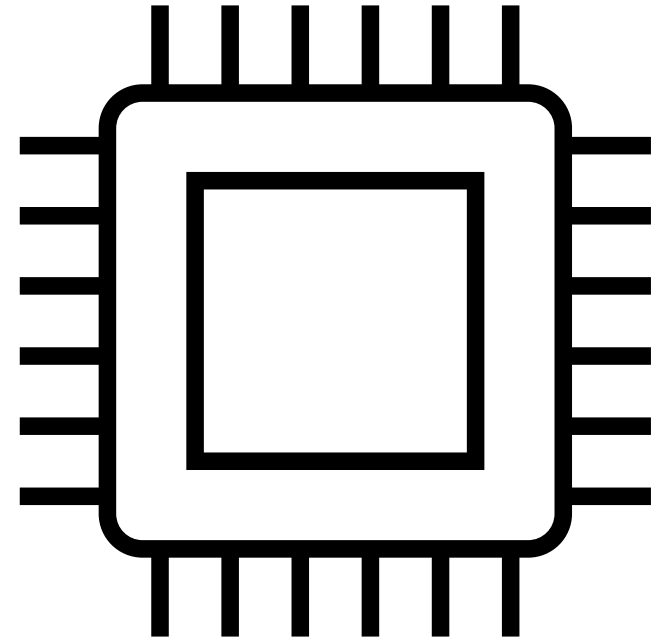


Object Detection (객체 탐지)

2025.07.08.

Copyright©2024 by 고재균



High-Level Vision (객체 탐지) [1]

• Convolutional Neural Network for Object Detection

▪ Definition

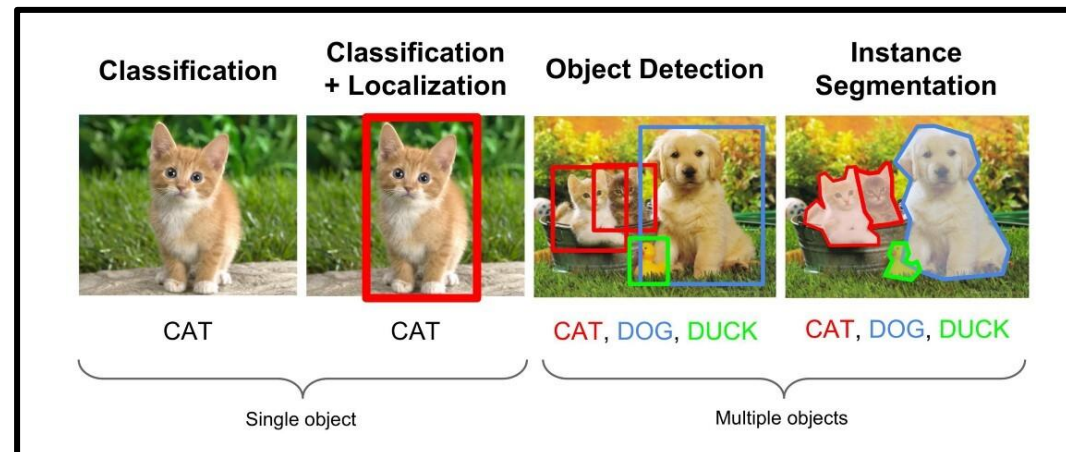
▶ 이미지 내 객체와 객체를 둘러싸는 가장 작은 직사각형으로 정의 되는 Bounding Box를 찾는 Task

▶ Image Classification / Image Localization / Object Detection 비교

▷ Image Classification → 하나의 Object에 대해 무엇인지를 분류

▷ Image Localization → 하나의 Object에 대해 어디에 위치해 있는지 파악

▷ Object Detection → 여러가지 물체에 대해서 위치를 파악하고, 각각이 무엇에 해당하는지를 파악



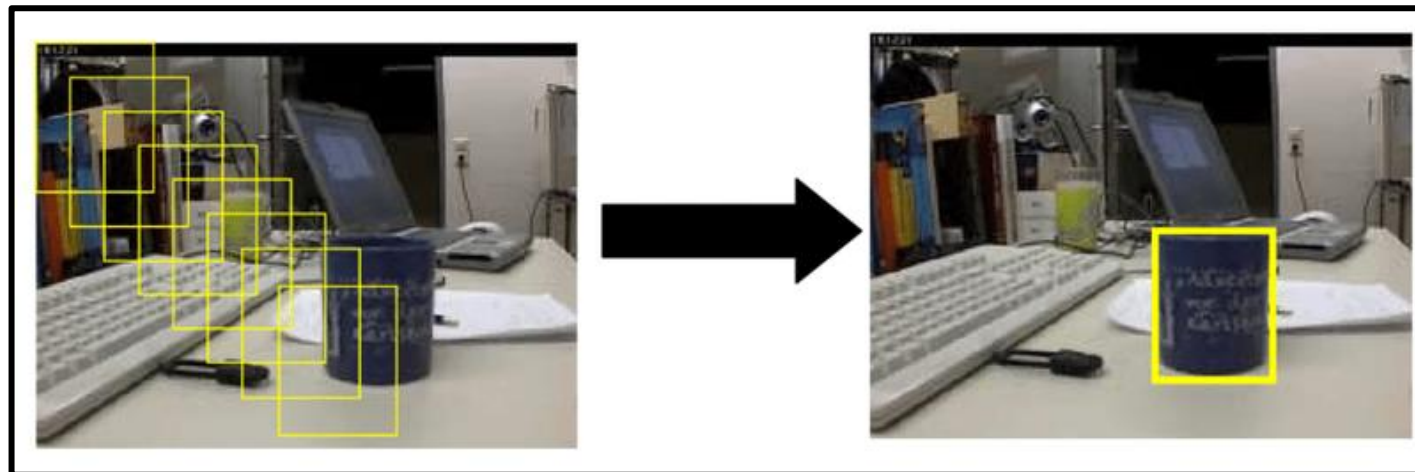
High-Level Vision (객체 탐지) [2]

• Convolutional Neural Network for Object Detection

▪ Localization

▶ Sliding Window

- ▷ 모든 Location에 대해 Scale을 고려하여 Object Candidate을 만드는 방법
- ▷ 임의의 크기를 가진 Patch (Window)가 Pixel-wise로 이미지 전체를 순회
- ▷ 크기가 작은 Object를 검출하기 위해 Patch Size를 변경해가며 순회
- ▷ Pixel-wise로 이미지 전체를 순회하며 Object를 검출하기에는 너무 많은 계산 비용이 요구



High-Level Vision (객체 탐지) [3]

- Convolutional Neural Network for Object Detection

- Localization

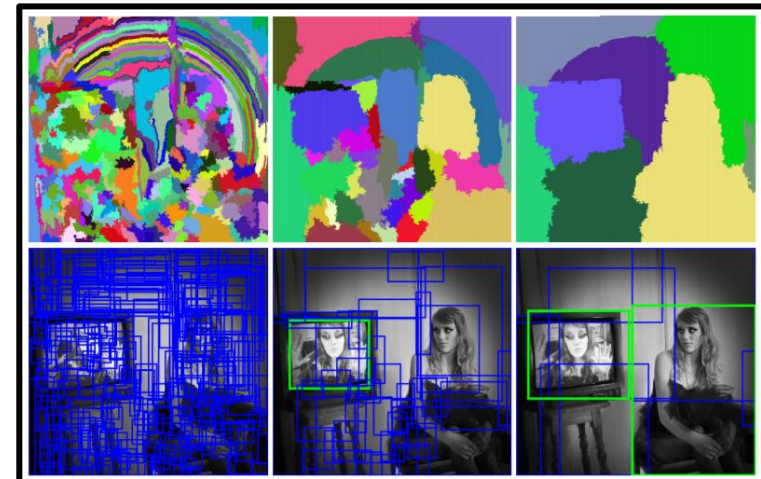
- ▶ Selective Search

- ▷ Sliding Window의 단점을 보완하기 위한 새로운 방법

- ▷ Region of Interest (RoI)를 찾아내어 Object가 존재할 만한 영역에서만 Classification을 진행

- ▷ Selective Search 알고리즘은 다음과 같은 방식으로 RoI를 탐색

1. 주변 Pixel 간 유사도를 평가 (계산)
2. 비슷한 영역을 합쳐 Segmented (Clustered) Area를 형성
3. 기존 Area를 합쳐 더 큰 Segmented Area를 형성
4. 2번 및 3번 과정을 반복하여 최종적으로 2000여개의 RoI를 형성



High-Level Vision (객체 탐지) [4]

- **Convolutional Neural Network for Object Detection**

- Localization

- ▶ Region Proposal Method (RPN) (1)

- ▷ Selective Search 또한 딥 러닝 모델에 적용해 객체 검출을 하기에는 많은 시간이 요구됨
 - ▷ 따라서, Anchor Box라는 개념에 기반한 Region Proposal Method (RPN)방법을 제시
 - ▷ Anchor Box → Bounding Box가 될 후보군으로, 미리 정해 둔 크기와 비율을 가짐
 - ▷ 추후 학습을 통해 Bounding Box가 아닐 것 같은 Anchor Box는 Non-Maximum Suppression (NMS)을 통해 후보군에서 탈락
 - ▷ 그리고 가능성 있는 Anchor Box에 대해서만 Regression 하여 최종 Bounding Box를 계산
 - ▷ 주로 2-Stage OD 모델들이 RPN을 사용

High-Level Vision (객체 탐지) [5]

- Convolutional Neural Network for Object Detection

- Localization

- ▶ Region Proposal Method (RPN) (2)

- ▷ RPN은 다음과 같은 방식으로 작동됨

1. 특정 비율과 크기를 가지는 $k(=9)$ 개의 Anchor Box를 설정
2. 입력 이미지를 Pretrained CNN (Backbone)에 넣어 $H \times W \times C$ 크기의 특징 맵을 추출
3. 출력 특징 맵에 대해 3×3 합성곱 적용 후 $H \times W \times 256$ 크기의 중간 특징 맵을 추출
4. 중간 특징 맵에 대해 1×1 합성곱 적용 후 Classification과 Bounding Box Regression을 각각 수행
5. 앞서 얻은 값들로 최종 Bounding Box를 찾는 과정을 진행
6. 우선 NMS를 통해 Object가 있을 확률이 높은 N 개의 Anchor Box를 추출
7. 이후 후 학습을 통해 Classification과 Bounding Box Regression을 동시에 End-to-End로 진행

- ▷ Selective Search가 2000개의 RoI를 제시하는데 반해 RPN은 800개 정도의 RoI를 제시

- ▷ CPU에서 작동하는 Selective Search와는 달리 RPN은 GPU에서 작동

High-Level Vision (객체 탐지) [6]

• Convolutional Neural Network for Object Detection

▪ Localization

▶ Region Proposal Method (RPN) (3)

▷ Classification Layer

- ◆ 중간 특징 맵의 특정 지점을 중심으로 하는 Anchor Box 9개를 그려보고, 이 Anchor Box에 Object가 있는지 없는지 판별 (Binary Classification)
- ◆ 그 결과 채널 수는 $2 \times k (=9)$ 가 되고 최종 특징 맵은 $H \times W \times k \times 2$ 의 크기를 가짐
- ◆ 이 값들을 적절히 Reshape 해준 다음 Softmax를 적용하여 해당 Anchor Box에 Object가 있을 확률 값을 계산

▷ Bounding Box Regression Layer

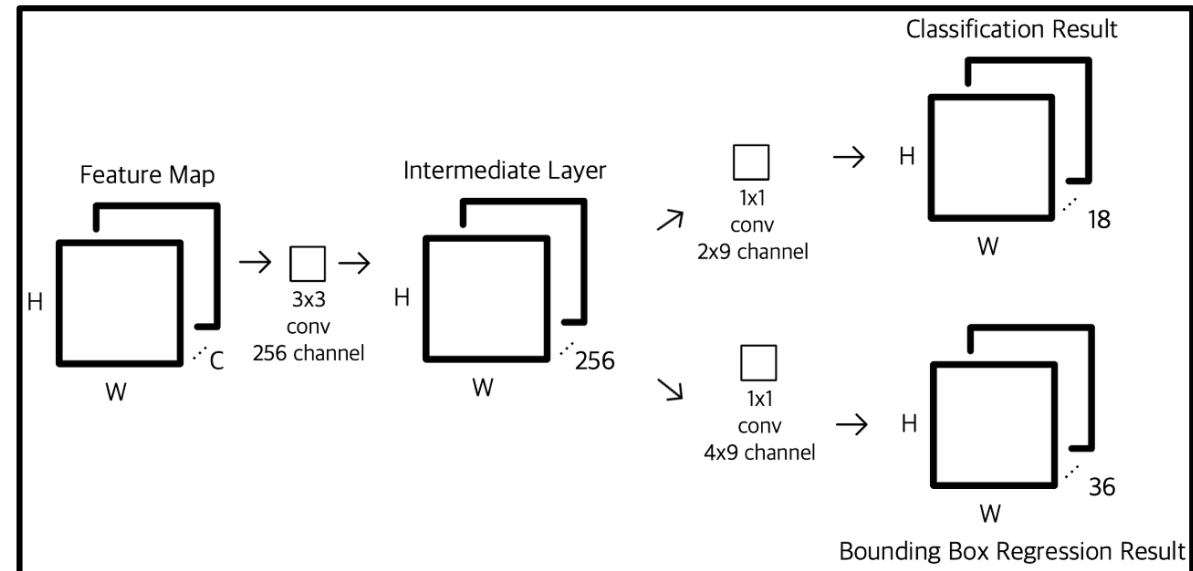
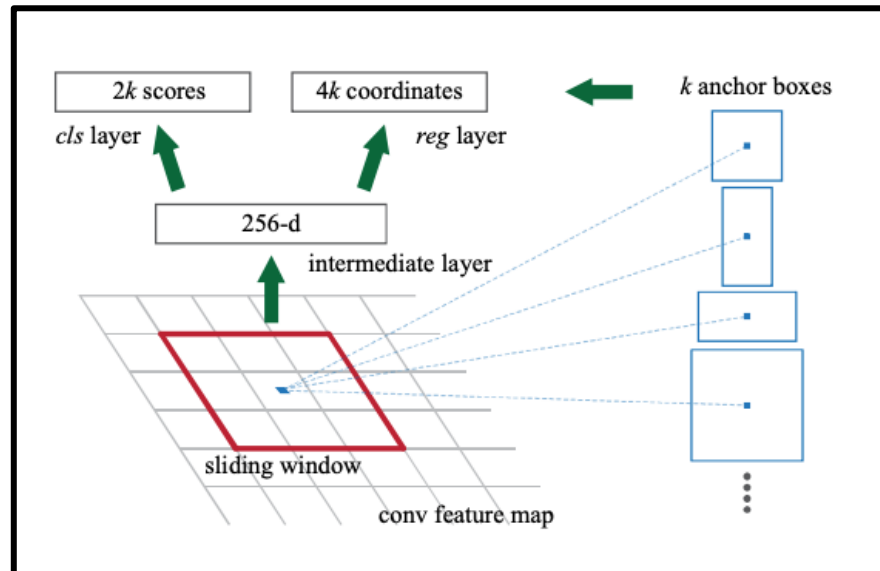
- ◆ 중간 특징 맵의 특정 지점을 중심으로 하는 Anchor Box 9개를 그려보고, Bounding Box의 위치를 추출하기 위해 채널 수를 $4 \times k (=9)$ 로 조정
- ◆ 그 결과 최종 특징 맵은 $H \times W \times k \times 4$ 의 크기를 갖고, 추후 습을 통해 Anchor Box의 위치 및 크기를 정밀하게 조정

High-Level Vision (객체 탐지) [7]

• Convolutional Neural Network for Object Detection

▪ Localization

▶ Region Proposal Method (RPN) (4)

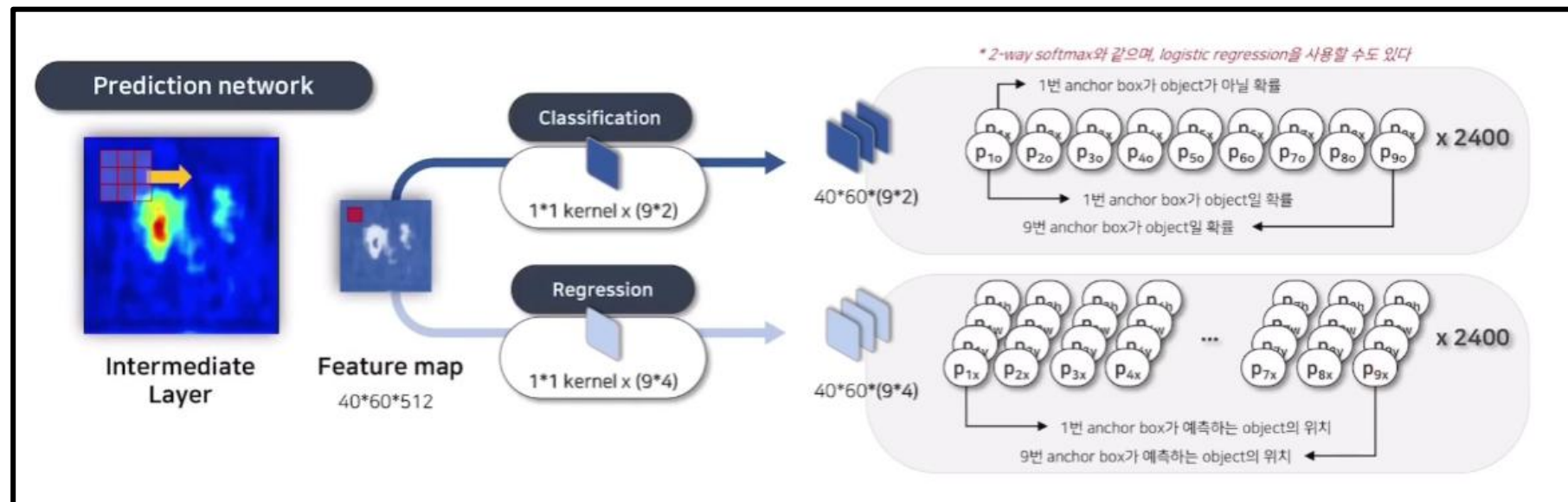


High-Level Vision (객체 탐지) [8]

• Convolutional Neural Network for Object Detection

▪ Localization

▶ Region Proposal Method (RPN) (5)



High-Level Vision (객체 탐지) [9]

- Convolutional Neural Network for Object Detection

- Localization

- ▶ Unified Detection (1)

- ▷ RPN와 달리 Unified Detection은 한 번에 Object Detection을 수행

- ▷ Unified Detection은 1-Stage OD 모델 중 하나인 You Only Look Once (YOLO)에서 사용

- ▷ Unified Detection은 다음과 같은 방식으로 작동됨

- 1. 입력 이미지를 $S \times S$ 개의 Grid Cell로 분할

- 2. 각 Grid Cell 마다 Bounding Box Prediction, Box Confidence Score, Conditional Class Probability를 계산

- ◆ Bounding Box Prediction → Object가 있을 만한 영역에 B개의 Bounding Box 및 Bounding Box의 위치 및 크기를 (x, y, w, h)로 표시

- ◆ Box Confidence Score (Pc) → 해당 Bounding Box마다 Box Confidence Score (Pc)를 계산 해당 Grid Cell에 Object가 있을 확률 Pr(Object)와 Intersection of Union (IoU)를 곱한 값

- ◆ Conditional class probability (Ci) → Ci는 Object가 Bounding Box 안에 있을 때 해당 Object가 i번째 Class에 해당할 확률

High-Level Vision (객체 탐지) [10]

• Convolutional Neural Network for Object Detection

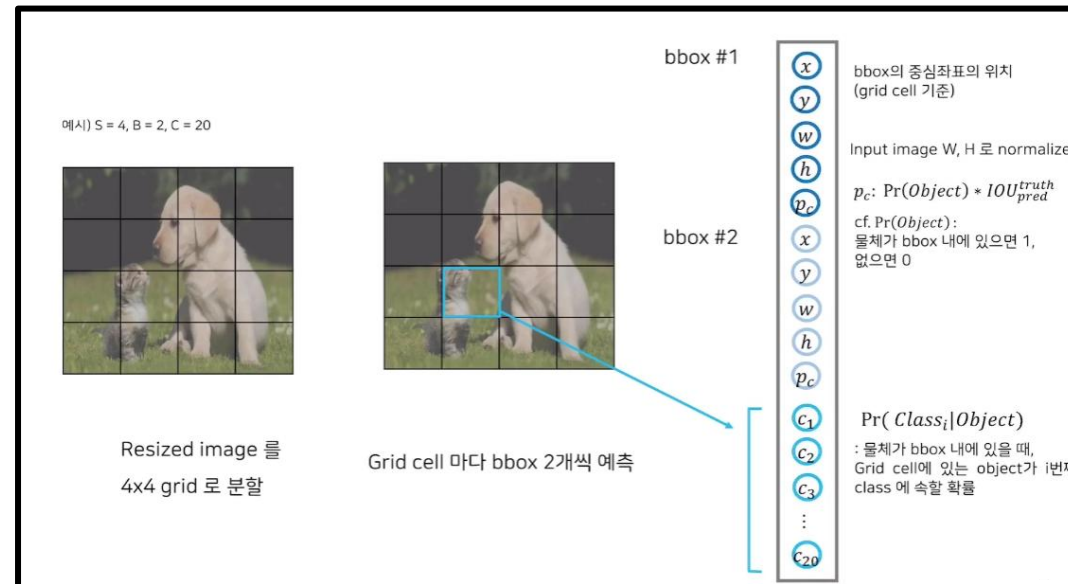
▪ Localization

▶ Unified Detection (2)

▷ Unified Detection은 다음과 같은 방식으로 작동됨

3. 최종적으로 각 Grid Cell에서 뽑아낸 벡터는 $B \times (4+1) + C$ 의 크기를 가짐

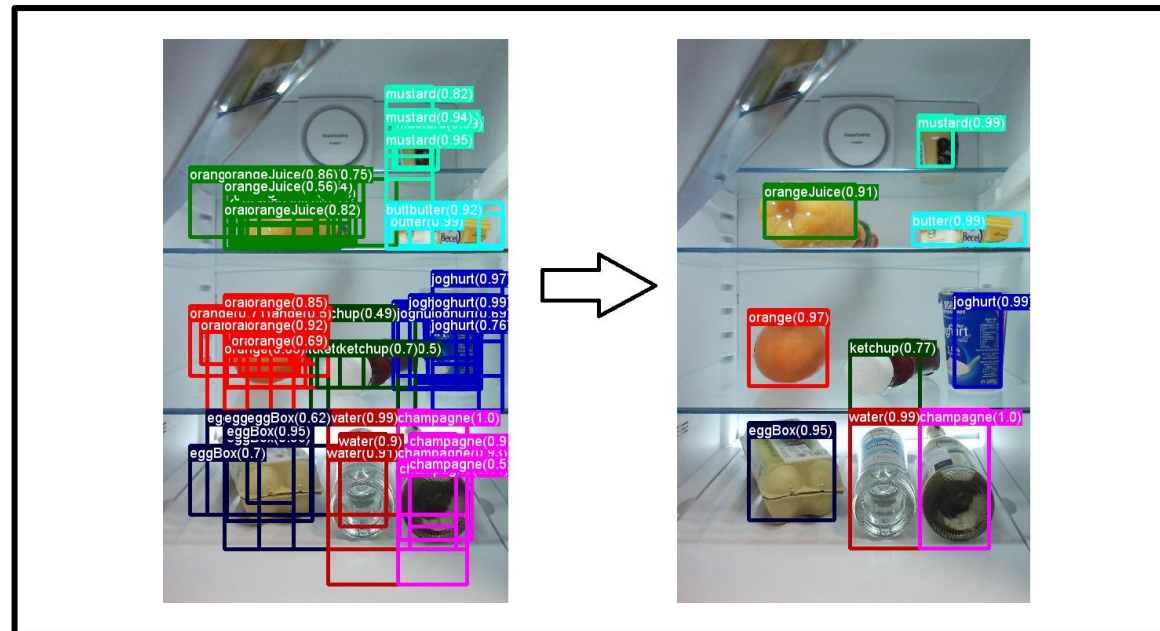
▷ YOLO는 입력 이미지를 작은 Grid Cell로 나누고, 각 Grid Cell 별로 Bounding Box 예측과 Classification을 동시에 수행



High-Level Vision (객체 탐지) [11]

- Convolutional Neural Network for Object Detection

- Non-Maximum Suppression (NMS)
 - ▶ 앞선 Localization 기법에 의거하여 Detection을 진행할 시 여러 개의 Bounding Box를 반환
 - ▶ 따라서 최종적으로 하나의 Bounding Box를 선정하기 위해 NMS를 사용
 - ▶ NMS는 Optimal한 Solution 대신 모델이 추론한 Bounding Box 중 Local Maxima를 찾음



High-Level Vision (객체 탐지) [12]

• Convolutional Neural Network for Object Detection

▪ Generalized NMS Process

▶ 일반적인 NMS는 다음과 같은 과정을 통해 진행

1. 모든 Bounding Box는 자신이 해당 객체를 얼마나 잘 잡아내지 나타내는 Confidence Score를 계산
2. 모든 Bounding Box에 대하여 Threshold 이하의 Confidence Score를 가지는 Bounding Box를 제거
 - ▷ Confidence Score가 일정 수준 이하인 Bounding Box들에 대해 일차적으로 필터링을 거치는 과정
3. 남은 Bounding Box들을 Confidence Score 기준 모두 내림차순으로 정렬
4. 맨 앞에 있는 Bounding Box를 기준으로 다른 Bounding Box와의 IoU 값을 계산
 - ▷ Bounding Box끼리 IoU가 높을수록, 즉 많이 겹쳐질수록 같은 물체를 검출하고 있다고 판단하기 때문에 IoU가 Threshold 이상인 Bounding Box들은 제거
5. 해당 과정을 순차적으로 시행하여 모든 Bounding Box를 비교하고 제거
6. Confidence Threshold가 높을수록, IoU Threshold가 낮을수록 더 많은 Bounding Box를 제거

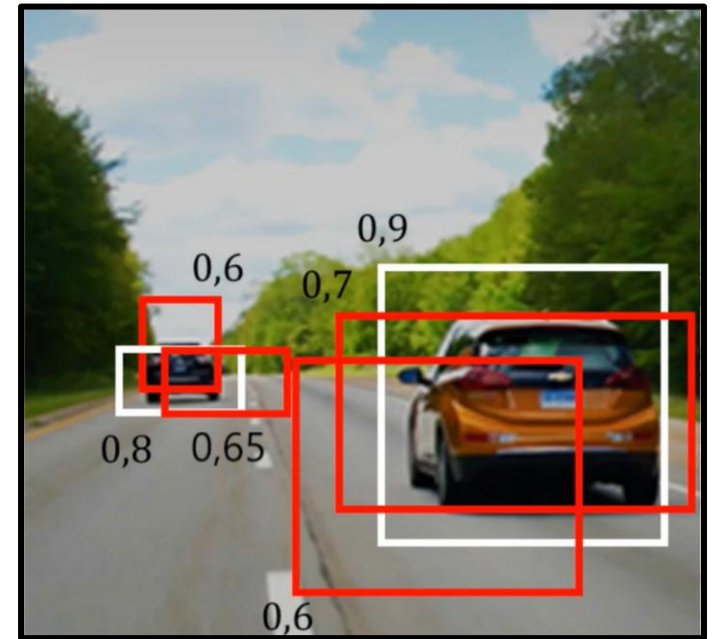
High-Level Vision (객체 탐지) [13]

• Convolutional Neural Network for Object Detection

▪ Generalized NMS Process

▶ Example

1. Confidence Score의 Threshold를 0.4라고 지정하면 Confidence Score가 0.4 이하인 Bounding Box들은 모두 제거
2. Bounding box를 Confidence Score 기준 내림차순 정렬
3. Confidence Score 0.9인 Bounding Box를 기준으로 뒤의 모든 박스를 비교
 - ▷ 0.8 박스와는 겹치지 않으므로 해당 보존
 - ▷ 0.7 박스와 IoU가 Threshold 이상이므로 해당 박스는 제거
 - ▷ 0.65 박스 및 0.6 박스(왼쪽)과는 겹치지 않으므로 보존
 - ▷ 0.6 박스(오른쪽)와 IoU가 Threshold 이상이므로 제거
4. Confidence Score 0.8인 Bounding Box를 기준으로 뒤의 모든 박스를 비교



High-Level Vision (객체 탐지) [14]

- Convolutional Neural Network for Object Detection

- Generalized NMS Process

- ▶ NMS의 문제점과 Anchor Box

- ▷ Unified Detection의 경우 각각의 Grid Cell이 하나의 Object만 감지

- ▷ 해당 상황에서 NMS를 적용시 필요한 Object의 Bounding Box가 제거되는 현상 발생

- ▷ 따라서, Grid Cell이 여러 개의 Object를 감지하기 위해 Anchor Box를 사용

- ▷ 만약 자동차와 사람이라는 Object가 겹쳐 있고, 만약 사람, 자동차, 오토바이 세 가지의 Class를 가진 벡터 y가 있다면 여기서 2개의 Class를 분류하는데 어려움이 존재

- ▷ 이를 보완하기 위해 Anchor Box를 사용하여 탐지하려는 객체의 모양을 정해 놓고 객체가 탐지되었을 때 어떤 Anchor Box와 유사한지 판단해서 벡터 값을 할당

High-Level Vision (객체 탐지) [15]

• Convolutional Neural Network for Object Detection

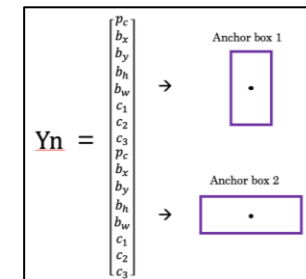
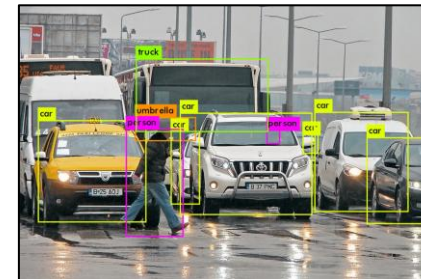
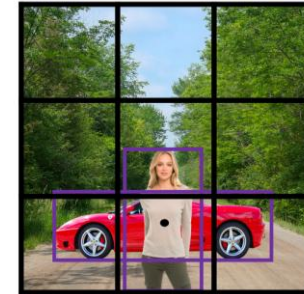
▪ Generalized NMS Process

▶ Example

1. 다음과 같이 Anchor box가 2개라면 벡터를 두개를 이어 붙인 것 동일 ($[p, x, y, w, h, c1, c2, c3, p, x, y, w, h, c1, c2, c3]$)

2. Detection을 할 때 Detection을 통해 예측한 Object의 Bounding Box가 Anchor Box 1에 유사한지 2에 유사한지 IoU를 비교

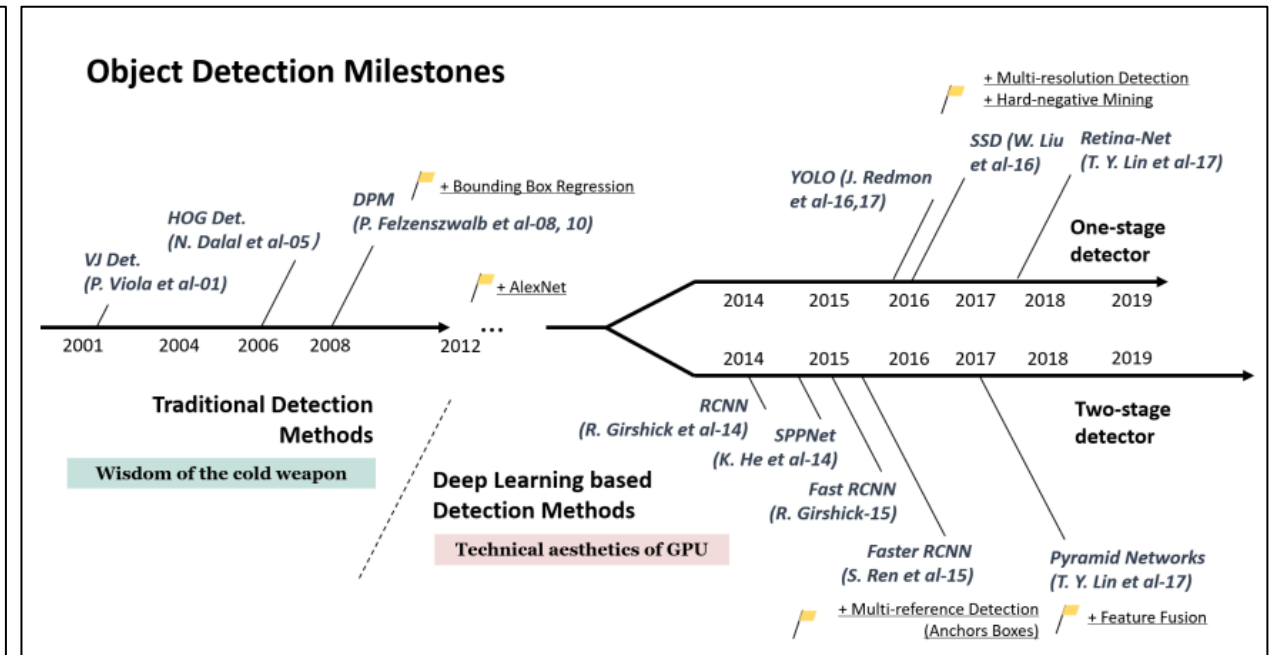
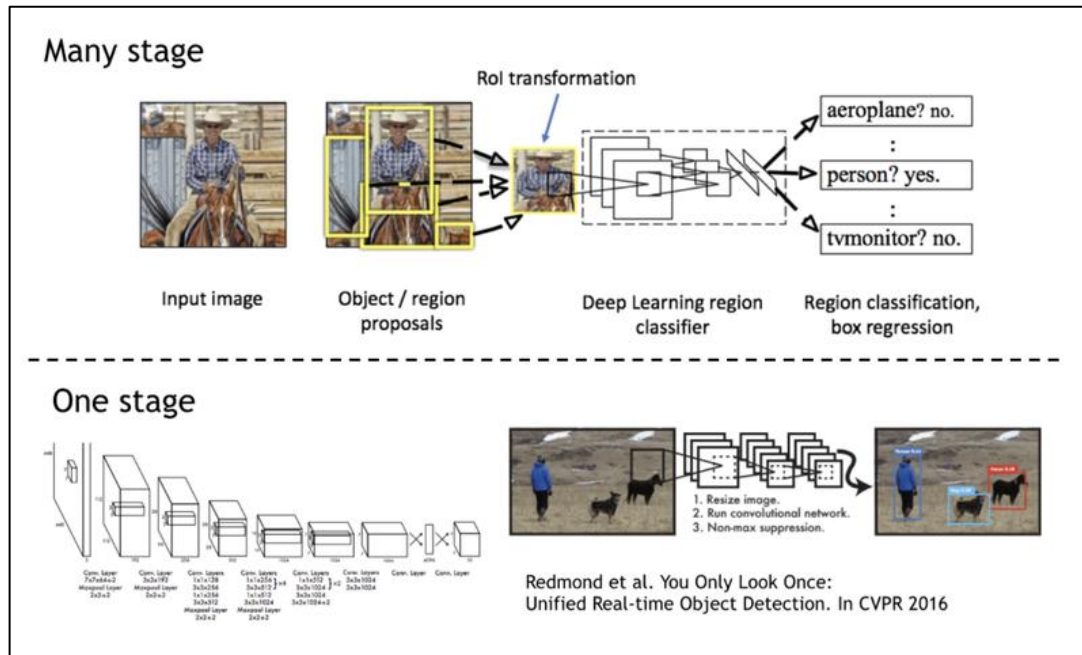
3. IoU 계산 후 높은 IoU를 갖는 Anchor Box 자리에 할당



High-Level Vision (객체 탐지) [16]

- **Convolutional Neural Network for Object Detection**

- Object Detection (OD) Models
 - ▶ 2-Stage vs. 1-Stage



High-Level Vision (객체 탐지) [17]

- **Convolutional Neural Network for Object Detection**

- Object Detection (OD) Models

- ▶ 2-Stage OD Models

- ▷ R-CNN

- ◆ [Rich feature hierarchies for accurate object detection and semantic segmentation](#) (arXiv, 2013)

- ▷ Fast R-CNN

- ◆ [Fast R-CNN](#) (ICCV, 2015)

- ▷ Faster R-CNN

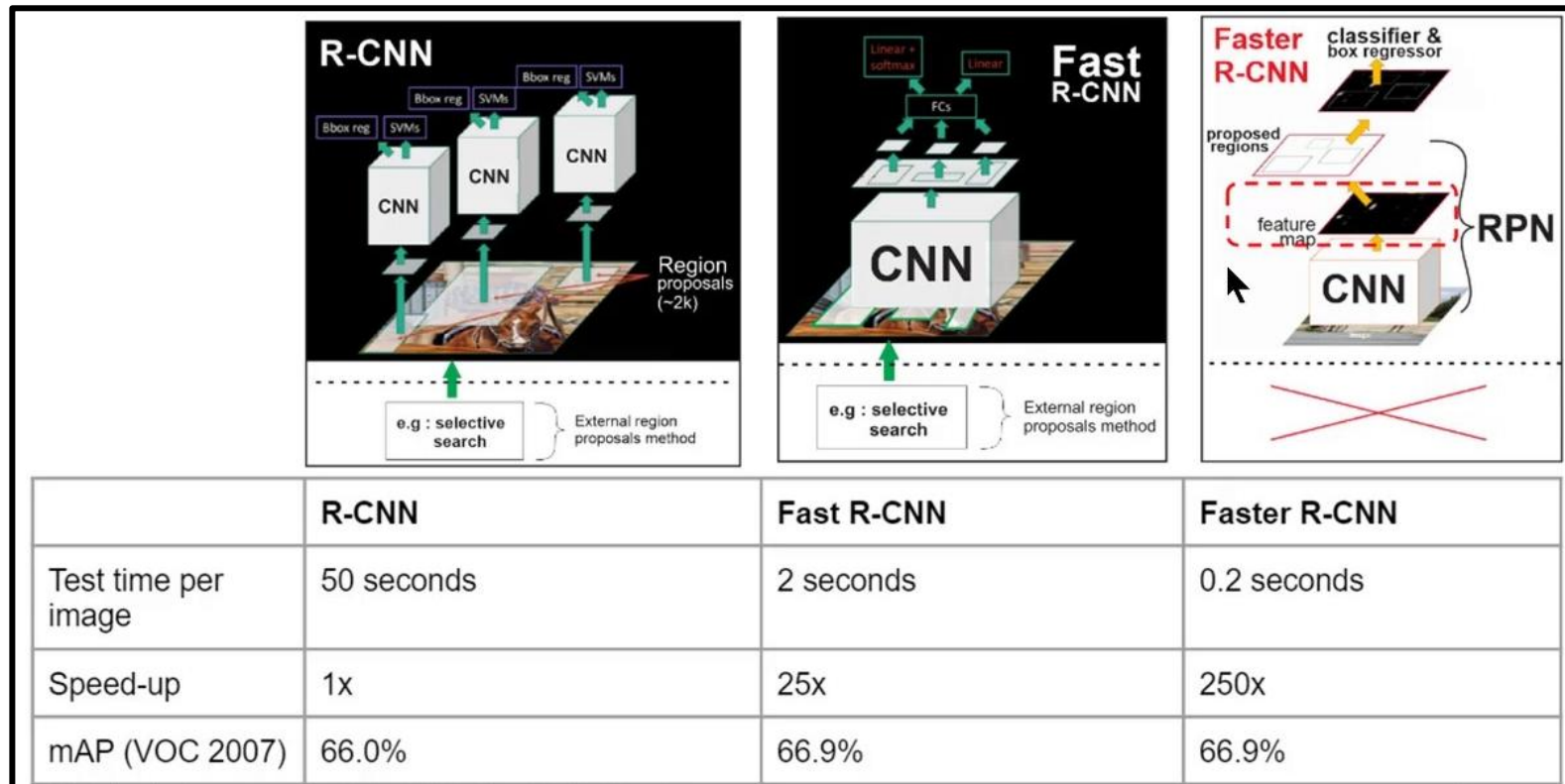
- ◆ [Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks](#) (NeuRIPS, 2015)

High-Level Vision (객체 탐지) [18]

• Convolutional Neural Network for Object Detection

▪ Object Detection (OD) Models

▶ 2-Stage OD Models



High-Level Vision (객체 탐지) [19]

- **Convolutional Neural Network for Object Detection**

- Object Detection (OD) Models

- ▶ 1-Stage OD Models

- ▷ You Only Look Once (YOLO)

- ◆ [You Only Look Once: Unified, Real-Time Object Detection](#) (arXiv, 2015)

- ▷ Single Shot MultiBox Detector (SSD)

- ◆ [SSD: Single Shot MultiBox Detector](#) (ECCV, 2016)

- ▷ RetinaNet

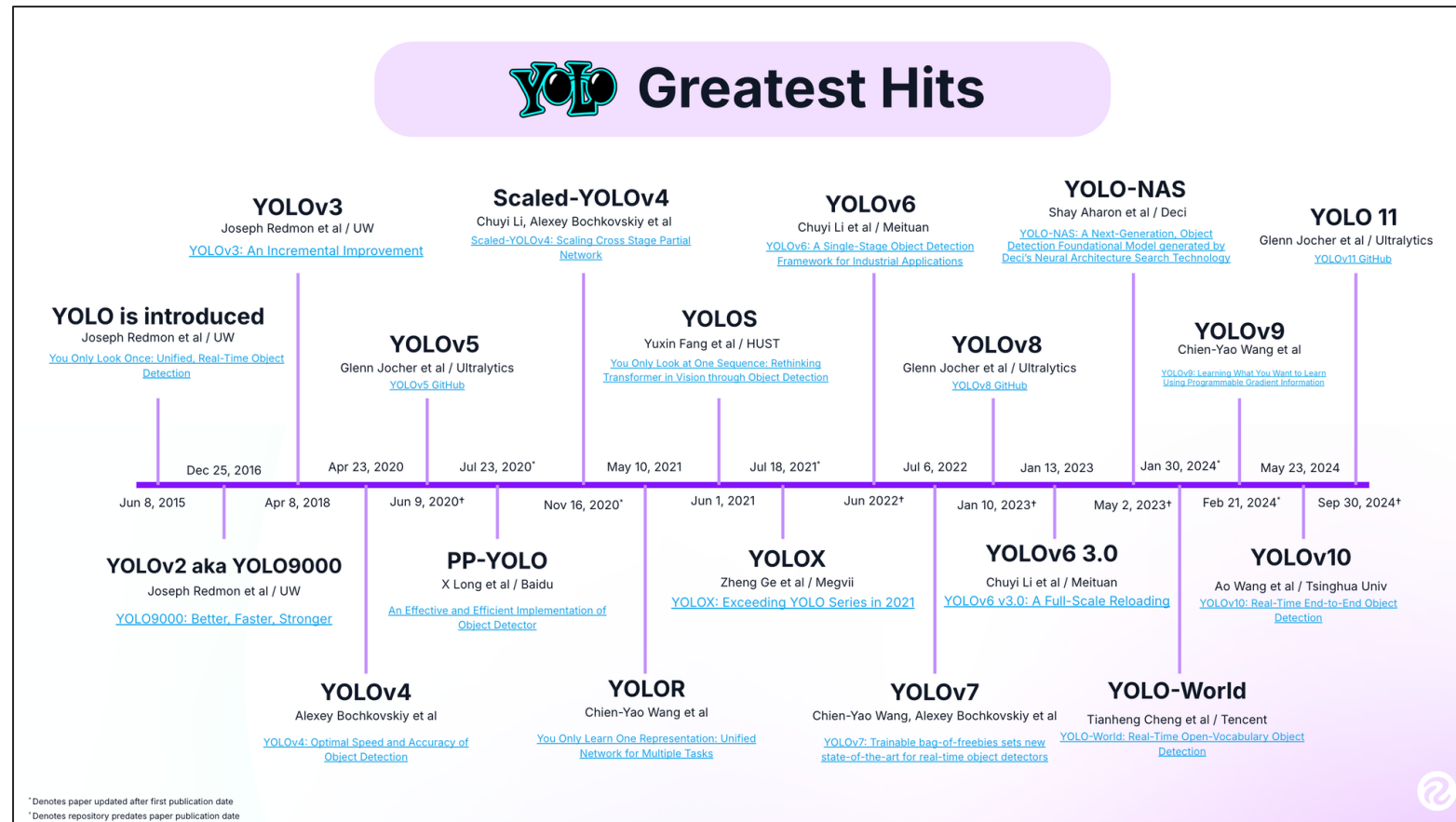
- ◆ [Focal Loss for Dense Object Detection](#) (arXiv, 2017)

High-Level Vision (객체 탐지) [20]

• Convolutional Neural Network for Object Detection

▪ Object Detection (OD) Models

▶ You Only Look Once (YOLO) → Real-Time 1-Stage Detector [Time Line]

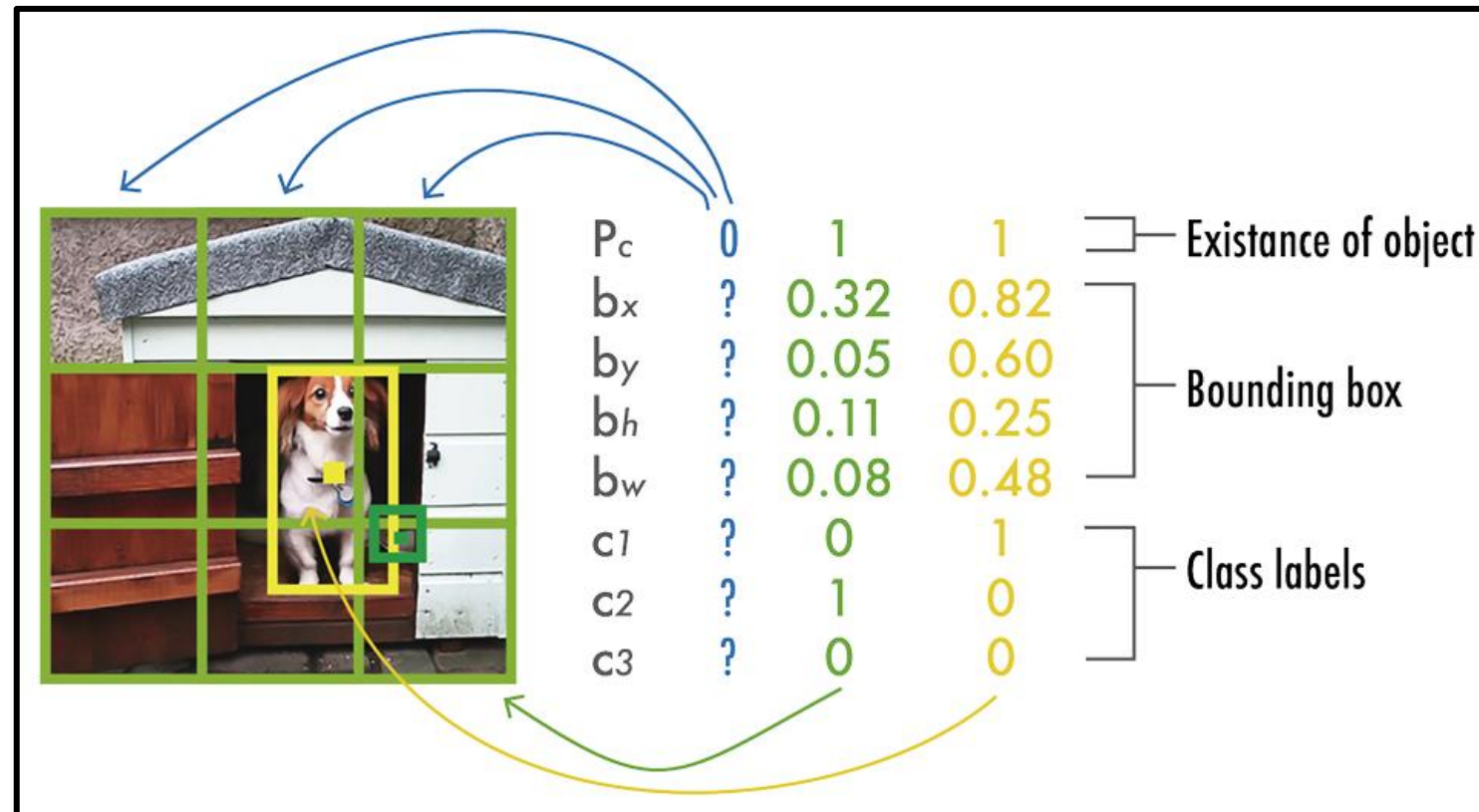


High-Level Vision (객체 탐지) [21]

- Convolutional Neural Network for Object Detection

- Object Detection (OD) Models

- ▶ You Only Look Once (YOLO) → Real-Time 1-Stage Detector [Output Predictions]



High-Level Vision (객체 탐지) [22]

• Convolutional Neural Network for Object Detection

▪ Object Detection (OD) Models

▶ You Only Look Once (YOLO) → Real-Time 1-Stage Detector [Loss Function]

▷ Localization Loss + Confidence Loss + Classification Loss

Localization loss

Set to 5 to increase the loss of bounding box predictions

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

For each grid cell For each grid box '1' if object appears in the ith cell and the jth box detect it, '0' otherwise

GT bbox x-coordinate in the ith cell Predicted bbox x-coordinate in the ith cell GT bbox y-coordinate in the ith cell Predicted bbox y-coordinate in the ith cell Sum-squared error

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

Square root to reduce the range of the values GT bbox width in the ith cell Predicted bbox width in the ith cell GT bbox height in the ith cell Predicted bbox height in the ith cell

Confidence loss

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

GT confidence score Predicted confidence score Confidence error when an object is detected in the ith cell

Set to 0.5 to decrease the loss for empty boxes

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

'1' if there is no object in the ith cell, '0' otherwise Confidence error when an object not detected in the ith cell

Classification loss

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

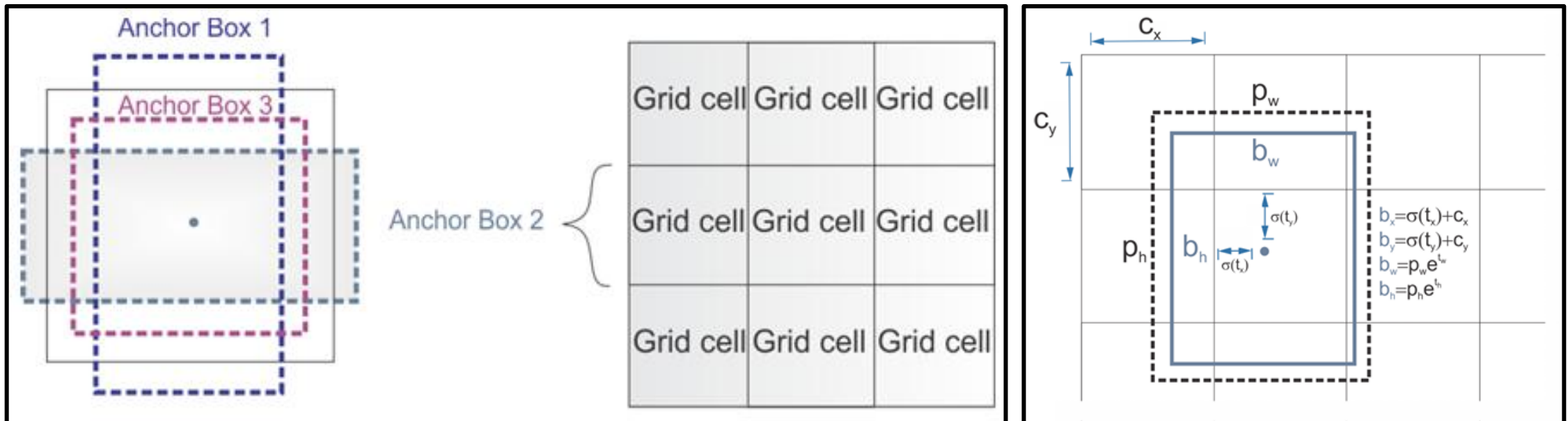
For each grid cell For each class Predicted conditional probability of an object of class c appearing in the ith cell GT conditional probability of class c appearing in the ith cell

High-Level Vision (객체 탐지) [23]

• Convolutional Neural Network for Object Detection

- Object Detection (OD) Models

- ▶ You Only Look Once (YOLO) → Real-Time 1-Stage Detector [Anchor Box & Bounding Boxes Prediction]

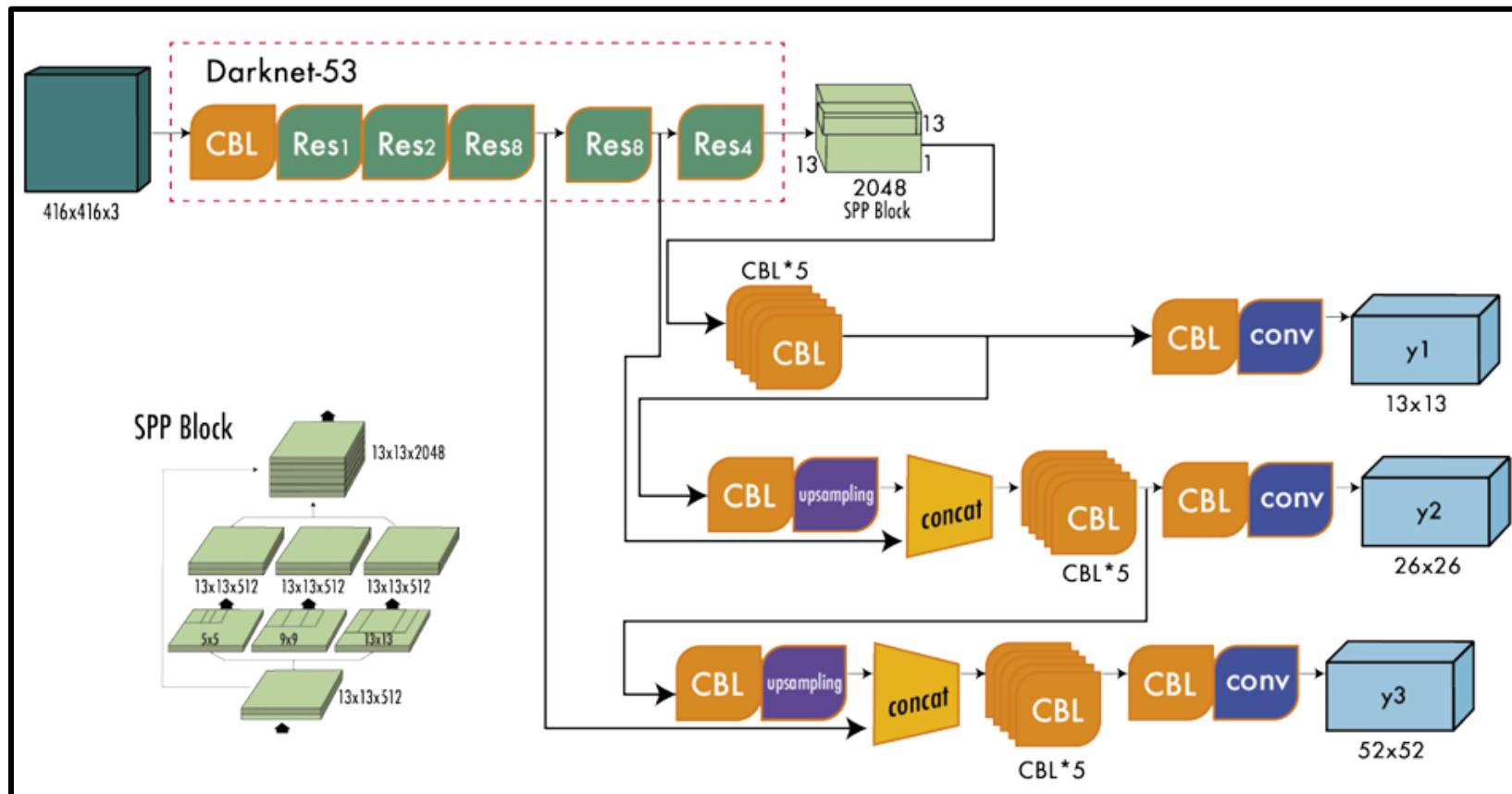


High-Level Vision (객체 탐지) [24]

• Convolutional Neural Network for Object Detection

▪ Object Detection (OD) Models

▶ You Only Look Once (YOLO) → Real-Time 1-Stage Detector [YOLOv3 Architecture]

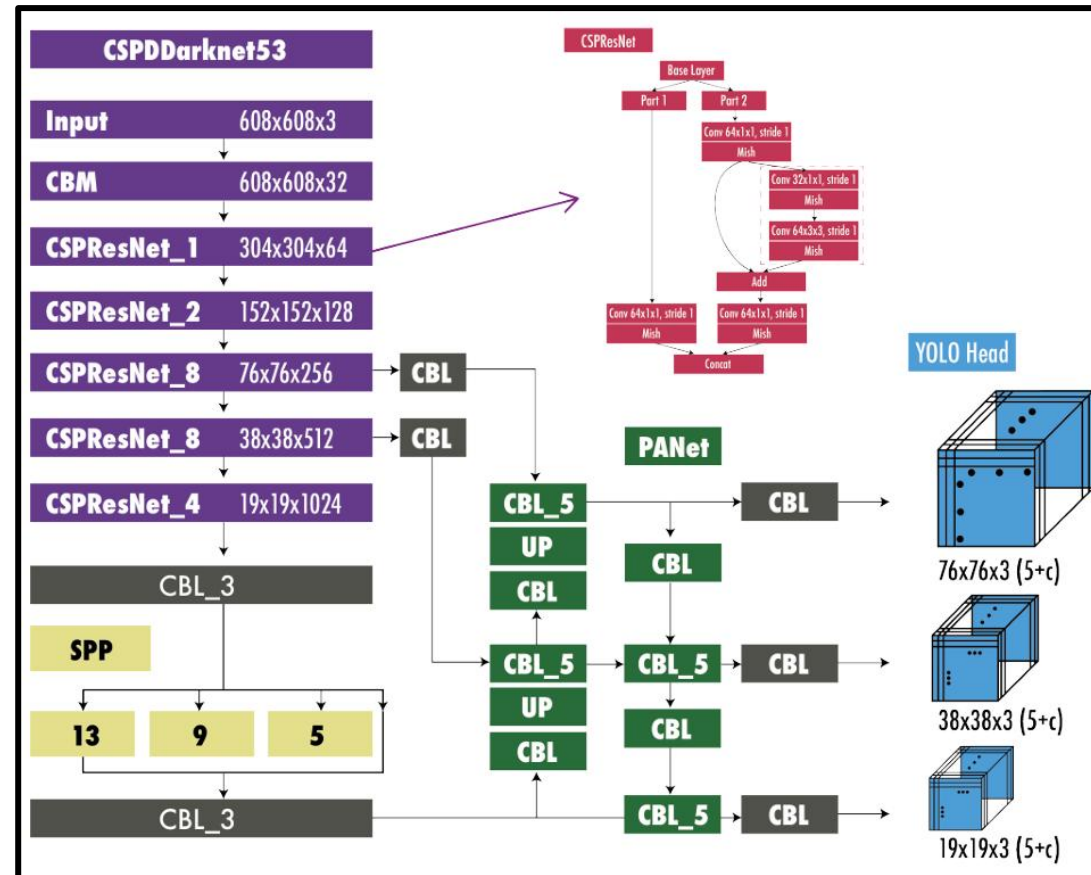


High-Level Vision (객체 탐지) [25]

- **Convolutional Neural Network for Object Detection**

- Object Detection (OD) Models

- You Only Look Once (YOLO) → Real-Time 1-Stage Detector [YOLOv4 Architecture]



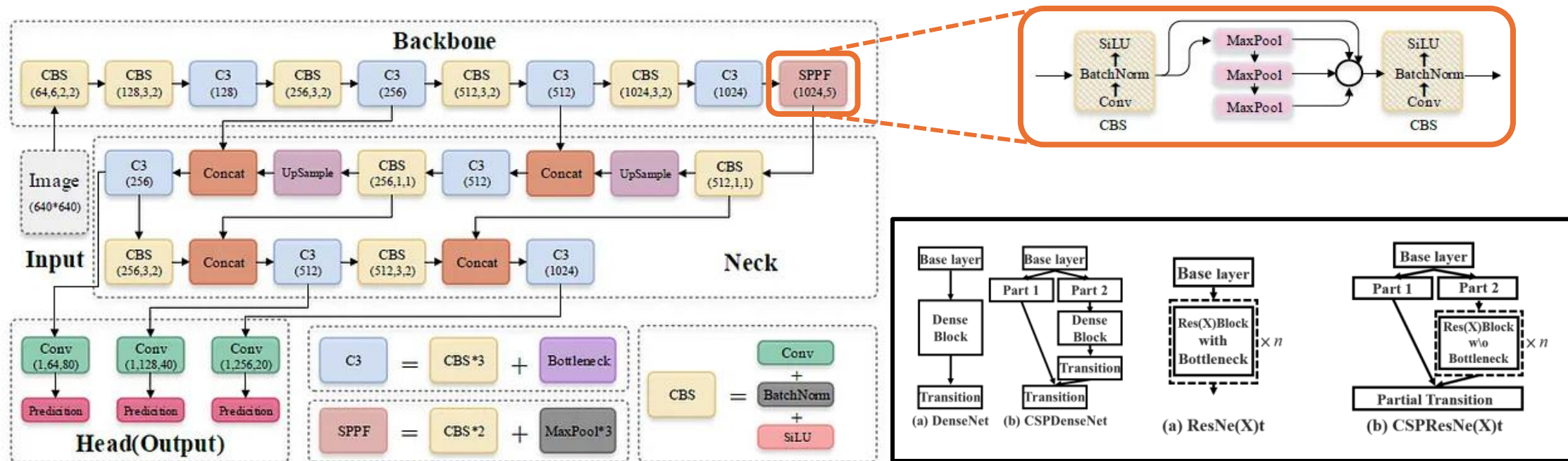
High-Level Vision (객체 탐지) [26]

• Convolutional Neural Network for Object Detection

▪ Object Detection (OD) Models

▶ You Only Look Once (YOLO) → Real-Time 1-Stage Detector [YOLOv5 Architecture]

▷ Backbone 구조 (CSP Darknet53)



Cross Stage Pyramid (CSP)

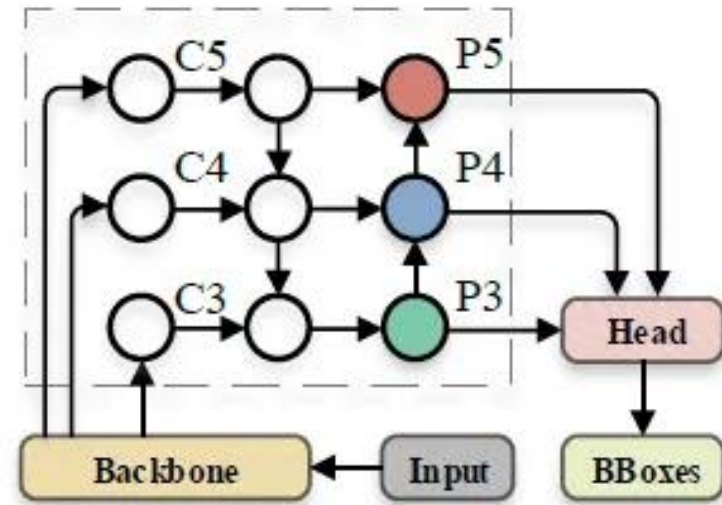
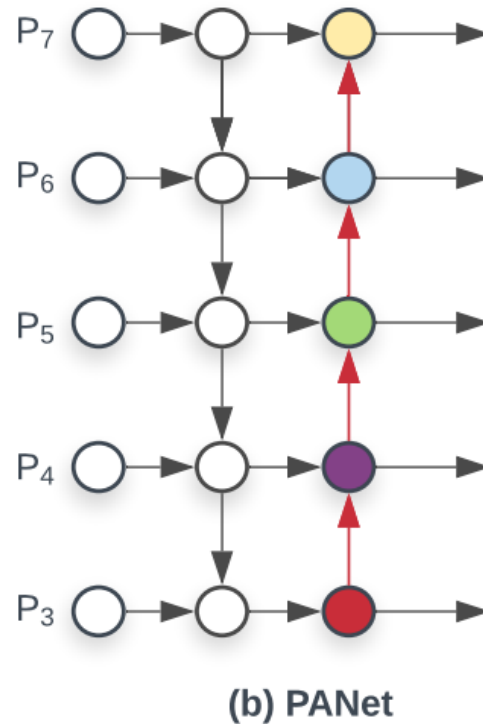
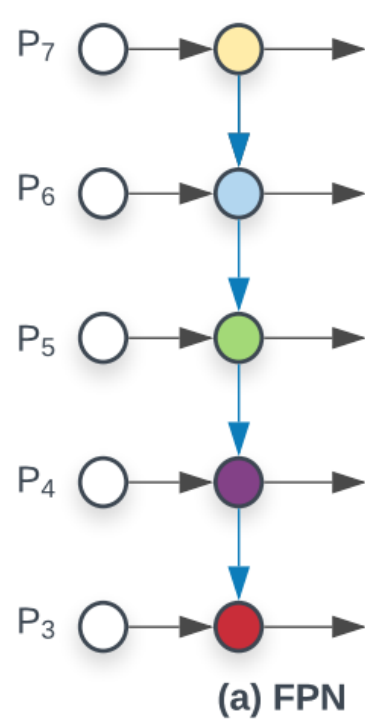
High-Level Vision (객체 탐지) [27]

• Convolutional Neural Network for Object Detection

▪ Object Detection (OD) Models

▶ You Only Look Once (YOLO) → Real-Time 1-Stage Detector [YOLOv5 Architecture]

▷ Neck 구조 (Feature Pyramid Network & Path Aggregation Network)



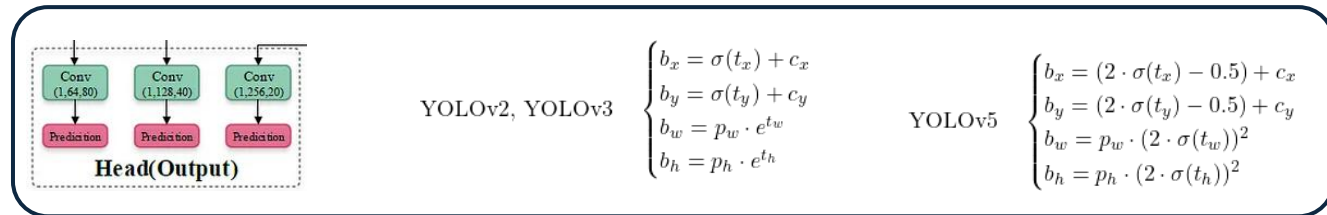
High-Level Vision (객체 탐지) [28]

• Convolutional Neural Network for Object Detection

▪ Object Detection (OD) Models

▶ Head 구조

- ▷ YOLOv3 및 YOLOv4와 동일한 헤드를 사용
- ▷ 경계 상자를 계산하는 방정식이 상이



▶ 활성화 함수

- ▷ SiLU와 Sigmoid를 사용

▶ 손실 함수

- ▷ 클래스 손실과 객체성 손실을 계산하기 위해 Binary Cross Entropy (BCE)를 사용
- ▷ 위치 손실을 계산하기 위해 CIoU(Complete Intersection over Union)를 사용

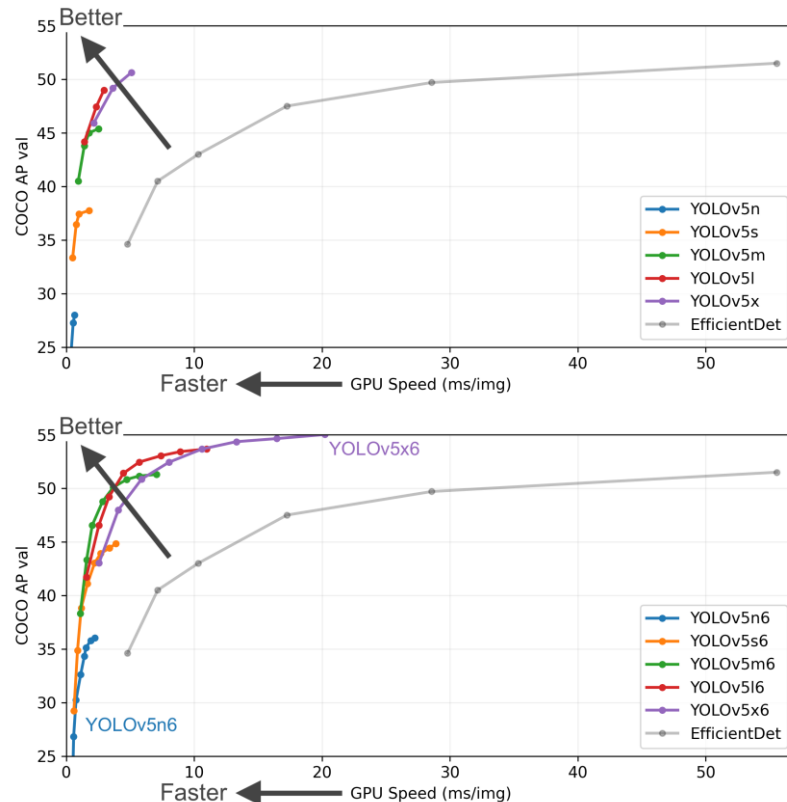
$$\text{Loss} = \lambda_1 \text{Loss}_{cls} + \lambda_2 \text{Loss}_{obj} + \lambda_3 \text{Loss}_{ciou}$$

High-Level Vision (객체 탐지) [29]

• Convolutional Neural Network for Object Detection

▪ Object Detection (OD) Models

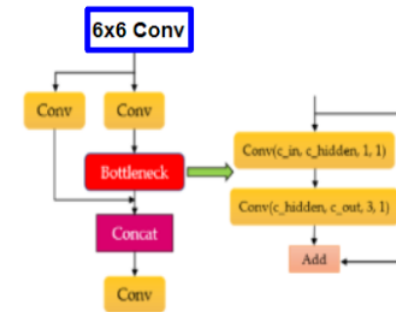
▶ YOLOv5의 다양한 버전



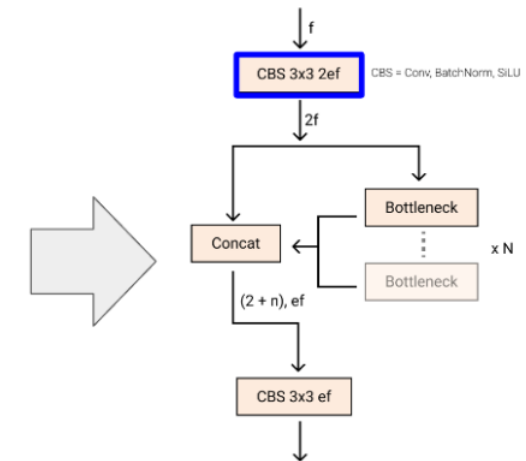
Model	size (pixels)	mAP ^{val} 50-95	mAP ^{val} 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6	1280	55.0	72.7	3136	26.2	19.4	140.7	209.8
YOLOv5x6 + TTA	1536	55.8	72.7	-	-	-	-	-

- Object Detection (OD) Models

Anchor Free



C3 Module (YOLOv5)



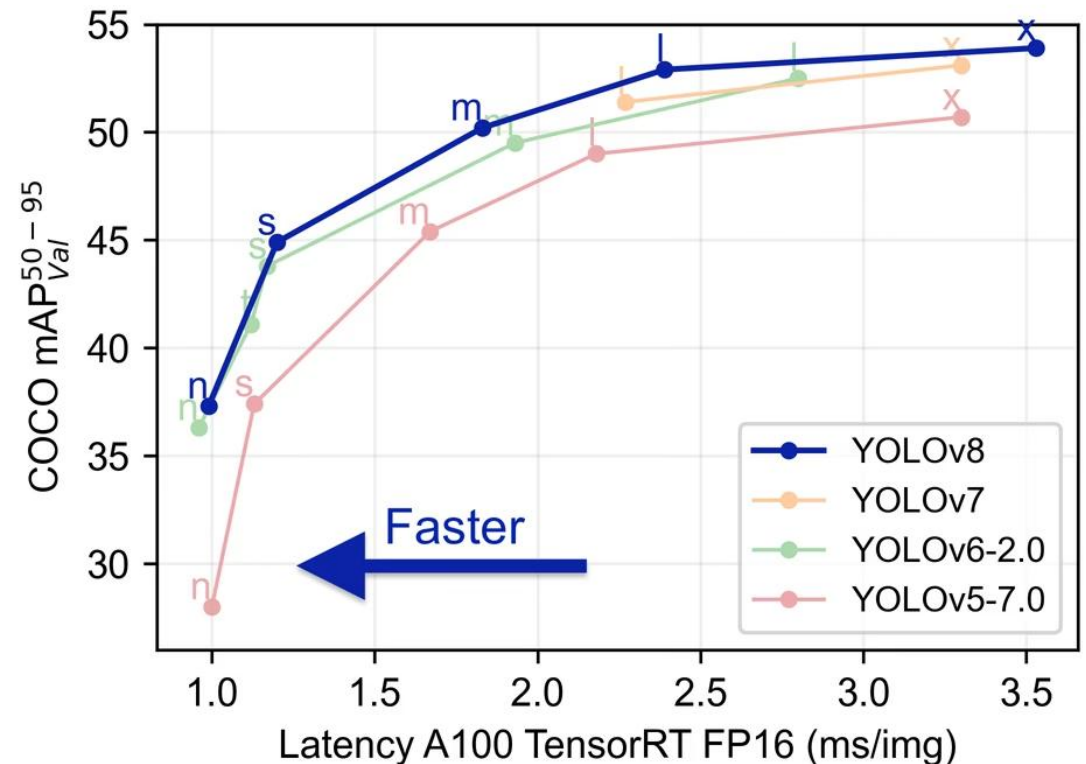
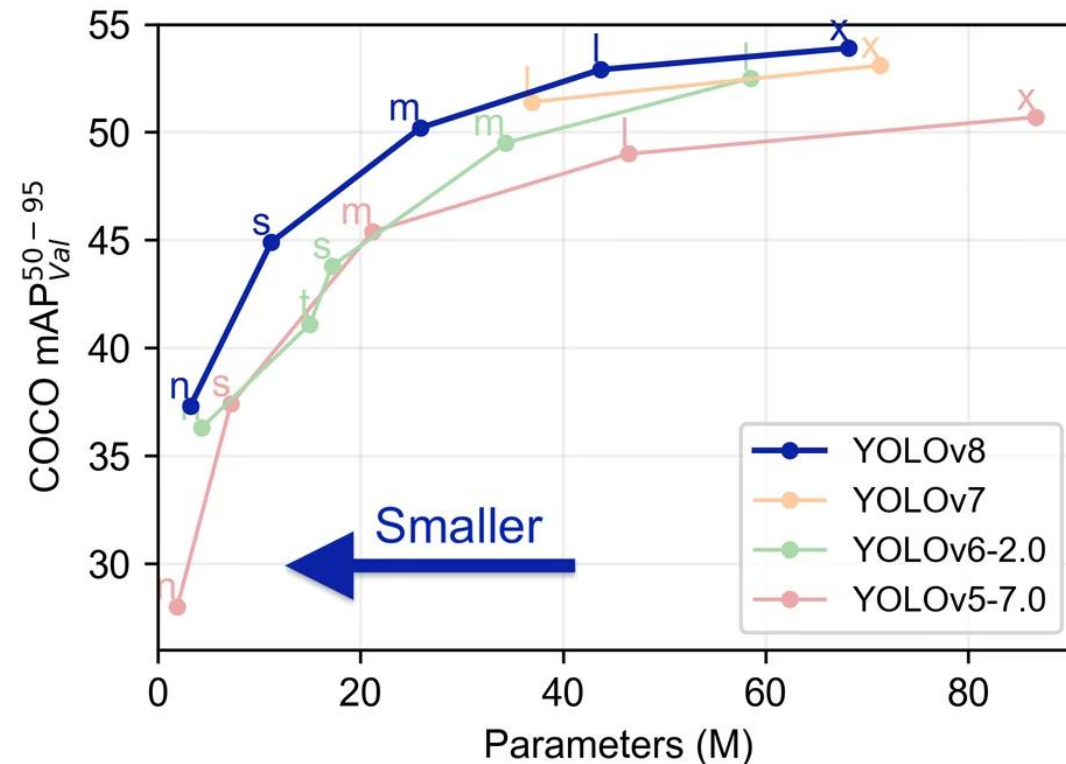
C2f Module (YOLOv8)

High-Level Vision (객체 탐지) [31]

• Convolutional Neural Network for Object Detection

▪ Object Detection (OD) Models

▶ YOLOv8의 성능 개선



High-Level Vision (객체 탐지) [32]

• Convolutional Neural Network for Object Detection

▪ Object Detection (OD) Models

▶ YOLOv8의 다양한 버전

Object Detection

Model	size (pixels)	mAP ^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Segmentation

Model	size (pixels)	mAP ^{box} 50-95	mAP ^{mask} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	36.7	30.5	96.1	1.21	3.4	12.6
YOLOv8s	640	44.6	36.8	155.7	1.47	11.8	42.6
YOLOv8m	640	49.9	40.8	317.0	2.18	27.3	110.2
YOLOv8l	640	52.3	42.6	572.4	2.79	46.0	220.5
YOLOv8x	640	53.4	43.4	712.1	4.02	71.8	344.1

Classification

Model	size (pixels)	acc top1	acc top5	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B) at 640
YOLOv8n	224	66.6	87.0	12.9	0.31	2.7	4.3
YOLOv8s	224	72.3	91.1	23.4	0.35	6.4	13.5
YOLOv8m	224	76.4	93.2	85.4	0.62	17.0	42.7
YOLOv8l	224	78.0	94.1	163.0	0.87	37.5	99.7
YOLOv8x	224	78.4	94.3	232.0	1.01	57.4	154.8

High-Level Vision (객체 탐지) [33]

- Convolutional Neural Network for Object Detection

- Object Detection (OD) Models

- YOLO Architectures (VOC2007 & COCO2017)

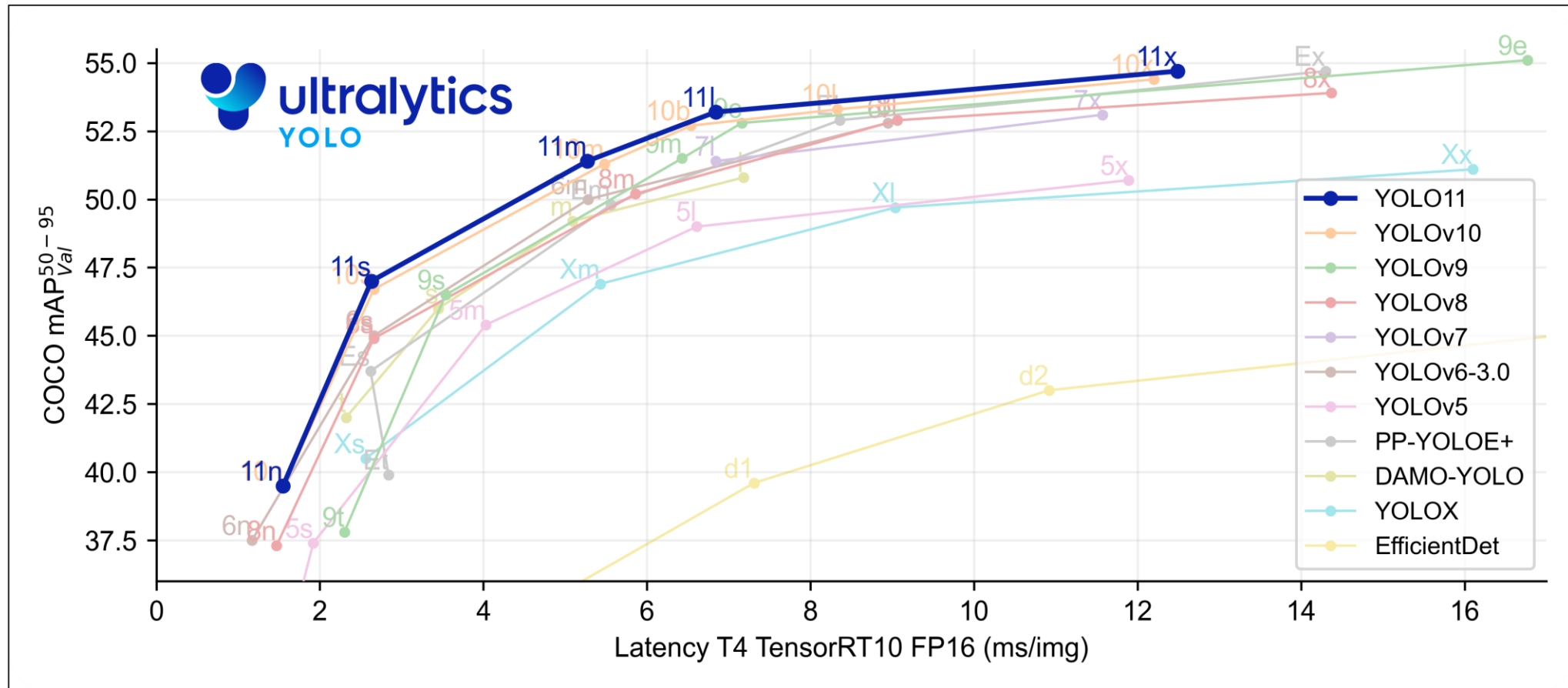
Version	Date	Anchor	Framework	Backbone	AP (%)
YOLO	2015	No	Darknet	Darknet24	63.4
YOLOv2	2016	Yes	Darknet	Darknet24	63.4
YOLOv3	2018	Yes	Darknet	Darknet53	36.2
YOLOv4	2020	Yes	Darknet	CSPDarknet53	43.5
YOLOv5	2020	Yes	Pytorch	Modified CSP v7	55.8
PP-YOLO	2020	Yes	PaddlePaddle	ResNet50-vd	45.9
Scaled-YOLOv4	2021	Yes	Pytorch	CSPDarknet	56.0
PP-YOLOv2	2021	Yes	PaddlePaddle	ResNet101-vd	50.3
YOLOR	2021	Yes	Pytorch	CSPDarknet	55.4
YOLOX	2021	No	Pytorch	Modified CSP v5	51.2
PP-YOLOE	2022	No	PaddlePaddle	CSPRepResNet	54.7
YOLOv6	2022	No	Pytorch	EfficientRep	52.5
YOLOv7	2022	No	Pytorch	RepConvN	56.8
DAMO-YOLO	2022	No	Pytorch	MAE-NAS	50.0
YOLOv8	2023	No	Pytorch	YOLO v8	53.9

High-Level Vision (객체 탐지) [34]

- Convolutional Neural Network for Object Detection

- Object Detection (OD) Models

- YOLOv11



High-Level Vision (객체 탐지) [35]

• Convolutional Neural Network for Object Detection

▪ Object Detection (OD) Models

▶ YOLOv11

Object Detection

Model	size (pixels)	mAP ^{val} 50-95	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n	640	39.5	56.1 ± 0.8	1.5 ± 0.0	2.6	6.5
YOLO11s	640	47.0	90.0 ± 1.2	2.5 ± 0.0	9.4	21.5
YOLO11m	640	51.5	183.2 ± 2.0	4.7 ± 0.1	20.1	68.0
YOLO11l	640	53.4	238.6 ± 1.4	6.2 ± 0.1	25.3	86.9
YOLO11x	640	54.7	462.8 ± 6.7	11.3 ± 0.2	56.9	194.9

Segmentation

Model	size (pixels)	mAP ^{box} 50-95	mAP ^{mask} 50-95	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n-seg	640	38.9	32.0	65.9 ± 1.1	1.8 ± 0.0	2.9	10.4
YOLO11s-seg	640	46.6	37.8	117.6 ± 4.9	2.9 ± 0.0	10.1	35.5
YOLO11m-seg	640	51.5	41.5	281.6 ± 1.2	6.3 ± 0.1	22.4	123.3
YOLO11l-seg	640	53.4	42.9	344.2 ± 3.2	7.8 ± 0.2	27.6	142.2
YOLO11x-seg	640	54.7	43.8	664.5 ± 3.2	15.8 ± 0.7	62.1	319.0

Classification


Model	size (pixels)	acc top1	acc top5	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B) at 224
YOLO11n-cls	224	70.0	89.4	5.0 ± 0.3	1.1 ± 0.0	1.6	0.5
YOLO11s-cls	224	75.4	92.7	7.9 ± 0.2	1.3 ± 0.0	5.5	1.6
YOLO11m-cls	224	77.3	93.9	17.2 ± 0.4	2.0 ± 0.0	10.4	5.0
YOLO11l-cls	224	78.3	94.3	23.2 ± 0.3	2.8 ± 0.0	12.9	6.2
YOLO11x-cls	224	79.5	94.9	41.4 ± 0.9	3.8 ± 0.0	28.4	13.7

High-Level Vision (객체 탐지) [36]

• Convolutional Neural Network for Object Detection

▪ Object Detection (OD) Models

▶ YOLOv11 튜토리얼



[中文](#) | [한국어](#) | [日本語](#) | [Русский](#) | [Deutsch](#) | [Français](#) | [Español](#) | [Português](#) | [Türkçe](#) | [Tiếng Việt](#) | [العربية](#)

[Ultralytics CI](#) [passing](#) [Run on Gradient](#) [Open in Colab](#) [Open in Kaggle](#)

[Discord](#) 837 online [Forums](#) 420 users [Reddit](#) 590

This **Ultralytics Colab Notebook** is the easiest way to get started with **YOLO models**—no installation needed. Built by **Ultralytics**, the creators of YOLO, this notebook walks you through running **state-of-the-art** models directly in your browser.

Ultralytics models are constantly updated for performance and flexibility. They're **fast**, **accurate**, and **easy to use**, and they excel at [object detection](#), [tracking](#), [instance segmentation](#), [image classification](#), and [pose estimation](#).

Find detailed documentation in the [Ultralytics Docs](#). Get support via [GitHub Issues](#). Join discussions on [Discord](#), [Reddit](#), and the [Ultralytics Community Forums](#)!

Request an Enterprise License for commercial use at [Ultralytics Licensing](#).

Train Ultralytics YOLO11 Model on Custom Dataset

Watch: How to Train [Ultralytics YOLO11](#) Model on Custom Dataset using Google Colab Notebook


High-Level Vision (객체 탐지) [37]

- Convolutional Neural Network for Object Detection

- Quantitative Metrics

- ▶ Intersection of Union (IoU)

- ▷ OD가 예측한 Bounding Box와 정답 Bounding Box간 중첩되는 부분의 면적을 측정
 - ▷ 이를 확률적으로 표현하기 위해 중첩되는 면적을 합집합 면적으로 나눔
 - ▷ 따라서, IoU Threshold 값을 설정하여 제대로 탐지를 하지 못한 Bounding Box를 제거 가능

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


High-Level Vision (객체 탐지) [38]

- Convolutional Neural Network for Object Detection

- Quantitative Metrics

- ▶ Confusion Matrix & Metrics

- ▷ Precision → 검출 결과들 중 옳게 검출한 비율

- ▷ Recall → 실제 옳게 검출된 결과물 중에서 옳다고 예측한 것의 비율

	True (Target)	False (Target)
True (Prediction)	TP (진양성)	FP (위양성)
False (Prediction)	FN (위음성)	TN (진음성)

- 재현율 (Recall)

$$REC = \frac{TP}{(TP + FN)}$$

- 정밀도 (Precision)

$$PRC = \frac{TP}{(TP + FP)}$$

High-Level Vision (객체 탐지) [39]

- **Convolutional Neural Network for Object Detection**

- Quantitative Metrics

- ▶ Precision-Recall 곡선

- ▷ Precision과 Recall은 반비례하는 성질을 가짐

- ▷ 따라서, Precision과 Recall의 성능 변화를 전체적으로 확인하기 위해 Precision-Recall 곡선을 사용

- ▶ Average Precision (AP)

- ▷ Precision-Recall 곡선의 면적을 의미 (0과 1 사이의 값)

- ▷ AP가 높을수록 OD 모델의 성능이 좋다고 판단

- ▶ Mean Average Precision (mAP)

- ▷ 각 Class별 AP를 계산한 후 평균을 낸 값