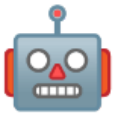


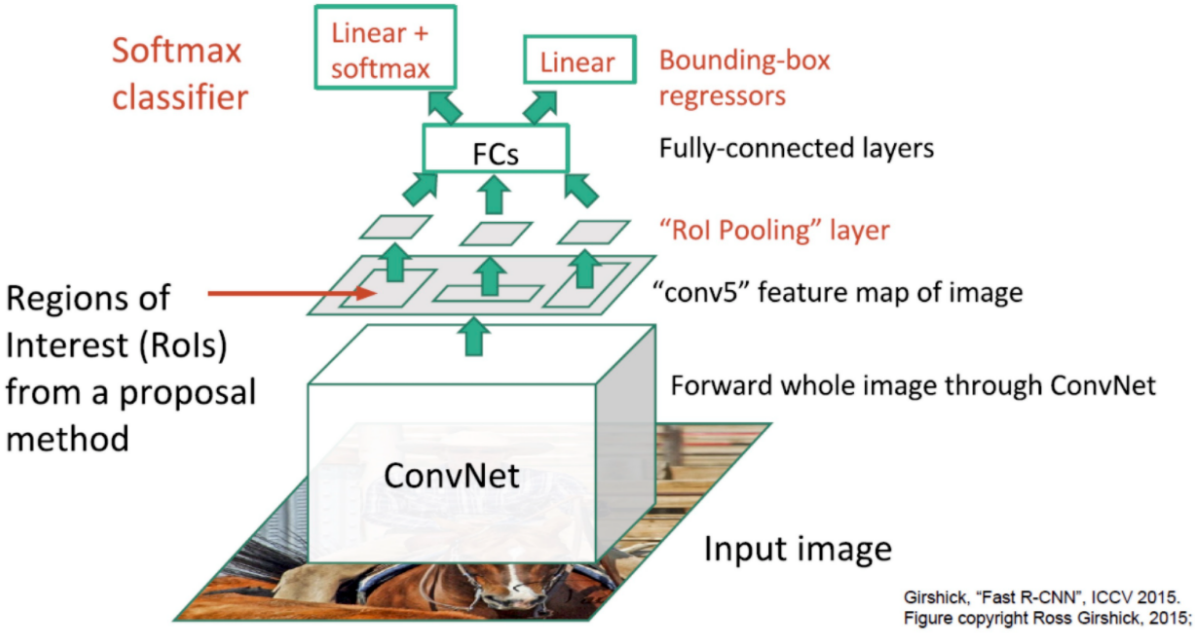
Faster R-CNN, OHEM

Created At	@2021년 8월 6일 오전 12:31
Created By	 조승제
Topics	Deep Learning Object Detection
Type	📒 Lesson
발표일	@2021년 8월 6일
발표자	 조승제
참고 링크 1	
참고 링크 2	
참석자	 조건우  엄현식
첨부 자료 1	
첨부 자료 2	
첨부 자료 3	

Faster R-CNN (2015)

1. Introduction

- Fast R-CNN의 단점을 보완
  - Fast R-CNN의 Test time 2.3초 중 2초가 Region Proposal에 소요됨
  - Fast R-CNN 또한 완벽한 의미의 end-to-end network는 아님 (SS)
- 이를 CNN 내부에서 수행할 수 있도록 Region Proposal Network(RPN)을 도입
- RPN은 region proposals를 보다 정교하게 수행하기 위해 Anchor box를 도입



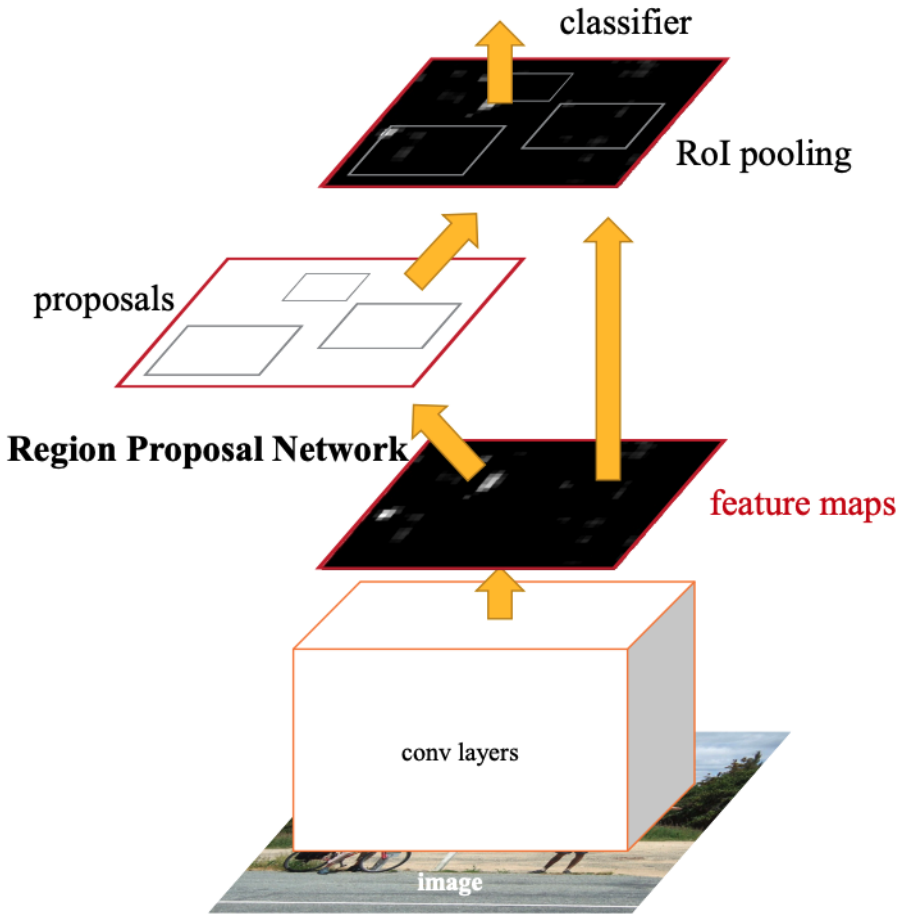


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

2. Anchor Box

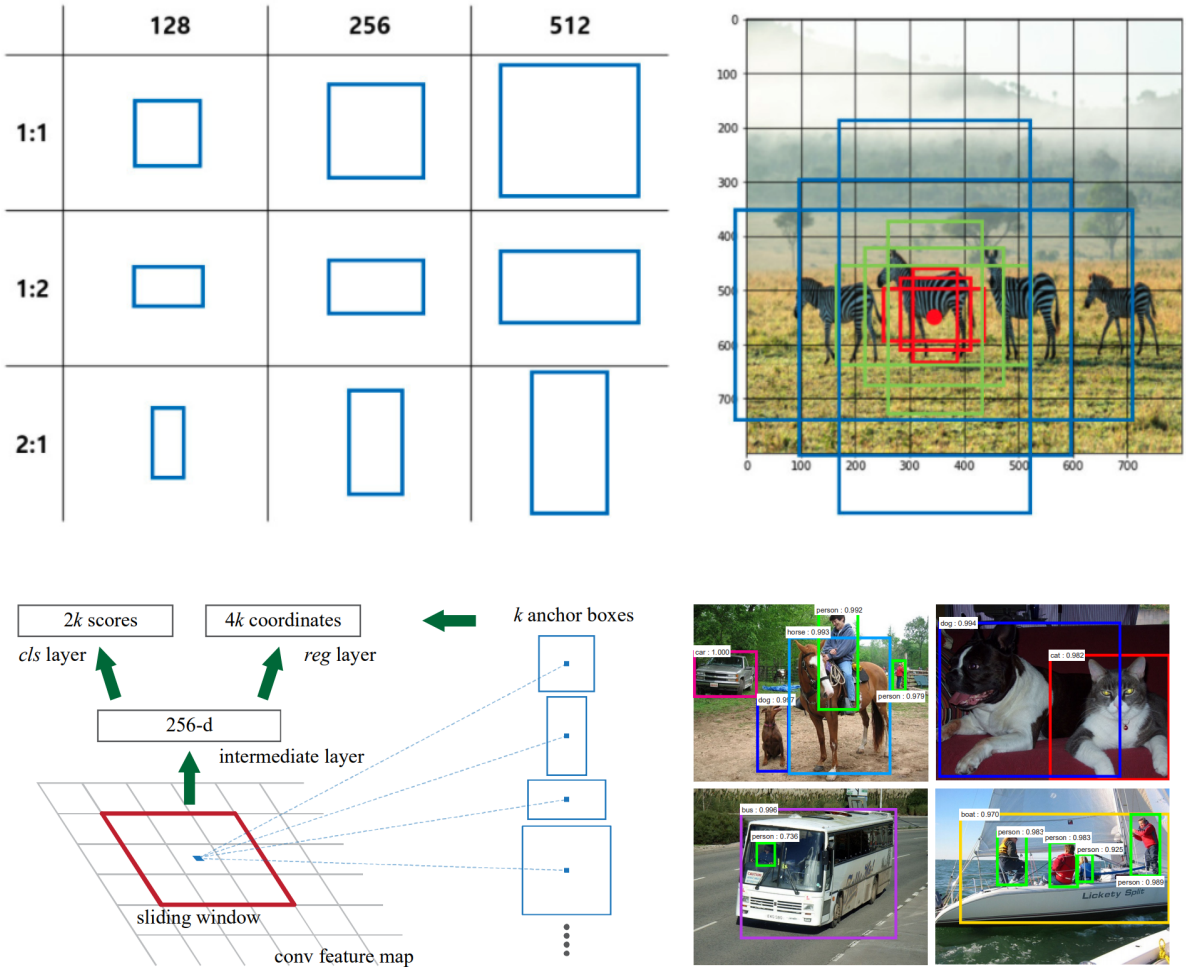
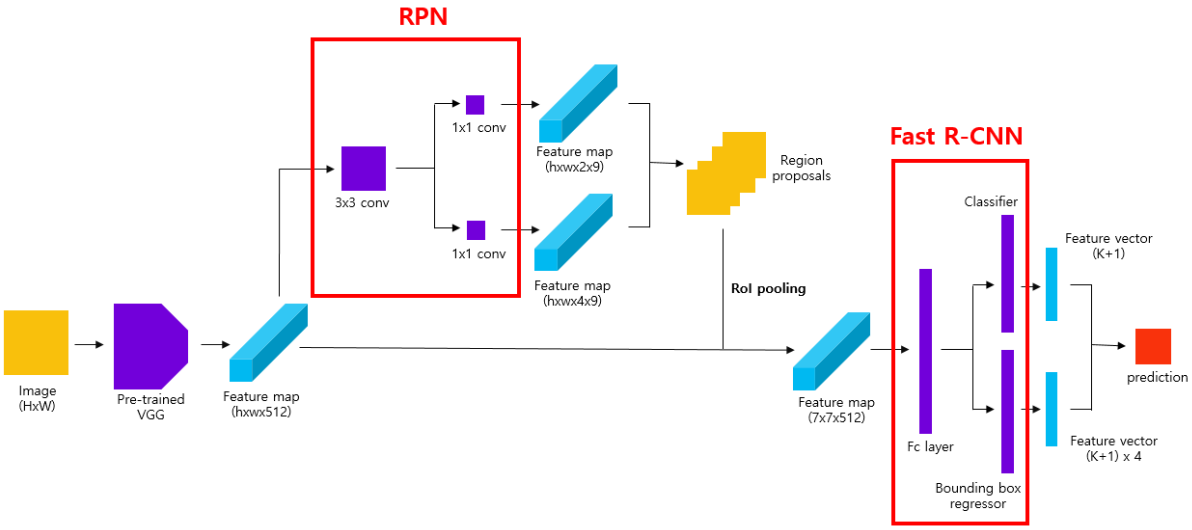


Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

- scale(w, h), aspect ratio(w, h의 비율)을 조정하여 다양한 크기의 anchor box 정의 가능
- 다양하게 정의하면 좀 더 정확한 localization 성능을 얻을 수 있고 다양한 크기의 객체를 포착하는 것이 가능
- selective search 방식보다 시간도 단축, mAP 또한 상승
- anchor box는 원본 이미지의 각 grid cell을 중심으로 생성하는데, 원본 이미지에서 sub-sampling ratio를 기준으로 anchor box가 생성됨

3. RPN (Region Proposal Network)

- CNN으로부터 feature map을 입력받아 anchor에 대한 class score, bounding box regressor를 반환하는 역할



1. 원본 이미지를 CNN을 통해 feature map을 얻음
2. feature map에 3x3 conv 연산 수행 (same padding)
3. class score 계산을 위해 1x1 conv 연산 수행 (RPN에서는 후보 영역에 객체가 포함되어있는지만을 분류)
4. bounding box regressor를 위해 1x1 conv 연산 수행
5. 이후 class score에 따라 상위 N개의 region proposals만을 추출한 후 NMS를 적용하여 Fast R-CNN에 전달

4. Loss Function

- For training RPNs, we assign a binary class label (of being an object or not) to each anch
- positive label 추출
  - the anchor/anchors with the highest Intersection-overUnion (IoU) overlap with a ground-truth box (가끔 적합하지 않은 데이터가 label되는 경우가 있어 사용하지 않음)
  - an anchor that has an IoU overlap higher than 0.7 with any ground-truth box

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

i = the index of an anchor in a mini-batch

p_i = the predicted probability of anchor i being an object

p_i^* = anchor가 positive일 경우 1, negative일 경우 0

t_i = vector representing the 4 parameterized coordinates of the predicted bounding box

t_i^* = ground-truth box associated with a positive anchor

L_{cls} = log loss over two classes (object vs. not object)

L_{reg} = Smooth L1 loss (bounding box regression)

N_{cls} = mini-batch size

N_{reg} = the number of anchor locations

λ = balancing parameter

5. Training

- RPN과 Fast R-CNN을 따로 훈련시킴

1. RPN을 훈련 (pre-trained model도 훈련됨)
    a. This network is initialized with an ImageNet-pre-trained model and fine-tuned end-to-end for the region proposal task.
2. Fast R-CNN 훈련 (pre-trained model도 훈련됨)
    a. we train a separate detection network by Fast R-CNN using the proposals generated by the step-1 RPN

b. At this point the two networks do not share convolutional layers

3. RPN을 훈련 (pre-trained model fix)
4. Fast R-CNN 훈련 (pre-trained model fix)

이 방법이 복잡해서 이후에 Alternating training(논문에서 사용) → Approximate Joint Training → Non-approximate joint training 으로 훈련 방법을 바꿈

6. EXPERIMENTS

Table 11: Object detection results (%) on the **MS COCO** dataset. The model is VGG-16.

method	proposals	training data	COCO val		COCO test-dev	
			mAP@.5	mAP@[.5, .95]	mAP@.5	mAP@[.5, .95]
Fast R-CNN [2]	SS, 2000	COCO train	-	-	35.9	19.7
Fast R-CNN [impl. in this paper]	SS, 2000	COCO train	38.6	18.9	39.3	19.3
Faster R-CNN	RPN, 300	COCO train	41.5	21.2	42.1	21.5
Faster R-CNN	RPN, 300	COCO trainval	-	-	**42.7**	**21.9**

Table 3: Detection results on **PASCAL VOC 2007 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: this number was reported in [2]; using the repository provided by this paper, this result is higher (68.1).

method	# proposals	data	mAP (%)
SS	2000	07	66.9†
SS	2000	07+12	70.0
RPN+VGG, unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	**73.2**
RPN+VGG, shared	300	COCO+07+12	**78.8**

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**

OHEM (2016)

1. Introduction

- In this paper, we propose a novel bootstrapping technique called online hard example mining (OHEM) for training state-of-the-art detection models based on deep ConvNets
- hard negative mining은 비효율적이다

2. Hard Negative Mining

- 모델이 잘못 예측한, 어려운(hard) sample을 추출하는 방법
- object detection 모델이 있다고 할 때, positive sample은 객체에 해당하는 영역이며, negative sample은 배경
- 여기서 모델이 예측하기 어려운 sample은 주로 False Positive sample
- 기존에는 Hard Negative Mining을 사용하여 모델이 예측하기 어려운 sample을 추출한 후, 학습 데이터에 포함시켜 모델이 False Positive 오류에 강건해지도록 학습

- 논문 저자는 Hard Negative Mining은 **휴리스틱(Heuristic)**하다고 언급.
- 지정해야 하는 하이퍼 파라미터(positive/negative sampling ratio, IoU threshold 등)가 많아 실험자의 개입과 시행착오가 많이 필요

→ False Positive sample를 mini-batch로 주는 대신 loss가 큰 데이터들로 훈련을 시키자

- loss가 큰 데이터에 gradiant가 크게 주어지니까

3. OHEM(Online Hard Example Mining)

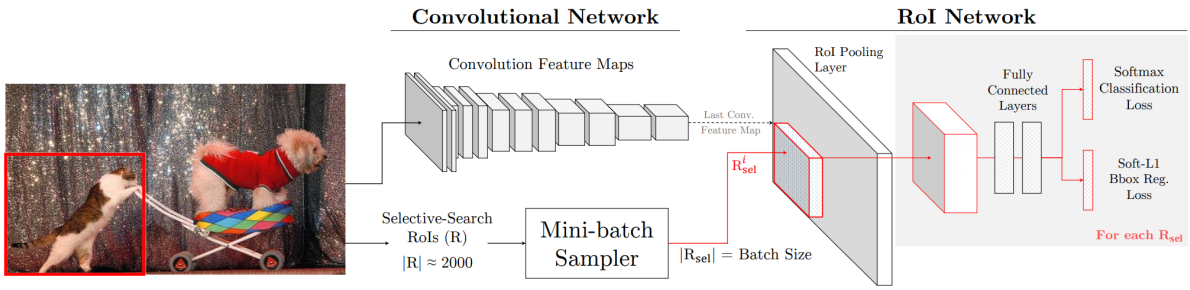


Figure 1: Architecture of the Fast R-CNN approach (see Section 3 for details).

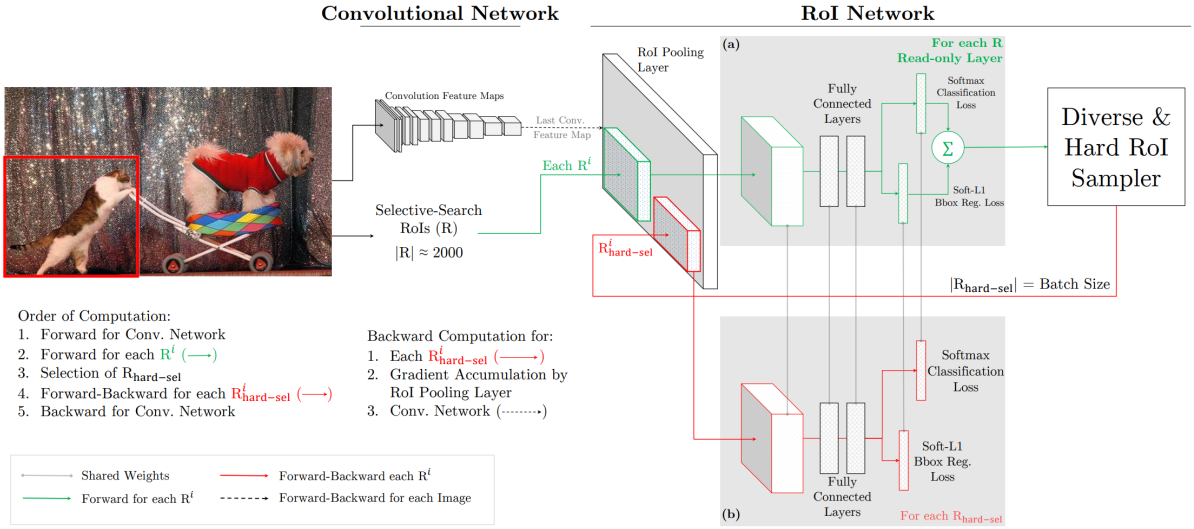
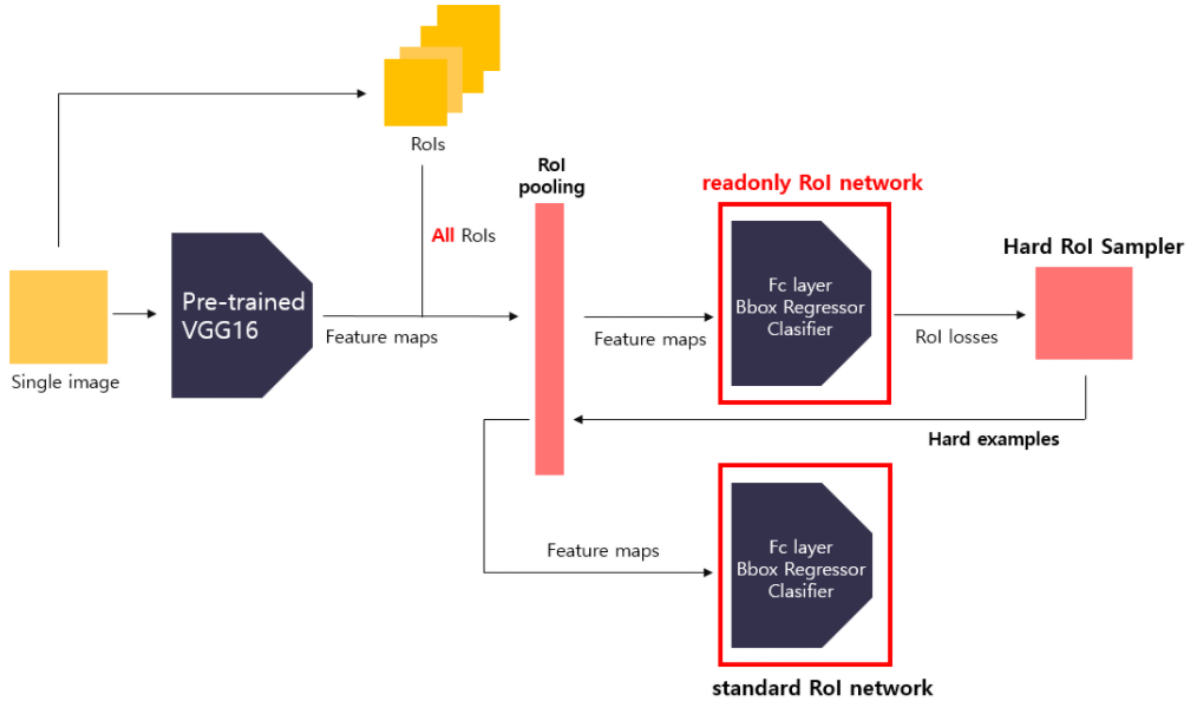


Figure 2: Architecture of the proposed training algorithm. Given an image, and selective search RoIs, the conv network computes a conv feature map. In (a), the *readonly* RoI network runs a forward pass on the feature map and all RoIs (shown in green arrows). Then the Hard RoI module uses these RoI losses 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).



4. OHEM Training

1. Fast R-CNN과 같은 Forward 수행
   a. Image → CNN → feature map
   b. Selective search → RoIs
2. a와 b 과정의 output으로 RoI pooling수행
3. FC Layer, Classifier, Bounding Box Regressor 거쳐 각 RoI별로 loss 계산
4. RoI를 내림차순으로 정렬 후 NMS(threshold = 0.7)를 적용해서 loss값이 큰 RoI만 back propagation

5. 장점

- Fast R-CNN은 mini-batch를 만들 때 RoI와 Ground Truth의 IoU가 0.5 이상이면 positive, 이하이면 negative로 sampling

  → Sampling 과정이 없어지면서 훈련 시간이 2배 이상 단축됨

- Sampling 과정에서 사용된 threshold등의 hyper parameter를 제거할 수 있게 됨

  → Heuristic 방식 X

- 인위적인 개입이 필요했던 부분들을 제거하면서 훈련 속도나 메모리 면에서도 빠르고 안정적으로 훈련이 가능해짐

6. EXPERIMENTS

Table 1: Impact of hyperparameters on FRCN training.

	Experiment	Model	N	LR	B	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

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