



# Object detection

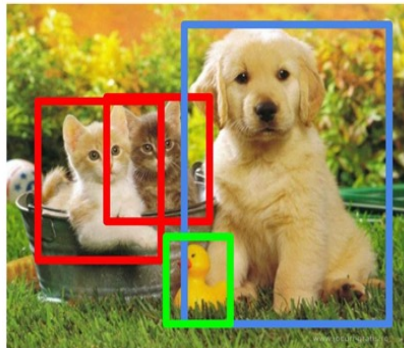
---

Sang Yup Lee

# Object detection

- Object detection (물체 탐지 또는 객체 탐지)
  - 여러 개의 사물을 담고 있는 이미지에 대해서 각 사물이 무엇이고, 각 이미지 내에서 각 사물의 위치가 어떻게 되는지를 맞추는 것

**Object Detection**



CAT, DOG, DUCK

vs.

**Classification**

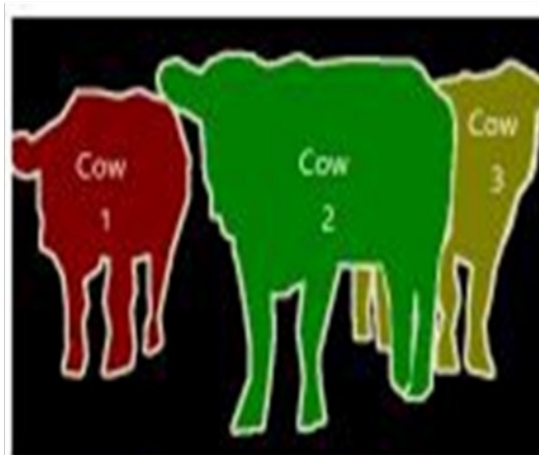


CAT

# 참고: detection vs. segmentation



Object detection



Instance segmentation



Semantic segmentation



# Object detection

---

- Object detection (cont'd)
  - 두 가지 태스크를 포함
    - Localization: 물체의 위치를 찾는 것
      - 정확하게는 물체가 있을 것 같은 경계상자를 찾는 것
    - Classification: 해당 상자에 포함되어 있는 물체가 무엇인지 맞히는 것
  - Localization과 classification을 수행하기 위해서 일반적으로 두 단계 필요
    - 물체가 있을 만한 경계상자 (이러한 경계상자를 일컬어 regions of interest, ROI라고 함)를 추출
    - 추출된 ROI을 이용하여 localization과 classification을 수행

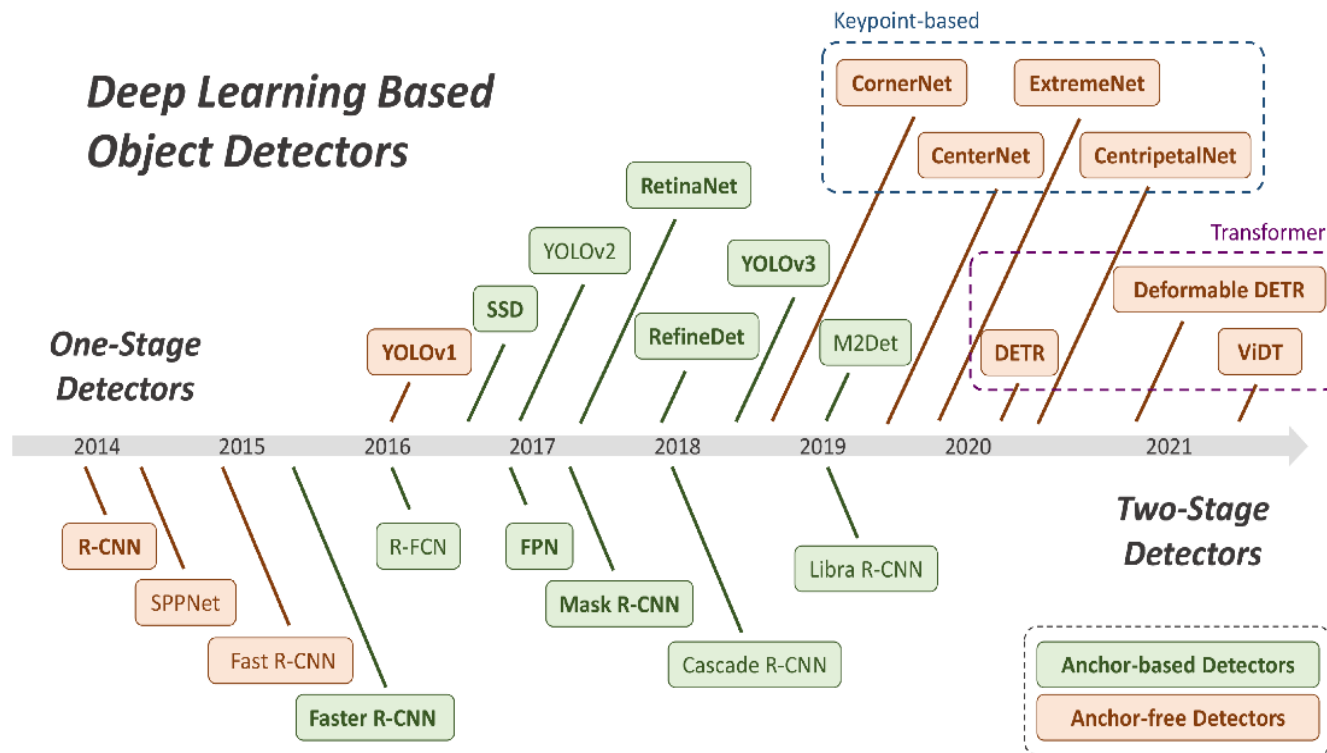


# Object detection

---

- Object detection (cont'd)
  - 두 가지 종류의 object detection 알고리즘들
    - One stage detectors
      - SSD, YOLO 등
    - Two stage detectors
      - 앞의 두 단계를 구분해서 수행
      - R-CNN family
      - 속도가 느리다는 단점

# Object detection



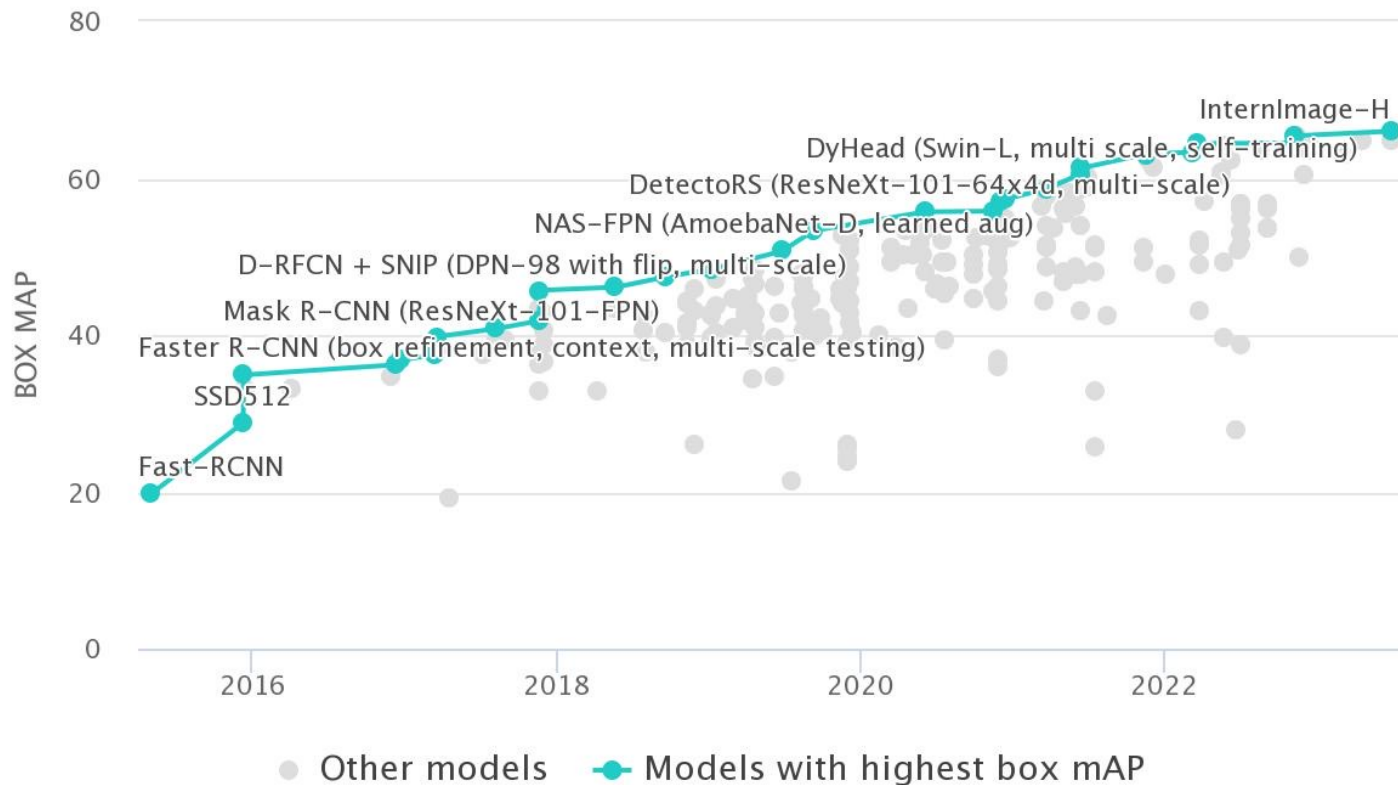
Anchor-free 소개:

<https://blog.si-analytics.ai/72>

<https://learnopencv.com/fcos-anchor-free-object-detection-explained/>

# Object detection

## ■ Primary models on MS COCO datasets





# Object detection

---

- Before 2014 (참고)
  - Viola-Jones in 2001
    - Viola, P., & Jones, M. (2001, December). Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001 (Vol. 1, pp. I-I). IEEE.
  - HOG detector in 2005
    - Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) (Vol. 1, pp. 886-893). IEEE.
  - Deformable Parts Model (DPM) in 2008
    - Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008, June). A discriminatively trained, multiscale, deformable part model. In 2008 IEEE conference on computer vision and pattern recognition (pp. 1-8). IEEE.





# Object detection

---

- 학습 데이터
  - PASCAL Visual Object Classes (VOC)
    - <http://host.robots.ox.ac.uk/pascal/VOC/>
    - 다운로드: [http://host.robots.ox.ac.uk/pascal/VOC/voc2012/VOCtrainval\\_11-May-2012.tar](http://host.robots.ox.ac.uk/pascal/VOC/voc2012/VOCtrainval_11-May-2012.tar)
    - Number of classes: 20
  - COCO (Common Objects in Context)
    - <https://cocodataset.org/#home>
    - 다운로드: <https://cocodataset.org/#download>
    - Number of classes: 80 (90개 중 80개만 사용)
  - Google Open Images
    - <https://storage.googleapis.com/openimages/web/index.html>
    - Number of classes: 600 +



# Object detection

---

- PASCAL VOC format
  - XML files (one file for each image)

```
<annotation>
  <folder>Kangaroo</folder>
  <filename>00001.jpg</filename>
  <path>./Kangaroo/stock-12.jpg</path>
  <source>
    <database>Kangaroo</database>
  </source>
  <size>
    <width>450</width>
    <height>319</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>kangaroo</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>233</xmin>
      <ymin>89</ymin>
      <xmax>386</xmax>
      <ymax>262</ymax>
    </bndbox>
  </object>
</annotation>
```

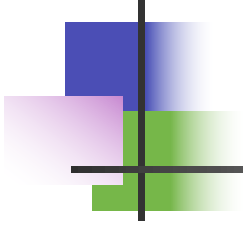
# Object detection

- MS COCO format
  - json file

COCO Bounding box: (x-top left, y-top left, width, height)



```
"annotations": [  
  {  
    "id": 125686,  
    "category_id": 2,  
    "iscrowd": 0,  
    "segmentation": [[164.81, 417.51, 164.81,  
    "image_id": 242287,  
    "area": 42061.80340000001,  
    "bbox": [19.23, 383.18, 314.5, 244.46]  
  },  
  {  
    "id": 1488610
```



# SSD (Single shot detection)

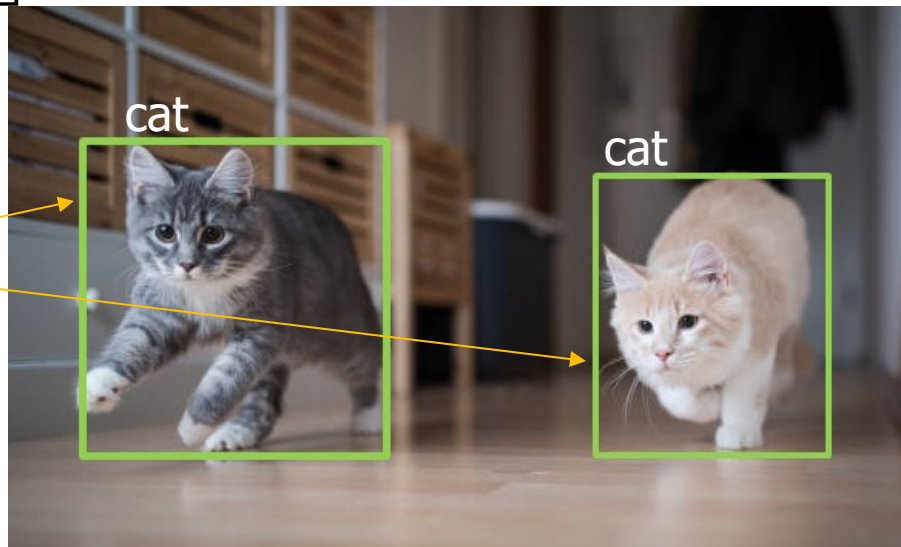
# SSD

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

## ■ SSD 소개

- 지도학습의 한 종류
- 다음과 같은 정답 이미지 존재
- SSD를 통해서 GTBB를 찾고, 각 GTBB에 속한 object를 예측해야 함

정답 경계상자  
(Ground Truth  
Bounding Box,  
GTBB)



# SSD

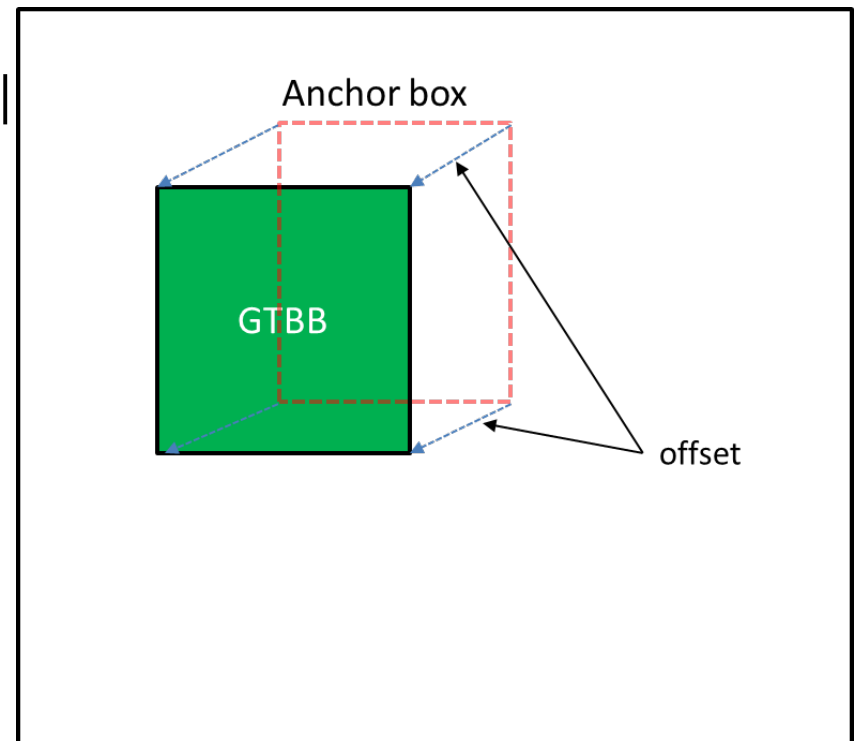
- Anchor box 사용

- 그러면 우리는 어떻게 다음과 같은 이미지가 입력되었을 때 정답 경계상자를 찾을 수 있을까요?
- 아무것도 없는 상태에서 GTBB를 예측하는 것은 어려움
- SSD는 이를 위해서 기본상자를 사용  $\Rightarrow$  이를 anchor box라고 함



# SSD

- Anchor box (AB)
  - 다양한 위치, 크기, 형태의 anchor box 사용
  - SSD는 이러한 AB를 기준으로 AB로부터 GTBB가 얼마나 떨어져 있는지 (즉, offset 정도) 그리고 해당 GTBB에 존재하는 object가 무엇인지를 예측





# SSD

---

- Anchor box (cont'd)
  - 그렇다면 SSD는 anchor box를 어떻게 혹은 몇 개를 설정할까요?
  - 원본 이미지를 그대로 사용하지 않고, VGG16과 같은 사전 학습 모형을 통해 추출된 feature map을 사용
  - 도출되는 feature map을 사용해서 크기와 형태가 다른 여러 개의 anchor box를 생성  $\Rightarrow$  anchor box를 이용하여 원본 이미지에 존재하는 GTBB를 찾는 방식으로 작동





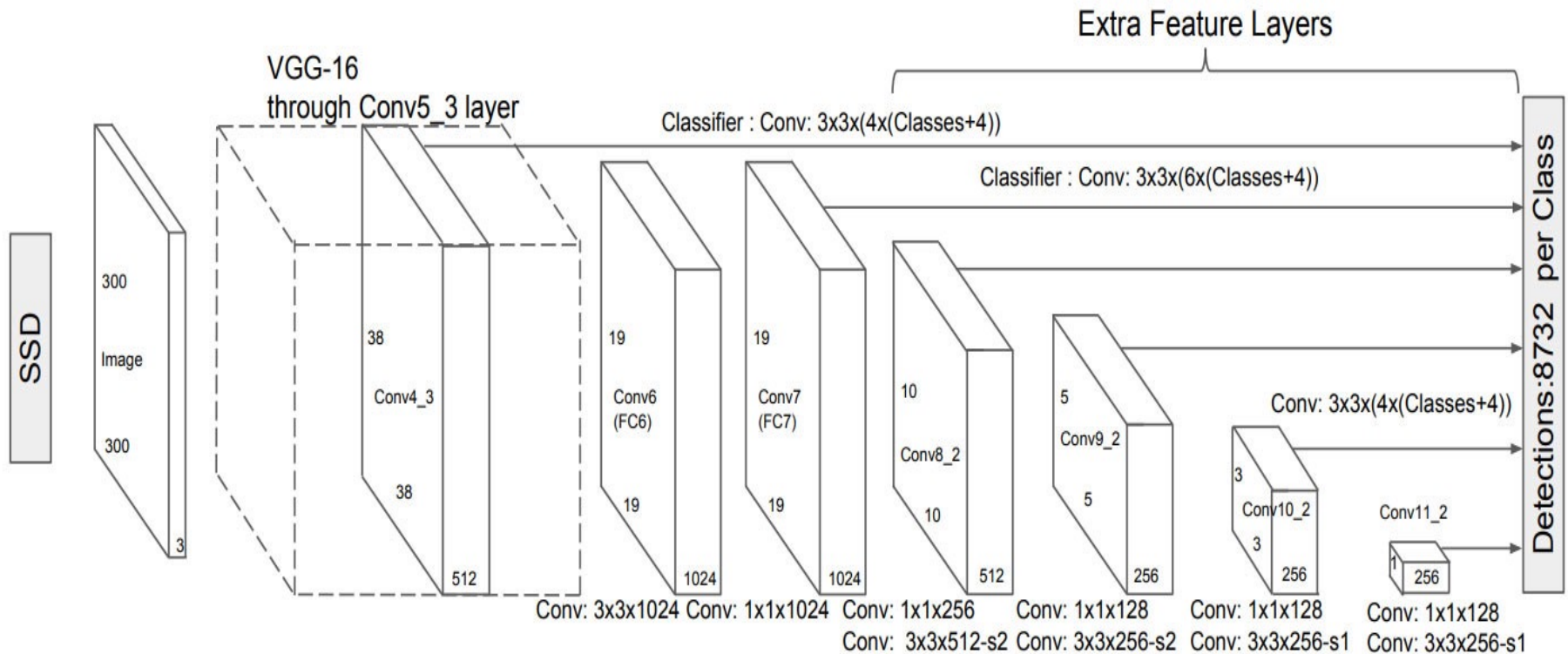
# SSD

---

- Feature map 추출
  - SSD의 경우, object detection을 위해서 사용하는 feature map의 수 6개
    - 더 많은 feature map이 도출되지만 그 중 특정 6개만 사용
  - 1차적으로 VGG16과 같은 사전학습모형 (backbone 네트워크라고 함)를 사용해서 feature map 추출, 이후 추가적인 합성곱 필터를 사용하여 6개의 feature map 추출
  - VGG16이 추출한 feature map 중 1개 사용 + 추가 convolutional layer가 추출한 6개 중 5개 사용
    - Feature map 마다의 크기가 다름 (receptive field의 크기가 다름)
  - multi-scale object detection
    - 크기가 다른 여러 개의 feature map을 사용하여 물체 탐지
    - 다양한 크기의 물체를 잘 찾을 수 있음

# SSD

## ■ Feature map 추출 (cont'd)



# SSD

- 각 feature map 에서 사용되는 anchor box의 수 상이

The number of anchor boxes in a cell

conv4_3	→	38x38x(4x(Classes+4))	=	5776x(Classes+4)
conv7	→	19x19x(6x(Classes+4))	=	2166x(Classes+4)
conv8_2	→	10x10x(6x(Classes+4))	=	600x(Classes+4)
conv9_2	→	5x5x(6x(Classes+4))	=	150x(Classes+4)
conv10_2	→	3x3x(4x(Classes+4))	=	36x(Classes+4)
conv11_2	→	1x1x(4x(Classes+4))	=	4x(Classes+4)

+ = 8732x(Classes+4)

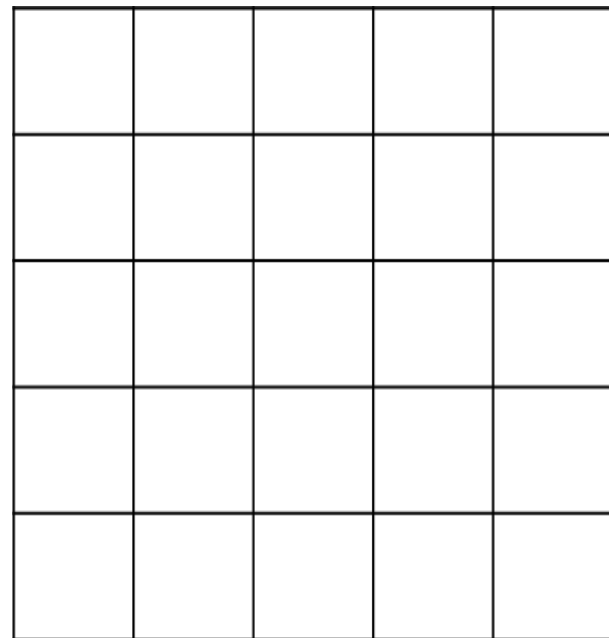
Anchor boxes 좌표정보



# SSD

---

- Feature map에 anchor box들 설정하기
  - 각 feature map를 이용해서 object detection을 하기 위해 각 feature map에 대해서 여러 개의 anchor box 설정
  - 하나의 feature map은 오른쪽과 같은 격자판 (grid)라고 생각할 수 있음 (depth 생략)
  - 설명을 위해 가로x세로가 5x5인 feature map을 예로 사용
    - 25개의 cell 존재





# SSD

- Feature map에 anchor box들 설정하기 (cont'd)
  - SSD는 feature map에 존재하는 각 cell에 4개 또는 6개의 anchor box를 설정
  - 6개의 feature map 중 두번째~네번째는 6개 그외는 4개를 사용
  - 그렇다면 각 셀에 대한 anchor box는 어떻게 설정되는가?
    - 보다 다양한 형태의 물체를 찾기 위해 다양한 형태의 anchor box 사용 (즉, 가로와 세로의 비율이 다른 여러가지 형태의 anchor box)
    - j번째 feature map의 특정 셀에 존재하는 i번째 anchor box의 가로와 세로의 길이는 다음과 같이 결정
      - $(w_i, h_i) = \left( s_j \sqrt{a_i}, s_j \frac{1}{\sqrt{a_i}} \right)$ 
        - $s_j$ 는 j번째 feature map의 scaling factor (anchor box의 상대적인 크기를 결정하는 역할)
        - $a_i$ 는 i번째 anchor box의 aspect ratio (즉, 가로세로 비율)

# SSD

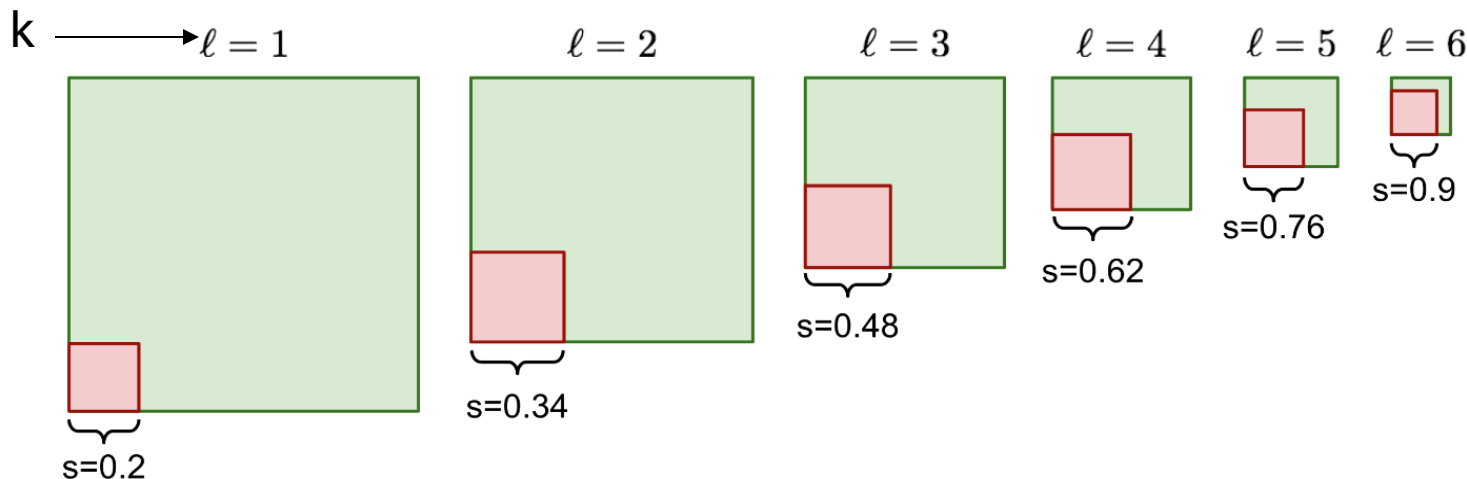
## ■ Feature map에 anchor box들 설정하기 (cont'd)

### ■ Scaling factor

- k번째 feature map에 대한 scaling factor, 이는 feature map에 대한 상대적 크기

- $s_k = s_{min} + \frac{s_{max} - s_{min}}{m-1}(k-1), k \in [1, m]$

- $s_{min} = 0.2, s_{max} = 0.9$





# SSD

---

- Feature map에 anchor box들 설정하기 (cont'd)
  - Aspect ratio
    - cell 당 anchor box가 6개인 경우
      - $a_i \in \{1, 2, 3, 1/2, 1/3\}$
      - $a_i = 1$  경우, 추가 scaling factor 사용:  $s'_j = \sqrt{s_j s_{j+1}}$  (하나의 anchor box 추가)
    - cell 당 anchor box가 4개인 경우
      - $a_i \in \{1, 2, 1/2\}$
      - $a_i = 1$  경우, 추가 scaling factor 사용:  $s'_j = \sqrt{s_j s_{j+1}}$



# SSD

---

- Example: 4번째 feature map (즉, 5x5 feature map)의 경우, 즉,  $k=4$ 
  - $s_4 = 0.2 + \frac{0.9-0.2}{5}(4-1) = 0.62$
  - 4번째 feature map의 경우는 6개의 anchor box를 생성하게 되나, 여기서는 설명을 간단하게 하기 위해서 4개만 생성한다고 가정
  - $a_i \in \{1, 2, 1/2\}$

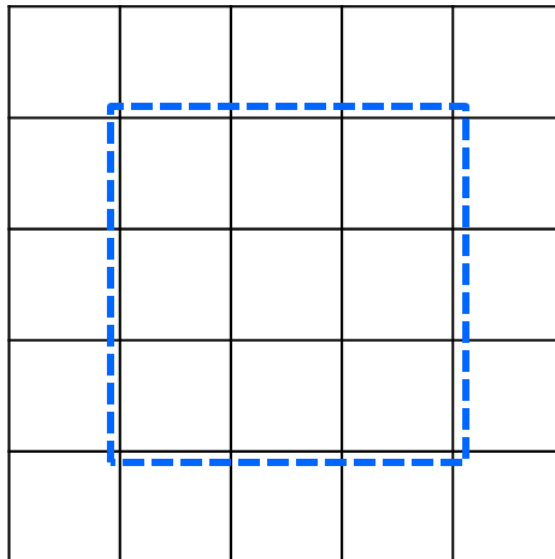


# SSD

- Example (cont'd)

- 첫 번째 anchor box:  $AR=1$

- $(w_0, h_0) = \left(0.62\sqrt{1}, 0.62\frac{1}{\sqrt{1}}\right) = (0.62, 0.62)$



# SSD

- Example (cont'd)

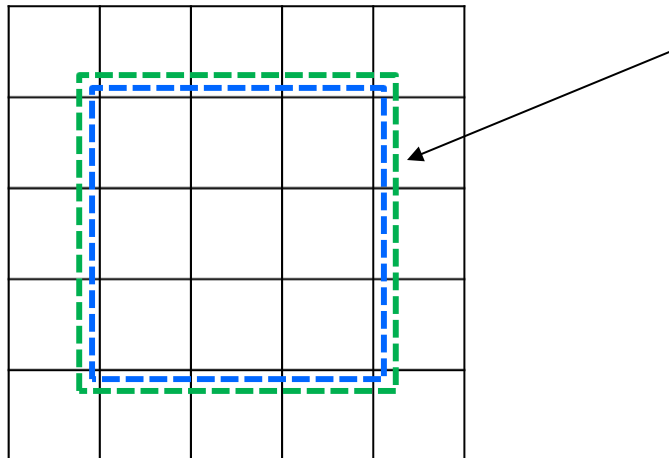
- 두 번째 anchor box: AR=1

- AR=1 인 경우에는 또 다른 scaling factor 값을 사용

$$s'_j = \sqrt{s_j s_{j+1}}, s_j = s_4 = 0.62, s_{j+1} = s_5 = 0.76$$

$$s'_4 = \sqrt{s_4 s_5} = \sqrt{0.62 * 0.76} \approx 0.69$$

- $(w_i, h_i) = (0.69\sqrt{1}, 0.69\frac{1}{\sqrt{1}}) = (0.69, 0.69)$



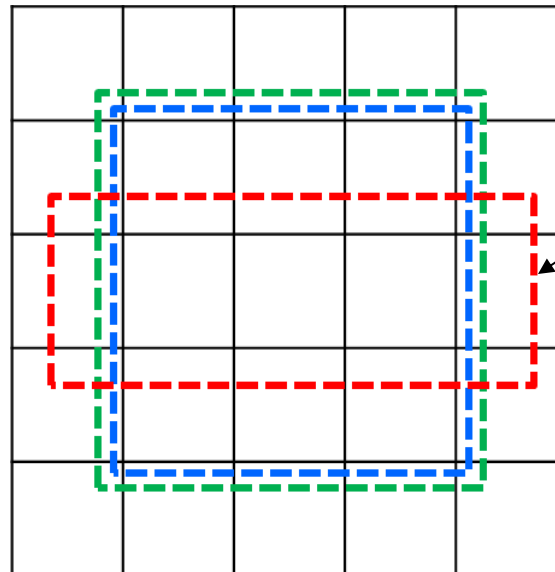
Object detection

# SSD

- Example (cont'd)

- 세 번째 anchor box: AR=2

$$(w_i, h_i) = \left( s_j \sqrt{a_i}, s_j \frac{1}{\sqrt{a_i}} \right) = \left( 0.62 * \sqrt{2}, 0.62 * \frac{1}{\sqrt{2}} \right) \approx (0.88, 0.44)$$



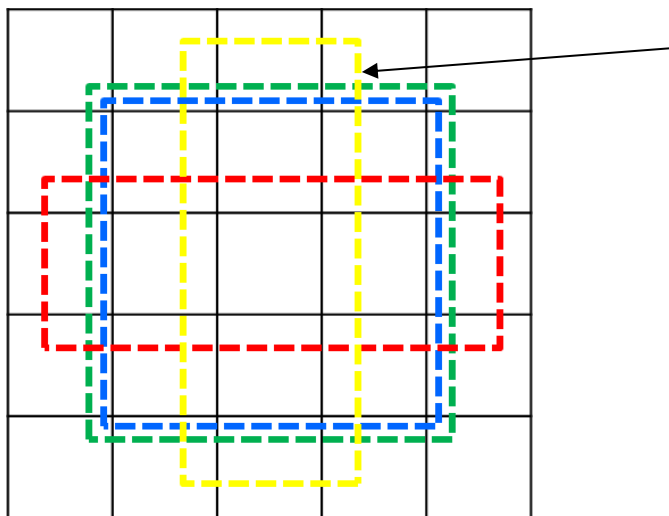
Object detection

# SSD

## ■ Example (cont'd)

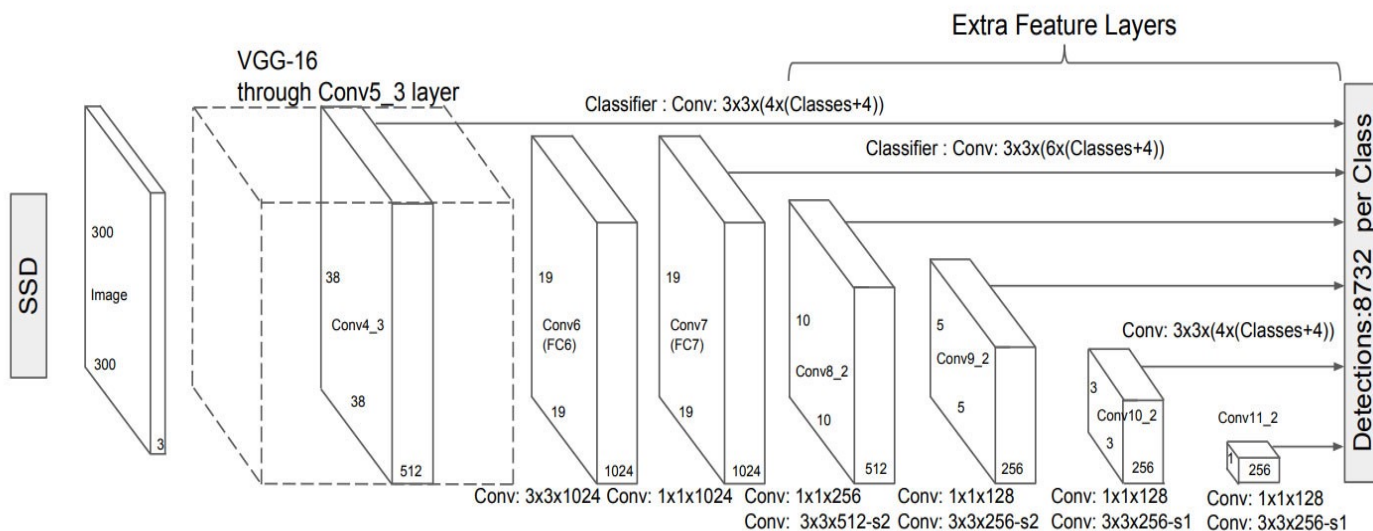
- 네 번째 anchor box:  $AR=1/2$

$$(w_i, h_i) = \left( s_j \sqrt{a_i}, s_j \frac{1}{\sqrt{a_i}} \right) = \left( 0.62 * \sqrt{1/2}, 0.62 * \frac{1}{\sqrt{1/2}} \right) \approx (0.44, 0.88)$$



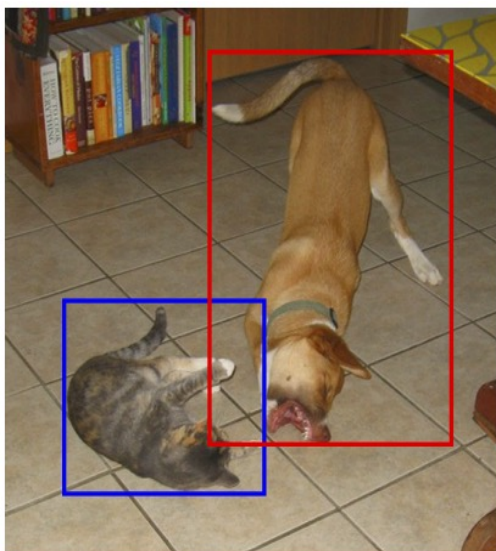
# SSD

- 크기가 다른 여러 개의 feature map을 사용하는 이유
  - 출력층에 가까울수록 feature map의 크기가 작아짐

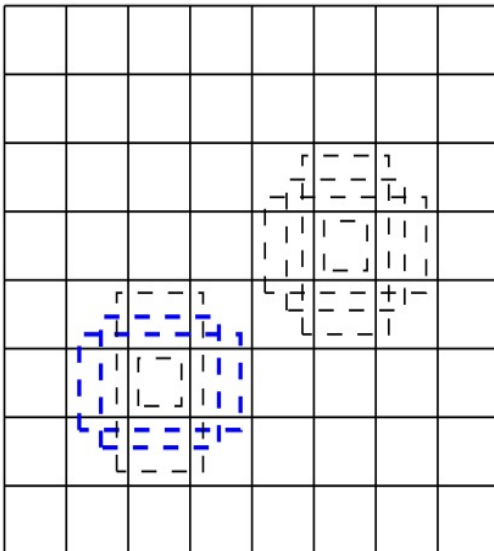


# SSD

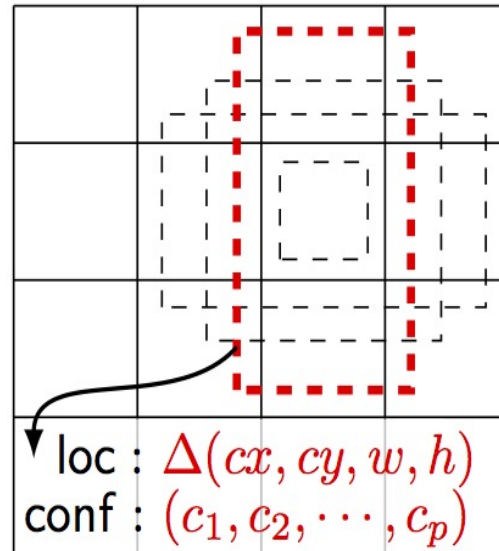
- 크기가 다른 여러 개의 feature map을 사용하는 이유
  - 각 feature map의 크기가 다르기 때문에 (receptive field가 다름), feature map별로 다양한 크기의 anchor box들이 생성  $\Rightarrow$  보다 다양한 크기의 물체들을 더 잘 찾을 수 있음
    - Feature map이 작을 수록 더 큰 이미지를 찾을 수 있다.



(a) Image with GT boxes



(b)  $8 \times 8$  feature map

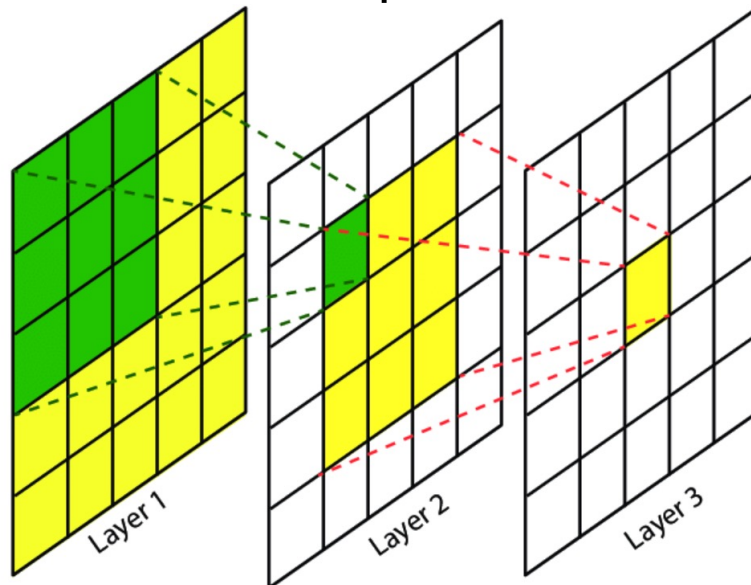


loc :  $\Delta(cx, cy, w, h)$   
 conf :  $(c_1, c_2, \dots, c_p)$

(c)  $4 \times 4$  feature map

# SSD

- 참고: Receptive field (수용 영역)
  - 필터를 이용해서 도출된 feature map의 각 셀이 원본 이미지에서 담당하는 영역
  - 층이 깊어질수록 receptive field 가 커짐





# SSD

---

- 학습하기 (training)
  - 생성된 각 anchor box가 하나의 관측치로 간주
    - 각 관측치에 대한 비용함수 계산 필요
  - 각 anchor box에 대해 비용함수를 계산하기 위해서는 각 anchor box에 대한 정답상자 (GTBB)를 지정해야 함
  - SSD에서 사용된 matching strategy
    - 각 anchor box에 대해서 GTBB하고 겹치는 정도를 계산
    - 이를 위해 IoU (Intersection over Union) 지표 사용 (Jaccard overlap 라고도 함)






# SSD

---

- 학습하기 (cont'd)
  - IoU (Jaccard overlap) (두 개의 경계 상자에 대해)
    - 겹치는 부분 / 전체 영역


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



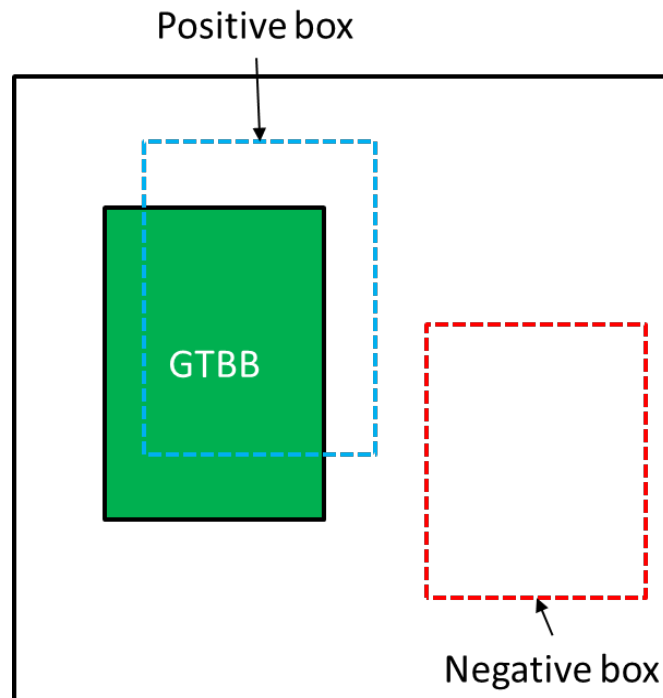
# SSD

---

- 학습하기 (cont'd)
  - SSD에서 사용된 matching strategy (cont'd)
    - 각 anchor box에 대해서 GTBB하고 겹치는 정도를 계산
    - 각 GTBB에 대해서 IoU값이 제일 큰 anchor box를 찾고, 해당 GTBB를 해당 anchor box의 정답으로 할당
    - GTBB를 할당 받지 못한 anchor box들 중에서 GTBB와의 IoU 값이 0.5 보다 큰 GTBB를 해당 anchor box의 GTBB로 할당
    - 0.5가 넘는 GTBB가 여러 개 있다면 그 중 가장 큰 IoU를 갖는 GTBB를 할당
    - GTBB가 할당된 anchor box를 positive box라고 하고, 그렇지 않은 anchor box를 negative box라고 함

# SSD

- 학습하기 (cont'd)
  - 아래 그림에서 점선은 Anchor boxes





# SSD

---

- Hard negative sampling

- 보통의 이미지의 경우에는 GTBB에 해당 하는 영역보다 그렇지 않은 영역 (background 라고 함)이 더 큼 (앞 사진의 경우)  $\Rightarrow$  # positive boxes  $\ll$  # negative boxes (심각한 불균형 문제)
- 모든 negative box를 사용한 것이 아니라, 일부의 negative box 만을 사용
- 백그라운드 클래스에 대한 확률이 낮은 순으로 선택해서 positive box의 수와 negative box의 수의 비율이 1:3이 되도록 함



# SSD

---

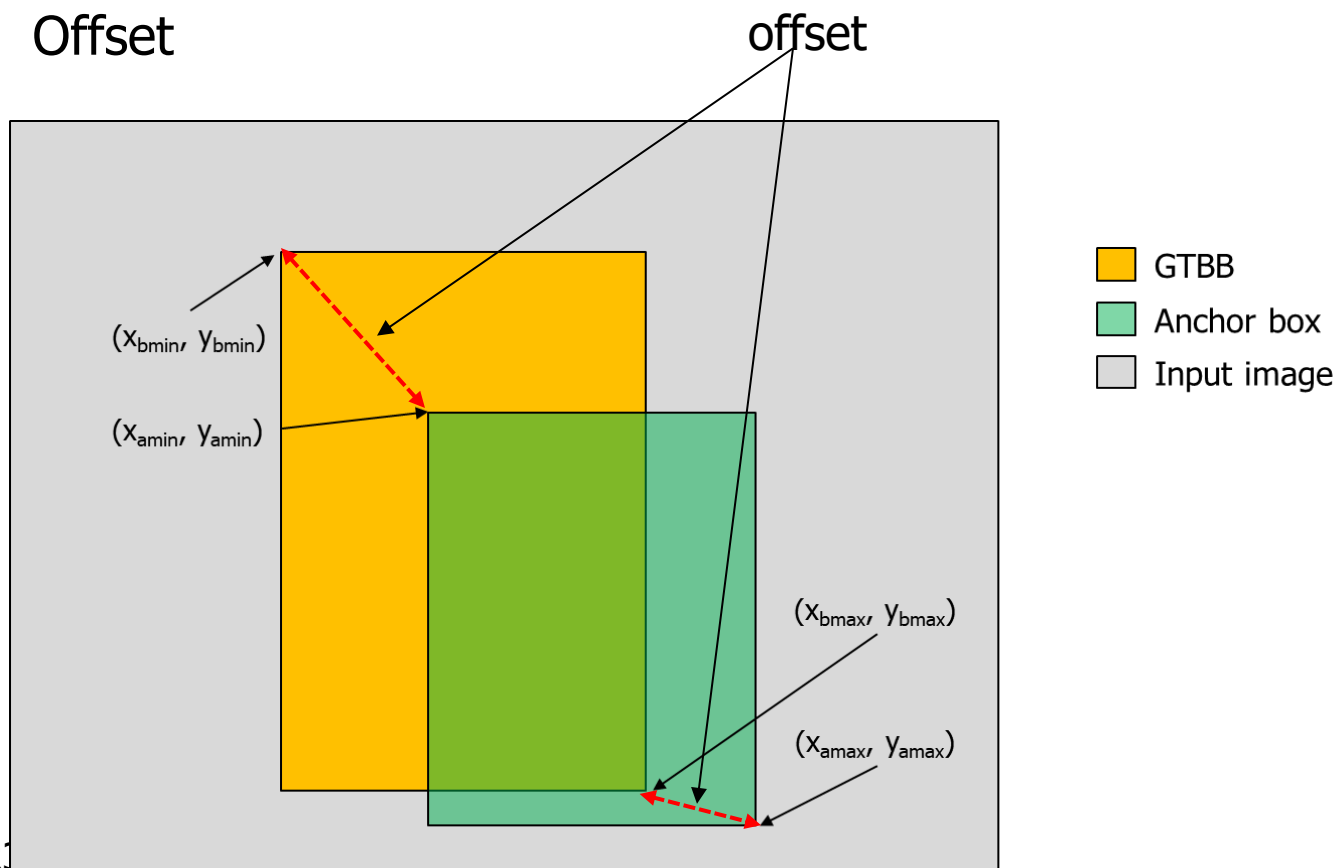
## ■ 예측치

- 각 anchor box에 대해서 SSD를 통해 예측되는 값들의 종류는 2가지
  - ① 해당 anchor box로부터 GTBB가 얼마나 떨어져 있는지 + 얼마나 더 큰지 (이러한 값을 offset 이라고 함)
    - 4 개의 offset 값 출력
  - ② 해당 GTBB에 존재하는 object의 클래스가 무엇일지에 대한 확률
    - 총 21개의 클래스 존재: 20개는 실제 object class, 1개는 background class

# SSD

## ■ 예측치 (cont'd)

### ■ Offset





# SSD

---

- 비용함수

- $L = \frac{1}{N}(L_{cls} + \alpha L_{loc})$

- 즉, classification loss + localization loss
    - $L_{loc}$ : Smooth L1 비용함수 (Fast R-CNN에서 사용)
    - $L_{cls}$ : 교차 엔트로피 비용함수
    - $N = \#$  of matched anchor boxes
    - $\alpha$  는 가중치,  $\alpha = 1$



# SSD

- 비용함수 (cont'd)

- $L_{loc}$

- How to calculate ground truth offset?
    - SSD does not recommend to directly predict the raw pixel error values yoff. Instead, **the offset values are used (중심 좌표와 너비, 높이에 대한 offsets)**. The ground truth bounding box and anchor box coordinates are first expressed in centroid-dimensions format

$$y_{gbox} = ((x_{gmin}, y_{gmin}), (x_{gmax}, y_{gmax})) \rightarrow (c_{gx}, c_{gy}, w_g, h_g)$$

$$y_{anchor} = ((x_{amin}, y_{amin}), (x_{amax}, y_{amax})) \rightarrow (c_{ax}, c_{ay}, w_a, h_a)$$

where

$$(c_{gx}, c_{gy}) = \left( x_{gmin} + \frac{x_{gmax} - x_{gmin}}{2}, y_{gmin} + \frac{y_{gmax} - y_{gmin}}{2} \right)$$

$$(w_g, h_g) = (x_{gmax} - x_{gmin}, y_{gmax} - y_{gmin})$$





# SSD

---

- The ground truth offsets

$$y_{gt\_off} = \left( \frac{c_{gx} - c_{gx}}{w_a}, \frac{c_{gy} - c_{gy}}{h_a}, \log \frac{w_g}{w_a}, \log \frac{h_g}{h_a} \right)$$

- Smooth L1 loss (a.k.a., Huber loss)

$$L(x, y) = \sum_{j \in Pos} \sum_{i=1}^4 z_i$$

where  $z_i$  is given by:      x는 정답, y는 예측치

$$z_i = \begin{cases} 0.5(x_i - y_i)^2 / \text{beta}, & \text{if } |x_i - y_i| < \text{beta} \\ |x_i - y_i| - 0.5 * \text{beta}, & \text{otherwise} \end{cases}$$



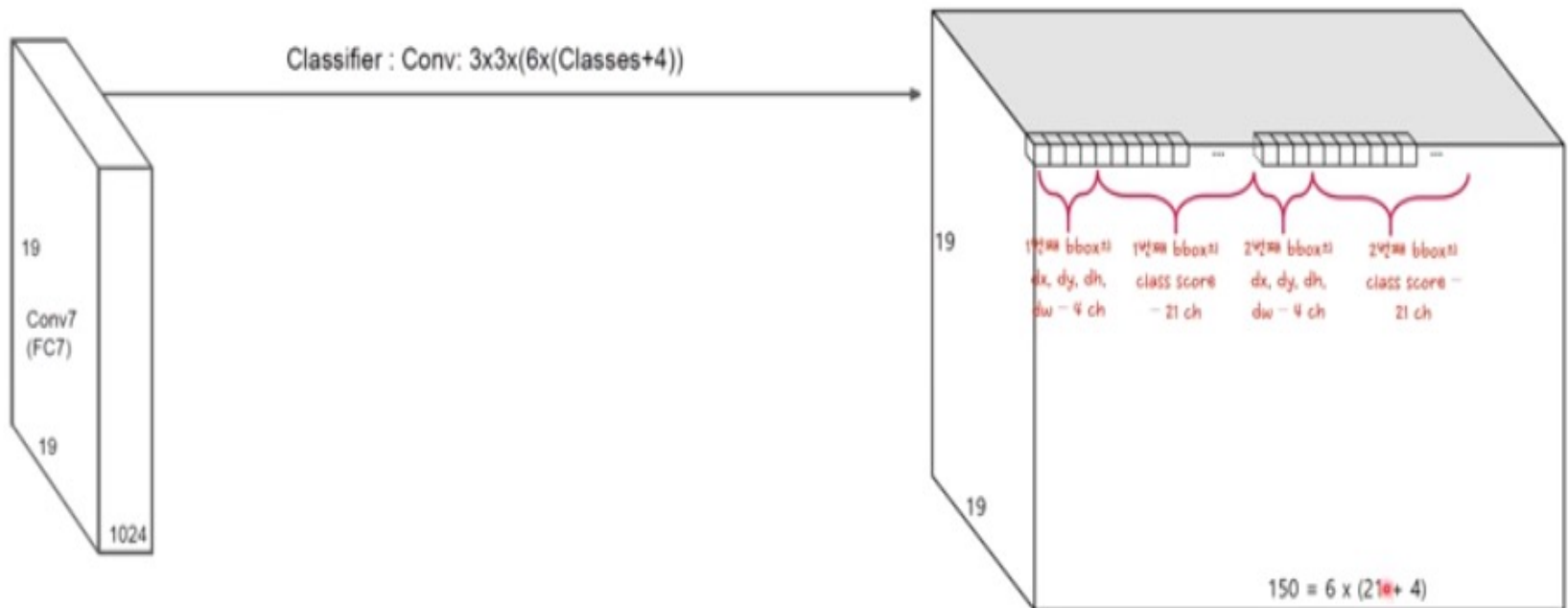
# SSD

---

- Convolution filter를 사용한 결과 출력
  - 각 feature map에 3x3 conv2d filter를 적용하여 종속변수에 대한 예측치를 출력
  - 각 anchor box별로 25개의 값 출력
    - 4개의 offset 값, 21개 클래스별 확률
  - 각 셀별로 6개의 anchor box가 존재하는 경우, 각 셀별로  $6*(4+21)$ 의 값 출력
  - 이를 위해, 3x3 conv2d filter를  $6*(4+21)$  개 적용 => 즉, 도출되는 activation map의 depth =  $6*(4+21)$
  - padding을 해서 입력 feature map과 출력 feature map의 크기가 동일하게 설정

# SSD

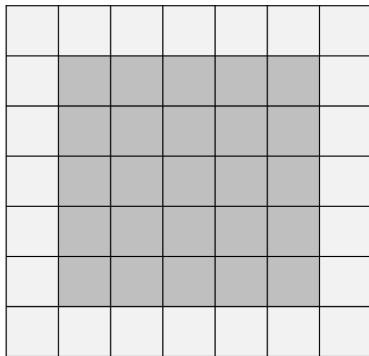
- Convolution filter를 사용한 결과 출력 (cont'd)



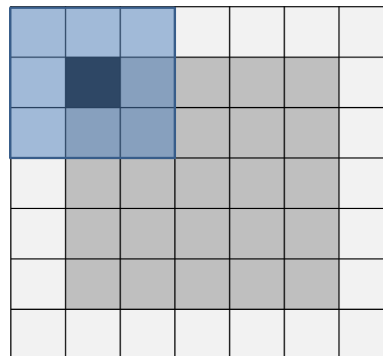
첫번째 (4+21)은 해당 cell에 대한 첫번째 anchor box에 대한 예측치가 됨, 두번째 (4+21)은 해당 cell에 대한 두번째 anchor box에 대한 예측치가 됨, ... 나머지는 동일한 방식으로 작동.

# SSD

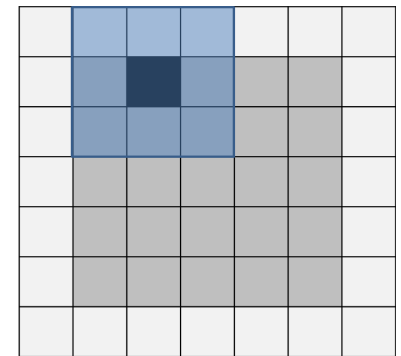
- Convolution filter를 사용한 결과 출력 (cont'd)
  - 필터 적용의 예



원본 feature map (진한 회색 부분)  
+ padding (밝은 회색 부분)



여기에 3x3 filter를 적용 (첫번째 셀  
에 대해서 적용하는 경우) 여러번  
적용 ( $6 \times (4+21)$ )



두번째 셀에 적용하는 경우



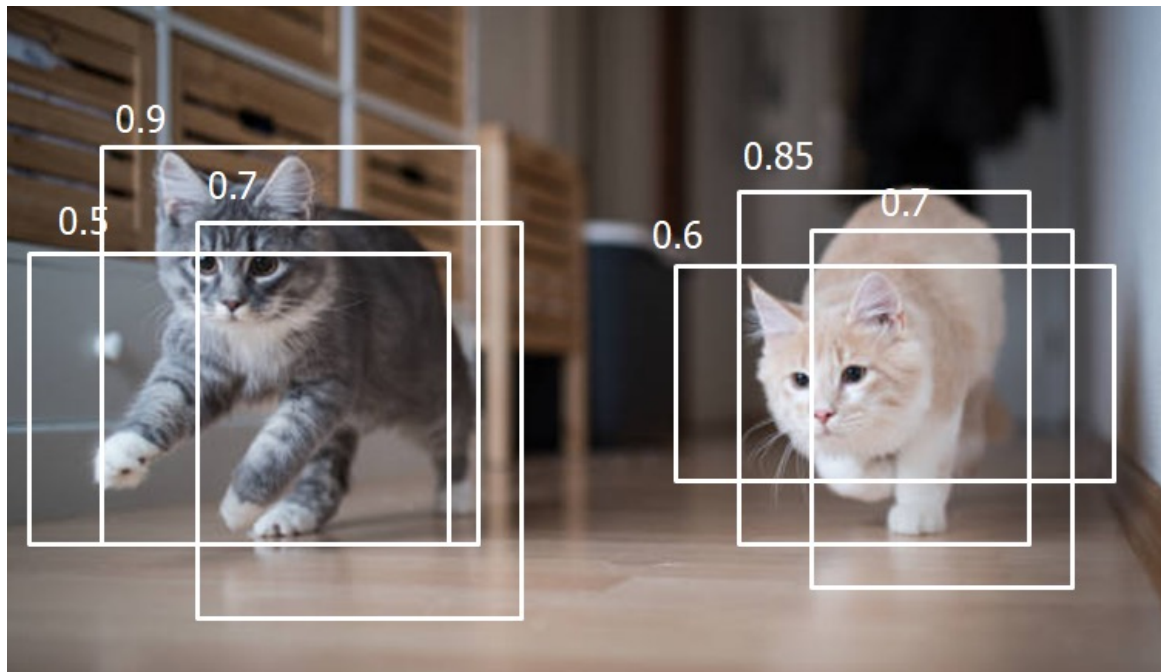
# SSD

---

- 새로운 이미지에 대한 정답 예측 (inference)
  - 새로운 이미지에 존재하는 물체를 찾기 위해
  - 학습에서 사용 방식과 동일한 방식으로 anchor box 생성
    - 각 anchor box에 대해서
      - offset 예측 (해당 anchor box 좌표에 offset을 더해서 최종 상자 예측)
      - 각 클래스 확률 예측
  - 특정 object 에 대해서 많은 상자들이 중복됨
  - 가장 정확도가 높은 상자를 추출하기 위해서 non-maximum suppression 방법 사용 (확률값이 최대가 아닌 상자들은 삭제를 하겠다는 의미)

# SSD

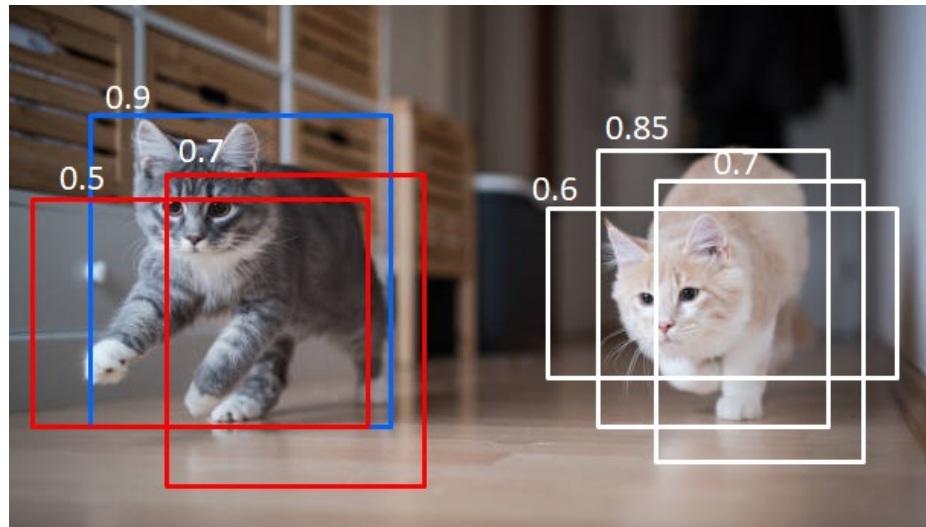
- Non-maximum suppression
  - 아래와 같은 사진이 있다고 가정



# SSD

- Non-maximum suppression (cont'd)
  - 특정 클래스에 대해서
    - 확률이 특정값 (예, 0.1) 이하인 상자는 모두 삭제
    - 남아있는 상자들에 대해서 확률값이 제일 큰 상자를 선택
      - 해당 상자와 IoU 값이 0.45 이상인 다른 상자들을 모두 삭제

그러면 오른쪽 그림에서  
파란색에 해당하는 상자가  
선택되어지고, 해당  
상자와의 중복 정도가 높은  
빨간색 상자들이 삭제됨

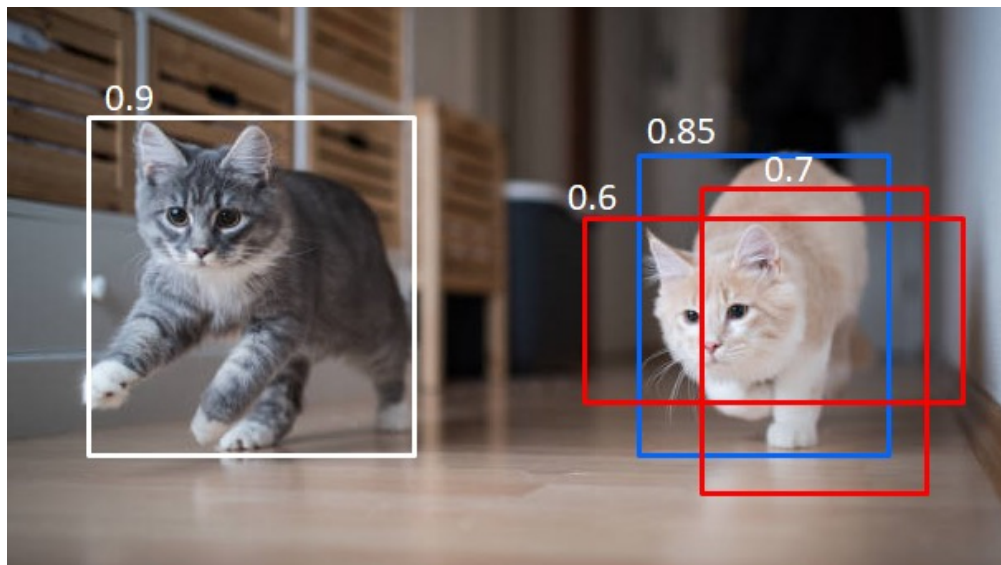


Object detection

# SSD

- Non-maximum suppression (cont'd)
  - 특정 클래스에 대해서
    - 남아 있는 상자들에 대해서 동일한 과정 반복
      - 확률값이 제일 큰 상자를 선택  $\Rightarrow$  해당 상자와의 IoU 값이 0.45 이상인 다른 상자들을 모두 삭제

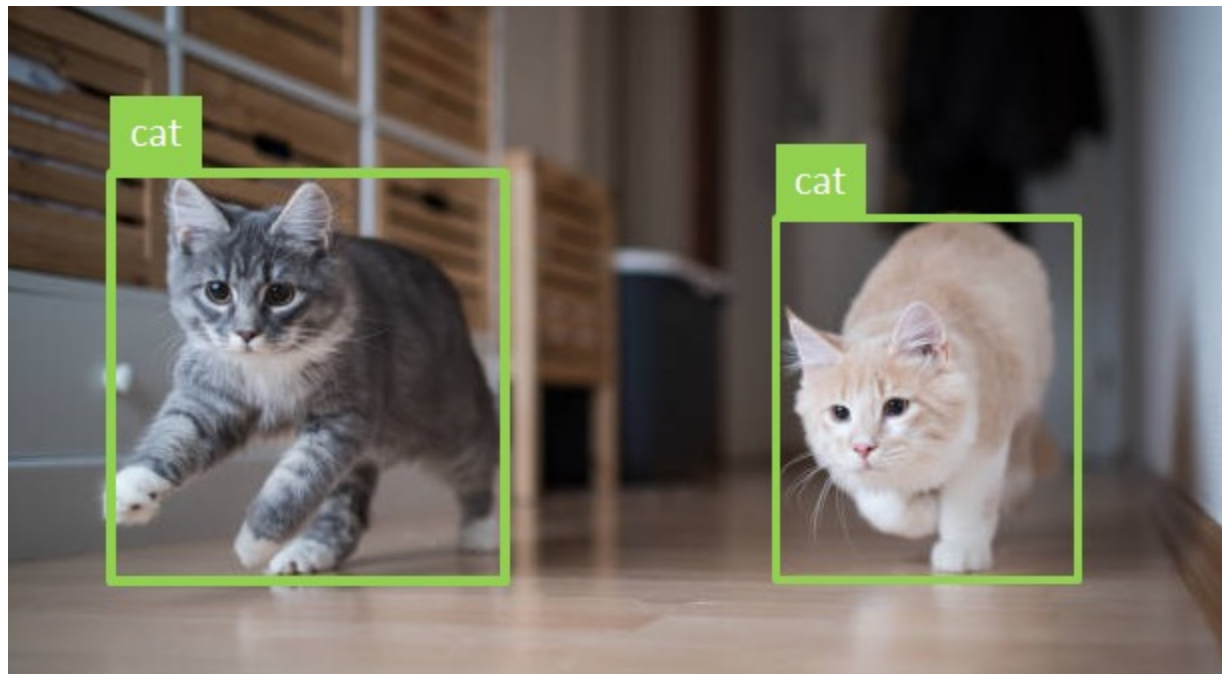
그러면 오른쪽 고양이에 대해서 파란색 상자가 선택되고 나머지 상자들이 삭제된다.





# SSD

## ■ 최종 결과물



# SSD

## ■ 성능

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast [6]	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
Fast [6]	07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Faster [2]	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
Faster [2]	07+12	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
Faster [2]	07+12+COCO	78.8	84.3	82.0	77.7	68.9	65.7	88.1	88.4	88.9	63.6	86.3	70.8	85.9	87.6	80.1	82.3	53.6	80.4	75.8	86.6	78.9
SSD300	07	68.0	73.4	77.5	64.1	59.0	38.9	75.2	80.8	78.5	46.0	67.8	69.2	76.6	82.1	77.0	72.5	41.2	64.2	69.1	78.0	68.5
SSD300	07+12	74.3	75.5	80.2	72.3	66.3	47.6	83.0	84.2	86.1	54.7	78.3	73.9	84.5	85.3	82.6	76.2	48.6	73.9	76.0	83.4	74.0
SSD300	07+12+COCO	79.6	80.9	86.3	79.0	<b>76.2</b>	57.6	87.3	88.2	88.6	60.5	85.4	<b>76.7</b>	<b>87.5</b>	<b>89.2</b>	84.5	81.4	55.0	81.9	<b>81.5</b>	85.9	78.9
SSD512	07	71.6	75.1	81.4	69.8	60.8	46.3	82.6	84.7	84.1	48.5	75.0	67.4	82.3	83.9	79.4	76.6	44.9	69.9	69.1	78.1	71.8
SSD512	07+12	76.8	82.4	84.7	78.4	73.8	53.2	86.2	87.5	86.0	57.8	83.1	70.2	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3
SSD512	07+12+COCO	<b>81.6</b>	<b>86.6</b>	<b>88.3</b>	<b>82.4</b>	76.0	<b>66.3</b>	<b>88.6</b>	<b>88.9</b>	<b>89.1</b>	<b>65.1</b>	<b>88.4</b>	73.6	86.5	88.9	<b>85.3</b>	<b>84.6</b>	<b>59.1</b>	<b>85.0</b>	80.4	<b>87.4</b>	<b>81.2</b>

지표: mAP (mean Average Precision)