

YOLO v1 – YOLO v5

(YOU ONLY LOOK ONCE)

TEAM : 카피바라

NAME : 배누리, 김호정, 전사영, 박현아

DATE : 2024. 11. 13

목 차



01 YOLO v1

02 YOLO v2

03 YOLO v3

04 YOLO v4

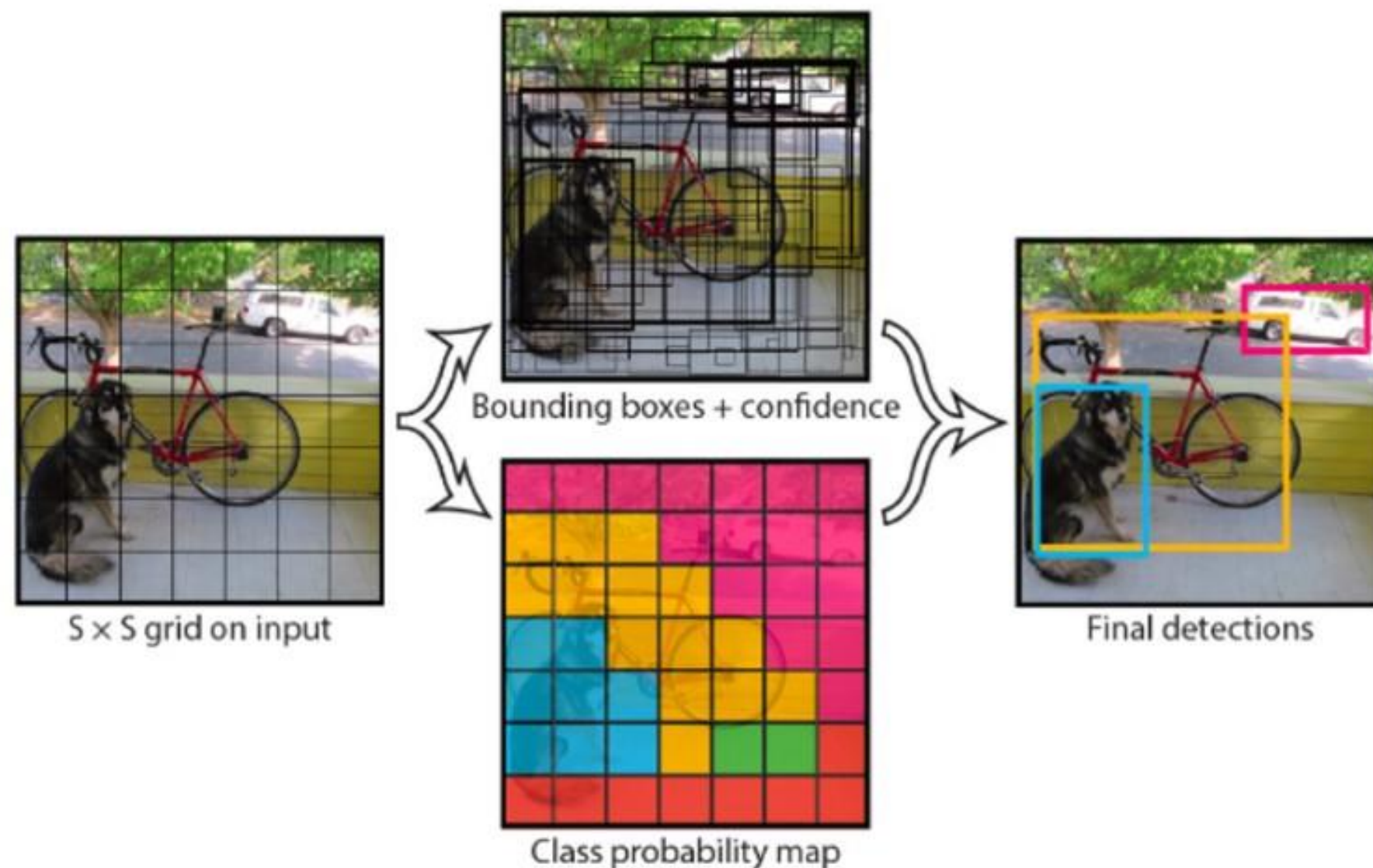
05 YOLO v1 – YOLO v5 비교

06 YOLO v1 구현



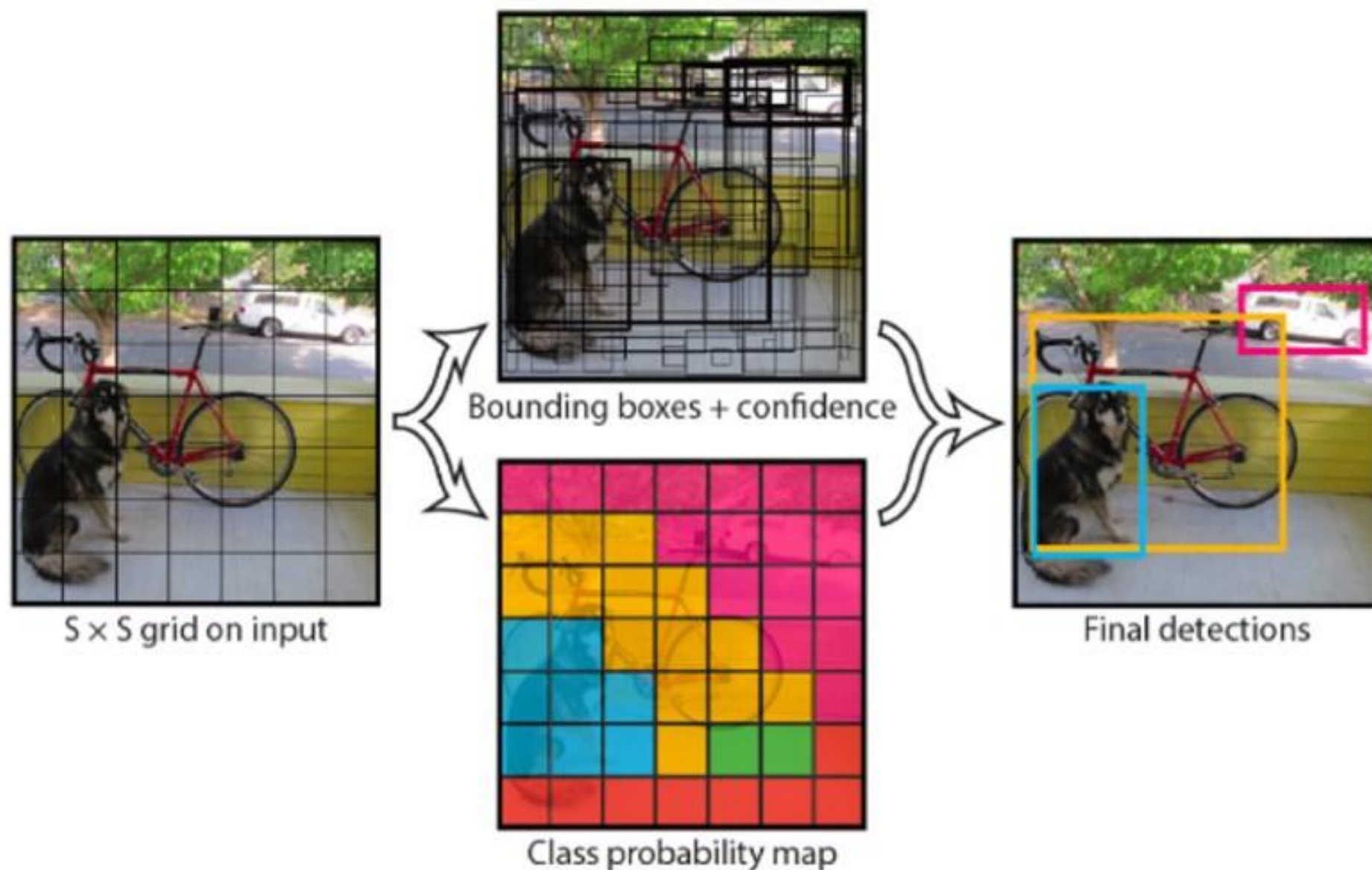
01 YOLO v1

YOLO v1



- 기존 객체 탐지 시스템들은 슬라이딩 윈도우나 region proposal 방법을 사용하여 이미지의 여러 위치에서 객체를 탐지하였음
- 이러한 방식은 처리 속도가 느리고 최적화가 어렵다는 단점이 있음.
- YOLO는 객체 탐지를 단일 신경망을 통해 단번에 수행하는 방식으로 이미지 전체를 입력으로 받아 바운딩 박스와 클래스 확률을 한 번에 예측함
- 따라서 YOLO는 통합된 단일 네트워크 아키텍처를 통해 end-to-end 학습이 가능함

YOLO v1



- Input된 이미지를 $S \times S$ 규격의 Grid cell로 나눔
- 각 Grid cell마다 **B개의 Bounding box**와 객체에 대한 **confidence score**를 예측
- 조건부 클래스 확률과 바운딩 박스 신뢰도를 곱하여, 각 박스가 특정 클래스에 속할 확률을 계산
- 출력 텐서 구조: $S \times S \times (B * 5 + C)$ 형태로 각 셀에서 여러 개의 바운딩 박스와 클래스 확률을 예측함

IoU

Confidence Score

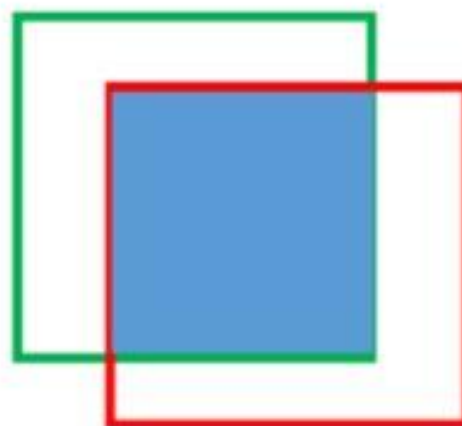
$$\text{Pr(object)} * \text{IOU}$$

		예측 Predicted Condition	
		Prediction Positive	Prediction Negative
실제 Actual Condition	Condition Positive	TP True Positive 실제 양성 예측 양성	FN False Negative 실제 양성 예측 음성
	Condition Negative	FP False Positive 실제 음성 예측 양성	TN True Negative 실제 음성 예측 음성

IOU

$B_p =$ 실제 (Gound Truth)

$B_{gt} =$ 예측 (Prediction)

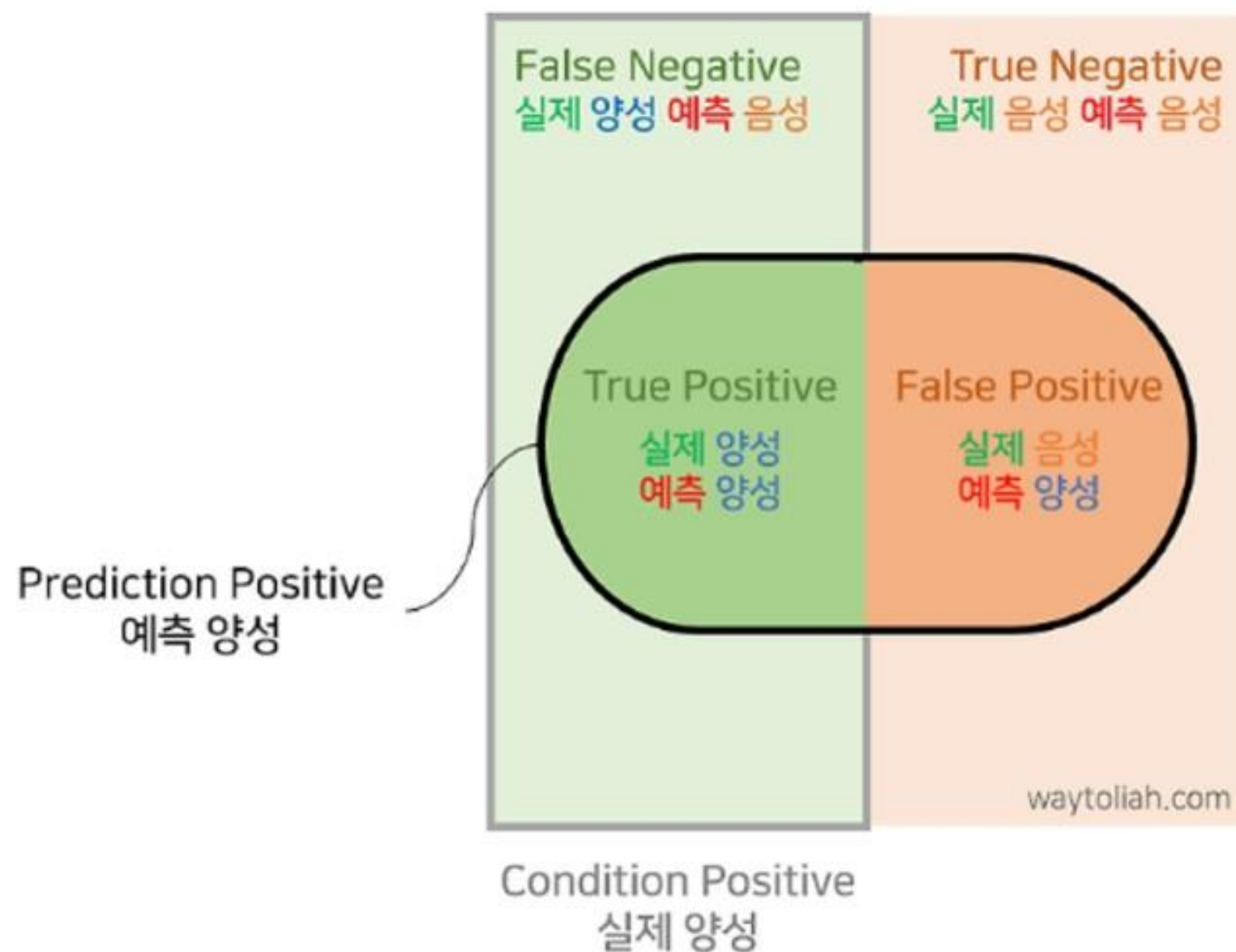


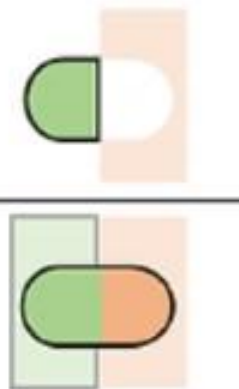
$$\text{IOU} = \frac{\text{area of overlap}}{\text{area of union}} =$$


$$= \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})}$$

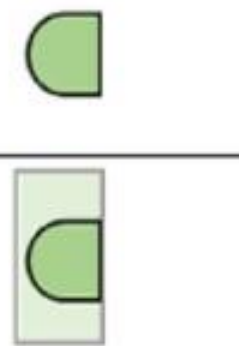
$$= \frac{\text{실제} \cap \text{예측 중복지역}}{\text{실제} \cup \text{예측 전체 영역}}$$

재현율, 정밀도

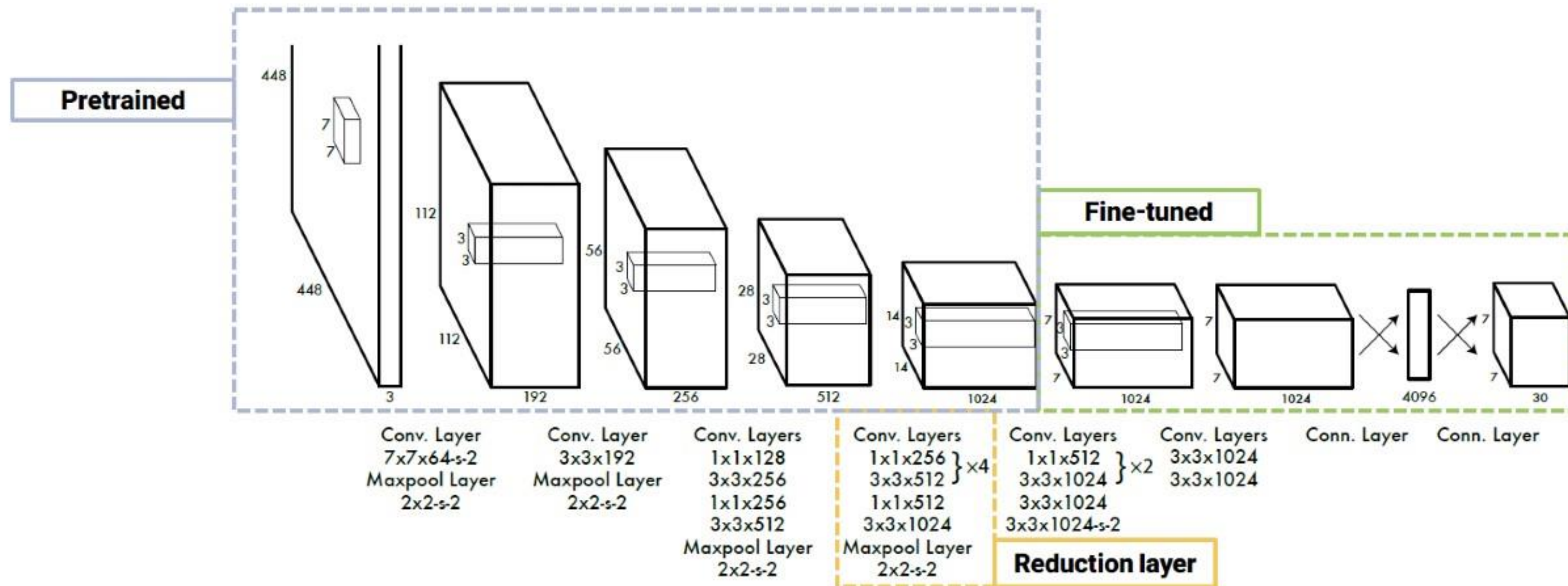


$$Accuracy = \frac{TP + TN}{\text{area of union}} = \frac{TP + TN}{TP + TN + FP + FN}$$


$$Precision = \frac{TP}{\text{all detection (Prediction Positive)}} = \frac{TP}{TP + FP}$$


$$Recall = \frac{TP}{\text{all ground truth (Condition Positive)}} = \frac{TP}{TP + FN}$$


YOLO v1 구조



- 24개의 conv layer + 2개의 fully connected layers
 - 20개의 conv layer : 사전학습된 1000-class ImageNet (input image : 224x224)
 - 4개의 conv layer + 2 fc layer : fine-tuned with PASCAL VOC (input image : 448x448)
- 1x1 필터를 사용하여 연산량 감소, 3x3 필터를 사용하여 특징 추출함

YOLO v1 손실함수

Localization Loss

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

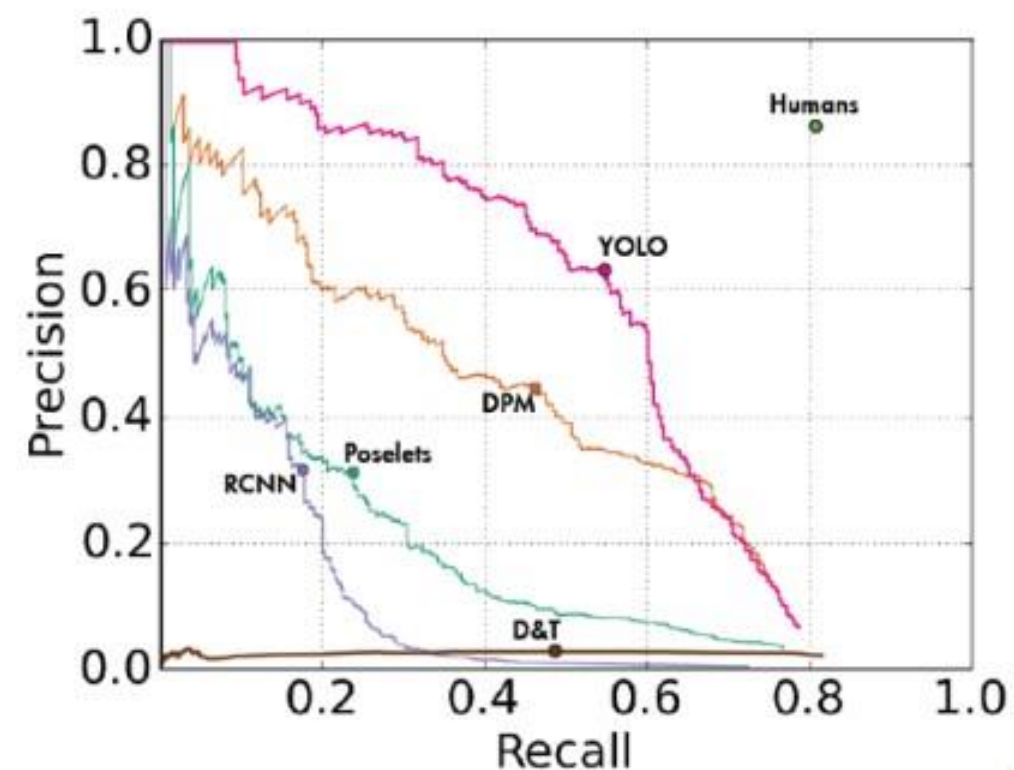
Confidence Loss

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

Classification Loss

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

YOLO v1 성능



	VOC 2007 AP	Picasso AP	Picasso Best F_1	People-Art AP
YOLO	59.2	53.3	0.590	45
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

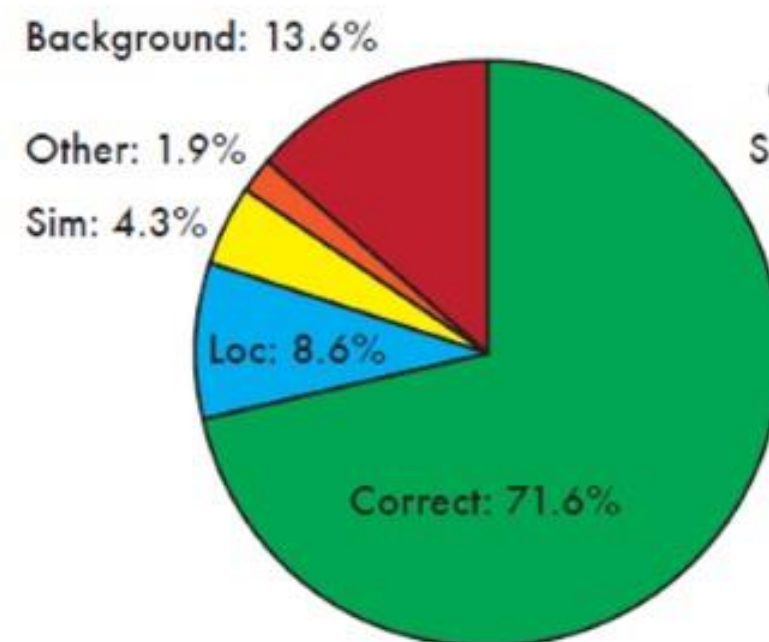


- 이미지당 45 프레임/초의 속도로 실시간 객체 탐지가 가능함. Fast YOLO 버전은 155 프레임/초까지 처리할 수 있음
- 이미지의 전체를 한 번에 보기 때문에 배경과 객체의 구분이 명확함
- 자연 이미지뿐만 아니라 예술 작품에서도 성능이 우수함

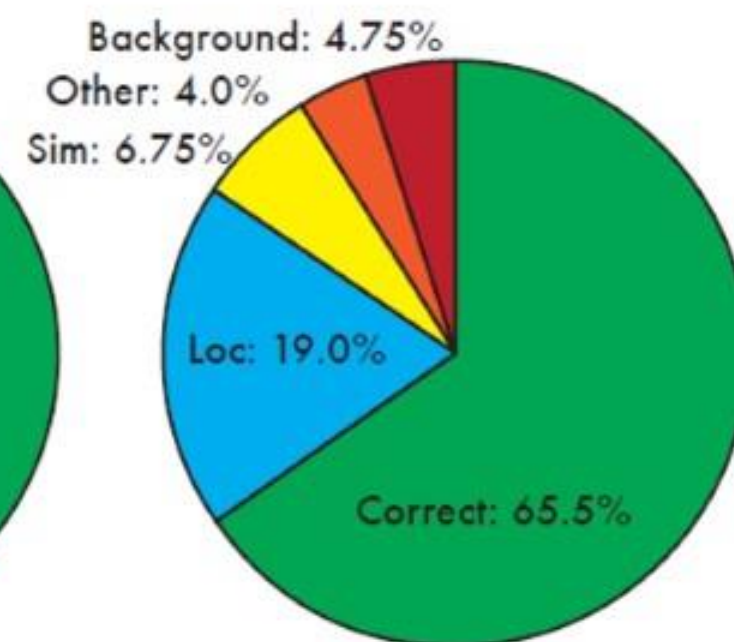
Fast R-CNN 과 YOLO v1

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

Fast R-CNN



YOLO



- PASCAL VOC 데이터셋 기준으로 기존 실시간 탐지 시스템보다 2배 이상의 mAP 성능을 보이며, Fast R-CNN보다 적은 배경 오류를 발생시킴.
- Fast R-CNN은 background error가 13.6%로, YOLO의 배경 오류 4.75%보다 훨씬 높음. (배경을 객체로 잘못 탐지하는 경우가 많다는 의미)
- 반면에 YOLO는 localization error가 19.0%로, Fast R-CNN의 8.6%보다 높음. (객체의 위치를 정확히 맞추는 데 더 어려움을 겪는다는 것을 의미)

YOLO v1 한계

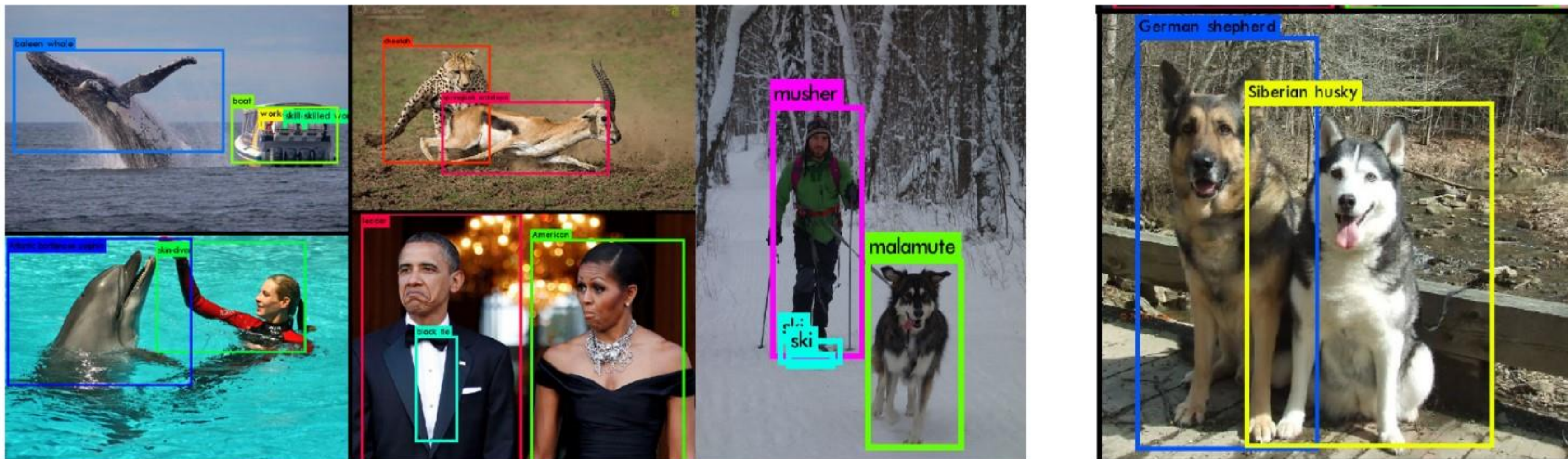


- 작은 객체의 위치를 정확히 예측하는 데 어려움이 있으며, 특히 여러 객체가 밀집된 경우 정확도가 떨어짐
- 각 그리드 셀에서 두 개의 바운딩 박스만 예측하기 때문에, 근접한 여러 객체 탐지에 한계가 있음
- Grid cell 당 하나의 클래스만 예측 할 수 있음



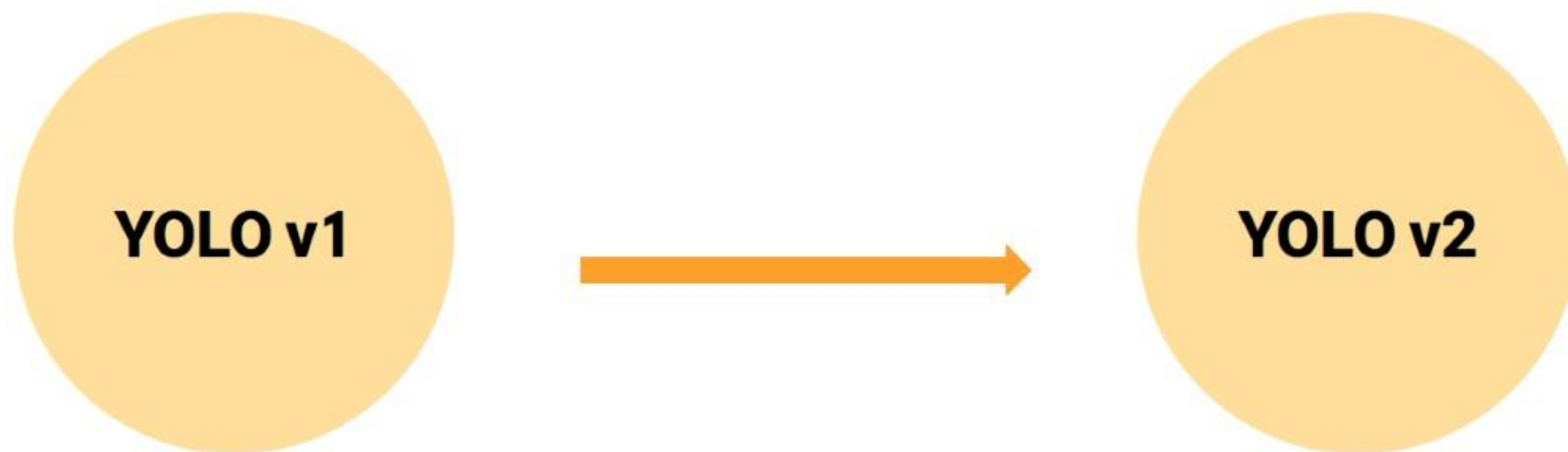
02 YOLO v2

02 YOLO v2



- YOLO v1 : VOC와 같은 데이터셋에 한정 (예: 20개 클래스)
- YOLO v2 : ImageNet을 통해 9000개 이상의 클래스 탐지 가능

02 YOLO v2



Better, Faster, Stronger

Better

Batch Normalization

- 모든 conv layer 뒤에 batch normalization을 추가함

High Resolution Classifier

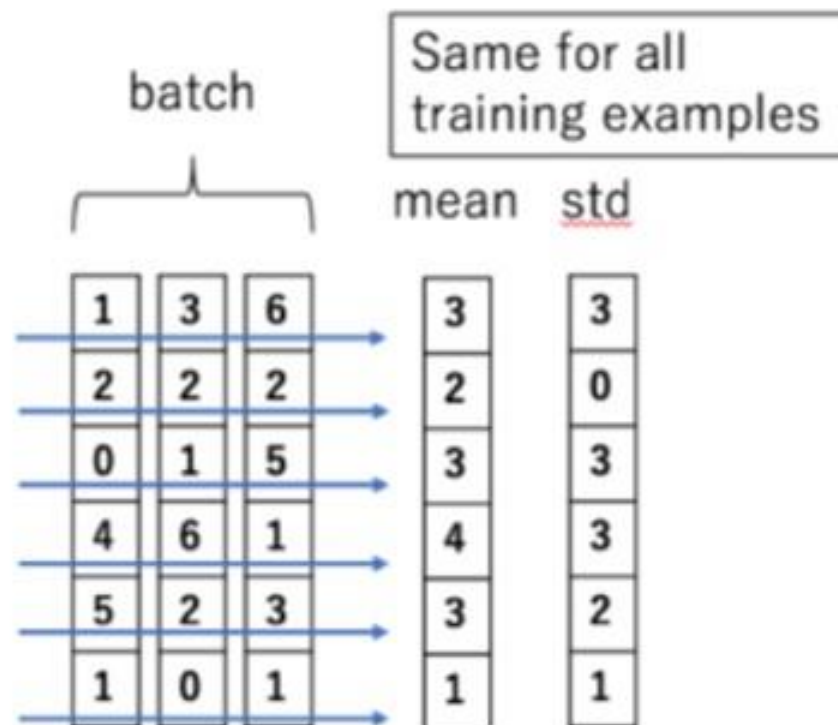
- YOLO v1 모델에서는 Darknet을 224x224 크기로 사전 학습시킨 후 detection 시 448x448 이미지를 입력으로 사용함
- YOLO v2 모델은 처음부터 Darknet을 448x448 크기로 사전 학습시켜 상대적으로 높은 해상도의 이미지에 적응할 시간을 제공함.

Convolutional with Anchor boxes

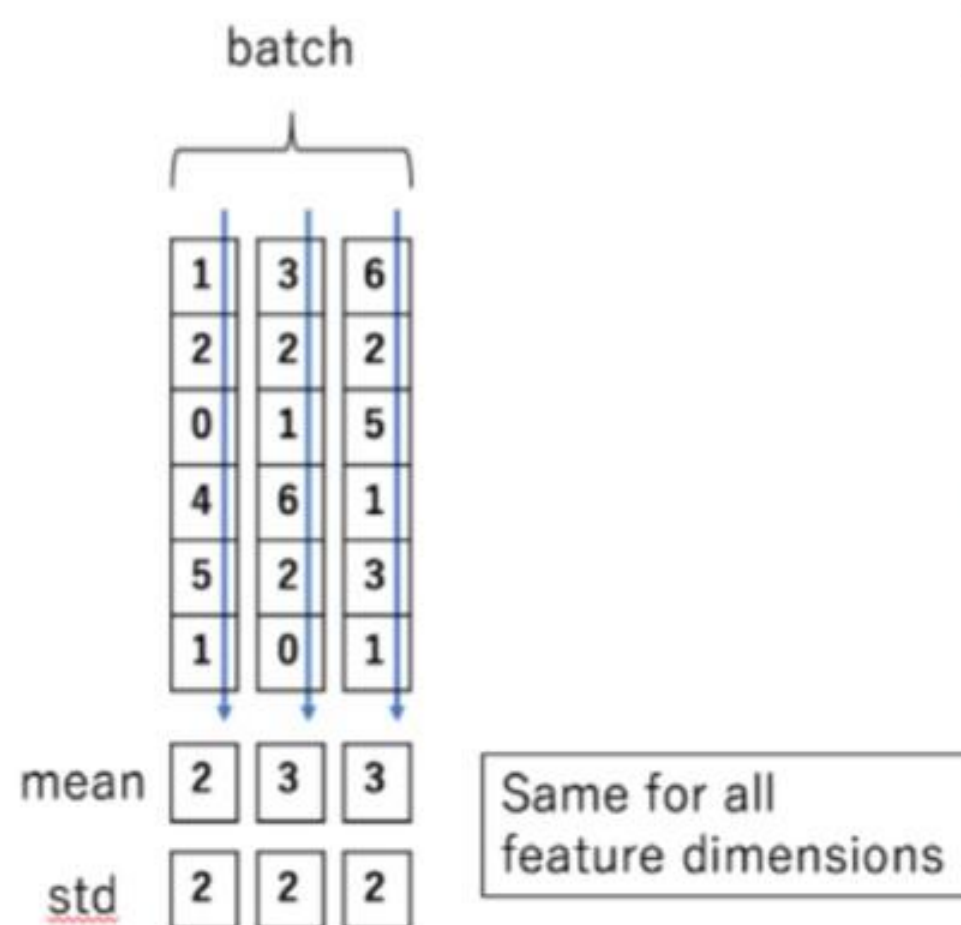
- 앵커 박스를 도입하고, Fully connected layer를 제거함
- 416x416 크기의 입력 이미지를 사용함

BN vs LN

Batch Normalization



Layer Normalization



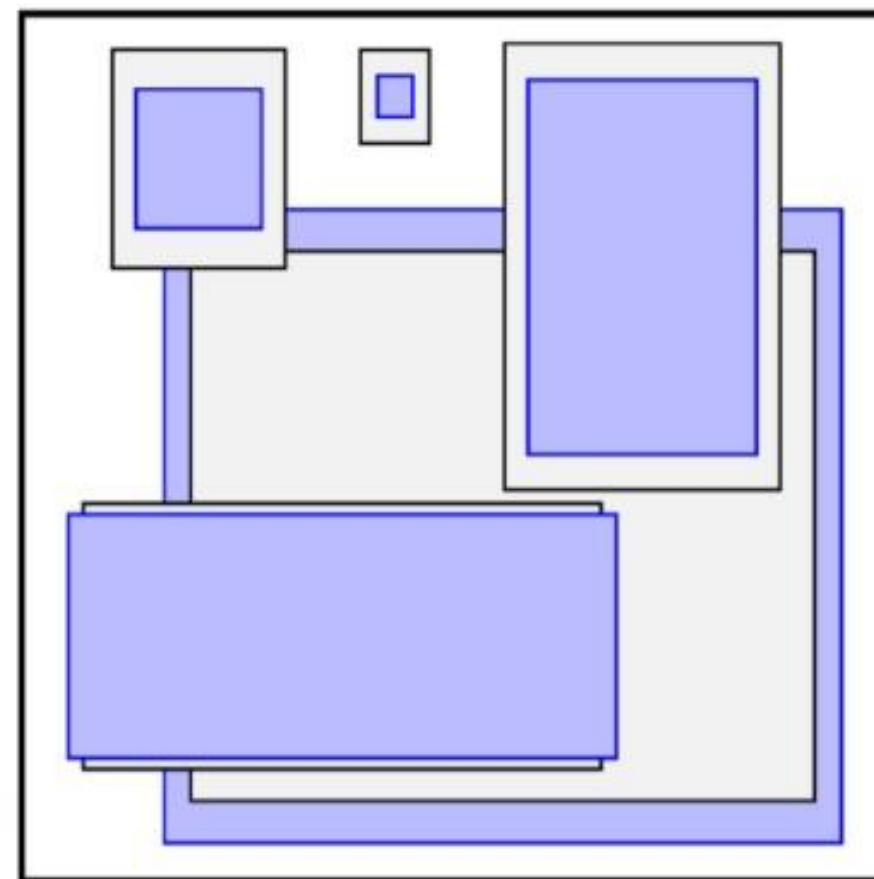
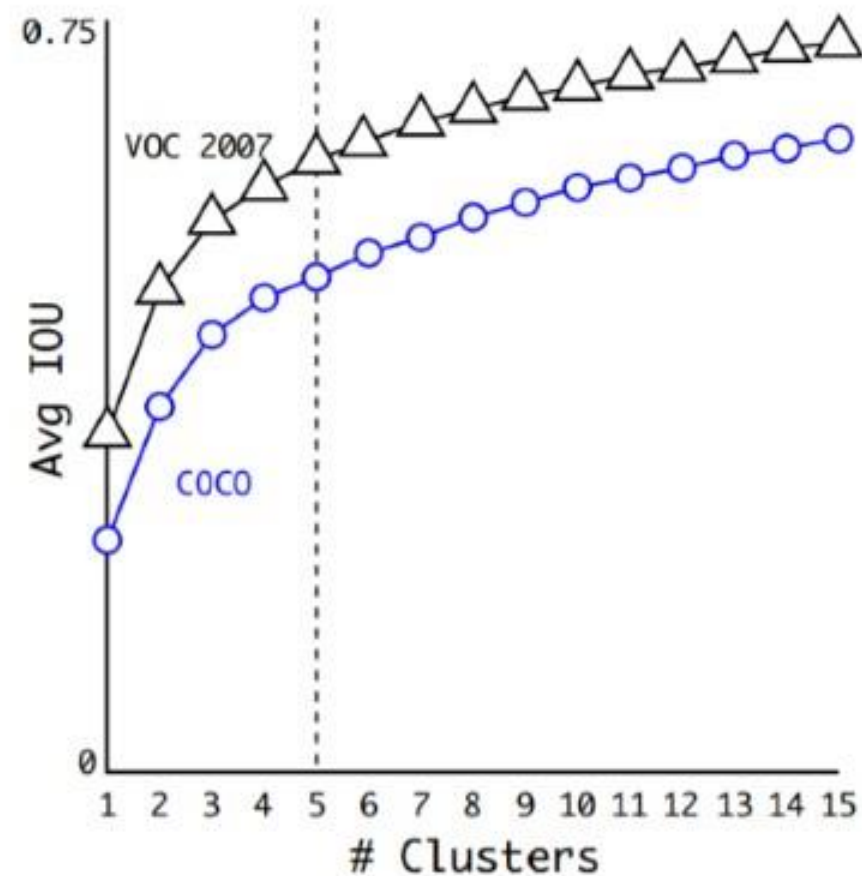
Batch Normalization (BN)

- 미니배치 단위로 정규화 수행, 배치 내의 각 채널별로 정규화
- 미니배치 크기에 의존적임

Layer Normalization (LN)

- 각 레이어의 뉴런 단위로 정규화를 수행하는 방법
- 미니배치 크기와 무관

Better - Dimension Clusters

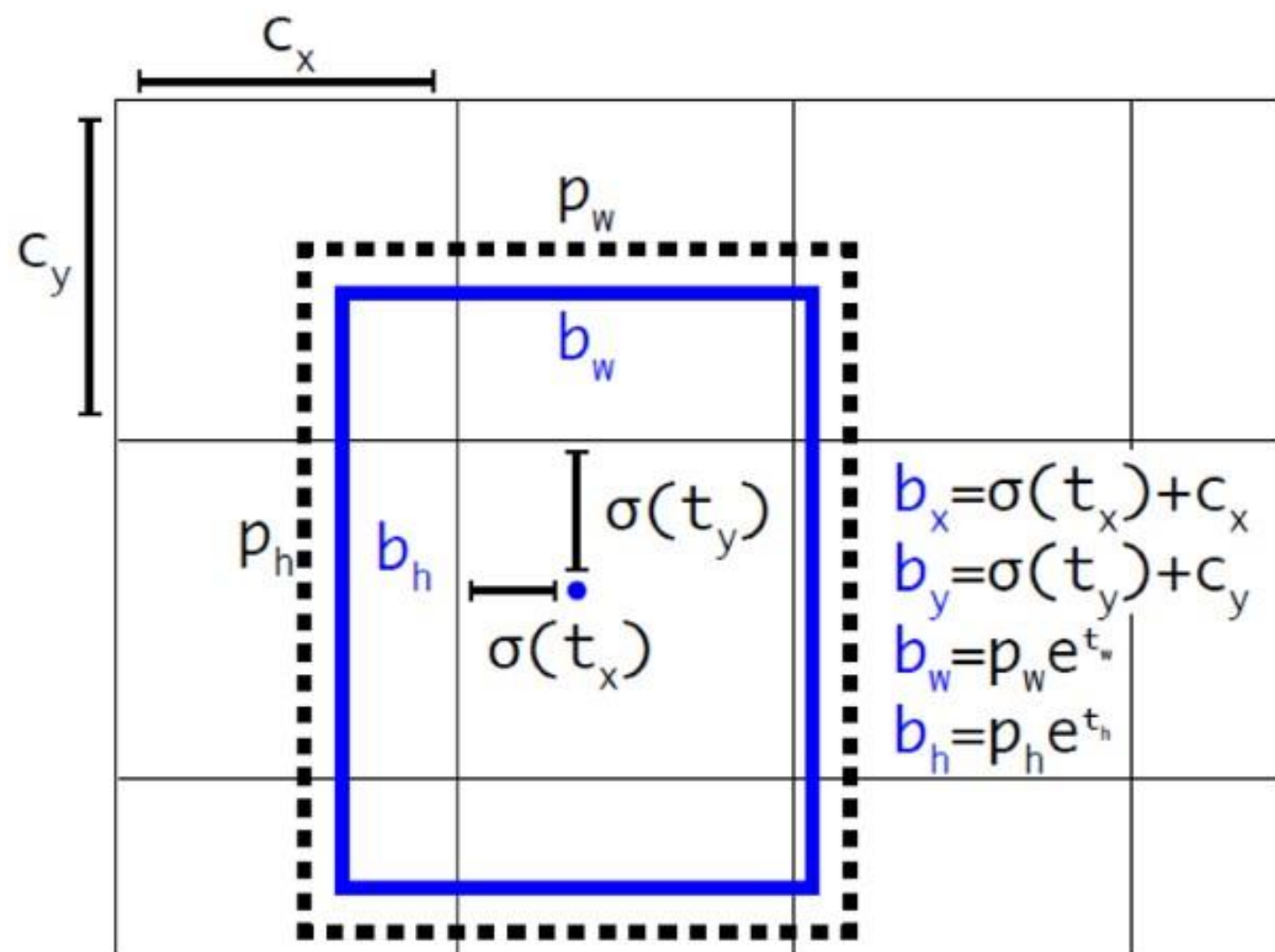


$$d(\text{box, centroid}) = 1 - \text{IOU}(\text{box, centroid})$$

- k-means clustering을 통해 최적의 앵커박스의 크기와 비율을 탐색하는 방법을 제시함
- 데이터셋에 있는 모든 ground truth box의 width, height 값을 사용하여 k-means clustering 수행

Better

Direct location prediction



$$x = (t_x * w_a) - x_a$$

$$y = (t_y * h_a) - y_a$$

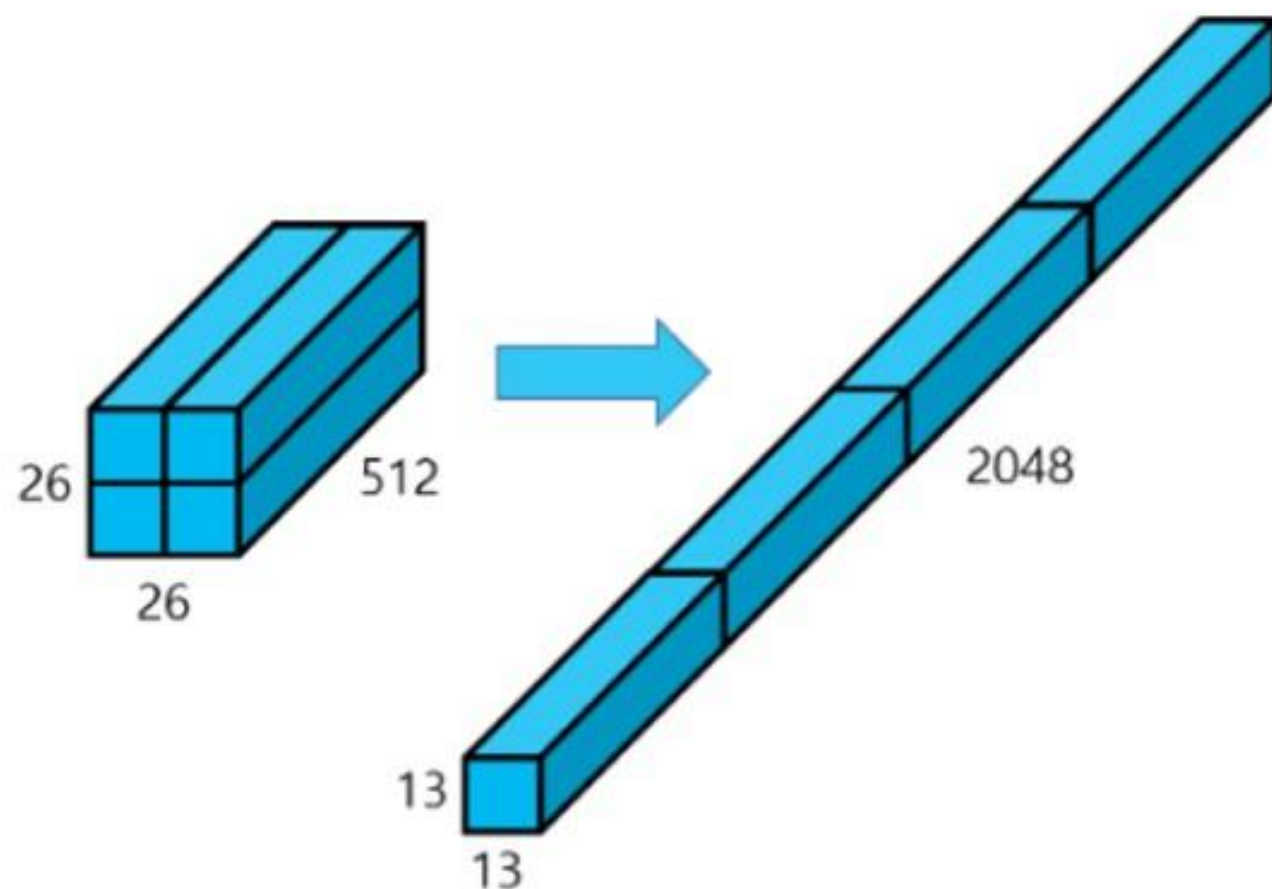
x, y : 앵커 박스를 기준으로 최종 바운딩 박스 위치를 조정하기 위한 중간 계산값

t_x, t_y : 앵커 박스를 기준으로 바운딩 박스의 위치를 조정하기 위해 사용되는 보정값

b_x, b_y : 최종 바운딩 박스의 중심 좌표

Better

Fine-Grained Features



Multi-Scale Training

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

- 작은 객체를 탐지하기 위해 중간 단계의 고해상도 feature map을 추출하여, 정보 손실을 줄이고 작은 객체의 특성을 보존함
- 26x26x512 feature map을 4개로 분할하여 13x13x2048로 변환하면, 낮은 해상도에서도 작은 객체의 정보가 유지된 채로 예측할 수 있게 됨
- 다양한 입력 이미지를 사용 (이미지를 1/32배로 다운샘플링시키기 때문에 32배수로 선택)

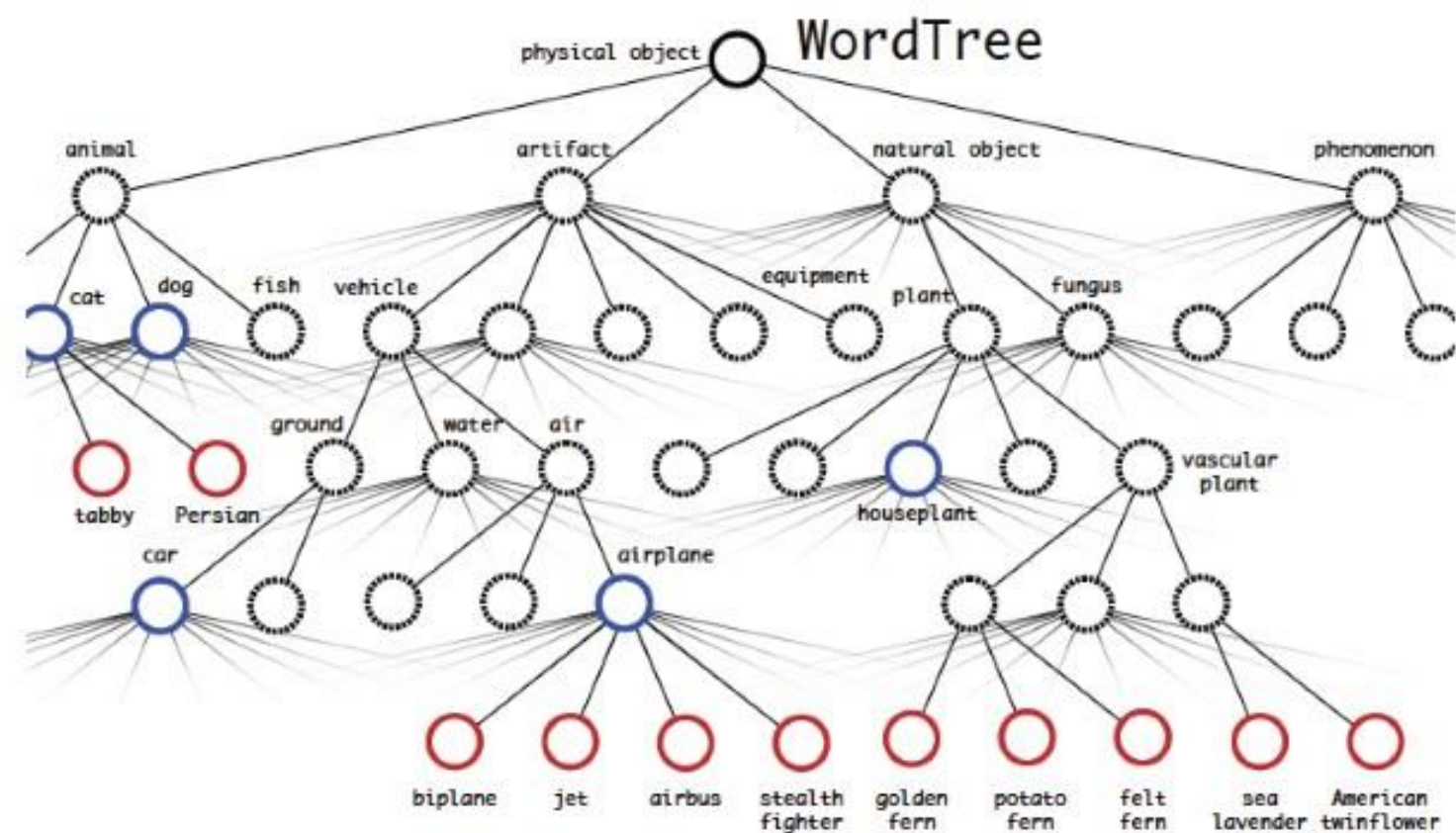
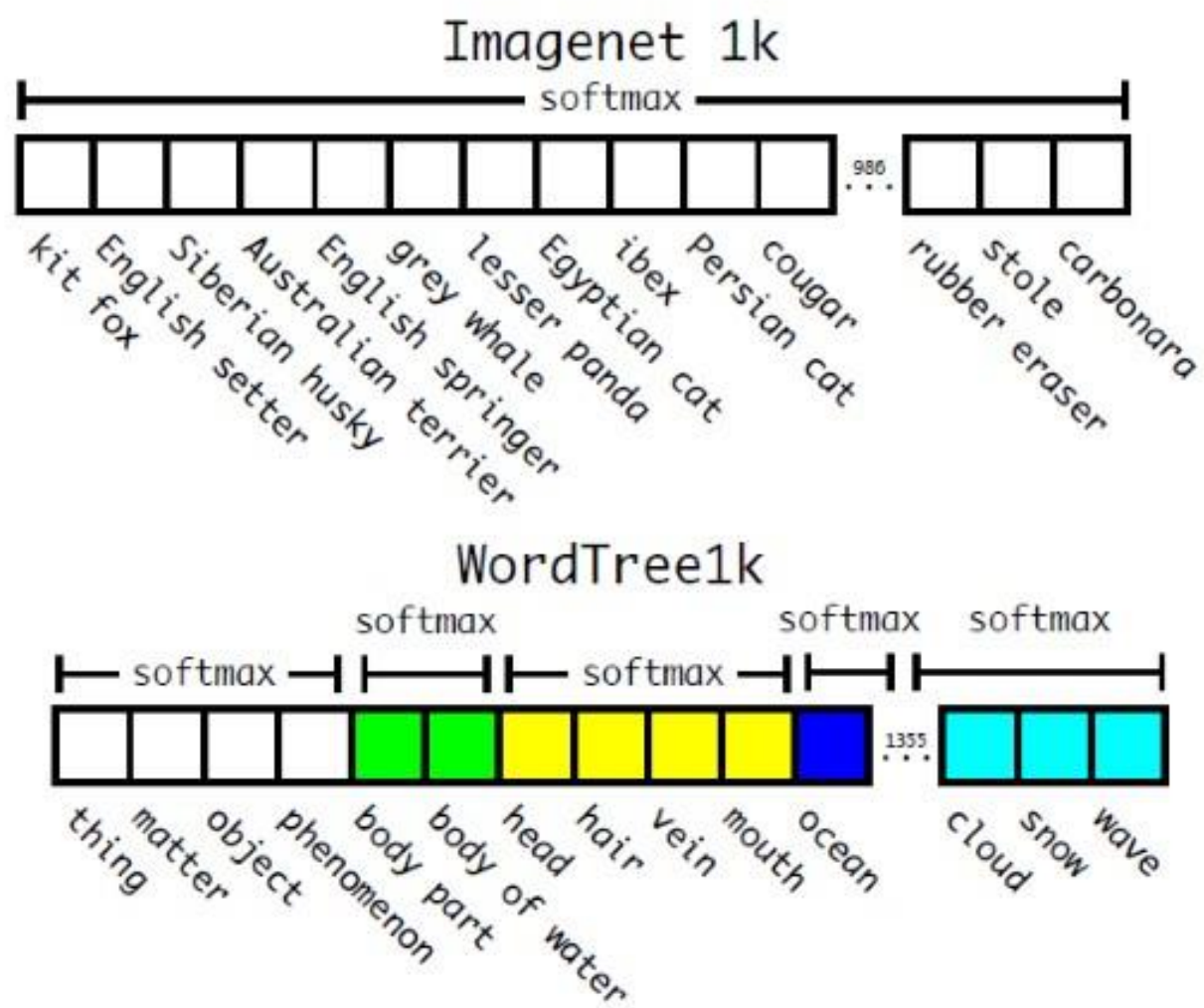
Faster

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			

Darknet-19

- Darknet-19는 마지막 layer에 global average pooling을 사용하여 fc layer를 제거하여 파라미터 수를 감소시키고, detection 속도를 향상시킴
- 각 grid cell마다 5개의 bounding box가 5개의 값(confidence score, x, y, w, h)과, PASCAL VOC 데이터셋을 사용하여 학습하기 때문에 20개의 class score를 예측함
- 따라서 1x1 conv layer에서 channel 수를 $125 (= 5 \times (5 + 20))$ 개로 지정

Stronger



- Wordtree는 클래스들 간의 계층 구조를 트리 형태로 구성하여, 다수의 클래스를 효과적으로 학습하고 예측할 수 있도록 돕는 기법
- 다중 소프트맥스(Multi Softmax) 레이어 사용 (각 계층 수준에서 별도로 소프트맥스 분포를 적용하여 클래스 예측을 수행)
- 기존 ImageNet처럼 1000개의 클래스에 한정되지 않고, 9000개 이상의 클래스를 효율적으로 탐지



03 YOLO v3

03 YOLO v3

Multiclass Classification



- Dog
- Cat
- Horse
- Fish
- Bird
- ...

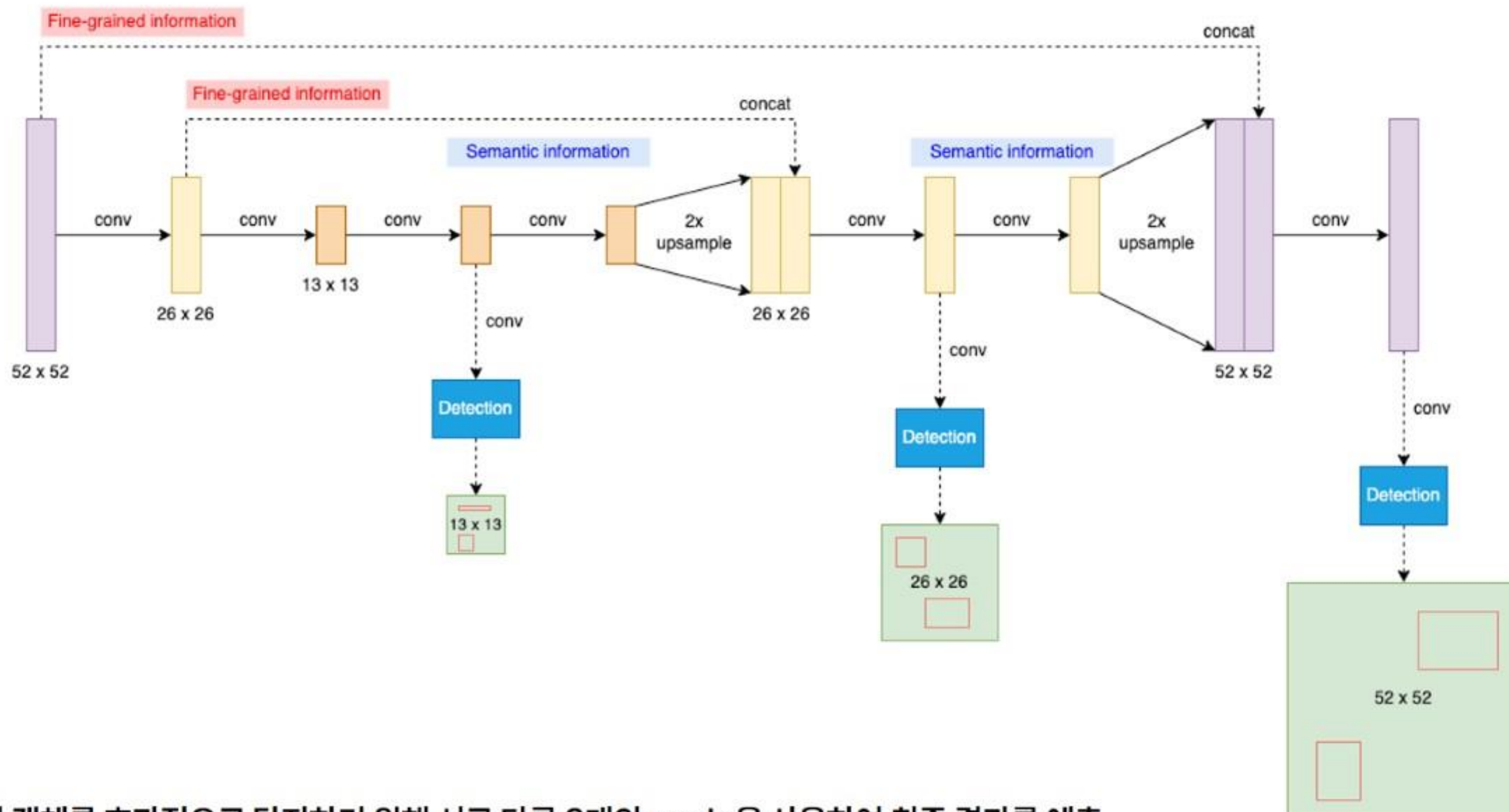
Multi-label Classification



- Dog
- Cat
- Horse
- Fish
- Bird
- ...

- multi-label classification 수행
- softmax 함수 대신 binary cross-entropy를 사용함

03 YOLO v3



- 다양한 크기의 객체를 효과적으로 탐지하기 위해 서로 다른 3개의 scale을 사용하여 최종 결과를 예측
- 이 때 각 scale의 feature map의 output channel 수가 $[3 \times (4 + 1 + 80)] (=255)$ 이 되도록 마지막 1x1 conv layer의 channel 수를 조정

Darknet-53

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

Darknet-53

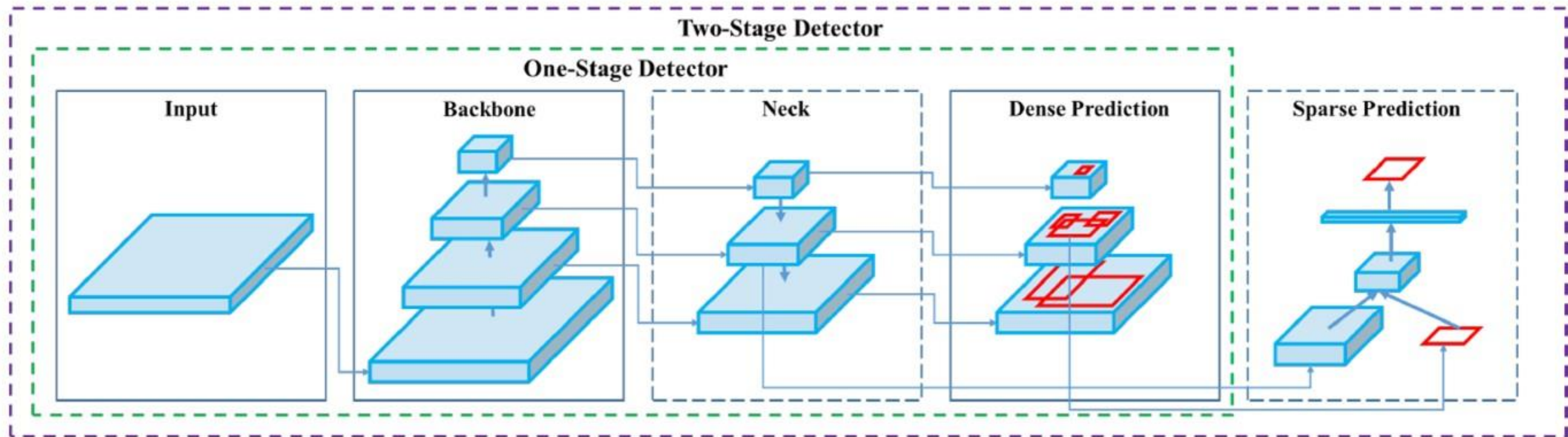
Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	171
ResNet-101 [5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	77.6	93.8	29.4	1090	37
Darknet-53	77.2	93.8	18.7	1457	78

- shortcut connection이 추가되어 53개의 layer를 가지는 Darknet-53을 backbone network로 사용
- ResNet-101보다 1.5배 빠르며, ResNet-152와 비슷한 성능을 보이지만 2배 이상 빠름



04 YOLO v4

04 YOLO v4



기존에는 ImageNet에서 pre-trained된 backbone과 class와 bounding box를 예측하는 head로 구성되었으나, backbone과 head 사이에서 서로 다른 scale의 feature map을 수집하는 layer인 neck이 추가되었음

YOLO v4 = YOLO v3 + CSPDarknet53 + SPP + PAN + BoF + BoS

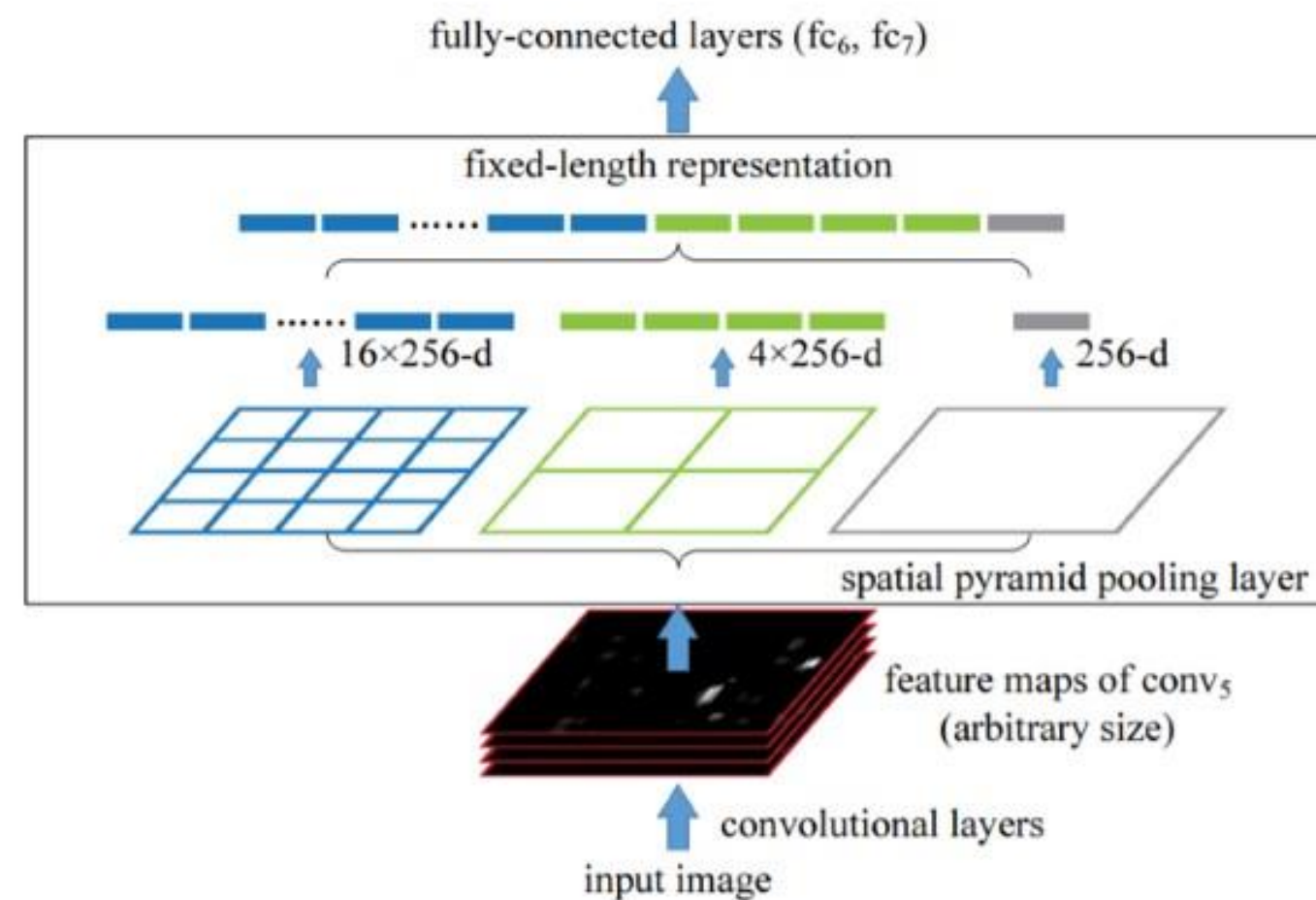
04 YOLO v4

• SPP (Spatial Pyramid Pooling)

- 다양한 크기의 Pooling을 통해 고정된 크기의 feature map 생성
- 입력 이미지의 크기와 무관하게 항상 동일한 크기의 feature map을 생성하므로, Fully Connected Layers와 결합이 용이

• PAN (Path Augmented Network)

- Bottom-up Path Augmentation을 통해 low-level feature의 정보를 high-level feature에 효과적으로 전달함으로써 객체 탐지 시 localization 성능을 향상시킨 네트워크



04 YOLO v4

• Bag of Freebies (BoF)

- 추가적인 추론 비용을 유발하지 않으면서 성능을 높이는 학습 기법
- Data augmentation
- Semantic distribution bias 해결: 클래스 간 데이터의 불균형 해결
- Objective function of Bounding box regression : IoU loss 사용하여 바운딩 박스의 정확한 위치 예측

• Bag of Specials (BoS)

- 추론 시 약간의 계산 비용을 추가하면서 성능을 높이는 기법
- receptive field 확장 : 모델이 더 넓은 영역을 보며 객체 탐지
- attention 기법 활용 : SAM 모델을 사용하여 특정 공간 영역에 대해 가중치를 부여하고 그 영역에 집중하게 함
- feature integration : PAN을 사용하여 저해상도 feature map과 고해상도 feature map의 세부 정보 결합

YOLO v4 추가 기법



aug_-319215602_0_-238783579.jpg



aug_-1271888501_0_-749611674.jpg



aug_1462167959_0_-1659206634.jpg



aug_1474493600_0_-45389312.jpg



aug_1715045541_0_603913529.jpg



aug_1779424844_0_-589696888.jpg

- **Mosaic**

- 네 개의 학습 이미지를 섞는 Data Augmentation 방법

- 데이터셋에서 무작위로 4개의 이미지를 선택 → 4개의 이미지를 불규칙한 위치에서 자르고, 이들을 하나의 이미지로 합침

- **SAT (Self-Adversarial Training)** : Forward, Backward 2번의 stage를 걸쳐 수행되는 Data Augmentation 방법

05 YOLO v1 – YOLO v5 비교

YOLO v1 – YOLO v5 비교

	주요 특징	개선 사항
YOLO v1	단일 신경망을 통한 객체 탐지, 속도는 빠르지만 정확도가 낮음	단일 프레임워크로 객체 탐지 문제 해결 시도
YOLO v2	Anchor Boxes, Batch Normalization, Multi-Scale Training 도입	작은 객체 탐지 성능 개선 및 다중 해상도 학습
YOLO v3	Darknet-53 백본, 다중 스케일 예측, Binary Cross-Entropy 사용	다양한 객체 크기에 대한 탐지 성능 개선
YOLO v4	CSPDarknet53, SPP, PAN, BoF, BoS, 다양한 데이터 증강 기법 도입	정확도와 속도, 학습 효율성을 동시에 개선 일반 GPU에서도 효율적으로 학습 가능하도록 개선
YOLO v5	PyTorch 기반, 경량화 모델 제공(YOLOv5s, YOLOv5m 등), AutoAnchor 도입	실용성을 높이고 다양한 모델 크기 제공



06 YOLO v11 구현

06 YOLO v11 구현

```
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt

# YOLOv11 모델 불러오기
model = YOLO('yolov5/yolo11x.pt')

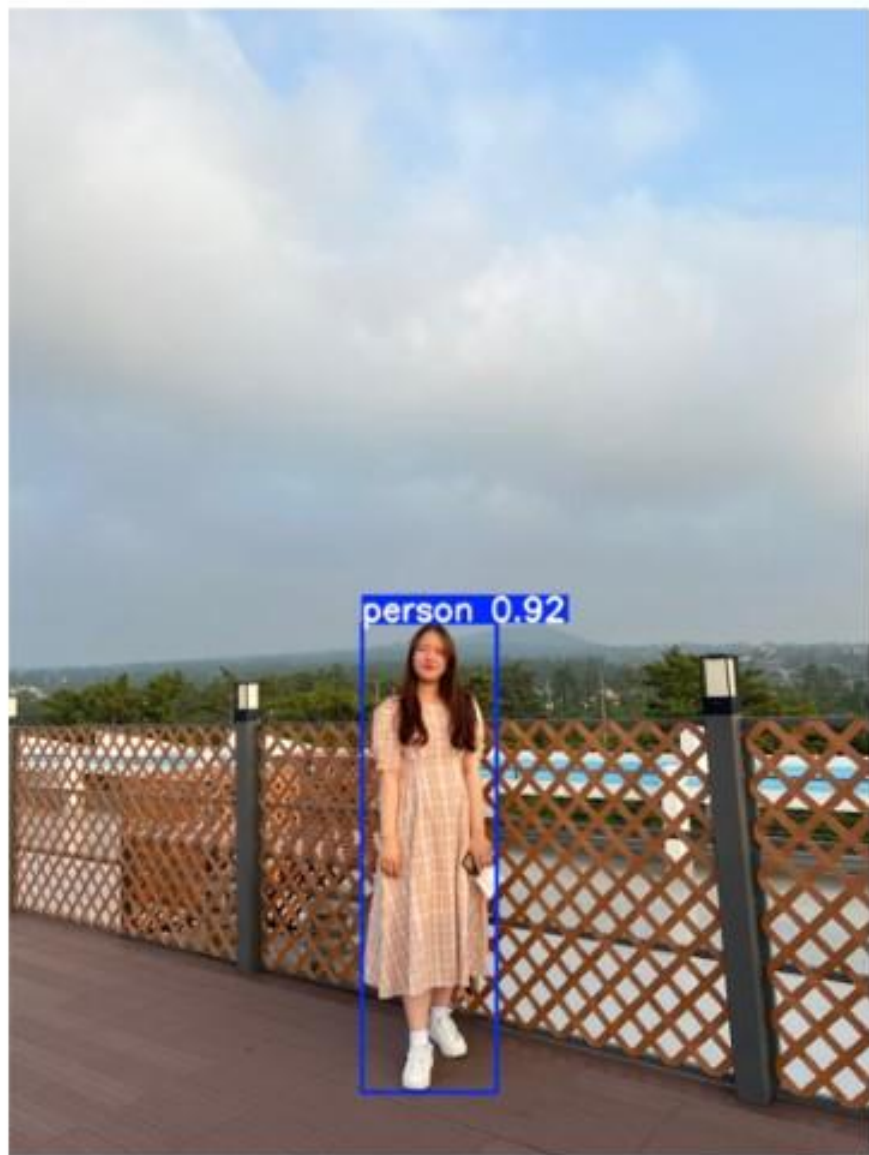
# 이미지 불러오기
image_path = 'C:/Users/Pictures/YOLO/사진1.jpg'
image = cv2.imread(image_path)

# 객체 탐지 수행
results = model(image)

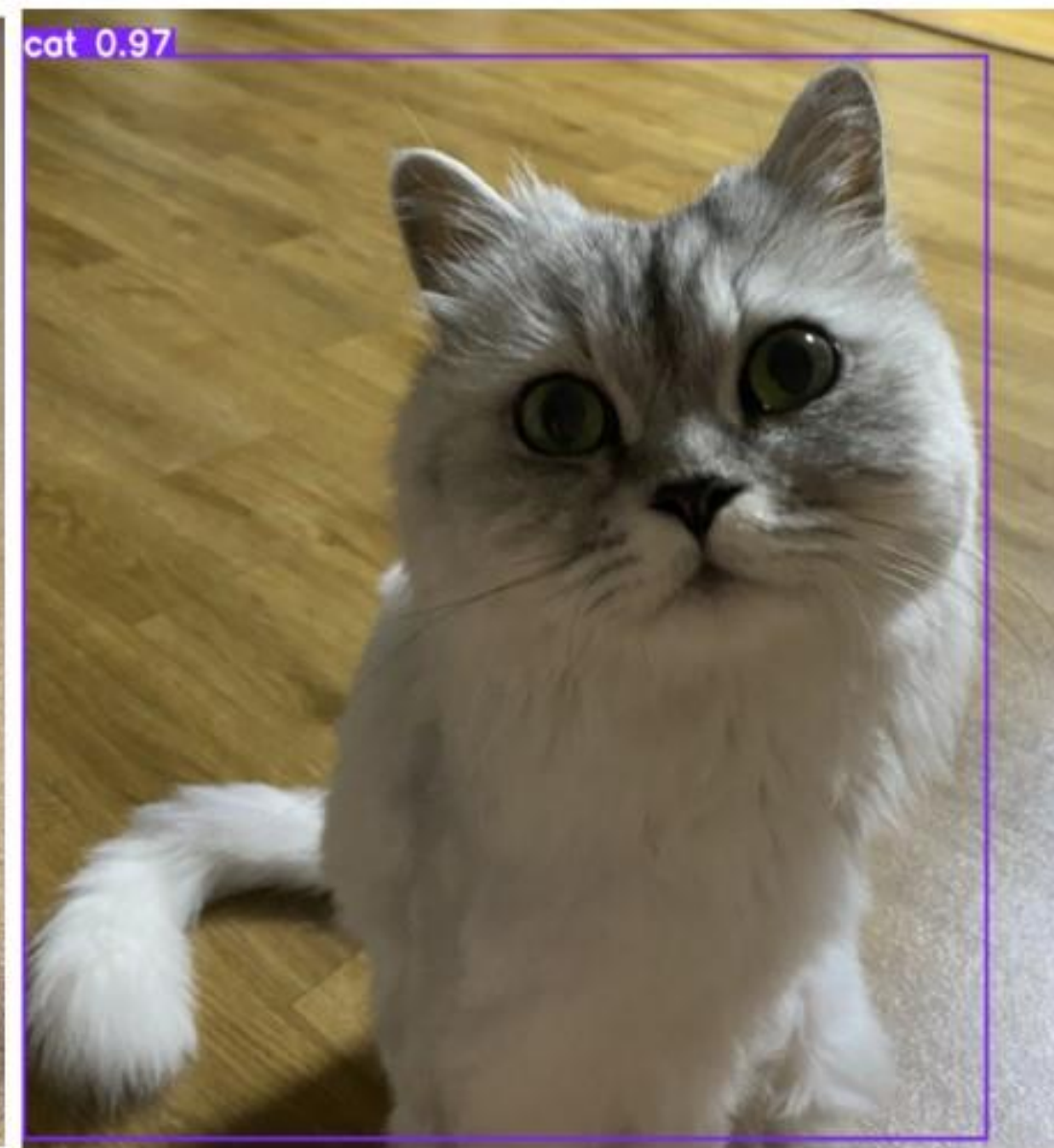
# 결과 시각화
annotated_image = results[0].plot()

# 이미지 표시
plt.imshow(cv2.cvtColor(annotated_image, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```


YOLO v11 결과 - 사람



YOLO v11 결과 - 동물



THANK YOU
Q & A