Faster R-CNN

Towards Real-Time Object Detection with Region Proposal Networks

▼ Abstract

- 객체 탐지 모델에서 객체 위치를 추정하기 위해서는 **영역 추정(region proposal algorithms)**이 중요!
- SPPnet이나 Fast R-CNN 같은 모델들의 발전을 통해 객체 탐지 시간은 줄였지만, 영역 추정 단계에서 bottle neck 현상이 발생
→ **RPN(Region Proposal Network)**: 영역 추정 네트워크 기법 제안
- 객체 탐지 네트워크와 합성곱 피쳐들을 공유하므로 영역 추정에 비용이 들지 x
- object bound와 objectness score를 동시에 예측하는 합성곱 네트워크
- 품질 좋은 영역 추정을 할 수 있도록 end-to-end 훈련됨 for Fast-RCNN
- 더 나아가 RPN과 Fast R-CNN을 결합해 하나의 신경망으로 만들어 합성곱 피처를 공유하도록 함 like attention

▼ Introduction

Problem :
최근 객체 탐지 분야의 많은 발전 by 영역 추정 방법(region proposal method) & R-CNN(region based convolutional neural networks)
but, R-CNN도 Fast R-CNN도 영역 추정 단계에서는
test-time computational bottleneck 문제ㅠ

Fast R-CNN : region proposal 방법인 selective search는 CPU에서 계산되었으며, CNN 외부에서 진행

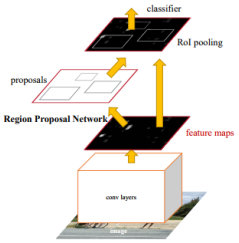
Solution:
→ Fast R-CNN을 보완
GPU의 이점을 최대한 활용하고 CNN 내부에서 진행하기 위해 Region Proposal Network(RPN) 도입
RPN : object bounds와 objectness score를 동시에 예측하는 fully convolutional network

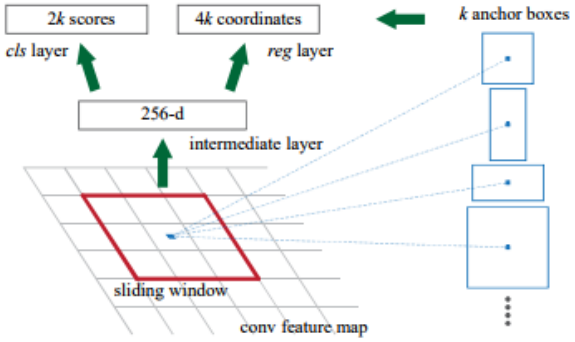
▼ **Faster R-CNN**

> 💡 2가지 모듈로 구성 → 이걸 합쳐서 single로!
> 1. deep fully convolutional network : proposes region
> 2. fast R-CNN detector : uses the proposed regions
> single로 region proposal을 생성하고 object detection도 수행하는 것!

3.1 Region Proposal Networks(RPN)

- 이미지 전체를 입력받음 → objectness score 알려주는 영역추정 경계 박스(object proposal) 반환!
- fully convolutional network를 이용해 처리 → Fast R-CNN과 계산 공유
- **Anchors**
  → 하나의 feature map에서 여러가지 크기, 모양의 object를 검출하기 위한 미리 정해진 후보 box





1. 이미지가 conv layer를 통과하여 feature map 추출
2. feature map의 각 위치에서 3x3 sliding window를 사용
3. 각 위치에 대해 region proposal을 생성하기 위해 k개의 anchor boxes 사용
4. cls layer는 k개의 박스에 대해 객체가 있는지/없는지 여부 2k scores 출력
5. reg later는 k개의 박스 좌표 정보(x,y,w,h) 4k 출력
6. feature map 크기가 w*h 이면 총 w*h*k개의 anchor가 이용되어 region proposals가 생성됨

→ proposal에는 object score 담겨져 있음 → detection network 거쳐서 class 확률 갖게됨

- **Translation- Invariant Anchors**

: 대상의 위치에 영향 받지 x, 위치 변화에 무관 → 물체의 위치가 바뀌면 바뀐 물체의 영역을 뽑아낼 수 있고, 모델의 크기 줄일 수 있음, parameter수 감소, overfitting 위험 감소

- **Loss Function**

: RPN 학습을 위해서는 각 Anchor에 Binary Classification을 할당해야함!

1. GroundTruth에 대한 IoU가 가장 높은 Anchor가 무엇인지

2. GroundTruth에 대한 IoU 값이 0.7보다 높은 Anchor가 무엇인지

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

i : Anchor의 index
pi: i번째 Anchor가 예측한 객체가 있을 확률값
pi*: Ground Truth label
ti: Bounding Box에 대한 4개 값 벡터
ti*: Ground-truth Box의 좌표값
Lcls: Classification을 위한 loss → Log loss
Lreg: Regression을 위한 Loss

L_cls - object가 있는지 없는지에 대한 classification loss

L_reg - object가 있을 때의 bounding box의 regression loss

Bounding Box Regression을 위한 4개의 좌표값

$$\begin{aligned} t_{\mathrm{x}} &= (x - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}} = (y - y_{\mathrm{a}})/h_{\mathrm{a}}, \\ t_{\mathrm{w}} &= \log(w/w_{\mathrm{a}}), \quad t_{\mathrm{h}} = \log(h/h_{\mathrm{a}}), \\ t_{\mathrm{x}}^* &= (x^* - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}}^* = (y^* - y_{\mathrm{a}})/h_{\mathrm{a}}, \\ t_{\mathrm{w}}^* &= \log(w^*/w_{\mathrm{a}}), \quad t_{\mathrm{h}}^* = \log(h^*/h_{\mathrm{a}}), \end{aligned}$$

x,y,w,h : box 중앙의 x,y좌표, 넓이, 높이
→ ground-truth box 근처에 있는 anchor box를 통해 bounding box regression을 진행!

→ paramterize를 통해 크기가 큰 박스에서 loss가 크게 발생되지 않도록 하며, anchor box의 크기로 나누어 큰 물체든 작은 물체든 동등하게 학습 가능

각 Anchor들을 자신이 맡은 크기에 대해서만 찾으며, 다른 anchor들과 weight 공유하지 x

- **Training RPNs**

Fast R-CNN에서 소개한 **Image Centric Sampling**을 이용하여 end-to-end로 학습!

Image Centric Sampling : 여러 RoI를 한번에 뽑는 것이 아니라, 먼저 큰 영역을 뽑고 그 영역에서 작은 영역들을 뽑아내는 방식

→ negative anchor의 bias가 발생하지 않도록 256개의 Anchor를 임의로 뽑고, positive와 negative의 비율을 1:1로 맞춰서 Loss 계산!

3.2 Sharing Features for RPN and Fast R-CNN

RPN에서 생성한 Proposal은 Detection Network(Fast R-CNN)으로 전달됨.

RPN과 Fast R-CNN을 통합하여 학습 진행, Convolution Layer를 공유하지만 각각 독립적으로 훈련!

→ **방법1. Alternating Training 사용!**

방법2. Approximate joint training

방법3. Non-approximate joint training

- **4-Step Alternating Training**

: 먼저 RPN을 학습하고, 여기서 추출한 Proposal을 통해 Fast R-CNN을 학습하는 방법

Step1. RPN 학습 : ImageNet으로 Pretrained 된 모델, RPN 학습을 통해 Fine-tuning 진행

Step2. Fast R-CNN을 Detector로 해서 RPN에서 뽑아낸 Proposal을 통해 학습 : Detector 모델 또한 ImageNet으로 pre-trained, fine tuning 진행

Step3. 공유된 Convolution layer를 고정시키고, RPN에 대해 고유한 layer만을 학습

Step4. 고정되었던 Convolution layer를 유지한 채, Fast R-CNN 부분을 학습

→ 통합된 Network에서 같은 Feature 공유!

▼ **Conclusion**

> 💡 RPN과 Detector가 Convolution layer를 공유함으로써 영역 추정 비용을 크게 줄일 수 있었음!
> Faster R-CNN은 실시간 객체 탐지가 가능할 정도로 빠르고, 전체적인 정확도도 뛰어남!