Faster R-CNN은 2015년 샤오칭 렌(Shaoqing Ren)이 제안한 객체 탐지(Object Detection) 모델로, Fast R-CNN을 개선하여 실시간 객체 탐지 모델

주요 특징 및 장점:

- Fast R-CNN의 병목 현상 개선: Fast R-CNN의 객체 영역 추정(Region Proposal) 단계에서 발생하는 병목 현상을 개선하여, PASCAL VOC 챌린지와 COCO 챌린지에서 우승

성능

- GPU 환경에서 깊은 VGG-16 모델 사용 시 5 FPS의 성능을 달성
  PASCAL VOC 2007 데이터셋에서 mAP 78.8%를 기록
- End-to-End 학습: 전통적인 파이프라인 방식과 달리, 중간 단계 처리나 변환 없이 입력과 출력 사이의 관계를 직접 학습(End-to-End) 하여 결과를 예측합니다. 이를 통해 모델 구성과 훈련이 간단하고 효율적
- 높은 유연성: 다른 모델과의 조합을 통해 성능을 높일 수 있으며, 다양한 변형 모델에서도 우수한 성능을 보임
- 지속적인 활용: 객체 탐지 분야에서 여전히 활발히 사용되며, 다른 딥러닝 기반 객체 탐지 모델들과 함께 연구 및 응용되고 있음
- Faster R-CNN은 Fast R-CNN의 개선 모델
- Fast R-CNN은 R-CNN의 개선 모델

R-CNN

- R-CNN은 2013년 로스 기르쉬크(Ross Girshick)가 제안한 딥러닝 기반 객체 탐지 모델
- 영역 제안(Region Proposal), 특징 추출(Feature Extraction), 서포트 벡터 머신(SVM)을 사용하여 객체 탐지를 수행
- 영역 제안과 특징 추출(합성곱 모델, CNN)을 결합하기 때문에 R-CNN (Regions-based CNN) 이라고 불림
- R-CNN은 객체 탐지를 위해 "선택적 탐색 (Selective Search)" 알고리즘을 사용하여 영역 제안을 수행
- 선택적 탐색은 규칙 기반 알고리즘으로, 이미지 내에서 객체가 있을 법한 후보 영역들을 생성합
- 후보 영역 생성 과정:
  - 군집화 (Grouping): 입력 이미지에서 인접 픽셀 중 색상, 질감, 밝기, 크기 등 유사한 특징을 갖는 영역들을 묶어 군집화
  - 반복적 확장: 군집화된 영역에 대해서도 위와 같은 군집화 과정을 반복하여 군집을 확장
  - 영역 추출: 최종적으로 확장된 군집들을 통해 객체가 존재할 가능성이 있는 영역들을 추출

- 이미지는 R-CNN의 선택적 탐색 과정을 시각적으로 보여줍니다.

그림 9.2 R-CNN의 선택적 탐색 과정

- R-CNN은 이미지 내에서 객체가 있을 법한 위치를 찾기 위해 "선택적 탐색"이라는 알고리즘을 사용하며, 이 알고리즘은 이미지의 픽셀들을 유사한 특징을 기준으로 묶고 확장하여 후보 영역을 찾아내는 방식
- 특징 추출 및 분류: 선택적 탐색 알고리즘으로 생성된 약 2,000개의 영역 제안 각각에 대해 AlexNet을 적용하여 4,096 크기의 특징 벡터를 추출합니다.(CNN사용) 그 후, 서포트 벡터 머신(SVM) 분류기에 입력하여 객체 분류를 수행
- 비최댓값 억제 (NMS): SVM 분류를 통과한 2,000개의 영역은 클래스와 확률값을 갖게 되지만, 동일한 객체에 대해 겹치는 영역들이 발생할 수 있습니다. NMS(Non-Maximum Suppression)는 이러한 겹치는 영역 중 가장 높은 확률값을 가진 영역만 남기고 나머지는 제거하여, 하나의 객체에 하나의 영역만 남도록 하는 후처리 기법
- 박스 회귀 (Box Regression): NMS를 통해 남겨진 영역은 객체의 대략적인 위치만 표현하므로, 정확한 위치를 나타내기 위해 박스 회귀를 사용. 박스 회귀는 후보 영역이 실제 객체 영역(Ground Truth)과 일치하도록 학습하여 영역의 위치와 크기를 미세 조정

> - R-CNN은 선택적 탐색 -> 특징 추출 -> SVM 분류 -> NMS -> 박스 회귀 단계를 거쳐 객체 탐지를 수행
> - NMS는 겹치는 영역 중 가장 좋은 영역만 남기는 역할을 합니다.
> - 박스 회귀는 객체의 위치와 크기를 정확하게 조정하는 역할을 합니다.

Fast R-CNN

- Fast R-CNN은 Ross Girshick이 2015년에 제안한 R-CNN의 개선 모델입니다.

- R-CNN의 문제점:
  - 많은 연산량과 메모리 사용: 2,000여 개의 후보 영역을 모두 CNN에 통과시키기 때문에 연산량과 메모리 사용량이 많습니다.
  - 분리된 학습 단계: 영역 제안, CNN, SVM이 각각 분리되어 학습되므로, 종단간 학습(End-to-End Learning)이 불가능하고 성능 향상에 제약이 있습니다.
- Fast R-CNN은 이러한 R-CNN의 단점을 개선하여 더 빠르고 정확한 객체 인식을 가능하게 합니다.

핵심 요약:

- Fast R-CNN은 R-CNN보다 빠르고 정확합니다.
- R-CNN은 연산량이 많고, 종단간 학습이 불가능한 단점이 있습니다.
- Fast R-CNN은 R-CNN의 이러한 단점을 개선했습니다.

간단히 말해, Fast R-CNN은 R-CNN의 느리고 비효율적인 부분을 개선하여 더 빠르고 정확하게 객체를 인식할 수 있도록 만든 모델

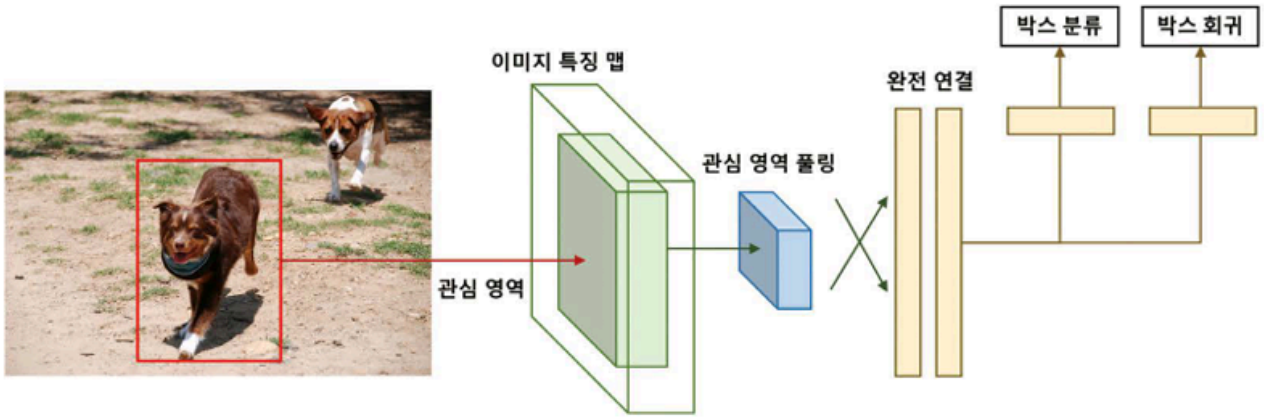

그림 9.3 Fast R-CNN 모델 구조

- Fast R-CNN은 입력 이미지를 사전 학습된 CNN에 통과시켜 특징 맵을 추출하고, 그 위에 관심영역(RoI, Region of Interest)를 투영
- RoI 풀링은 다양한 크기의 RoI를 고정된 크기의 특징 맵으로 변환하는 기법
- 고정된 크기의 특징 맵은 완전 연결 계층(fully connected layer)에 전달되어 객체 분류와 경계 상자 회귀(bounding box regression)를 수행
- Fast R-CNN은 특징 추출, RoI 풀링, 객체 분류, 박스 회귀를 종단간(end-to-end) 학습하여, 학습 단계를 간소화하고 성능을 향상

Faster R-CNN

- Faster R-CNN은 특징 추출, 영역 제안 네트워크(RPN), 관심 영역 풀링(RoI Pooling), 분류기로 구성
- 이미지에서 합성곱 신경망을 통과해 특징 맵을 추출
- 추출된 특징 맵은 영역 제안 네트워크와 Fast R-CNN에서 사용한 분류기에 각각 전달

- 영역 제안 네트워크로 전달된 특징 맵에서 관심 영역을 계산하고추출된 관심 영역은관심 영역 폴링을 적용해 분류기에서 객체를 검출

1. Faster R-CNN의 구성 요소:
   - 특징 추출: 입력 이미지에서 합성곱 신경망(CNN)을 통해 특징 맵을 추출
   - 영역 제안 네트워크 (Region Proposal Network, RPN): 특징 맵에서 객체가 있을 법한 후보 영역(관심 영역)을 생성
   - 관심 영역 풀링 (RoI Pooling): 다양한 크기의 후보 영역들을 고정된 크기로 변환하여 분류기에 입력할 수 있도록 함
   - 분류기: 고정된 크기로 변환된 후보 영역들을 입력받아 최종적으로 객체를 분류하고, 바운딩 박스 좌표를 미세 조정
2. R-CNN의 문제점과 RPN의 등장 배경:
   - R-CNN은 객체 탐지를 위해 가능한 모든 영역에서 특징 벡터를 추출하고 분류하는 비효율적인 방식을 사용하는데 이는 이미지 처리 속도가 느리고 연산량이 많다는 문제점
   - Faster R-CNN은 R-CNN의 문제점을 해결하기 위해, 특징 추출과 영역 제안을 통합한 RPN을 도입

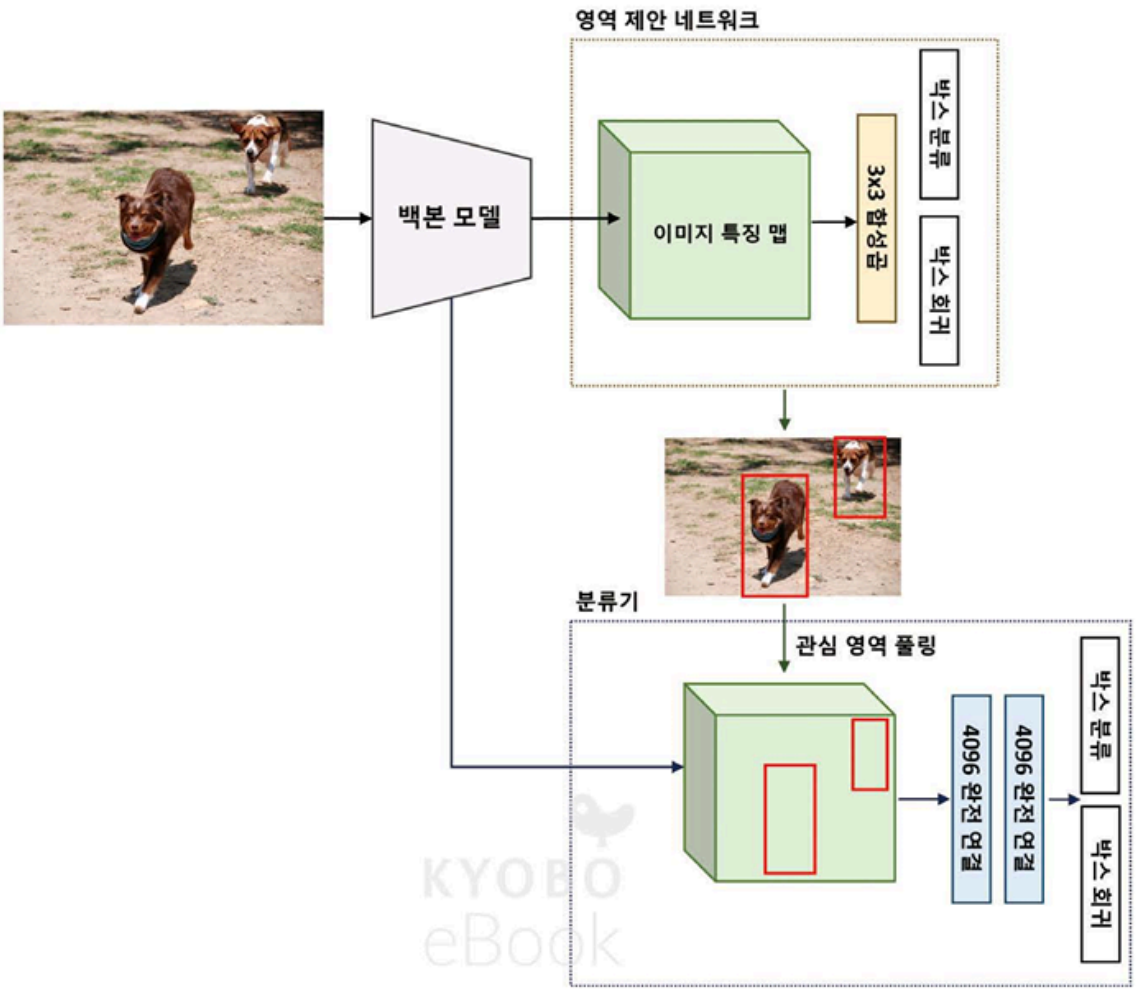

그림 9.4 Faster R-CNN 구조

RPN의 작동 방식

Pasted image 20250121234236.png

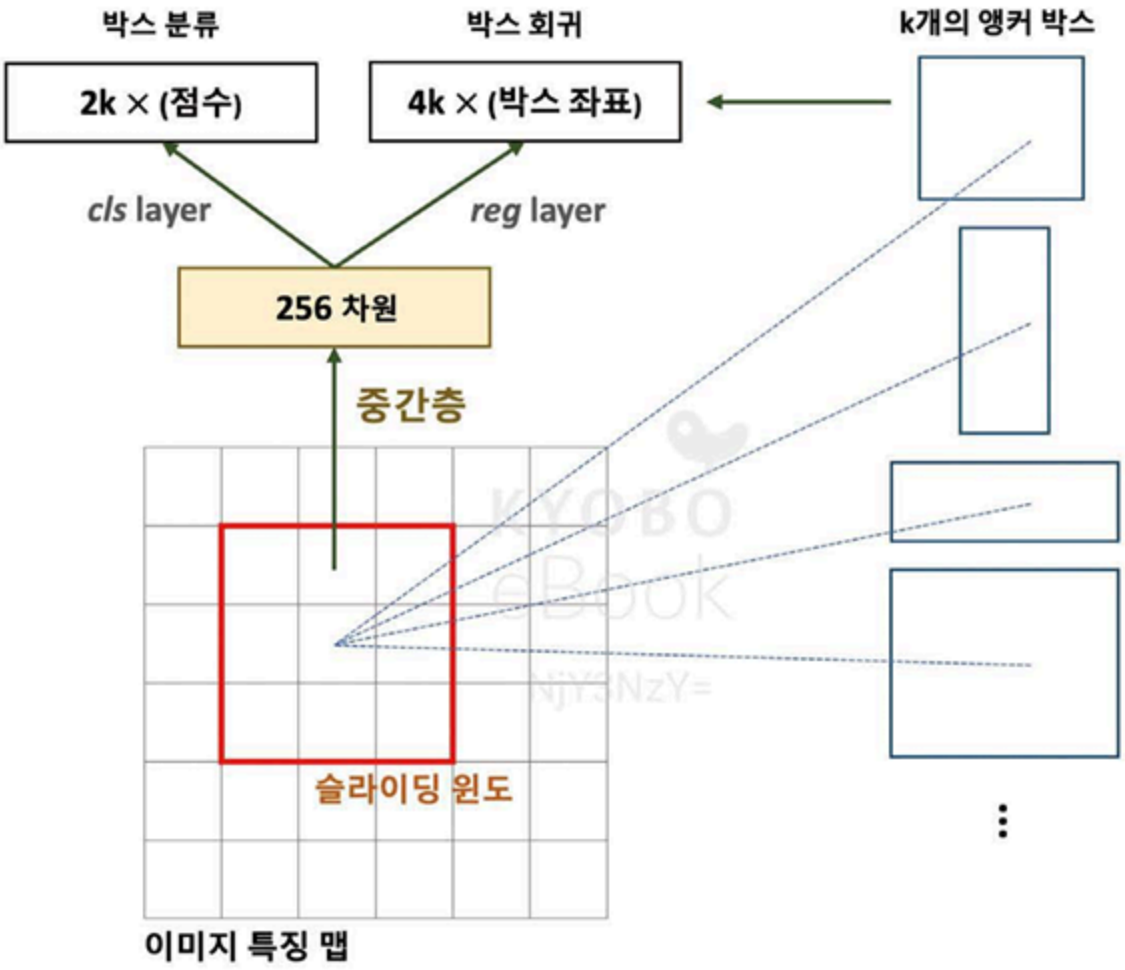

그림 9.5 영역 제안 네트워크의 구조

- 입력: RPN은 CNN을 통해 추출된 특징 맵을 입력으로 받음
- 슬라이딩 윈도우: 특징 맵 위를 일정 크기의 윈도우(슬라이딩 윈도우)가 이동하며 각 위치의 특징을 중간층(Intermediate layer)에 전달
- 중간층: 슬라이딩 윈도우의 특징은 256차원 또는 512차원의 벡터로 변환
- 앵커 박스: 각 슬라이딩 윈도우 위치마다 k개의 앵커 박스가 생성됩니다. 앵커 박스는 서로 다른 크기와 비율을 가진 미리 정의된 박스
- 박스 분류 계층(cls layer): 각 앵커 박스에 대해 객체인지 아닌지를 나타내는 점수(2k 점수 -각 앵커별 객체/배경 분류 점수)를 출력합니다.
- 박스 회귀 계층(reg layer): 각 앵커 박스에 대한 바운딩 박스 좌표(4k 좌표 - 각 앵커별 바운딩 박스 좌표 보정값)를 출력합니다.

RPN의 장점:

- GPU 연산: RPN은 GPU 연산이 가능하여 CPU를 사용하는 선택적 탐색 알고리즘보다 훨씬 빠름
- 효율성: 특징 추출과 영역 제안을 통합하여 R-CNN의 비효율성을 개선

앵커박스

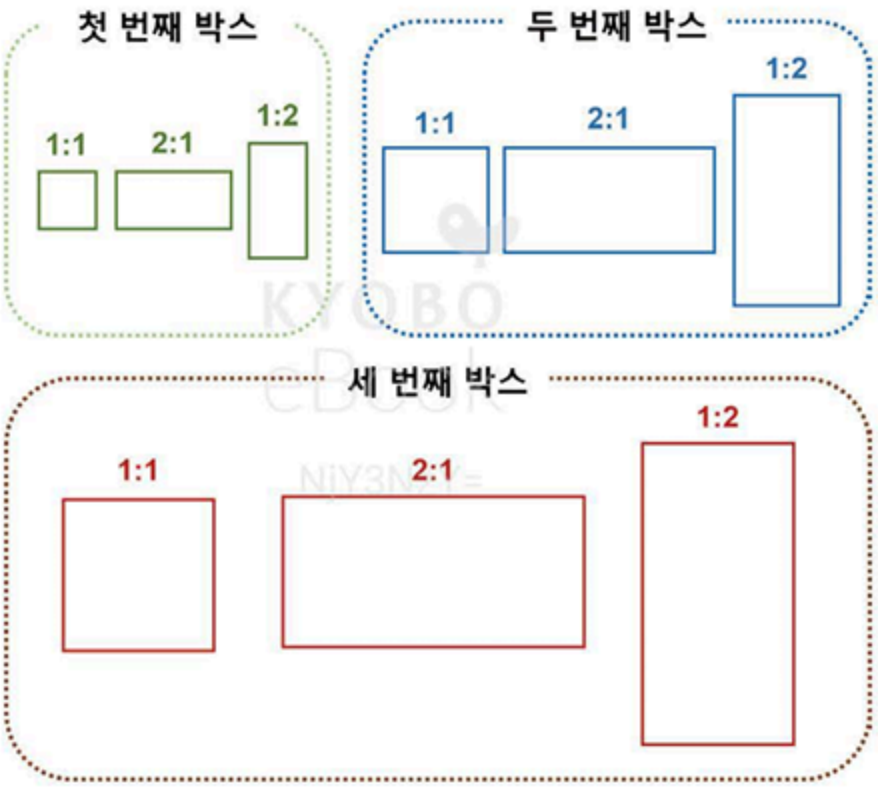

그림 9.6 앵커 박스 형태

- 앵커 박스는 객체의 위치와 크기를 예측하기 위해 사용되는 미리 정의된 사각형 영역
- 앵커 박스는 슬라이딩 윈도우와 함께 사용되며, 다양한 크기와 종횡비(aspect ratio)
- Faster R-CNN은 9개의 앵커 박스를 사용합니다 (3가지 크기 x 3가지 종횡비).
- 앵커 박스는 이미지 내에서 객체의 존재 여부와 객체의 크기 및 위치를 예측하는 데 사용됩니다.

앵커 박스 (Anchor Box) 정의

- 객체의 위치와 크기를 예측하기 위해 사용하는 사각형 영역
- 슬라이딩 윈도우와 함께 작동하여 이미지 내의 객체를 탐지
- 다양한 종류의 객체를 탐지하기 위해 여러 크기와 종횡비를 갖도록 정의
- 3가지 크기(작은 박스, 중간 박스, 큰 박스)와 3가지 종횡비(1:1, 2:1, 1:2)를 조합하여 총 9개의 앵커 박스를 보여줍니다.
  - 첫 번째 박스: 작은 크기, 3가지 종횡비 (1:1, 2:1, 1:2)
  - 두 번째 박스: 중간 크기, 3가지 종횡비 (1:1, 2:1, 1:2)
  - 세 번째 박스: 큰 크기, 3가지 종횡비 (1:1, 2:1, 1:2)

- Faster R-CNN은 128, 256, 512의 3가지 크기와 1:1, 2:1, 1:2의 3가지 종횡비를 조합하여 총 9개의 앵커 박스를 사용
- 예를 들어, 1,024개의 셀(특징 맵의 각 위치)을 추출했다면, 각 셀마다 9개의 앵커 박스가 생성되어 총 9,216(1024 x 9)개의 앵커 박스가 생성
- 각 앵커 박스에 대해 객체의 존재 여부와 객체의 크기 및 위치를 예측

박스 분류 (Box Classification)

- 박스 분류: 9개의 앵커 박스 각각에 대해 객체 존재 여부(객체 있음/없음)를 나타내는 점수를 계산 (출력: 18개 = 9개 앵커 박스 x 2개 클래스)
- 박스 회귀: 앵커 박스의 위치와 크기를 조정하여 객체의 정확한 위치를 예측(출력: 36개 = 9개 앵커 박스 x 4개 조정값(x, y, w, h))
- 1x1 합성곱: 박스 분류와 박스 회귀 모두 1x1 합성곱 연산을 사용하여 효율적으로 계산
- 조정값 계산: 앵커 박스의 위치와 크기 조정값(d)을 계산하기 위해 수식이 사용되며, 이 값들을 통해 조정된 앵커 박스(G)를 계산
- 지수 함수 사용: 박스 회귀에서 지수 함수를 사용하여 작은 값의 변화를 크게 반영하고, 다양한 크기와 비율의 객체를 탐지할 수 있도록 합니다.
- 목표: 각 앵커 박스에 객체가 존재하는지 여부(객체 있음/없음)를 판별
- 출력: 9개의 앵커 박스 각각에 대해 2개의 클래스(객체 없음, 객체 있음)에 대한 점수를 출력합니다. 따라서 총 18개(9 x 2)의 출력이 생성
- 계산:
  - 슬라이딩 윈도우의 크기가 H x W x C라면, 1x1x18 합성곱 연산을 적용하여 H x W x 18 크기의 특징 맵을 생성
  - 1x1 합성곱 연산을 사용하면 입력 이미지 크기와 상관없이 동작할 수 있으며, 한 번의 연산으로 H x W 개의 앵커 박스에 대한 예측을 수행

박스 회귀 (Box Regression):

- 목표: 앵커 박스의 위치와 크기를 조정하여 객체의 정확한 위치를 예측합
- 출력: 9개의 앵커 박스 각각에 대해 4개의 조정값(x, y, w, h)을 출력합니다. 따라서 총 36개(9 x 4)의 출력이 생성됩니다.

수식 9.1 앵커 박스 정보

$$P^i = \left(P_x^i, P_y^i, P_w^i, P_h^i\right)$$

수식 9.2 앵커 박스 조정값

$$d^i = \left(d_x(P^i), d_y(P^i), d_w(P^i), d_h(P^i)\right)$$

- 계산:
  - 박스 분류와 유사하게 1x1 합성곱 연산을 사용하지만, 36개의 차원을 갖도록 합니다.
  - 4개의 클래스는 각각 위치(x, y)와 크기(w, h) 값을 조정하는 데 사용됩니다.
  - 수식 9.1은 앵커 박스의 정보(P), 수식 9.2는 앵커 박스 조정값(d)

수식 9.3 최종 앵커 박스

$$\begin{aligned}\hat{G}_x^i &= P_w^i d_x(P^i) + P_x^i \\ \hat{G}_y^i &= P_h^i d_y(P^i) + P_y^i \\ \hat{G}_w^i &= P_w^i e^{d_w}(P^i) \\ \hat{G}_h^i &= P_h^i e^{d_w}(P^i)\end{aligned}$$

  - 수식 9.3은 조정값(d)을 사용하여 조정된 앵커 박스(G)를 계산하는 방법을 보여줌
    - $G_x = P_w d_x(P) + P_x$: 조정된 앵커 박스의 x 좌표
    - $G_y = P_h d_y(P) + P_y$: 조정된 앵커 박스의 y 좌표
    - $G_w = P_w exp(d_w(P))$: 조정된 앵커 박스의 너비
    - $G_h = P_h exp(d_h(P))$: 조정된 앵커 박스의 높이
  - $d_x(P)$와 $d_y(P)$ 계산에 P_w와 P_h를 곱하는 이유는 관심 영역의 크기가 다르기 때문에 비율을 맞추기 위함입니다.
  - $d_w(P)$와 $d_h(P)$ 계산에 지수 함수를 사용하는 이유는 작은 값의 변화를 크게 반영하여 미세한 조정을 가능하게 하고, 다양한 크기와 비율의 객체를 탐지할 수 있도록 하기 위함입니다.
- 결론적으로, Faster R-CNN은 앵커 박스를 기반으로 박스 분류를 통해 객체의 존재 여부를 판별하고, 박스 회귀를 통해 객체의 정확한 위치와 크기를 예측합니다. 이 과정에서 1x1 합성곱 연산과 수식, 지수 함수 등을 활용하여 효율적이고 정확한 계산을 수행합니다.

관심 영역 선별

- 관심 영역 선별은 앵커 박스 중 객체가 있을 확률이 높은 영역을 추려내는 과정입니다.
- 학습 시와 추론 시 RoI 선별 개수 및 기준이 다릅니다. (학습: 2,000개 → 1,000개, 추론: 1,000개 → 100개)
- RoI 선별 과정:
  1. 객체 점수 정렬
  2. 높은 점수 앵커 박스 추출 (학습: 1,000개, 추론: 100개)
  3. 박스 회귀 적용
  4. NMS 적용
  5. 최종 RoI 선정
- 학습 시 추가적인 RoI 선별 기준:
  - 가장자리 앵커 박스 제외

    - IoU 기반 긍정/부정 샘플 선별
- IoU(Intersection over Union)는 앵커 박스와 정답 박스 간 겹치는 비율을 나타내는 지표로, RoI 선별에 활용됩니다.

1. 관심 영역 선별 개요:

- 목적: RPN(영역 제안 네트워크)에서 생성된 수많은 앵커 박스 중 객체가 존재할 가능성이 높은 영역만을 선별하여 후속 처리(분류 및 박스 회귀)의 효율성을 높이는 것
- 학습 시 vs. 추론 시:
    - 학습 시: 일반적으로 2,000개의 앵커 박스 중 1,000개를 선별합니다.
    - 추론 시: 일반적으로 1,000개의 앵커 박스 중 100개를 선별합니다.
- 선별 방법: 앵커 박스에 대한 객체 점수, 박스 회귀, NMS(Non-Maximum Suppression) 등을 활용하여 최종적인 RoI를 선별합니다.

2. 관심 영역 선별 과정 (공통):

- 1단계: 박스 분류에서 계산된 점수 정렬: 앵커 박스마다 부여된 객체 점수(객체 존재 확률)를 기준으로 내림차순 정렬
- 2단계: 정렬된 객체 점수 값이 0.7(또는 특정 값) 이상인 앵커 박스 추출: 객체 점수가 높은 앵커 박스를 우선적으로 선별 (학습 시: 2000개 → 1000개, 추론 시: 1000개 → 100개)
- 3단계: 상위 N개의 앵커 박스에만 박스 회귀 적용: 선별된 상위 N개의 앵커 박스에 대해 박스 회귀를 적용하여 위치와 크기를 미세 조정
- 4단계: 비최댓값 억제(NMS) 연산 적용: 겹치는 영역이 많은 앵커 박스 중 가장 높은 점수를 가진 앵커 박스만 남기고 나머지는 제거
- 5단계: 남은 앵커 박스를 관심 영역으로 사용: NMS를 거친 앵커 박스를 최종적인 RoI로 사용

1. 학습 시 추가적인 관심 영역 선별 기준:

- 1단계: 특징 맵 가장자리의 윈도에 할당된 앵커 박스 제외: 이미지 가장자리에 걸쳐있는 앵커 박스는 유의미한 정보를 담고 있을 가능성이 적기 때문에 제외
- 2단계: IoU 기반 긍정 샘플 선별:
    - 실제 객체(Ground Truth) 박스와 앵커 박스의 IoU가 0.7 이상인 경우, 해당 앵커 박스를 긍정 샘플(객체가 존재하는 영역)로 간주
    - 만약 긍정 샘플로 선별된 앵커 박스가 없다면, 객체 점수가 가장 높은 앵커 박스를 긍정 샘플로 사용
- 3단계: IoU 기반 부정 샘플 선별: 정답 박스와 IoU가 0.3 이하인 앵커 박스를 부정 샘플(배경 영역)로 간주

2. IoU (Intersection over Union):

- 정의: 앵커 박스와 정답 박스 간 겹치는 영역의 비율을 나타내는 지표

- 계산: (앵커 박스와 정답 박스의 교집합 영역 넓이) / (앵커 박스와 정답 박스의 합집합 영역 넓이)
- 활용: RoI 선별 과정에서 긍정/부정 샘플을 판별하는 데 사용됩니다.
- 시각적 표현: 이미지 하단의 그림 9.7은 IoU 값에 따른 겹침 정도를 시각적으로 보여줌줌
- Faster R-CNN은 RPN에서 생성된 수많은 앵커 박스 중 객체 존재 확률이 높은 영역을 효율적으로 선별하기 위해 객체 점수, 박스 회귀, NMS, IoU 등의 방법을 활용합니다. 특히, 학습 시에는 IoU를 기반으로 긍정/부정 샘플을 추가적으로 선별하여 모델의 정확도를 높입니다.

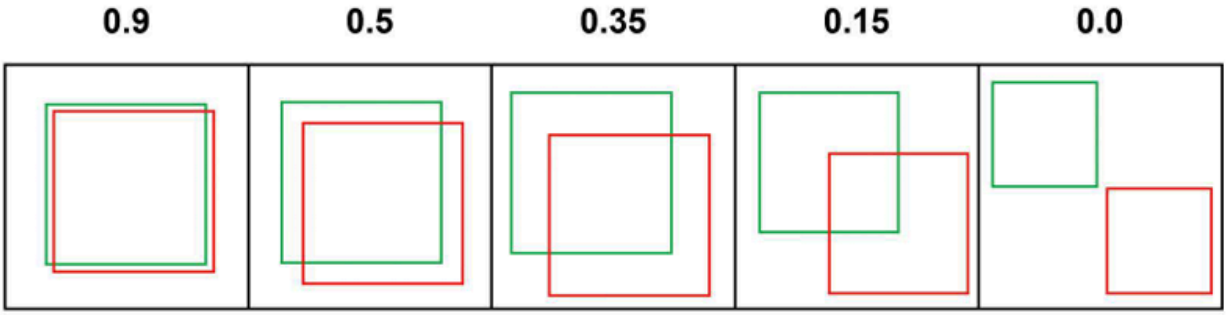

그림 9.7 IoU 지표

모델 학습 과정

- Faster R-CNN은 2단계 객체 탐지 모델입니다.
  - 1단계 (영역 제안 네트워크): 입력 이미지에서 객체가 있을 가능성이 있는 관심 영역(RoI)을 추출하여 후보 영역을 생성합니다.
  - 2단계 (객체 탐지 네트워크): 후보 영역에 대해 객체 탐지를 수행합니다.
- 관심 영역 풀링(RoI Pooling): 후보 영역들을 고정된 크기(7x7)로 변환하여 객체 탐지 네트워크에 입력합니다.
- Faster R-CNN의 손실 함수는 분류 손실 함수(L_cls)와 박스 회귀 손실 함수(L_reg)로 구성됩니다.
- 손실 함수의 균형을 맞추기 위해 하이퍼파라미터 λ를 사용합니다.
- 정규화 상수 N_cls와 N_reg는 각각 미니 배치 크기와 앵커 박스 개수를 의미합니다.

상세 요약:

1. Faster R-CNN의 2단계 구조:

- 1단계: 영역 제안 네트워크 (Region Proposal Network, RPN):
  - 입력 이미지에서 객체가 있을 가능성이 있는 영역, 즉 관심 영역(Region of Interest, RoI)을 추출합니다.
  - 이를 통해 객체가 있을 법한 후보 영역들을 생성합니다.
- 2단계: 객체 탐지 네트워크:
  - RPN에서 생성된 후보 영역들에 대해 객체 탐지를 수행합니다.

- 관심 영역 풀링(RoI Pooling)을 사용하여 후보 영역들을 고정된 크기로 변환합니다.
- 변환된 후보 영역들을 입력받아 객체를 분류하고, 경계 상자(bounding box)를 회귀합니다.

2. 관심 영역 풀링 (RoI Pooling):

- 목적: 다양한 크기의 후보 영역들을 고정된 크기(Faster R-CNN에서는 7x7)로 변환하여 객체 탐지 네트워크에 입력할 수 있도록 합니다.
- 과정:
  - N개의 특징 맵을 모두 결합하면 [N, 7, 7, 256] 또는 [N, 7, 7, 512] 크기의 텐서가 생성됩니다. (ZFNet 사용 시 256, VGG 사용 시 512)
  - 이 텐서를 평탄화(flatten)하면 [N, 25088] 또는 [N, 7, 7, 512] 크기의 벡터가 생성됩니다.

3. 객체 탐지 네트워크 출력:

- 박스 분류 계층 (cls layer): 각 후보 영역에 대한 클래스 예측 결과를 출력합니다. 클래스 개수가 C개라면 [N, C] 형태의 출력이 생성됩니다.
- 박스 회귀 계층 (reg layer): 각 후보 영역에 대한 경계 상자 조정 값을 출력합니다. 클래스마다 조정 값이 생성되므로 [N, C x 4] 형태의 출력이 생성됩니다.
- 최종 출력: Faster R-CNN 모델의 최종 출력값은 객체의 클래스 예측 결과와 경계 상자입니다.

4. Faster R-CNN 손실 함수:

Faster R-CNN은 2단계 구조를 통해 객체 탐지를 수행하며, RoI 풀링을 통해 다양한 크기의 후보 영역을 처리합니다. 손실 함수는 분류 손실과 박스 회귀 손실을 결합하여 모델을 학습시키며, 하이퍼파라미터와 정규화 상수를 통해 손실 함수의 균형을 맞춥니다.

- 구성: Faster R-CNN의 손실 함수는 객체 분류를 위한 분류 손실 함수(L_{cls})와 객체의 경계 상자 회귀를 위한 박스 회귀 손실 함수(L_{reg})로 구성됩니다.

수식 9.4 Faster R-CNN 손실 함수

$$L = \frac{1}{N_{cls}} L_{cls} + \lambda \frac{1}{N_{reg}} L_{reg}$$

- λ (하이퍼파라미터): 분류 손실 함수와 박스 회귀 손실 함수의 균형을 맞추기 위한 하이퍼파라미터입니다. 논문에서는 초기값을 10으로 사용합니다.
- N_{cls} (정규화 상수): 손실 값을 계산할 때 사용되는 정규화 상수로, 미니 배치의 크기를 의미합니다. 논문에서는 256을 사용합니다.
- N_{reg} (정규화 상수): 손실 값을 계산할 때 사용되는 정규화 상수로, 앵커 박스의 개수를 의미합니다. 논문에서는 2,304(256 x 9)를 사용합니다.

수식 9.5 분류 손실 함수

$$L_{cls} = -\sum_i p_i^* \log(p_i) + (1 - p_i^*) \log(1 - p_i)$$

- p_i : i번째 앵커 박스에 대한 예측 확률
- p_i^* : i번째 앵커 박스에 대한 실제 레이블 (0 또는 1)
- 분류 손실 함수: 교차 엔트로피(Cross-Entropy)를 사용하며, 각 앵커 박스에 대한 클래스(객체/배경) 예측 확률과 실제 레이블 간의 차이를 계산합니다.
- 배경 클래스: 객체가 없는 영역을 나타내는 클래스로, 객체와 배경을 구분하는 데 사용됩니다. 배경 클래스에 대해서는 박스 회귀 손실을 계산하지 않습니다.

분류 손실 함수 (Classification Loss Function):

- 교차 엔트로피 사용: 교차 엔트로피를 사용하여 분류 손실을 계산합니다.
- p_i : i번째 앵커 박스(관심 영역)에 대한 클래스 예측 확률을 의미합니다.
- p_i : i번째 앵커 박스에 대한 클래스 실젯값(ground truth)을 의미하며, 앵커 박스가 객체일 경우 1, 배경일 경우 0의 이진값(binary)을 갖습니다.
- 배경 클래스 추가:
  - 객체가 없는 영역을 구분하기 위해 배경 클래스를 추가합니다.
  - 배경 클래스는 객체가 아닌 영역을 나타내며, 학습 데이터에 포함되어 있어야 합니다.
  - 객체를 감지할 때와 배경을 구분할 때 각 클래스에 대한 확률이 동시에 계산됩니다.
  - 배경 클래스에 대해서는 박스 회귀 손실 함수에 영향을 미치지 않도록 0으로 설정합니다. 즉, 배경에 대해서는 경계 상자 조정을 수행하지 않습니다.

수식 9.6 박스 회귀 손실 함수

$$L_{reg} = -\sum_i p_i^* smooth_{L_1}(t_i - t_i^*)$$

- 박스 회귀 손실 함수: $smooth_L1$ 함수를 사용하며, 예측된 경계 상자 정보와 실제 경계 상자 정보 간의 차이를 계산합니다. 배경 클래스가 아닐 때만(객체일 때) 계산됩니다.
- $smooth_L1$ 함수: L1 손실 함수와 L2 손실 함수의 장점을 결합한 함수로, 이상치(outlier)에 덜 민감하고 미분이 가능하여 역전파(backpropagation)가 가능합니다.

1. 박스 회귀 손실 함수 (Box Regression Loss Function):

- 수식 9.6: 박스 회귀 손실 함수를 나타냅니다.

NjY3NzY=

수식 9.6 박스 회귀 손실 함수

$$L_{reg} = -\sum_i p_i^* smooth_{L_1}(t_i - t_i^*)$$

- p_i: i번째 앵커 박스에 대한 클래스 실젯값 (배경일 때는 0, 객체일 때는 1)
- t_i: 모델이 예측한 i번째 앵커 박스의 경계 상자 정보 (x, y, w, h)
- t_i: i번째 앵커 박스의 실젯값 경계 상자 정보 (x, y, w, h)
- $smooth_L1$ 함수: L1 손실 함수와 L2 손실 함수의 장점을 결합한 함수 (자세한 설명은 아래 참조)
- 배경 클래스 제외: p_i가 1일 때만(객체일 때만) 박스 회귀 손실 함수를 계산합니다. 즉, 배경 클래스에 대해서는 경계 상자를 조정하지 않습니다.

수식 9.7 앵커 박스 예측값

$$t_x = d_x(P) = (\hat{G}_x - P_x)/P_w$$
$$t_y = d_y(P) = (\hat{G}_y - P_y)/P_h$$
$$t_w = d_w(P) = \log(\hat{G}_w/P_w)$$
$$t_h = d_h(P) = \log(\hat{G}_h/P_h)$$

수식 9.8 앵커 박스 실젯값

$$t_x^* = d_x(P) = (G_x - P_x)/P_w$$
$$t_y^* = d_y(P) = (G_y - P_y)/P_h$$
$$t_w^* = d_w(P) = \log(G_w/P_w)$$
$$t_h^* = d_h(P) = \log(G_h/P_h)$$

- 경계 상자 정보: 수식 9.7과 9.8은 앵커 박스의 예측 정보(t)와 실제 정보(t)를 계산하는 방법을 보여줍니다.

3. $smooth_L1$ 함수:

수식 9.9 smoothL1

$$smooth_{L_1}(x)\begin{cases}0.5x^2 & if: |x| < 1\\ |x| - 0.5 & otherwise\end{cases}$$

- 특징:
  - 입력값의 절댓값이 1보다 작은 경우 L2 손실 함수($0.5x^2$)를 적용합니다.
  - 입력값의 절댓값이 1보다 큰 경우 L1 손실 함수($|x| - 0.5$)를 적용합니다.
- 장점:
  - L2 손실 함수의 문제 완화: 큰 값에 대해 손실이 지나치게 커지는 L2 손실 함수의 문제를 완화합니다.
  - L1 손실 함수의 문제 해결: 미분값이 계산되지 않는 지점이 있는 L1 손실 함수의 문제를 해결하여 역전파가 가능하도록 합니다.

Faster R-CNN은 분류 손실 함수와 박스 회귀 손실 함수를 사용하여 객체 탐지를 수행합니다. 배경 클래스를 추가하여 객체와 배경을 명확히 구분하고, $smooth_L1$ 함수를 사용하여 이상치에 덜 민감하고 미분이 가능한 박스 회귀를 수행합니다. 이를 통해 Faster R-CNN은 객체의 위치와 크기를 정확하게 예측할 수 있습니다.

모델 실습

- MS COCO (Microsoft Common Objects in Context) 데이터세트
- Faster R-CNN 모델을 훈련하는 실습
- MS COCO 데이터세트를 사용하여 Faster R-CNN 모델을 미세 조정(fine-tuning)하여 이미지 분류를 수행합니다.
- MS COCO 데이터세트는 경계 상자 탐지, 객체 분할, 캡션 생성을 위한 대규모 이미지 데이터세트입니다. (약 328,000장 이미지, 2,500만 개 레이블, 80개 클래스)
- 실습에서는 MS COCO 데이터세트 중 개와 고양이 클래스만 소규모로 샘플링하여 사용합니다.
- 제공된 `coco.zip` 파일을 압축 해제하면 `annotations`, `train`, `val` 디렉터리로 구성된 데이터세트 구조를 얻을 수 있습니다.
- `annotations` 디렉터리에는 JSON 형식의 데이터 주석(Data Annotation) 파일이 포함되어 있습니다. (`train_annotations.json`, `val_annotations.json`)
- 데이터 주석 파일에는 정보(info), 라이선스(licenses), 카테고리(categories), 이미지(images), 어노테이션(annotations) 정보가 포함되어 있습니다.
- 카테고리 정보: 클래스 정보
- 이미지 정보: 파일 이름, 이미지 ID 등

- 어노테이션 정보: 분할 마스크 좌표, 픽셀 크기, 군집 여부, 이미지 ID, 경계 상자 좌표, 카테고리 ID, 어노테이션 ID 등

상세 요약:

1. 실습 목표: MS COCO 데이터세트를 활용하여 Faster R-CNN 모델을 미세 조정하고, 이미지 분류를 수행합니다.
2. MS COCO 데이터세트:
   - 2014년에 처음 소개된 대규모 이미지 데이터세트입니다.
   - 경계 상자 탐지, 객체 분할 및 캡션 생성을 위한 데이터를 제공합니다.
   - 약 328,000장의 이미지와 약 2,500만 개의 레이블을 포함합니다.
   - 80개의 클래스로 구성되어 있습니다.
3. 실습용 데이터:
   - MS COCO 데이터세트는 대규모이기 때문에, 실습에서는 개와 고양이 클래스만 소규모로 샘플링하여 사용합니다.
   - `coco.zip` 파일을 제공하며, 압축을 해제하면 다음과 같은 구조의 디렉터리가 생성됩니다.
     - `annotations` : 데이터 주석 파일이 저장된 디렉터리
       - `train_annotations.json` : 훈련용 데이터 주석 파일
       - `val_annotations.json` : 검증용 데이터 주석 파일
     - `train` : 훈련용 이미지 데이터가 저장된 디렉터리
     - `val` : 검증용 이미지 데이터가 저장된 디렉터리
4. 데이터 주석 (Data Annotation):
   - `annotations` 디렉터리에 저장된 JSON 파일은 모델 학습에 필요한 정보를 제공합니다.
   - JSON 파일은 다음 정보를 포함합니다.
     - 정보(info): MS COCO 데이터세트에 대한 간략한 정보
     - 라이선스(licenses): 데이터를 사용할 때 준수해야 하는 라이선스 정보
     - 카테고리(categories): 데이터세트에서 제공되는 클래스 정보 (실습에서는 개, 고양이)
     - 이미지(images): 파일 이름, 이미지 ID, 라이선스 ID 등 이미지 관련 정보
     - 어노테이션(annotations): 분할 마스크 좌표, 픽셀 크기, 군집 여부, 이미지 ID, 경계 상자 좌표, 카테고리 ID, 어노테이션 ID 등 객체 관련 정보

결론적으로, 이 이미지는 MS COCO 데이터세트를 활용한 Faster R-CNN 모델 훈련 실습을 소개하며, 실습에 필요한 데이터의 구조와 내용을 설명하고 있습니다.