SSD

Single Shot MultiBox Detector

▼ 1. 어떤 문제를 풀고자 했는가? (Abstract)

💡 SSD : 단일 깊은 신경망(single deep neural network)를 이용한 객체 탐지 기법
→ Faster R-CNN과 같은 2-stage detecter의 정확도를 가지면서 YOLO와 같은 1-stage detecter의 성능을 가짐
- 예측 단계에서 디폴트 박스 내 객체 카테고리 존재 여부를 점수화
- 디폴트 박스가 객체 모양과 더 일치하도록 조정
- 다양한 크기의 객체를 탐지하도록 서로 다른 해상도를 갖는 여러 피처맵의 예측 결과 활용!
- 영역 추정 단계와 반복적인 피쳐 샘플링을 제거하여 모든 연산을 단일 네트워크로 수행!

▼ 2. 어떤 동기/상황/문제점에서 이 연구가 시작되었는가? (Introduction)

💡 - 이전 객체 탐지 모델들의 접근법
1. 경계 박스 추정 2. 각 경계 박스마다 피처 재추출(resample) 3. 성능 좋은 분류기 적용
→ 정확도는 높지만 실시간 객체 탐지 서비스에 적용하기엔 속도 느림

In this 논문,,,
'경계 박스 추정을 위한 피처 재추출'을 하지 않는 딥러닝 기반 객체 탐지 모델 제안
- 경계 박스 추정, 피처 재추출 단계 제거 → 속도 향상
- 고정된 경계 박스를 활용해 클래스 레이블과 경계 박스 좌표 예측! 이때, 작은 합성곱 필터를 피처 맵에 적용
- 다양한 스케일의 피처 맵에서 여러 스케일로 예측
- end to end 방식, 입력 이미지 해상도가 낮더라도 높은 정확도 보임

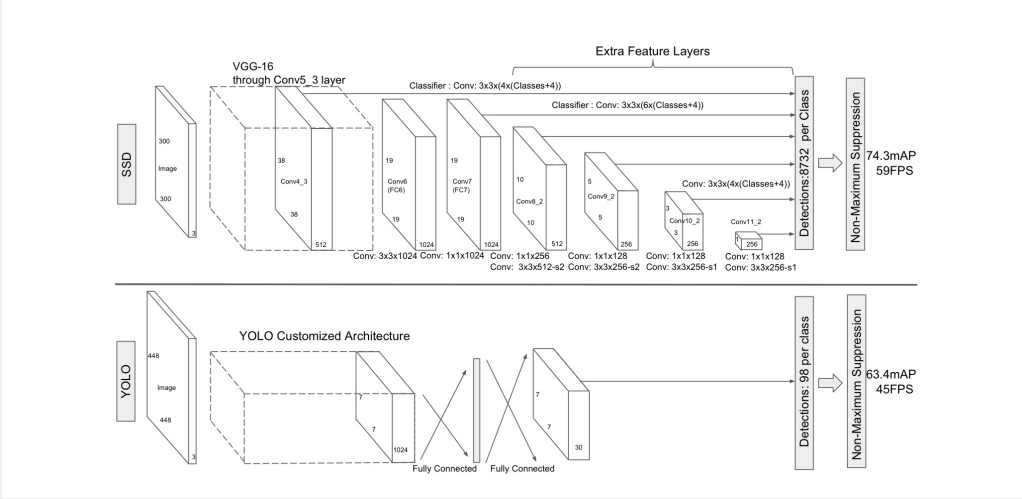
▼ 3. 이 연구의 접근 방법은 무엇인가?(Method)

1. **Model**

- Base Network로 **VGG-16**을 사용하고, 그 뒤에 또 다른 구조 덧붙임

1. Multi-scale feature maps for detection

- 기본 네트워크 다음에 여러 합성곱 피처 계층을 덧붙임,
- 계층의 크기가 차례대로 줄어들면서 다양한 스케일로 객체를 탐지함!



2. **Convolutional predictors for detection**

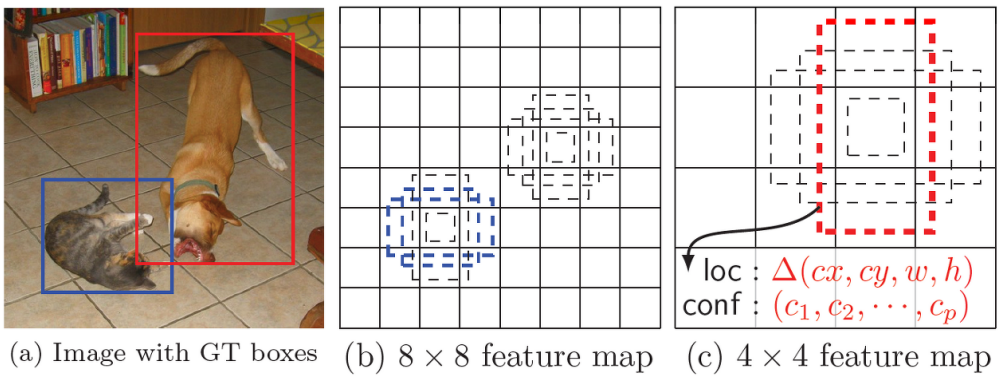
(1)의 피처 계층은 합성곱 필터를 통해 객체 탐지를 수행

- 각 피처 계층마다 '다양한 스케일과 가로세로 비율을 갖는 디폴트 박스의 좌표값' 과 '해당 디폴트 박스가 나타내는 객체의 존재 여부 신뢰도'를 예측

3. Default boxes and aspect ratios

- 다양한 피처맵마다 디폴트 박스와 피처 맵의 각 픽셀과 연관시킴
- 각 픽셀마다 경계 박스 좌표와 객체 클래스 신뢰도 점수(해당 디폴트 박스에 객체가 존재하는지 여부를 점수화)를 구함

- Framework



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

(A) 훈련 시, 입력 이미지와 각 객체 별 참 경계박스만 필요로 함
(B) 입력 이미지에 합성곱 연산을 해서 피처맵을 구함, 여러차례 합성곱 연산을 해서 다양한 크기의 피처 맵을 생성, 각 피처맵 마다 각 픽셀당 서로 다른 비율을 갖는 디폴트 박스가 존대
(C) 각 디폴트 박스마다 '경계 박스 좌표'와 '그 경계 박스의 객체 신뢰도 점수'를 예측, 훈련 단계에서 디폴트 박스를 참 경계박스와 매칭!

2. **Training**

💡 다른 객체 탐지 모델과의 훈련 방식의 차이점 : **참 경계 박스(ground truth boxes)가 필요**하다! → end-to-end 훈련에 손실함수와 역전파를 적용할 수 있음

1. Matching Strategy
- 참 경계 박스에 매칭되는 디폴트 박스를 찾는 방식으로 훈련 진행!
- **참 경계 박스와 IoU가 0.5보다 큰 디폴트 박스**를 찾음!

2. Training objective
- 각 객체 클래스마다 디폴트 박스와 참 경계 박스가 매칭되는지 여부를 x로 설정
- 전체 손실 함수:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

- 앞부분 = 신뢰도 손실 (confidence loss)
  → 각 객체 클래스마다 소프트맥스 함수로 구함!

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

- 뒷부분 = localization 손실
  → 예측한 디폴트 박스와 참 경계 박스 사이의 Smooth L1 손실

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

3. **Choosing scales and aspect ratios for default boxes**

- 합성곱 계층의 피처 맵을 이용해 다양한 객체 크기를 다룸!
- 모든 객체 스케일끼리 파라미터를 공유할 수 있음
- K개 디폴트 박스 크기는 피처맵마다 일정 → 큰 피처맵에서는 디폴트 박스가 작은 객체 탐지, 작은 피처맵에서는 큰 객체를 탐지

4. **Hard negative mining**
   - 참 경계 박스와 매칭된 디폴트 박스는 작으므로, 대부분 남은 디폴트 박스는 negative일거임 → 신뢰도 손실 점수를 기반으로 몇몇의 negative 샘플은 제외하고, negative와 positive의 비율을 맞춰줌!

▼ 4. 실험은 어떻게 이루어졌는가? (Experiments)

💡 **1. Pascal VOC 2007**
- 데이터가 많을수록 입력 이미지가 클수록 성능이 좋음!

2**. Model Analysis**
- 각 요소별로 성능 향상에 얼마나 도움되는지를 실험함
- (1) 데이터 증강은 SSD 성능 향상에 중요!
- (2) 다양한 모양의 Default box를 사용하는 게 성능 향상에 good!

-(3) resolution이 다른 다양한 출력 계층을 사용할 수록 성능이 좋다!

3.Inference time
- SSD는 디폴트 박스가 굉장히 많기 때문에 추론단계에서는 비최댓값 억제를 적용!
- SSD 모두 Faster R-CNN보다 속도 빠르고 mAP 높음!

-YOLO보다 속도는 느리지만 mAP 훨씬 높음!
- 빠른 기본 네트워크를 사용한다면 실시간 탐지가 가능한 수준!

▼ 5. 결론 및 요약 (Conclusion)

💡 - 여러 객체 클래스를 탐지할 수 있는 single-shot 객체 탐지기
- 여러 스케일의 합성곱 계층 피처 맵마다 경계 박스를 예측 → 다양한 크기를 갖는 객체 효율적으로 탐지 가능!
- 성능 good, 간단한 구조이기 때문에 더 큰 모델 시스템의 구성 요소로 사용 가능!