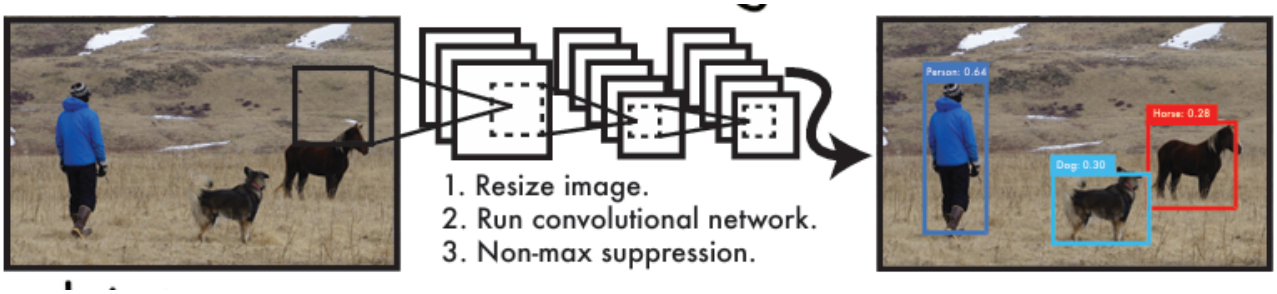
Yolo 는 객체 탐지를 위한 새로운 접근 방식. 이전 연구들이 객체 탐지를 위해 분류기(classifier)를 재활용했다면 YOLO는 객체 탐지를 spatially separated bounding box 와 그에 대응하는 class probablility 를 예측하는 회귀 문제로 설정

- 단일 싱경망이 한 번의 연산으로 전체 이미지로부터 bounding box 와 class probability 를 직접 예측
- 탐지 파이프라인이 하나의 네트워크로 되어 있다.
- 이미지 처리 속도가 빠르다
- 다만 다른 최신 탐지 시스템보다 localization 오류가 조금 더 많음
- 배경의 false positive 오류는 더 적음
- general representation 을 학습하는데 용이함
  DPM 이나 R-CNN 같은 기존 탐지 기법보다 자연 이미지 외 영역 (ex: artworks) 에서도 더 나은 성능을 보임

Intro

- 기존 모델들의 문제점
  - DPM (Deformable Parts Model): 분류기를 repurpose 하는 방식으로 객체를 탐지. 이미지의 여러 위치와 크기에 대해 분류기를 반복적으로 적용하여 객체가 있는지 판단.
    - 슬라이딩 윈도우 기반으로 이미지 전체를 일정 간격으로 스캔
  - R-CNN: region proposal(후보 영역) 을 먼저 생성한 뒤, 각 영역마다 분류기를 돌려야 하므로 학습과 최적화가 복잡
    -> **느리고, 각 구성요소를 따로 학습해야 하므로 최적화가 어렵다는 문제**
- YOLO 의 접근
  : 객체 탐자를 단일 회귀 문제로 다시 정의. > 이미지의 픽셀에서 바로 bounding box cordinates 와 class probabilities 를 탐지하도록 함.
- YOLO 의 구조



  YOLO 는 여러 바운딩 박스와 클래스 확률을 동시에 예측하는데, 이미지 전체를 입력으로 탐지 성능을 end to end 로 최적화 한다.
- YOLO 의 장점
  1. Extremly Fast: 복잡한 파이프라인이 없기 때문에 매우 빠름
     - 기본 YOLO 네트워크: 초당 45 프레임

    - 기본 YOLO 네트워크: 초당 45 프레임
      real time 스트리밍에서도 25ms 이하의 latency 가짐
2. Global Reasoning
   이미지 전체를 한 번에 학습 및 추론하기 때문에 객체 간의 context 를 함께 고려할 수 있음
   <> R-CNN 이 region proposal 만 고려하기 때문에 back ground patches 를 객체로 잘못 인식하는 경우가 많음
   -> Fast R-CNN 보다 background error 를 절반 이하로 줄임
3. Generalization
   자연 이미지로 학습했더라도 artwork 같은 새로운 도메인에서도 좋은 성능
   YOLO 가 generalizable representaion 을 학습하기 때문

- YOLO 의 한계
  - small objects 를 localization하는데 어려움이 있음
  - state-of-art 모델들에 비해 세밀한 정밀도가 떨어짐

Unified Detection

YOLO 파이프라인

- 입력 이미지를 SxS 그리드로 나누기
- 어떤 객체의 중심이 특정 그리드 셀 안에 위치한다면 그 셀은 해당 객체를 탐지할 책임을 가짐
  각 그리드 셀은
  - B 개의 바운딩 박스
  - 각 박스의 confidence점수
  - C개의 조건부 클래스 확률 Pr(Class_i | Object) : 객체가 있을 때, 그 객체가 특정 클래스 i 일 확률를 예측
    - confidence 점수
      1. 해당 박스에 객체가 존재할 확률 Pr(Object)
      2. 예측된 박스와 실제 박스의 겹치는 정도 IOU(pred, truth)

$$\text{confidence} = Pr(Object) \times IOU_{pred}^{truth}$$

      따라서 객체가 없으면 이 값은 0이 된다. 객체가 있을 경우 예측된 박스의 정확도를 반영
    - 바운딩 박스 (x, y, w, h, confience)
      - **(x, y)**: 객체 중심 좌표 (해당 그리드 셀 내에서 상대적 위치)
      - **(w, h)**: 객체의 폭과 높이 (전체 이미지 크기에 대한 비율)
      - **confidence**: 예측된 박스와 실제 박스의 IOU를 나타내는 신뢰도

- Inference 시 클래스 확률과 박스 신뢰도를 곱하여 최종 클래스별 확률을 계산

$$Pr(Class_i|Object) \times Pr(Object) \times IOU_{pred}^{truth} = Pr(Class_i) \times IOU_{pred}^{truth}$$

최종 클래스별 확률

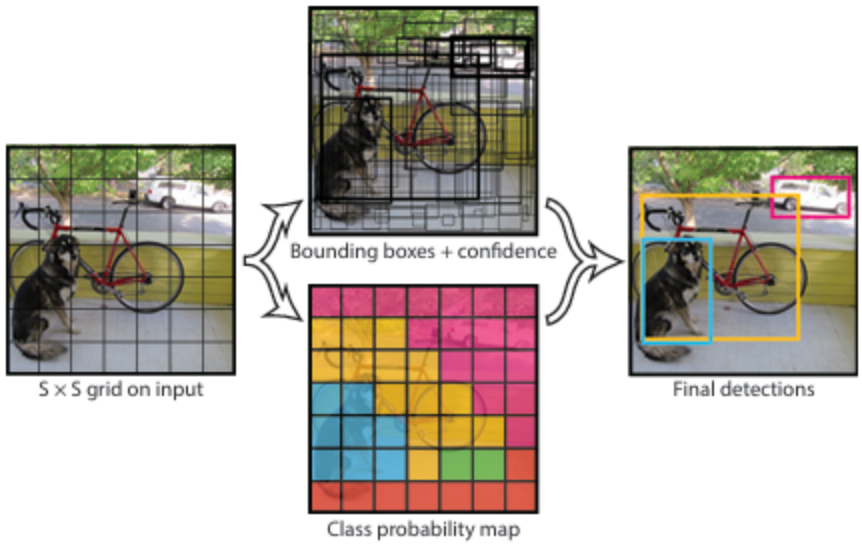


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

-> 예측된 결과는 SxSx(Bx5 + C) 형태의 텐서로 표현

1. **Network Design**

- 합성곱 신경망(CNN) 으로 구현, PASCAL VOC 데이터셋을 기준으로 학습과 평가
- 네트워크의 초기 합성곱 계층은 이미지의 특징(feature)을 추출하고, 완전 연결층(fully connected layer)은 **출력 확률과 좌표(prediction tensor)** 를 계산
- GoogLeNet 모델 기반
- 24개의 합성곱 층(convolutional layers)
- 2개의 완전 연결층(fully connected layers)

으로 구성. 대신 Inception 모듈 대신, 1×1 축소 계층(reduction layer) 과 3×3 합성곱 계층을 번갈아 사용합니다.

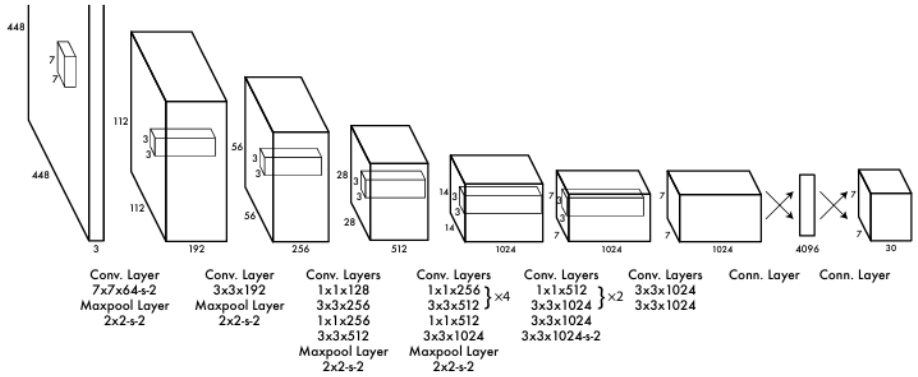


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

2. **Training

- 사전 학습

- IaageNet 1000-class 데이터셋에서 사전 학습

- 처음 20개 합성곱 층을 사용

- 1주일간 학습 → Top-5 정확도 88%

- Darknet 프레임워크로 구현

이후 탐지 작업을 위해

- 4개의 합성곱층, 2개의 완전 연결층 추가

- 입력 크기 또한 448x448 로 확대

- 학습 시 prediction 방식

- 마지막 출력층: 클래스 확률(class probabilities) **&** 바운딩 박스 좌표(bounding box coordinates) **를 예측**

(x, y, w, h)은 [0,1] 사이로 정규화,

(x, y)는 셀 내부의 상대 좌표, (w, h)는 전체 이미지 비율로 표현.

- 활성화 함수 Leaky ReLU

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

- Loss Function

- Sum Squared Error 사용

- average precision 과 완전히 일치하지 않기 때문에 조정 과정을 거침

1. 좌표 손실(λ_coord = 5) **↑ -> 바운딩 박스 좌표 정확도 향상**

2. 객체가 없는 셀 손실(λ_noobj = 0.5) **↓ -> 배경에서의 불필요한 confidence 감소**

3. 작은 박스와 큰 박스 오차 균형 조정 -> w, h 대신 √w, √h 를 사용해 큰 박스의 오차 영향을 줄임

==3. Inference (테스트 시)==

한 번의 네트워크 실행으로 전체 이미지의 탐지를 수행
→ 실시간 추론 가능 (classifier 기반 방법과 달리 반복 실행 불필요)
- 이러한 구조는 spatial diversity 를 보장하지만 경계 근처의 큰 객체를 여러 셀이 동시에 감지하는 문제가 생김
→ None-Max Suppression (NMS) 로 중복 박스 제거
→ mAP 향상 효과
==4. Limitations of YOLO==
- 공간적 제약(spatial constraint)**
- 각 셀은 2개의 박스, 1개의 클래스만 예측 가능
- 인접하거나 작은 객체를 동시에 탐지하기 어려움. (ex: 새 떼, 작은 물체 집단 등)
- aspect ratio 를 가지거나 새로운 형태의 객체에 대한 일반화 부족
- localization 에 약함
- 이런 문제들이 IOU 에 크게 영향을 준다.

Comparison to Other Detection Systems

전통적인 파이프라인

1. 이미지에서 특징추출 – Haar, SIFT, HOG, CNN 등
2. 분류기또는 로컬라이저 적용
3. 슬라이딩 윈도우나 region proposal 방식으로 탐지

- DPM
  - 전통적인 슬라이딩 윈도우 기반 탐지
  - 특징 추출 → 분류 → 바운딩 박스 예측 → 후처리(NMS) 등의 단계가 모두 분리되어 있음
  - YOLO는 모든 단계를 **하나의 CNN으로 통합**, 특징 추출과 박스 예측을 end-to-end로 수행
- R-CNN
  - Selective Search 를 통해 region proposal 방식 사용
  - 단계가 많고 독립적으로 조정되어야 함
  - 추론 시간 또한 매우 느림
    YOLO 는 R-CNN 과 달리 절차를 간소화하고 하나의 네트워크가 바로 위치와 클래스 확률을 동시에 예측하므로 시간 효율적
- Other Fast Detectors (Fast/Faster R-CNN)
  - R-CNN의 속도를 개선하기 위해 CNN feature map을 공유하거나, region proposal을 네트워크가 직접 예측
  - 여전히 real time 수준은 아님
  - YOLO 는 이런 구조를 아예 제거하여 근본적으로 빠름
- Deep Multibox
  - CNN을 이용해 후보 영역(region of interest) 예측
  - YOLO는 이와 달리 완전한 통합형 탐지 시스템(complete detection system)

    - MultiBox는 단일 객체 탐지에 그치지만 YOLO는 multi-object를 동시에 탐지 가능
- OverFeat
    - CNN을 사용하여 localization 과 detection 을 수행하지면 그래도 disjoint system 임
    - 슬라이딩 윈도우 방식을 사용하고 전체 context 를 고려하지 못함
    - YOLO 는 이미지를 한 번에 처리해서 global context 를 이해할 수 있다
- MultiGrasp
    - MultiGrasp는 그리드 기반 회귀 로 grasp 위치를 예측하는 방식
    - YOLO 와 비슷한 형태를 가지지만 grasp detection 은 object detection 보다 훨씬 단순한 문제
    - YOLO 가 Multigrasp 의 일반적이고 확장된 형태의 모델이라고 할 수 있다.

Experiments

PASCAL VOC 2007 데이터셋 사용

- YOLO는 background false positives 을 줄이는 데 강점
- Fast R-CNN은 탐지 결과를 rescore 하는 데 활용
  PASCAL VOC 2012 데이터셋 사용

1. Comparison to Other Real-Time System
   기존의 실시간 탐지 연구들은 대부분 기존 파이프라인을 빠르게 만드는 데 집중
   -> DPM(Deformable Parts Model)의 GPU 버전은 30Hz나 100Hz로 동작하지만, 정확도가 낮음
    - YOLO는 이보다 훨씬 높은 mAP를 가지면서도 실시간 성능을 유지 가능.
        - PASCAL VOC 2007에서 52.7% mAP
        - 초당 155 프레임(fps)
          를 달성하여 기존 실시간 탐지기보다 두 배 이상 정확하다.
    - Fast/Faster R-CNN은 높은 정확도를 보이지만 실시간 속도에는 한참 못 미침
2. VOC 2007 Error Analysis
   YOLO와 Fast R-CNN의 오류 특성
    - Fast R-CNN은 배경 오류(background error)
    - YOLO는 위치 오류(localization error)
      가 많음
3. Combining Fast R-CNN and YOLO
   YOLO의 장점(낮은 배경 오탐)을 이용해 Fast R-CNN의 단점을 보완하기
   -> YOLO로 Fast R-CNN의 탐지 결과 중 배경일 가능성이 높은 것을 제거하면 성능이 크게 향상됨
    - 단순한 앙상블 효과 가 아니라 YOLO와 R-CNN이 서로 다른 유형의 오류를 보완하기 때문에 가능.
    - 두 모델을 함께 사용할 경우에도 전체 속도에 큰 손실은 없음
4. VOC 2012 Results
    - VOC 2012 테스트 세트에서 YOLO는 57.9% mAP를 기록
    - 병목이 되는 작은 객체들에서 어려움을 겪기 때문에 state of art VGG 기반 CNN 보다 낮음

    - bottle, sheep, tv/monitor 등에서는 R-CNN보다 8~10% 낮은 점수를 보이지만 cat, train 같은 큰 객체에서는 더 높은 성능

5. Generalizability - Person Detection in artwork
   YOLO의 일반화 능력을 평가하기 위해 예술 작품(artwork) 에서 인물 탐지 성능을 실험
    - R-CNN은 VOC 2007에서는 높은 AP를 보이지만 예술 작품으로 도메인이 바뀌면 성능이 크게 하락 -> 이는 R-CNN의 Selective Search가 natural 이미지용으로 최적화 되어 있기 때문
    - DPM 은 객체의 모양과 배치를 모델링하기 때문에 예술 작품에서도 AP가 덜 떨어짐
    - YOLO 는 예술 작품 데이터에서도 다른 탐지기보다 성능 저하가 적음. 픽셀 단위에서는 달라도 전반적인 구조가 비슷한 예술 이미지에서 적절한 바운딩 박스를 예측할 수 있기 때문

Real World Detection In the Wild

YOLO를 웹캠에 연결해 실제 환경에서도 실시간 성능이 유지되는지 실험

- 카메라로부터 이미지를 받아오고 탐지 결과를 화면에 표시하는 시간을 모두 포함해도 YOLO는 실시간으로 작동
- YOLO는 개별 이미지를 하나씩 처리하지만, 웹캠에 연결하면 객체가 움직이거나 모양이 변할 때 이를 연속적으로 탐지하기 때문에 마치 추적 시스템처럼 작동한다.