YOLO Object Detection

-Gitanjit Medhi
-Anant Shankhdhar
-Francis Saikia

Introduction

Abstract:-

1.YOLO is a new approach to object detection that frames object detection as a regression problem to spatially seperated bounding boxes and associated class probabilities.

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

3.YOLO performs object detection much faster than all previous methods. YOLO processes processes real time images at 45 frames per second . Fast YOLO , can do the same at 155 frames per second with double mAP(mean Average Precision)

4.As it processes the full images YOLO gives a more general representation of objects in a picture.Gives less false positives but may miss out small objects.

Previous Methods for object detection:-

These methods take a classifier for the object and evaluate it at various locations and scales of the image.

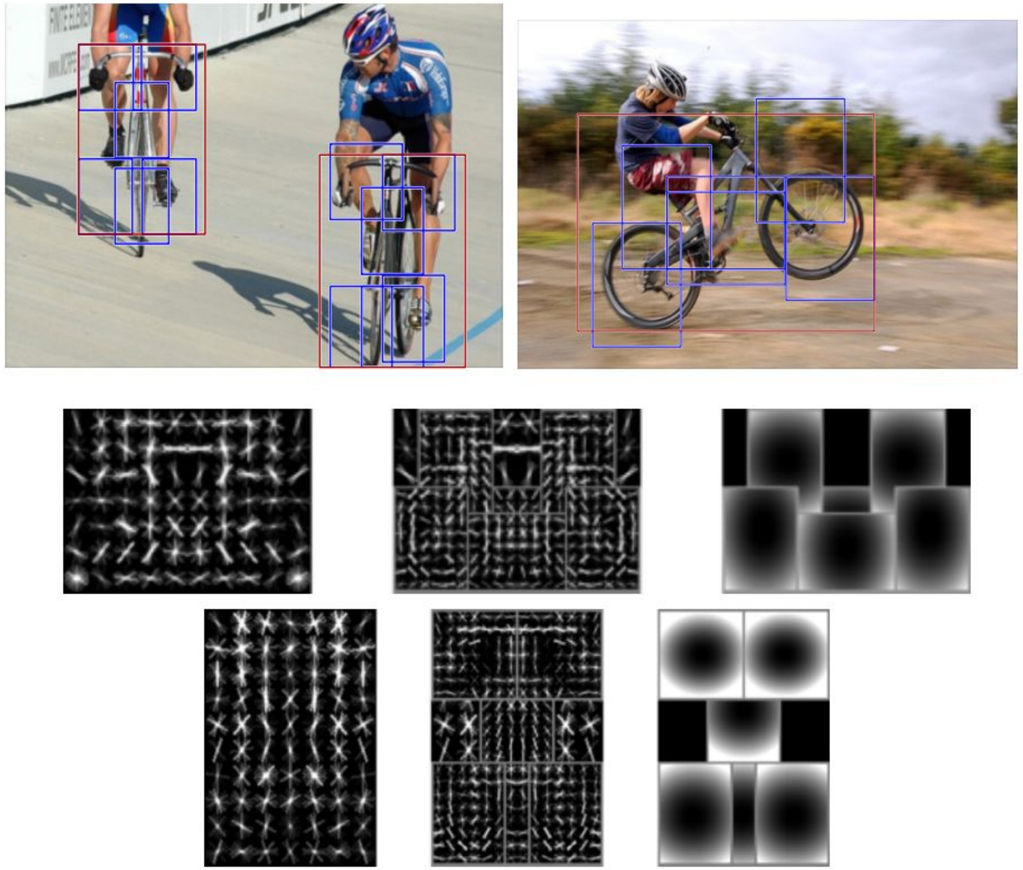
1.Deformable parts Model:-

- Sliding Window -> Classifier (evenly spaced locations)

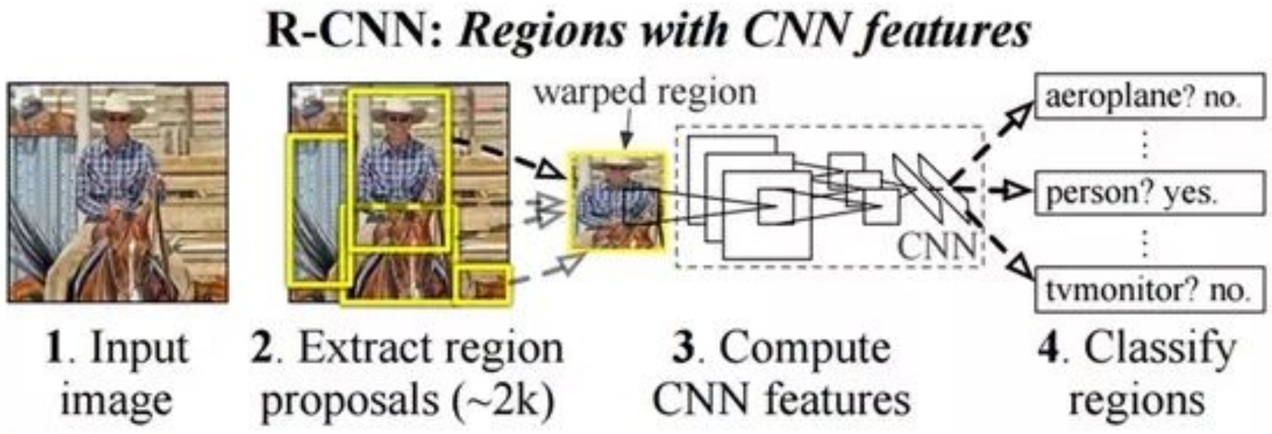
2.Region based CNNs(RCNNs)-

- Region Proposal:- Identifying the potential bounding boxes
- Run Classifiers on each bounding box
- Post processing (Refining bounding boxes,eliminate duplicate detections,and rescore the boxes based on other boxes in the scene)

DPM



RCNN



R-CNN workflow

YOLO Intuition

A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance.

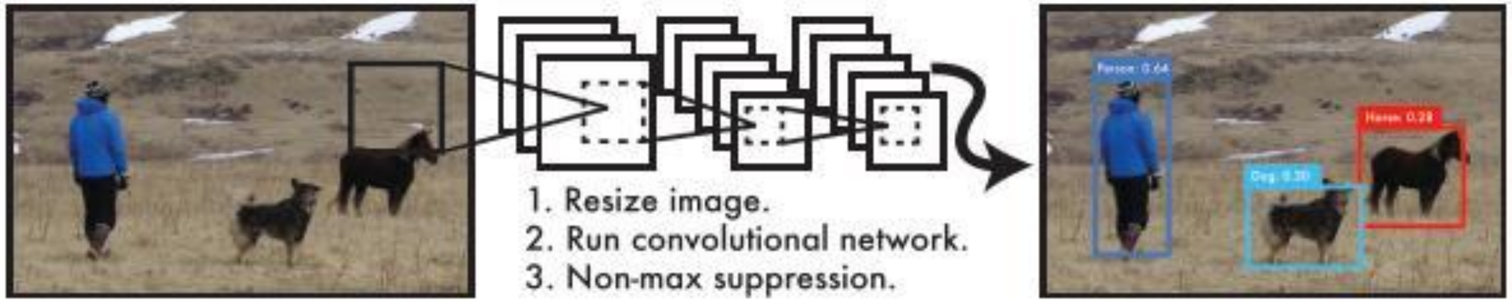


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

YOLO Intuition

Benefit of this process:-

1.Faster detection

2.More generalized representation of objects

3.Less background errors

Problem:-

Lacks in accuracy as compared to networks like rcnns

Unified Detection

1.Our network uses features from the entire image to predict each bounding box
2.For an image all bounding boxes corresponding to all classes are determined simultaneously
3.For this the entire ima  grid

Unified detection

1.Each grid cell predicts B bounding boxes and the confidence scores of all the boxes.

2.The confidence scores shows how confident the model is that an object is present in a bounding box

3.Formula for calculating confidence for each cell

Confidence = Pr(Object) ∗ IOUtruth pred .

Pr(Object)- Probability whether an objects exists in a bounding box

IOU = area of intersection of predicted box and actual object / area of union of predicted box and actual object.

4.Therefore each bounding box gives 5 outputs:-x(x coord of centre),y(y coord of centre),w(width),h(height) and confidence.

5.Confidence = IOU(if image exists) and 0 if it doesn’t

Unified Detection

1.Each grid cell predicts C class probabilities Pr(Classi |Object).

2.These are calculated only if object exists in the grid

3.Only one set of class probabilities are calculated for each cell

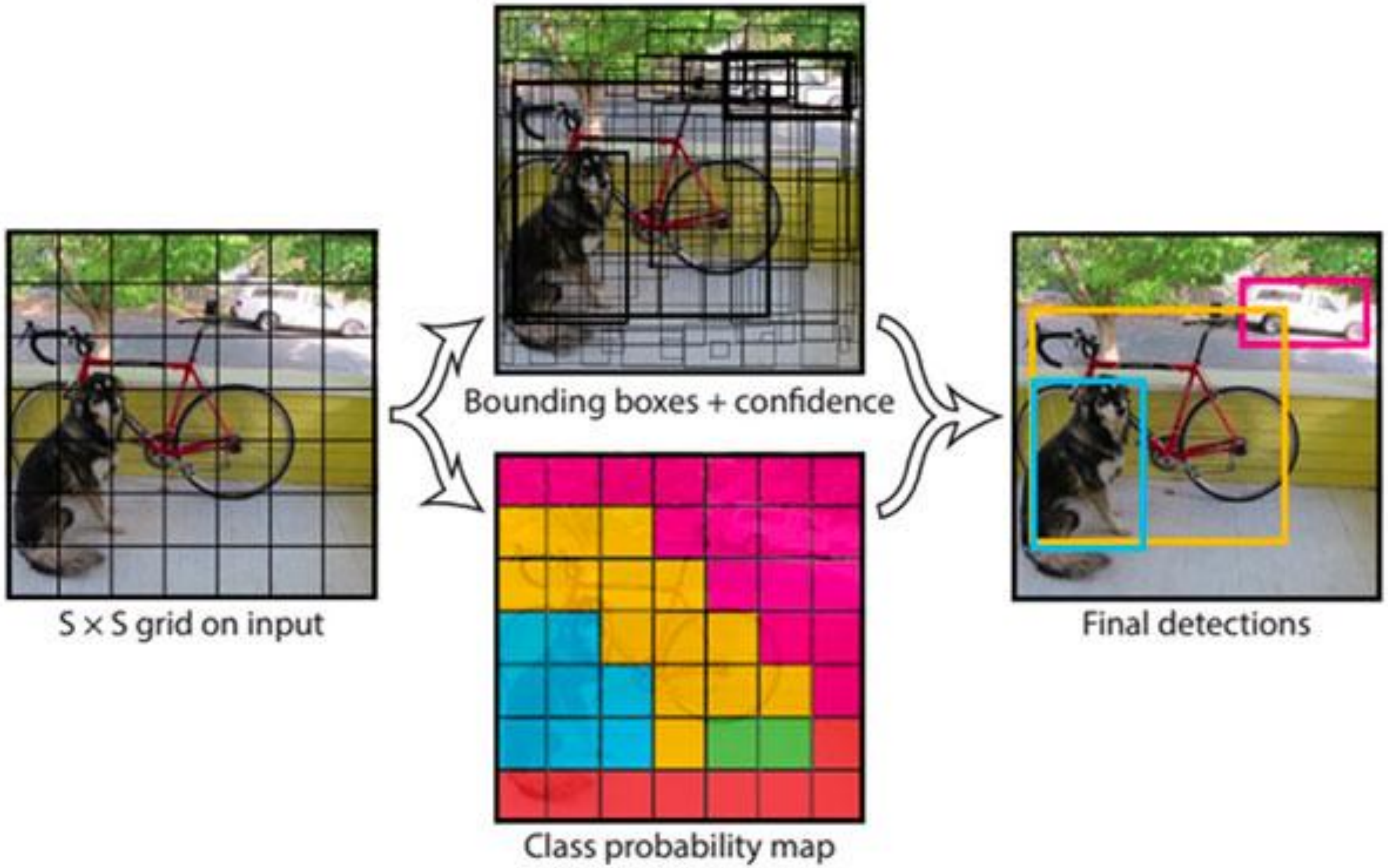
4.At test time we multiply the conditional class probabilities and the individual box confidence predictions

Pr(Classi|Object) ∗ Pr(Object) ∗ IOUtruth pred = Pr(Classi) ∗ IOUtruth pred which gives us class probability scores for each bounding box in a cell.

The prediction will therefore be, an SxSx(B*5+C) tensor.

For evaluating the model on PASCAL VOC dataset , S = 7,B=2,C=20 therefore output 7x7x30

Unified detection



Network Design

1.We implement it as a CNN

2.Initial layers perform feature extraction.Fully connected layers predict outputs

3.The architecture is inspired by GoogleNet model for image classification

4.24 convolutional layers and 2 fully connected layers.

5.Fast YOLO has 9 convolutional layers

Network design

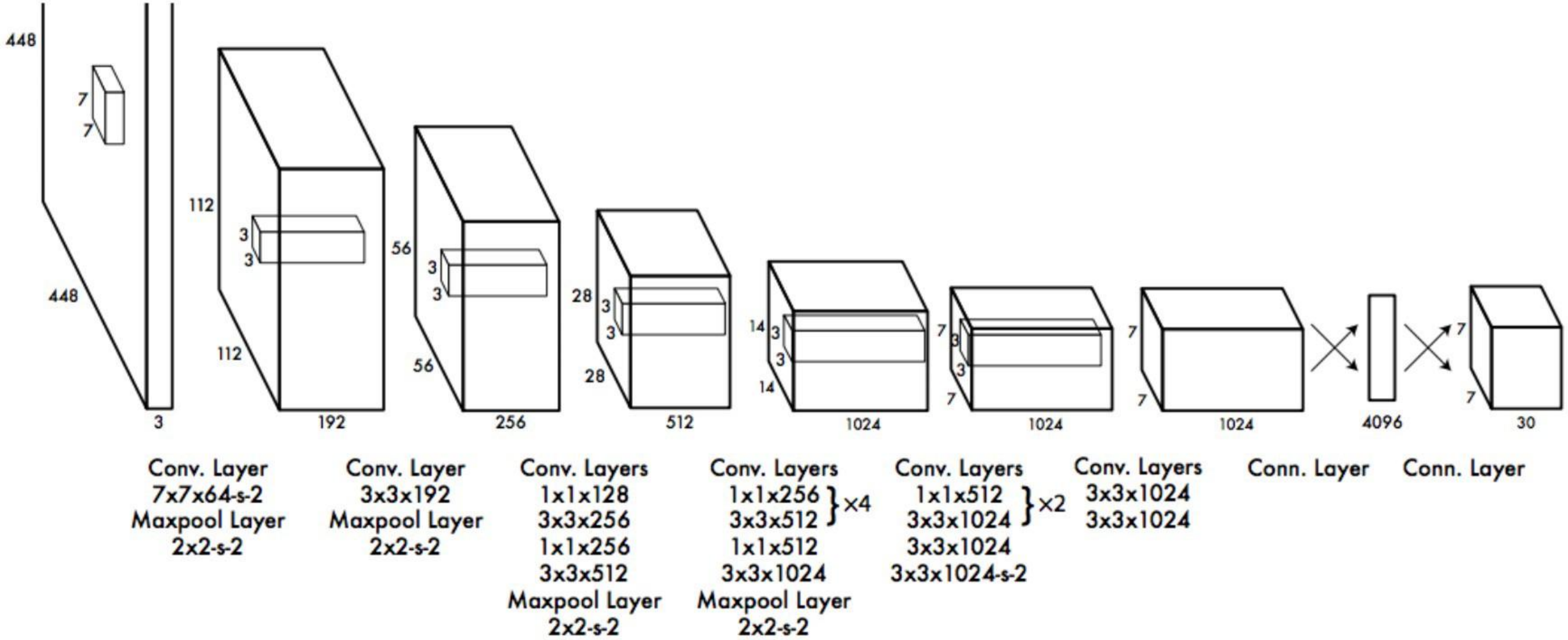


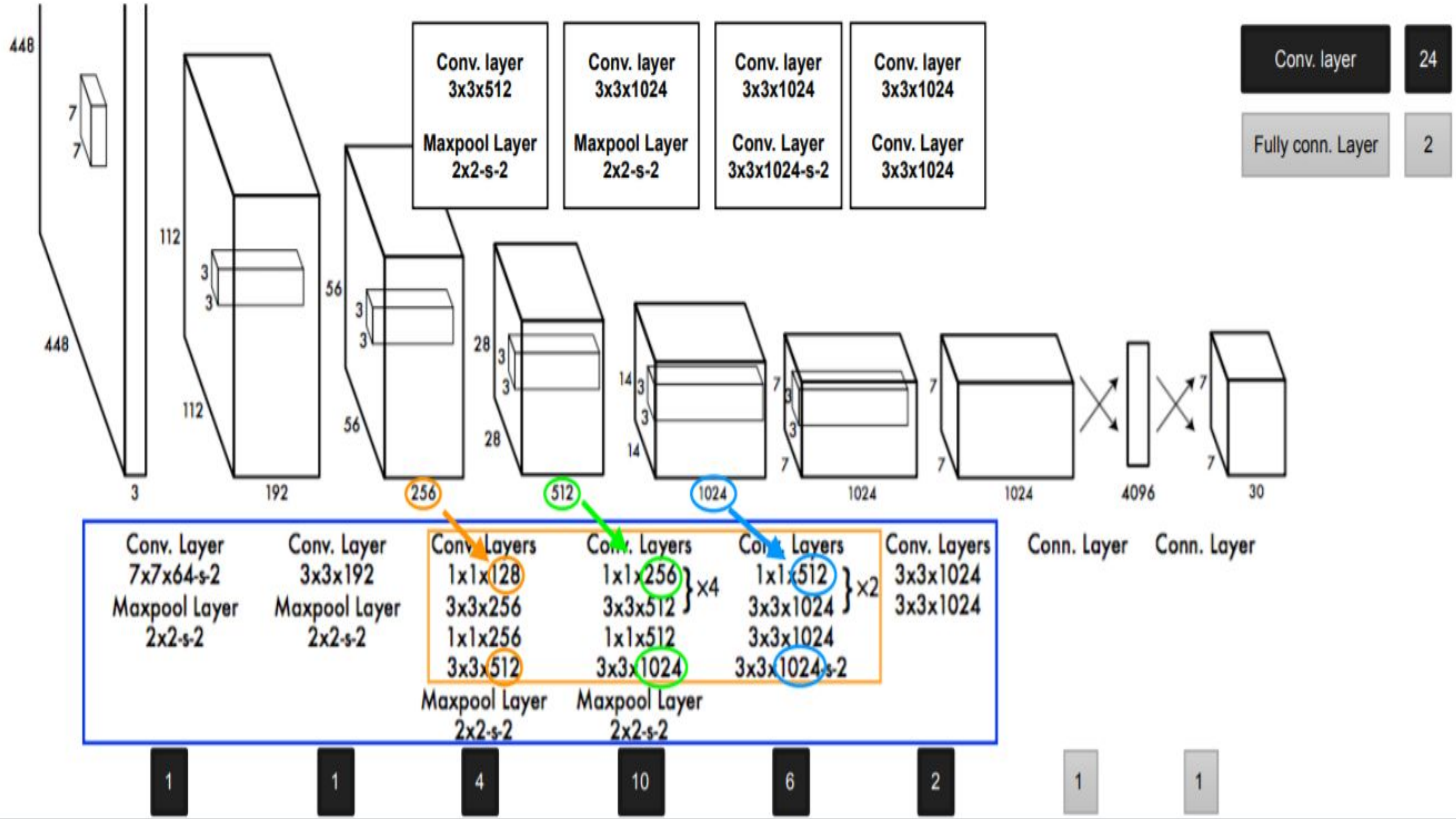
Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Name	Filters	Output Dimension
Conv 1	7 x 7 x 64, stride=2	224 x 224 x 64
Max Pool 1	2 x 2, stride=2	112 x 112 x 64
Conv 2	3 x 3 x 192	112 x 112 x 192
Max Pool 2	2 x 2, stride=2	56 x 56 x 192
Conv 3	1 x 1 x 128	56 x 56 x 128
Conv 4	3 x 3 x 256	56 x 56 x 256
Conv 5	1 x 1 x 256	56 x 56 x 256
Conv 6	1 x 1 x 512	56 x 56 x 512
Max Pool 3	2 x 2, stride=2	28 x 28 x 512
Conv 7	1 x 1 x 256	28 x 28 x 256
Conv 8	3 x 3 x 512	28 x 28 x 512
Conv 9	1 x 1 x 256	28 x 28 x 256
Conv 10	3 x 3 x 512	28 x 28 x 512
Conv 11	1 x 1 x 256	28 x 28 x 256
Conv 12	3 x 3 x 512	28 x 28 x 512
Conv 13	1 x 1 x 256	28 x 28 x 256
Conv 14	3 x 3 x 512	28 x 28 x 512
Conv 15	1 x 1 x 512	28 x 28 x 512
Conv 16	3 x 3 x 1024	28 x 28 x 1024
Max Pool 4	2 x 2, stride=2	14 x 14 x 1024
Conv 17	1 x 1 x 512	14 x 14 x 512
Conv 18	3 x 3 x 1024	14 x 14 x 1024
Conv 19	1 x 1 x 512	14 x 14 x 512
Conv 20	3 x 3 x 1024	14 x 14 x 1024
Conv 21	3 x 3 x 1024	14 x 14 x 1024
Conv 22	3 x 3 x 1024, stride=2	7 x 7 x 1024
Conv 23	3 x 3 x 1024	7 x 7 x 1024
Conv 24	3 x 3 x 1024	7 x 7 x 1024
FC 1	-	4096
FC 2	-	7 x 7 x 30 (1470)

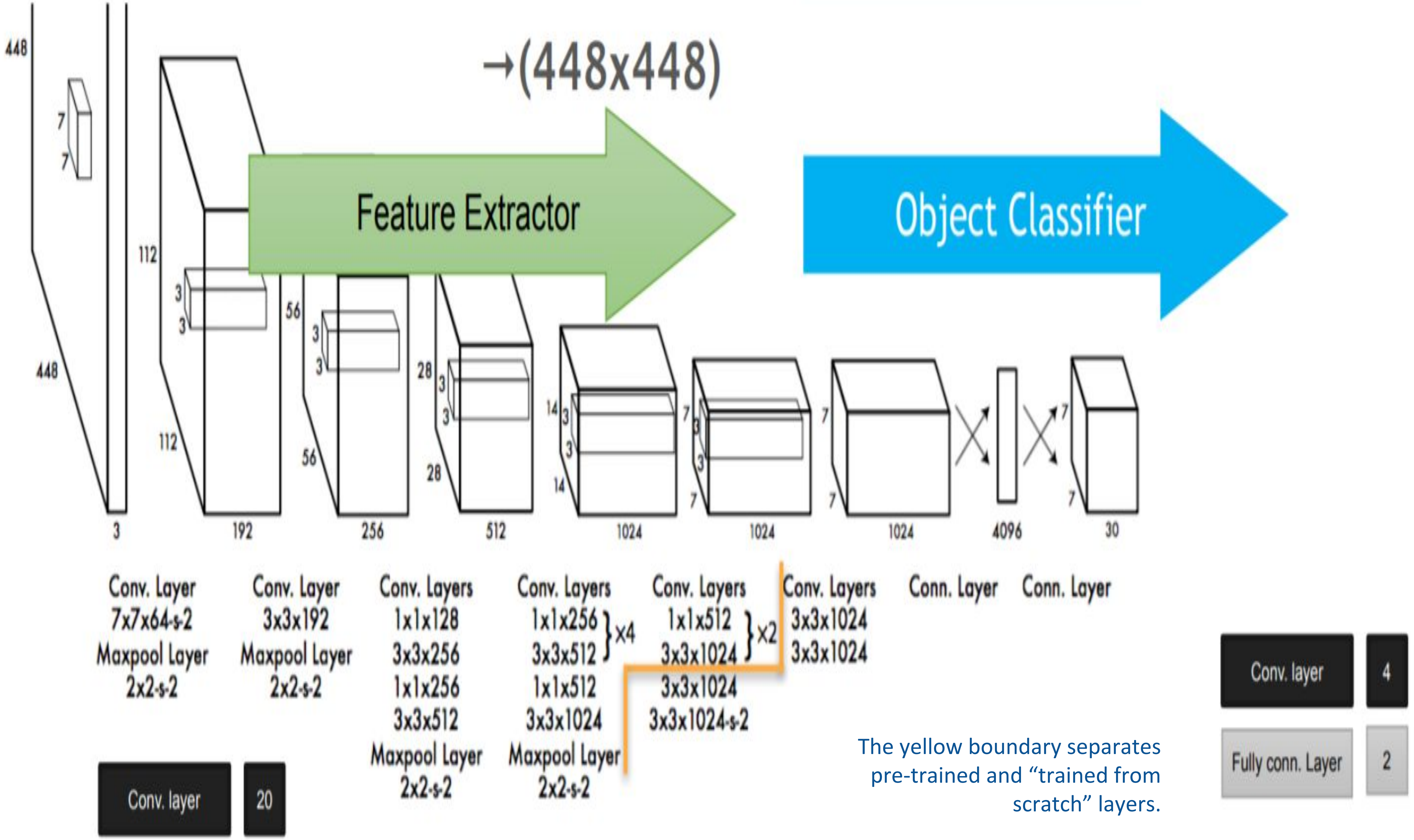
TRAINING

Pretraining

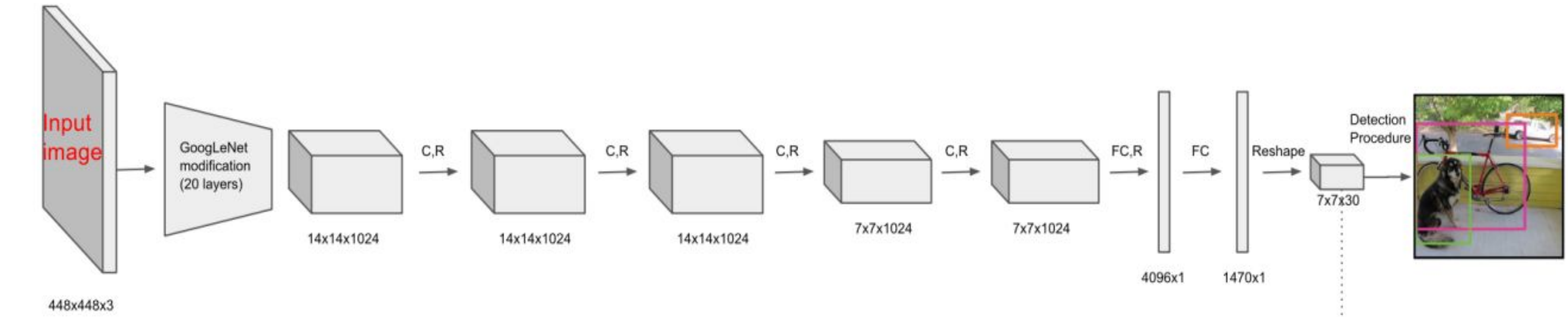
- We shall pretrain our convolutional layers on the ImageNet 1000-class competition dataset.
- We'll use only the first 20 layers (fig. next slide) followed by a average-pooling layer and a fully connected layer.
- Adding both convolutional and connected layers to pretrained networks can improve performance
- Four convolutional layers and two fully connected layers with randomly initialized weights are added to the pretrained network.
- Detection often requires fine-grained visual information so we increase the input resolution of the network from 224×224 to 448×448.

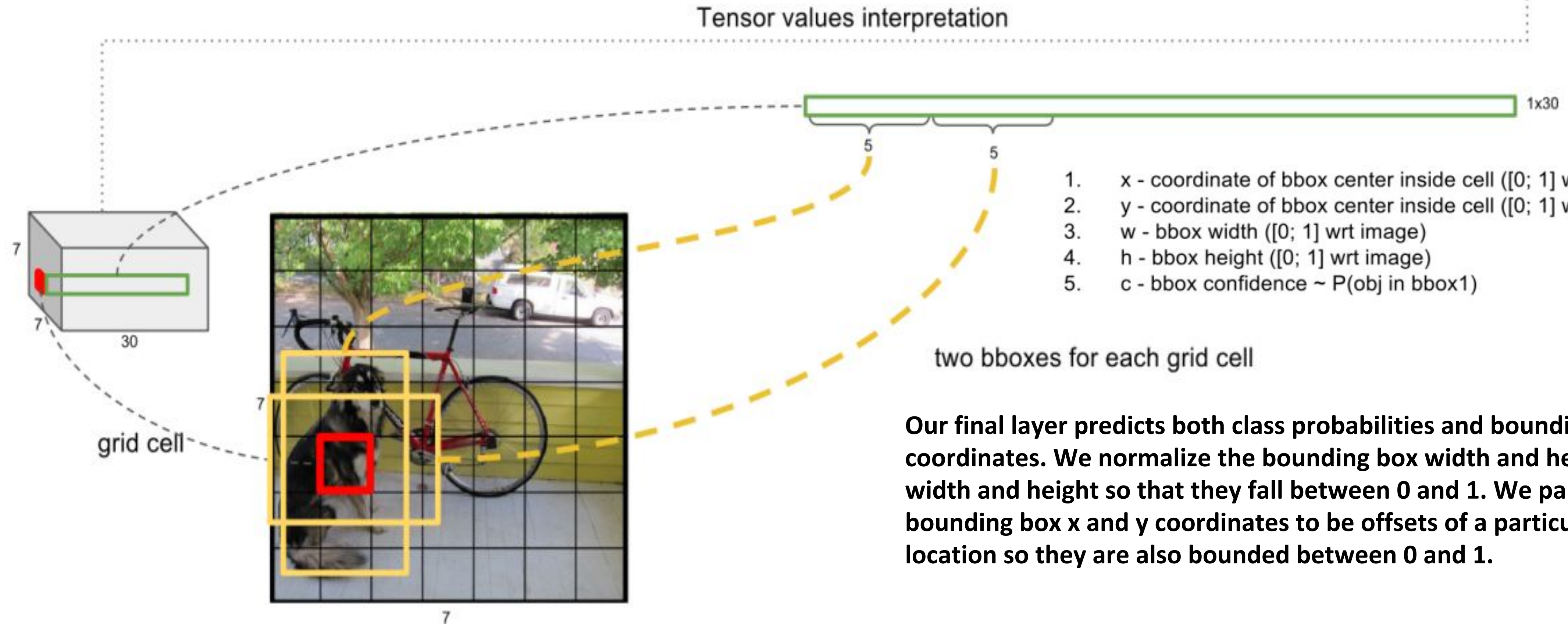


448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
Conv. layer
3x3x512
Maxpool Layer
2x2-s-2
28
3
3
28
512
Conv. layer
3x3x1024
Maxpool Layer
2x2-s-2
14
3
3
14
1024
Conv. layer
3x3x1024
Conv. Layer
3x3x1024-s-2
7
3
3
7
1024
Conv. layer
3x3x1024
Conv. Layer
3x3x1024
7
7
1024
4096
7
7
30
Conv. layer
24
Fully conn. Layer
2
Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2
Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x256
3x3x512
}×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2
Conv. Layers
1x1x512
3x3x1024
}×2
3x3x1024
3x3x1024-s-2
Conv. Layers
3x3x1024
3x3x1024
Conn. Layer
Conn. Layer
1
1
4
10
6
2
1
1



The yellow boundary separates pre-trained and "trained from scratch" layers.



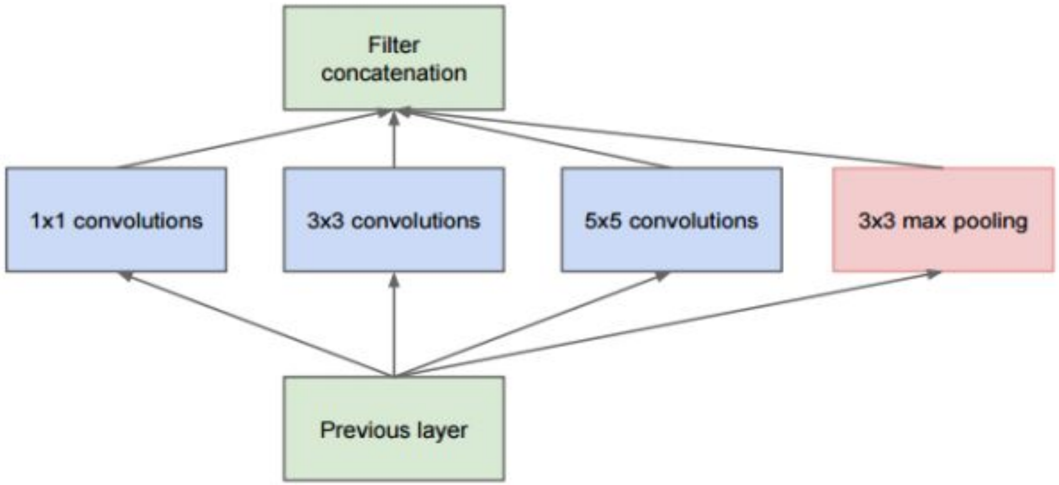


Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they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

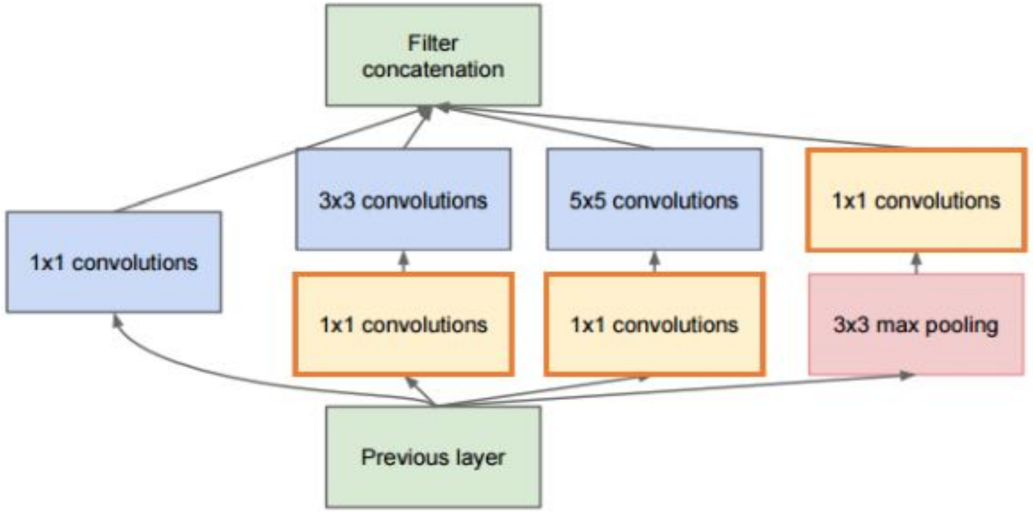
LOSS FUNCTION (SSE)

- Sum-squared error has been used because it is easy to optimize, however it does not perfectly align with our goal of maximizing average precision.
- It considers localisation loss to be equal to the classification error.
- This pushes the “confidence” scores of no-object cells towards zero, often overpowering the gradient from cells that do contain objects.
- This can lead to model instability(kind of overfitting to object containing cells), causing training to diverge early on.
- We increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don’t contain objects. We use two parameters, λcoord(>1) and λnoobj(<1) to accomplish this. We set λcoord = 5 and λnoobj = .5.

Inception Module (1x1 convolution for dimension reductions)



(a) Inception module, naïve version



(b) Inception module with dimension reductions

sum of squared errors of prediction (SSE), is the sum of the squares of residuals (deviations predicted from actual empirical values of data). It is a measure of the discrepancy between the data and an estimation model. A small RSS indicates a tight fit of the model to the data. It is used as an optimality criterion in parameter selection and model selection.

$$SSE = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

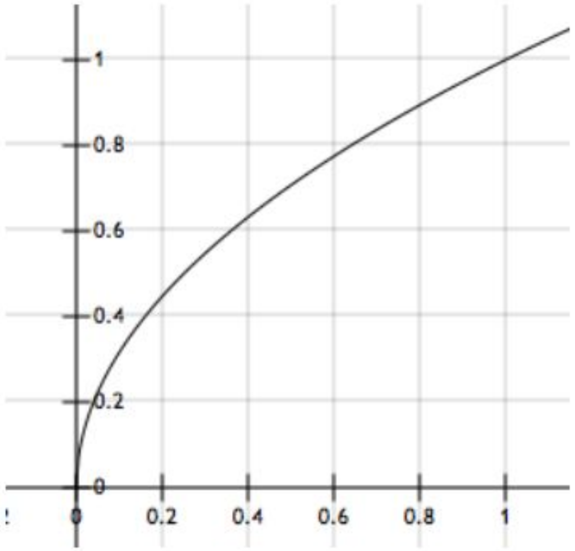
The loss function

loss function:

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

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

Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.



The loss function

loss function:

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

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

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

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

Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

Train Strategy

epochs=135

batch_size=64

momentum_a = 0.9

decay=0.0005

lr=[10^{-3}, 10^{-2}, 10^{-3}, 10^{-4}]

dropout_rate=0.5

augmentation
=[scaling, translation, exposure, saturation]

- We train the network for about 135 epochs on the training and validation data sets from PASCAL VOC 2007 and 2012.
- We use a batch size of 64, a momentum of 0.9 and a decay of 0.0005.
- For the first epochs we slowly raise the learning rate from 10^−3 to 10^−2 . We continue training with 10^−2 for 75 epochs, then 10^−3 for 30 epochs, and finally 10^−4 for 30 epochs.
- A dropout layer with rate = .5 after the first connected layer prevents co-adaptation between layers
- Augmentation: random scaling and translations of up to 20% of the original image size.Random adjustment of exposure and saturation of the image by up to a factor of 1.5 in the HSV color space.

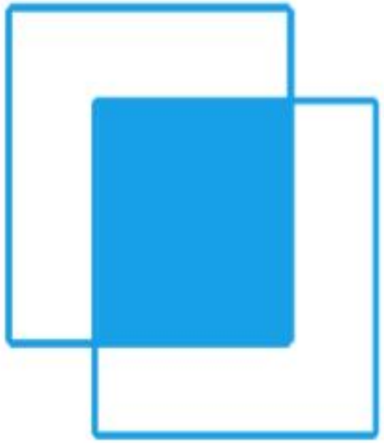
INFERENCE

- We obtained a 7X7X30 tensor(matrix) as an output after running our model on a test image.
- The four convolutional layers in the middle(fig.) , can be replaced with more conv. layers in order to improve performance.
- On PASCAL VOC the network predicts 98(2 bboxes for each of the 7X7=49 cells) bounding boxes per image and class probabilities for each box.
- It is extremely fast at test time since it only requires a single network evaluation, unlike classifier-based methods.
- Normally it’s clear which box an object will belong to, though its possible for large objects or boundary-nearing objects that multiple boxes claim to contain that object.
- **Non-maximal suppression** can be used to fix these multiple detections. While not critical to performance as it is for R-CNN or DPM, non-maximal suppression adds 2- 3% in mAP.

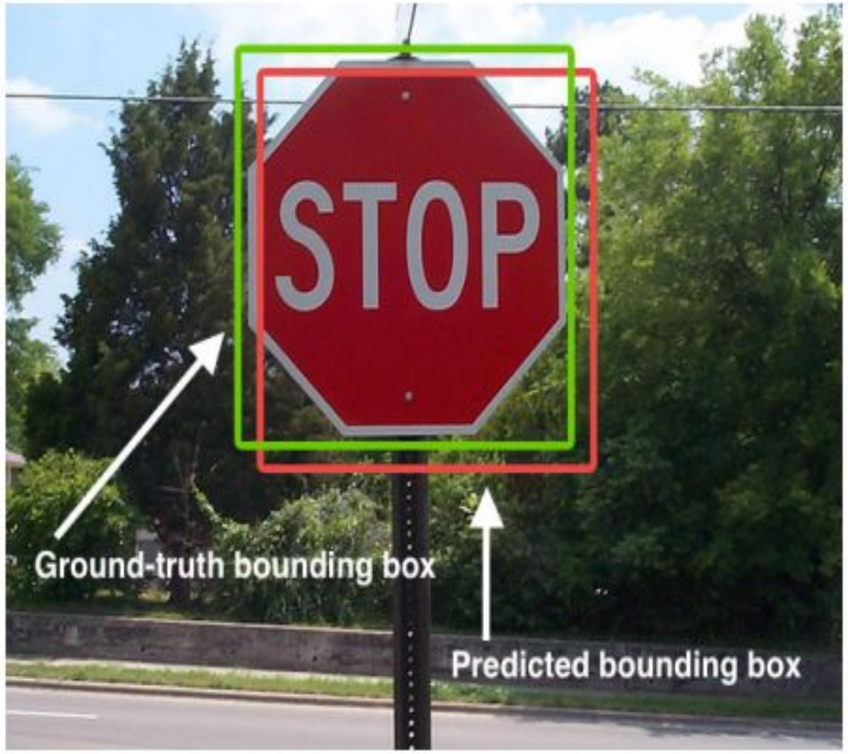
INTERSECTION OVER UNION(IOU)

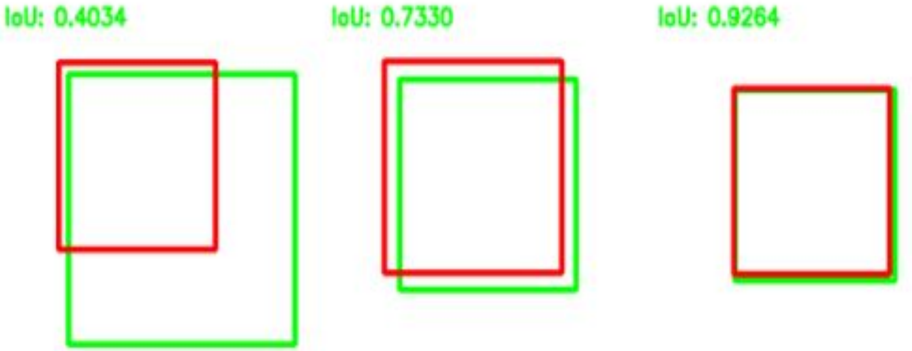
- IoU(pred, truth)=[0, 1]

$$\mathrm{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

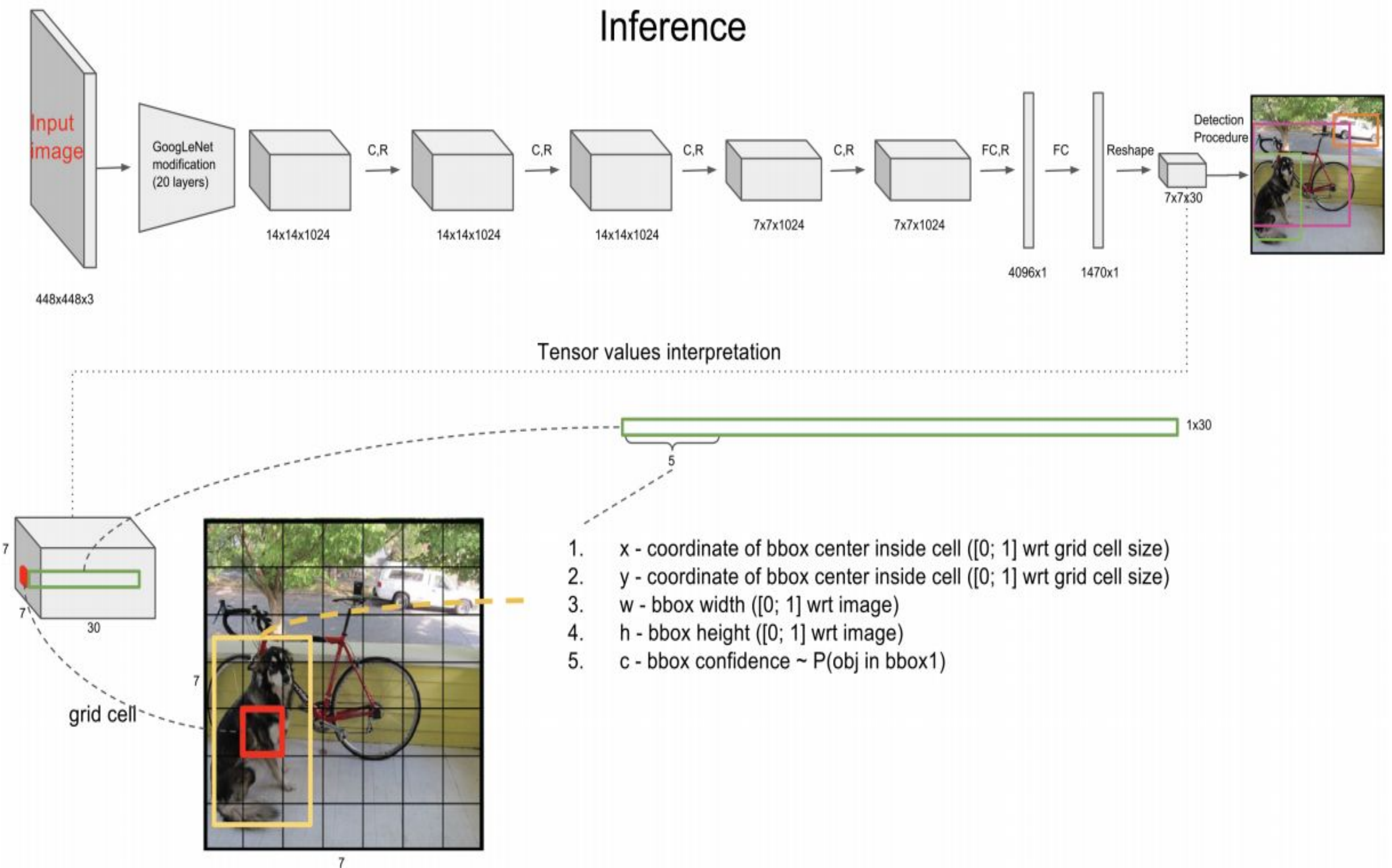


$$\mathrm{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

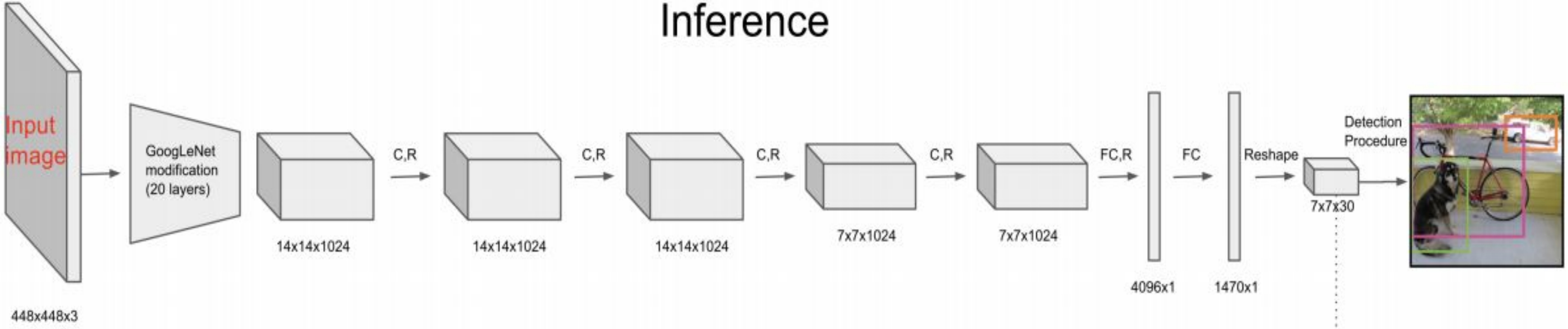




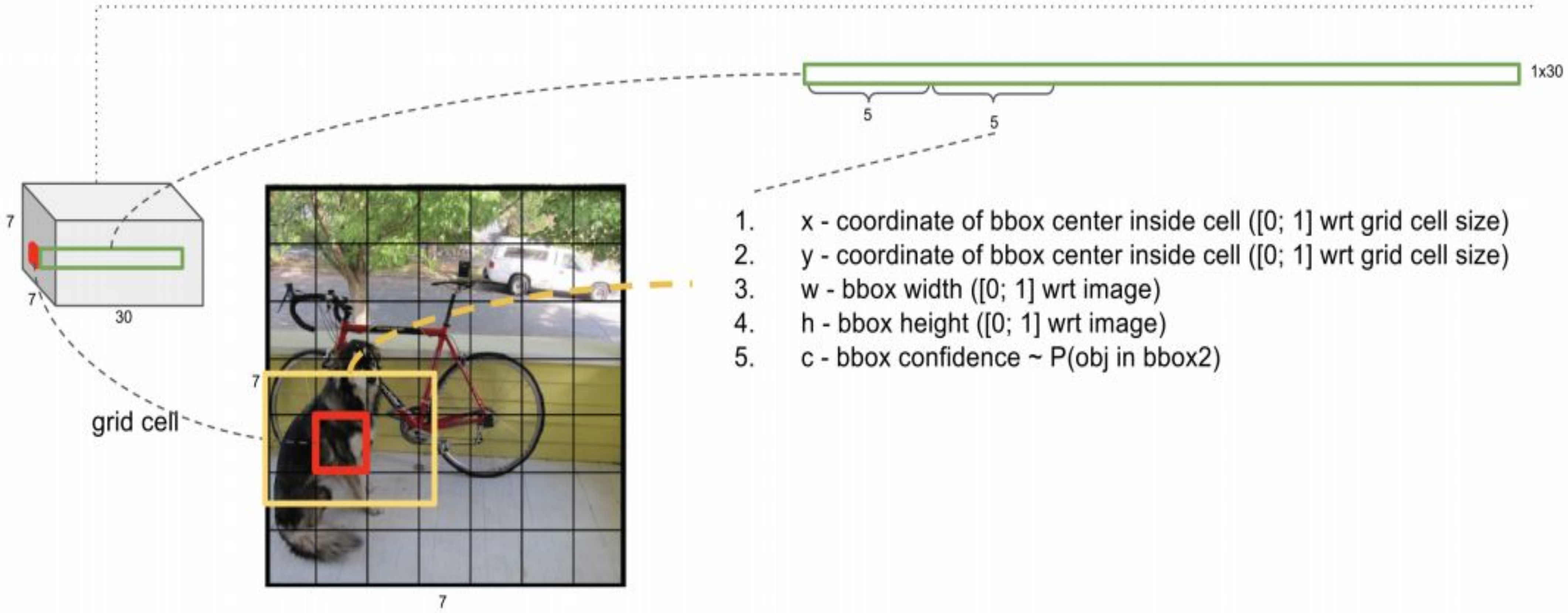
Poor Good Excellent


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)

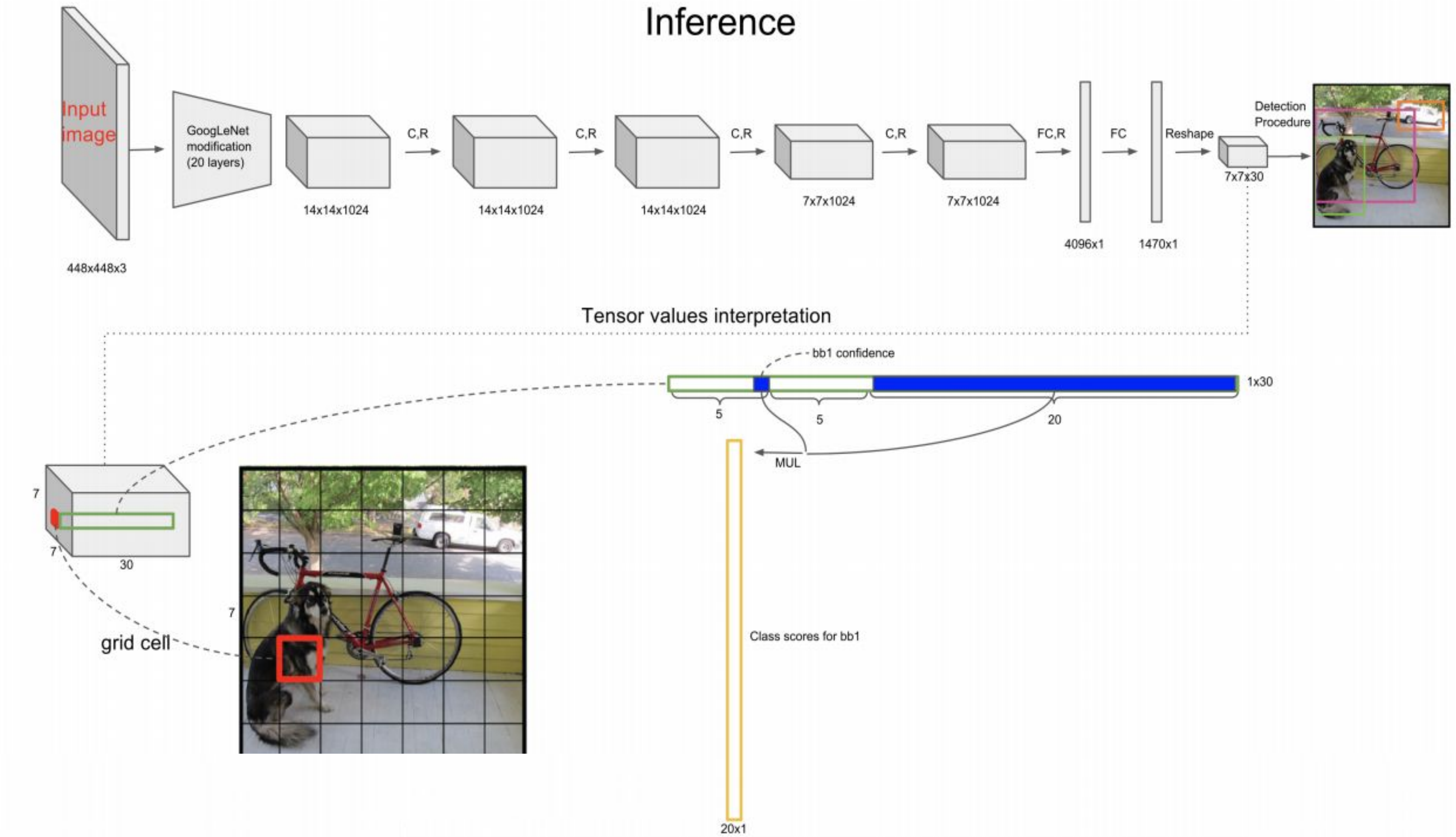
Inference



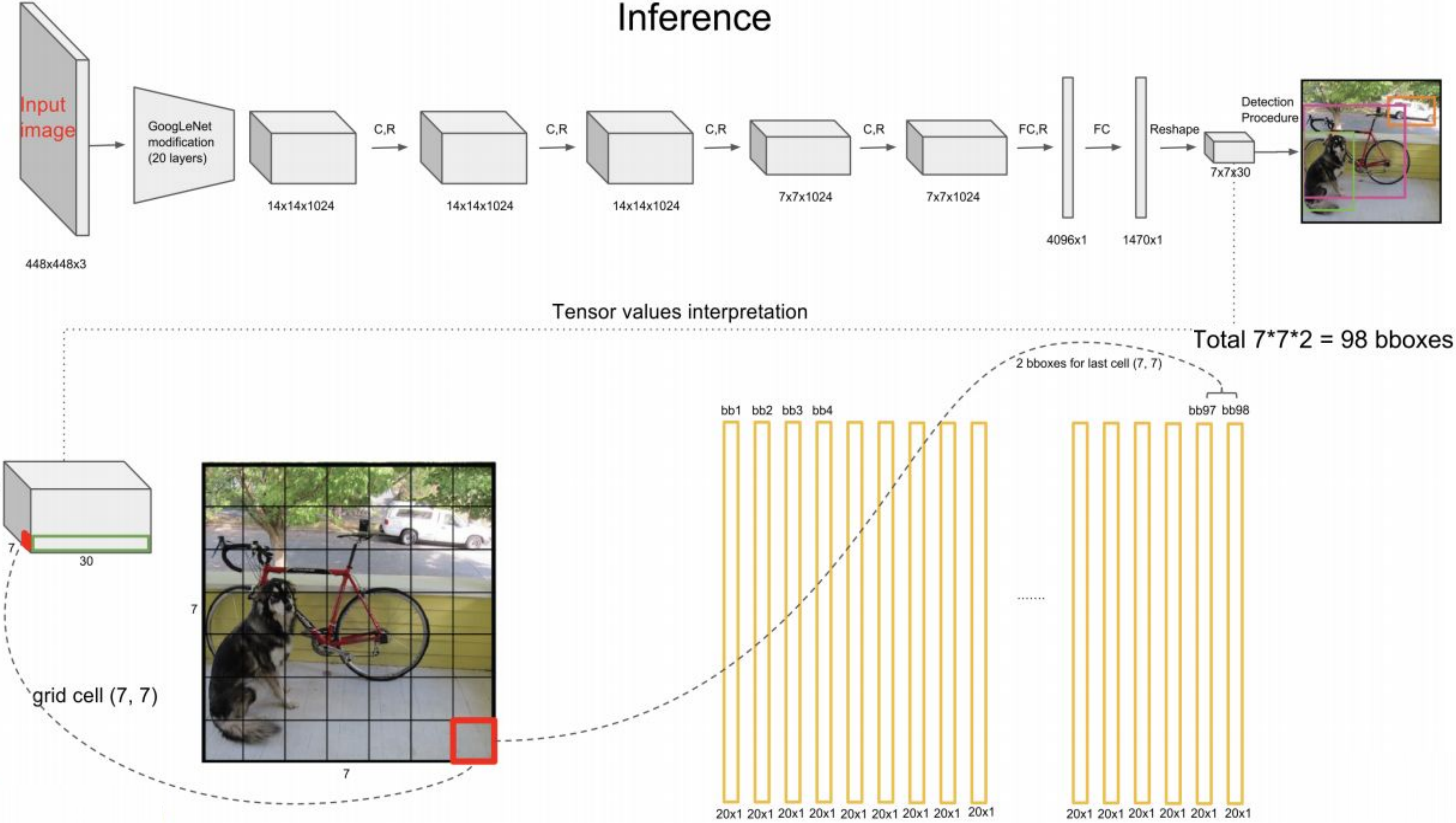
Tensor values interpretation



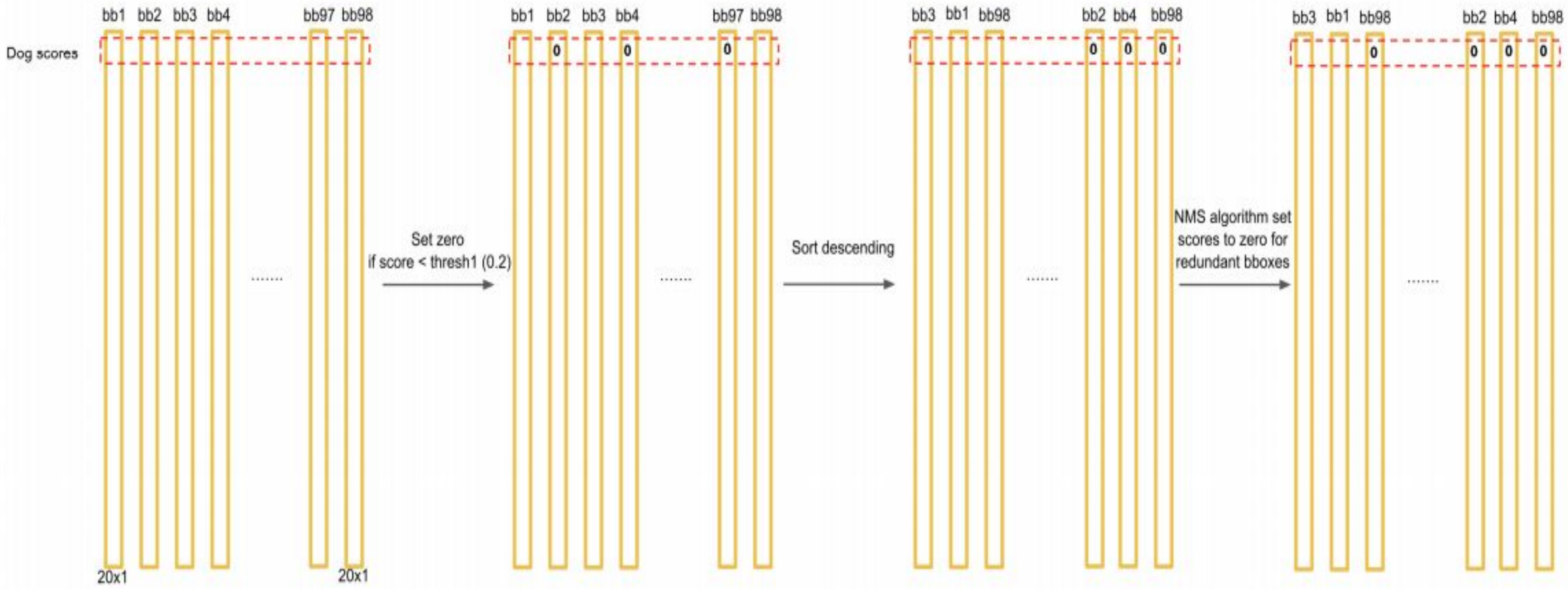
1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox2)


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

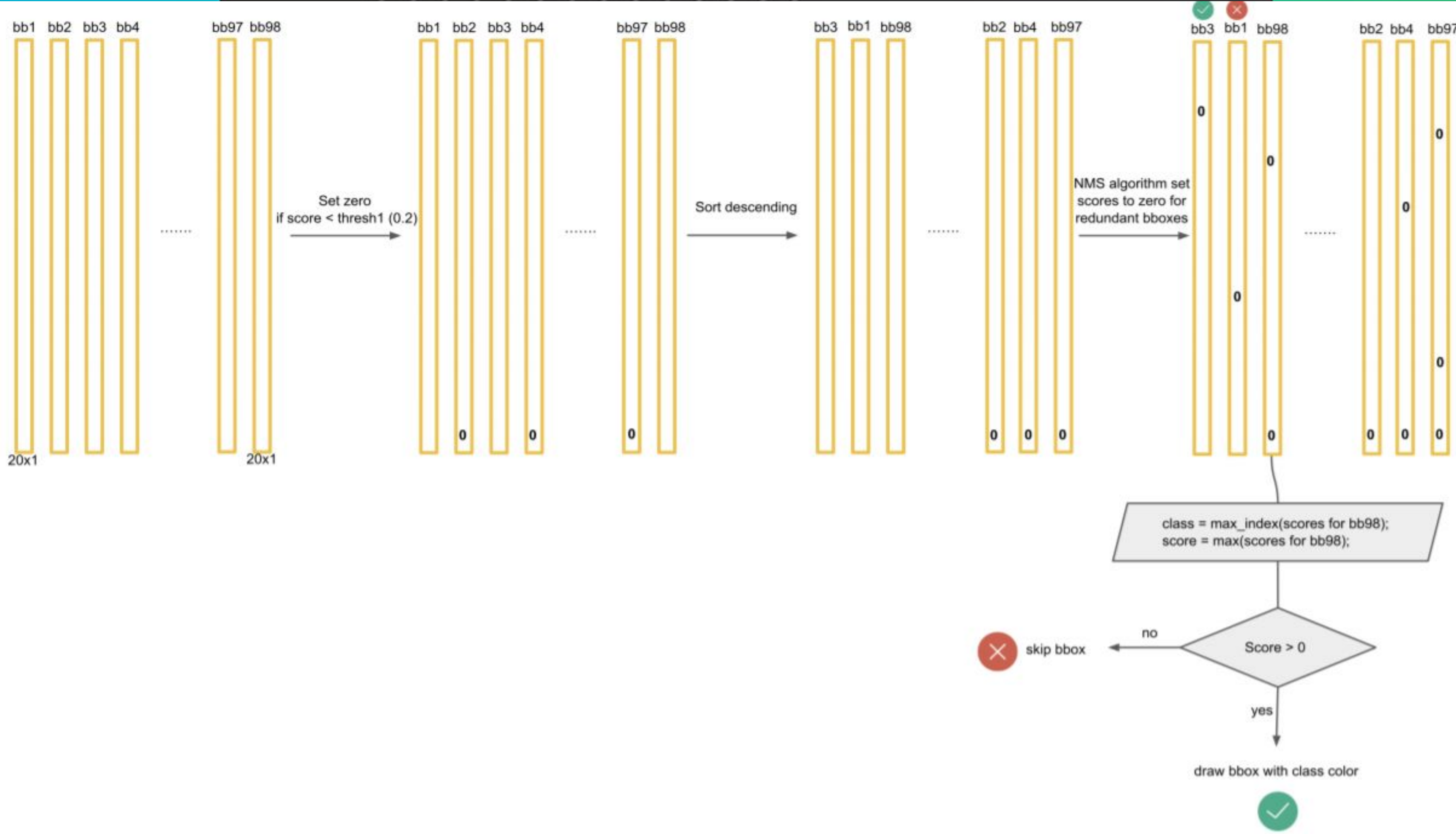
Inference



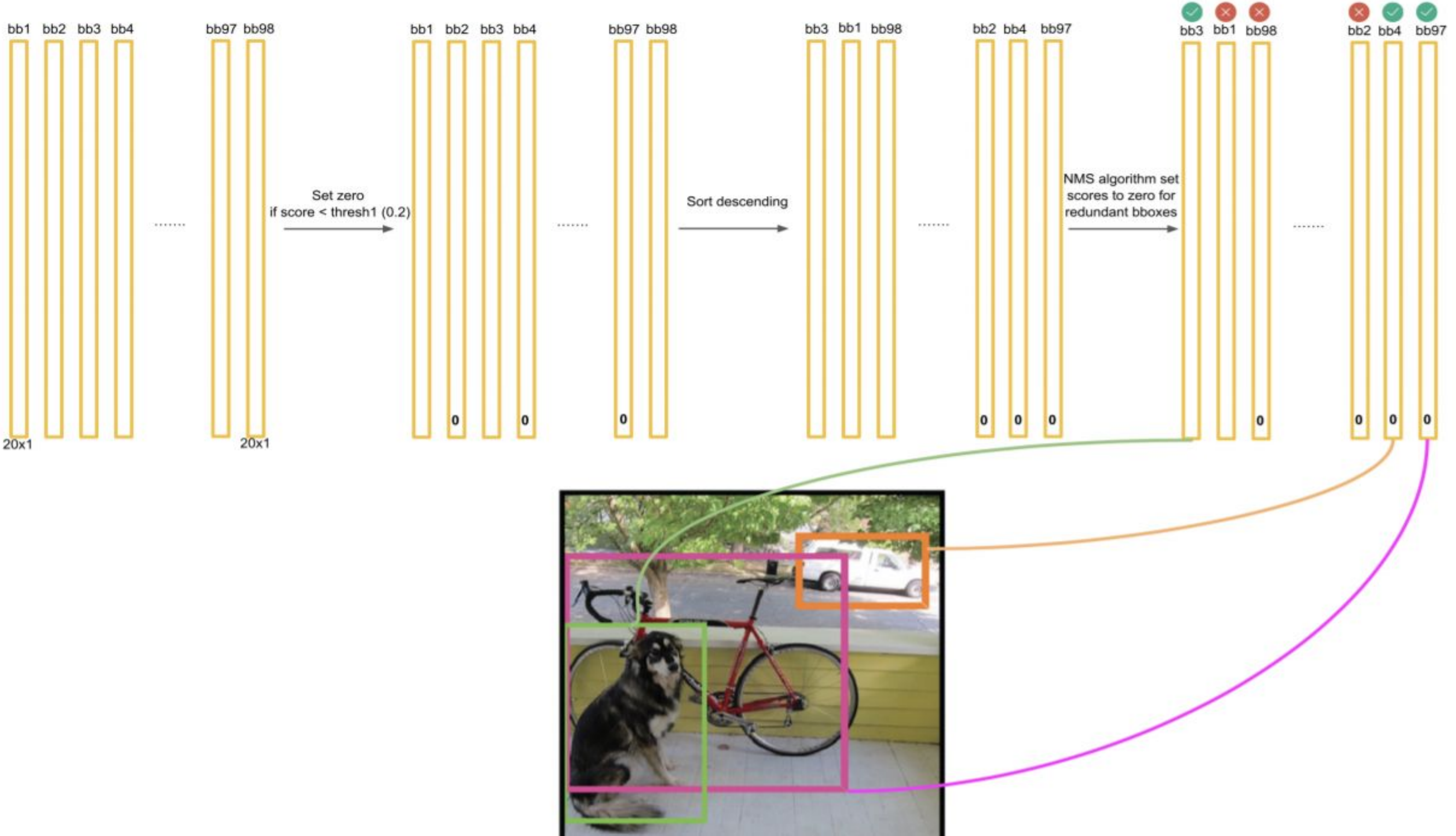
Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
Total 7*7*2 = 98 bboxes
2 bboxes for last cell (7, 7)
7
7
30
grid cell (7, 7)
7
7
bb1 bb2 bb3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



We will look at the NMS algorithm mentioned here in a separate pdf which step-by-step shows the complete algorithm illustratively



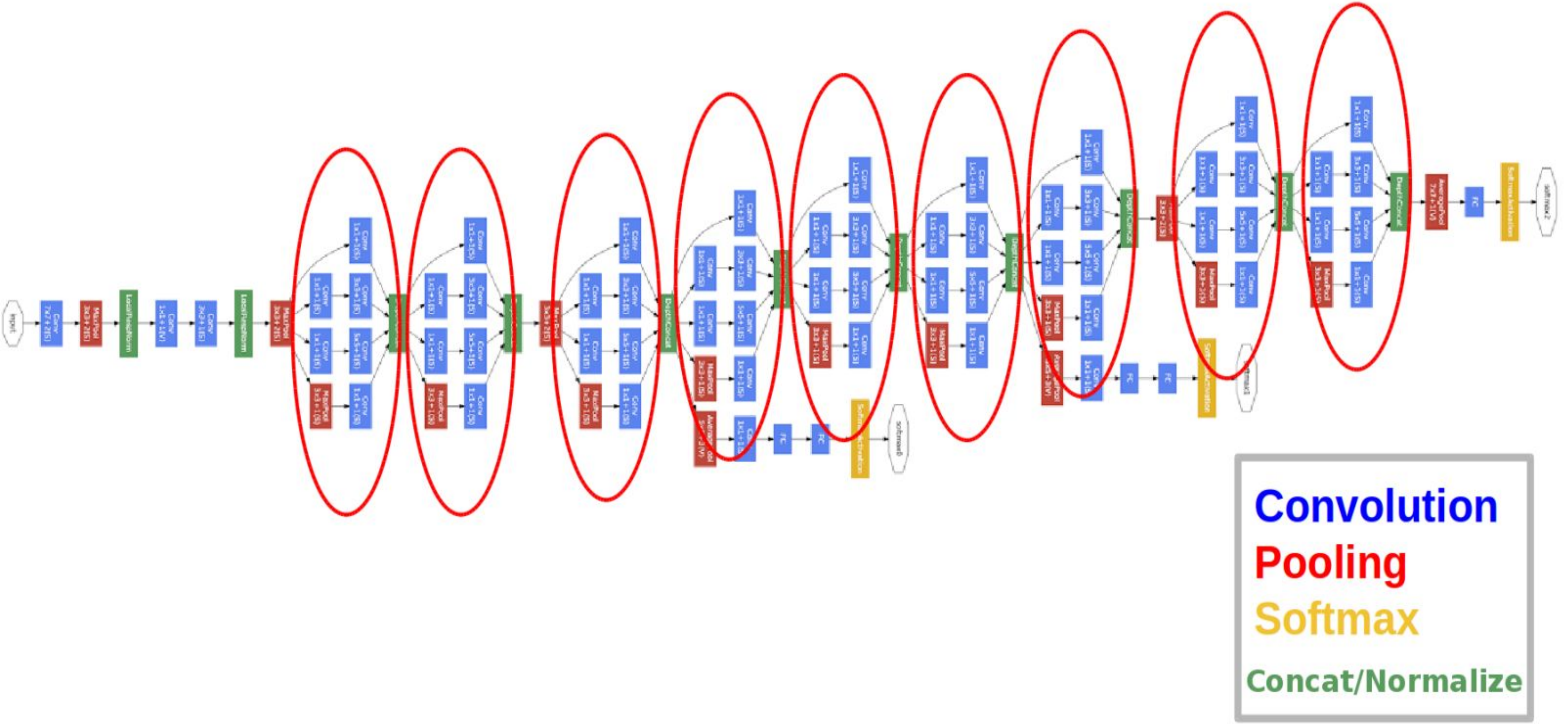
bb1 bb2 bb3 bb4 bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 bb97 bb98
0 0 0
Sort descending
bb3 bb1 bb98 bb2 bb4 bb97
0 0 0
NMS algorithm set scores to zero for redundant bboxes
bb3 bb1 bb98 bb2 bb4 bb97
class = max_index(scores for bb98);
score = max(scores for bb98);
Score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


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

Limitations of YOLO

- **Group of small objects-** model struggles with small objects that appear in groups, such as flocks of birds.
- **Unusual Aspect Ratios-**struggles to generalize to objects in new or unusual aspect ratios or configurations(different from data on which it was trained)
- **Coarse Feature**-uses relatively coarse features for predicting bounding boxes since our architecture has multiple downsampling layers from the input image
- **Localization error of bounding box -**while we train on a loss function that approximates detection performance, our loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU. Our main source of error is incorrect localizations.

GoogLeNet Model Diagram



OTHER MODELS AND EXPERIMENTS

Comparison to other Models

- DPM(Deformable Parts Model) & Overfeat:
  - sliding window
  - use of a disjoint pipeline

- R-CNN & Fast-RCNN
  - selective search to predict b boxes
  - SVM scores boxes
  - linear model prediction

- Deep Multibox
  - region of interest
  - single class prediction
  - just a piece in a bigger pipeline

- Multi-Grasp
  - only grasps a certain region of the image

Experiments

Contents

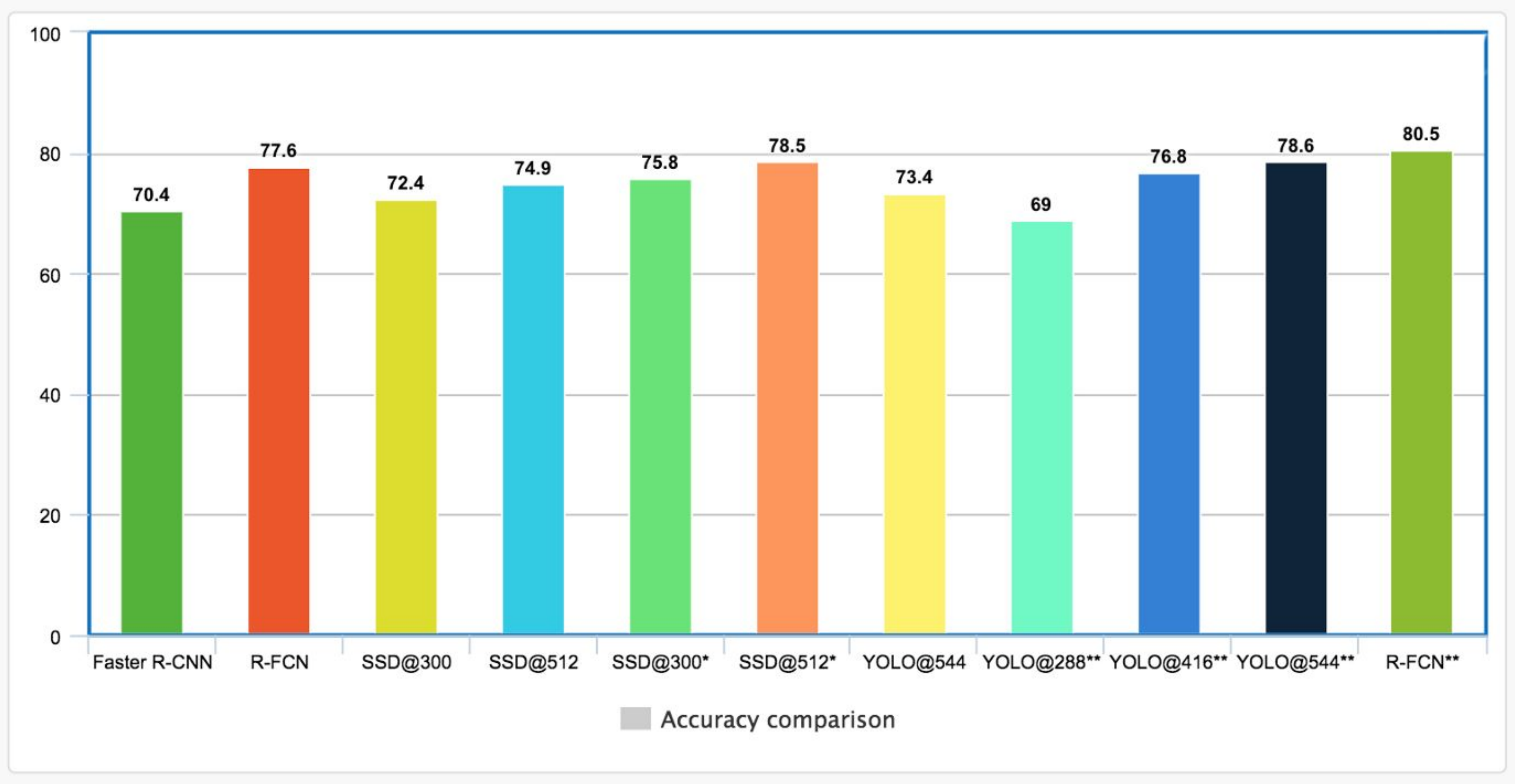
Comparison

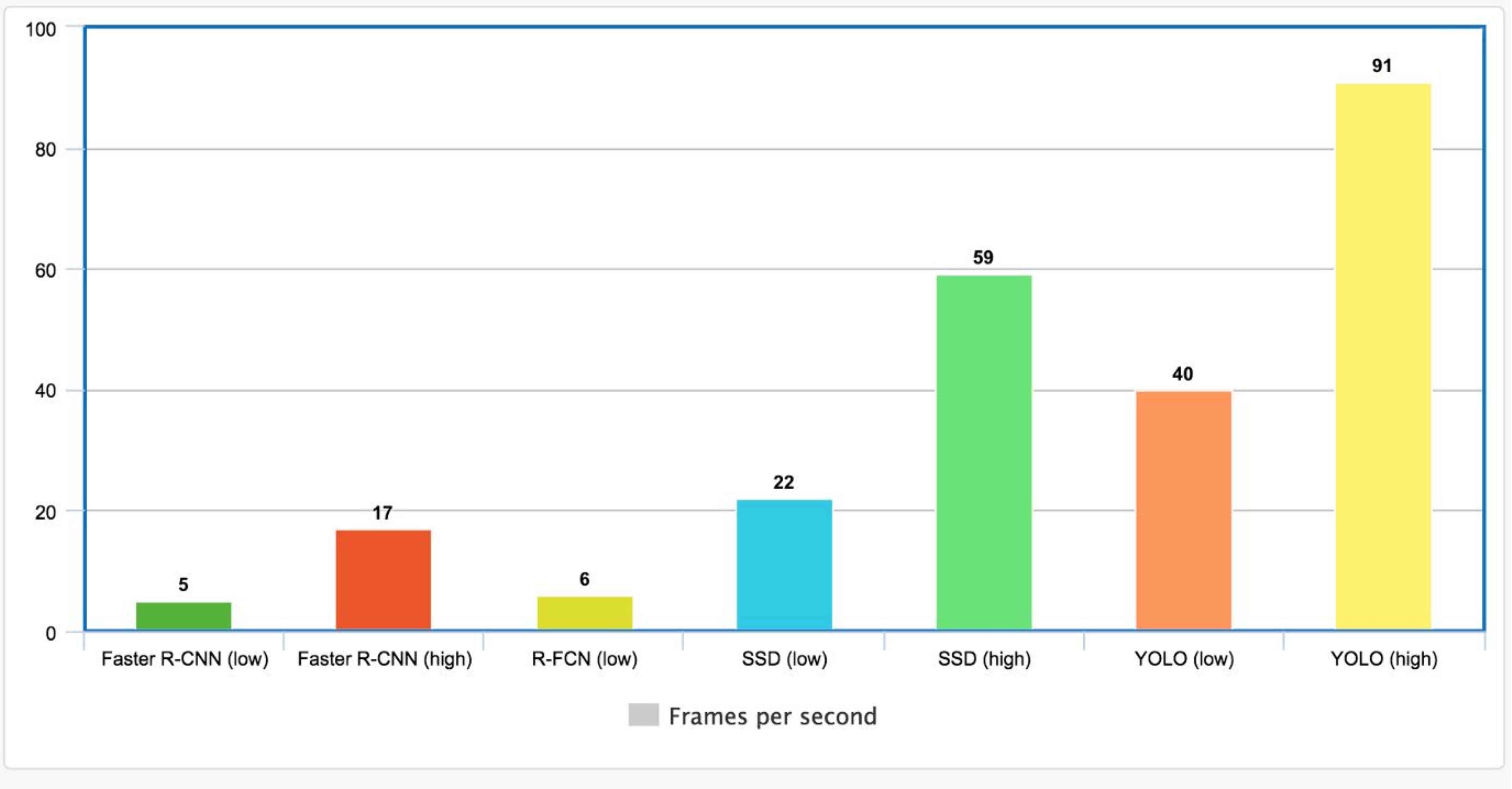
- mAPs and speed to examine the **accuracy-performance tradeoffs**
- **mAP** : mean average Precision
- mAP tracks average precision i.e., **accuracy of predictions**

Training set, mAP and FPS comparison

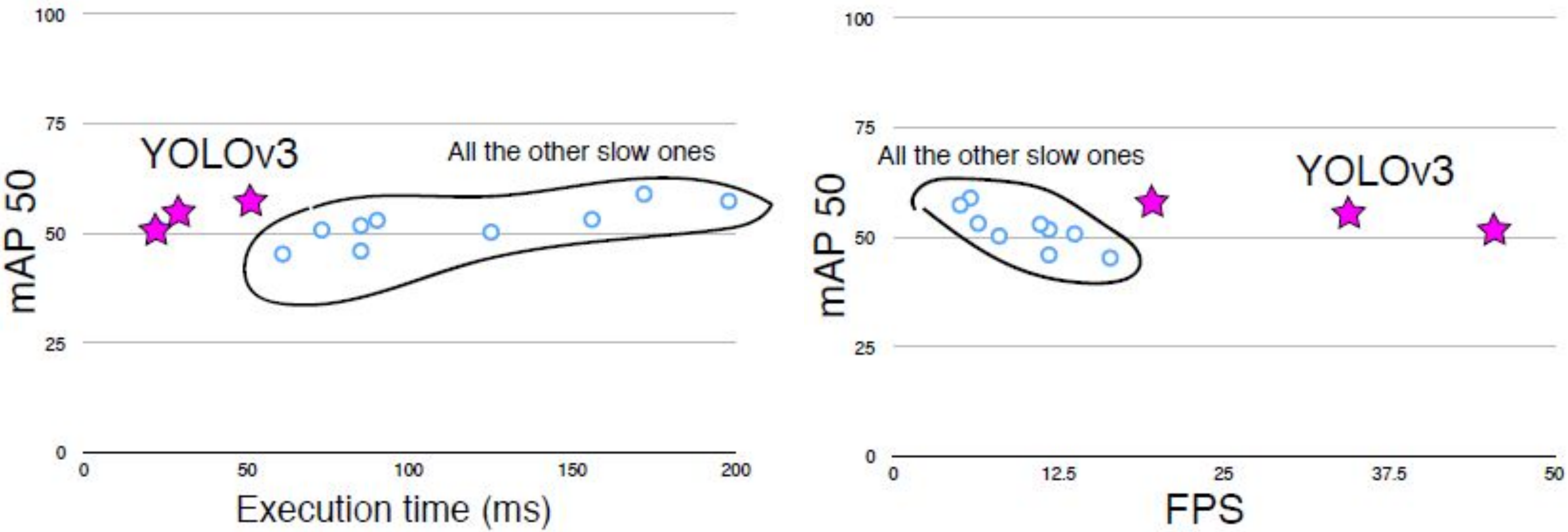
Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

- fastest DPMs **miss** real time speed even at **cost** of mAP
- For real-time mAP and FPS, **YOLO** remains **undefeated**
- Considering **highest mAP**, Fast R-CNN beats it considerably at **cost of speed**
- **YOLO** trained with **VGG-16** gets **best mAP** among YOLO models but **lower FPS**
- **Fast RCNN** use of **Selective Search** slows down detection even with faster classification


100
80
60
40
20
0
70.4
77.6
72.4
74.9
75.8
78.5
73.4
69
76.8
78.6
80.5
Faster R-CNN
R-FCN
SSD@300
SSD@512
SSD@300*
SSD@512*
YOLO@544
YOLO@288**
YOLO@416**
YOLO@544**
R-FCN**
Accuracy comparison


100
80
60
40
20
0
91
59
40
22
17
6
5
Faster R-CNN (low)
Faster R-CNN (high)
R-FCN (low)
SSD (low)
SSD (high)
YOLO (low)
YOLO (high)
Frames per second

Training set, mAP and FPS comparison



PASCAL VOC Error Analysis

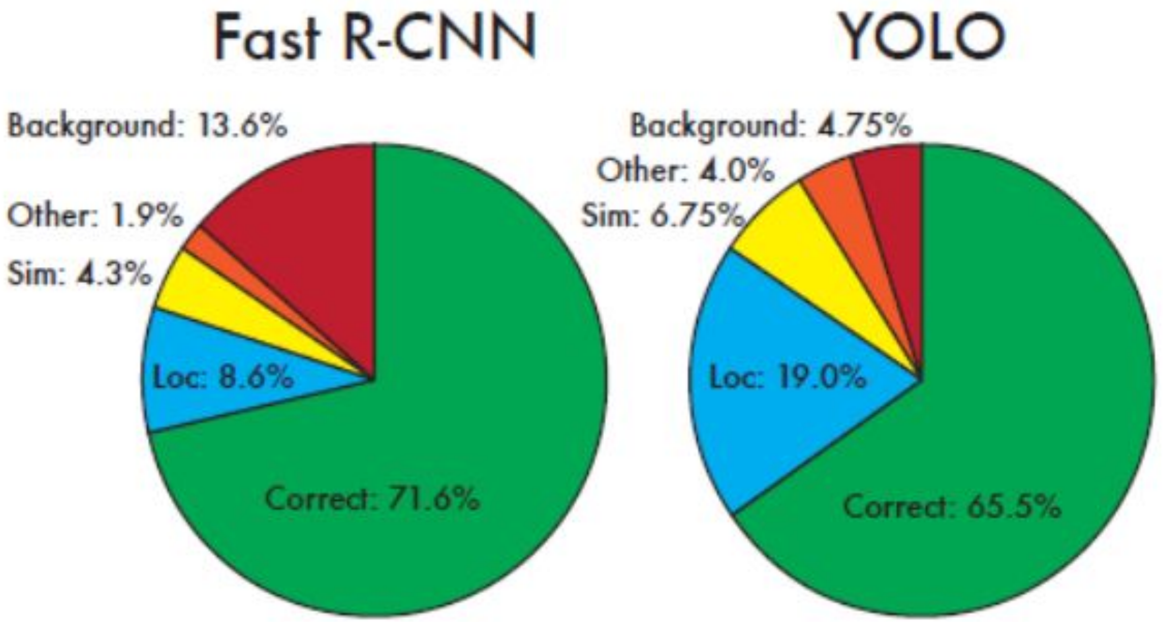
Correct: correct class and IOU > 0.5

Localization: correct class, 0.1 < IOU < 0.5

Similar: class is similar, IOU > 0.1

Other: class is wrong, IOU > :1

Background: IOU < :1 for any object



- **Top N** predictions for a category
- YOLO struggles at **localization**
- Fast R-CNN's struggles in background object prediction

- Fast R-CNN is almost 3x more likely to predict background detections than YOLO.

WHAT IF WE COMBINE THE SPEED OF YOLO AND LOCALIZATION STRENGTH OF FAST-RCNN??

Combining Fast-RCNN and YOLO

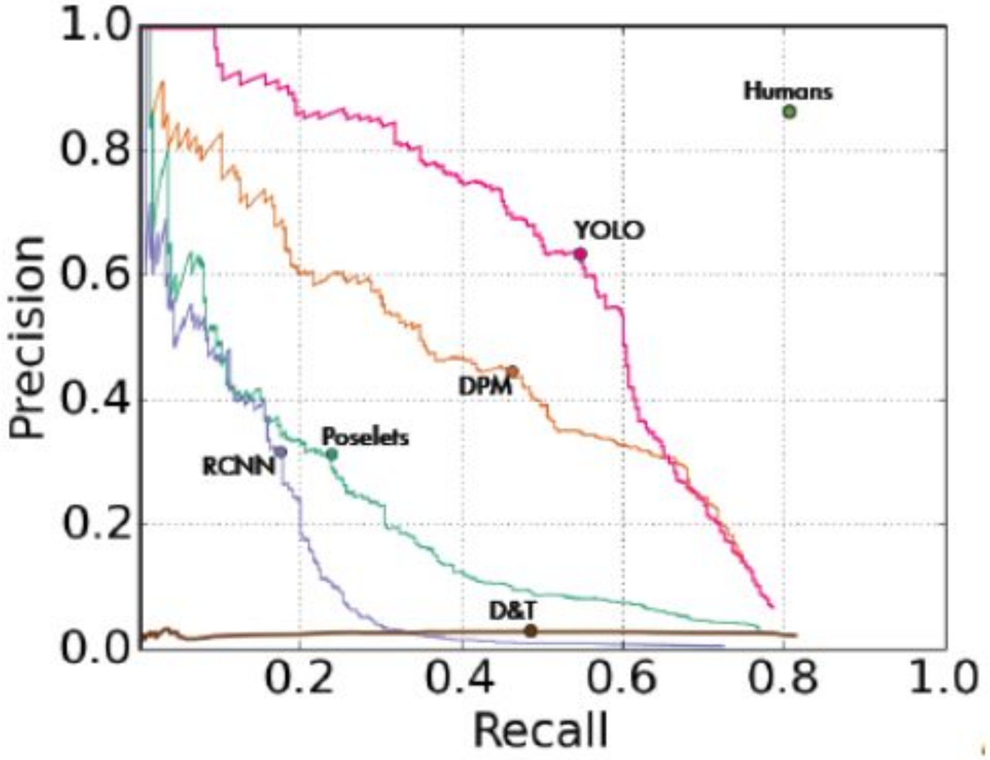
	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

- As noticed, combining with YOLO gives a **3.2% gain** in mAP
- Unfortunately, **no improvement in speed** as both models run separately
- Though YOLO doesn't add much inference time as it is very fast relatively

- **VOC 12 Results show YOLO's inaccuracy to predict small objects**

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [28]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [33]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

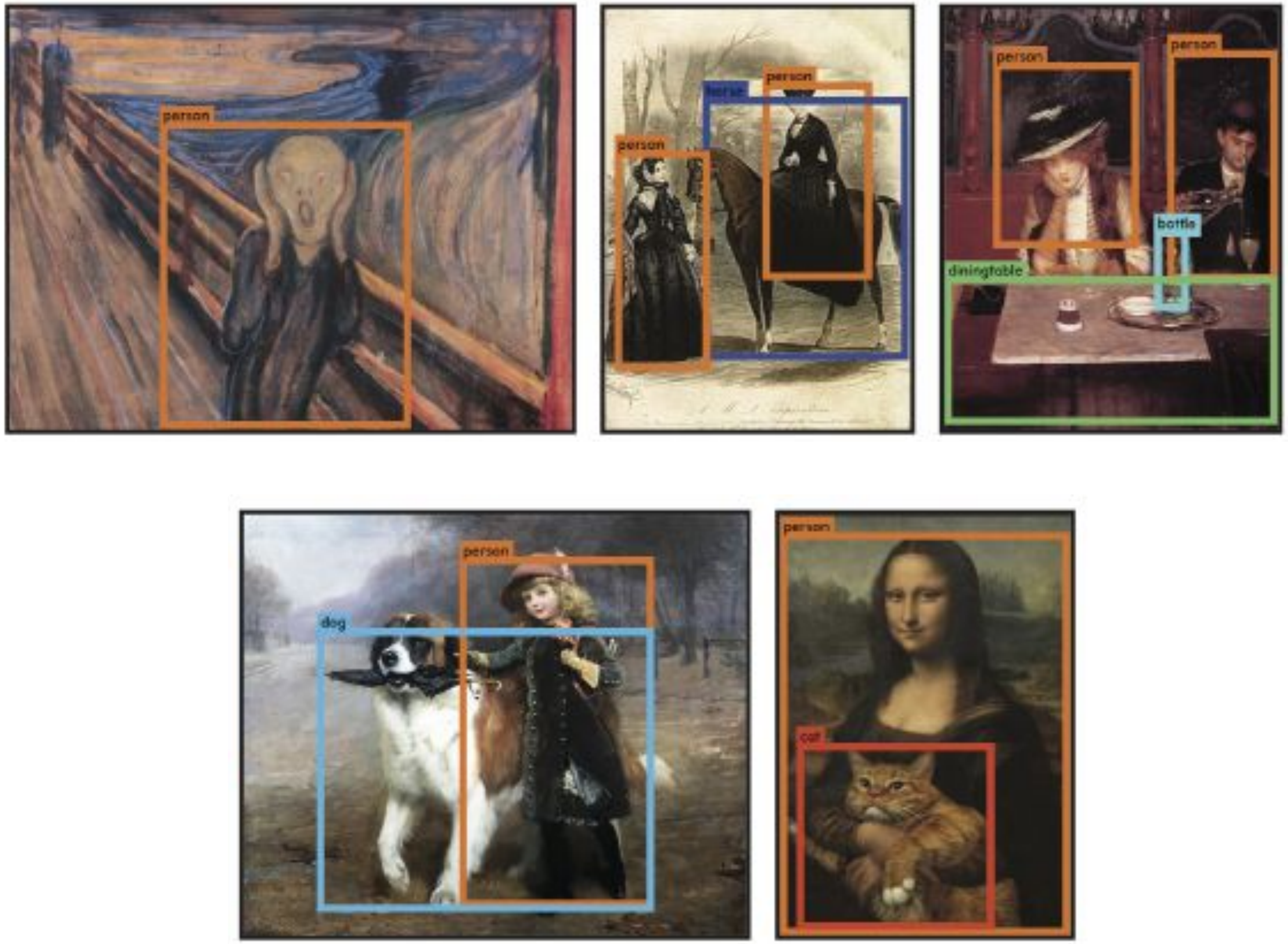
Generalizability



Models trained over real-life images…

- This Section explores the **artwork datasets** like **picasso painting dataset**
- **YOLO** performs **fairly**
- **RCNN suffers** due to its selective Search algo.
- **DPM maintains** a considerate AP

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	


person
person
horse
person
person
person
bottle
diningtable
person
dog
person
cat

YOLO v3 Insights-
Things tried but didn’t work

- Anchor Box x, y offset prediction:
  x, y as multiple of box width height
  model unstable

- LInear x, y prediction instead of Logistic:
  drop in **mAP**

- Focal Loss:
  drop in **mAP**

- **Dual IOU**(**range** of IOU [lower, upper]):
  IOU lower than ‘lower’ **negative** for **all classes**
  used in **RCNN**, proved **efficient**
  but **YOLO didn’t** much **benefit**

Thank You!