



ELSEVIER


Contents lists available at [ScienceDirect](#)

Franklin Open

journal homepage: [www.elsevier.com/locate/fraope](http://www.elsevier.com/locate/fraope)



# Architecture review: Two-stage and one-stage object detection

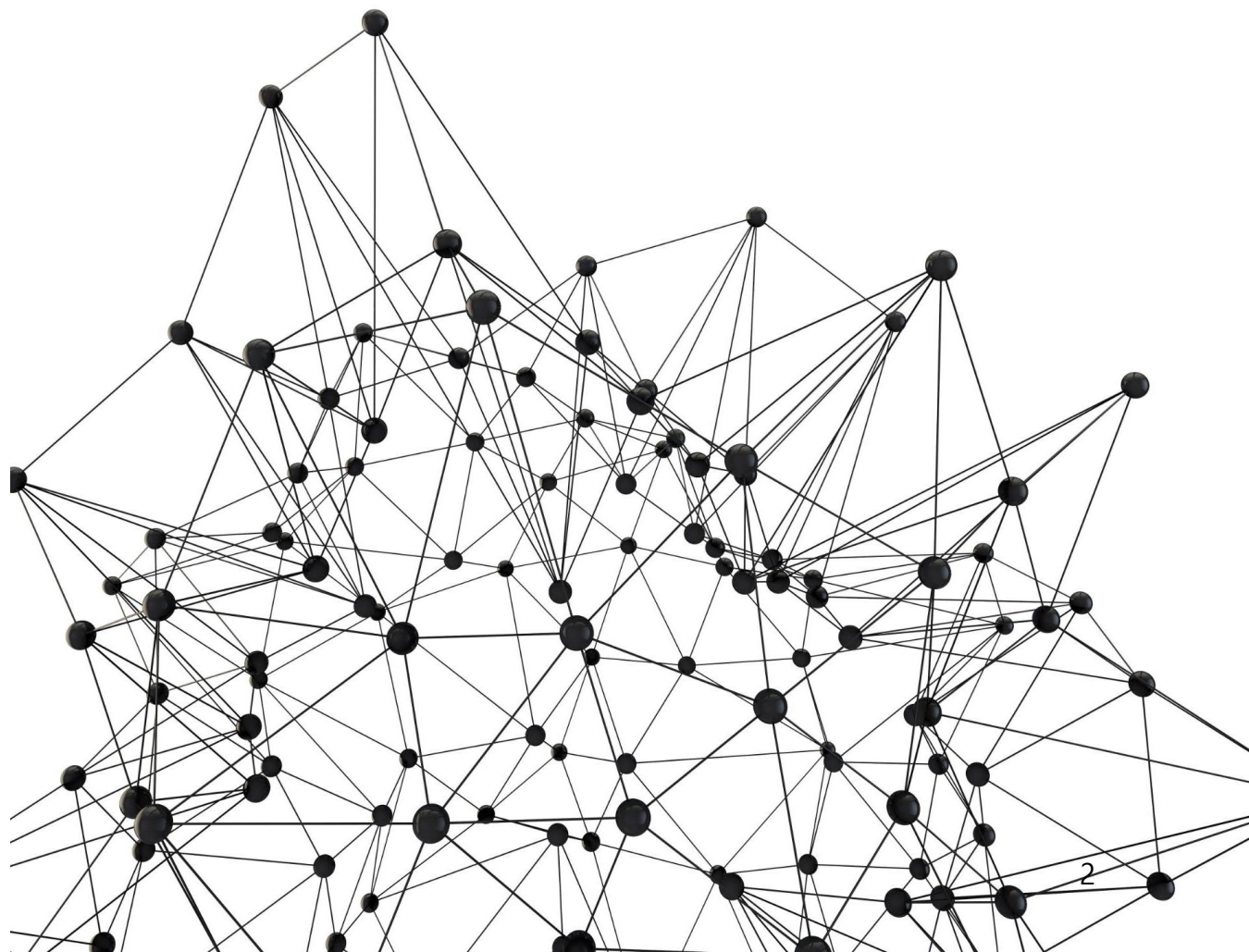
Samiyaa Yaseen Mohammed 

*Mustansiriyah University, College of Engineering, Department of Highway and Transportation Engineering, Baghdad, Iraq*

STC | 이상엽, 배정윤, 이상진, 김준희

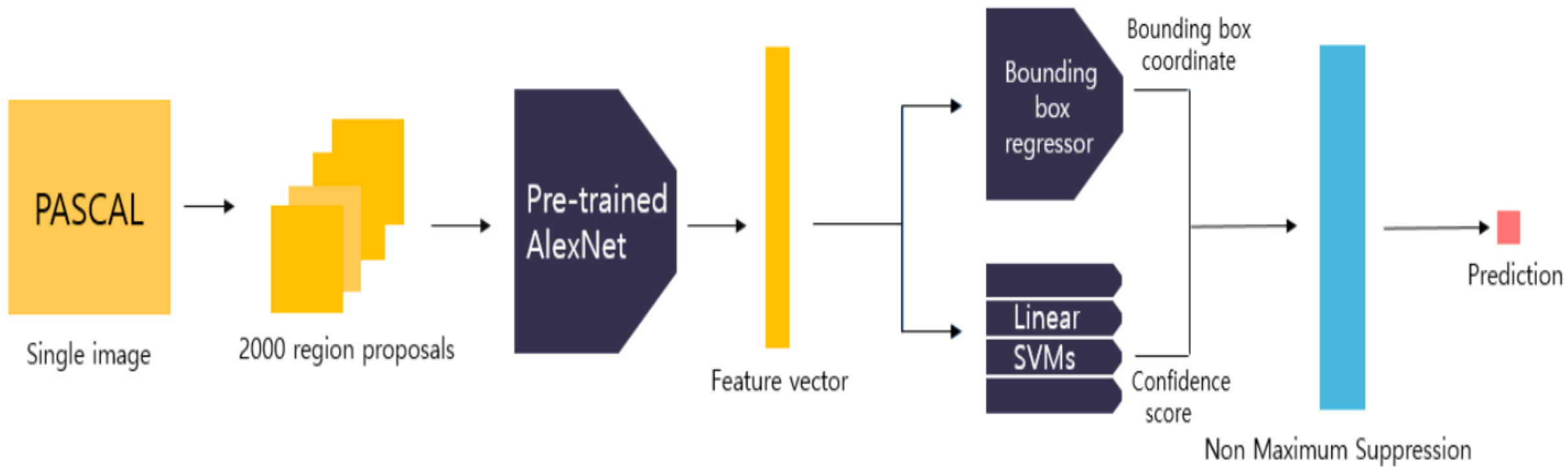
# 목차

1. R-CNN
2. Fast R-CNN과 Faster R-CNN
3. Mask R-CNN
4. 구현

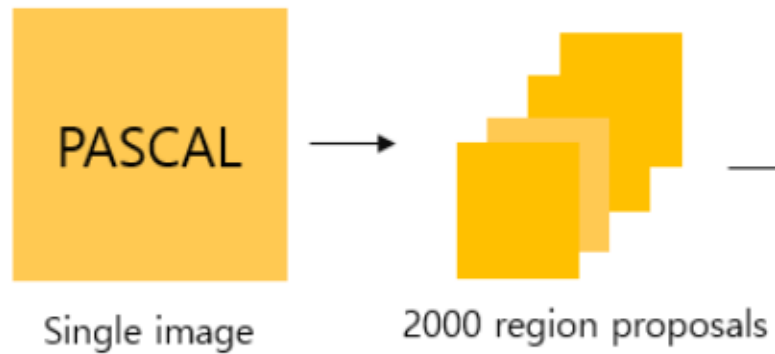


# 1. R-CNN

# R-CNN 아키텍처



# R-CNN 아키텍처



입력 이미지 → 후보 영역(Region proposals)

- Selective Search 알고리즘을 사용해 후보 영역 (Region proposals) 생성

각 후보 영역을 고정 크기로 resize

- 각 영역을 이미지에서 잘라낸 뒤( $227 * 227$ )로 리사이즈(resize)

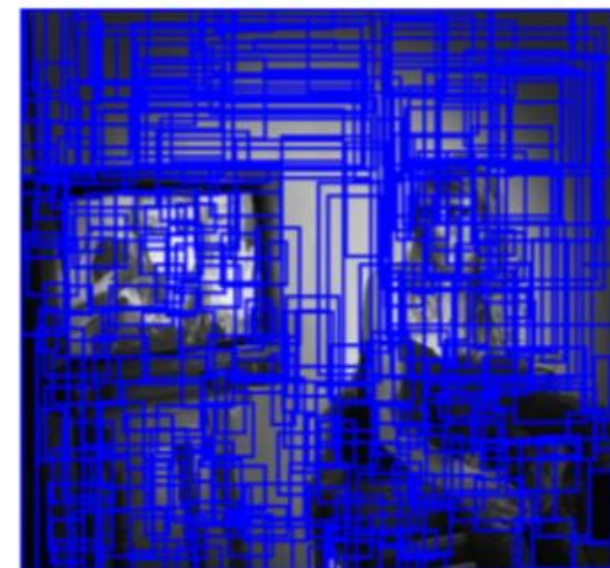
# Selective Search

## 1. 초기 분할(superpixel)

- 그래프 기반 세그멘테이션 (Felzenszwalb & Huttenlocher, 2004) 을 사용
- 각 픽셀을 노드(node) 로 보고, 이웃 픽셀 사이의 색상/밝기 차이를 가중치(weight) 로 설정

$\text{Dif}(A, B) < \min(\text{Int}(A) + \tau(A), \text{Int}(B) + \tau(B))$  이면 병합

$\text{Dif}(A, B)$	영역 A와 B의 경계 차이(혹은 유사도)	두 영역이 얼마나 다른 색/텍스처를 가지는지 나타내는 수치
$\text{Int}(A)$	영역 A 내부의 내부 차이(internal difference)	영역 A 안에서 픽셀들이 얼마나 비슷한가
$\tau(A)$	작은 영역 보정값 (threshold function)	영역 크기가 작을수록 병합이 쉽게 되도록 조절하는 값
$\min(\cdot)$	두 영역의 내부 차이 중 더 작은 쪽을 기준으로 비교	두 영역 중 덜 복잡한 쪽 기준으로 판단



# Selective Search

## 2. 계층적 병합

- 유사도 높은 영역끼리 병합하며 모든 중간 결과 저장
- 기준: 색상, 질감, 밝기/명암, 크기, 인접성
- 병합 과정 중 나온 **모든 중간 영역**을 잠재적 객체 후보로 봄

## 3. 후보 필터링

- 중복/작은, 큰 영역 제거
- **IoU(영역 겹침)** 을 기준으로 비슷한 후보들을 묶거나 제거



# R-CNN 아키텍처

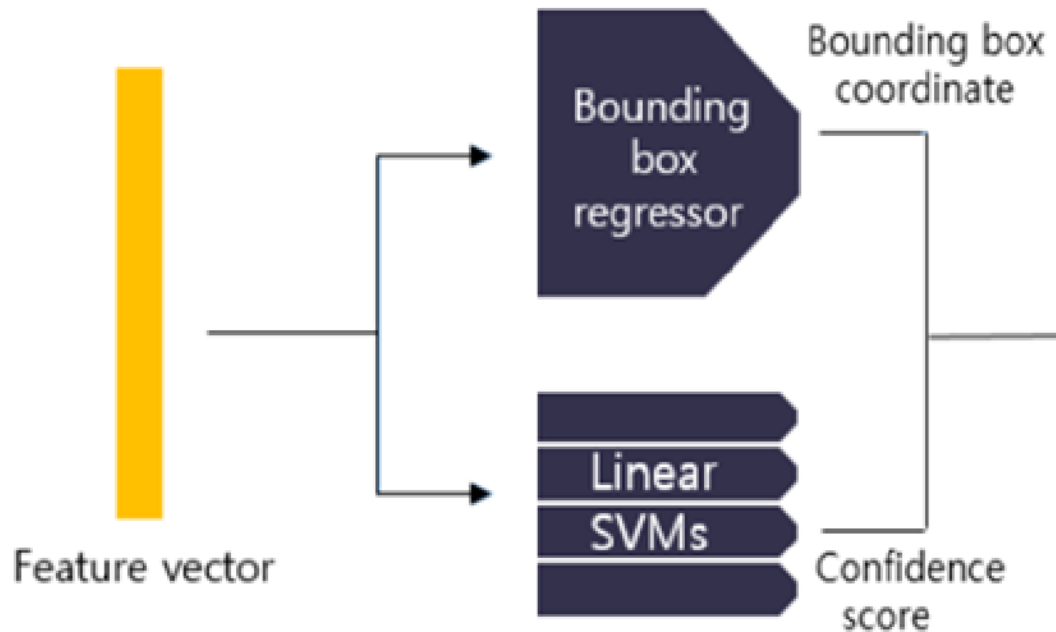


CNN에 개별 region proposal 입력

- 모든 후보 영역을 파인 튜닝(**fine tuning**)된 **AlexNet**을 통과시켜 고정 길이 (2000x4096) feature 벡터를 얻음



# R-CNN 아키텍처



1. **Linear SVMs(Support Vector Machine)** 모델은  $2000 \times 4096$  feature vector를 입력 받아 **class**를 예측하고 신뢰도 점수(confidence score)를 반환

이 때 linear SVM 모델은 특정 class에 해당하는지 여부만을 판단하는 **이진 분류기(binary classifier)**

2. **Bounding box regressor** 모델은 bounding box의 좌표를 변환하여 객체의 위치를 세밀하게 조정

# Confidence score

모델이 특정 객체가 존재한다고 확신하는 정도를 0에서 1 사이의 확률 값으로 표현한 점수

R-CNN

$$f(x) = w^T x + b$$

SVM의 결정 함수 값 자체가 **confidence score** 역할

Yolo

객체 존재 확률  $P(\text{object})$  따로 예측

클래스 확률  $\text{Softmax} \rightarrow P(\text{class} \mid \text{object})$

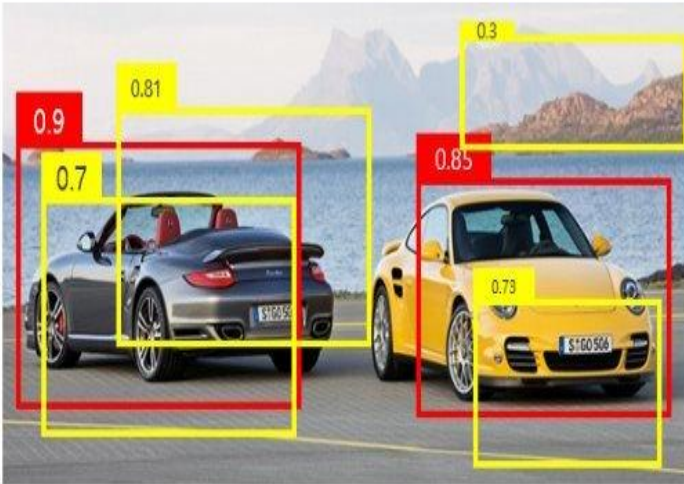
최종 confidence  $P(\text{object}) \times P(\text{class} \mid \text{object})$

# NMS(Non-Maximum Suppression)

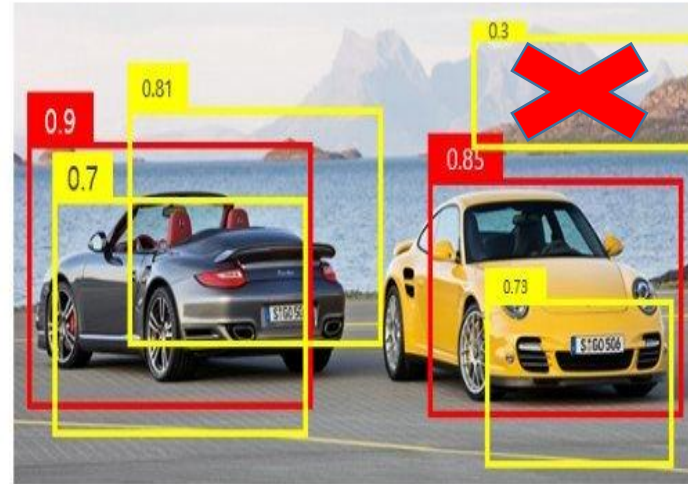
겹치는 여러 후보 박스들 중 예측 점수가 가장 높은 박스를 남기고, 그와 일정 IoU 이상 겹치는 나머지 박스들을 제거하여 중복 검출을 억제하는 알고리즘

1. bounding box별로 지정한 confidence score threshold 이하의 box를 제거.

- ex) confidence score threshold = 0.5 로 지정.
- 그림에서 confidence score가 0.3인 box를 제거.



Before Non Maximum Suppression



Before Non Maximum Suppression

# NMS(Non-Maximum Suppression)

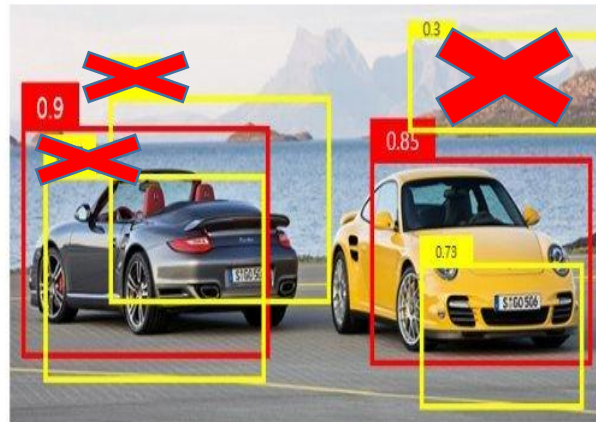
2. 남은 bounding box를 confidence score에 따라 내림차순으로 정렬.

- confidence score에 따라 내림차순으로 box를 정렬합니다. **[0.9, 0.85, 0.81, 0.73, 0.7]**

3. confidence score가 높은 순의 bounding box부터 다른 box와의 IoU 값을 조사하여 IoU threshold 이상인 box를 모두 제거. (IoU threshold = 0.4 로 지정.)

- confidence score가 0.9인 box와 나머지 box와의 IoU 값 조사.
- ex) **[0.85 : 0, 0.81 : 0.44, 0.73 : 0, 0.7 : 0.67]**

- IoU threshold 이상인 confidence score가
- **0.81, 0.7**인 box 제거. **[0.9, 0.85, 0.73]**
- 남은 box에 대하여 위의 과정 반복.



Before Non Maximum Suppression



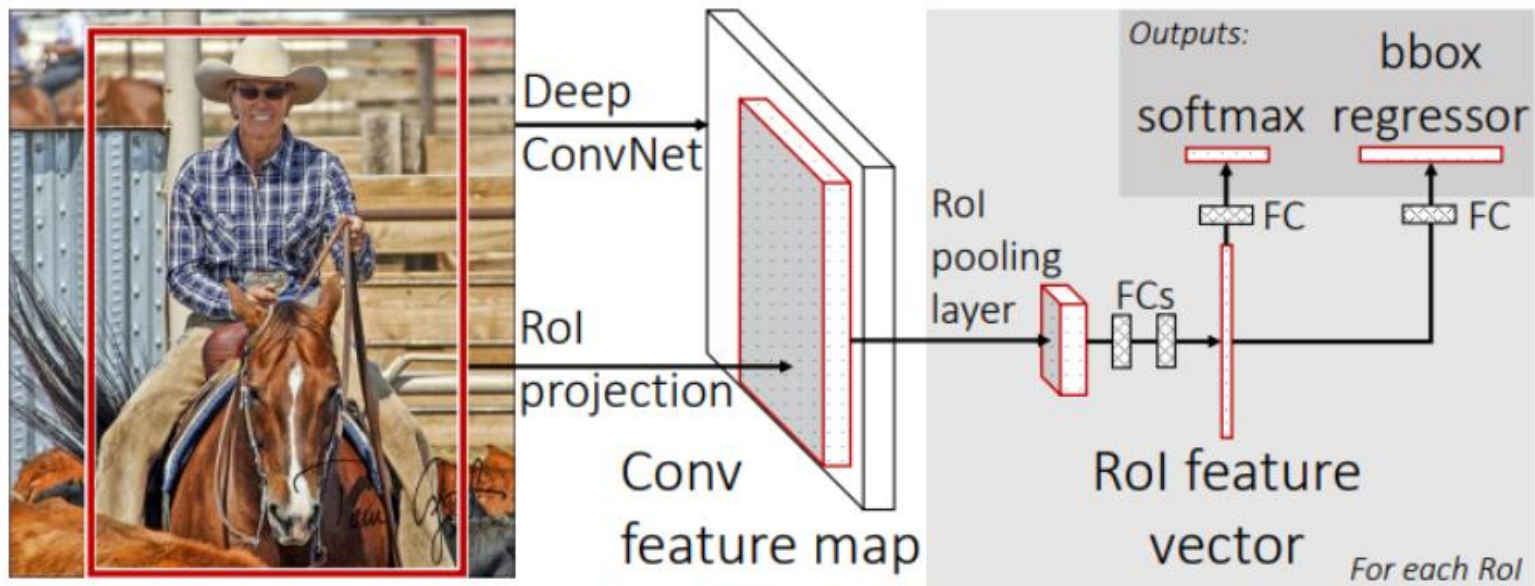
After Non Maximum Suppression

# 2. Fast R-CNN

## R-CNN과 Fast R-CNN 차이점

구분	R-CNN	Fast R-CNN
연산 구조	모든 Region Proposal(후보 영역)을 각각 CNN에 통과시켜 특징 추출 → 연산량 많고 느림	원본 이미지를 한 번만 CNN에 통과시켜 <b>Feature Map</b> 을 공유 → 속도 향상
Backbone 모델	<b>AlexNet</b> 사용	<b>VGG16</b> 사용
ROI 처리 방식	잘라낸 이미지를 개별로 CNN에 통과시킴	Feature Map 위에서 <b>ROI Pooling</b> 을 수행하여 고정 크기 feature 추출

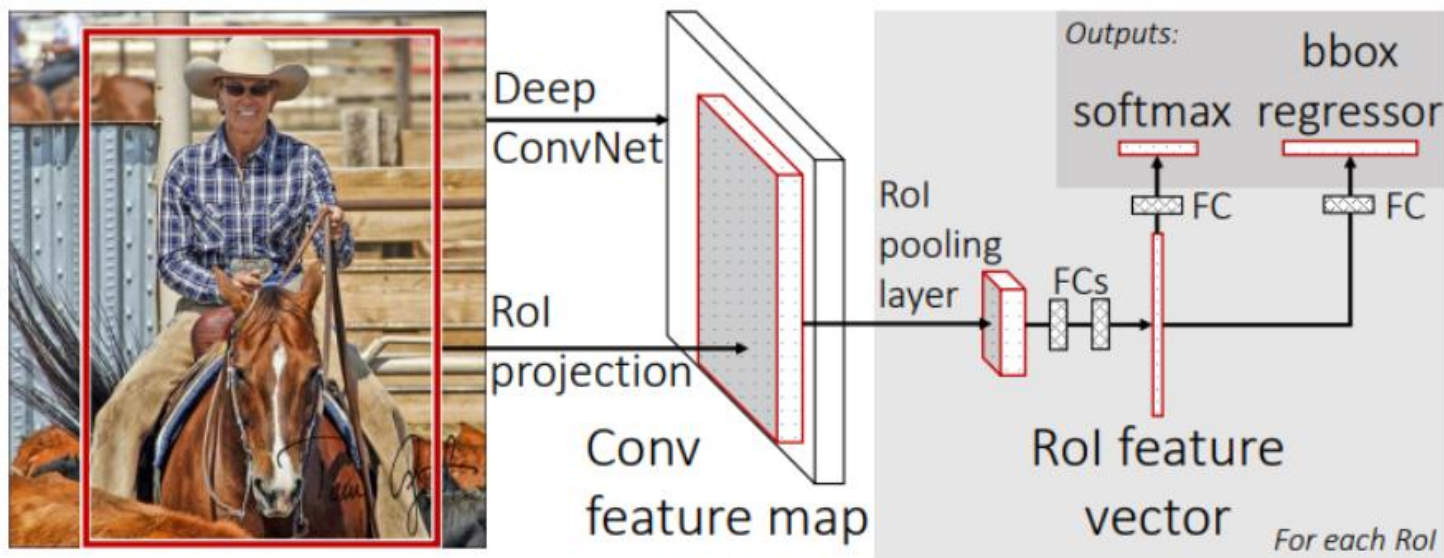
# Fast R-CNN 아키텍처



1. 원본 이미지에 대하여 Selective search 알고리즘을 적용하여 미리 후보 영역(region proposals)를 추출
2. 원본 이미지에 CNN을 통과시켜 feature map을 추출.
3. 후보 영역(region proposals)를 feature map에 대하여 투영(projection)을 진행한 후, **RoI pooling**을 수행 이 과정을 거쳐 고정된 크기의 feature map을 추출



# Fast R-CNN 아키텍처



4. region proposal별로 feature map을 flatten한 후 FC layer에 입력하여 feature vector를 얻는다.
5. softmax를 거치며(왼쪽 FC) object Classification 을 진행.
6. bounding box regression을 거쳐 예측한 Bounding box를 Ground Truth와 비교하며 조정.

# RoI Pooling



feature map에서 region proposals에 해당하는 **관심 영역(Region of Interest)**을 지정한 크기(정수)의 grid로 나눈 후 max pooling을 수행하는 방법

**RoI pooling**을 이용함으로써 랜덤한 크기를 가지는 ROI projection이 FC layer에 들어갈 수 있도록 고정된 크기를 갖게 됨.



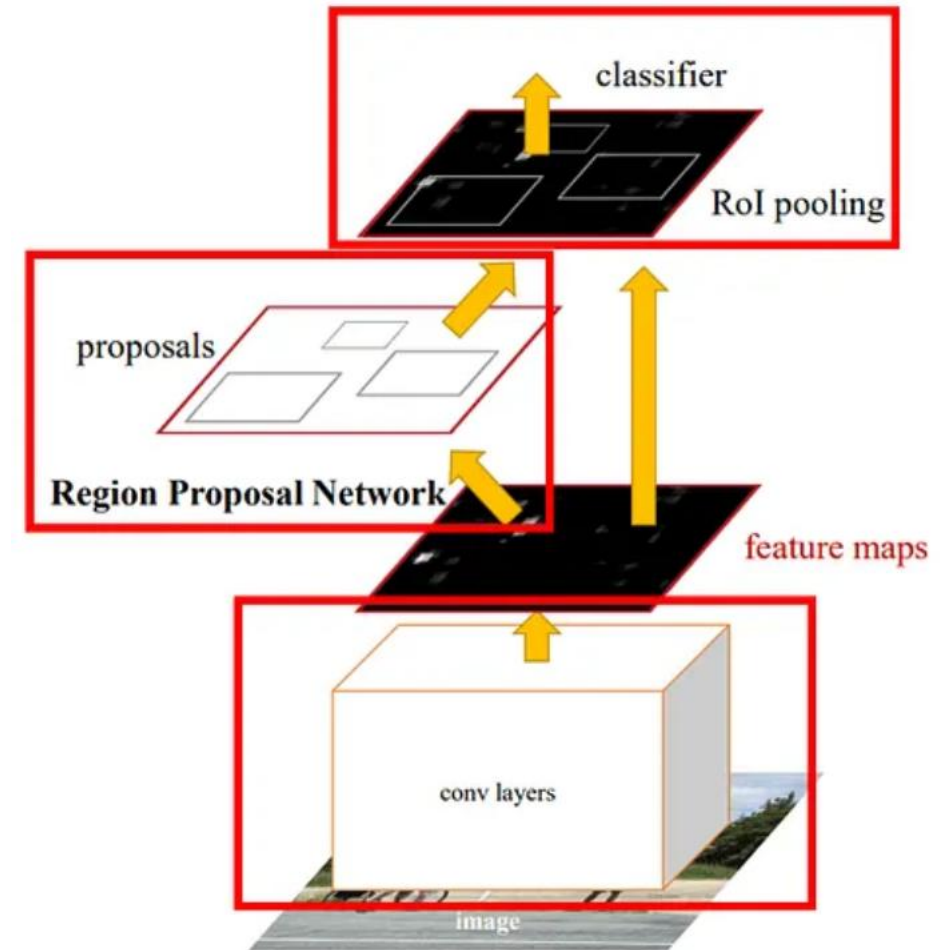
# 3. Faster R-CNN

Fast R-CNN 과 Faster R-CNN 차이점

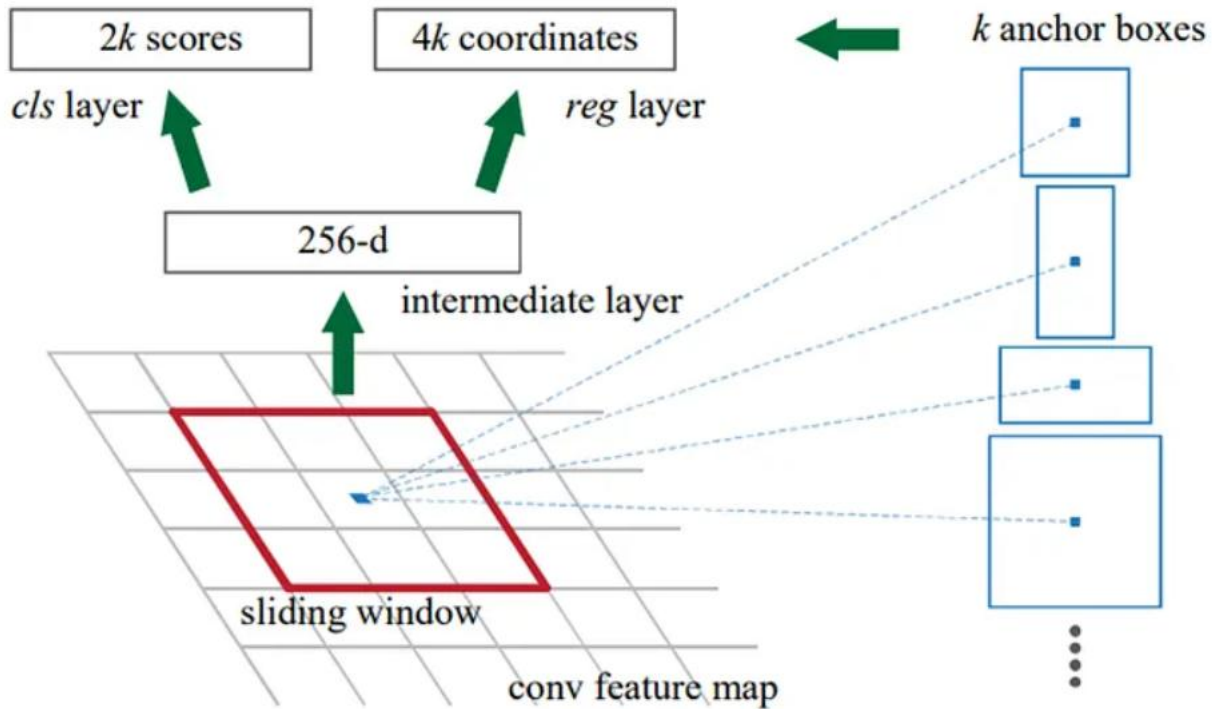
구분	Fast R-CNN	Faster R-CNN
Region Proposal 생성 방식	Selective Search (비학습적, CNN 외부에서 동작)	RPN (Region Proposal Network, CNN 내부에서 학습적으로 생성)
Feature Map 생성	원본 이미지를 한 번만 CNN에 통과해 Feature Map 생성	동일 — CNN에서 Feature Map 생성
Feature Map 공유 범위	RoI 간 공유 (모든 후보 영역이 하나의 Feature Map 사용) Region Proposal 단계에서는 공유 불가 (Selective Search는 CNN 밖에서 실행)	RPN과 RoI Head 모두 Feature Map 공유 (End-to-End 구조)

# Faster R-CNN 아키텍처

1. 원본 이미지를 사전 훈련 (pre-train)된 CNN 모델에 입력하여 feature map을 얻는다.
2. Feature map은 RPN(Region Proposal Network) 모듈에 전달되어 후보 영역(region proposals)을 산출
3. 후보 영역(region proposals)과 이전 과정에서 얻은 feature map을 통해 RoI pooling을 수행하여 고정된 크기의 feature map을 얻는다.
4. Fast R-CNN 모델에 고정된 크기의 feature map을 입력하여 분류와 Bounding box regression을 수행



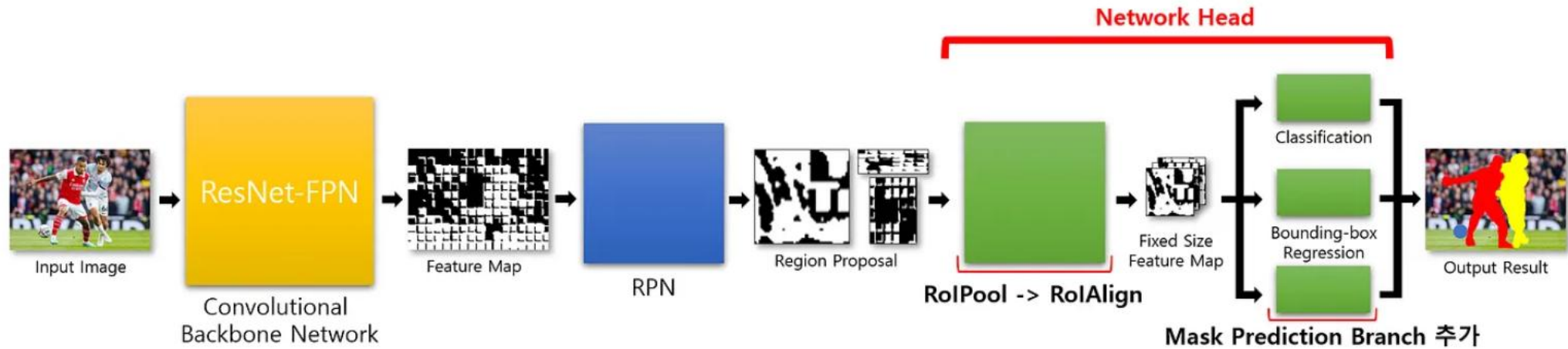
# RPN(Region Proposal Network)



!이미지에서 객체가 있을만한 위치를 효율적으로 찾아내는 역할!

1. 원본 이미지를 사전 훈련(pre-trained)된 VGG16모델에 입력하여 feature map 추출.
2. 추출한 feature map에 대하여 3x3 conv 연산을 적용. 이때 feature map의 크기가 유지될 수 있도록 padding을 추가
3. output의 결과는 각각 classification layer 와 regressoin layer로 들어간다
4. 후처리  
NMS로 중복박스제거

## 4. Mask R-CNN



Mask R-CNN과 Faster R-CNN의 차이점.

1. VGG16 -> Resnet
2. RoI Pooling 대신에 RoI Align을 사용
3. Mask Branch를 추가

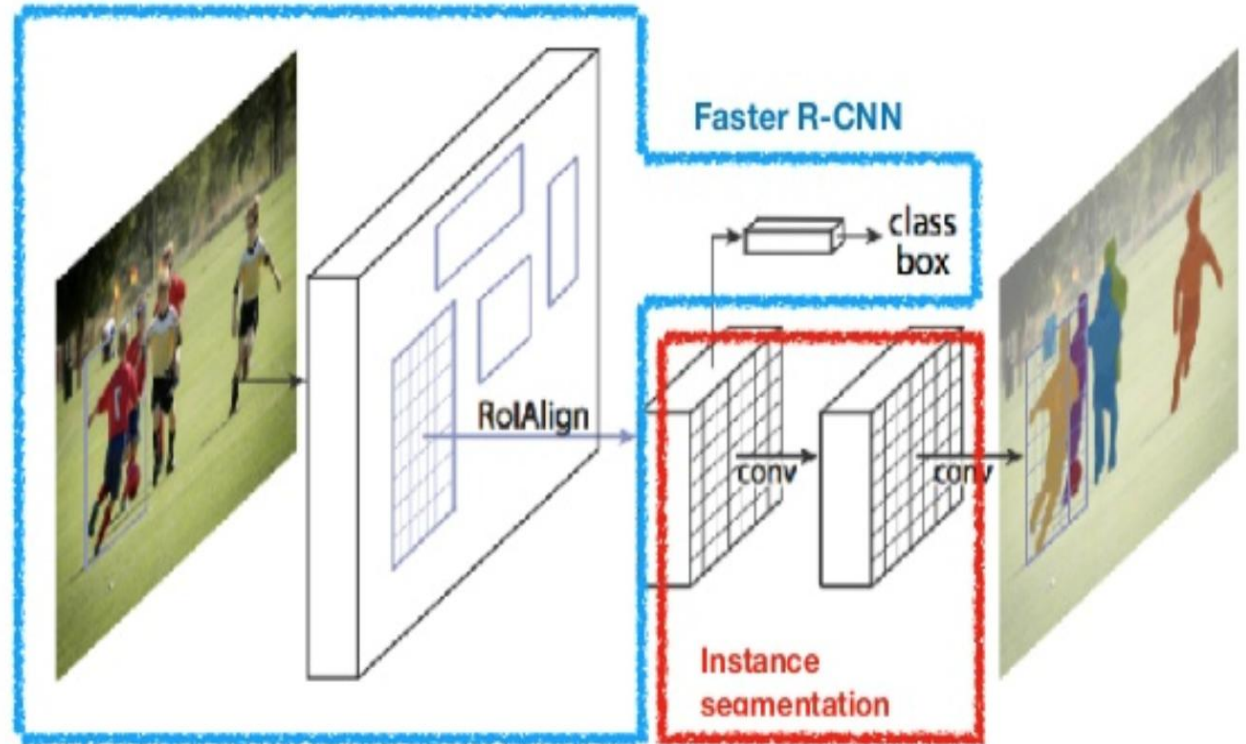
# Mask R-CNN 아키텍처

## 1. 첫 번째 Stage는 Convolutional Backbone Network

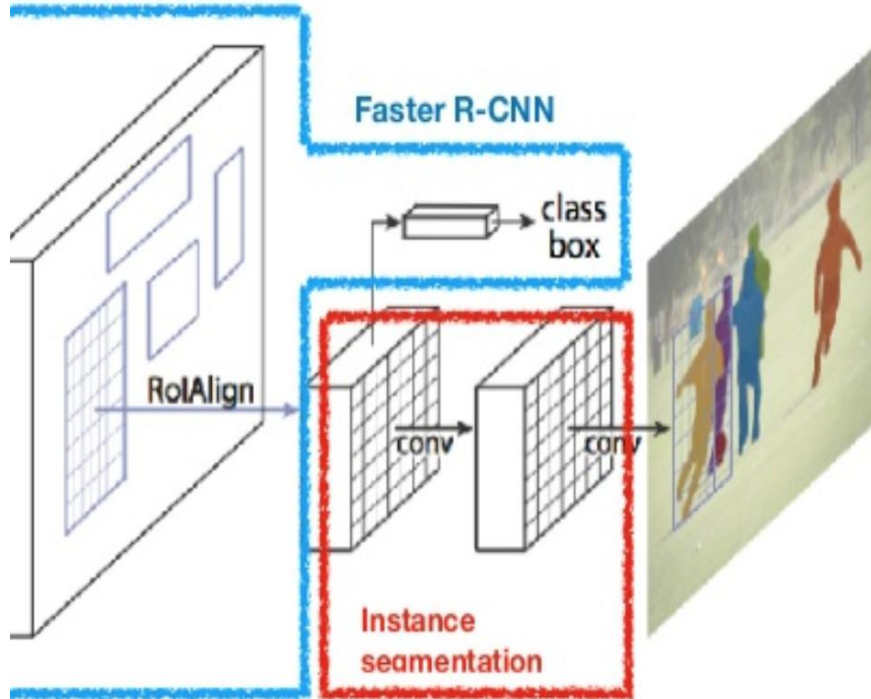
- Backbone Network의 주된 목적은 입력된 이미지로부터 Feature Map을 생성하는 것

### RPN(Region Proposal Network)

- Convolutional Backbone Network로부터 생성된 Feature Map을 입력 받고 이미지 속에서 객체가 속할 가능성이 있는 RoI(Region of interest)를 생성



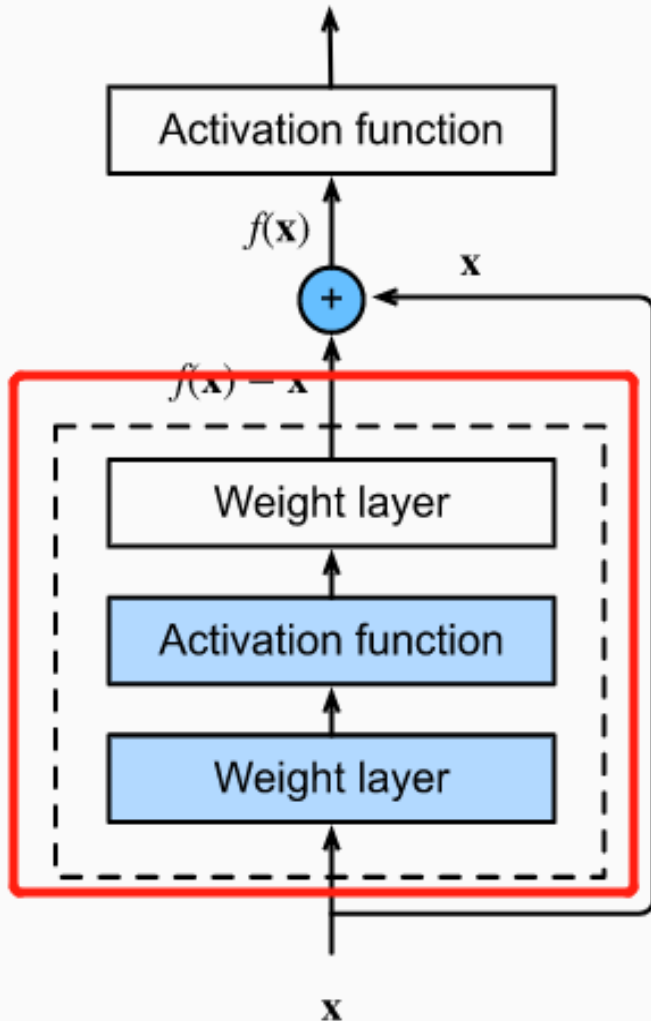
# Mask R-CNN 아키텍처



2번째 stage는 RPN이 생성한 RoI가 RoIAlign Layer를 거쳐 고정된 크기의 RoI로 변환.

- 고정된 크기의 RoI들은 Classification, Bounding-box Regression, 그리고 Mask Prediction Branch에 각각 입력.
- Classification Branch는 Feature Map에 속한 객체를 예측하고, Bounding-box Regression Branch에서는
- Bounding-box가 객체를 더 잘 포함할 수 있도록 Bounding-box의 위치를 수정.
- 마지막으로 Mask Prediction Branch에서는 Segmentation Mask를 생성

# ResNet-FPN(Feature Pyramid Network)



왜 백본 네트워크의 변경이 이뤄졌는가?

- 기존 VGG-16 모델의 한계

VGG-16 모델은 단일 feature map(C5)만 사용.

고해상도 정보가 손실되어 작은 객체 검출 성능이 떨어짐.

- ResNet 도입 — 더 깊고 안정적인 특징 추출

**Residual Connection** 도입을 통해, 네트워크 깊이가 깊어질수록 성능이 저하되는 **Gradient Vanishing** 문제를 해결

입력  $x$ 를 출력에 직접 더해주는 Residual Function

$$H(x) = F(x) + x$$



# ResNet-FPN(Feature Pyramid Network)

FPN과 ResNet의 결합으로 객체 검출 분야 시너지 발휘

## FPN의 구조

### 1. Bottom-up Pathway

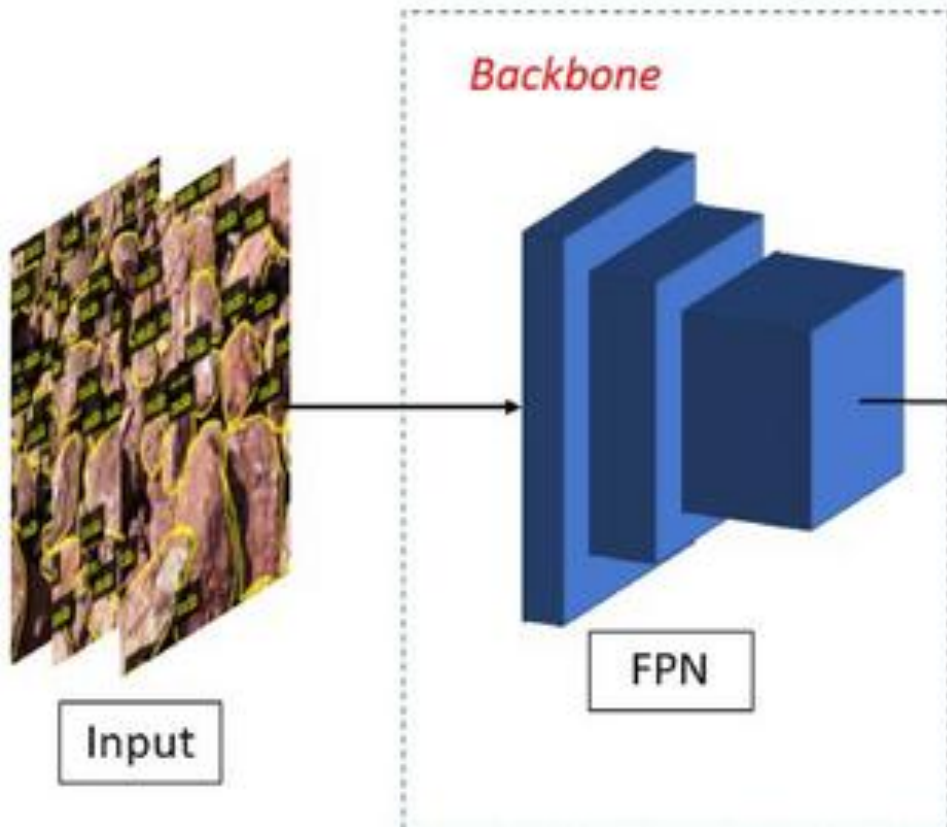
ResNet의 순방향 연산(forward pass)을 통해 저해상도이지만 의미적 정보가 풍부한 고수준 특징 맵을 추출

### 2. Top-down Pathway

상위 레벨의 특징 맵을 업샘플링하여 하위 레벨로 전파하며, 이를 통해 고수준의 의미적 정보를 보강

### 3. Lateral Connections

Bottom-up pathway의 동일 스케일 특징 맵과 Top-down pathway의 특징 맵을 연결하여, 위치 정보의 손실을 최소화



*ResNet: 깊은 계층으로 이미지에서 추상적인 특징을 효과적으로 추출하고, 안정적인 기울기 흐름을 유지*

*FPN: 다양한 해상도와 의미 정보를 가진 특징 피라미드를 생성하여 여러 크기의 객체 검출에 최적화*



# ROI Pooling → ROI Align

0.8	1.1	2.1	0.8	1.1	2.1	3.9	0.4
2.2	4.3	4.8	2.2	4.3	4.2	1.8	6.3
0.2	5.5	4.8	0.2	5.5	4.8	7.7	9.5
0.8	1.1	2.1	0.8	1.1	2.1	3.9	0.4
2.2	4.3	4.2	2.2	4.3	4.2	1.8	6.3
0.2	5.5	4.8	0.2	5.5	4.8	7.7	9.5
1.1	0.8	2.2	1.1	0.8	2.2	0.8	5.2
5.0	1.8	1.9	5.0	1.8	1.9	2.0	4.1

8 x 8 Feature Map  
Red Dotted Line = Roi



2.65	4.2
3.6	3.7

- 빨간 점 = 샘플링 포인트
- 숫자 = 보간된 값
- 오른쪽의 2x2 결과 = 고정 크기 ROI

## ◆ 개념 요약

ROI Pooling은 feature map을 정수 좌표로 잘라서 위치 정보가 손실되는 문제가 있었음.

ROI Align은 이걸 **bilinear interpolation**(양선형보간)으로 보정함.

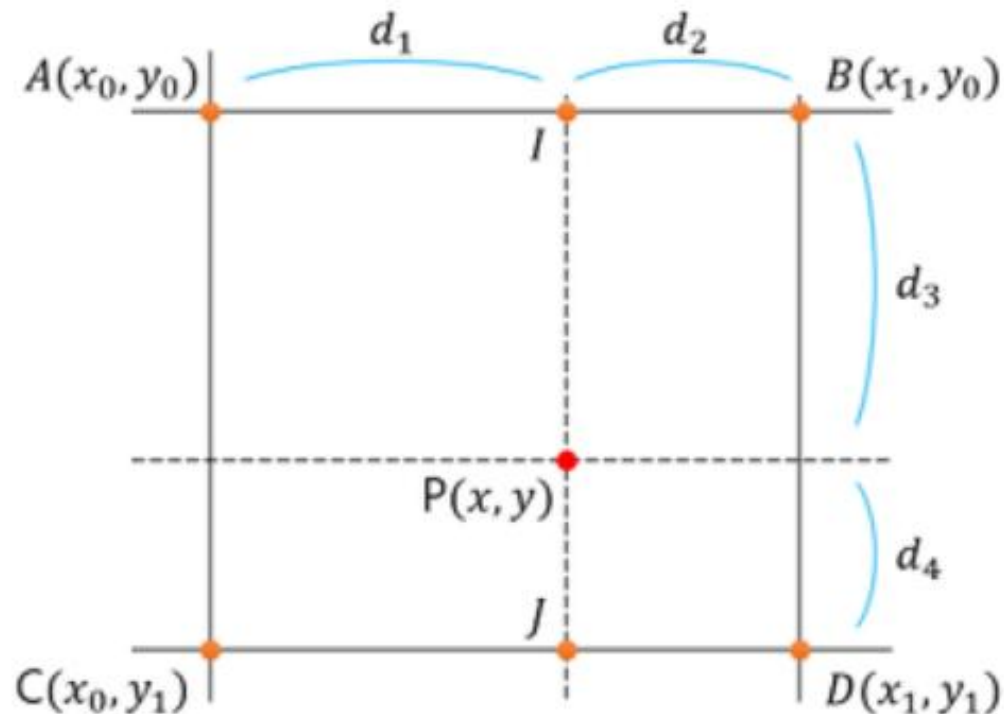
## ◆ 동작 과정

- 1) ROI를 세분화해서 bin 생성
- 2) 각 bin 안에서 4개의 sampling point를 생성
- 3) **bilinear interpolation**으로 sampling point 값 보간
- 4) 4개의 보간 값의 평균(Average Pooling)으로 feature map 생성

## ◆ 효과

- Feature map의 **정밀한 위치 정보 유지**
- Bounding box / Mask 예측 시 **픽셀 단위 정확도 향상**

# bilinear interpolation

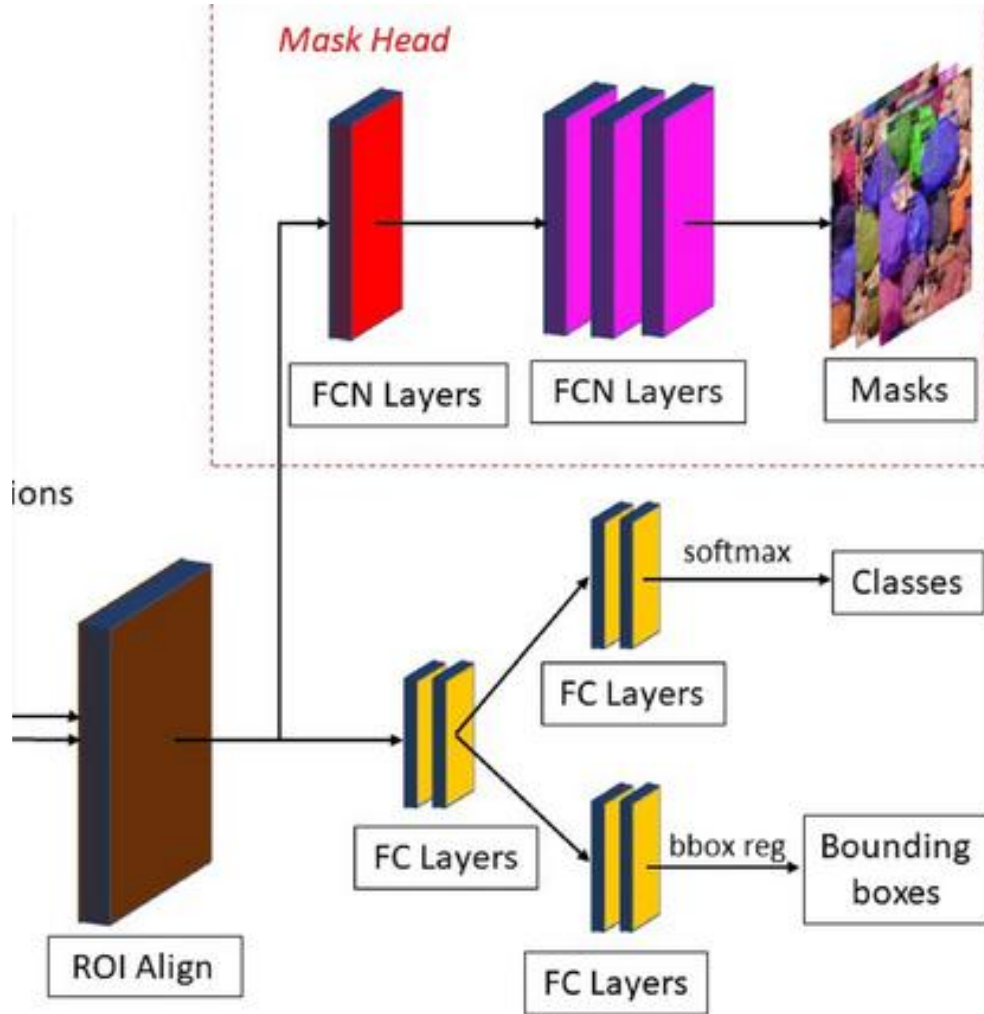


RoI Align에서는 feature map의 실수 좌표에서 값을 얻기 위해 이 방식을 사용

$$f(x, y) = f(0, 0)(1 - x)(1 - y) + f(1, 0)x(1 - y) + f(0, 1)(1 - x)y + f(1, 1)xy$$

좌표 사이의 값을 주변 네 개 픽셀의 가중평균으로 계산하는 방법

# Mask Head



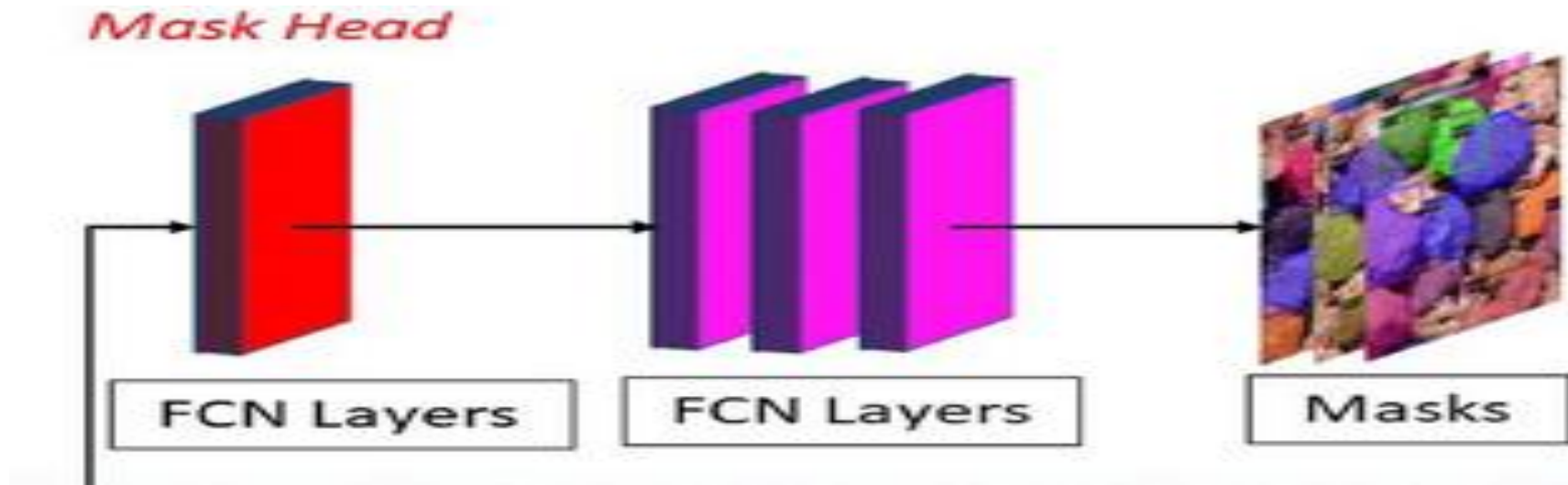
## ◆ 개념 요약

- Mask Head는 RoIAlign 결과를 입력으로 받아 각 객체의 픽셀 단위 마스크(Instance Segmentation)를 예측합니다
- 기존 Faster R-CNN의 Classification + BBox Head와 병렬 구조로 추가된 브랜치입니다
- FCN(Fully Convolutional Network) 기반으로 Conv 연산을 통해 공간적 정보를 보존합니다

## ◆ 주요 특징과 효과

- **Bounding Box + Mask 예측 병렬 수행** → Detection + Segmentation 동시 달성
- **FCN 구조**로 공간 정보 보존 → 픽셀 단위 정확도 상승
- **효율적 구조**: 28×28 크기의 작은 마스크로 예측 후 원본 크기로 확대하여 계산 효율성을 확보합니다

# Mask Head



## ◆ 동작 과정

1. ROI Align으로 추출된 feature → Mask Head 입력
2. 여러 개의 Conv Layer와 Deconv Layer 거친 후 각 ROI 마다  $N \times N$  크기의 Binary Mask 출력
3. 클래스마다 별도의 mask 예측 → **Sigmoid 활성화**로 독립 처리 (Softmax X : 클래스 간 mask 중첩 가능)

# 비교표

구분	R-CNN (2014)	Fast R-CNN (2015)	Faster R-CNN (2016)	Mask R-CNN (2017)
핵심 기술	Selective Search로 영역 제안 + CNN 특징 추출	하나의 CNN feature map 에서 RoI pooling	RPN(Region Proposal Network) 도입	Segmentation branch 추가
속도 (FPS)	~0.02 (47초/이미지)	~0.43 (2.3초/이미지)	5 (0.2초/이미지)	3-5 (0.2-0.3초/이미지)
주요 혁신점	최초 CNN 기반 객체 검출	공유 연산으로 속도 향상	완전한 end-to-end 구조	Instance segmentation 확장

- 정확도 향상: R-CNN: 53.3% mAP → Fast R-CNN: 65.7% mAP → Faster R-CNN: 73.2% mAP → Mask R-CNN: 37.1% mAP (COCO)
- 기능 확장: 검출 → 빠른 검출 → 통합 학습 → 세그멘테이션

# Mask R-CNN구현

```
def load_model(device):  
    """COCO 데이터셋으로 사전 학습된 Mask R-CNN 모델 로드"""  
    print("\n Mask R-CNN 모델 로딩 중...")  
  
    try:  
        from torchvision.models.detection import MaskRCNN_ResNet50_FPN_Weights  
        weights = MaskRCNN_ResNet50_FPN_Weights.DEFAULT  
        model = maskrcnn_resnet50_fpn(weights=weights)  
    except:  
        model = maskrcnn_resnet50_fpn(pretrained=True)  
  
    model.to(device)  
    model.eval()  
    print(" 모델 로딩 완료!")  
    return model
```

# Mask R-CNN 구현

```
for i, (box, label) in enumerate(zip(boxes, labels)):
    # person 클래스만 처리
    if COCO_CLASSES[label.item()] != 'person':
        continue

    # 바운딩 박스 좌표
    x1, y1, x2, y2 = map(int, box.tolist())

    # 여백 추가 (얼굴 검출 성공률 향상)
    h, w = image_cv.shape[:2]
    margin = 20
    x1 = max(0, x1 - margin)
    y1 = max(0, y1 - margin)
    x2 = min(w, x2 + margin)
    y2 = min(h, y2 + margin)

    # 얼굴 영역 추출
    face_region = image_cv[y1:y2, x1:x2]

    # 임시 파일로 저장 (DeepFace는 파일 경로 필요)
    temp_path = "temp_face.jpg"
    cv2.imwrite(temp_path, face_region)

    try:
        # DeepFace로 임베딩 추출
        embedding_obj = DeepFace.represent(
            img_path=temp_path,
            model_name='Facenet',
            enforce_detection=False # 검출 실패해도 계속 진행
        )
```



# Mask R-CNN구현





# 참조

Girshick, R. (2015). *Fast R-CNN*. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Ren, S., He, K., Girshick, R., & Sun, J. (2016). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. *Advances in Neural Information Processing Systems (NeurIPS)*.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., & Smeulders, A. W. M. (2013). *Selective Search for Object Recognition*. *International Journal of Computer Vision*, 104(2), 154–171

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). *Mask R-CNN*. *ICCV*

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). *Rich feature hierarchies for accurate object detection and semantic segmentation*. *Proceedings of CVPR*.

Karen Simonyan, Andrew Zisserman(2015). VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

# Q&A

감사합니다.