Object Detection의 흐름

2025.05.13

발표자 : 강민성

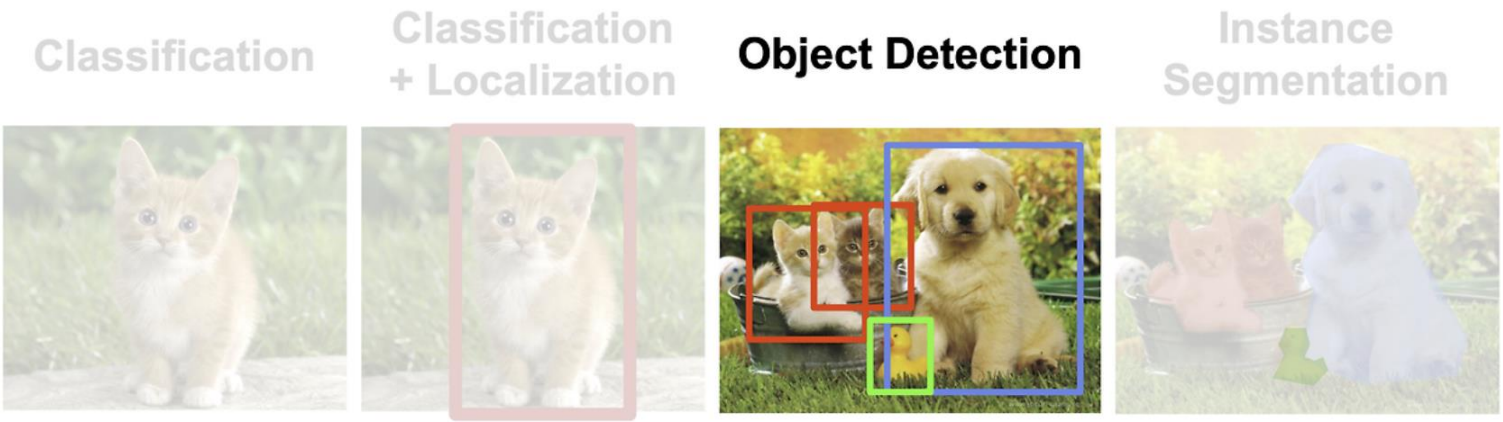
목차

Object detection의 기본 개념

1. Object detection의 정의

이미지 또는 영상에서 객체의 위치(Bounding Box)와 Label을 예측하는 문제

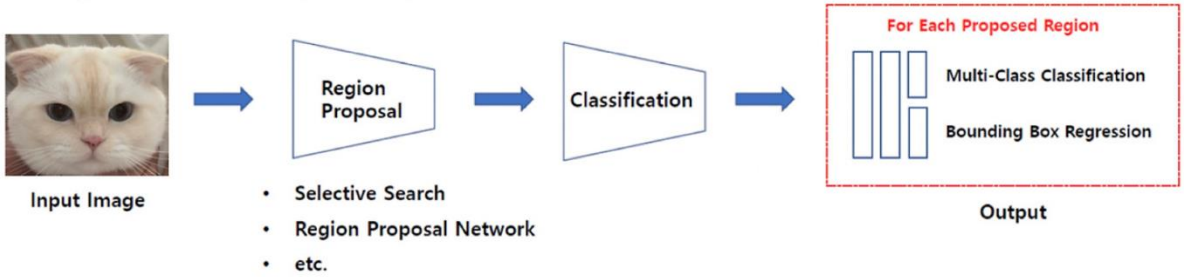
Classification + Localization 수행



Object detection의 기본 개념

1. Two stage object detector vs One stage object detector

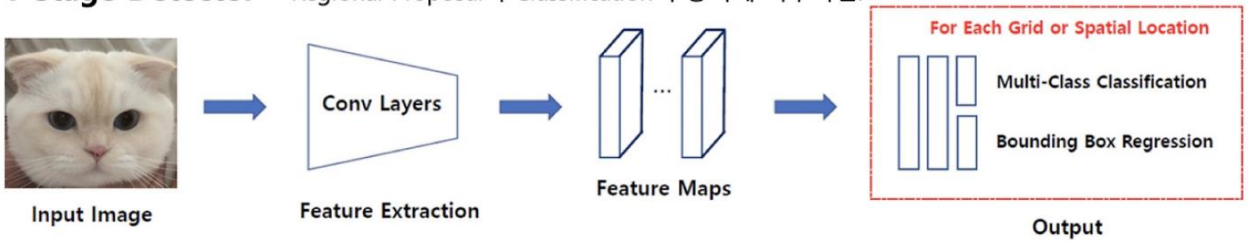
2-Stage Detector - Regional Proposal와 Classification이 순차적으로 이루어짐.



- Selective Search
- Region Proposal Network
- etc.

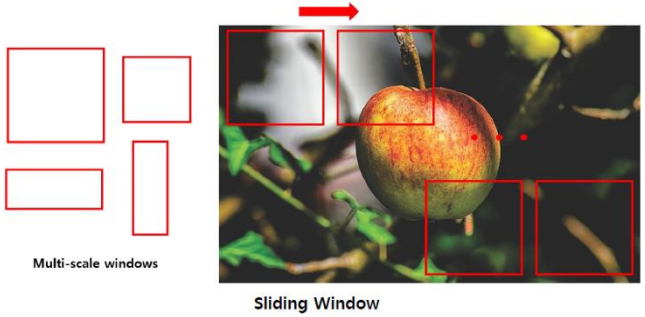
Ex) **R-CNN 계열** (R-CNN, Fast R-CNN, Faster R-CNN, R-FCN, Mask R-CNN ...)

1-Stage Detector - Regional Proposal와 Classification이 동시에 이루어짐.

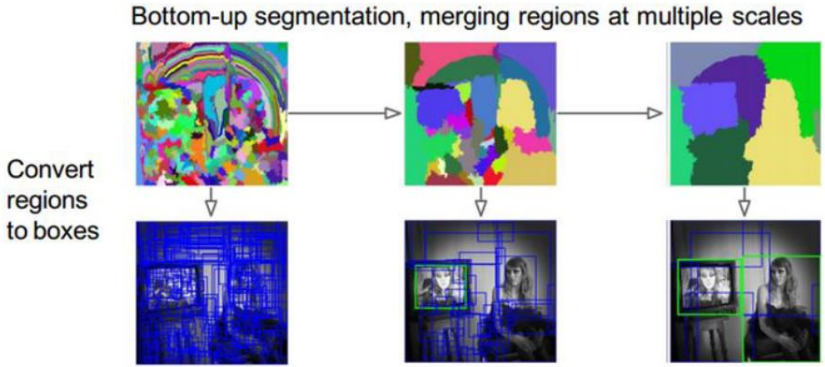


Ex) **YOLO 계열** (YOLO v1, v2, v3) , **SSD 계열** (SSD, DSSD, DSOD, RetinaNet, RefineDet ...)

Sliding Window & Selective Search(SS)

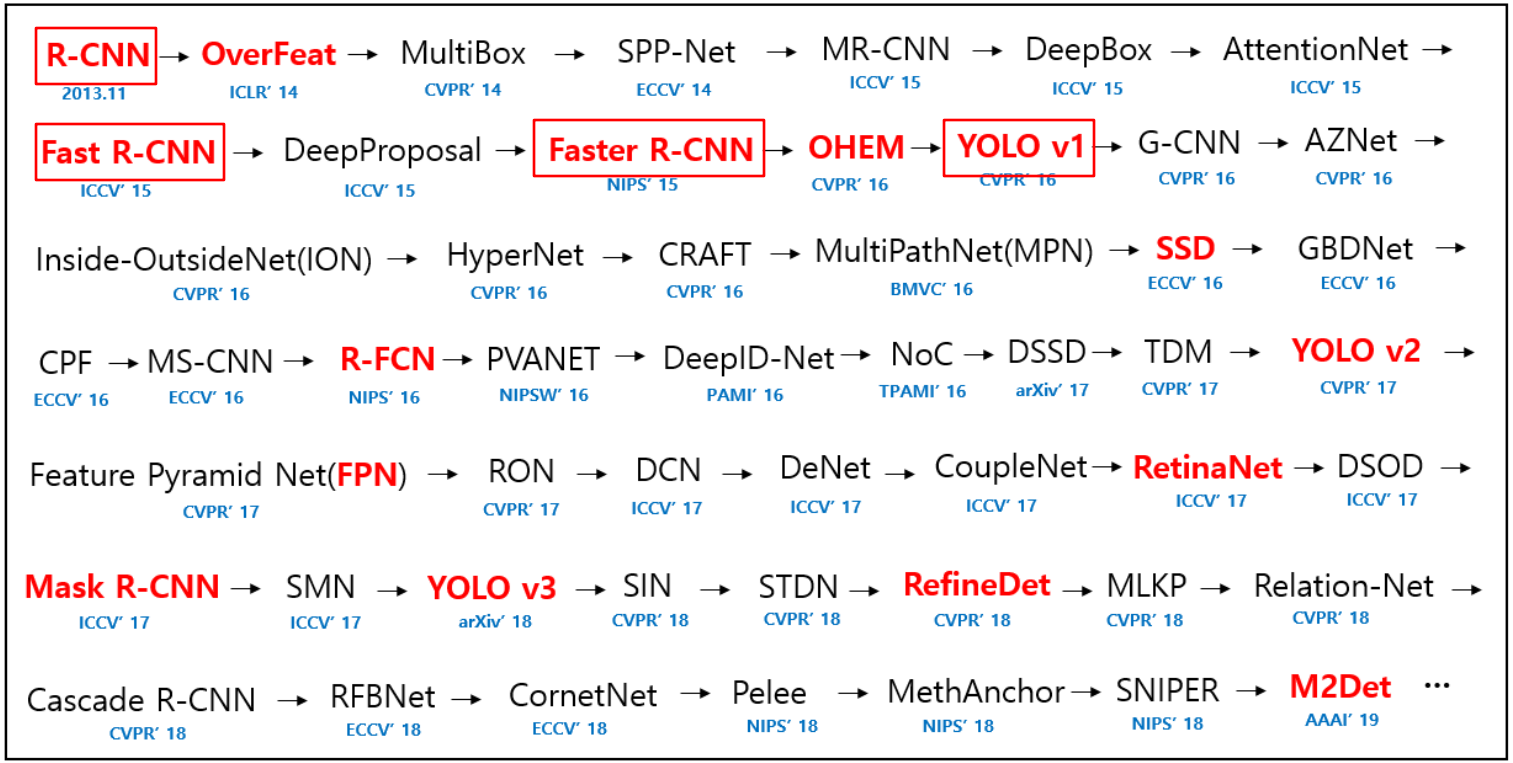


Window를 이미지의 왼쪽 위부터 오른쪽 아래까지 sliding하며 score를 계산하는 방법



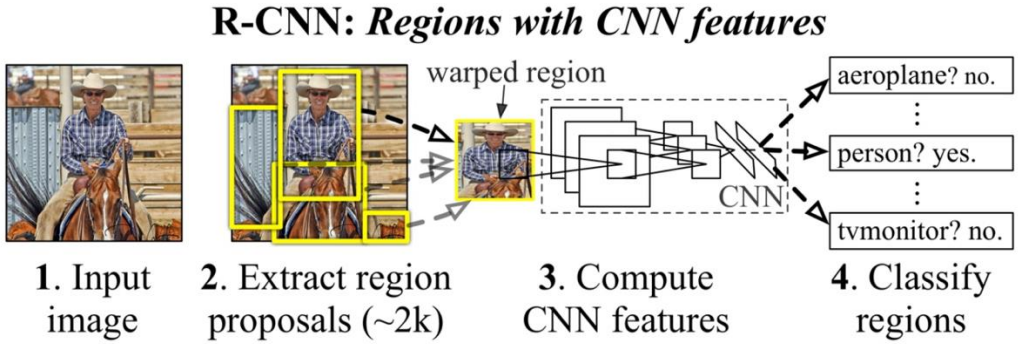
계층적 구조를 활용하여 영역을 탐색하고 그룹화하는 과정을 반복하며 객체의 위치를 proposal

Object detection flows



R-CNN (2-stage detector)

Regions with CNN features (CNN과 Region proposals의 결합)



1.Input Image

2.약 2000개의 bottom-up region proposals를 추출

3.CNN을 이용해 각각 region proposal의 특징을 계산

4.linear SVMs를 이용해 각각의 label을 분류

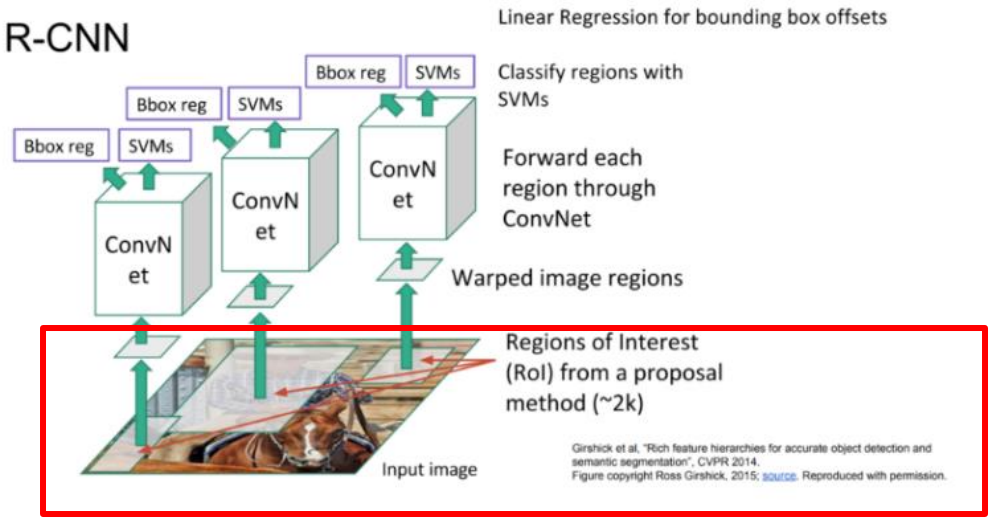
Three Modules for R-CNN

1.Region Proposals: detector가 이용가능한 영역 후보의 집합

2.CNN: 각각의 region에 대한 고정된 크기의 feature vector를 추출

3.Linear SVMs: 분류를 진행

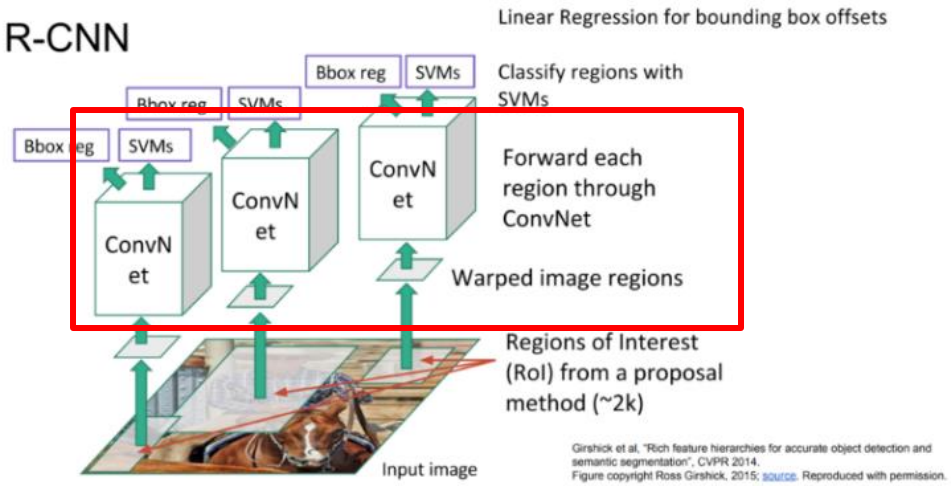
Region Proposals



1. category-independent region proposals을 위해 **selective search algorithm**을 사용하여 각 이미지에서 2000장의 후보 영역을 찾음

2.Warped image regions 과정에서 227x227로 사이즈를 통합 (for CNN architecture)

CNN

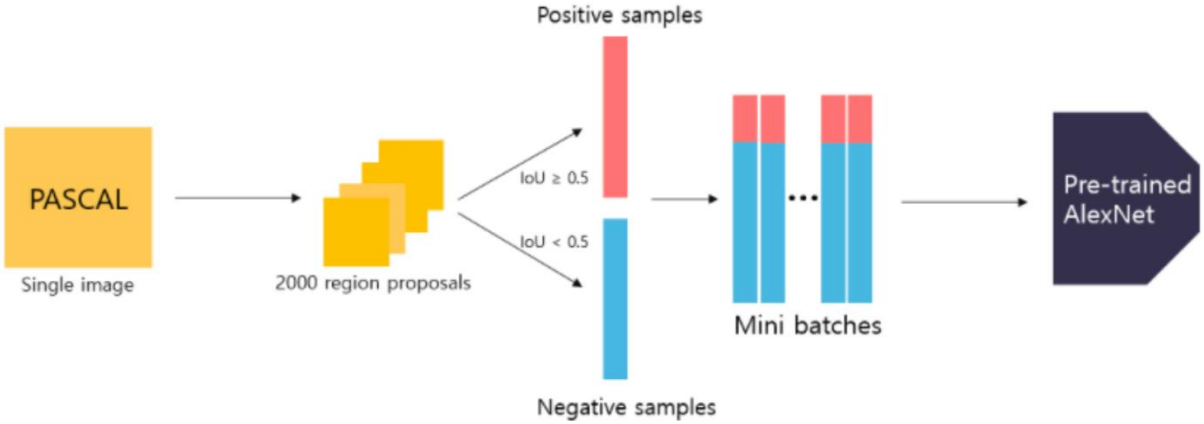


1. classification dataset을 이용해 pre-trained된 AlexNet 구조 사용

2.Domain-specific-fine-tuning을 통해 CNN을 다시 학습

=> Region proposals 2000개의 이미지를 입력받아 4096-dimensional feature vector 추출

CNN – Domain specific fine-tuning



1. 2000개의 Region Proposal과 Ground Truth의 IoU를 비교하여

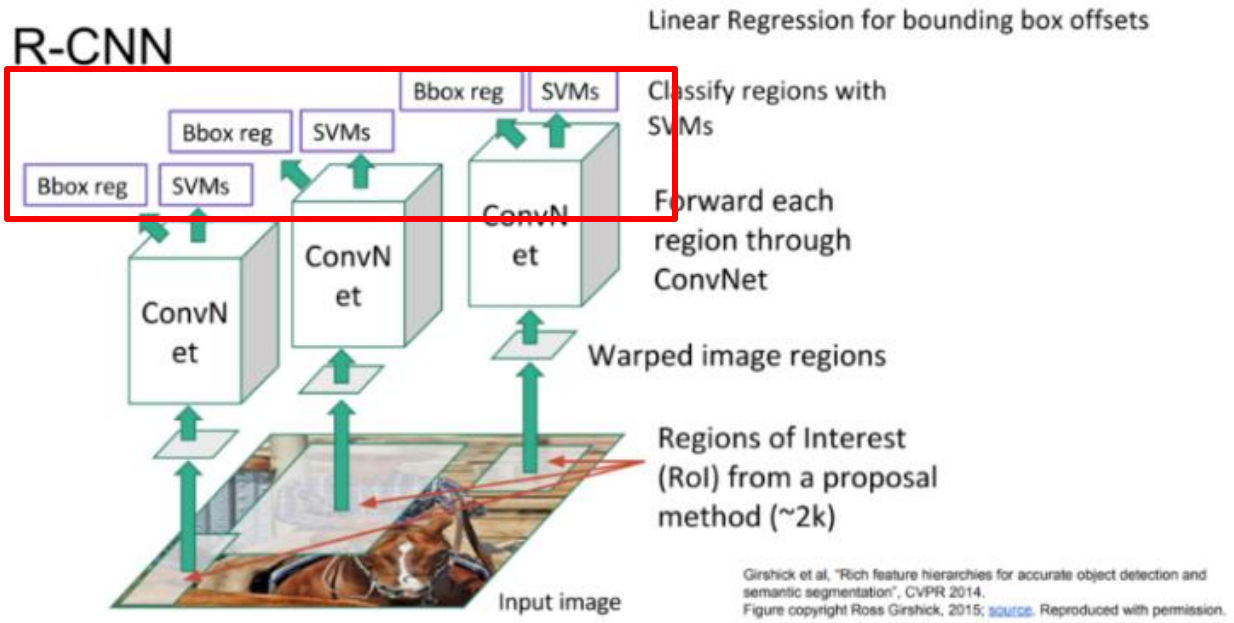
   IoU ≥ 0.5: **Positive sample**, IoU < 0.5: **Negative sample**

2. Positive 32개 + Negative 96개로 구성된 **Mini-batch (총 128)**

3.Pre-trained **AlexNet**의 마지막 Softmax layer를 N+1 way로 수정하여 fine-tuning

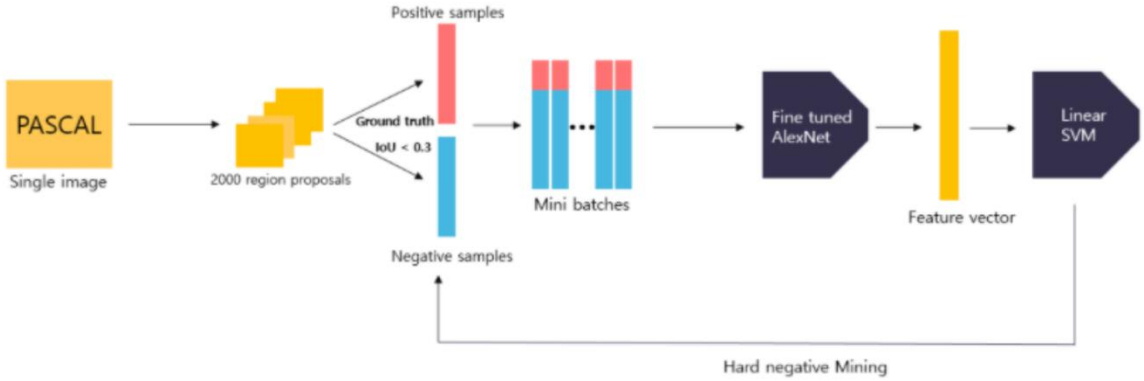
4. 학습 후에는 Softmax 제거 → CNN은 **4096-D feature vector 추출용**으로 사용

Linear SVMs



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

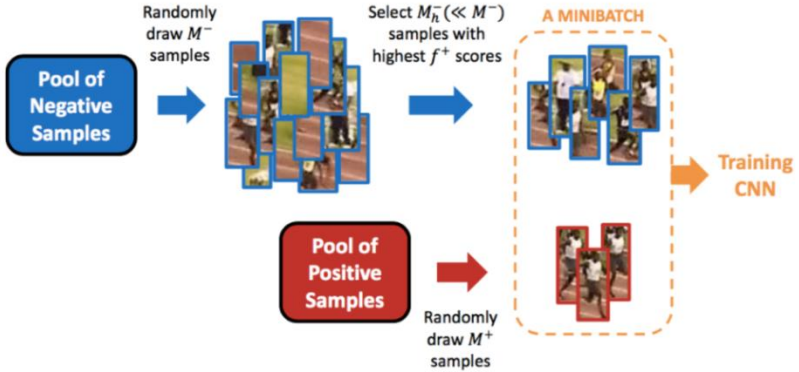
Linear SVMs



1. 학습 시 **IoU ≥ 0.5** → **Positive**, **IoU ≤ 0.3** → **Negative**, 그 외는 **제외**
2. Positive 32 + Negative 96 = **128개 Mini-batch 구성**
3. 추출된 **4096-D feature**를 바탕으로 각 클래스별 **Linear SVM 학습**
4. SVM은 **Hard Negative Mining** 기법을 통해 성능 향상
5. Softmax 대신 SVM 사용 → 더 정밀한 Positive 정의 + False Positive 억제
6. 결과: **클래스 + Confidence Score** 반환

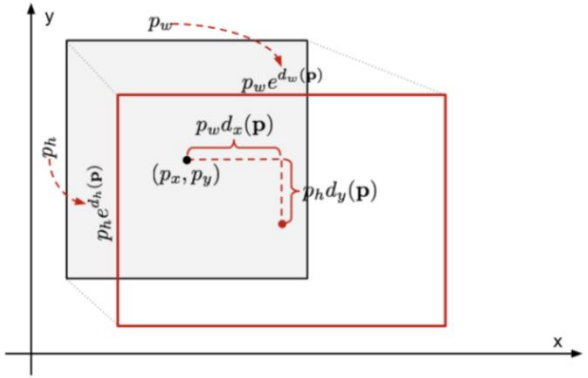
Hard negative mining & Bounding Box Regressor

Hard negative mining

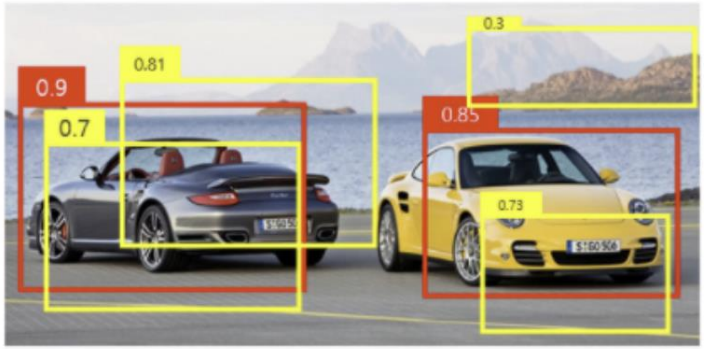


Bounding Box Regressor

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



Non maximum Supression



Before Non Maximum Suppression

After Non Maximum Suppression

1. bounding box별로 지정한 confidence scroe threshold 이하의 box를 제거

2. 남은 bounding box를 confidence score에 따라 내림차순으로 정렬한다. 그 다음 confidence score가 높은 순의 bounding box부터 다른 box와의 IoU값을 조사하여 IoU threshold 이상인 box를 모두 제거

1.2의 과정을 반복하여 남아있는 box만 선택

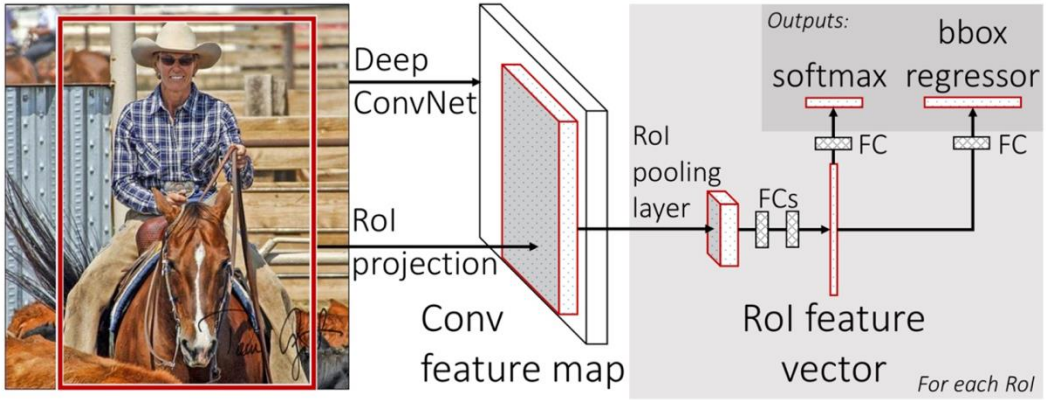
R-CNN Weakness

1.이미지 한 장당 2000개의 **region proposal을 추출하므로** 학습 및 추론의 속도가 느리다.

2.3가지 모델을 사용 하다 보니 구조와 학습 과정이 복잡하다.

3.end-to-end 학습을 수행할 수 없다.

Fast R-CNN (2-stage detector)

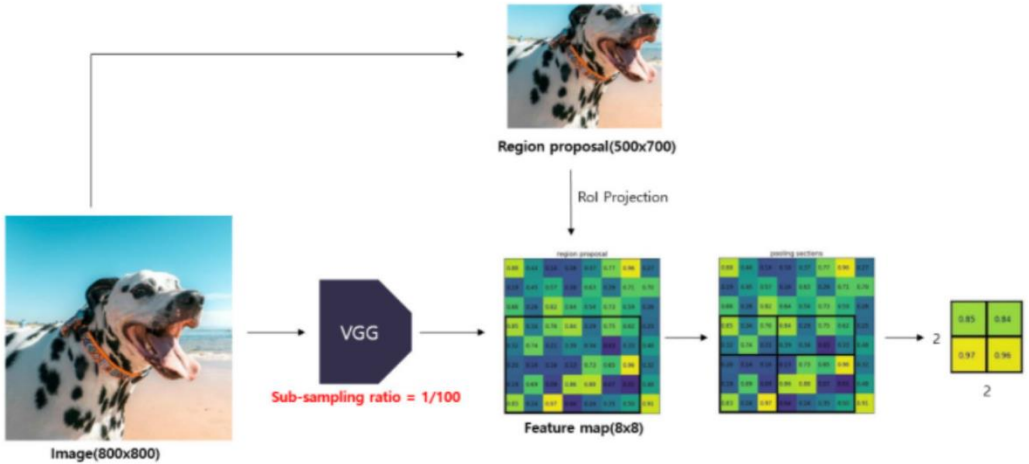


1. input image와 multiple regions of interset(RoIs)가 입력으로 사용

2.각각의 RoI는 ConvNet 연산을 통해 고정된 크기의 feature map으로 pooling되고, FCs을 통해 feature vector로 매핑(mapping)된다.

3.RoI별 두 개의 output을 갖는다. 하나는 softmax probabilities이고, 다른 하나는 per-class bounding-box regression offsets

RoI pooling layer

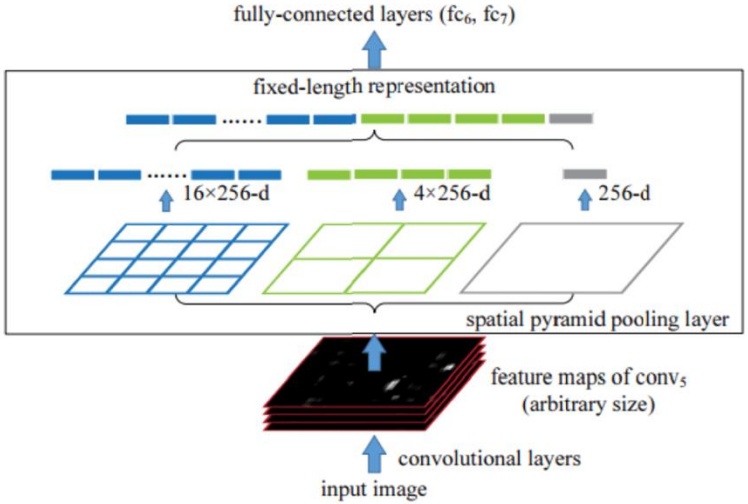


전체 이미지를 CNN(VGG)으로 한 번만 연산하여 **Feature Map**을 생성한 뒤,
→ 각 Region Proposal을 **RoI Projection**을 통해 feature map 상에 매핑

RoI projection까진 **spatial pyramid pooling**과 동일,

차이점: 하나의 level 사용. feature map을 7x7로 분할

Spatial pyramid pooling



입력 이미지에서 추출된 **feature map**에 대해,
여러 수준의 크기로 분할하여 **다단계 max-pooling**을 수행
→ 각 레벨에서 고정된 개수의 벡터를 얻고 **Concat**하여 고정된 길이의 feature vector 생성

- 1×1 → 1×256
- 2×2 → 4×256
- 4×4 → 16×256 →총 21×256 크기의 벡터 생성 (3개 레벨 concat)

장점:

- 입력 이미지나 RoI의 크기가 달라도 **고정된 길이의 표현 가능 오버피팅 방지**

Fine-tuning for detection

Fast R-CNN : 역전파를 통해 네트워크의 모든 가중치를 학습

1. 효율적인 학습을 위해 **hierarchical sampling**

+

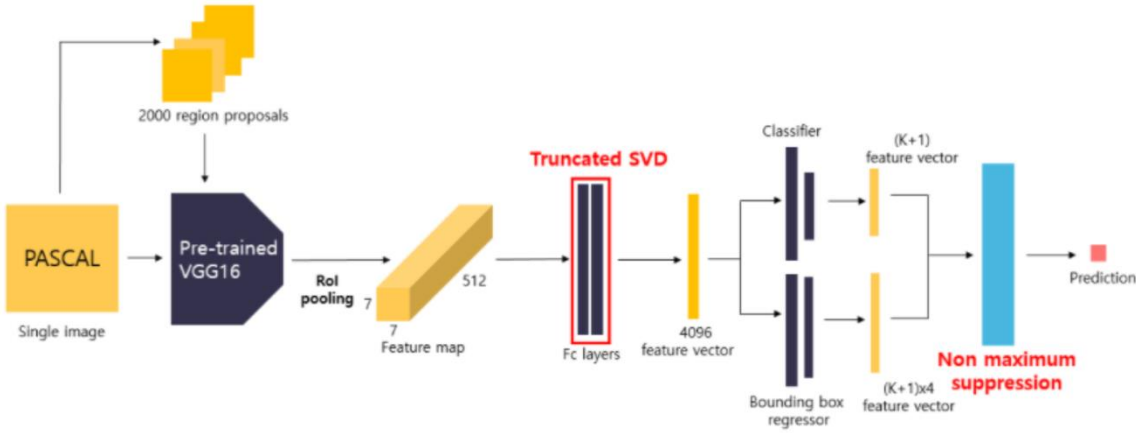
2. Multi-task loss -> softmax classifier 와 bounding box regressors를 **한번의 fine-tuning**으로 학습 $L(p, t, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$

Hierarchical sampling

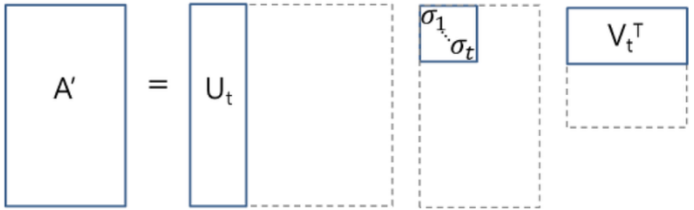
. SGD mini-batch를 구성할 때 N개의 이미지를 sampling하고, 총 R개의 region proposal을 사용한다고 할 때, 각 이미지로부터 R/N개의 region proposals를 sampling하는 방법

이를 통해 같은 이미지에서 추출된 region proposals끼리는 forward, backward propogation 시, **연산과 메모리를 공유**

Fast R-CNN detection



1.Truncated SVD를 이용해 fully connected layers의 연산 속도↑



Truncated SVD

2.Non maximum suppression으로 최적의 bounding box만 출력

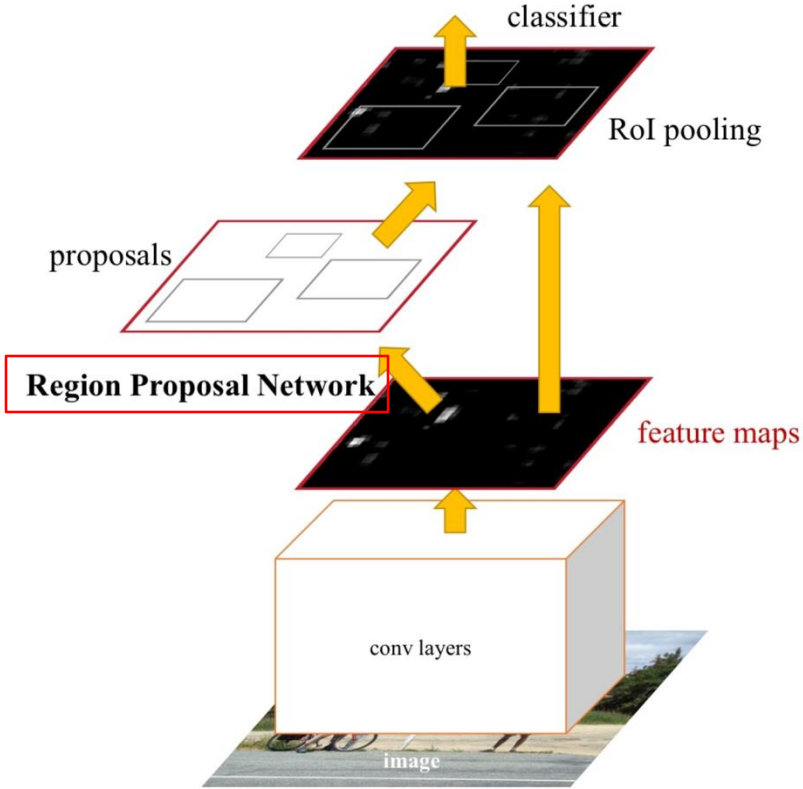
Faster R-CNN

- **기존 한계**
  - Fast R-CNN은 여전히 **Selective Search** 사용
  - Region Proposal 단계가 전체 시스템의 **속도 병목**

- **RPN 도입**
  - Region Proposal Network(RPN)는
    Pre-trained CNN feature map 위에 얕은 Conv layers를 추가해
    각 위치의 **Objectness Score + Bounding Box 좌표**를 동시에 예측
  - RPN은 **Fully Convolutional Network (FCN)** 구조로 동작
  - 다양한 크기 대응 위해 **Anchor Boxes** 사용

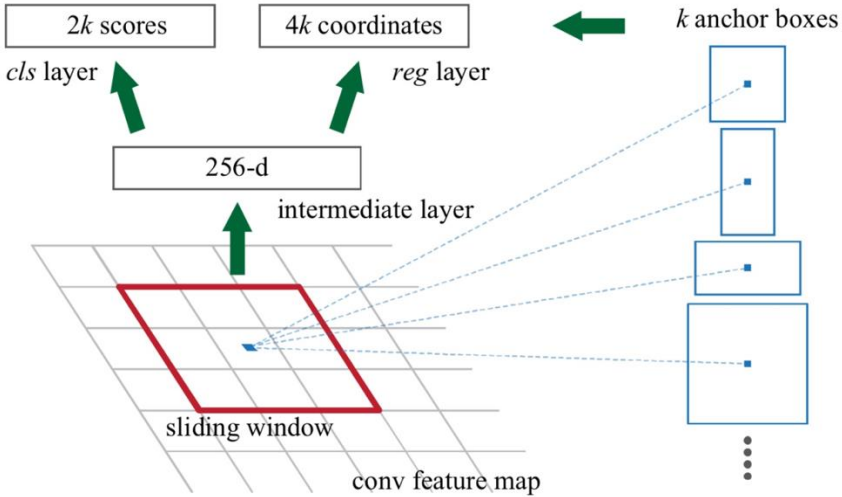
- **통합 네트워크 구조**
  - **RPN과 Fast R-CNN이 Convolutional Features를 공유**
  - 이를 통해 두 모듈을 **하나의 통합 네트워크**로 결합
  - 전체 구조가 **End-to-End 학습 가능**

Faster R-CNN


classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image

Region Proposal Network (RPN)

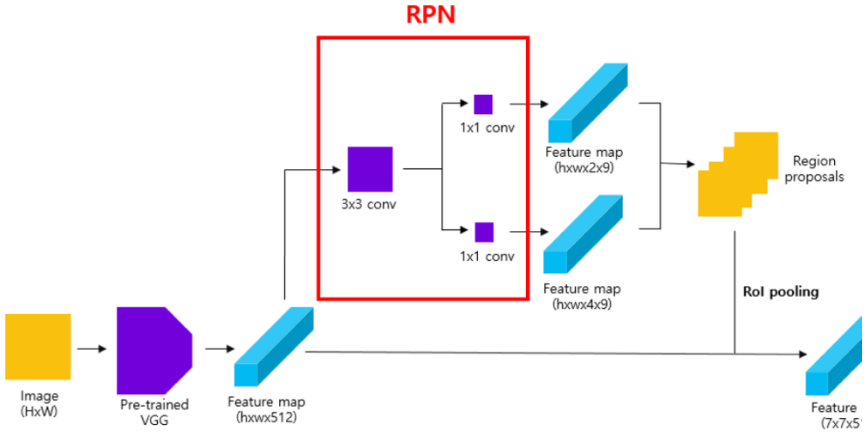
anchor



scale([128, 256, 512]),
aspect ratio([1:1, 1:2, 2:1])
=> 총 9개의 서로 다른 anchor box를 사전에 정의

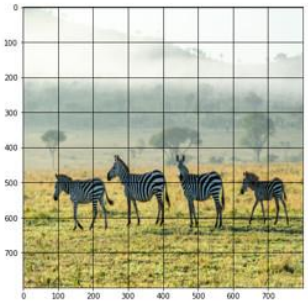
1. 기존 **sliding window**는 고정 크기의 윈도우로 객체 크기 다양성 반영에 한계
2. **Anchor box**는 한 위치에서 여러 scale과 aspect ratio(비율)를 동시에 고려
3. Faster R-CNN은 각 위치당 **9개의 anchor**(3 scales × 3 ratios)를 사용하여 다양한 객체 크기에 대응

Region Proposal Network (RPN)

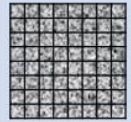


1) 원본 이미지를 pre-trained된 VGG 모델에 입력하여 feature map을 구함
2) 위에서 얻은 feature map에 대하여 3x3 conv 연산을 적용합니다. 이때 feature map의 크기가 유지될 수 있도록 padding을 추가
3) RPN은 각 위치의 9개 anchor에 대해 **객체 유무만 판단**, class score 출력을 위한 1×1 conv의 **채널 수는 2(Object 여부)×9(anchor box)**로 설정
4) bounding box regressor를 얻기 위해 feature map에 대하여 1x1 conv 연산을 적용
5) **4(bounding box regressor)x9(anchor box 9개)**

Region Proposal Network (RPN)



	object O	object X
Anchor type1		
...	...	...
Anchor type 9		

	t_x	t_y	t_w	t_h
Anchor type 1				
...	...	...	...	...
Anchor type 9				

좌측 Feature Map:
각 anchor box에 대해 **객체 포함 여부**(Object / Not Object)를 판단 (2×9 채널)

우측 Feature Map:
각 anchor box에 대해 **bounding box 조정값**(t_x, t_y, t_w, t_h)을 예측 (4×9 채널)

예측된 class score에 따라 **상위 N개 proposal**을 선택한 후,
Non-Maximum Suppression(NMS) 적용하여 최적의 region proposals만 Fast R-CNN에 전달

YOLO (1-stage detector)

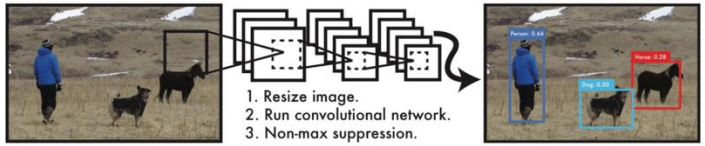
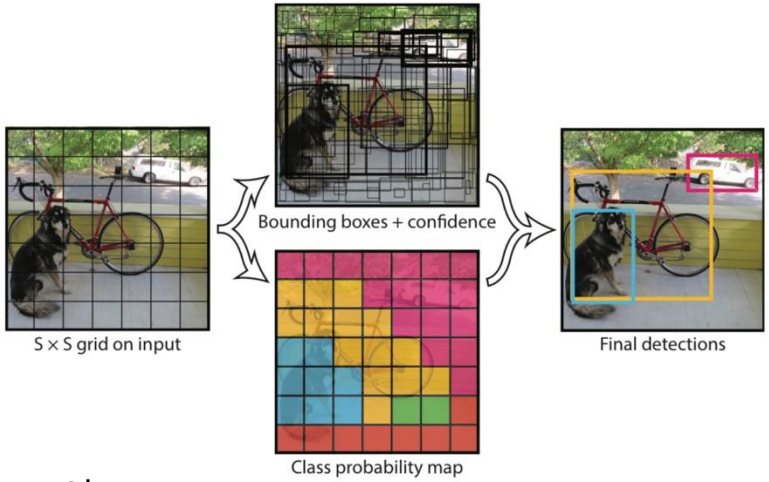


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

- YOLO는 기존 R-CNN 계열과 달리,
  하나의 CNN 네트워크로 위치 + 클래스 정보를 동시에 예측하는 **One-Stage Detector**
  End-to-End 학습 가능

- **빠른 속도**
  - 기존 real-time 모델 대비 2배 높은 mAP
- **전역적 추론(Global reasoning)**
  - Sliding window나 Region Proposal 없이 이미지 전체를 기반으로 추론 → background error
- **우수한 일반화 성능**
  - Natural image로 학습 후 **artwork**에도 잘 작동
  - **객체의 본질적인 표현(Representation)**을 잘 학습

YOLO (1-stage detector)



1. 이미지를 **S×S grid**로 나눈 뒤, 각 grid cell이
   B개의 bounding box + confidence + C개의 클래스 확률을 예측

2. 각 box는 **(x, y, w, h, confidence)** 형태

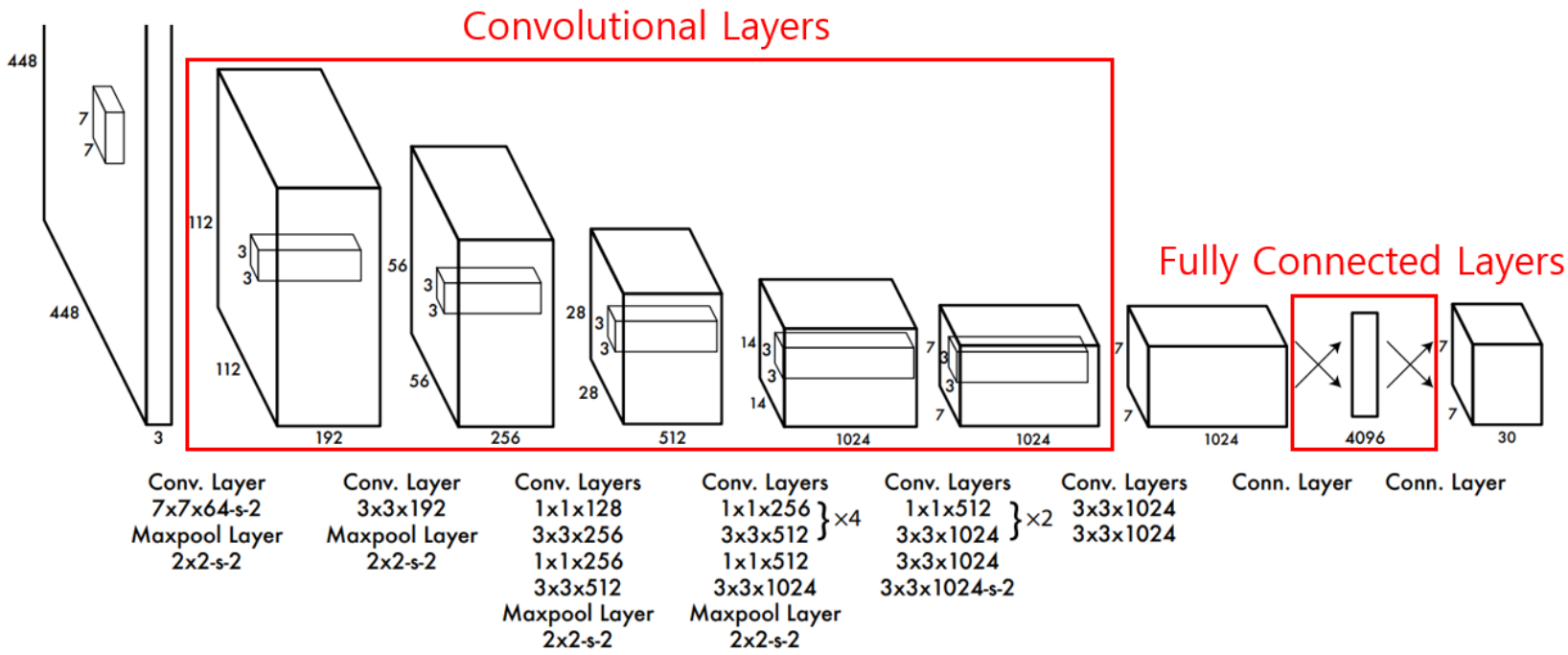
$$confidence = Pr(object) * IOU(pred, truth)$$

3. 각 grid cell은 **1개의 class만 예측**,**class-specific confidence 계산**

$$Pr(Class_i|object) * Pr(object) * IoU(pred, truth) = Pr(class_i) * IoU(pred, truth)$$

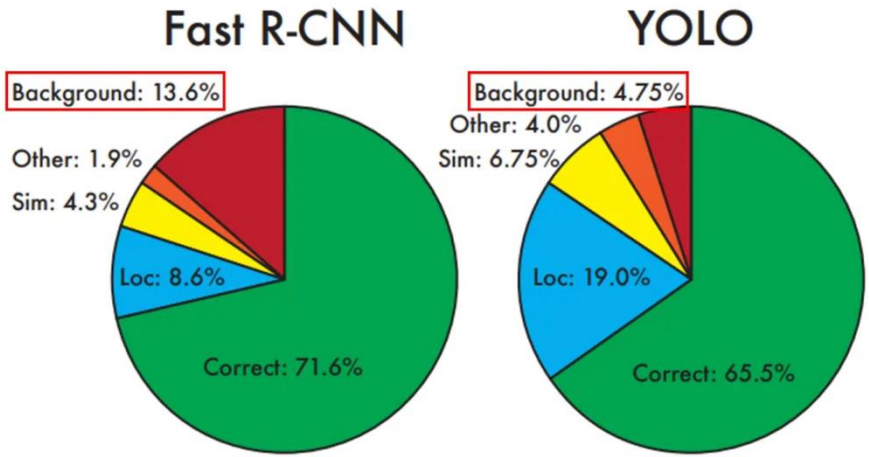
4. 최종 출력은**S × S × (B×5 + C)**
   → 논문에서는 S=7, B=2, C=20 → **7×7×30 tensor**

Architecture



Experiments

Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21



감사합니다.