[week9] 논문리뷰

YOLO

1. Introduction

💡 논문에서 다루고 있는 주제가 무엇인지와 해당 주제의 필요성이 무엇인가
논문에서 제안하는 방법이 기존 방법의 문제점에 대응되도록 제안 되었는가

논문이 다루는 분야

- **CV - Object Detection**
- **인간** : 이미지를 한 번 쓱 보기만 해도 어떤 객체가 있는지, 어디에 있는지, 서로 어떻게 상호작용하는지 바로 알아챔. 인간의 시각 시스템은 빠르고 정확해서, 운전 같은 복잡한 일도 별생각 없이 함
- **기계** : 빠르고 정확한 객체 탐지 알고리즘이 있다면, 컴퓨터도 특수 센서 없이 자율 주행을 할 수 있고, 사람에게 실시간 장면 정보를 전달하는 보조 장치나 범용 로봇 시스템에도 활용할 수 있음

해당 task에서 기존 연구 한계점

1. **기존 객체 탐지 방식**
    - 기존의 객체 탐지 시스템은 classifier를 재활용해서 detection
    - 객체를 탐지하기 위해, 특정 객체에 대한 classifier를 테스트 이미지의 여러 위치와 크기에서 반복적으로 적용
    - DPM(Deformable Parts Model) 같은 시스템은 sliding window 방식으로 전체 이미지에서 classifier를 반복적으로 실행
2. **R-CNN 방식의 복잡성**
    - R-CNN 같은 최신 방법은 먼저 region proposal을 사용해 후보 bounding box를 만들고, 거기에 classifier를 적용
    - 분류가 끝나면, bounding box를 정제하거나 중복된 탐지를 제거하고, 주변 객체를 고려해 다시 점수를 매기는 post-processing 단계를 거침

      ⇒ 파이프라인이 복잡하고, 각 구성요소를 따로 학습시켜야 해서 느리고 최적화도 어려움

논문의 contributions

1. **YOLO의 접근 방식**
    - 객체 탐지를 regression problem으로 다시 정의
    - 이미지 픽셀을 입력으로 받아 바운딩 박스 좌표와 클래스 확률을 바로 예측

- 이 시스템에서는 이미지를 딱 한 번만 보면("You Only Look Once!!") 어떤 객체가 어디에 있는지를 바로 예측할 수 있음

2. **YOLO의 구조 및 장점**

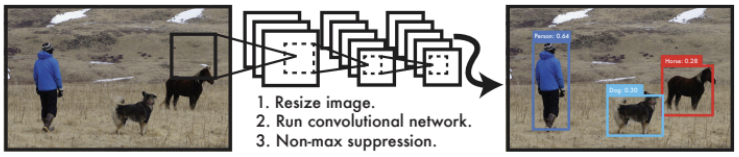


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

- 간단한 구조 → 하나의 convolutional network가 여러 바운딩 박스와 각 박스의 클래스 확률을 동시에 예측
- YOLO는 전체 이미지를 기반으로 학습하며 탐지 성능을 직접적으로 최적화
- 이 unified model은 기존 방식과 비교해서 여러 장점을 가짐

    1. **속도가 빠름**

        : detection을 회귀 문제로 보기 때문에 복잡한 파이프라인이 필요 없음. 테스트할 때는 새 이미지를 입력해 네트워크를 한 번 실행하면 detection 결과가 나옴.
        : 기본 모델은 Titan X GPU에서 배치 처리 없이 초당 45프레임으로 동작, 빠른 버전은 초당 150프레임 이상. 즉, 실시간 스트리밍 영상도 지연 시간 25ms 이하로 처리할 수 있음.
        : 또한 YOLO는 다른 실시간 시스템보다 두 배 이상 높은 평균 정밀도(mAP)를 보임.

    2. **이미지 전체를 보고 판단하여 예측**

        : sliding window나 region proposal 방식과 달리, YOLO는 훈련과 테스트 시 전체 이미지를 보기 때문에 클래스의 외형뿐 아니라 문맥적 정보도 함께 학습함.
        : 예를 들어, Fast R-CNN은 전체 맥락을 보지 못해서 배경 일부를 객체로 착각하기도 함. YOLO는 Fast R-CNN보다 배경 오탐을 절반 이하로 줄였음.

    3. **일반화 가능한 객체 표현을 학습함**

        : 자연 이미지를 학습하고 예술 작품에서 테스트해도 YOLO는 DPM이나 R-CNN보다 훨씬 더 좋은 성능을 보임.

: 일반화 성능이 뛰어나서 새로운 도메인이나 예기치 않은 입력에서도 성능이 급격히 떨어지지 않음.

3. **YOLO의 한계**
    - 정확도 면에서는 SOTA보다는 조금 떨어짐
    - 이미지를 빠르게 분석하긴 하지만, 작은 객체의 정확한 위치를 잡는 데는 어려움이 있음

2. 제안 방법론

Introduction에서 언급된 내용과 동일하게 작성되어 있는가
Introduction에서 언급한 제안 방법이 가지는 장점에 대한 근거가 있는가
제안 방법에 대한 설명이 구현 가능하도록 작성되어 있는가

- **제안 방법론 Reference** : https://bkshin.tistory.com/entry/%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-YOLOYou-Only-Look-Once

Unified Detection

- 객체 검출의 개별 요소를 단일 신경망(single neural network)으로 통합
- 각각의 bounding box를 예측하기 위해 이미지 전체의 특징을 활용
- 높은 정확성을 유지하면서 end-to-end 학습과 실시간 객체 검출 가능

- 입력 이미지를 S x S grid로 나눔 → 만약 어떤 객체의 중심이 특정 그리드 셀(grid cell) 안에 위치 → 그 그리드 셀이 해당 객체를 검출
- 각각의 그리드 셀은 B개의 bounding box와 그 bounding box에 대한 confidence score를 예측

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

    - confidence score : bounding box가 객체를 포함한다는 것을 얼마나 믿을만한지, 그리고 예측한 bounding box가 얼마나 정확한지 (아래의 식으로 계산)
    - IOU = intersection over union
      = 객체의 실제 bounding box와 예측 bounding box의 합집합 면적 대비 교집합 면적의 비율

= (실제 bounding box와 예측 bounding box의 교집합) / (실제 bounding box와 예측 bounding box의 합집합)

  - 그리드 셀에 아무 객체가 없으면 → Pr(Obejct)=0 → confidence score= 0
    그리드 셀에 어떤 객체가 확실히 있다고 예측 → Pr(Object)=1일 때가 가장 이상적

⇒ 따라서 confidence score가 IOU와 같다면 가장 이상적인 score입니다.

- 각각의 bounding box는 5개의 예측치로 구성되어 있음 [x, y, w, h, confidence]
  - (x, y) : bouning box 중심의 그리드 셀(grid cell) 내 상대 위치. (절대 위치X → 0~1 사이의 값을 가짐) : bounding box의 중심이 정확히 그리드 셀 중앙에 위치 → (x, y)=(0.5, 0.5)
  - (w, h) :bounding box의 상대 너비와 상대 높이. 이미지 전체의 너비와 높이를 1이라고 가정→ (w, h)도 역시 0~1 사이의 값을 가짐.
  - confidence = confience score
- 각각의 그리드 셀은 conditional class probabilities(C)를 예측. (다음과 같이 계산)

$$C(conditional\ class\ probabilities) = \Pr(Class_i|Object)$$

  - 객체가 어떤 클래스(class)인지에 대한 조건부 확률입니다.
  - 그리드 셀에 몇 개의 bounding box가 있는지와는 무관하게 하나의 그리드 셀에는 오직 하나의 클래스(class)에 대한 확률 값만을 구함.
    하나의 그리드 셀은 B개의 bounding box를 예측한다고 했습니다. B의 개수와는 무관하게 하나의 그리드 셀에서는 클래스 하나만 예측.
- 테스트 단계 : 각 bounding box에 대한 class-specific confidence score 계산 (다음과 같이 계산)

$$\begin{aligned} &class\ specific\ confidence\ score \\ &= \Pr(Class_i|Object) * \Pr(Object) * IOU_{pred}^{truth} \\ &= \Pr(Class_i) * IOU_{pred}^{truth} \end{aligned}$$

  - conditional class probability(C)와 개별 boudning box의 confidence score를 곱함.
  - 이 score는 bounding box에 특정 클래스 객체가 나타날 확률(=Pr(Classi))과 예측된 bounding box가 그 클래스 객체에 얼마나 잘 들어맞는지(=IOU)를 나타냄

- YOLO 연구진은 S=7, B=2로 세팅, 파스칼 VOC는 총 20개의 라벨링 된 클래스 → C=20

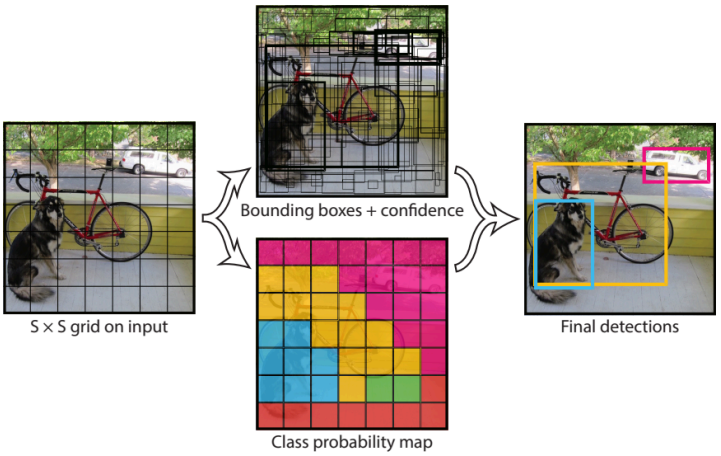


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

  - S=7 → 인풋 이미지는 7 x 7 그리드로 나뉨
  - B=2 → 하나의 그리드 셀에서 2개의 bounding box를 예측.
  - 결과 : S x S x (B*5 + C) 텐서 생성 → 최종 예측 텐서의 dimension = 7 × 7 × 30

Network Design

- YOLO 모델은 하나의 CNN(Convolutional Neural Network) 구조로 디자인되었음.
- convolutional layer (이미지로부터 특징을 추출) + fully-connected layer (클래스 확률과 bounding box의 좌표 예측)
- YOLO의 신경망 구조 : 이미지 분류(image classification)에 사용되는 GoogLeNet에서 따왔음
  - 총 24개의 convolutional layers와 2개의 fully connected layers으로 구성.
  - 1 × 1 reduction layer와 3 x 3 convolutional layer의 결합이 GoogLeNet의 인셉션 구조를 대신함
  - 모델의 최종 아웃풋은 7 x 7 × 30의 prediction tensors
  - Figure3 : YOLO 모델의 전체 구조.

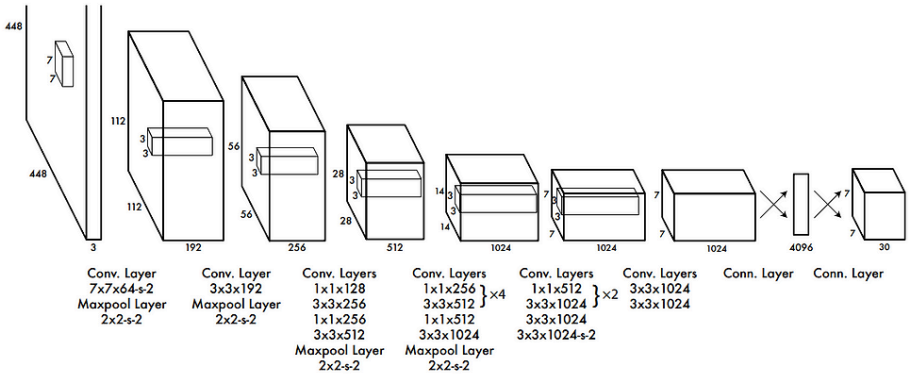


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

- Fast YOLO : 좀 더 빠른 객체 인식 속도를 위해 YOLO보다 더 적은 convolutional layer(24개→9개)와 필터를 사용.
    - 크기만 다르고 훈련 및 테스트 시 사용하는 나머지 파라미터는 YOLO와 모두 동일함

Training

- ImageNet 데이터 셋으로 YOLO의 앞단 20개의 convolutional layer을 pre-training
- 사전 학습된 20개의 convolutional layer 뒤에 4개의 convolutional layer 및 2개의 fully connected layer을 추가
- YOLO 신경망의 마지막 layer에는 linear activation function를 적용, 나머지 모든 layer에는 leaky ReLU를 적용

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

- 구조상 문제 해결을 위해 아래 3가지 방법 사용 (아래 사진은 loss function)
    1. localization loss와 classification loss 중 localization loss의 가중치를 증가시킴
    2. 객체가 없는 그리드 셀의 confidence loss보다 객체가 존재하는 그리드 셀의 confidence loss의 가중치를 증가시킴

3. bounding box의 너비(widht)와 높이(hegith)에 square root를 취해준 값을 loss function으로 사용함

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

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

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

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

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

- 과적합(overfitting)을 막기 위해 드롭아웃(dropout)과 data augmentation을 적용

Inference

- 훈련 단계와 마찬가지로, 추론 단계에서도 테스트 이미지로부터 객체를 검출하는 데에는 하나의 신경망 계산만 하면 됨. → 테스트 단계에서 굉장히 빠름
- 파스칼 VOC 데이터 셋에 대해서 YOLO는 한 이미지 당 98개의 bounding box를 예측 → bounding box마다 클래스 확률을 구함.
- YOLO의 그리드 디자인(grid design)의 단점

  → 하나의 객체를 여러 그리드 셀이 동시에 검출하는 경우가 있음
  → 객체의 크기가 크거나 객체가 그리드 셀 경계에 인접해 있는 경우, 그 객체에 대한 bounding box가 여러 개 생길 수 있음 = 다중 검출(multiple detections) 문제
  → non-maximal suppression이라는 방법을 통해 개선 가능. YOLO는 비 최대 억제를 통해 mAP를 2~3%가량 향상시켰음

Limitations of YOLO

- YOLO는 하나의 그리드 셀마다 두 개의 bounding box를 예측 & 하나의 그리드 셀마다 오직 하나의 객체만 검출.

→ 이는 공간적 제약(spatial constraints)을 야기. (하나의 그리드 셀에 두 개 이상의 객체가 붙어있으면 이를 잘 검출하지 못한다는 의미).
→ 예를 들어, 새 떼와 같이 작은 물체가 몰려 있는 경우 공간적 제약 때문에 객체 검출이 제한적.

- YOLO 모델은 데이터로부터 bounding box를 예측하는 것을 학습하기 때문에 훈련 단계에서 학습하지 못했던 새로운 종횡비(aspect ratio, 가로 세로 비율)를 마주하면 고전함
- YOLO 모델은 큰 bounding box와 작은 bounding box의 loss에 대해 동일한 가중치를 둔다는 단점이 있음. ⇒ 부정확한 localization 문제 발생

  → 크기가 큰 bounding box는 위치가 약간 달라져도 비교적 성능에 별 영향을 주지 않음
  → BUT 크기가 작은 bounding box는 위치가 조금만 달라져도 성능에 큰 영향을 줄 수 있음
  → 큰 bounding box에 비해 작은 bounding box가 위치 변화에 따른 IOU 변화가 더 심하기 때문

3. Related Work

💡 Introduction에서 언급한 기존 연구들에 대해 어떻게 서술하는가
제안 방법의 차별성을 어떻게 표현하고 있는가

> **Comparison to Other Detection Systems**
> : 일반적인 객체 탐지 파이프라인은 먼저 이미지에서 feature를 추출하고 classifier나 localizer를 이용해 feature space에서 객체를 식별함 (EX > Haar, SIFT, HOG, convolutional features)
> : 이러한 classifier나 localizer는 전체 이미지를 sliding window 방식으로 훑거나, 이미지의 특정 부분에서만 실행됨

Deformable Parts Models (DPM)

- sliding window 기반 객체 탐지 방식
- 여러 단계를 분리된(disjoint) 파이프라인으로 처리
- 고정된(static) feature를 추출하고, 영역을 분류하고, 높은 점수를 가진 영역에 대해 바운딩 박스를 예측하는 방식

> YOLO는 이러한 모든 분리된 단계를 하나의 convolutional neural network로 대체함
> YOLO는 feature 추출, 바운딩 박스 예측, non-max suppression, contextual reasoning을 동시에 수행함
> YOLO는 고정된 feature를 사용하지 않고, 학습 중 직접 특징을 최적화함. 이러한 통합 구조 덕분에 DPM보다 더 빠르고 정확한 모델을 만들 수 있음

R-CNN

- R-CNN과 이에서 비롯한 여러 모델은 sliding window 대신 region proposal 방식을 사용해 이미지에서 객체를 찾음
- Selective Search : 후보 바운딩 박스 생성 → Convolutional Network : 특징 추출 → SVM : 박스에 점수 매김 → Linear Model : 바운딩 박스 조정 → Non-max Suppression : 중복 탐지 제거
- 파이프라인 복잡 & 각 단계는 따로 정교하게 튜닝되어야 함
- 전체 시스템 매우 느려서 테스트 시 이미지 한 장당 40초 이상 걸릴 수 있음.

> YOLO와 R-CNN은 각 grid cell이 바운딩 박스 후보를 생성하고 convolutional features를 이용해 점수를 매긴다는 공통점 있음
> BUT, YOLO는 grid cell proposal에 공간적 제약(spatial constraints)을 걸어 같은 객체가 중복 탐지되는 문제를 줄임
> 또한 Selective Search는 이미지 당 약 2000개의 박스를 생성하는 반면, YOLO는 이미지당 오직 98개의 박스만 제안함
> YOLO는 이 모든 단계를 하나의 통합된 모델로 묶어 **jointly optimized**된 구조로 만듬

Other Fast Detectors

- Fast R-CNN과 Faster R-CNN은 Selective Search 대신 **신경망을 사용해 region proposal을 생성**하고, 연산을 공유함으로써 R-CNN 프레임워크를 더 빠르게 만듬
- R-CNN보다 속도와 정확도는 향상되었지만, 여전히 real-time 성능은 미치지 못합니다.

- 많은 연구들이 DPM 파이프라인의 속도를 높이는 데 초점을 맞추고 있습니다. (예를 들어, HOG 계산을 빠르게 하고, cascade를 사용하며, 연산을 GPU로 옮기는 등의 방식이 사용됩니다.)
- 하지만, 이 중 실제로 실시간으로 작동하는 경우는 드물며, 30Hz 속도의 DPM만이 실제로 실시간에 가까운 성능을 냅니다.

> YOLO는 파이프라인을 단계별로 최적화하려 하지 않고, 아예 전체 파이프라인을 없애고 처음부터 빠른 구조로 설계되었음.
> 단일 클래스 탐지를 위한 classifier(얼굴이나 사람 등 탐지)는 객체의 모습의 변화폭이 작아 성능을 높게 최적화할 수 있음.
> YOLO는 단일 클래스 탐지가 아닌 다양한 객체를 동시에 탐지할 수 있는 general purpose detector임.

Deep MultiBox

- Selective Search 대신, convolutional neural network를 이용해 region of interest을 예측하도록 학습
- MultiBox는 single object detection도 가능, confidence prediction을 단일 클래스 예측으로 바꾸어 작동 가능
- 하지만 MultiBox는 general object detection은 수행하지 못하고, 전체 파이프라인의 한 조각으로써 이미지 패치를 분류하는 추가 과정이 필요함

> YOLO와 MultiBox 모두 convolutional network를 이용해 바운딩 박스를 예측한다는 공통점이 있음. BUT, YOLO는 하나의 unified detection system임

OverFeat

- convolutional network를 학습시켜 localization을 수행하게 하고, 그 localizer를 객체 탐지에 사용
- sliding window 방식으로 동작하지만, 여전히 disjoint 시스템임 (↔)
- localization은 잘 하지만, 객체 탐지 성능 자체는 높지 않음.

- DPM처럼 localizer는 예측할 때 local information만 보기 때문에, 전체 맥락을 이해할 수 없음. (↔)
- 결과적으로 많은 post-processing 을 거쳐야만 일관된 탐지 결과를 얻을 수 있음.

MultiGrasp

- YOLO는 MultiGrasp 시스템과 유사한 구조를 가지고 있음.
- YOLO의 바운딩 박스 예측을 위한 grid 기반 접근 방식은 MultiGrasp 시스템의 grasp regression 방식에서 영감을 받았음.
- BUT, grasp detection은 object detection보다 훨씬 단순한 작업
  - MultiGrasp는 하나의 객체가 있는 이미지에 대해 grasp 가능한 하나의 영역만 예측하면 됨.
  - 객체의 크기, 위치, 경계, 클래스 등을 추정할 필요 없이, 그저 잡을 수 있는 영역만 찾으면 됨

> 반면 YOLO는 이미지 안의 여러 객체에 대해, 바운딩 박스와 클래스 확률 모두를 예측함

4. 실험 및 결과

💡 Introduction에서 언급한 제안 방법의 장점을 검증하기 위한 실험이 있는가

- **실험 및 결과 해설 Reference** : https://bkshin.tistory.com/entry/%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-YOLOYou-Only-Look-Once

Dataset

- PASCAL VOC 2007 / VOC 2012 데이터셋
- 두 가지 예술 이미지 데이터셋

Baseline

- YOLO vs real-time detection systems

- VOC 2007
  - YOLO와 Fast R-CNN이 만든 오류들 분석

    → YOLO랑 Fast R-CNN은 서로 다른 실수를 하기 때문에, YOLO를 이용해 Fast R-CNN의 탐지 결과를 다시 점검하면, 실수를 줄이고 정확도를 더 높일 수 있음
- VOC 2012
  - SOTA와의 mAP(정확도) 비교
- 두 가지 예술 이미지 데이터셋
  - YOLO가 다른 detector보다 새로운 도메인에 더 잘 일반화됨

4.1. Comparison to Other Real-Time Systems

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.

- 객체 검출에 대한 많은 연구들은 표준화된 객체 검출 파이프라인을 빠르게 만드는데 초점을 두고 있음.
- GPU 버전의 YOLO와 30Hz/100Hz에서의 DPM의 성능을 비교

- Fast YOLO

  → 파스칼 데이터 셋 기준으로 가장 빠른 객체 검출 모델
- R-CNN minus R

  → selective search를 정적 bounding box proposal로 대체한 모

→ mAP : 52.7% = 30Hz/100Hz에서의 DPM보다 2배 높은 정확도

- YOLO

  → 실시간(real-time) 성능은 그대로 유지하며 mAP를 63.4%까지 높였음

- VGG-16을 사용하여 YOLO를 훈련시킴

  → 일반 YOLO에 비해 mAP가 더 높지만 속도는 다소 느렸음.

- Faster R-CNN 모델

  → bounding box proposal을 위해 selective search를 사용하지 않고 신경망을 사용

  → 기존의 R-CNN 계열보다는 속도가 빨라졌지만 여전히 YOLO에 비해서는 몇 배나 느림

델.

→ R-CNN에 비해서는 훨씬 빠르지만 실시간(real-time) 검출에는 부족.

→ Fast R-CNN은 높은 정확도를 보이고 R-CNN보다 빠르지만 초당 0.5 프레임만 처리할 수 있어 실시간 검출에 사용하기엔 부족

- Fast DPM

  → DPM의 mAP를 약간 하락시키면서 속도를 향상시킨 모델

  → 실시간(real-time) 검출에 사용하기에는 속도가 느림.

  → 신경망을 이용한 객체 검출 모델에 비해 정확도도 떨어짐

- [Table1]

  각종 객체 검출 모델 별 정확도(mAP)와 속도(FPS)를 보여줌. FPS가 30 = 1초에 30 프레임의 영상을 처리. (30 이상이어야 실시간 처리 가능하다고 판단하는듯함)
  정확도는 Fast R-CNN과 Faster R-CNN VGG-16이 가장 높지만 이 모델들의 FPS는 너무 낮아 실시간 객체 검출 모델로 사용할 수는 없음.
  반면, YOLO 계열 모델은 정확도도 적당히 높고 속도도 빨랐음.

4.2. VOC 2007 Error Analysis

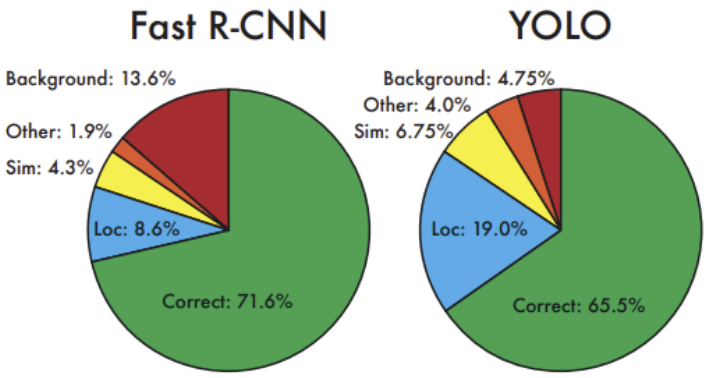


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

- 파스칼 VOC 2007 데이터 셋에 대해 YOLO와 Fast R-CNN의 성능을 비교
- Diagnosing Error in Object Detectors 논문에 소개된 에러 측정 방법론을 사용.
  - Correct : class가 정확하며 IOU > 0.5 인 경우
  - Localization : class가 정확하고, 0.1 < IOU < 0.5 인 경우
  - Similar : class가 유사하고 IOU > 0.1 인 경우
  - Other : class는 틀렸으나, IOU > 0.1 인 경우
  - Background : 어떤 Object라도 IOU < 0.1 인 경우

- YOLO는 localization error 가 상대적으로 큼.
- localization error가 19.0%로 나머지 error를 모두 합한 15.5% (6.75%+4.0%+4.75%) 보다 큼
- Fast R-CNN은 YOLO에 비해 localization error가 작은 반면, background error가 상대적으로 큼. YOLO에 비해 background error가 3배나 더 큼.
  - backgound error는 배경에 아무 물체가 없는데 물체가 있다고 판단하는 false positive error임.

4.3. Combining Fast R-CNN and YOLO

	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

Table 2: Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

- YOLO는 Fast R-CNN에 비해 background error가 훨씬 적음.

  ⇒ Fast R-CNN에 YOLO를 결합하여 background error를 줄인다면 굉장히 높은 성능을 낼 수 있을 것

  : R-CNN이 예측하는 모든 boudning box에 대해 YOLO도 유사하게 예측하는지를 체크.
  : 만약 R-CNN이 예측한 bounding box와 YOLO가 예측한 bounding box가 유사하다면 두 bounding box가 겹치는 부분을 bounding box로 잡으면 됨.
- 파스칼 VOC 2007 데이터 셋에 대해 가장 성능이 좋은 Fast R-CNN 모델은 71.8%의 mAP를 기록했음.
- Fast R-CNN과 YOLO를 결합

  → mAP가 3.2% 올라서 75.0%가 됨.
  → Fast R-CNN과 다른 모델 앙상블했으나 mAP 향상은 0.3%, 0.6%로 미미
  → Fast R-CNN과 YOLO를 결합한 모델은 YOLO에 비해 느림. Fast R-CNN과 YOLO를 독립적으로 돌려 결과를 앙상블 하는 방식이기 때문. 그래도 YOLO가 워낙 빨라서 Fast R-CNN을 단독으로 돌리는 것과 앙상블 모델을 돌리는 것의 속도는 거의 유사함. 따라서, Fast R-CNN을 사용하는 것보다는 Fast R-CNN과 YOLO를 결합한 모델을 사용하는 것이 더 나음.

4.4. VOC 2012 Results

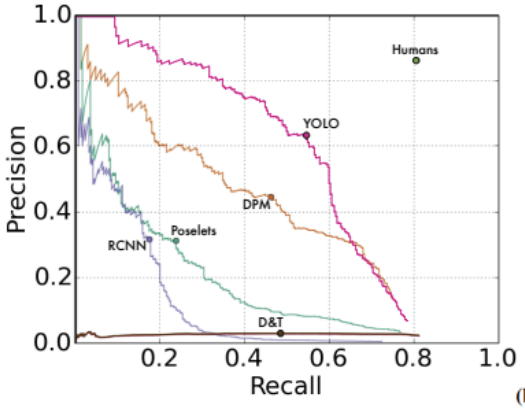
- 파스칼 VOC 2012 데이터 셋에서 YOLO는 57.9%의 mAP를 달성했음. VGG-16을 사용한 R-CNN의 mAP와 비슷
- 속도 측면에서는 YOLO가 빠르고, 정확도 측면에서는 Fast R-CNN과 YOLO를 결합한 모델이 가장 좋음.

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [28]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [33]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

Table 3: PASCAL VOC 2012 Leaderboard. YOLO compared with the full `comp4` (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

4.5. Generalizability: Person Detection in Artwork

- 객체 검출 연구를 위해 사용하는 데이터 셋 : 훈련 데이터 셋과 테스트 데이터 셋이 동일한 분포.
- 실제 이미지 데이터 : 훈련 데이터 셋과 테스트 데이터 셋의 분포가 다를 수 있음
- YOLO 연구진은 훈련 데이터 셋과 다른 분포는 지닌 테스트 데이터 셋(=훈련 데이터 셋에서 보지 못한 새로운 데이터 셋)을 활용하여 테스트했음. → 피카소 데이터 셋과 일반 예술 작품을 사용



(a) Picasso Dataset precision-recall curves.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

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

- 파스칼 VOC 2007에서 학습한 YOLO, R-CNN, DPM 등의 성능을 서로 비교
- R-CNN은 VOC 2007(훈련 데이터셋)에서는 높은 정확도를 보이지만 예술작품(훈련 데이터셋 분포와 비슷하지 않은 테스트 데이터셋)에 대해서는 굉장히 낮은 정확도를 보

임

- DPM은 예술 작품에 대해서도 정확도가 크게 떨어지지는 않았음. 다만 애초에 VOC 2007에서의 정확도도 그리 높은 편은 아님.
- 반면 YOLO는 VOC 2007에서도 가장 높은 정확도를 보였고, 예술 작품에 대해서도 정확도가 크게 떨어지지 않았음.

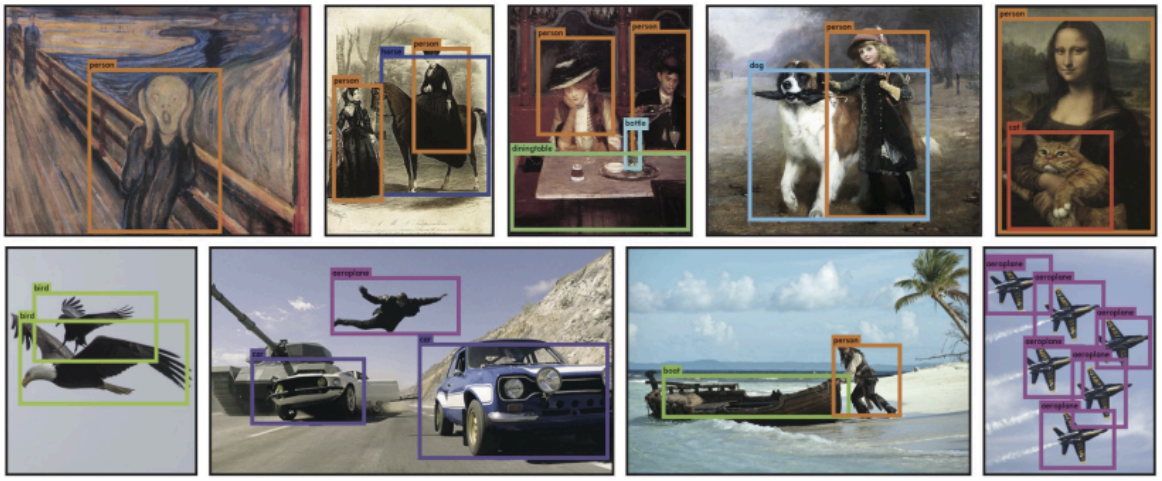


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

Real-Time Detection In The Wild

- YOLO를 웹캠에 연결하고, 카메라에서 이미지를 가져와서 real-time 성능이 유지되는지 확인함

  → YOLO는 이미지를 개별적으로 처리하지만, 웹캠에 연결되었을 때는 마치 추적 시스템처럼 작동해, 움직이거나 모습이 바뀌는 객체를 탐지하였음

5. 결론 (배운점)

💡 연구의 의의 및 한계점, 본인이 생각하는 좋았던/아쉬웠던 점 (배운점)

[의의]

- YOLO라는 객체 탐지를 위한 통합 모델을 제안
  : 구조가 간단하고 전체 이미지를 기반으로 직접 학습할 수 있음
- 기존의 classifier-based 접근 방식과 달리, detection 성능과 직접적으로 연결된 손실 함수를 기반으로 학습되며, 전체 모델이 함께 학습됨

- Fast YOLO는 지금까지 나온 general-purpose object detector 중 가장 빠르고, YOLO는 real-time 객체 탐지 분야에서 SOTA 달성. 또한 YOLO는 새로운 도메인에도 잘 일반화되어, 빠르고 견고한 객체 탐지가 중요한 응용 분야에 적합함.