YOLO 논문요약

Written by 박 철(e2g1234@naver.com)

YOLO(You Only Look Once)

.02640v5 [cs.CV] 9 May 2016

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/

Abstract

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 predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

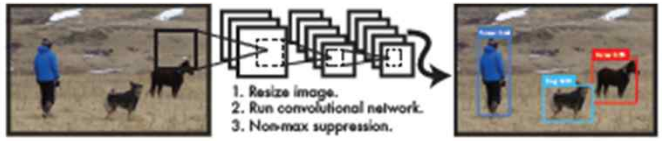


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

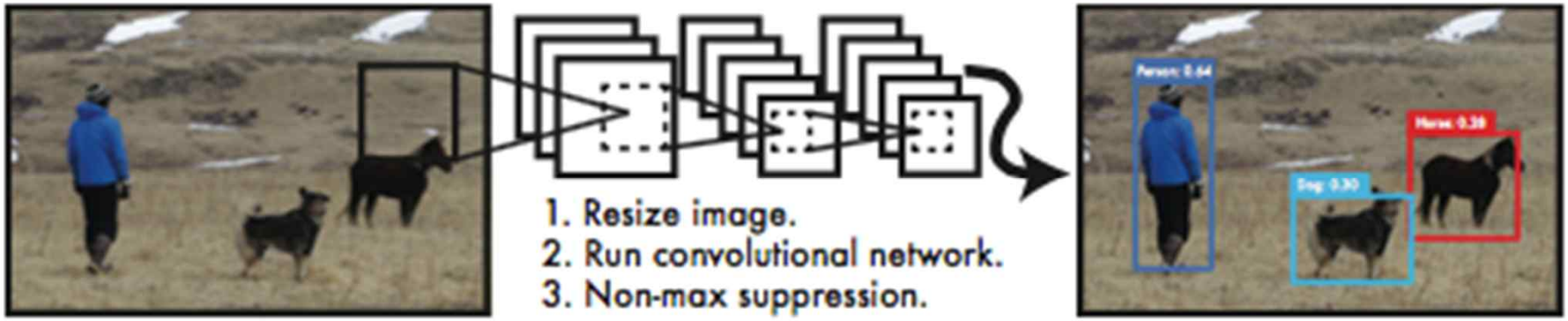
methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

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

요약

- ✓ object detection에 있어서 새로운 접근법인 YOLO를 소개합니다.
- ✓ object detection은 사전 작업으로 detection을 먼저 수행합니다.
- ✓ object detection을 공간적으로 분리된 bounding box들과 class(분류할 object 종류들)에 대한 확률과 연관된 regression 문제로 재정의
- ✓ Single neural network는 전체 이미지에서 한번의 평가를 통해서 직접적으로 bounding box들을 예측하고 class의 확률을 계산
- ✓ 전체 detection pipeline이 Single network이기 때문에, 탐지 성능이 최적화 됨
  - 통합된 아키텍쳐를 사용하여 극단적으로 빠릅니다.
  - YOLO model은 45 FPS로 실시간 이미지 연산이 가능
  - 더 작은 버젼의 네트워크인 Fast YOLO는 매우 놀랍게도 다른 실시간 탐지모델보다 2배의 mAP(Mean Accuracy Precision:: 평균 정확도)를 가지면서 155 FPS의 성능을 보여줌
  - 최신의 탐지 시스템들과 비교했을 때, YOLO는 localization error가 조금 더 높지만, 배경으로 인한 예측 실패율은 더 낮음
  - 마지막으로 YOLO는 object의 아주 일반적인 표현들을 학습합니다. 이 말은 DPM이나 R-CNN같은 다른 탐지 대비 artwork같은 다른 영역에서의 자연스러운 이미지로부터 일반적인 표현을 학습하는데 있어서 결과가 더 좋다는 것을 의미함

The YOLO Detection System



YOLO를 이용하여 이미지를 처리하는 과정은 간단하고 직관적임.

(1) input image를 448 x 448의 이미지로 resize

(2) image에서 작동하는 single convolution network를 실행

(3) 해당 모델을 통해서 나온 확률 값을 threshold로 잘라서 결과값을 보여줌

Detection 방식의 비교

- ✓ R-CNN
  - 더 최신의 detection 시스템
  - region proposal 방법 사용함
  - region proposal은 image안에서 가능성이 높은 bounding box를 생성
  - 그 box속의 이미지를 classifier에 넣어 실행
  - 분류 후에는 이전에 진행 했던 과정들을 중복된 detection 제거 하거나
  - 다른 장면에서의 object를 기반으로 re-score(재평가)를 하는 것 등을 위해서 성능 향상을 위해 재사용
  - 이러한 복잡한 pipeline은 각각의 component들을 따로 학습시켜야하기 때문에 느리고 최적화하기가 힘듬
- ✓ YOLO
  - object detection을 이미지 pixel로부터 bounding box좌표와 분류 과정을 single regression problem으로 재정의
  - 이미지를 한번만 보면 현재 image에 있는 object가 무엇이고 어디 있는지 알 수 있음.(쉽다)
  - 위 그림의 Single Convolution network는 물체의 bounding box와 bounding box안의 object가 무엇인지 동시에 예측
  - YOLO는 전체 image에서 학습하며 직접적으로 detection performence를 최적화함
  - 이런 통합적인 모델은 전통적인 object detection방법에 비교해서 몇가지 이점을 갖음

YOLO 장점

- ✓ YOLO는 매우 빠르다.
  - detection을 single regression problem으로 정의 - 복잡한 pipeline이 필요하지 않음
  - 테스트 시간에 새로운 이미지로 신경망을 실행시킬 수 있습니다.
  - Titan X GPU상에서 batch 과정없이 45FPS가 나옴
  - 빠른 버젼은 150 FPS를 상회함.
  - YOLO는 다른 real-time system비해 2배의 mean average precision(mAP)의 성능
- ✓ image를 예측 시에 globally 하다
  - sliding window나 region proposal 기반의 기술들과는 다르게 학습이나 테스트 중에 전체 이미지를 보기 때문에, 은연중에 각 class의 대표적인 표현들에 대해서 맥락적인 정보를 encoding 함.
  - 강력한 detection방법인 Fast R-CNN경우에는 배경정보에 따라서 object를 detect하는데 있어서 실수가 있는데, 이것은 Fast R-CNN이 큰 맥락을 보지 못하기 때문임
  - YOLO는 Fast R-CNN과 비교했을 때, background error가 절반 정도 밖에 되지 않음.
- ✓ object의 일반적인 특징을 학습
  - object의 일반화를 잘 함.
  - 예술작품이나 자연의 이미지를 학습하거나 테스트할 때, R-CNN같은 강력한 detection방법들보다 학습을 더 잘합니다.
  - YOLO는 일반화에 강력하기 때문에, 새로운 도메인이나 예상하지 못한 input이 들어와도 망가질 확률이 적습니다.

YOLO Trade off

- ✓ 분류 능력은 다른 분류시스템보다 떨어짐
- ✓ 빠르게 object를 식별하나, 작은 물체를 검출하는 데는 애를 먹음
- ✓ 실험을 통하여 해당 내용과 관련된 trade off를 확인함

Unified Detection

- ✓ YOLO는 component들을 single convolution network에 통합
- ✓ 이미지로 얻은 feature들을 예측하고 탐지하는데 사용
- ✓ 네트워크는 이미지로부터 bounding box과 class들을 동시에 찾아냄
  - image에서 모든 image 종류와 모든 category를 찾는데 있어서 globally하다는 것을 의미
  - 높은 mAP를 유지하면서 실시간으로 학습이 가능하도록 end-to-end 방식으로 설계됨
- ✓ YOLO 시스템은 input image를 S x S grid로 나눔
  - object의 중심이 grid cell로 떨어지게 되면, 해당 grid cell은 해당 object를 detect하도록 책임이 주어짐
  - 각각의 grid cell은 B bounding box와 해당 bounding box에 대한 confidence score를 예측
  - confidence score는 해당 모델이 해당 box안에 object가 있을 확률이 얼마나 되는지, 그리고 해당 object가 자신이 예측한 object가 맞을 확률이 얼마나 되는지에 대한 확률에 대한 score
  - 이를 수식으로 표현하게 되면, 우리는 confidence를 $Pr(Class_i) * IOU_{pred}^{truth}$ 로 표현
  - grid cell에 object가 없으면 confidence score는 0

Unified Detection

- ✓ confidence score가 예측된 상자의 intersection over union(IOU)와 ground truth(label에서 나타낸 실제값) 같기를 원함.

- ✓ 각 bounding box는 5가지 요소로 구성되어있습니다.
  - x, y, w, h 와 confidence score
  - (x, y) 는 grid cell의 경계를 기준으로한 상자의 중심 좌표
  - width와 height는 예측된 object와 전체 이미지의 width와 height의 비율 의미
  - confidence는 예측된 box와 ground truth(실제값) 사이의 IOU값을 의미

- ✓ 각각의 cell은 C에 대해서 예측
  - C는 class의 조건부 확률 $Pr(Class_i|Object)$
  - C는 객체를 포함하는 grid cell에 따라 달라짐.
  - 하나의 grid cell은 예측된 상자수 B와는 상관없이 한가지 종류의 Class의 확률만 계산
  - 테스트 시에는 class의 조건부 확률과 각각의 개별적인 box에 대해서 confidence 예측

Unified Detection

- ✓ 각각의 box를 통해서 특정한 class에 대한 confidence score를 얻을 수 있음
- ✓ 이러한 score는 해당 box가 class를 나타내는지, 얼마나 box가 object와 잘 맞춰져 있는지에 대한 정보를 encoding하고 있음

$$Pr(Class_i|Object)*Pr(object)*IOU_{pred}^{truth} = Pr(Class_i)*IOU_{pred}^{truth} \quad (1)$$

P(A∩B)=P(A,B) : **교집합**
P(A | B) = P(A,B) / P(B) : **조건부 확률**
P(A,B) = P(A | B) x P(B) : **교집합**

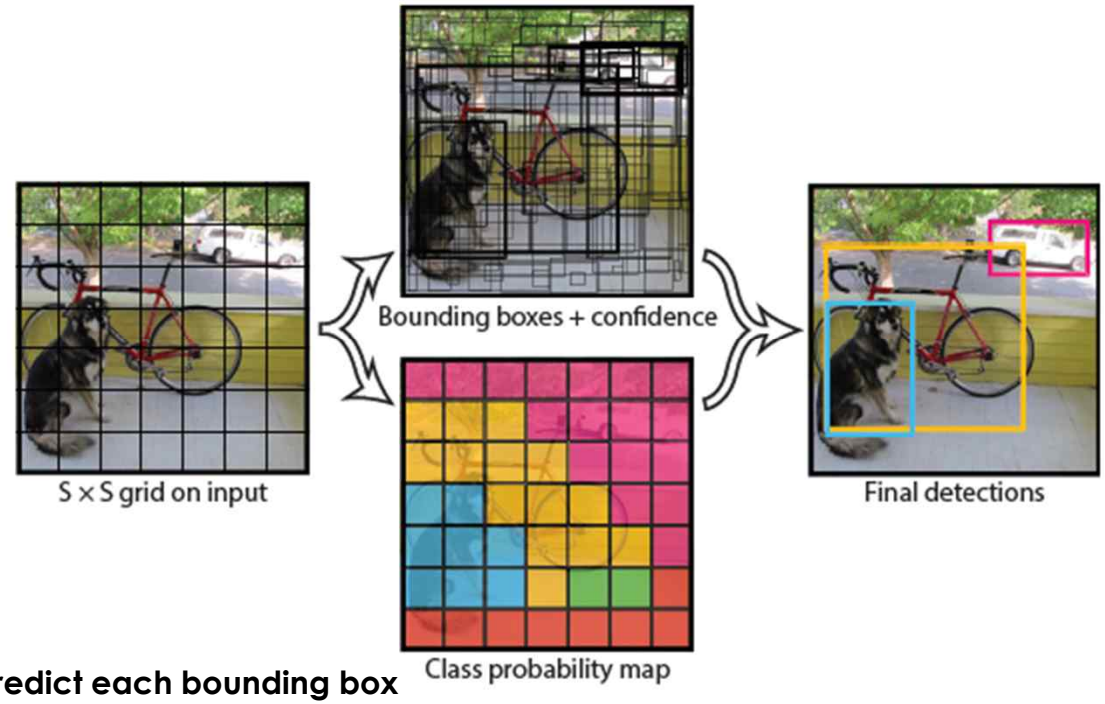
P(Class ∩ Object)=P(A, Object) : **교집합**
P(Class | Object) = P(Class, Object) / P(Object) : **조건부 확률**
P(Class, Object) = P(Class | Object) x P(Object) : **교집합**

How unified detection works?

regression problem으로 object를 detection
image를 S x S의 grid cell로 나눔
각각의 cell은 B개의 bounding box와 confidence score 예측
예측값은 S x S X (B * 5 + C) 크기의 tensor로 구성됨

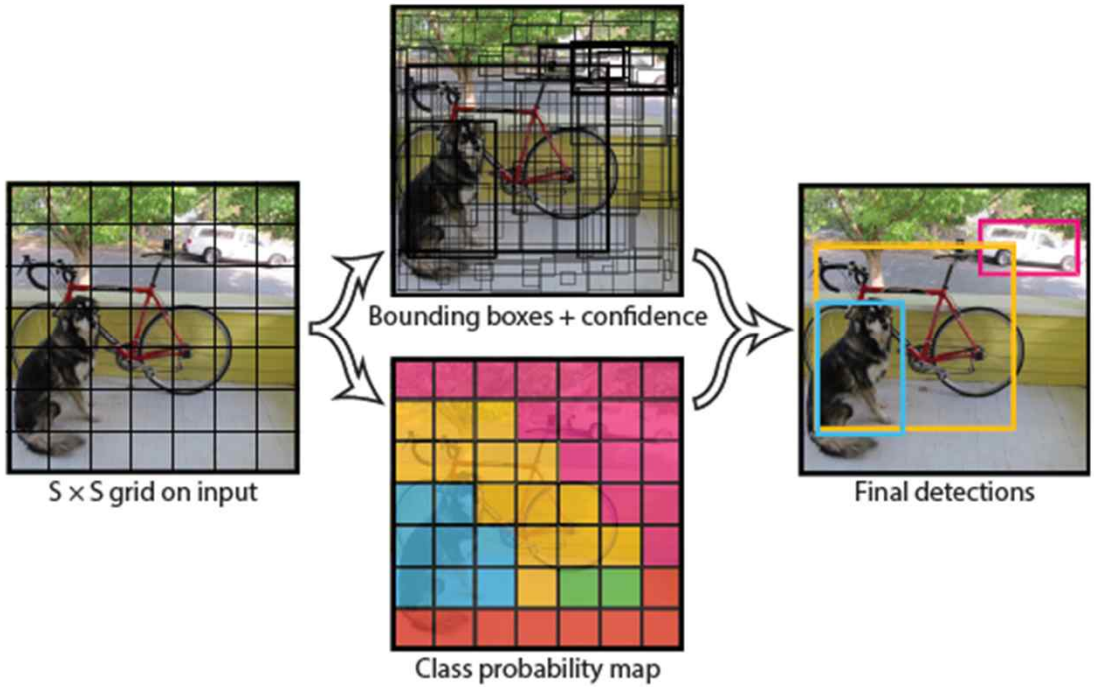
Pascal VOC **데이터를 이용해 평가**

S=7, B=2**사용**
Pascal VOC data**는** 20**개의** class**로** label **됨**
C=20
최종 예측 값 7 x 7 x 30 tensor**가 됨**



- **uses features from the entire image to predict each bounding box**
- **predicts all bounding boxes across all classes for an image simultaneously?**
  - **divides the input image into an s*s grid. If the center of an object falls into a grid cell, the cell is responsible for detecting that object.**
  - **each grid cell predicts B bounding boxes and confidence scores for those boxes**
  - **Each grid also predicts C conditional(conditioned on the grid cell containing an object) class probabilities**

How unified detection works?



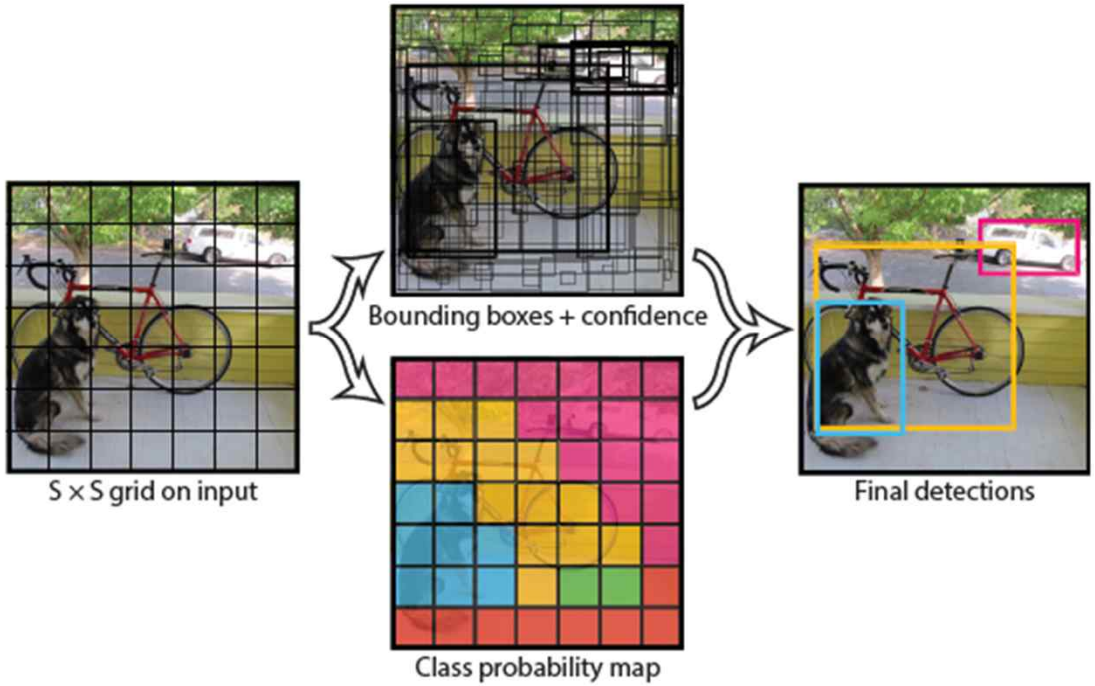
<u>confidence scores</u>: reflect how confident is that the box contains an object+how accurate the box is .

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

conditional class probabilities: conditioned on the grid cell containing an object

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

How unified detection works?



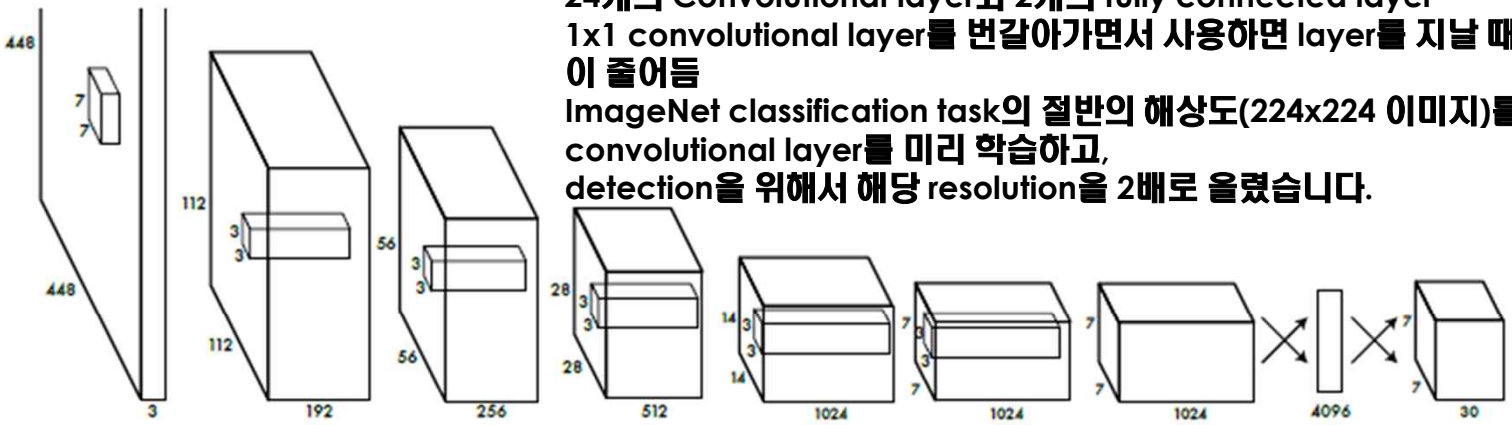
$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

- At test time, multiply the conditional class probabilities and the individual box confidence predictions
- giving class-specific confidence scores for each box
- Showing both the probability of that class appearing in the box and how well the predicted box fits the object

Network Design

- ✓ Convolutional neural network로 구현
- ✓ Pascal VOC detection dataset으로 평가
- ✓ 네트워크의 첫번째 Convolutional layer는 image로부터 특징을 추출
- ✓ fully connected layer는 출력확률과 좌표를 예측
- ✓ network 아키텍쳐는 이미지 분류를 위한 GoogLeNet 모델로부터 영감 받음
  - network는 24개의 Convolutional layer와 2개의 fully connected layer로 구성
  - GoogLeNet이 사용한 inception 모듈 대신 간단하게 1x1 reduction layer와 3x3 convolutional layer 사용

24개의 Convolutional layer와 2개의 fully connected layer
1x1 convolutional layer를 번갈아가면서 사용하면 layer를 지날 때마다 특징 공간이 줄어듬
ImageNet classification task의 절반의 해상도(224x224 이미지)를 이용하여 convolutional layer를 미리 학습하고,
detection을 위해서 해당 resolution을 2배로 올렸습니다.



Network Design

- ✓ 빠른 object detection의 한계를 넘게 설계된 YOLO의 빠른 버전을 학습
- ✓ Fast YOLO는 기존에 24개의 Convolution Layer를 사용하던 YOLO에 비교해서 9개로 더 적은 Convolution Layer를 사용
- ✓ 나머지 network의 크기나 학습, 테스트 파라미터들은 기존 YOLO와 Fast YOLO는 같음.
- ✓ network의 최종 output은 7x7x30 tensor로 출력

Training

- ✓ ImageNet 1000-class competition dataset을 이용하여 convolutional layer를 미리 학습
- ✓ Pretraining을 위해서 20개의 convolutional layer와 average-pooling layer그리고 fully connected layer를 사용
- ✓ Caffe의 모델 Zoo에서 GoogLeNet과 비교할 수 있게, ImageNet 2012 validation set을 이용해서 top-5의 정확도가 88%에 다를 때까지 1주일동안 학습
- ✓ 모든 학습을 Darknet framework을 사용
- ✓ detection을 수행할 수 있도록 모델 변경
- ✓ convolutional과 connected layer를 pretrained network에 추가하면 성능이 향상될 수 있다는 것을 보여주었다.
- ✓ 예제를 따라서 4개의 convolutional layer와 2개의 fully connected layer와 해당 layer의 weight들을 랜덤하게 초기화하여 기존 네트워크에 추가
- ✓ detection에는 종종 세부적인 시각정보를 요구하기 때문에, input image의 resolution을 향상시켜서 224 x 224의 image를 448 x 448 image로 변경해서 input image로 사용

Training

- ✓ 마지막 layer는 bounding box의 좌표와 class의 확률 모두 예측
- ✓ image의 width와 height와 bounding box의 width, height의 비율로 파라미터를 정규화
- ✓ 그 결과로 해당 값들은 0과 1사이로 정규화
- ✓ bounding box의 x와 y의 좌표를 특정한 grid cell의 위치에 offset되도록 매개변수화하여 0과 1사이의 경계 값을 만듬
- ✓ 마지막 layer에는 linear activation function을 사용하였고 나머지 다른 layer에는 leaky를 사용
- ✓ leaky rectified linear activaion은 다음 수식을 따름.

$$\phi(x) = \begin{cases} x, & if\ x > 0 \\ 0.1x, & otherwise \end{cases} \quad (2)$$

Training

- ✓ 모델의 output의 sum-squared error를 최적화
- ✓ error function을 sum-squared error를 사용
- ✓ sum-squared error를 사용한 이유는 최적화시키기 제일 쉽기 때문
- ✓ 하지만 sum-squared-error는 우리의 목적인 average precision을 최대화하는 데 있어서 완벽하게 부합하지는 않음
- ✓ 왜냐면 sum-squared-error는 weight들의 localization error는 classification error와 같다라고 판정하는데, 실제 우리의 세상은 이렇게 이상적이지 않기 때문
- ✓ 또한 모든 image는 많은 grid cell을 가지고 있고, 많은 grid cell은 어느 object도 포함하고 있지 않음.
- ✓ 이런 상태는 해당 cell의 confidence score를 zero로 만들고, 이것은 종종 object가 포함된 image의 grid cell의 gradient를 압도함
- ✓ 이러한 현상은 network를 불안정하게 만들고, 학습이 일찍 끝나도록 만듬 위의 현상을 방지하기 위해서 bounding box의 좌표 예측으로부터 생성된 loss값을 증가시키고, object가 포함되지 않을 거다라고 예측된 box의 confidence로부터 생성된 loss는 감소시켰습니다.

Training

✓ 학습하는 동안 다음과 같은 multi-part loss function을 따릅니다.

(1)Object가 존재하는 grid cell i의 predictor bounding box j에 대해, x와 y의 loss를 계산.

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

(2) Object가 존재하는 grid cell i의 predictor bounding box j에 대해, h의 loss를 계산. 큰 box에 대해서는 small deviation을 반영하기 위해 제곱을 취한 후, sum-squared error를 한다.(같은 error라도 larger box의 경우 상대적으로 IOU에 영향을 적게 준다.)

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

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

(3) Object가 존재하는 grid cell i의 predictor bounding box j에 대해, confidence score의 loss를 계산. (Ci= 1)

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

(4) Object가 존재하지 않는 grid cell i의 bounding box j에 대해, confidence score의 loss를 계산. (Ci = 0)

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

(5) Object가 존재하는 grid cell i에 대해, conditional class probability의 loss 계산. (Correct class c: pi(c) =1, otherwise: pi(c) =0)

λcoord: coordinates(x,y,w,h)에 대한 loss와 다른 loss들과의 균형을 위한 balancing parameter.

λnoobj: obj가 있는 box와 없는 box간에 균형을 위한 balancing parameter. (일반적으로 image내에는 obj가 있는 cell보다는 obj가 없는 cell이 훨씬 많으므로)

Training

- ✓ 여기서 은 만약에 i번째 cell에 object가 나타났다는 것을 의미합니다. 그리고 는 cell i의 j번째 bounding box predictor는 해당 예측에 대한 책임이 있다는 것을 의미함.
- ✓ 이러한 loss function은 object가 해당 grid cell에 존재할 때만, classification error에 대해서 처벌함.(앞에 이야기했던 조건부 클래스 확률).
- ✓ 마찬가지로 predictor가 ground truth box에 대해서 책임이 있는 경우에만 coordinate error에 대한 처벌을 함.(grid cell에서 가장 큰 IOU값을 갖는 특정한 predictor)
- ✓ Pascal VOC 2007과 2012의 training & validation data set을 사용해서 135번의 epoch로 Network를 학습시킴.
- ✓ VOC 2012를 학습시킬때, 2007 test data도 training용으로 포함했습니다. 우리는 64의 batch size와 0.9의 momentum 그리고 0.0005의 decay값을 설정.

Training

- ✓ 여기서 다음과 같은 2개의 파라미터 λ_{coord} 와 λ_{noobj} 를 사용해 다음과 같은 방법을 구현 여기서 $\lambda_{coord} = 5$ $\lambda_{noobj} = 0.5$ 설정하였습니다.
- ✓ Sum-squared-error는 또한 큰 box와 small box의 weights error와 같습니다.
- ✓ error metric은 large box들의 small deviation의 문제를 small box의 small deviation문제보다 더 작게 반영해야 함.
- ✓ 이런 문제를 처리하기 위해서 부분적으로 width와 height를 직접 예측하는 대신, width와 height의 제곱근을 직접 예측함.
- ✓ YOLO는 각 grid cell마다 multiple bounding boxes를 예측
- ✓ 학습 시 오직 각 object 마다 한 predictor에게 하나의 bounding box만을 찾도록 책임을 부여하며, 하나의 predictor에게 어느 예측이 ground truth와 제일 높은 highest IOU를 갖는지에 기반해서 object를 예측하도록 책임을 할당
- ✓ 이러한 형태는 bounding box predictor들 사이에서 전문성을 갖도록 유도함
- ✓ 각각의 predictor는 predicting certain size, aspect ratios, class 예측이 더 좋아지므로 전체적인 recall향상에 도움이 됨.

Training

- ✓ learning rate 스케쥴은 다음과 같다.
- ✓ 첫 epoch에서는 learning rate를 에서 으로 천천히 증가.
- ✓ 만약에 high learning rate으로 시작한다면 모델은 종종 불안정한 gradient들로 인해서 발산될 것임.
- ✓ 계속해서 으로 75번의 epoch를 돌렸고 그리고 으로 30번의 epochs 그리고 마지막으로 으로 30번의 epoch를 사용
- ✓ overfitting을 피하기 위해서 dropout과 확장된 데이터를 사용해서 데이터의 수를 늘렸음.
- ✓ dropout layer의 rate parameter는 0.5를 사용
- ✓ 이는 첫번째 connected layer와 layer들 사이의 동기화를 막아줌.
- ✓ data의 확장으로 인한 data 수를 확보는 여기서 임의적으로 scaling하거나 원본 이미지 크기의 최대 20%정도에서의 데이터 변환을 이용
- ✓ 또한 임의적으로 이미지의 exposure과 saturation를 최대 1.5배까지 임의로 조정.

Inference

- ✓ training 과정과 마찬가지로 test image의 detection를 예측하려면 네트워크 평가가 하나 필요합니다.
- ✓ PASCAL VOC에서 네트워크는 이미지 당 98 bounding box와 각 box에 대한 class 확률을 예측합니다.
- ✓ YOLO는 분류 기준 기반 방법과 달리 단일 네트워크 평가 만 필요하므로 테스트 시간이 매우 빠릅니다.
- ✓ grid 디자인은 bounding box 예측에서 공간적 다양성을 적용합니다.
- ✓ object가 어느 grid cell에 속하고 network가 각 object에 대해 하나의 box만 예측하는지는 분명합니다. 그러나 여러 cell의 테두리 근처에있는 큰 object나 object는 여러 셀로 잘 지역화 될 수 있습니다. 극대치가 아닌 억제를 사용하여 이러한 문제를 해결할 수 있습니다. multi object detection. R-CNN 또는 DPM의 경우처럼 성능에 중요하지는 않지만 최대가 아닌 억제는 mAP를 2-3 %향상시킵니다.

DATA

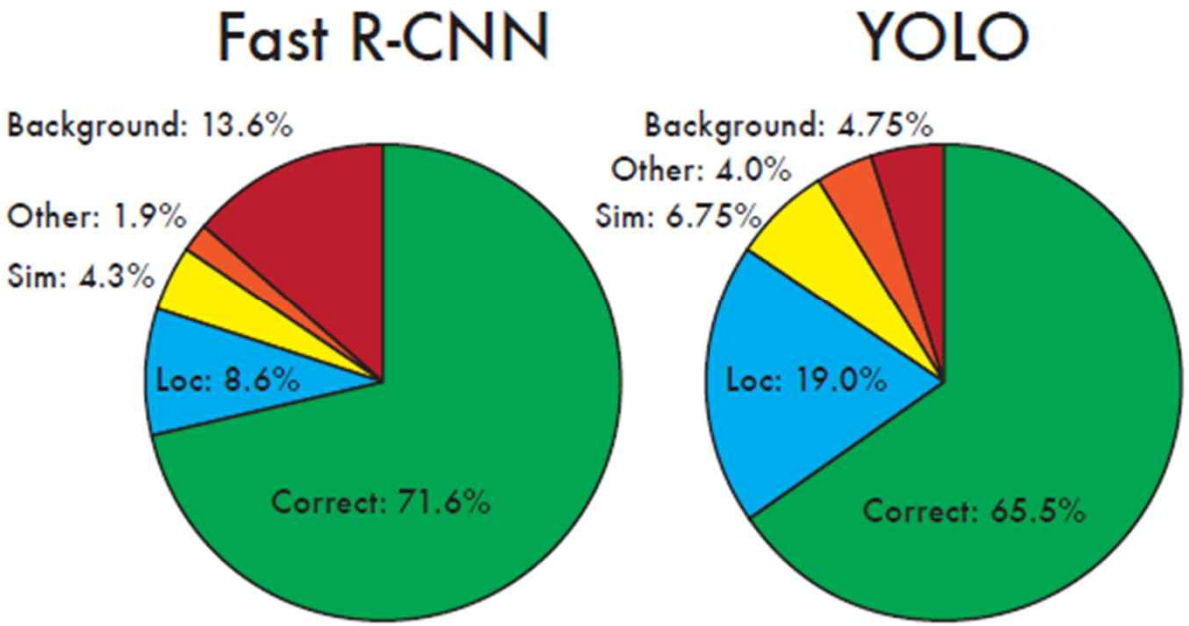
- Pretraining convolutional layers on the ImageNet 1000-class competition dataset
- Training data: VOC 2007 and 2012
- 135 epochs on the training
- Learning rate from 10^{-3} to 10^{-4}

Detection Analysis PASCAL VOC 2007test

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

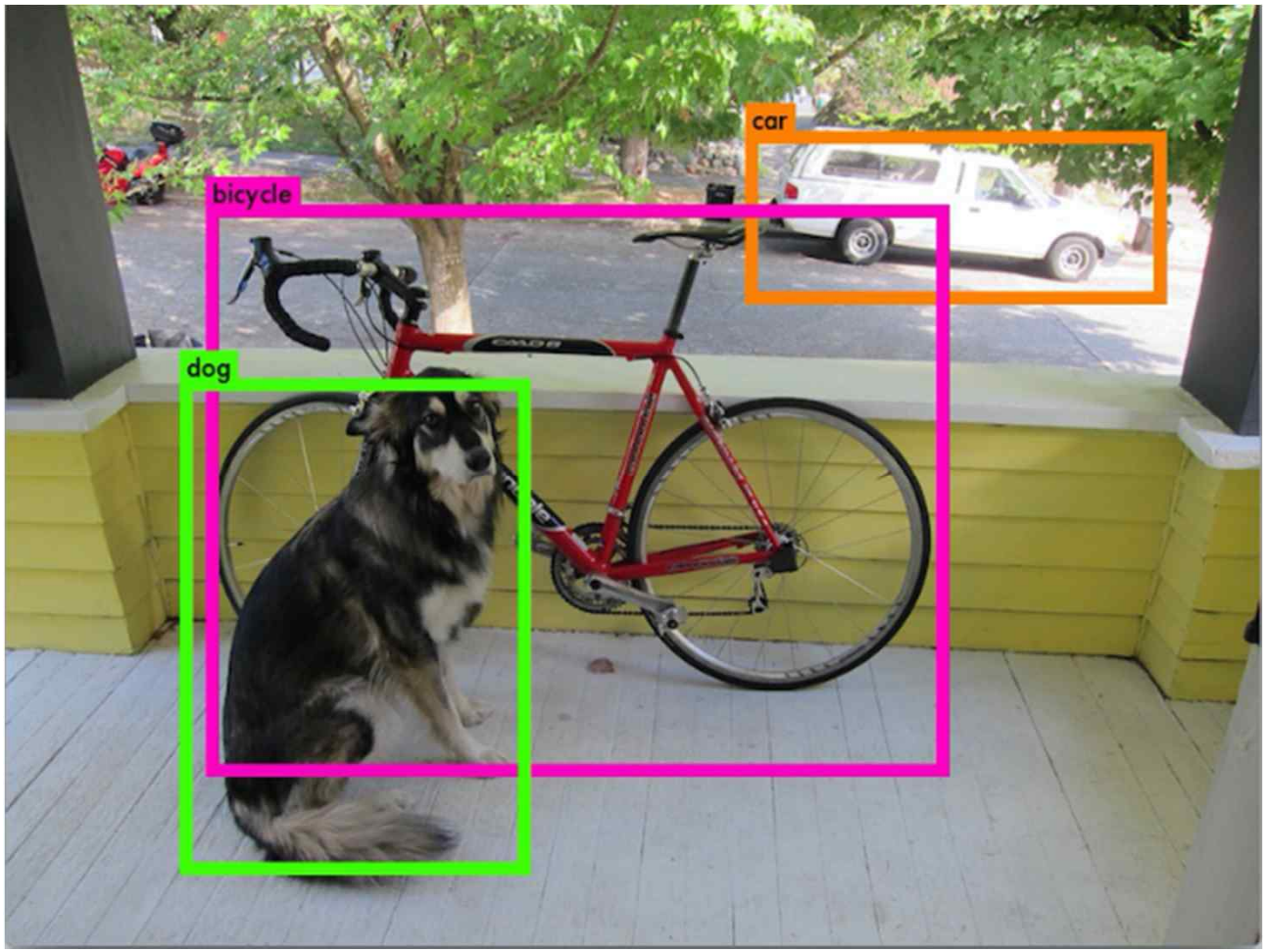
- **Fast YOLO is the fastest object detection method on PASCAL; With 52:7% mAP, it is more than twice as ac curate as prior work on real-time detection**
- **YOLO pushes mAP to 63:4% while still maintaining re al-time performance**

Detection Analysis：PASCAL VOC 2007test

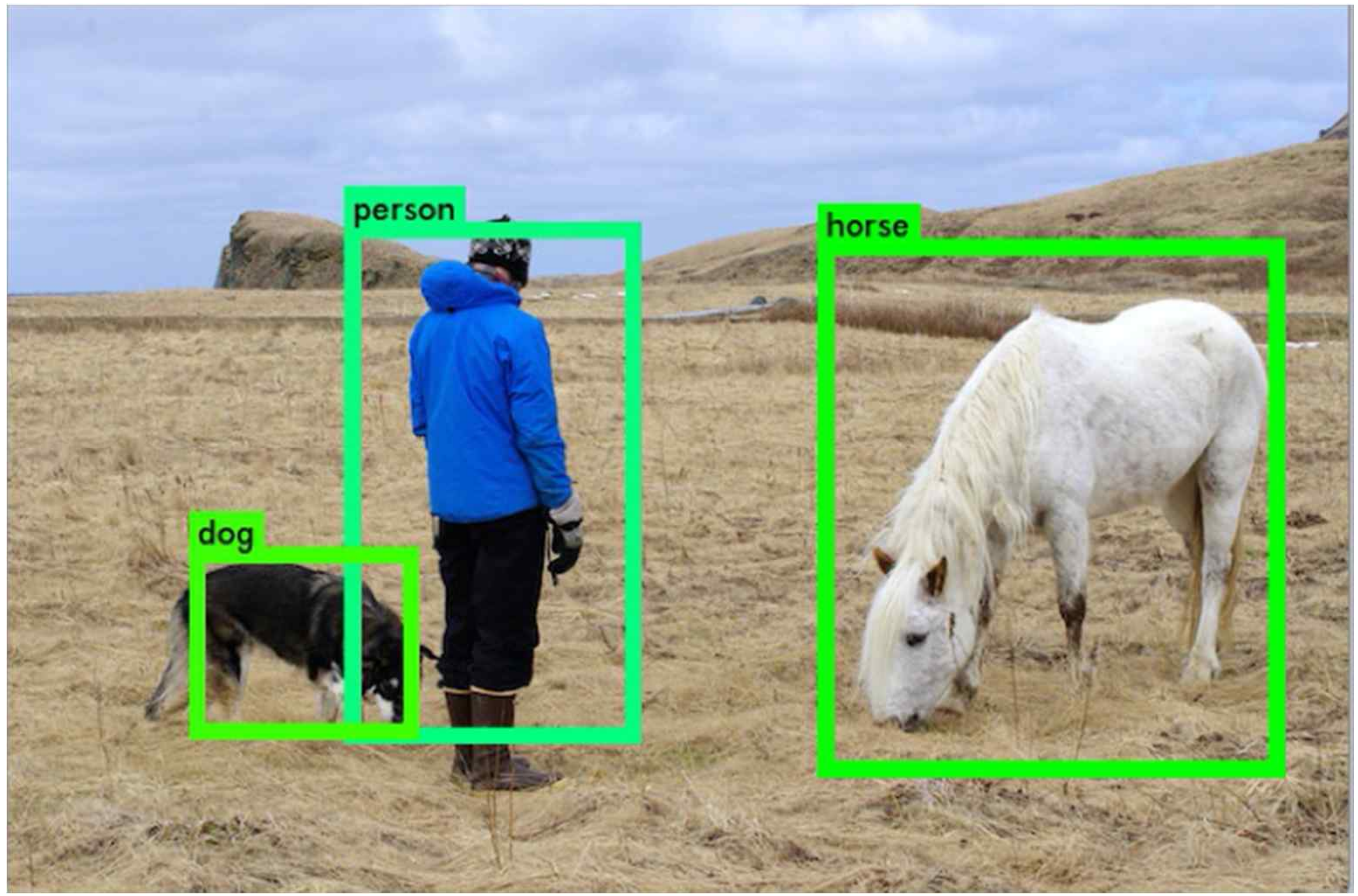


- Better ability to eliminate background error(sees the entire image during training and testing)
- Struggle to localize objects correctly(each grid cell only predicts two boxes and can only have one clas s)

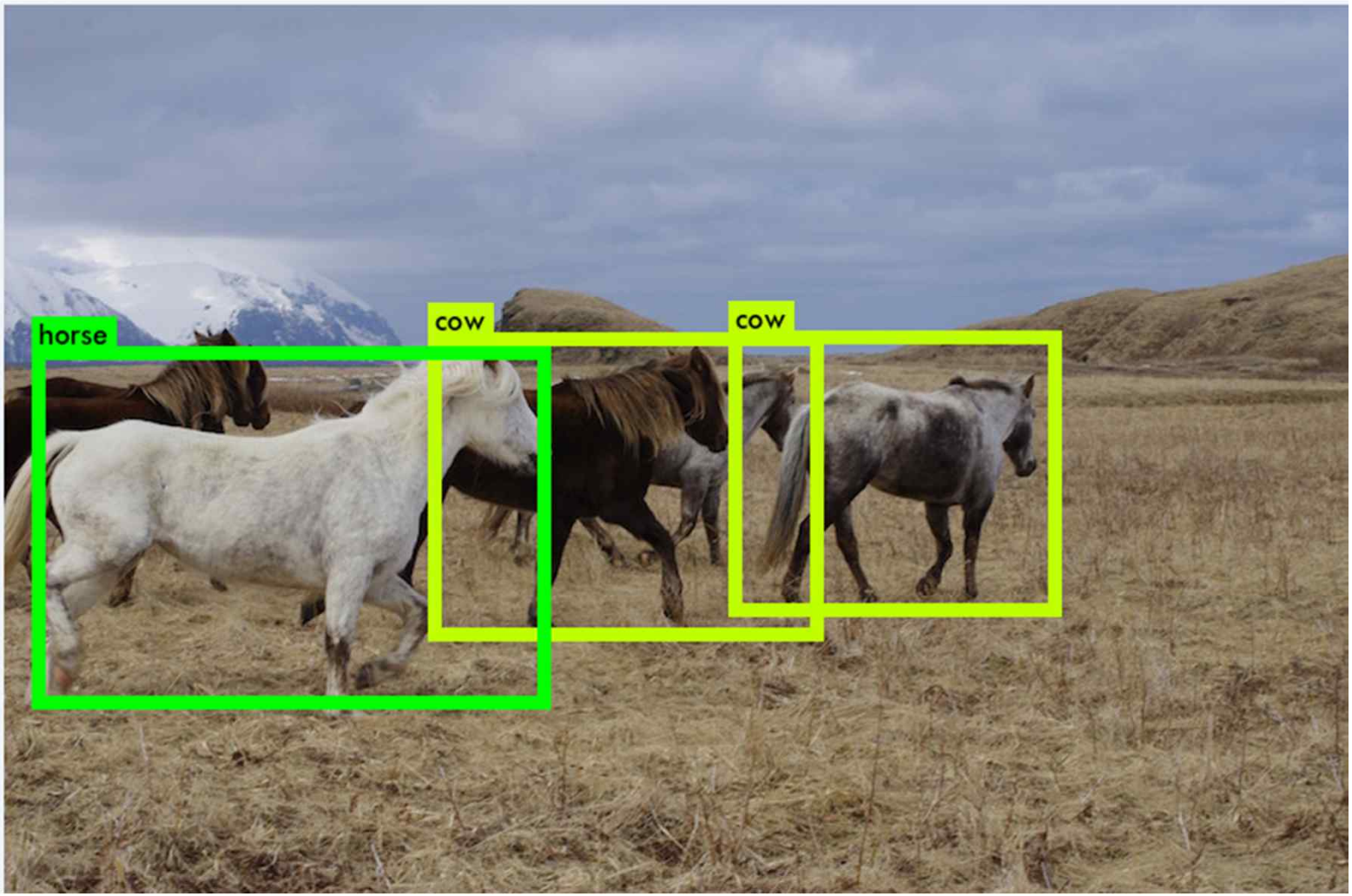
Detection 예



Detection 예



Detection 예



용어 정리

ground truth
사물의 실제 위치를 나타내는 '실제(ground truth; **이하** GT)**' 바운딩 박스 정보가 이미지 레이블 상에 포함되어 있습니다.**
학습하는 과정에서 실제 데이터 라벨(**또는 클래스**) - **이를** Ground Truth Label**이라고 합니다**

IOU(intersection over union)

Intersection : 교집합(Bp∩Bgt)
Union : 합집합 (Bp∪Bgt)
Detection 문제의 경우, 사물의 클래스 및 위치에 대한 예측 결과를 동시에 평가해야 하기 때문에, 사물의 실제 위치를 나타내는 '실제(ground truth; 이하 GT)' 바운딩 박스 정보가 이미지 레이블 상에 포함되어 있습니다. 검출 모델의 경우 복수 개의 예측 바운딩 박스를 제출할 수 있기 때문에, 이들 중 어떤 것을 GT 바운딩 박스와 매칭시킬지에 대한 규정이 마련되어 있습니다.

이를 위해, 각 예측 바운딩 박스 Bp와 GT 바운딩 박스 Bgt에 대하여, 아래와 같이 정의되는 IOU(intersection over union)를 사용하여 Bp와 Bgt가 서로 얼마나 '겹쳐지는지' 를 평가합니다.

Bp와 Bgt 의 IOU = Bp∩Bgt 영역 넓이 / Bp∪Bgt 영역 넓이

confidence score
해당 모델이 해당 box안에 object가 있을 확률이 얼마나 되는지, 그리고 해당 object가 자신이 예측한 object가 맞을 확률이 얼마나 되는지에 대한 확률에 대한 score

$$Pr(Class_i|Object)*Pr(object)*IOU_{pred}^{truth} = Pr(Class_i)*IOU_{pred}^{truth} \quad (1)$$

용어 정리

mAP(mean Average Precision)

$$Pr(Class_i|Object)$$

각각의 cell은 C에 대해서 예측
C는 class의 조건부 확률
C는 객체를 포함하는 grid cell에 따라 달라짐.
하나의 grid cell은 예측된 상자수 B와는 상관없이 한가지 종류의 Class의 확률만 계산
테스트 시에는 class의 조건부 확률과 각각의 개별적인 box에 대해서 confidence 예측

$$Pr(Class_i|Object)*Pr(object)*IOU_{pred}^{truth} = Pr(Class_i)*IOU_{pred}^{truth} \quad (1)$$

$$Pr(object)$$

$$Pr(Class_i)$$

P(A∩B) = P(A,B) : **교집합**
P(A | B) = P(A,B) / P(B) : **조건부 확률**
P(A,B) = P(A | B) x P(B) : **교집합**

P(Class ∩ Object)=P(A, Object) : **교집합**
P(Class | Object) = P(Class, Object) / P(Object) : **조건부 확률**
P(Class, Object) = P(Class | Object) x P(Object) : **교집합**