人工智能之深度学习

目标检测

主讲人：Vincent Ying

课程要求

- 课上课下“九字”真言
  - 认真听，**善摘录，勤思考**
  - **多温故，乐实践**，再发散
- 四不原则
  - 不懒散惰性，不迟到早退
  - 不请假旷课，不拖延作业
- 一点注意事项
  - 违反“四不原则”，不推荐就业

课程内容

- YOLO v3

YOLO

- You only look once (YOLO) is a state-of-the-art, real-time object detection system。
- website： https://pjreddie.com/darknet/yolo/
- version：
  - YOLO v1: https://arxiv.org/pdf/1506.02640.pdf
  - YOLO v2: https://arxiv.org/pdf/1612.08242.pdf
  - YOLO v3: https://arxiv.org/pdf/1804.02767.pdf

YOLO v1

- 从R-CNN到Faster R-CNN网络的发展中，都是基于proposal+分类的方式来进行目标检测的，检测精度比较高，但是检测速度不行，YOLO提供了一种更加直接的思路：**直接在输出层回归bounding box的位置和bounding box所属类别的置信度**，相比于R-CNN体系的目标检测，YOLO将目标检测从分类问题转换为回归问题。其主要特点是：
  - 速度快，能够达到实时的要求，在Titan X的GPU上达到45fps；
  - 使用全图Context信息，背景错误(把背景当做物体)比较少；
  - 泛化能力强；

YOLO v1

- Yolo处理图像比较简单明了，可以划分为如下三个步骤：
  - 1. 将输入图像resize到448x448；
  - 2. 在图像上运行单个卷积网络；
  - 3. 根据模型的置信度对检测结果进行NMS非极大值抑制。

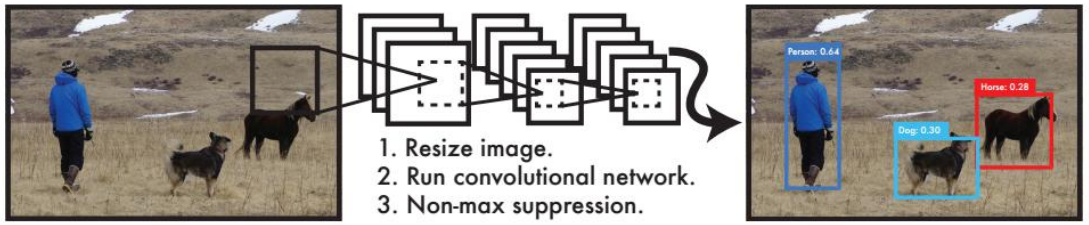
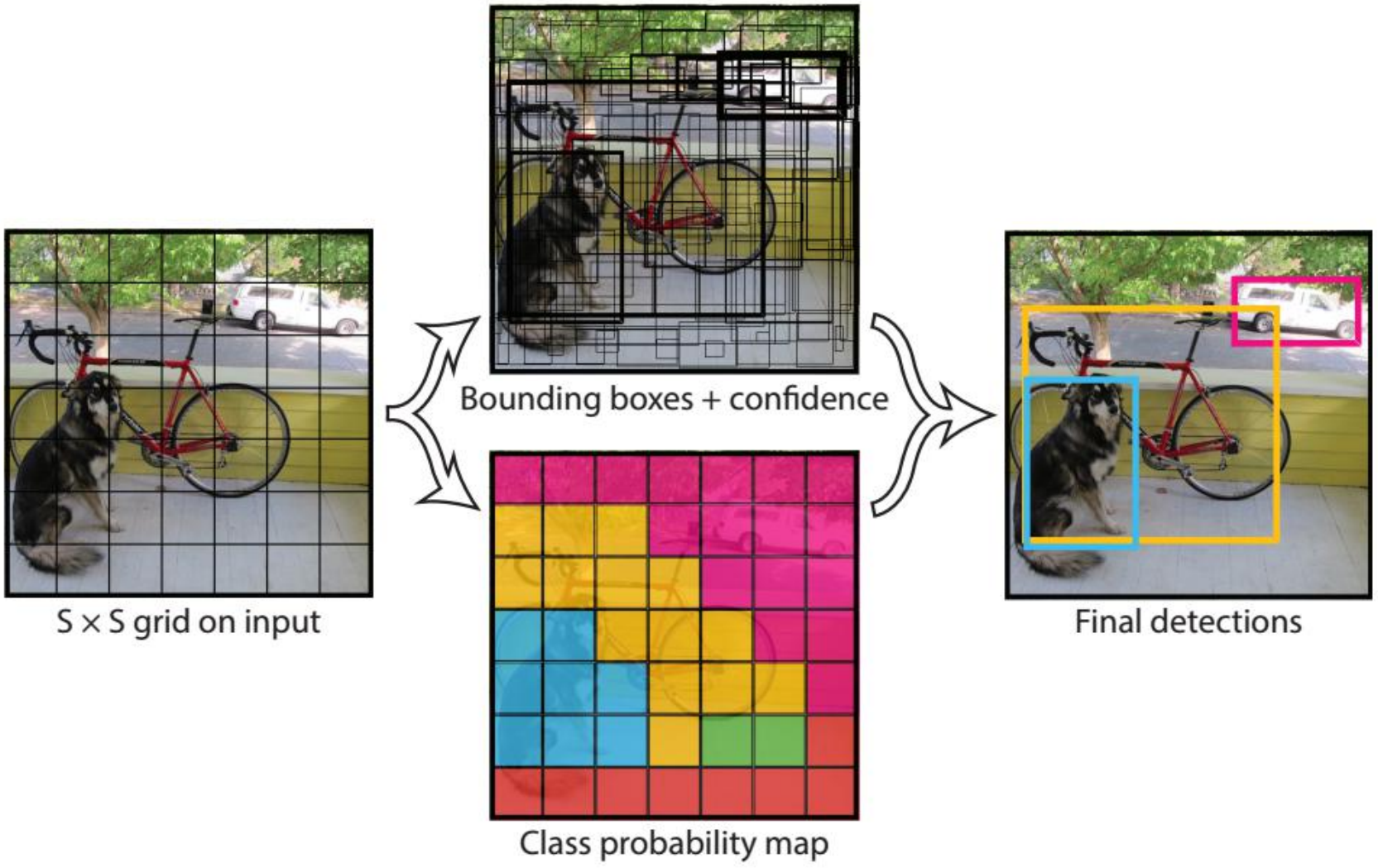


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

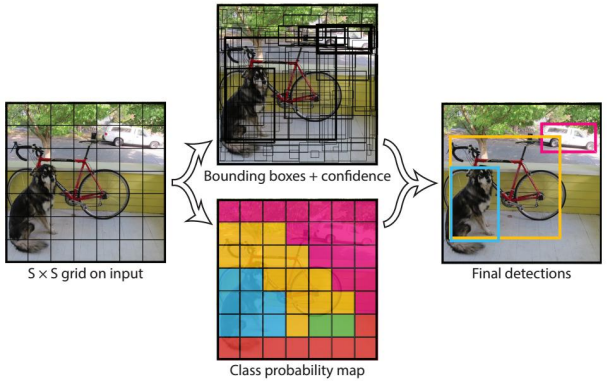
YOLO v1


Bounding boxes + confidence
S × S grid on input
Class probability map
Final detections

YOLO v1

- 在YOLO v1中其核心思想为**Unified Detection**(统一检测)，将目标检测的分离组件统一为单个神经网络，使用整个图像的特征来预测每个bounding boxes边框，YOLO的设计使得训练和实时速度成为可能性，同时保证相对高的平均精度。

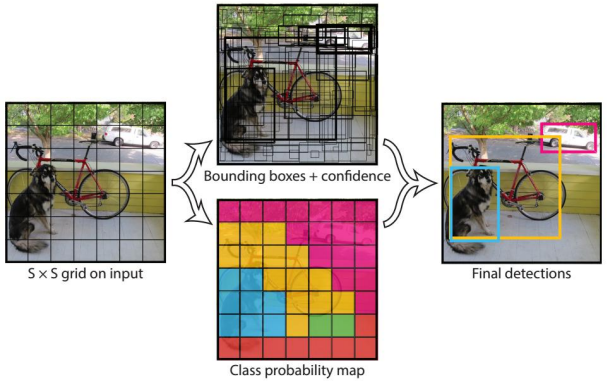
YOLO v1



- **Unified Detection(统一检测)核心思想:**
  - 1. 将输入图像划分为S*S个grid cell(网格单元)；如果一个object物体的中心是位于这个grid cell内，那么这个grid cell就负责检测这个物体；
  - 2. 每一个grid cell需要预测B个bounding boxes坐标信息(x,y,w,h)以及这些bounding boxes的置信度confidence。
    - YOLO v1中的坐标信息(x,y,w,h)表示的是boxes中心点坐标相对于当前图像宽度、高度的比值，w、h是boxes的宽度、高度和实际图像的宽度、高度的比值；
    - 置信度confidence反映的是模型对于这个预测的bounding box中包含物体的可能性大小。

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

YOLO v1



- **Unified Detection(统一检测)核心思想：**
  - 3. 每个grid cell还需要预测C个类别的概率(conditional class probability)，以判断当前cell属于哪个类别C。
    - NOTE：Confidence置信度表示的是bounding boxes包含物体的概率值，而conditional class probability表示的是每个grid cell属于某个类别的概率值。
  - 4. 每个grid cell的class信息和每个bounding box预测的confidence信息相乘，就可以得到每个bounding box的**class-specific confidence score**。

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

YOLO v1

- 论文建议：
  - S=7, B=2
- 在PASCAL VOC数据集中，最终预测值形状为7 x 7 x 30的Tensor对象
  - C=20

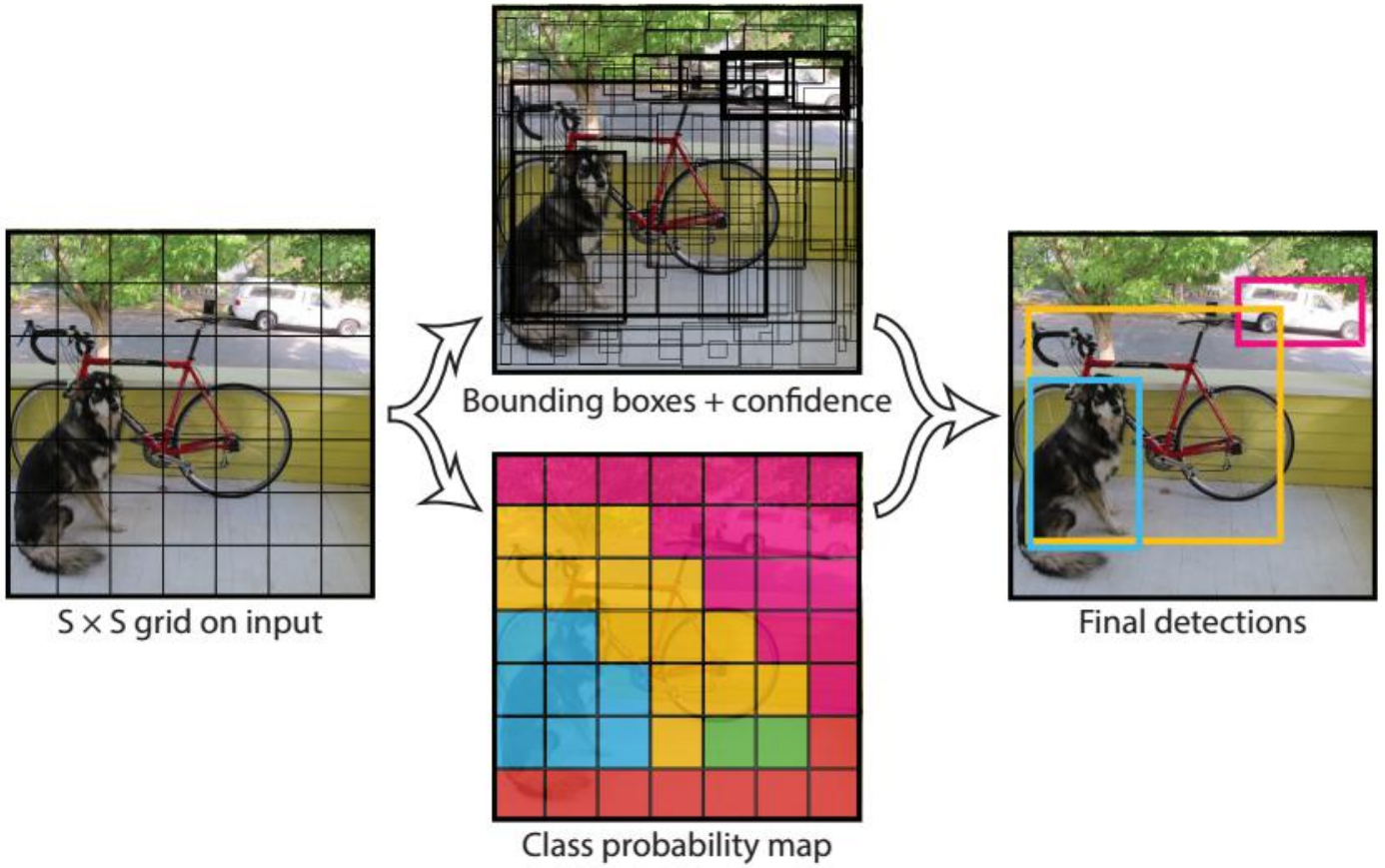


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

YOLO v1

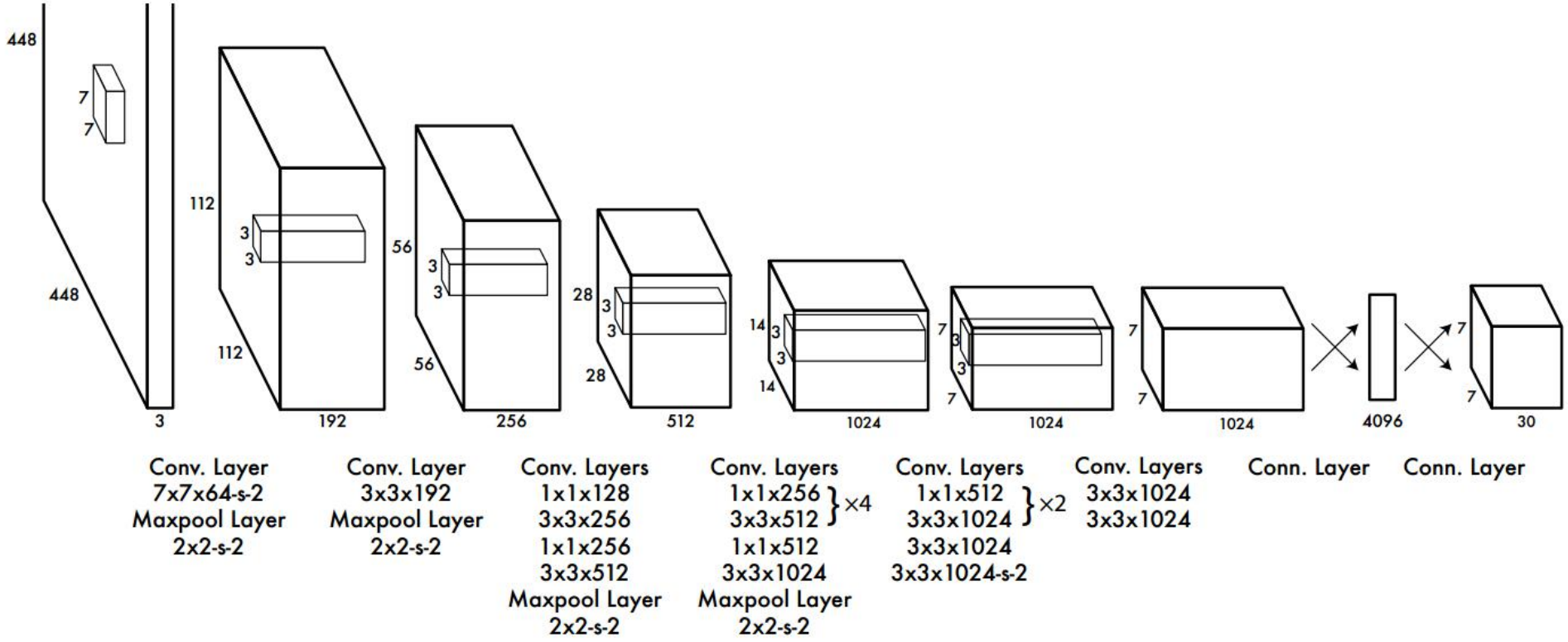
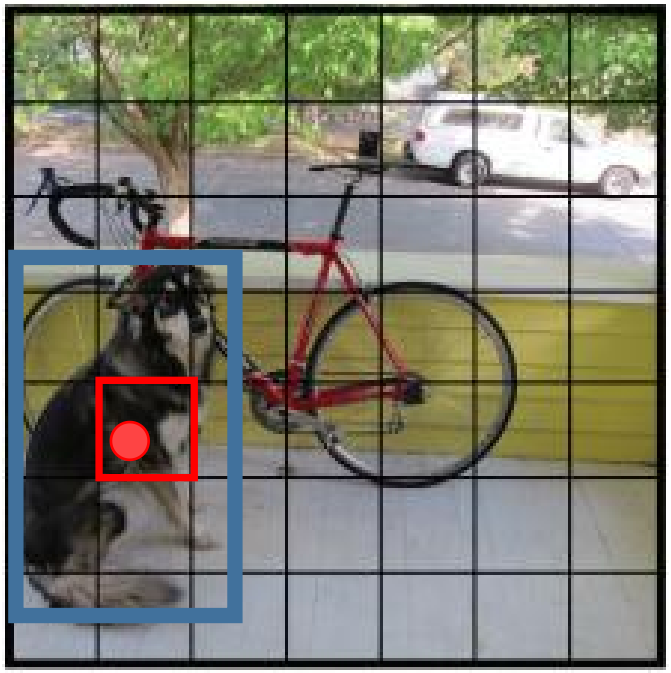


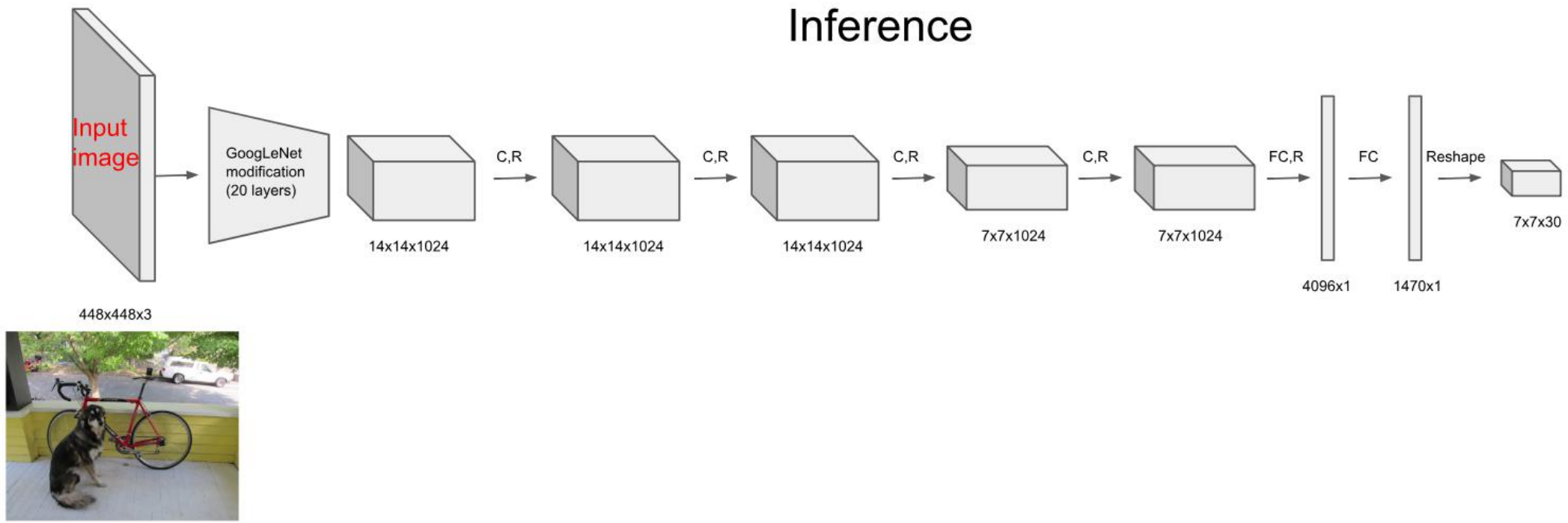
Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

YOLO v1

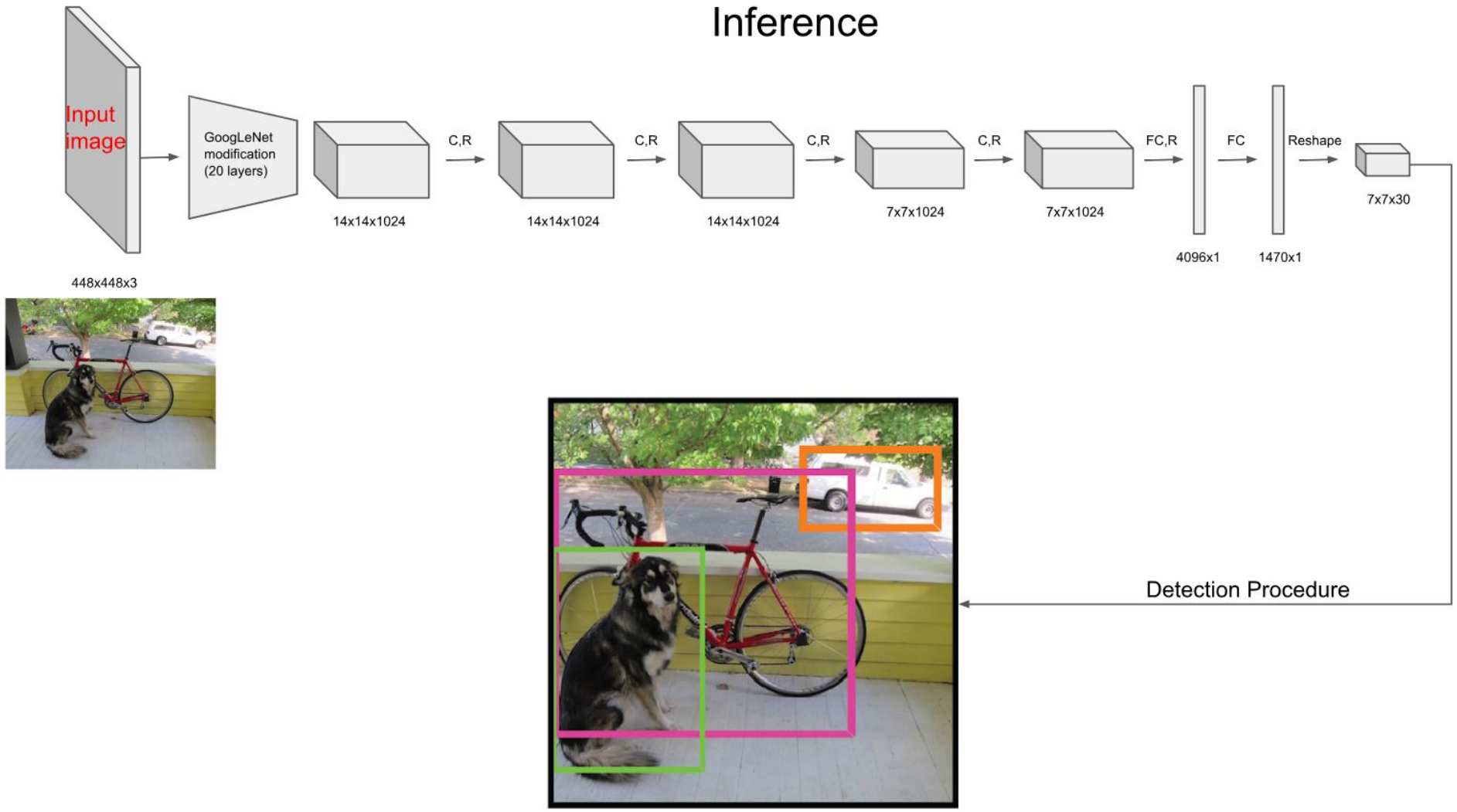
- 图片分成7x7个网格(grid cell)，某个物体的中心落在这个网格中此网格就负责预测这个物体。图中物体狗的中心点（红色框）落入第5行、第2列的格子内，所以这个格子负责预测图像中的物体狗。



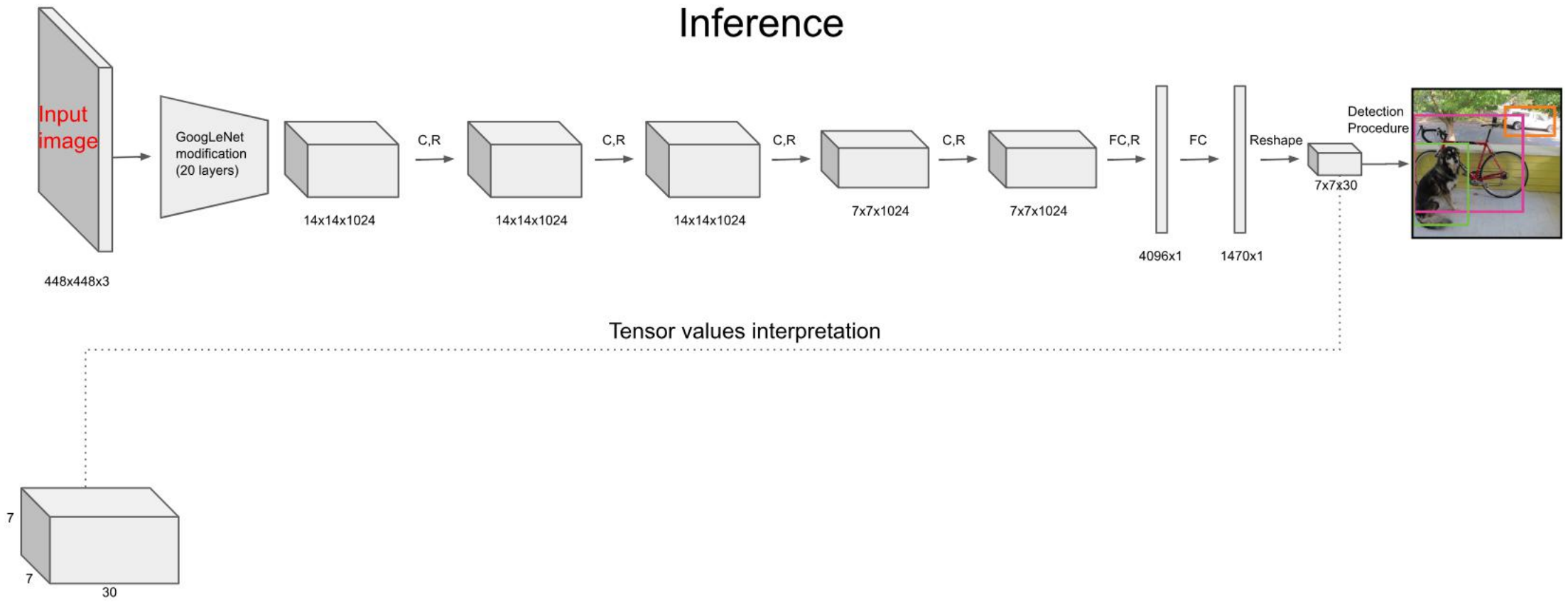
YOLO v1


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

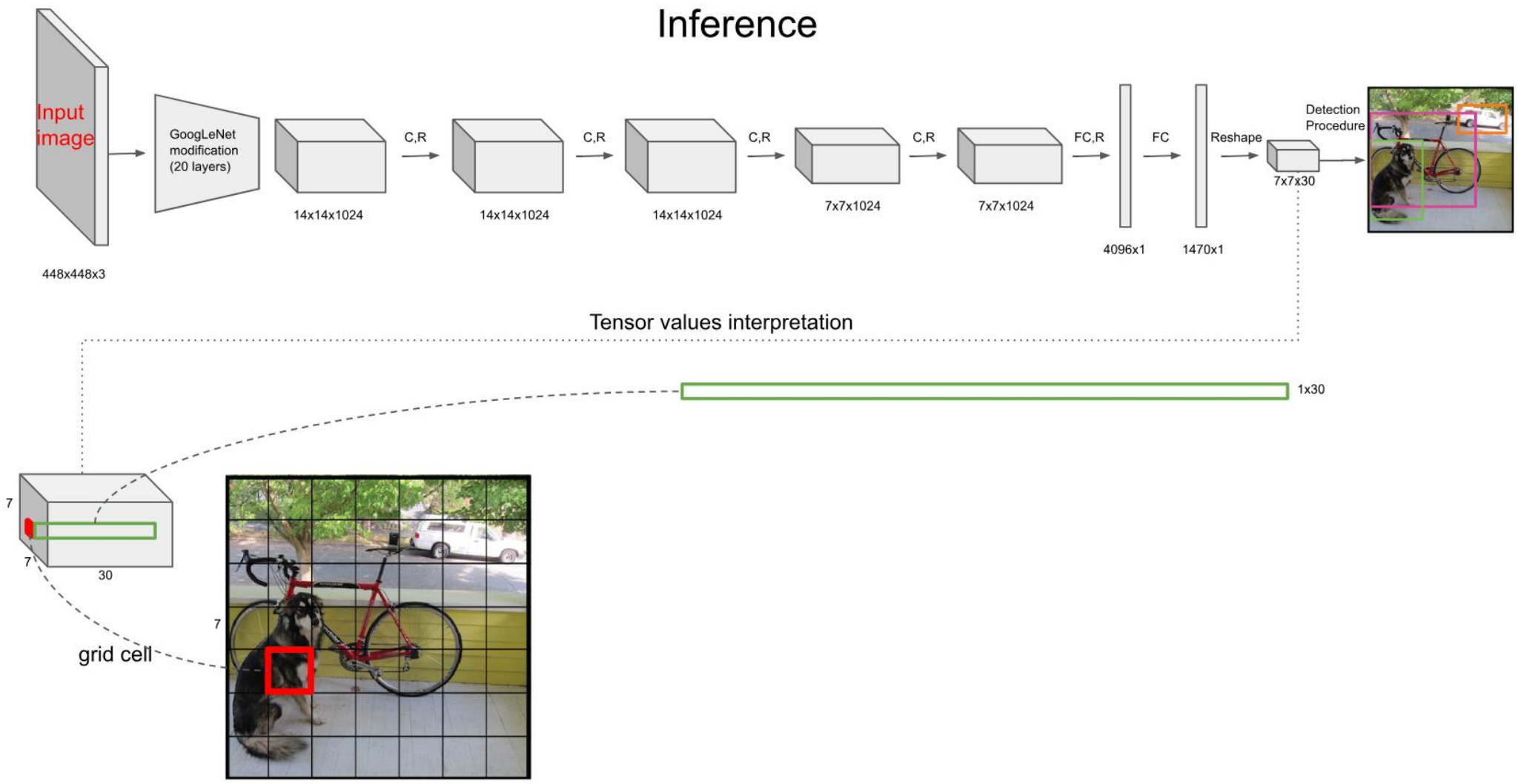
YOLO V1


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

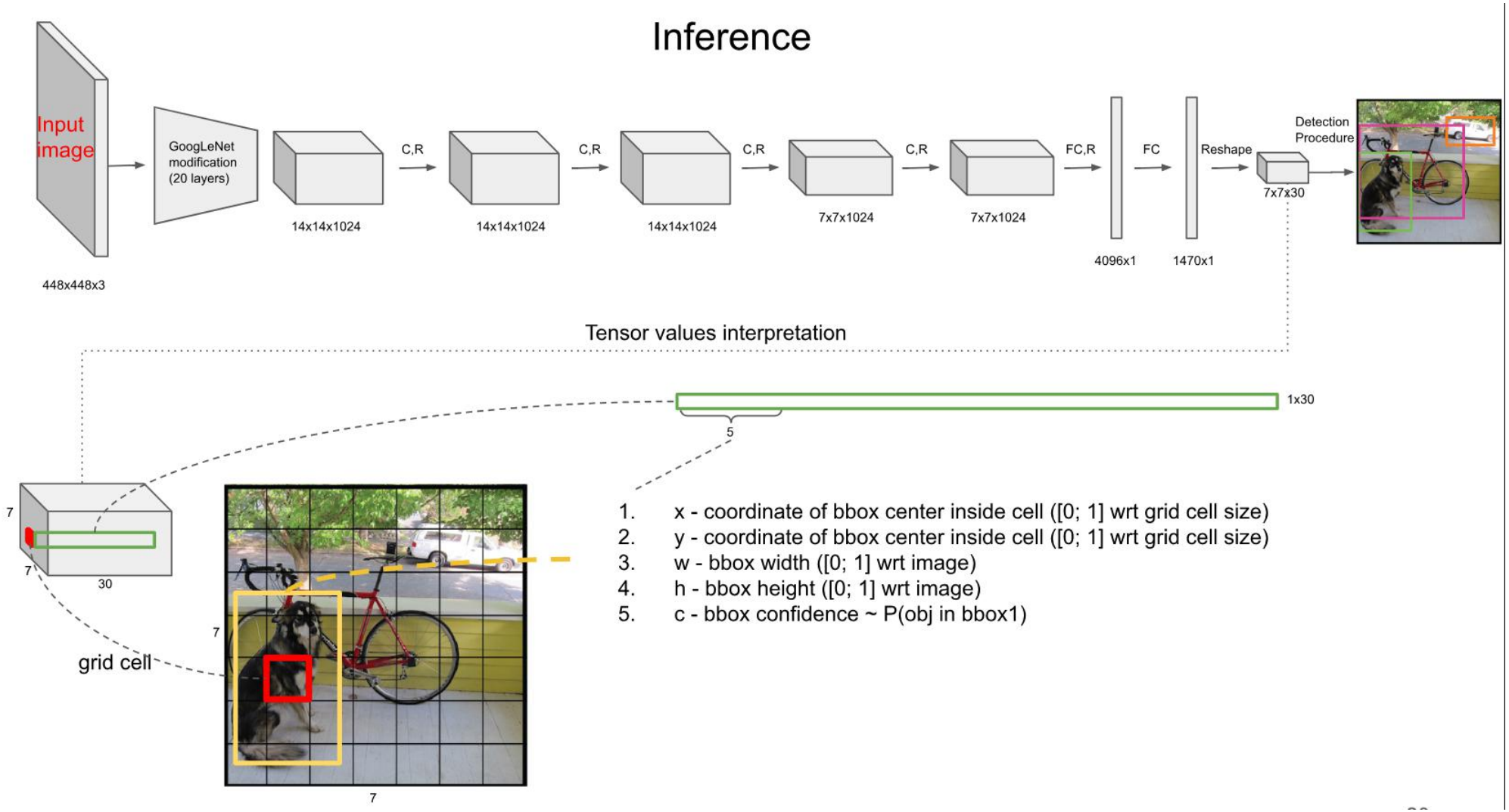
YOLO v1


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

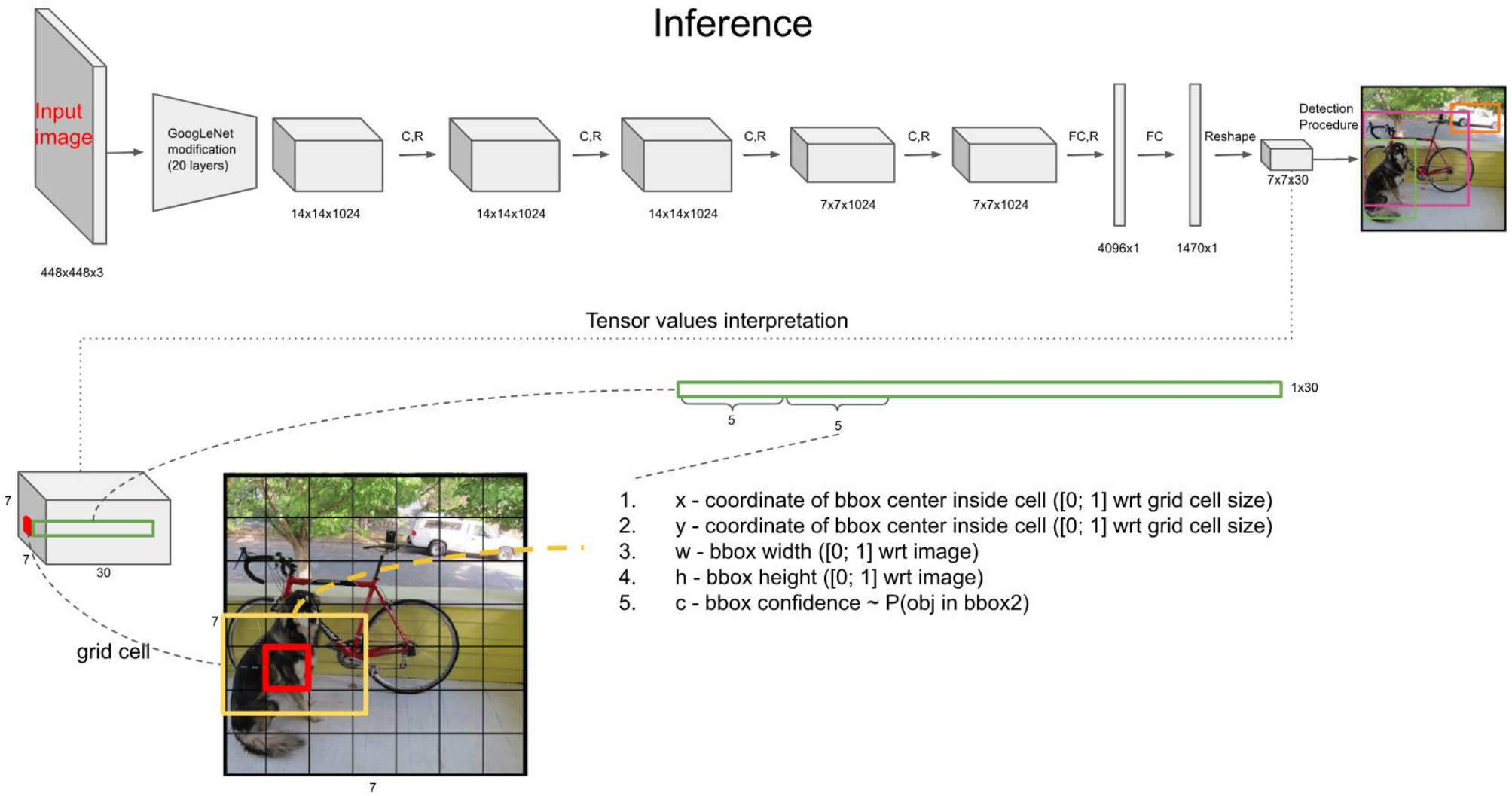
YOLO v1


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

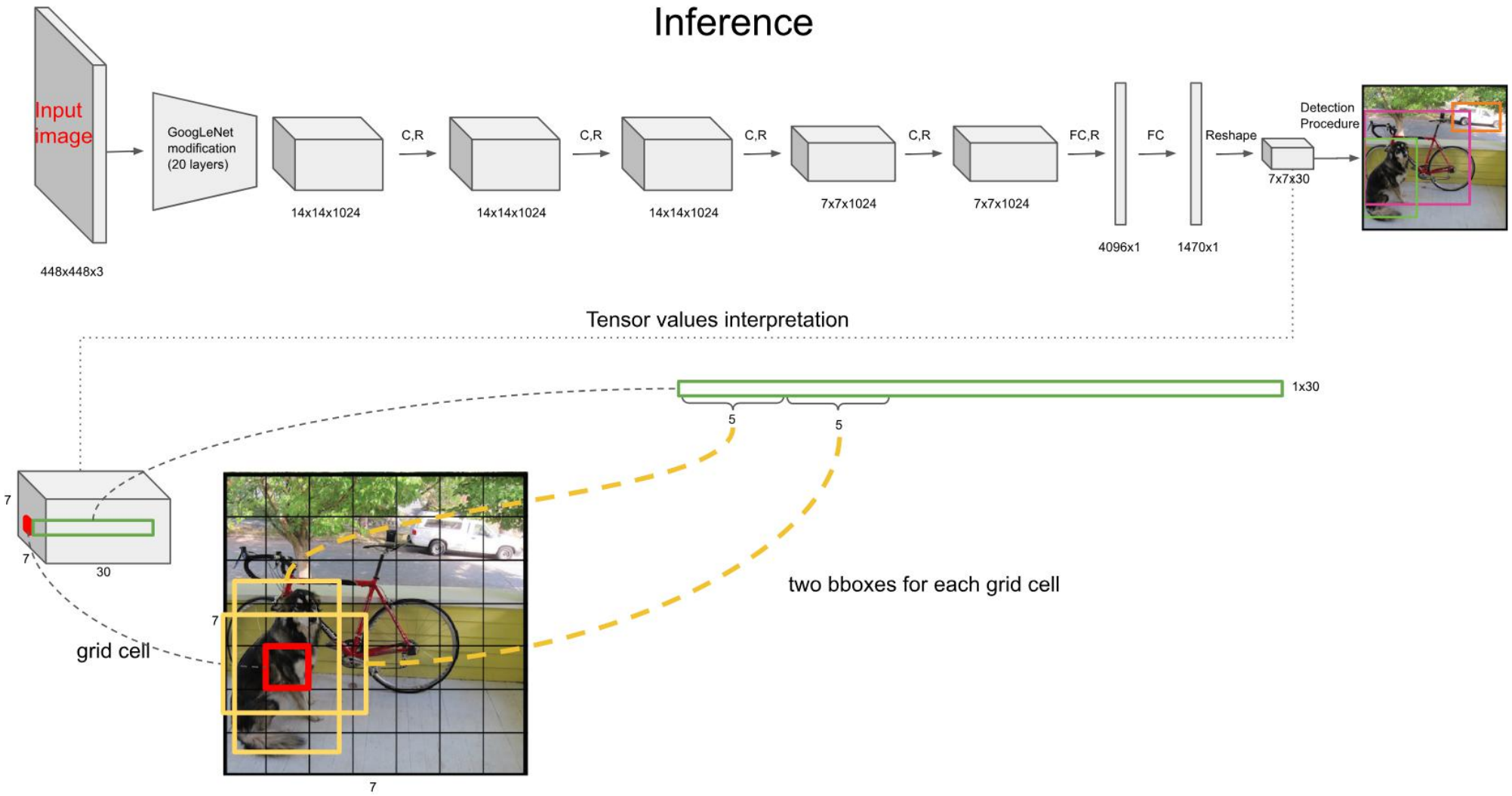
YOLO v1


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)

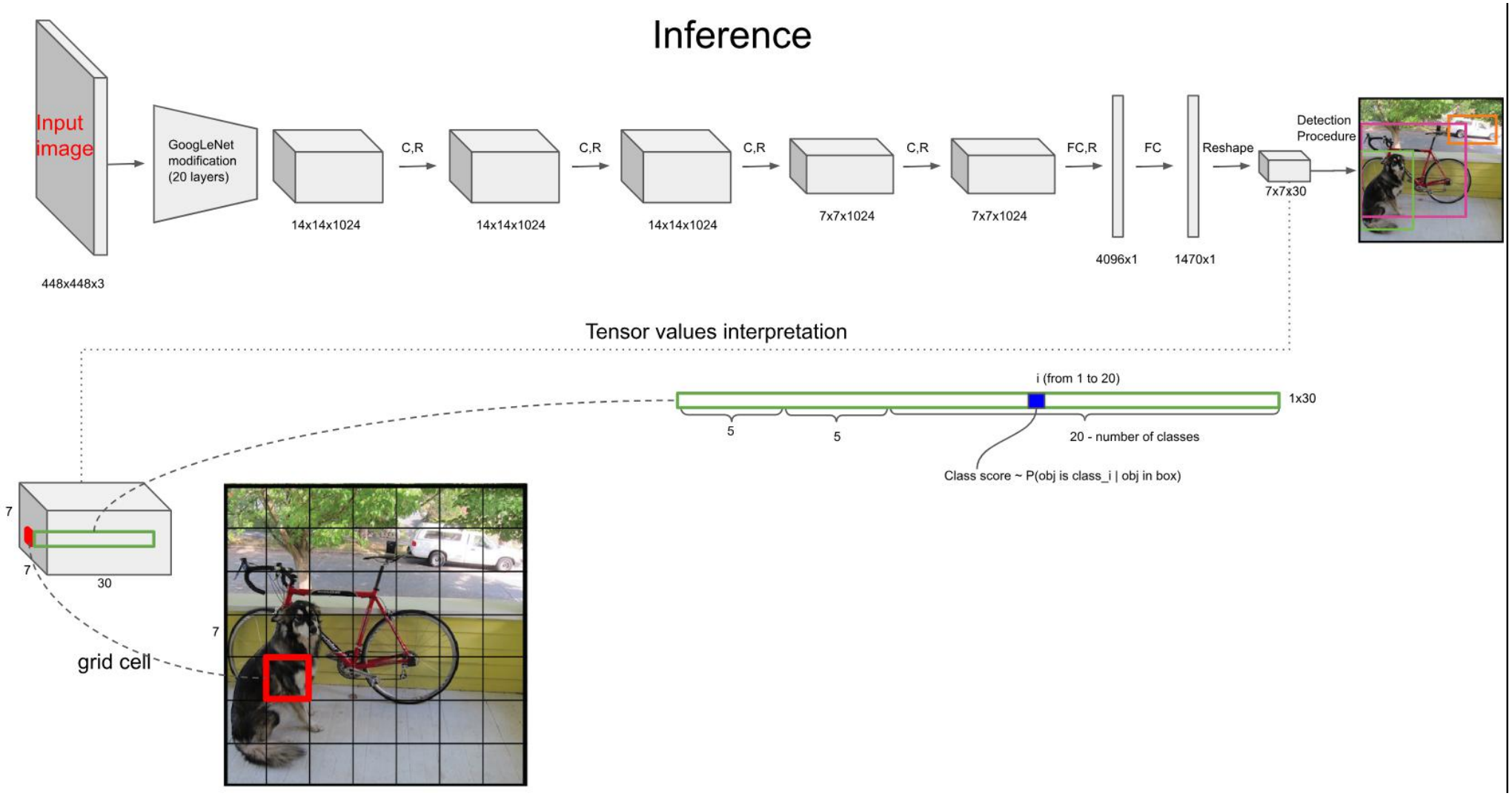
YOLO v1


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)

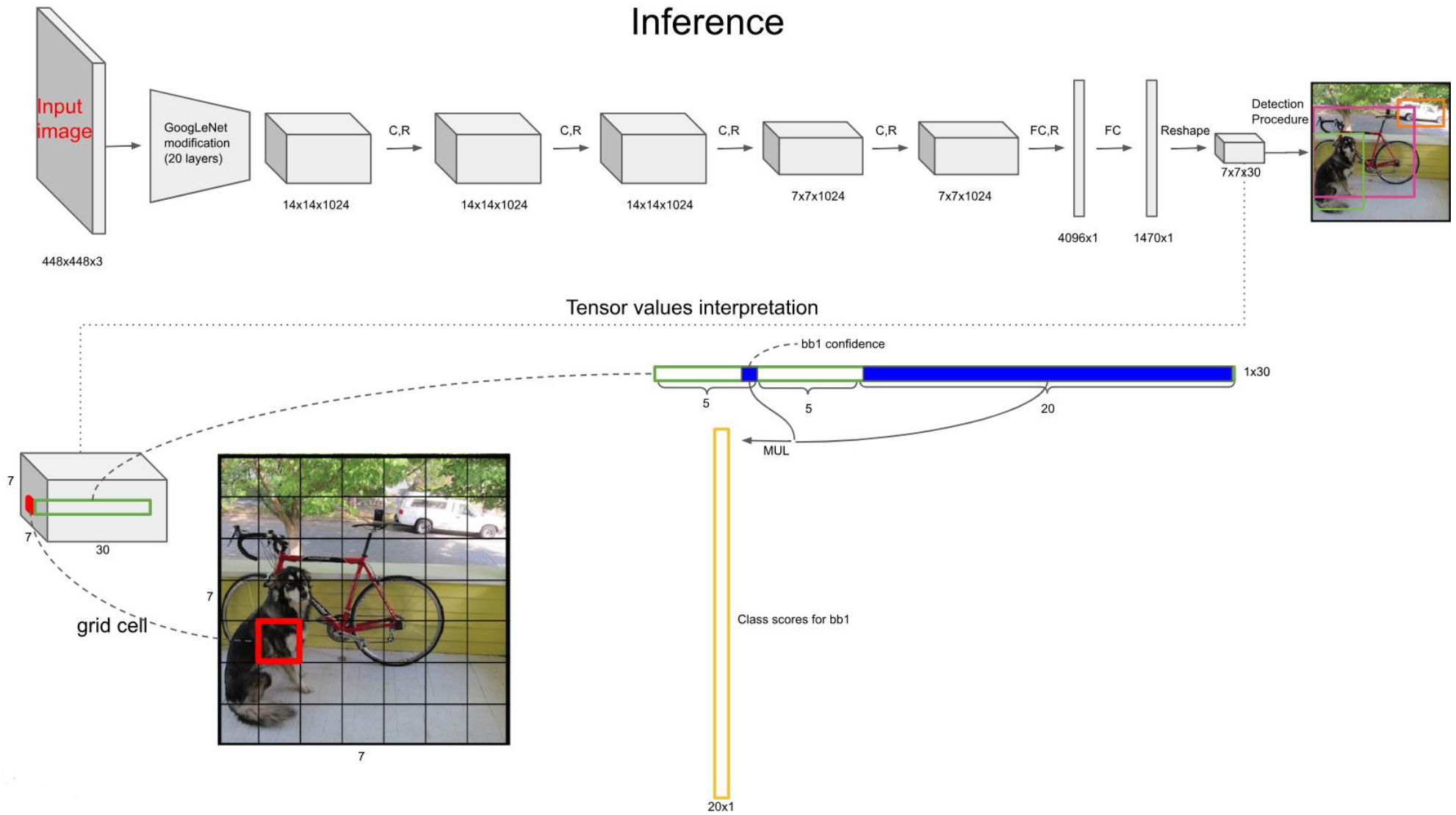
YOLO v1


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

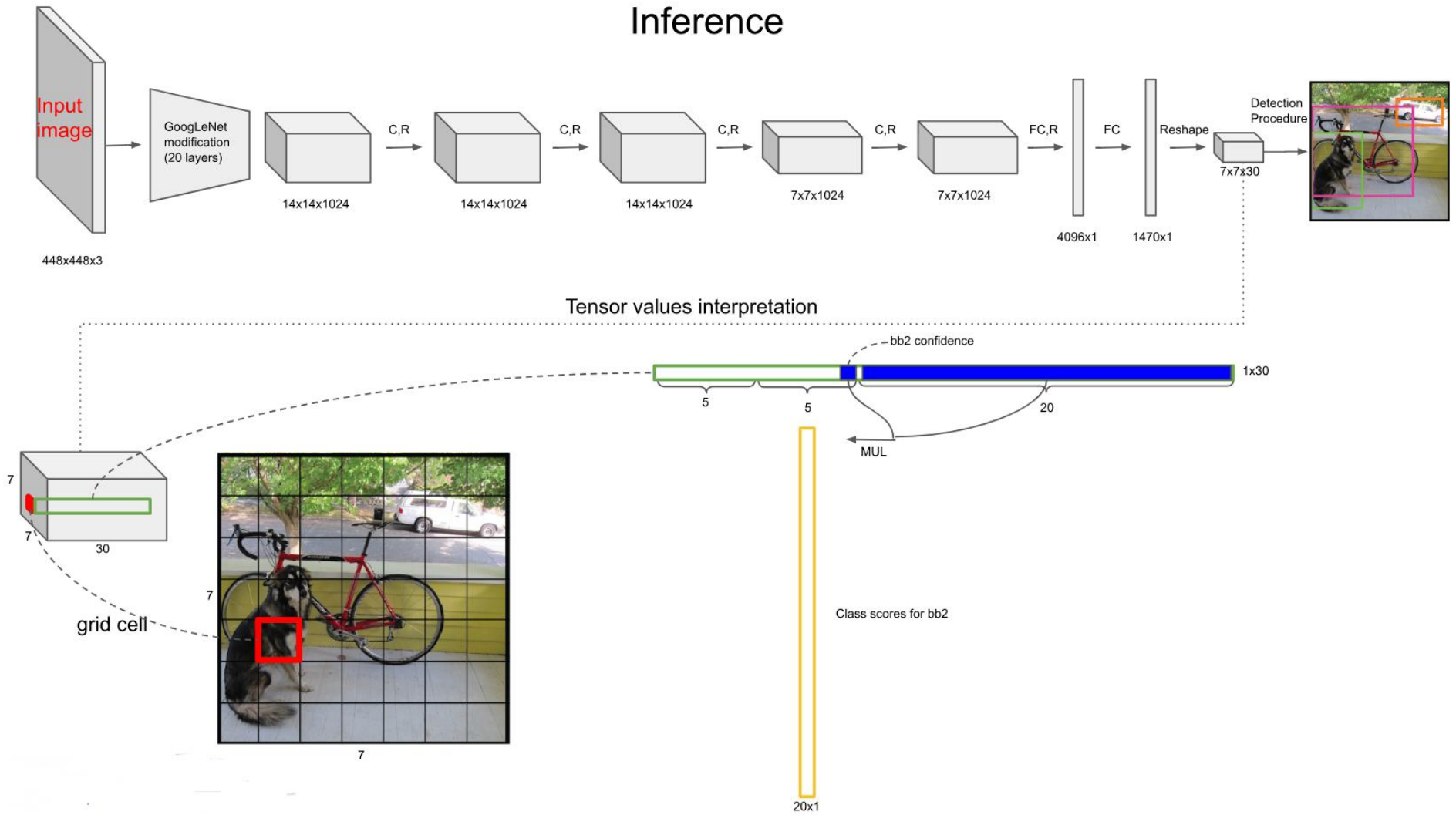
YOLO v1


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
7
grid cell

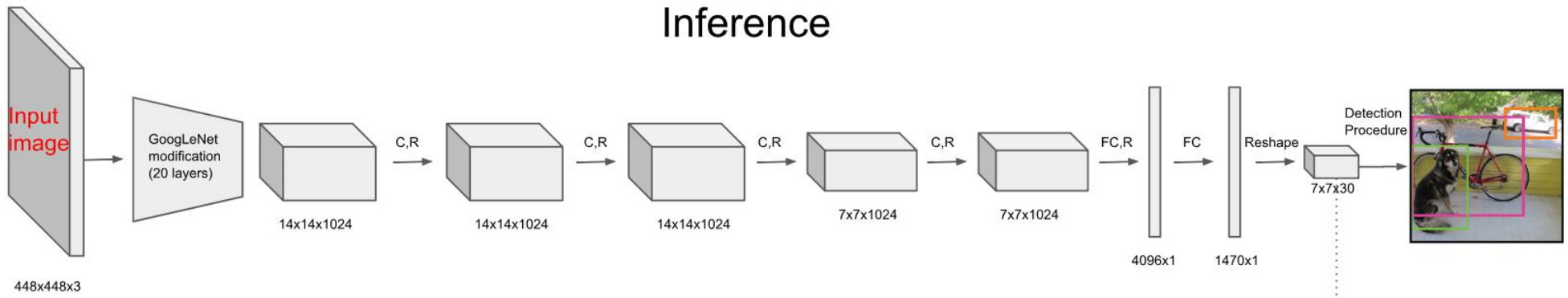
YOLO v1


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

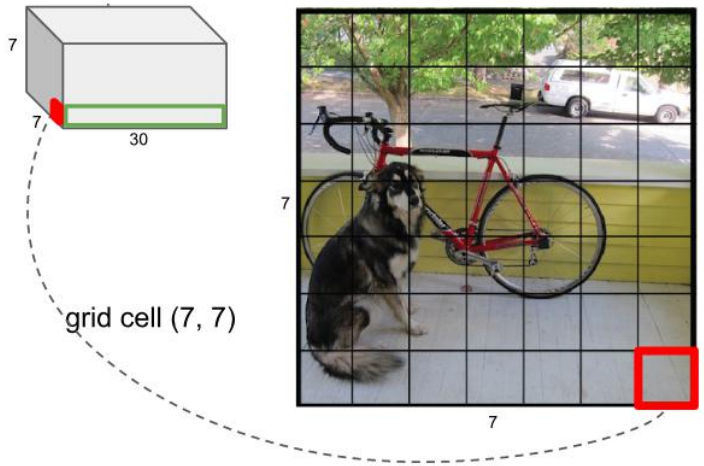
YOLO v1


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

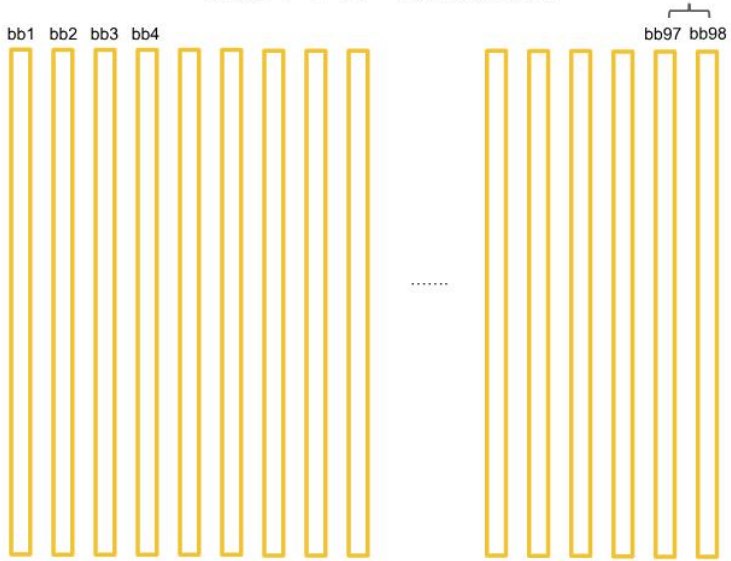
YOLO v1



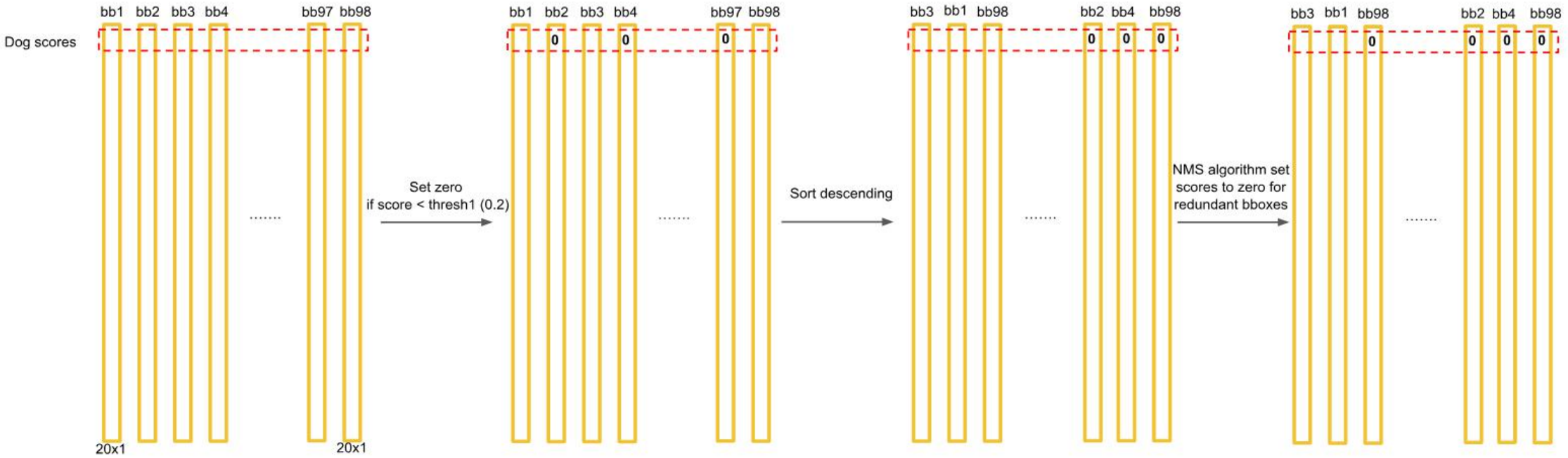
Tensor values interpretation



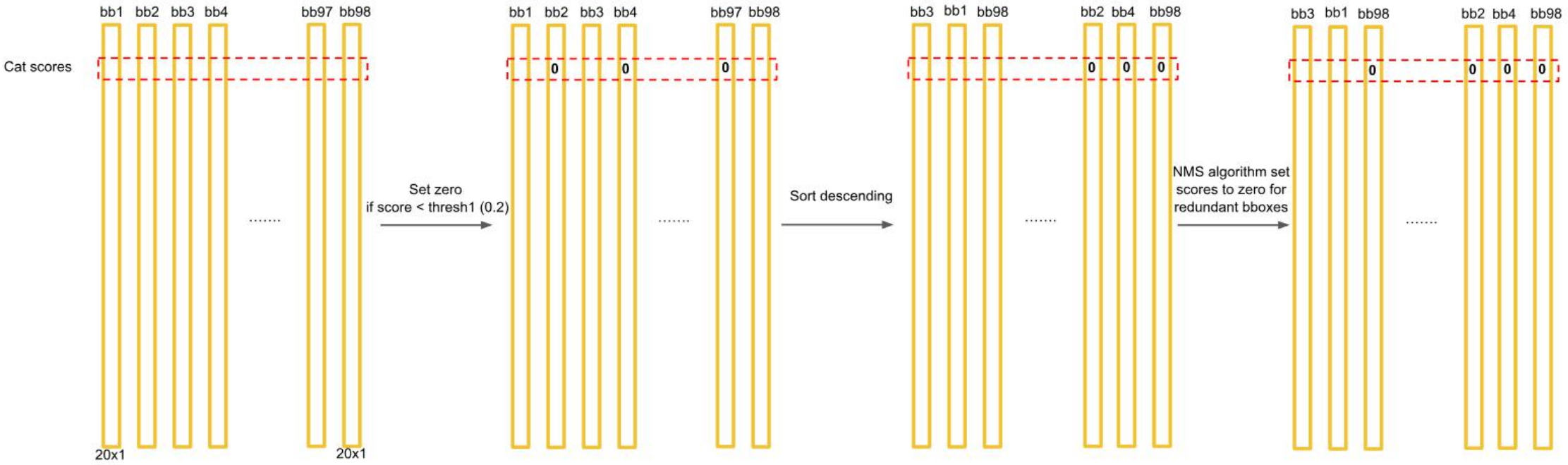
Total 7*7*2 = 98 bboxes



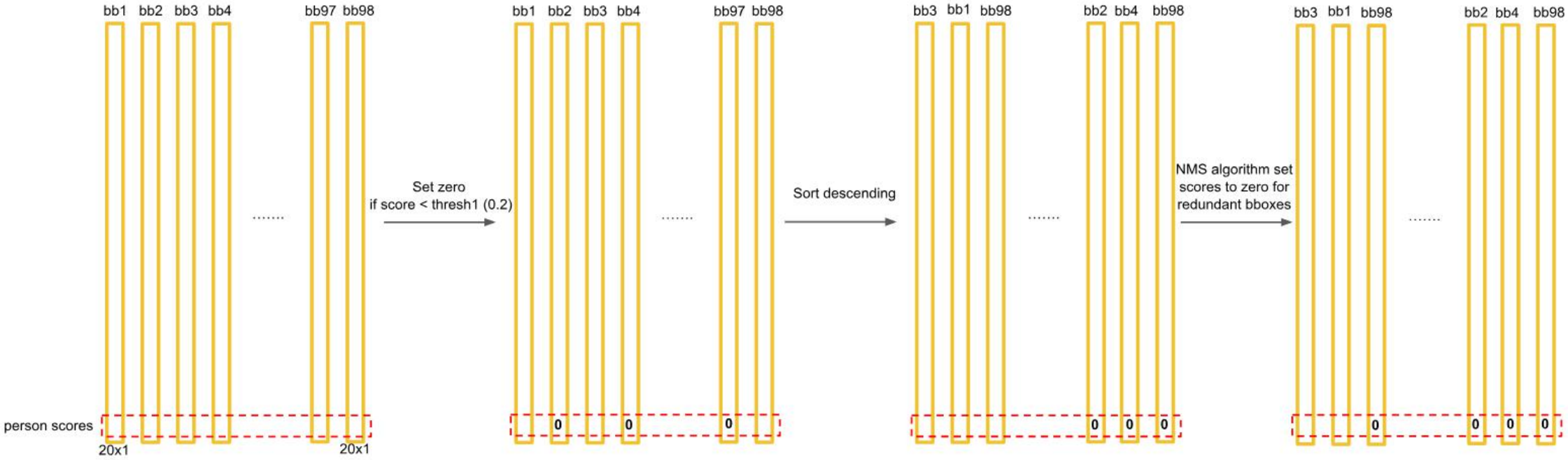
YOLO v1


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

YOLO v1


Cat 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

YOLO v1


bb1 bb2 bb3 bb4 ..... bb97 bb98
person scores
20x1
20x1
Set zero
if score < thresh1 (0.2)
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

YOLO v1


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

YOLO v1

- 损失函数

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

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

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

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

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

YOLO v1

- 损失函数

第i个cell中的第j个bounding boxes负责预测obj类别

位置信息的预测

采用平方根之差兼顾大框和小框的预测，让大框接受较大的偏移，让小框不接受较大的偏移

5.0: 加强坐标信息预测失败所带来的损失

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

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

第i个cell中的第j个bounding boxes不负责预测obj类别

包含obj类别的box的confidence预测

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

0.5：减弱不包含物体的Bounding Box的损失

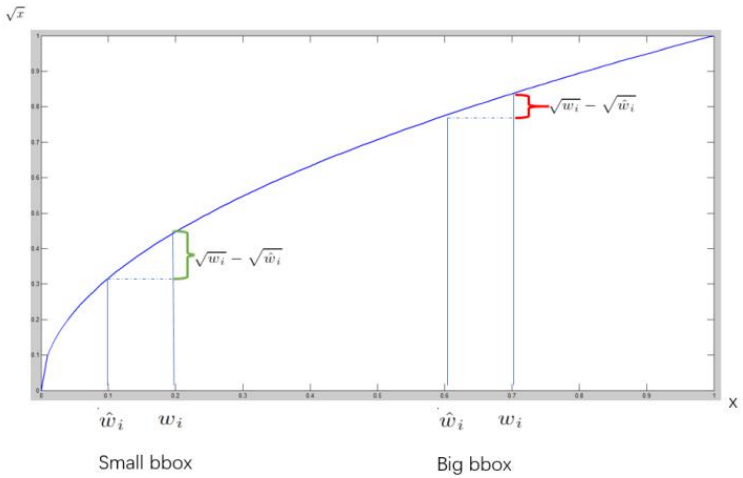
不包含obj类别的box的confidence预测

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

类别判断

判断是否有object的中心落在这个grid cell中

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



YOLO v1

- 特征数据: 原始图像，大小必须为448*448；
- 训练用的目标属性(VOC数据集，20个类别)：
  - 7*7=49个cell预测各个类别的概率矩阵，是一个49*20矩阵，如果真实边框的中心点落在当前cell范围，那么对应索引的value值为1，其它值为0.
  - 位置信息，直接使用实际边框的中心点坐标(x,y)、宽度、高度除以原始图像的宽度和高度，得到的就是实际值(x,y,w,h);
  - 置信度的值，如果真实边框的中心点落在当前cell中，那么当前cell中的所有Bounding Box和实际边框的IoU值当做置信度。

YOLO v1

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

YOLO v1

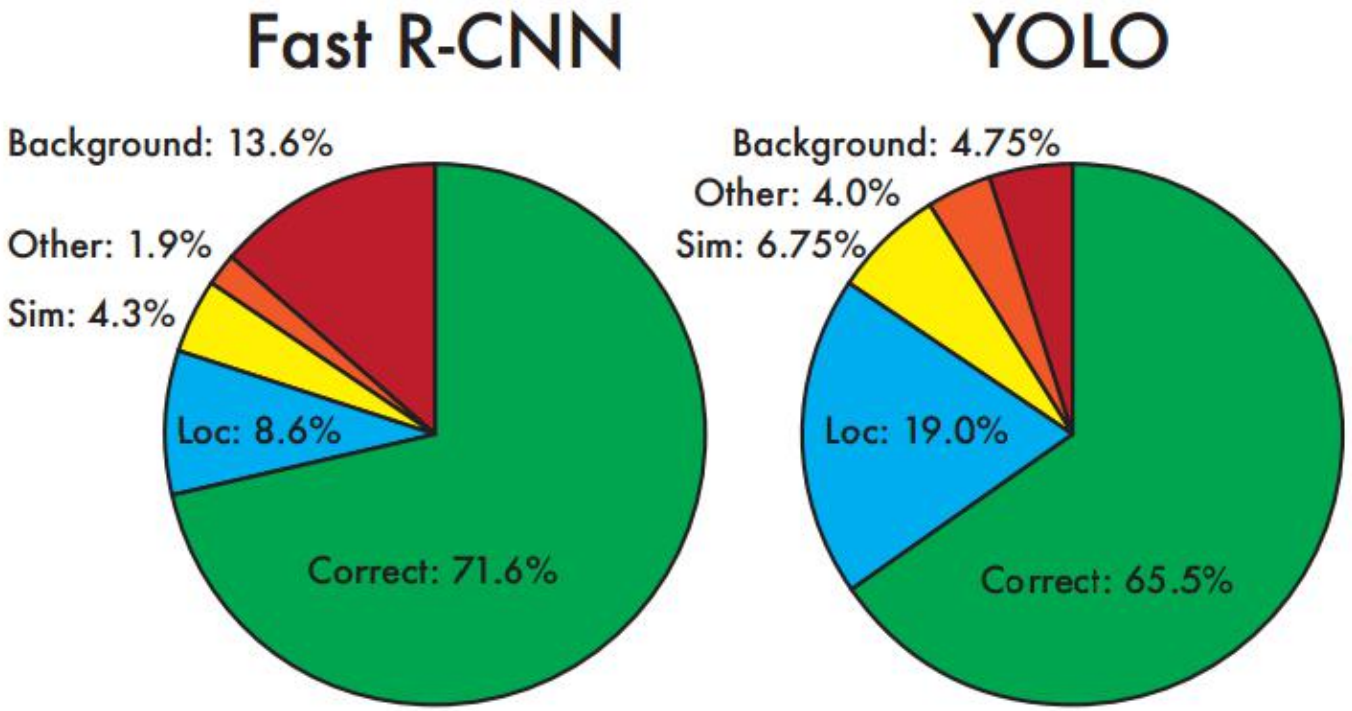
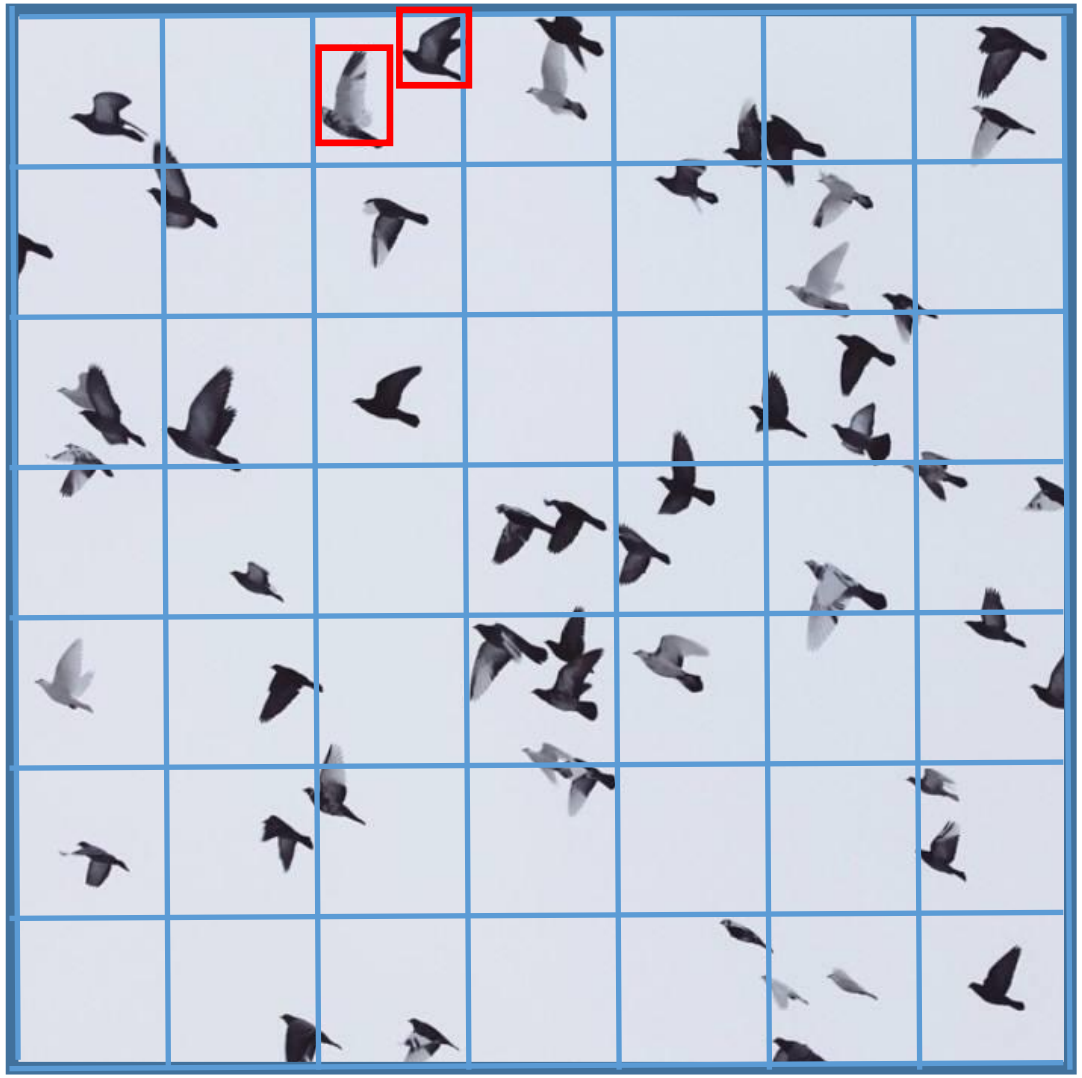


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

YOLO v1

- 优点：
  - 运行速度快；
  - 背景预测错误的情况比较少；
- 缺点：
  - 对于实际物体的效果没有Faster R-CNN效果好；
  - 如果一个grid cell中包含多个相同类别的小物体，那么YOLO v1仅可以检测出一个物体。



YOLO v2

- YOLO9000:Better, Faster, Stronger， 主要分成三个方面对模型进行改进：
  - 1. Better：从精度方面进行改进，让效果从YOLO v1的63.4%mAP上升到YOLO v2的78.6%mAP，基本和Faster R-CNN以及SSD持平。
  - 2. Faster：网络结构方面做了更改，让模型速度更快；
  - 3. Stronger：对这个损失函数做一个变化；

YOLO v2

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

YOLO v2

- Batch Normalization：
  - 在每一个卷积层后加入Batch Normalization，mAP提升2%，Batch Normalization有助于规范化模型，防止过拟合。
- High Resolution Classifier：
  - 一般的目标检测方法中，基本上会使用ImageNet预训练的模型来提取特征，比如使用AlexNet或者VGG网络，那么输入的图片会被resize到不足256*256的大小，这样会导致分辨率不够高，目标检测比较困难；在YOLO v2中自定义了darknet分类网络，将图像的输入分辨率更改为448*448，然后在ImageNet上训练10轮，训练后的网络可以适应高分辨率的输入；应用到检测的时候，对检测部分网络进行fine tune，mAP提升4%。

YOLO v2

- Convolutional with Anchor Boxes：
  - 借鉴Faster R-CNN中的anchor思想，产生多个bounding boxes候选框，通过这种方式可以提升模型的recall召回率。
  - 删除全部的全连接层，去掉最后一个池化层，以确保输出的特征图具有更高的分辨率，然后通过缩减网络让图片输入分辨率为416*416，这样最终输出的特征图为13*13；

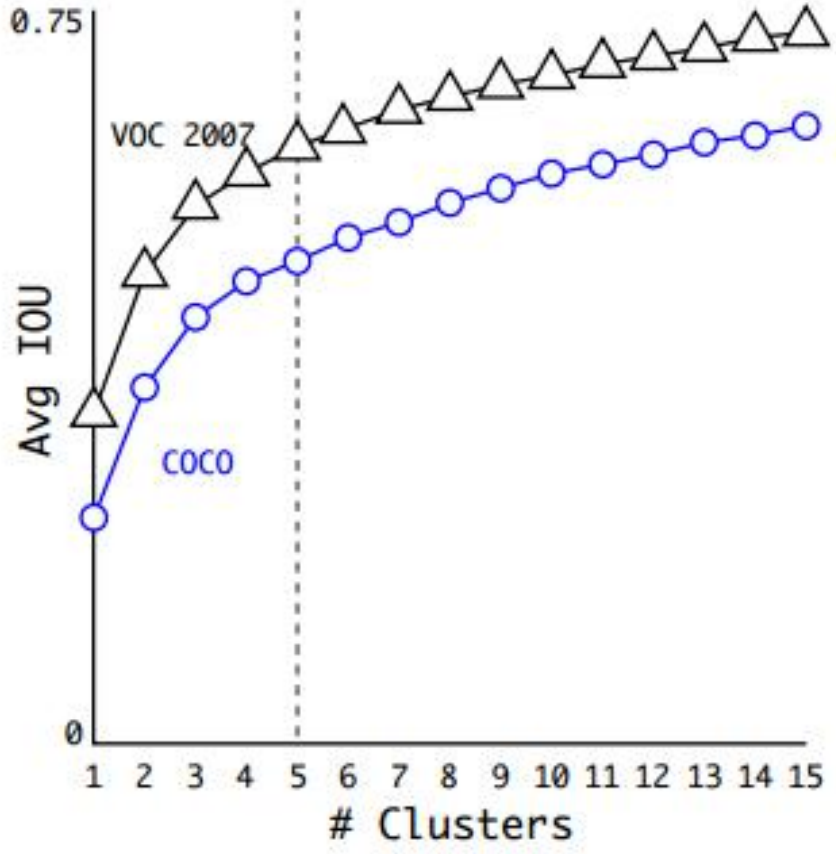
YOLO v2

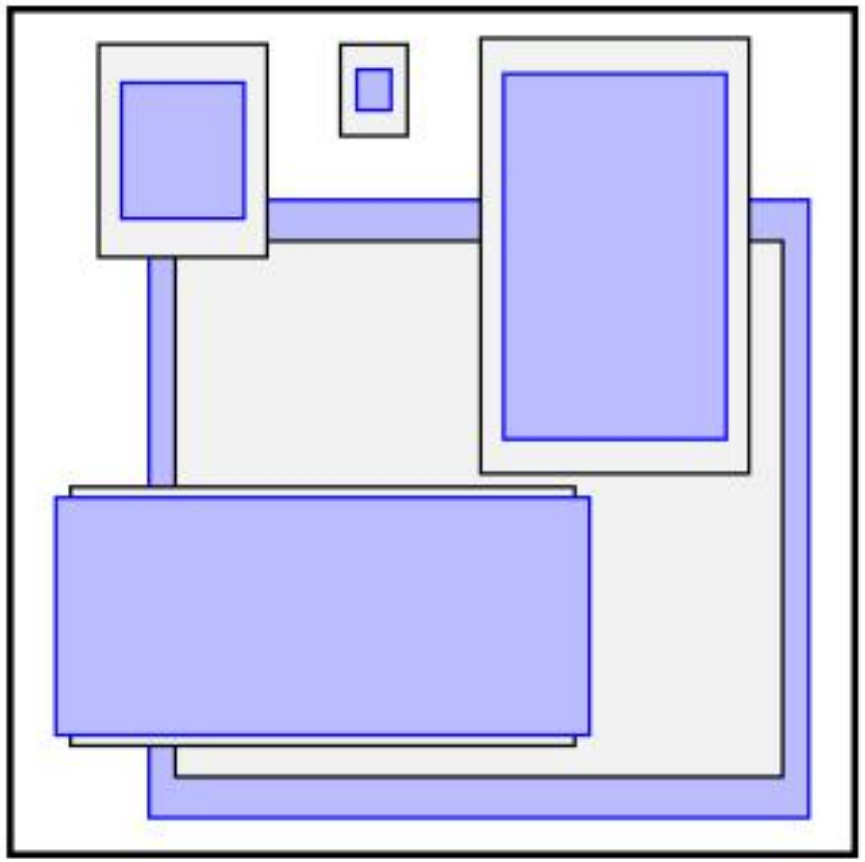
- Dimension Cluster(维度聚类):
  - Anchor Boxes的宽高纬度通常需要通过精选的先验框来给定，然后通过网络学习转换系数，最终得到准确的bounding box候选框；如果可以通过维度聚类一开始就给定更具有代表性的boxes维度，那么网络就会更容易预测位置。
  - 使用K-Means聚类方法来训练bounding boxes；采用IoU作为KMeans聚类的距离公式。

$$d(\text{box}, \text{centroid}) = 1 - \text{IOU}(\text{box}, \text{centroid})$$

YOLO v2

Box Generation	#	Avg IOU
Cluster SSE	5	58.7
Cluster IOU	5	61.0
Anchor Boxes [15]	9	60.9
Cluster IOU	9	67.2





YOLO v2

- Direct Location Prediction(直接位置预测)：
  - 在Anchor Boxes中，模型不是特别稳定，原因是：模型的位置预测值为偏移量的值(在整个图像上的)，在模型中相当于anchor可以检测很远目标的box的情况，这样就会导致模型收敛比较慢。
  - YOLO v2中不采用直接的offset方法，使用了预测相对于grid cell的坐标位置的方法，并且将ground truth通过logistic函数限制在0~1之间。
  - 通过Dimension Cluster+Direct Location Prediction的改进，mAP提升5%。

YOLO v2

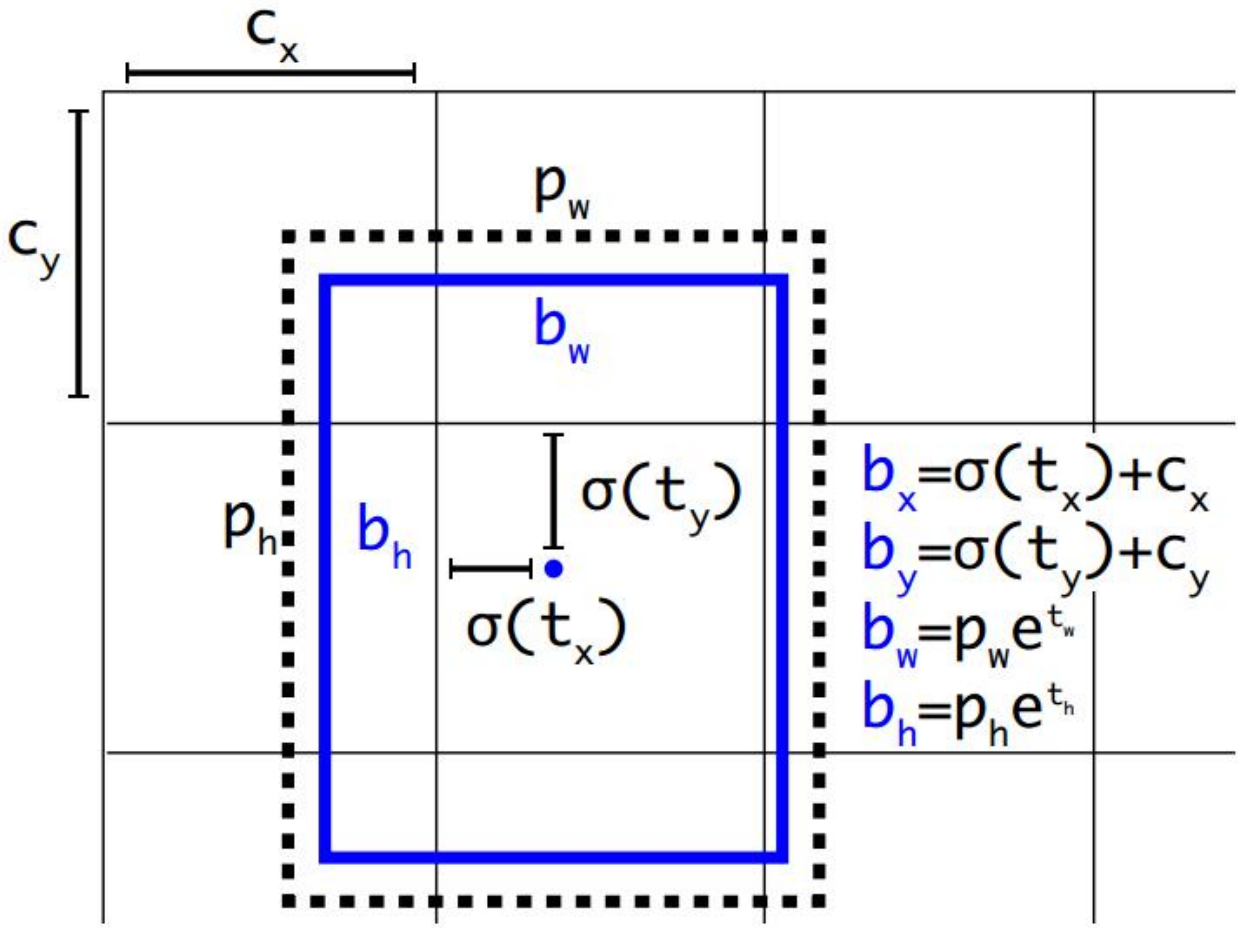
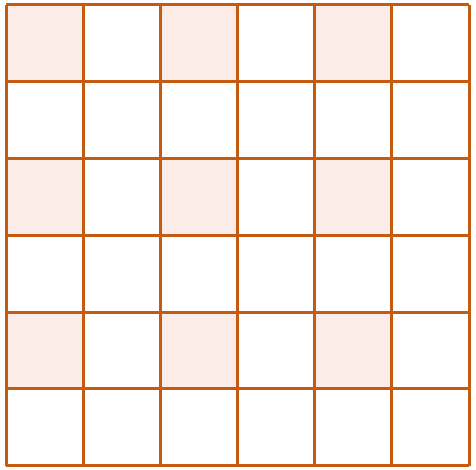


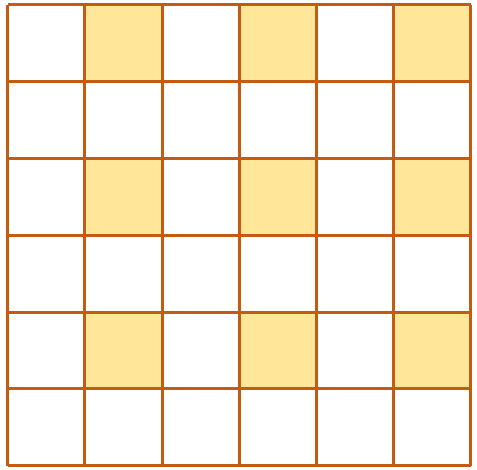
Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

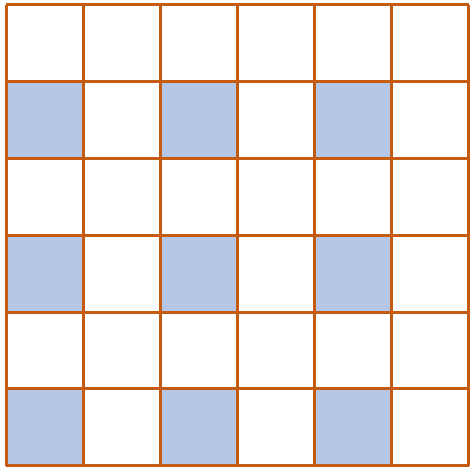
YOLO v2

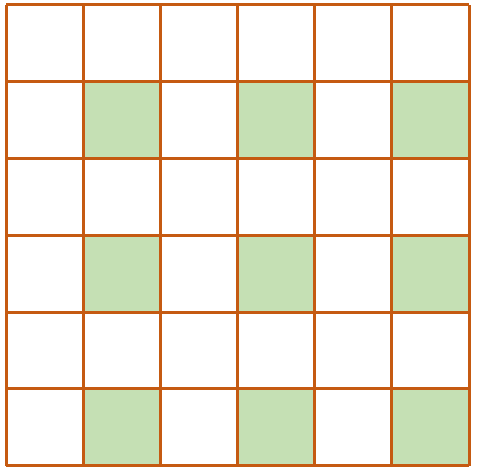
- Fine-Grained Features(细粒度特征)
  - 在Faster R-CNN和SSD中，通过不同的方式获得了多尺度的适应性，Faster R-CNN中使用不同的scale，SSD直接从不同大小的feature map上来提取ROI区域；为了提升对小尺度物体的检测，在YOLO v2中加入了转移层(passthrough layer)，这个层次的功能就是讲浅层特征图(26*26)连接到深层特征图(13*13)中，类似ResNet的残差网络结构。
  - 对于26*26*512的特征图按行、按列隔一个进行采样，产生4个13*13*512维度的新特征图，然后concat合并得到13*13*2048的特征图，最后将其连接到深层特征图上。相当于做了特征融合，对于小目标检测比较有效。
  - 通过这种结构，mAP提升了1%。

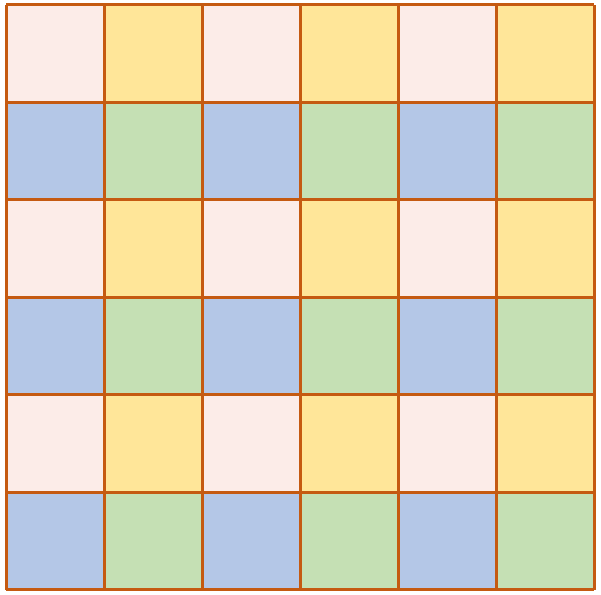
YOLO v2







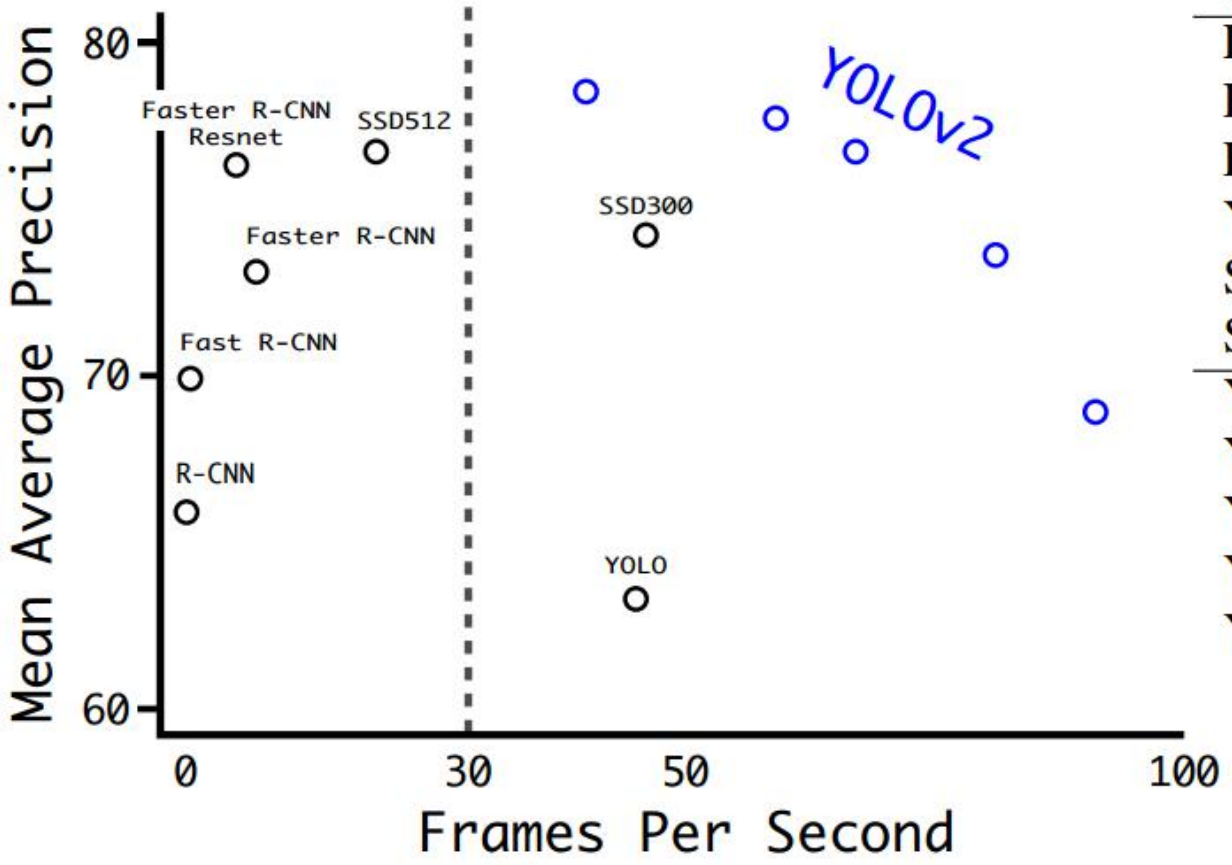




YOLO v2

- Multi-Scale Training：
  - 由于YOLO v2中仅存在卷积层和池化层，所以可以进行动态调整，每经过10个epoch，随机选择新的图片尺寸进行训练。由于YOLO v2中降采样的参数为32，所以以32个像素为增量值设置不同大小的图像来进行训练，最小尺寸320，最大尺寸608，尺寸可选值{320,352,....,608}总共十个不同尺寸的的图像。
  - 这种设计让YOLO在低性能的GPU、高帧率视频等场景的应用更加适合。

YOLO v2



Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

Figure 4: Accuracy and speed on VOC 2007.

YOLO v2

- 为了改善YOLO模型的检测速度，YOLO v2在Faster方面也做了一些改进。
- 大多数神经网络依赖于VGG Net来提取特征，VGG-16特征提取功能是非常强大的，但是复杂度有点高，对于224*224的图像，前向计算高达306.9亿次浮点数运算。
- YOLO v2中使用基于GoogleNet的定制网络DarkNet，一次前向传播仅需要85.2亿次浮点数计算。精度相比来讲低2%(88%/90%)

YOLO v2

- 大量使用3*3的卷积；
- 在3*3卷积前使用1*1卷积来压缩通道；
- 在每一个卷积后加入BN，稳定模型；
- 使用Global Average Pooling；
- 使用Max Pooling。

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

Table 6: Darknet-19.

YOLO v2

- 分类模型训练：使用下列参数及数据处理方式，将Darknet-19在标准的1000类ImageNet上训练160epoch；
  - 随机梯度下降
  - starting learning rate:0.1
  - polynomial rate decay: 4
  - weight decay: 0.0005
  - momentum: 0.9
  - 输入数据大小: 224*224
  - 数据增强： random crops、rotations、hue/saturation/exposure shifts.

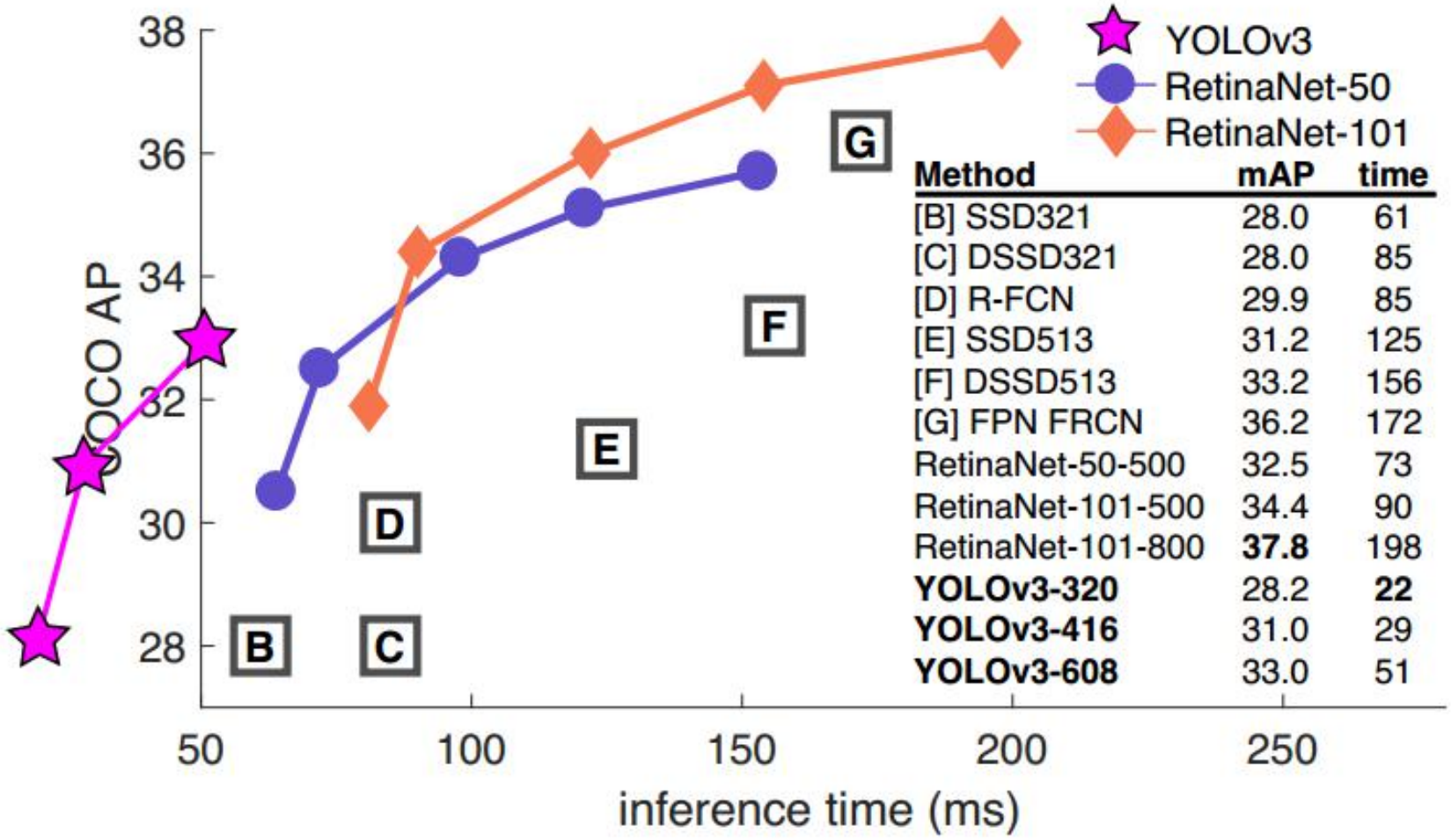
YOLO v2

- 分类模型训练：当初始224*244的训练完成后(160epochs)：进行更高分辨率下的图像分类训练(训练10epochs)：
  - learning_rate: 0.001
  - 输入数据大小: 448*448

YOLO v2

- 检测网络的训练；将分类网络的最后一个卷积层去掉，更改为三个3*3*1024的卷积层，并且每个卷积层后跟一个1*1的卷积层，输出维度为检测需要的数目；比如在VOC数据集中，需要预测5个框，每个框4个坐标值+1个置信度+20个类别概率值，也就是输出为125维。同时将转移层(passthrough layer)从倒数第二层做一个连接的操作。
- 训练方式以及Loss的定义和YOLO v1类似。

YOLO v3



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

YOLO v3

- YOLO v3对YOLO v1和YOLO v2的内容既有保留又有改进，其保留的内容如下：
  - 从YOLO v1开始，YOLO算法都是通过划分单元格的方式来检测的，只是划分的数量不一样；
  - 全部都是采用Leaky ReLU激活函数；
  - 采用端到端的训练；
  - 从YOLO v2开始，将BN和Leaky ReLU放到每一个卷积层之后；
  - 从YOLO v2开始，多尺度训练，在速度和准确率之间tradeoff。
  - 从YOLO v2开始，位置信息的预测全部是预测相对于grid cell的位置信息。

YOLO v3

- 在YOLO v3中借鉴了大量其它网络结构的优点，其主要改进如下：
  - 1. YOLO v3的改进主要是backbone网络的提升，从v2的darknet-19到v3的deaknet-53。在v3中，还提供了轻量高速的网络tiny-darknet；
  - 2. 借鉴SSD/FPN在不同scala的feature map上提取特征信息，相比于v2中仅仅通过多尺度图像的训练来讲，v3对于小目标的检测效果更好；
  - 3. Anchor Boxes数目从v2中的5个提升到v3中的9个；
    - 10x13, 16x30, 33x23, 30x61, 64x45, 59x119, 116x90, 156x198, 373x326
  - 4. 采用Logistic分类器替换Softmax分类器；
  - 5. Loss损失函数进行了改进；

YOLO v3

- backbone

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

Table 6: Darknet-19.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

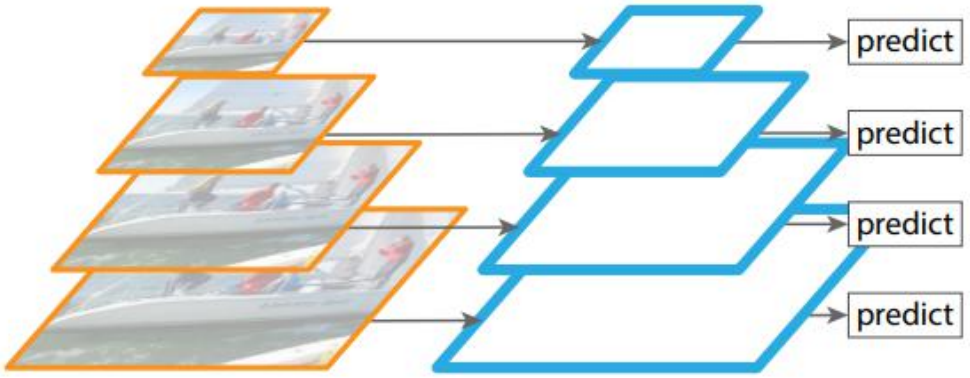
Table 1. **Darknet-53.**

YOLO v3

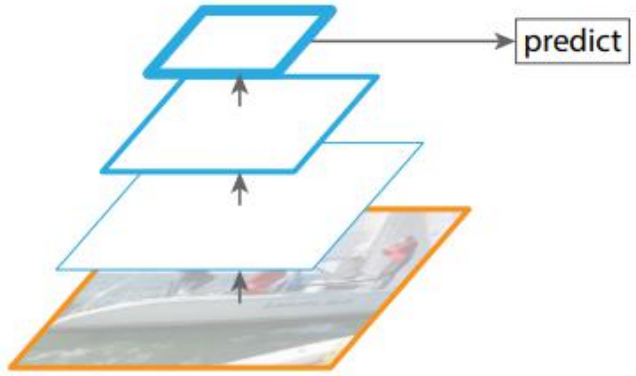
- YOLO v3在保证实时性(fps>=36)要求的情况下，尽量要求这个performance。

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

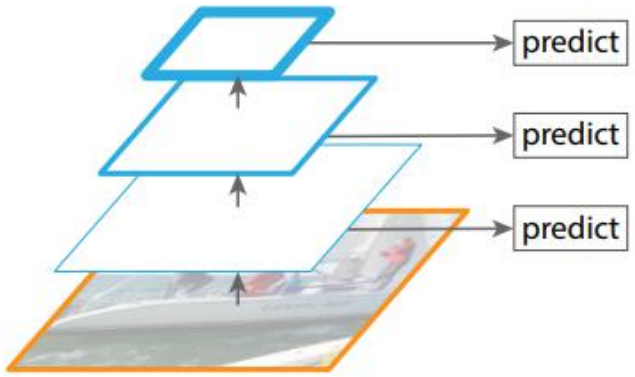
YOLO v3



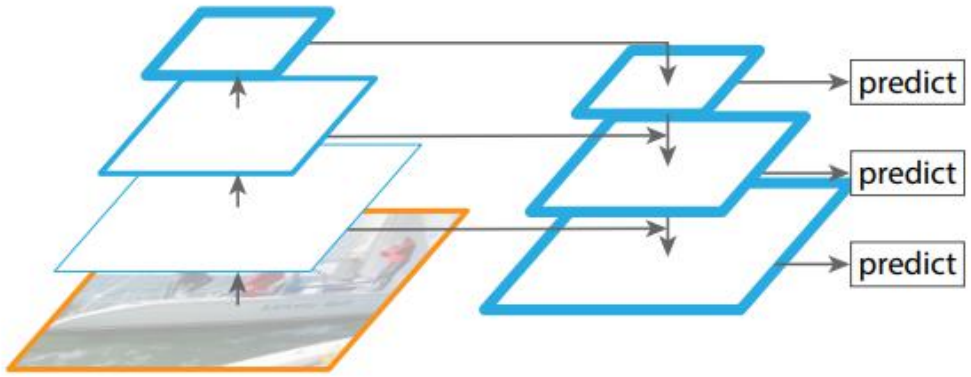
(a) Featurized image pyramid



(b) Single feature map

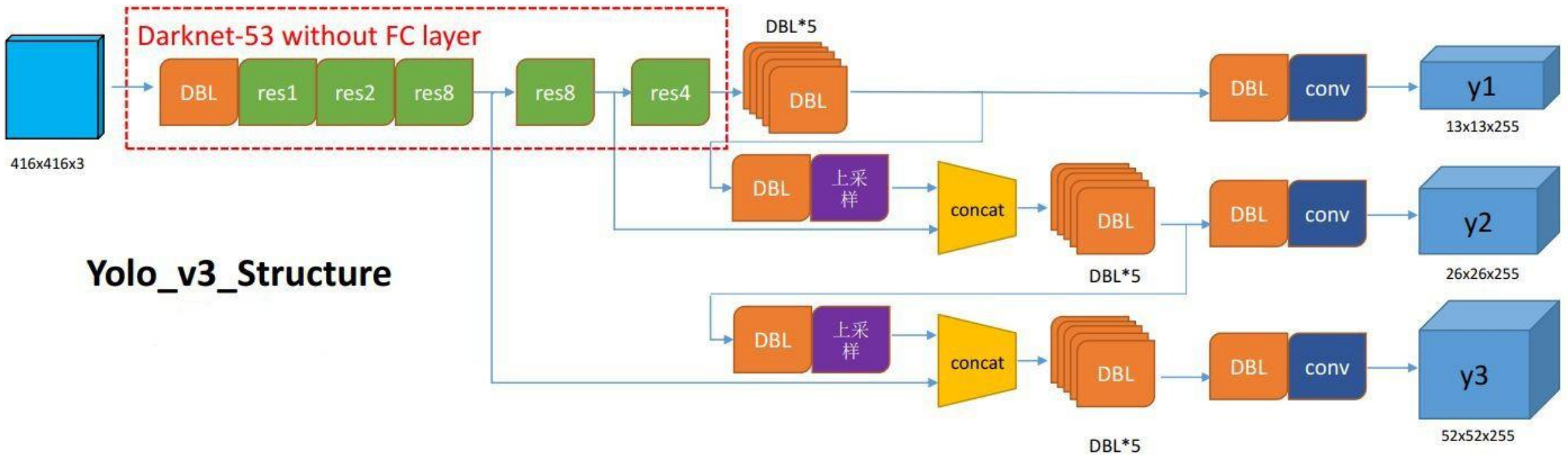


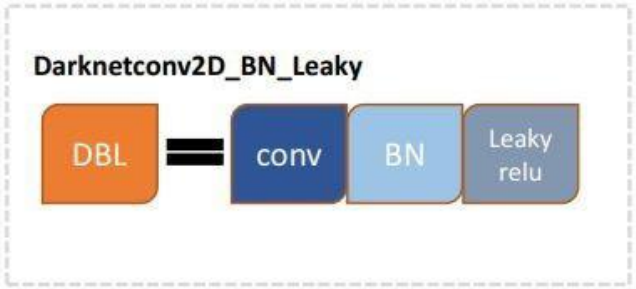
(c) Pyramidal feature hierarchy

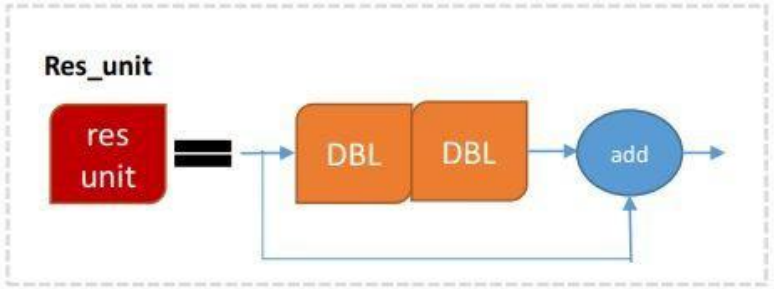


(d) Feature Pyramid Network

YOLO v3


Darknet-53 without FC layer
DBL
res1
res2
res8
res8
res4
DBL*5
DBL
416x416x3
DBL
conv
y1
13x13x255
DBL
上采样
concat
DBL
DBL*5
DBL
conv
y2
26x26x255
DBL
上采样
concat
DBL
DBL*5
DBL
conv
y3
52x52x255
Yolo_v3_Structure


Darknetconv2D_BN_Leaky
DBL
conv
BN
Leaky relu


Res_unit
res unit
DBL
DBL
add


Resblock_body
resn
zero padding
DBL
res unit
Res_unit*n

YOLO v3

- YOLO v3之所以不使用Softmax对每个框进行分类的主要原因如下：
  - 1. Softmax使得每个框分类一个类别(score最大的一个)，但是在实际情况下，目标可能会重复，但是Softmax不适合多标签分类；
  - 2. Softmax可以被多个独立的logistic分类器替代，并且效果不会下降；
  - NOTE: 使用Logistic回归分类器可以直接判断是不是属于这个类别即可。

YOLO v3

- YOLO v3 Loss函数采用类似YOLO v1的损失函数，其主要更改的地方是将v1中关于confidence和class类别概率部分的损失函数更改为logistic交叉熵损失函数(tf.nn.sigmoid_cross_entropy_with_logits)。

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

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

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

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

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

更改为 binary_crossentropy

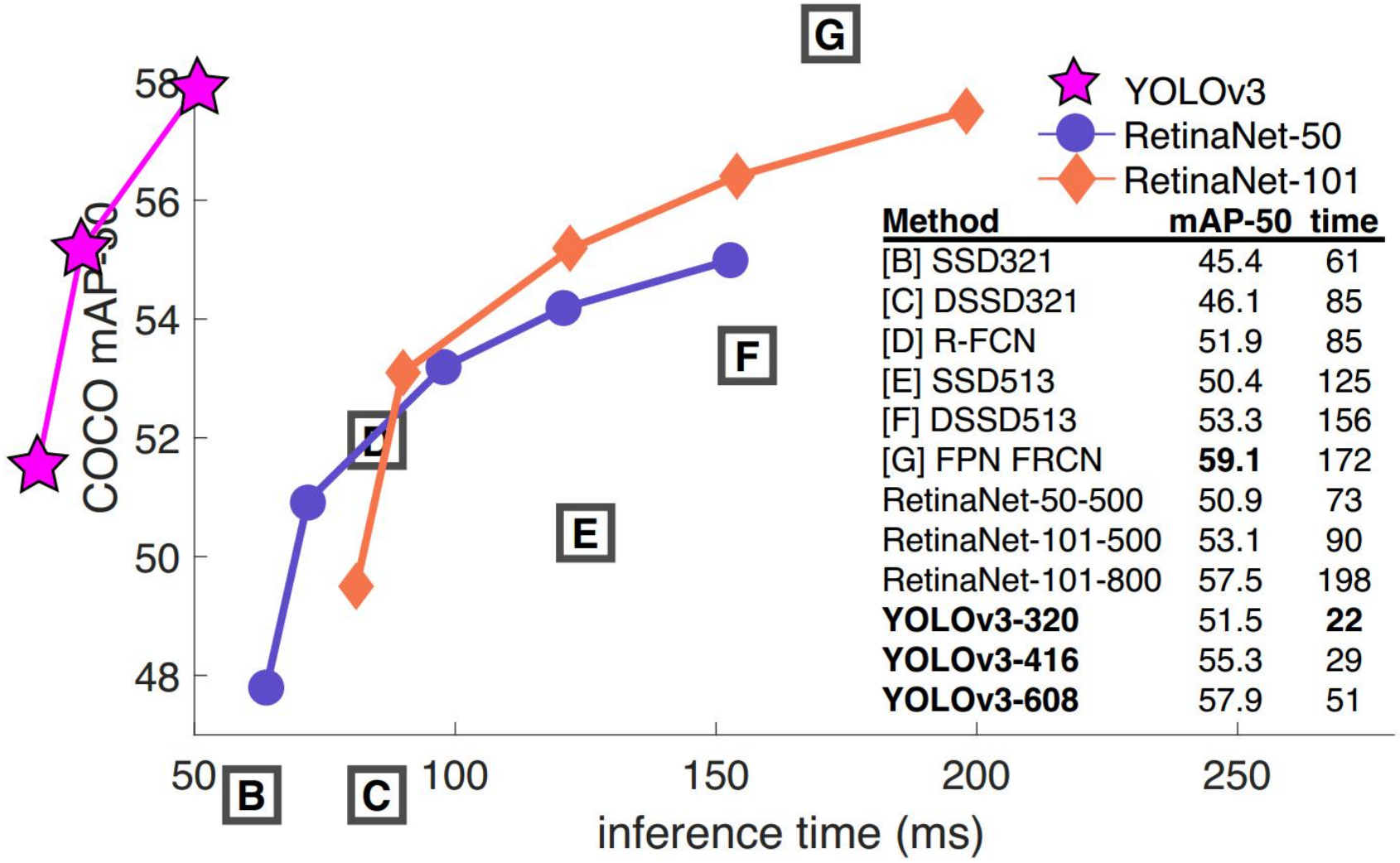
YOLO v3

- 训练操作类似YOLO v1和v2；主要特点如下：
  - 网络结构darknet；
  - 输入数据完整图像，不进行hard negative mining或者其它操作；
  - 多尺度训练multi scale；
  - 数据增强data augmentation;
  - 批归一化batch normalization;

YOLO v3

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

YOLO v3



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51