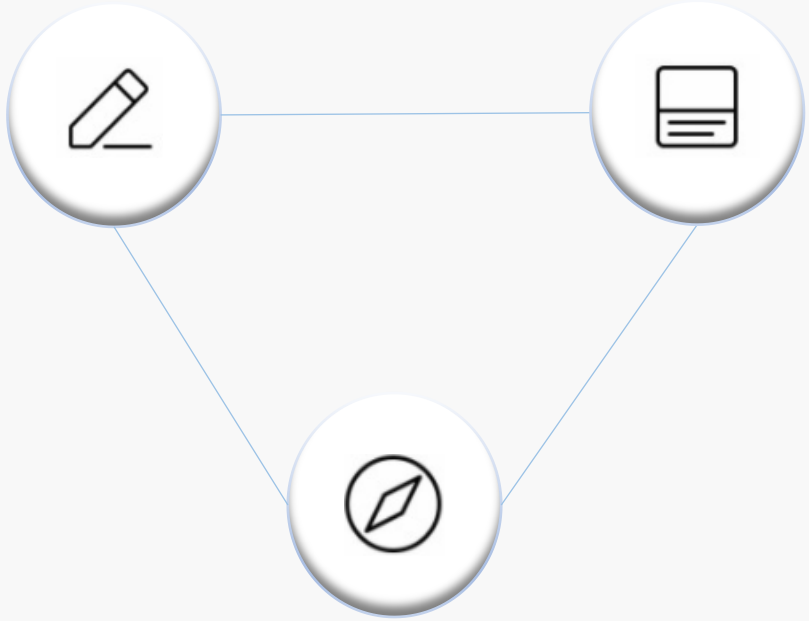
Faster R-CNN: Towards Real-Time Object Detection With Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

김성민

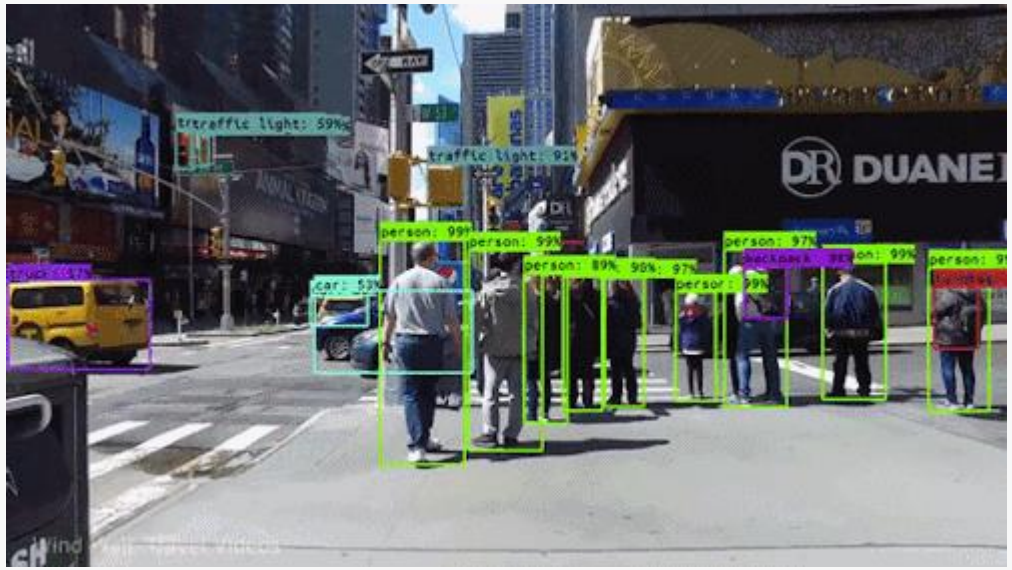
SAI

CONTENTS



CONTENTS

01 Object Detection



Object Detection

이미지 출처 : https://github.com/abhishek1605/Detect-Me

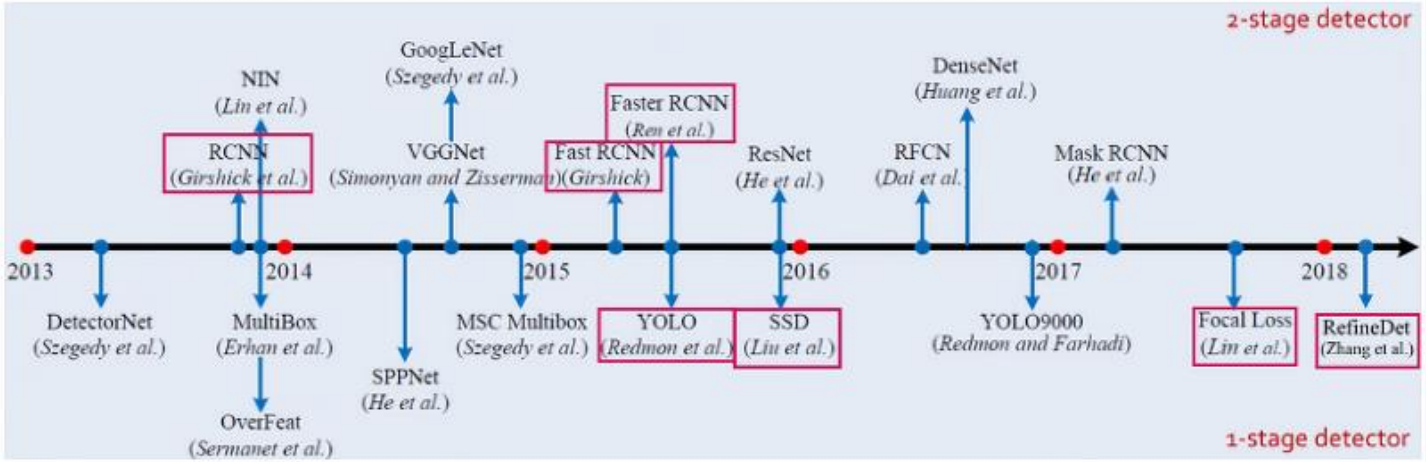
01 Object Detection

1-stage Detector

Localization과 Classification을 동시에 해결
-> 빠르지만 정확도가 떨어진다.

2-stage Detector

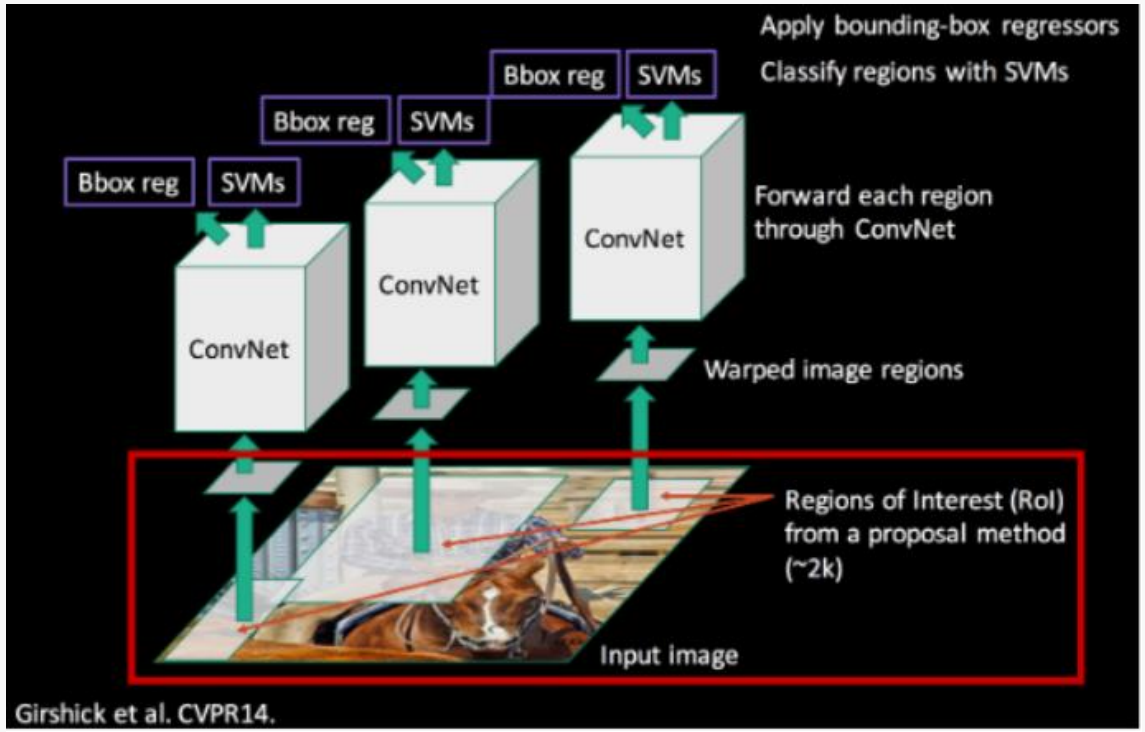
Localization과 Classification을 순차적으로 해결
-> 느리지만 정확도가 올라간다.



이미지 출처 : https://ganghee-lee.tistory.com/34

02 R-CNN

- 딥러닝을 이용한 2-stage Detector
- PASCAL VOC 2012에서 이전의 방법보다 30%가 넘는 큰 향상
- 이후 다른 모델들에 큰 영향



이미지 출처 : https://velog.io/@skhim520/R-CNN-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0

02 R-CNN

1. Region Proposal

물체가 있을 만한 영역을 찾는다.

2. CNN (Convolution Neural Network)

각 영역으로부터 고정된 크기의 Feature Vector를 뽑아낸다.

3. SVM (Linear Support Vector Machine)

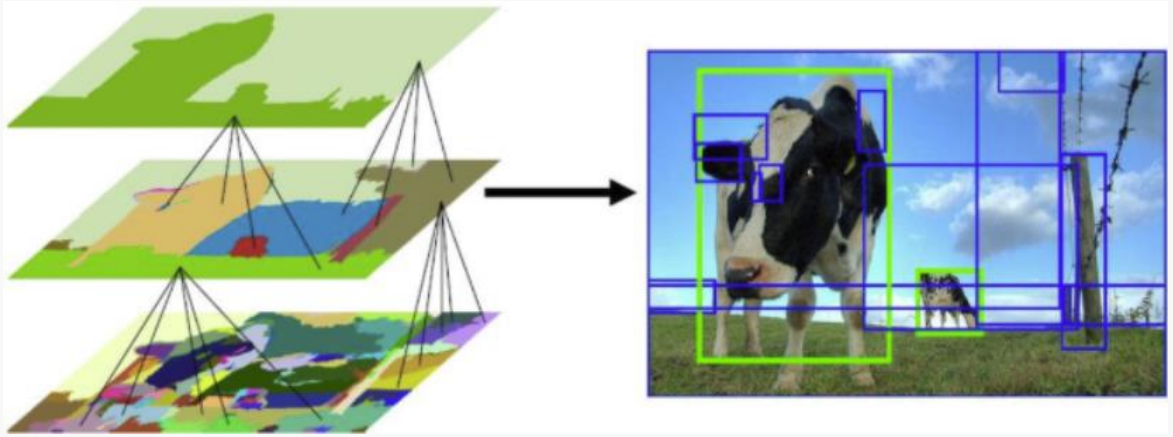
Classification을 위한 선형 지도 학습 모델

02 R-CNN

Region Proposal

Selective Search 알고리즘을 이용해 2000개의 Region을 선정

객체와 주변 간의 색감(color), 질감(texture) 차이, 다른 물체에 둘러 쌓여 있는지 (Enclosed) 여부 등을 파악해 물체의 위치를 파악할 수 있도록 하는 알고리즘



2000개의 Region은 CNN에 넣기 위해 같은 사이즈(224x224 pixel)로 통일 시키는 작업을 (Wrap) 거친다. (For Fully Connected Layer)

이미지 출처 : https://velog.io/@skhim520/R-CNN-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0

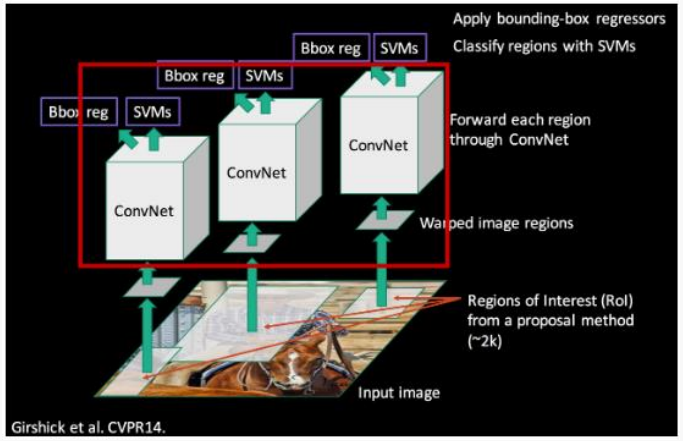
02 R-CNN

CNN (Convolution Neural Network)

앞의 단계에서 나온 결과물을 CNN에 넣어준다.

논문에서는 AlexNet의 구조를 사용하며 Object Detection을 위해 끝 부분만 수정

각각의 Region Proposal로부터 4096 차원의 feature vector를 뽑아내고, 고정된 길이의 feature vector를 만든다.



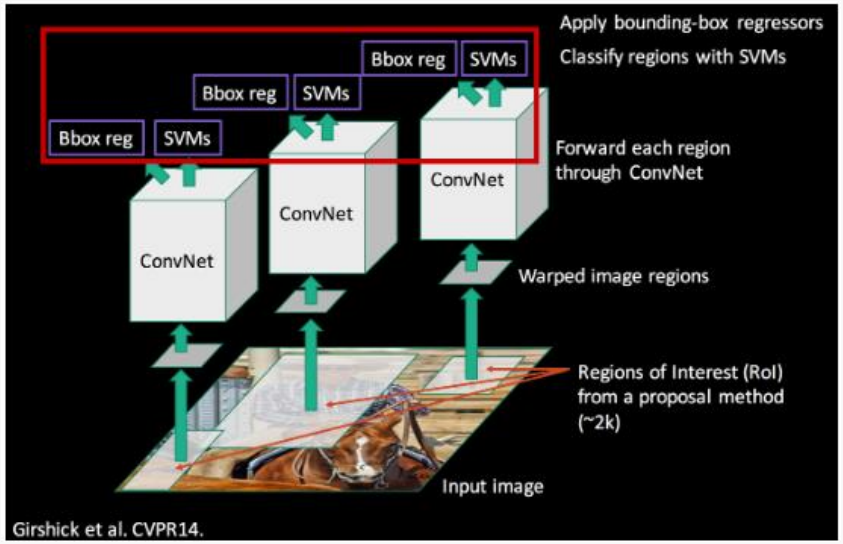
이미지 출처 : https://ganghee-lee.tistory.com/35

SVM (Support Vector Machine)

CNN에서 나온 결과를 Linear SVM을 통해 classification을 진행한다.

Softmax보다 더 좋은 성능을 보였기에 채택되었다.

Bounding Box Regression 작업을 통해 경계 박스를 더 정확하게 예측하도록 하는 작업도 추가된다.



이미지 출처 : https://ganghee-lee.tistory.com/35

02 R-CNN

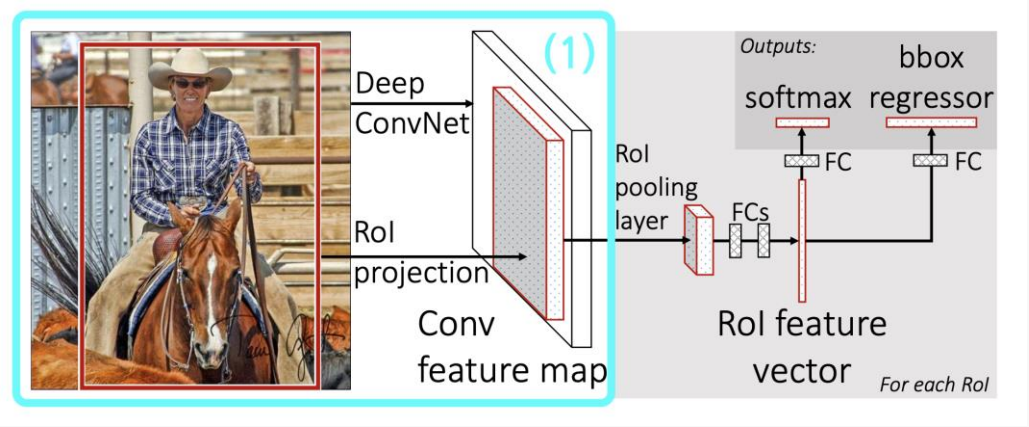
단점

1. CNN 연산을 2000번이나 해야 하므로 수행 시간이 느리다.
2. 총 세 가지의 모델이 한 번에 학습되지 않는다. 서로 연산을 공유하지 않는다.

-> Fast R-CNN 등장

03 Fast R-CNN

1. CNN (Convolutional Neural Network)



먼저, 전체 이미지에서

1) Selective Search로 Region Proposal을 얻어낸다.

2) 1번과 별개로 CNN을 통과시켜 feature map을 얻어낸다.

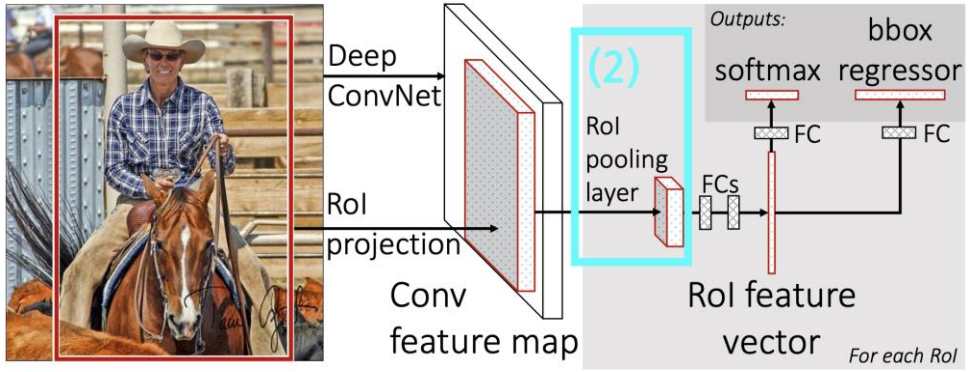
Region Proposal을 변형하지 않고 가지고 있고, CNN을 통해 얻은 feature map은 RoI Projection을 함

→ input image 1장으로부터 CNN Model에 들어가는 이미지는 2000장에서 1장으로 줄었다.

이미지 출처 : https://nuggy875.tistory.com/33

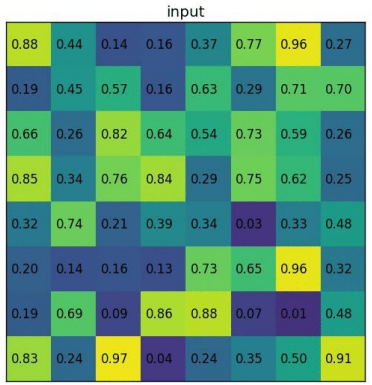
03 Fast R-CNN

2. RoI (Region of Interest) Pooling



앞서 Projection한 Bounding Box들을 RoI Pooling 하는 것이 Fast R-CNN의 핵심이다.

Projection시킨 RoI를 FCs에 넣기 위해서는 같은 크기의 Feature map이 필요하다. 이를 위해 RoI Pooling 수행

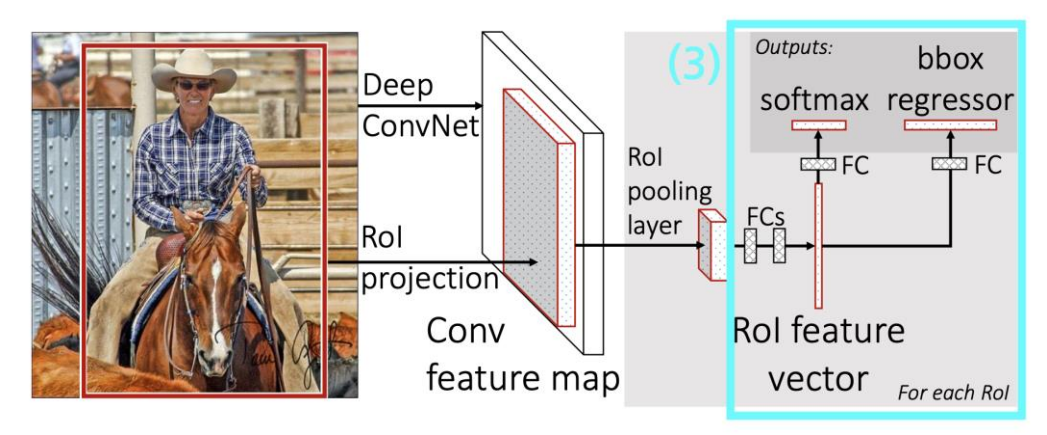


크기가 다른 Feature Map의 Region마다 Stride를 다르게 Max Pooling을 진행

이미지 출처 : https://nuggy875.tistory.com/33

03 Fast R-CNN

3. Classification & Bounding Box Regression



2번에서 얻은 Fixed Length Feature Vector를 FCs에 넣은 후

Classification과 Bounding Box Regression 진행

이 단계는 R-CNN과 비슷하지만 여기서는 Softmax를 사용하여 분류 진행

→ 전체적으로 속도 개선, end-to-end 방식으로 학습 가능
But, 여전히 Selective Search가 외부에서 진행되므로 이 부분이 속도의 Bottleneck이다.

이미지 출처 : https://nuggy875.tistory.com/33

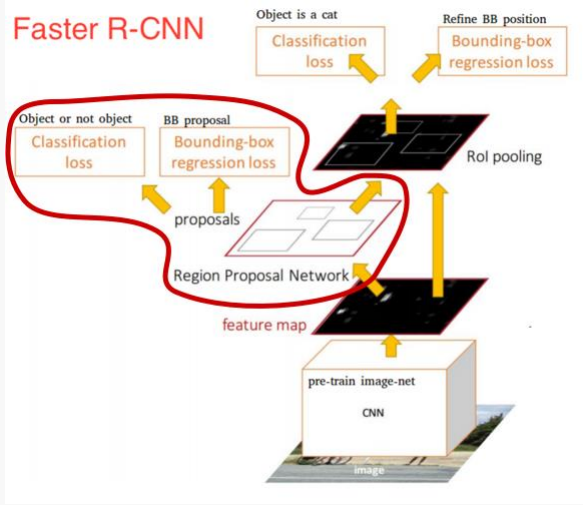
04 Faster R-CNN

Fast R-CNN보다 더 빠르게!

Region Proposal도 네트워크 구조 안에서 만들어보자!

Faster R-CNN = RPN + Fast R-CNN

기존의 Fast R-CNN은 그대로 사용하되, Region Proposal을 만드는 RPN을 추가적으로 도입하였다.



이미지 출처 : https://nuggy875.tistory.com/33

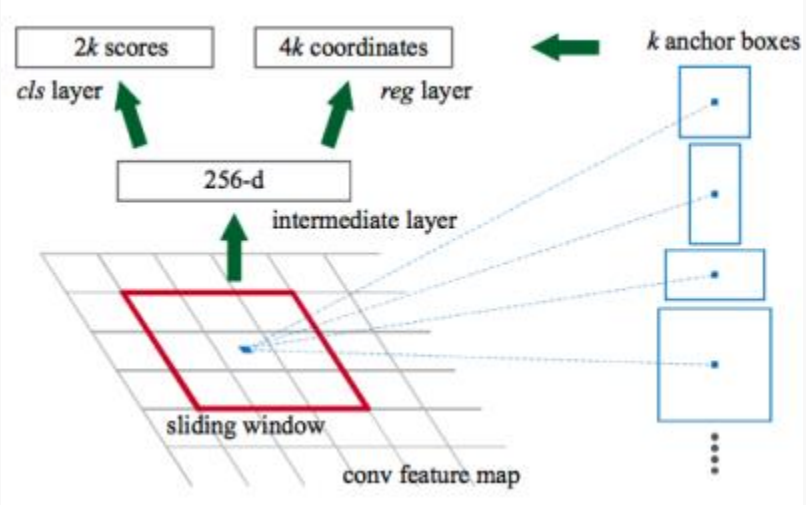
04 Faster R-CNN

Region Proposal Network (RPN)

RPN의 입력 값은 이전 CNN 모델(ZF Net or VGG-16)에서 뽑아낸 feature map이다.

Region proposal을 생성하기 위해 n x n window를 슬라이딩 시킨다.
그 결과, low dimensional feature를 얻게 되고, 이는 reg/cls layer로 보내진다. (n=3)

출력 값은 Region Proposal과 객체성 점수를 반환한다.



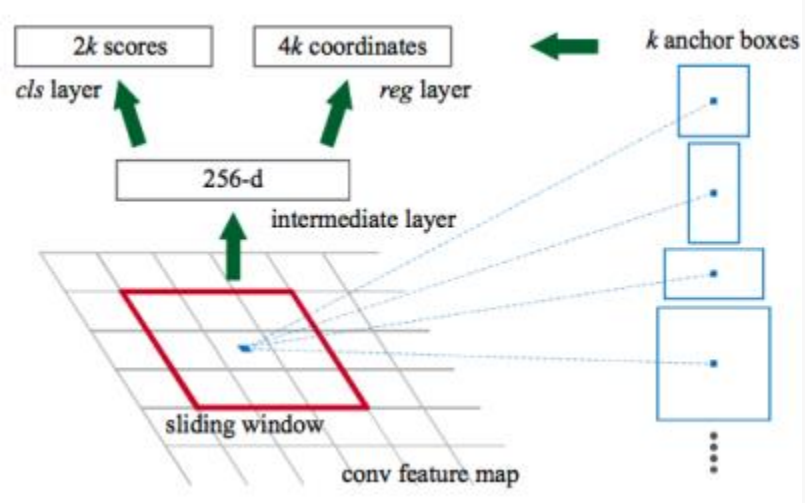
이미지 출처 : https://arxiv.org/abs/1506.01497

04 Faster R-CNN

Anchors

앞의 내용처럼 sliding window에 대해 다양한 region proposal을 예측하는데,
이 때, object의 비율이 어떻게 될 지 모르므로 미리 k개의 anchor box를 정의해 둔다.

논문에서는 서로 다른 3 개의 비율을 서로 다른 3 개의 크기의 박스로 anchor box를 정의하였다. (총 9개)



이미지 출처 : https://arxiv.org/abs/1506.01497

04 Faster R-CNN

Translation-Invariant Anchors

Object의 이동에 영향을 받지 않는 성질
Object가 어디에 있던 Proposal을 예측할 수 있어야 한다.

Multi-Scale Anchors as Regression Reference

Faster R-CNN은 anchor-based method를 사용하기 때문에 scale을 다루기 위한 여분의 비용이 들지 않는다는 장점이 있다.

RPN 학습을 위한 label

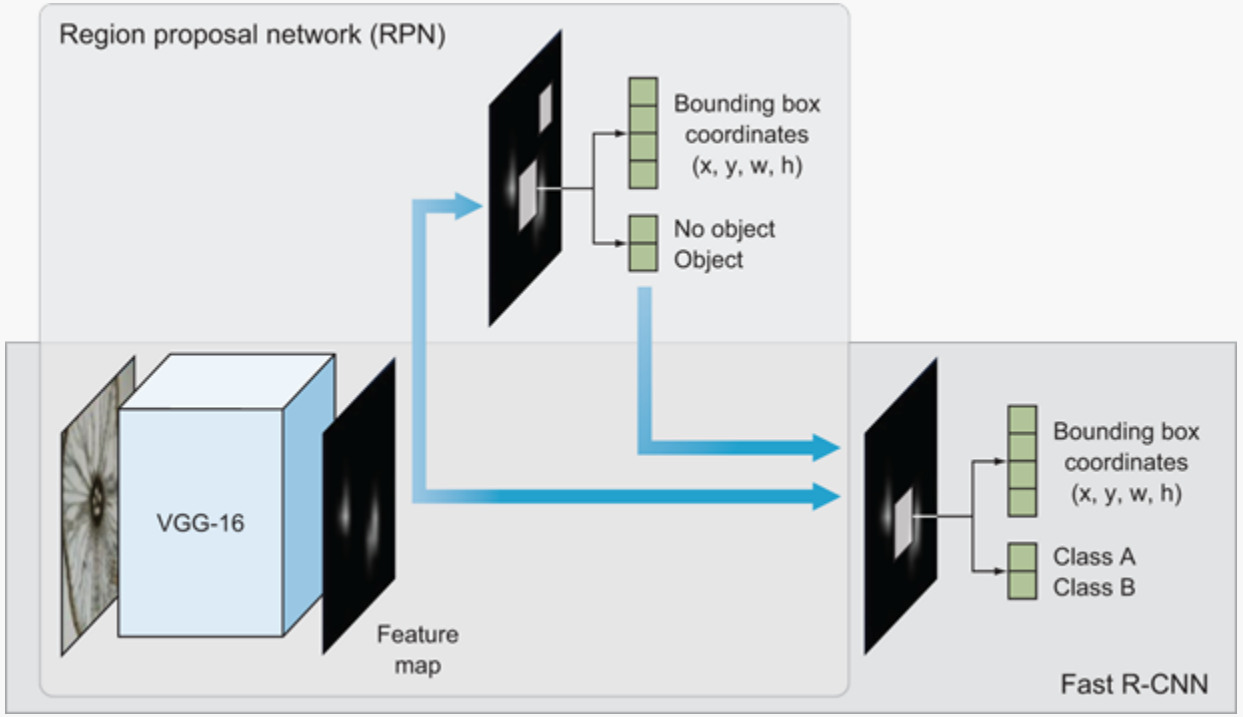
정답 box와 예측된 box의 IoU가 0.7 이상이거나 가장 높은 한 가지 박스에 positive 라벨
정답 box와 예측된 box의 IoU가 0.3 이하인 경우 negative 라벨

둘 다 아니면 학습에 도움이 되지 않으므로 무시

04 Faster R-CNN

Training RPN

미니배치는 하나의 이미지로 얻어진 anchor들 중에서 positive 128개 + negative 128개 총 256개의 anchor로 구성 (random으로 선정)



이미지 출처 : https://livebook.manning.com/concept/deep-learning/bounding-box

04 Faster R-CNN

어떻게 RPN과 Fast R-CNN이 특징을 공유할까?

1. Alternating Training : 말 그대로 번갈아 가며 학습한다는 의미로, 먼저 RPN 학습 후 Fast R-CNN을 학습한다. 그리고 이를 다시 RPN을 학습하기 위해 사용

→ 번갈아 가며 학습, 논문에서 사용한 방법

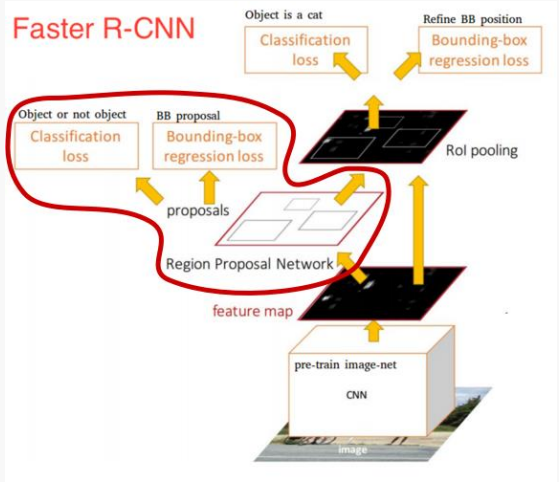
2. Approximate Joint Training : RPN과 Fast R-CNN을 완전히 한 개의 네트워크로 묶어서 학습을 진행하는 방법으로, 구현이 쉽고 학습시간이 적게 걸리지만, 정확도가 떨어진다.
3. Non-approximate Joint Training : 경계 박스 좌표의 기울기를 포함해서 역전파를 진행하는 방식으로 꽤 어려운 문제라 논문에서 디테일하게 다루지 않음

04 Faster R-CNN

4-Step Alternating Training

1. RPN만 학습
2. RPN에서 만든 Proposal을 이용해서 Fast RCNN 학습
3. RPN 앞의 Conv Layer를 완전히 고정한 상태로 RPN에 포함되어 있는 추가적인 Conv Layer에 대해서만 fine tuning 진행
4. 앞의 Conv Layer를 고정한 상태에서 Fast RCNN에만 포함되어 있는 레이어에 대한 학습 진행

3번과 4번을 통해 앞의 Conv 레이어가 RPN과 Fast RCNN이 서로 공유할 수 있게 된다.



이미지 출처 : https://nuggy875.tistory.com/33

04 Faster R-CNN

Implement Details

Anchor box는 3 scale : 128x128, 256x256, 512x512
3 aspect ratio : 1:1, 2:1, 1:2

이미지의 경계를 넘는 anchor는 무시
이로 인해 2만 개의 anchor가 있다고 하면 약 6천 개의 anchor만 학습

또한, 많은 proposal이 중복되는데, 이 중복을 줄이기 위해 non-maximum suppression (NMS)를 사용한다. NMS가 정확도에 크게 영향을 끼치지는 않는다.

NMS를 거친 후 top N개의 Proposal만 사용

학습 때의 N과 평가 때의 N을 다르게 사용 가능

04 Faster R-CNN

Table 2: Detection results on **PASCAL VOC 2007 test set** (trained on VOC 2007 trainval). The detectors are Fast R-CNN with ZF, but using various proposal methods for training and testing.

train-time region proposals		**test-time region proposals**		
method	# boxes	method	# proposals	mAP (%)
SS	2000	SS	2000	58.7
EB	2000	EB	2000	58.6
RPN+ZF, shared	2000	RPN+ZF, shared	300	**59.9**
ablation experiments follow below				
RPN+ZF, unshared	2000	RPN+ZF, unshared	300	58.7
SS	2000	RPN+ZF	100	55.1
SS	2000	RPN+ZF	300	56.8
SS	2000	RPN+ZF	1000	56.3
SS	2000	RPN+ZF (no NMS)	6000	55.2
SS	2000	RPN+ZF (no *cls*)	100	44.6
SS	2000	RPN+ZF (no *cls*)	300	51.4
SS	2000	RPN+ZF (no *cls*)	1000	55.8
SS	2000	RPN+ZF (no *reg*)	300	52.1
SS	2000	RPN+ZF (no *reg*)	1000	51.3
SS	2000	RPN+VGG	300	59.2

Table 5: **Timing** (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. "Region-wise" includes NMS, pooling, fully-connected, and softmax layers. See our released code for the profiling of running time.

model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	**10**	47	**198**	**5 fps**
ZF	RPN + Fast R-CNN	31	**3**	25	**59**	**17 fps**

Table 10: **One-Stage Detection *vs.* Two-Stage Proposal + Detection**. Detection results are on the PASCAL VOC 2007 test set using the ZF model and Fast R-CNN. RPN uses unshared features.

	proposals		detector	mAP (%)
Two-Stage	RPN + ZF, unshared	300	Fast R-CNN + ZF, 1 scale	58.7
One-Stage	dense, 3 scales, 3 aspect ratios	20000	Fast R-CNN + ZF, 1 scale	53.8
One-Stage	dense, 3 scales, 3 aspect ratios	20000	Fast R-CNN + ZF, 5 scales	53.9

Table 8: Detection results of Faster R-CNN on PASCAL VOC 2007 test set using **different settings of anchors**. The network is VGG-16. The training data is VOC 2007 trainval. The default setting of using 3 scales and 3 aspect ratios (69.9%) is the same as that in Table 3.

settings	anchor scales	aspect ratios	mAP (%)
1 scale, 1 ratio	128^2	1:1	65.8
	256^2	1:1	66.7
1 scale, 3 ratios	128^2	{2:1, 1:1, 1:2}	68.8
	256^2	{2:1, 1:1, 1:2}	67.9
3 scales, 1 ratio	$\{128^2, 256^2, 512^2\}$	1:1	**69.8**
3 scales, 3 ratios	$\{128^2, 256^2, 512^2\}$	{2:1, 1:1, 1:2}	**69.9**

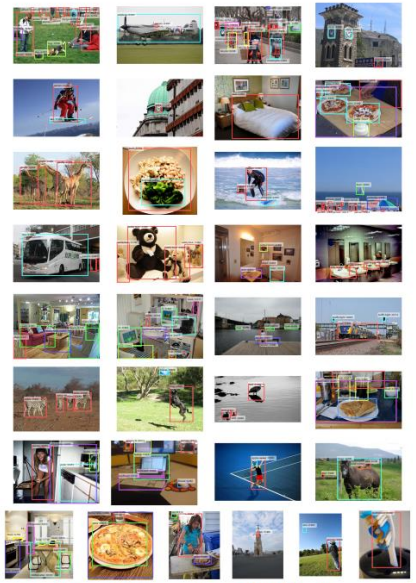


Figure 6: Selected examples of object detection results on the MS COCO test-dev set using the Faster R-CNN system. The model is VGG-16 and the training data is COCO trainval (42.7% mAP@0.5 on the test-dev set). Each output box is associated with a category label and a softmax score in [0, 1]. A score threshold of 0.6 is used to display these images. For each image, one color represents one object category in that image.

THANK YOU -

경청해주셔서 **감사합니다.**