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

**Abstract**

SPPnet, Fast R-CNN과 같이 발달된 Object Detection Network는 running time 단축은 성공하였지만, Region proposal 자체를 별도로 Selective Search(SS)와 같은 알고리즘을 사용해서 CPU기반으로 수행했기에 Region Proposal이 굉장히 느린 문제(각 이미지당 기본 2초)가 있다.

따라서, 본 논문에서는 Selective Search Algorithm을 이용하지 않고 Region Proposal Network(RPN)을 제안해서, 이러한 RPN이 CPU가 아닌 GPU에서 돌아갈 수 있도록 함으로써 Region Proposal에 걸리는 시간을 획기적으로 단축했다.

이러한 RPN은 Fully Convolutional Network 형태로 되어 있으며, 그 물체가 고양이인지 강아지인지에 대한 Class정보까지 예측하지는 않고 어떠한 위치에 물체가 실존하는가 그렇지 않은가 정도만 판단한다.

또한, Region Proposal까지 GPU에서 돌릴 수 있게 되면서, 모델 전체를 GPU로 처리할 수 있게 되었고, 이로 인해 end-to-end 방식으로 학습이 가능해지면서 전체적인 학습시간이 비약적으로 단축되었다. (초당 5장의 이미지 처리가 가능해지면서 상당히 Real-Time에도 적용할 정도로 빠른 수준까지 도달)

RPN에서 만들어진 각각의 위치에 대한 정보와 그 물체의 크기(Bounding Box)에 대한 정보를 바탕으로, 그 위치에 있는 사물이 어떤 Class로 분류가 되는지는 Fast-RCNN구조를 가져와서 사용하였다.

즉, 전체적인 구조의 경우, Faster R-CNN은 RPN와 Fast R-CNN으로 구성이 되는데, 이 두가지 Network는 동일한 Feature를 공유하도록, 즉, 제일 앞부분에 일종의 Feature Extractor로써 CNN Layer를 두어서, Feature를 추출하는 과정이 동일하게 했다. 이렇게 해서 성능도 좋고, 시간도 단축시켰다.

한편, 본 논문에서는 RPN을 통해서 사물의 위치가 감지가 되면, 그 위치에 있는 사물이 어떤 사물인지 맞추는 방식으로 진행이 되기 때문에 이러한 RPN을 이 논문에서는 Attention Mechanism과 유사하다고 언급했다.

성능 검증 및 테스트를 위해 사용한 이미지는 PASCAL VOC 2007, PASCAL VOC 2012, MS COCO Dataset

**Introduction & Related Work**

Abstract에서 설명했듯이, 본 논문의 RPN는 물체가 있을 법한 위치에 대해 Regression을 이용해서 Bounding Box를 더 잘 그리도록 했고, 그 위치에 실제로 물체가 존재하는가 아닌가 예측하는 작업까지 진행하였다.

또한, 속도 및 Detect 정확성 상의 이점을 얻기 위해 앵커박스의 크기와 스케일을 다양하게 지정했으며,

**Area :   
Ratio : 1:1, 1:2, 2:1**

이와 같이 본 논문에서는 총 **9개의 앵커박스**를 이용하였다.

**Anchor Box**

기본적인 크기는 3x3의 이며, 이러한 박스를 왼쪽 위부터 슬라이딩하면서 256 차원으로 mapping을 수행.  
**k개의 anchor box**가 있다고 가정하면**,**

각각의 box에 대해 물체가 있는지 없는지 Yes/No를 Softmax로 판별(Classification Layer) - 2k 개  
각각의 box에 대해 좌표를 출력(Regression Layer) – **x, y, w, h**  - 4k 개

예) 만약, Convolutional Feature Map의 Size가 W X H 라면, 그리고, k개의 anchor box Type이 있다면,  
총 W X H X k개의 Anchor Box를 사용하는 것이다.  
W=60, H=40이면, 총 2400 X k만큼의 Anchor 박스를 이용한 것! (본 논문엔 약 20,000개의 Anchor)

단, 본 논문에서는 Cross Boundary Anchor는 지워서, 각 이미지당 약 6000개 정도의 Anchor만 이용!

***전체적인 FLOW***

1. **CNN을 통해 이미지의 Feature Map을 뽑는다.**  
   한 장의 이미지를 Convolutional Layer에 넣어서 Convolutional Feature Map을 뽑고   
   이러한 Feature Map은 RPN와 Fast RCNN에 공유될 수 있는 구조이며,  
   이러한 Feature Map은 ZF net, VGGNET과 같은 다양한 CNN Architecture(=Backbone)으로 구성될 수 있다.

또, 이러한 Feature Map에 대해서 다양한 크기와 비율을 가지는 Anchor Box로 Sliding을 하면서 각 위치에 대해 Prediction(Propose)을 진행한다.

1. **Region Proposal Network는 어떤 위치에 물체가 존재하는지 존재하지 않는지 예측하고 존재하면 어떤 바운딩 박스안에 위치하는지 예측하는 역할을 한다.**
2. **RPN에서 나온 결과를 Classsifier에 넣어서 Region Proposal영역의 물체가 어떤 것인지 맞춘다.**즉, RPN를 통해 특정 영역에 대해서만 Classification을 시키기 때문에 Attention 기법과 유사하다고 함.

Faster RCNN의 또 다른 장점은, **translation-invariant property**

이미지에 translation 변형이 가해져도 이러한 이동에 있어서 invariant한 장점이 있음.

일반적으로 **Multi scale로 Prediction**을 하는 방법에는 대표적으로 2가지

1. **이미지나 피처의 피라미드 방식**

이미지나 피처의 크기를 피라미드 형식으로 변형하면서 각각의 스케일에 대해 Feature Map을 일일이 계산해야 되기에 시간이 오래 걸리는 문제가 있다.

1. 다양한 스케일 내에서 슬라이딩 윈도우 진행(**Sliding window 방식**)

Sliding Window를 진행하고, regression으로 정확한 위치를 찾도록 한다.

1. 본 논문에서는, 기본적으로 Sliding Window를 사용하되, **앵커 기반의 Sliding Window Method**이다.

이미지나 Feature Map은 Convolutional Layer를 한 번만 통과하며, 그렇게 추출된 Feature Map에 대해 Pyramid형식의 다양한 Anchor로 Sliding Window하며 Prediction을 진행한다. 비용과 시간면에서 효율적!

**Loss Function**

RPN에서는 각각의 앵커에 대해 이진 분류를 진행. 즉, 그 위치에 물체가 존재하는지 여부를 확인.

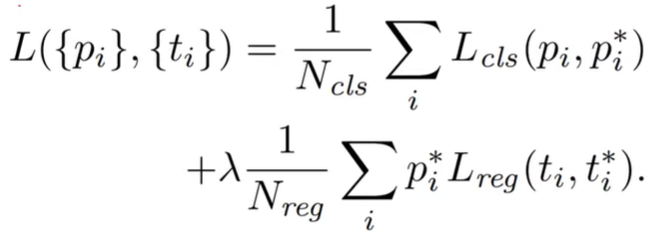
이때 정답 레이블로 positive를 주는 경우는 IoU (Intersection over Union) 교차 면적이 0.7보다 클 때만 준다. 따라서, 하나의 정답 ground truth 박스(물체를 Detecting한 Bounding Box가 물체에 대해 완전히 꽉 채워진 경우)가 있을 때, 이것을 예측하는 여러 Bounding Box가 있을 텐데, IoU가 0.7보다 큰 Bounding Box는 모두 positive로 될 수 있다. 즉, 하나의 ground truth box는 여러 anchor 박스에 대해 positive로 label될 수 있다는 점을 기억하기!

또, IoU가 0.7을 넘는 경우가 하나도 없을 수 있기에, 그렇다면 그나마 가장 IoU값이 높은 경우를 0.7이 안되었음에도 채택할 필요가 있다.

한편, IoU가 0.3보다 낮으면 Negative Label을 붙임.

또, Pos도 Neg도 아닌 Anchor박스는 Training objective function에 영향을 주지 않도록 함.

따라서, **Loss Function Formula**는 다음과 같이 정의될 수 있다.



Cls= classification Reg=regression

* = ground truth value(정답 레이블) :: 해당 앵커가 pos면 1부여, neg면 0 부여
* 아래쪽 식은 해당 앵커가 neg면 =0 이므로, 식 전체가 무시되는 영역

왜냐하면, 현재 앵커의 위치에 물체가 존재할 때만, 즉 =1일 때만,  
그 물체가 어디에 있는지 판단할 수 있도록 하는 것이 효율적이기 때문!

* = tx ty tw th 로 Bounding Box의 정보를 가지는 tuple data.
* 각각의 loss에 대해 가중치를 두고 학습시키기 위해 Weight parameter를 곱함.

본 논문에서는 미니 배치 사이즈나, Anchor location의 계수(람다)와 같은 정보로 Normalization을 진행,

Lamda는 10으로 할 때 충분히 좋은 성능이 나왔다고 하지만,

가중치 파라미터가 결과에 그렇게 크게 영향을 주지 않았기에 적절히 좋은 값을 잘 설정하라고 함.

* tx ty tw th의 값은 R-CNN논문에서 제시된 식을 그대로 사용
* 기본적으로 x랑 y는 Bounding Box의 중간점 위치에 대한 정보
* RPN은 모든 앵커에 대해 전부 로스를 계산하지 않고 랜덤하게 256개의 앵커만 샘플링해서 학습을 진행할 수 있도록 하고 pos앵커와 neg앵커의 비율이 1:1정도가 되도록 학습을 진행시킴.
* Faster RCNN에 맞게 기존의 CNN을 학습시킬 최적의 parameter에 대해서도 소개함. (Learning rate, weight에 대한 정보)

**RPN과 Fast R-CNN이 어떻게 Feature Map을 공유하는가?**

기존 Fast R-CNN의 Detection Network는 그대로 사용함.

본 논문에선 Feature를 share하면서 학습할 수 있는 방법에 대해 언급함. 총 4가지 단계!

1. **Alternation training - 교차 학습**

먼저 RPN을 학습하고, Fast RCNN을 학습하고,

Fast RCNN으로 다시 학습이 된, 즉 fine tuning이 된 network를 다시 RPN으로 학습

이 과정을 반복.

**전체 학습을 4단계로 진행.**

1. RPN 만 학습

2. RPN에서 만든 Proposal Region에 대해 Fast RCNN(Classifier)을 학습.

3. 여기까지는 앞쪽에있는 conv layer가 온전히 공유되지가 않는 문제가 있음.

따라서, 세번째 부터는 conv layer를 완전히 고정한 채로,  
RPN에 포함되어 있는 추가적인 Convolution Layer에 대해 Fine Tuning을 진행.

4. 마찬가지로, conv layer를 완전히 고정한 채로, fast RCNN에만 포함되어있는 Convolution Layer에 대해서만 학습을 진행.

**이를 통해 앞쪽의 conv layer가 RPN및 fast RCNN에 공유되는 셈.**

**+) 대부분의 Bounding Box는 서로 상당히 중복되는 경우가 잦음.**

* **Class가 동일한 Bounding Box는 가지치기하기 위해서**

**NMS(Non-Maximum Suppression)이라는 기법을 이용**

NMS를 위한 IoU의 Threshold값으로는 0.7을 선정!

**이렇게 하면 최종적으로 이미지당 약 2000개 정도의 Region Proposal만 나온다고 한다.**

**Experiments**

**PASCAL VOC 2007**

5000개의 TrainVal Data  
5000개의 Test Data로 구성

20개의 Class

**테이블이(가) 표시된 사진

자동 생성된 설명**

**MS COCO Dataset**

80개의 Class

**Ablation**

특정한 메소드의 Component를 한 개씩 제거해보면서, 그 Component가 얼마나 중요한 특성을 가지고 있는지 확인해보고 있는 과정.