Fast R-CNN

Abstract

- 본 논문은 object detection을 위한 Fast R-CNN을 제안하였다.
- 본 논문에서 제안한 model은 이전의 결과와 비교하여 정확도와 학습 속도를 개선하였다.

Fast R-CNN architecture and training

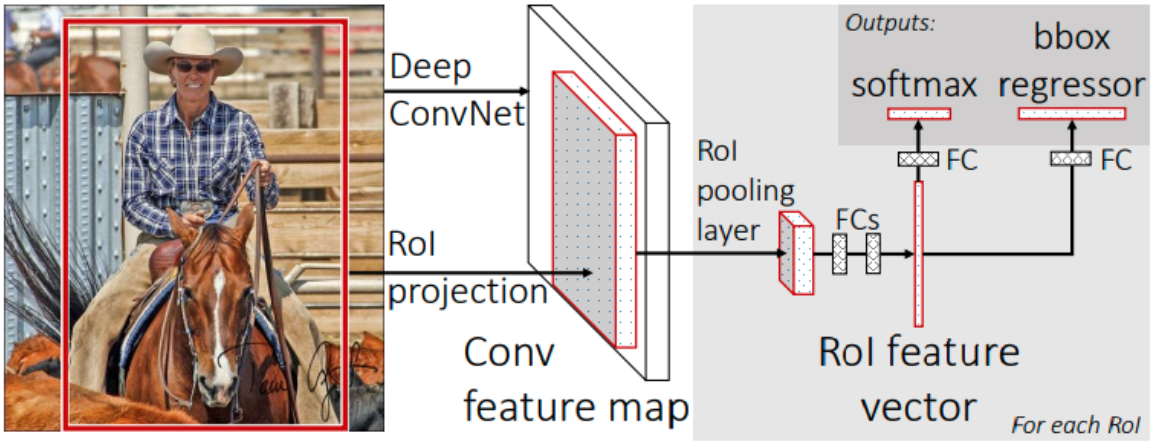

Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax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

위의 이미지에서 Fast R-CNN의 architecture을 확인할 수 있다. 본 network는 전체 이미지의 input과 object proposal의 set을 입력받는다. Network는 먼저 여러 convolutional을 통해 전체 이미지를 처리하고, 그 결과로 나온 feature map에 max pooling을 적용시킨다.

그 후에 각 object proposal 마다 region of interest(RoI) pooling layer이 feature map에서 fixed-length의 feature vector을 뽑아낸다.

그 결과는 fc layer에 입력드로 들어가 2개의 결과로 출력된다. 그 중 첫 번째 결과는 softmax를 통한 class별 확률값을 출력하고(background class 포함), 나머지 하나는 각 object class별 bounding-box를 위한 4개의 수를 출력해준다.

The RoI pooling layer

RoI pooling layer은 max pooling을 통해 유효한 RoI의 feature들을 $H \times W$ (height and weight)크기의 작은 feature map으로 변환한다.

본 논문에서 RoI는 rectangular window를 conv feature map으로 변환시킨다. 각 RoI는 (r, c, h, w) 의 형태로 정의되며, 그 중 왼쪽 위는 (r, c) 로 정의되고, height & width는 (h, w) 로 정의된다.

RoI max pooling은 $h \times w$ RoI window를 $H \times W$ 의 sub-window $(approximate\ size\ h/H \times w/W)$ 로 나누고, sub-window의 값들을 대응되는 output grid cell로 max-pooling한다. Pooling은 각 feature map channel에 독립적으로 적용된다.

Initializing from pre-trained networks

본 논문의 저자들은 3개의 pre-trained ImageNet network로 실험을 진행하였다.

구조는 5개의 5, 13개의 conv layer 사이에 max pooling layer이 위치한 형태이다. Pre-trained network가 Fast R-CNN을 initialize 할 때에 3 종류의 transformation이 진행된다.

첫 번째로 마지막 pooling layer이 RoI pooling layer으로 대체된다.

두 번째로 network의 마지막 fc layer과 softmax가 2개의 sibling layers로 대체된다(앞서 확인한 2개의 결과로 출력됨).

마지막으로 network가 2개의 input을 받도록 수정된다. 2개의 input은 이미지와 RoI 이미지의 list로 구성된다.

Fine-tuning for detection

모든 network를 back-propagation을 통해 학습시키는 것은 Fast R-CNN에서 중요한 capability이다.

이전의 SPPnet이 spatial pyramid pooling layer아래의 weight를 업데이트 하지 못하는 이유는 back-propagation을 시작하는 root가 효율적이지 못하였기 때문이다.

따라서 본 논문의 저자들은 더욱 효율적인 학습 방안을 제시하였다.

Fast R-CNN 학습 과정에서 SGD mini-batch가 계층적으로 구성된다. 먼저 N 장의 이미지를 sampling하고, R/N RoI를 각 이미지에서 뽑아낸다. 더하여 계층적 sampling에서 fine-tuning stage를 거친다.

Multi-task loss

Fast R-CNN network는 2개의 sibling output layer을 가진다. 첫 번째는 RoI별 확률 분포를 $K+1$ category별로 출력한다. 계산은 softmax를 이용한다. 두 번째는 bounding-box regression offset인 $t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$ 를 각 K class별로 출력한다(index by k).

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

- u : ground-truth class
- v : ground-truth bounding-box target
- $L(p, u) = -log\ p_u$: log loss for true class u
- L_{loc} : second task loss
- $[u \geq 1]$: 1 when $u \geq 1$ else 0

bounding-box regression에는 다음과 같은 loss가 사용된다.

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

in which

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

Mini-batch sampling

Fine-tuning 작업 중에 각 SGD mini-batch는 $N = 2$ 개의 이미지에서 uniform ramdom하게 구축된다. 본 논문에서는 $R = 128$의 mini-batch, 64 RoI per image가 사용된다. 자세한 수치들은 논문을 참고하도록 하자.

Back-propagation through RoI pooling layers

$x_i \in \mathbb{R}$ 을 $i-th$ RoI pooling layer에 들어가는 activation input이라고 하고, y_{rj} 를 RoI로 부터의 $r-th$ output이라고 가정하자. RoI pooling layer은 $y_{ri} = x_{i*(r,j)}, \ in \ which \ i*(r,j) = argmax_{i\prime \in R(r,j)} x_{i\prime}$ 이다. 이 때, $R(r,j)$ 는 y_{rj} max pools의 output unit인 sub-window의 index set of input이다.

RoI pooling layer의 backward function은 다음과 같다.

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

Scale Invariance

본 논문의 저자들은 scale invariant object detection을 achieve 하기 위해 2가지의 방법을 실험하였다.

1. 브루트 포스 학습
2. 이미지 피라미드

하지만 GPU limit 때문에 작은 실험만 진행할 수 있었다.

Fast R-CNN detection

Fast R-CNN이 finetune 된 후 detection은 forward pass를 하는 것보다 조금 더 많은 양이 가능하다. Input으로 한 이미지와 score으로 R object proposal의 list를 받는다. Test-time에 R 은 약 2000정도가 된다.

각 test RoI r 별로 forward pass은 class posterior probability distribution p 와 r 과 연관된 set of predicted bounding-box offset를 뽑아낸다. 그리고 $Pr(class = k|r) \triangleq p_k$ 를 사용하여 object class인 k 별로 detection confidence를 r 별로 할당한다. 그 후 non-maximum suppression을 각 class 별로 독립적으로 진행한다.

Truncated SVD for faster detection

전체 이미지 detection에서 fc layer의 computation time이 conv layer의 time 보다 빠르다. 반면 RoI를 detection하는 경우 많은 시간이 걸린다. 따라서 SVD를 사용해 그 문제를 해결하였다.

$$W \approx U \Sigma_t V^T$$

- U : $u \times t$ matrix comprising the first t left-singular vectors of W
- $\sum_t$: $t \times t$ diagonal matrix containing the top t singular values of W
- V : $v \times t$ matrix comprising the first t rightr-singular vectors of W

Main Results

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.

나머지 결과는 논문을 참고하자.