Deep Learning

RCNN 모델 개요

(Regions with CNN features)

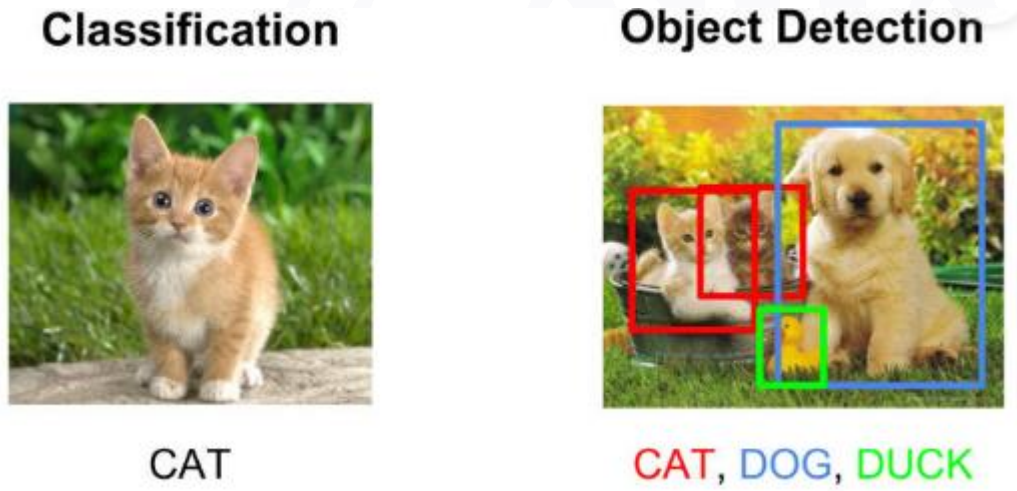
강사 양석환

RCNN 모델

CNN 모델의 등장 이후

- 2012년
  - ILSVRC(ImageNet Large Scale Visual Recognition Challenge) 대회에서 AlexNet 공개
  - CNN은 이미지 분류(Classification) 분야의 표준이 됨
  - 그러나 CNN이 이미지 분류(Classification) 분야에서 엄청난 성적을 거두었음에도 불구하고 Object Detection 분야에 바로 적용되지는 못함

- 이미지 분류(classification)
  - 한 개의 객체(Object)가 그려져 있는 이미지가 있을 때 이 객체가 무엇인지 알아내는 문제
- 물체 인식(Object Detection, 객체 검출)
  - 이미지 내에 관심이 있는 객체의 위치(Region of Interest)를 파악하고 분류하는 문제
  - 입력된 이미지에 물체의 위치를 알려주기 위한 Bounding Box를 그려줘야 하고
  - 다수의 Bounding Box를 다양한 Object 종류에 대하여 찾아줘야 하기 때문에
  - 이미지 분류 보다는 훨씬 복잡한 문제임



RCNN 모델

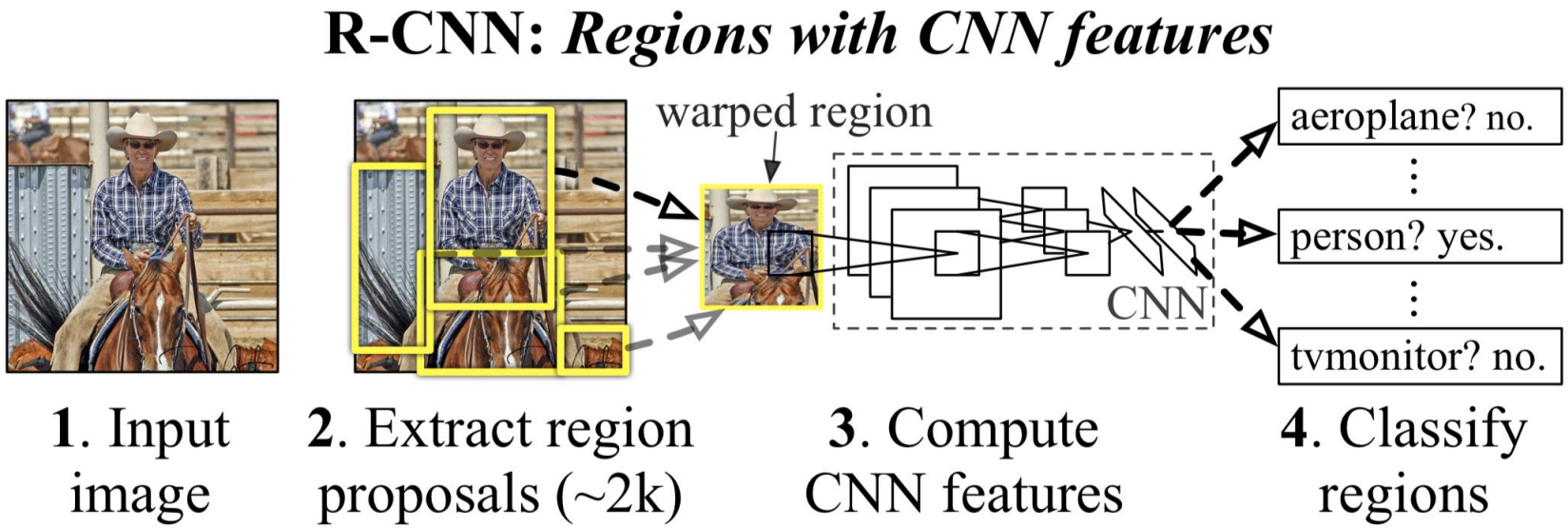
- R-CNN (CVPR 2014)
  - Rich feature hierarchies for accurate object detection and semantic segmentation
    (정확한 객체 감지 및 시맨틱 분할을 위한 풍부한 기능 계층)
  - https://arxiv.org/pdf/1311.2524.pdf
  - CNN을 Object Detection 분야에 최초로 적용
    → CNN을 이용한 검출 방식이 Classification 뿐만 아닌 Object Detection 분야에도 높은 수준의 성능을 이끌어 낼 수 있다는 것을 증명함
  - VOC2012 (Visual Object Classes Challenge)에서 기존 방법보다 30%이상의 성능향상을 보임

RCNN 모델

- **RCNN 모델의 확장**
  - **R-CNN을 수정, 보완하여 성능, 속도 등을 향상시킨 R-CNN 계열의 논문들**
    - R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN...

RCNN 모델의 구조와 프로세스

RCNN 모델의 구조와 처리 방식



1. 이미지를 input으로 집어 넣는다.
2. 2000개의 영역(Bounding Box)을 Selective Search 알고리즘을 통해 추출하여 잘라낸다(Cropping). 이를 CNN모델에 넣기 위해 같은 사이즈(227x227 pixel size)로 찌그러뜨린다(Warping).
3. 2000개의 Warped image를 각각 CNN 모델에 집어 넣는다.
4. 각각 Classification을 진행하여 결과를 도출한다.

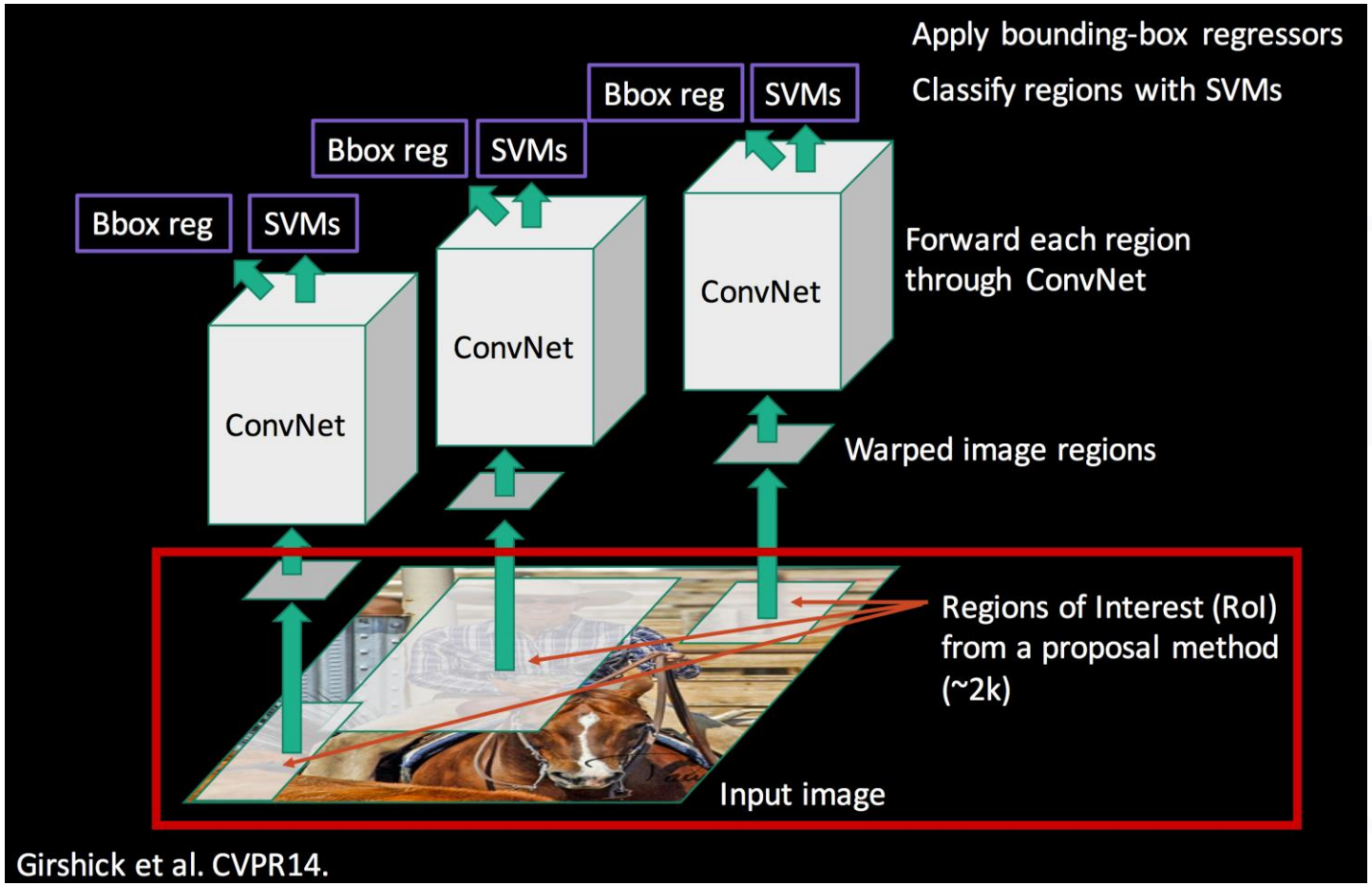
RCNN 모델의 처리 방식

- R-CNN은 2-stage Detector → 전체 Task를 두 가지 단계로 나누어 진행
  - 첫 번째 단계: Region Proposal (물체의 위치를 찾는 일)
  - 두 번째 단계: Region Classification (물체를 분류하는 일)

- RCNN 모델의 처리 단계별 구성요소와 역할
  1. Region Proposal: 카테고리와 무관하게 물체의 영역을 찾는 모듈
  2. CNN: 각각의 영역으로부터 고정된 크기의 Feature Vector를 뽑아내는 Large Convolutional Neural Network
  3. SVM: Classification 을 위한 선형 지도학습 모델 Support Vector Machine(SVM)

Region Proposal (영역 찾기)

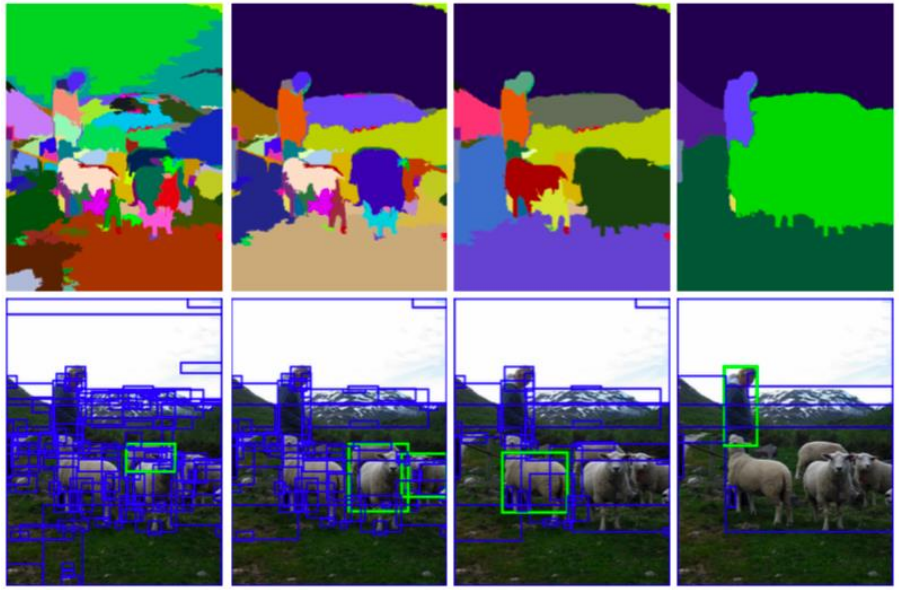
- **Selective Search라는 알고리즘을 이용하여 각 영역 탐색**



Region Proposal (영역 찾기)

- **Selective Search알고리즘**
  - **객체와 주변 간의 색감(Color), 질감(Texture) 차이, 다른 물체에 둘러싸여있는지(Enclosed) 여부 등을 파악해서 다양한 전략으로 물체의 위치를 파악할 수 있도록 하는 알고리즘**
  - **Segmentation 분야에 많이 사용됨**

Bounding box들을 Random 하게 많이 생성을하고 이들을 조금씩 Merge 해나가면서 물체를 인식해나가는 방식
→ 물체의 위치를 파악하기 위한 알고리즘

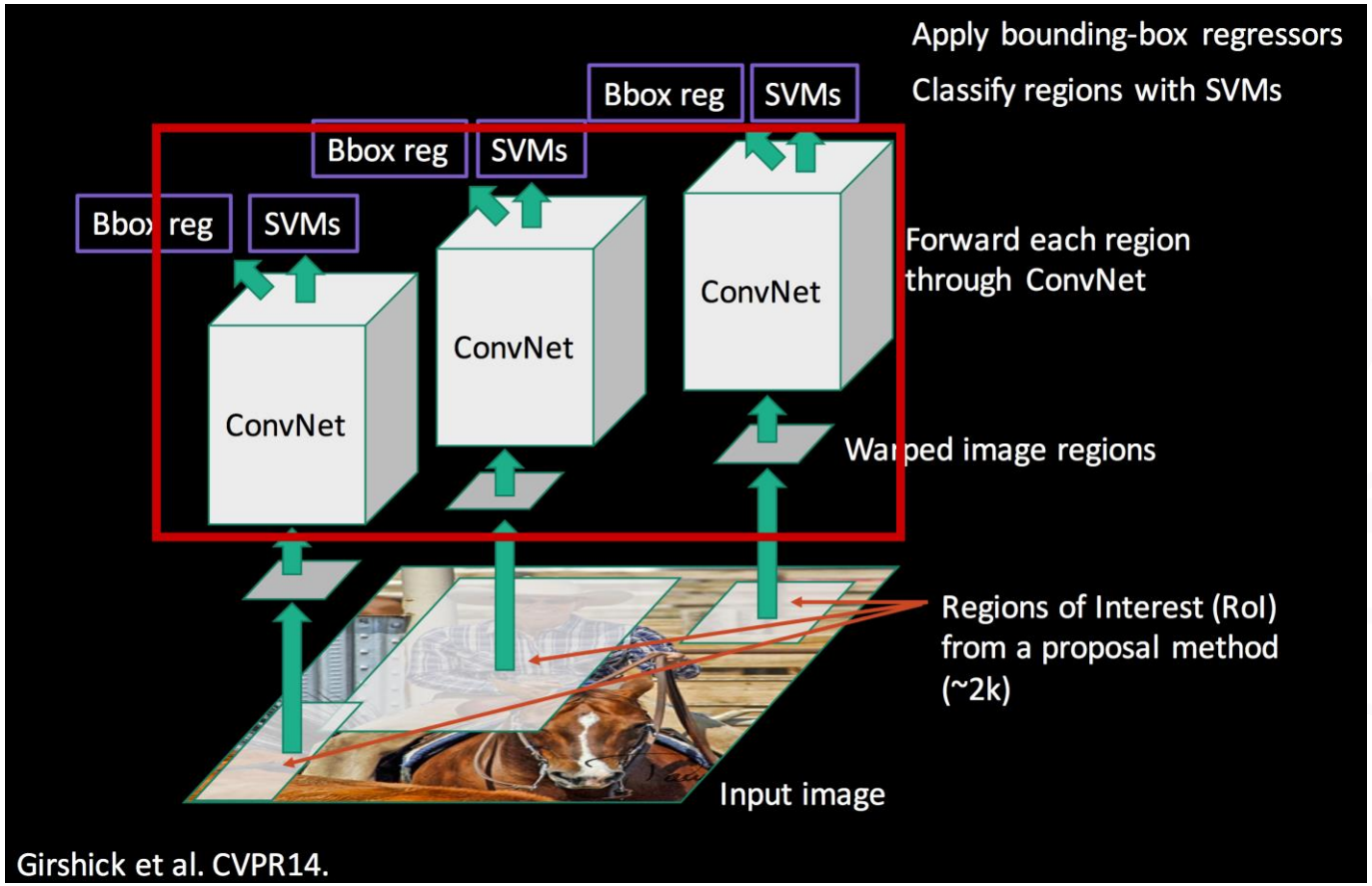


(a)

(b)

CNN 모델

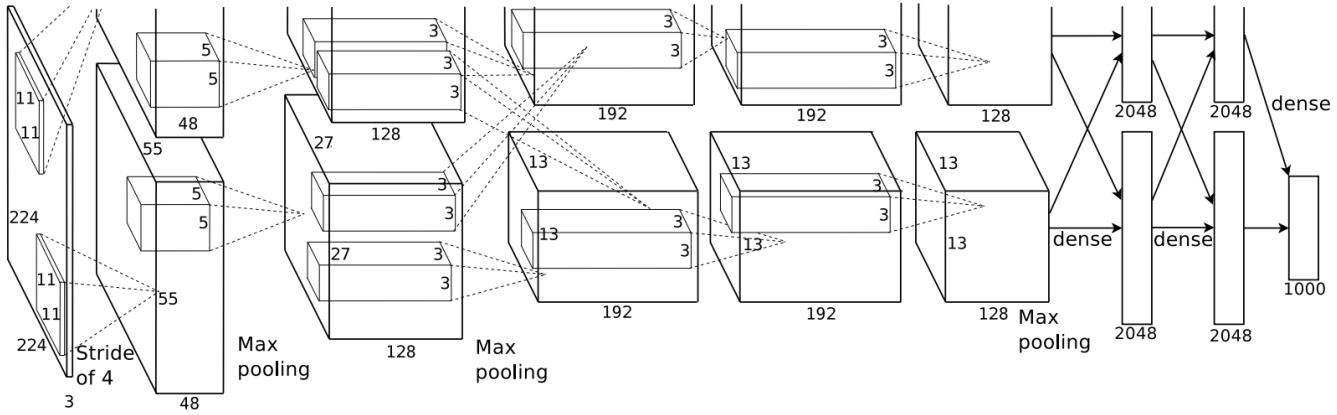
- Selective Search를 통해 생성된 2000개의 224x224 Pixel Size로 Warping된 이미지를 각각 CNN에 적용



224x224 사이즈로 Warping된 VOC2007 train 이미지들

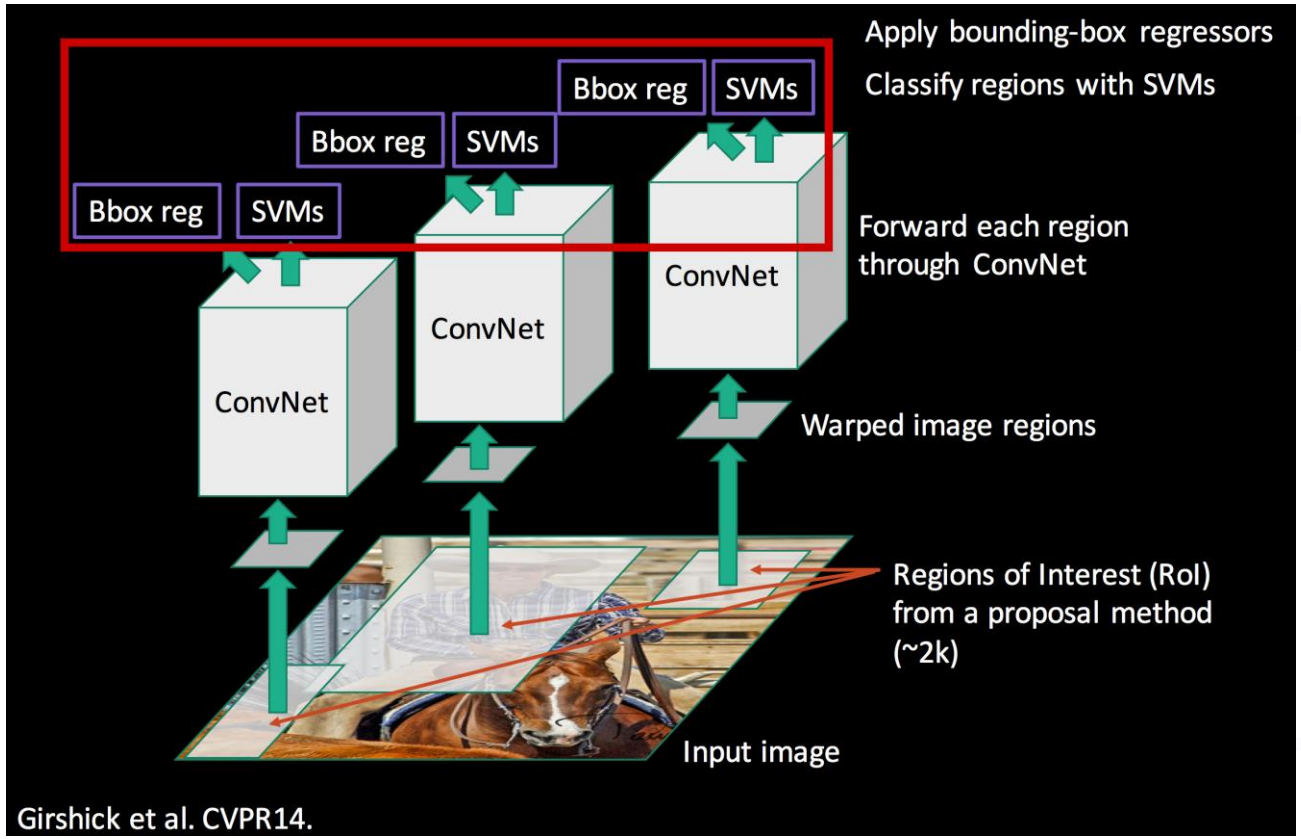
CNN 모델

- **RCNN 모델에서 CNN 모델 부분은**
  - **AlexNet의 구조를 거의 그대로 사용하고 Object Detection 용으로 마지막 부분만 조금 수정**
    - AlexNet Network 마지막 부분을 Detection을 위한 Class 수 만큼 바꾸고
      - (1000 -> PASCAL VOC 기준 20) (1000-> ILSVRC2013 기준 200)
    - Object Detection용 Dataset을 집어 넣어 Fine-Tuning 진행
    - 각각의 region proposal로부터 4096-dimentional feature vector를 뽑아내고,
    - 이를 이용하여 Fixed-length Feature Vector를 만들어냅니다.



SVM (Support Vector Machine)

- CNN 모델로부터 Feature가 추출이 되고 Training Label이 적용되고 나면 Linear SVM을 이용하여 classification을 진행 (Category-Specific Linear SVMs)



SVM (Support Vector Machine)

- **R-CNN에서 Classifier로 Softmax를 쓰지 않고 SVM을 사용하는 이유**
  - **RCNN 논문의 Appendix B에 따르면**
    - VOC2007 데이터셋 기준으로 Softmax를 사용하였을 때 mAP값이 54.2%에서 50.9%로 떨어졌음
  - **논문에서는 CNN을 fine-tuning 할 때**
    - 이미지의 positive/negative examples와 SVM을 학습할 때 이미지의 positive/negative examples를 따로 정의함

SVM (Support Vector Machine)

- **CNN fine-tuning에서는**
  - IoU가 0.5가 넘으면 positive, 이 외에는 "background"라고 labeled
- **SVM을 학습할 때는**
  - ground-truth boxes만 positive example로 설정
  - IoU가 0.3미만인 영역은 모두 negative
  - 나머지는 전부 무시
- **SVM을 CNN fine-tuning과 같은 값을 두고 학습을 했을 때**
  - 성능이 훨씬 좋지 않게 나옴
    - IoU가 0.5에서 1 사이인 영역들(fine-tuning에서 positive으로 정의했던)을 "jittered examples"라고 정의

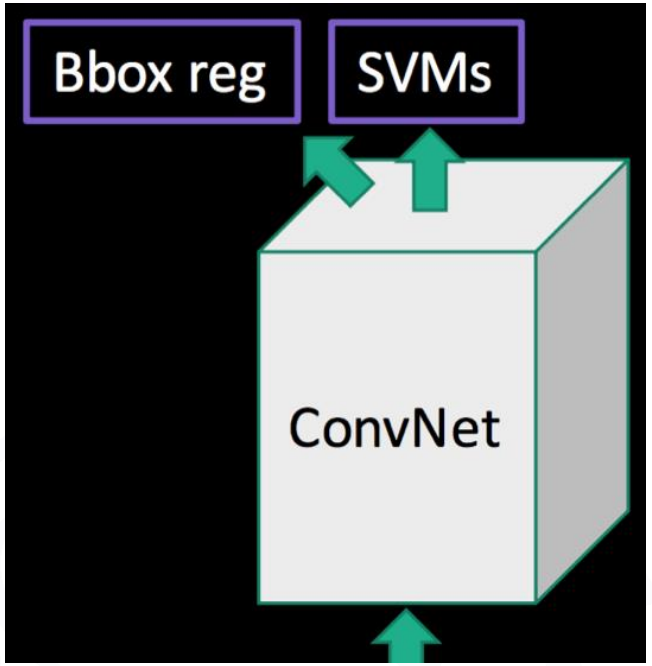
Network의 Overfitting을 피하기 위해선 "Jittered examples" 들이 분명 필요하지만,
시기상으로 fine-tuning 학습 데이터가 많지 않았기 때문에, 이러한 "Jittered examples" 들이 다소 정확하지 않았고,
때문에 바로 Softmax Classifier를 적용시켰을 때 성능이 좋지 않아서 SVM을 학습하는 과정이 필요했던 것으로 추측됨

SVM (Support Vector Machine)

- 결국 SVM(Support Vector Machine)은
  - CNN으로부터 추출된 각각의 Feature Vector들의 점수를 Class별로 매기고,
  - 객체인지 아닌지, 객체라면 어떤 객체인지 등을 판별하는 역할을 하는 Classifier

Bounding Box Regression

- 논문에서 소개했던 전체적인 구조는 앞의 세 가지이지만 제시된 그림에는 bBox reg라고 쓰여진 상자가 추가되어 있음
  - Selective Search로 만들어낸 Bounding Box는 완전히 정확하지는 않기 때문에 물체를 정확히 감싸도록 조정해주는 선형회귀 모델 (Bounding Box Regression)을 추가 적용함



Bounding Box Regression

- bBox의 input값은 N개의 Training Pairs로 구성

$$\{(P^i, G^i)\}_{i=1,\ldots,N}, \text{where } P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$$

  - x, y, w, h는 각각 Bounding Box의 x, y 좌표 (위치), width(너비), height(높이)
  - P는 선택된 Bounding Box이고 G는 Ground Truth(실제 값) Bounding Box
  - 선택된 P를 G에 맞추도록 transform 하는 것을 학습하는 것이 Bounding Box Regression의 목표

$$\hat{G}_x = P_w d_x(P) + P_x \quad (1)$$
$$\hat{G}_y = P_h d_y(P) + P_y \quad (2)$$
$$\hat{G}_w = P_w \exp(d_w(P)) \quad (3)$$
$$\hat{G}_h = P_h \exp(d_h(P)). \quad (4)$$

$$t_x = (G_x - P_x)/P_w \quad (6)$$
$$t_y = (G_y - P_y)/P_h \quad (7)$$
$$t_w = \log(G_w/P_w) \quad (8)$$
$$t_h = \log(G_h/P_h). \quad (9)$$

$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t^i_\star - \hat{\mathbf{w}}^{\mathrm{T}}_\star \boldsymbol{\phi}_5(P^i))^2 + \lambda \|\hat{\mathbf{w}}_\star\|^2. \quad (5)$$

$$d_\star(P) = \mathbf{w}^{\mathrm{T}}_\star \boldsymbol{\phi}_5(P)$$

R-CNN계열의 방법의 Bounding Box Regression에서 모두 이 공식을 사용

Bounding Box Regression

- 수식 설명
  - (1), (2), (3), (4) 식에서 G hat들은 G(Ground Truth)와 최대한 가까워질 변수
  - (1), (2), (3), (4) 식에서 G hat을 G(Ground Truth)로 바꾸고 d를 t로 치환하여 t에 대해 나타낸 것이 (6), (7), (8), (9) 식
  - (6), (7), (8), (9) 는 실제 값
  - (5)식에서 시그마(Sigma) 안에 있는 식이 Loss Function
  - t와 d의 차이인 Loss를 줄여 나가는 방향으로 학습하는 것이 Bounding Box Regression 수식의 목표
  - 뒤에 더해지는 람다식은 Regularization 이고, 논문에서는 이 값이 중요하다고 기재
    - 논문에서는 validation set을 기반으로 람다값을 1000으로 지정

Bounding Box Regression

- R-CNN Class별 성능 및 mAP

VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 [20]†	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA [39]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets [41]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM [18]†	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN BB	**71.8**	**65.8**	**53.0**	**36.8**	**35.9**	**59.7**	**60.0**	**69.9**	**27.9**	**50.6**	**41.4**	**70.0**	**62.0**	**69.0**	**58.1**	**29.5**	**59.4**	**39.3**	**61.2**	**52.4**	**53.7**

- R-CNN BB라고 기재되어있는 맨 아래 행은 Bounding Box Regression을 적용한 경우
- Bounding Box Regression을 적용시켰을 때 성능이 더 향상 되는 것을 확인할 수 있음
- 또 VOC 2010 데이터셋 기준으로 이전 방법들 보다 뛰어난 성능을 보임

R-CNN의 단점

- 처리 시간이 길다
  - Selective Search에서 뽑아낸 2000개의 영역 이미지들에 대해서 모두 CNN모델을 적용
  - Region Proposal에 사용되는 Selective Search가 CPU를 사용하는 알고리즘
    - RCNN의 수행 시간
      - Training Time: 약 84시간(GPU K40 사용 기준으로 frame당 13초, CPU를 사용하였을 때 frame당 53초 소요)
- 복잡하다
  - Multi-Stage Training을 수행하며, CNN, SVM, 그리고 Bounding Box Regression까지 총 세 가지의 모델을 필요로 하는 복잡한 구조
- Back Propagation이 안된다
  - SVM, Bounding Box Regression에서 학습한 결과가 CNN을 업데이트 시키지 못함

RCNN 모델 정리

- 앞에서 소개한 단점들이 존재하지만
  - R-CNN은 최초로 Object Detection에 Deep Learning 방법인 CNN을 적용
  - 이후 2-stage detector들의 구조에 막대한 영향을 미침