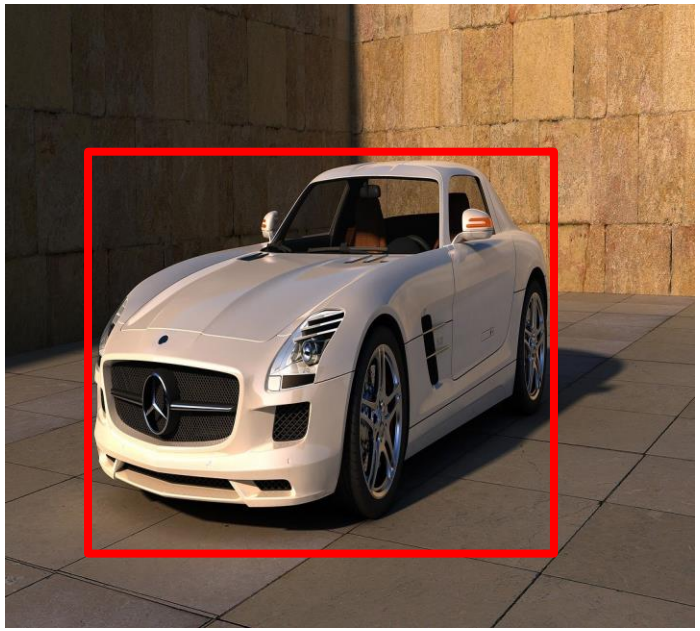
연구노트

2022.01.19

홍진우

Bounding Box



$$Y = \begin{bmatrix} Pc \\ bx \\ by \\ bw \\ bh \\ C1 \\ C2 \\ C3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.4 \\ 0.6 \\ 0.7 \\ 0.6 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

1 – car
2 – dog
3 - cat

- Pc는 Bounding box안에 물체가 있는지
- bx, by, bw, bh 는 Bounding Box에 대한 정보(전체 이미지에 대한 상대값)
- C1, C2, C3 는 예측할 클래스에 대한 정보
- Pc가 0이라면 아래 정보들은 처리하지 않음

YOLOv1

Unified Detection

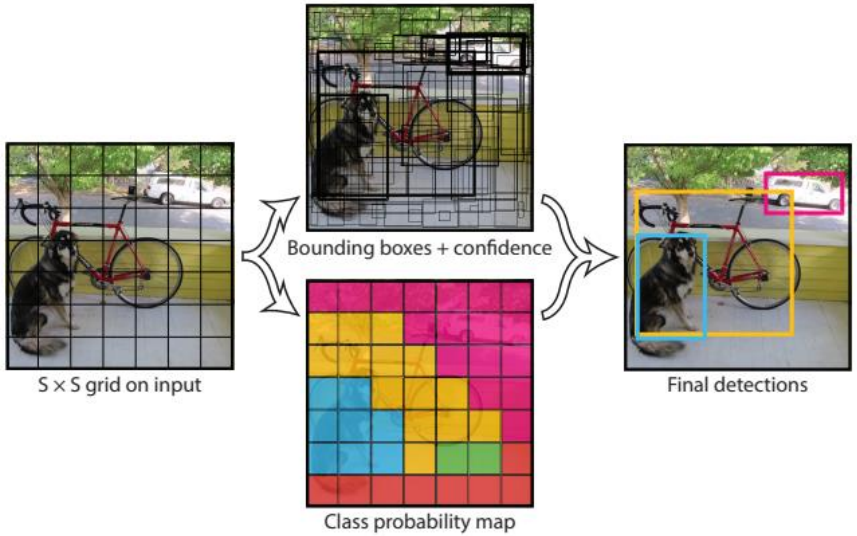


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

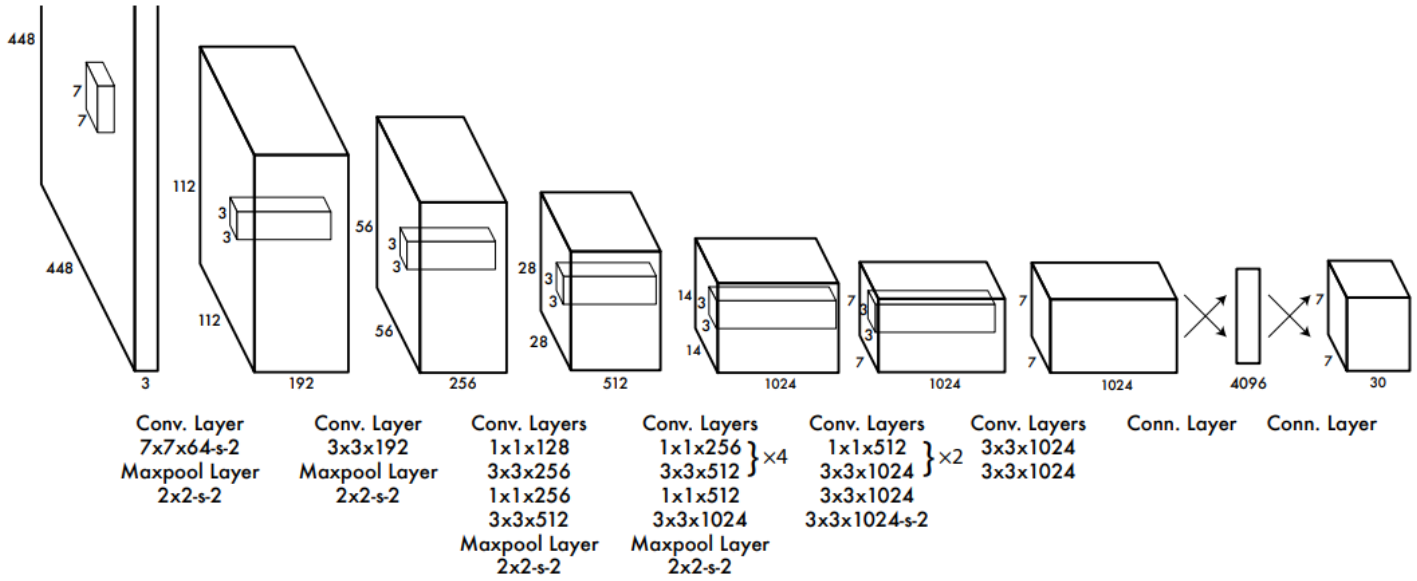
YOLO 성능 평가를 위해 PASCAL VOC를 사용했다.
S = 7, B = 2, C = 20

- Input Image를 S x S grid로 나눈다.
- 각각의 grid cell이 B개의 bounding box와 각 bounding box에 대한 confidence score를 갖는다. $\Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}}$
- 각각의 grid cell은 C개의 conditional class probabilities를 갖는다. $\Pr(\text{Class}_i|\text{Object})$
- 각 bounding box는 x, y, w, h, confidence로 구성된다.
  - x, y는 grid cell 범위에 대한 상대값
  - w, h는 전체 이미지에 대한 상대값
- Test time에는 confidence score 와 conditional class probabilities를 곱하여 class-specific confidence score를 얻는다.

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

YOLOv1

Network Design



- Network architecture는 GoogLeNet model for image classification을 기반으로 한다.
- 24개의 Convolution layers와 2개의 FC Layers로 구성.

YOLOv1

Training

loss function:

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

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

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

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

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

- 여러 bounding box들 중에서 ground-truth와 IOU가 가장 높은 bounding box를 predictor로 정한다.
- S^2 : grid cell의 수
- B : grid cell 별 bounding box의 수

$\mathbb{1}_{ij}^{\text{obj}}$: Object가 존재하는 grid cell i의 predictor bounding box j

$\mathbb{1}_{ij}^{\text{noobj}}$: Object가 존재하지 않는 grid cell i의 bounding box j

$\mathbb{1}_{i}^{\text{obj}}$: Object가 존재하는 grid cell i

YOLOv1

Limitation of YOLO

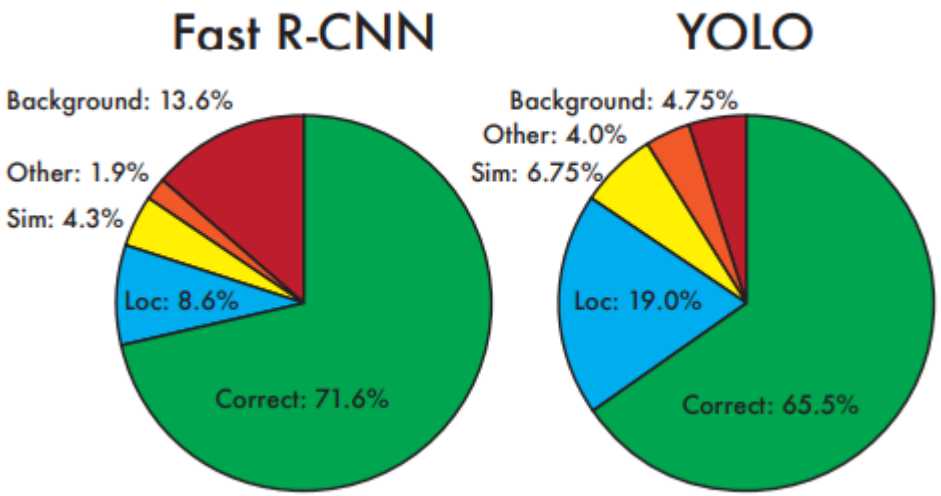
- 각각의 Grid cell이 하나의 클래스만 예측할 수 있다.

- Bounding box 형태가 training data를 통해서만 학습이 되어 새로운 가로•세로 비율에 대한 bounding box에 대해서 정확히 예측하지 못한다.

- 작은 Bounding box의 작은 오류가 IOU에 훨씬 더 큰 영향을 미쳐 localization 오류가 있다.

YOLOv1

Experiments



- Correct: correct class and IOU $> .5$
- Localization: correct class, $.1 <$ IOU $< .5$
- Similar: class is similar, IOU $> .1$
- Other: class is wrong, IOU $> .1$
- Background: IOU $< .1$ for any object

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

	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**

YOLOv1

Experiments

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

Table 3: PASCAL VOC 2012 Leaderboard. YOLO compared with the full `comp4` (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

YOLOv2

Better

Batch Normalization

- 모든 convolution layer 뒤에 batch Normalization을 추가
- mAP 값이 2% 정도 향상
- Regularization 도움이 되었고 Overfitting 없이 dropout 제거할 수 있었음

High Resolution Classifier

- YOLOv1에서는 Darknet을 224x224 크기로 pre-train 시켰지만 detection task 에서는 448x448 크기의 이미지 입력으로 사용
- Darknet을 448x448 크기로 pre-train시켜 네트워크가 상대적으로 높은 해상도의 이미지에 적응할 시간을 제공
- mAP 값이 4% 정도 향상

YOLOv2

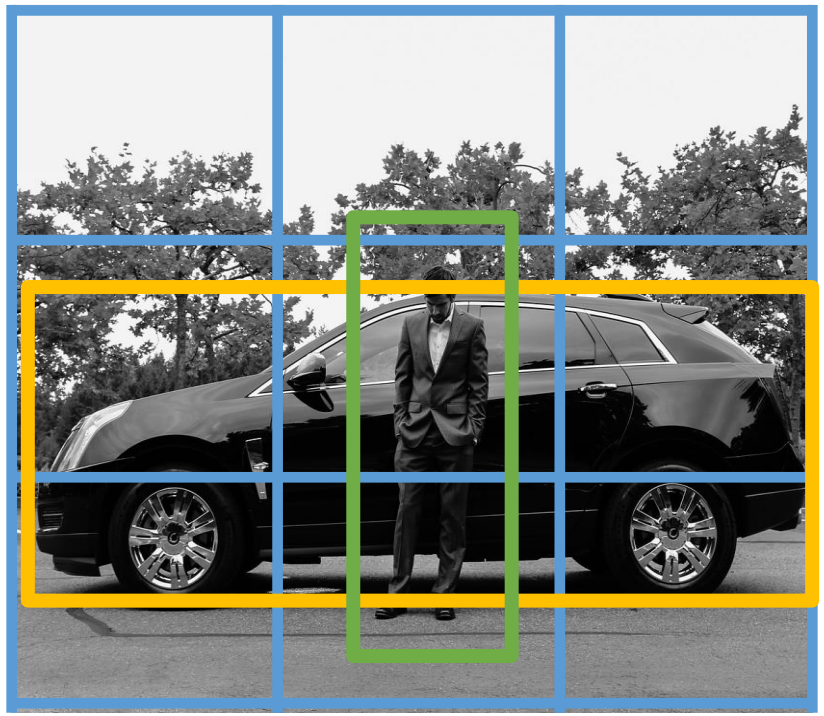
Better

Convolutional with Anchor boxes

- 좌표 대신 offset을 예측하는 문제가 보다 단순하고 네트워크가 학습하기 쉬움
- Output이 높은 resolution을 가지도록 pooling layer를 제거
- 416x416 크기의 입력 이미지를 사용해 output feature map의 크기가 홀수가 되도록 해 feature map 내에 하나의 중심 cell이 존재할 수 있도록 함
- YOLOv2에서는 anchor box를 사용해 보다 많은 수의 bounding box를 예측
- Anchor box를 사용하지 않은 경우 mAP 값이 69.5%, recall 값은 81%
- Anchor box를 사용한 경우 mAP 값이 69.2%, recall 값은 88%
- Anchor box를 사용하면 mAP 값이 감소하지만 recall 값이 상승하고 모델이 더 향상될 여지가 있음을 나타냄

- **Object detection task에서 recall 값이 높다는 것은 모델이 실제 객체의 위치를 예측한 비율이 높음을 의미**

Anchor Boxes

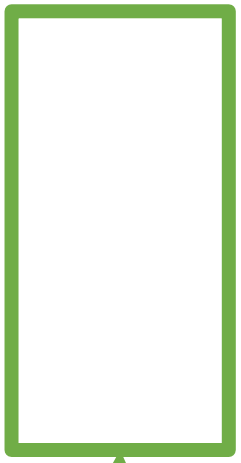


1 – car
2 – person
3 - cat

Y =

Anchor box 1:

Anchor box 2:



Pc
bx
by
bw
bh
C1
C2
C3
Pc
bx
by
bw
bh
C1
C2
C3

- 하나의 Grid Cell에서 여러 물체를 탐지하고 싶을 때 사용
- 학습 알고리즘을 전문화 시킬 수 있음

YOLOv2

Better

Dimension Clusters

- 기존에는 anchor box의 크기와 aspect ratio를 사전에 미리 정의했음.
- K-means clusterin을 통해 최적의 값을 찾음.

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

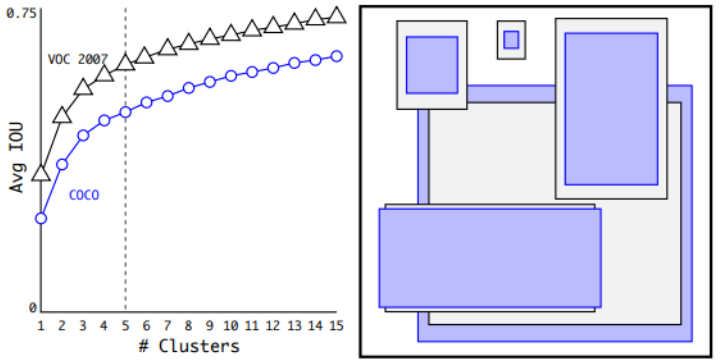


Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. We find that $k = 5$ gives a good tradeoff for recall vs. complexity of the model. The right image shows the relative centroids for VOC and COCO. Both sets of priors favor thinner, taller boxes while COCO has greater variation in size than VOC.

YOLOv2

Better

Direct location prediction

- YOLO와 anchor box를 같이 사용했을 때 초기 iteration시 모델이 불안정
- Network는 5개의 bounding boxes를 예측함.
- tx, ty, tw, th, to를 학습을 통해 예측하고 우측 식과 같은 방식을 적용하여 bounding box를 구함
- Anchor 중심부 좌표를 직접 예측함으로써 mAP 값이 5% 향상

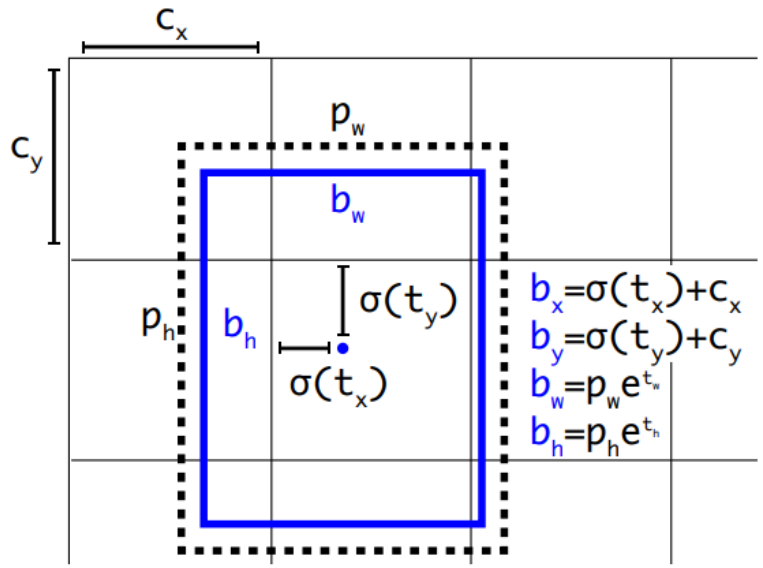


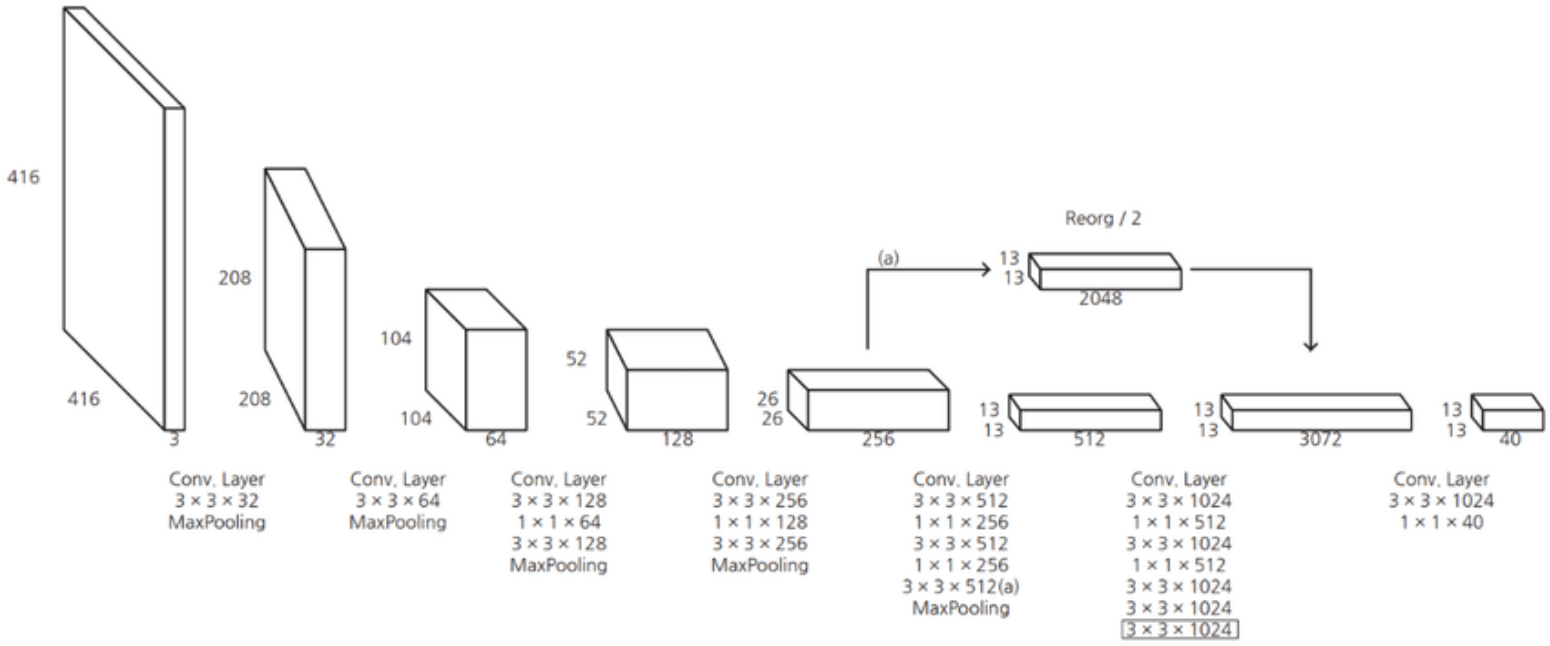
Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

YOLOv2

Better

Fine-Grained Features

- YOLOv2는 13x13 크기의 feature map을 출력함
- Feature map의 크기가 작은 경우 큰 객체를 예측하기 쉽지만 작은 객체는 예측하기 어려움
- 마지막 pooling을 수행하기 전에 feature map을 추출하여 26x26x512 크기의 feature map을 얻고 4개로 분할 후 concat하여 13x13x2048 크기의 feature map을 얻음.(작은 객체에 대한 정보 함축)



YOLOv2

Better

Multi-Scale Training

- 모델을 강건하게 하기 위해 다양한 입력 이미지를 사용하여 네트워크를 학습시킴
- 10 batch마다 입력 이미지의 크기를 랜덤하게 학습하도록 설계
- 이미지 크기를 32배수:{320, 352,…,608} 중에서 선택하도록 함

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

YOLOv2

Faster

Darknet-19

- Darknet-19라는 독자적인 classification 모델을 backbone network로 사용
- 기존 YOLOv1의 마지막 fc layer를 제거하고 global average pooling을 사용
- Parameter 수를 감소시키고 detection 속도를 향상

Training for classification

- Class 수가 1000개인 ImageNet을 학습시킴
- top-1 정확도는 76.5%, top-5 정확도는 93.3%

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	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		$2 \times 2/2$	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.

YOLOv2

Stronger

- YOLOv2를 classification 데이터와 detection 데이터를 함께 사용하여 학습시켜 보다 많은 class를 예측하는 YOLO9000
- 이 두 데이터를 섞어 학습시킬 때 detection 데이터셋은 모든 개 이미지를 “개 " 라는 하나의 class로 분류
- Classification 데이터셋은 “요크셔 테리어”, “불독 " 등 개를 종류별로 세부적인 class로 분류

YOLOv2

Stronger

Hierachical classification

- ImageNet label로 부터 계층적인 트리인 WordTree를 구성하는 방법을 제안
- WordTree에서 각 노드는 범주를 의미하고 하위 범주는 자식 노드가 되는 구조
- 이러한 트리에서 특정 범주에 속할 확률을 루트 노드로부터 해당 범주의 노드까지의 조건부 확률의 곱으로 표현 가능

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

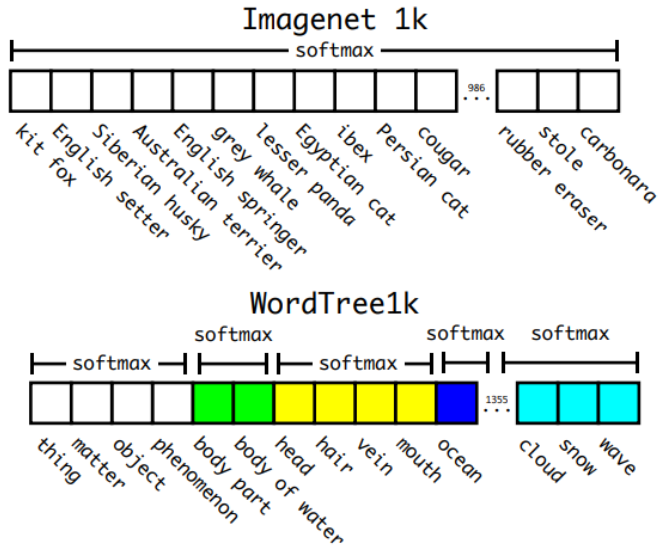


Figure 5: Prediction on ImageNet vs WordTree. Most ImageNet models use one large softmax to predict a probability distribution. Using WordTree we perform multiple softmax operations over co-hyponyms.

YOLOv2

Stronger

Joint classification and detection

- COCO 데이터셋과 ImageNet 데이터 셋을 합쳐 9418개의 범주를 가지는 WordTree를 구성
- ImageNet과 COCO 데이터셋의 비율이 4:1이 되도록 조정
- Classification loss의 경우 특정 범주와 상위 범주에 대해서만 loss 계산
- Ground truth box와 IOU값이 0.3이상인 경우에만 역전파 수행

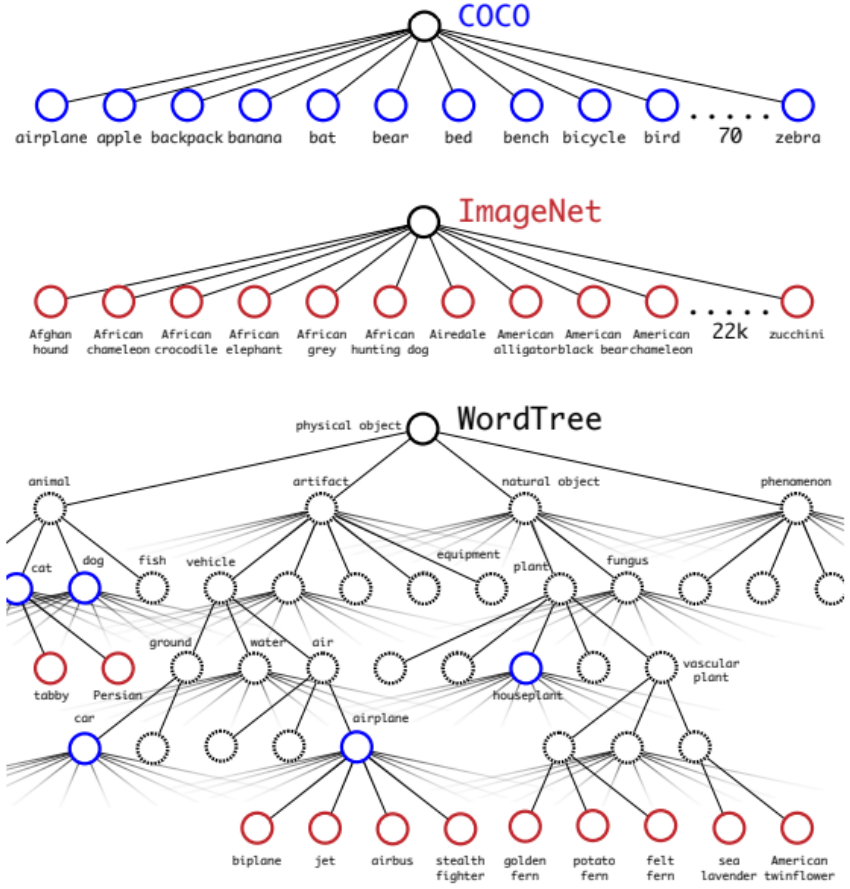


Figure 6: Combining datasets using WordTree hierarchy. Using the WordNet concept graph we build a hierarchical tree of visual concepts. Then we can merge datasets together by mapping the classes in the dataset to synsets in the tree. This is a simplified view of WordTree for illustration purposes.

YOLOv2

Conclusion

	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**

YOLOv3

Darknet-53

- Backbone network를 Darknet-53으로 변경
- Maxpooling 대신 convolution의 stride를 2로 주어 feature map resolution을 줄여나감
- Skip connection을 활요해 residual 값을 전달

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

Table 2. **Comparison of backbones.** Accuracy, billions of operations, billion floating point operations per second, and FPS for various networks.

	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.**

YOLOv3

Class Prediction

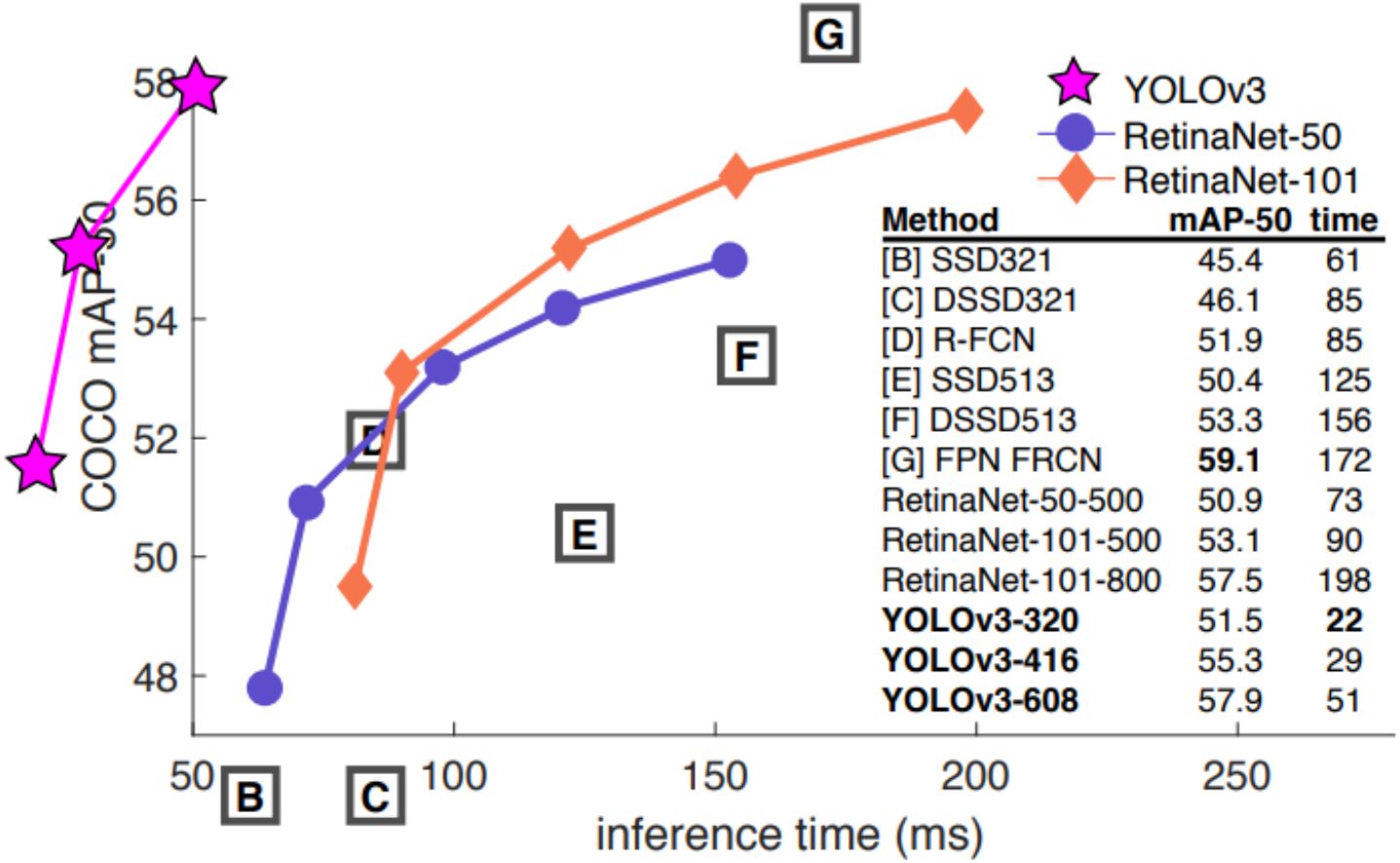
- 각각의 bounding box는 multi-label classification을 수행
- Softmax 함수를 사용하여 class를 예측할 경우 성능면에서 좋지 않아 binary cross-entropy를 사용

Prediction across Scales

- YOLO v3는 서로 다른 3개의 scale을 사용하여 최종 결과를 예측
- Feature pyramid network와 유사
- 3개의 box를 예측하며 N x N x [3*(4+1+80)] 텐서의 크기를 갖는다.
- COCO dataset에서 K-means clustering을 통해 9개의 Anchor box 크기를 정의

  (10 x 13), (16 x 30), (33 x 23), (30 x 61), (62 x 45), (59 x 119), (116 x 90), (156 x 198), (373 x 326)

YOLOv3



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