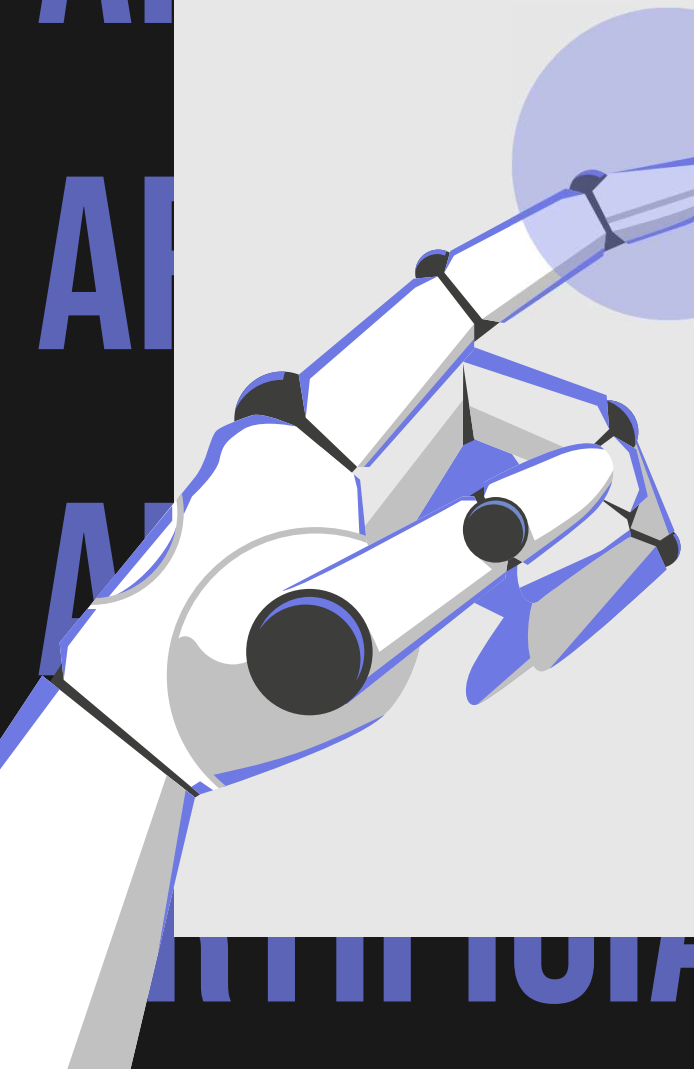


YOLO V1

학부연구생 2단계 과정 세미나

18011816 김동영

TABLE OF CONTENTS

ARTIFICIAL INTELLIGENCE (AI)

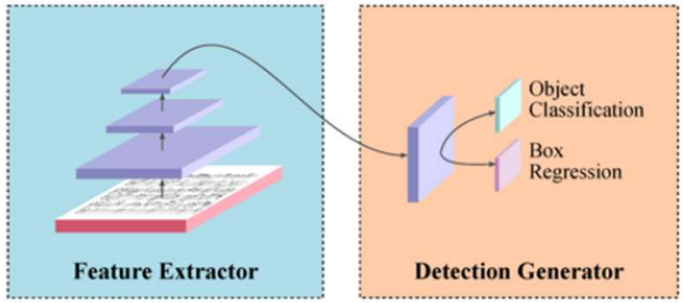
01.

INTRODUCTION

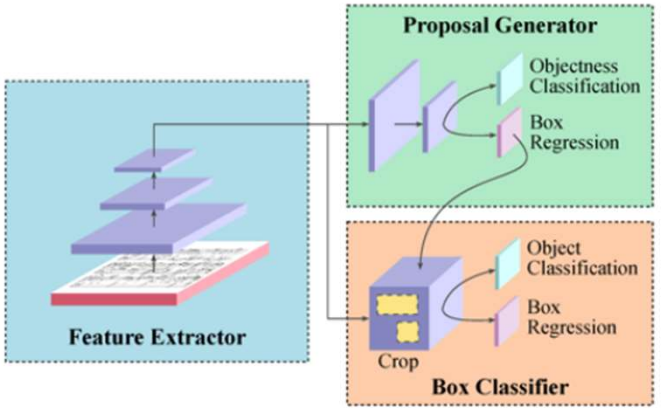
Let's find out What YOLO V1 is

INTRODUCTION

- One-Stage Detectors
- Two-Stage Detectors



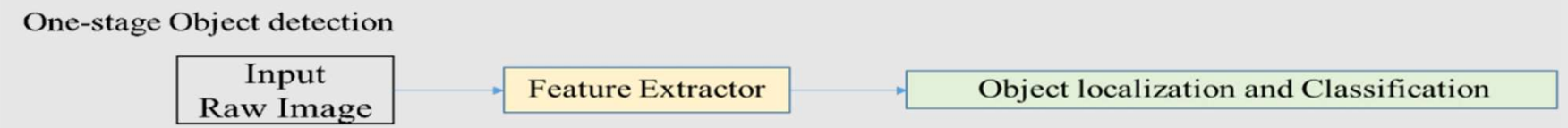
(a) Basic architecture of a one-stage detector.



(b) Basic architecture of a two-stage detector.

RCNN, SPP, Fast RCNN, Faster RCNN 모델은 Two-Stage 모델입니다. 여기서 Two란, 위의 오른쪽 그림처럼 Region Proposal과 Object Detection 단계를 분리해 전개되는 모델들을 말합니다. 반면에 YOLO와 같은 One-Stage 종류의 모델들은 Region Proposal 과 Object Detection을 따로 분리하지 않고 한 번에 수행하는 모델입니다.

INTRODUCTION

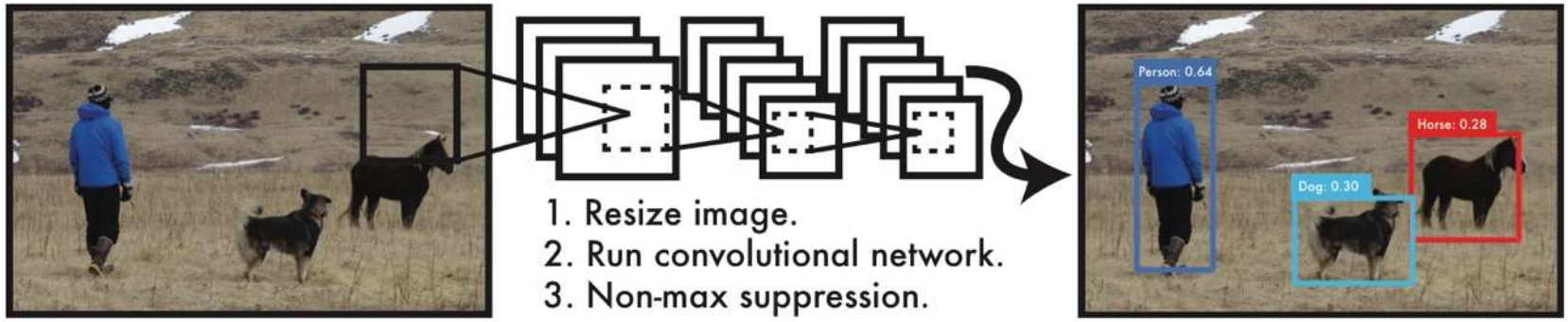


Two-Stage 모델들의 학습 속도가 느리기 때문에 One-Stage 모델들이 등장하게 되었습니다. Object Detection 분야에서 객체를 얼마나 탐지를 잘 하느냐 도 중요하지만 탐지를 얼마나 '빨리' 하느냐 도 마찬가지로 중요하기 때문입니다.

논문에서 나온 것처럼, 컴퓨터가 자동차를 특정한 센서 없이 운전할 수 있으려면, 실시간으로 들어오는 정보를 빠르고 정확하게 분석 할 수 있어야 가능하기 때문에 Object-Detection에서는 검출 속도 또한 정확성과 함께 가져가야 합니다.

그런데 Two-Stage 모델들은 탐지 성능은 뛰어나지만 상대적으로 탐지 속도는 매우 느린 편입니다. 따라서 연구자들은 이러한 느린 탐지 속도문제를 해결하고자 One-Stage 모델인 YOLO(You Only Look Once)라는 Real-Time Object Detection을 개발했습니다.

INTRODUCTION



YOLO의 동작원리를 간단하게 본다면, 위 그림에 보이듯이 한 개의 Convolutional Network가 동시에 수 많은 Bounding Box들을 예측하고 각각 Class들에 대한 확률 값들을 계산합니다.

또한, YOLO는 전체 이미지를 직접적으로 학습할 수 있기 때문에 과거에 사용되었던 방법들과 비교해 여러 장점들을 가지고 있습니다.

/(AI)

02.

PROS AND CONS

What is the
Advantage & Disadvantage

PROS

YOLO의 장점에 대해 먼저 알아보겠습니다.

1. YOLO의 속도는 엄청나게 빠릅니다.
   Batch Processing 없이 초당 45프레임을 처리할 수 있고, Fast YOLO의 경우 초당 150프레임을 처리할 수 있습니다. 이러한 처리 속도 때문에 25 ms 미만의 지연속도로 real-time detection이 가능하며, 기존의 다른 real-time detector들 보다 2배 이상의 mAP(Mean Average Precision)를 보여줍니다.

2. 예측할 때 이미지를 전역적으로 파악합니다.
   기존의 sliding window와 region proposal과 다르게 이미지의 전체를 학습할 때와 예측할 때 봅니다. 따라서 해당 클래스의 맥락적 정보 뿐만 아니라 모습을 encode해서 fast R-CNN보다 background error를 절반 미만으로 줄일 수 있습니다.

3. 물체의 일반화 가능한 표현을 학습합니다.
   YOLO는 일반화하여 학습하므로 새로운 환경 또는 예상치 못한 Input들에 대해서 잘 무너지지 않아, 자연 이미지(natural image)로 학습한 후 그림(artwork)에서 test를 진행해도 다른 모델들 보다 좋은 성능을 보입니다.

CONS

다음으로는 YOLO의 한계점에 대해 먼저 알아보겠습니다.

1. YOLO는 각각의 grid cell에서 **오직 2개의 bounding boxes만을 예측할** 수 있고, **하나의 class**만 갖을 수 있으므로 근처에 있는 여러 물체를 검출하는데 한계점이 있습니다. 따라서 그룹에서 나타나는 근접한 작은 물체인 양떼나 조류에 대해 잘 감지 못 합니다.

2. YOLO는 Train dataset에 있는 bounding box를 통해 학습하기 때문에, 검출하려는 물체가 Train dataset에 **존재하는 bounding box의 형태와 다를 경우** 일반화하는데 어려움이 있습니다.
또한 입력 이미지를 **여러 번 down-sampling해 coarse features(거친 특징)를** 사용하여 bounding box를 예측해야 하는 단점이 있습니다.

3. Bounding box가 큰 경우에서 작은 error보다 bounding box가 작은 경우에서 작은 error를 더 크게 영향을 받아 localization의 부정확한 경우가 있습니다.

03.

HOW IT WORKS

Let’s find out How the network works

HOW IT WORKS

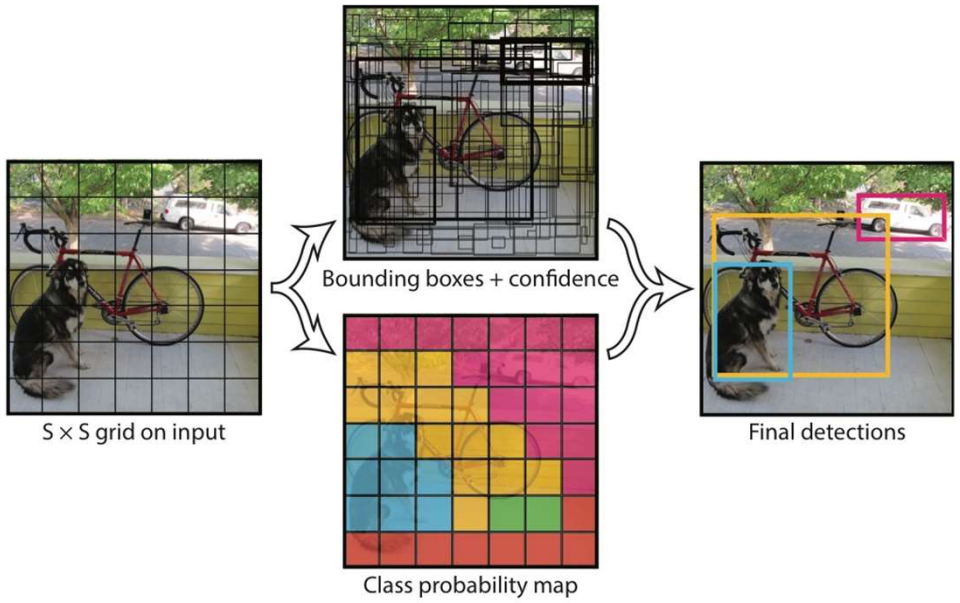


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

(1) 입력 이미지를 S x S grid로 분할합니다.

(2) 객체의 중심이 grid cell에 맞아 떨어지면 그 grid cell은 객체를 탐지했다고 표기합니다.

(3) 각 grid cell은 B개의 bounding box와 각 bounding box에 대한 confidence score를 예측합니다.

(4) 각 bounding box 박스는 5개의 정보를 갖고 있습니다. x, y, w, h와 confidence 입니다. x, y는 bounding box 중심을 나타내고, w, h는 width와 height 값입니다. 마지막으로 confidence는 predicted box와 ground truth 사이의 IOU(intersection over union) 입니다.

HOW IT WORKS

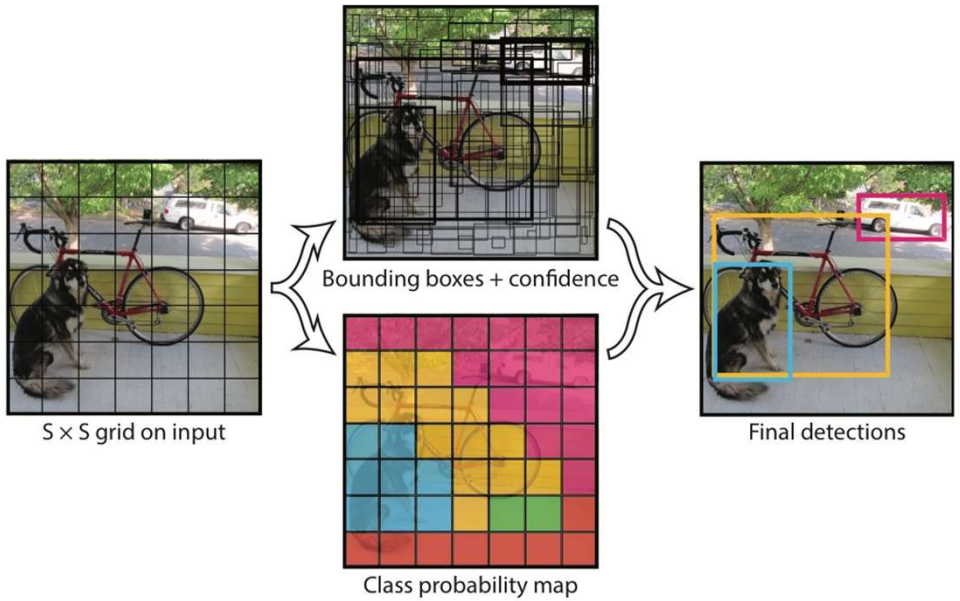
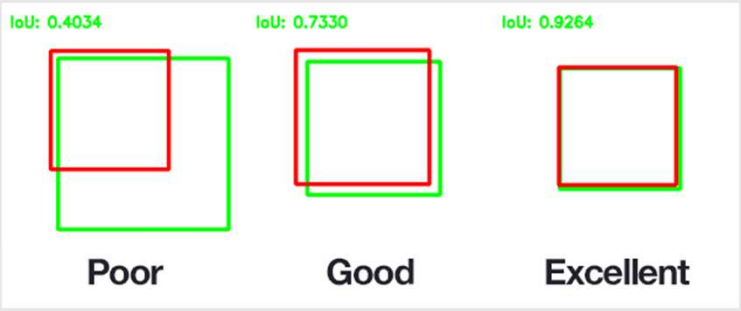


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

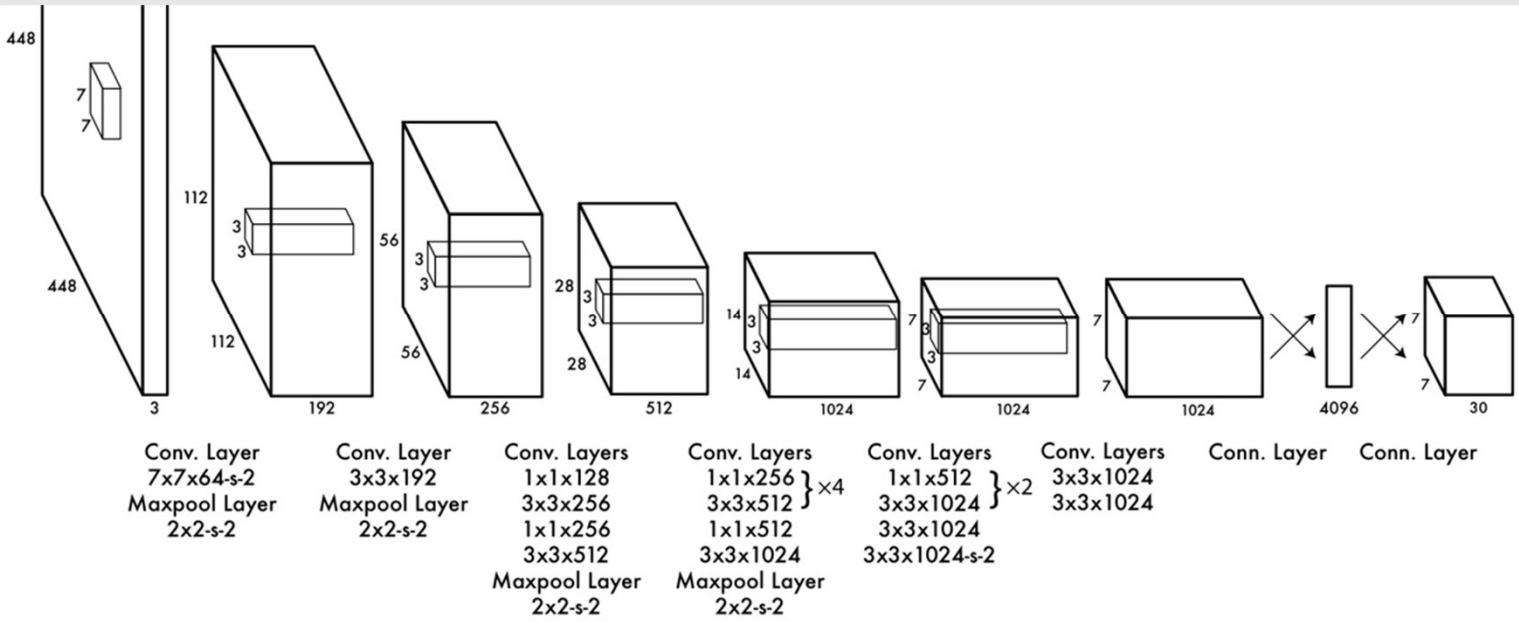
(5) 각 grid cell은 bounding box 이외에도 class 확률을 예측합니다. 최종적으로 예측 값은 (S x S x (B * 5 + C)) 크기의 tensor를 갖습니다.

- S = feature size
- 5 = (x, y, w, h, confidence)
- B = # of boxes
- C = # of classes



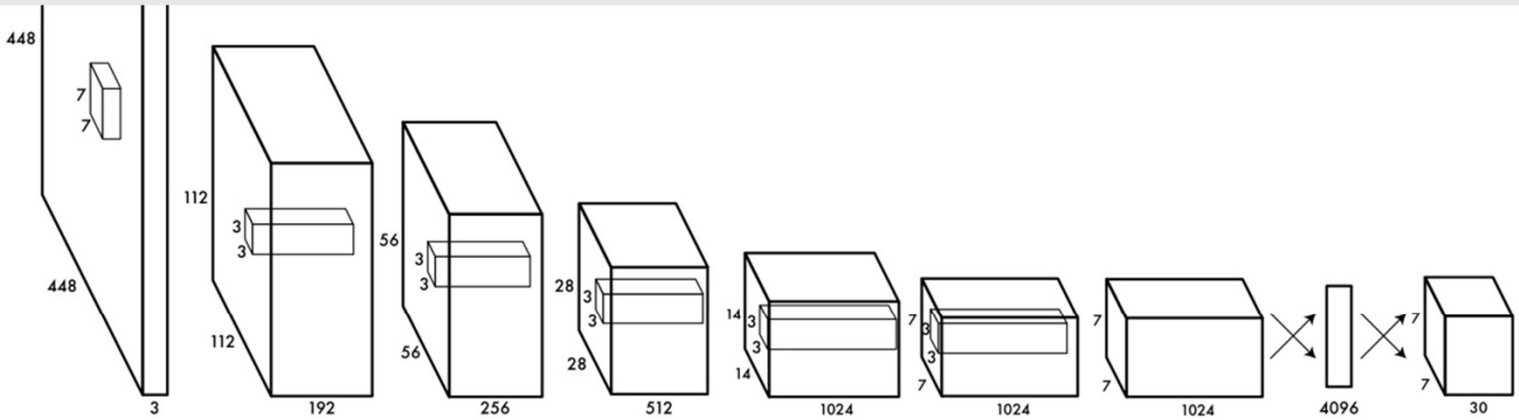
IOU between the predicted box and the ground truth.

NETWORK LAYERS



YOLO의 네트워크 구조: Image classification은 GoogLeNet 구조에 영감을 받아 설계됐습니다. 기존의 GoogLeNet은 Inception module(병렬 구조)을 사용한 반면, YOLO는 Inception module을 일자로 나열한 모델을 사용했습니다. Fast version YOLO의 경우 convolutional layer의 수를 24개에서 9개로 줄이고, filter 또한 적게 사용했습니다. 네트워크 구조 이외에 모든 parameter는 기존 YOLO와 동일합니다.

NETWORK LAYERS



Training을 하기 전 앞의 20개의 convolutional layers를 ImageNet dataset으로 pre-train 합니다. Object detection을 수행하기 4개의 convolutional layers와 2개의 fully-connected layers를 추가했습니다. Pre-train시 224x224 image를 input으로 사용했으나, object detection을 더 효과적으로 수행하기 위해 448x448 image의 input을 사용합니다. 마지막 layer는 class probabilities 와 bounding box 좌표를 예측하는데, 이 값들을 정규화 하여 사용하게 됩니다.

LOSS FUNCTION

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

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

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

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

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

최적화를 쉽게 하기 위해 sum-squared error를 사용하지만, localization error와 classification error를 동등하게 가중치를 주는 것이 이상적이지 않고, 객체가 존재하지 않는 grid cell이 많기 때문에 객체가 존재하는 cell의 confidence가 0을 향하게 하여 조기 탈선을 야기 시킬 수 있습니다.
따라서 bounding box coordinate loss를 증가시키고, 객체가 존재하지 않는 box의 경우 confidence loss를 감소시켰습니다. 이를 위해 $\lambda coord$=5, $\lambda noobj$=0.5를 사용했습니다.

또한 sum-squared error는 box 크기에 상관없이 같은 error를 사용합니다. 같은 error여도 큰 박스에 존재하는 것이 작은 박스에 존재하는 것보다 덜 중요하기 때문에 이를 구분하기 위해 bounding box의 width와 height을 제곱근을 이용해 계산하였습니다.

ARTIFICIAL INTEL (AI)

THANKS A LOT FOR LISTENING

(AI)