

YOLO

You Only Look Once: Unified, Real-Time Object Detection

< 한 번만 보면 돼: 통합된 실시간 객체 탐지 >

2024.08.09(금)

YOLO 논문 발표

Contents

| | |
|----|--------|
| 1. | 소개 |
| 2. | 모델 |
| 3. | 비교 |
| 4. | 실험 |
| 5. | 실시간 검출 |
| 6. | 결론 |
| 7. | 실습 |

01. Introduction (소개)



1. 새로운 접근법: 객체 검출을 회귀 문제로 프레임화, 단일 신경망 사용

2. 단일 평가: 전체 이미지를 한 번 평가 -> 경계 상자와 클래스 확률 예측



3. 빠른 처리 속도: 실시간으로 45프레임/초, Fast YOLO는 155프레임/초 처리

4. 성능: Fast YOLO는 다른 실시간 검출기보다 두 배의 mAP를 달성

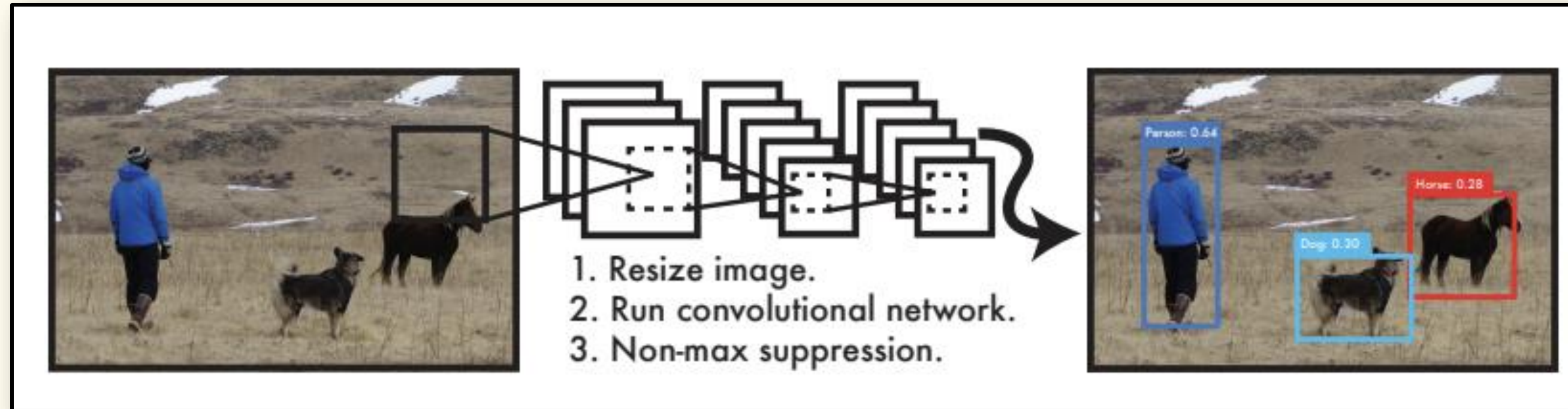
* mAP (mean Average Precision) : 다양한 객체를 얼마나 정확하게 검출하는지를 종합적으로 평가하는 지표



5. 오류 및 오탐지: 위치 오류가 더 많지만 배경에 대한 오탐지가 적음

6. 일반화 능력: 다양한 도메인에서 다른 검출 방법보다 더 좋은 성능

[그림 1 : YOLO 탐지 시스템]



1. 이미지
조정

2. 합성곱
신경망

3. 비최대
억제

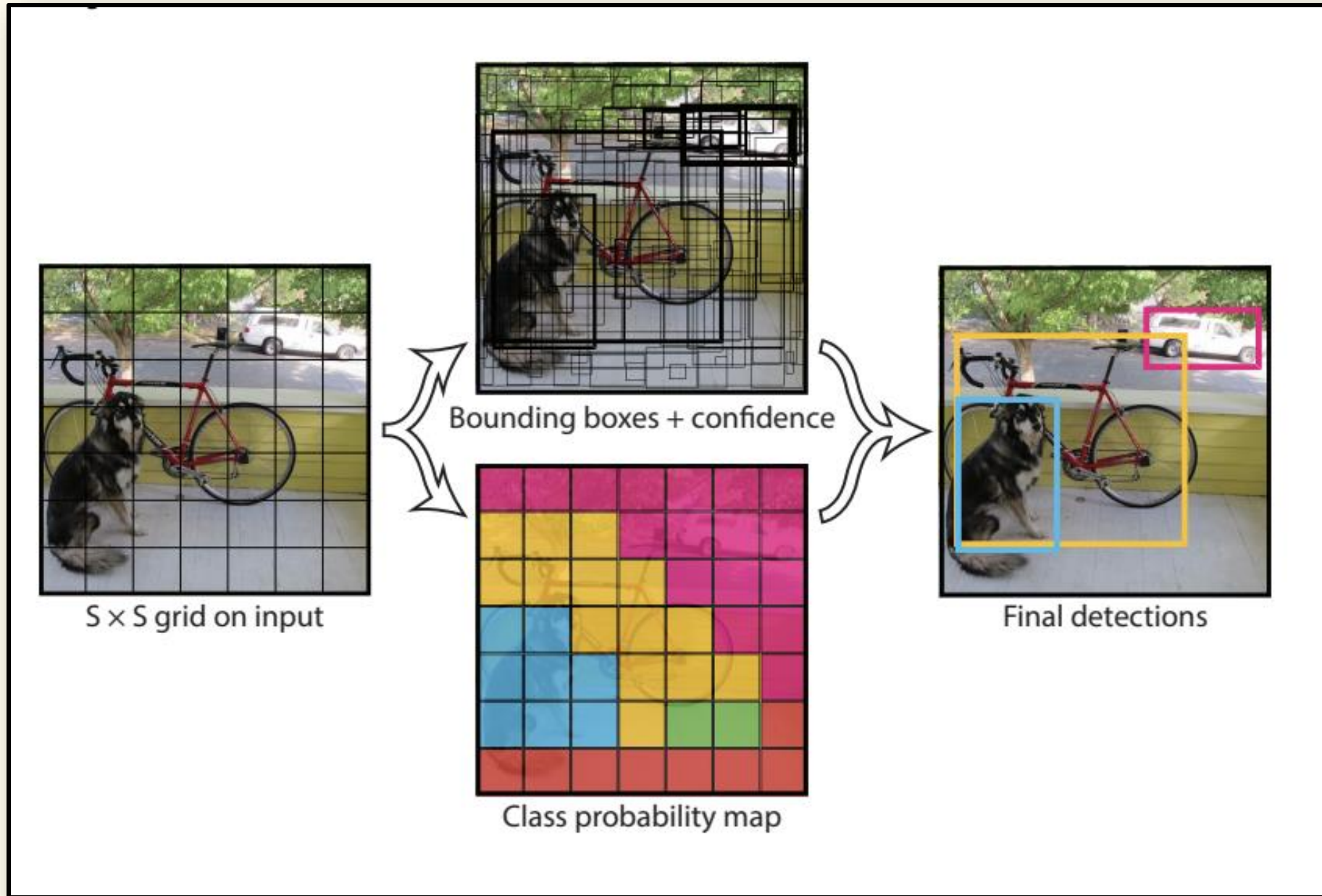
입력 이미지
 448×448 픽셀
크기로 조정

이미지 내 객체
탐지 각 객체의
위치와 클래스
예측
(사람, 개, 말 등)

가장 신뢰도가
높은 예측만을
남기고 겹치는
예측은 제거

02. Unified Detection (통합된 검출)

[그림 2 : 모델]



1. $S \times S$ 그리드

- 입력 이미지를 $S \times S$ 그리드로 나눔

2. 경계 상자 + 신뢰도

- 그리드 셀에서 B개의 경계 상자를 예측
- 각 경계 상자는 그 상자의 위치와 크기 그리고 그 상자가 객체를 포함하고 있을 확률(신뢰도)

3. 클래스 확률 맵

- 각 그리드 셀에서 C개의 클래스에 대한 확률을 예측
- 해당 그리드 셀에 특정 클래스의 객체가 있을

4. 최종 탐지 결과

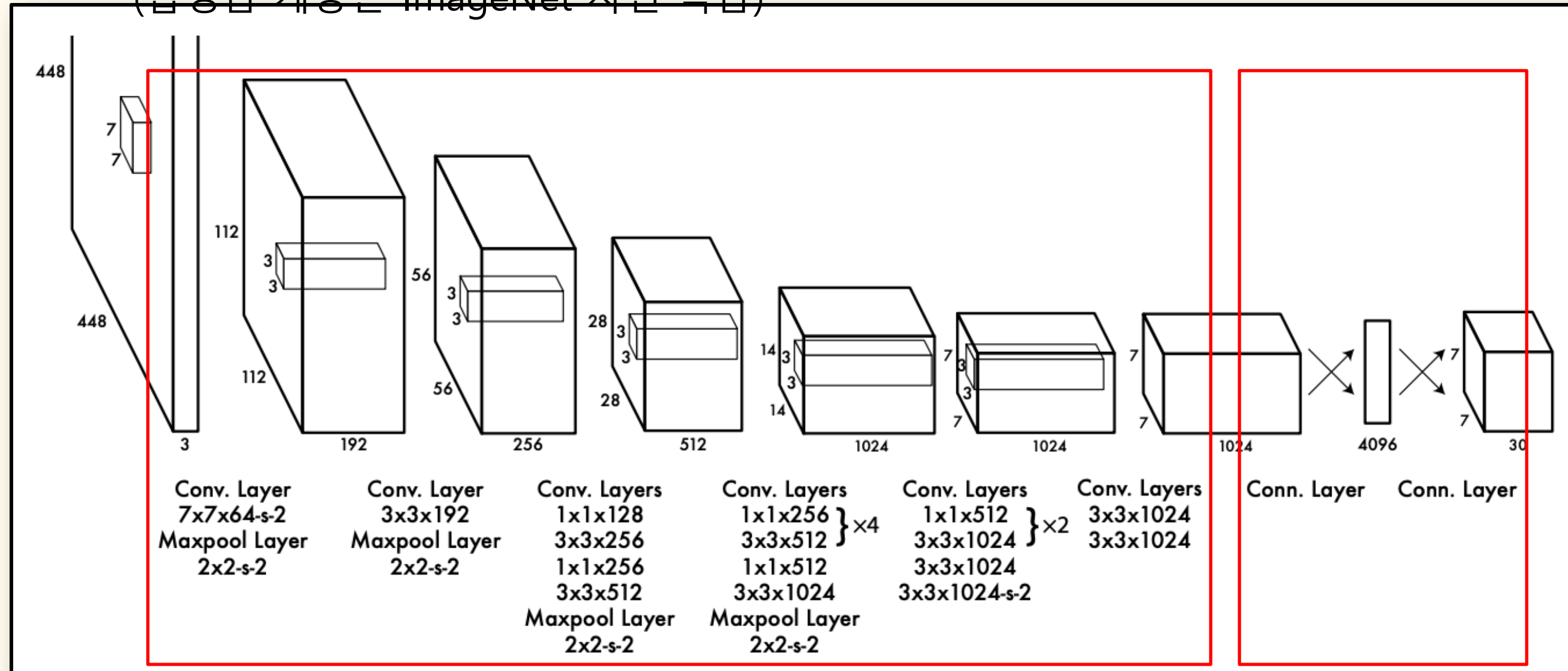
- 예측된 경계 상자 + 클래스 확률 -> 최종 객체 탐지 결과 생성
- 객체들의 위치와 클래스를 나타내는 경계 상자로 시각화

2-1. Network Design (네트워크

설계)

[그림 3 :
아키텍처]

총 26개의 계층 중 24개는 합성곱 계층, 나머지 2개는 완전 연결 계층
(합성곱 계층은 ImageNet 사전 학습)



이미지에서 특징(feature) 추출

추출된 특징을 바탕으로 최종
예측

2-2. Training (훈련)

1. 훈련 과정:

- PASCAL VOC 2007 및 2012 데이터 세트로 약 135 에포크 동안 네트워크를 훈련함.
- 2012년 테스트 시 2007년 VOC 테스트 데이터도 포함함.

2. 훈련 설정:

- 배치 크기: 64
- 모멘텀: 0.9
- 가중치 감소율: 0.0005

3. 학습률:

- 첫 에포크 동안 학습률을 10^{-3} 에서 10^{-2} 로 증가.
- 75 에포크 동안 10^{-2} 로 학습.
- 30 에포크 동안 10^{-3} 로 학습.
- 마지막 30 에포크 동안 10^{-4} 로 학습.

4. 과적합 방지:

- 드롭아웃과 광범위한 데이터 증강 사용.
- 드롭아웃 비율: 0.5
- 데이터 증강: 원본 이미지 크기의 최대 20%까지 무작위 스케일링 및 변환.
- 노출과 채도 최대 1.5배까지 무작위 조정.

2-3. Limitations of YOLO (YOLO의 한계점)

1. 강한 공간적 제약:

- 각 그리드 셀이 두 개의 박스만 예측하고 하나의 클래스만 가질 수 있음
-> 인접한 객체의 수를 제한
- 새 떼와 같은 그룹으로 나타나는 작은 객체를 다루는 데 어려움이 있음

2. 일반화 문제:

- 바운딩 박스를 예측하도록 학습 -> 새로운 또는 비정상적인 구성의 객체에 일반화하는 데 어려움이 있음

3. 거친 특징(coarse features) 사용:

- 입력 이미지에서 여러 다운샘플링 레이어를 사용 -> 상대적으로 거친 특징을 사용하여 바운딩 박스를 예측함

4. 손실 함수의 한계:

- 작은 바운딩 박스와 큰 바운딩 박스에서의 오류를 동일하게 취급함
- 큰 박스에서 작은 오류는 무해하지만 작은 박스에서 작은 오류는 IOU에 큰 영향을 미침
- 주요 오류 원인은 잘못된 위치 설정 (incorrect localizations)

* IOU (Intersection Over union) : 두 영역의 겹치는 부분(Intersection) / 전체 영역(Union), 객체 검출 성능 지표

3. Comparison to Other Detection Systems (다른 검출 시스템과의 비교)

1. DPM

Deformable Parts Models

- 정적인 특징 추출, 영역 분류, 바운딩 박스 예측 등을 **분리된 파이프라인**으로 처리
- **슬라이딩 윈도우** 접근 방식 사용
- * 슬라이딩 윈도우: 이미지 전체를 탐색하며 객체가 위치할 가능성이 있는 영역을 찾는 방식

2. R-CNN

- **영역 제안**을 사용하여 이미지에서 객체를 찾음
- Selective Search, 컨볼루션 네트워크, SVM, 선형 모델, 비최대 억제로 구성된 **복잡한 파이프라인**

YOLO

하나의 컨볼루션 신경망으로

대체

-> 더 빠르고 정확한 모델 제공

- 그리드 셀이 **잠재적 바운딩 박스** 제안
- **공간적 제약**을 두어 동일 객체의 여러 탐지 완화

- **적은 바운딩 박스**를 제안
- 단일 최적화된 모델로 결합.

VS

VS

3. Comparison to Other Detection Systems (다른 검출 시스템과의 비교)

3. 기타 빠른 검출기

Fast 및 Faster R-CNN은 R-CNN 프레임워크를
가속화하지만 여전히 **실시간 성능**에는 미치지
못함

VS

파이프라인을 없애고
설계 자체가 빠르도록 만듦

YOLO

4. Deep MultiBox

- 관심 영역을 예측하기 위해 컨볼루션 신경망을
사용
- 단일 객체 탐지 가능, 일반적인 객체 탐지는
불가능

VS

MultiBox와 YOLO
모두 CNN으로 바운딩 박스를 예측하지만
YOLO는 **완전한 객체 탐지 시스템**

MultiBox와 YOLO

3. Comparison to Other Detection Systems (다른 검출 시스템과의 비교)

5. OverFeat

슬라이딩 윈도우 접근 방식
더 높은 정확도 but 시간과 비용이 큼
전반적인 맥락을 고려하지 못해 후처리
필요

YOLO

단일 신경망을 통해 모든 객체 검출
접근 방식은 빠르지만 OverFeat처럼
작은 객체나 근접한 객체에 대한 정확도는
떨어짐

VS

6. MultiGrasp

이미지 내에 하나의 물체가 있는 경우
그 물체를 잡을 수 있는 영역을 하나만 예측
물체의 크기, 위치, 경계, 종류 등을 추정할 필요가
없음

이미지 내에 여러 개의 물체가 있는 경우
각각의 물체에 대한 경계 상자와 클래스 확률을
예측해야 하는 훨씬 복잡한 작업

VS

4. Experiments (실험)

1. PASCAL VOC 2007 비교: (20개의 서로 다른 객체 클래스로 이루어진 9,963개의 이미지로 구성된 데이터셋)

- YOLO를 다른 실시간 검출 시스템과 비교.
- YOLO와 Fast R-CNN의 오류를 분석하여 차이를 이해.

2. 오류 프로파일 분석:

- YOLO를 사용하여 Fast R-CNN의 배경에서 발생하는 잘못된 긍정 오류를 줄임으로써 성능 향상.

3. VOC 2012 결과:

- VOC 2012에서의 결과를 제시하고 현재 최첨단 방법과 mAP를 비교.

4. 새로운 도메인 일반화:

- YOLO가 다른 검출기보다 새로운 도메인에 더 잘 일반화됨을 두 가지 예술작품 데이터셋에서 보여줌.

4.1 Comparison to Other Real-Time Systems
(다른 실시간 시스템과의 비교)

| Real-Time Detectors | 데이터셋 | 정확도 | 속도 |
|-------------------------|-----------|------|-----|
| | 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 |

일반적으로 30 FPS 이상의 속도는
실시간 응용프로그램에서 원하는 성능을 제공
45 FPS는 영상 스트리밍 및 실시간 비디오 분석에
충분

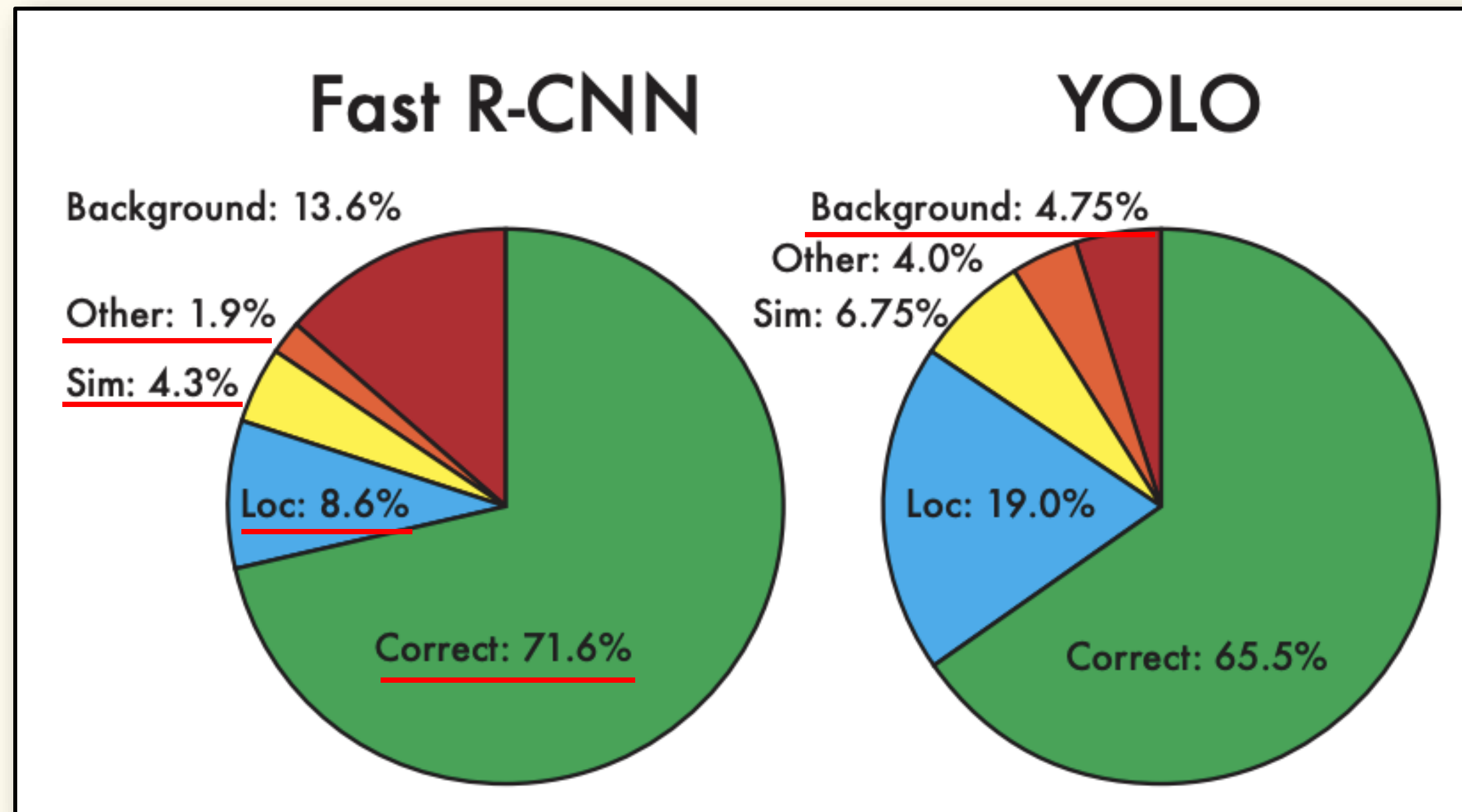
→ YOLO가 다른 모델들에 비해
얼마나 빠르고 정확한지 강조

YOLO는 실시간 성능을
유지하면서
높은 정확도를 제공

4.2 VOC 2007 Error Analysis

(VOC 2007 오류 분석)

(Fast R-CNN: PASCAL VOC 데이터셋에서 가장 높은 성능을 보이는 검출 모델)

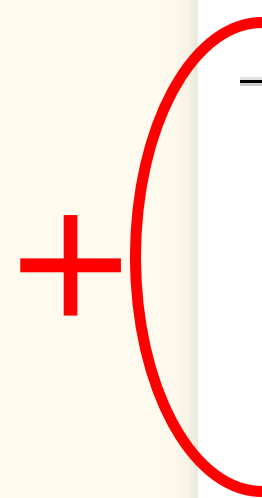


- . Correct (정확한 탐지)
- . Loc (위치 오류)
- . Sim (유사한 객체 오류)
- . Other (기타 오류)
- . Background (배경 오류)

- Fast R-CNN은 YOLO보다 더 높은 정확도를 가지고 있으며, 위치 오류와 기타 오류에서 더 나은 성능
- 반면 YOLO는 배경 오류가 적으며, 실시간 탐지 시스템에서 중요한 측면인 빠른 속도
- YOLO의 더 높은 위치 오류와 기타 오류는 개선 필요, 배경 오류가 적다는 점이 강점

4.3 Combining Fast R-CNN and YOLO

(Fast R-CNN과 YOLO 결합)



| | 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 ↑ |

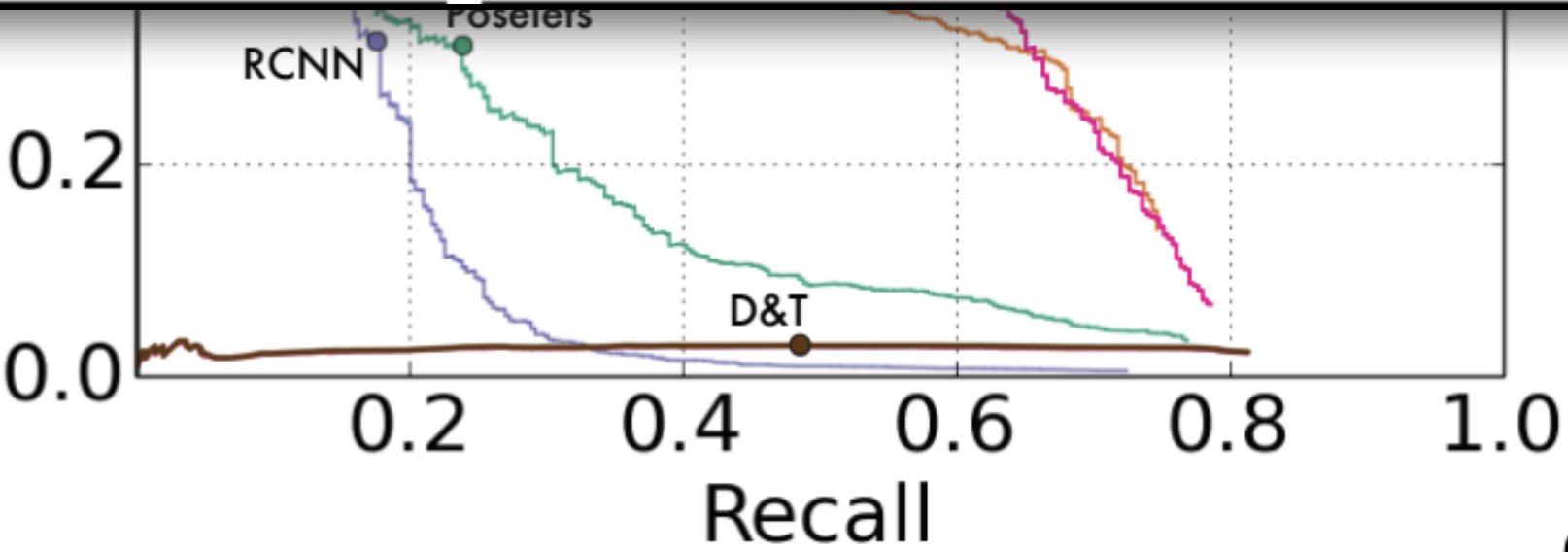
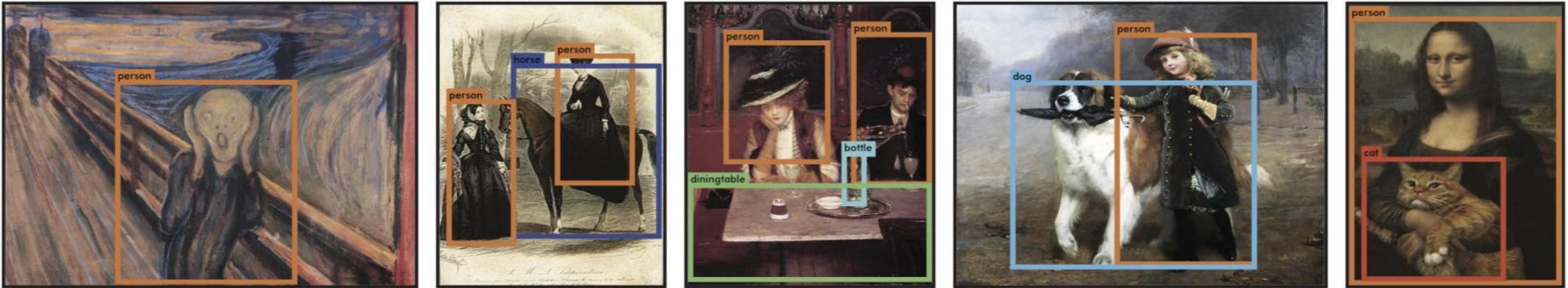
Fast R-CNN의 검출 결과 중
13.6%가 실제로는 객체가 없는 배경을
포함
YOLO보다 약 3배 더 자주
배경을 객체로 잘못 예측

- YOLO를 사용해 Fast R-CNN의 배경 검출을 제거하여 성능 향상
- 최고의 Fast R-CNN 모델은 VOC 2007에서 71.8% mAP를 달성
- YOLO와 결합 시 mAP가 3.2% 증가하여 75.0% 달성
- 다른 Fast R-CNN 버전과 결합 시 mAP 증가폭은 0.3%에서 0.6%에 불과

YOLO의 빠른 속도를
충분히 활용하지 못함 but
YOLO의 빠른 속도 덕분에 전체
시스템의 계산 시간에 큰 영향을
미치지 않는

4.4 Generalizability: Person Detection in Artwork
(일반화 가능성: 예술 작품에서의 사람 검출)

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



(a) Picasso Dataset precision-recall curves.

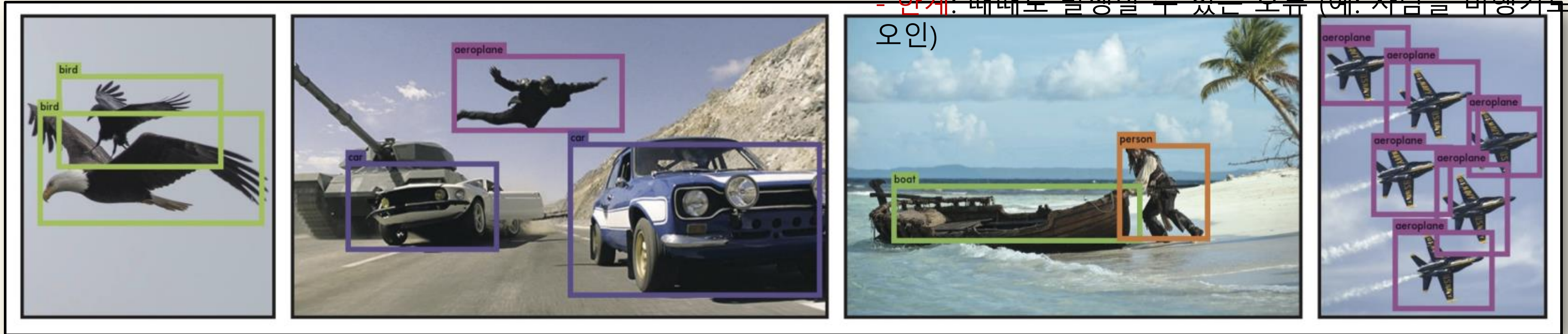
| | VOC 2007 AP | Picasso AP | Picasso Best F_1 | People-Art AP |
|--------------|----------------|---------------|-----------------------|------------------|
| YOLO | 59.2 | 53.3 | 0.590 | 45 |
| R-CNN | 54.2 | 10.4 | 0.226 | 26 |
| DPM | 43.2 | 37.8 | 0.458 | 32 |
| Poselets [2] | 36.5 | 17.8 | 0.271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

05. Real-Time Detection In The Wild

(실제 환경에서의 실시간 검출)

- **강점**: 다양한 이미지 유형에서 대체로 정확한 객체 탐지 능력

- **한계**: 때때로 발생할 수 있는 오류 (예: 사람을 비행기로 오인)



1. YOLO의 실시간 성능:

- YOLO는 빠르고 정확한 객체 검출기로, **웹캠과 연결하여** 실시간 성능을 유지함

2. 인터랙티브 시스템:

- YOLO는 개별 이미지를 처리하면서도 웹캠에 연결되면 **객체의 이동과 변화하는 모습을 추적**할 수 있는 시스템처럼 작동함

3. 데모 및 소스 코드:

- 시스템의 데모와 소스 코드는 프로젝트 웹사이트(<http://pjreddie.com/yolo/>)에서 제공

06. Conclusion (결론)

YOLO 소개

- YOLO라는 통합 객체 검출 모델은 **단순하게 구성**할 수 있으며, **전체 이미지를 대상으로 직접 학습**할 수 있음

YOLO와 기존 방식의 차이점

- YOLO는 분류기 기반 접근법과 달리, **객체 검출 성능에 직접 대응**하는 손실 함수를 사용하여 학습
 - 모델 전체가 **통합적으로 학습**

Fast YOLO의 성능

- Fast YOLO는 문헌에 등장한 **가장 빠른 범용 객체 검출기**
- YOLO는 실시간 객체 검출에서 **최첨단 성능**을 자랑합니다.

YOLO의 일반화 능력

- YOLO는 **새로운 도메인**에 잘 일반화됨
- **빠르고 견고한 객체 검출**이 필요한 응용 프로그램에 이상적

07. 실습

Time: 1h

Epoch [1/10], Loss: 37578555.9922
Epoch [2/10], Loss: 1530.6875
Epoch [3/10], Loss: 428.8274
Epoch [4/10], Loss: 141.3663
Epoch [5/10], Loss: 86.1959
Epoch [6/10], Loss: 70.8615
Epoch [7/10], Loss: 53.3619
Epoch [8/10], Loss: 16.4923
Epoch [9/10], Loss: 17.9992
Epoch [10/10], Loss: 7.2352

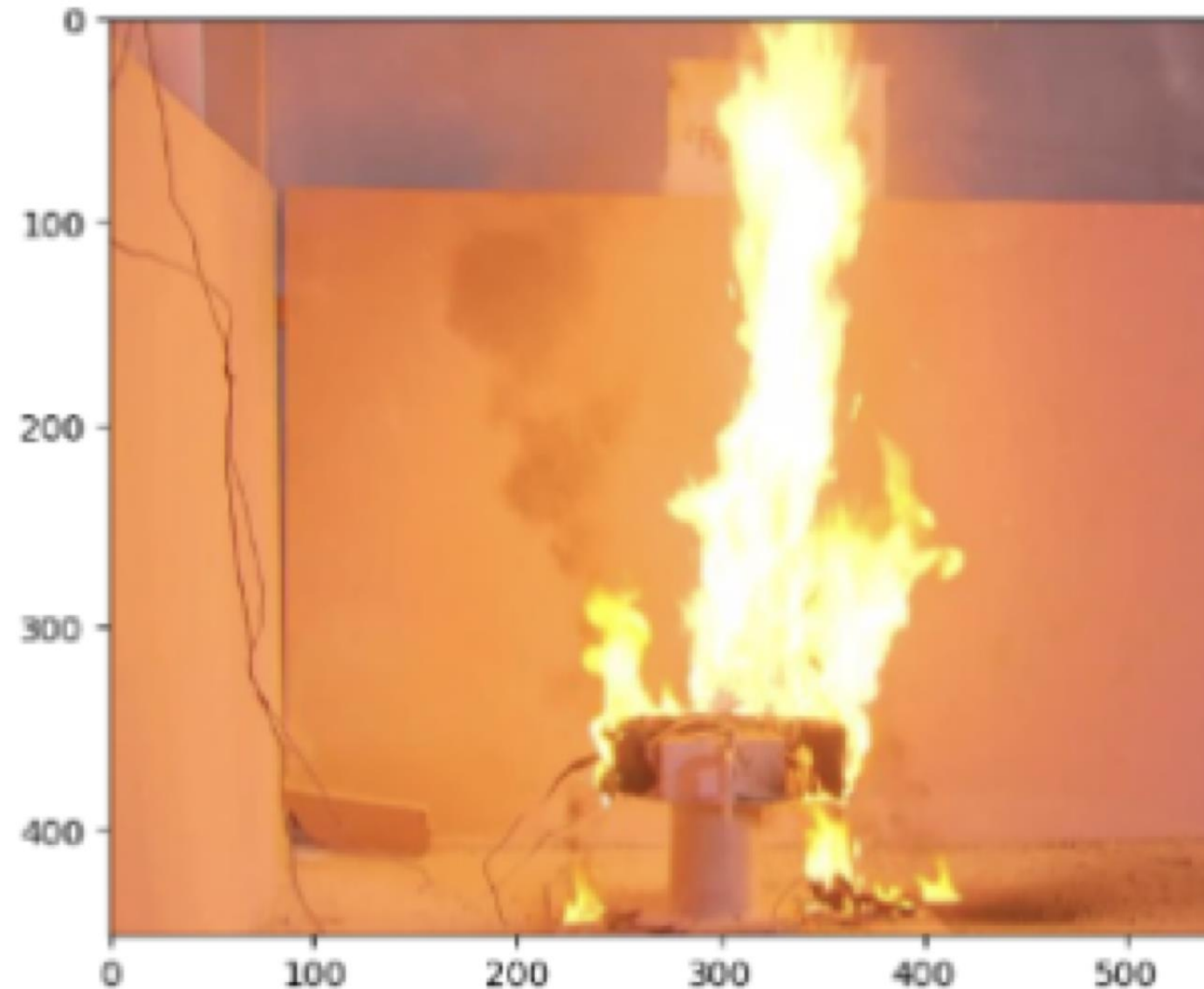
227 # 결과 시각화
--> 228 display_image(Image.open(image_path), filtered_predictions)

<img alt="A visualization of a model's prediction for a fire image. The image shows a fire burning in a container. The prediction is a heatmap overlaying the original image, with the fire area highlighted in red. The heatmap is generated using the function display_image(Image.open(image_path), filtered_predictions). The image is converted from RGB to BGR using the function OR_RGB2BGR. The heatmap is generated using the function grad_fn=<UnbindBackward0>." data-bbox="488 250 875 898"/>

[-0.1140, 0.1324, 0.0254, ..., -0.1154, -0.1431, -0.0514],
[-0.0179, -0.0795, -0.1592, ..., 0.2308, -0.0972, 0.1153]]].
grad_fn=<UnbindBackward0>)

-->

Value



YOLO Review

감사합니
다
