

#hutom_ai

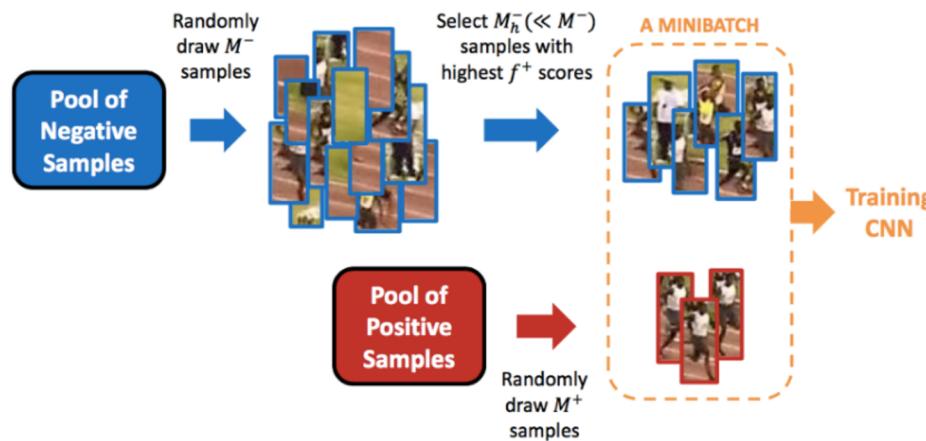
Training Region-based Object Detectors with Online Hard Example Mining

2021-09-01 | JiHyun Lee

1. Hard Example Mining
 - Hard negative mining
 - heuristics method (Fast R-CNN)
1. Training Region-based Object Detectors with Online Hard Example Mining ([link](#))
 - Fast R-CNN
 - OHEM
 - Experiment
 - Conclusion

Hard negative mining

- Object detection
 - 일반적으로 object detection 시, 배경 영역에 해당하는 region proposals 수가 더 많아 클래스 불균형 (class imbalance) 발생.
 - imbalance ratio 70:1
- Hard Negative Mining
 - 클래스 불균형 문제 해결 (i.e. **positive example** (객체) 과 **negative example** (배경) 을 균형적으로 학습하기 위한 방법)
 - 모델이 잘못 예측한, 어려운 (hard) sample 을 추출하는 방법
 - 모델이 예측하기 어려운 sample: **False Positive sample**
 - 객체라고 예측했지만, 실제로는 배경인 것
 - object detection 모델은 positive에 해당하는 객체의 영역만을 detect 하기 때문에 배경을 잘못 예측한 False Negative sample 은 object detection task 에서 고려하지 않기 때문
 - hard negative mining 으로 얻은 데이터를 원래의 데이터에 추가해서 재학습하면 **false positive** 오류에 강해짐



negative sample 을 confidence score 를 기준으로 내림차순 정렬

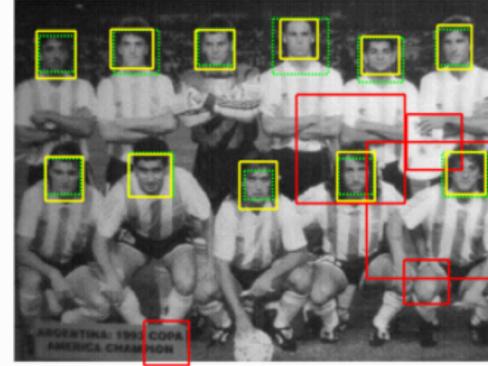
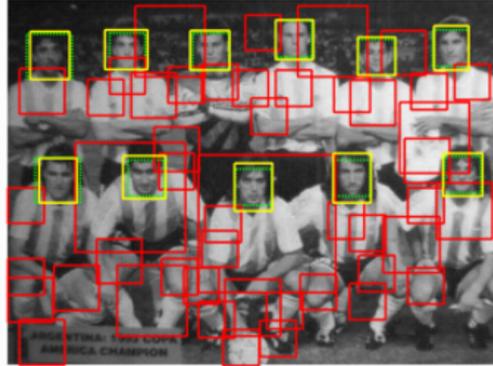
- **confidence score**
 - 특정 바운딩 박스 안에 있는 객체가 어떤 물체의 클래스일 확률과 IoU 를 곱한 값
 - negative sample 중 confidence score 가 높을수록 hard sample

이후 confidence score가 높은 순으로 negative sampling + random하게 선정한 positive sample 들과 구성하여 하나의 mini-batch 형성 후 CNN 학습

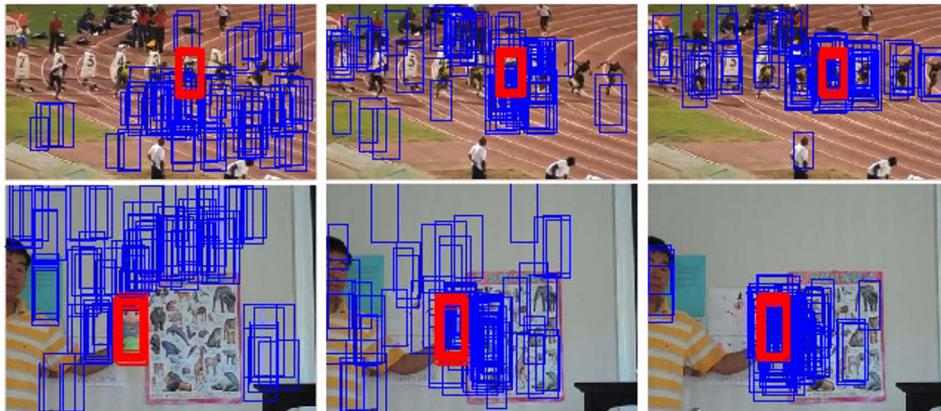
⇒ 전체 negative sample 을 모두 사용할 필요가 없기 때문에 memory 의 양이 줄어들게 됨

Hard negative mining

image: "Argentina.jpg" (green=true pos, red=false pos, yellow=ground truth), 11/11 found image: "Argentina.jpg" (green=true pos, red=false pos, yellow=ground truth), 11/11 found



* hard negative 데이터를
기존의 데이터에 추가해 다시 학습



(a) 1st minibatch

(b) 5th minibatch

(c) 30th minibatch

* 빨간색 box : positive sample
* 파란색 box : negative sample

⇒ 학습이 거듭될수록 점차 mini-batch에 False positive sample 들이 더해지면서 객체라고 착각할만큼 어려운 negative sample 들이 추가됨.

Hard negative mining

Hard negative mining 한계점

- (1) fixed model 이 새로운 샘플을 찾아서 active training set 에 넣음
- (2) 모델이 이 active training set 을 기반으로 학습

SGD를 사용하여 학습되는 ConvNet에서 모델이 false positive를 판단하고 이를 학습 데이터셋에 추가하고, mini-batch를 구성하는 과정이 끝날 때까지 모델을 update 할 수 없어 학습 속도 저하.

⇒ 즉, SGD를 이용해 학습하는 Fast R-CNN의 경우 SGD가 수천개의 iteration 이 필요한데, 이렇게 freeze 하고 학습하는 과정에서 freeze 에서는 학습을 할 수 없음

Hard negative mining 한계점 ⇒ **Fast R-CNN** 에서는 **hard negative mining** 대신에 **heuristics method** 사용

Heuristics method

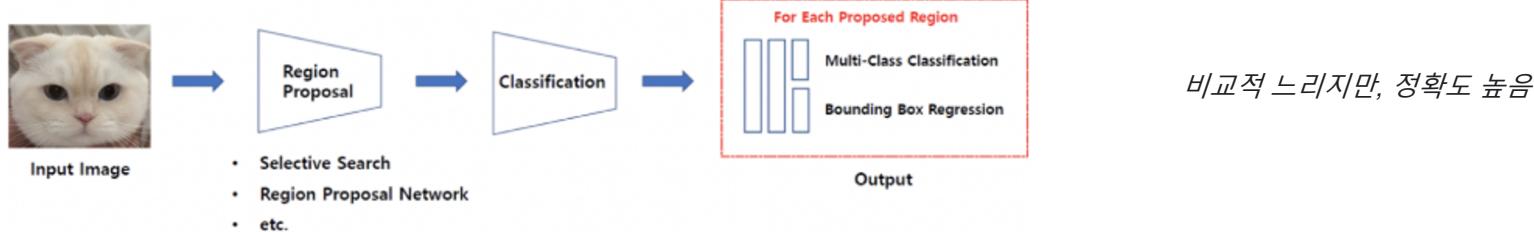
- 하이퍼 파라미터
 - IoU threshold, positive/negative sampling ratio 등..
- 2장의 이미지에서 각각 64개의 region proposal 추출 ⇒ 총 128 mini-batch 구성
- 128개의 region proposal 는 25% 는 positive sample, 75% 는 negative sample 로 추출
- positive sample : IoU 가 0.5 이상, negative sample: IoU가 0.1~0.5 사이의 proposal
- 한계점
 - 하이퍼 파라미터가 많아지면, 실험자의 개입과 시행착오 많아짐..
 - 또한 이러한 방식으로 mini-batch 를 생성하는 방법은 효과적이지 않음

Object detection 방식 : 2-Stage 방식 & 1-Stage 방식

- **2-Stage 방식**

- 물체의 ① 위치를 찾는 문제 (localization) 와 ② 분류 (classification) 문제를 순차적으로 해결
- R-CNN 계열 (R-CNN, **Fast R-CNN**, Faster R-CNN, Mask R-CNN ...)

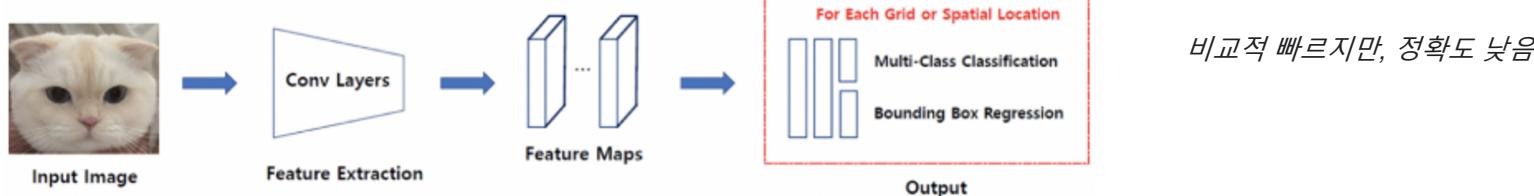
2-Stage Detector - Regional Proposal와 Classification이 순차적으로 이루어짐.



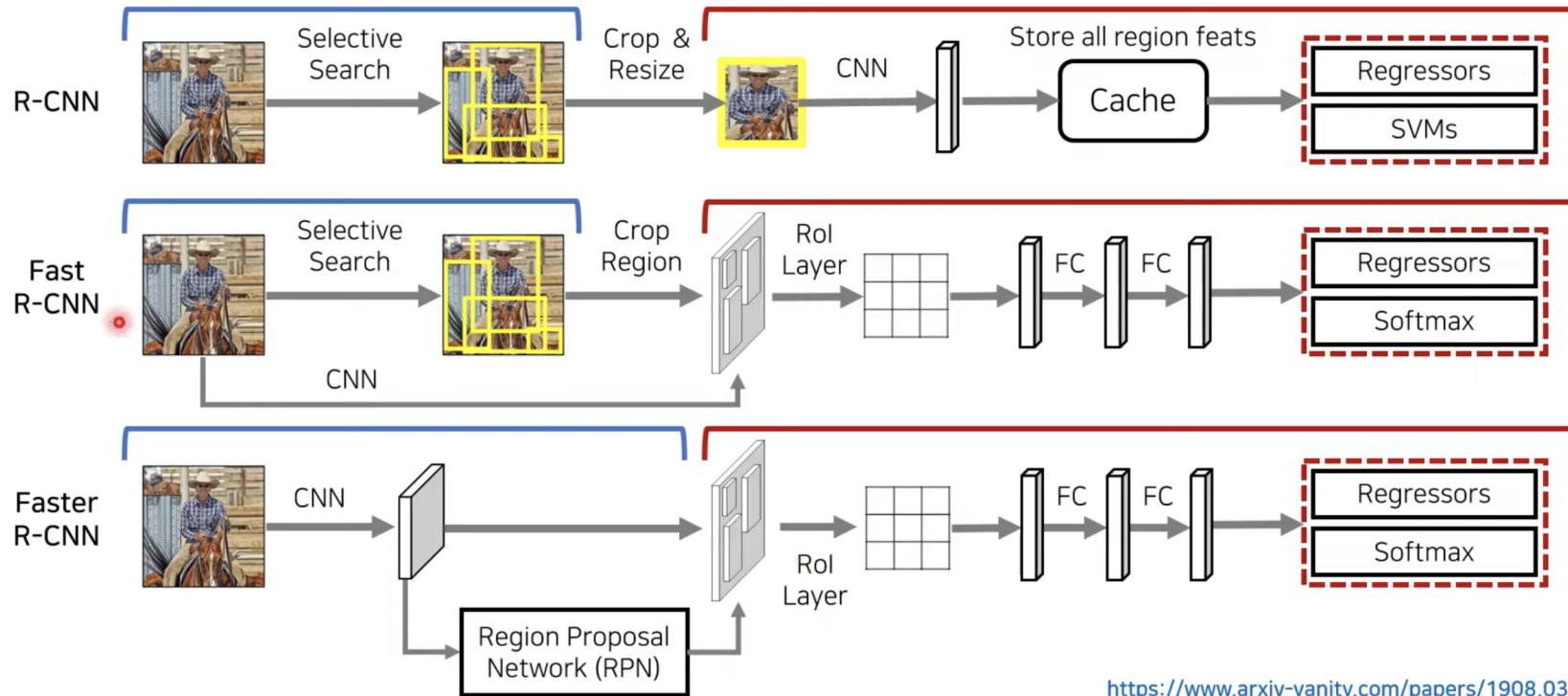
- **1-Stage 방식**

- 물체의 위치를 찾는 문제 (localization) 와 분류 (classification) 문제를 한 번에 해결
- YOLO 계열, SSD 계열 (SSD, RetinaNet, RefineDet ...)

1-Stage Detector - Regional Proposal와 Classification이 동시에 이루어짐.



Fast R-CNN



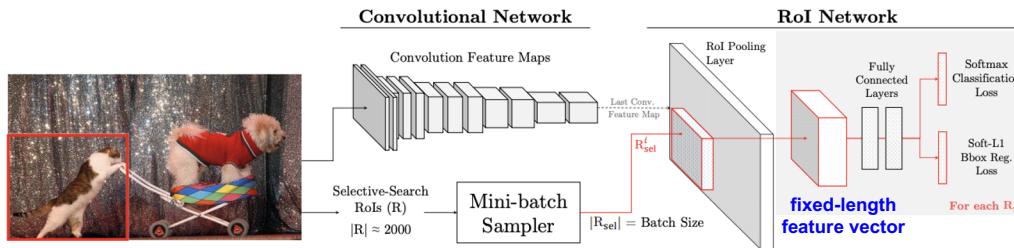
<https://www.arxiv-vanity.com/papers/1908.03673/>

Fast R-CNN

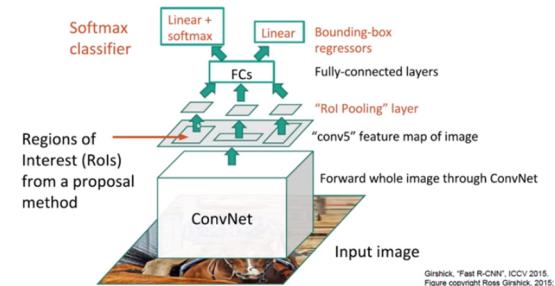
R-CNN (link) (CVPR 2014)	장점	CNN을 이용해 각 Region 의 클래스를 분류할 수 있음
	단점	전체 프레임워크를 end-to-end 방식으로 학습할 수 없음 따라서 global optimal solution 을 찾기 어려움
Fast R-CNN (link) (ICCV 2015)	장점	(selective search 를 제외한 다른 부분) feature extraction, RoI pooling, region classification, bounding box regression 단계 (step) 를 모두 end-to-end 로 묶어서 학습할 수 있음
	단점	여전히 첫 번째 selective search 는 cpu 에서 수행되므로 속도 느림 (bottleneck)
Faster R-CNN (link) (NIPS 2015)	장점	RPN을 제안하여, 전체 프레임워크를 end-to-end 로 학습할 수 있음
	단점	여전히 많은 컴포넌트로 구성되며, region classification 단계에서 각 특징 벡터 (feature vector)는 개별적으로 fc layer 로 forward 됨

Fast R-CNN

Fast R-CNN



Fast R-CNN



이전 R-CNN의 한계점 극복

1. RoI마다 CNN 연산을 함으로써 속도 저하
2. multi-stage pipelines 으로써 모델을 한번에 학습시키지 못함

Fast R-CNN

1. RoI pooling
2. CNN 특징 추출부터 classification, bounding box regression까지 하나의 모델에서 학습

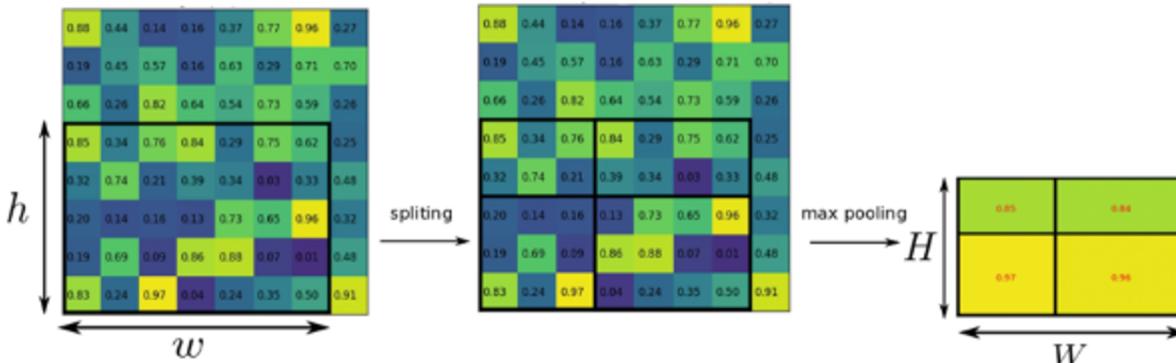
Fast R-CNN process

1. selective search 를 통해 2000개의 region proposal (RoI) 위치 찾음 ⇒ CPU 에서 진행 (Faster R-CNN 과 차이)
2. 전체 이미지를 CNN 에 통과시켜 하나의 feature map 추출
3. selective search 로 찾았던 RoI를 feature map 크기에 맞춰서 projection
4. projection 시킨 RoI 에 대해 RoI pooling 진행하여, 고정된 크기의 feature vector 얻음
5. 고정된 크기의 feature vector 는 fc layer 를 통과한 뒤, 두 브랜치로 나뉨
6. (a) softmax 를 통하여, RoI 에 대해 object classification
(b) bounding box regression 을 통해, selective search 로 찾은 box 의 위치 조정

Fast R-CNN

RoI Pooling

- fast r-cnn에서 먼저 입력 이미지를 cnn에 통과시켜 feature map 추출
- 그 후, 이전에 미리 selective search로 만들어 놨던 RoI를 feature map에 projection



input image and region proposal

pooling section

result

feature map

- $h \times w$ 크기의 검은색 box : 투영된 RoI

- (1) 미리 설정한 $H \times W$ 크기로 만들어주기 위해서 $(h/W) \times (w/H)$ 크기만큼 grid 를 RoI 위에 만들
- (2) RoI 를 grid 크기로 split 시킨 뒤, max pooling 을 적용하여 각 grid 간마다 하나의 값 주출

⇒ 이를 통해, feature map에 투영했던 $h \times w$ 크기의 RoI는 $H \times W$ 크기의 고정된 feature vector로

원래 이미지를 CNN에 통과시킨 후 나온 feature map에 이전에 생성한 RoI를 projection 시키고, 이 RoI를 FC layer input 크기에 맞게 고정된 크기로 변형

⇒ 따라서 더 이상 2000번의 CNN 연산이 필요하지 않고, 1번의 CNN 연산으로 속도 향상

End-to-End : Trainable

- 기존 R-CNN
 - CNN 을 통과한 후, 각각 서로 다른 모델인 SVM (classification), bounding box regression (localization) 안으로 들어가 forward 했기 때문에 연산 공유되지 않음.
 - bbox regression 은 CNN을 거치기 전의 region proposal 데이터가 input 으로 들어가고, SVM 은 CNN을 거친 후의 feature map 이 input 으로 들어가기에 연산이 겹치지 않음.
- ROI pooling 을 통해 각각의 region 들에 대해서 feature 에 대한 정보 추출 가능
 - CNN의 구조상 feature map 은 input 이미지에 대한 각각의 위치 정보를 어느 정도 보존하고 있음.
⇒ ROI 영역을 CNN을 거친 후의 feature map 에 투영시킬 수 있음.
 - 동일 data 가 각자 softmax (classification), bbox regressor (localization) 으로 들어가기에 연산 공유
- 모델이 end-to-end 로 한 번에 학습 가능하다는 뜻!
 - Multi-task loss

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v)$$

Where $L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x}, \text{y}, \text{w}, \text{h}\}} \text{smooth}_{L_1}(t_i^u - v_i)$.

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

classification 과 **localization loss** 를 합친 function 으로
써 한 번의 학습으로 둘 다 학습시킬 수 있음.

$p = (p_0, \dots, p_K)$: 예측된 Class Score
in $K + 1$ categories (0은 background)

u = 실제 Class Score

$t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$: 예측된 tuple
 (v_x, v_y, v_w, v_h) : 실제 Bounding box 좌표 값

$u \neq 0$ (background) 일때 0, 나머지일때 1

Heuristics method

Hard negative mining 한계점

- (1) fixed model 이 새로운 샘플을 찾아서 active training set 에 넣음
- (2) 모델이 이 active training set 을 기반으로 학습

Hard negative mining 한계점 ⇒ **Fast R-CNN** 에서는 **hard negative mining** 대신에 **heuristics method** 사용

Heuristics method

- 하이퍼 파라미터
 - IoU threshold, positive/negative sampling ratio 등..
- 2장의 이미지에서 각각 64개의 region proposal 추출 ⇒ 총 128 mini-batch 구성
- 128개의 region proposal 는 25% 는 positive sample, 75% 는 negative sample 로 추출
- positive sample : IoU 가 0.5 이상, negative sample: IoU가 0.1~0.5 사이의 proposal
- 한계점
 - 하이퍼 파라미터가 많아지면, 실험자의 개입과 시행착오 많아짐..
 - 또한 이러한 방식으로 mini-batch 를 생성하는 방법은 효과적이지 않음



OHEM (Online Hard Example Mining)

- base object detector : FRCN

Fast R-CNN 을 사용한 이유

- (1) basic two network (conv and RoI) 는 (당시) 최신 detectors (SPPnet, MR-CNN) 의 구조에 동일하게 사용되기 때문에, 넓게 사용하기 유용함
- (1) basic setup 은 유사함에도 불구하고, 고정된 conv network 를 사용하는 SPPnet, MR-CNN 과는 다르게, FRCN은 전체 conv network 를 학습함.
- (1) SPPnet, MR-CNN 은 따로 SVM 을 사용하여 RoI network 에서 얻은 feature를 학습하는 반면, FRCN의 경우 RoI network 자체에서 classifiers 를 학습함.

OHEM (Online Hard Example Mining)

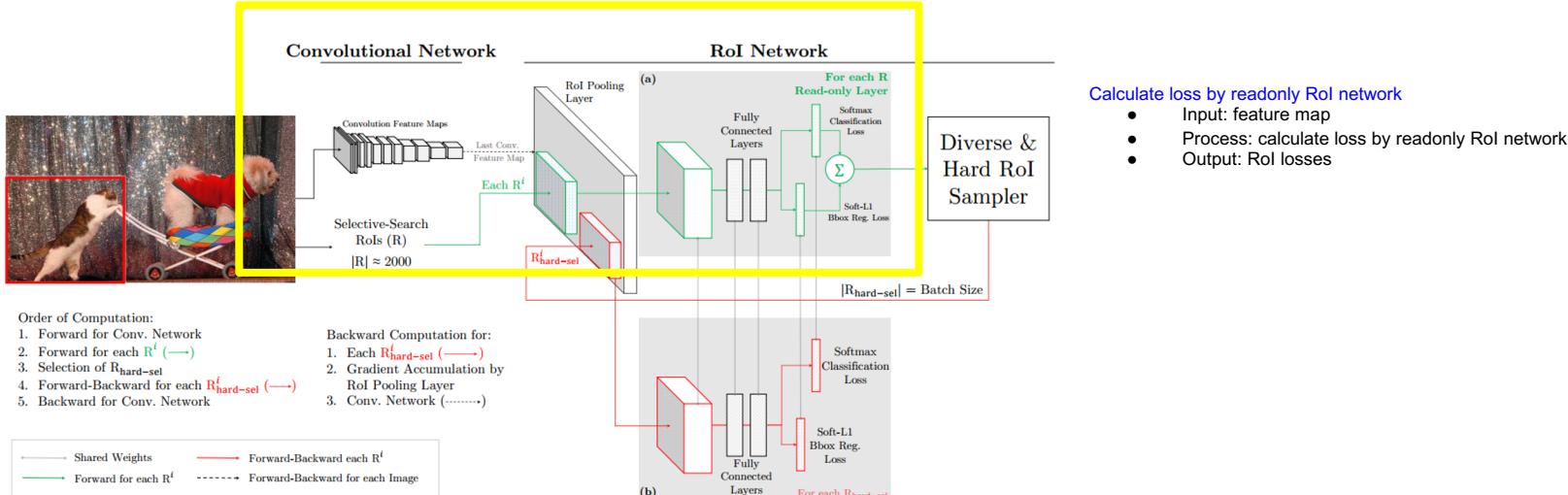


Figure 2: Architecture of the proposed training algorithm. Given an image, and selective search RoIs, the conv network computes a conv feature map. In (a), the *readonly* RoI network runs a forward pass on the feature map and all RoIs (shown in green arrows). Then the Hard RoI module uses these RoI losses to select B examples. In (b), these hard examples are used by the RoI network to compute forward and backward passes (shown in red arrows).

Fast R-CNN 모델에 OHEM 방법 적용

* RoI pooling layer 추가 : readonly RoI network - readonly (읽기 전용) *

- hard example을 샘플링하는 역할
- RoI pooling을 통해 얻은 모든 RoI에 대한 feature map을 readonly RoI network에 입력.
- 이 때, readonly RoI network는 fc layer, bbox regressor, classifier로 구성되어 있으며, forward pass만을 수행
 - 이를 통해 각각의 RoI에 대한 loss 계산
 - The loss represents how well the current network performs on each RoI

OHEM (Online Hard Example Mining)

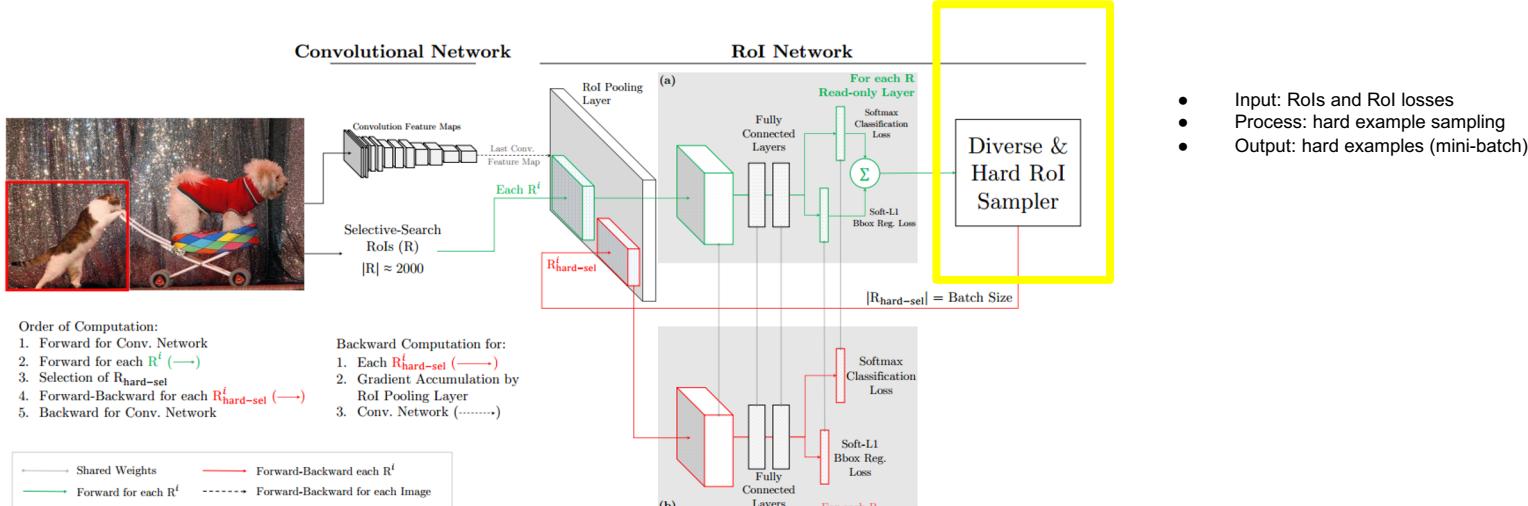


Figure 2: Architecture of the proposed training algorithm. Given an image, and selective search RoIs, the conv network computes a conv feature map. In (a), the *readonly* RoI network runs a forward pass on the feature map and all RoIs (shown in green arrows). Then the Hard RoI module uses these RoI losses to select B examples. In (b), these hard examples are used by the RoI network to compute forward and backward passes (shown in red arrows).

* select hard examples by Hard RoI sampler*

- 중복되는 sample 제거하기 위해 Non maximum suppression (NMS) 수행
- loss 를 내림차순으로 정렬한 뒤, 이미지별로 상위 B/N 개의 RoI만 선택
 - e.g. $N=2$, $B=128$ (2장의 이미지에서 128개의 RoI 선택하여) mini-batch 를 구성한다고 할 때, 각각의 이미지에서 loss 가 가장 높은 상위 64개의 RoI 만 선택
- $R_{\{hard-set\}}$

OHEM (Online Hard Example Mining)

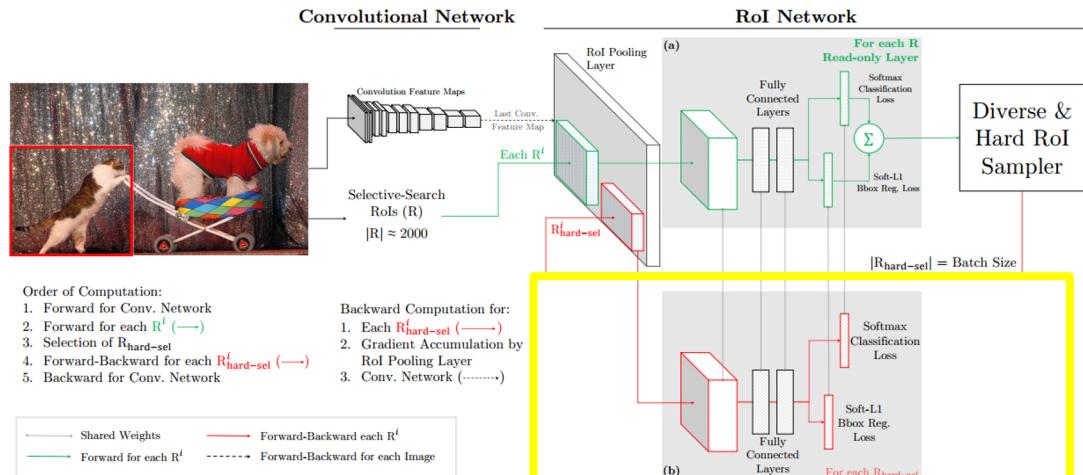


Figure 2: Architecture of the proposed training algorithm. Given an image, and selective search RoIs, the conv network computes a conv feature map. In (a), the *readonly* ROI network runs a forward pass on the feature map and all RoIs (shown in green arrows). Then the Hard ROI module uses these ROI losses to select B examples. In (b), these hard examples are used by the ROI network to compute forward and backward passes (shown in red arrows).

* Max pooling by ROI pooling *

- Hard ROI Sampler에서 얻은 hard example과 CNN을 통해 얻은 feature map을 사용하여 ROI pooling
 - 이를 통해, hard example 수 만큼의 feature map 얻음

* Train standard ROI network *

- feature map을 입력 받아 fc layer, bbox regressor, classifier를 거쳐 loss 계산한 후, backward pass를 통해 모델 학습.
- 이 과정에서 오직 hard example에 해당하는 ROI만이 학습에 참여함.

OHEM 장점

(1) 모델의 학습을 빠르게 할 수 있음

- 학습 도중 sampling 과정 필요 없음
- 기존 FRCN 의 hard example mining 은 10, 100 개의 이미지에서 example 을 모두 뽑을 때까지 모델 업데이트가 없었음

(1) 휴리스틱한 방법을 상대적으로 덜 사용

- positive/negative 를 균형있게 sampling 하기 위한 별도의 하이퍼 파라미터 필요 없음
(만약 특정 클래스의 example 이 backward pass 되지 않았다면, loss 는 상승함. -> 다음 iteration 때 선택되어 backward pass 될 가능성이 높아짐)

(1) detection 성능 향상

- sampling 을 통해 모델 성능 향상

Table 1: Impact of hyperparameters on FRCN training.

	Experiment	Model	<i>N</i>	LR	<i>B</i>	bg_lo	07 mAP
1	Fast R-CNN [14]	VGGM	2	0.001	128	0.1	59.6
2		VGG16					67.2
3	Removing hard mining heuristic (Section 5.2)	VGGM	2	0.001	128	0	57.2
4		VGG16					67.5
5	Fewer images per batch (Section 5.3)	VGG16	1	0.001	128	0.1	66.3
6						0	66.3
7	Bigger batch, High LR (Section 5.4)	VGGM	1	0.004	2048	0	57.7
8			2				60.4
9		VGG16	1	0.003	2048	0	67.5
10			2				68.7
11	Our Approach	VGG16	1	0.001	128	0	69.7
12		VGGM	2	0.001	128	0	62.0
13		VGG16					69.9

1) Heuristic sampling [1-2, 3-4, 11-13]

- Fast F-CNN에서의 heuristic 부분의 효과를 확인하기 위해, bg_lo (background threshold) 를 0.1로 둔 것과 0으로 둔 것의 차이)
- mAP가 2.4 정도 떨어짐.

1) Robust gradient estimate [5-6, 11]

- batch 당 1개의 이미지 사용
- 일반적으로 한 batch 당 2개의 이미지 사용, 그러나 OHEM에서는 loss 가 높은 것들만 골라서 훈련시키기 때문에, 한 이미지에서 샘플이 다 나올 수 있음.
- OHEM을 사용하지 않는 경우) 1 포인트 정도 떨어짐
- OHEM을 사용하는 경우) 거의 동일
⇒ 경우에 따라서 GPU 메모리를 위해 1개의 이미지만 배치에 넣어도 상관 없음.

1) hard examples [7-10, 11-13]

- what if we train with all the Rols, not just the hard ones?
- 어차피 easy example들은 쉽게 판별이 가능하여 loss 가 낮고, gradient descent에 큰 영향을 미치지 않을 것이기 때문.
- FRCN
 - $B = 2048, \text{bg_lo} = 0, N = \{1, 2\}$
- 원래 값인 1, 2 결과보다 1 포인트 정도 올랐지만, OHEM이 더 우수함.
- 또한 속도 역시 선별된 몇개의 example 만을 사용하여 back prop 하기 때문에 OHEM이 더 우수함

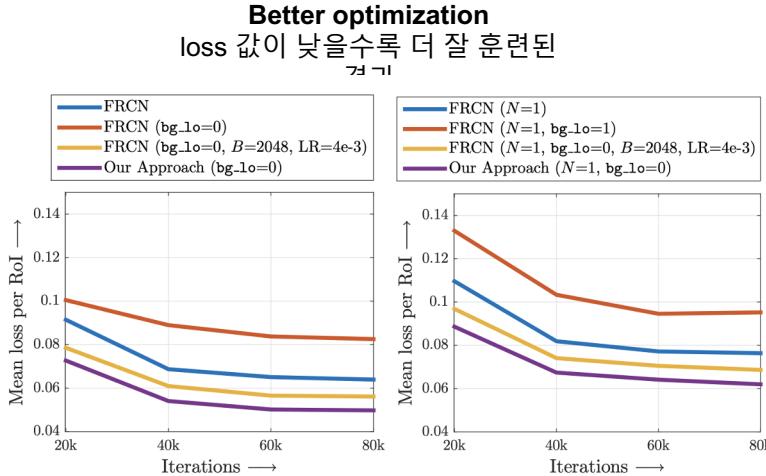


Figure 3: Training loss is computed for various training procedures using VGG16 networks discussed in Section 5. We report mean loss per ROI. These results indicate that using hard mining for training leads to lower training loss than any of the other heuristics.

Table 2: Computational statistics of training FRCN [14] and FRCN with OHEM (using an Nvidia Titan X GPU).

	VGGM		VGG16		
	FRCN	Ours	FRCN	FRCN*	Ours*
time (sec/iter)	0.13	0.22	0.60	0.57	1.00
max. memory (G)	2.6	3.6	11.2	6.4	8.7

*: uses gradient accumulation over two forward/backward passes

F-RCN이 빠른 detector 임을 감안할 때,
이 정도의 시간 증가는 수용 가능해 보임

Table 3: **VOC 2007 test** detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07+12**: union of **07** and VOC12 trainval. All methods use bounding-box regression. Legend: **M**: using multi-scale for training and testing, **B**: multi-stage bbox regression. FRCN* refers to FRCN [14] with our training schedule.

method	M	B	train set	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
FRCN [14]			07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
FRCN*			07	67.2	74.6	76.8	67.6	52.9	37.8	78.7	78.8	81.6	42.2	73.6	67.0	79.4	79.6	74.1	68.3	33.4	65.9	68.7	75.4	68.1
Ours			07	69.9	71.2	78.3	69.2	57.9	46.5	81.8	79.1	83.2	47.9	76.2	68.9	83.2	80.8	75.8	72.7	39.9	67.5	66.2	75.6	75.9
FRCN*	✓	✓	07	72.4	77.8	81.3	71.4	60.4	48.3	85.0	84.6	86.2	49.4	80.7	68.1	84.1	86.7	80.2	75.3	38.7	71.9	71.5	77.9	67.8
MR-CNN [13]	✓	✓	07	74.9	78.7	81.8	76.7	66.6	61.8	81.7	85.3	82.7	57.0	81.9	73.2	84.6	86.0	80.5	74.9	44.9	71.7	69.7	78.7	79.9
Ours	✓	✓	07	75.1	77.7	81.9	76.0	64.9	55.8	86.3	86.0	86.8	53.2	82.9	70.3	85.0	86.3	78.7	78.0	46.8	76.1	72.7	80.9	75.5
FRCN [14]			07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Ours			07+12	74.6	77.7	81.2	74.1	64.2	50.2	86.2	83.8	88.1	55.2	80.9	73.8	85.1	82.6	77.8	74.9	43.7	76.1	74.2	82.3	79.6
MR-CNN [13]	✓	✓	07+12	78.2	80.3	84.1	78.5	70.8	68.5	88.0	85.9	87.8	60.3	85.2	73.7	87.2	86.5	85.0	76.4	48.5	76.3	75.5	85.0	81.0
Ours	✓	✓	07+12	78.9	80.6	85.7	79.8	69.9	60.8	88.3	87.9	89.6	59.7	85.1	76.5	87.1	87.3	82.4	78.8	53.7	80.5	78.7	84.5	80.7

Table 4: **VOC 2012 test** detection average precision (%). All methods use VGG16. Training set key: **12**: VOC12 trainval, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval. Legend: **M**: using multi-scale for training and testing, **B**: iterative bbox regression.

method	M	B	train set	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
FRCN [14]			12	65.7	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7
Ours¹			12	69.8	81.5	78.9	69.6	52.3	46.5	77.4	72.1	88.2	48.8	73.8	58.3	86.9	79.7	81.4	75.0	43.0	69.5	64.8	78.5	68.9
MR-CNN [13]	✓	✓	12	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Ours²	✓	✓	12	72.9	85.8	82.3	74.1	55.8	55.1	79.5	77.7	90.4	52.1	75.5	58.4	88.6	82.4	83.1	78.3	47.0	77.2	65.1	79.3	70.4
FRCN [14]			07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Ours³			07++12	71.9	83.0	81.3	72.5	55.6	49.0	78.9	74.7	89.5	52.3	75.0	61.0	87.9	80.9	82.4	76.3	47.1	72.5	67.3	80.6	71.2
MR-CNN [13]	✓	✓	07++12	73.9	85.5	82.9	76.6	57.8	62.7	79.4	77.2	86.6	55.0	79.1	62.2	87.0	83.4	84.7	78.9	45.3	73.4	65.8	80.3	74.0
Ours⁴	✓	✓	07++12	76.3	86.3	85.0	77.0	60.9	59.3	81.9	81.1	91.9	55.8	80.6	63.0	90.8	85.1	85.3	80.7	54.9	78.3	70.8	82.8	74.9

¹<http://host.robots.ox.ac.uk:8080/anonymous/XNDVK7.html>, ²<http://host.robots.ox.ac.uk:8080/anonymous/H49PTT.html>,

³<http://host.robots.ox.ac.uk:8080/anonymous/LSANTB.html>, ⁴<http://host.robots.ox.ac.uk:8080/anonymous/R7EAMX.html>

Table 5: **MS COCO 2015 test-dev** detection average precision (%). All methods use VGG16. Legend: **M**: using multi-scale for training and testing.

AP@IoU	area	FRCN [†]	Ours	Ours [+M]	Ours* [+M]
[0.50 : 0.95]	all	19.7	22.6	24.4	25.5
0.50	all	35.9	42.5	44.4	45.9
0.75	all	19.9	22.2	24.8	26.1
[0.50 : 0.95]	small	3.5	5.0	7.1	7.4
[0.50 : 0.95]	med.	18.8	23.7	26.4	27.7
[0.50 : 0.95]	large	34.6	37.9	38.5	40.3

[†]from the leaderboard, *trained on trainval set

Conclusion

1. Heuristic 한 부분을 줄이고, 자동으로 hard example 이 선택될 수 있게 함
2. 이로 인해 훈련과정 단순화
3. 논문에서는 FRCN 을 이용하였지만, region-based ConvNet detector 를 이용하는 모델이라면 모두 적용 가능
4. PASCAL VOC 2007, 2012 등에서 SoTA 성능

End of the Document