Thesis review study

You Only Look Once:

Unified, Real-Time Object Detection (2016)

Redmon Joseph, et al.

조은비, 전민하

CONTENT

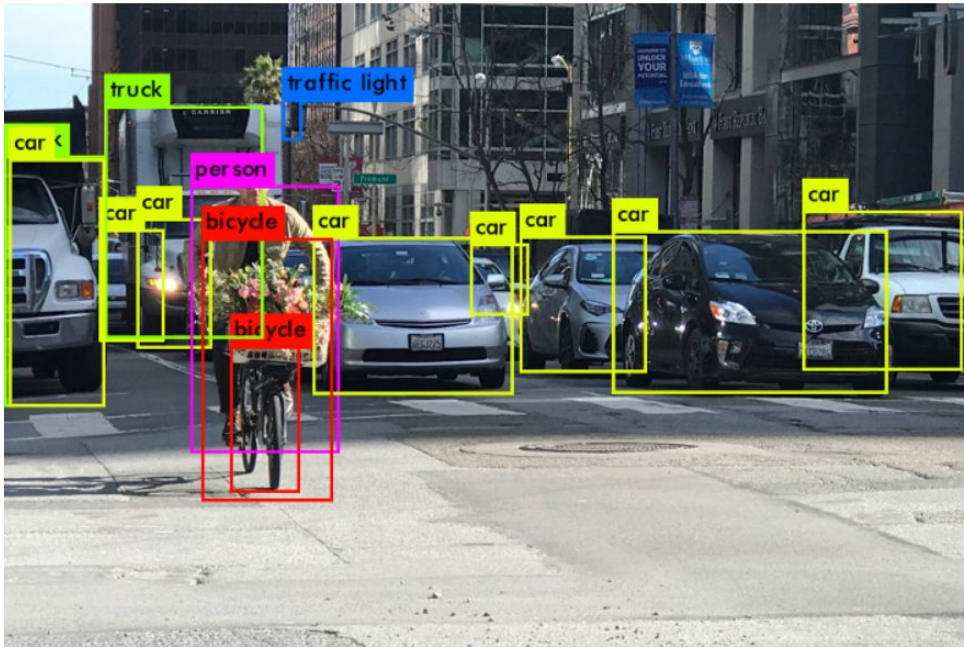
1. Introduction
2. Unified Detection
3. Comparison to Other Detection Systems
4. Experiments
5. Real-Time Detection In The Wild
6. Conclusion

CONTENT

1. Introduction
2. Unified Detection
3. Comparison to Other Detection Systems
4. Experiments
5. Real-Time Detection In The Wild
6. Conclusion

연구 목적

1) 기존 객체 탐지 방법의 한계를 해결하고,
2) 빠르고 단순하며 실시간 탐지가 가능한 객체 탐지 모델을 개발하는 것

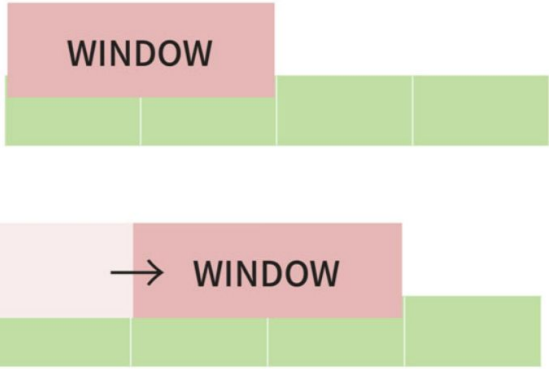


1. Introduction

기존

분류기(classifier)를 재활용하여 탐지

- DPM: 슬라이딩 윈도우(Sliding Window)
- R-CNN: 영역 제안(Region Proposal)
  - 이미지를 여러 개의 작은 영역으로 나누어 각각 분류: 주변 환경 반영 X
    ex) 나무에 비친 그림자를 사람으로 인식



1. 연산량 多
2. 복잡한 파이프라인으로 인해 속도 👎
   (영역 제안 → 분류 → 후처리)
3. 최적화 👎

1. Introduction

기존	YOLO

분류기(classifier)를 재활용하여 탐지
 	
- DPM: 슬라이딩 윈도우(Sliding Window)
- R-CNN: 영역 제안(Region Proposal)
  - 이미지를 여러 개의 작은 영역으로 나누어 각각 분류: 주변 환경 반영 X
  ex) 나무에 비친 그림자를 사람으로 인식
	단일 회귀 문제(single regression problem)
	
	- 단일 신경망이 이미지 픽셀에서 바운딩 박스 좌표와 클래스 확률을 직접 예측
	- **"You Only Look Once"**
	  - 이미지를 한번에 통째로 보고 모든 객체를 동시 탐지: 주변 환경 반영 O

1. 연산량 **多** 2. 복잡한 파이프라인으로 인해 속도 👎 (영역 제안 → 분류 → 후처리) 3. 최적화 👎	1. 단일 연산 2. 전체 파이프라인이 하나의 네트워크로 구성 3. 탐지 성능에 맞추어 종단간 최적화 가능 (end-to-end)

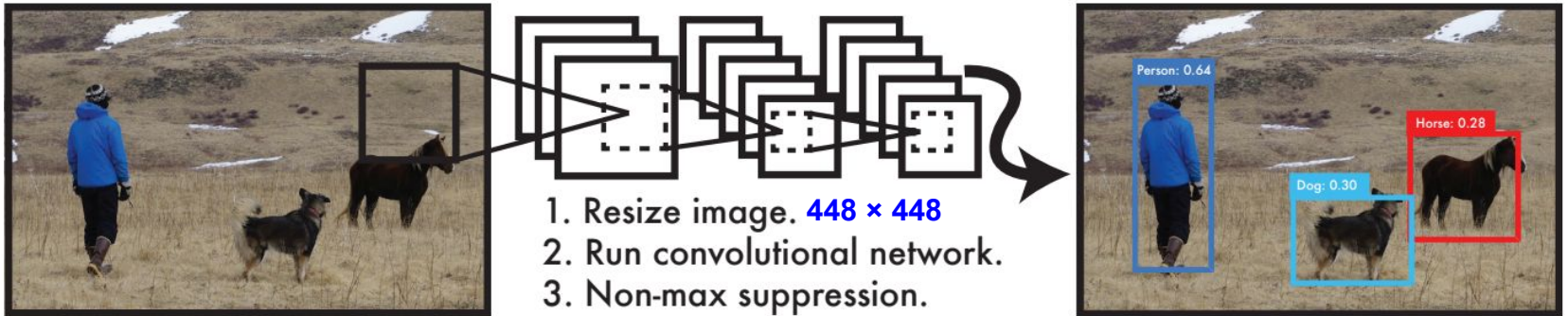
1. Introduction

장점

1. **속도**: 복잡한 파이프라인 X
   - 45 FPS(기본), 155 FPS(Fast YOLO)
   - 실시간 비디오 처리 가능
2. **전역적 추론**: 이미지 전체를 고려해 예측
   - 이미지의 맥락 정보를 암묵적으로 인코딩
   - 배경 오류: Fast R-CNN보다 절반 이하
3. **일반화 가능**: 자연 이미지에서 예술 작품까지 다양한 도메인에 적용 가능

한계

1. **정확도**
   - SOTA(최신 탐지 시스템)보다 뒤처짐
   - 특히 작은 객체의 정확한 위치 파악이 어려움



CONTENT

1. Introduction
2. Unified Detection
3. Comparison to Other Detection Systems
4. Experiments
5. Real-Time Detection In The Wild
6. Conclusion

2. Unified Detection

- x, y: 그리드 셀 내 박스 중심 좌표
- w, h: 이미지 전체 크기에 대한 상대 너비, 높이
- **신뢰도(confidence)**

모델이 해당 박스에 객체가 존재한다고 **확신하는 정도**, 예측된 박스가 실제 객체와 얼마나 일치하는지 나타냄 **(IOU)**

입력 이미지를 S × S 그리드로 나눔

신뢰도 계산:
Pr(Classi) × IOU_truth/pred

클래스 확률 예측
▶ 해당 셀이 객체를 포함할 경우의 확률

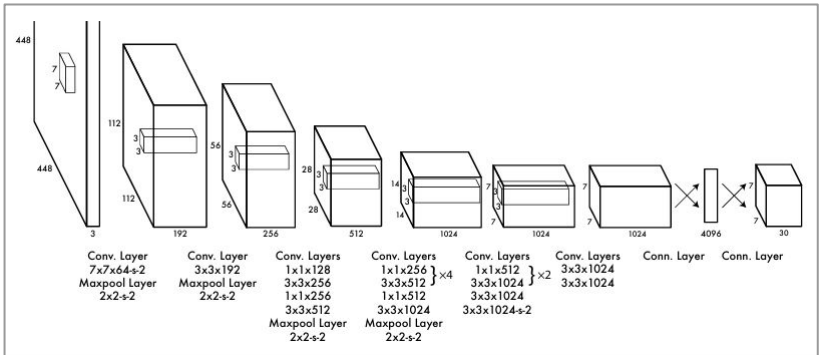
2-1. Network Design

- 합성곱 신경망(CNN) 구조 기반
- PASCAL VOC 데이터셋 사용 객체 탐지 수행
- **구조**: 24개의 합성곱 계층(Convolutional Layer) + 2개의 완전 연결 계층(fully connected layer)

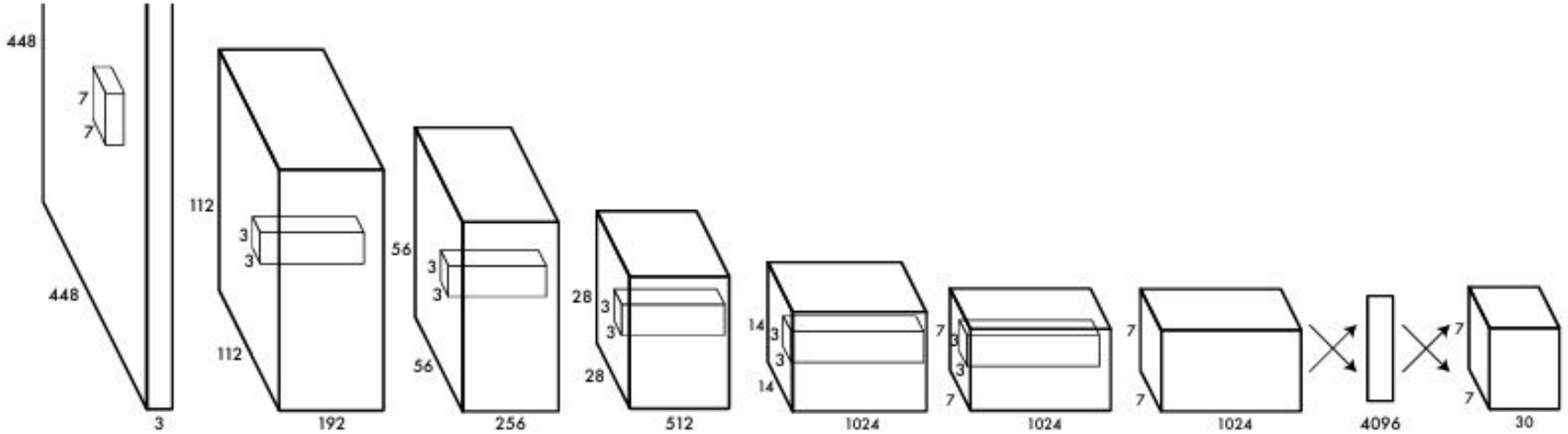
  이미지 **특징 추출**

  객체의 클래스 확률 및 바운딩 박스 좌표를 **예측**

- GoogLeNet에서 영감을 받음
  - Inception 모듈 대신 **1×1으로 차원을 줄인 후, 3×3 합성곱 계층을 적용**
- Fast YOLO
  - 합성곱 계층(24개 → 9개), 각 계층 필터 수 ↓
  - 더 적은 연산으로 빠른 탐지 가능
  - 훈련 및 테스트 과정은 YOLO와 동일
- 최종 output: 7 × 7 × 30 tensor
- 전체 네트워크 구조



2-1. Network Design



2-2. Training

1. 사전학습
   - ImageNet 데이터셋에서 사전학습 진행 (프레임워크: Darknet)
   - 성능: top-5 정확도 88%
2. 객체 탐지 모델로 변환하여 성능 향상
   - 레이어 추가: 4개의 convolution layer + 2개의 FC layer
   - 입력 이미지 크기 증가: 224×224 → **448×448**
3. **최종 출력**
   - 클래스 확률, 바운딩 박스 좌표 예측
   - 바운딩 박스 좌표를 **0~1** 범위로 정규화
4. 손실 함수: Sum-Squared Error(SSE, 제곱합 오차) 사용
   - 문제: classification error, localization error의 동일한 비중
   - 객체가 없는 셀에서도 손실 발생, 모델이 불안정해짐
   - 해결: Localization Loss가중치 증가(λ_coord=5), **바운딩 박스 좌표** 예측을 더 중요하게 반영
   - 객체가 없는 셀의 confidence 손실 가중치 감소(λ_noobj=0.5), 불필요한 손실을 줄이고 안정적인 학습 유도

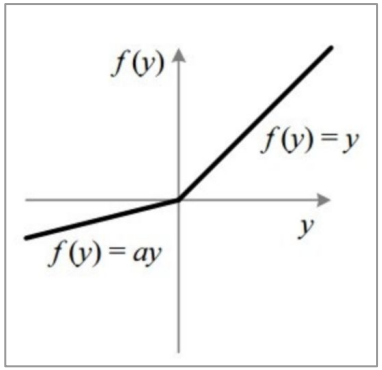
⭐ 학습률 변화

- 총 135 epoch 학습
- 처음: 10^{-3} → 10^{-2}로 점진적으로 증가 (불안정 방지)
- 75 epochs: 10^{-2} 유지
- 30 epochs: 10^{-3} 로 감소
- 30 epochs: 10^{-4}로 더 낮춰서 미세 조정

⭐ 활성화 함수

- 마지막 layer: 선형 활성화 함수 사용 → 바운딩 박스 좌표 등을 직접 예측하기 위함
- 나머지 layer: Leaky ReLU 사용

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



입력값이 0이하일 때 작은 값을 반환하여 완전한 **기울기 소실을 방지**
음수 값에서도 작은 기울기를 유지하여 **안정적인 학습**이 가능

2-3. Inference

- **단일 신경망 평가**로 탐지를 수행하므로 매우 빠름
- 98개의 바운딩 박스 예측
- 탐지된 바운딩 박스 중 겹치는 박스를 제거하기 위해
  Non-Maximum Suppression(NMS, 비최대 억제) 기법을 적용

2-3. Inference

- **단일 신경망 평가**로 탐지를 수행하므로 매우 빠름
- 98개의 바운딩 박스 예측
- 탐지된 바운딩 박스 중 겹치는 박스를 제거하기 위해
  Non-Maximum Suppression(NMS, 비최대 억제) 기법을 적용

2-4. Limitations of YOLO

- **근접한 객체 탐지 한계**
  - 각 셀은 두 개의 바운딩 박스만 예측 가능하며, 하나의 클래스만 가짐
  - 가까운 작은 객체, 무리 지은 객체(ex. 새 떼) 탐지가 어려움
- **새로운 객체 형태 일반화 부족**
  - 새로운 구성을 가진 객체 탐지가 어려움
- **작은 객체 탐지 성능 저하**
  - 작은 객체에서 바운딩 박스 오차가 탐지 성능에 큰 영향을 줌
- 주요 오류 원인
  - 잘못된 위치 예측

CONTENT

1. Introduction
2. Unified Detection
3. Comparison to Other Detection Systems
4. Experiments
5. Real-Time Detection In The Wild
6. Conclusion

3. Comparison to Other Detection Systems

시스템	특징	YOLO
DPM (Deformable Parts Models)	슬라이딩 윈도우 사용 정적 특징 기반 예측	단일 합성 신경망으로 전체 과정을 처리 더 빠르고 정확함
R-CNN	Selective Search로 바운딩 박스 제안 (약 2,000개) CNN으로 특징 추출	Grid 셀에서 바운딩 박스를 예측 더 적은 박스 예측(98개)
Fast/Faster R-CNN	Selective Search 대신 신경망 사용 R-CNN보다 속도 개선	파이프라인 제거 실시간 성능을 목표로 설계되어 빠른 속도 제공
단일 클래스 탐지기	얼굴, 사람 등 특정 클래스에 최적화 적은 변동성으로 고도의 최적화 가능	여러 객체와 클래스를 동시에 탐지 가능
Deep MultiBox	CNN으로 관심 영역 예측 단일 객체 탐지	하나의 통합된 모델로 탐지 수행 바운딩 박스와 클래스를 동시에 예측
OverFeat	지역화 후 후처리 필요 맥락 고려 안함	맥락을 고려하여 더 나은 성능 제공
MultiGrasp	Grid 형태로 나누고, **각 영역**에서 물체를 잡을 수 있는 가능성 예측	여러 객체의 바운딩 박스와 클래스를 예측

CONTENT

1. Introduction
2. Unified Detection
3. Comparison to Other Detection Systems
4. Experiments
5. Real-Time Detection In The Wild
6. Conclusion

4. Experiments

성능 평가 지표 (Performance Metrics)

- **mAP (mean Average Precision, 평균 정밀도)**
  - 정확도를 평가하는 대표적인 지표
  - 숫자가 높을수록 탐지 성능이 좋음을 의미함
- **FPS (Frames Per Second, 초당 프레임 수)**
  - 1초 동안 몇 개의 이미지를 처리할 수 있는지 측정
  - FPS가 높을수록 실시간 처리에 적합함

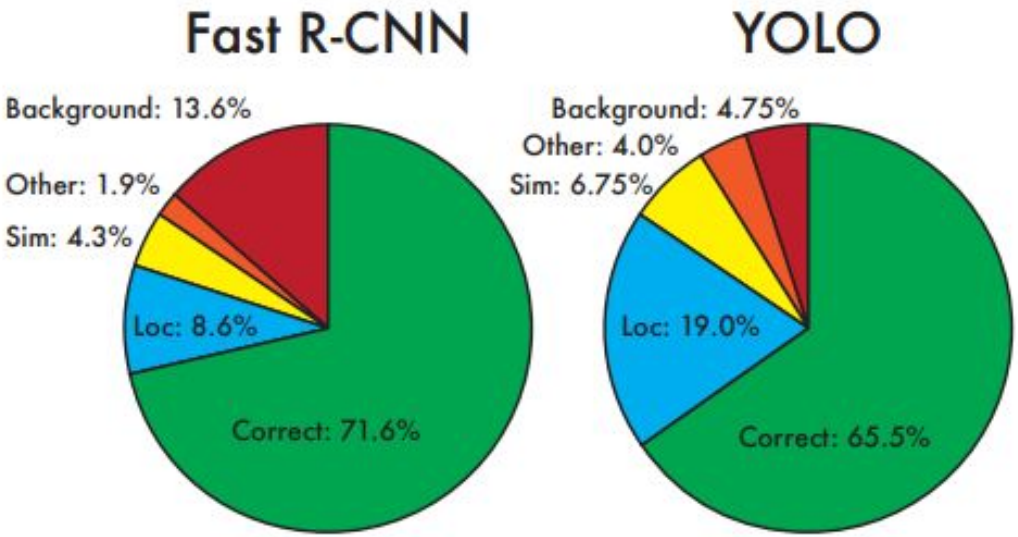
- YOLO vs 기존 객체탐지 모델들과 비교

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	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[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

- YOLO는 Fast R-CNN보다 훨씬 빠르다 (0.5 vs 45)
- 정확도는 Fast R-CNN보다 낮다
- Fast YOLO는 속도를 더 높였지만 정확도가 크게 떨어짐

4. Error Analysis(YOLO의 한계 분석)

- Fast R-CNN과 비교한 오류 분석 그래프



Localization Error	모델이 객체의 존재 여부는 잘 판단했지만, 박스의 위치나 크기가 정확하지 않을 때 발생하는 오류
• YOLO는 이미지를 Grid 방식으로 나누고, 각 칸에서 예측을 수행하는 방식이기 때문에, 작은 객체나 경계에 있는 객체를 탐지할 때 위치가 부정확할 가능성이 커짐 • Fast R-CNN은 객체 후보영역(Region Proposal)을 먼저 찾고 정확한 위치를 조정하기 때문에, Localization Error가 낮음	
YOLO가 Fast R-CNN보다 2배 이상 높게 나타남	
Background Error	객체가 없는 곳을 잘못 탐지하는 오류
• YOLO는 이미지를 한 번에 분석하는 방식이므로, 전체적인 맥락을 보고 객체와 배경을 더 잘 구분할 수 있다. • Fast R-CNN은 Region Proposal을 통해 ‘이게 객체일 수도?’라고 추측하는 일이 많아서, 배경을 객체로 잘못 탐지하곤 한다.	
YOLO는 배경을 객체로 잘못 탐지하는 비율이 Fast R-CNN보다 3배 적다.	

Fast R-CNN

- Localization error(위치 예측 오류) 8.6%
- Background error(배경 오탐) 13.6%

YOLO

- Localization error 19%
- Background error 4.75%

- **YOLO는 객체 존재 여부는 잘 맞추지만 객체의 정확한 위치를 정교하게 잡지는 못한다. 대신 배경은 잘 구분한다.**
- **Fast R-CNN은 위치도 정확하게 맞히지만, 배경을 객체로 착각하는 경우가 많음.**

4. Experiments

	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-CNN과 다른 모델들을 결합했을 때 성능이 어떻게 향상되는지 비교해봤다.

- Fast R-CNN에 YOLO를 결합했을 때 성능(mAP)이 3.2 증가
- YOLO를 결합했을 때 가장 큰 성능 향상이 나타남

YOLO와 Fast R-CNN은 상호 보완적

- YOLO는 빠르지만 정확도가 낮음
- Fast R-CNN은 정확도가 높지만 속도가 느림
- 두 모델을 결합하면 빠른 속도 + 높은 정확도를 동시에 얻을 수 있음

4. Experiments

객체 탐지 모델을 비교한 성능 순위표 - mAP(모델 전체 성능), 특정 객체 탐지 성능

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 [27]	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 [28]	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 [32]	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

- YOLO 단독 성능(mAP 57.9)은 다른 모델들에 비해 크게 높지 않다.
- Fast R-CNN과 합쳤을 때 기존 Fast R-CNN보다 성능이 향상됨
- 가장 높은 MR_CNN_MORE_DATA는 Multi-Region CNN 모델에 데이터 양을 늘려서 학습한 결과다.

CONTENT

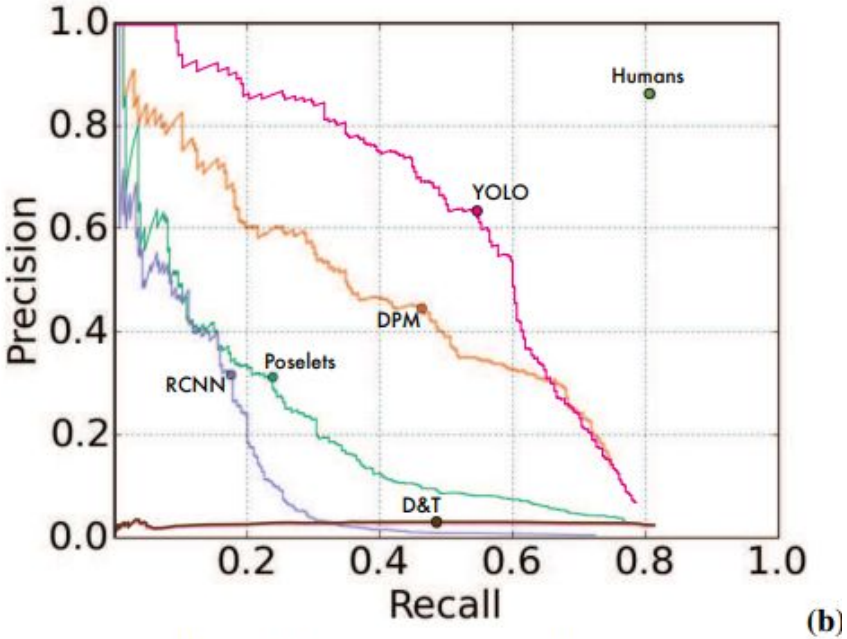
1. Introduction
2. Unified Detection
3. Comparison to Other Detection Systems
4. Experiments
5. Real-Time Detection In The Wild
6. Conclusion

5. Real-Time Detection in the Wild

PASCAL VOC 2007에서 학습한 YOLO가 Picasso Dataset 및 People-Art Dataset에서도 잘 작동하는가? (새 데이터셋)

- Precision(정밀도)과 Recall(재현율)의 관계를 나타낸 곡선.
- 객체 탐지 모델의 성능을 평가하는 중요한 지표.

- Recall(재현율) = 전체 객체 중에서 모델이 탐지한 비율.
- Precision(정밀도) = 모델이 탐지한 객체 중에서 실제로 맞는 비율. Precision이 높으면? → 잘못된 탐지가 적다는 뜻.



(a) Picasso Dataset precision-recall curves. (b) Th

YOLO (핑크색)
- 다른 모델보다 높은 Precision-Recall 성능을 보임.

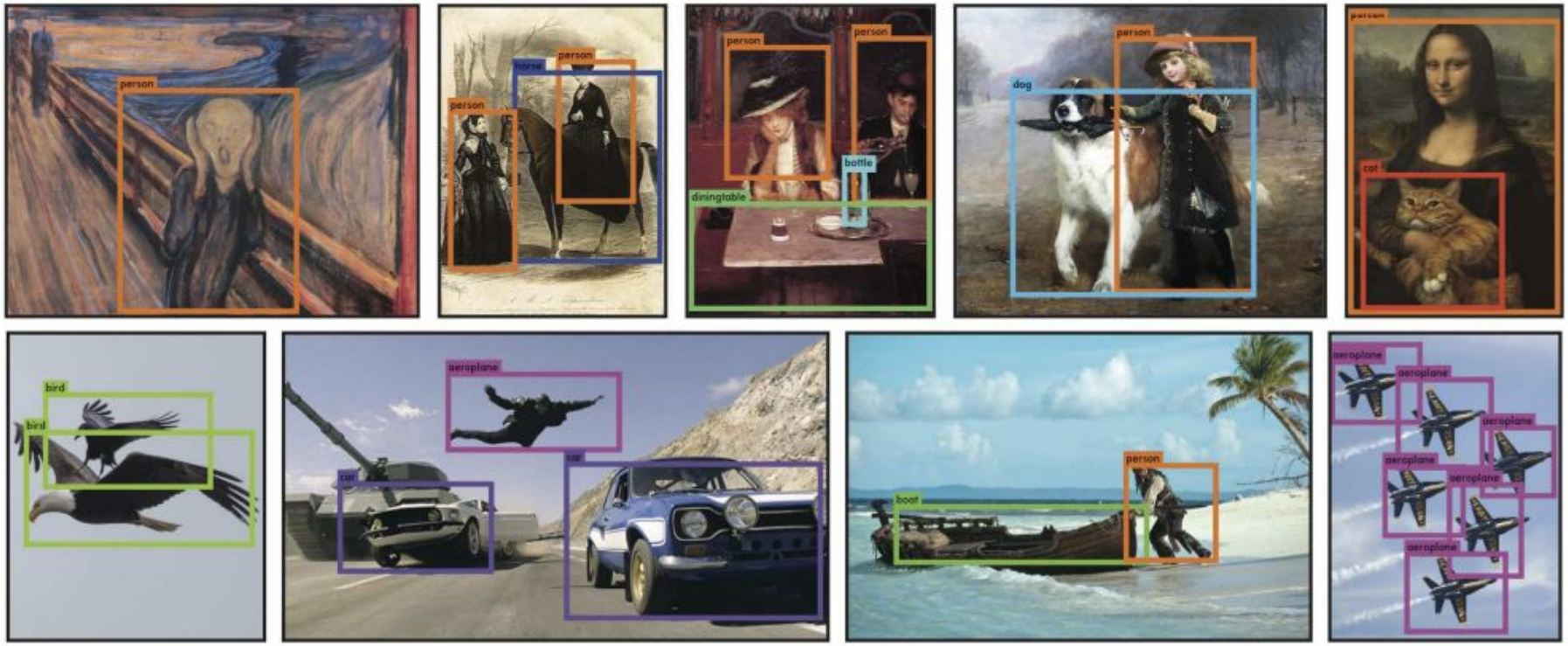
DPM (주황색)
- YOLO보다는 성능이 낮지만, 기존의 대표적인 탐지 모델 중 하나

R-CNN (파란색)
- R-CNN은 YOLO보다 새로운 데이터에 대한 적응력이 부족함

Poselets, D&T (청록색 & 갈색)
- 특히 D&T는 거의 Precision이 0에 가까울 정도로 탐지 성능이 떨어짐

결론: YOLO는 다른 모델보다 새로운 데이터셋에서도 강한 일반화 성능을 보인다.

5. Real-Time Detection in the Wild



YOLO는 예술 작품처럼 일반적이지 않은 이미지에서도 객체를 감지할 수 있다. 이는 YOLO가 훈련 데이터를 넘어서 일반화 성능이 뛰어나다는 것을 의미.

CONTENT

1. Introduction
2. Unified Detection
3. Comparison to Other Detection Systems
4. Experiments
5. Real-Time Detection In The Wild
6. Conclusion

6. Conclusion

- YOLO는 속도가 빠르고 실시간 객체 탐지가 가능한 혁신적인 모델
- 기존 객체 탐지 모델보다 단순한 구조 & End-to-End 학습 가능
- 웹캠, 자율주행, 감시 시스템 등 다양한 분야에서 실시간 적용 가능
- 하지만 작거나 겹친 객체 탐지의 한계가 있으며, 이후 연구에서 개선될 것