

Object Detection

Localization, Detection, Segmentation (semantic/instance)

Single Object

• Classification

- 이미지 내 object를 통해 class 구분 (1 image : 1 class)

Single Object

• Localization

- 이미지 내 object의 위치 찾기 (regression)
- 주로 Bounding Box 이용 & Bounding Box의 좌표값을 출력

Multiple Objects

• Object Detection

- 이미지 내 object를 classification & localization 동시에 수행

Multiple Objects

• Segmentation

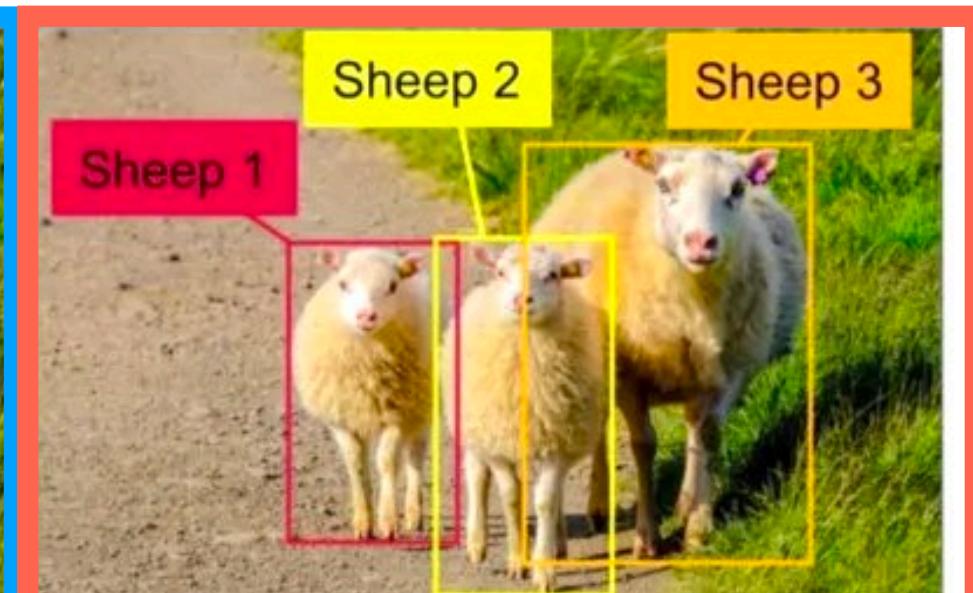
- 이미지의 pixel 단위로 classification
 - ▶ Semantic Segmentation: 클래스 내 구분 x
 - ▶ Instance Segmentation: 같은 클래스이어도 instance를 구분

Single Object

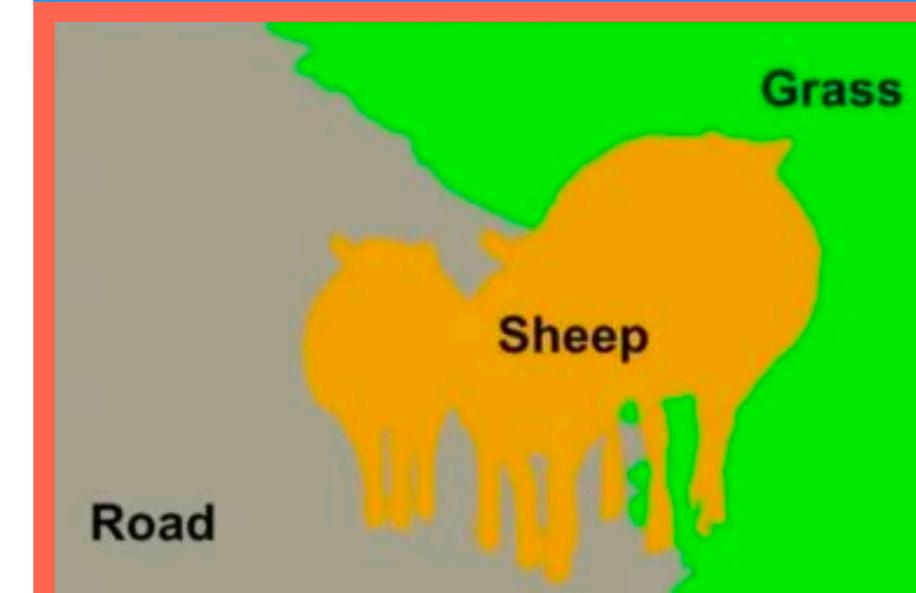


Classification + Localization

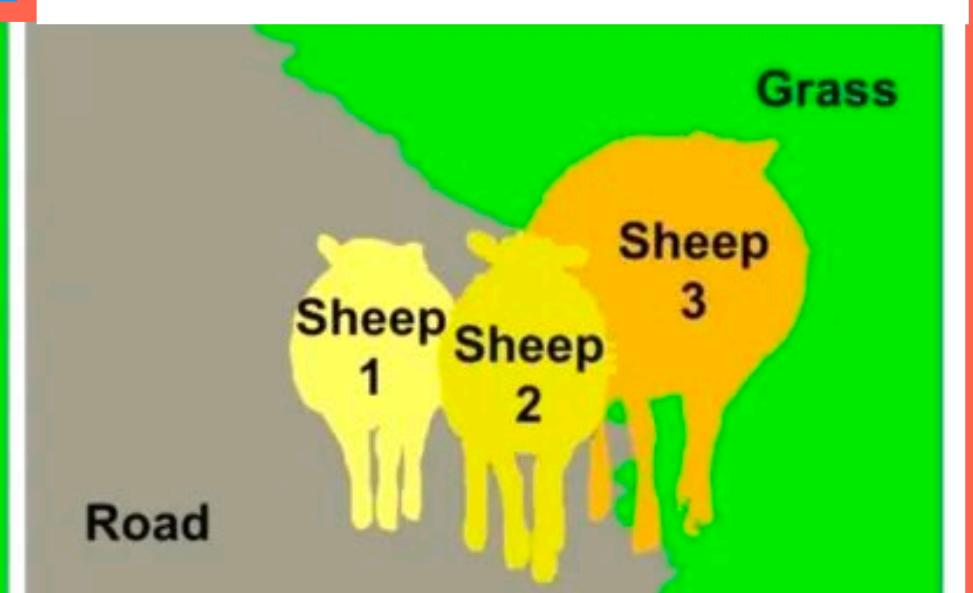
Multiple Objects



Object Detection



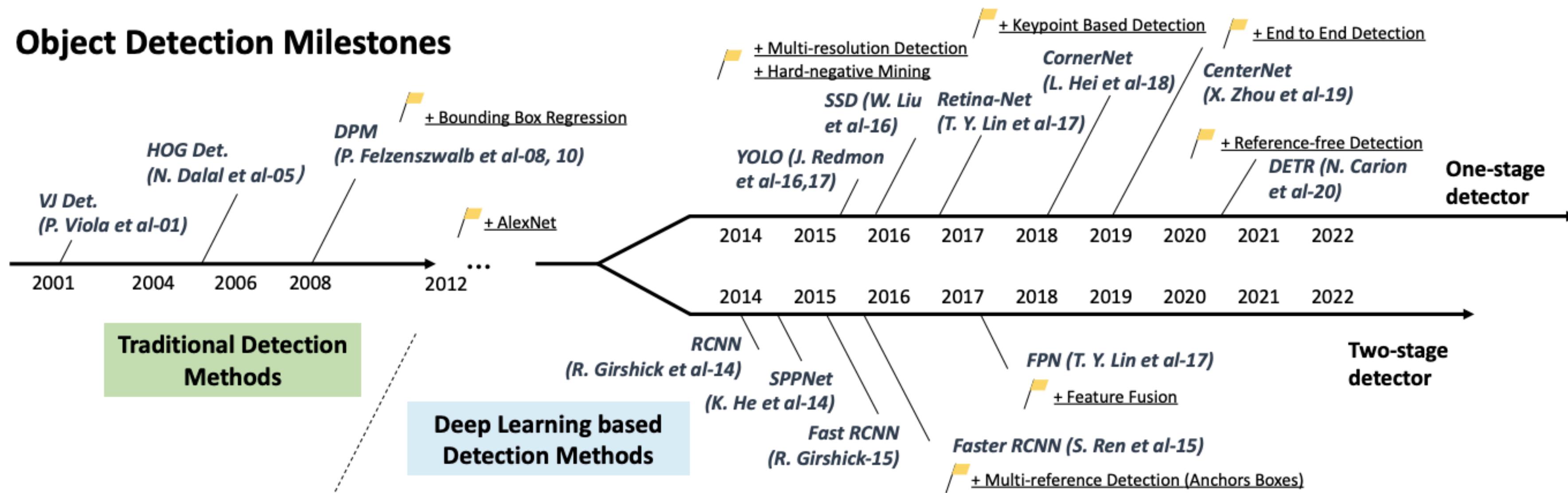
Semantic Segmentation



Instance Segmentation

Object Detection

two-stage VS one-stage



- **localization**
 - 위치 찾는 문제 (output: 이미지내 위치 (x, y, w, h))
- **classification**
 - 분류 문제 (output: 클래스 (softmax의 경우, 각 클래스의 확률))
- **two stage Detector**
 - localization + classification를 순차적으로 해결
- **one stage Detector**
 - localization & classification를 한 번에 해결

Object Detection

개념 정리 - 목차

1. region proposal

Sliding Window

Selective Search

2. Deep Learning Network

Backbone

Neck

Head

3. 그 외

IOU

NMS

xxxxxxxxxxxxx 다루지 않음xxxxxxxx

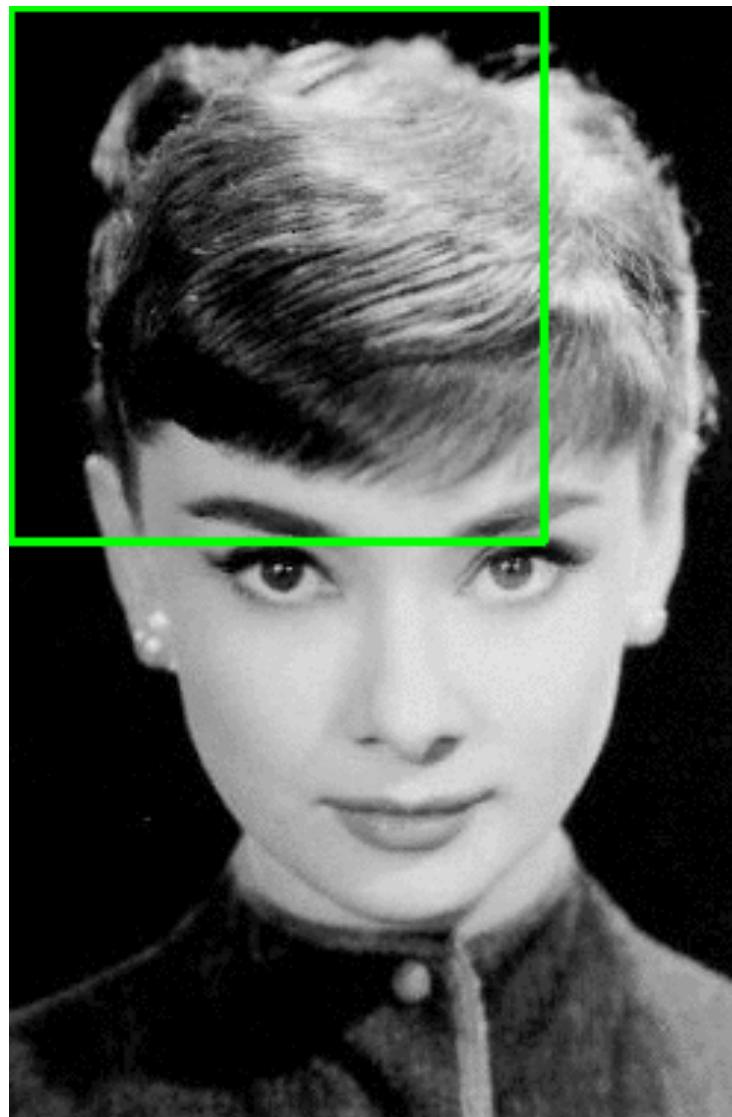
모델성능지표 (recall, precision, AP, mAP)

앵커 박스(Anchor Box) - YOLO v2에서 도입

Concept 1

region proposal

bounding box regression(bBox Regression) 학습만으로 inference 하기 어려움 -> Object가 있을만한 위치를 미리 예측하는 과정 추가



- **Sliding Window**

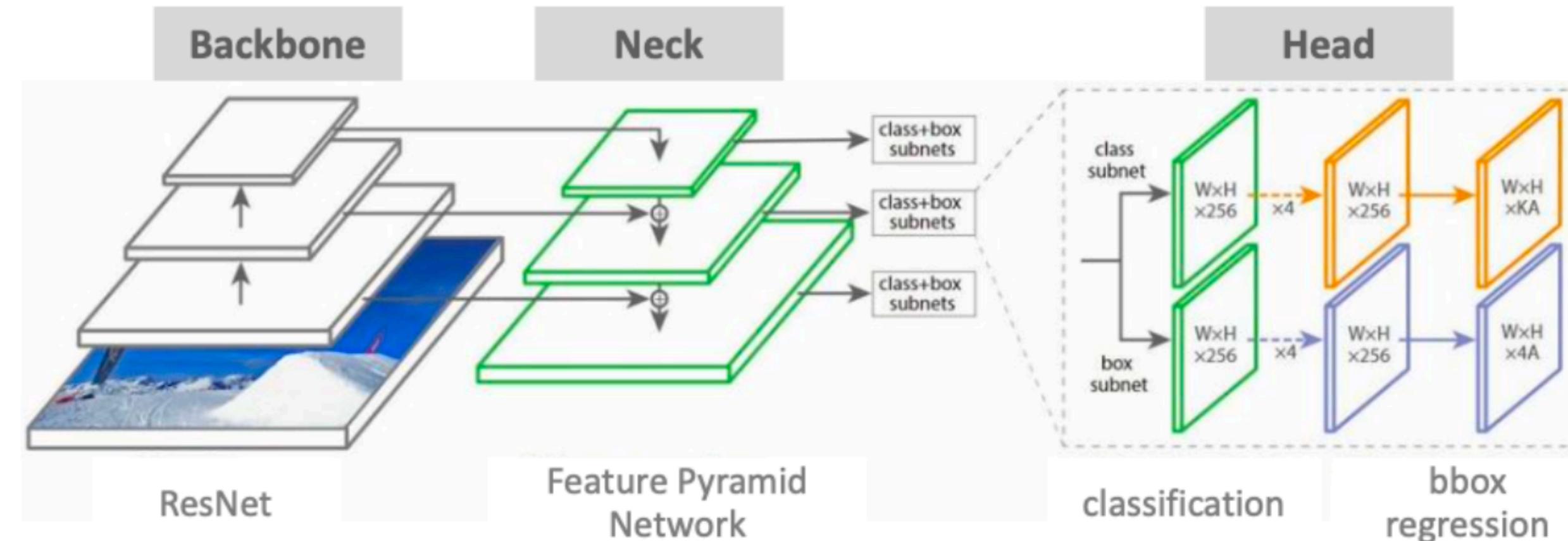
- 모든 영역의 이미지 -> window를 sliding하면서 detect
- (단점) 오래 걸림 - object가 없어도 무조건 sliding 해야함
 - 여러 형태의 window를 각각 sliding
 - window scale은 고정하고, 이미지 scale 변경하는 방식
- Deformable Parts Models(DPM) 모델 등 초기 기법

- **Selective Search**

- 1) segmentation
 - 인접한 영역끼리 유사성을 측정해 큰 영역으로 통합
- 2) 개별 segment를 bounding box로 만듬
- R-CNN 모델 등

Concept 2

Object Detection 일반적 Deep Learning Network 구조



- **Backbone**
 - Feature Extraction
- **Neck**
 - 작은 objects에 대한 정보를 체계화
-> BackBone 처리를 용이하게
- **Head**
 - Classification + bBox Regression

Concept 3 (1)

IoU (Intersection Over Union)

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



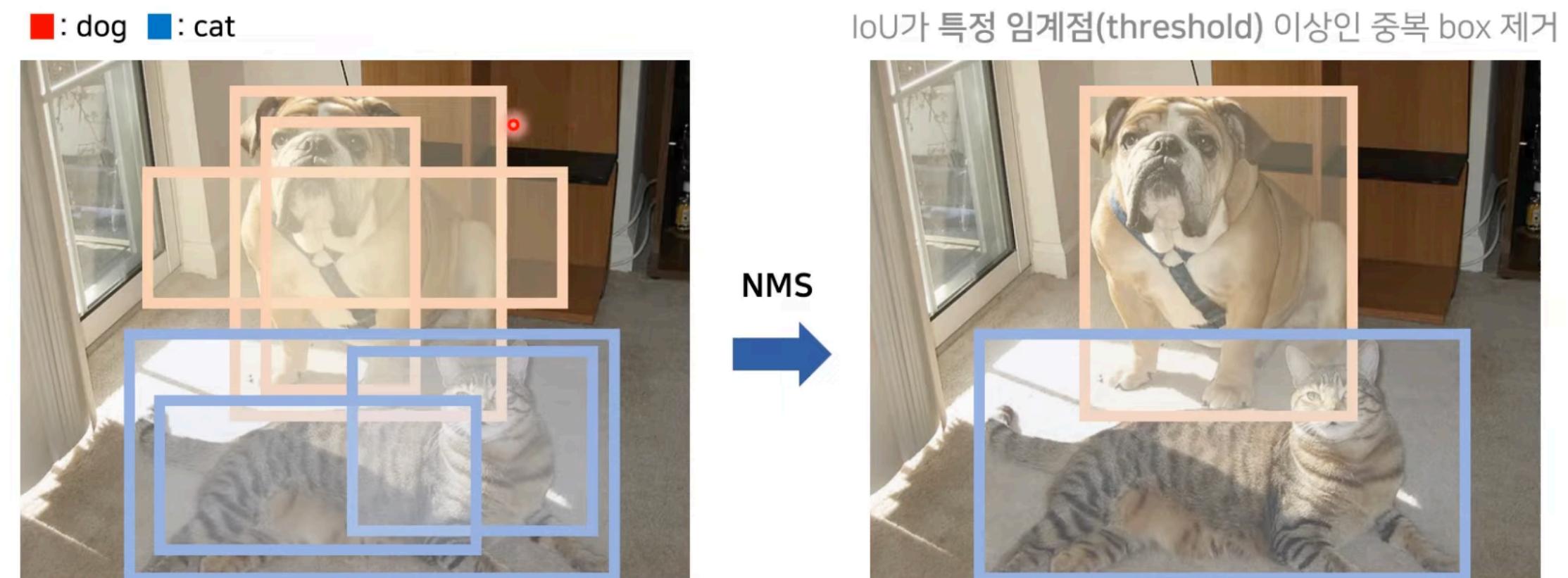
- IOU
 - 실제값(Ground Truth)과 예측값이 얼마나 겹치는지를 나타내는 지표 (교집합/합집합)

- IOU가 높을수록 잘 예측한 모델 (green: GT)

Concept 3 (2)

NMS (Non-Maximum Suppression)

- one instance에 one bounding box가 적용
- 하나의 클래스 내 겹치는 상자들을 제거하는 과정
- NMS 알고리즘
 - 같은 클래스 내의 bounding box 후보를 box의 confidence score 기준으로 정렬
 - IoU가 confidence threshold 이상인 중복 box 제거 (같은 객체를 detect한 box이므로)
 - 위 두 단계를 반복하면,
 - 최종적으로 각 object마다 하나의 bounding box만 남게 됨



YOLO

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

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

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network pre-

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only look once (YOLO) at an image to predict what objects are present and where they are.

Abstract

1. Introduction

- **Unified:** one-stage detector (localization & classification를 한번에)
 - Detection 문제를 하나의 회귀 문제로 재정의
 - Bounding-box와 Class probability를 하나의 문제로 간주
 - Bounding-box coordinates & class probabilities를 한번에 예측
- **Real-Time:** fast -> 실시간 가능 (중요도: accuracy < speed)
 - YOLO: 45 fps, Fast YOLO(A smaller version of the network): 155 fps
 - 이후, YOLO가 version up이 되면서, speed 뿐만 아니라 accuracy(mAP)도 확연히 증가함
- **sliding window** 방식이 아닌 CNN 사용 -> 이미지를 한 번만 봄 -> Global Context가 잘 고려되어 학습됨
 - false positives on background 적지만, 작은 객체를 탐지하는데는 어려움

YOLO

CNN의 중요성 (Unified Detection 핵심 역할)

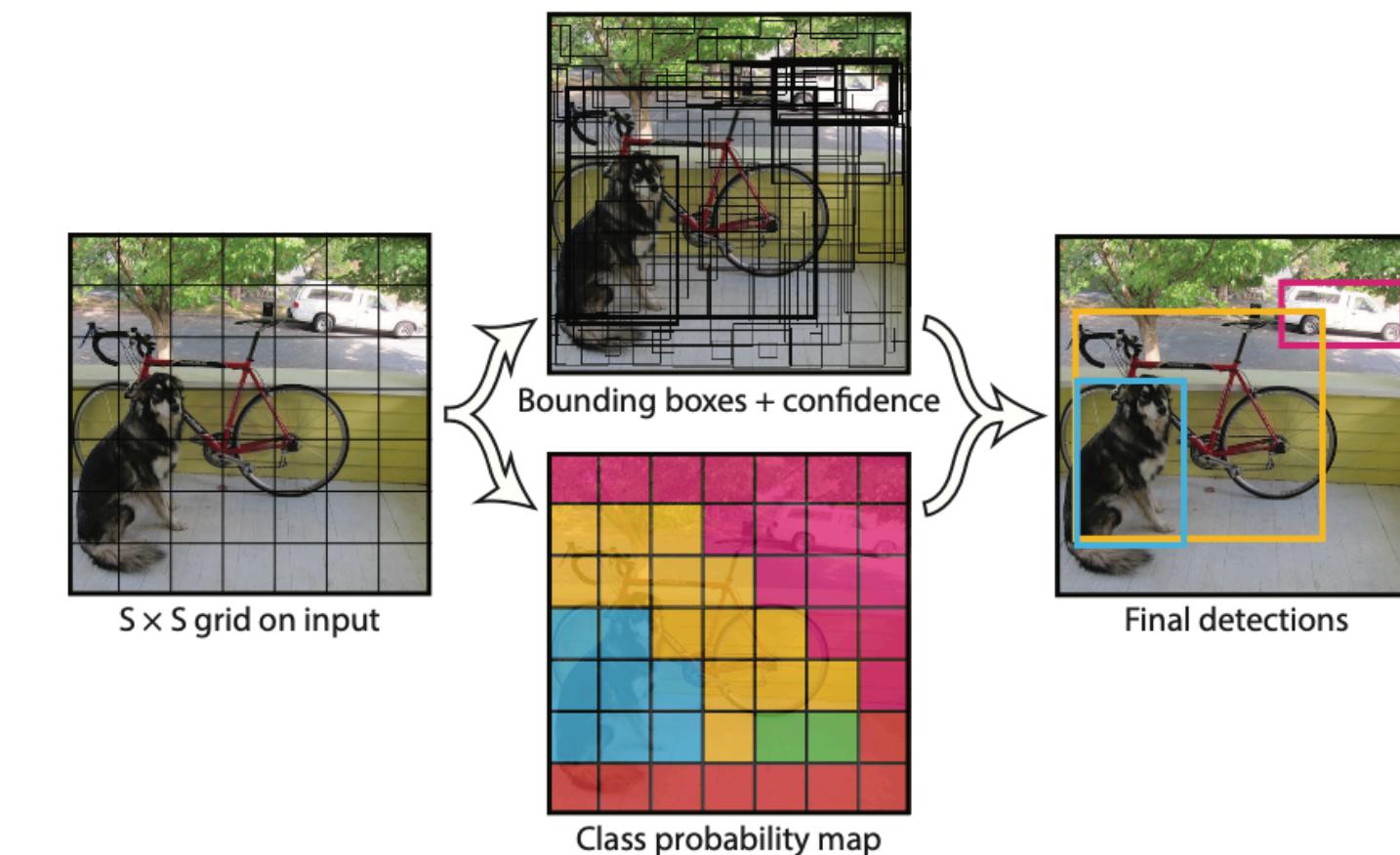
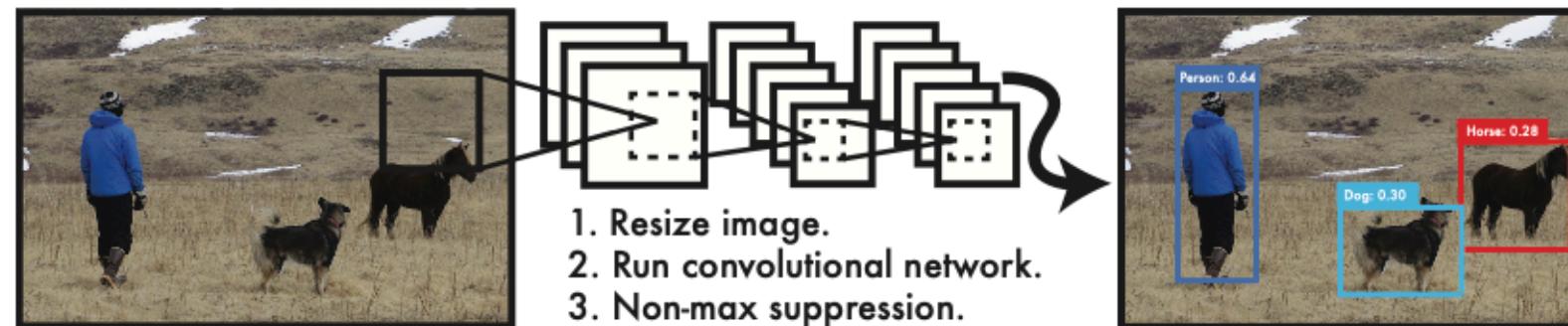


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448 , (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

1. Introduction

- **End-to-End 방식의 통합된 구조로, feature extraction, classification, bBox regression를 하나의 CNN으로 수행**

- 이미지를 CNN에서 한 번 inference -> 동시에 다수의 bounding box와 class 확률을 구함

- **output: $S \times S \times (B * 5 + C)$ tensor**

- $S \times S$ 그리드 셀로 나눔 -> 각 셀마다 (bBox regression + confidence) & classification 결과를 구함

- **confidence (score)**

- 바운딩 박스 내에 객체가 존재할 확률

- we define **confidence** as $\text{Pr}(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$.

- $0 \leq \text{confidence} \leq 1$ (셀 내 object가 없으면, confidence=0)

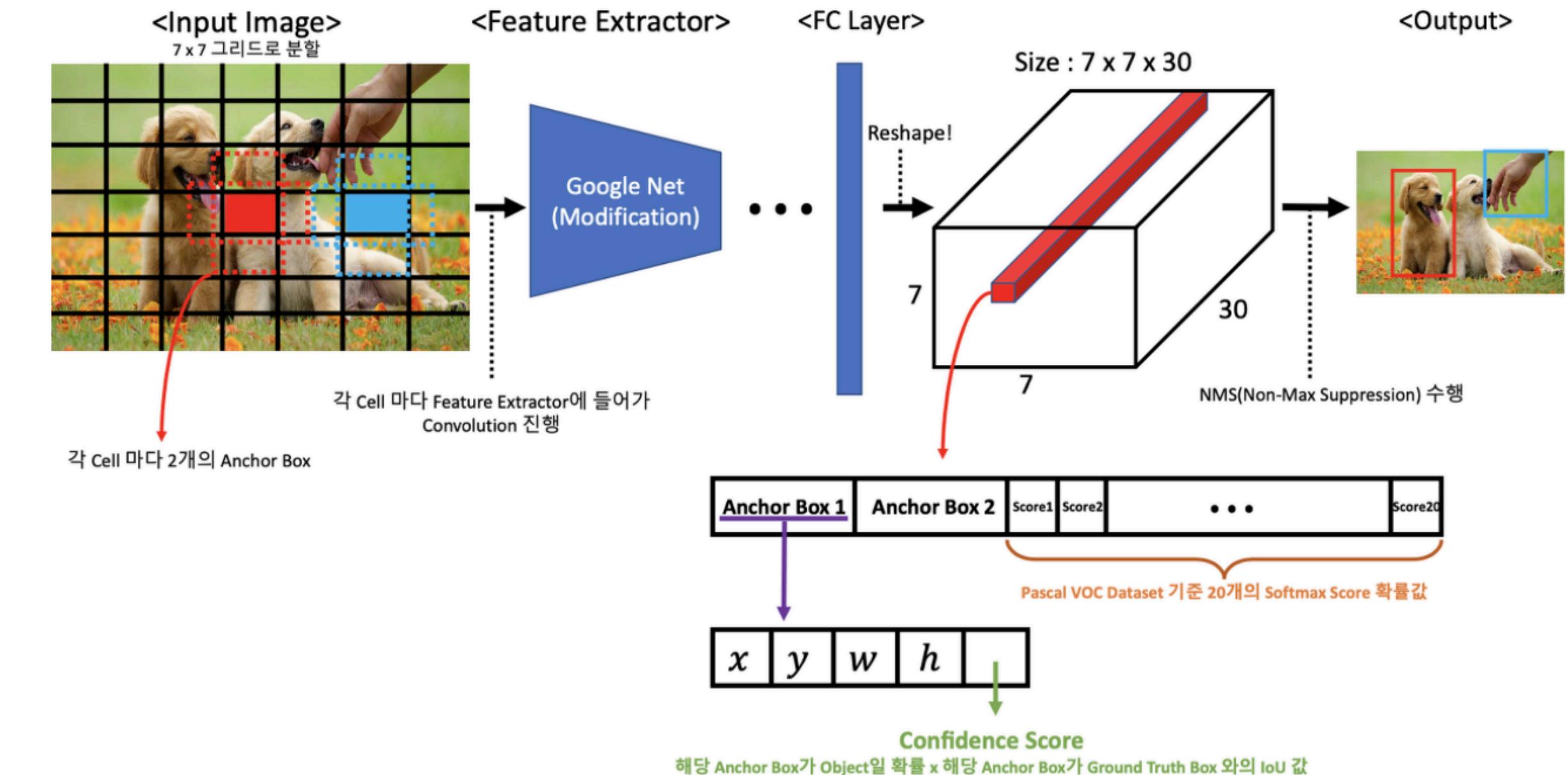
- $\text{Pr}(\text{Object})$: Object가 bbox내에 있으면 1, 아니면 0

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.

2. Unified Detection

YOLO

2. Unified Detection



- Resized Image -> output vector
 - Resize된 이미지를 7×7 로 나눔 -> Grid cell마다 bbox 2 개씩 예측 -> output vector 생성
- 하나의 셀의 Object Detection 수행 -> 셀의 개수만큼 Output vector가 생김
 - 크기: $S \times S \times (B * 5 + C)$
 - $\rightarrow 7 \times 7 (2 * 5 + 20) = 30$

- output vector - bBox 마다 5개의 값 매핑
 - (x, y) : 바운딩 박스의 중심점 (셀 기준으로 normalize)
 - (w, h) : 바운딩 박스의 width와 height (이미지 기준으로 normalize)
 - confidence: (= confidence score)
 - we define confidence as $\text{Pr}(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$.
- output vector - 확률값 추가
 - 클래스 개수 만큼 추가
 - Conditional Class Probability: C_i 의 score
 - Object가 bBox 내에 있을 때, Grid Cell에 있는 Object가 i 번째 class에 속할 확률

Each grid cell also predicts C conditional class probabilities, $\text{Pr}(\text{Class}_i | \text{Object})$. These probabilities are condi-

YOLO

2.1. Network Design

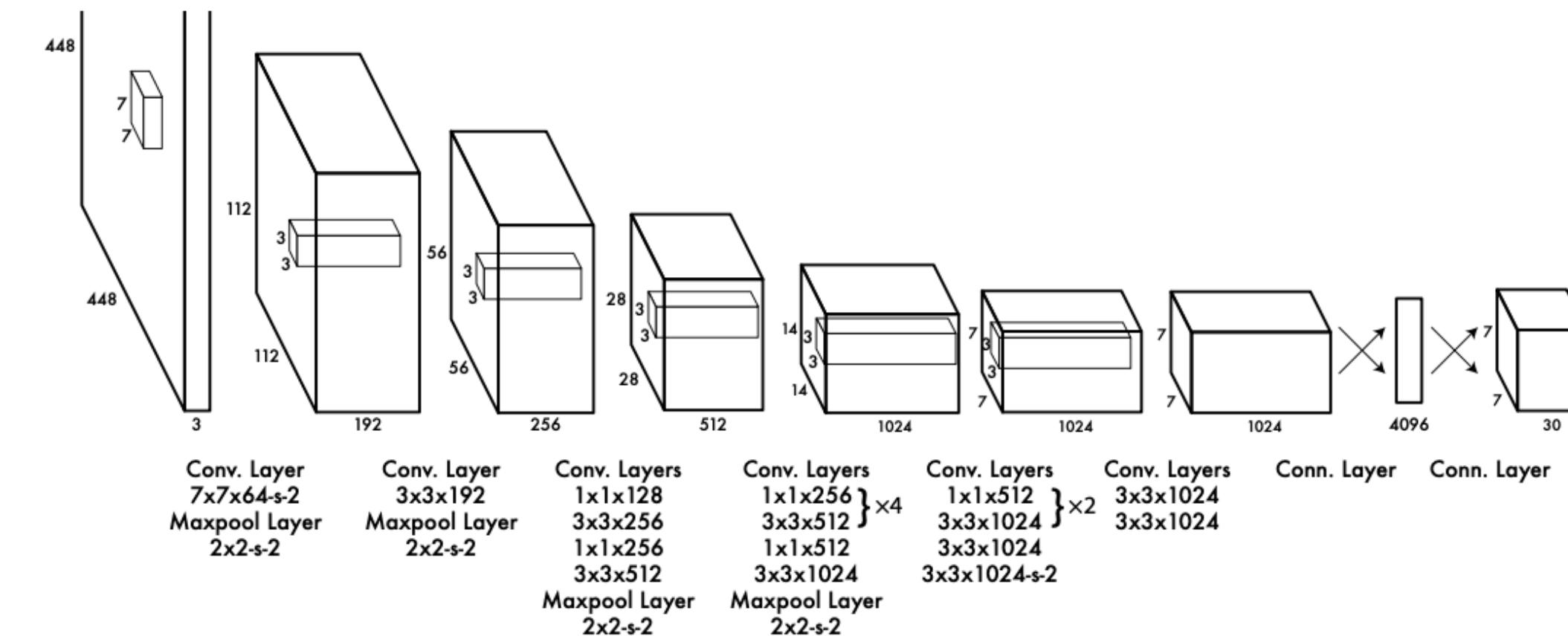
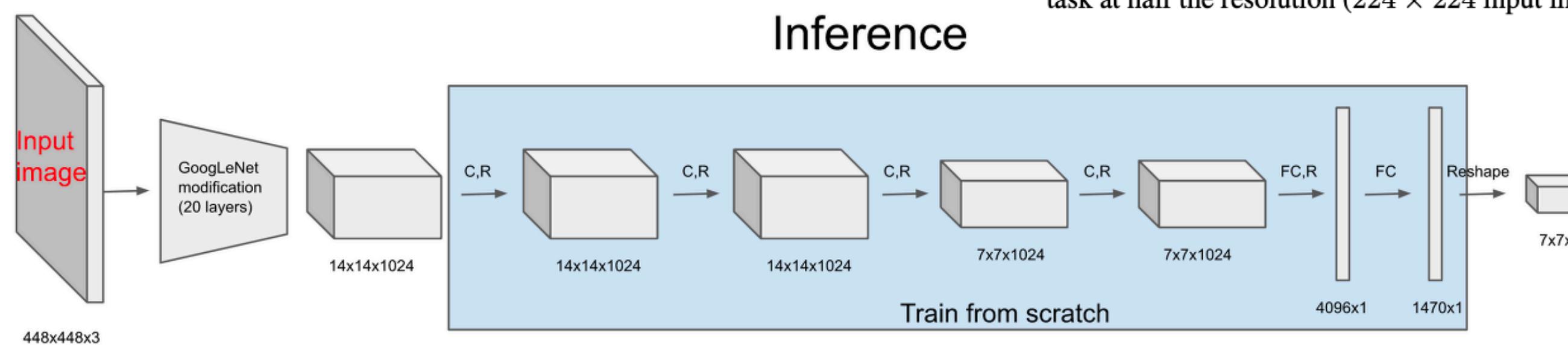


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

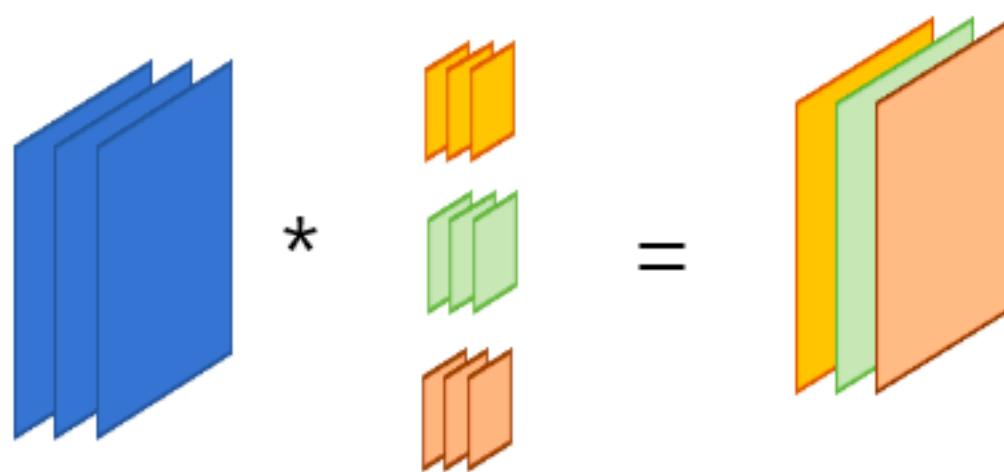
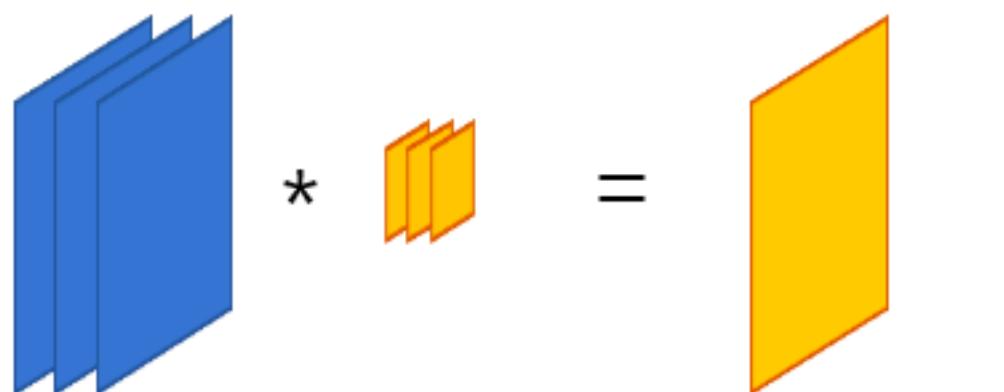


- **GoogLeNet model for image classification** 영감을 얻음
- **24개의 Conv layer와 2개의 Fully connected layer로 네트워크를 구성**
(Fast YOLO: 9개의 Conv layer)
- **1 x 1 reduction layer & 3 x 3 Conv layer**
- **GoogleNet's inception module 대신 1 x 1 을 사용하여 연산량 줄임**
- **input image: 448x448 이미지**
- **학습**
 - 24개의 Conv Layer 중 앞의 20개의 Conv layer는 고정
 - 뒤 4개의 Conv layer와 2개의 Fully Connected Layer만 object detection task에 맞게 학습
- **prediction tensor: 7 x 7 x 30 텐서**

1 x 1 Conv layer

1 x 1 conv -> 3 x 3 conv의 의미

$$\begin{array}{|c|c|c|c|} \hline 1 & 2 & 1 & 1 \\ \hline 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 2 & 0 \\ \hline 4 & 1 & 2 & 0 \\ \hline \end{array} * \boxed{2} = \begin{array}{|c|c|c|c|} \hline 2 & 4 & 2 & 2 \\ \hline 2 & 0 & 0 & 2 \\ \hline 0 & 0 & 4 & 0 \\ \hline 8 & 2 & 4 & 0 \\ \hline \end{array}$$



- output channel의 수 = Filter 수
 - Filter 수를 조절함으로써 파라미터의 수를 조절
 - 주로 연산량을 줄이기 위해 사용하기 때문에, filter 수를 줄임
 - 차원 축소
 - GoogleNet's inception module도 같은 역할을 함

- Bottleneck 구조 (>< 같은 모양)
 - 1 x 1 convolution & 3 x 3 convolution
 - 1 x 1 convolution으로 좁아졌다가, 3x3 convolution 수행(원래 하고자 했던 연산)
 - 비용이 많이 드는 3x3 convolution을 하기 전에, 1x1 convolution을 통해 차원 축소를 함.