Faster R-CNN final-report

ByeongGeun Shin[1],
[1]Pusan National University.

1 Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks Paper Summary

1.1 Intro

이 논문에서는 기존 Object Detection 모델의 Region Proposal Algorithm의 한계를 극복하기 위해 RPN(Region Proposal Network)을 제안한다. 이전의 Fast R-CNN[1]은 Region Proposal 계산이 Bottleneck이라 판단해 이를 줄이는 방식으로 속도를 높였다. RPN은 Fully Convolutional Network로, 이미지 내 모든 픽셀에 대해 물체의 경계와 확률을 예측할 수 있도록 설계되었다. 이를 Fast R-CNN과 결합해 하나의 네트워크로 구성함으로써 더 효율적이고 빠른 모델을 만들 수 있었다. 이러한 구조 덕분에 Faster R-CNN은 추론 속도가 빠르면서도 높은 정확도를 유지할 수 있다.

1.2 Introduction

이전에 사용되던 Region Proposal Method는 계산량이 많아 연산 속도가 느린 단점이 있었다. Selective Search는 높은 정확도를 제공했으나 속도가 느렸고, EdgeBoxes는 속도와 정확성의 Trade-off가 적절하게 이루어진 방법이었다. GPU를 통해 region-based CNN의 속도가 개선되었지만, Proposal을 추출하는 알고리즘이 여전히 CPU에서 수행되어 전체 프로세스가 느렸다. Faster R-CNN은 이를 해결하기 위해 CNN을 사용한 Region Proposal 추출 방식을 제안하였다. RPN은 Convolution 연산을 통해 Region Proposal을 생성하기 때문에 속도와 정확도 면에서 효율적이다.

1.3 Faster R-CNN

Faster R-CNN은 두 가지 모듈로 구성된다. 첫번째 모듈은 Region Proposal을 생성하는 Deep Convolutional Network이다. 두 번째 모듈은 Fast R-CNN Detector로 첫 번째 모듈에서 뽑아낸 Region Proposal을 사용한다. 이 두 모듈을 통합하여 Object Detection을 수행한다.

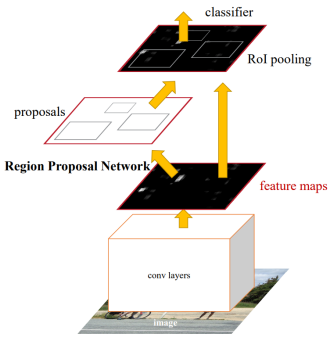

Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Figure 1: Faster R-CNN Process.

1.3.1 Region Proposal Networks

RPN은 이미지를 입력으로 받아 각 픽셀 위치에서 객체의 Score와 사각형 형태의 Proposal을 추출한다. Fully Convolutional 구조를 채택하여 Fast R-CNN과의 Feature를 공유해 연산 효율을 높였다. RPN에서는 Feature Map의 각 위치에 대한 다양한 크기와 비율을 갖는 앵커를 통해 여러 Proposal을 생성한다. 이를 통해 다양한 크기의 객체를 인식할 수 있도록 했다. 앵커는 Sliding Window 기반의 Centralized Region Proposal로 활용되어 모델의 크기를 줄이고 Overfitting 위험을 낮추었다.

1.3.2 Loss Function

RPN의 학습 과정에서는 각 Anchor에 대해 Positive와 Negative Label을 할당한다. Positive Label은 Ground Truth와의 IoU가 가장 높은 Anchor와, IoU가 0.7 이상인 Anchor에 부여된다. Negative Label은 IoU가 0.3 이하인 경우에 할당된다. 이렇게 부여된 Label을 바탕으로 RPN은 Box-regression과 Box-classification을 동시에 수행하도록 Multi-task Loss를 계산한다. 또한, 각 앵커들은 자신이 맡은 한 개의 크기에 대해서 찾기 때문에 다른 크기의 앵커들과 Weight를 공유하지 않는다. 이 덕분에 Feature의 크기가 변하지 않아도 다양한 크기의 객체를 탐지할 수 있다.

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

Figure 2: Faster R-CNN Multi-task Loss.

1.3.3 Training RPN

RPN은 일반적인 Neural Network처럼 end-to-end로 학습이 가능하다. 학습 시 Image Centric Sampling 방식을 통해 다양한 RoI를 샘플링하며, 각 mini-batch마다 Positive와 Negative Anchor의 비율을 조정해 학습한다. Negative Anchor가 Positive Anchor보다 훨씬 많기 때문에, 256개의 Anchor 중 Positive와 Negative Anchor를 1:1 비율로 선정하여 Loss를 계산한다. 이를 통해 모델이 Negative Anchor에 대한 Bias를 줄이고 안정적으로 학습할 수 있도록 한다. 또한, 작은 영역을 포함하는 RoI에서 특정 객체를 감지하는 데 유리한 학습 환경을 제공한다.

1.4 Sharing Features for RPN and Fast R-CNN

RPN과 Fast R-CNN은 Convolution Feature를 공유함으로써 연산 효율을 크게 높였다. 이 논문에서는 Feature Sharing을 위한 세 가지 학습 방식을 제안했다. 첫 번째는 Alternating Training으로, 먼저 RPN을 학습하고 이를 통해 생성된 Proposal을 이용해 Fast R-CNN을 학습하는 방식이다. 두 번째는 Approximate Joint Training으로, 두 모듈을 결합하여 한 번에 학습시키는 방법이다. 마지막으로 Non-approximate Joint Training은 Fast R-CNN의 RoI Pooling Layer가 Box 좌표도 gradients에 포함되도록 하여 좀 더 정확한 Joint Training을 수행한다.

RPN을 통해 뽑아낸 Proposal은 겹치는 부분이 많이 존재한다. 이를 줄이기 위해 Class Score에 기반한 Non-maximum Suppression (NMS)를 도입한다.

1.5 Experiments

PASCAL VOC 2007 Dataset을 이용해 진행한 실험에서 평가지표는 mAP로, Selective Search나 EdgeBoxes보다 ZF[3]에 RPN을 사용한 것이 mAP도 높았고, 사용한 Proposal 수도 적었다. 또한 RPN과 Fast R-CNN의 Convolution Feature Sharing이 성능에 기여함을 확인했으며, RPN의 Region Proposal은 Selective Search 대비 적은 Proposal 수에서도 높은 정확도를 유지했다. 또한, VGG Backbone을 사용하는 RPN이 ZF와 Selective Search를 사용했을 때보다 성능이 우수한 것으로 나타났다.

VGG-16[2]을 Proposal과 Detection에 모두 적용하여 실험한 결과, RPN+VGG에 Feature Sharing을 하지 않은 것이 68.5%로 Selective Search보다 높은 성능을 보였다. Feature Sharing을 진행하면서 성능이 더 올라갔고, 2012, COCO Dataset을 추가하면서 더욱 성능이 증가하였다. 실행 속도에선 Selective Search보다 RPN이 훨씬 빠른 속도를 보였다.

IoU Threshold 값에 따른 Recall을 계산하는 실험에선 RPN은 Proposal이 2000부터 300까지 줄어도 성능 차이가 많이 나지 않는 것을 볼 수 있다. 이를 통해 RPN은 적은 Proposal을 통해서도 물체를 잘 찾아낼 수 있음을 알 수 있다.

1.6 Conclusion

RPN은 Region Proposal을 효과적으로, 정확하게 생성하는 방법을 제안한다. Convolution Layer를 공유하며 Region Proposal을 추출하는 과정에서 Cost를 줄였다. 이 당시 Real-time에 근접한 속도를 보였고 Region Proposal의 Quality를 향상 시켰고 따라서 Object Detection Task의 성능도 올랐다.

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

2.1 Intro

이번 실험의 목적은 Faster R-CNN 네트워크를 구현하여 객체 탐지(Object Detection) 작업에서의 성능을 평가하고, 다양한 실험을 통해 모델의 구현방법을 연습하는 것이다. Faster R-CNN은 Region Proposal Network (RPN)를 통해 객체가 있을 가능성이 높은 영역을 빠르게 탐지하는 메커니즘을 기반으로 한 모델이다. 본 실험에서는 Faster R-CNN의 성능을 이미지 내 여러 객체의 위치와 클래스 탐지에서 측정하며, 모델 구조와 각 단계의 역할에 대한 깊이 있는 이해를 목표로 한다.

2.2 Practice Methods

2.2.1 Dataset

본 실험에서는 Faster R-CNN 모델을 훈련하고 평가하기 위해 주어진 링크에서 MS COCO 어노테이션 규격의 pascal voc 데이터셋을 다운로드하여 사용하였다. 데이터셋은 훈련용 데이터와 테스트 데이터로 구성되고, 각 데이터는 JSON 파일 형식으로 주석 정보가 제공된다. 이러한 구조의 데이터셋은 다양한 객체 종류와 위치 정보를 포함하고 있어 객체 탐지 모델의 학습과 평가가 가능하다. 이후 객체가 포함된 여러 예시 이미지들을 모델에 입력하여 추론을 수행하였다.

2.3 Mission

2.3.1 Anchor Box

Anchor Box 좌표값 생성에선 Anchor Box Generator는 이미지 내 객체가 존재할 가능성이 높은 위치에 anchor box 좌표를 생성하여 탐지할 영역을 제안하는 역할을 한다. Faster R-CNN에서는 각 중점 좌표에서 다양한 크기와 비율로 9개의 anchor box를 생성하는데, 이는 다양한 객체 크기에 대응하기 위해 필요하다. 기본 Anchor Box는 이미지의 중간 지점인 base size의 절반을 기준으로 설정된다. 이 중점은 각 anchor box의 중심 위치가 되며, 이를 기준으로 다양한 비율(ratio)과 스케일(scale)을 적용해 anchor box 크기를 조정한다.

2.3.2 Proposal Creator

이후 Proposal Creator는 RPN에서 계산된 위치와 anchor 정보를 바탕으로 RoI(Region of Interest)를 생성하는 함수이다. RoI는 학습 또는 테스트 시점에 객체가 존재할 가능성이 높은 영역을 제안하며, 최종적으로는 NMS(Non-Maximum Suppression) 알고리즘을 통해 최종 anchor box를 선택하여 RoI를 줄이는 역할을 수행한다. 이 과정은 RPN이 제안한 모든 영역에서 일정 크기 이상의 영역만 선택하고, 이 영역 중에서도 중복되는 부분을 제거해 가장 가능성 높은 상위 2000개의 RoI를 선택하여 성능을 최적화한다.

2.3.3 Region Proposal Network & Faster R-CNN

RPN은 이미지에서 객체가 존재할 가능성이 높은 후보 영역을 제안하기 위해 사용한다. VGG16 백본에서 추출한 feature map을 입력받아 중간 feature map을 생성하고, 최종적으로 bounding box의 위치와 객체 가능성을 나타내는 feature map을 생성한다.

실험에서는 1024x1024 이미지 크기를 기준으로 백본 모델(VGG16)에서 나온 feature map에 3x3 convolution 연산을 적용하여 중간 feature map을 생성하고, 중간 feature map에 1x1 convolution을 적용하여 bounding box 좌표 예측을 위한 feature map을 생성해 Bounding Box 좌표 예측을 수행한다. 해당 출력은 (torch.Size([1, 36, 64, 64]))로, 각 위치에서 9개의 anchor box 좌표값을 예측할 수 있도록 36개의 channel을 갖는다. 각 anchor box마다 (y min, x min, y max, x max) 형태의 bounding box 좌표가 예측된다.

그리고 중간 feature map에 또 다른 1x1 convolution을 적용하여 객체 유무를 판단하는 feature map도 생성한다. 출력 형상은 (torch.Size([1, 18, 64, 64]))로, 각 위치에서 9개의 anchor box에 대해 객체가 있을 가능성을 2개의 channel로 구분하여 나타낸다. 이로써 anchor box마다 객체 유무를 예측할 수 있다.

마지막으로 앞서 생성된 feature map을 활용하여 최종 RoI를 생성한다. Proposal Creator는 anchor box 좌표와 객체 가능성 점수를 기반으로 NMS를 수행해 중복되는 박스를 제거하고, 상위 2000개의 RoI만 남긴다. 이러한 과정을 통해 RPN은 최종적으로 중요한 후보 영역만을 모델에 제공하여 객체 탐지 성능을 높인다.

이렇게 구성된 네트워크를 합쳐 최종적으로 Faster R-CNN 클래스를 구현하였다.

2.4 Training

FasterRCNNTrainer 클래스는 Faster R-CNN 학습 과정을 효율적으로 관리하는 역할을 수행한다. 먼저, AnchorTargetCreator와 ProposalTargetCreator를 통해 anchor와 RoI에 대한 ground truth를 생성하여 학습에 필요한 데이터를 준비한다. 이 데이터는 RPN 및 RoI 네트워크에서 위치 손실과 분류 손실을 구하는 데 사용된다. forward 메서드는 모델의 주요 모듈(VGG, RPN, RoI 헤드)을 순차적으로 호출하여 예측을 수행하고 손실을 계산한다. train step 메서드는 매 학습 단계에서 손실을 역전파하고 가중치를 최적화하며, update meters와 reset meters로 평균 손실 값을 기록 및 초기화하여 학습 상태를 모니터링한다. save와 load는 모델의 학습 상태를 저장 및 로드하여 학습 중단 후 재개를 가능하게 한다.

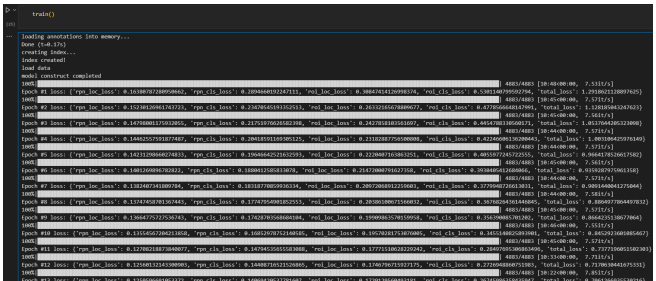

Figure 3: Faster R-CNN Training Print.

학습에는 3시간정도가 소요되었으며, 학습 시간이 길어 VSCode를 계속 띄워두는 게 아닌, 터미널에서 Screen으로 백그라운드 터미널을 띄우고 Nbconverter로 백그라운드 주피터 실행을 통해 훈련을 수행하고 실험 결과를 관측하였다.

최종 15에포크 결과는 rpn loc loss': 0.1241322684034365, 'rpn cls loss': 0.1388325185219161, 'roi loc loss': 0.17160542999412712, 'roi cls loss': 0.26338986655441243, 'total loss': 0.6979600836324727 이다.

2.5 Inference

추론은 테스트 데이터셋에 대해 Faster R-CNN 모델을 사용하여 객체 탐지 예측을 수행한다. 테스트 데이터셋을 로드하고, 학습된 모델을 불러와 평가를 진행한 후, 예측된 객체의 라벨, 점수, bounding box 좌표를 추출한다. 추출된 예측 정보를 기준으로 결과를 CSV 파일 형식으로 저장하여 제출 파일을 생성한다.

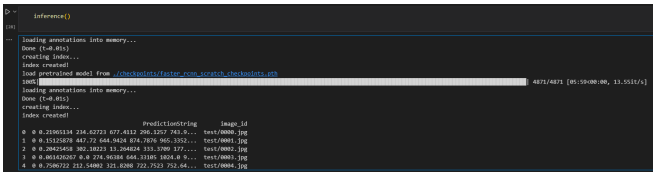

Figure 4: Faster R-CNN Inference.

위 추론 결과는 객체 탐지 모델(Faster R-CNN)을 통해 생성된 예측 결과이다. 4871개 데이터에 대해 5분 59초 정도가 걸렸고 초당 13개 정도의 이미지를 처리하였다. 각 행은 하나의 이미지에 대한 예측을 나타내며, PredictionString 컬럼에는 해당 이미지에서 검출된 객체들의 정보가 담겨 있다. 각 객체의 정보는 "라벨, 점수, bounding box 좌표(좌측 상단 x, 좌측 상단 y, 우측 하단 x, 우측 하단 y)" 형식으로 제공된다. 예를 들어, 첫 번째 행에서는 이미지 test/0000.jpg에서 검출된 객체에 대한 예측 결과가 포함되어 있다. image id는 해당 이미지 파일의 경로를 나타내고, 이 파일은 CSV 형태로 저장하여 결과로 같이 제출하였다.

[1] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[3] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.