Fast R-CNN

Created At	@2021년 7월 29일 오전 1:49
Created By	조승제
Topics	Deep Learning Object Detection
Type	Lesson
발표일	@2021년 7월 29일
발표자	조승제
참고 링크 1	
참고 링크 2	
참석자	조건우 최소은 유정민 엄현식
첨부 자료 1	
첨부 자료 2	
첨부 자료 3	

1. Introduction

- R-CNN과 SPP-Net의 단점을 극복하기 위해 제안됨

- R-CNN 단점
    1. Training is a multi-stage pipeline
    2. Training is expensive in space and time
        a. For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk.
    3. Object detection is slow
        a. Detection with VGG16 takes 47s / image (on a GPU).
- SPP-Net 단점
    1. SPP-Net에서는 (4x4, 2x2, 1x1)의 spatial bin을 이용하면 다양한 resolution을 학습하게 되는 장점이 있다고 설명
    2. 하지만 하나의 객체에 대해서 다양한 resolution을 학습하게 되면 overfitting이 발생할 가능성이 높아짐
- Contributions

    We propose a new training algorithm that fixes the disadvantages of R-CNN and SPPnet, while improving on their speed and accuracy.

    1. Higher detection quality (mAP) than R-CNN, SPPnet
    2. Training is single-stage, using a multi-task loss
    3. Training can update all network layers
    4. No disk storage is required for feature caching

    Fast R-CNN is written in Python and C++ github

2. Fast R-CNN architecture and training

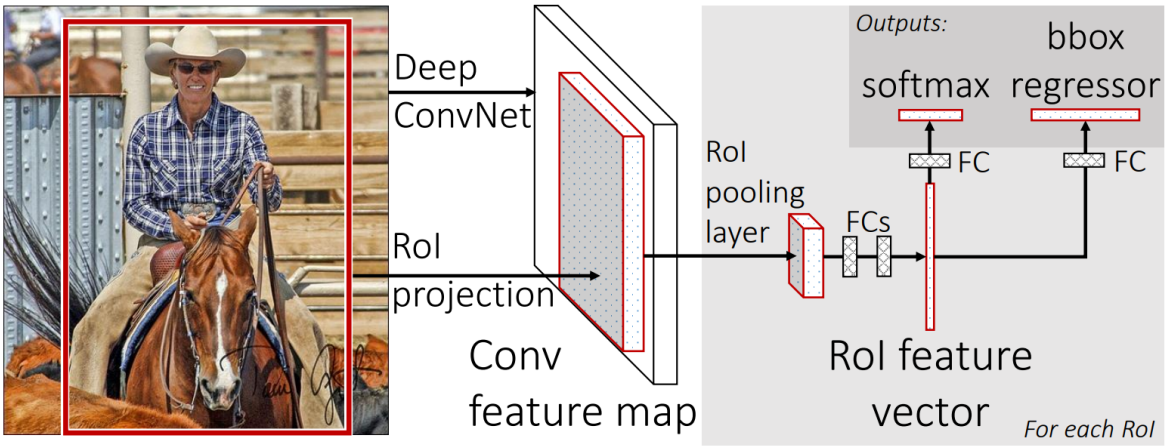
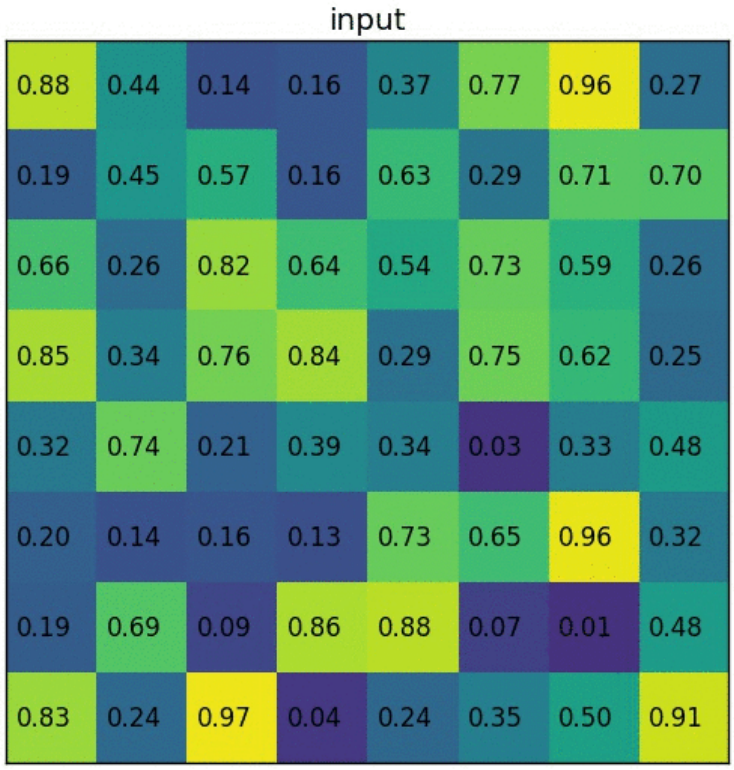


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

1) The RoI pooling layer

- region of interest into a small feature map with a fixed spatial extent of H × W (e.g., 7 × 7)
  - H and W are layer hyper-parameters
- RoI is a rectangular window into a conv feature map. Each RoI is defined by a four-tuple (r, c, h, w) that specifies its top-left corner (r, c) and its height and width (h, w)
- The RoI layer is simply the special-case of the spatial pyramid pooling layer used in SPPnets in which there is only one pyramid level.



2) Initializing from pre-trained networks

- 5개의 max pooling layer, 5~13개의 conv layers가 있는 ImageNet으로 Pre-trained 된 모델 3개로 Fast R-CNN의 CNN을 initializes해서 비교
- Pre-trained network는 3가지 변환을 거침
    1. 마지막 max pooling layer는 FC layer와 호환되도록 H, W값을 설정한 RoI Pooling layer로 대체
    2. Pre-trained network의 마지막 FC layer와 softmax는 위에서 언급한 서로 다른 레이어 2개로 대체
    3. 2가지 Input을 받을 수 있도록 수정 (Image, list of Rols)

3) Fine-tuning for detection

1. Hierarchical Sampling
    a. R-CNN에서는 여러 장의 이미지에서 랜덤하게 N개의 영역을 샘플링한 mini-batch를 구성하여 학습

b. Fast R-CNN에서는 1장 또는 2장(N)의 이미지에서 R개의 영역을 샘플링한 mini-batch를 사용하여 학습에 필요한 CNN 연산량을 효율적으로 줄임

c. 논문에선 N은 2, R은 128 사용

2. Multi-task loss

a. two sibling output layers를 가짐 (classification, bounding-box regression)

b. L1 distance를 계산하는 이유는 L2 distance를 계산할 시 gradient가 explode 되는 현상을 관찰했기 때문

$$p = (p_0, \ldots, p_K)$$

$$t^k = (t^k_{\mathrm{x}}, t^k_{\mathrm{y}}, t^k_{\mathrm{w}}, t^k_{\mathrm{h}})$$

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

$$L_{\mathrm{cls}}(p, u) = -\log p_u$$

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

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

u: fully-connacted 정답 벡터 배경 0 은 제외하고 계산한다

v : bounding box 정답 값

The hyper-parameter λ in Eq. 1 controls the balance between the two task losses

(λ = 1로 실험)

4) Mini-batch sampling

- N=2, R=128 → sampling 64 RoIs from each image
- mini-batch의 25%는 IoU값이 0.5 이상인 RoIs, 나머지 75%는 IoU값이 0.1~0.5의 RoIs
- 학습 과정에서 50% 확률로 Horizontal flip

5) Back-propagation through RoI pooling layer

- 이전 R-CNN, SPP Net은 각 Task 별로 fine-tuning을 진행했는데, 이는 성능 향상에 제약이 있다고 주장
- RoI Pooling Layer 이전까지 back-propagation이 가능한 것인지 검증

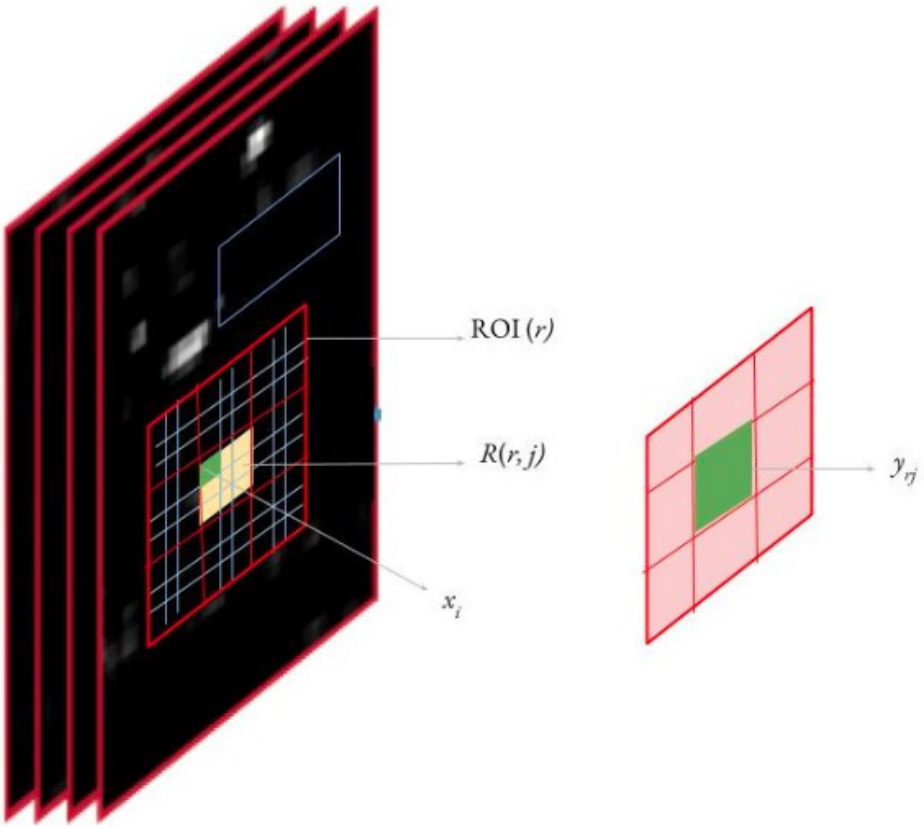
$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j \left[i = i^*(r,j) \right] \frac{\partial L}{\partial y_{rj}}.$$

Let xi ∈ R be the i-th activation input into the RoI pooling layer

Let yrj be the layer's j-th output from the r-th RoI

$$y_{rj} = x_{i^*(r,j)}$$

$$i^*(r,j) = \operatorname{argmax}_{i' \in \mathcal{R}(r,j)} x_{i'}$$



3. Main results

1. State-of-the-art mAP on VOC07, 2010, and 2012
2. Fast training and testing compared to R-CNN, SPPnet
3. Fine-tuning conv layers in VGG16 improves mAP

4. Experiments

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
SPPnet BB [11]†	07 \ diff	73.9	72.3	62.5	51.5	44.4	74.4	73.0	74.4	42.3	73.6	57.7	70.3	74.6	74.3	54.2	34.0	56.4	56.4	67.9	73.5	63.1
R-CNN BB [10]	07	73.4	77.0	63.4	45.4	**44.6**	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	**35.6**	66.8	67.2	70.4	**71.1**	66.0
FRCN [ours]	07	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	66.9
FRCN [ours]	07 \ diff	74.6	**79.0**	68.6	57.0	39.3	79.5	**78.6**	81.9	**48.0**	74.0	67.4	80.5	80.7	74.1	69.6	31.8	67.1	68.4	75.3	65.5	68.1
FRCN [ours]	07+12	**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	**70.0**

Table 1. **VOC 2007 test** detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07 \ diff**: **07** without "difficult" examples, **07+12**: union of **07** and VOC12 trainval. †SPPnet results were prepared by the authors of [11].

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
SegDeepM	12+seg	**82.3**	75.2	67.1	50.7	**49.8**	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	**41.5**	**71.9**	62.2	73.2	**64.6**	67.2
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1
FRCN [ours]	07++12	82.0	**77.8**	**71.6**	**55.3**	42.4	**77.3**	**71.7**	**89.3**	**44.5**	**72.1**	**53.7**	**87.7**	**80.0**	**82.5**	**72.7**	36.6	68.7	**65.4**	**81.1**	62.7	**68.8**

Table 2. **VOC 2010 test** detection average precision (%). BabyLearning uses a network based on [17]. All other methods use VGG16. Training set key: **12**: VOC12 trainval, **Prop.**: proprietary dataset, **12+seg**: **12** with segmentation annotations, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval.

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	63.2
NUS_NIN_c2000	Unk.	80.2	73.8	61.9	43.7	**43.0**	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	**38.6**	**68.3**	58.0	68.7	63.3	63.8
R-CNN BB [10]	12	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3	62.4
FRCN [ours]	12	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	65.7
FRCN [ours]	07++12	**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**	**68.4**

Table 3. **VOC 2012 test** detection average precision (%). BabyLearning and NUS_NIN_c2000 use networks based on [17]. All other methods use VGG16. Training set key: see Table 2, **Unk.**: unknown.

- 모델 크기 별 실험
  - S = CaffeNet (essentially AlexNet)
  - M = VGG_CNN_M_1024
  - L = VGG16

	SPPnet **ZF**		**S**		**M**		**L**
scales	1	5	1	5	1	5	1
test rate (s/im)	0.14	0.38	**0.10**	0.39	0.15	0.64	0.32
VOC07 mAP	58.0	59.2	57.1	58.4	59.2	60.7	**66.9**

Table 7. Multi-scale vs. single scale. SPPnet **ZF** (similar to model **S**) results are from [11]. Larger networks with a single-scale offer the best speed / accuracy tradeoff. (**L** cannot use multi-scale in our implementation due to GPU memory constraints.)

- SVM vs Softmax

method	classifier	S	M	L
R-CNN [9, 10]	SVM	**58.5**	**60.2**	66.0
FRCN [ours]	SVM	56.3	58.7	66.8
FRCN [ours]	softmax	57.1	59.2	**66.9**

Table 8. Fast R-CNN with softmax vs. SVM (VOC07 mAP).

- Speed Up

	Fast R-CNN			R-CNN			SPPnet
	S	**M**	**L**	**S**	**M**	**L**	†**L**
train time (h)	**1.2**	2.0	9.5	22	28	84	25
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	**0.06**	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	**213**×	-	-	-	-
VOC07 mAP	57.1	59.2	**66.9**	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

Table 4. Runtime comparison between the same models in Fast R-CNN, R-CNN, and SPPnet. Fast R-CNN uses single-scale mode. SPPnet uses the five scales specified in [11]. †Timing provided by the authors of [11]. Times were measured on an Nvidia K40 GPU.

5. Conclusion

- R-CNN, SPP-Net에 비해 뛰어난 성능
- multi-task loss 사용으로 single-stage training이 가능해짐
- Test time이 매우 단축됨
    - 하지만 이미지 1장 당 2.3초의 시간이 소요되므로 실시간 탐지에는 역부족
    - Region Proposal에 2.3초 중 2초가 걸림