2020.11.02

YOLO

2020 fall ESC

산업공학과 박상재

Introduction

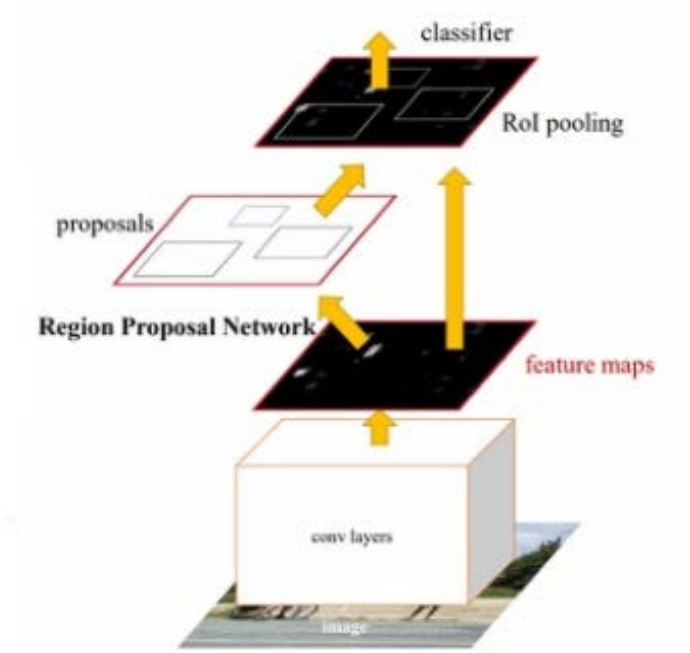


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Table 5: **Timing** (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. "Region-wise" includes NMS, pooling, fully-connected, and softmax layers. See our released code for the profiling of running time.

model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	**10**	47	**198**	**5 fps**

기존의 object detection 모델인 Faster R-CNN 모델은 속도가 느리다는 단점

Faster R-CNN의 fps 성능은 5fps로 real-time (30fps)에 한참 못 미치는 수준

Detection 성능은 약간은 떨어지더라도 높은 fps 성능을 갖는 object detection 모델을 구현해볼 수 없을까 ? -> YOLO

Introduction

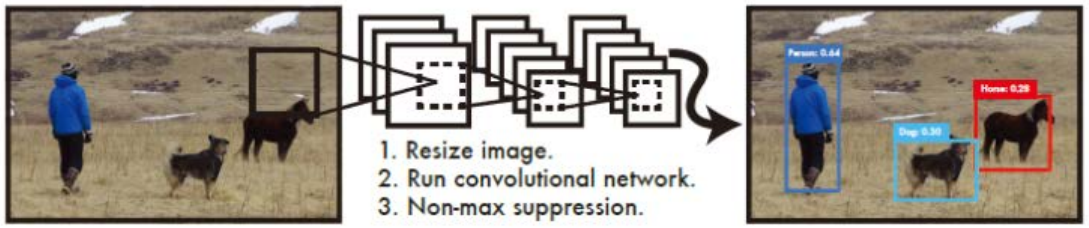


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

R-CNN과 달리 YOLO 에서는 이미지 내의 bounding box 와 class probability를 single regression problem 으로 간주

즉, 이미지를 한 번 보는 것으로 object의 위치와 종류를 동시에 추측(You Only Look Once)

Introduction

장점

간단한 처리과정으로 빠른 속도. + 기존의 다른 real-time detection system과 비교했을 때, 2배정도 높은 mAP

Image 전체를 보고 추측하기 때문에 낮은 background error(False-positive)

Object의 일반적인 특징을 학습하기 때문에 보편적으로 활용 가능

단점

최신 모델들에 비해 정확도는 조금 떨어짐

물체가 작을수록 위치를 파악하는데 어려움

Unified Detection

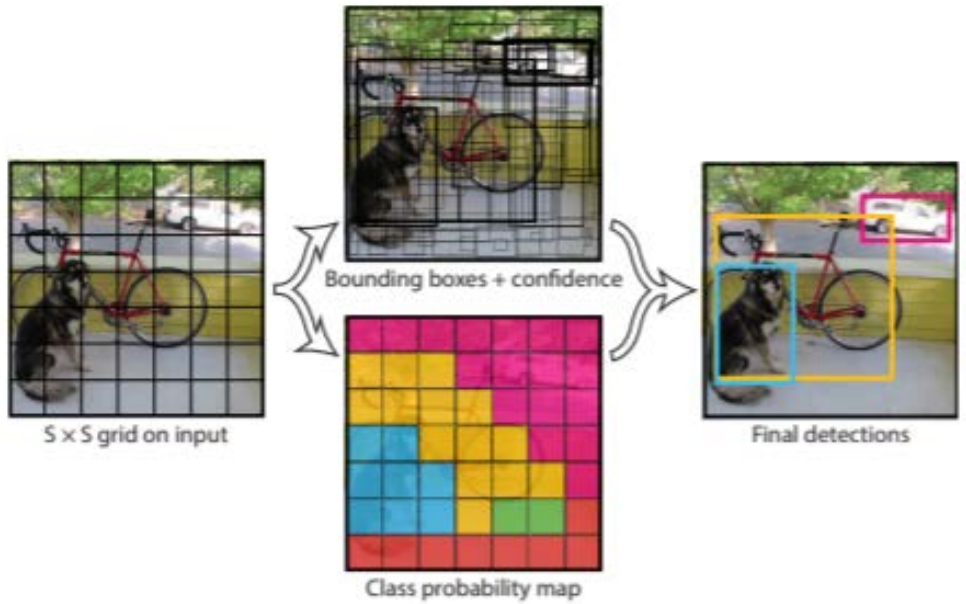


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

1. Input image를 S x S grid로 나눈다

2. 각각의 grid cell은 B개의 Bounding box와 각 box는 confidence score를 가진다

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

3. 각각의 grid cell은 C개의 conditional class probability를 가진다.

$$\Pr(\text{Class}_i | \text{Object}).$$

Unified Detection

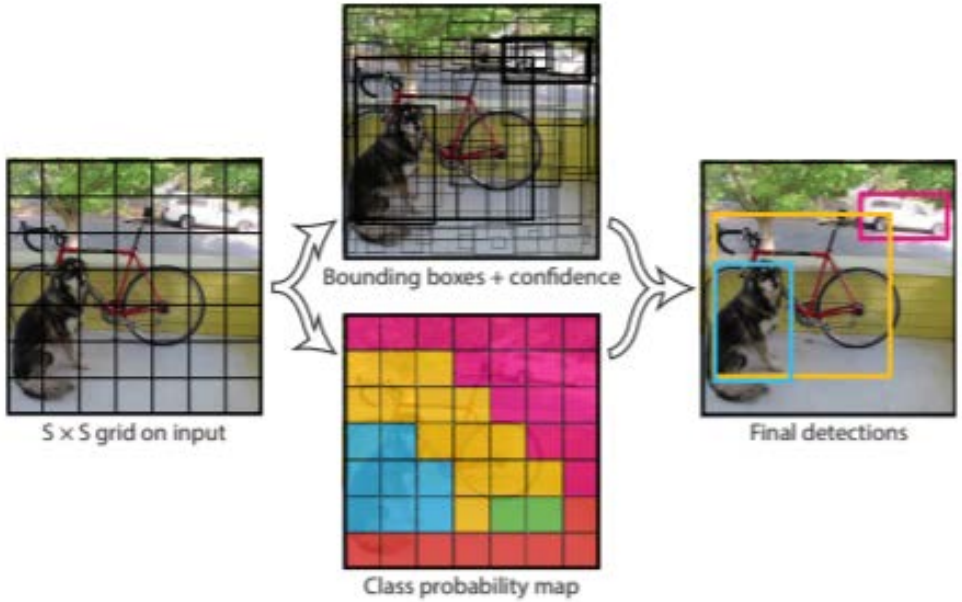


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

4. 각각의 Bounding box는 x, y, w, h confidence 5개의 값을 가짐

x, y – box의 중심점, grid cell에 대한 상대값 (0~1)
w, h – input image에 대한 상대값 (0~1)

Test 에는 conditional class probability 와 Bounding box의 confidence score를 곱해 class specific confidence score를 얻음

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}} = \Pr(\text{Class}_i) * \text{IOU}^{\text{truth}}_{\text{pred}}$$

Network Design

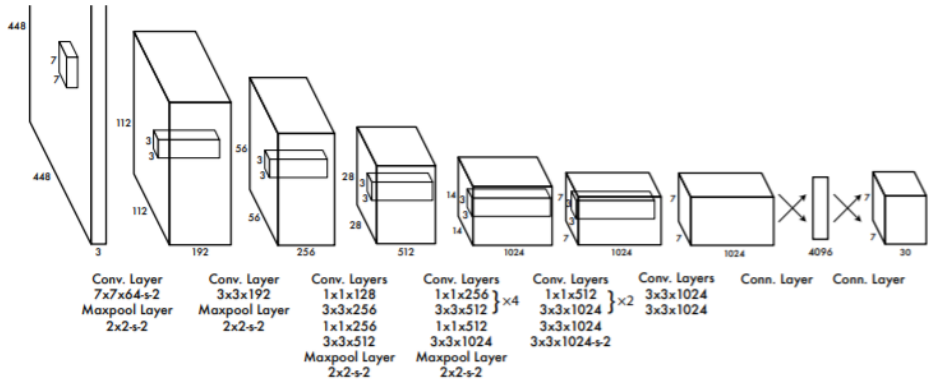


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

GoogLeNet의 모델로 부터 영감을 받아 24개의 Convolutional layers와 2개의 Fully connected layers로 구성

Convolutional layers에서 feature 추출 (pretrained network)
Fully connected layers에서 확률과 좌표 예측 (training network)

Network Design

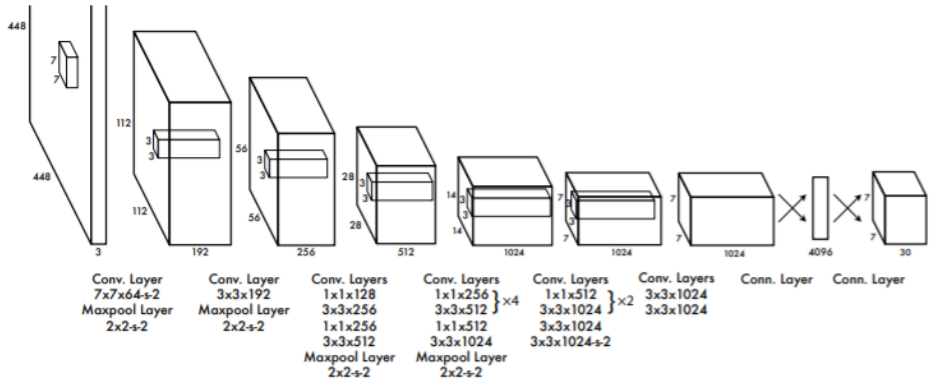


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

Pre-trained network

GoogLeNet을 이용하여 ImageNet 1000-class dataset을 사전에 학습하여 Fine Tuning (20개의 convolutional layers)

4개의 convolutional layers와 2개의 fully connected layers 를 추가

224x224 크기의 이미지 데이터를 448 x 448 크기의 이미지 데이터로 바꿔 줌

Network Design

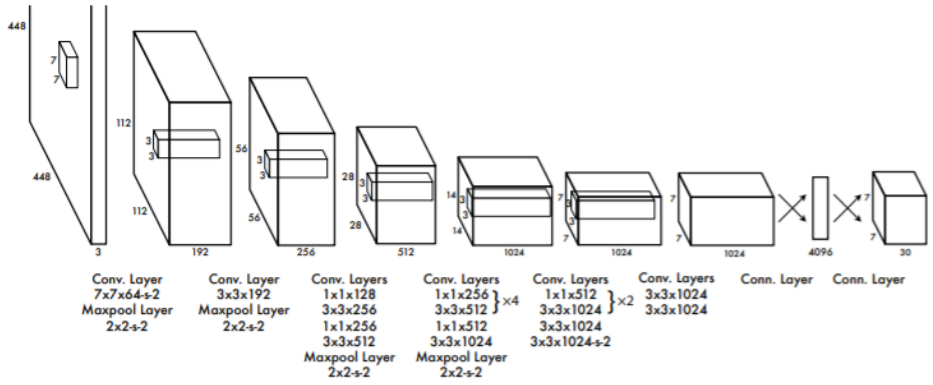


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

Training Network

앞의 Pre-trained Network에서 학습한 feature를 이용하여 Class probability와 Bounding box를 학습하고 예측

YOLO 예측 모델은 S*S*(B*5 +C)개의 값을 결과로 출력

Training

loss function:

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

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

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

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

$$+ \sum_{i=0}^{S^2} 1_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (3)$$

좋은 loss functio을 만들기 위해 ..

두가지 loss의 비중을 다르게 함

-> set $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{noobj}} = .5$.

큰 box의 변화가 작은 box의 변화보다 덜 중요함

-> w, h의 제곱근 값을 사용함으로써 부분적으로 반영

1개의 bounding box가 1개의 object를 예측하도록

->가장 큰 IOU값을 갖는 class 예측

Training

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

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

Object가 존재하는 grid cell i의 predictor bounding box j에 대해, x와 y의 loss를 계산

Object가 존재하는 grid cell i의 predictor bounding box j에 대해, w와 h의 loss를 계산.

Training

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

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

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

Object가 존재하는 grid cell i의 predictor bounding box j에 대해, confidence score의 loss를 계산 C_i = 1

Object가 존재하지 않는 grid cell i의 bounding box j에 대해, confidence score의 loss를 계산 C_i = 0

Object가 존재하는 grid cell i에 대해, conditional class probability의 loss 계산

Correct class – p_i(c) =1

Other class – p_i(c) =0

Training

loss function:

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

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

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

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

$$+ \sum_{i=0}^{S^2} 1_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

Batch size: 64

Momentum: 0.9 and a decay of 0.0005

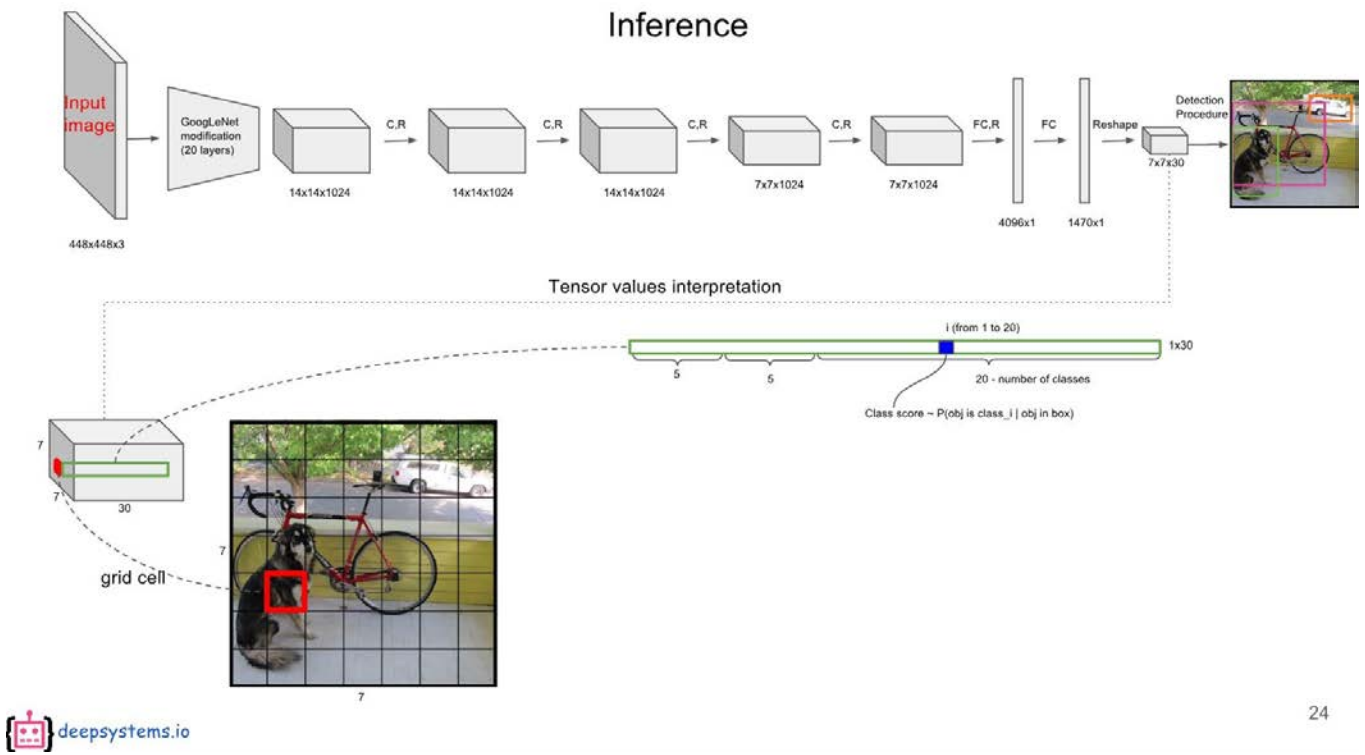
Learning Rate: 0.001에서 0.01로 epoch마다 천천히 상승시킴. 이후 75 epoch동안 0.01, 30 epoch동안 0.001, 마지막 30 epoch동안 0.0001

Dropout Rate: 0.5

Data augmentation: random scailing up to 20 %

Activation function: leaky rectified linear activation

Inference

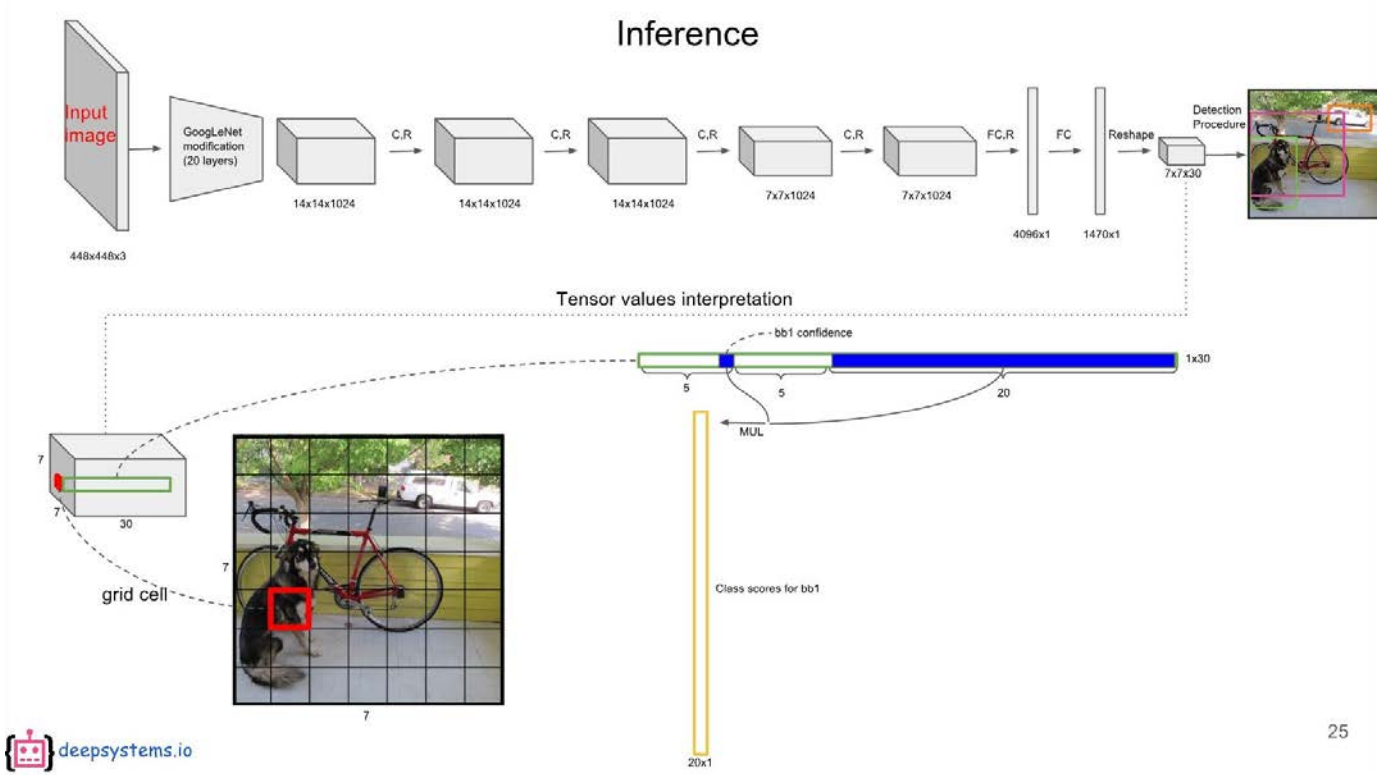


앞 5개의 값은 해당 Grid cell의 첫 번째 bounding box에 대한 값

6~10번째 값은 두번째 box에 대한 값

나머지 20개의 값은 각 class에 대한 conditional class probability

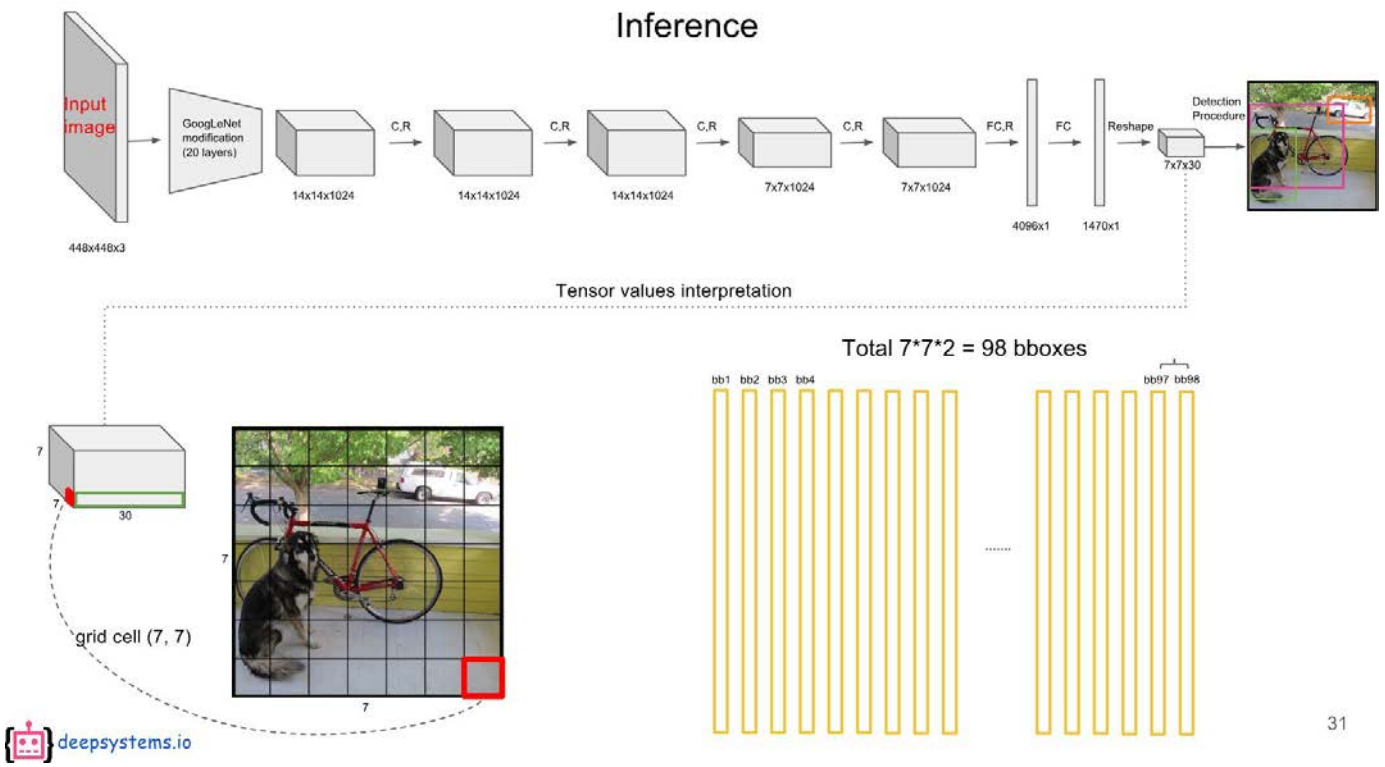
Inference



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

두번째 box도 똑같이 계산하면 class specific confidence score가 나온다.

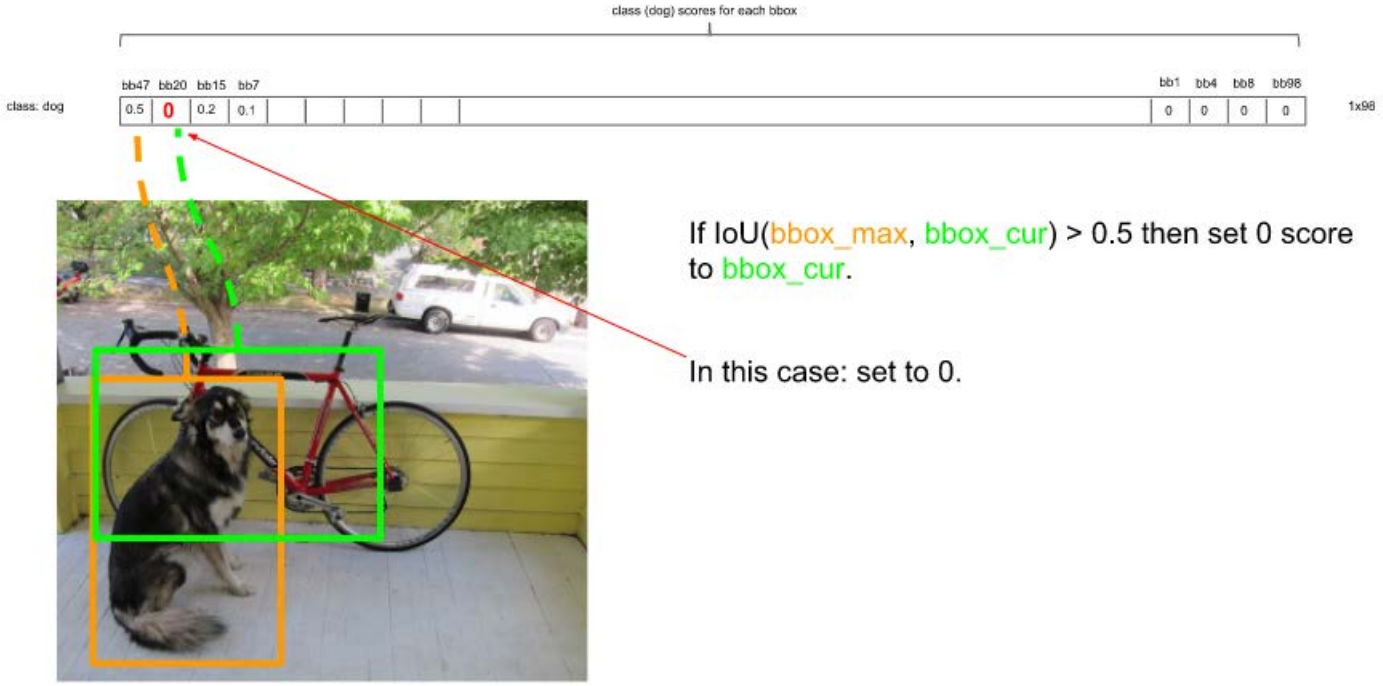
Inference



bounding box에 대해 계산하게되면 총 98개의 class specific confidence score를 얻을 수 있다.

이 98개의 class specific confidence score에 대해 non-maximum suppression을 통해 Object에 대한 Class 및 bounding box Location를 결정한다.

Inference

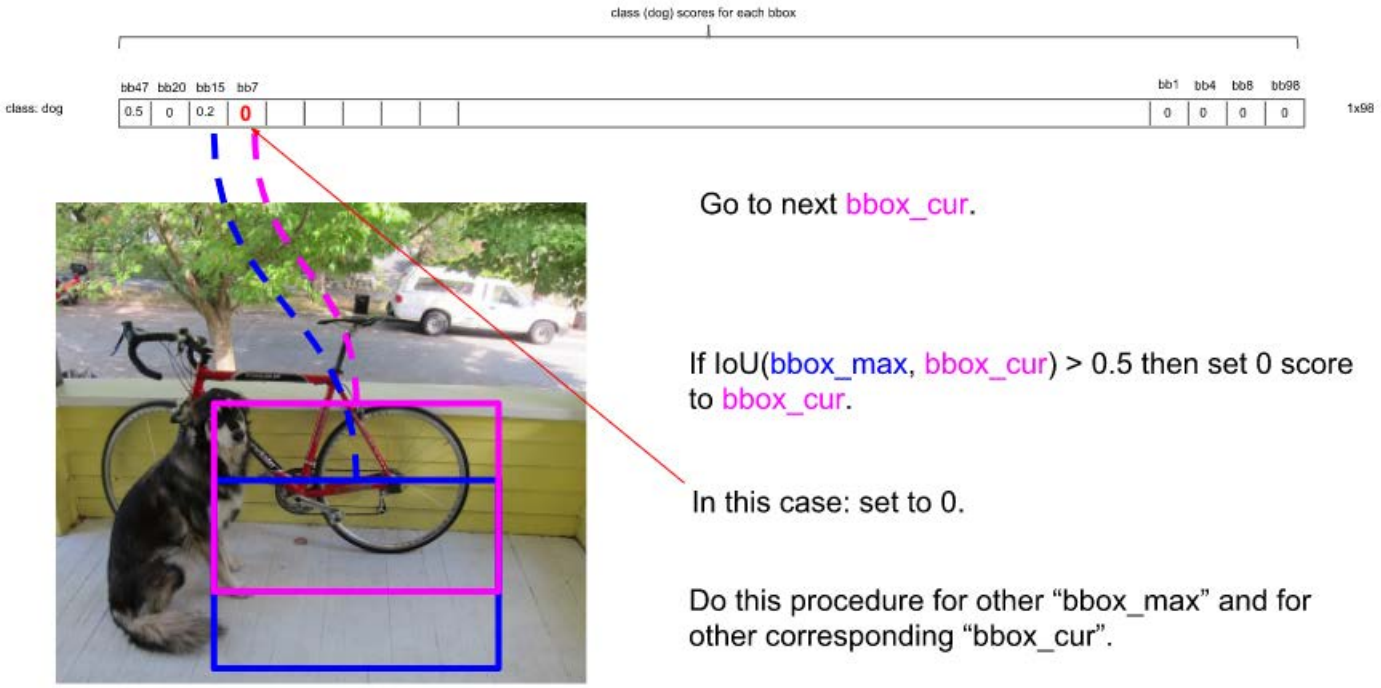


Dog 라는 class에 대해 각 box의 score 를 비교해서 제일 값이 큰 것을 기준으로 잡는다.

최대값을 기준으로 자기보다 score가 작은 box의 IOU가 0.5 이상인 score들을 다 0 으로 바꾼다.

이전의 최대값을 제외하고 계속 반복한다.

Inference



Dog 라는 class에 대해 각 box의 score 를 비교해서 제일 값이 큰 것을 기준으로 잡는다.

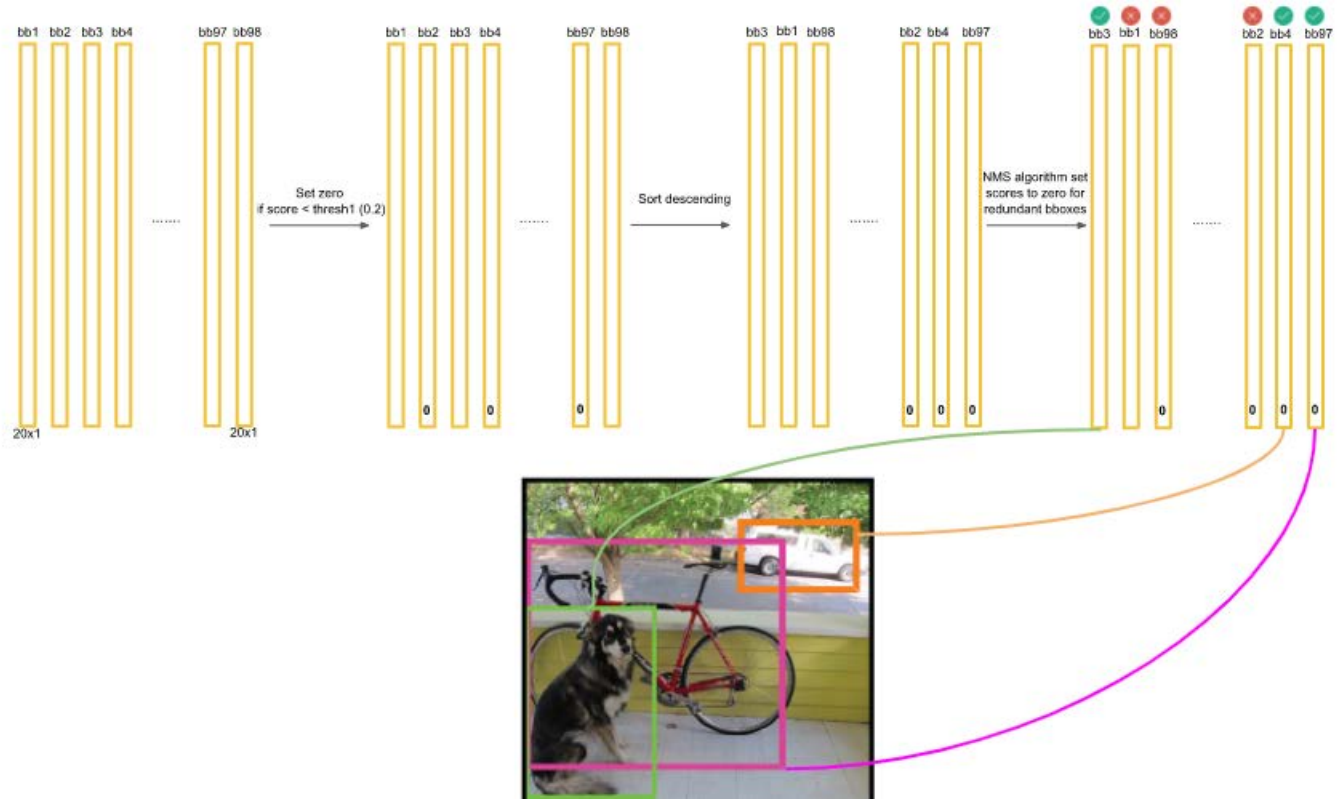
최대값을 기준으로 자기보다 score가 작은 box의 IOU가 0.5 이상인 score들을 다 0 으로 바꾼다.

이전의 최대값을 제외하고 계속 반복한다.

Inference

각 box의 class specific confidence score의 max값이 0이 아니면 예측을 한다.

Inference



각 box의 class specific confidence score의 max값이 001 아니면 예측을 한다.

Experiments

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

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

타 모델에 비해 훨씬 뛰어난 초당 프레임 수(FPS)성능

Fast YOLO의 경우에는 155FPS

약간 부족한 정확도(mAP) 성능

Experiments

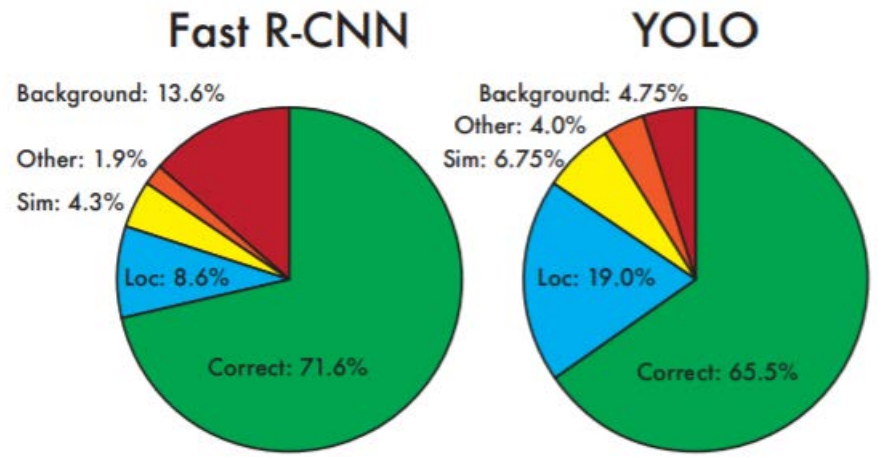


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

Fast R-CNN 의 background error 13.6%
YOLO의 background error 4.75%

훨씬 적은 False-positive error
(low background error)

Limitations, Conclusions

작고 많은 물체(EX: 새 떼, 개미 떼 …)는 인식하기 어려움

YOLO가 데이터를 제공받고 Bounding box를 학습하기
때문에 극단적인 비율을 가지는 object는 인식하기 어려움.

But, 간단하고 빠르다!

Thank you