Faster R-CNN & YOLO

날짜	@2025년 3월 27일
분야	분석
주차	과제

Faster R-CNN

1. 개요

기존 객체 탐지 모델들은 영역 추정(region proposal) 알고리즘을 사용하여 객체의 위치를 예측해왔습니다. SPP-net과 Fast R-CNN은 속도를 크게 향상시켰지만, 여전히 영역 추정 단계에서 병목(bottleneck) 현상이 발생하는 문제가 있었습니다.

Faster R-CNN은 이러한 문제를 해결하기 위해 **영역 추정 네트워크(Region Proposal Network, RPN)** 를 도입하였습니다. RPN은 객체 탐지 네트워크와 합성곱 피처를 공유하여, 영역 추정을 거의 비용 없이(cost-free) 수행합니다. 또한, 객체의 경계 박스와 객체 존재 여부를 동시에 예측할 수 있도록 설계되었으며, end-to-end로 훈련할 수 있습니다.

Faster R-CNN은 깊은 합성곱 신경망을 활용하면서도 실시간에 가까운 속도를 보이며, 2015년 **ILSVRC와 COCO 대회에서 1위를 차지**하는 등 우수한 성능을 입증하였습니다.

---

2. 기존 객체 탐지 방식과 한계

- **R-CNN**: 객체 탐지를 위해 사전에 영역 후보(region proposals)를 생성한 후, 각 영역을 CNN을 사용하여 개별적으로 분류 → 속도가 매우 느림
- **Fast R-CNN**: 피처 맵을 공유하여 속도를 향상 → 하지만 여전히 영역 추정을 선택적 탐색(selective search) 방식으로 수행하여 병목 현상이 존재
- **선택적 탐색과 EdgeBox**:
  - 선택적 탐색은 CPU에서 수행되며, 이미지당 2초가 걸려 속도가 느림
  - EdgeBox는 0.2초로 개선되었지만, 여전히 빠른 실시간 처리를 어렵게 만듦

이러한 문제를 해결하려면 영역 추정을 GPU에서 수행해야 하며, Faster R-CNN은 이를 위해 RPN을 도입했습니다.

---

3. Faster R-CNN의 구조

Faster R-CNN은 두 가지 주요 모듈로 구성됩니다.

1. **RPN(Region Proposal Network)**:
   - 입력 이미지에서 합성곱 신경망을 사용해 영역 후보를 생성
   - 영역을 예측하는 동시에 객체 존재 여부를 점수로 예측
   - 슬라이딩 윈도우 기반으로 피처 맵을 활용
2. **Fast R-CNN 기반 객체 탐지기**:
   - RPN이 생성한 영역 후보를 기반으로 Fast R-CNN을 적용하여 최종 객체 탐지 수행

이 과정에서 RPN과 Fast R-CNN은 **합성곱 피처 맵을 공유**하여 연산량을 크게 줄였습니다.

---

4. RPN (Region Proposal Network)

4.1 개념

- 영역 추정을 CNN 기반으로 수행하여 속도를 획기적으로 개선
- 피처 맵을 기반으로 **슬라이딩 윈도우 방식**을 적용하여 영역을 예측
- **경계 박스와 객체 존재 여부를 동시에 예측**하는 네트워크

4.2 앵커 박스(Anchor Box)

- 기존의 객체 탐지 모델들은 다양한 크기의 객체를 다루기 위해 **이미지 피라미드**나 **필터 피라미드** 방식을 사용 → 속도 저하
- Faster R-CNN은 **앵커 박스(anchor box)** 개념을 도입하여 다양한 크기와 비율을 고려한 영역 후보를 생성
  - 일반적으로 **9개의 앵커 박스(k=9)** 사용 (3가지 크기 × 3가지 가로세로 비율)
  - 피처 맵의 각 위치에서 9개의 앵커 박스를 생성 → 다양한 크기의 객체를 효과적으로 탐지 가능

---

5. Loss Function (손실 함수)

RPN은 두 가지 손실 함수를 사용하여 훈련됩니다.

1. **이진 분류 손실(Binary Classification Loss, L_cls)**
   - 앵커 박스가 객체를 포함하는지 여부를 판단하는 손실
   - IoU(Intersection over Union) 기준으로 **0.7 이상 → Positive, 0.3 이하 → Negative**
2. **경계 박스 회귀 손실(Bounding Box Regression Loss, L_reg)**
   - 앵커 박스 좌표를 실제 객체 경계 박스와 일치하도록 조정하는 손실

최종 손실 함수는 다음과 같이 정의됩니다.

$L = L_{\text{cls}} + \lambda L_{\text{reg}}$

$L = L_{cls} + \lambda L_{reg}$

---

6. Faster R-CNN의 장점

1. **연산량 감소 및 속도 향상**
   - RPN과 Fast R-CNN이 피처 맵을 공유하여 연산량 절감
   - GPU에서 영역 추정을 수행하여 CPU 기반 선택적 탐색보다 훨씬 빠름
2. **객체 탐지 정확도 향상**
   - 앵커 박스를 활용하여 다양한 크기와 비율의 객체를 효과적으로 탐지
   - end-to-end 훈련으로 최적화 수행 가능
3. **실제 응용 가능성**
   - PASCAL VOC 데이터셋에서 선택적 탐색을 적용한 Fast R-CNN보다 높은 정확도를 기록
   - 깊은 신경망(VGG-16) 사용 시에도 **5 fps 속도** 달성 → 실시간 응용 가능

---

7. 결론

Faster R-CNN은 영역 추정과 객체 탐지를 통합한 최초의 end-to-end 모델로, **빠른 속도와 높은 정확도를 동시에 제공**하는 혁신적인 접근법입니다. 특히, RPN을 도입하여 기존 선택적 탐색 방식의 병목 현상을 해결하였으며, 앵커 박스를 활용하여 다양한 크기의 객체를 효과적으로 탐지할 수 있습니다.

Faster R-CNN은 **ILSVRC 및 COCO 2015 대회에서 1위를 차지**하며 우수한 성능을 입증하였으며, 이후 Mask R-CNN, YOLO, RetinaNet 등 다양한 객체 탐지 모델에 영향을 미쳤습니다.

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

1. 개요

기존 객체 탐지 모델(R-CNN, DPM)은 **classifier 기반 탐지 방식**을 사용했습니다. 하지만 YOLO(You Only Look Once)는 **객체 탐지를 회귀 문제(regression)로 변환**하여 **단일 신경망(end-to-end)** 으로 최적화할 수 있도록 설계되었습니다.

- 기존 2-stage 모델(R-CNN, Fast R-CNN 등)은 **Region Proposal → Classification** 구조로 되어 있어 속도가 느렸음.
- YOLO는 단일 CNN을 사용하여 **이미지를 한 번만 통과**시켜 **bounding box와 class 확률을 동시에 예측**하는 방식으로 설계됨.

- Localization(객체 위치 예측)이 다소 부정확할 수 있지만, **속도와 mAP(mean Average Precision)에서 기존 모델 대비 뛰어난 성능**을 보임.

---

2. 기존 객체 탐지 모델과 YOLO의 차이점

- **DPM (Deformable Parts Model)**
  - Sliding window 방식 사용 → **모든 위치에서 탐지**를 수행하여 연산량이 많음.
- **R-CNN (Region-based CNN)**
  - Selective Search 기반 **Region Proposal** 방식 도입 → **2-stage 구조** (Region Proposal → Classification).
  - 하지만, **속도가 느리고 최적화가 어려움**.
- **YOLO (You Only Look Once)**
  - 2-stage가 아닌 **1-stage 탐지기**로 동작.
  - **객체 탐지를 회귀(regression) 문제로 변환**, classification을 따로 수행하지 않음.
  - Sliding window나 Region Proposal 없이 **이미지를 한 번만 CNN에 통과시켜 전체적으로(global) 탐지** 가능.
  - **NMS(Non-Maximum Suppression)** 을 사용하여 최종 탐지 결과 결정.

---

3. YOLO의 동작 방식

1. **S x S Grid 생성**
   - 입력 이미지를 **S × S 크기의 격자(grid)로 나눔**.
   - 각 grid cell은 **B개의 bounding box와 신뢰도(confidence score)를 예측**.
   - 신뢰도 점수 = 객체 포함 여부 + bbox 정확도 (IoU 기반).
2. **Bounding Box + Confidence & Class Probability 예측**
   - 각 bbox는 **(x, y, w, h, confidence score)** 5가지 정보를 예측.
   - 추가적으로, **C개의 클래스 확률(class probabilities)** 도 함께 예측.
   - 최종적으로 각 grid cell은 **(5 × B + C)개의 값을 예측하는 output tensor**를 생성.
3. **최종 탐지 결과 결정**
   - 신뢰도(confidence score)와 class probability를 곱하여 최종 점수 산출.
   - **NMS(Non-Maximum Suppression)** 을 사용하여 중복된 박스 제거 및 최종 결과 확정.

---

4. YOLO 네트워크 구조

- **24개의 합성곱 층(convolution layers) + 2개의 완전 연결층(fully connected layers)로 구성**.
- **1 × 1 conv layer**를 활용하여 feature map의 차원을 줄여 연산량 감소.
- 최종 output tensor의 크기는 **S × S × (5 × B + C)**.
  - 기본적으로 **7 × 7 × 30** (B=2, C=20).
  - 클래스 수(C)나 bbox 개수(B)를 조정하여 fine-tuning 가능.
- **ReLU 대신 Leaky ReLU 사용**하여 작은 값도 잘 반영할 수 있도록 설계.

---

5. Loss Function (손실 함수)

- 객체가 있는 bbox의 위치 정보에는 **5배 가중치**를 부여하여 중요도를 높임.
- 객체가 없는 bbox에 대한 손실은 **0.5배 가중치**를 줘서 불필요한 오차를 줄임.
- Localization, confidence, class 예측 모두를 고려하는 복합적인 SSE(Sum of Squared Errors) 기반 손실 함수 사용.

---

6. YOLO의 장점과 단점

✅ 장점

- **빠른 속도**: 이미지 한 번만 CNN에 통과시키므로 실시간 탐지가 가능.

- **Global Context 활용 가능**: 전체 이미지를 한 번에 보고 탐지 → 객체 간 관계 고려 가능.
- **배경에 대한 false positive(Type 1 error) 발생 가능성이 낮음**.
- **train과 다른 도메인의 test 이미지에도 성능이 잘 유지됨 (generalization 가능)**.

❌ **단점**

- **Localization 정확도가 다소 떨어짐**: 작은 객체를 탐지하는 데 어려움.
- **Type 2 error(객체를 놓치는 오류)가 발생할 가능성이 있음**.
- **여러 개의 객체가 가까이 있을 경우, bbox가 겹치는 문제 발생 가능**.

---

7. 결론

YOLO는 **1-stage 탐지기**로, 기존 R-CNN 계열 모델보다 **훨씬 빠르면서도 높은 정확도를 유지하는 객체 탐지 모델**입니다.

- Classification 기반 탐지 방식이 아닌 **회귀(regression) 방식으로 객체 탐지** 수행.
- **단일 CNN을 사용하여 end-to-end 학습 가능**.
- Sliding window나 Region Proposal 없이, **이미지 전체를 한 번만 보고 탐지**.
- **빠른 속도, 높은 mAP, 낮은 false positive** 등의 장점을 가짐.