Focal Loss for Dense Object Detection

Abstract

지금까지 one-stage Network(YOLO,SSD 등) 의 Dense Object Detection 는 two-stage Network(R-CNN 계열) 에 비해 속도는 빠르지
우리는 그 이유가 극단적인 클래스 불균형 문제 때문이라는 것을 발견했다.
이 문제를 해결하기 위해서 클래스 분류에 일반적으로 사용되는 크로스 엔트로피 로스 함수를 조금 수정한 Focal Loss 를 제안한다.
Focal Loss 는 잘 분류된 예제들에 대해서는 작은 가중치를 부여하는 반면 분류하기 어려운 일부 예제들에는 큰 가중치를 부여해서 학
그래서 쉽게 분류되는 대부분의 negative 샘플들에 의해서 학습이 압도 되는 문제를 해결 할 수 있다.
또한 Focal Loss 를 사용하면 one-stage Network 에서도 two-stage Network 이상의 성능을 얻을 수 있다.
RetinaNet 은 one-stage detector 만큼 빠르면서도 기존의 모든 최고 성능의 detector 들을 능가하는 정확도를 보여준다.

1. Introduction

현재 최고의 성능을 내는 object detecots 는 two-stage 기반의 proposal-driven 방식이다.
R-CNN 방식은 1-stage 에서 후보 오브젝트 위치들의 sparse set 을 예측한다.
2-stage 에서는 후보 위치들의 오브젝트의 클래스를 분류한다.
그동안 COCO benchmark 에서는 이러한 two-stage 프레임워크가 줄곧 top 의 위치를 지켜왔다.

one-stage 방식은 two-stage 방식의 성능을 따라잡을 수 없을까?
One stage detector 는 영상 전체 위치에서 오브젝트의 위치, 크기, 비율을 dense 하게 샘플링한다.
이 방식으로 YOLO, SSD 등은 최고 성능의 two-stage 방식에 비해 굉장히 빠르지만 성능면에서는 단지 10~40% 부족하다.
이 논문은 여기서 한 발 더 나아간다.

COCO AP 에 대해서 기존 최고 성능의 모델은 Feature Pyramid Network(FPN)이나 Faster R-CNN 의 변형인 Mask R-CNN 과 같은 tw
우리는 one-stage object detector 로서는 최초로 이들 성능을 뛰어 넘었다.

우리는 one-stage detector 의 학습과정에서 클래스 불균형이 가장 큰 문제였음을 확인하였다.
그래서 새로운 로스 함수를 제안하여 이 문제의 장벽을 제거함으로써 이러한 좋은 성능을 낼 수 있었다.

R-CNN 계열의 detectors 에서는 two-stage cascade 나 sampling heuristics 을 사용해서 클래스 불균형 문제를 해결했다.
proposal 단계에서(Selective Search, EdgeBoxes, DeepMask, RPN) 후보들의 대부분인 배경 샘플들을 필터링한다.
두번째 분류 단계에서는 foreground-to-background ratio 가 1:3 이 되도록하거나 online hard example mining(OHEM)등의 sampling h
이러한 샘플링 작업들이 foreground 와 background 클래스의 균형을 맞춰서 학습이 잘되게 해준다.

반면 one-stage detector 는 영상 전체에서 후보 오브젝트가 매우 많이 나온다.
학습과정에서 loss 가 너무나 많은 background examples 에 의해 압도되어 버리기 때문에 R-CNN 에서 사용하던 샘플링 기법을 적용하

이 논문에서는 새로운 로스 함수를 사용하여 이전의 방법들에 비해 훨씬 효율적으로 클래스 불균형 문제를 해결하였다.
우리의 로스 함수는 동적(Dynamic)으로 크로스 엔트로피 로스의 크기를 조절한다.
그림 1 에서 나타나듯이 클래스 분류가 잘 된 예제에 대해서는 scaling factor 가 0 에 가까워진다.
직관적으로 scaling factor 는 자동으로 학습과정에서 분류하기 쉬운 예제(easy example)들이 학습에 기여하는 정도를 낮춘다(down-we
반면 분류하기 어려운 예제(hard example)들에 대해서는 매우 집중한다.

실험의 결과를 통해서 우리가 제안하는 Focal Loss 가 one-stage detector 에서 매우 높은 성능을 낼 수 있으며, 기존의 sampling he
효율적인 대안이 될 수 있다는 것을 보여준다.
focal loss 의 효과를 보여주기 위해서, 우리는 간단한 one-stage object detector 인 RetinaNet 을 디자인했다.
입력 이미지로부터 object locations 을 dense sampling 한다는 의미에서 RetinaNet 이라고 이름을 지었다.
디자인면에서 효율적으로 in-network feature pyramid 및 앵커 박스를 사용하는 것이 특징이다.
RetinaNet 은 효율적이고 정확하다.
ResNet-101-FPN backbone 으로 하는 우리의 최고 성능의 모델에서 COCO test-dev AP 39.1 , 5 fps 의 성능을 낸다.
그림 2 에서 나타난것처럼, 지금까지 published 된 모든 single-model 과 모든 two-stage ectectors 의 성능을 능가한다.

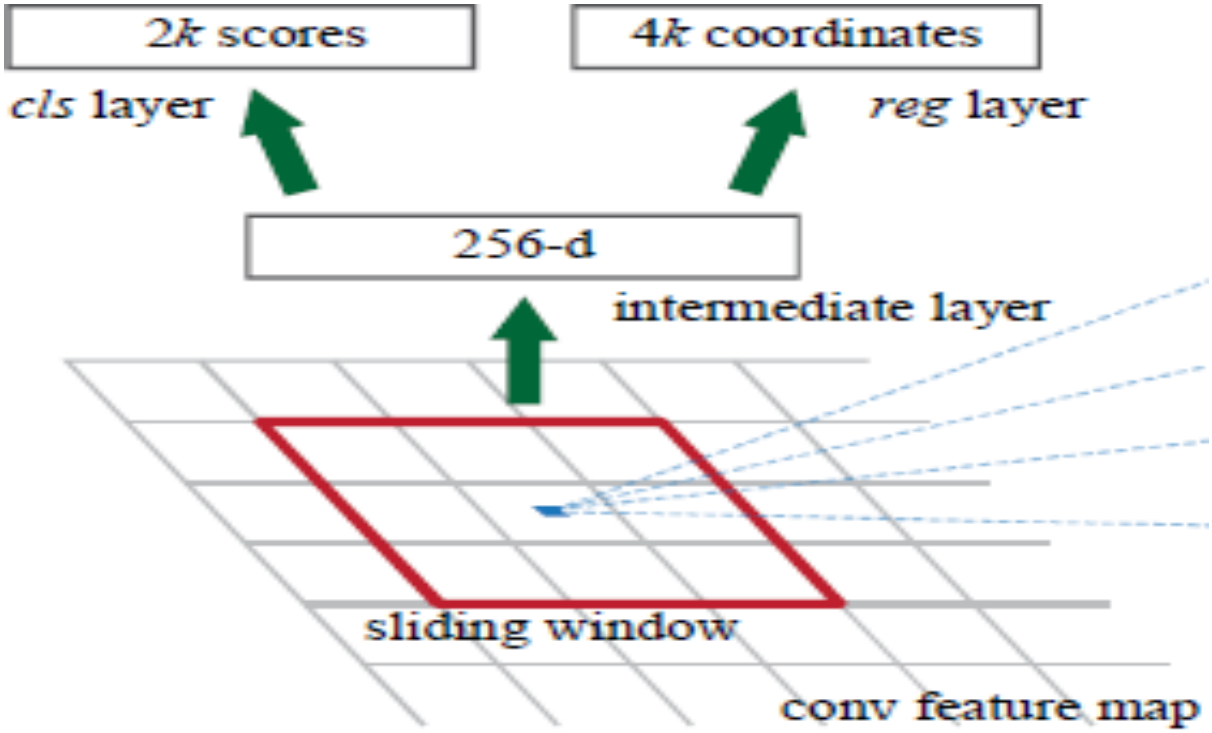
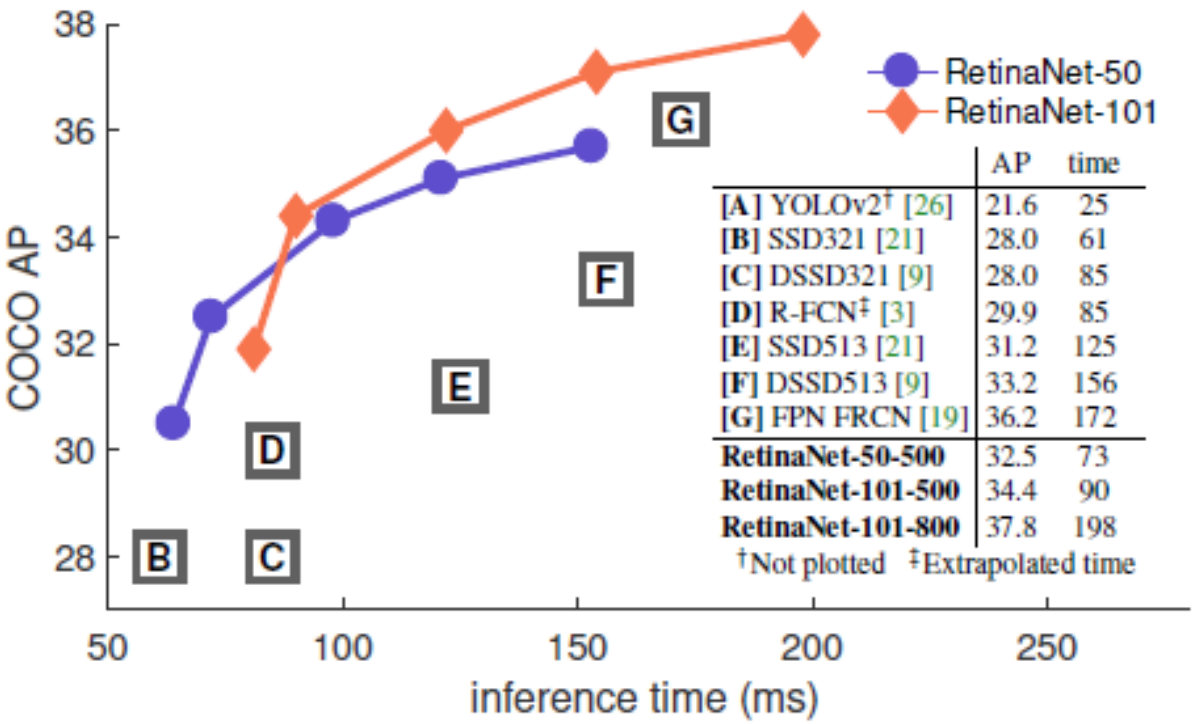


그림 1. 기존 크로스 엔트로피 함수에 $(1-p_t)^\gamma$ 를 추가한 새로운 포컬 로스를 제안한다.

γ (감마)>0 으로 셋팅하면 상대적으로 잘 분류된 예제들(well-classified examples: Pt> 0.5)의 로스를 줄일 수 있다.
그러면 잘못 분류된 예제들에게 더욱 집중할 수 있게 된다.
대부분이 구분하기 쉬운 배경 예제들이 많은 경우 포컬 로스를 사용하면 높은 성능을 갖는 dense object detector 를 학습시킬 수 있다



	AP	time
[A] YOLOv2† [26]	21.6	25
[B] SSD321 [21]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [21]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [19]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

그림 2. COCO test-dev 셋에서 여러 모델의 속도와 정확도 간의 관계.
포컬 로스를 사용하는 간단한 one-stage 의 RetinaNet Detector 는 이전의 one-stage 와 two-stage detectors 보다 성능이 더 좋다.
비교 대상에는 지금까지 최고의 성능을 내는 것으로 알려진 Faster R-CNN([G] FPN FRCN) 도 포함되었다.

그래프에는 다양한 입력 크기를 받는(400-800 픽셀) ResNet-50-FPN(파란색 원)과 ResNet-101-FPN(오렌지 다이아몬드) 을 바탕
나타나있다.
25 미만을 뺀 AP 성능에서 RetinaNet 은 지금까지의 어떠한 모델들보다도 성능보다 좋다.
그래프에는 나타나있지 않지만 가장 높은 성능을 보인 RetinaNet 의 AP 는 39.1 이다.

2. Related Work

Classic Object Detectors: 슬라이딩 윈도우, HOG, DPMs 등은 그동안 효율적으로 사용되어 왔다.
그동안 슬라이딩-윈도우 방식이 detection 패러다임을 이끌었지만 딥러닝의 부활로 인해서, two-stage detectors 가 object detection

Two-stage Detector: 최근의 object detection 에서의 dominant paradigm 은 two-stage approach 이다.
1 단계에서 sparse set 한 후보를 제안하고 2 단계에서는 분류를 한다.
R-CNN 은 2 단계 분류기에서 convolutional network 를 사용하여 성능면에서 큰 발전을 이루었다.
Region Proposal Networks(RPN) 과 2 단계의 classfier 와 single convolution network 으로 통합되어 Faster R-CNN framework 가 만들

One-stage Detectors: OverFeat 은 딥러닝을 사용한 최초의 one-stage object detector 다.
최근에는 SSD, YOLO 등이 one-stage method 로서 좋은 성능을 냈다.
이들 detectors 는 speed 에 초점을 맞추었지만 성능도 나쁘지 않았다.
SSD 는 단지 AP 가 10-20% 가 낮다.
YOLO 는 스피드/정확도 trade-off 에 더욱 집중했다.

RetinaNet 은 기존의 dense detectors 와 매우 유사하다.
RPN 에서 소개된 'anchor' 와 SSD 와 FPN 에서 사용한 features pyramids 를 사용한다.
중요한 것은 RetinaNet 의 높은 성능을 보이는 이유는 네트워크 디자인이 아니라 새로운 loss 함수 때문이라는 점이다.

Class Imbalance: boosted detector,DPMs,최신의 SSD 같은 일반적인 one-stage object detection 에서는 학습 과정에서 클래스 불
이들 detectors 는 한장의 이미지에서 10^4~10^5 개의 후보 위치를 제안하지만 실제로는 몇개의 오브젝트만 있을 뿐이다.
이러한 불균형 현상은 2 가지 문제를 일으킨다
(1) 분류하기 쉬운 easy negative 들이 대부분인데 이들이 학습에 기여하는 것이 거의 없기 때문에 학습이 비효율적이다.
(2) 수많은 easy negative 들이 학습 과정을 압도하므로 비일반적인 모델이 학습되어 버린다.
일반적인 해결방법은 학습과정에서 hard examples 을 샘플링하는 hard negative minining(학습이 잘 안되는 샘플들을 모으는것) 을 하기
방법을 쓰는 것이다.
반면, 우리가 제안하는 focal loss 는 one-stage detector 에서는 피할 수 없는 클래스 블균형문제를 자연스럽게 해결한다.
샘플링 기법을 사용하지 않고도 모든 샘플에 대해서 효율적인 학습이 이루어지도록 해준다.
focal loss 는 easy negative 에게 압도당하지 않는 로스와 그라디언트를 계산해준다.

Robust Estimation: 매우 큰 에러를 갖는(hard example) outliers 의 loss 를 down-weighting 시킴으로써 loss 에 대한 영향력을 줄
sampling 기법보다 더욱 흥미로워 보인다.
outlier 에 대한 접근하는 robust loss function 과는 반대로 focal loss 는 inliers(easy examples) 을 down-weiginting 시키므로서 그들
비록 하나하나의 inliers 들의 loss 는 작지만 inliers 들이 매우 많기 때문에 total loss 에 끼치는 영향력은 크다.
다른 말로 하자면 focal loss 는 robust loss 와 정반대로 동작한다고 할 수 있다.
focal loss 는 일부의 hard examples 에 대해 집중해서 학습한다.

3. Focal Loss

Focal Loss 는 one-stage object detection 에서 foreground 와 background 의 클래스간 불균형이 극도로 심한 상황(예를 들면 1:100
binary classification 을 위한 cross entropy(CE) loss 에서부터 focal loss 의 소개를 시작한다.

$$\mathrm{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise.} \end{cases} \qquad (1)$$

(1) 에서 y 는 1 or -1 이며 ground-truth class 를 가리킨다.
p 는 y=1 인 클래스에 대해서 모델이 예측한 클래스 확률(Probability)로 0 과 1 사이의 값이다.
조금 더 편리하게 식을 다루기 위해서 (2)와 같이 Pt 항을 정의하겠다.

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise,} \end{cases} \qquad (2)$$

이므로 CE(p,y) = CE(Pt) = −log(Pt) 이다.

CE loss 는 그림 1 의 파란색 커브의 형태를 갖는다.
CE loss 의 가장 눈에 띄는 특징은 쉽게 예측된 샘플들(Pt >> 0.5)의 로스가 그다지 작지 않다는 것이다.
그 결과 이들 loss 가 커지게 되면(혹은 많아지게 되면) rare(드문) class 들의 loss 를 압도해 버리는 상황이 생길 수 있기 때문에 문제

3.1 Balanced Cross Entropy

클래스 불균형을 해결하는 가장 일반적인 방법은 0~1 사이의 가중치 a(알파) 를 사용하는 것이다.
클래스 1 에는 a 를, 클래스 -1 에는 1-a 를 적용한다.
실제로 a 는 클래스 빈도의 역수(1:90 클래스 비율의 경우 a=90/1=90)를 사용하거나 하이퍼파라미터로서 크로스 밸리데이션과정을 통
클래스 t 에 대한 알파 t 를 다음과 같이 a-balance CE loss 식으로 정의한다.

$$\mathrm{CE}(p_t) = -\alpha_t \log(p_t). \qquad (3)$$

이 로스함수는 단순히 CE 를 확장한 것으로 실험에서는 이 식을 기본적으로 사용했다.

3.2 Focal Loss Definition

우리의 실험에서 나타나듯이, 클래스 dense detector 의 학습과정에서 불균형 문제가 심각한 경우에는 (easy samples 에 의해) 크로스
예측하기 쉬운 negative 샘플들이 대다수를 이루게 되면서 gradient 에 막대한 영향을 준다.
위 3.1 에서 balance 파라미터인 알파는 positive/negative example 의 중요한 정도를 밸런스가 맞도록 조절하는 역활을 한다.
그러나 알파는 easy/hard example 을 따로 구분하지 않는다.
반면 우리가 제안하는 변형된 loss function 은 easy example 의 중요도를 down-weight 시키며 그 결과 hard negative 의 학습에 집중

cross entropy loss 에 조절 항 $(1 - p_t)^\gamma$ 을 추가했다.

γ 는 조절가능한 focusing parameter 로 0 이상의 값으로 셋팅한다.
다음과 같이 focal loss 를 정의한다.

$$\mathrm{FL}(p_t) = -(1 - p_t)^\gamma \log(p_t). \qquad (4)$$

이 focall loss 에서 γ 를 0~5 사이의 값으로 조절했을 때의 함수 형태를 그림 1 에서 볼 수 있다.
(1) 만약 example 을 잘못 분류했는데, Pt 가 작은 값인 경우 조절항이 1 에 가까워지면서 조잘 항의 값이 거의 0 이 되므로 결국 로스

((1-0.00001)^γ=~1)

Pt -> 1 에 가까워 지면, 조절 항은 거의 0 이 된다. ((1-0.999)^γ=~0)
이것이 잘 분류된 샘플을 down-weight 시키는 경우이다.

(2) focusing parameter γ 는 부드럽게 easy example 을 down-weight 시킨다.

만약 γ = 0 이라면, FL 은 CE 와 같다.

γ 의 값이 커질수록 FL 의 down-weight 시키는 영향력이 커진다.(실험에서는 γ=2 인경우 가장 성능이 좋았다)
직관적으로 보면, modulating factor 은 easy example 들의 loss 에 대한 영향력을 줄이며, loss 를 낮추는 example 의 범위를 결정한

예를 들어 γ = 2 인 경우 Pt = 0.9 로 예측되었던 example 의 원래 CE 로스에 비해서 FL 에서는 CE 의 Pt=~0.968 일때 수준의
작아 진다.
이것은 잘못 분류되었던 examples 의 중요도를 높이는 역활을 한다.

(γ = 2 로 셋팅되었을 때, Pt <= 0.5 로 예측된 example 의 loss 는 단지 4x 작아질 뿐이다.)

실제로 우리는 알파-balance variant 의 focal loss 를 다음과 같이 정의했다.

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t). \qquad (5)$$

위 로스 함수를 모든 실험에 적용했는데, 알파-balacned 항을 쓰지 않을 때에 비해 약간의 성능 향상이 있었다.
마지막으로, loss layer 에서 p 를 계산하는 과정에서 sigmoid operation 을 사용할 때 굉장히 큰 성능 향상을 보였다.
그러나 전체적으로는 loss layer 를 무엇을 사용하던지간에 결과에 큰 차이는 없었다.

3.3 Class Imbalance and Model Initialization

기본적으로 Binary classificatino Model 은 클래스 y 가 -1, 1 일 확률이 같도록 초기화된다.
이렇게 초기화했을 때 클래스 불균형 현상이 나타나면 더 많은 class 가 total loss 에서 더 많은 비중을 차지하게 되어 학습이 초기부
이 문제를 해결하기 위해 'prior' 라는 개념의 p 항을 사용하였다.
p 항은 rare class(foreground) 의 비율로 최초 학습시에 사용하는 값이다.
rare class 의 비율이 0.01 처럼 낮은 경우 이 비율을 p 라고 추정하며 파이에 의해 정의된다.
p 항은 모델 초기화에서 사용되는 것일 뿐이고 loss function 이 아니다.
클래스 불균형문제가 심각한 경우에 p 항을 사용하면 cross entropy 나 focal loss 에 관계 없이 학습 안정화에 도움을 준다.

3.4 Class Imbalance and Two-stage Detectors

Two-stage Detectors 는 보통 cross entropy loss 를 사용하고 알파-밸런싱은 사용하지 않는다.
대신 클래스 불균형 문제를 해결하기 위해 2 가지 메카니즘을 사용한다.
(1) two-stage cascade (2) baised minibatch sampling
1 번째 cascade stage 는 object proposal mechanism 으로 거의 무한대의 가능한 오브젝트의 위치를 1~2 천개 정도로 줄이는 것이다
선택할 proposals 은 물론 랜덤한 것이 아니라 true object location 에 가까운 것이어야만 한다.
그렇게 함으로써 대부분을 차지하는 easy negatives 를 제외할 수 있다.
2 단계 학습에서는, biased sampling 으로 minibatch 를 구성한다.
positive/negative 비율을 알파-밸런싱 factor 와 비슷하게 샘플링한다.
우리가 제안하는 focal loss 는 one-stage detection syste 에서 loss function 을 통해서 이러한 메카니즘들을 직접 디자인한다.

4. RetinaNet Detector

RetinaNet 은 하나로 통합 네트워크로 backbone network 와 2 개의 전용 subnetwork 로 구성된다.
backbone 은 입력 이미지 전체 영역에 대해서 convolutional feature map 을 계산하는 역활이며, 누구나 사용할 수 있는 공개된 conv
첫번째 subnet 은 backbone 의 output 으로 부터 convolutional 을 통해서 object classification 을 수행한다.
두번째 subnet 은 backbone 의 output 으로 부터 convolutional 을 통해서 경계박스의 좌표(앵커와 정답간의 offset)를 구한다.
2 개의 subnetwork 의 디자인은 간단하다.
이것으로 그림 3 과 같이 one-state, dense detection 을 구성했다.
설계에는 많은 선택사항이 있지만, 실험 결과처럼 대부분의 디자인 파라미터들은 성능에 대해 크게 민감하지 않다.

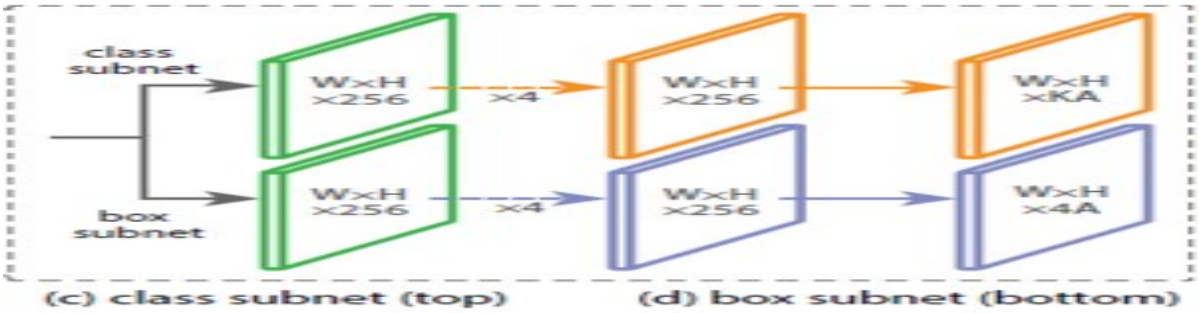


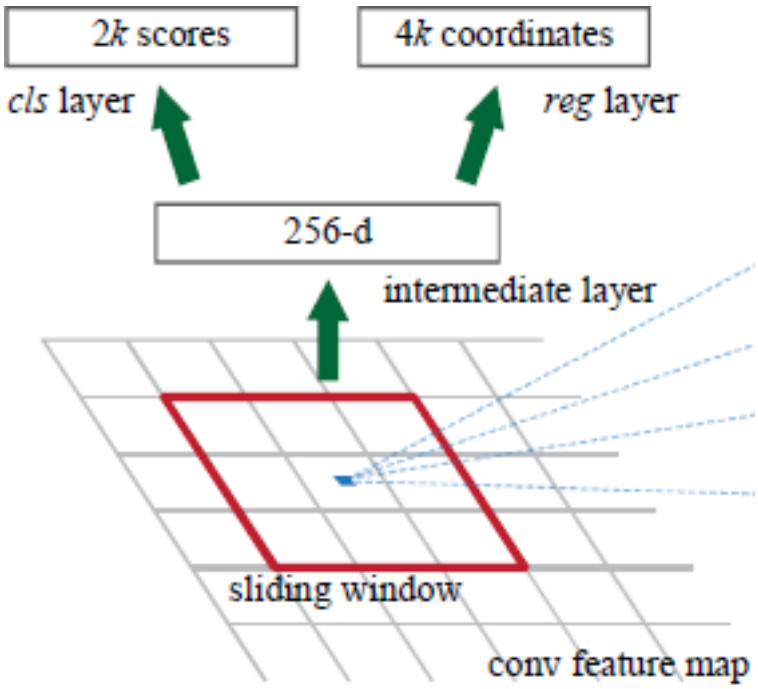
그림 3. one-stage RetinaNet network 구조로 Feature Pyramid Network(FPN)을 사용한다.
backbone 은 ResNet architecture 를 사용해서 (a)의 rich,multi-scale convolutional feature pyramid 를 생성한다.
(b) backbone 에 2 개의 subnetwork 를 붙여서 RetinaNet 이 된다. 하나는 앵커 박스의 클래스 분류, 다른 하나는 앵커 박스와 groun
(d) 네트워크 디자인은 일부러 간단하게 했다.
우리의 목표는 새로운 네트워크 구조를 제안하는 것이 아니다.
focal loss function 를 통해서 one-stage detector 와 그동안 최고 성능을 보인 Faster R-CNN with FPN 등의 two-stage detector
동작시키는 것이다.

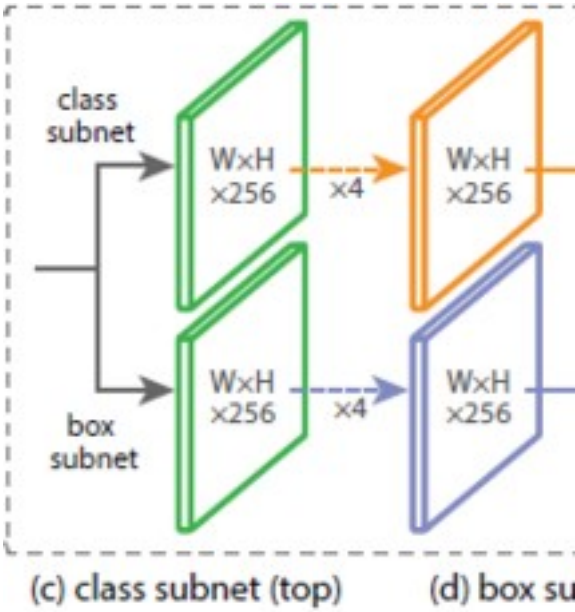
Feature Pyramid Network Backbone: Feature Pyramid Network(FPN)을 RetinaNet 의 backbone 으로 채택했다.
간단하게 말하면, FPN 은 일반 convolutional network 를 top-down 방향으로 augment 한다.
그림 3(a)-(b) 와 같이 lateral connection(수평 연결) 을 통해서 하나의 resolution 입력 이미지로부터 rich 한 multi-scale feature pyra
각각의 피라미드 레벨은 서로 다른 크기의 object 를 검출하는데 사용한다.
FPN 은 fully convolutional network(FCN) 에 비해서 multi-scale 예측 성능이 더 좋다는 것이 이미 RPN, DeepMask-style proposal, tw
R-CNN 등을 통해서 알려져 있다.
ResNet 구조 위에 FPN 을 구성하였다.
피라미드 레벨을 P3 에서 P7 이 되도록 했다.
숫자의 의미는 입력 이미지 크기에 대해 2 의 승수만큼 작다는 것이다.(P1 = input size, P2 = input size/2, P3 = input size/2/2 ...)
모든 피라미드 레벨의 채널 C 는 256 으로 동일하다.
설계에서 다양한 디자인을 선택할 수 있었지만 성능에 큰 영향을 주지는 않는다.
다만 FPN 을 backbone 으로 쓰는 것은 중요하다.
FPN 없이 ResNet 의 최종 레이어만을 사용해 예측하는 경우에는 AP 가 낮았다.

Anchor: RPN(Region Proposal Network) 에서 사용하는 translation-invariant anchor box(이동 불변 앵커박스) 를 사용했다.
앵커박스의 크기는 피라미드 레벨 P3~P7 에서 32^2 ... 512^2 의 크기를 갖는다.
각 피라미드 레벨에서 3 개의 종횡비{1:2, 1:1, 2:1} 를 갖는 앵커박스를 사용했다.
모든 스케일에 대응(denser scale coverage) 하기 위해서 각 피라미드 레벨에서 원래의 3 개의 앵커박스에 대해서 각각 3 개를 더 추가
이렇게 하는것이 AP 를 올리는데 도움이 되었다.
각 레벨에서 총 A = 9 개의 앵커박스를 갖고 있으며 입력이미지의 32 - 813 픽셀까지 담당할 수 있다.
각 앵커박스에는 K 개의 one-hot vector classification target 와 4-vector 의 box regression target 이 할당된다.
K 는 number of ojbect class 이다.
GT 와의 intersection-over-union(IoU)가 0.5 이상인 앵커박스만 ground-trugh object box 로 할당된다.
IoU 가 0.4 이하의 앵커박스는 background 로 할당된다.
앵커박스는 최대(오직) 1 개의 object box 에만 할당된다.
K label vector 는 클래스에 해당하는 entry 만 1, 나머지는 0 으로 채운다.(one-hot)
만약 앵커박사가 아무것에도 할당되지 않는다면, 즉 GT 와의 IoU 가 0.4~0.5 인 경우라면, 해당 앵커박스는 학습과정에서 무시된다.
Box regression targets 은 각 앵커박스와 할당된 object box 와의 offset(거리)를 계산한다.
할당되지 않은 앵커박스들은 계산에서 빠진다.

Classification Subnet: classification subnet 은 각 위치에서 A 앵커에의 K 개의 object class 확률을 예측한다
이 subnet 은 각 FPN 레벨에 붙은 작은 FCN 이다.
subnet 의 네트워크 구조는 모든 피라미드 레벨에서 전부 동일하다.
디자인도 매우 단순하다.
그림 3 (c) 와 같이 주어진 피라미드 레벨에서 C 채널의 feature map 을 받는다.
3x3 컨볼루션을 256 개의 필터로 컨볼루션하고 ReLU 활성함수를 적용한다. 이것을 4 회반복 (x 4)
마지막으로 KA(K*A) 개의 필터로 컨불루션한 후 매 공간 위치에서 binary prediction 을 위해서 Sigmoid 활성함수를 적용한다.
(Sigmoid 와 FL 조합의 성능이 좋았다)

RPN 과는 대조적으로 우리의 object classification subnet 은 레이어가 깊고 3x3 컨볼루션만 사용한다.
모든 피라미드 레벨에서 파라미터를 공유한다.





참고) RPN 과 RetinaNet RPN 의 클래스,경계박스 예측 레이어 : RPN 에서는 cls layer 와 reg layer 가 intermediate layer 의 출력 feature

이렇게 전체 레벨에서 디자인 하는 것이 하이퍼 파라미터를 조절하는 것보다 더 중요하다는 것을 확인했다.

Box Regression Subnet: 각 피라미드 레벨에서 object classification subnet 과는 병렬적으로 또다른 작은 FCN 을 붙였다.
이것은 그림 3(d) 와 같이 각 앵커박스에서 근처에서 ground-truth object 가 있는 경우 둘 간의 offset 을 예측하는 것이 목표이다.
각각의 위치에서의 A 개의 앵커박스들은 자신과 GT 간의 상대적인 좌표 offset 4 개를 예측한다.
(R-CNN 에서 사용하는 일반적인 박스 파라미터를 사용했다. cx, cy, w, h)
최근의 work 와는 다르게, 우리는 클래스에 관계없이 좌표를 예측하도록 했는데, 성능의 차이가 없음을 확인했다.
(일부 최근의 work 에서는 클래스마다 좌표 예측을 하는 파라미터들이 다르다).
object classification subnet 과 box regression subnet 은 모두 공통의 구조를 가지지만, box regression subnet 은 파라미터는 공유하지

4.1. Inference and Training

Inference: 속도를 위해서 detector threshold 0.05 로 걸러낸 이후 top-scoring proection 1k 개의 box prediction 만 사용했다.
모든 피라미드 레벨에서의 예측들은 최종 detection 을 위해서 threshold 0.5 의 non-maximum suppression 을 통해서 중복 예측들이

Focal Loss: classification subnet 의 output 에 focal loss 를 사용했다.

γ 가 0.5~5 일때 모두 괜찮지만 γ = 2 일때 가장 성능이 좋다.
RetinaNet 을 학습할때, focal loss 에서 계산되는 anchor 의 갯수가 all~100k 개의 합이고 각 로스들이 앵커와 GT 박스의 갯수의 비율
전체 앵커박스가 아니라 할당이 된 앵커박스의 수로 정규화했다.
그 이유는 대부분의 앵커박스는 easy negative 여서 focal loss 를 사용하게 되면 매우 작은 loss 를 받기 때문이다.
마지막으로, rare class 에 할당되는 weight 인 알파도 역시 그리 민감하지 않다.
Table 1a 와 1b 에서 감마와 알파 값에 따른 성능이 나타나있다.
일반적으로 감마값을 키울수록 알파값은 감소시켜야 한다.
(감마=2, 알파=0.25 가 가장 좋았다)

Initialization: -ResNet-50-FPN 과 ResNet-101-FPN 두개를 backbone 으로 실험했다.
둘 다 ImageNet1k 로 pre-train 시켰다.
'Feature pyramid networks for object detection'처럼 FPN 을 위해 새로운 레이어를 추가했다.
최종 레이어를 제외하고는 bias b = 0, Gaussian weight fill with 델타 = 0.01 로 초기화했다.
classification subnet 의 마지막 레이어의 bias 는 b = -log((1-파이)/파이) 로 초기화 했다.
파이는 최초 학습에서 모든 앵커의 라벨의 foreground 에 대한 confidence 값으로 모든 실험에서 0.01 을 사용했다.
이렇게 초기화하면 너무 많은 background anchor 가 생성되는 것을 방지해 주기 때문에 첫번째 iteration 에서 학습이 안정된다.

Optimization: RetinaNet 은 stochastic gradient descent(SGD)로 학습된다.
8 개의 GPU 에서 synchronized SGD 를 사용했고 GPU 당 배치 2 개이므로 전체 총 배치는 16 개다.
(개별 GPU 에서 개별 배치에 대해 Gradient 를 계산 - 동기화 - 모든 Gradient 를 모아 평균내서 parameter 를 update - 개별 GPU 에
모든 모델은 90k iteration 이고, 학습율은 0.01 이고 60k 와 80k 에서 10 으로 나눈다.
augmentation 은 horizontal image flipping 만을 사용했다.
Weight decay 는 0.0001, momentum 0.9 다.
training loss 는 focal loss 와 box regression 의 standard smooth L1 loss 의 합이다
학습 시간은 모델에 따라서 10~35 시간으로 Table 1e 에 나타나 있다.

5. Experiments

경계 박스의 성능은 COCO benchmark 로 측정했다.

α	AP	AP_{50}	AP_{75}
.10	0.0	0.0	0.0
.25	10.8	16.0	11.7
.50	30.2	46.7	32.8
.75	31.1	49.4	33.0
.90	30.8	49.7	32.3
.99	28.7	47.4	29.9
.999	25.1	41.7	26.1

(a) **Varying α for CE loss ($\gamma = 0$)**

γ	α	AP	AP_{50}	AP_{75}
0	.75	31.1	49.4	33.0
0.1	.75	31.4	49.9	33.1
0.2	.75	31.9	50.7	33.4
0.5	.50	32.9	51.7	35.2
1.0	.25	33.7	52.0	36.2
2.0	.25	**34.0**	**52.5**	**36.5**
5.0	.25	32.2	49.6	34.8

(b) **Varying γ for FL** (w. optimal α)

#sc	#ar	AP	AP_{50}	AP_{75}
1	1	30.3	49.0	31.8
2	1	31.9	50.0	34.0
3	1	31.8	49.4	33.7
1	3	32.4	52.3	33.9
2	3	**34.2**	**53.1**	**36.5**
3	3	34.0	52.5	**36.5**
4	3	33.8	52.1	36.2

(c) **Varying anchor scales and aspects**

method	batch size	nms thr	AP	AP_{50}	AP_{75}
OHEM	128	.7	31.1	47.2	33.2
OHEM	256	.7	31.8	48.8	33.9
OHEM	512	.7	30.6	47.0	32.6
OHEM	128	.5	32.8	50.3	35.1
OHEM	256	.5	31.0	47.4	33.0
OHEM	512	.5	27.6	42.0	29.2
OHEM 1:3	128	.5	31.1	47.2	33.2
OHEM 1:3	256	.5	28.3	42.4	30.3
OHEM 1:3	512	.5	24.0	35.5	25.8
FL	n/a	n/a	**36.0**	**54.9**	**38.7**

(d) **FL *vs.* OHEM baselines** (with ResNet-101-FPN)

depth	scale	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	time
50	400	30.5	47.8	32.7	11.2	33.8	46.1	64
50	500	32.5	50.9	34.8	13.9	35.8	46.7	72
50	600	34.3	53.2	36.9	16.2	37.4	47.4	98
50	700	35.1	54.2	37.7	18.0	39.3	46.4	121
50	800	35.7	55.0	38.5	18.9	38.9	46.3	153
101	400	31.9	49.5	34.1	11.6	35.8	48.5	81
101	500	34.4	53.1	36.8	14.7	38.5	49.1	90
101	600	36.0	55.2	38.7	17.4	39.6	49.7	122
101	700	37.1	56.6	39.8	19.1	40.6	49.4	154
101	800	37.8	57.5	40.8	20.2	41.1	49.2	198

(e) **Accuracy/speed trade-off** RetinaNet (on `test-dev`)

5.1 Training Dense Detection

Network Initialization: 처음 시도한 방법은 RetinaNet 을 standard cross entropy(CE) loss 로 학습시키는 것이었다.
그런데 네트워크가 학습 중에 발산해버리는 바람에 금방 실패했다.
그래서 마지막 레이어의 검출 사전 확률을 파이 =0.01 로 초기화 해주었더니 효율적으로 학습이 가능했다.
이렇게 했을 때의 ResNet-50 을 backbone 으로하는 RetinaNet 의 정확도는 COCO 에서 30.2 AP 었다.
파이의 초기값은 크게 중요하지 않았으므로 앞으로의 모든 실험에서 0.01 을 사용했다.

Balanced Cross Entropy: 다음으로는 알파-밸런스 CE loss 를 사용해 보았다.
Table 1a 에 알파를 조절했을때의 성능이 나타나는데 알파를 0.75 로 셋팅했을 때 AP 가 0.9 상승했다.

Focal Loss: focal loss 를 사용했을 때의 성능은 Table 1b 에 나타나 있다.
focal loss 에는 새로운 하이퍼파라미터인 감마를 통해 focus 정도를 조절할 수 있다.
감마가 0 일때 focal loss 는 CE loss 와 똑같다.
그림 1 처럼 감마를 증가시키면, "easy" example 의 loss 가 작아진다.
감마를 키울수록 성능이 올라갔다.
FL 에서 감마가 2 일때 알파-밸런스 CE loss 에 비해서 2.9AP 의 성능 향상이 있었다.
여러 알파와 감마의 조합의 실험 결과는 Table 1b 에서 볼 수 있다.
작은 알파는 높은 감마를 필요로 했다.(easy negative 가 down weight 된다. 대신 positive 에 좀 더 집중한다)
전체적으로 감마를 키우는 것이 이득이었고, 가장 좋은 알파는 [.25, .75] 사이의 값이었다.
알파는 [.01, .999] 사이의 값으로 테스트했다.

모든 실험에서 감마 2, 알파 .25 에서 가장 성능이 좋았다.
여기서 알파를 .5 로 하면 .4 AP 낮아진다

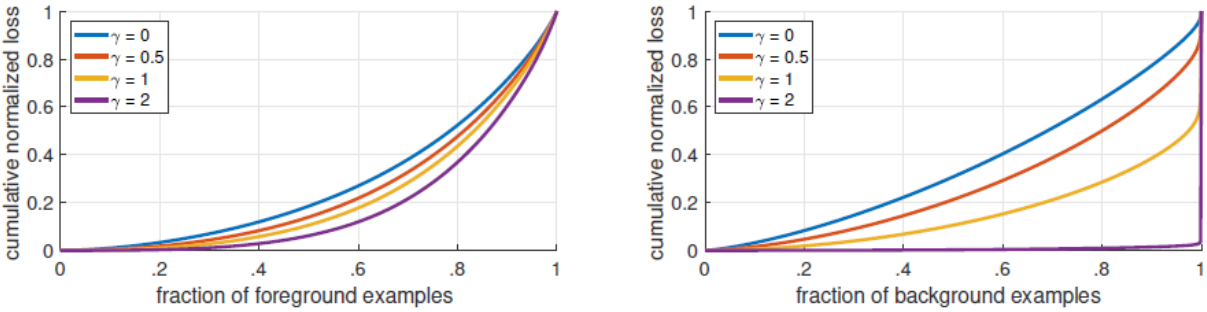


그림 4. 수렴된 모델로부터 각각 다른 감마를 사용해서 positive 와 negative 샘플들의 정규화된 로스를 구해서 그린 Cumulative distrib
positive 로스 분산 그래프에서 감마의 효과는 작지만 negative 에서는 감마가 커질수록 hard example 에 대부분의 집중력을 사용한다

Analysis of the Focal Loss: focal loss 를 이해하기 위해서 수렴한 모델에서 loss 의 분포를 분석했다.
기본 ResNet 101 600 픽셀 모델을 감마 2 로 학습시키고(AP 36.0), 아무 이미지나 넣어서 예측 확률을 ~10^7 개의 negative 와 10^
다음으로 positive 와 negative 를 구분하여, 이들 샘플의 FL 를 구했고, 합이 1 이 되도록 정규화했다.
positive 와 negative 의 정규화된 로스를 낮은값부터 높은값 순으로 정렬해서 cumulative distribution function(CDF) 커브를 각각 그렸다
이때 positive 와 negative 는 학습에서는 둘다 감마 2 가 사용되었지만 FL 을 구할 때는 서로 다른 감마를 사용했다.
cumulative distribution function 은 그림 4 에 나타나있다.
positive sample 에서는 CDF 가 감마의 변화에 따라 크게 변하지 않는다.
예를 들여 대략 20% 정도의 hardest positive sample 이 전체 loss 의 절반을 차지한다.(오른쪽 .8 ~ 1 부분)
감마가 커질수록 top 20% 의 example 에 더욱 집중하지만 그 효과는 작은 편이다.

반면 negative sample 에서의 감마의 효과는 매우 크다.
감마가 0 일때 positive 와 negative 의 CDF 는 거의 똑같다.
그러나 감마를 키우게 되면 부분적으로 더 큰 가중치가 hard negative example 에 집중된다.
감마가 2 일때(실험 기본 셋팅), 대부분의 loss 는 아주 일부분의 hard negative example 에서 나온다(오른쪽 끝부분)
여기서 볼 수 있듯이, FL 은 효과적으로 easy negative 의 효과를 줄이고, hard negative example 에 모든 집중력을 쏟는다.

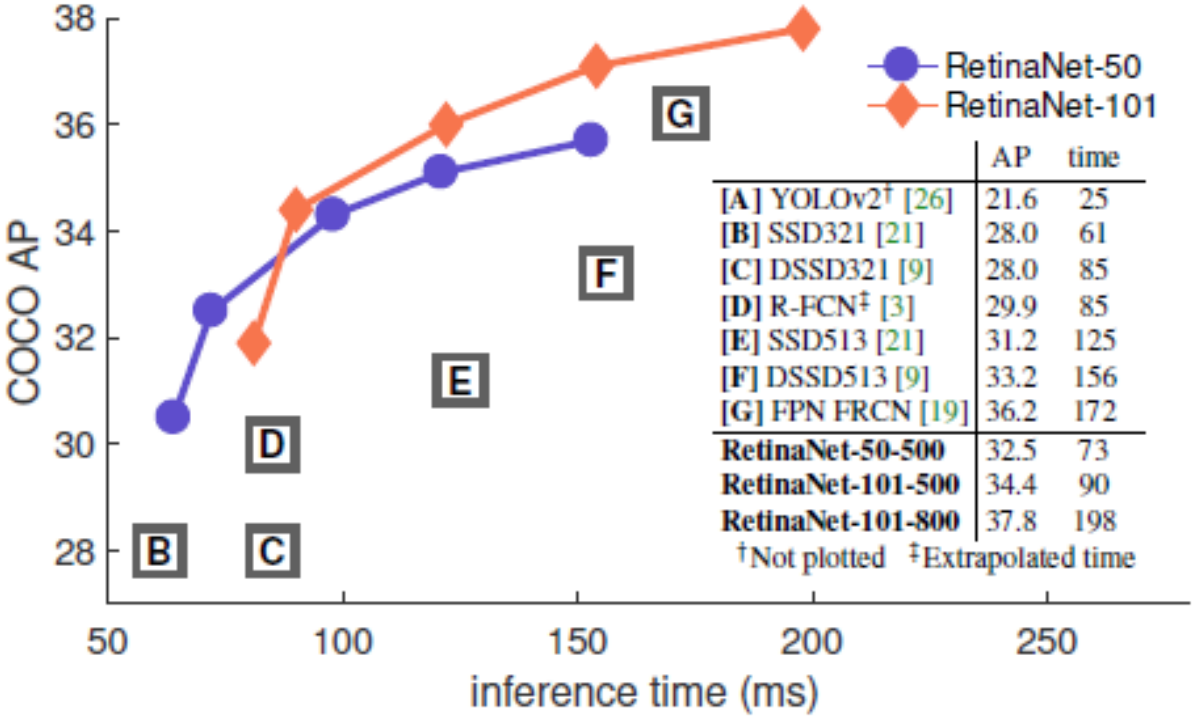
Online Hard Example Mining (OHEM): 'Training regionbased object detectors with online hard example mining' 에서는 two-s
minibach 를 구성해서 성능을 높이는 방법을 제안했다.
OHEM 은 각 example 의 score 에 non-maximum suppression(nms)를 적용해서 loss 가 큰 example 로 minibatch 를 구성한다.
nms 의 threshold 와 batch size 는 선택가능한 파라미터다.
focal loss 와 비슷하게 OHEM 도 잘못 분류된 example 에 집중한다.
하지만 FL 과는 다르게 OHEM 은 easy example 을 일시적으로 완전히 버린다.
우리는 OHEM 을 SSD 에서 구현해서 테스트해보았다.
모든 example 에 nms 를 적용한 후에, positive 와 negative example 이 1:3 의 비율이 되도록 미니배치를 구성했다.
클래스 불균형 문제가 심각한 one-stage detection 에서도 여러 종류의 셋팅을 해서 OHEM 을 테스트해보았다.
original OHEM 을 사용한 성능을 Table 1d 에서 'OHEM 1:3' 로 나타냈다.
같은 셋팅으로 FL 을 이용해 학습한 ResNet-101 backbone 의 RetinaNet 의 성능은 36.0 AP 였다.
반면 OHEM 에서의 최고의 셋팅을 했을 때의 성능은 32.8 AP 였다.
3.2 AP 의 성능의 차이는 OHEM 보다 FL 이 dense detector 에서 더 효율적이라는 것을 말해준다.
다른 종류의 셋팅에서는 OHEM 이 좋은 결과를 얻지 못했다.

Hinge Loss: 마지막으로, 초기 실험에서, Pt 에 대해 hinge loss 를 테스트해보았다.
hinge loss 는 일정값 이상의 Pt 에 대해서는 loss 를 0 으로 하는 방법이다.
그러나 이방법은 불안정했으며 의미 있는 결과를 얻지 못했다.
성능은 부록에 있다.

5.2. Model Architecture Design

Anchor Density: one-stage detection system 에서 가장 중요한 디자인 요소는 모든 가능한 이미지 박스 공간을 얼마나 빈틈없이 커
Two-state detector 는 모든 위치에서 box 를 분류할 수 있다.
Speed versus Accuracy: backbone network 는 정확도가 더 높지만 반면 느리다.
입력 이미지의 크기에 대해서도 마찬가지이다.

그 효과를 테이블 1e 에 나타냈다.
그림 2 에서 RetinaNet 과 최신 method 들의 speed/accuracy trade-off 커브를 그렸다.



	AP	time
[A] YOLOv2† [26]	21.6	25
[B] SSD321 [21]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [21]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [19]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

그림 2 다시 가져옴) COCO test-dev 셋에서 여러 모델의 속도와 정확도 간의 관계.

RetinaNet 이 다른 method 들의 커브보다 위에 있는 것을 볼 수 있다.
표에는 나와있지 않지만 ResNet-101-FPN-600 의 성능이 최근에 publish 된 ResNet-101-FPN Faster R-CNN 과 성능은 비
빠르다.(Nvidia M40 GPU 에서 측정)
더 큰 이미지를 사용하면 더 좋은 성능을 얻을 수 있는데, 이미지가 커져도 속도는 충분히 빠르다.
속도를 중요하게 여길 경우 RetinaNet-50-500 을 사용한다.
high frame rate 를 원한다면 네트워크 디자인을 잘 하거나, 다른 공개된 네트워크를 쓰는 것이 좋겠지만 그런 것은 이 논문의 주제가

5.3. Comparison to State of the Art

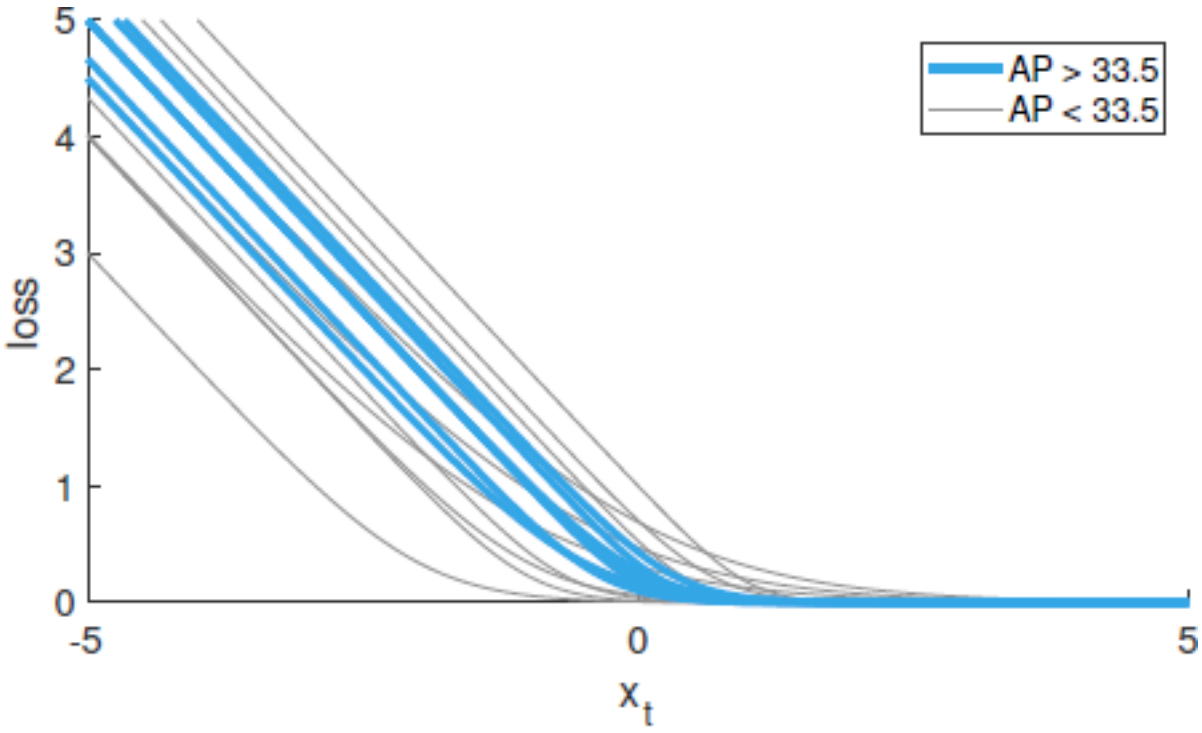
	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [15]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [19]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [16]	Inception-ResNet-v2 [33]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [31]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [26]	DarkNet-19 [26]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [21, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	**39.1**	**59.1**	**42.3**	**21.8**	**42.7**	50.2

표 2. **Object detection** single-model 성능(경계 박스 AP) vs state-of-the-art on COCO test-dev.
표 1e 의 모델을 스케일을 1.5x 까지 조금씩 다르게 해서 학습시킨 RetinaNet-101-800 model 이다.
우리 모델의 최고 성능은 기존의 one-stage 와 two-stage model 을 모두 넘는다.
speed versus accuracy 의 자세한 관계표는 Table 1e 와 그림 2 를 참조
최근에 최고 성능을 낸다고 알려진 모델은 Inception-ResNet-v2-TDM 을 backbone 으로한 two-stage method 인 Faster R-CNN w T
이것과 비교했을 때 RetinaNet 은 2.3 point 더 높은 성능을 보인다.

6. Conclusion

우리는 one-stage object detector 가 Faster R-CNN 류의 two-stage method 만큼의 성능이 나오지 못하는 가장 큰 이유가 클래스 불
이문제를 해결하기 위해 크로스 엔트로피에서 대부분의 easy negative 의 loss 를 상대적으로 줄이면서 hard example 에 집중하도록 ㅈ

이 방법은 간단하고 매우 효율적이다.
이것의



Appendix B : Derivatives

$$\frac{dCE}{dx} = y(p_t - 1) \tag{9}$$

$$\frac{dFL}{dx} = y(1 - p_t)^{\gamma}(\gamma p_t \log(p_t) + p_t - 1) \tag{10}$$

$$\frac{dFL^*}{dx} = y(p_t^* - 1) \tag{11}$$

코드 https://github.com/zhangxujinsh/keras-retinanet