YOLO 9000

TAEWAN KIM

DNN-based Object Detection

R-CNN→ MultiBox → SPP-Net → DeepID-Net →NoC → **Fast R-CNN** → DeepBox → **MR-CNN** →
2013.11 ECCV '14 PAMI '16 ICCV '15 ICCV '15

Faster R-CNN →**YOLO** → AttentionNet → DenseBox → **SSD** →Inside-OutsideNet(ION) →G-CNN →
NIPS '15 CVPR '16 ICCV '15 ECCV '16 CVPR '16

HyperNet → MultiPathNet → CRAFT → OHEM → **R-FCN** → **MS-CNN** → PVANET → GBDNet →
BMVC '16 CVPR '16 CVPR '16 NIPS '16 ECCV '16

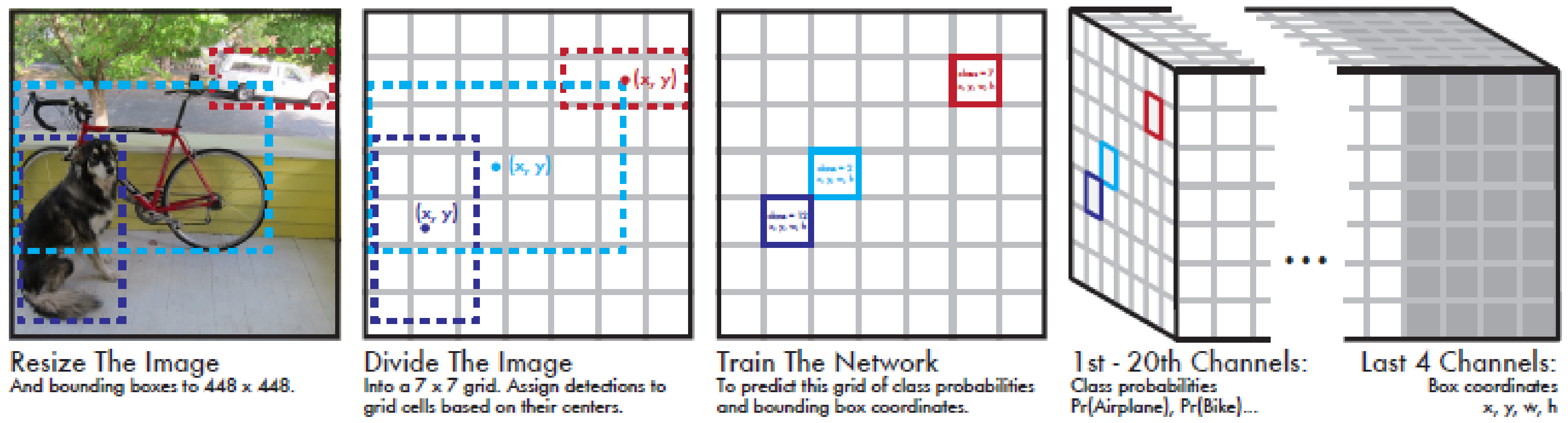
StuffNet → Feature Pyramid Net **(FPN)** → **YOLO v2** → DSSD → CC-Net → Mask R-CNN …
CVPR '17

YOLO v2

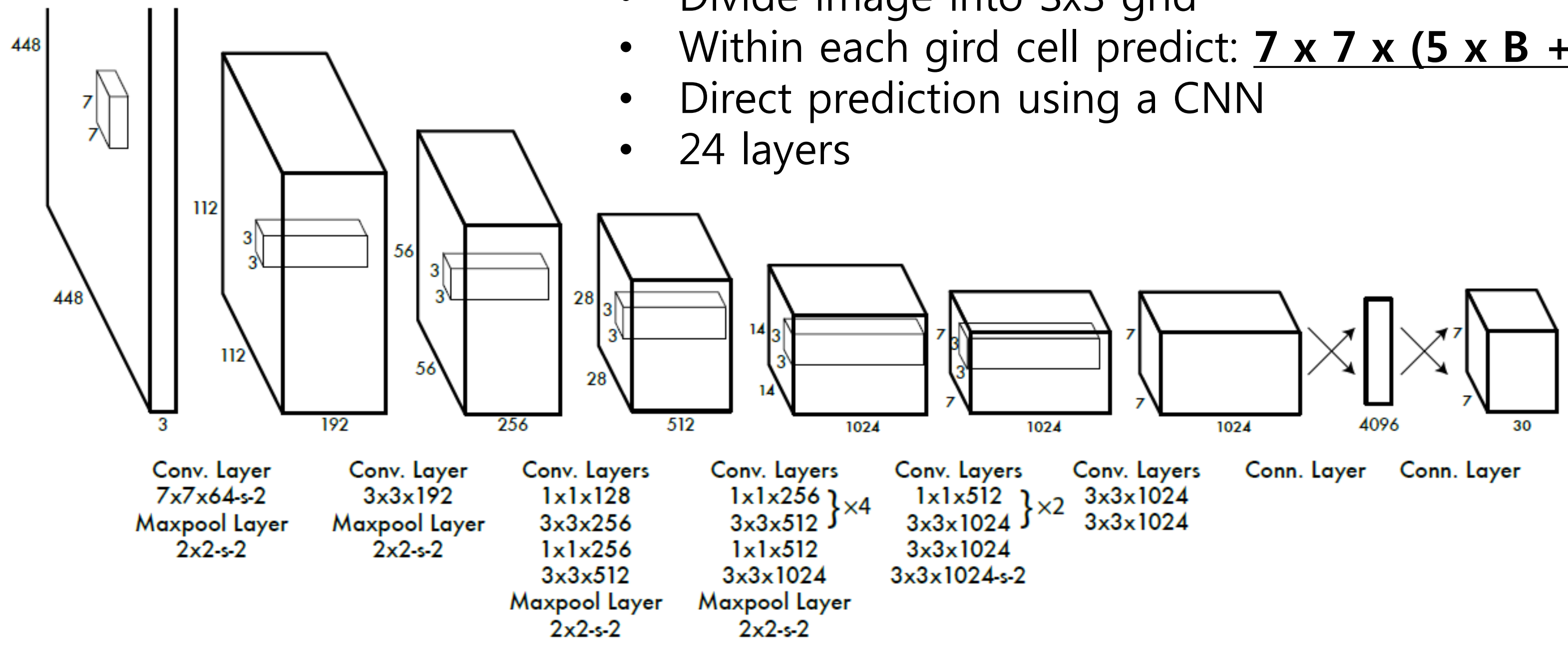
- YOLO9000: Better, Faster, Stronger
  - arxiv: https://arxiv.org/abs/1612.08242
  - code: http://pjreddie.com/yolo9000/
  - github(Chainer): https://github.com/leetenki/YOLOv2
  - github(Keras): https://github.com/allanzelener/YAD2K
  - github(PyTorch): https://github.com/longcw/yolo2-pytorch
  - github(Tensorflow): https://github.com/hizhangp/yolo_tensorflow

- Yolo_mark: GUI for marking bounded boxes of objects in images for training Yolo v2
  - github: https://github.com/AlexeyAB/Yolo_mark

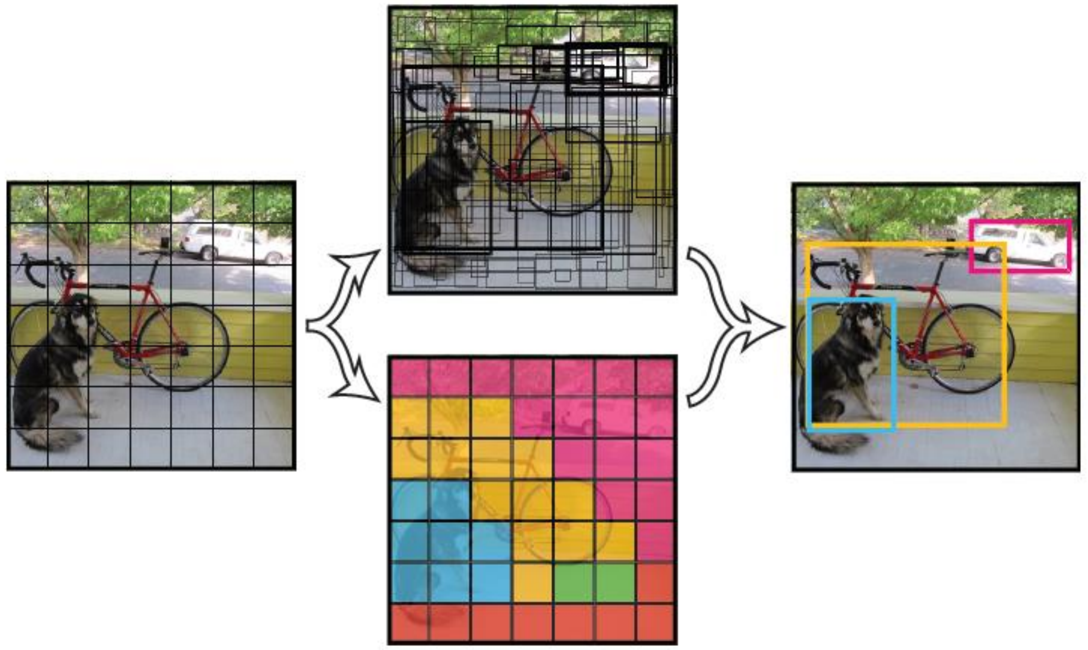
YOLO version 1 (CVPR 2016)


(x, y)
(x, y)
(x, y)
Resize The Image
And bounding boxes to 448 x 448.
Divide The Image
Into a 7 x 7 grid. Assign detections to grid cells based on their centers.
Train The Network
To predict this grid of class probabilities and bounding box coordinates.
1st - 20th Channels:
Class probabilities
Pr(Airplane), Pr(Bike)...
Last 4 Channels:
Box coordinates
x, y, w, h

- Divide image into SxS grid
- Within each gird cell predict: <u>**7 x 7 x (5 x B + C) tensor**</u>
- Direct prediction using a CNN
- 24 layers



YOLO version 1 (CVPR 2016)

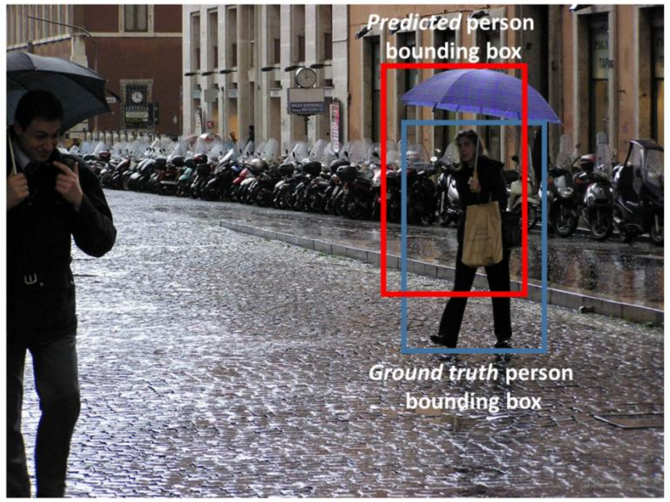


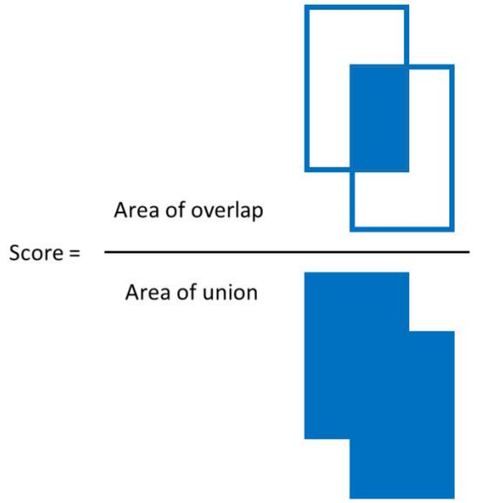
- 입력 이미지를 SxS 그리드로 나눔
- 각 grid cell 당 B개의 bounding box 예측
- 각 bounding box 는 5개의 예측 값
  - (x, y) : 물체의 중심점
  - (w, h) : 물체의 width, height
  - p : confidence 확률
- 최종적으로 S x S x (5*B + C), S = 7, B = 2, C = 20 (Pascal VOC)
  - 그러므로 최종 CNN 결과는 7x7x30 = 1,470
- 물체의 중심점 2개가 모두 같은 grid cell안에 있어도 검출 가능
- 검정색 선이 굵을 수록 더욱 확률이 높은 bounding box

YOLO version 1 (CVPR 2016)

- 이미지 내 물체의 중심점이 grid cell에 속한다면 그 grid cell은 물체를 검출하는 책임이 주어짐
- Grid Cell 당 B개의 bounding boxes를 구한다. (예전 v1에서는 1개 였음, source 도 업데이트함)
- Cell당 확률값은 물체일 확률 x box의 교차 영역 비율
    - IOU : intersection over union

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

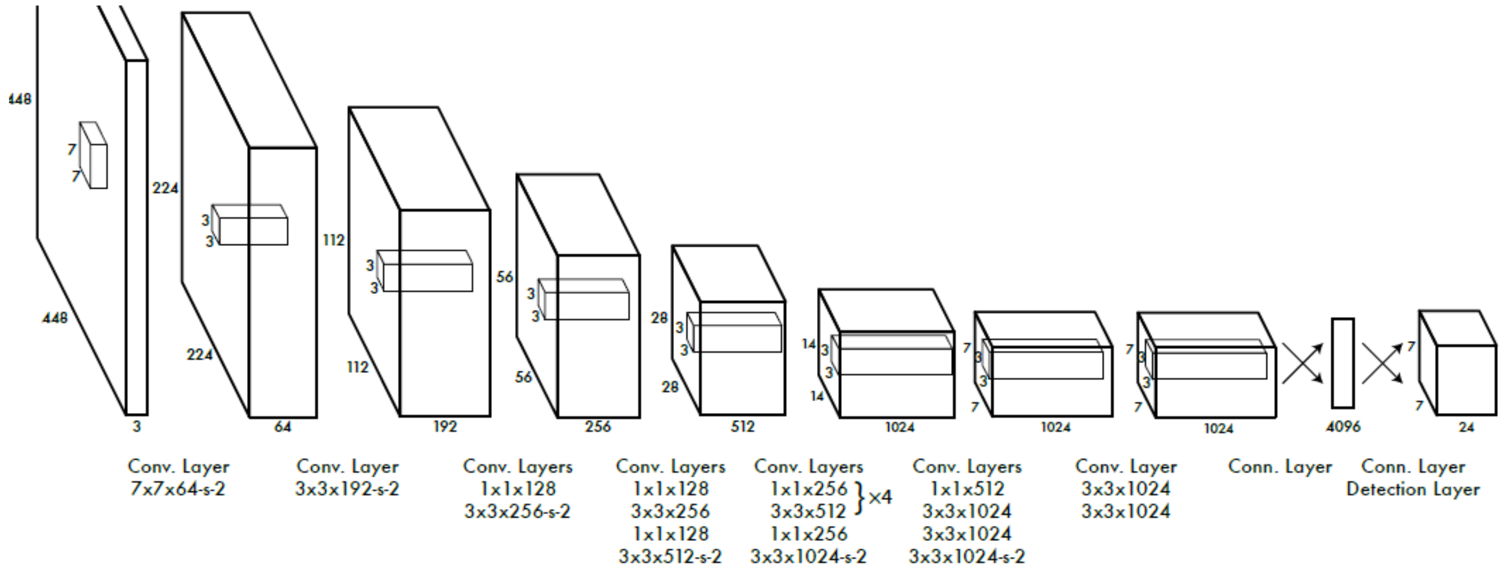

Predicted person bounding box
Ground truth person bounding box



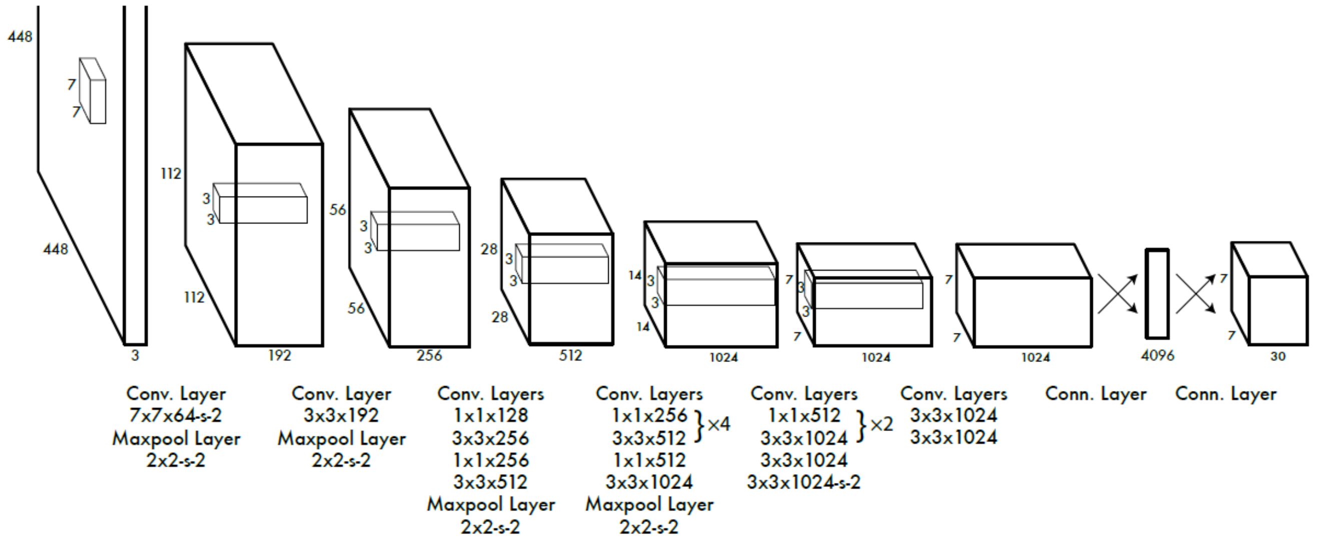
YOLO version 1 (CVPR 2016)

- V1에서는 7x7x(20 class + 4 (x,y,w,h)
- V4에서는 7x7x(20 class + 5 (x,y,w,h,p) * 2)

V1
NIPS
2015



V4
CVPR
2016



YOLO version 1 (CVPR 2016)

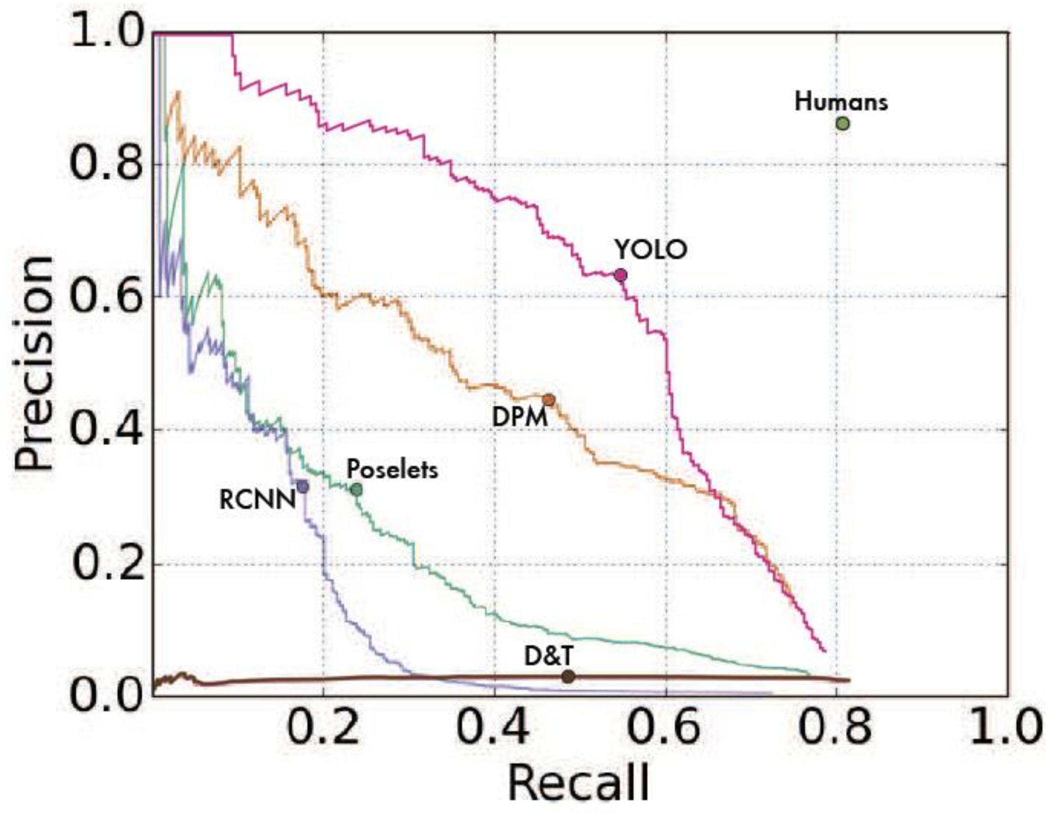
- Loss function

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

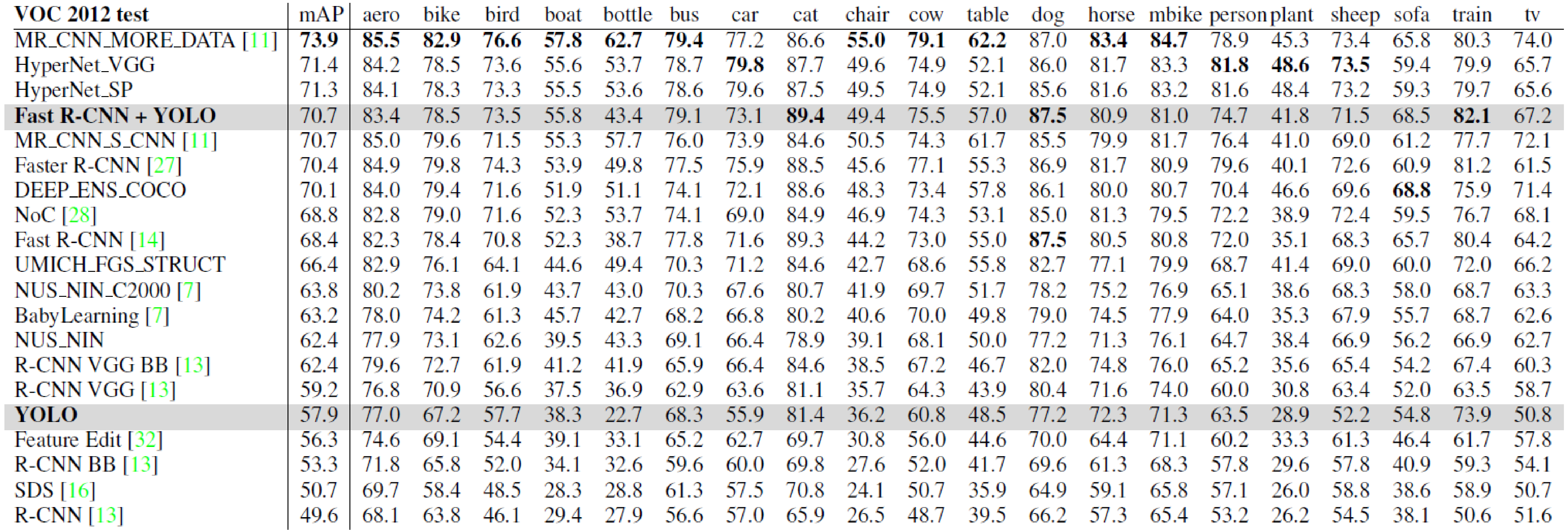
YOLO version 1 (CVPR 2016)

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	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[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

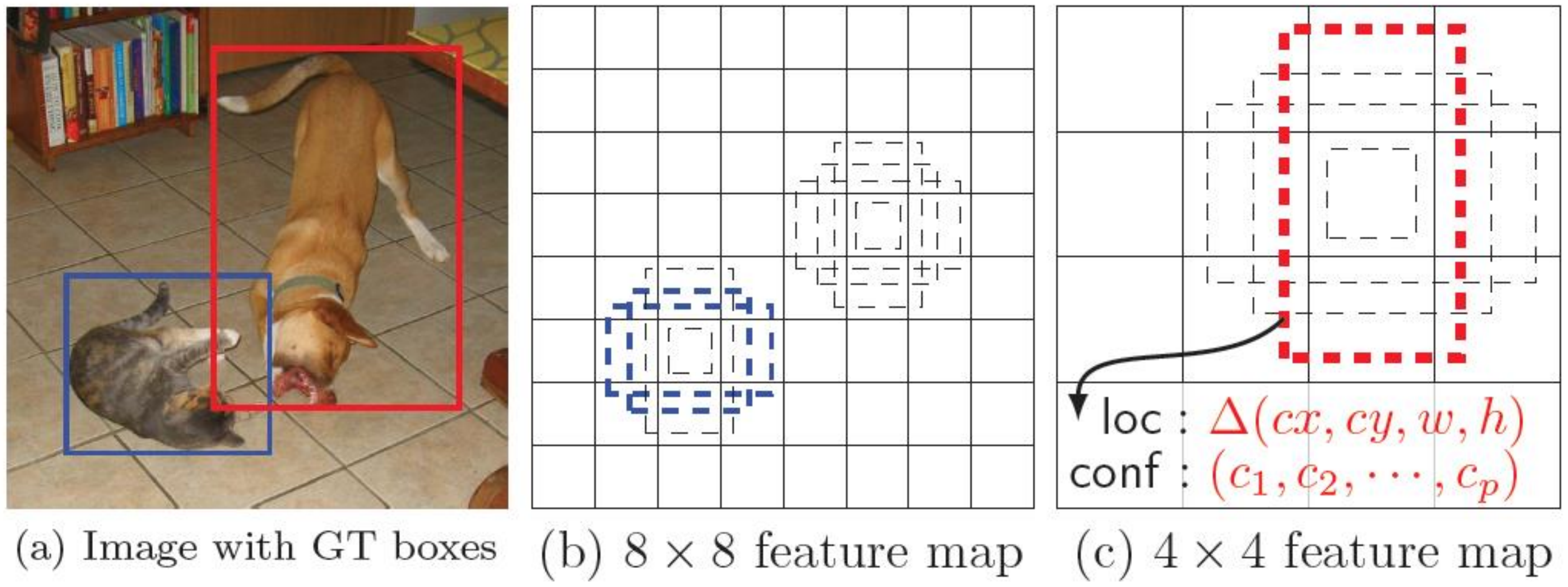


YOLO version 1 (CVPR 2016)



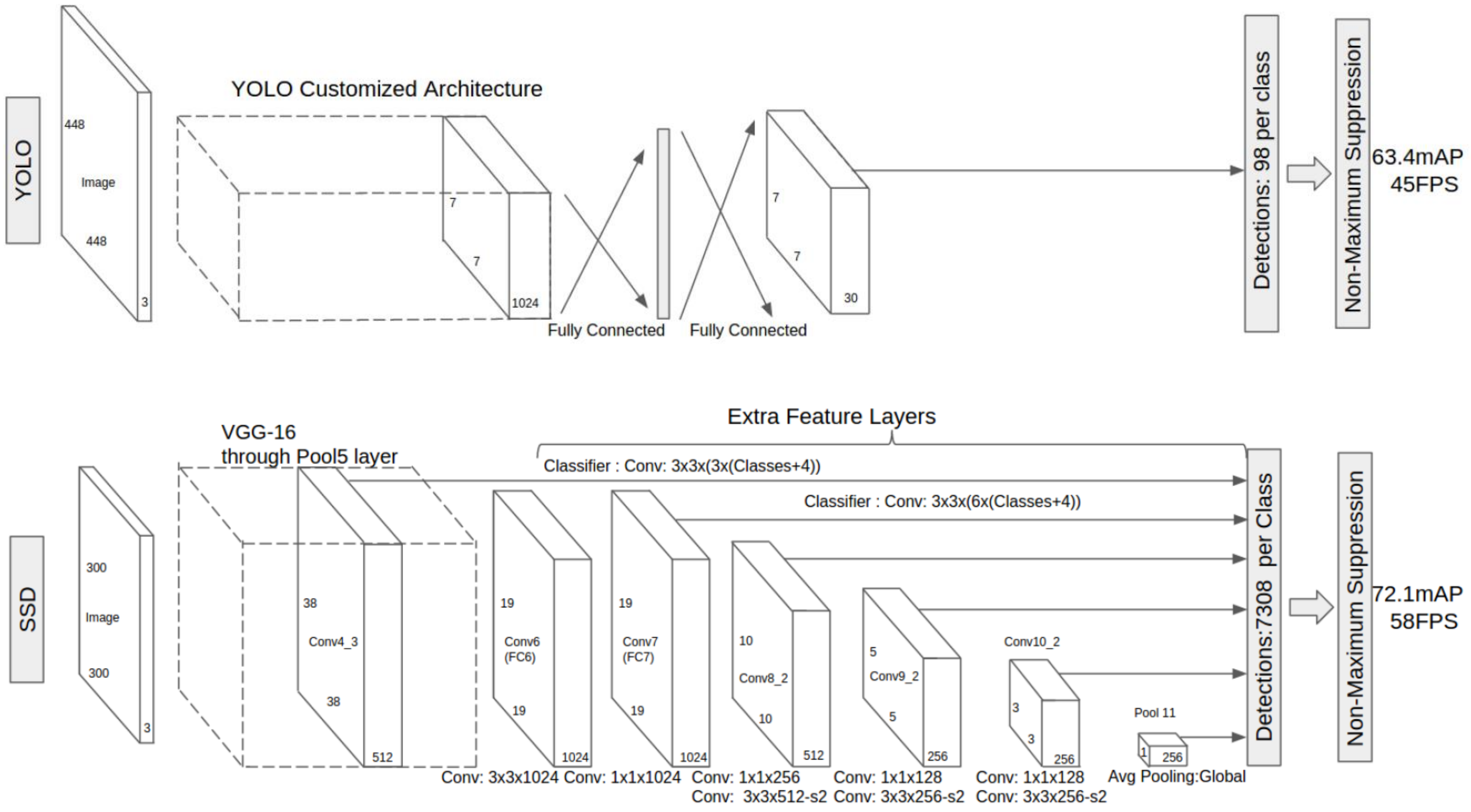
VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	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 [27]	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 [28]	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 [32]	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

참고 SSD : Single Shot MultiBox Detector (ECCV 2016)

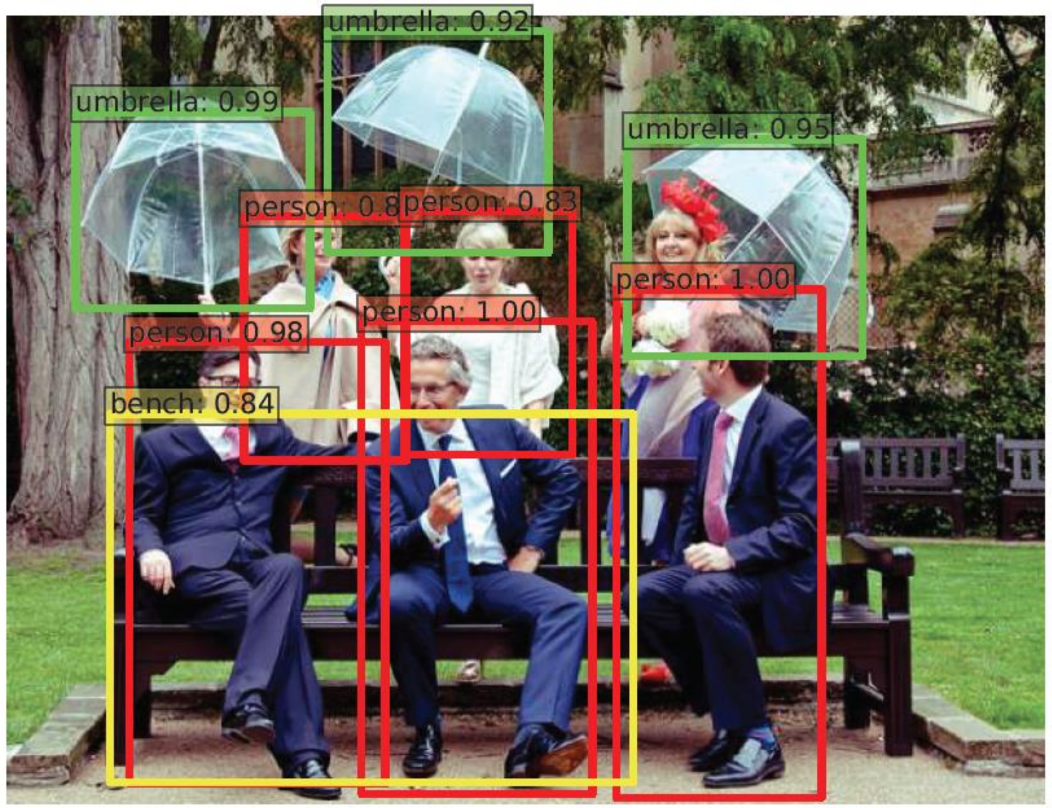


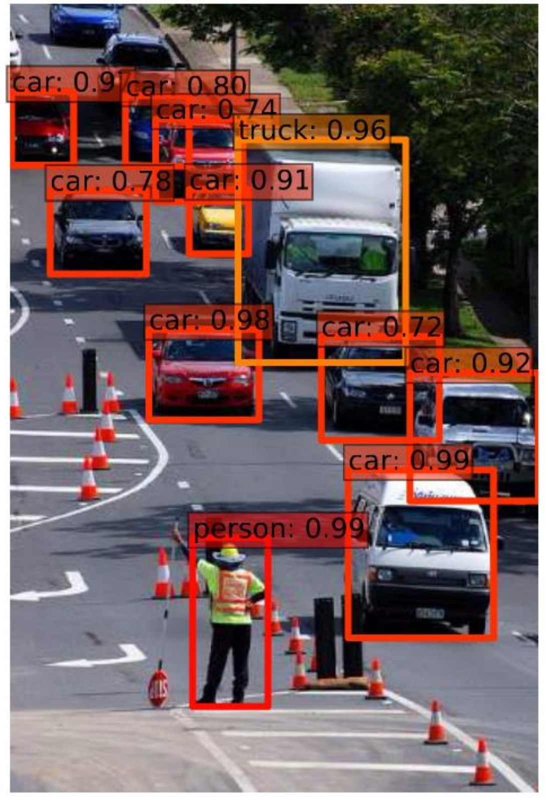
- Key Idea = Multi-scale Convolutional Bouding Box
- VGG 16을 사용했지만 다른 CNN을 사용해도 됨
- YOLO v1 보다 빠르고 작은 물체에서도 검출이 잘 되며 accuracy도 좋음
  - YOLO v1 : 45FPS, 63.4 mAP
  - Faster R-CNN = 7FPS, 73.2 mAP
  - SSD : 58 FPS, 72.1 mAP

참고 SSD : Single Shot MultiBox Detector (ECCV 2016)



참고 SSD : Single Shot MultiBox Detector (ECCV 2016)





Method	*mAP*	FPS	# Boxes
Faster R-CNN [2](VGG16)	73.2	7	300
Faster R-CNN [2](ZF)	62.1	17	300
YOLO [5]	63.4	45	98
Fast YOLO [5]	52.7	155	98
SSD300	72.1	58	7308
SSD500	**75.1**	23	20097

YOLO 9000

a state-of-the-art, real-time object detection system that can detect over 9,000 object categories

1. Propose various improvements to the YOLO: YOLO v2
   - 67fps: 76.8 mAP on VOC 2007
   - 40fps: 78.6 mAP on VOC 2007

2. Propose a method to jointly train on object detection and classification
   - Train YOLO9000 simultaneously on COCO detection dataset & ImageNet classification dataset
   - 19.7 mAP on ImageNet despite only having detection data for 44 of the 200 classes
   - 16.0 mAP on COCO despite only having detection data for 156 of the 200 classes
   - 9000 different object categories while satisfying real-time speed

YOLO v2

http://pjreddie.com/yolo

YOLO 9000

Goal : Improving recall and localization while maintaining classification accuracy

1. Better
   - Batch normalization
   - High resolution classifier
   - Convolutional with Anchor boxes
   - Dimension clusters
   - Direct location prediction
   - Fine-grained features
   - Multi-scale training
2. Faster
   - Darknet-19
   - Training for classification
   - Training for detection
3. Stronger
   - Hierarchical classification
   - Dataset combination with Word-tree
   - Joint classification and detection

Better – Batch normalization

- **Mini-batch normalization**: training data 전체에 대해 mean과 variance를 구하는 것이 아니라, mini-batch 단위로 접근하여 계산
- **2% improvement in mAP**

Better – High resolution classifier

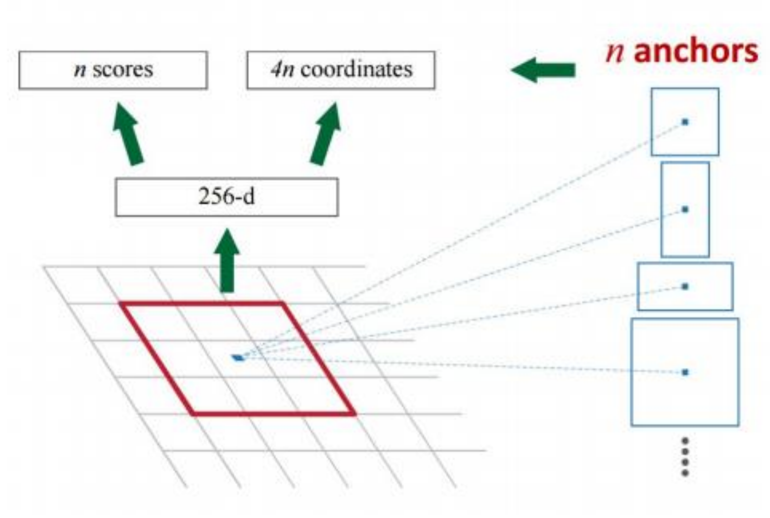
- General classifier: 256 x 256
- Original YOLO classifier: 224 x 224 → detector 448 x 448
- YOLO v2: fine tune the classification network at 448 x 448 for 10 epochs on ImageNet → detector
- **4% improvement in mAP**

Better – Convolutional with Anchor Boxes

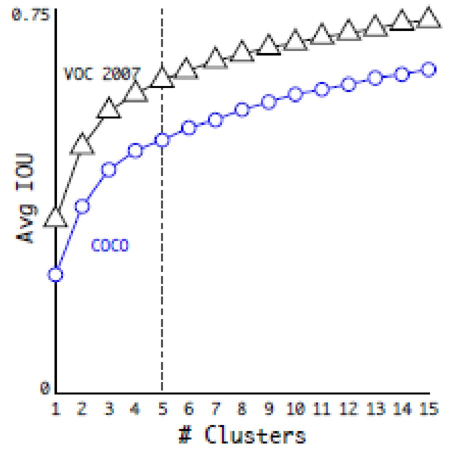
- Faster R-CNN: predicts bounding boxes using hand-picked priors
- Original YOLO: predicts the coordinates of bounding boxes directly using FC
- YOLO v2
  - remove FC and use **anchor boxes**
  - shrink network to operate on 416 input images instead of 448
  - convolutional layers: down-sample the image by a factor of 32 from 416x416 to 13x13
  - Because of anchor boxes, accuracy is slightly decrease
    - 98 boxes → 9000 boxes
    - **69.5% mAP & recall 81% → 69.2% mAP & recall 88%**

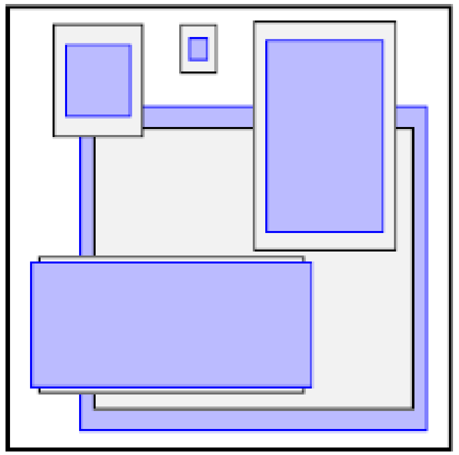
Better – Dimension Clusters

- Anchor boxes: box dimensions are hand picked → prior information!



- k-means clustering: k =5





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

Better – Direct location prediction

- Anchor boxes: model instability because of predicting the (x,y) locations for the box
- General region proposal networks:

$$x = (t_x * w_a) - x_a$$
$$y = (t_y * h_a) - y_a$$

For example, a prediction of $t_x = 1$ would shift the box to the right by the width of the anchor box, a prediction of $t_x = -1$ would shift it to the left by the same amount.

- YOLO v2: predict location coordinates relative to the location of the grid cell (offset X)

The network predicts 5 bounding boxes at each cell in the output feature map. The network predicts 5 coordinates for each bounding box, t_x, t_y, t_w, t_h, and t_o. If the cell is offset from the top left corner of the image by (c_x, c_y) and the bounding box prior has width and height p_w, p_h, then the predictions correspond to:

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$
$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$

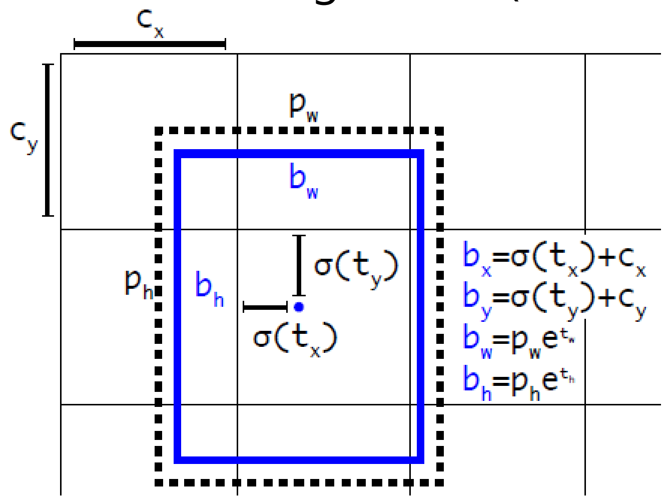
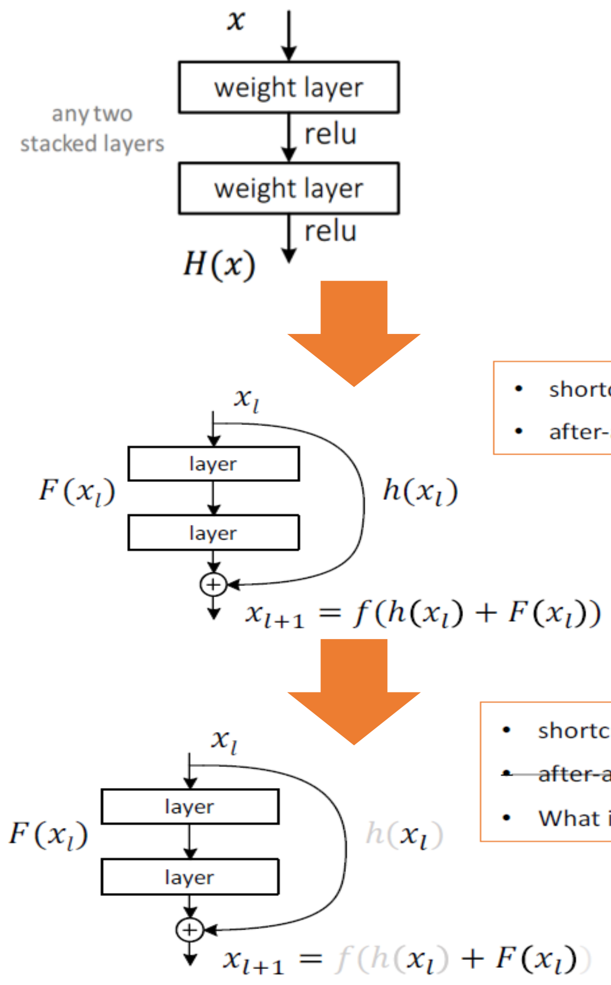


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

- **5% improvement in detection**

Better – Fine-Grained Features

- Simply add a **passthrough layer** which brings features from an earlier layer at 26x26 resolution
- 26x26x512 feature map → 13x13x2048 feature map
- **1% improvement**



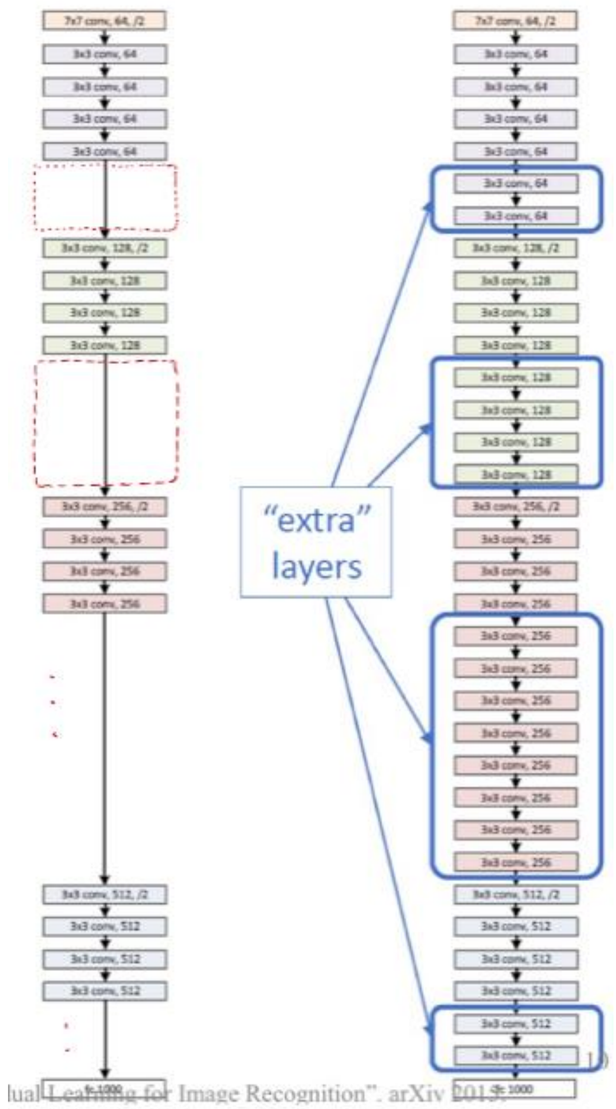
- shortcut mapping: h = identity
- after-add mapping: f = ReLU

- shortcut mapping: h = identity
- ~~after-add mapping: f = ReLU~~
- What if f = identity?

$$x_{l+1} = x_l + F(x_l) \Rightarrow x_{l+2} = x_{l+1} + F(x_{l+1})$$

$$x_{l+2} = x_l + F(x_l) + F(x_{l+1})$$

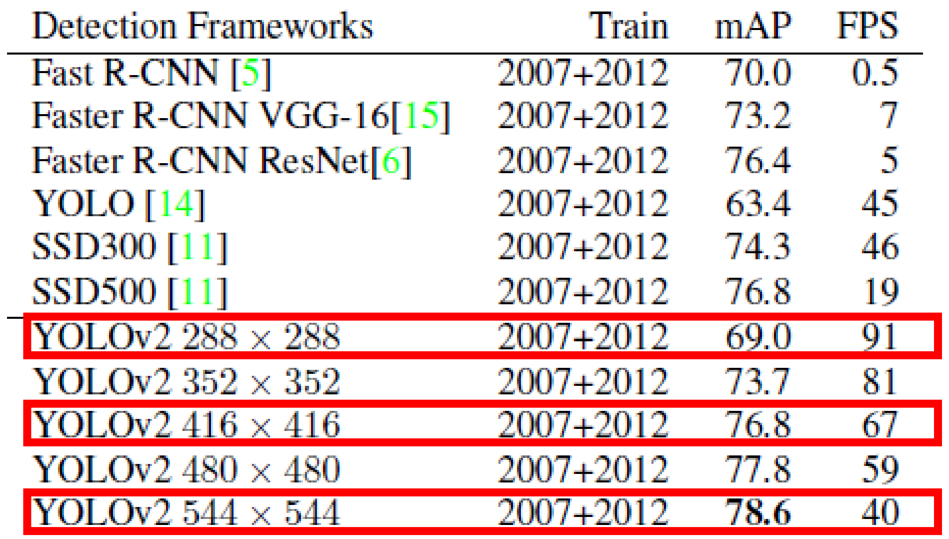
$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$



ual Learning for Image Recognition", arXiv 2015.

Better – Multi-scale Training

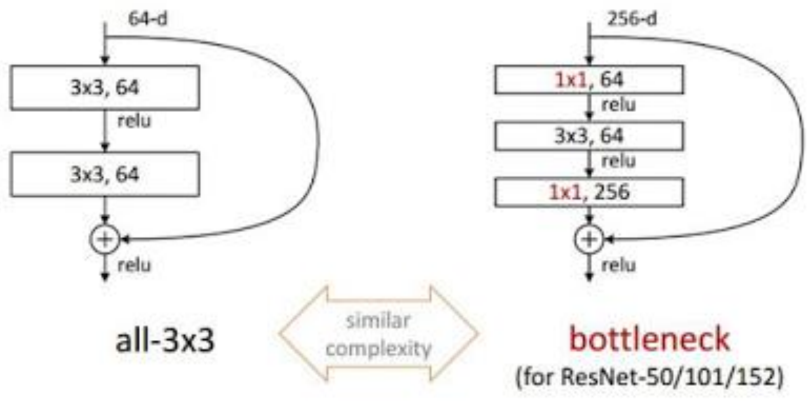
- Instead of fixing the input image size, change the network every few iterations
- Every 10 batches our network, randomly chooses a new image dimension size {320, 352, …, 608}



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

Faster – Darknet-19

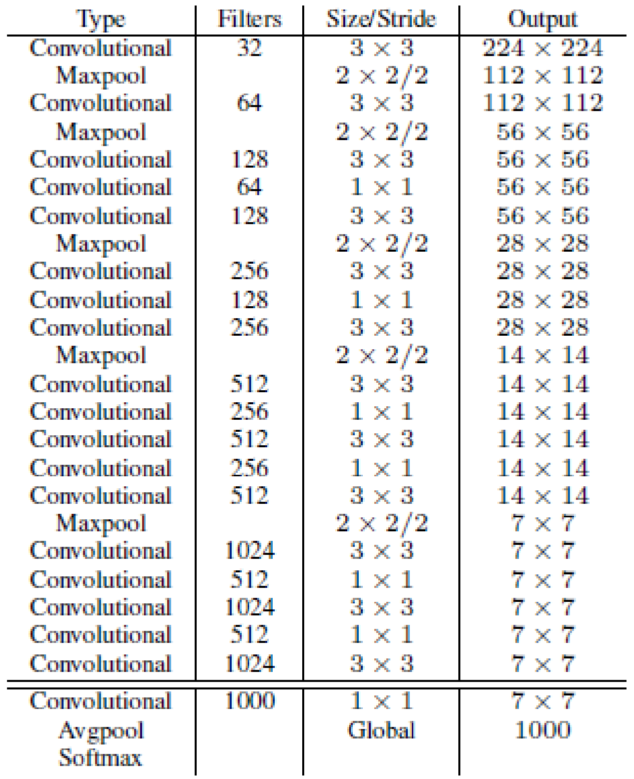
- Most detection frameworks rely on VGG-16 as the base feature extractor (ex. SSD)
  - 30.69 billion floating point operations at 224x224
- YOLO: custom network based on the Googlenet (faster than VGG-16)
  - 8.52 billion operations
- YOLO v2
  - mostly 3x3 filters (similar to VGG)
  - Following the work on Network in Network (NIN), use global average pooling to make predictions as well as 1x1 filters to compress the feature representation between 3x3 convolutions



3x3x64x64+3x3x64x64=73,728

1x1x256x64+3x3x64x64+1x1x256x64=69,632

  - Darknet-19: 19 convolutional layers & 5 maxpooling layers
  - **5.58 billion operations: 72.9% top-1 & 91.2% top-5 accuracy**



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			

Faster – Training

- Training for classification
  - ImangeNet 1000 class for 160 epochs
  - Standard data augmentation: random, crops, rotations, and hue, saturation, and exposure shifts
  - Initial training: 224x224 → 448x448 fine-tuning

- Training for detection
  - For VOC, predict 5 boxes with 5 coordinates each and 20 classes per box, so 125 filters
  - 160 epochs with a start learning rate of 10^-3 dividing it by 10 at 60 and 90 epochs

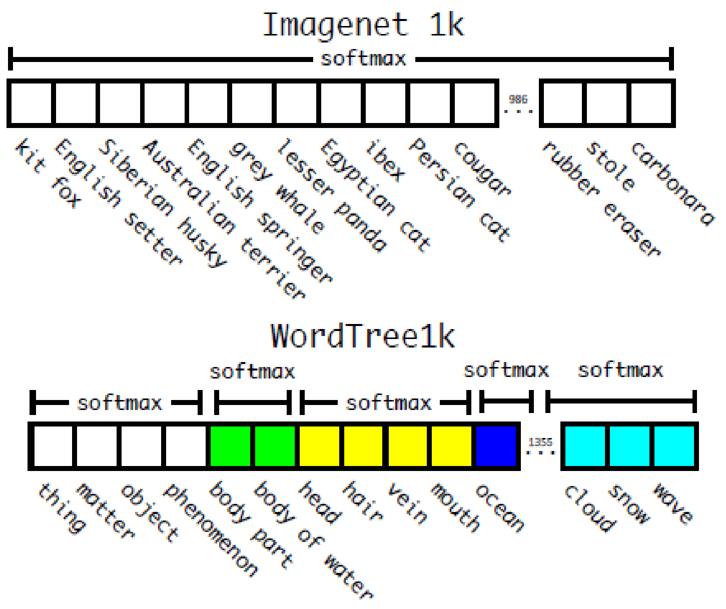
Stronger – Jointly training on classification and detection data

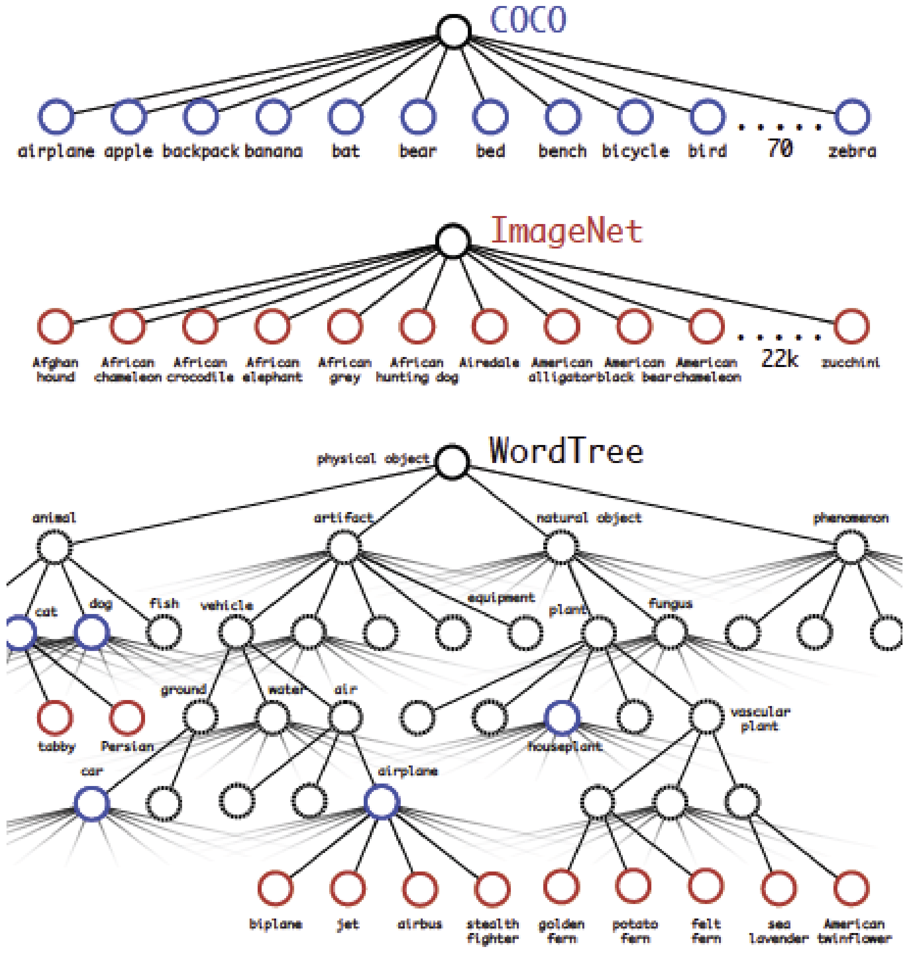
- WordTree: a hierarchical model of visual concepts

$$
\begin{aligned}
&Pr(\text{Norfolk terrier}|\text{terrier})\\
&Pr(\text{Yorkshire terrier}|\text{terrier})\\
&Pr(\text{Bedlington terrier}|\text{terrier})\\
&\ldots
\end{aligned}
$$

$$
\begin{aligned}
Pr(\text{Norfolk terrier}) = \; & Pr(\text{Norfolk terrier}|\text{terrier})\\
& *Pr(\text{terrier}|\text{hunting dog})\\
& *\ldots*\\
& *Pr(\text{mammal}|Pr(\text{animal})\\
& *Pr(\text{animal}|\text{physical object})
\end{aligned}
$$

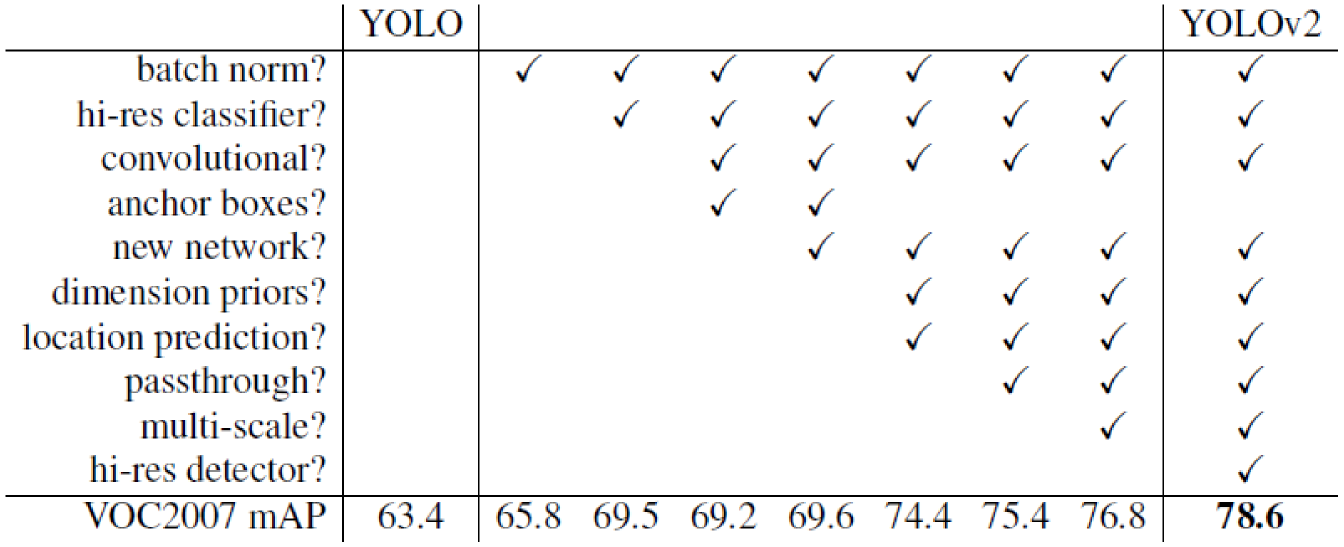
- Using the joint training, **YOLO 9000** learns to find objects in images using the detection data in COCO and it learns to classify a wide variety of these objects using data from ImageNet





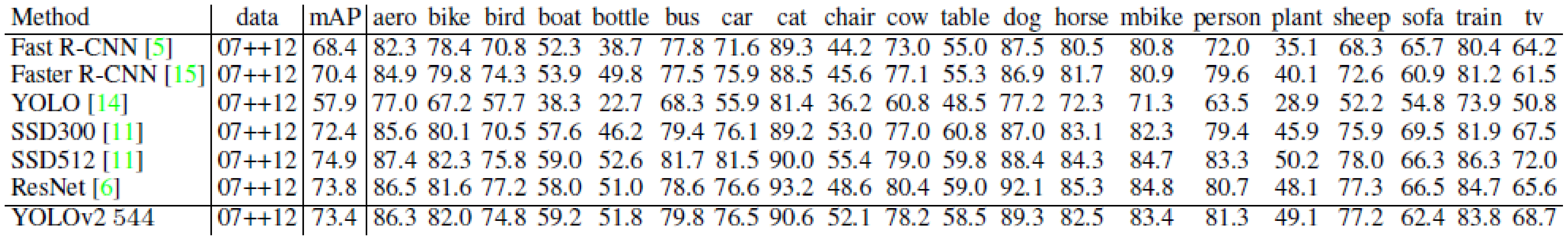
Conclusion

- YOLO v2 & YOLO 9000
  - State-of-the-art real-time detection systems
  - YOLO9000 is a real-time framework for detection more than 9000 object categories by jointly optimizing detection and classification using WordTree
- PASCAL VOC 2007



	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**

- PASCAL VOC 2012



Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN [5]	07++12	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
Faster R-CNN [15]	07++12	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
YOLO [14]	07++12	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
SSD300 [11]	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD512 [11]	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
ResNet [6]	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLOv2 544	07++12	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7