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

<"한 번만 보면 돼: 통합된 실시간 객체 탐지" >

Joseph Redmon∗, Santosh Divvala∗†, Ross Girshick¶, Ali Farhadi∗† 워싱턴 대학교∗, Allen Institute for AI†, Facebook AI Research¶

**개요**

1. **새로운 접근법**: YOLO는 객체 검출을 회귀 문제로 프레임화하여 단일 신경망을 사용합니다.
2. **단일 평가**: 전체 이미지를 한 번 평가하여 경계 상자와 클래스 확률을 예측합니다.
3. **빠른 처리 속도**: YOLO 모델은 실시간으로 45프레임/초, Fast YOLO는 155프레임/초를 처리합니다.
4. **성능**: Fast YOLO는 다른 실시간 검출기보다 두 배의 mAP를 달성합니다.
5. **오류 및 오탐지**: YOLO는 위치 오류가 더 많지만 배경에 대한 오탐지가 적습니다.
6. **일반화 능력**: YOLO는 다양한 도메인에서 다른 검출 방법보다 더 좋은 성능을 보입니다.

**1. Introduction (소개)**

1. **새로운 접근법**: YOLO는 객체 검출을 회귀 문제로 프레임화하여 단일 신경망을 사용합니다.
2. **단일 평가**: 전체 이미지를 한 번 평가하여 경계 상자와 클래스 확률을 예측합니다.
3. **빠른 처리 속도**: YOLO 모델은 실시간으로 45프레임/초, Fast YOLO는 155프레임/초를 처리합니다.
4. **성능**: Fast YOLO는 다른 실시간 검출기보다 두 배의 mAP를 달성합니다.
5. **오류 및 오탐지**: YOLO는 위치 오류가 더 많지만 배경에 대한 오탐지가 적습니다.
6. **일반화 능력**: YOLO는 다양한 도메인에서 다른 검출 방법보다 더 좋은 성능을 보입니다.
7. **속도**: YOLO는 복잡한 파이프라인 없이 테스트 시 신경망을 실행하여 매우 빠르게 검출을 예측합니다. 기본 네트워크는 초당 45프레임, 빠른 버전은 초당 150프레임 이상을 실행합니다.
8. **실시간 처리**: 실시간 스트리밍 비디오를 25밀리초 미만의 지연 시간으로 처리할 수 있습니다.
9. **정확도**: YOLO는 다른 실시간 시스템보다 두 배 이상의 평균 정밀도를 달성합니다.
10. **전역적 고려**: YOLO는 훈련 및 테스트 시 전체 이미지를 보기 때문에 클래스의 컨텍스트 정보를 포함한 예측을 합니다.
11. **배경 오류 감소**: YOLO는 Fast R-CNN보다 배경 오류를 절반 이하로 줄입니다.
12. **일반화 가능성**: YOLO는 다양한 도메인에서 뛰어난 성능을 보이며 새로운 도메인이나 예기치 않은 입력에 잘 대응합니다.
13. **정확도 트레이드오프**: YOLO는 정확도가 다소 떨어지며 특히 작은 객체의 위치를 지정하는 데 어려움을 겪습니다.
14. **오픈 소스**: 모든 훈련 및 테스트 코드는 오픈 소스로 제공되며 다양한 사전 훈련된 모델도 다운로드할 수 있습니다.

포유류, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**[그림1: YOLO 탐지 시스템]**

**1. Resize Image (이미지 크기 조정):**

* 입력 이미지를 448 × 448 픽셀 크기로 조정합니다. 이는 합성곱 신경망에 입력하기 위한 표준 크기입니다.

**2.** **Run Convolutional Network (합성곱 신경망 실행):**

* 크기가 조정된 이미지를 단일 합성곱 신경망에 입력합니다. 이 신경망은 이미지 내 객체를 탐지하고, 각 객체의 위치와 클래스(사람, 개, 말 등)를 예측합니다.

**3.** **Non-max Suppression (비최대 억제):**

* 신경망이 예측한 여러 객체들 중에서 가장 신뢰도가 높은 예측만을 남기고 나머지 겹치는 예측은 제거합니다. 이렇게 하면 중복된 탐지를 줄이고 최종 결과를 더욱 명확하게 합니다.

**2. Unified Detection (통합된 검출)**

1. **통합 신경망**: YOLO는 객체 검출의 개별 구성 요소를 단일 신경망으로 통합하여 전체 이미지의 특징을 사용해 경계 상자를 예측합니다.
2. **그리드 기반 예측**: 입력 이미지를 S × S 그리드로 나누고, 객체의 중심이 포함된 그리드 셀이 해당 객체를 검출합니다.
3. **경계 상자와 신뢰도**: 각 그리드 셀은 B개의 경계 상자와 신뢰도 점수를 예측하며, 신뢰도는 객체 포함 여부와 상자의 정확성을 반영합니다.
4. **예측값**: 각 경계 상자는 5가지 예측값(x, y, w, h, 신뢰도)으로 구성됩니다.
5. **조건부 클래스 확률**: 각 그리드 셀은 C개의 조건부 클래스 확률을 예측하며, 그리드 셀에 객체가 포함되어 있을 때만 이를 예측합니다.
6. **테스트 시 결합**: 테스트 시 조건부 클래스 확률과 신뢰도 예측값을 곱하여 클래스별 신뢰도 점수를 생성하며, 이는 해당 클래스가 상자에 나타날 확률과 상자가 객체에 얼마나 잘 맞는지를 나타냅니다.

자전거, 자전거 바퀴, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**[그림 2: 모델]**

1. **S × S Grid on Input (입력 이미지의 S × S 그리드):**
   * 입력 이미지를 S × S 그리드로 나눕니다. 각 그리드 셀은 이미지의 작은 부분을 나타냅니다.
2. **Bounding Boxes + Confidence (경계 상자 + 신뢰도):**
   * 각 그리드 셀에서 B개의 경계 상자를 예측합니다. 각 경계 상자는 그 상자의 위치와 크기, 그리고 그 상자가 객체를 포함하고 있을 확률(신뢰도)을 나타냅니다.
3. **Class Probability Map (클래스 확률 맵):**
   * 각 그리드 셀에서 C개의 클래스에 대한 확률을 예측합니다. 이는 해당 그리드 셀에 특정 클래스의 객체가 있을 가능성을 나타냅니다.
4. **Final Detections (최종 탐지 결과):**
   * 예측된 경계 상자와 클래스 확률을 결합하여 최종 객체 탐지 결과를 생성합니다. 이는 이미지 내 객체들의 위치와 클래스를 나타내는 경계 상자로 시각화됩니다.

**2.1. Network Design (네트워크 설계)**

1. **모델 구현**: YOLO 모델은 합성곱 신경망으로 구현되었으며 PASCAL VOC 검출 데이터셋에서 평가되었습니다.
2. **초기 합성곱 층**: 초기 합성곱 층은 이미지에서 특징을 추출합니다.
3. **완전 연결 층**: 완전 연결 층은 출력 확률과 좌표를 예측합니다.
4. **아키텍처 영감**: 네트워크 아키텍처는 GoogLeNet 모델에서 영감을 받았습니다.
5. **네트워크 구성**: 24개의 합성곱 층과 2개의 완전 연결 층으로 구성되어 있습니다.
6. **축소 및 합성곱 층**: 1 × 1 축소 층과 3 × 3 합성곱 층을 사용합니다.
7. **빠른 버전 (Fast YOLO)**: Fast YOLO는 9개의 합성곱 층과 적은 수의 필터를 사용하여 빠른 객체 검출을 목표로 합니다.
8. **훈련 및 테스트 매개변수**: YOLO와 Fast YOLO의 훈련 및 테스트 매개변수는 네트워크 크기를 제외하고 동일합니다.
9. **최종 출력**: 네트워크의 최종 출력은 7 × 7 × 30 예측 텐서입니다.

**스케치, 도표, 기술 도면, 평면도이(가) 표시된 사진

자동 생성된 설명**

**[그림 3: 아키텍처]**

**1. 입력 이미지:**

* 크기가 448 × 448 픽셀인 이미지가 입력됩니다. 이 이미지는 3개의 색상 채널(RGB)을 갖고 있습니다.

**2.** **Conv. Layer 7x7x64-s-2 (합성곱 계층 7x7x64-s-2)와 Maxpool Layer 2x2-s-2 (최대 풀링 계층 2x2-s-2):**

* 첫 번째 합성곱 계층은 7 × 7 커널과 64개의 필터를 사용하며, 스트라이드(stride)는 2입니다.
* 그 다음 최대 풀링 계층은 2 × 2 풀링 창과 스트라이드 2를 사용합니다.

**3. Conv. Layer 3x3x192와 Maxpool Layer 2x2-s-2:**

* 다음 합성곱 계층은 3 × 3 커널과 192개의 필터를 사용합니다.
* 그 다음 최대 풀링 계층은 2 × 2 풀링 창과 스트라이드 2를 사용합니다.

**4. Conv. Layers 1x1x128, 3x3x256, 1x1x256, 3x3x512:**

* 1 × 1 합성곱 계층(128 필터), 3 × 3 합성곱 계층(256 필터), 1 × 1 합성곱 계층(256 필터), 3 × 3 합성곱 계층(512 필터)의 순서로 이어집니다.
* 이러한 구조가 4번 반복됩니다.
* 그 다음 최대 풀링 계층은 2 × 2 풀링 창과 스트라이드 2를 사용합니다.

5. **Conv. Layers 1x1x256, 3x3x512:**

* 1 × 1 합성곱 계층(256 필터), 3 × 3 합성곱 계층(512 필터)의 순서로 이어집니다.
* 이러한 구조가 2번 반복됩니다.

**6. Conv. Layers 3x3x1024:**

* 3 × 3 합성곱 계층(1024 필터)이 2번 반복됩니다.
* 최대 풀링 계층은 2 × 2 풀링 창과 스트라이드 2를 사용합니다.

**7. Fully Connected Layers (완전 연결 계층):**

* 4096개의 뉴런을 가진 완전 연결 계층이 뒤따릅니다.
* 최종 출력은 7 × 7 × 30의 크기를 갖는 텐서로, 각 그리드 셀에 대한 경계 상자와 클래스 확률을 나타냅니다.
* **과정 요약:**

1. **입력 이미지 크기 조정 및 전처리:**
   * 입력 이미지를 448 × 448 픽셀로 크기 조정합니다.
2. **합성곱 계층과 풀링 계층:**
   * 24개의 합성곱 계층을 통해 이미지의 특징을 추출합니다.
   * 1 × 1 합성곱 계층을 사용하여 특징 공간을 줄입니다.
3. **완전 연결 계층:**
   * 2개의 완전 연결 계층을 사용하여 최종 예측을 생성합니다.
4. **출력:**
   * 최종 출력은 각 그리드 셀에 대한 경계 상자와 클래스 확률을 포함하는 7 × 7 × 30 텐서입니다.

YOLO 모델이 이미지 내 객체를 실시간으로 빠르고 정확하게 탐지할 수 있게 합니다.

**2.2. 훈련**

1. **사전 훈련**: 합성곱 층을 ImageNet 1000-클래스 대회 데이터셋에서 사전 훈련합니다.
2. **훈련 구조**: 처음 20개의 합성곱 층에 평균 풀링 층과 완전 연결 층을 추가하여 약 일주일 동안 훈련합니다.
3. **성능**: 사전 훈련된 네트워크는 ImageNet 2012 검증 세트에서 88%의 단일 크롭 상위 5위 정확도를 달성합니다.
4. **Darknet 프레임워크**: 모든 훈련과 추론에 Darknet 프레임워크를 사용합니다.
5. **모델 변환**: 사전 훈련된 모델을 검출 작업에 맞게 변환합니다.
6. **추가 층**: 랜덤 초기화된 가중치를 가진 4개의 합성곱 층과 2개의 완전 연결 층을 추가합니다.
7. **입력 해상도 증가**: 검출 작업에는 세밀한 시각 정보가 필요하므로 입력 해상도를 224 × 224에서 448 × 448로 증가시킵니다.
8. **최종 층 예측**: 최종 층은 클래스 확률과 경계 상자 좌표를 모두 예측합니다.
9. **정규화**: 경계 상자의 너비와 높이는 이미지의 너비와 높이로 정규화하여 0에서 1 사이의 값으로 만듭니다.
10. **매개변수화**: 경계 상자의 x와 y 좌표는 특정 그리드 셀 위치의 오프셋으로 매개변수화하여 0에서 1 사이의 값으로 제한합니다.
11. **활성화 함수**: 최종 층에는 선형 활성화 함수를 사용하고, 다른 층에는 Leaky Rectified Linear 활성화 함수를 사용합니다.

-------------------------------------------------------------------------------------------------------------

1. **합-제곱 오류의 사용**:
   * 모델 출력에서 합-제곱 오류를 최적화한다.
   * 합-제곱 오류는 평균 정밀도와 완전히 일치하지 않음.
   * 위치 오차와 분류 오차를 동일하게 취급함.
2. **그리드 셀의 신뢰도 점수 문제**:
   * 많은 그리드 셀이 객체를 포함하지 않음.
   * 이러한 셀의 신뢰도 점수가 0으로 밀려 모델 불안정성 초래.
3. **손실 함수 조정**:
   * 바운딩 박스 좌표 예측 손실 증가.
   * 객체를 포함하지 않는 박스의 신뢰도 예측 손실 감소.
   * 매개변수 λcoord와 λnoobj를 사용 (λcoord = 5, λnoobj = 0.5).
4. **크기별 오류 가중치**:
   * 합-제곱 오류가 큰 박스와 작은 박스의 오류를 동일하게 가중치 줌.
   * 작은 박스에서의 작은 편차가 더 중요함.
   * 너비와 높이 대신 너비와 높이의 제곱근을 예측함.
5. **바운딩 박스 예측기 책임 할당**:
   * 각 객체에 대해 하나의 바운딩 박스 예측기가 책임을 지도록 함.
   * 현재 IOU와 정답을 비교해 가장 높은 IOU를 가진 예측기를 책임자로 지정.
   * 이는 예측기들 간의 특수화를 초래하여 전체적인 리콜 향상.
6. **훈련 과정**:
   * PASCAL VOC 2007 및 2012 데이터 세트로 약 135 에포크 동안 네트워크를 훈련함.
   * 2012년 테스트 시 2007년 VOC 테스트 데이터도 포함함.
7. **훈련 설정**:
   * 배치 크기: 64
   * 모멘텀: 0.9
   * 가중치 감소율: 0.0005
8. **학습률 스케줄**:
   * 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.
9. **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 Inference (추론)**

1. **단일 네트워크 평가**:
   * 테스트 이미지에 대한 예측은 단 하나의 네트워크 평가만 필요.
   * PASCAL VOC에서 이미지당 98개의 바운딩 박스와 각 박스의 클래스 확률을 예측함.
   * YOLO는 분류기 기반 방법들과 달리 테스트 시간에 매우 빠름.
2. **그리드 디자인**:
   * 그리드 디자인은 바운딩 박스 예측에서 공간적 다양성을 강제함.
   * 객체가 어느 그리드 셀에 속하는지 명확한 경우가 많아 네트워크는 각 객체에 대해 하나의 박스만 예측함.
   * 큰 객체나 여러 셀의 경계에 있는 객체는 여러 셀에 의해 잘 위치할 수 있음.
3. **비최대 억제(non-maximal suppression)**:
   * 여러 번의 탐지를 수정하기 위해 비최대 억제를 사용할 수 있음.
   * R-CNN이나 DPM처럼 성능에 결정적이지는 않지만, mAP를 2-3% 향상시킴.

**2.4 Limitations of YOLO (YOLO의 한계점)**

1. **강한 공간적 제약**:
   * 각 그리드 셀이 두 개의 박스만 예측하고 하나의 클래스만 가질 수 있음.
   * 이러한 제약으로 인해 인접한 객체의 수를 제한함.
   * 새 떼와 같은 그룹으로 나타나는 작은 객체를 다루는 데 어려움이 있음.
2. **일반화 문제**:
   * 모델이 데이터에서 바운딩 박스를 예측하도록 학습하기 때문에 새로운 또는 특이한 종횡비나 구성의 객체에 일반화하는 데 어려움이 있음.
3. **조잡한 특징 사용**:
   * 입력 이미지에서 여러 다운샘플링 레이어를 거치는 아키텍처로 인해 상대적으로 조잡한 특징을 사용하여 바운딩 박스를 예측함.
4. **손실 함수의 한계**:
   * 탐지 성능을 근사화하는 손실 함수로 학습하지만 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 설정임.

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

1. **변형 가능한 파트 모델(DPM: Deformable Parts Models):**
   * 슬라이딩 윈도우 접근 방식을 사용.
   * 정적인 특징 추출, 영역 분류, 바운딩 박스 예측 등을 분리된 파이프라인으로 처리.
   * YOLO는 이러한 부분들을 하나의 컨볼루션 신경망으로 대체하여 더 빠르고 정확한 모델을 제공.
2. **R-CNN**:
   * 영역 제안을 사용하여 이미지에서 객체를 찾음.
   * Selective Search, 컨볼루션 네트워크, SVM, 선형 모델, 비최대 억제로 구성된 복잡한 파이프라인.
   * YOLO는 그리드 셀이 잠재적 바운딩 박스를 제안하고 공간적 제약을 두어 동일 객체의 여러 탐지를 완화.
   * YOLO는 적은 바운딩 박스를 제안하며, 단일 최적화된 모델로 결합.
3. **기타 빠른 검출기**:
   * Fast 및 Faster R-CNN은 R-CNN 프레임워크를 가속화하지만 여전히 실시간 성능에는 미치지 못함.
   * YOLO는 파이프라인을 없애고 설계 자체가 빠르도록 만듦.
4. **Deep MultiBox**:
   * 관심 영역을 예측하기 위해 컨볼루션 신경망을 사용.
   * 일반 객체 검출을 수행할 수 없으며, 여전히 더 큰 검출 파이프라인의 일부.
   * YOLO는 완전한 검출 시스템임.
5. **OverFeat**:
   * 슬라이딩 윈도우 검출을 수행하지만 분리된 시스템.
   * 위치 성능을 최적화하며 전역 컨텍스트를 고려하지 않음.
   * 상당한 후처리가 필요함.
6. **MultiGrasp**:
   * 그립 검출 작업과 유사한 설계.
   * 그립 검출은 객체 검출보다 간단한 작업.
   * YOLO는 여러 클래스의 여러 객체에 대해 바운딩 박스와 클래스 확률을 예측.

**4. Experiments (실험)**

1. **PASCAL VOC 2007 비교**:
   * 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)**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* **실시간 탐지기:**
  + **100Hz DPM**과 **30Hz DPM**은 PASCAL VOC 2007 데이터셋에서 훈련되었으며, 각각 100 FPS와 30 FPS의 속도를 가지고 있지만, 정확도는 비교적 낮습니다 (각각 16.0 mAP와 26.1 mAP).
  + **Fast YOLO**는 PASCAL VOC 2007과 2012 데이터셋에서 훈련되었으며, 52.7 mAP의 정확도와 155 FPS의 매우 빠른 속도를 가지고 있습니다.
  + **YOLO**는 동일한 데이터셋에서 훈련되었으며, 63.4 mAP의 더 높은 정확도를 가지고 있지만 45 FPS의 속도를 가집니다. 이는 실시간 성능을 유지하면서 더 높은 정확도를 제공합니다.
* **실시간 미만 탐지기:**
  + **Fastest DPM**은 15 FPS의 속도로 실시간 성능에 가깝지만, 30.4 mAP로 정확도가 낮습니다.
  + **R-CNN Minus R**은 6 FPS로 느리지만, 53.5 mAP로 높은 정확도를 가집니다.
  + **Fast R-CNN**은 매우 높은 정확도(70.0 mAP)를 가졌지만, 0.5 FPS로 매우 느립니다.
  + **Faster R-CNN VGG-16**과 **Faster R-CNN ZF**는 각각 7 FPS와 18 FPS의 속도를 가지며, 높은 정확도(73.2 mAP와 62.1 mAP)를 제공합니다.
  + **YOLO VGG-16**은 21 FPS의 속도로 실시간에 가깝고, 66.4 mAP의 높은 정확도를 가집니다.

이 표는 다양한 객체 탐지 모델의 성능과 속도를 비교하여 YOLO와 Fast YOLO가 다른 모델들에 비해 얼마나 빠르고 정확한지를 강조합니다. YOLO는 실시간 성능을 유지하면서도 높은 정확도를 제공하며, Fast YOLO는 가장 빠른 속도를 자랑합니다.

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

Fast R-CNN과 YOLO를 비교

**텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명**

**1.** **Fast R-CNN:**

* **Correct (정확한 탐지):** 71.6%
* **Loc (위치 오류):** 8.6%
* **Sim (유사한 객체 오류):** 4.3%
* **Other (기타 오류):** 1.9%
* **Background (배경 오류):** 13.6%

**2.** **YOLO:**

* **Correct (정확한 탐지):** 65.5%
* **Loc (위치 오류):** 19.0%
* **Sim (유사한 객체 오류):** 6.75%
* **Other (기타 오류):** 4.0%
* **Background (배경 오류):** 4.75%

- Fast R-CNN은 YOLO보다 더 높은 정확도를 가지고 있으며, 위치 오류와 기타 오류에서 더 나은 성능을 보입니다.

- 반면 YOLO는 배경 오류가 적으며, 실시간 탐지 시스템에서 중요한 측면인 빠른 속도를 가지고 있습니다.

- YOLO의 더 높은 위치 오류와 기타 오류는 개선이 필요한 부분이지만, 배경 오류가 적다는 점에서 강점을 가집니다.

**4.3 Combining Fast R-CNN and YOLO (Fast R-CNN과 YOLO 결합)**

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

1. **배경 오류 감소**:
   * YOLO는 Fast R-CNN보다 배경 오류를 훨씬 적게 발생시킴.
2. **YOLO와 Fast R-CNN 결합**:
   * YOLO를 사용해 Fast R-CNN의 배경 검출을 제거하여 성능 향상.
   * YOLO가 예측한 유사 박스와의 겹침 정도와 확률을 바탕으로 Fast R-CNN의 예측값을 보강.
3. **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 불과함.
4. **효과의 원인**:
   * YOLO의 성능 향상 효과는 모델 앙상블의 부산물이 아님.
   * YOLO는 테스트 시 서로 다른 종류의 실수를 하여 Fast R-CNN의 성능을 크게 향상시킴.
5. **속도 문제**:
   * YOLO의 속도 이점을 활용하지 못함 (각 모델을 별도로 실행 후 결과를 결합).
   * YOLO가 매우 빠르기 때문에 계산 시간에 큰 추가는 없음.

**4.4. VOC 2012 Results ( VOC 2012 결과 )**

1. **YOLO의 VOC 2012 성능**:
   * VOC 2012 테스트 세트에서 YOLO는 57.9% mAP를 기록.
   * 최신 기술에는 미치지 않으며, 원래 R-CNN과 VGG-16에 더 가까운 성능.
2. **성능 차이**:
   * 작은 객체 검출에서 어려움을 겪음.
   * 병, 양, TV/모니터 카테고리에서 R-CNN이나 Feature Edit보다 8-10% 낮은 성능.
   * 고양이와 기차 카테고리에서는 더 높은 성능을 달성.
3. **Fast R-CNN + YOLO 결합 성능**:
   * 결합된 모델은 높은 성능을 보임.
   * Fast R-CNN은 YOLO와 결합하여 2.3% 성능 향상.
   * 공공 리더보드에서 5위 상승.

**4.5. Generalizability: Person Detection in Artwork (일반화 가능성: 예술 작품에서의 사람 검출)**

텍스트, 라인, 그래프, 도표이(가) 표시된 사진

자동 생성된 설명

1. **일반화 문제**:
   * 학술 데이터셋은 같은 분포에서 훈련 및 테스트 데이터를 추출함. 실제 상황에서는 테스트 데이터가 시스템이 본 데이터와 다를 수 있음.
2. **YOLO vs. 다른 방법**:
   * **R-CNN**: VOC 2007에서는 높은 AP 기록, 하지만 예술 작품에서 성능이 크게 저하됨. Selective Search가 자연 이미지에 맞춰져 있음.
   * **DPM**: 예술 작품에서 AP를 잘 유지하지만 원래 AP가 낮음. 객체의 형태와 배치에 대한 강력한 모델을 가짐.
   * **YOLO**: VOC 2007에서 좋은 성능을 보이며, 예술 작품에서의 성능 저하가 적음. 객체의 크기와 형태, 관계를 모델링하여 좋은 검출을 유지함.

**5. Real-Time Detection In The Wild ( 실제 환경에서의 실시간 검출 )**

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 추적할 수 있는 시스템처럼 작동함.
3. **데모 및 소스 코드**:
   * 시스템의 데모와 소스 코드는 프로젝트 웹사이트(http://pjreddie.com/yolo/)에서 제공됨.

콜라주, 페인팅, 사진 액자, 예술이(가) 표시된 사진

자동 생성된 설명

**[그림 6. 질적인 결과]**

- 강점: 다양한 이미지 유형에서 대체로 정확한 객체 탐지 능력

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

**6. 결론**

1. **YOLO의 소개**:
   * YOLO는 객체 검출을 위한 통합 모델로, 전체 이미지에서 직접 훈련할 수 있고, 간단하게 구성할 수 있음.
2. **모델 훈련 방식**:
   * YOLO는 검출 성능에 직접 대응하는 손실 함수로 훈련되며, 전체 모델이 함께 훈련됨.
3. **속도 및 성능**:
   * Fast YOLO는 가장 빠른 범용 객체 검출기이며, 최신 기술의 정점을 찍음.
   * YOLO는 새로운 도메인에 잘 일반화되며, 빠르고 강력한 객체 검출이 필요한 애플리케이션에 적합함.