2. Fast R-CNN

Abstract

- 빠른 공간 기반 합성공 신경망 모델(Fast Resion-based Convolution Network method, Fast - RCNN)
- Fast R-CNN은 이전의 R-CNN과 비교해 속도도 빠르고 성능도 좋음
- Fast R-CNN은 PASCAL VOC 2012에서 R-CNN에 비해 VGG16 네트워크를 9배나 빠르게 훈련
- Fast R-CNN은 파이썬과 C++로 구현되어있음

Introduction

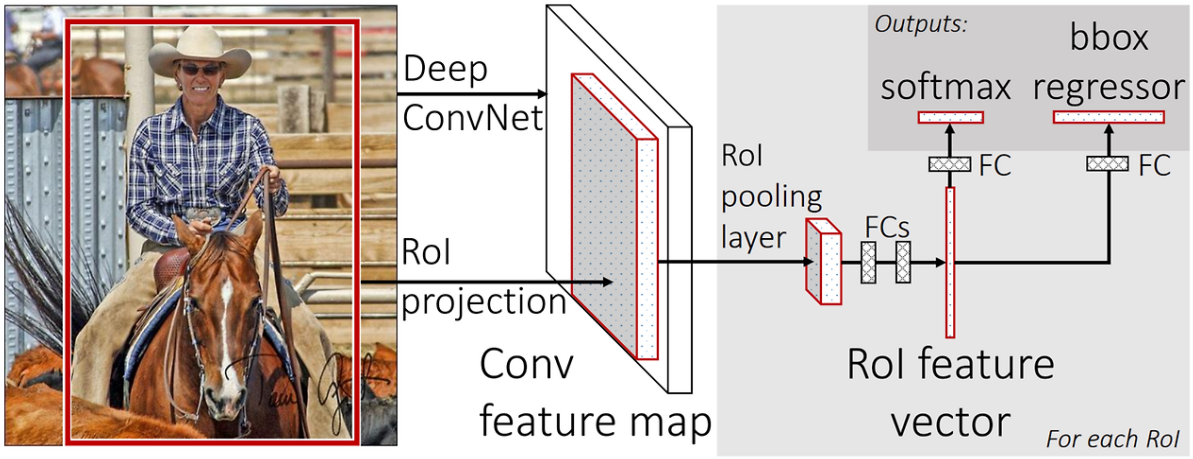
- 깊은 합성곱 신경망이 이미지 분류나 객체 탐지 성능을 크게 향상시켰으나 이미지 분류와 비교해서 객체 탐지 분야는 객체 위치까지 찾아야해 더 어려운 분야이고 객체 위치를 찾아야 하는 작업이 추가로 필요해 탐지 모델은 multi-stage 파이프라인을 가짐
    - multi-stage 파이프라인 때문에 속도가 느리고 파이프라인이 복잡함
- 객체의 위치를 정확히 찾아야 하는 작업 때문에 복잡해 짐
- 구체적으로 작업이 복잡해 지는 이유 2가지
    1. 여러 후보 영역(후보 경계 박스)을 처리해야 함
    2. 후보 영역은 대략적인 위치만 나타내는데 이를 바탕으로 정확한 위치를 알아내야 함
        a. 이런 복잡함을 해결하려면 속도, 정확성, 파이프라인의 간결성 가운데 하나와 타협해야함
- 이 논문에서는 합성곱 신경망 기반 객체 탐지 모델의 훈련 절차를 간소화 하는 방법을 소개함
- 객체를 분류하는 일(classify object proposals)과
  후보 영역에서 정확한 객체 위치를 탐색하는 일(refine their spatial locations)을 동시에 배우는 single - stage 훈련 알고리즘을 제안
- R-CNN and SPP-net
    - R-CNN은 객체의 후보 영역을 분류하는데 깊은 신경망을 사용하기 때문에 객체 탐지 성능이 우수하지만 단점이 있음

1. Training is a multi-stage pipeline
    a. R-CNN은 먼저 로그손실(long-loss)을 이용해 신경망을 Fine-Tuning함
    이어서 SVM으로 합성곱 신경망으로 구한 피처들을 처리함
    SVM이 객체 탐지기 역할을 해주고, 마지막으로 경계 박스 회귀를 진행함
        i. 합성곱 신경망도 훈련하고 SVM으로 분류를하고 경계박스 휘귀도 진행하여,
        세 가지 단계가 필요하여 multi-stage 파이프라인이라고함 → 복잡 그자체
2. Training is expensive in space and time
    a. R-CNN은 시간도 오래 걸릴뿐더러 저장공간이 많이 필요함
    b. SVM과 경계 박스 회귀 훈련을 위해서 이미지의 각 객체 후보 영역에서 피처를 추출
    c. 피처들은 디스크 공간을 차지함
3. Object Detection is slow
    a. 테스트 단계에서, 각 테스트 이미지마다 각 객체 후보 영역에서 피처를 추출
        i. 그렇기 때문에 GPU에서 객체를 탐지하는데 이미지 당 47초 소모 (VGG16 기준)
    b. 각 후보 영역마다 합성곱 신경망을 훈련해야 하기 때문에 R-CNN은 느림
    c. R-CNN은 후보 영역끼리 연산 결과를 공유하지 않지만, SPP-net은 연산 결과를 공유하기 때문에 R-CNN보다 속도가 빠름
    d. SPP-net은 먼저 입력 이미지 전체에서 합성곱 계층의 피처 맵을 구하고, 피처 맵에서 후보 영역들을 추출
        i. 후보 영역을 하나의 피처 맵에서 구한다는 말 → 피처 맵을 공유함
        ii. 각 후보 영역마다 spatial pyramid 풀링을 적용해 고정된 크기의 피처 벡터를 구함
        iii. 피처 벡터가 전결합 계층의 입력값이 됨
        iv. SPP-net의 단점
            1. R-CNN과 마찬가지로 훈련 구조가 multi-stage 파이프라인 구조

2. 피처 추출, 로그손실로 신경망 파인 튜닝, SVM 훈련, 경계 박스 회귀 구조
3. 피처들은 저장 공간을 많이 차지하지만 R-CNNN과 다르게 SPP-net에서 수행하는 파인 튜닝은 spatial pyramid 풀링 계층 앞에 있는 합성곱 신경망 계층을 업데이트 하지 않음
    a. 이런 한계는 깊은 신경망 모델의 정확성을 끌어올리는 데 지장을 줌

- Contributions
    - R-CNN과 SPP-net의 단점을 극복할 새로운 알고리즘인 Fast R-CNN을 제시
    - Fast R-CNN의 장점
        1. R-CNN, SPP-net보다 객체 탐지 성능이 좋음
        2. 훈련이 single-stage, multi-task loss를 사용함
        3. 훈련을 하면서 네트워크 전체 파라미터를 업데이트할 수 있음
        4. 피처 캐칭(feature caching)을 하기 위한 저장 공간이 필요 없음
            a. SPP-net에서 spatial pyramid 풀링 계층 앞에 있는 합성곱 신경망 계층은 업데이트 되지 않았음
            b. Fast R-CNN은 네트워크 전체 파라미터를 업데이트 할 수 있음

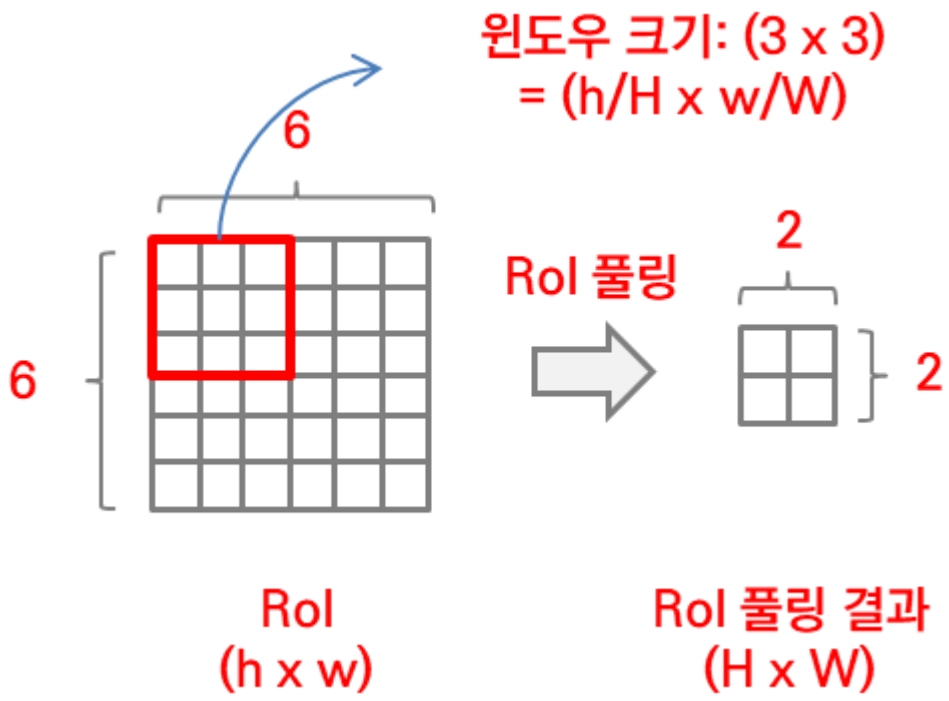
Fast R-CNN architecture and training



1. 전체 이미지를 먼저 여러 합성곱 신경망으로 처리함
2. 최대 풀링을 거쳐 피처맵을 구함

3. RoI(Resion of Interest)풀링 계층은 피처 맵에서 고정된 크기의 피처 벡터를 추출
4. 각 피처 벡터는 두 가지 전결합 계층의 입력값이 되어줌
    a. 첫 번째 전결합 게층은 소프트맥스로 클래스의 확률을 구함
    클래스 개수가 K 개 라고 할 때, 소프트맥스 결과값 개수는 총 (K+1)개 → 배경까지 포함하기 때문
    b. 두 번째 전결합 게층은 경계 박스의 좌표를 구함
        - 이러한 Fast R-CNN 구조는 multi-task loss값을 사용해 end-to-end 훈련을 할 수 있음

- The RoI Pooling Layer
    - RoI 풀링 계층은 최대풀링을 사용해 RoI 영역 안에 있는 피처를 작업 피처 맵으로 변환
        - 이 피처맵은 크기가 고정된 형태 → (H x W)형태
        - H와 W는 특정 RoI와 독립적인 하이퍼파라미터
        - RoI는 피처 맵으로 사상된(projection) 사각형 윈도우를 뜻함
        - RoI는 네가지 값을 갖는 튜플 형태(r, c, h, w)
            - (r, c) → 좌상단 좌표값
            - (h, w) → 높이와 너비
        - RoI풀링에 사용하는 윈도우 크기는 (h/H x w/W)



    - RoI계층은 Spp-net에서 사용하는 spatial pyramid pooling 계층과 비슷함 spatial pyramid pooling 계층에서 pyramid level이 딱 하나만 있는 경우로 볼 수 있음

- Initializing from pre-trained networks
    - Fast R-CNN도 이미지넷에서 사전 훈련된 네트워크를 뼈대로 사용함, 사전 훈련된 네트워크를 초기화할 때 세가지 변형을 가짐
        1. 마지막 풀링 계층을 RoI풀링 계층으로 바꿈
            a. 첫 번째 전결합 계층이(H x W)크기를 받을 수 있게(VGG16 H = W = 7)
        2. 네트워크의 마지막 전결합 계층과 소프트맥스는 두 가지 계층으로 나뉨
            a. {전결합+소프트맥스}계층과 경계 박스 회귀 계층
            - 이미지 분류에서 사용하는 VGG16은 마지막 부분에 전결합 계층을 거쳐 소프트 맥스를 적용함, 이미지를 분류만 하면 되기 때문임
            - Fast R-CNN은 RoI 풀링 계층을 거친 후에 두 가지 분기로 나뉘는데 이 때{전결합 + 소프트맥스}계층과 경계 박스 회귀 계층으로 나뉨
        3. 네트워크가 입력 데이터 두 개를 받도록 수정함
            a. 이미지 데이터와 해당 이미지의 RoI 데이터를 받을 수 있도록
- Fine-tuning for detection

- 역전파로 네트워크의 모든 가중치를 훈련할 수 있다는 점은 Fast R-CNN의 가장 큰 장점
- SPP-net은 spatial pyramid 풀링 계층 이전의 가중치를 왜 업데이트하지 못 했는지
  - 훈렴샘플, 즉 RoI를 서로 다른 이미지에서 구할 때는 SPP 계층을 통해 역전파를 적용하는 것이 굉장이 비효율적 → R-CNN과 SPP-net은 이런 방식으로 네트워크를 훈련함
  - 각 RoI가 굉장히 큰 receptive field를 갖기 때문에 비효율적,
    RoI가 입력 이미지의 꽤 많은 영역을 차지함 → 거의 입력 이미지 전체를 훈련하는 꼴
- Fast R-CNN은 훈련 하는 동안 피처를 공유하는 방식으로 제안함
  - Fast R-CNN 훈련에서 SGD 미니 배치가 샘플링 됨
  - 먼저 N개 이미지를 샘플링하고, 이어서 샘플링한 각 이미지에서 R/N개의 RoI를 샘플링
  - 중요한점은 같은 이미지에서 뽑은 RoI는 순전파와 역전파를 하면서 연산 결과와 메모리를 공유하는 것
  - N이 작을수록 미니 배치 연산은 줄어듬
    - 예를들어, N = 2, R = 128일 때 서로 다른 128개의 이미지에서 RoI 한개를 뽑을 때 보다 64배 빠름

  i. 2개(=N)의 이미지를 샘플링하여 각 이미지마다 RoI 64개(=R/N=128/2)를 샘플링 하기 때문에 128개를 샘플링하는 경우와 2개를 샘플링하는 경우의 속도 차이는 64배가 남
  2개의 이미지에서 64개의 RoI를 샘플링할 때는 RoI가 연산과 메모리를 공유하기 때문에 손실이 발생하지 않음
  - Fast R-CNN은 소프트맥스 분류기, 경계 박스 회귀를 동시에 최적화하도록 한 번에 파인 튜닝을 진행함(one-stage) → R-CNN은 multi-stage

Multi-task loss

- Fast R-CNN은 두 가지 출력 계층을 가짐
  1. 첫 번째 계층은 RoI마다 (K+1)개 확률값을 출력
     a. K는 클래스 개수인데 배경까지 더해 K+1이 됨
  2. 두 번째 계층은 경계 박스 좌표를 출력

a. K개 객체마다 (좌상단 x좌표, 좌상단 y좌표, 너비, 높이)를 구해줌

- multi-task loss 수식
  - p = 예측 클래스 값
  - u = 실제 클래스 값,
  - t = 예측 경계 박스 좌표
  - v = 실제 경계 박스 좌표

① 클래스 분류 손실값 : 로그손실 ② 경계 박스 회귀 손실값

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

Multi-task loss ④ 람다 ③ 아이버슨 괄호

  - ①은 클래스 분류 손실값. 로그손실로 계산
  - ②는 경계 박스 회귀의 손실값 이때 ③ 아이버슨 괄호를 곱해줌
    - 아이버슨 괄호는 u가 1 이상일 땐 1, 1 미만일 땐 0을 반환하는 함수
    - u = 1은 객체가 있다는 뜻, u = 0은 객체가 없다는 뜻 → u = 0은 배경
    - 따라서, 객체가 있을 때만 경계 박스 회귀 손실값이 필요하고 배경일 땐 필요 없음
  - ④는 클래스 분류 손실값과 경계 박스 회귀 손실값 사이의 균형을 맞춰주는 하이퍼 파라미터
  - ②경계 박스 회귀 손실값은 다음과 같이 구함

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

in which

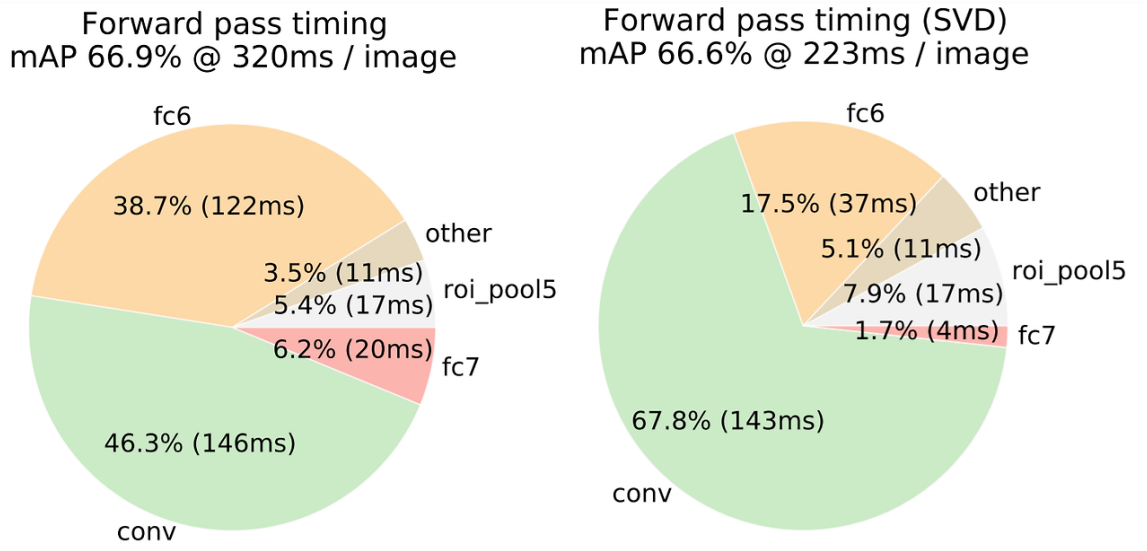
$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

Mini-batch sampling

- 파인 튜닝하는 동안 각 SGD 미니 배치는 N=2가 되게끔 이미지를 샘플링함
  - R = 128이므로, 각 이미지마다 RoI를 64개씩 뽑음
- 후보영역에서 실제 경계 박스와 IoU가 0.5이사인 RoI를 25% 뽑음
  - 이 RoI를 Positive 샘플로 간주 즉, u = 1
- 후보 영역에서 뽑은 나머지 RoI 중 실제 경계 박스와 IoU가 0.1 이상 0.5 미만인 RoI는 배경으로 간주
  - negative 샘플로 간주 즉, u = 0
- 데이터 증강을 위해 훈련하는 동안 50% 확률로 좌우 대칭 진행
- Scale invariance
  - 스케일 불변한 객체 탐지를 수행하기 위해 두 가지 실ㄹ험을 진행
    1. brute-foce 학습
       a. brute-force 방식에서는 훈련과 테스트에서 모두, 각 이미지를 사전에 정의한 픽셀 크기로 처리
    2. 이미지 피라미드 방식
       a. 이미지 피라미드를 통해 스케일 불변성을 갖게하여 사전에 정의한 크기로만 학습하는게 아닌 여러 이미지 크기를 활용
       b. 테스트 단계에서는 정규화된 후보 영역을 위해 이미지 피라미드를 사용
          i. 훈련 단계에서는 피라미드 스케일을 무작위로 뽑음 → 데이터 증강형식
          ii. GPU 메모리 한계 때문에 작은 네트워크에서만 multi-scale훈련을 진행

Fast R-CNN Detection

- Truncated SVD for faster detection
  - 이미지 분류 시 순전파를 할 때, 전결합 계층에서 소요하는 시간이 합성곱 계층에서 소요하는 시간 대비 상대적으로 짧음
    - 하지만 객체 탐지를 할 때는 그렇지 않음, 처리해야 할 RoI가 많기 때문에 전결합 계층에서 소요하는 시간이 거의 절반을 차지
  - Truncated 특이값 분해(SVD)를 적용하면 전결합 계층에서 소요하는 시간이 훨씬 짧아



- Truncated SVD를 적용하기 전과 후의 각 계층별 순전파 소요 시간을 나타

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.

- 기존 모델과 Fast R-CNN의 성능을 비교한 표 → FRCN이 Fast R-CNN의 약자
- Fast R-CNN이 다른 모델들 보다 mAP가 더 좋으며 데이터셋을 많이 사용할수록 mAP가 더 높아짐

- Training and testing time

	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	-	-	-	-

  - Fast R-CNN의 속도
  - 훈련시간(hours) 테스트 ㅅ속도(seconds per image) 그리고 mAP
  - S, M, L은 각각 small, medium, large의 약자로 모델의 크기를 의미
  - Fast R-CNN(L)은 R-CNN보다 테스트 처리 속도가 146배 빠름(SVD 미적용)
  - SVD 적용 시, 213배 더 빠름 훈련시간 84시간 → 9.5시간으로 단축
  - Fast R-CNN은 피처를 공유하기 때문에 저장 공간도 수백 GB 절약 가능
- Which layers to fine-tune?
  - SPP-net은 Fast R-CNN보다 깊은 신경망이 아니기 때문에 전결합 계층만 파인 튜닝해줘도 충분했음
  - Fast R-CNN이 꽤나 깊은 신경망이라서 파인 튜닝을 합성곱 계층까지 적용해보았음

	layers that are fine-tuned in model **L**			SPPnet **L**
	≥ fc6	≥ conv3_1	≥ conv2_1	≥ fc6
VOC07 mAP	61.4	66.9	**67.2**	63.1
test rate (s/im)	**0.32**	**0.32**	**0.32**	2.3

        - SPP-net은 전결합 계층만 파인튜닝했는데 mAP가 63.1인 반면, Fast R-CNN은 전결합 계층만 파인 튜닝할 경우 mAP가 SPP-net보다 낮은 61.4이지만, conv2_1 이후 합성곱까지 파인튜닝하면 67.2로 늘어가며, conv3_1 이후 합성곱까지 파인튜닝 시 66.9가 됨
    - 모든 합성곱 계층을 파인 튜닝하는 것이 항상 좋다는 건 아니며 크기가 작은 네트워크의 첫 번째 합성곱 계층(conv1)은 task와 무관함(task - idenpendent)
        - 태스크와 무관하다는 말은 어떤 이미지를 분류하든 관계없이 파라미터가 같다는 말
            - 고양이와 강아지를 구분하는 작업이든, 소와 말을 구분하는 작업이든 첫 번째 계층에서 수행하는 일은 같다는 의미
            - 세세한 분류가 아니라 형상이나 선 등을 구분하는 일을 하는데는 객체 종류와 상관없기 때문
    - 첫 번째 계층을 파인 튜닝하든 하지 않든 mAP에는 영향을 주지 않음, VGG16에서는 conv3_1 이후로 파인 튜닝을 진행하면됨

      (1) conv2_1 이후로 파인 튜닝하면 conv3_1 이후 파인 튜닝하는 것에 비해 훈련 속도가 1.3배 느려짐

      (2) conv1_1 이후로 파인튜닝하면 GPU메모리를 초과하며, conv2_1이후로 파인튜닝한 mAP는 conv3_1 이후 파인 튜닝한 mAP에 비해 겨우 0.3점 높음
        - 본 논문에서 사용한 모든 Fast R-CNN 결과는 VGG16의 conv3_1 이후로 파인 튜닝한 결과
        - 또한 모든 작업 모델(S, M)은 conv2 이후로 파인튜닝 진행

Design evaluation

- Fast R-CNN을 이해하기 위해 PASCAL VOC 2007 데이터셋에서 실험 진행한 결과
- Dose multi-task training help?
    - multi-task 훈련은 편리함, task를 연속으로 훈련하는 파이프라인을 사용하지 않아도 되기 때문
    - multi-task 훈련을 할 때는 합성곱 계층의 파라미터를 공유하기 때문에 성능이 더 좋아질 가능성이 있음
    - 그렇다면 multi-task 훈련이 Fast R-CNN의 객체 탐지 성능을 높여주는가?

① 클래스 분류 손실값 : 로그손실 ② 경계 박스 회귀 손실값

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

Multi-task loss ④ 람다 ③ 아이버슨 괄호

- 전체 손실에서 분류 손실만 남겨둔상태로 베이스라인 네트워크를 훈련해봐야 함
- 위 수식이 손실을 구하는 수식인데, 해당 수식에서 λ = 0으로 설정하면 ①분류 손실만 남음

	S				M				L			
multi-task training?		✓		✓		✓		✓		✓		✓
stage-wise training?			✓				✓				✓	
test-time bbox reg?			✓	✓			✓	✓			✓	✓
VOC07 mAP	52.2	53.3	54.6	**57.1**	54.7	55.5	56.6	**59.2**	62.6	63.4	64.0	**66.9**

- 두 번째 열은 multi-task 손실(즉, 전체 손실 수식에서 λ=1로 설정한 경우)을 사용해 훈련한 모델을 나타냄
  - 단, 테스트 단계에서 경계 박스 회귀는 사용하지 않았음
  - 이렇게하면 multi-task 손실을 사용하지 않은 베이스라인 모델과 성능을 일대일로 비교할 수 있음
  - multi-task 훈련을 할 때가 그렇지 않을 때 보다 mAP가 높음 → multi-task 훈련 효과 검증
  - 분류손실로만 훈련된 베이스라인 모델에 경계 박스 회귀 계층을 덧붙이고 ② 경계 박스 회귀 손실만을 사용해 훈련
    - 세 번째 열에 결과 → 곧 stage-wise 훈련은 아무것도 하지 않은 베이스라인 보다는 성능이 좋지만 multi-task 훈련보다는 성능이 나쁨

- Scale invariance : to brute force or finesse?
  - 스케일 불변한(scale-invariant) 객체 탐즈를 위해 두 가지 기법을 비교
    1. 무차별 대입 학습(brute-force-learning)
       a. 무차별 대입 학습은 단일 스케일 훈련
    2. 이미지 피라미드 기법
       a. 이미지 피라미드 기법은 멀티 스케일 훈련
  - 두 기법을 사용할 때, 이미지 너비와 높이 중 더 작은 값을 s로 정의

- 모든 단일 스케일 실험에서는 s = 600 pixel
- 너비 혹은 높이 중 긴 부분이 1,000 pixel을 넘지 않도록 한도를 정하기 때문에 s가 600 pixel보다 작기도함
- PASCAL VOC 데이터셋 이미지는 평균적으로 384 * 473 크기
  - 단일 스케일 훈련을 할 때는 1.6배 키워 사용(384 * 1.6 = 약 600)
- 멀티 스케일 훈련을 할 때는 다섯 가지 s 값을 사용
  - s ∈ {480, 576, 688, 864, 1200}

	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**

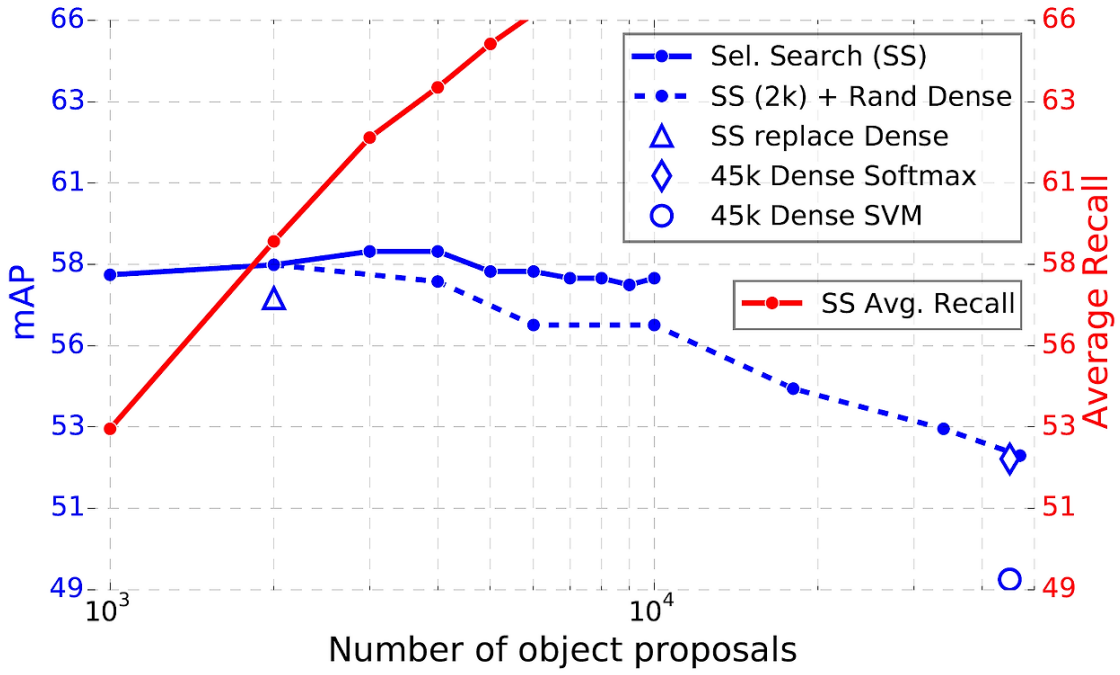
- 단일 스케일과 멀티 스케일로 훈련과 테스트를 한 모델 S와 M의 결과, L 모델은 GPU 메모리 한계로 멀티 스케일 훈련을 하지 못함

- Do SVMs outperform softmax?
  - Fast R-CNN은 SVM 분류기 대신 소프트맥스 분류기를 사용함
    - R-CNN과 SPP-net에서는 SVM 분류기를 사용함

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**

    - Fast R-CNN의 소프트 맥스 분류기 vs SVM 분류기
      - 같은 환경에서 비교하여 훈련 알고리즘 및 하이퍼파라미터는 동일하게 설정
      - 소프트맥스 분류기를 사용한 경우 SVM을 사용한 경우보다 mAP가 더 높음

- Are more proposals always better?



- 후보 영역이 많을수록 mAP는 대개 좋아지지만, 과하게 많으면 오히려 mAP가 떨어짐
- 위 그래프에서 보다시피 AR이 높다고 mAP가 반드시 비례해서 높아지지 않음

- 참고하여 정리한 글

논문 리뷰 - Fast R-CNN 톺아보기

본 글에서 주요 내용 위주로 Fast R-CNN 논문을 번역/정리했습니다. 글 중간에 로 부연 설명을 달아놓기도 했습니다. 틀린 내용이 있으면 피드백 부탁드립니다. 논문 제목: Fast R-CNN 저자: Ross Girshick

https://bkshin.tistory.com/entry/논문-리뷰-Fast-R-CNN-톺아보기