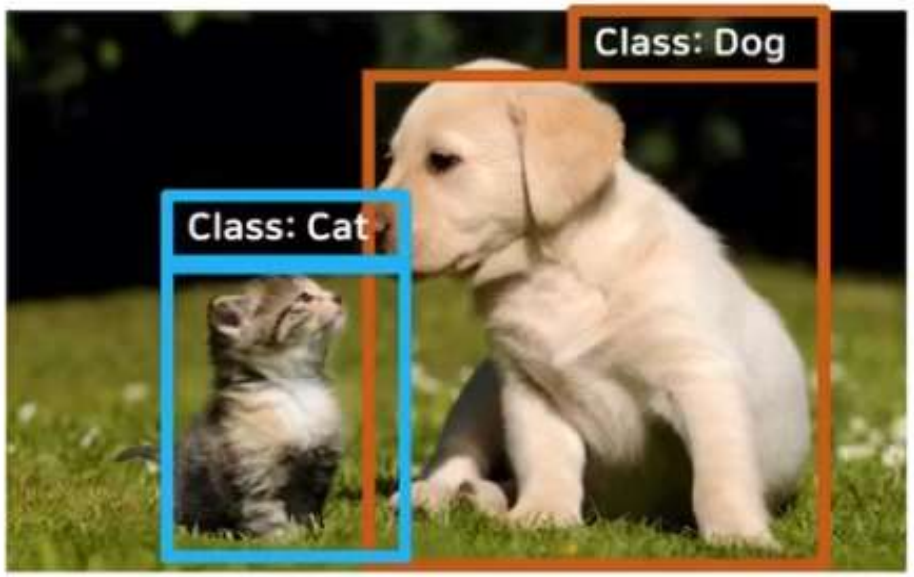
Object Detection

Object classification

- 이미지 내 single object
- Object class
- output: class probability

Object Localization

- 이미지 내 single object
- Object class
- Bounding Box(물체의 위치)
- output: (x, y, w, h)



Object Detection

- 이미지 내 **multiple object**
- Object class
- Bounding Box
- output:
class probabilities + (x, y, w, h)
class probabilities + (x, y, w, h)

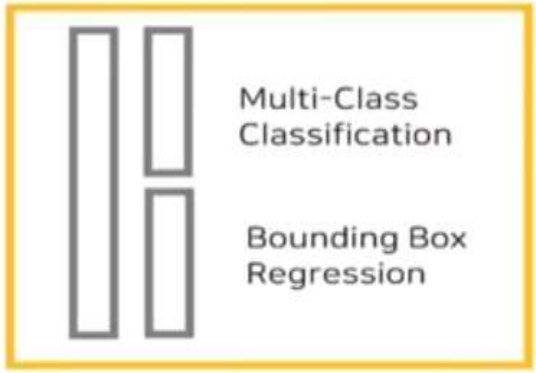
출처: https://www.youtube.com/watch?v=O78V3kwBRBk

Object Detection

One-Stage Detector

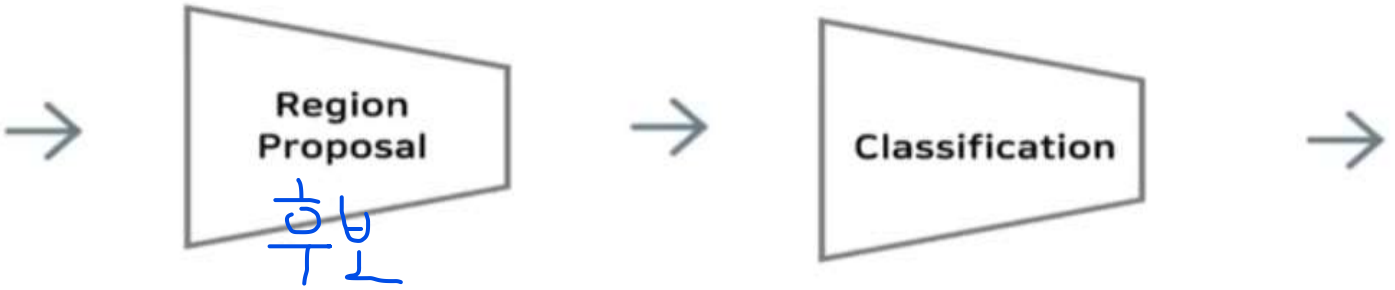
- Localization & Classification 동시에 수행하여 결과 얻음
- 이미지 내 모든 위치를 object 의 잠재영역으로 보고 각 후보영역에 대해 class 예측

For each grid cell

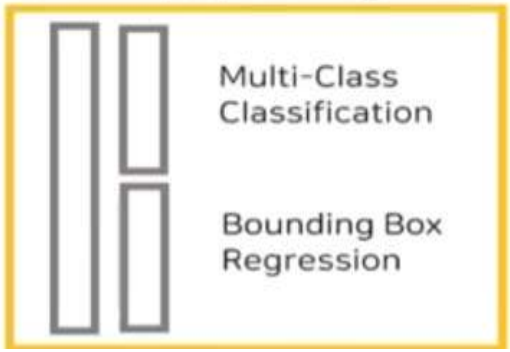


Two-Stage Detector

- Localization → Classification 순차적으로 수행하여 결과 얻음
- 후보 object 위치 제안한 후, object class 예측

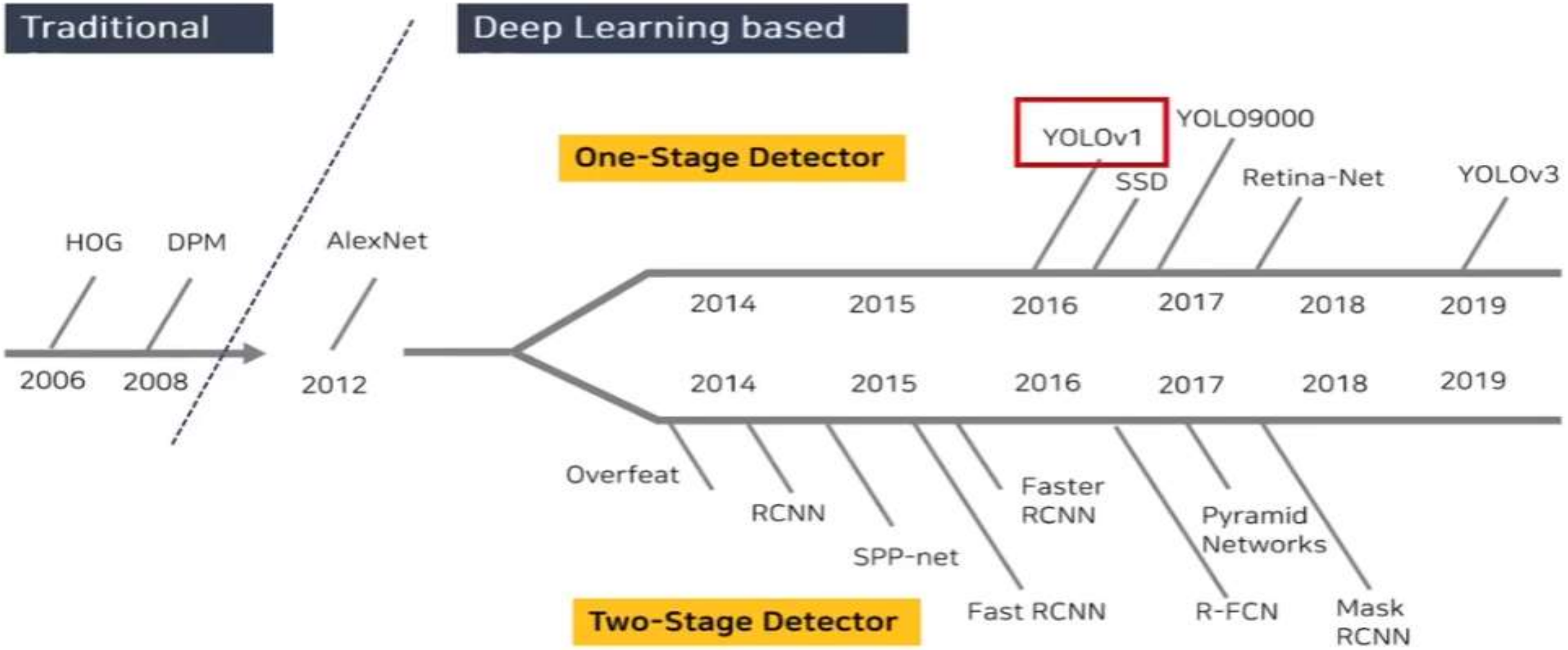


For each proposed region



출처: https://www.youtube.com/watch?v=O78V3kwBRBk

Object Detection Algorithm History



출처: https://www.youtube.com/watch?v=O78V3kwBRBk

YOLO

① 전체 이미지 보는 횟수: 1회

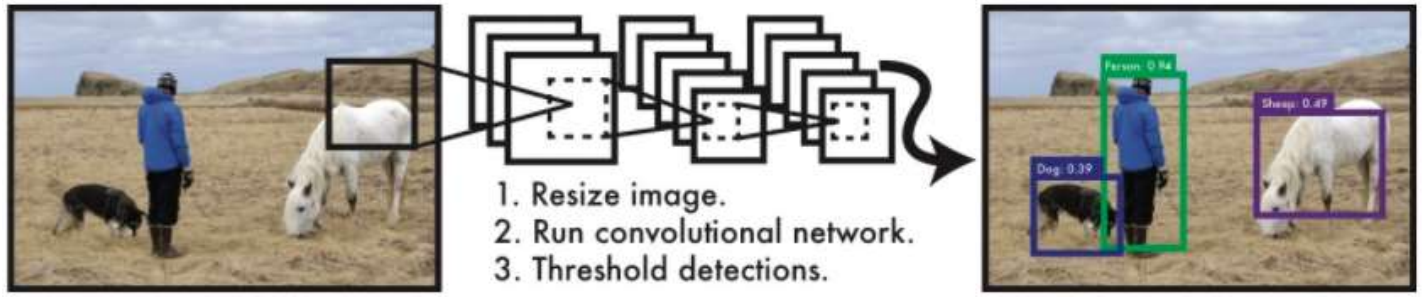
You Only Look Once:

Unified, Real-Time Object Detection

② Classifcation & Localization 단계 단일화

③ 속도 개선:
- test time: 45 fps (Fast YOLO: 155 fps)
- RCNN: 6 fps
- DPM(30Hz): 30 fps

YOLO(You Only Look Once)는 이미지 내의 bounding box와 class probability를 single regression problem으로 간주하여, 이미지를 한 번 보는 것으로 object의 종류와 위치를 추측한다. 아래와 같이 single convolutional network를 통해 multiple bounding box에 대한 class probablility를 계산하는 방식이다.



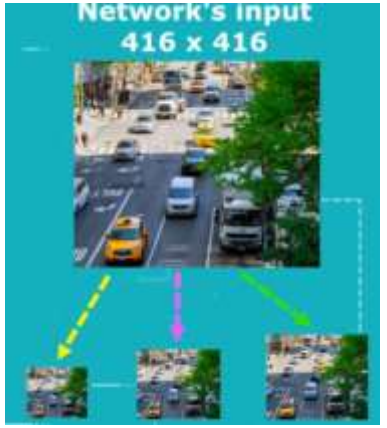
1. detect multiple objects
2. Predict classes
3. Identify locations

< source>https://curt-park.github.io/2017-03-26/yolo/

<source: https://www.youtube.com/watch?v=O78V3kwBRBk

Yolo

- YOLO (2018)
  - YOLOv3: An Incremental Improvement (https://arxiv.org/abs/1804.02767)
  - YOLO는 v1, v2, v3의 순서로 발전하였는데, v1은 정확도가 너무 낮은 문제가 있었고 이 문제는 v2까지 이어짐
  - 엔지니어링적으로 보완한 v3는 v2보다 살짝 속도는 떨어지더라도 정확도를 대폭 높인 모델
  - RetinaNet과 마찬가지로 FPN을 도입해 정확도를 높임
  - RetinaNet에 비하면 정확도는 4mAP정도 떨어지지만, 속도는 더 빠르다는 장점



Feature pyramid Network

- YOLOv4 (2020)
  - YOLOv4: Optimal Speed and Accuracy of Object Detection (https://arxiv.org/pdf/2004.10934v1.pdf)
  - YOLOv3에 비해 AP, FPS가 각각 10%, 12% 증가
  - YOLOv3와 다른 개발자인 AlexeyBochkousky가 발표
  - v3에서 다양한 딥러닝 기법(WRC, CSP ...) 등을 사용해 성능을 향상시킴
  - CSPNet 기반의 backbone(CSPDarkNet53)을 설계하여 사용

Unified Detection

1. Input image를 S X S grid로 나눈다.

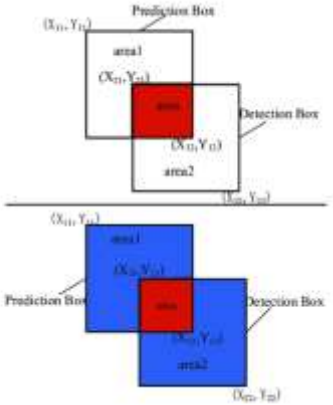
2. 각각의 grid cell은 B개의 bounding box와 각 bounding box에 대한 confidence score를 갖는다. (만약 cell에 object가 존재하지 않는다면 confidence score는 0이 된다.)
   Confidence Score: $Pr(Object) * IOU_{pred}^{truth}$

3. 각각의 grid cell은 C개의 conditional class probability를 갖는다.
   Conditional Class Probability: $Pr(Class_i|Object)$

4. 각각의 bounding box는 x, y, w, h, confidence로 구성된다.
   (x,y): Bounding box의 중심점을 의미하며, grid cell의 범위에 대한 상대값이 입력된다.
   (w,h): 전체 이미지의 width, height에 대한 상대값이 입력된다.

   - *예1: 만약 x가 grid cell의 가장 왼쪽에 있다면 x=0, y가 grid cell의 중간에 있다면 y=0.5*
   - *예2: bbox의 width가 이미지 width의 절반이라면 w=0.5*

IOU =



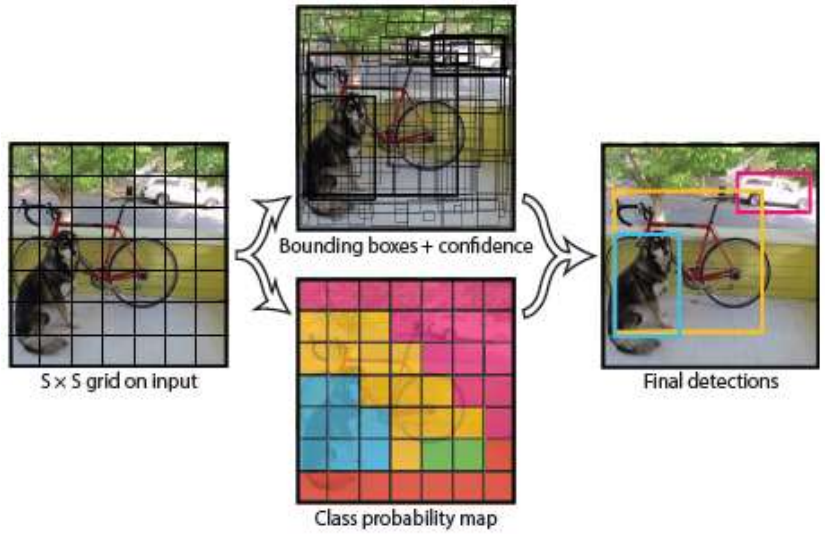
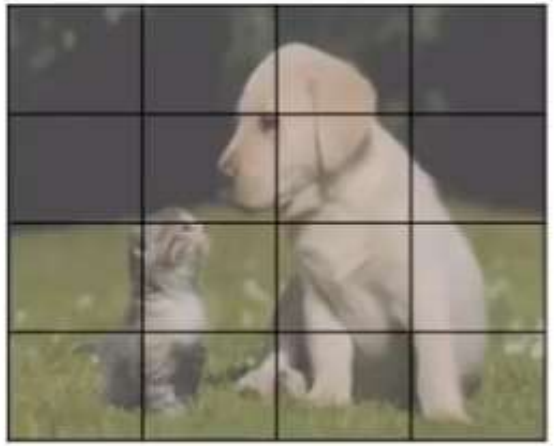


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

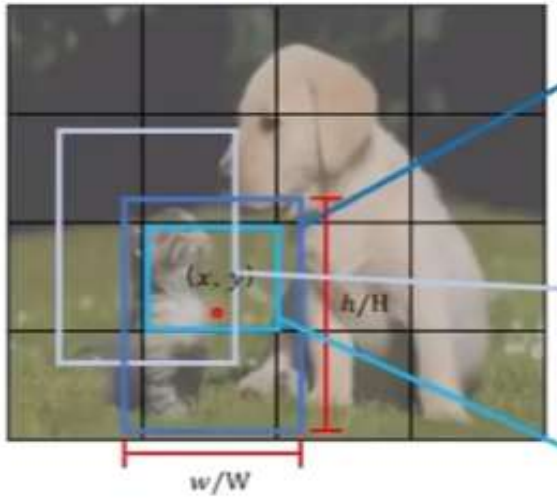
< source>https://curt-park.github.io/2017-03-26/yolo/

Unified Detection

예시) S = 4, B = 2, C = 20



Resized image 를
4x4 grid 로 분할



Grid cell 마다 bbox 2개씩 예측

bbox #1

bbox #2

x
y
w
h
p_c
x
y
w
h
p_c
c_1
c_2
c_3
⋮
c_{20}

bbox의 중심좌표의 위치
(grid cell 기준)

Input image W, H 로 normalize

p_c: $\Pr(Object) * IOU_{pred}^{truth}$

cf. $\Pr(Object)$:
물체가 bbox 내에 있으면 1,
없으면 0

$\Pr(Class_i|Object)$

: 물체가 bbox 내에 있을 때,
Grid cell에 있는 object가 i번째
class 에 속할 확률

출처: https://www.youtube.com/watch?v=O78V3kwBRBk

Unified Detection

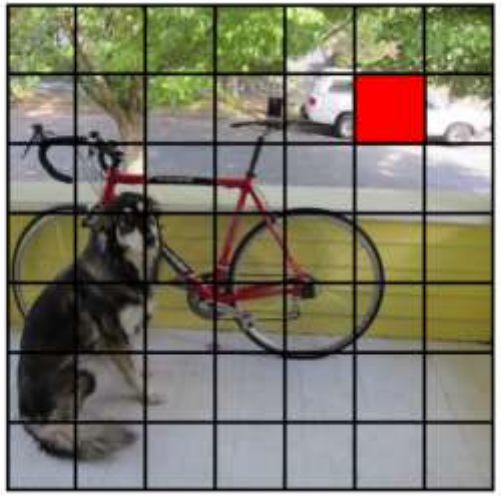
Test time에는 conditional class probability와 bounding box의 confidence score를 곱하여 class-specific confidence score를 얻는다.

$$
\begin{aligned}
Class Specific Confidence Score &= Conditional Class Probability * Confidence Score \\
&= Pr(Classi|Object) * Pr(Object) * IOU_{pred}^{truth} \\
&= Pr(Classi) * IOU_{pred}^{truth}
\end{aligned}
$$

논문에서는 YOLO의 성능평가를 위해 PASCAL VOC을 사용하였으며, S, B, C에는 각각 7, 2, 20이 할당되었다.

Unified Detection

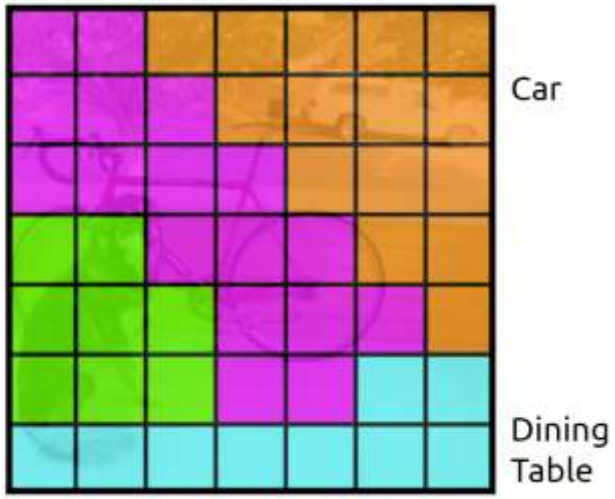
Each cell predicts boxes and confidences: P(Object)



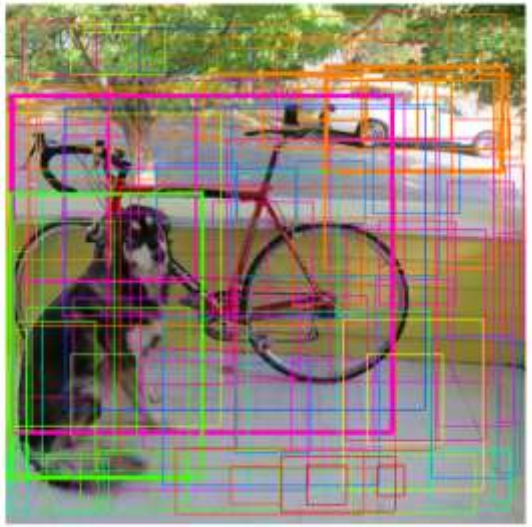
Each cell predicts boxes and confidences: P(Object)

Each cell predicts boxes and confidences: P(Object)

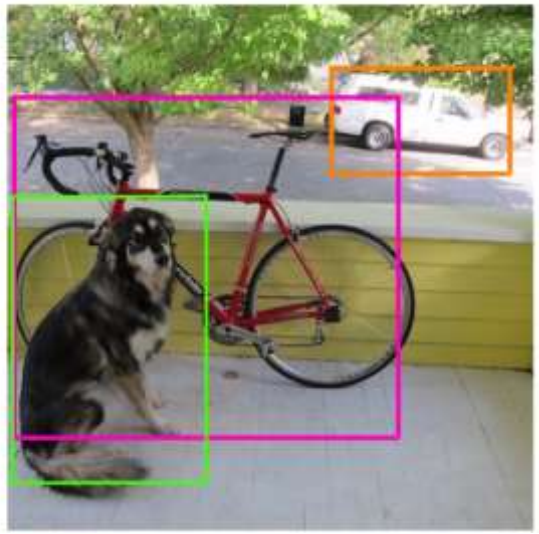
Conditioned on object: P(Car | Object)



Then we combine the box and class predictions.



Finally we do NMS and threshold detections



MAXIMUM 아니면 버림

Network Design : Yolo v1 GoogLeNet

Yolo v3 uses 53 CNNs layers(called Darknet-53)

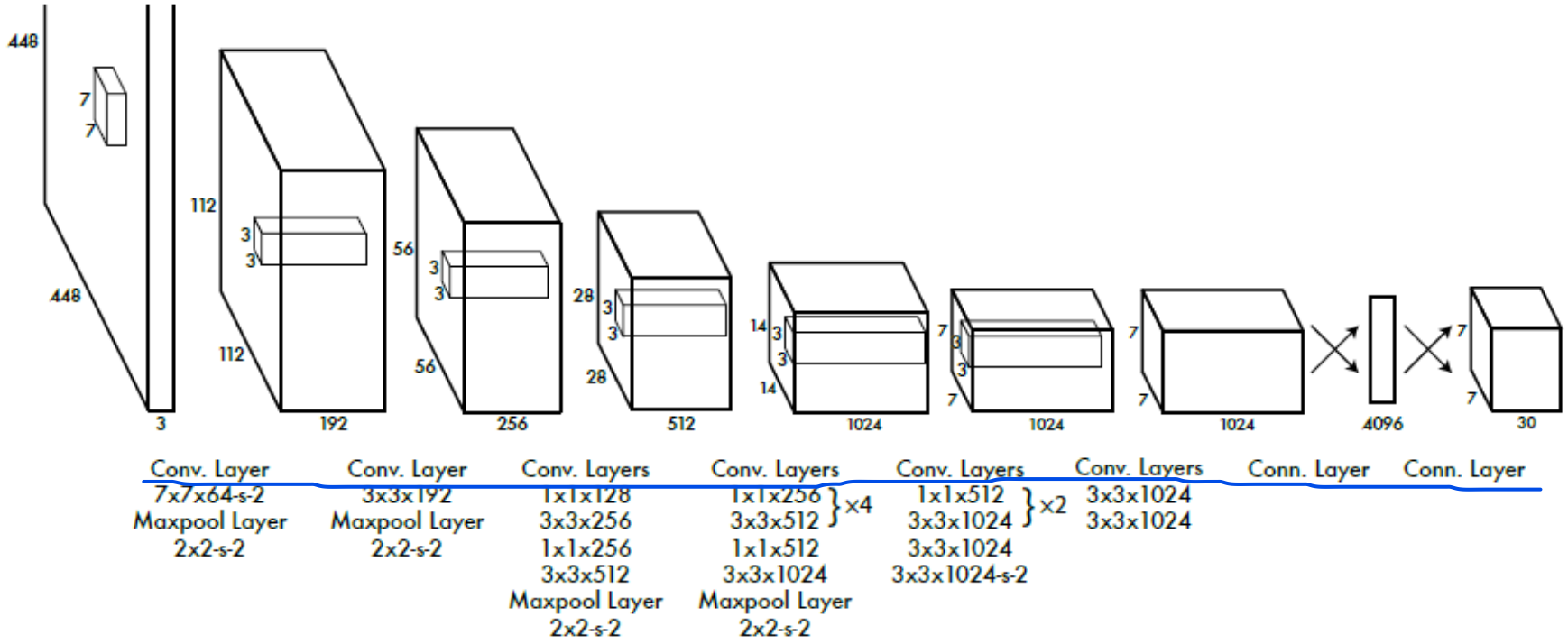
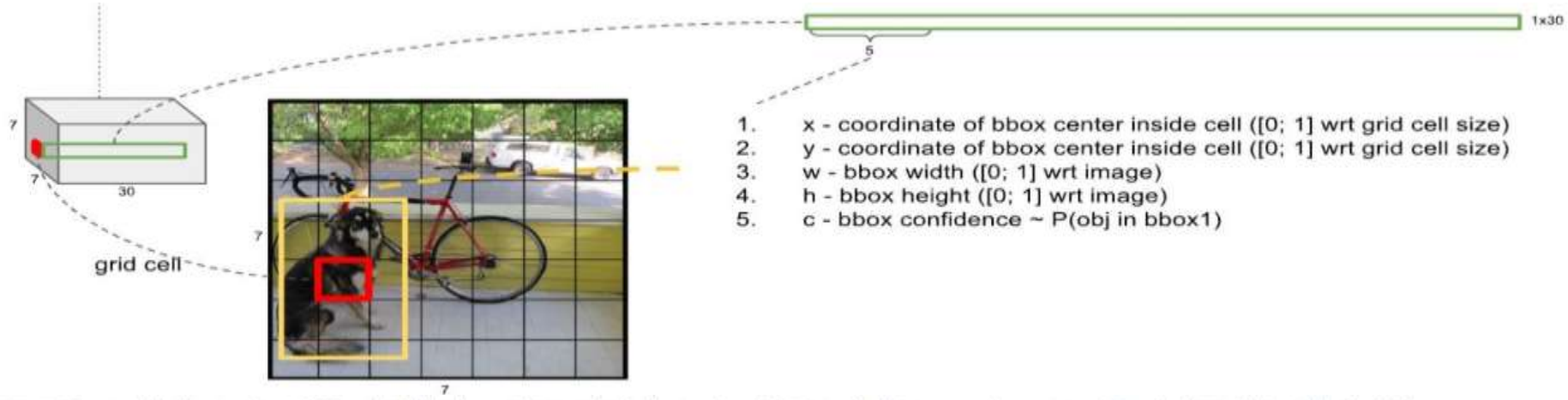
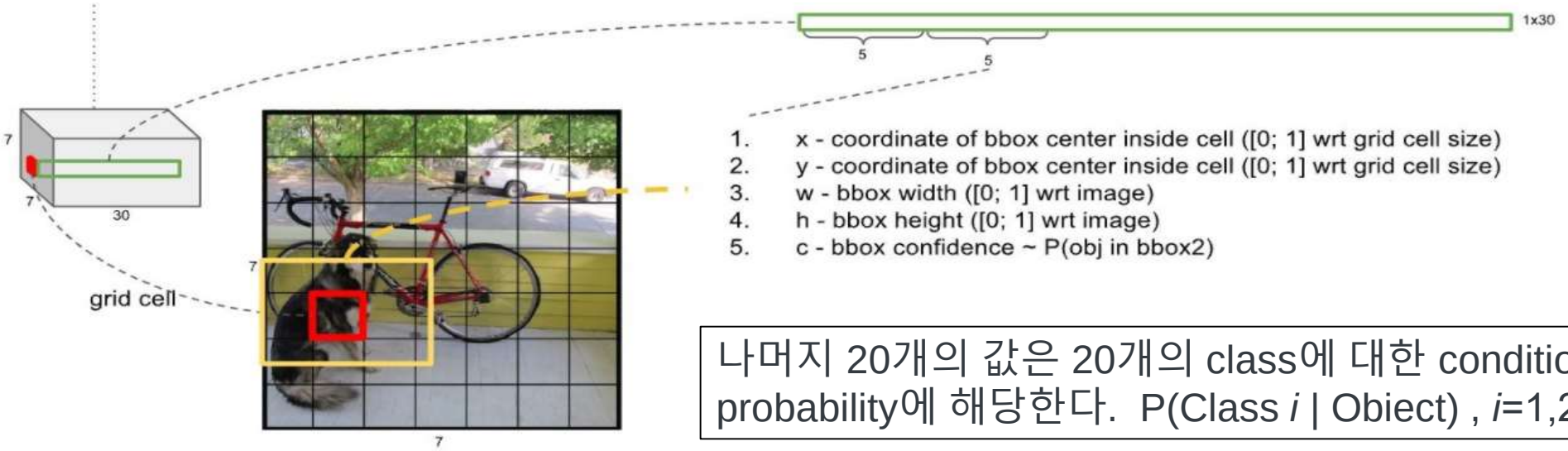


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.



7X7은 49개의 Grid Cell을 의미한다. 그리고 각각의 Grid cell은 B개의 bounding Box를 가지고 있는데(여기선 B=2), 앞 5개의 값은 해당 Grid cell의 첫 번째 bounding box에 대한 값이 채워져있다.



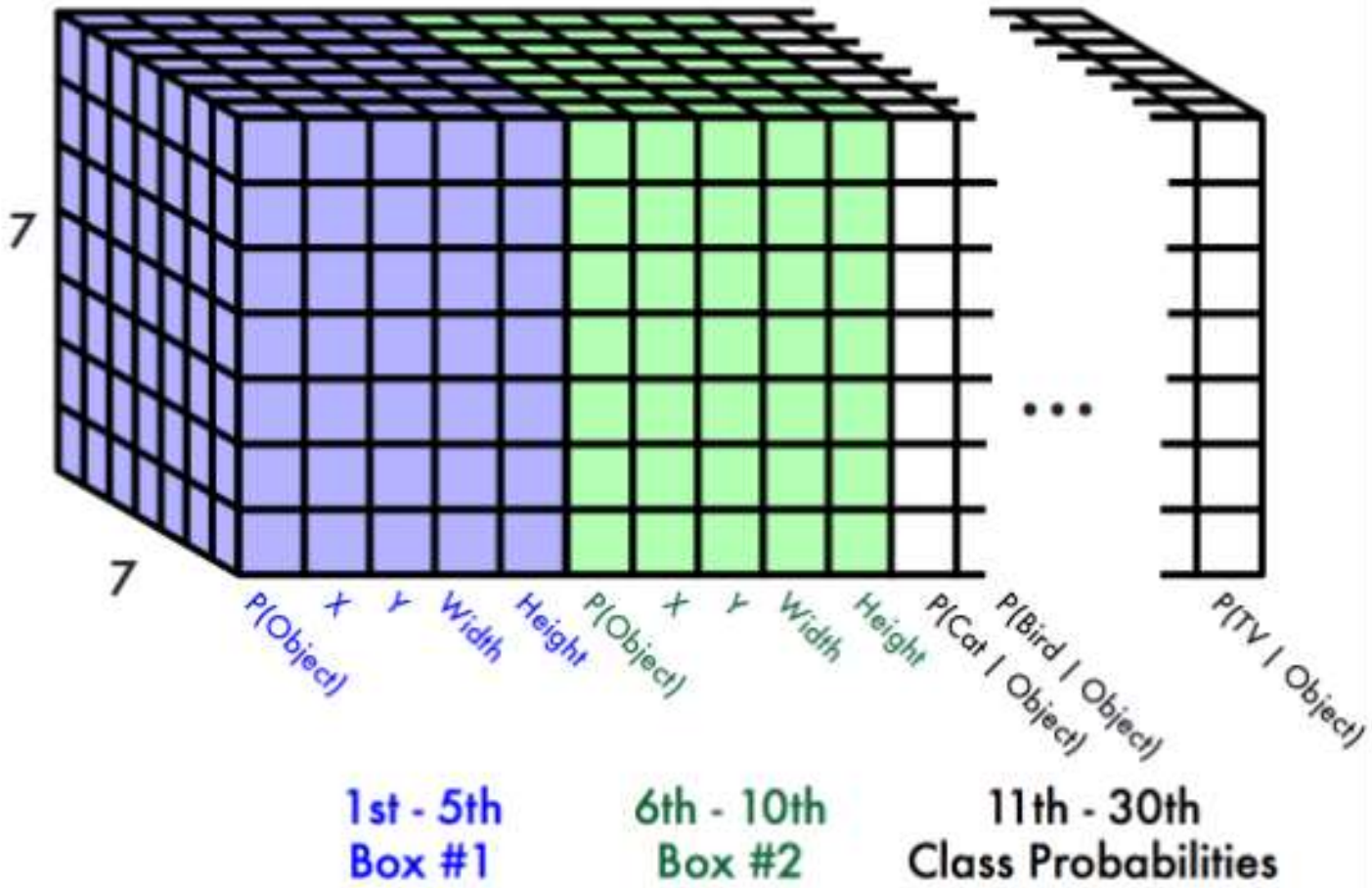
나머지 20개의 값은 20개의 class에 대한 conditional class probability에 해당한다. P(Class *i* | Obiect) , *i*=1,2 …,20

6~10번째 값은 두 번째 bounding box에 대한 내용이다.

Output(Yolo v1는 2개 , Yolo v3는 3개의 bounding box)

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities



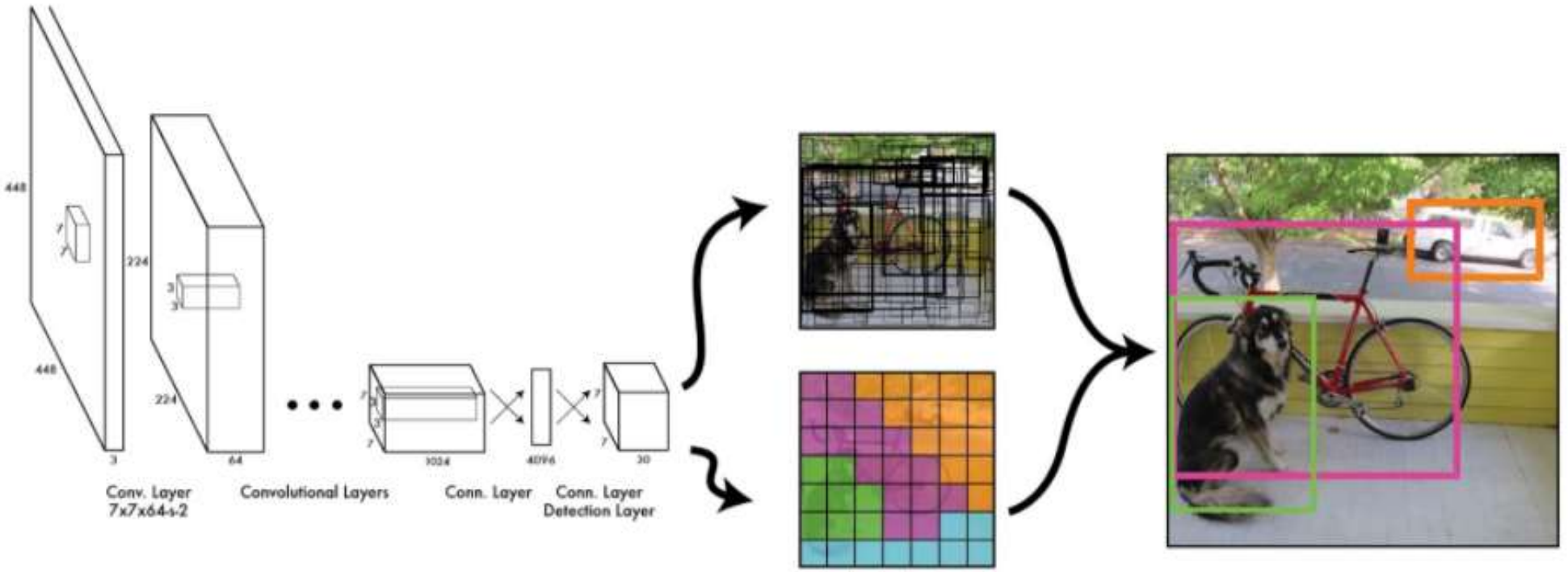
For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**

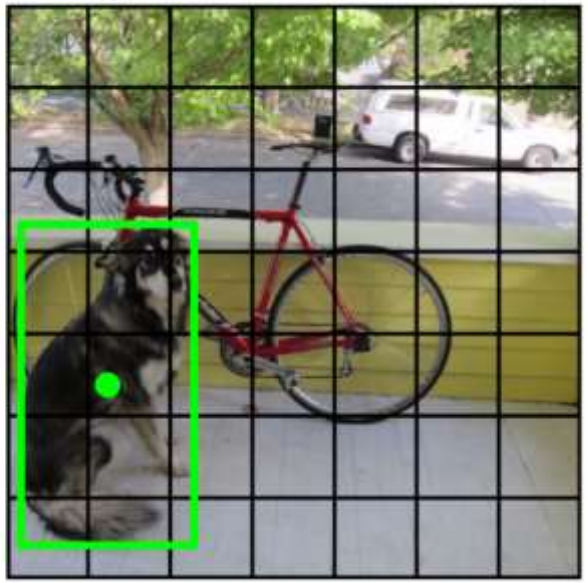
Output

Thus we can train one neural network to be a whole detection pipeline

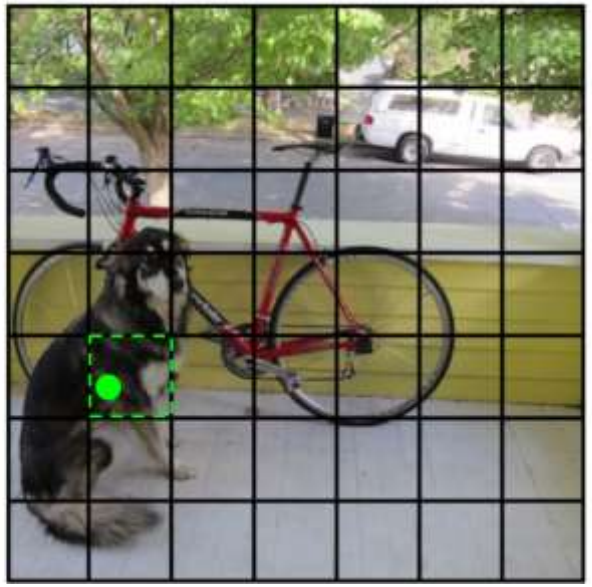


Training

During training, match example to the right cell

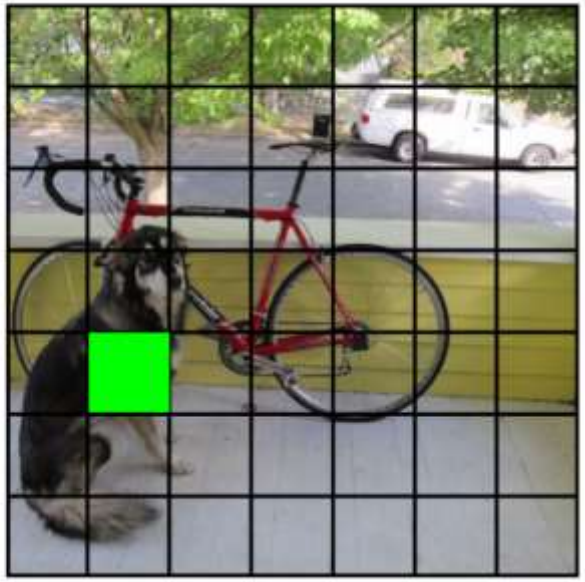


During training, match example to the right cell



Adjust that cell's class prediction

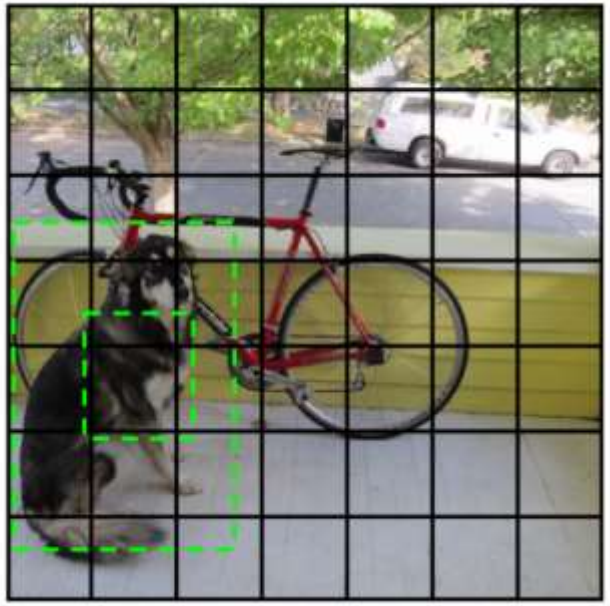
Dog = 1
Cat = 0
Bike = 0
...



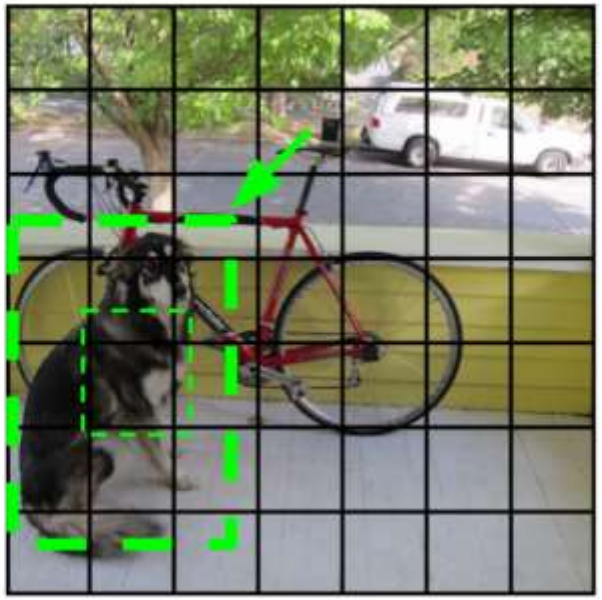
1. 특정 object에 responsible 한 cell을 찾는다:
   object를 포함하는 ground truth(GT) box의 중심을 포함하는 cell을 말함.
   위 예에서 dog에 대한 초록색 박스(GT)의 중심점을 포함하는 cell 이 dog에 responsible 한 cell이 됨
2. cell 에 대한 P(Class i | Object) 확률을 할당한다. P(dog | Object)=1,
   다른 classes에 대한 확률은 0으로 할당한다.

Training

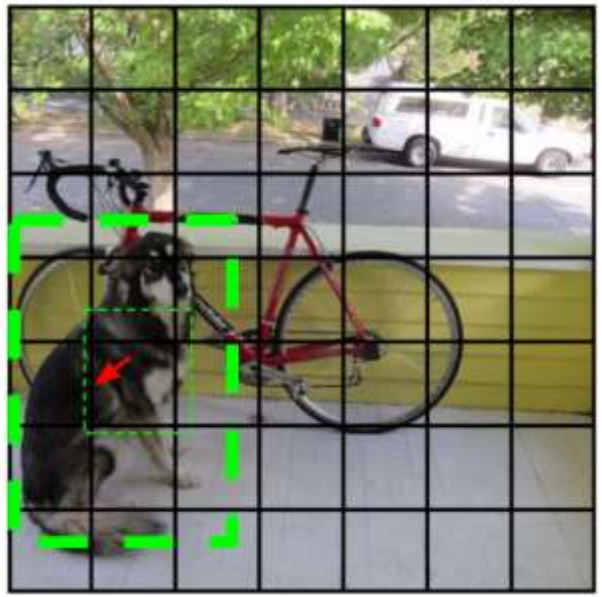
Look at that cell's predicted boxes



Find the best one, adjust it, increase the confidence



Decrease the confidence of other boxes



3. Yolo Conv layer를 통해 얻은 B개의 Bounding Boxes를 얻는다. 여기서는 B=2임.

4. 예측된 B(2)개의 bounding box 중에서 GT box와 IOU가 가장 높은 Bounding box를 사용해서 학습한다. 그리고 cell i 에서 선택된 박스 j 에 해당하는 $\mathbb{1}_{ij}^{\text{obj}} = 1$ 로 설정하여 loss function에 반영한다.

Train : Loss function

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

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

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \quad (3)$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \quad (4)$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \quad (5)$$

모든 grid cell에서 예측한 B개의 bbox의 좌표와 크기를 GT box와 비교

모든 grid cell에서 예측한 B개의 bbox의 Confidence Score를 GT box와 비교

모든 grid cell에서 예측한 P(class i| Object)를 GT 값과 비교

Train : Loss function 수식 설명

(1) Object가 존재하는 grid cell i의 predictor bounding box j에 대해, x와 y의 loss를 계산.
(2) Object가 존재하는 grid cell i의 predictor bounding box j에 대해, w와 h의 loss를 계산. 큰 box에 대해서는 small deviation을 반영하기 위해 제곱근을 취한 후, sum-squared error를 한다.(같은 error라도 larger box의 경우 상대적으로 IOU에 영향을 적게 준다.)
(3) Object가 존재하는 grid cell i의 predictor bounding box j에 대해, confidence score의 loss를 계산. (C_i = 1)
(4) Object가 존재하지 않는 grid cell i의 bounding box j에 대해, confidence score의 loss를 계산. (C_i = 0)
(5) Object가 존재하는 grid cell i에 대해, conditional class probability의 loss 계산. (Correct class c: $p_i(c)$ =1, otherwise: $p_i(c)$=0)

λ_{coord}: coordinates(x,y,w,h)에 대한 loss와 다른 loss들과의 균형을 위한 balancing parameter.
λ_{noobj}: obj가 있는 box와 없는 box간에 균형을 위한 balancing parameter. (일반적으로 image내에는 obj가 있는 cell보다는 obj가 없는 cell이 훨씬 많으므로)

Inference

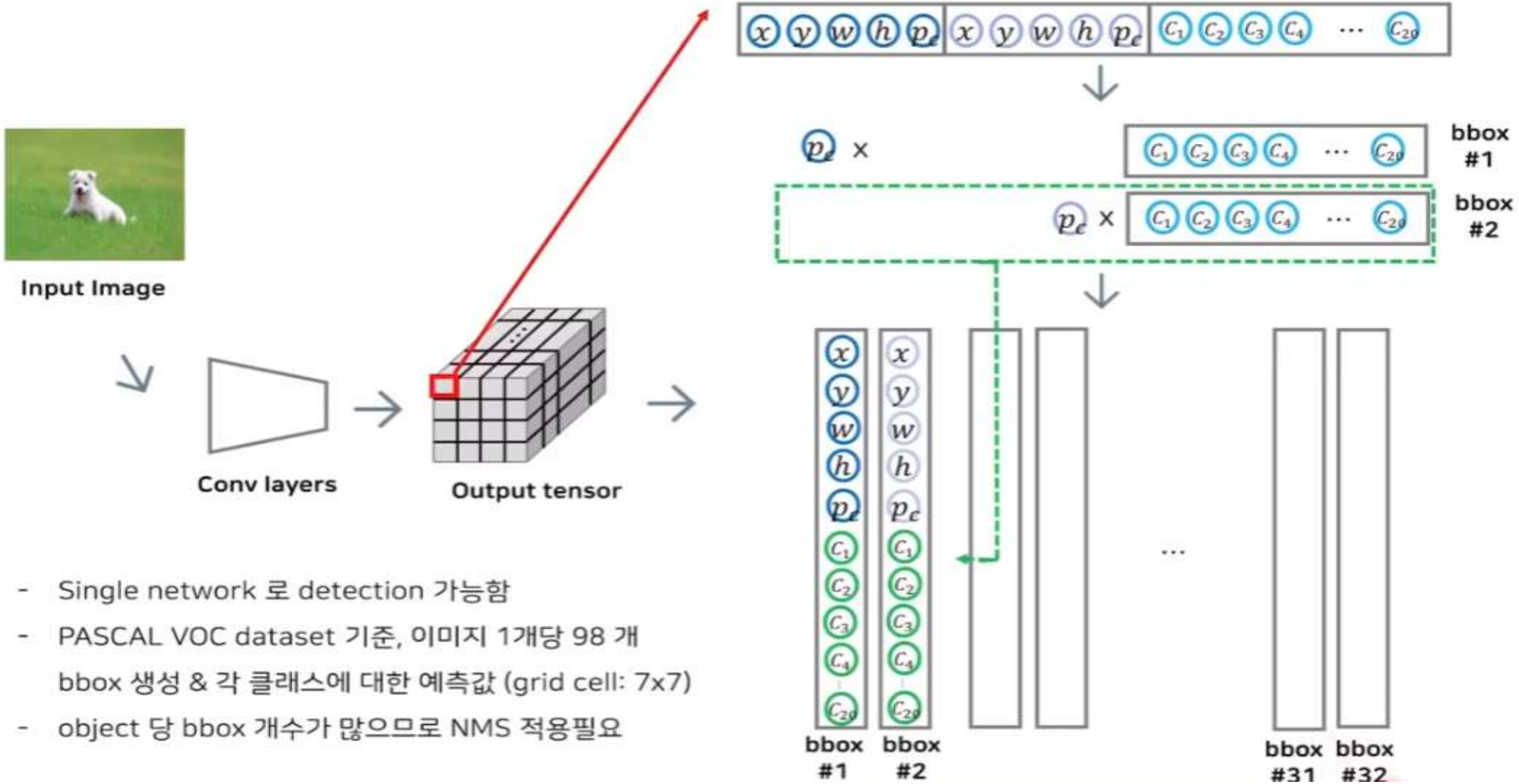
예측

첫 번째 bounding box의 confidence score와 각 conditional class probability를 곱하면 첫 번째 bounding box의 class specific confidence score가 나온다.

마찬가지로, 두 번째 bounding box의 confidence score와 각 conditional class probability를 곱하면 두 번째 bounding box의 class specific confidence score가 나온다.

이 계산을 각 bounding box에 대해 하게 되면 총 98개의 class specific confidence score를 얻을 수 있다.
이 98개의 class specific confidence score에 대해 각 20개의 클래스를 기준으로non-maximum suppresion을 하여, Object에 대한 Class 및 bounding box Location를 결정한다.

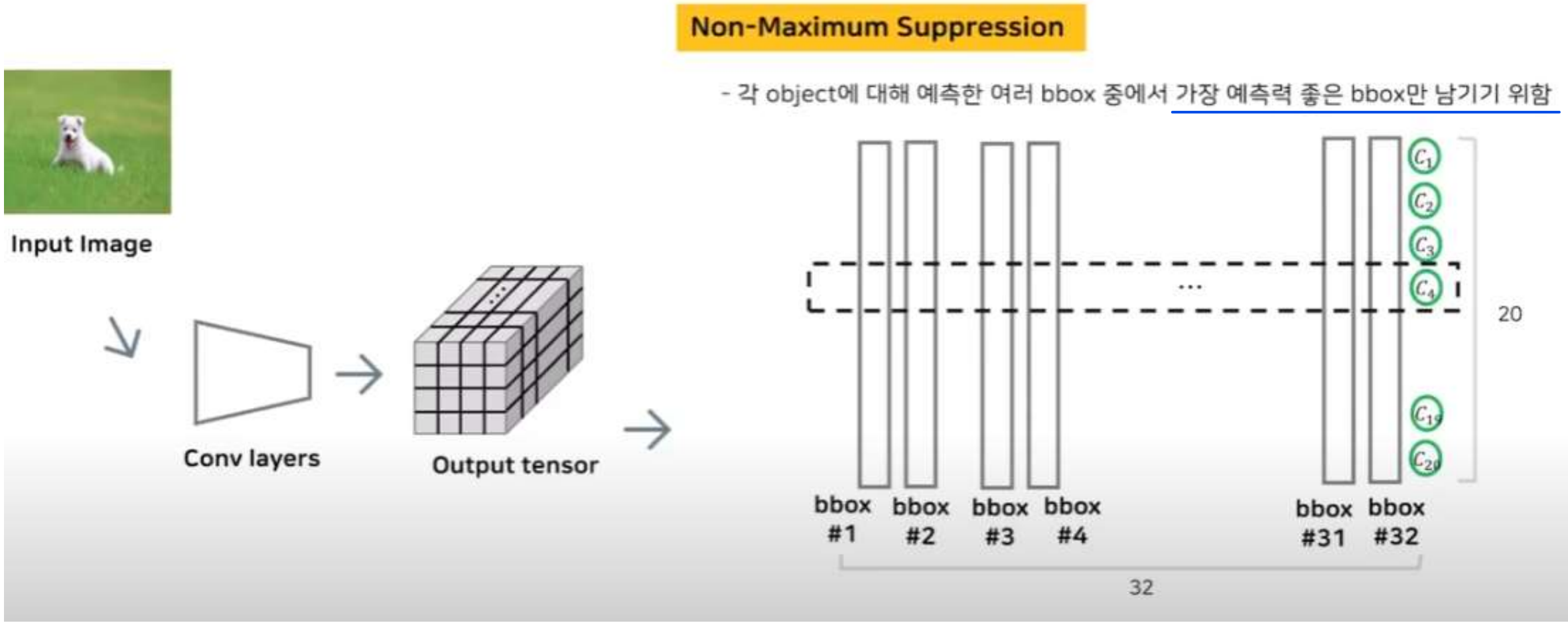
Inference



- Single network 로 detection 가능함
- PASCAL VOC dataset 기준, 이미지 1개당 98 개 bbox 생성 & 각 클래스에 대한 예측값 (grid cell: 7x7)
- object 당 bbox 개수가 많으므로 NMS 적용필요

출처: https://www.youtube.com/watch?v=O78V3kwBRBk

Inference

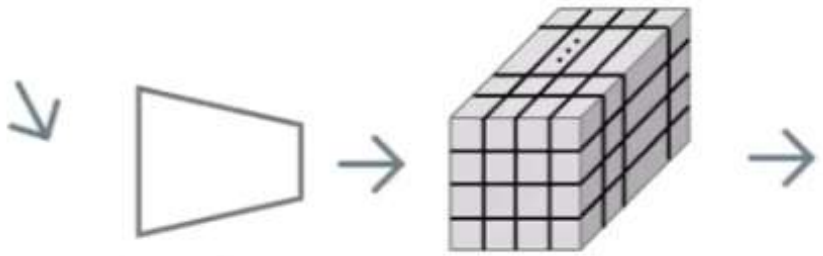


출처: https://www.youtube.com/watch?v=O78V3kwBRBk

Inference

3) 다른 class 속하는 Object

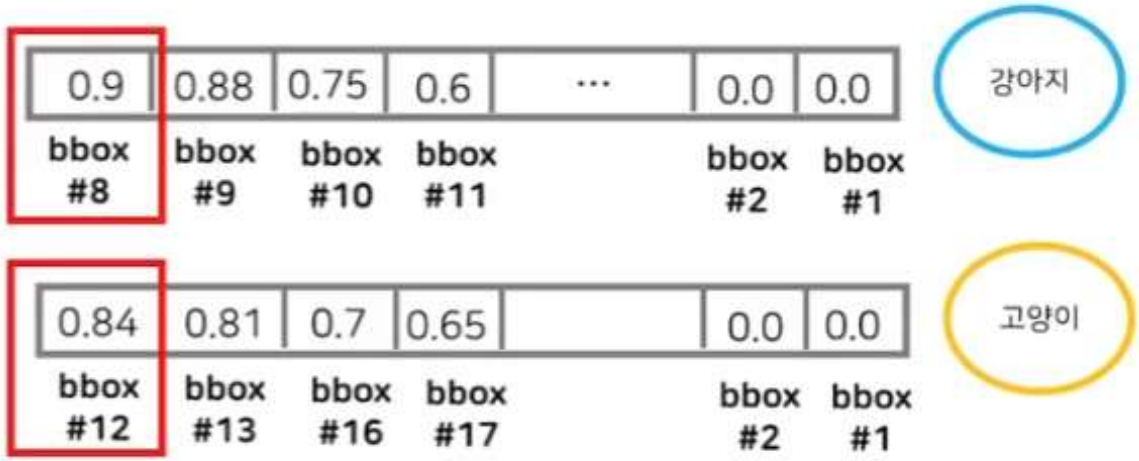
Input Image

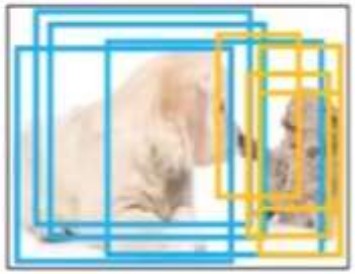


Non-Maximum Suppression

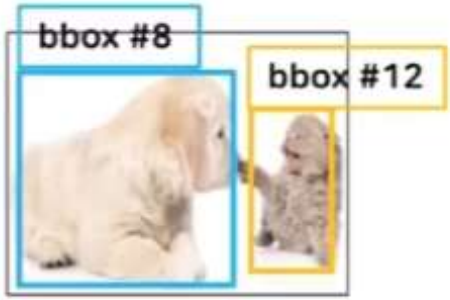
best bbox 와의 IOU를 비교해서 높은 값을 갖는 bbox는 best bbox와 겹치는 bbox이므로 삭제

예시)





NMS 적용 전



NMS 적용 후

출처: https://www.youtube.com/watch?v=O78V3kwBRBk

Limitation of YOLO

1. 각각의 grid cell이 하나의 클래스만을 예측할 수 있으므로, 작은 object 여러개가 다닥다닥 붙으면 제대로 예측하지 못한다.
2. bounding box의 형태가 training data를 통해서만 학습되므로, 새로운/독특한 형태의 bouding box의 경우 정확히 예측하지 못한다.
3. 몇 단계의 layer를 거쳐서 나온 feature map을 대상으로 bouding box를 예측하므로 localization이 다소 부정확해지는 경우가 있다.

You Only Look Once:
Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Generalizability: Person Detection in Artwork & Real-Time Detection In The Wild

Figure 5: Generalization results on Picasso and People-Art datasets.

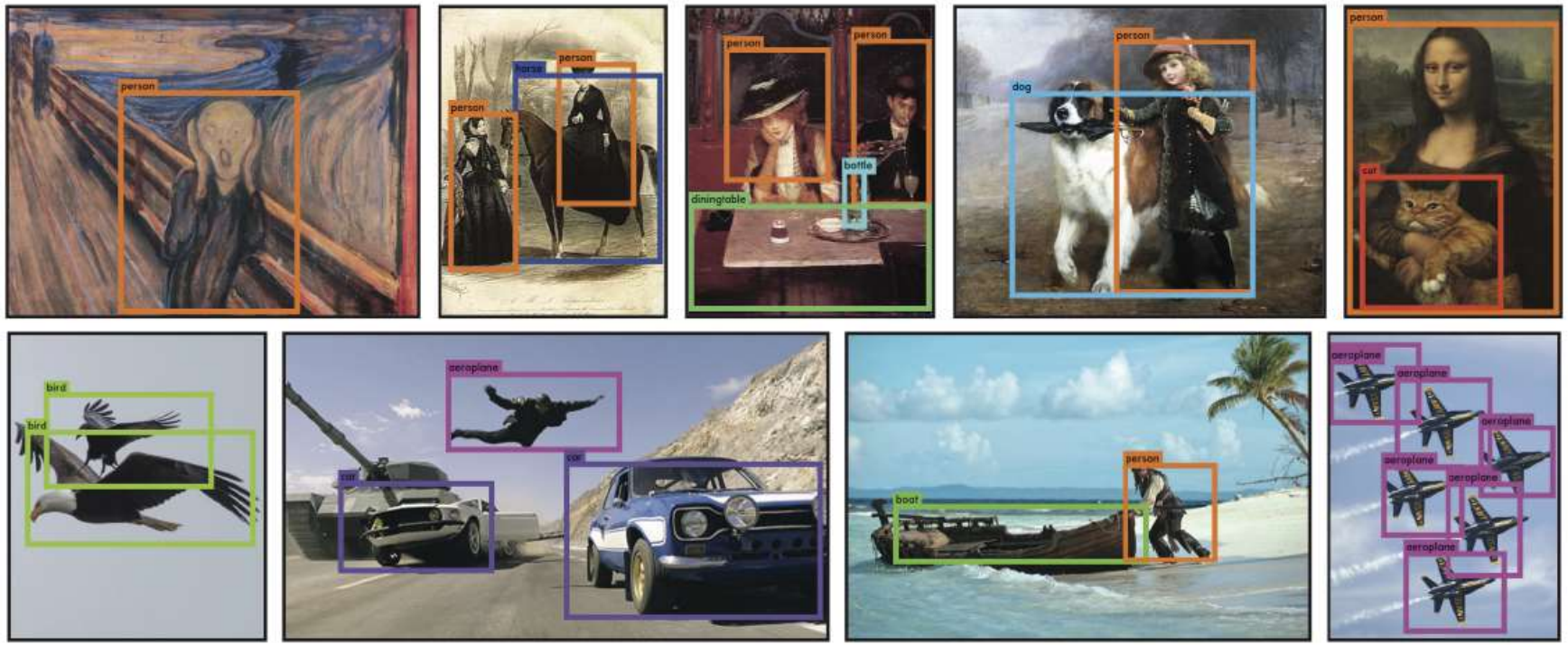


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

9.3.4 컨볼루션 신경망: YOLO 계열

■ YOLO의 버전업

- YOLO v3은 여러 스케일을 표현하여 다양한 크기의 물체 검출

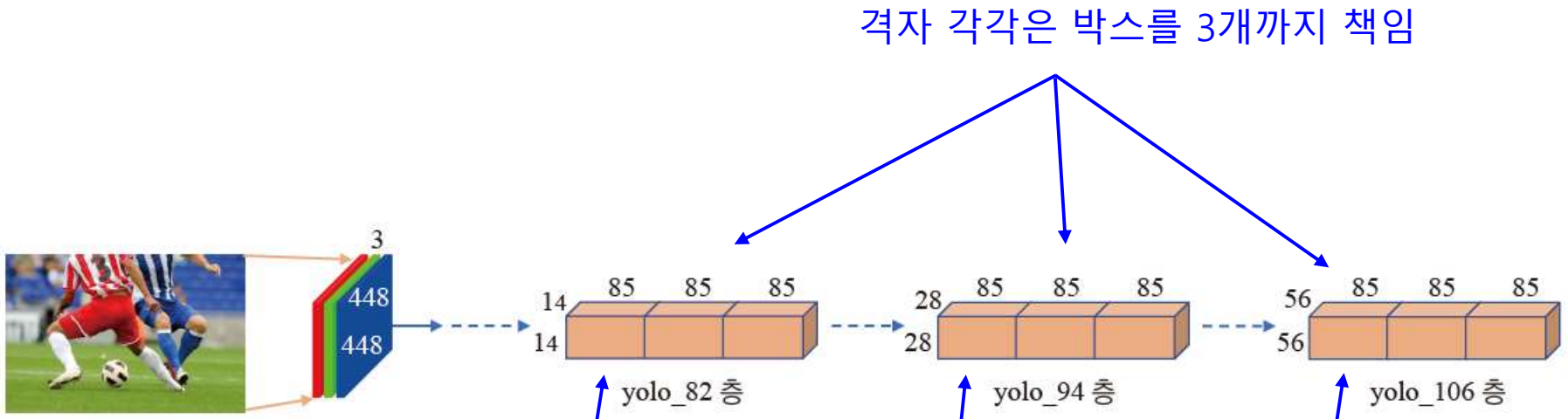


그림 9-26 다중 스케일 물체 검출을 수행하는 YOLO v3의 구조

14*14 격자로 나눔

28*28 격자로 나눔

56*56 격자로 나눔

9.3.4 컨볼루션 신경망: YOLO 계열

■ **정지 영상에서 물체 검출하는 프로그래밍:** COCO_names.txt, yolov3.weights, yolov3.cfg 필요

프로그램 9-1 YOLO v3으로 정지 영상에서 물체 검출하기

```
01  import numpy as np
02  import cv2 as cv
03  import sys
04
05  def construct_yolo_v3():
06      f=open('coco_names.txt', 'r')
07      class_names=[line.strip() for line in f.readlines()]
08
09      model=cv.dnn.readNet('yolov3.weights','yolov3.cfg')
10      layer_names=model.getLayerNames()
11      out_layers=[layer_names[i-1] for i in model.getUnconnectedOutLayers()]
12
13      return model,out_layers,class_names
14
```

사전 학습 모델을 읽어 YOLO 구성

readlines():각 줄을 element 로 하는 리스트를 반환
각 줄의 텍스트 마지막 줄바꿈 문자(₩n)도 포함

9.3.5 프로그래밍 실습: YOLO v3으로 물체 검출

```
15  def yolo_detect(img,yolo_model,out_layers):
16      height,width=img.shape[0],img.shape[1]
17      test_img=cv.dnn.blobFromImage(img,1.0/256,(448,448),(0,0,0),swapRB=True)
18
19      yolo_model.setInput(test_img)
20      output3=yolo_model.forward(out_layers)
21
22      box,conf,id=[],[],[]                    # 박스, 신뢰도, 부류 번호
23      for output in output3:
24          for vec85 in output:
25              scores=vec85[5:]
26              class_id=np.argmax(scores)
27              confidence=scores[class_id]
28              if confidence>0.5:              # 신뢰도가 50% 이상인 경우만 취함
29                  centerx,centery=int(vec85[0]*width),int(vec85[1]*height)
30                  w,h=int(vec85[2]*width),int(vec85[3]*height)
31                  x,y=int(centerx-w/2),int(centery-h/2)
32                  box.append([x,y,x+w,y+h])
33                  conf.append(float(confidence))
34                  id.append(class_id)
35
36      ind=cv.dnn.NMSBoxes(box,conf,0.5,0.4)
37      objects=[box[i]+[conf[i]]+[id[i]] for i in range(len(box)) if i in ind]
38      return objects
```

YOLO 모델로 물체를 검출

setInput: Sets the new input value for the network

9.3.5 프로그래밍 실습: YOLO v3으로 물체 검출

```
39
40  model,out_layers,class_names=construct_yolo_v3()          # YOLO 모델 생성
41  colors=np.random.uniform(0,255,size=(len(class_names),3)) # 부류마다 색깔
42
43  img=cv.imread('soccer.jpg')
44  if img is None: sys.exit('파일이 없습니다.')
45
46  res=yolo_detect(img,model,out_layers)                     # YOLO 모델로 물체 검출
47
48  for i in range(len(res)):                                 # 검출된 물체를 영상에 표시
49      x1,y1,x2,y2,confidence,id=res[i]
50      text=str(class_names[id])+'%.3f'%confidence
51      cv.rectangle(img,(x1,y1),(x2,y2),colors[id],2)
52      cv.putText(img,text,(x1,y1+30),cv.FONT_HERSHEY_PLAIN,1.5,colors[id],2)
53
54  cv.imshow("Object detection by YOLO v.3",img)
55
56  cv.waitKey()
57  cv.destroyAllWindows()
```

```
random.uniform(low=0.0, high=1.0, size=None)
    Draw samples from a uniform distribution.
```

다음은 Python 스트립 기능을 사용하는 이유입니다.

Python String strip(문자)

- 원래 문자열에서 제거하도록 지정된 문자를 기반으로 문자열 시작과 문자열 끝의 문자를 제거하는 데 도움이 됩니다.
- 주어진 문자가 원래 문자열과 일치하지 않으면 문자열을 있는 그대로 반환합니다.
- 제거할 문자를 지정하지 않으면 원래 문자열의 시작과 끝에서 공백이 제거됩니다.
- 시작 또는 끝에 공백이 없으면 문자열을 있는 그대로 반환합니다.

- getLayerNames():
  - Get the name of all layers of the network.
- getUnconnectedOutLayers():
  - Get the index of the output layers.

- Blob (Binary Large Object)
  - 이진 데이터를 나타내는 불변 객체다.
  - 텍스트와 이진 데이터를 다룰 수 있으며, 주로 이미지, 사운드, 비디오 같은 멀티미디어 객체

이 #Blob 라는 것은 동일한 방식으로 전처리 된 동일한 너비, 높이 및 채널 수를 가진 하나 이상의 이미지 말합니다.

◆ forward() [2/4]

```
void cv::dnn::Net::forward ( OutputArrayOfArrays  outputBlobs,
                             const String &       outputName = String()
                           )
```

Python:

```
cv.dnn.Net.forward(                [, outputName]                    ) -> retval
cv.dnn.Net.forward(                [, outputBlobs[, outputName]] )  -> outputBlobs
cv.dnn.Net.forward(                outBlobNames[, outputBlobs] )    -> outputBlobs
cv.dnn.Net.forwardAndRetrieve(  outBlobNames                        ) -> outputBlobs
```

Runs forward pass to compute output of layer with name `outputName`.

Parameters

outputBlobs contains all output blobs for specified layer.

outputName name for layer which output is needed to get

◆ blobFromImages() [1/2]

```
Mat cv::dnn::blobFromImages ( InputArrayOfArrays images,
                              double             scalefactor = 1.0 ,
                              Size               size = Size() ,
                              const Scalar &     mean = Scalar() ,
                              bool               swapRB = false ,
                              bool               crop = false ,
                              int                ddepth = CV_32F
                            )
```

Python:

cv.dnn.blobFromImages(images[, scalefactor[, size[, mean[, swapRB[, crop[, ddepth]]]]]]) -> retval

4-dimension: NCHW

Creates 4-dimensional blob from series of images. Optionally resizes and crops `images` from center, subtract `mean` values, scales values by `scalefactor`, swap Blue and Red channels.

Parameters

images	input images (all with 1-, 3- or 4-channels).
size	spatial size for output image
mean	scalar with mean values which are subtracted from channels. Values are intended to be in (mean-R, mean-G, mean-B) order if `image` has BGR ordering and `swapRB` is true.
scalefactor	multiplier for `images` values.
swapRB	flag which indicates that swap first and last channels in 3-channel image is necessary.
crop	flag which indicates whether image will be cropped after resize or not
ddepth	Depth of output blob. Choose CV_32F or CV_8U.

◆ NMSBoxes() [1/3]

```
void cv::dnn::NMSBoxes ( const std::vector< Rect > &  bboxes,
                         const std::vector< float > &  scores,
                         const float                   score_threshold,
                         const float                   nms_threshold,
                         std::vector< int > &          indices,
                         const float                   eta = 1.f ,
                         const int                     top_k = 0
                        )
```

Python:

```
cv.dnn.NMSBoxes(          bboxes, scores, score_threshold, nms_threshold[, eta[, top_k]] ) -> indices
cv.dnn.NMSBoxesRotated( bboxes, scores, score_threshold, nms_threshold[, eta[, top_k]] ) -> indices
```

```
#include <opencv2/dnn/dnn.hpp>
```

Performs non maximum suppression given boxes and corresponding scores.

Parameters

bboxes	a set of bounding boxes to apply NMS.
scores	a set of corresponding confidences.
score_threshold	a threshold used to filter boxes by score.
nms_threshold	a threshold used in non maximum suppression.
indices	the kept indices of bboxes after NMS.
eta	a coefficient in adaptive threshold formula: $nms\_threshold_{i+1} = eta \cdot nms\_threshold_i$.
top_k	if `>0`, keep at most `top_k` picked indices.

nms_threshold is the IOU threshold used in non-maximum suppression.

9.3.5 프로그래밍 실습: YOLO v3으로 물체 검출

1.00 신뢰도로 person 부류

0.941 신뢰도로 person 부류

0.999 신뢰도로 sports ball 부류

9.3.5 프로그래밍 실습: YOLO v3으로 물체 검출

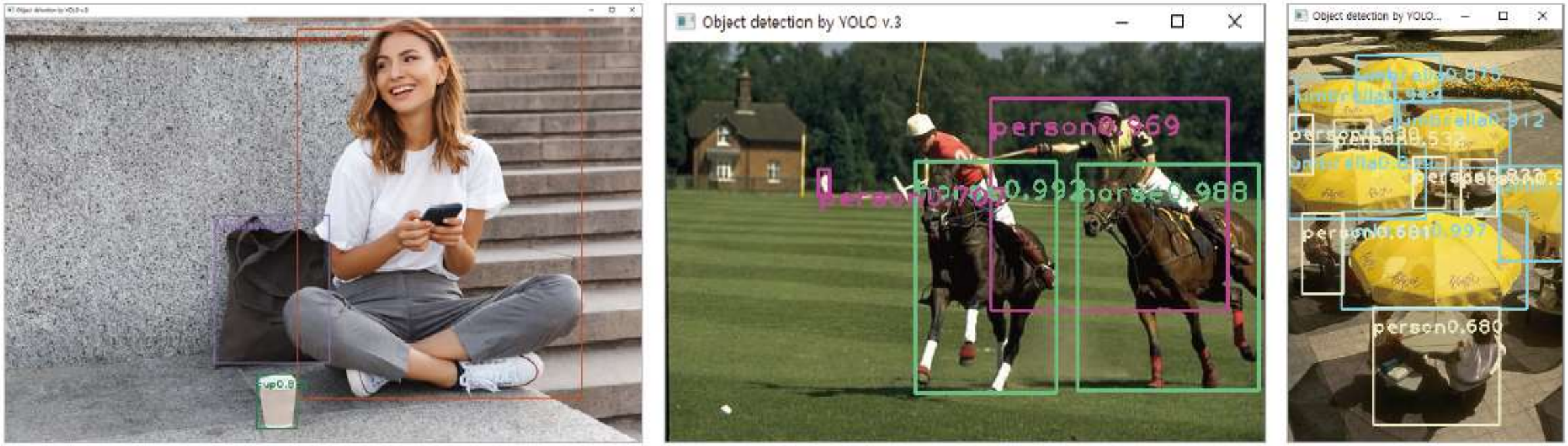


그림 9-27 YOLO v3으로 물체를 검출하는 여러 사례

9.3.5 프로그래밍 실습: YOLO v3으로 물체 검출

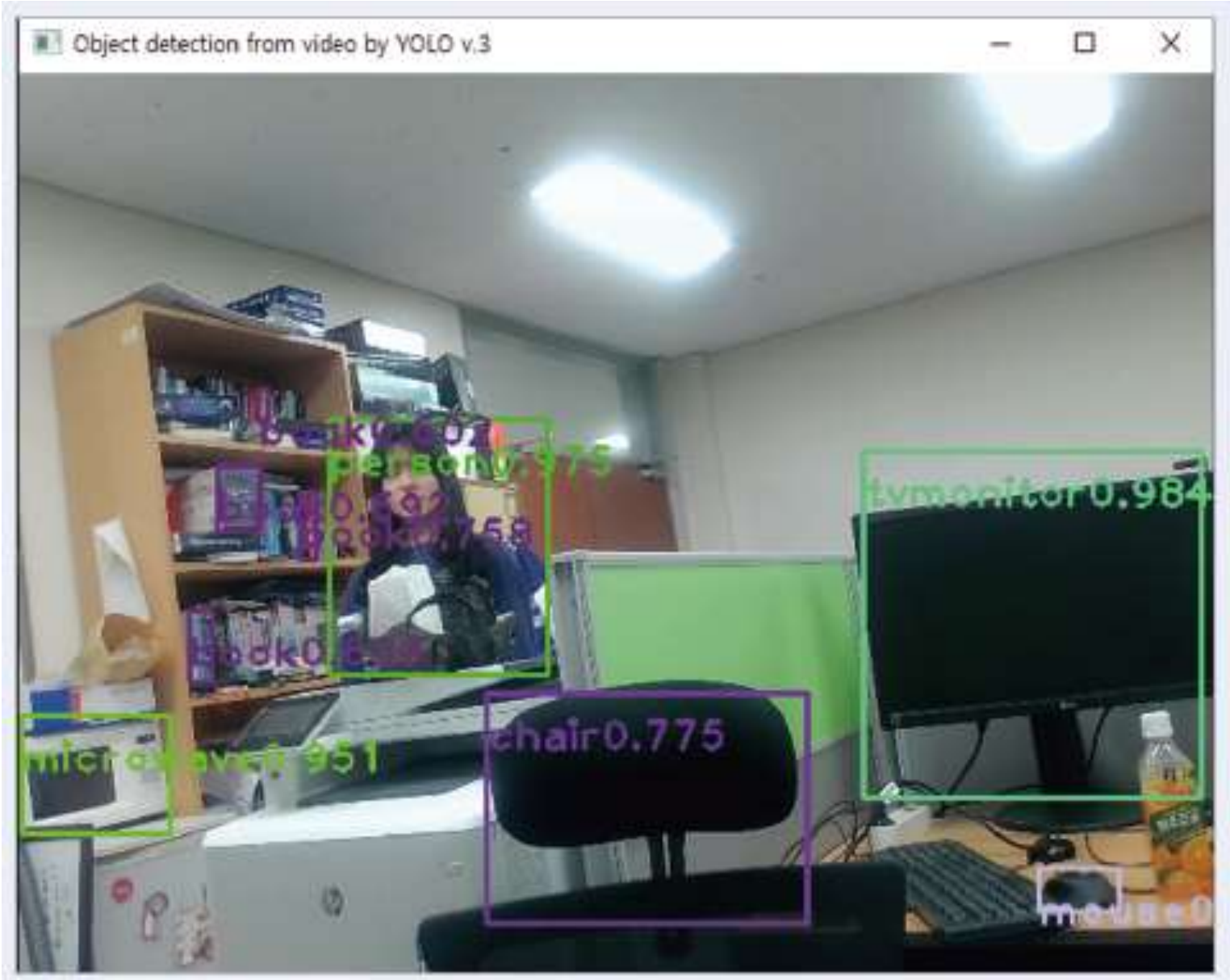
■ 비디오에서 물체 검출하는 프로그래밍

프로그램 9-2 YOLO v3으로 비디오에서 물체 검출하기

```
    1~41행은 [프로그램 9-1]과 같음

43  cap=cv.VideoCapture(0,cv.CAP_DSHOW)
44  if not cap.isOpened(): sys.exit('카메라 연결 실패')
45
46  while True:
47      ret,frame=cap.read()
48      if not ret: sys.exit('프레임 획득에 실패하여 루프를 나갑니다.')
49
50      res=yolo_detect(frame,model,out_layers)
51
52      for i in range(len(res)):
53          x1,y1,x2,y2,confidence,id=res[i]
54          text=str(class_names[id])+'%.3f'%confidence
55          cv.rectangle(frame,(x1,y1),(x2,y2),colors[id],2)
56          cv.putText(frame,text,(x1,y1+30),cv.FONT_HERSHEY_PLAIN,1.5,colors[id],2)
57
58      cv.imshow("Object detection from video by YOLO v.3",frame)
59
60      key=cv.waitKey(1)
61      if key==ord('q'): break
62
63  cap.release()     # 카메라와 연결을 끊음
64  cv.destroyAllWindows()
```

9.3.5 프로그래밍 실습: YOLO v3으로 물체 검출



9.3.5 프로그래밍 실습: YOLO v3으로 물체 검출

■ 비디오 처리량 측정하기

프로그램 9-3 YOLO v3의 비디오 처리량 측정하기

```
    1~41행은 [프로그램 9-1]과 같음

43  cap=cv.VideoCapture(0,cv.CAP_DSHOW)
44  if not cap.isOpened(): sys.exit('카메라 연결 실패')
45
46  import time
47
48  start=time.time()
49  n_frame=0
50  while True:
51      ret,frame=cap.read()
52      if not ret: sys.exit('프레임 획득에 실패하여 루프를 나갑니다.')
53
54      res=yolo_detect(frame,model,out_layers)
55
56      for i in range(len(res)):
57          x1,y1,x2,y2,confidence,id=res[i]
58          text=str(class_names[id])+'%.3f'%confidence
59          cv.rectangle(frame,(x1,y1),(x2,y2),colors[id],2)
60          cv.putText(frame,text,(x1,y1+30),cv.FONT_HERSHEY_PLAIN,1.5,colors[id],2)
61
62      cv.imshow("Object detection from video by YOLO v.3",frame)
63      n_frame+=1
64
65      key=cv.waitKey(1)
66      if key==ord('q'): break
```

9.3.5 프로그래밍 실습: YOLO v3으로 물체 검출

```
67
68  end=time.time()
69  print('처리한 프레임 수=',n_frame,', 경과 시간=',end-start,'\n초당 프레임 수=',n_frame/
    (end-start))
70
71  cap.release()        # 카메라와 연결을 끊음
72  cv.destroyAllWindows()
```

```
처리한 프레임 수= 82, 경과 시간= 29.883725881576538
초당 프레임 수= 2.7439684169554437
```

초당 2.74 프레임 처리

과제

- 다양한 물체(차, 오토바이, 자전거 버스, 의자,개 등)가 포함된 사진을 사용하여 주어진 YOLO3 프로그램을 사용하여 물체에 대한 검출과 분류 확률을 구하여서 원본과 결과물을 제출하시오.