SSD: Single Shot MultiBox Detector

Wei Liu(UNC Chapel Hill)
Dragomir Anguelov(Zoox Inc.)
Dumitru Erhan(Google Inc.)
Christian Szegedy(Google Inc.)
Scott Reed(University of Michigan, Ann-Arbor)
Cheng-Yang Fu(UNC Chapel Hill)
Alexander C. Berg(UNC Chapel Hill)

INDEX

Abstract

Introduction

The Single Shot Detector(SSD)

- Model

- - Multi-scale feature maps for detection

- - Convolutional predictors for detection

- - Default boxes and aspect ratios

- Training

- - Matching strategy

- - Training objective

- - Choosing scales and aspect ratios for default boxes

- - Hard negative mining

- - Data augmentation

Experimental Results

- Base network

- PASCAL VOC2007

- Model analysis

- - Data augmentation is crucial

- - More default box shapes is better

- - Atrous is faster

- - Multiple output layers at different resolution is better

- PASCAL VOC2012

- COCO

- Preliminary ILSVRC results

- Data Augmentation for Small Object Accuracy

- Inference time

Related Work

Conclusions

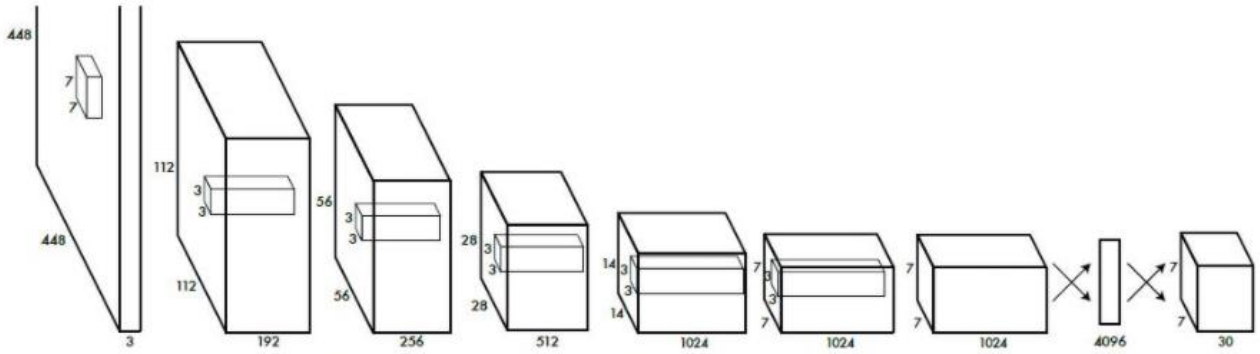
ABSTRACT

SSD가 기여한 점:

1. Two-stage detection과는 다르게 단일 네트워크로 구성되어 있음. 따라서 속도가 더 빠름.
   (Faster R-CNN -> Base CNN, RPN, Detection Network)

2. YOLOv1과는 다르게 여러 크기의 특징맵에서 Detection 수행. 따라서 다양한 크기의 객체 탐지 용이.

YOLOv1:



INTRODUCTION

Two-stage 방식의 문제점:

1. 이미지 안에 객체 들어 있을 법한 패치를 추정하고 나서 그 패치에 대한 후속 처리.

   -> 속도가 느리다. Faster R-CNN도 7 FPS.

   -> 실시간성이 중요한 애플리케이션이나 임베디드 시스템에 부적절.

2. 적절한 패치를 추출하기 위해서 원본 이미지에 어떤 변형을 가함.

Rich feature hierarchies for accurate object detection and semantic segmentation (warping)

INTRODUCTION

SSD의 방식

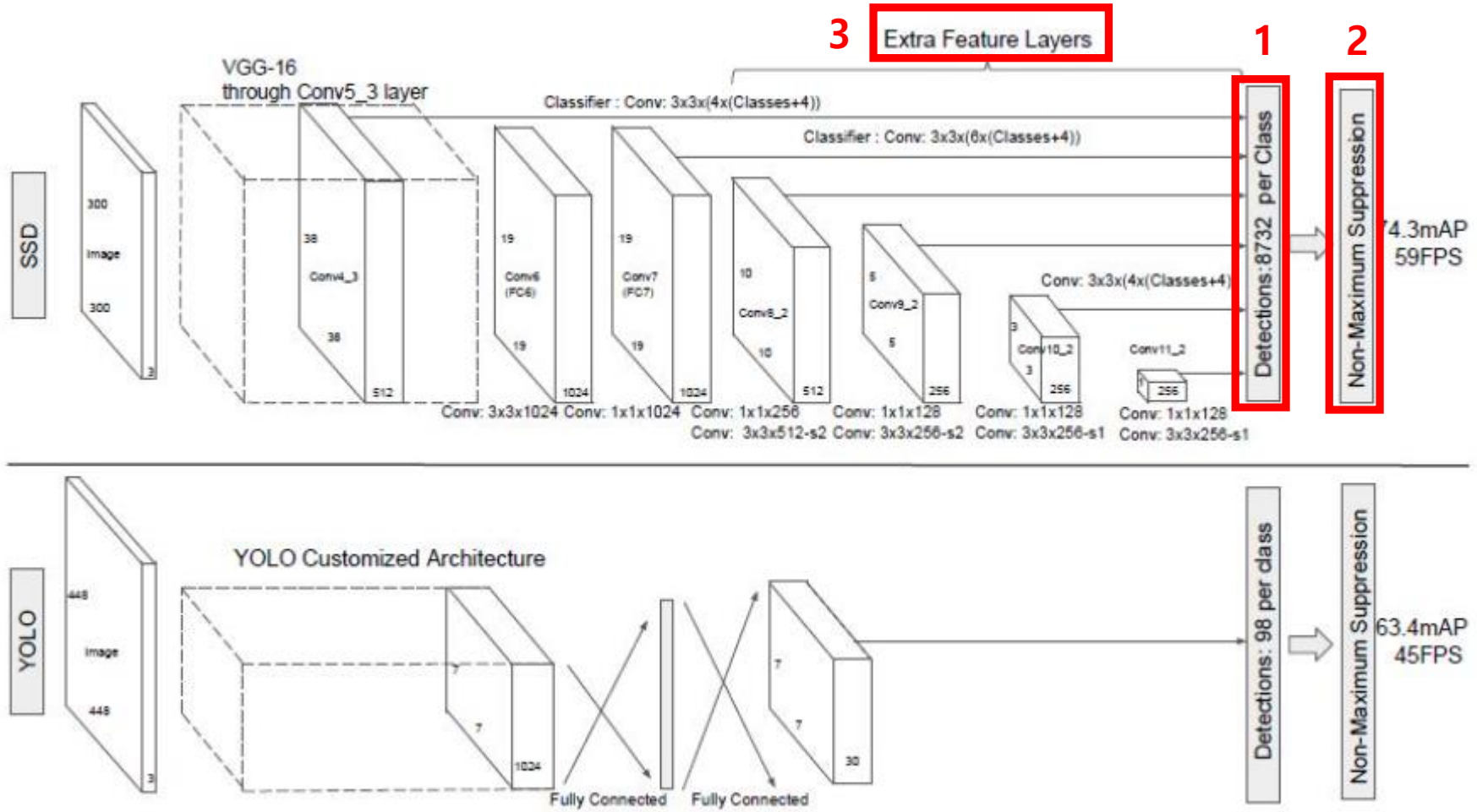
1. CNN에서 완전 연결 계층을 제거하고 여러 컨볼루션 계층을 붙임.

   -> CNN, SVM 등을 각각 훈련시키지 않고 하나의 단일 네트워크를 종단간으로 훈련 가능.

2. 각 계층의 출력 특징맵에서 Default box들을 이용해서 클래스 Score, 박스 Offset을 예측.

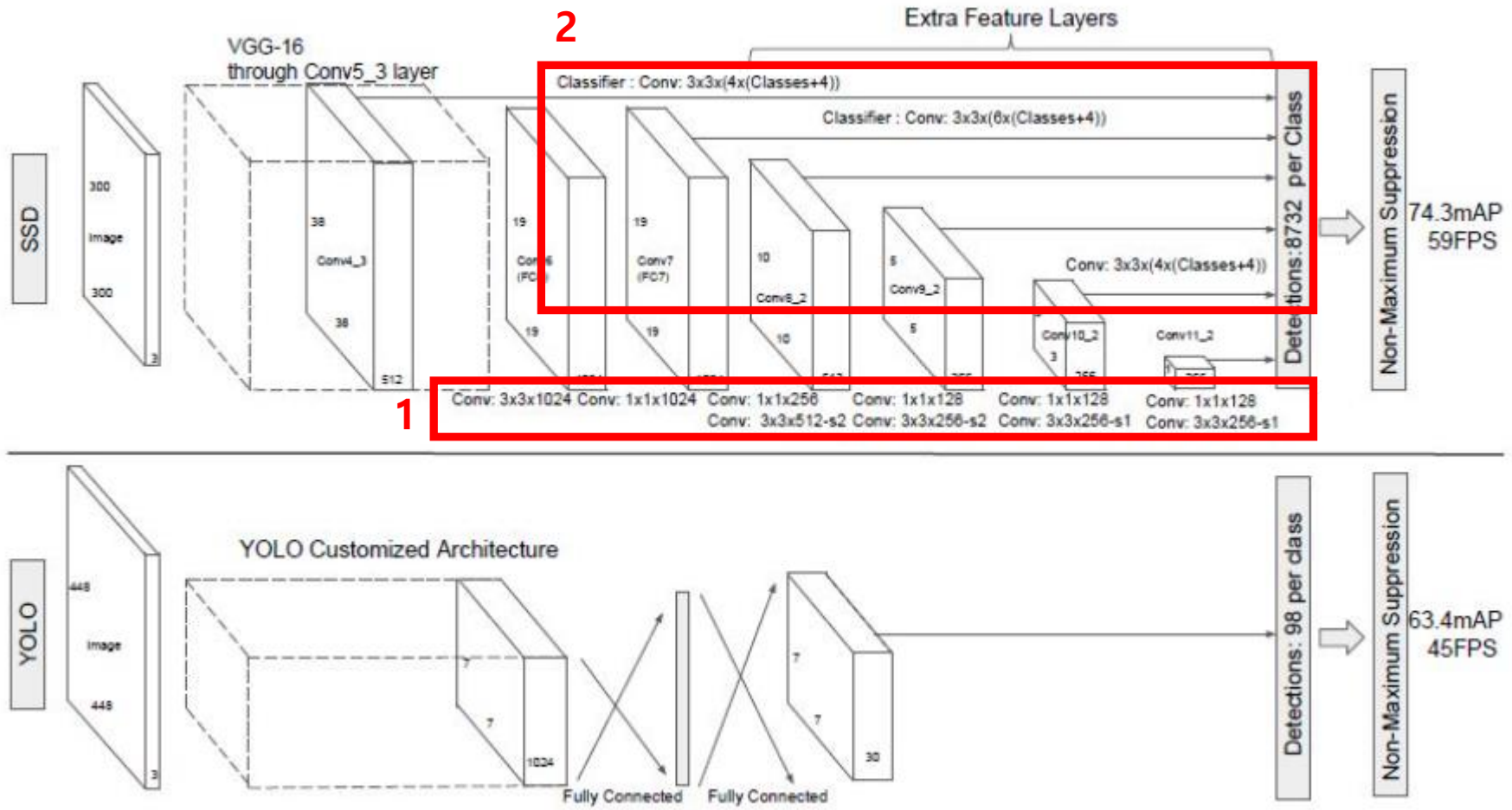
   -> 각 특징맵의 크기가 다르기 때문에 담당하는 객체 사이즈가 다름.

THE SINGLE SHOT DETECTOR(SSD) - MODEL



1. 정해진 박스 수 만큼 예측 값을 뽑아냄(300x300 입력 사이즈 기준 클래스당 8732 박스).
2. 예측 수행 후에 NMS 수행.
3. 이를 위해서 완전 연결 계층을 제거하고 추가적인 컨볼루션 계층을 추가.

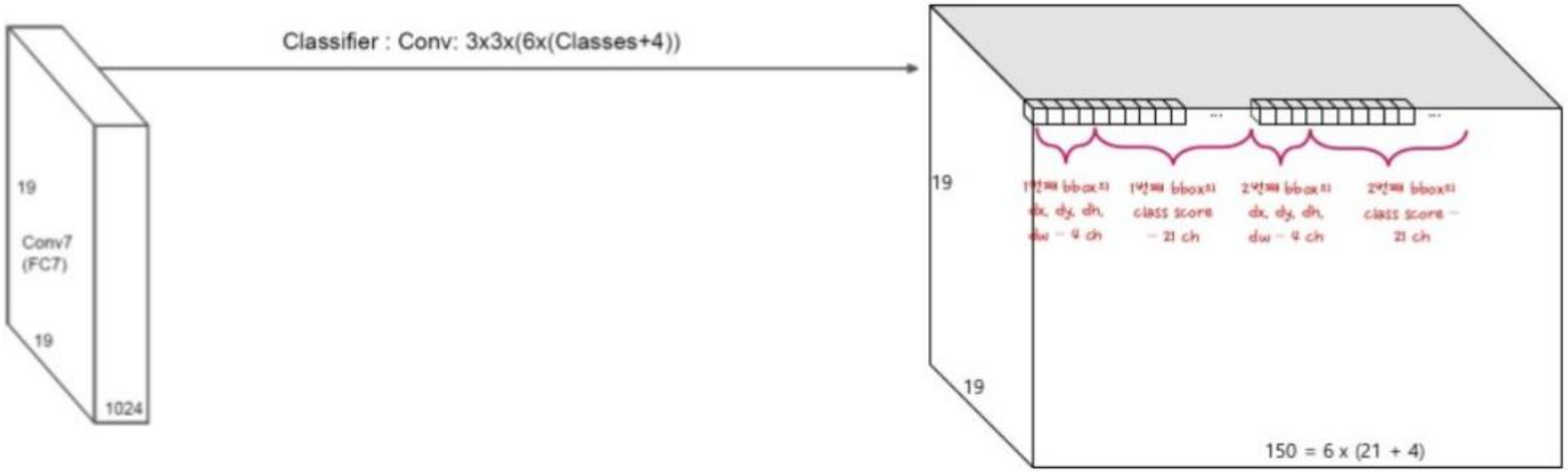
THE SINGLE SHOT DETECTOR(SSD) – MULTI-SCALE FEATURE MAPS



1. 작은 사이즈를 가진 필터로 컨볼루션 연산을 진행해서 점차 특징맵의 크기를 줄여감.
   그러면서 각 특징맵에서 예측 수행.

2. Detection을 수행하는 필터는 입력 특징맵의 채널 수가 p일때 3x3xp

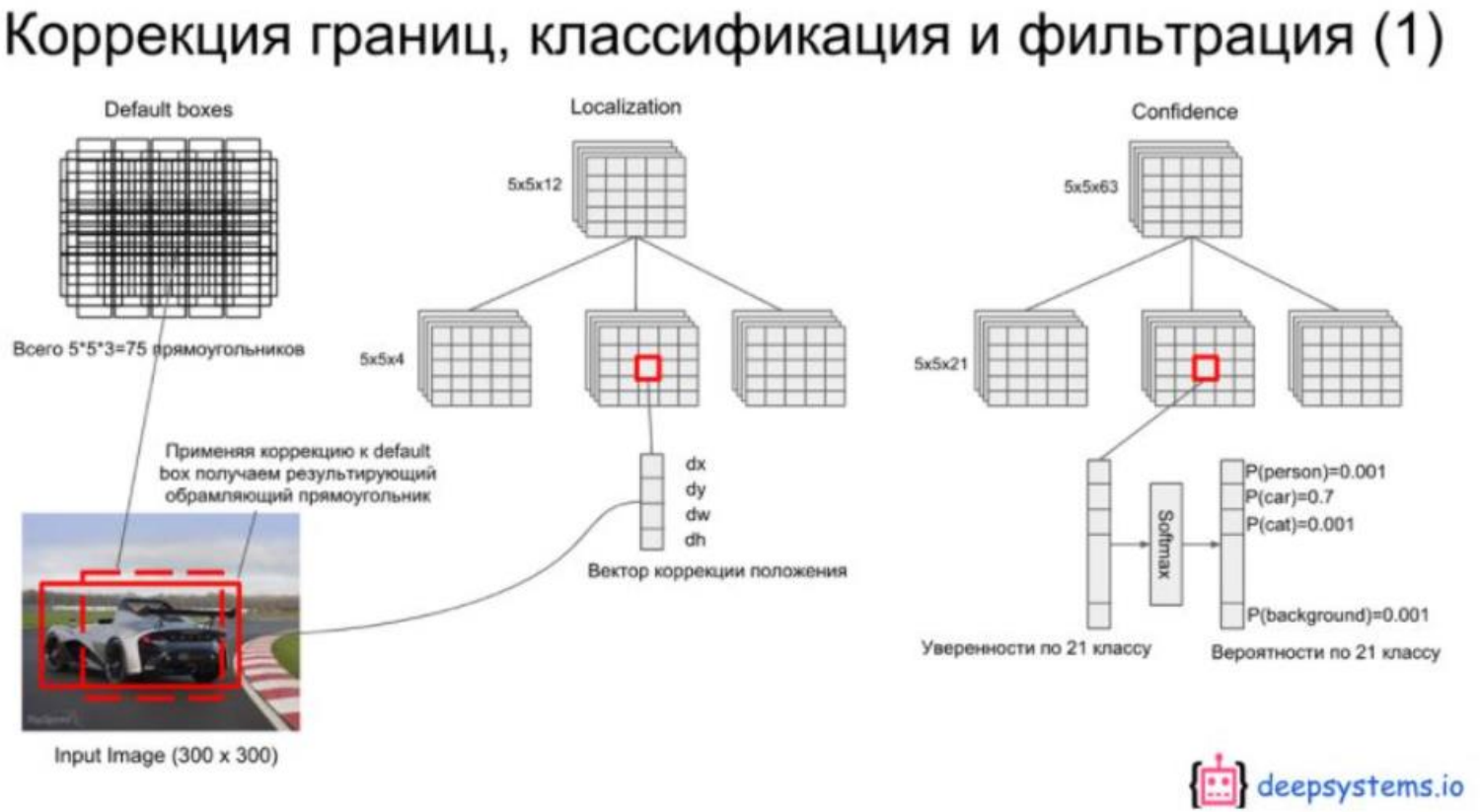
THE SINGLE SHOT DETECTOR(SSD) – DEFAULT BOXES AND ASPECT RATIOS



Jinwon Lee, Samsung Electronics Enginner - PR-132: SSD: Single Shot MultiBox Detector

특징맵의 각 셀에 객체의 중심 좌표가 존재한다면(GT의 중심 좌표) 그와 관련하여 Default box가 매핑됨.
예측 값은 매핑된 박스의 x, y좌표의 Offset, Width, Height와 관련된 값을 예측함.
그리고 각 박스마다 Class score도 예측함.
따라서 각 셀마다 (#class + 4)*#box 크기의 벡터가 생성되고
특징맵의 크기가 m*n이라면 최종적으로 m * n * #box *(#class + 4)의 아웃풋이 각 특징맵마다 생성됨.

THE SINGLE SHOT DETECTOR(SSD) – DEFAULT BOXES AND ASPECT RATIOS



예를 들어서 5x5의 격자의 각 셀에 3개씩 Box가 할당되어 있다고 하면 총 5x5x3=75개의 박스가 존재함.

Localization과 관련된 특징맵만 모아보면 5x5x12(12=4(dx, dy, dw, dh)x3(3개의 박스))

Class score와 관련된 특징맵을 모아보면 5x5x63(63=21(클래스 개수(배경 포함))x3(3개의 박스))

THE SINGLE SHOT DETECTOR(SSD) – MATCHING STRATEGY

각 GT 박스에 대해 다양한 종횡비와 크기를 가진 Default 박스를 매칭하는 방법:

1. GT 박스와 가장 Jaccard overlap(IoU)가 높은 박스를 매칭한다.
2. Default 박스 중에서 GT 박스와 Jaccard overlap이 Threshold(여기서는 0.5) 이상이라면 그 박스는 해당 GT 박스에 매칭.

이렇게 해서 여러 상자 예측 값을 뽑아내게 만든다.

THE SINGLE SHOT DETECTOR(SSD) – TRAINING OBJECTIVE

$x_{ij}^{p} = \{1, 0\}$ i번째 박스가 클래스 p의 j번째 GT 박스와 매칭되었는지 여부.

$\sum_i x_{ij}^{p} \geq 1$ 위 조건을 만족하는 Default 박스는 1개 이상이다.

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

N은 매칭된 Default box의 숫자이다. 매칭된 박스가 없다면 손실은 0이 됨. l은 예측된 박스와 관련된 파라미터, g는 GT 박스와 관련된 파라미터. α는 Localization과 관련된 손실의 비중을 결정하는 요소인데 교차 검증에 의해서 1로 설정됨.

THE SINGLE SHOT DETECTOR(SSD) – TRAINING OBJECTIVE

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^{w} = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^{h} = \log\left(\frac{g_j^h}{d_i^h}\right)$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

Default 박스 d의 중심 좌표 cx, cy 그리고 넓이와 높이인 w, h의 offset을 Regression한다.

즉 매칭된 박스를 좌표 이동하고 넓이와 높이를 변경해서 최대한 GT 박스에 맞게 한다.

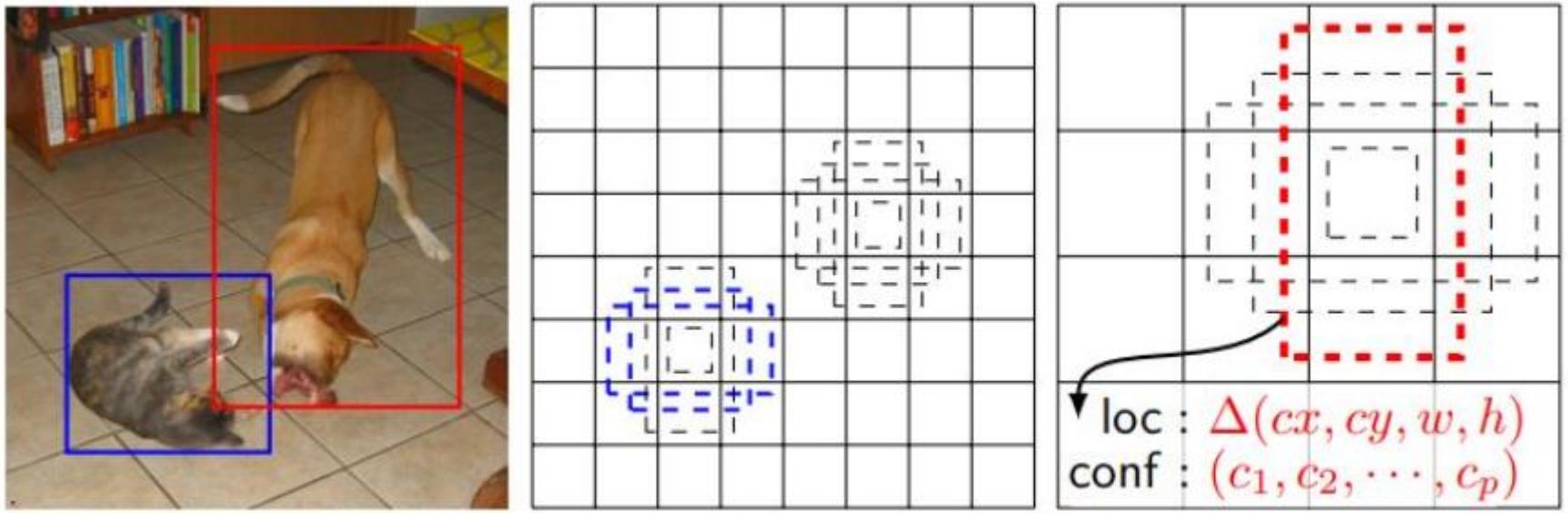
GT 박스의 값을 그대로 쓰지 않고 정규화 해서 터무니 없이 박스의 값이 변하지 않도록 함.

THE SINGLE SHOT DETECTOR(SSD) – TRAINING OBJECTIVE

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

Confidence Loss는 Positive box(GT와 매칭된 박스)와 Negative box(배경으로 분류된 박스)에 대한 Cross entropy의 합을 나타낸다.

THE SINGLE SHOT DETECTOR(SSD) – CHOOSING SCALES AND ASPECT RATIOS



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

각 특징맵의 크기가 다르므로 각 특징맵에서 담당하는 객체의 크기가 달라야 한다.

예를 들어서, 8x8 특징맵에서는 고양이 크기에 맞는 Default 박스를 GT 박스에 매칭할 수 있지만 개의 크기에 맞는 박스는 없다. 4x4 특징맵에서는 8x8 특징맵에서 매칭하지 못한 개에 대한 GT 박스에 Default 박스를 매칭할 수 있게 된다. High Resolution에서는 크기가 큰 객체를, Low Resolution에서는 크기가 작은 객체를 담당하게 된다.

THE SINGLE SHOT DETECTOR(SSD) – CHOOSING SCALES AND ASPECT RATIOS

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

Default 박스들이 전담할 Scale은 위와 같이 계산됨. smin = 0.2, smax = 0.9로 상수 값. 가장 낮은 레벨 계층의 Default 박스들은 원래 크기의 0.2, 가장 높은 레벨 계층의 Default 박스들은 원래 크기의 0.9
k는 각 특징맵의 순서.

$$a_r \in \{1, 2, 3, \tfrac{1}{2}, \tfrac{1}{3}\}$$

각 Default 박스들은 각기 다른 종횡비를 가질 수 있다. 위는 가질 수 있는 종횡비의 집합을 나타냄.

넓이	높이
$(w_k^a = s_k\sqrt{a_r})$	$(h_k^a = s_k/\sqrt{a_r})$

$$s'_k = \sqrt{s_k s_{k+1}}$$

똑같은 Scale이라면 종횡비가 1일때는 정사각형 박스가 될 것이고 2라면 넓이가 높이보다 커진다. 1/2라면 반대로 높이가 넓이보다 커진다. 종횡비가 1일때는 위의 오른쪽과 같은 스케일이 더 추가된다. 그래서 6(종횡비마다 스케일 1개씩 , 1일때는 스케일 2개)개의 Default 박스가 특징맵의 각 셀마다 할당된다. 4개일때는 3, 1/3을 버린다.

THE SINGLE SHOT DETECTOR(SSD) – CHOOSING SCALES AND ASPECT RATIOS

중심좌표	분모	i, j
$(\frac{i+0.5}{\lvert f_k \rvert}, \frac{j+0.5}{\lvert f_k \rvert})$	$\lvert f_k \rvert$	$i, j \in [0, \lvert f_k \rvert)$

Default 박스의 중심 좌표는 위와 같이 계산한다. 분모는 k번째 특징맵의 크기이고 i, j는 k번째 특징맵에서의 각 셀 인덱스를 나타낸다.

8x8의 특징맵에서는 개의 GT 박스와 매칭하는 Default 박스를 찾을 수 없는데 Scale에 의해서 조정된 Default box가 개의 GT를 커버하기에는 크기가 충분히 크지 않기 때문이다. 이때 개의 위치와 관련된 Default 박스들은 모두 Negative로 설정된다.

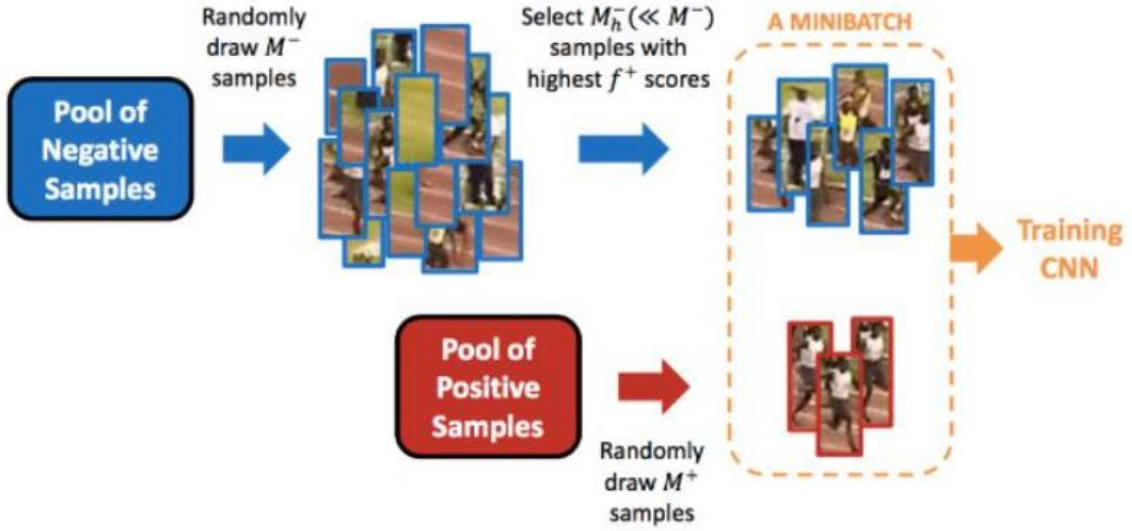
THE SINGLE SHOT DETECTOR(SSD) – HARD NEGATIVE MINING

훈련 중에는 대부분의 Default 박스들은 Negative(객체와 스케일이 안 맞거나 종횡비가 안 맞거나)

이때문에 Positive와 Negative 간의 심한 불균형이 생기는데 이를 해결하기 위해서

Negative 박스들을 Confidence score의 내림차순으로 정렬하고 그 중에서 Hard Example을 골라서

Positive와 Negative의 비율이 1:3이 되게 한다.



(그림 출처: 한보형 님의 슬라이드 "Lecture 6: CNNs for Detection, Tracking, and Segmentation")

THE SINGLE SHOT DETECTOR(SSD) – DATA AUGMENTATION

모델에 다양한 데이터를 입력을 줄 수 있도록 다음의 옵션을 랜덤하게 적용.

1. 그냥 전체 원본 이미지 전체를 사용.

2. 최소 Jaccard overlap을 {0.1, 0.3, 0.5, 0.7. 0.9} 중에 하나로 설정하고 이를 충족하는 패치를 샘플링한다.

3. 그냥 랜덤하게 패치를 샘플링한다.

패치를 샘플링해서 사용할 경우 원본 이미지 크기의 [0.1, 1]이고 종횡비는 ½과 2 사이.

GT의 중심 값이 샘플링된 패치 안에 있다면 패치와 GT가 겹치는 부분 주시.

위와 같은 샘플링 과정을 거치고 나서 각 패치들은 네트워크 입력 크기에 맞게 조정되고

0.5의 확률로 Horizontally flipped 된다. 그리고 다음의 연구에서 제시한 몇가지 Photo-metric distortion을 적용한다.

Howard, A.G.: Some improvements on deep convolutional neural network based image classification. arXiv preprint arXiv:1312.5402 (2013)

EXPERIMENTAL RESULTS – BASE NETWORK

기본적으로 ILSVRC CLS-LOC 데이터셋으로 미리 훈련한 VGG16 네트워크를 실험에 사용한다.

완전 연결 계층 6, 7을 컨볼루션 계층으로 바꾸되 완전 연결 계층에서 몇몇 파라미터를 가져온다.

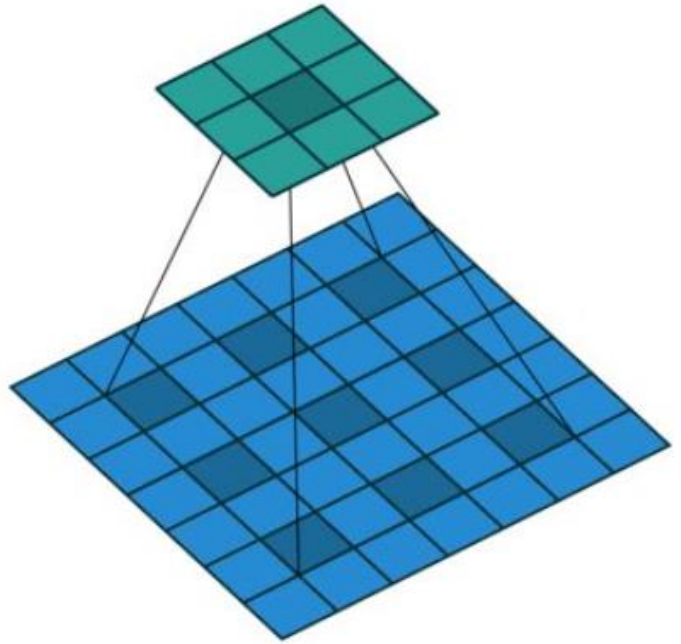
완전 연결 계층 6 바로 전의 POOL5를 2x2 크기의 Stride 2에서 3x3 크기의 Stride1로 바꾼다.

컨볼루션 6, 7은 다음과 같은 Atrous convolution을 사용한다.

FC8과 Dropout은 제거한다.

Batch size 32, Weight Decay 0.0005, Learning Rate 10^{-3}. Momentum 0.9 의 SGD로 Fine tuning 진행.

Receptive field의 크기는 커지면서 파라미터 수는 원래의 컨볼루션보다 적어지는 효과가 있음.



Atrous convolution / Hole algorithm / Dilated convolution

Sik-Ho Tsang - Review: SSD — Single Shot Detector (Object Detection)

EXPERIMENTAL RESULTS – PASCAL VOC2007

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	**76.2**	57.6	87.3	88.2	88.6	60.5	85.4	**76.7**	**87.5**	**89.2**	84.5	81.4	55.0	81.9	**81.5**	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	**81.6**	**86.6**	**88.3**	**82.4**	76.0	**66.3**	**88.6**	**88.9**	**89.1**	**65.1**	**88.4**	73.6	86.5	88.9	**85.3**	**84.6**	**59.1**	**85.0**	80.4	**87.4**	**81.2**

모든 방법들은 PASCAL 2007 test 셋으로 Pre-trained된 Base Network(VGG16)에서 수행됨.

SSD300은 Counterpart의 Fast R-CNN보다 성능이 좋다.

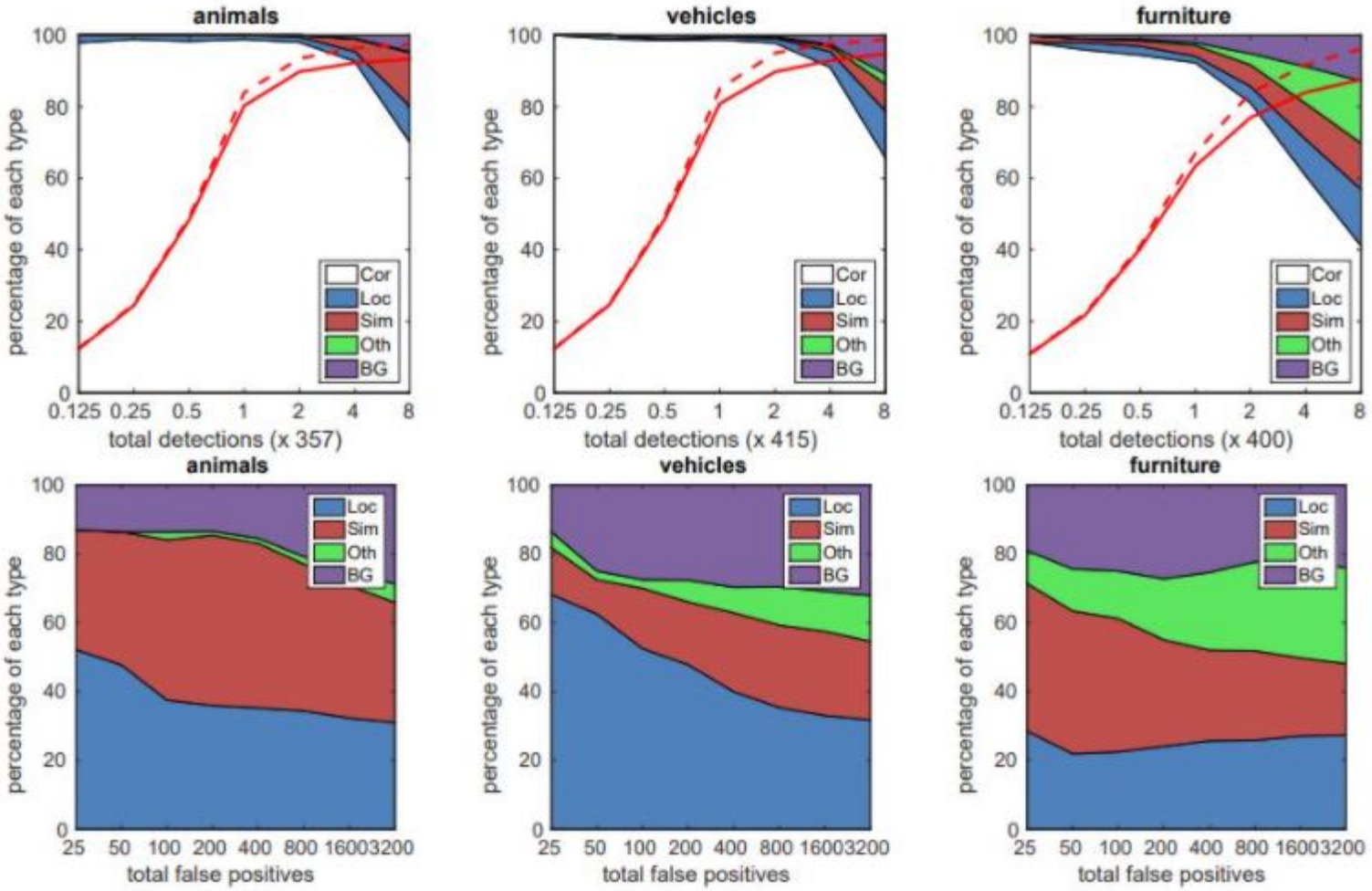
SSD512는 Counterpart의 Faster R-CNN보다 성능이 좋다.

가장 좋은 성능이 좋은 모델은 COCO trainval135k에서 훈련된 모델을 07+12 데이터셋으로 Fine-tuning한 SSD512모델.

EXPERIMENTAL RESULTS – PASCAL VOC2007

SSD 모델의 성능을 자세히 이해하기 위해서 저자들은 다음 연구의 Detection analysis tool을 사용했다.

Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS. (2010)

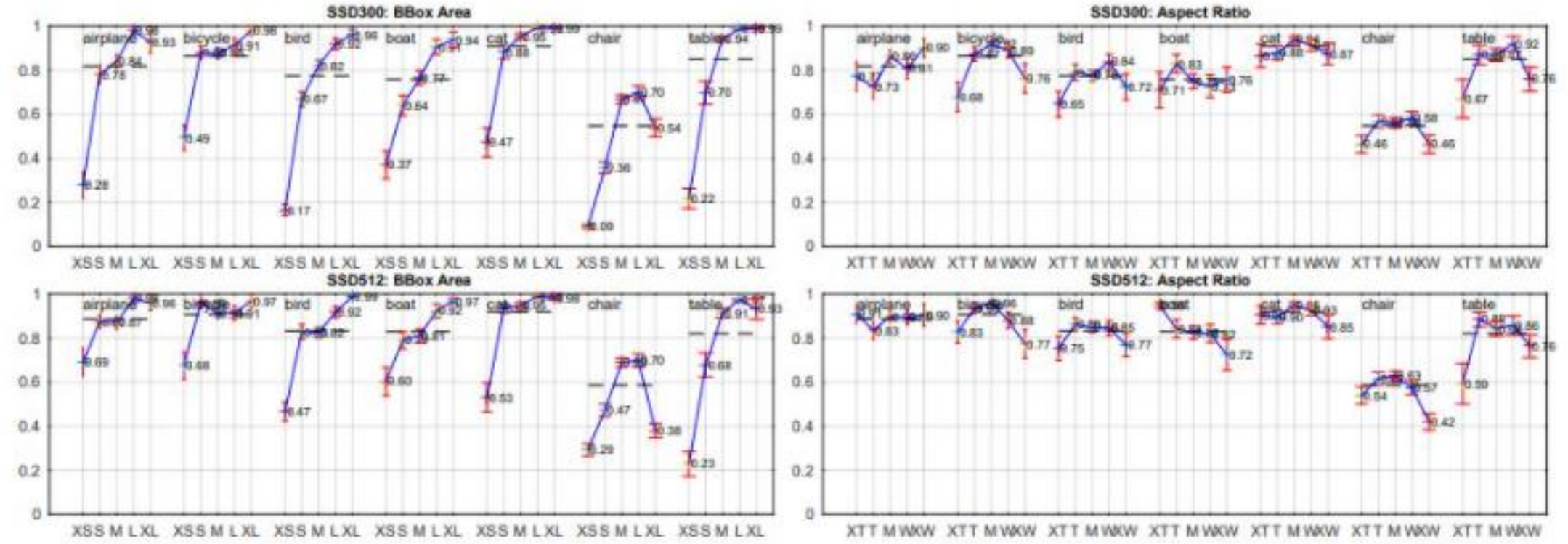


위쪽 행의 대부분의 영역을 흰색이 차지하므로 결과가 대부분 정확하다고 할 수 있다.

R-CNN과 비교했을 때 특히 Localization error가 더 적다.

동물과 관련된 카테고리의 경우 비슷한 클래스끼리의 혼동 에러가 많은데 여러 카테고리에 대해서 Location을 공유하기 때문(?)

EXPERIMENTAL RESULTS – PASCAL VOC2007



왼쪽은 카테고리별로 바운딩 박스의 영역 크기와 관련된 실험 결과를 오른쪽은 종횡비와 관련된 실험 결과를 보여준다.

왼쪽을 보면 SSD가 바운딩 박스의 크기에 민감한 것을 알 수 있다. 특히 크기가 작은 객체에 대한 성능이 떨어지는데 이는 고 레벨 계층에서 크기가 작은 객체에 대한 정보가 손실되었기 때문이다(FPN을 적용하면 개선될 듯).

EXPERIMENTAL RESULTS – DATA AUGMENTATION IS CRUCIAL

	SSD300				
more data augmentation?		✔	✔	✔	✔
include $\{\frac{1}{2}, 2\}$ box?	✔		✔	✔	✔
include $\{\frac{1}{3}, 3\}$ box?	✔			✔	✔
use atrous?	✔	✔	✔		✔
VOC2007 `test` mAP	65.5	71.6	73.7	74.2	**74.3**

Table 2: **Effects of various design choices and components on SSD performance.**

Data augmentation을 적용했을 때의 가장 낮은 성능이 적용하지 않았을 때보다 약 8.8% mAP가 더 높은 것으로 보아 Data augmentation이 중요하다는 것을 알 수 있다.

EXPERIMENTAL RESULTS – MORE DEFAULT BOX SHAPES IS BETTER

	SSD300				
more data augmentation?		✔	✔	✔	✔
include $\{\frac{1}{2}, 2\}$ box?	✔		✔	✔	✔
include $\{\frac{1}{3}, 3\}$ box?	✔			✔	✔
use atrous?	✔	✔	✔		✔
VOC2007 `test` mAP	65.5	71.6	73.7	74.2	**74.3**

Table 2: **Effects of various design choices and components on SSD performance.**

다양한 Aspect ratio를 사용한 모델이 성능이 더 좋다.

EXPERIMENTAL RESULTS – ATROUS IS FASTER

	SSD300				
more data augmentation?		✔	✔	✔	✔
include $\{\frac{1}{2}, 2\}$ box?	✔		✔	✔	✔
include $\{\frac{1}{3}, 3\}$ box?	✔			✔	✔
use atrous?	✔	✔	✔		✔
VOC2007 `test` mAP	65.5	71.6	73.7	74.2	**74.3**

Table 2: **Effects of various design choices and components on SSD performance.**

Atrous 알고리즘을 적용하지 않고 POOL5를 2x2-s2로 유지하고 FC6, 7의 파라미터를 서브 샘플링하지 않고 컨볼루션을 추가하면 정확도는 비슷한데 속도가 20%더 느리다.

EXPERIMENTAL RESULTS – MULTIPLE OUTPUTS WITH DIFFERENT RESOLUTION

Prediction source layers from:						mAP use boundary boxes?		# Boxes
conv4_3	conv7	conv8_2	conv9_2	conv10_2	conv11_2	Yes	No	
✔	✔	✔	✔	✔	✔	74.3	63.4	8732
✔	✔	✔	✔	✔		**74.6**	63.1	8764
✔	✔	✔	✔			73.8	68.4	8942
✔	✔	✔				70.7	69.2	9864
✔	✔					64.2	64.4	9025
	✔					62.4	64.0	8664

Table 3: **Effects of using multiple output layers.**

계층을 하나씩 제거하면서 성능을 관측함. 이때 계층을 제거해도 상자의 숫자는 비슷하게 유지하도록 함.

각기 다른 크기의 Default 박스들을 각 계층에 고루 분포하는 것이 중요하다.

EXPERIMENTAL RESULTS – PASCAL VOC2012

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++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[2]	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
Faster[2]	07++12+COCO	75.9	87.4	83.6	76.8	62.9	59.6	81.9	82.0	91.3	54.9	82.6	59.0	89.0	85.5	84.7	84.1	52.2	78.9	65.5	85.4	70.2
YOLO[5]	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	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
SSD300	07++12+COCO	77.5	90.2	83.3	76.3	63.0	53.6	83.8	82.8	92.0	59.7	82.7	63.5	89.3	87.6	85.9	84.3	52.6	82.5	**74.1**	**88.4**	74.2
SSD512	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
SSD512	07++12+COCO	**80.0**	**90.7**	**86.8**	**80.5**	**67.8**	**60.8**	**86.3**	**85.5**	**93.5**	**63.2**	**85.7**	**64.4**	**90.9**	**89.0**	**88.9**	**86.8**	**57.2**	**85.1**	72.8	**88.4**	**75.9**

EXPERIMENTAL RESULTS – COCO

Method	data	Avg. Precision, IoU:			Avg. Precision, Area:			Avg. Recall, #Dets:			Avg. Recall, Area:		
		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast [6]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast [24]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster [2]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [24]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster [25]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512	trainval35k	**26.8**	**46.5**	**27.8**	**9.0**	**28.9**	**41.9**	**24.8**	**37.5**	**39.8**	**14.0**	**43.5**	**59.0**

Table 5: **COCO `test-dev2015` detection results.**

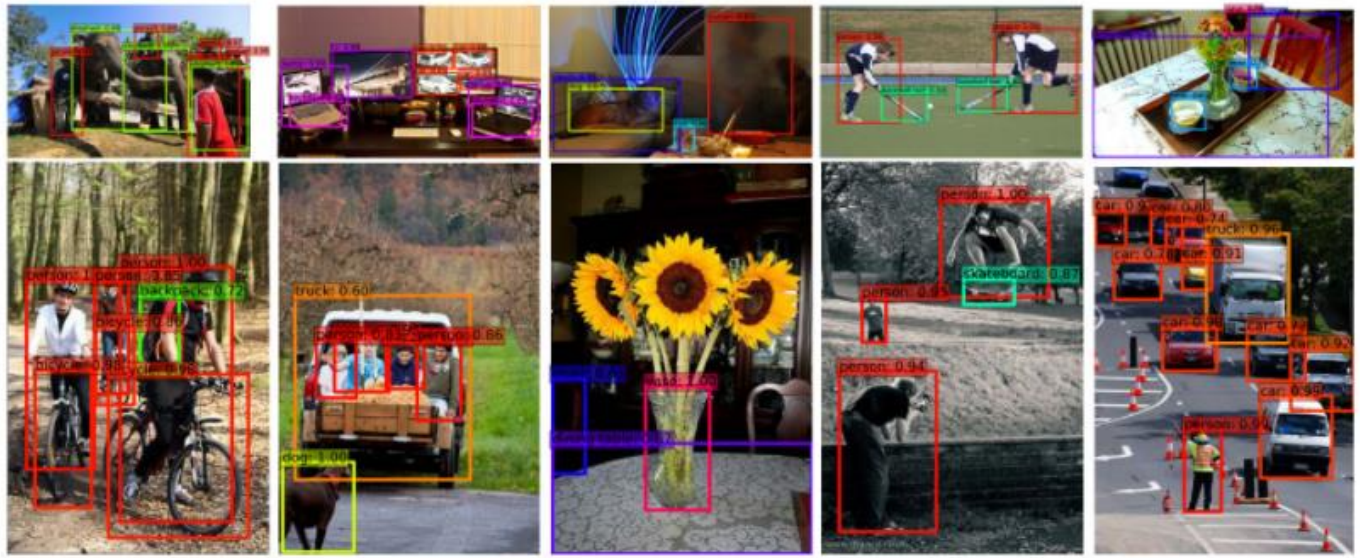
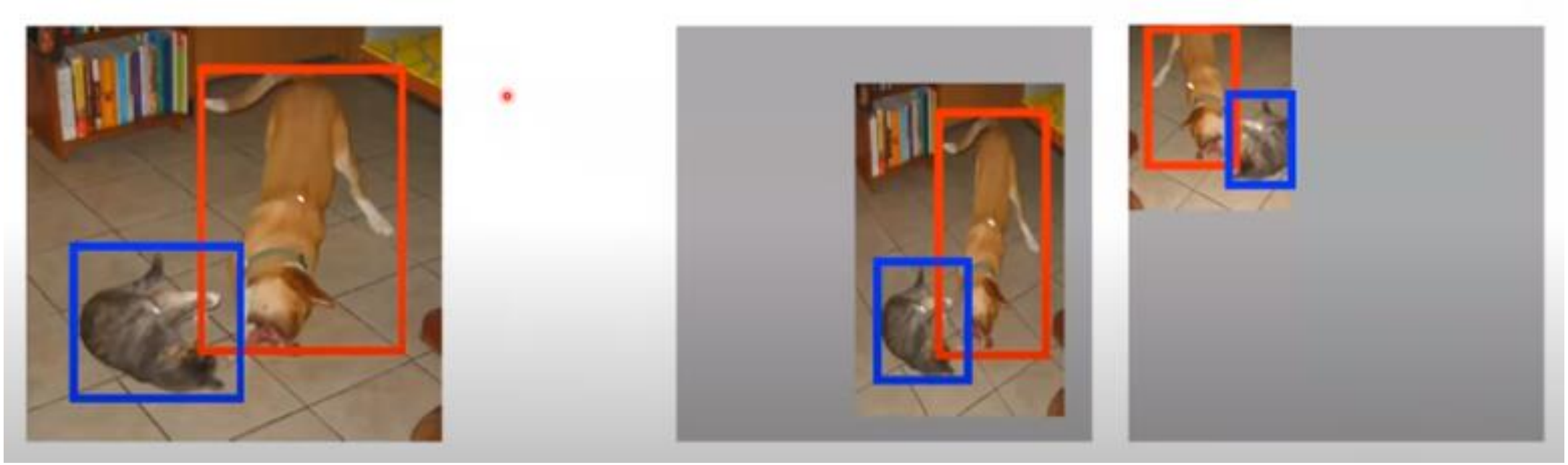


Fig. 5: **Detection examples on COCO `test-dev` with SSD512 model.** We show detections with scores higher than 0.6. Each color corresponds to an object category.

EXPERIMENTAL RESULTS – DATA AUGMENTATION FOR SMALL OBJECT ACCURACY



크기가 작은 객체에 대한 탐지 성능이 좋지 않은 것을 위와 같은 Data augmentation 방법으로 해결하고자 했다. 원본 이미지의 16배 정도 되는 캔버스에 이미지를 랜덤으로 위치시키고 이미지 이외의 지역은 이미지 픽셀 값의 평균으로 채운다.

Method	VOC2007 test		VOC2012 test		COCO test-dev2015 trainval35k		
	07+12	07+12+COCO	07++12	07++12+COCO			
	0.5	0.5	0.5	0.5	0.5:0.95	0.5	0.75
SSD300	74.3	79.6	72.4	77.5	23.2	41.2	23.4
SSD512	76.8	81.6	74.9	80.0	26.8	46.5	27.8
SSD300*	77.2	81.2	75.8	79.3	25.1	43.1	25.8
SSD512*	**79.8**	**83.2**	**78.5**	**82.2**	**28.8**	**48.5**	**30.3**

EXPERIMENTAL RESULTS – INFERENCE TIME

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

SSD에서는 많은 양의 박스들이 만들어지기 때문에 추론 시에 효율적으로 NMS를 수행하는 것이 중요하다.

Confidence threshold를 0.01로 해서 대부분의 박스를 제거하고 Best 박스와 IoU가 0.45이상인 박스들을 제거한다.

최종적으로 Top 200의 박스들만 남겨놓는다.

SSD의 키포인트:

“The use of multi-scale convolutional bounding box outputs attached to multiple feature maps at the top of the network”