1. You Only Look Once: Unified, Real-Time Object Detection 논문 리뷰

1. Abstract

- **YOLO**: object detection에 사용
  - 기존 객체 탐지 모델: 주로 classifiers 이용
  - YOLO는 regression problem으로 변환하여 **단일 신경망**이 이미지의 bounding box와 class probability 직접 예측

  ⇒ 기본 모델: 초당 45 프레임 처리 가능

  ⇒ Fast YOLO: 초당 155 프레임 처리 가능 & 다른 실시간 탐지기보다 두 배 높은 mAP 달성

💡 **mAP이란?**

- **mean average precision** (평균 정확도의 평균)로 Object Detection에서 모델의 성능을 평가하는 지표

⇒ 여러 클래스가 있을 때, 각 클래스의 AP 값을 평균낸 것

⇒ 모델이 여러 클래스에 대해 얼마나 잘 동작하는지를 평가

- **Precision**: 모델이 True로 예측한 것들 중에서 실제 True인 비율

⇒ 모델이 검출한 object 중 ground truth object의 비율

- **Recall:** 실제로 True인 것들 중에서 모델이 True로 예측한 것의 비율
- **IoU:** Intersection over Union(전체 영역 중 겹치는 영역 정도)

⇒ 모델이 검출한 bounding box가 ground truth box와 얼마만큼 겹치는지를 나타내는 지표

- **PR Curve:** IoU에 대한 threshold 값이 변화함에 따른 Precision과 Recall의 변화량을 나타낸 그래프
- **AP:** PR Curve의 아래 영역

- YOLO는 최신 detection 시스템과 비교했을 때, 위치 예측 오류는 더 많지만, background에서 잘못된 탐지를 할 확률은 적음
  - **Localization**: 객체의 정확한 위치를 예측하는 능력을 의미
  - **False Positives**: 실제로 객체가 아닌 부분을 객체로 잘못 탐지하는 오류

- 자연 이미지, 예술 작품과 같은 다른 도메인에서도 DPM이나 R-CNN 같은 다른 탐지 방법보다 더 좋은 성능을 보임
  - YOLO는 이미지를 전체적으로 보고 각 객체를 global context 기반으로 탐지하지만, R-CNN은 이미지의 작은 부분을 처리하는 방식으로 객체를 탐지

2. Introduction

- YOLO: 인간의 시각 시스템처럼 이미지를 한 번에 보고, 어떤 객체가 있는지, 어디에 있는지, 그리고 그 객체들이 어떻게 상호 작용하는지를 즉각적으로 알 수 있는 빠르고 정확

한 알고리즘

- **기존 객체 탐지 모델들의 문제점**
  - 분류기를 이용: 복잡하고 비효율적

  ⇒ 객체를 탐지하려면 이미지에서 잠재적인 bounding box를 생성한 후, 각 box에 대해 classifier를 적용하고, 그 결과에 따라 box를 정제하는 등의 후처리 과정이 필요

- **YOLO 차별점**

  1. **빠른 속도**: 복잡한 파이프라인을 거치지 않기 때문.

     → YOLO의 기본 네트워크는 초당 45 프레임을 처리하며, 빠른 버전은 초당 150 프레임 이상을 처리.

  2. **globally 정보 처리:**
     이미지 전체를 한 번에 보고 예측하기 때문에, 객체의 클래스와 모양뿐만 아니라 **문맥 정보**도 함께 인식 기능

     → Fast R-CNN과 같은 방법은 이미지의 배경 패치를 객체로 오인 가능

  3. **일반화 능력:** 새로운 입력이나 예상치 못한 도메인에서도 안정적인 성능 유지 가능

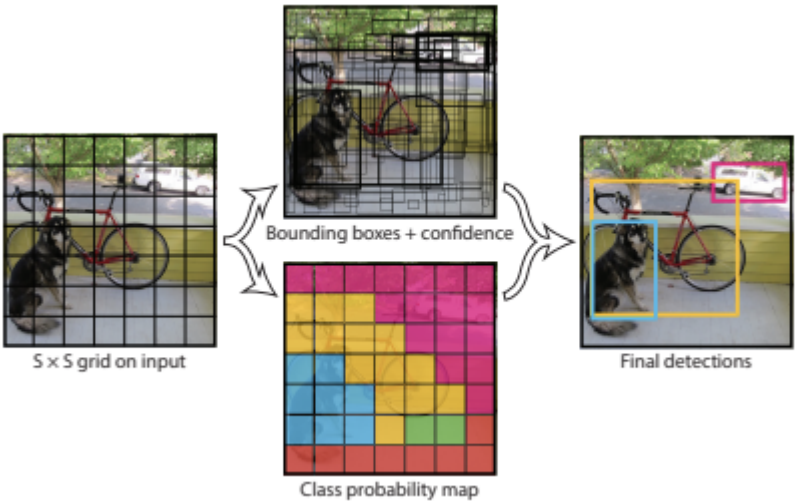
  → 자연 이미지를 학습한 후에도 예술작품과 같은 새로운 도메인에서도 다른 탐지 방법들보다 더 좋은 성능을 발휘

- **YOLO의 한계:** 작은 객체의 위치를 정확하게 예측하기 어려움

⇒ YOLO는 객체 탐지에서 속도와 효율성을 극대화한 모델

3. Unified Detection

- YOLO는 객체 탐지를 회귀 문제로 재구성하여, 이미지를 여러 그리드로 나누고 각 그리드 셀에서 **bounding box**와 **객체가 있을 확률**을 예측



1. **이미지 분할 및 그리드 생성**

- 입력 이미지가 S×S 크기의 그리드로 나누어짐.
- EX) YOLO가 PASCAL VOC 데이터셋에서 사용할 때 S=7, 즉 7x7 크기의 그리드로 이미지를 나누게 됨 & 각 그리드 셀이 해당 구역에 있는 객체를 탐지하게 됩니다.

2. **Bounding Box 예측**

- 각 그리드 셀은 객체의 경계 상자와 관련된 정보들을 예측
- 각 경계 상자는 5개의 값을 예측하는데, 여기에는 **중심 좌표(x, y), 너비(w), 높이(h)**, 그리고 그 상자가 정확히 맞는지를 나타내는 **confidence score (신뢰도)**가 포함됨.
- 각 그리드 셀은 B개의 **Bounding Box**를 예측하며, PASCAL VOC에서 B=2

  ⇒ 즉, 각 그리드 셀은 두 개의 경계 상자에 대한 정보를 출력

3. **객체 탐지와 신뢰도 계산**

- **신뢰도(Confidence)**: 예측된 **Bounding Box**가 실제 객체와 얼마나 일치하는지를 나타냄.

  ⇒ 객체가 존재할 확률과 실제 Bounding Box와의 일치 정도 (IOU: Intersection Over Union)를 곱한 값.
- EX) 한 그리드 셀 안에 객체가 있으면, 모델은 이 셀이 객체를 포함하는지 여부에 대한 확신을 $Pr(Object) \times IOU$로 표현

4. **클래스 확률 예측**

- 각 그리드 셀은 그 셀에 객체가 있을 때 **그 객체가 어떤 클래스에 속하는지**에 대한 확률을 예측
- PASCAL VOC 데이터셋에는 20개의 클래스가 있으므로, YOLO는 각 셀에서 20개의 클래스 확률을 출력

5. **최종 예측**

- 최종적으로, 각 그리드 **셀에서 예측된 클래스 확률**과 **Bounding Box의 신뢰도**를 곱하여 해당 상자에 특정 클래스가 있을 확률을 계산
- 이 결과는 객체가 그 상자에 존재할 확률과 Bounding Box가 얼마나 정확한지(신뢰도)를 포함하는 **클래스별 신뢰도**로 표현됨.

3-1. Network Design

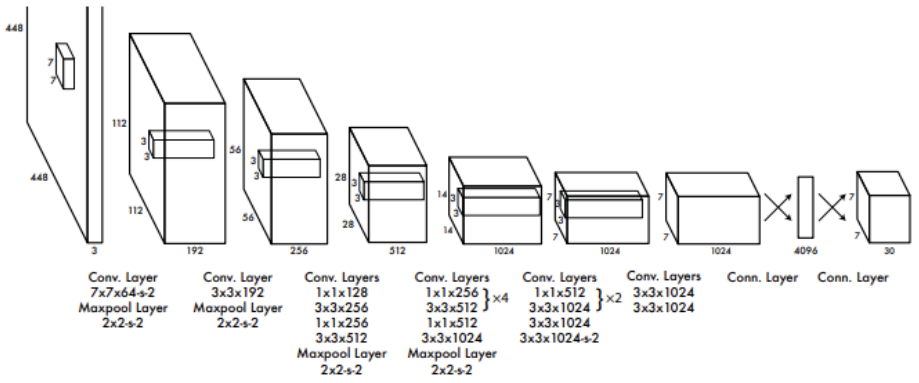


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

- 전체 구조: 24개의 **합성곱(convolutional) 층**과 2개의 **완전 연결(fully connected) 층**으로 구성

⇒ 첫 번째 단계에서는 합성곱 층들이 이미지를 처리하여 특성 맵을 추출하고, 마지막 완전 연결 층들이 객체 탐지 결과(bounding box와 클래스 확률)를 출력

- **GoogLeNet에서 영감을 받은 설계**
  - GoogLeNet의 inception 모듈 대신 더 간단한 1x1 및 3x3 합성곱 층을 사용
  - feature 맵을 효과적으로 축소하면서도 중요한 정보를 유지하는 것이 가능

- **Fast YOLO**
  - 빠른 버전인 Fast YOLO는 성능을 희생하더라도 속도를 높이기 위해 24개가 아닌 9개의 합성곱 층을 사용 ⇒ 네트워크 크기를 줄여 처리 속도를 극대화

3-2. Training

- **사전 훈련 (Pretraining)**
  - YOLO의 **합성곱 층**은 **ImageNet** 데이터셋에서 미리 훈련됨.
  - ImageNet은 1000개의 클래스를 가지고 있으며, YOLO는 처음 20개의 합성곱 층을 훈련한 후, 이 층 위에 **평균 풀링(average pooling)과 완전 연결(fully connected) 층을 추가하여 학습**.
  - 사전 훈련된 모델은 ImageNet 2012 검증 셋에서 88%의 top-5 정확도를 달성했으며, 이는 GoogLeNet과 비슷한 성능.

- **Detection 모델로 변환 (Converting to Detection Model)**
  - 사전 훈련된 네트워크에 **Detection 기능**을 추가.

  ⇒ 4개의 합성곱 층과 2개의 완전 연결 층을 추가해 변환하며, 이 층들은 무작위로 초기화됨.
  - Detection은 보다 세밀한 정보를 필요로 하기 때문에 네트워크의 입력 해상도를 224x224에서 448x448로 증가시킴.

- **최종 출력**
  - 최종 레이어에서는 **클래스 확률**과 **경계 상자(bounding box) 좌표**를 예측. 경계 상자의 너비와 높이는 이미지 크기에 대해 정규화(normalization)하여 0과 1 사이의 값으로 제한됩니다.
  - 경계 상자의 중심 좌표인 x와 y도 각 그리드 셀 내에서 오프셋으로 계산되며, 0과 1 사이로 제한됨

- **활성화 함수 (Activation Function)**

- 최종 출력에는 **linear activation function**를 사용하고, 그 외의 층에는 **leaky ReLU**를 사용
- **Leaky ReLU**는 x>0일 때는 x, x≤0일 때는 0.1x를 적용하는 함수.

- **손실 함수 (Loss Function)**
  - YOLO는 제곱 오차(sum-squared error)를 최적화함.
  - 위치 예측 오류와 분류 오류에 동일한 가중치를 부여하지만, 이는 최적의 방식이 아닐 수 있음.
  - 많은 그리드 셀이 객체를 포함하지 않기 때문에, 이 셀들은 신뢰도(confidence) 점수를 0으로 맞추려는 경향이 있으며, 이로 인해 모델이 불안정해질 수 있음

- **손실 함수 조정 (λcoord and λnoobj)**:
  - **경계 상자 좌표 손실**에 대한 가중치 λcoord=5를 사용하여 손실을 더 크게 하고, 객체가 없는 셀에 대한 **신뢰도 손실**에 대해 λnoobj=0.5로 낮춤.
  - 작은 경계 상자와 큰 경계 상자에 대한 오차를 동일하게 처리하지 않기 위해, **너비와 높이의 제곱근을 예측하는 방식**으로 대체.

- **다중 경계 상자 예측**
  - 각 그리드 셀은 여러 개의 경계 상자를 예측.
  - 학습 중에는 가장 높은 **IOU**를 가진 경계 상자가 객체에 대한 책임을 맡게 됩니다.

  ⇒ 이렇게 하면 특정 예측기가 특정 크기, 비율 또는 클래스의 객체에 대해 더 잘 예측할 수 있도록 전문화**됨**.

- **훈련 설정**
  - YOLO는 PASCAL VOC 2007 및 2012 데이터셋으로 훈련되며, 훈련 시 배치 크기는 64, 모멘텀은 0.9, 가중치 감쇠는 0.0005로 설정.
  - **학습률 스케줄:** 처음에는 낮은 학습률에서 시작해 점차 증가시키고, 75 에포크 동안 10^−2, 그 후 30 에포크 동안 10^−3, 마지막으로 30 에포크 동안 10^−4를 사용.
  - **오버피팅 방지**를 위해 **드롭아웃(dropout)과 데이터 증강(data augmentation)**을 사용

⇒ 드롭아웃 레이어는 0.5의 비율로 첫 번째 완전 연결 층에서 적용되며, 데이터 증강은 이미지 크기의 20%까지 랜덤으로 크기 조정 및 이동을 허용.

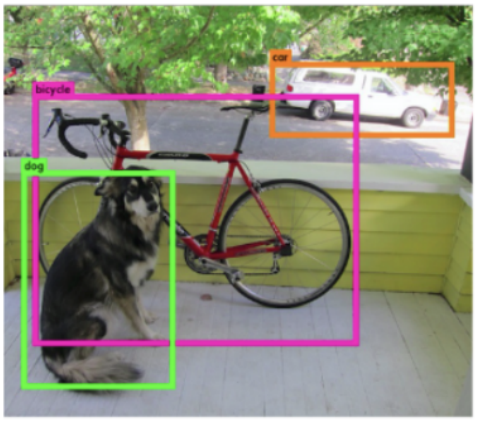
3-3. Inference

- **추론 과정**
  - YOLO는 학습 때와 마찬가지로 **한 번의 네트워크 평가**로 이미지에서 객체를 탐지. PASCAL VOC 데이터셋에서는 이미지당 98개의 경계 상자와 각 상자에 대한 클래스 확률을 예측
  - 한 번의 네트워크 평가만 필요하기 때문에 매우 빠르게 동작하며, 다른 분류 기반 탐지 방법들과 달리 추가적인 반복 평가가 필요 X
- **공간적 다양성**
  - YOLO의 그리드 기반 설계는 경계 상자 예측에서 **공간적 다양성**을 제공합니다.
  - 객체가 어느 그리드 셀에 속하는지 명확하므로 한 그리드 셀에서 하나의 상자만 예측됨
  - 하지만, 큰 객체나 여러 셀에 걸쳐 있는 객체는 여러 셀에서 동시에 잘 탐지될 수 있음.

⇒ 중복된 예측은 **Non-Maximum Suppression (NMS)**을 사용해 제거 가능. YOLO는 R-CNN이나 DPM과 달리 필수적으로 NMS를 사용하지는 않지만, 이를 적용하면 약 2-3%의 성능 향상

💡 **NMS란?**
: object detector가 예측한 bounding box 중에서 정확한 bounding box를 선택하도록 하는 기법

Multiple Bounding Boxes



Final Bounding Boxes

3-4. Limitations of YOLO

- **강한 공간적 제약 (Strong Spatial Constraints)**
  - YOLO는 **그리드 셀**을 기반으로 경계 상자를 예측. 각 그리드 셀은 두 개의 경계 상자를 예측할 수 있으며, 하나의 클래스만 탐지 가능
  - **한 셀당 하나의 클래스만 예측 가능**하다는 제약 때문에, 같은 그리드 셀 내에 여러 개의 객체가 있는 경우 (특히 **작은 객체들이 가까이** 붙어 있는 경우), YOLO는 이런 객체들을 잘 구분하지 못함.
  - Ex) 새떼나 작은 물체들이 그룹으로 모여 있는 경우, 하나의 셀에서 여러 객체를 정확하게 예측하는 것이 어렵기 때문에 YOLO는 이러한 상황에서 성능이 저하됨

- **일반화의 어려움 (Generalization to New or Unusual Aspect Ratios)**
  - YOLO는bounding box를 데이터에서 학습.

  ⇒ 새로운 형태의 객체나 비정상적인 비율을 가진 객체에 대해서는 잘 일반화하지 못하는 경향

- Ex) 길쭉하거나 매우 넓은 물체처럼 **학습 데이터에 자주 나타나지 않는 객체**는 YOLO가 정확하게 예측 X
- YOLO가 **훈련된 데이터에 강하게 의존**하기 때문에, 새로운 상황이나 다른 비율의 객체에 대해 적응하는 데 어려움을 겪게 됨

- **Coarse Features**
  - YOLO는 이미지에서 특징(feature)을 추출하는 여러 **다운샘플링 (downsampling) 층**을 포함.

    ⇒ 다운샘플링 층은 이미지 크기를 줄여서 더 효율적으로 처리할 수 있게 하지만, **세부적인 정보 손실** 가능

    ⇒ YOLO가 작은 객체를 탐지하는 데 어려움을 겪게 하는 요인
  - 작은 물체의 경우, 이러한 Coarse Features을 사용하면 세부적인 경계 상자를 정확하게 예측하기 어렵
  - **손실 함수의 한계 (Limitations of the Loss Function)**
    - YOLO의 손실 함수는 객체 탐지 성능을 대략적으로 반영하는 방식으로 설계되었지만, **큰 경계 상자와 작은 경계 상자에서의 오차를 동일하게 취급**.
    - **큰 경계 상자**에서는 약간의 오류가 IOU(Intersection over Union)에 미치는 영향이 상대적으로 작지만, **작은 경계 상자**에서는 작은 위치 오차가 IOU에 큰 영향을 미침.
    - 작은 객체에 대한 위치 예측이 부정확할 경우, 탐지 성능이 크게 저하될 수 있음.

  - **주요 오류 원인: 잘못된 위치 예측 (Incorrect Localization)**
    - YOLO의 주된 오류: **경계 상자의 위치를 정확하게 예측하지 못하는 경우**에 발생.
    - **위치 예측 오류:** 모델의 성능을 떨어뜨리는 주요 요인 중 하나로, 특히 작은 객체에서 두드러지게 나타남.

4. Comparison to Other Detection Systems

1. **Deformable Parts Models (DPM)**

- **DPM:** 객체 탐지에 **슬라이딩 윈도우(sliding window)** 방식을 사용. 이는 이미지 전체를 작은 창으로 나눠서 각 창에서 객체를 탐지하는 방식.
- **단계적 파이프라인(disjoint pipeline)**을 사용하여, 먼저 고정된 특징을 추출하고, 그 다음에 특정 영역을 분류하며, 고득점 영역에 대해 경계 상자를 예측하는 과정을 거침.
- YOLO는 이러한 개별적인 단계를 없애고, **하나의 통합된 신경망**을 통해 **특징 추출, 경계 상자 예측, 중복 탐지 제거, 맥락적 추론을 동시에** 처리.
- YOLO는 고정된 특징 대신 **학습된 특징**을 사용하여 탐지 성능을 최적화하며, 이로 인해 DPM보다 **더 빠르고 정확한** 모델이 됩니다.

2. **R-CNN**

- **R-CNN**과 그 변형 모델들은 **영역 제안(region proposals)** 방식을 사용.

  ➔ **Selective Search** 알고리즘으로 잠재적인 경계 상자를 생성한 후, 각 상자에서 특징을 추출하고, SVM을 사용해 해당 상자의 점수를 평가하며, 마지막으로 경계 상자를 조정하는 방식
- R-CNN은 각 단계를 개별적으로 최적화해야 하며, 이로 인해 복잡하고 느린 시스템이 됨.

  ⇒ R-CNN은 테스트 시 이미지당 40초 이상 소요됩니다.
- **YOLO**는 R-CNN과 유사하게 **경계 상자를 제안**하지만, 각 **그리드 셀에 공간적 제약을 부여**하여 동일한 객체를 여러 번 탐지하는 것을 방지.
- YOLO는 이미지당 **98개의 경계 상자**만 예측하지만, R-CNN은 **약 2000개의 경계 상자**를 생성.
- YOLO는 이 모든 단계를 하나의 신경망으로 통합하여 더 간단하고 빠른 시스템을 제공.
-

3. **Fast and Faster R-CNN**

- **Fast R-CNN**과 **Faster R-CNN**은 R-CNN의 속도를 개선하기 위한 변형 모델.
  - **Fast R-CNN:** features을 공유하여 연산을 최적화
  - **Faster R-CNN:** 영역 제안 네트워크(Region Proposal Network, RPN)를 사용하여 Selective Search를 대체
  - 이 모델들은 R-CNN보다 빠르고 정확하지만, 여전히 SOTA에는 미치지 못함

- **YOLO**는 복잡한 파이프라인을 없애고 디자인 자체가 빠르도록 만들어짐. YOLO는 실시간 탐지를 목표로 하고 있어, Fast/Faster R-CNN보다 더 빠름.

4. **단일 클래스 탐지기 (Single-Class Detectors)**

- 특정 객체(예: 얼굴, 사람)만을 탐지하는 시스템들은 탐지해야 하는 **변화의 범위**가 적기 때문에 최적화될 수 있음.
    - 하지만 이런 탐지 시스템들은 **단일 클래스**에만 특화되어 있음
- 반면 **YOLO**는 여러 클래스의 객체를 동시에 탐지할 수 있는 **일반적인 탐지기**.

  ⇒ 다양한 객체를 학습하고 탐지할 수 있다는 점에서 더 광범위한 적용이 가능.

5. **Deep MultiBox**

- **MultiBox:** R-CNN과 달리 Selective Search를 사용하지 않고, **신경망**을 통해 관심 영역(Region of Interest)을 예측. 하지만, MultiBox는 **일반 객체 탐지**가 아니라 **단일 객체 탐지**를 수행하며, 여전히 더 큰 탐지 파이프라인의 한 부분일 뿐
- **YOLO**는 경계 상자를 예측하는 전체적인 시스템을 제공하는 반면, MultiBox는 경계 상자만 예측하고 이를 바탕으로 추가적인 이미지 분류 작업이 필요.

6. **OverFeat**

- **OverFeat:** 신경망을 사용하여 **로컬라이제이션(localization)** 작업을 수행하고 이를 탐지 작업으로 적응시킴.
    - 슬라이딩 윈도우 방식을 효율적으로 사용하지만, 여전히 분리된 시스템(disjoint system)
- OverFeat는 **로컬 정보**만을 사용하여 예측하므로, 전역 맥락을 고려하지 X.
- **후처리(post-processing)**가 많이 필요.
- **YOLO**는 이미지 전체를 보고 맥락 정보를 함께 사용하므로, 더 일관된 탐지 수행 가능

7. **MultiGrasp**

- **MultiGrasp:** 그리드 기반 접근 방식을 사용해 **물체를 잡을 수 있는 영역(graspable region)**을 예측하는 시스템입니다. YOLO와 비슷한 방식으로 경계 상자를 예측하지만, MultiGrasp는 **단일 객체**를 처리하며, 물체의 경계나 크기를 예측할 필요 X

- **YOLO**는 **여러 객체에 대해 경계 상자와 클래스 확률을 동시에 예측**해야 하므로 훨씬 더 복잡한 작업을 수행

5. Experiments

1. **다른 실시간 탐지 시스템과의 비교 (Comparison to Other Real-Time Systems)**

- 기존 objection 파이프라인을 빠르게 만드는 데 집중. But, 대부분 실시간으로 작동 X
- **Fast YOLO:** PASCAL VOC 데이터셋에서 가장 빠른 탐지 시스템으로, 초당 155 프레임(FPS)을 처리
  - 기존의 실시간 탐지 시스템은 보통 30FPS 수준
  - 정확도 면에서도 YOLO가 더 뛰어남
  - YOLO는 **52.7% mAP**를 기록하고, 이는 기존 실시간 탐지 방법보다 **2배 이상 정확**한 수치

- VGG-16을 사용한 YOLO: 다른 Objection 시스템과 비교할 때는 유용하지만, 실시간 탐지에 적합하지 X

2. **VOC 2007 오류 분석 (VOC 2007 Error Analysis)**

- YOLO와 Fast R-CNN의 **오류**를 비교하는 실험 진행
- 실험 방법: 테스트 이미지에서 각 카테고리의 상위 N개의 예측을 분석하고, 이를 다섯 가지 유형으로 분류
  - **Correct (정확도)**: 올바른 클래스와 Bounding Box (IOU > 0.5)
  - **Localization (위치 오류)**: 올바른 클래스지만 Bounding Box가 정확하지 않음 (0.1 < IOU < 0.5)
  - **Similar** : 유사한 클래스이지만 다른 클래스 (IOU > 0.1)
  - **Other** : 완전히 다른 클래스 (IOU > 0.1)
  - **Background (배경 오류)**: Bounding Box가 아무 객체도 포함하지 않음 (IOU < 0.1)

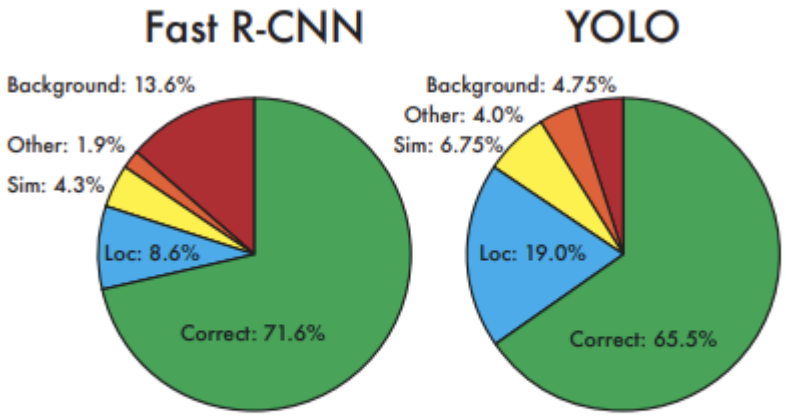


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

- **YOLO는 주로 위치 예측 오류**가 많음(전체 오류 중 19%가 경계 상자의 위치를 잘못 예측한 경우)
- **Fast R-CNN**은 위치 오류는 적지만 **배경 오류**가 많음Fast R-CNN의 전체 오류 중 13.6%가 배경 오류)

3. **Fast R-CNN과 YOLO의 결합 (Combining Fast R-CNN and YOLO)**

- **YOLO**는 배경을 잘못 탐지하는 경우가 적기 때문에, **Fast R-CNN**과 결합했을 때 성능이 크게 향상됨
    - YOLO는 Fast R-CNN이 탐지한 경계 상자 중 **배경 오류를 줄이는 역할**
- 실험 결과: Fast R-CNN의 **mAP**가 원래 71.8%였지만, YOLO와 결합하자 75.0%로 **3.2%** 상승
  ⇒ 단순한 모델의 앙상블 효과가 아니라, YOLO가 **다른 종류의 오류**를 보완해주기 때문
- But, 결합이 YOLO의 빠른 속도를 활용하지는 X
    - 두 모델을 각각 실행한 후 결과를 결합해야 하기 때문에, 속도 면에서는 개선 X
- YOLO는 매우 빠르기 때문에 Fast R-CNN과 결합하더라도 큰 속도 손실이 발생 X

4. **VOC 2012 결과 (VOC 2012 Results)**

- **VOC 2012** 테스트 결과에서, YOLO는 **57.9% mAP**를 기록

- YOLO는 특히 **작은 객체**를 탐지하는 데 약점이 있음
  - 병(bottle), 양(sheep), TV 모니터 같은 카테고리에서는 YOLO가 R-CNN이나 다른 시스템보다 **8-10%** 정도 낮은 성능
  - 작은 객체가 그리드 셀에 잘 맞지 않기 때문에 위치 예측에서 어려움을 겪는 경우가 많기 때문
- 고양이(cat)나 기차(train)와 같은 카테고리에서는 YOLO가 더 높은 성능을 기록

5. **일반화 성능: 예술작품에서의 인물 탐지 (Generalizability: Person Detection in Artwork)**

- 학술 데이터셋은 보통 **훈련 데이터**와 **테스트 데이터**가 같은 분포를 따르지만, 실제 응용에서는 X
  - Ex) 훈련 데이터: 자연 이미지, 테스트 데이터: 예술작품

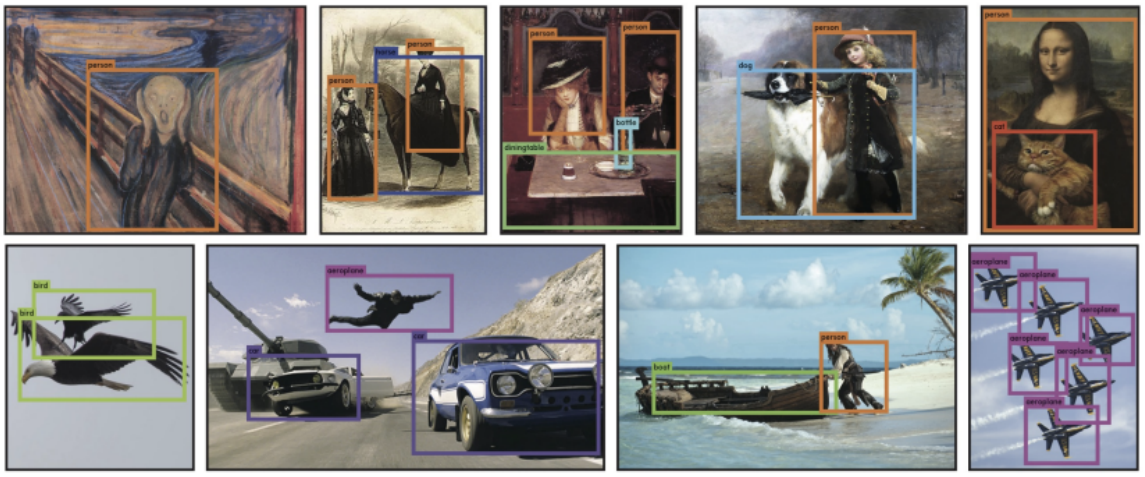


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

- 예술작품에서의 인물 탐지 실험에서, YOLO는 다른 시스템보다 **일반화 성능 뛰어남**
- **R-CNN**은 자연 이미지에 특화된 **Selective Search** 방식을 사용하기 때문에, 예술작품에서는 성능이 크게 저하
- DPM(Deformable Parts Model은 성능 저하가 덜함

  ⇒ DPM이 객체의 **크기와 형태에 대한 공간적 모델**을 강하게 사용하기 때문
- YOLO도 DPM처럼 객체의 크기, 형태, 그리고 객체 간의 **관계**를 모델링
  ⇒ 예술작품과 자연 이미지가

픽셀 수준에서는 다르지만, 객체의 크기와 모양이 유사하기 때문에 좋은 성능을 보이는 것

6. Real-Time Detection In The Wild

- YOLO는 **빠르고 정확한 객체 탐지기**이기 때문에, 실시간으로 이미지나 비디오에서 객체를 탐지해야 하는 컴퓨터 비전 응용 프로그램에 적합
- YOLO를 웹캠에 연결하여 실시간 성능을 확인
  - 웹캠에서 이미지를 가져오는 시간과 탐지 결과를 표시하는 시간을 포함하더라도 실시간 성능을 유지
  - 이미지를 하나씩 처리하지만, 웹캠에 연결하면 **추적 시스템(tracking system)**처럼 동작하여, 객체가 움직이거나 모양이 변할 때도 실시간으로 탐지
  - **상호작용적**이고 **참여감**을 높이는 방식으로 동작

7. Conclusion

- YOLO는 **단일 통합 모델(unified model)**로 객체 탐지를 수행하며, 전체 이미지에서 직접 학습 가능
- 기존의 분류기 기반 접근 방식과는 달리, YOLO는 **탐지 성능과 직접적으로 관련된 손실 함수**로 훈련됨
  - 모델이 Bounding Box를 예측하고 분류하는 모든 작업이 하나의 손실 함수로 통합되어 최적화
- **Fast YOLO**는 현재 가장 빠른 **일반 목적 객체 탐지기**
- YOLO는 **실시간 객체 탐지**에서 최신 성능을 제공하며, 여러 도메인에서 **잘 일반화**할 수 있어 다양한 응용 프로그램에 적합
- 새로운 영역(new domains)에서도 잘 작동하기 때문에, 빠르고 견고한 객체 탐지가 필요한 응용 프로그램에 이상적임