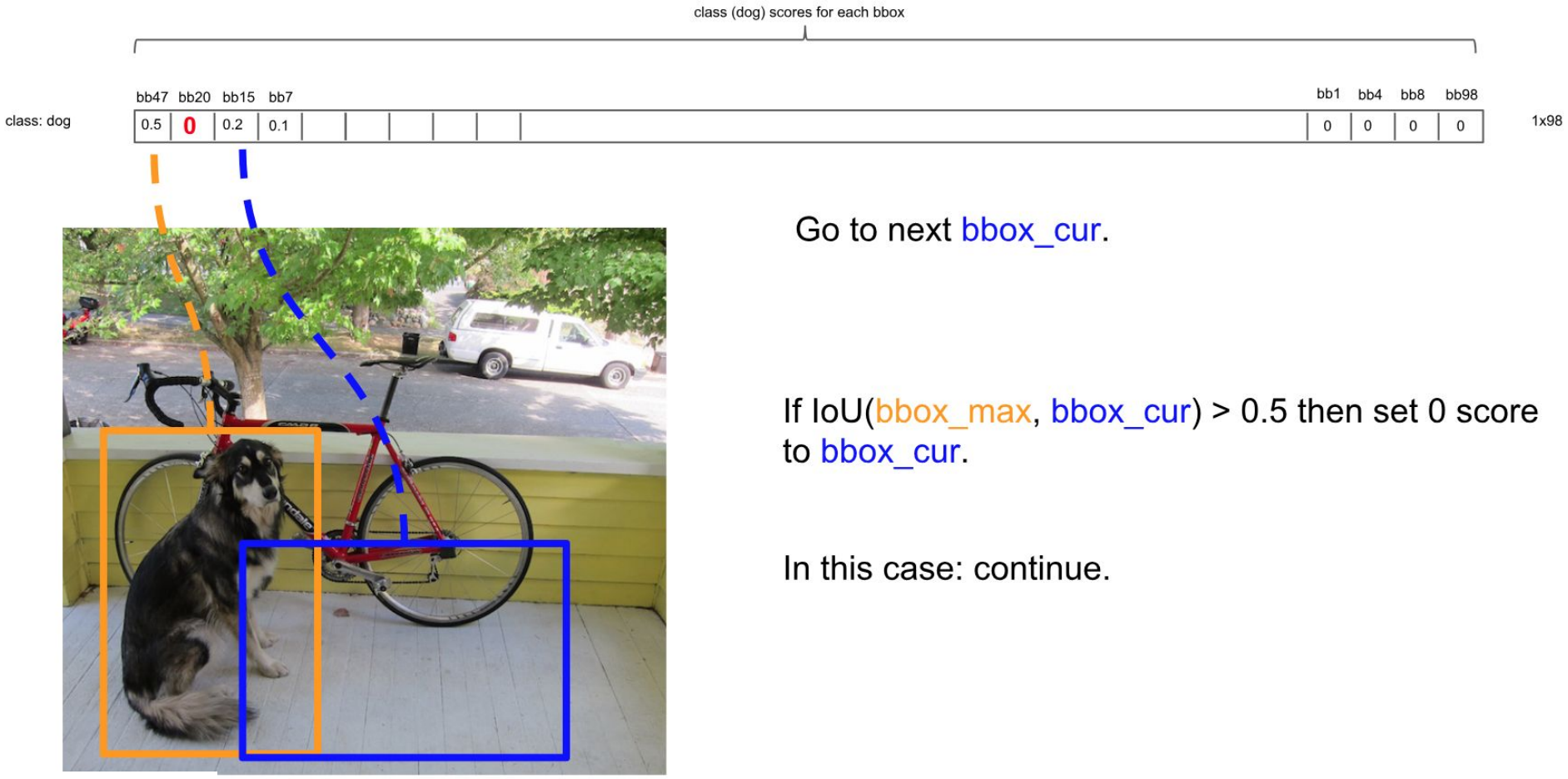
Non-Maximum Suppression: intuition



Go to next bbox_cur.

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

Non-Maximum Suppression: intuition

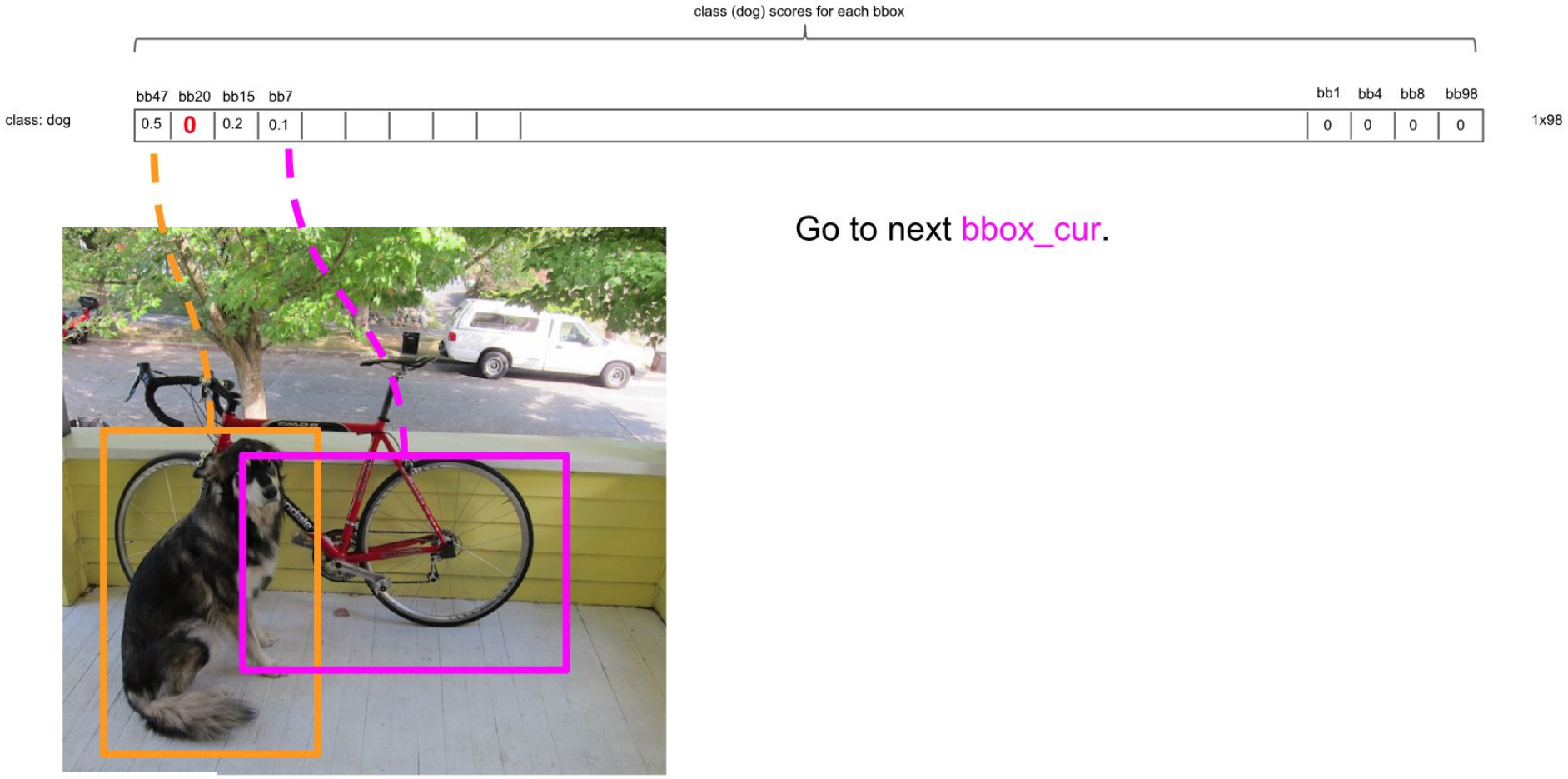


Go to next bbox_cur.

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

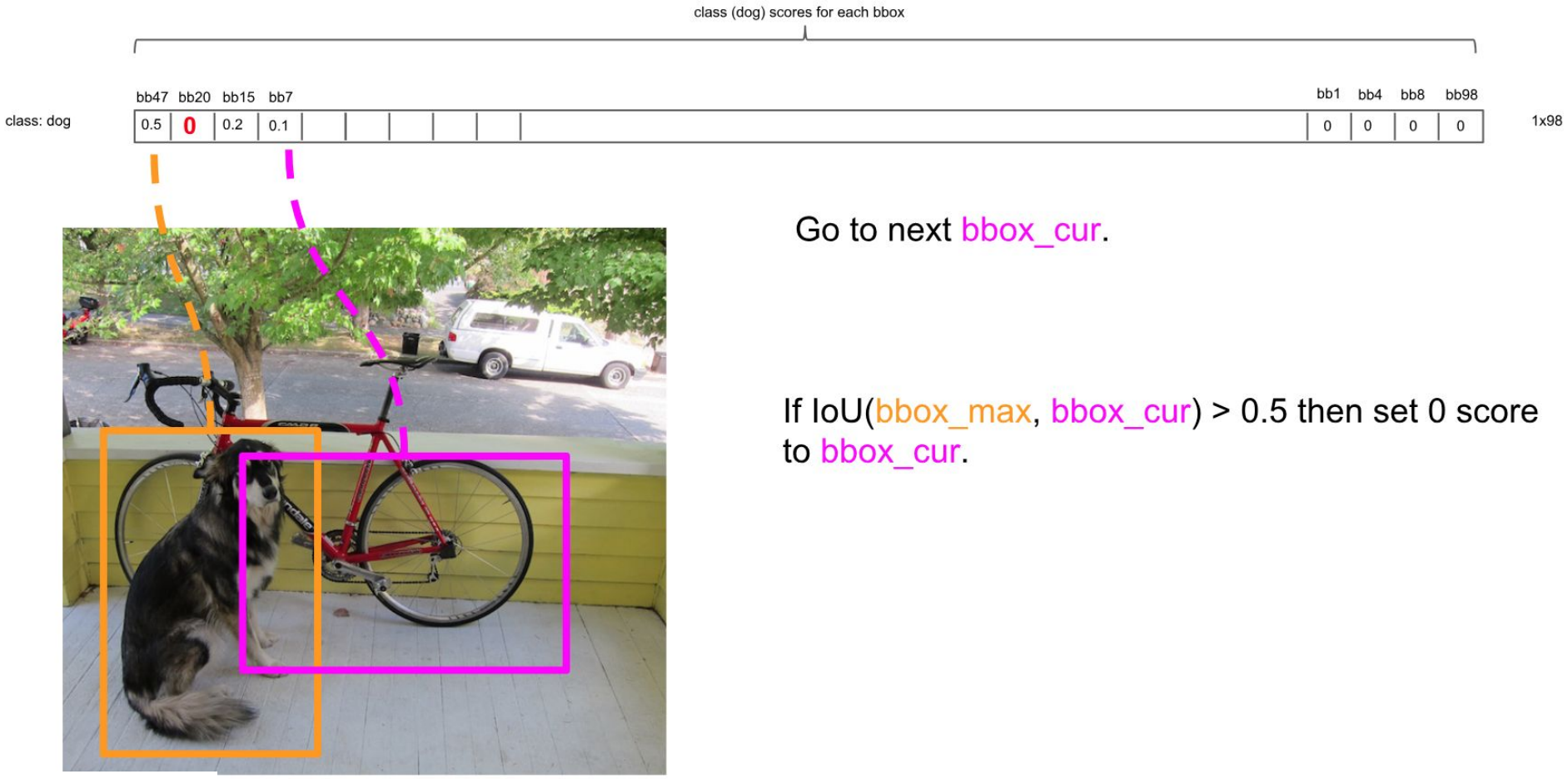
In this case: continue.

Non-Maximum Suppression: intuition



Go to next bbox_cur.

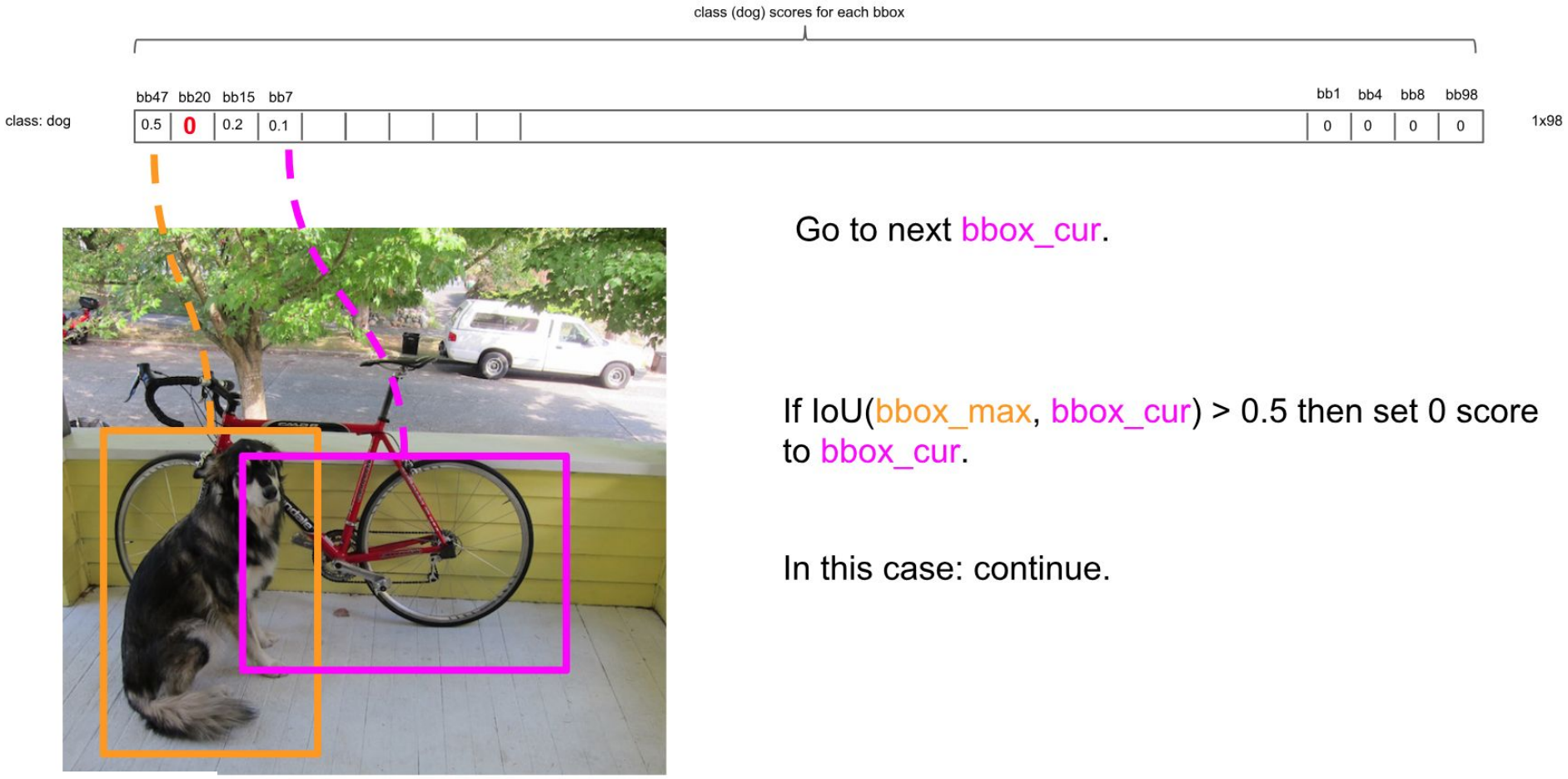
Non-Maximum Suppression: intuition



Go to next bbox_cur.

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

Non-Maximum Suppression: intuition

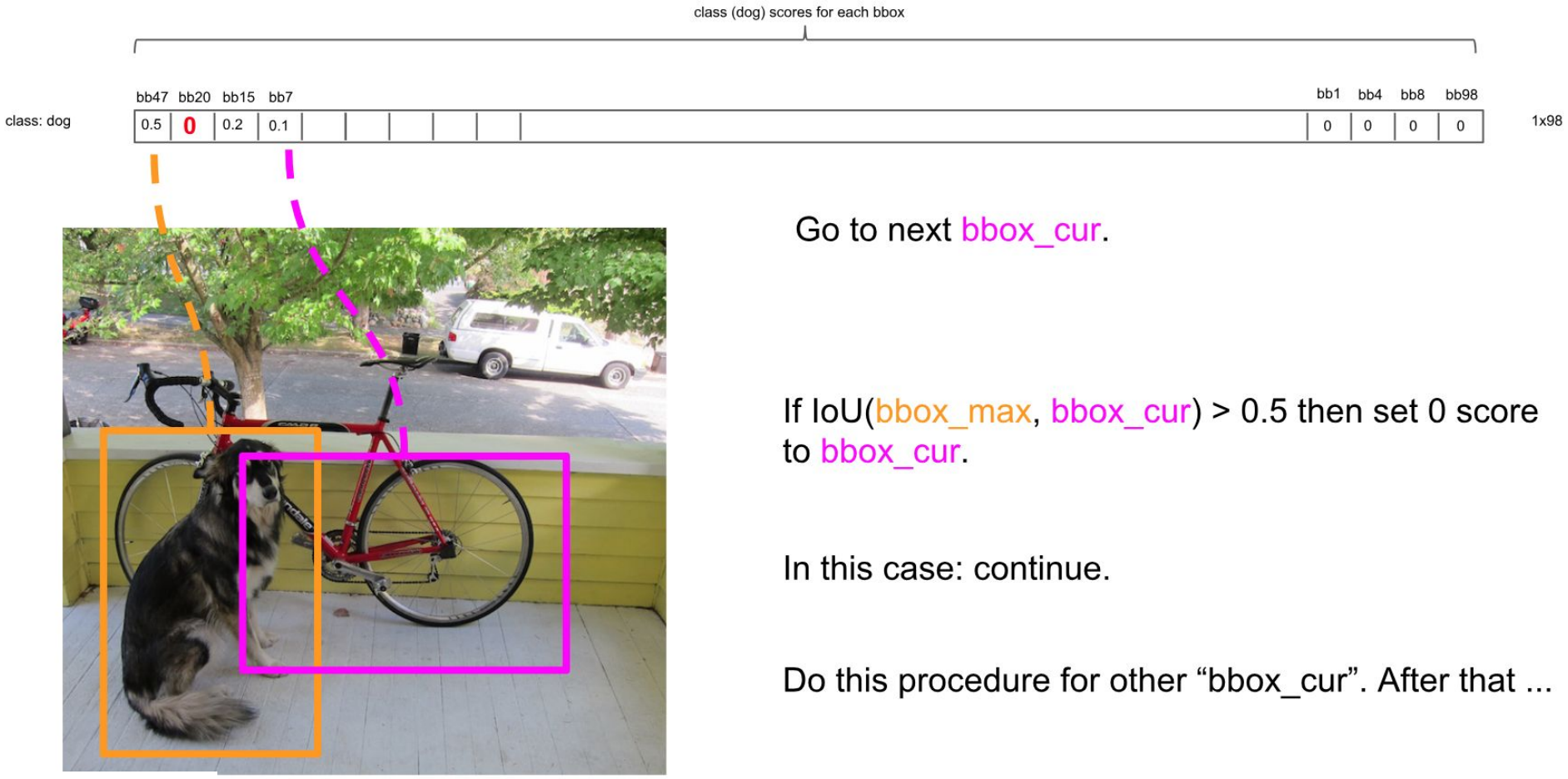


Go to next bbox_cur.

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

In this case: continue.

Non-Maximum Suppression: intuition



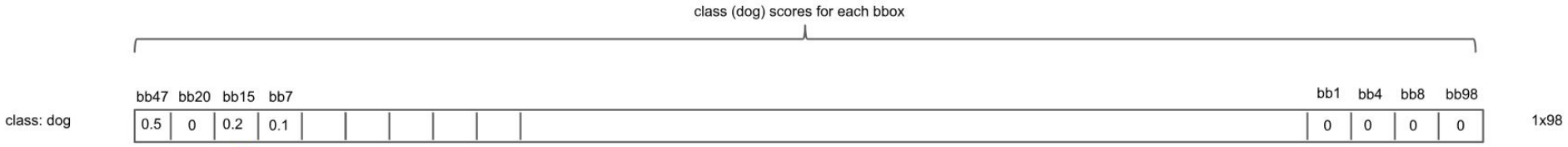
Go to next bbox_cur.

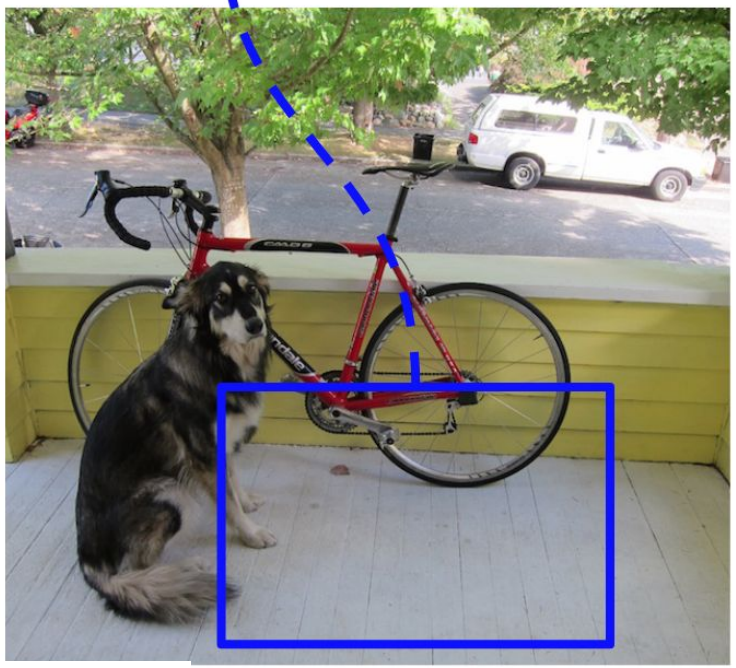
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: continue.

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

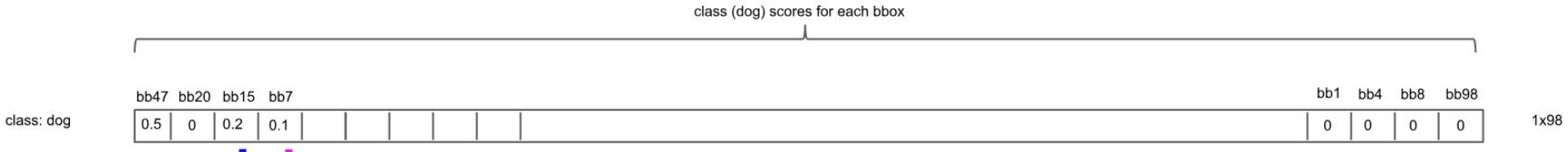
Non-Maximum Suppression: intuition

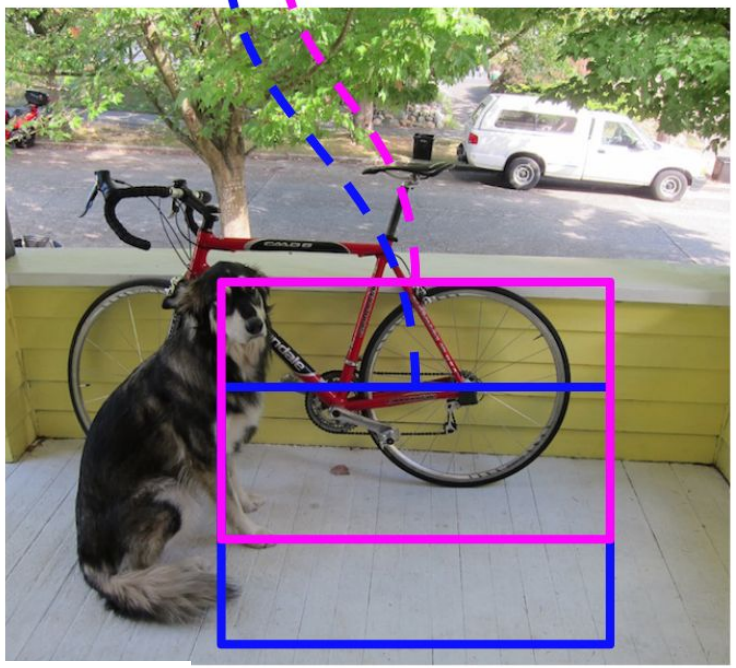




Go to next bbox with big score.
Let's denote it "bbox_max"

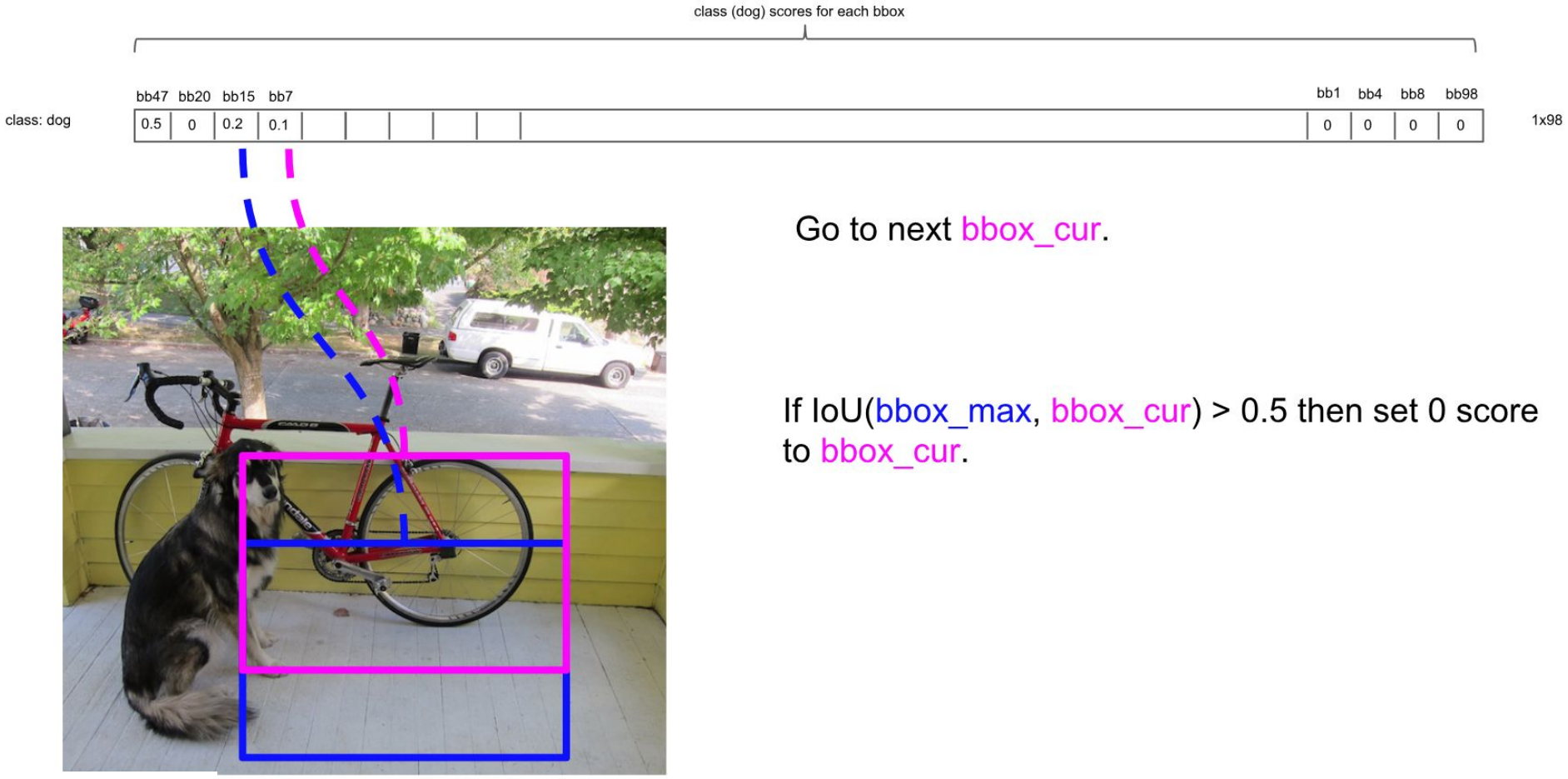
Non-Maximum Suppression: intuition





Go to next bbox_cur.

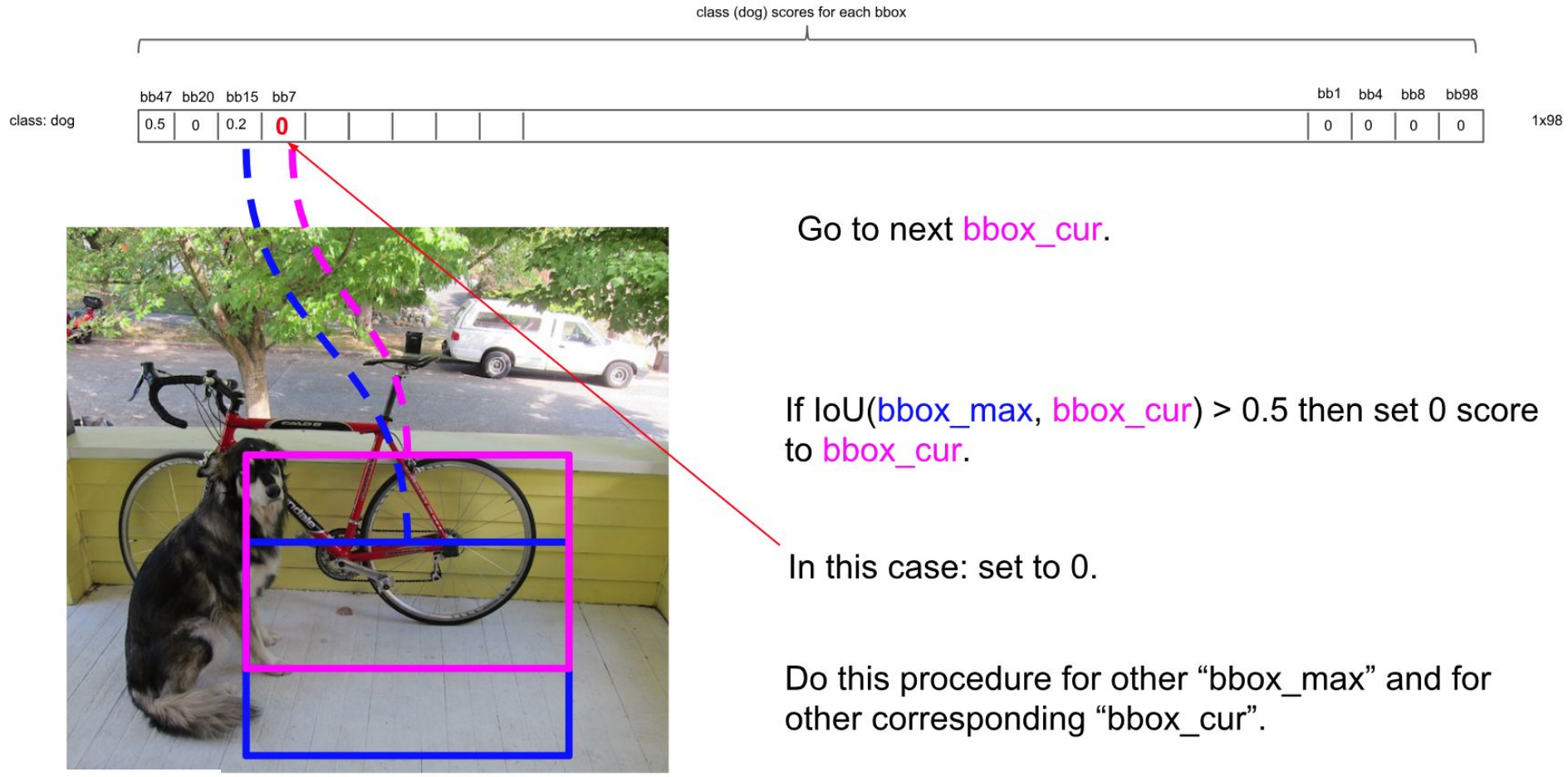
Non-Maximum Suppression: intuition



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.

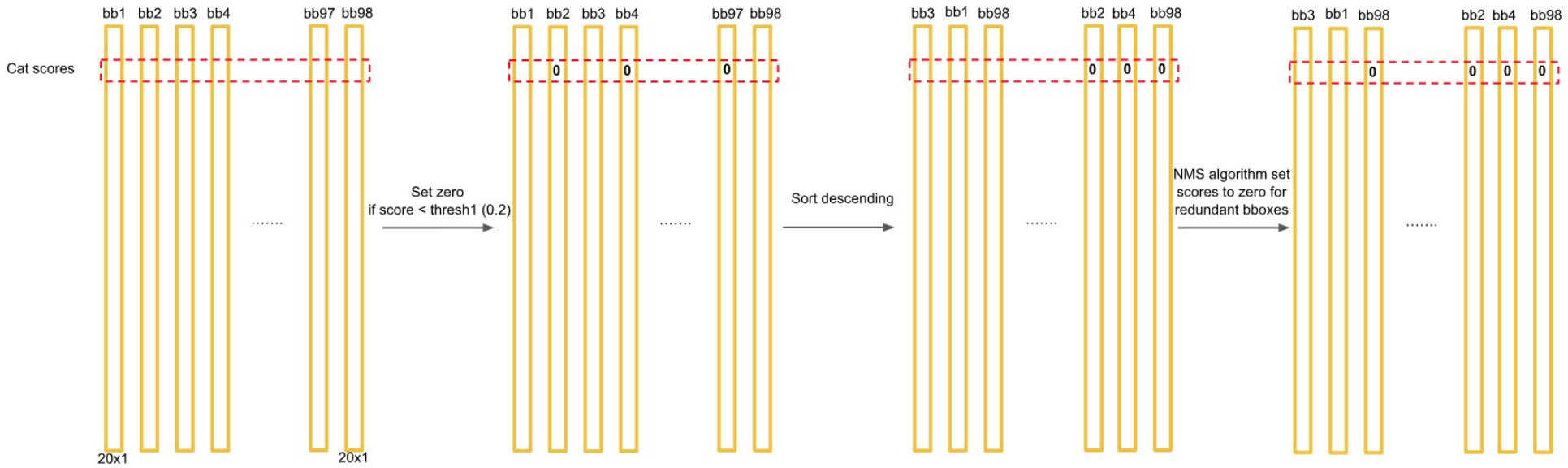
In this case: set to 0.

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

Non-Maximum Suppression: intuition

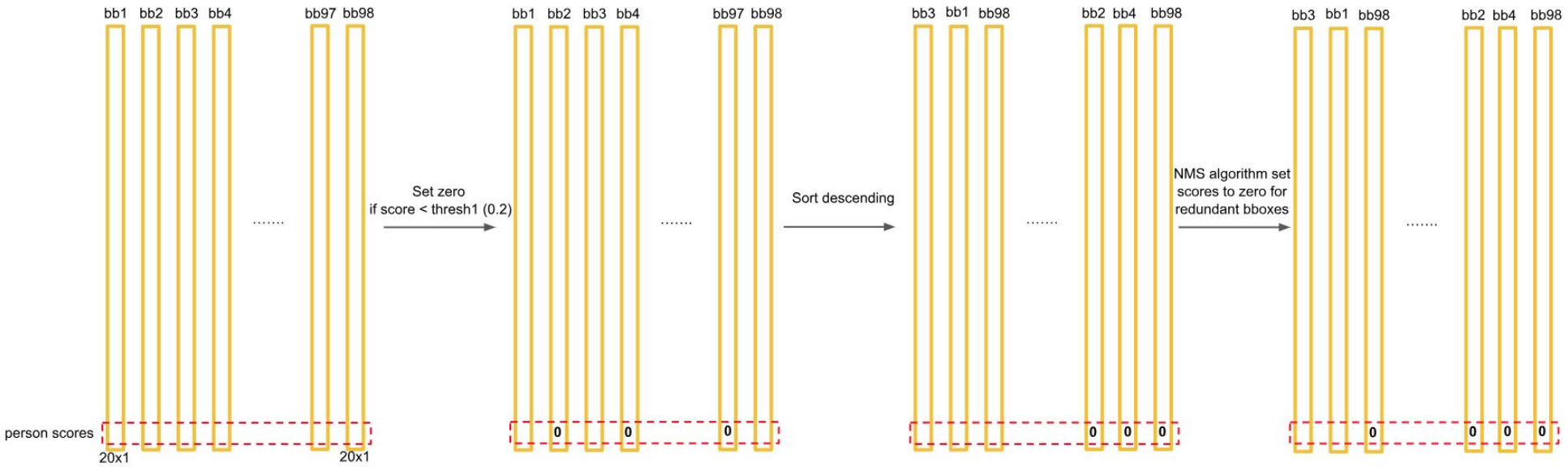
After comparison almost all pairs of bboxes the only two bboxes left with non-zero class score value.

YOLO



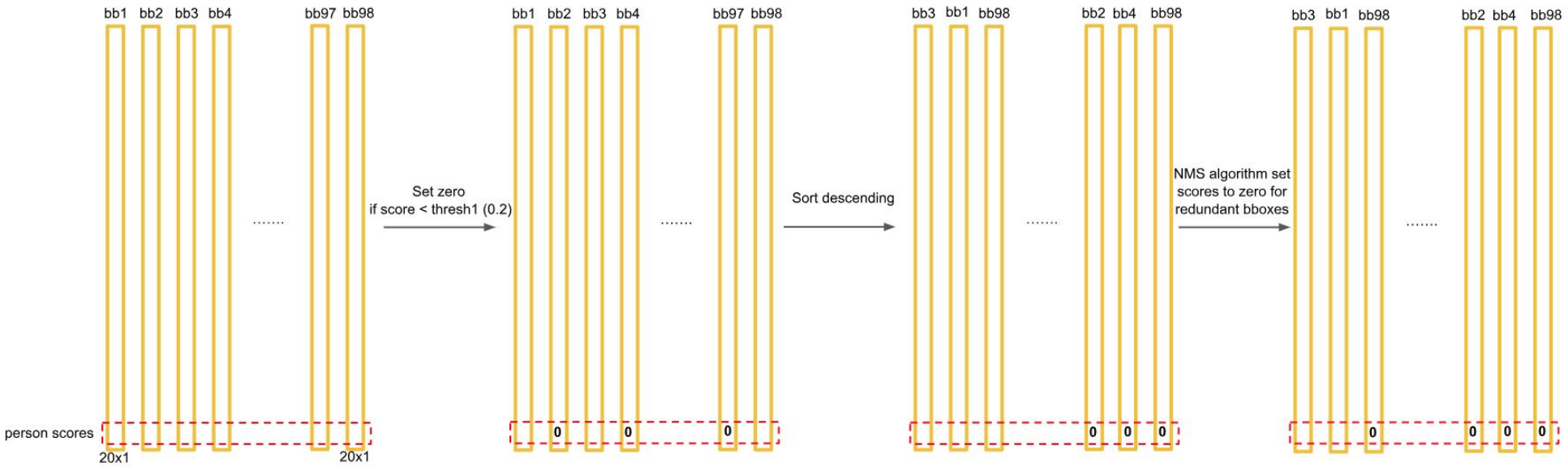
Do this procedure for next class

YOLO



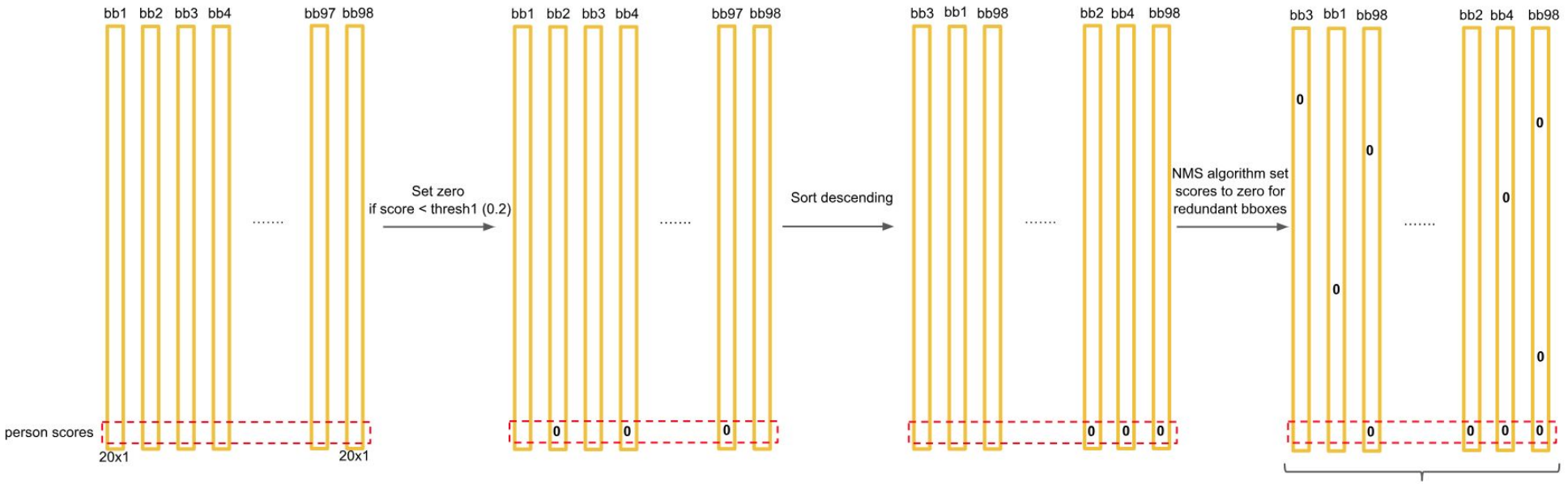
Do this procedure for all classes

YOLO



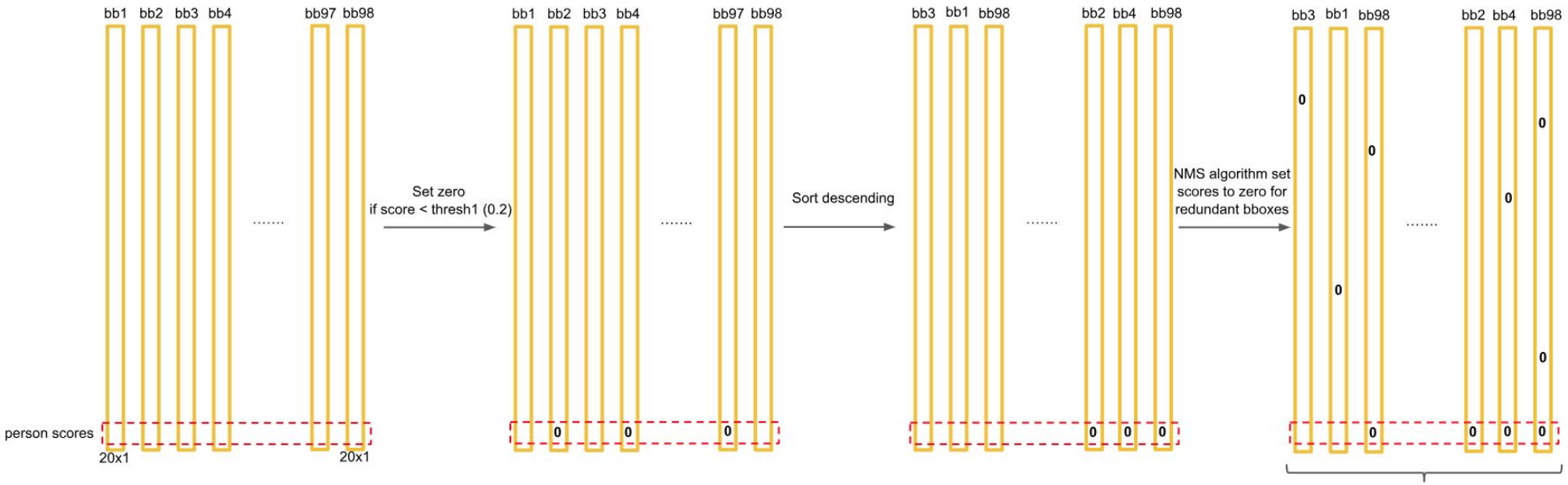
Do this procedure for all classes

YOLO



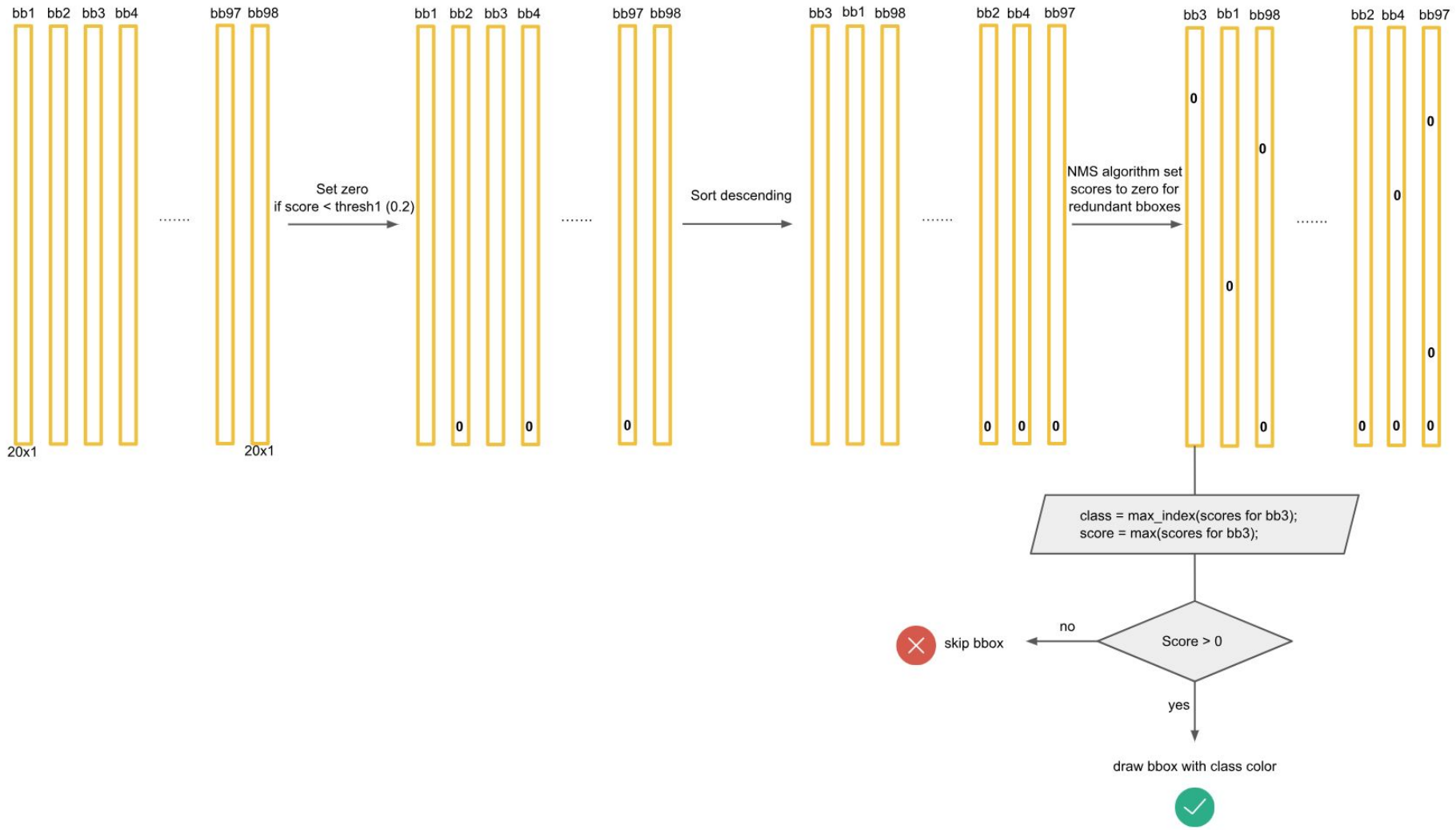
After this procedure -
a lot of zeros

YOLO

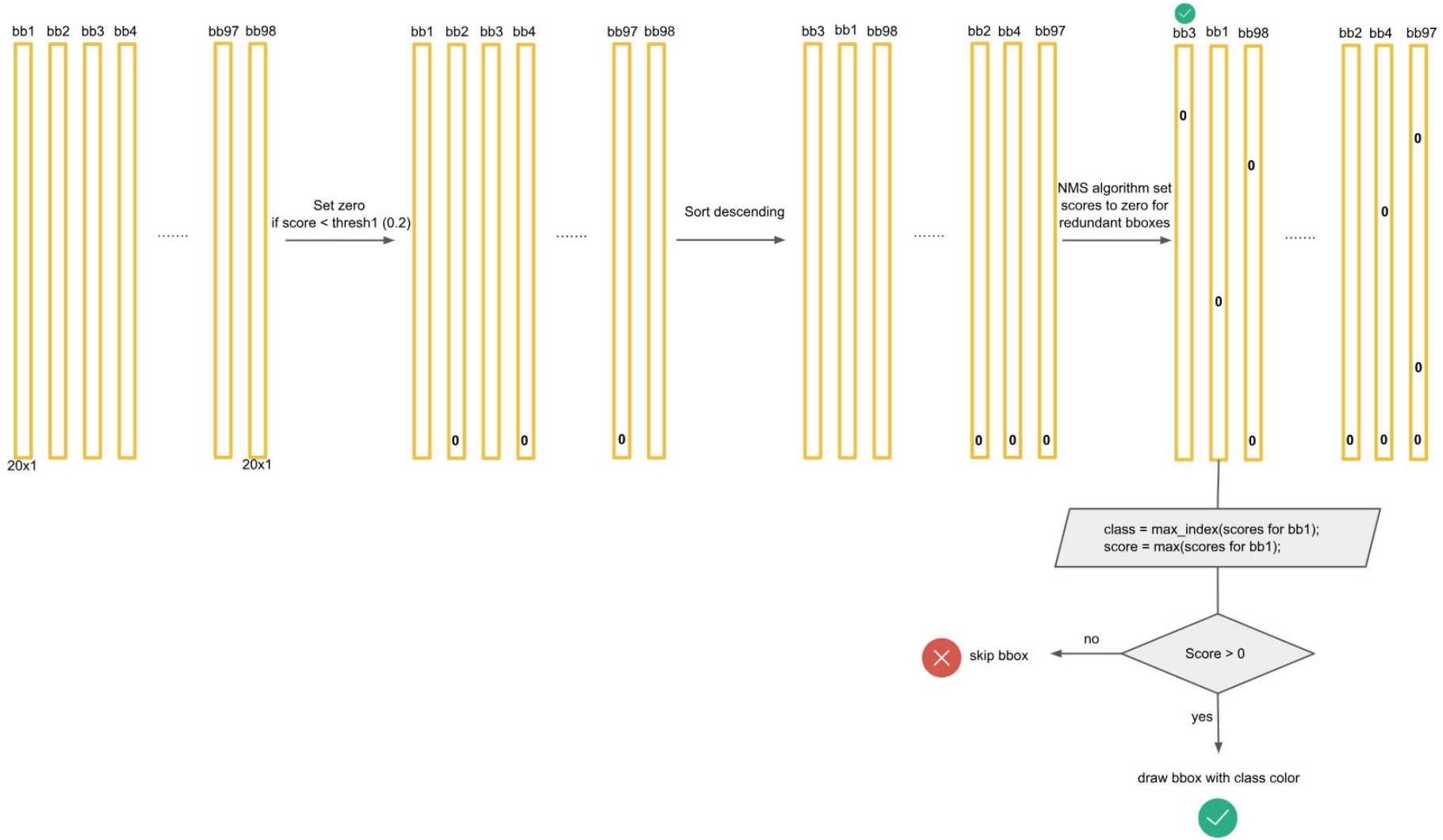


Select bboxes to draw by class score values

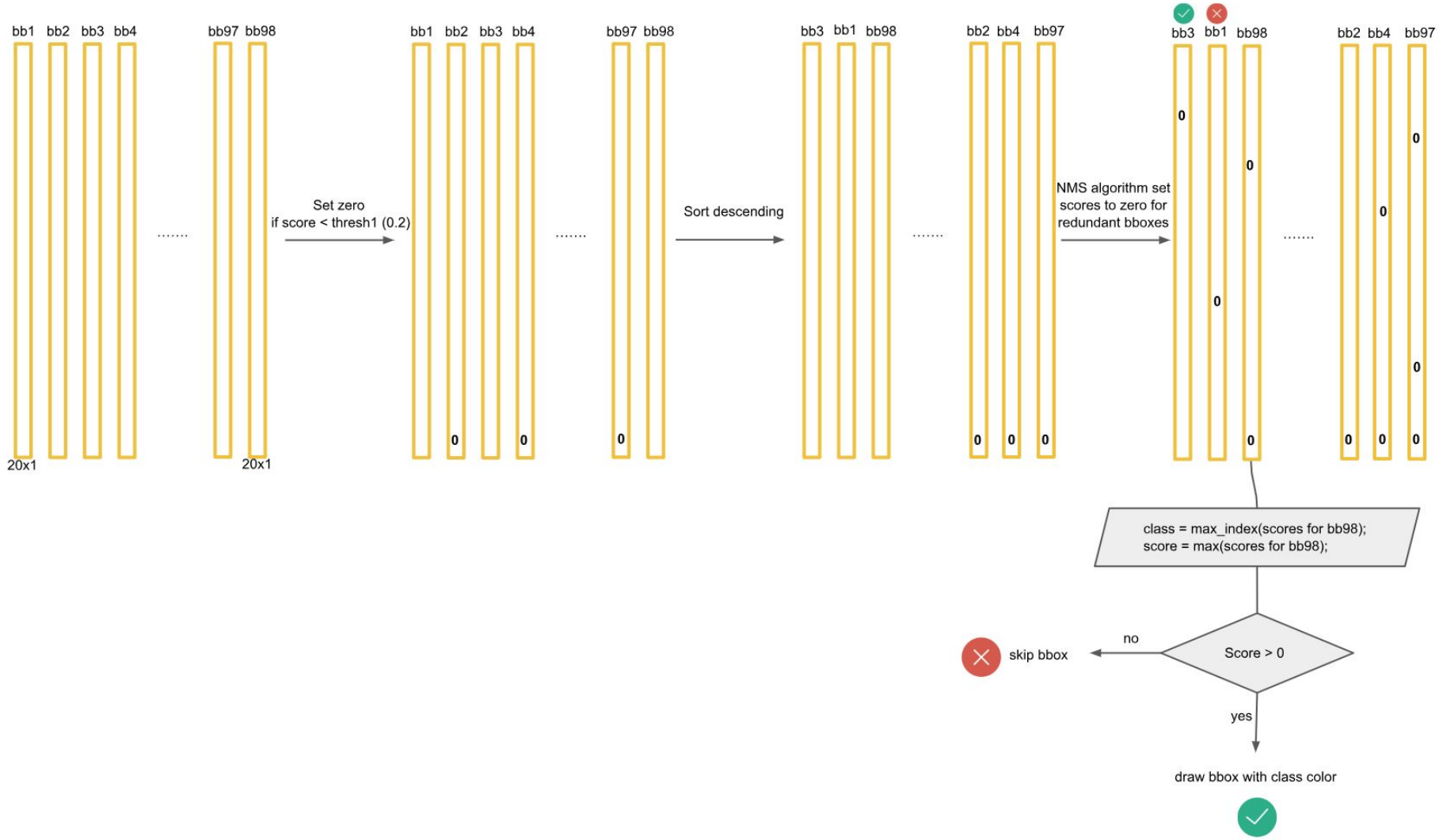
YOLO


bb1 bb2 bb3 bb4
20x1
bb97 bb98
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

YOLO


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

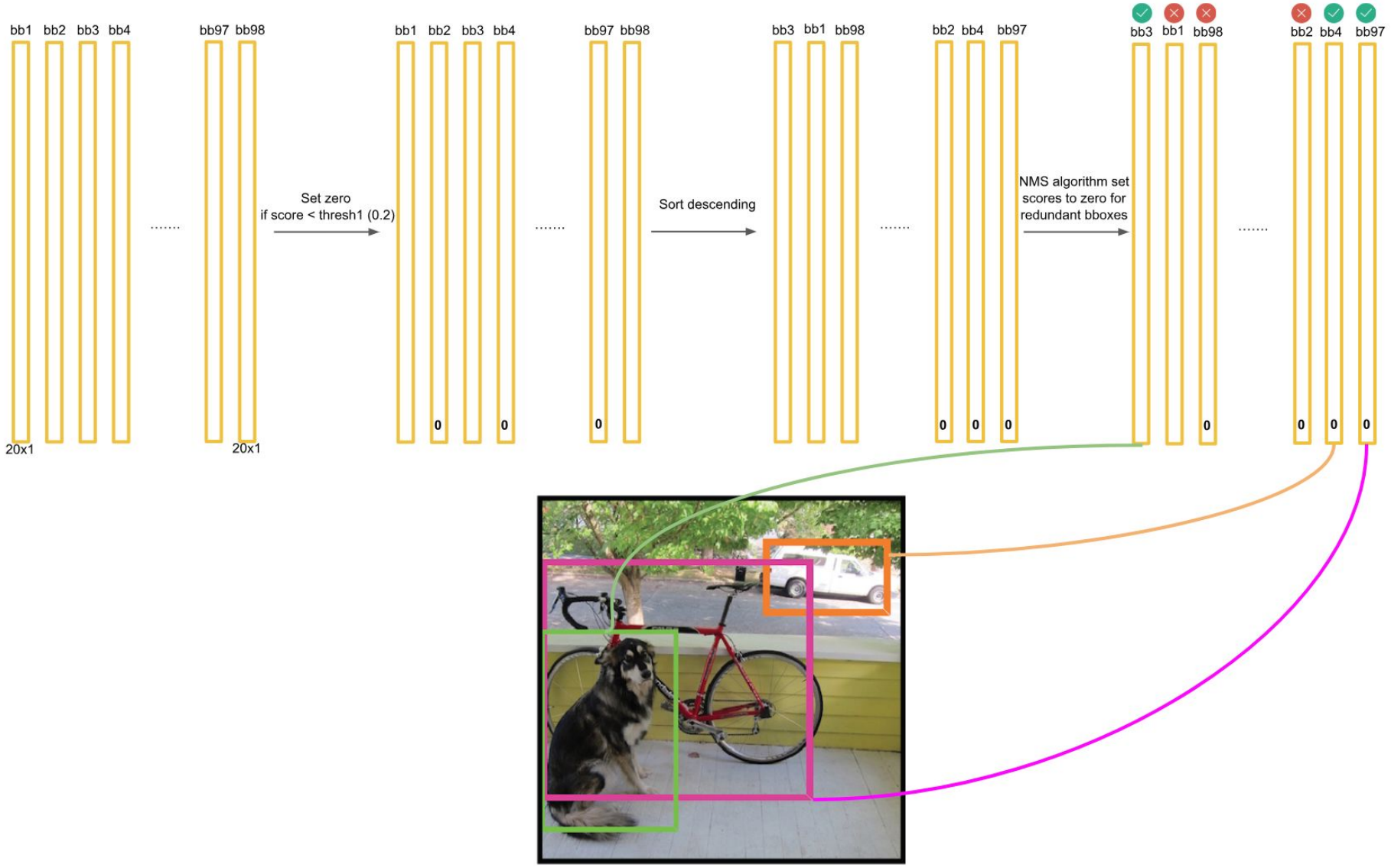
YOLO


bb1 bb2 bb3 bb4
20x1
bb97 bb98
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 bb98);
score = max(scores for bb98);
Score > 0
no
skip bbox
yes
draw bbox with class color

YOLO

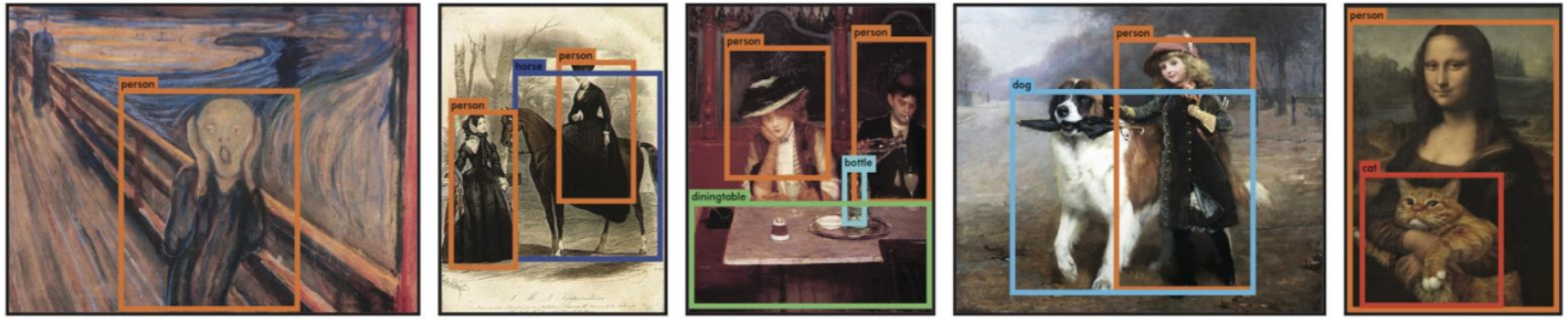

bb1 bb2 bb3 bb4
20x1
bb97 bb98
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4
bb97 bb98
Sort descending
bb3 bb1 bb98
bb2 bb4 bb97
NMS algorithm set
scores to zero for
redundant bboxes
bb3 bb1 bb98
bb2 bb4 bb97
class = max_index(scores for bb97);
score = max(scores for bb97);
Score > 0
no
skip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

YOLO


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

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



YOLO V1 - V8 (Improvement Summaries)

YOLO1

1. The main ideas of target detection

- **Regression-Based Detection**: YOLOv1 treats object detection as a regression problem, directly predicting bounding box coordinates and class probabilities from the input image.
- **Grid Division**: The image is divided into an S x S grid. Each grid cell is responsible for detecting objects whose centers fall within it.
- **Bounding Box Prediction**: Each grid cell predicts B bounding boxes, each with coordinates (x, y, w, h), a confidence score, and class probabilities.
- **Single Convolutional Network**: Uses a single neural network to perform both localization and classification.

2. Overall network structure

- **Architecture**: Inspired by GoogLeNet but uses a simpler design with 24 convolutional layers followed by 2 fully connected layers.
- **Feature Extraction**: Alternating 1x1 and 3x3 convolutional layers extract features from the input image.
- **Detection Layer**: The final layers output predictions for bounding boxes and class probabilities.

3. Loss function (sum of squared errors)

- **Components**:
  - **Localization Loss**: Penalizes errors in bounding box coordinates (x, y, w, h).
  - **Confidence Loss**: Measures the accuracy of the objectness score (confidence that an object is present).
  - **Classification Loss**: Computes the error in class probability predictions.
- **Sum of Squared Errors**: Uses mean squared error (MSE) for all components, which can cause imbalance due to the different scales of localization and classification errors.

4. Advantages of YOLO1

- **High Speed**: Capable of real-time object detection due to its unified architecture.
- **Global Context**: Considers the entire image when making predictions, leading to fewer false positives in background regions.
- **Simplified Pipeline**: Eliminates the need for region proposals and complex pipelines used in two-stage detectors.

YOLO V1 - V8 (Improvement Summaries)

YOLO2

1. The main ideas of target detection

- **Improved Accuracy and Speed**: Aims to enhance both the accuracy and speed of YOLOv1.
- **Batch Normalization**: Introduces batch normalization to all convolutional layers to stabilize training and improve convergence.
- **High-Resolution Classifier**: Trains the classifier network at higher resolutions for better detection performance.
- **Anchor Boxes**: Incorporates anchor boxes (similar to Faster R-CNN) to predict bounding boxes with predefined shapes.

2. Overall network structure

- **Darknet-19 Backbone**: A new, more efficient network with 19 convolutional layers and 5 pooling layers.
- **Feature Extractor**: Uses continuous 3x3 and 1x1 convolutional layers for feature extraction.
- **Detection Layer**: Outputs predictions using anchor boxes, allowing for better localization.

3. Improvements over YOLO1

- **Batch Normalization**: Reduces internal covariate shift, leading to faster training and improved accuracy.
- **Anchor Boxes**: Uses k-means clustering to determine good priors for anchor boxes, improving recall.
- **Dimension Priors**: Adjusts the network to predict offsets instead of absolute coordinates.
- **Passthrough Layer**: Introduces a passthrough layer to bring fine-grained features from earlier layers to the detection layer, improving small object detection.
- **Multi-Scale Training**: Trains the network with images of varying sizes to make the model robust to different input resolutions.

4. Loss function (sum of squared errors)

- **Similar to YOLOv1**: Continues to use the sum of squared errors loss function.
- **Adjustments**:
    - **Coordinate Prediction**: Predicts offsets relative to the anchor boxes.
    - **Confidence Prediction**: Computes objectness score loss.
    - **Class Prediction**: Calculates classification loss for each predicted box.

5. Advantages of YOLO2

- **Improved Recall and Precision**: Achieves better accuracy than YOLOv1 due to the use of anchor boxes and batch normalization.
- **Real-Time Detection**: Maintains high detection speeds suitable for real-time applications.
- **Better Small Object Detection**: Passthrough layer helps in detecting smaller objects by combining fine-grained features.

YOLO3

1. The main ideas of target detection

- **Multi-Scale Predictions**: Predicts bounding boxes at three different scales to improve detection of objects of various sizes.
- **Residual Blocks**: Incorporates residual connections to ease the training of deeper networks.
- **Feature Pyramid Network (FPN)**: Uses FPN-like structures to combine low-level and high-level features.

2. Overall network structure

- **Darknet-53 Backbone**: An improved network with 53 convolutional layers, utilizing residual connections similar to ResNet.
- **Multi-Scale Detection**: Performs detection at layers with strides of 32, 16, and 8, corresponding to different feature map sizes.
- **No Fully Connected Layers**: Fully convolutional network that can accept images of any size.

3. Improvements over YOLO2

- **Deeper Network**: Darknet-53 is deeper and more powerful than Darknet-19.
- **Residual Connections**: Helps in training deeper networks by mitigating vanishing gradient problems.
- **Multi-Scale Detection**: Enhances detection of small, medium, and large objects by aggregating features from different scales.
- **Bounding Box Predictions**: Each grid cell predicts 3 bounding boxes using 9 anchors in total (3 per scale).

4. Loss function

- **Bounding Box Regression**: Uses logistic regression to predict bounding box coordinates relative to the grid cell.
- **Class Prediction**: Switches from softmax to independent logistic classifiers (sigmoid activations) for multi-label classification.
- **Binary Cross-Entropy Loss**: Applies binary cross-entropy loss for both objectness score and class probabilities.

5. Advantages of YOLO3

- **Improved Accuracy**: Higher mAP (mean Average Precision) due to deeper network and multi-scale predictions.
- **Better Small Object Detection**: Multi-scale approach significantly improves detection of smaller objects.
- **Fast Inference**: Despite being more accurate, it maintains real-time detection speeds.

YOLO4

1. The main ideas of target detection

- **Balancing Speed and Accuracy**: YOLOv4 focuses on enhancing both detection performance and speed for real-time applications.
- **Bag of Freebies (BoF)**: Incorporates techniques that improve accuracy without affecting inference time.
- **Bag of Specials (BoS)**: Adds modules that slightly increase inference cost but provide significant accuracy gains.

2. Overall network structure

- **CSPDarknet53 Backbone**: Uses Cross Stage Partial connections to improve learning capability and reduce memory overhead.
- **Neck**:
  - **Spatial Pyramid Pooling (SPP)**: Increases receptive field and separates the most significant context features.
  - **Path Aggregation Network (PANet)**: Enhances parameter utilization for different detector levels.
- **Head**: Uses YOLOv3 head for prediction, making it compatible with previous versions.

3. Improvements over YOLO3

- **CSPDarknet53**: Improves backbone network efficiency and performance.
- **Mish Activation Function**: Replaces leaky ReLU with Mish for smoother gradients.
- **Cross Mini-Batch Normalization (CmBN)**: Addresses small batch size issues during training.
- **Self-Adversarial Training (SAT)**: Helps the model to generalize better by augmenting images to fool itself.
- **Multiple Data Augmentation Techniques**: Uses Mosaic and CutMix to improve robustness.

4. Loss function

- **Bounding Box Regression**: Employs Complete IoU (CIoU) loss to improve localization by considering overlap area, center distance, and aspect ratio.
- **Class Prediction**: Continues to use binary cross-entropy loss with sigmoid activation for multi-label classification.

5. Advantages of YOLO4

- **State-of-the-Art Performance**: Achieves high accuracy while maintaining real-time speed.
- **Training Accessibility**: Designed to be trained on a single GPU, making it accessible for broader research and application.
- **Enhanced Features**: Incorporates several modern techniques to boost performance without significant computational overhead.

YOLO5

1. The main ideas of target detection

- **Ease of Use**: YOLOv5 emphasizes simplicity and practicality, providing a user-friendly framework for training and deployment.
- **PyTorch Implementation**: Developed in PyTorch for flexibility and accessibility.
- **Model Scalability**: Offers different model sizes (s, m, l, x) to suit various applications.

2. Overall network structure

- **Backbone**: Utilizes CSPDarknet, similar to YOLOv4, for efficient feature extraction.
- **Neck**:
  - **Path Aggregation Network (PANet)**: For feature fusion across scales.
  - **Spatial Pyramid Pooling (SPP)**: To capture features at different scales.
- **Head**: Employs YOLO head layers for multi-scale predictions.

3. Improvements over YOLO4

- **Focus Layer**: Introduces a new layer that slices the input and concatenates channels to reduce spatial dimensions efficiently.
- **AutoAnchor**: Automatically calculates optimal anchor boxes based on the dataset.
- **Enhanced Data Augmentation**: Incorporates Mosaic and MixUp techniques for better generalization.
- **Simplified Training Process**: Streamlines the training pipeline for ease of use.

4. Loss function

- **Bounding Box Regression**: Uses Complete IoU (CIoU) or Generalized IoU (GIoU) loss for better bounding box regression.
- **Class Prediction**: Continues with binary cross-entropy loss with sigmoid activation.
- **Objectness Loss**: Applies binary cross-entropy loss for objectness score.

5. Advantages of YOLO5

- **User-Friendly**: Easy to set up and train with detailed documentation.
- **Fast and Accurate**: Offers a good balance between speed and accuracy, suitable for various applications.
- **Active Community Support**: Frequent updates and a supportive user community.

YOLO6

1. The main ideas of target detection

- **Industry-Oriented Design**: YOLOv6 is tailored for industrial applications, focusing on practical deployment and efficiency.
- **Optimization for Edge Devices**: Designed to perform well on resource-constrained hardware.

2. Overall network structure

- **EfficientRep Backbone**: Inspired by RepVGG, it simplifies the architecture for faster inference.
- **Neck**: Uses a lightweight FPN for feature aggregation.
- **Head**:
  - **Decoupled Head**: Separates classification and regression tasks for better optimization.
  - **Anchor-Free Mechanism**: Optionally uses anchor-free detection to reduce complexity.

3. Improvements over YOLO5

- **Re-parameterization Techniques**: Simplifies the model during inference by merging layers.
- **SimOTA Label Assignment**: Adopts a more efficient label assignment strategy during training.
- **Advanced Training Strategies**: Implements methods like EMA (Exponential Moving Average) and careful hyperparameter tuning.

4. Loss function

- **Bounding Box Regression**: Employs Efficient IoU (EIoU) loss to enhance localization precision.
- **Class Prediction**: Uses binary cross-entropy loss with sigmoid activation.
- **Additional Losses**: May include focal loss to address class imbalance.

5. Advantages of YOLO6

- **High Efficiency**: Optimized for speed without a significant loss in accuracy.
- **Deployment Flexibility**: Suitable for a range of devices, including CPUs and mobile platforms.
- **Simplified Architecture**: Easier to implement and deploy in industrial settings.

YOLO7

1. The main ideas of target detection

- **Maximizing Resource Utilization**: YOLOv7 focuses on efficient use of parameters and computations to achieve better performance.
- **Unified Architecture**: Streamlines the model design for consistency and efficiency.

2. Overall network structure

- **E-ELAN Backbone**: Enhances learning capability by expanding the network's width and depth.
- **Neck**: Utilizes a modified PANet for effective feature aggregation.
- **Head**:
  - **Decoupled Head**: Separates classification and localization tasks.
  - **Anchor-Based Detection**: Continues to use anchor boxes for prediction.

3. Improvements over YOLO6

- **Extended ELAN (E-ELAN)**: Improves upon the ELAN architecture for better feature representation.
- **Model Re-parameterization**: Merges layers during inference to optimize speed.
- **Dynamic Label Assignment**: Enhances training efficiency and accuracy by better matching predictions with ground truth.

4. Loss function

- **Bounding Box Regression**: Uses Extended IoU (EIoU) loss for precise localization.
- **Class Prediction**: Continues with binary cross-entropy loss with sigmoid activation.
- **Auxiliary Losses**: May include additional loss terms to improve convergence.

5. Advantages of YOLO7

- **State-of-the-Art Accuracy**: Achieves high accuracy on benchmark datasets.
- **Efficient Training and Inference**: Optimized for both speed and resource utilization.

YOLO8

1. The main ideas of target detection

- **Unified Task Framework**: YOLOv8 is designed to handle object detection, segmentation, and pose estimation within a single architecture.
- **Modularity and Flexibility**: Allows easy customization and extension for various tasks and datasets.

2. Overall network structure

- **Backbone**: Improved CSP-based network with potential integration of transformer modules.
- **Neck**: Combines PANet with advanced feature fusion techniques.
- **Head**:
  - **Task-Specific Heads**: Separate heads for detection, segmentation, and pose estimation.
  - **Anchor-Free Option**: Supports both anchor-based and anchor-free detection methods.

3. Improvements over YOLO7

- **Transformer Integration**: Incorporates transformer layers for capturing global context.
- **Advanced Augmentation**: Uses methods like self-distillation to enhance model performance.
- **Adaptive Computation**: Adjusts the computational effort based on input complexity for efficiency.
- **Enhanced Segmentation and Pose Estimation**: Extends capabilities beyond object detection.

4. Loss function

- **Bounding Box Regression**: Utilizes Distance IoU (DIoU) or Complete IoU (CIoU) loss for better localization.
- **Class Prediction**: Continues with binary cross-entropy loss with sigmoid activation.
- **Segmentation Loss**: Adds losses like Dice loss or Cross-Entropy loss for segmentation tasks.
- **Pose Estimation Loss**: Incorporates keypoint regression losses for pose estimation.

5. Advantages of YOLO8

- **Versatility**: Handles multiple computer vision tasks in one model.
- **Cutting-Edge Performance**: Employs the latest techniques to achieve high accuracy.
- **User-Friendly**: Designed for ease of use with customizable components.

YOLO V1 - V8 (Improvement Summaries)

From YOLO1 to YOLOv8, the YOLO series has evolved significantly, introducing various innovations to improve object detection performance. Each version builds upon the previous one, enhancing accuracy, speed, and versatility. The progression includes:

- **YOLO1**: Introduced the concept of treating object detection as a regression problem for real-time detection.
- **YOLO2**: Improved accuracy with batch normalization, anchor boxes, and multi-scale training.
- **YOLO3**: Enhanced small object detection with multi-scale predictions and a deeper network.
- **YOLO4**: Balanced speed and accuracy with CSPDarknet53 and introduced advanced data augmentation techniques.
- **YOLO5**: Emphasized ease of use and deployment with a PyTorch implementation and model scalability.
- **YOLO6**: Optimized for industrial applications and edge devices with efficient architecture.
- **YOLO7**: Achieved state-of-the-art performance through efficient architecture and training strategies.
- **YOLO8**: Expanded capabilities to include segmentation and pose estimation within a unified framework.