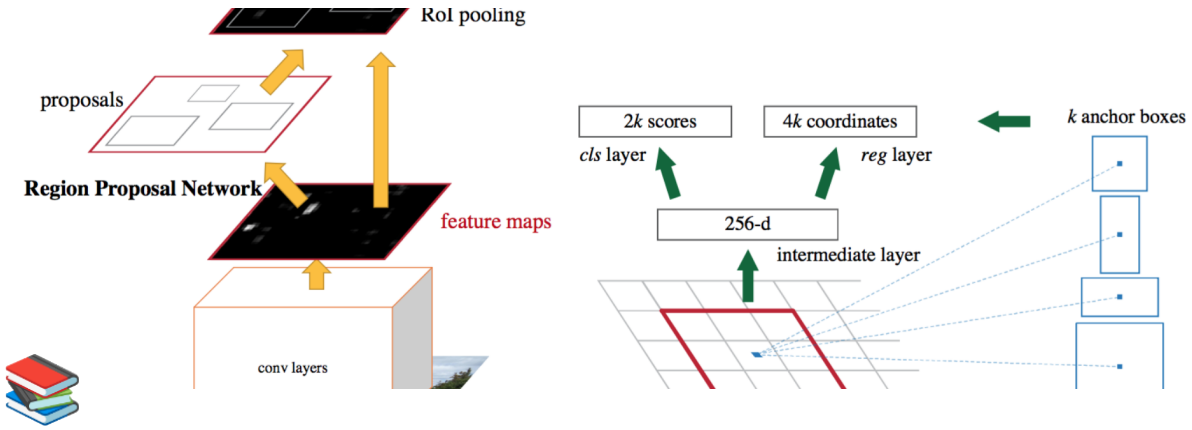


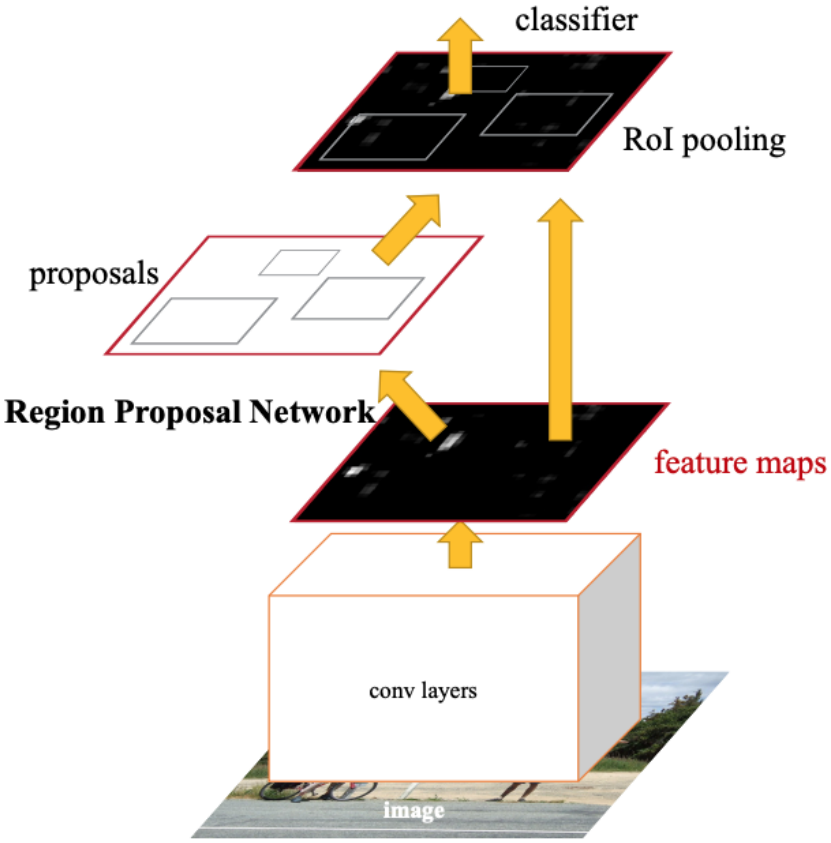
Faster R-CNN

Faster R-CNN

- 기존의 모델들은 후보 영역 추출을 위해 사용되는 Selective search 알고리즘은 CPU 상에서 동작하고 이로 인해 네트워크에서 병목현상이 발생
- Faster R-CNN은 이러한 문제를 해결하고자 **후보 영역 추출 작업을 수행하는 네트워크인 Region Proposal Network** (이하 RPN)를 도입
- RPN은 region proposals를 보다 정교하게 추출하기 위해 다양한 크기와 가로세로비를 가지는 bounding box인 **Anchor box**를 도입

⇒ **Faster R-CNN 모델을 간략하게 보면 RPN과 Fast R-CNN 모델이 합쳐졌다고 볼 수 있음**

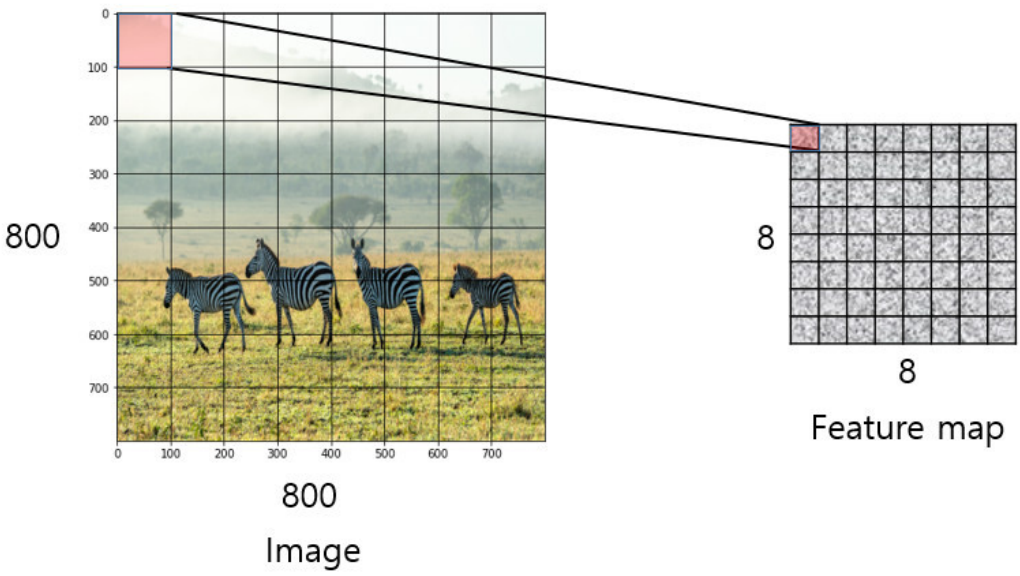
- RPN에서 region proposals를 추출하고 이를 Fast R-CNN 네트워크에 전달하여 객체의 class와 위치를 예측
- 모델의 전체 과정이 GPU 상에서 동작하여 병목 현상이 발생하지 않으며, end-to-end로 네트워크를 학습시키는 것이 가능



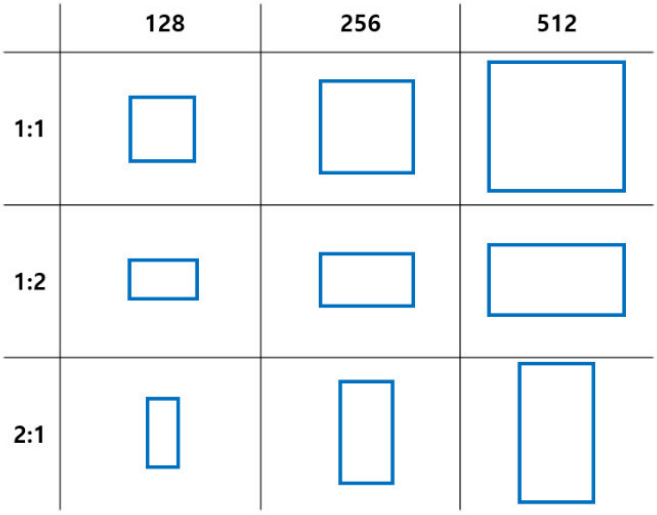
1. 원본 이미지를 pre-trained된 CNN 모델에 입력하여 feature map을 얻음
2. feature map은 RPN에 전달되어 적절한 region proposals을 산출
3. region proposals와 1) 과정에서 얻은 feature map을 통해 RoI pooling을 수행하여 고정된 크기의 feature map을 얻음
4. Fast R-CNN 모델에 고정된 크기의 feature map을 입력하여 Classification과 Bounding box regression을 수행합니다.

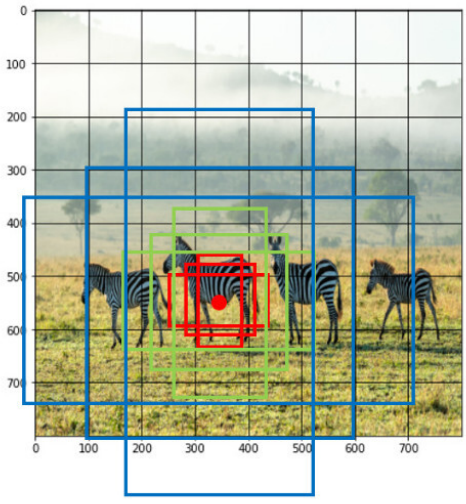
Main Ideas

1. Anchor box



- Selective search를 통해 region proposal을 추출하지 않을 경우, 원본 이미지를 일정 간격의 grid로 나눠 각 grid cell을 bounding box로 간주하여 feature map에 encode하는 **Dense Sampling** 방식을 사용
- 이같은 경우 sub-sampling ratio를 기준으로 grid를 나누게 됨
- 가령 원본 이미지의 크기가 800x800이며, sub-sampling ratio가 1/100이라고 할 때, CNN 모델에 입력시켜 얻은 최종 feature map의 크기는 8x8(800x1/100)가 됨
- 여기서 feature map의 각 cell은 원본 이미지의 100x100만큼의 영역에 대한 정보를 함축하고 있다고 할 수 있음
- 원본 이미지에서는 8x8개만큼의 bounding box가 생성된다고 볼 수 있음





- but, 이처럼 고정된 크기(fixed size)의 bounding box를 사용할 경우, 다양한 크기의 객체를 포착하지 못할 수 있다는 문제가 있음
- 본 논문에서는 이러한 문제를 해결하고자 지정한 위치에 사전에 정의한 서로 다른 크기(scale)와 가로세로비(aspect ratio)를 가지는 bounding box인 **Anchor box**를 생성하여 다양한 크기의 객체를 포착하는 방법을 제시

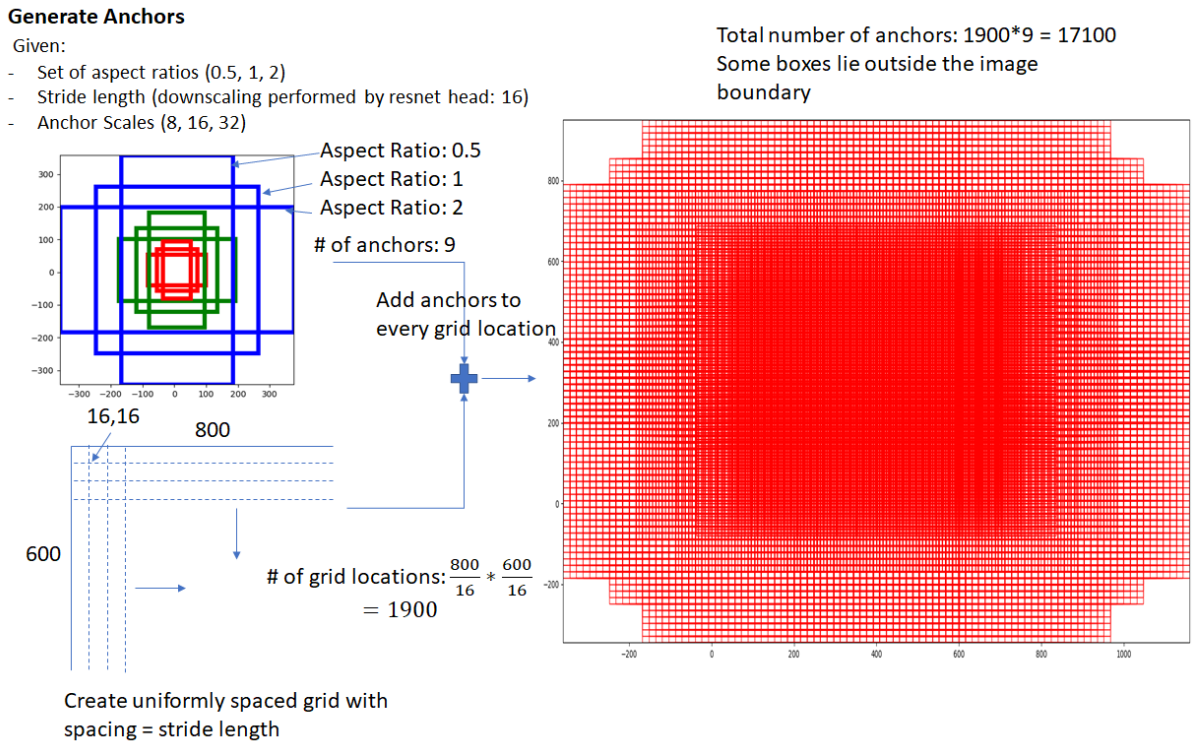
- 논문에서 3가지 scale([128, 256, 512])과 3가지 aspect ratio([1:1, 1:2, 2:1])를 가지는 총 9개의 서로 다른 anchor box를 사전에 정의(pre-define)

$$w \times h = s^2$$
$$w = \frac{1}{2} \times h$$
$$\frac{1}{2} \times h^2 = s^2$$
$$h = \sqrt{2s^2}$$
$$w = \frac{\sqrt{2s^2}}{2}$$

- 여기서 scale은 anchor box의 width(=w), height(=h)의 길이를, aspect ratio는 width,height의 길이의 비율을 의미
- 여기서aspect ratio에 따른 width, height의 길이는 **aspect ratio가 1:1일 때의 anchor box의 넓이를 유지한 채** 구함

ex)

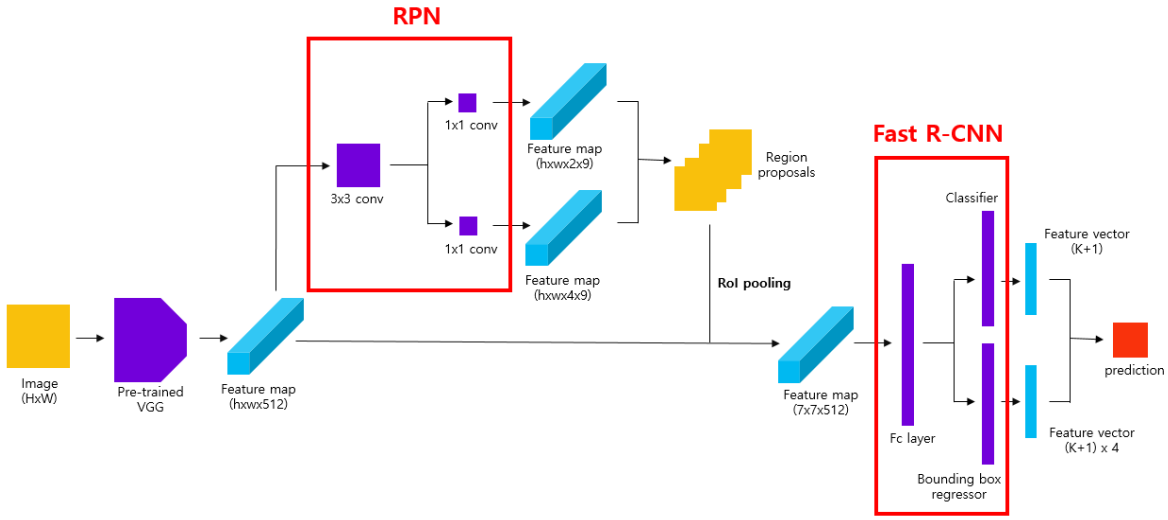
- scale이 s이며, aspect ratio가 1:1 일 때의 anchor box의 넓이는 $s^2 (= s \times s)$임
- 여기서 aspect ratio가 1:2, 즉 height가 width의 2배일 때 위와 같은 수식에 따라 width, height를 구함
- aspect ratio가 2:1인 경우에도 마찬가지로 구함



- anchor box는 원본 이미지의 각 grid cell의 중심을 기준으로 생성
- 원본이미지에서 sub-sampling ratio를 기준으로 anchor box를 생성하는 기준점인 anchor를 고정
- 이 anchor를 기준으로 사전에 정의한 anchor box 9개를 생성

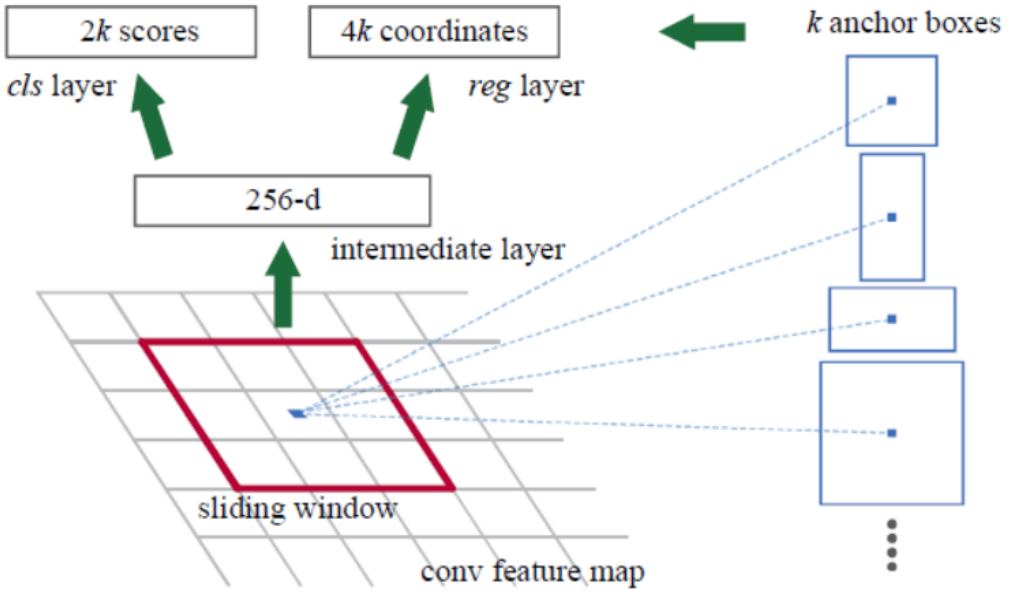
- 위의 그림에서 원본 이미지의 크기는 600x800이며, sub-sampling ratio=1/16 임
- 이 때 anchor가 생성되는 수는 1900(=600/16 x 800/16)이며, anchor box는 총 17100(=1900 x 9)개가 생성
- 이같은 방식을 사용할 경우, 기존에 고정된 크기의 bounding box를 사용할 때보다 9배 많은 bounding box를 생성하며, 보다 다양한 크기의 객체를 포착하는 것이 가능

2. RPN(Region Proposal Network)

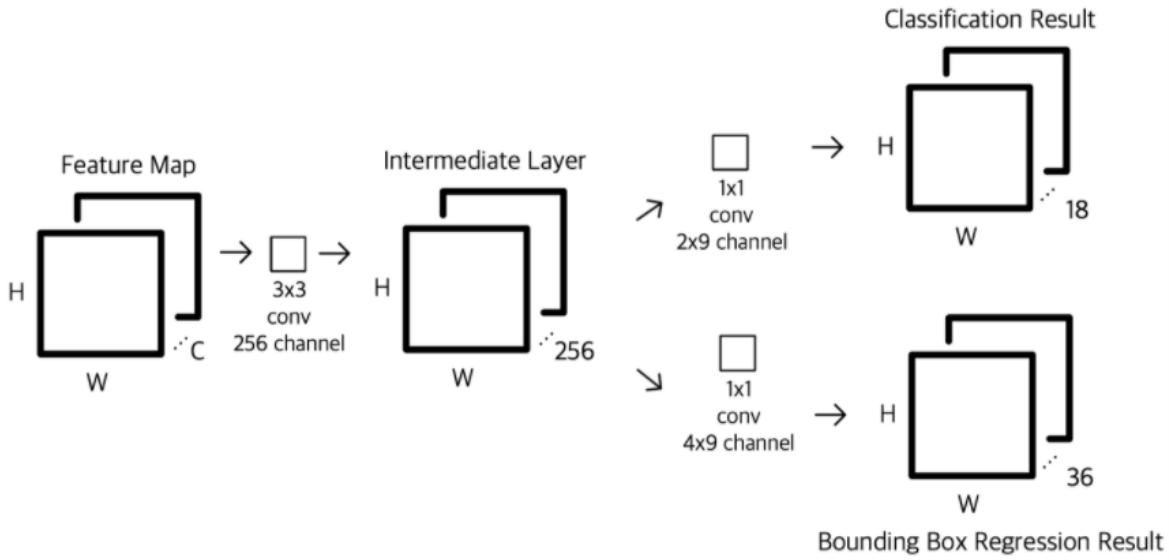


- RPN은 원본 이미지에서 region proposals를 추출하는 네트워크임
- 원본 이미지에서 anchor box를 생성하면 수많은 region proposals가 만들어짐
- RPN은 region proposals에 대하여 class score를 매기고, bounding box coefficient를 출력하는 기능을 함

동작 과정



The Output of RPN



Bounding Box Regression Result

1) 원본 이미지를 pre-trained된 VGG 모델에 입력하여 feature map을 얻음

→ 원본 이미지의 크기가 800x800이며, sub-sampling ratio가 1/100이라고 했을 때 8x8 크기의 feature map이 생성(channel 수는 512개).

2) 위에서 얻은 feature map에 대하여 3x3 conv 연산을 적용, 이때 feature map의 크기가 유지될 수 있도록 padding을 추가

→ 8x8x512 feature map에 대하여 3x3 연산을 적용하여 8x8x512개의 feature map이 출력

3) class score를 매기기 위해서 feature map에 대하여 1x1 conv 연산을 적용

- 이 때 출력하는 feature map의 channel 수가 2x9가 되도록 설정
- **RPN에서는 후보 영역이 어떤 class에 해당하는지까지 구체적인 분류를 하지 않고 객체가 포함되어 있는지 여부만을 분류**
- 또한 anchor box를 각 grid cell마다 9개가 되도록 설정

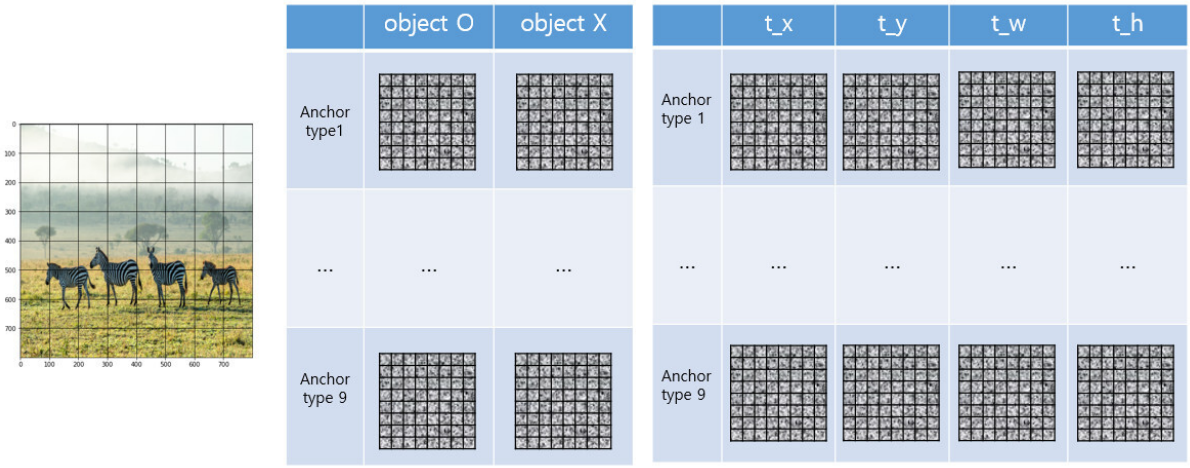
- 따라서 **channel 수는 2(object 여부) x 9(anchor box 9개)**가 됨

  → 8x8x512 크기의 feature map을 입력받아 8x8x2x9크기의 feature map을 출력

4) bounding box regressor를 얻기 위해 feature map에 대하여 1x1 conv 연산을 적용

- 이 때 출력하는 feature map의 channel 수가 **4(bounding box regressor)x9(anchor box 9개)**가 되도록 설정

  → 8x8x512 크기의 feature map을 입력받아 8x8x4x9크기의 feature map을 출력합니다.



- RPN의 출력 결과는 위와 같음
- 좌측 표는 anchor box의 종류에 따라 객체 포함 여부를 나타낸 feature map이며, 우측 표는 anchor box의 종류에 따라 bounding box regressor를 나타낸 feature map
- 이를 통해 8x8 grid cell마다 9개의 anchor box가 생성되어 총 576(=8x8x9)개의 region proposals가 추출되며, feature map을 통해 각각에 대한 객체 포함 여부와 bounding box regressor를 파악할 수 있음
- 이후 class score에 따라 상위 N개의 region proposals만을 추출하고, Non maximum suppression을 적용하여 최적의 region proposals만을 Fast R-CNN에 전달하게 됨

3. Multi-task loss

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

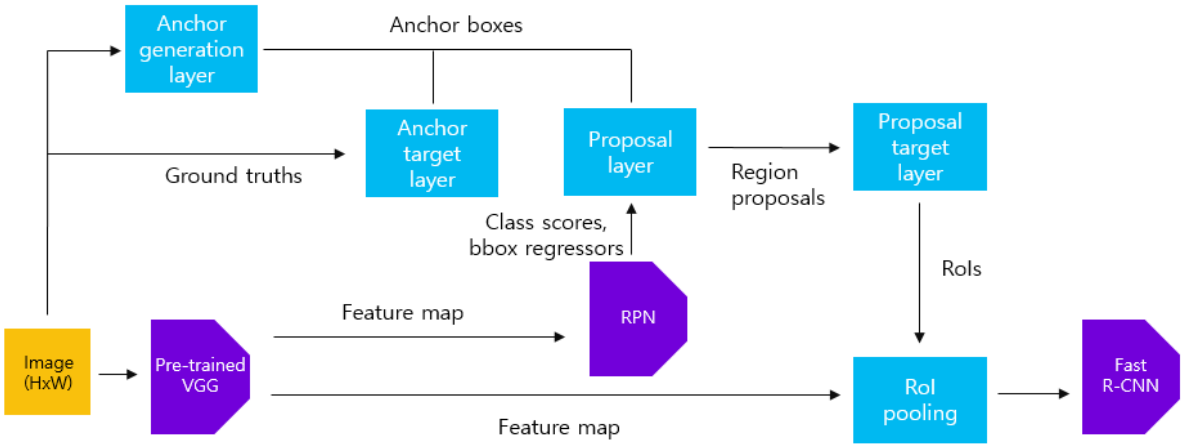
- i: mini-batch 내의 anchor의 index
- P_i: anchor i에 객체가 포함되어 있을 예측 확률
- P_i^*: anchor가 양성일 경우 1, 음성일 경우 0을 나타내는 index parameter
- t_i: 예측 bounding box의 파라미터화된 좌표(coefficient)
- t_i^*: ground truth box의 파라미터화된 좌표
- L_{cls} : Loss loss
- L_{reg} : Smooth L1 loss

- N_{cls} : mini-batch의 크기(논문에서는 256으로 지정)
- N_{reg} : anchor 위치의 수
- λ : balancing parameter(default=10)

- RPN과 Fast R-CNN을 학습시키기 위해 Multi-task loss를 사용
- But, RPN에서는 객체의 존재 여부만을 분류하는 반면, Fast R-CNN에서는 배경을 포함한 class를 분류한다는 점에서 차이가 있음

Faster R-CNN 구조와 Training

- Faster R-CNN = RPN + Fast R-CNN 이라고 단순하게 설명
- anchor를 생성하고 처리하는 작업과 적절한 region proposals를 추출하는 작업이 있음



1) feature extraction by pre-trained VGG16

- pre-trained된 VGG16 모델에 800x800x3 크기의 원본 이미지를 입력하여 50x50x512 크기의 feature map을 얻음
- 여기서 sub-sampling ratio는 1/16임
  - **Input** : 800x800x3 sized image
  - **Process** : feature extraction by pre-trained VGG16
  - **Output** : 50x50x512 sized feature map

2) Generate Anchors by Anchor generation layer

- region proposals를 추출하기에 앞서 **원본 이미지에 대하여 anchor box를 생성**하는 과정이 필요
- 원본 이미지의 크기에 sub-sampling ratio를 곱한만큼의 grid cell이 생성 되며, 이를 기준으로 각 grid cell마다 9개의 anchor box를 생성

- 즉, 원본 이미지에 50x50(=800x1/16 x 800x1/16)개의 grid cell이 생성되고, 각 grid cell마다 9개의 anchor box를 생성하므로 총 22500(=50x50x9)개의 anchor box가 생성
  - **Input** : 800x800x3 sized image
  - **Process** : generate anchors
  - **Output** : 22500(=50x50x9) anchor boxes

3) Class scores and Bounding box regressor by RPN

- RPN은 VGG16으로부터 feature map을 입력받아 anchor에 대한 **class score, bounding box regressor를 반환**하는 역할을 함
  - **Input** : 50x50x512 sized feature map
  - **Process** : Region proposal by RPN
  - **Output** : class scores(50x50x2x9 sized feature map) and bounding box regressors(50x50x4x9 sized feature map)

4) Region proposal by Proposal layer

- Proposal layer에서는 2)번 과정에서 생성된 anchor boxes와 RPN에서 반환한 class scores와 bounding box regressor를 사용하여 **region proposals를 추출**하는 작업을 수행
- 먼저 Non maximum suppression을 적용하여 부적절한 객체를 제거한 후, class score 상위 N개의 anchor box를 추출
- 이후 regression coefficients를 anchor box에 적용하여 anchor box가 객체의 위치를 더 잘 detect하도록 조정
  - **Input**
    - 22500(=50x50x9) anchor boxes
    - class scores(50x50x2x9 sized feature map) and bounding box regressors(50x50x4x9 sized feature map)
  - **Process** : region proposal by proposal layer
  - **Output** : top-N ranked region proposals

5) Select anchors for training RPN by Anchor target layer

- Anchor target layer의 목표는 **RPN이 학습하는데 사용할 수 있는 anchor를 선택**하는 것
- 먼저 2)번 과정에서 생성한 anchor box 중에서 원본 이미지의 경계를 벗어나지 않는 anchor box를 선택
- 그 다음 positive/negative 데이터를 sampling해줌
- 여기 positive sample은 객체가 존재하는 foreground, negative sample은 객체가 존재하지 않는 background를 의미

- 전체 anchor box 중에서 1) ground truth box와 가장 큰 IoU 값을 가지는 경우 2) ground truth box와의 IoU 값이 0.7 이상인 경우에 해당하는 box를 positive sample로 선정
- 반면 ground truth box와의 IoU 값이 0.3 이하인 경우에는 negative sample로 선정
- IoU 값이 0.3~0.7인 anchor box는 무시
- 이러한 과정을 통해 RPN을 학습시키는데 사용할 데이터셋을 구성하게 됩니다.
  - **Input** : anchor boxes, ground truth boxes

- **Process** : select anchors for training RPN
- **Output** : positive/negative samples with target regression coefficients

6) Select anchors for training Fast R-CNN by Proposal Target layer

- Proposal target layer의 목표는 proposal layer에서 나온 region proposals 중에서 **Fast R-CNN 모델을 학습시키기 위한 유용한 sample을 선택**하는 것
- 여기서 선택된 region proposals는 1)번 과정을 통해 출력된 feature map에 RoI pooling을 수행하게 됩니다. 먼저 region proposals와 ground truth box와의 IoU를 계산하여 0.5 이상일 경우 positive, 0.1~0.5 사이일 경우 negative sample로 label
  - **Input** : top-N ranked region proposals, ground truth boxes
  - **Process** : select region proposals for training Fast R-CNN
  - **Output** : positive/negative samples with target regression coefficients

7) Max pooling by RoI pooling

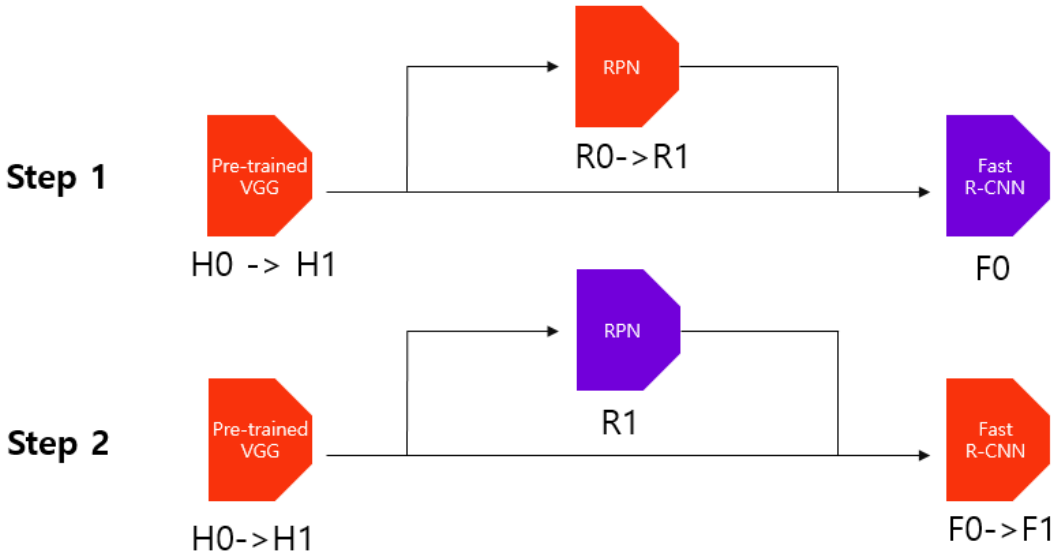
- 원본 이미지를 VGG16 모델에 입력하여 얻은 feature map과 6) 과정을 통해 얻은 sample을 사용하여 RoI pooling을 수행
- 이를 통해 고정된 크기의 feature map이 출력
  - **Input**
    - 50x50x512 sized feature map
    - positive/negative samples with target regression coefficients
  - **Process** : RoI pooling
  - **Output** : 7x7x512 sized feature map

8) Train Fast R-CNN by Multi-task loss

- 나머지 과정은 Fast R-CNN 모델의 동작 순서와 동일
- 입력받은 feature map을 fc layer에 입력하여 4096 크기의 feature vector를 얻음
- 이후 feature vector를 Classifier와 Bounding box regressor에 입력하여 (class의 수가 K라고 할 때)각각 (K+1), (K+1) x 4 크기의 feature vector를 출력
- 출력된 결과를 사용하여 Multi-task loss를 통해 Fast R-CNN 모델을 학습
  - **Input** : 7x7x512 sized feature map
  - **Process**
    - feature extraction by fc layer
    - classification by Classifier
    - bounding box regression by Bounding box regressor
    - Train Fast R-CNN by Multi-task loss
  - **Output** : loss(Loss loss + Smooth L1 loss)

*Alternating Training

- 논문의 저자는 Faster R-CNN 모델을 학습시키기 위해 RPN과 Fast R-CNN을 번갈아가며 학습시키는 **Alternating Training** 방법을 사용



1) 먼저 Anchor generation layer에서 생성된 anchor box와 원본 이미지의 ground truth box를 사용하여 Anchor target layer에서 RPN을 학습시킬 positive/negative 데이터셋을 구성,

- 이를 활용하여 **RPN을 학습**
- 이 과정에서 pre-trained된 VGG16 역시 학습

2) Anchor generation layer에서 생성한 anchor box와 학습된 RPN에 원본 이미지를 입력하여 얻은 feature maps를 사용하여 proposals layer에서 region proposals를 추출

- 이를 Proposal target layer에 전달하여 Fast R-CNN 모델을 학습시킬 positive/negative 데이터셋을 구성
- 이를 활용하여 **Fast R-CNN을 학습**
- 이 때 pre-trained된 VGG16 역시 학습

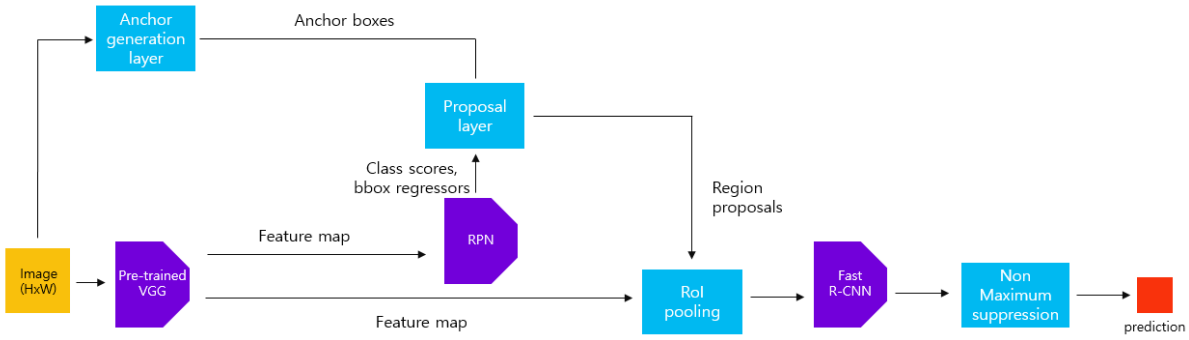
3) 앞서 학습시킨 RPN과 Fast R-CNN에서 **RPN에 해당하는 부분만 학습(fine tune)**시킴

- 세부적인 학습 과정은 1)과 같음
- 이 과정에서 두 네트워크끼리 공유하는 convolutional layer, 즉 **pre-trained된 VGG16은 고정(freeze)**

4) 학습시킨 RPN(3)번 과정)을 활용하여 추출한 region proposals를 활용하여 **Fast R-CNN을 학습(fine tune)**시킴

- 이 때 **RPN과 pre-trained된 VGG16은 고정(freeze)**

Detection



- 실제 detection(=inference) 시에는 Anchor target layer와 Proposal target layer는 사용되지 않음
- 두 layer 모두 네트워크를 학습시키기 위한 데이터셋을 구성하는데 사용되기 때문
- Fast R-CNN은 Proposal layer에서 추출한 region proposals를 활용하여 detection을 수행
- 그리고 최종적으로 얻은 predicted box에 **Non maximum suppression을 적용**하여 최적의 bounding box만을 결과로 출력

Result

- Faster R-CNN 모델은 PASCAL VOC 2012 데이터셋에서 mAP 값이 75.9를 보이면서 Fast R-CNN 모델보다 더 높은 detection 성능을 보임
- 또한 Fast R-CNN 모델이 0.5fps(frame pre second)인 반면 Faster R-CNN 모델은 17fps를 보이며, 이미지 처리 속도 면에서 발전한 결과를 보임
- 또한 feature extraction에 사용하는 convolutional layer의 feature를 공유하면서 end-to-end로 학습시키는 것이 가능
- But, 논문의 저자는 detection 속도에 대해 "near real-time"이라고 언급하며, 실시간 detection에는 한계가 있음을 인정

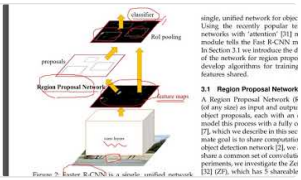
참고자료

- 유튜브

[논문 읽기] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (NIPS' 15)

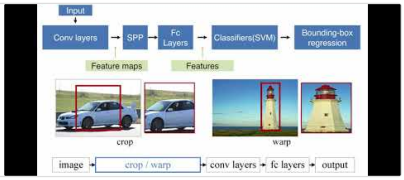
논문 읽기] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (NIPS' 15)

https://www.youtube.com/watch?v=46SjJbUcO-c&list=PLRx0vPvlEmdADpce8aoBhNnDaaHQN1Typ&index=28



양우식 - Fast R-CNN & Faster R-CNN

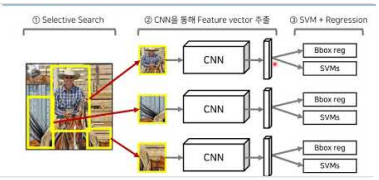
https://www.youtube.com/watch?v=Jo32zrxr6l8



객체 검출(Object Detection) 딥러닝 기술: R-CNN, Fast R-CNN, Faster R-CNN 발전 과정 핵심 요약

객체 검출(Object Detection) 딥러닝 기술: R-CNN, Fast R-CNN, Faster R-CNN 발전 과정 핵심 요약

https://www.youtube.com/watch?v=jqNCdjOB15s&list=PLRx0vPvlEmdADpce8aoBhNnDaaHQN1Typ&index=26

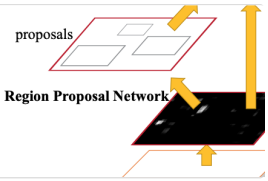


- 블로그

Faster R-CNN 논문(Faster R-CNN: Towards Real-Time ObjectDetection with Region Proposal Networks) 리뷰

이번 포스팅에서는 Faster R-CNN 논문(을 읽고 정리해봤습니다. 기존 Fast R-CNN 모델은 여전히 Selective search 알고리즘을 통해 region proposals 추출하기 때문에 학습 및 detection 속도를 향상시키는데 한계가 있습니다. 또한 detection을 위한 과정을 end-to-end로 수행하지 못한다는 문제가 있습니다. 이러한 문제를 해결하여 속도와 모델의 완성도 측면에서 더 좋은 모습을 보인 Faster

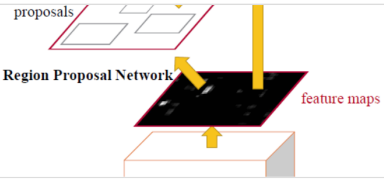
https://herbwood.tistory.com/10



[논문 읽기] Faster R-CNN (2015) 리뷰

이번에 읽어볼 논문은 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks' 입니다. Fast R-CNN과 R-CNN에서 region proposals는 selective search로 생성되었습니다. Faster R-CNN에서는 region proposals를 생성하는 작업과 object detection이 동일한 CNN에서 수행됩니다. 즉, region proposal

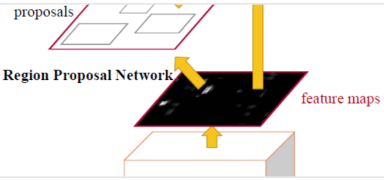
https://deep-learning-study.tistory.com/464?category=968059



[논문 읽기] Faster R-CNN (2015) 리뷰

이번에 읽어볼 논문은 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks' 입니다. Fast R-CNN과 R-CNN에서 region proposals는 selective search로 생성되었습니다. Faster R-CNN에서는 region proposals를 생성하는 작업과 object detection이 동일한 CNN에서 수행됩니다. 즉, region proposal

https://deep-learning-study.tistory.com/464



[논문 리뷰] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

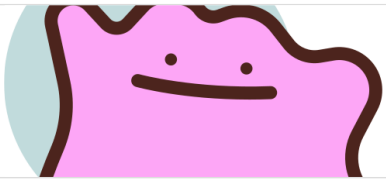
본 논문 리뷰는 v1 : 4/6/15 (NIPS 2015) 이 아닌 을 기준으로 작성하였습니다. 이때 당시 SOTA는 SPPnet 과 Fast R-CNN 입니다. bottleneck 같은 region proposal(RP) 계산을 사용합니다. 본 논문에서는 Region Proposal Network(RPN)을 소개합니다. RPN이란 전체 이미지를 CNN시킨 feature map을 공유하는 네트워크로 RP를 cost

https://cocopambag.tistory.com/4

Faster R-CNN 논문 리뷰

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks 을 정리한 글입니다!

https://zzsza.github.io/data/2018/05/09/Faster-RCNN-review/



- 깃허브

https://github.com/wllvcxz/faster-rcnn-pytorch