

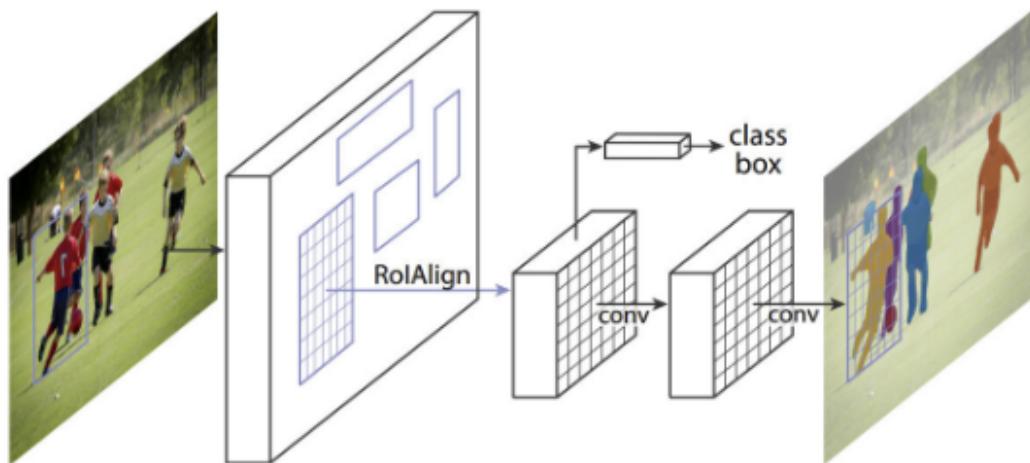
MASK R-CNN

Abstract

- 5fps로 실행되는 Faster R-CNN에 약간의 오버헤드(Mask Branch)를 추가한 모델
: 병렬로 object mask를 예측하기 위한 branch 추가
- 훈련이 간단하고 유연하며 일반화하기 쉬움
ex) human pose-estimation

1. Introduction

- Faster R-CNN은 네트워크 입력과 출력 간의 픽셀 대 픽셀 정렬을 위해 설계되지 않음
- Faster R-CNN의 Roi pooling에서는 Roi가 소수점 좌표를 가질 때 각 좌표를 반올림해서 pooling해줌 → 원본 위치 정보가 왜곡됨
- Mask R-CNN에서는 Roi Align을 사용해 이러한 단점을 해결
- Instance Segment

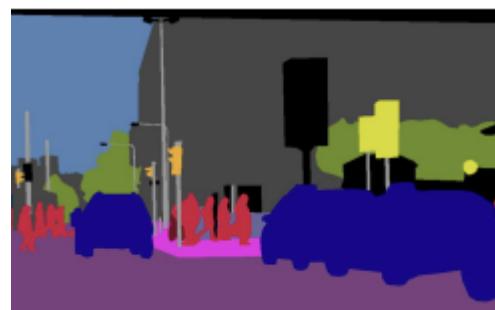


- 인스턴스 분할은 이미지의 모든 객체를 올바르게 감지하는 동시에 각 인스턴스를 정확하게 분할해야 하기 때문에 어려움
- bounding box를 사용해 개별 object를 분류하고 각 개체를 지역화하며 각 픽셀을 구별하지 않고 고정된 범주 집합으로 분류하고자 함
- Faster R-CNN에 각 ROI(Region of Interest)에서 segmentation mask를 예측하기 위한 mask branch를 추가

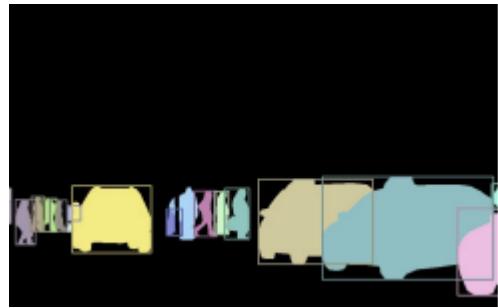
- mask branch
 - 각 ROI에 적용되는 작은 FCN
 - 픽셀 대 픽셀 방식으로 segmentation mask를 예측
 - 작은 계산 오버헤드만 추가하므로 빠른 실험 가능
- Segmentation



- Semantic segmentation



- 사진에 있는 모든 픽셀을 해당하는 class로 분류하는 것
- 동일한 class의 객체는 따로 분류하지 않고, 그 픽셀 자체가 어떤 class에 속하는지만 분류함
- 동일한 object는 같은 색으로 표현되며, 한번에 masking을 수행함
- FCN
- instance segmentation



- Semantic segmentation과 비교했을 때 각각의 같은 class의 object를 다른 label로 취급 (class가 부여되지는 않음)
- 각자 객체별로 masking 수행
- Mask R-CNN
 - panoptic segmentation

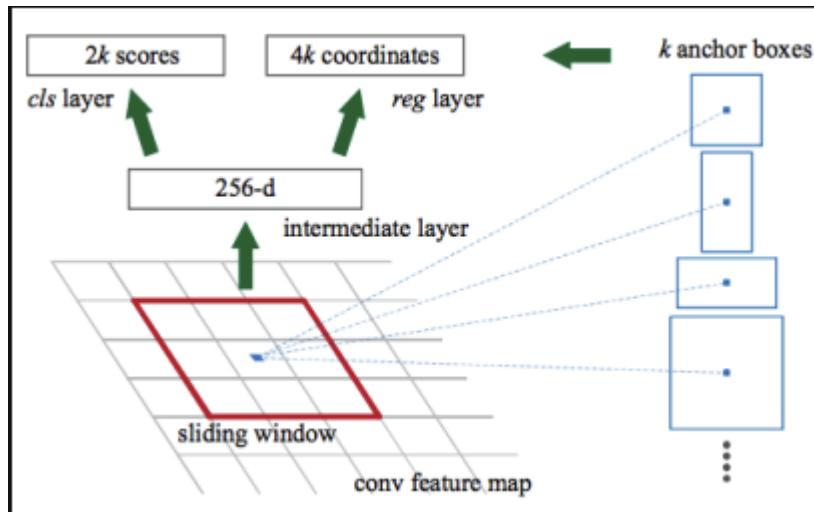


- Semantic segmentation + Instance Segmentation가 합쳐진 것
- object와 class와 instance를 함께 구분

2. Related Work

R-CNN

- 관리 가능한 수의 후보 객체 영역에 집중하고 Convolution Network를 독립적으로 평가 함
- RoIPool을 사용해 RoI feature map에 집중할 수 있도록 확장되어 빠른 속도와 더 나은 정확도를 보여줌
- Faster R-CNN은 RPN(Region Proposal Network)으로 attention 메커니즘을 학습해 이 스트림을 발전시킴
 - RPN



CNN에서 뽑아낸 feature map을 Region Proposal을 생성하기 위해 $n \times n$ window를 sliding window시키는 방식

1. 입력

: anchor box, delta, probability

2. Bounding Box 계산

: anchor와 delta를 결합해 값을 조정해 실제 객체 위치를 정확하게 표현

3. Sort 단계

: 2번 과정에서 산출된 bounding box들 중 확률이 높은 객체 사용

4. Non Maximum Suppression

a. 동일한 클래스에 대해 높은-낮은 confidence 순서로 정렬

b. 가장 confidence가 높은 bounding box와 IOU가 일정 이상인 bounding box는 동일한 물체를 detect했다고 판단해 지움





- R-CNN

본래의 CNN은 한 객체만 있는 이미지를 판별하는 용도로 만들어짐

2단계를 거쳐 이미지 내에 있는 객체를 찾아내는 방법

1. ROI 선별 : 객체가 있을 장소를 대략적으로 파악
 2. Object Detection : 얻어낸 ROI를 CNN으로 판단해 어떤 객체인지, 정확한 위치는 어디인지 등을 판별
- ⇒ 2-stage detector 임
- ex) YOLO (대표적인 1-stage detector)
- 찾아낸 영역을 잘라내 CNN에 넣어 특성을 찾아내기 때문에 시간이 많이 걸림
- Fast R-CNN
 - 어느 지점 까지는 같은 CNN에서 진행하자!
 - 같은 모델에서 연산을 수행하기 때문에 연산 횟수가 확실히 줄어듦
 - CPU에서 돌아가 속도에 그래도 한계가 존재함
 - Faster R-CNN
 - Region Proposal을 하는 부분을 GPU에서 돌아가게 함

3. Mask R-CNN

Mask R-CNN

Mask R-CNN은 Faster R-CNN의 두 개의 출력(클래스 레이블, bounding box offset)에 object mask까지 출력하고자 함

- object mask 출력 : 클래스 및 bounding box 출력과 구별되므로 더 미세한 공간 레이아웃을 추출해야 함
- 과정
 1. RPN(Region Proposal Network) : 후보 object bounding box를 제안
 2. 각 후보 box에서 ROI Pooling을 사용해 특징을 추출하고 분류 및 bounding box regression을 수행함
클래스 및 box offset을 예측 + 각 ROI에 대해 binary mask 출력
병렬적으로 예측이 이뤄짐
- loss function
각 ROI에 대한 multi-task loss를 적용 (서로 독립)

$$L = L_{cls} + L_{box} + L_{mask}$$

`mask branch` : Km²차원 (K : class 개수, m : 이미지 해상도)

k 클래스 각각에 대해 하나씩 해상도 m × m의 k 이진 mask를 인코딩 average binary cross-entropy loss 사용

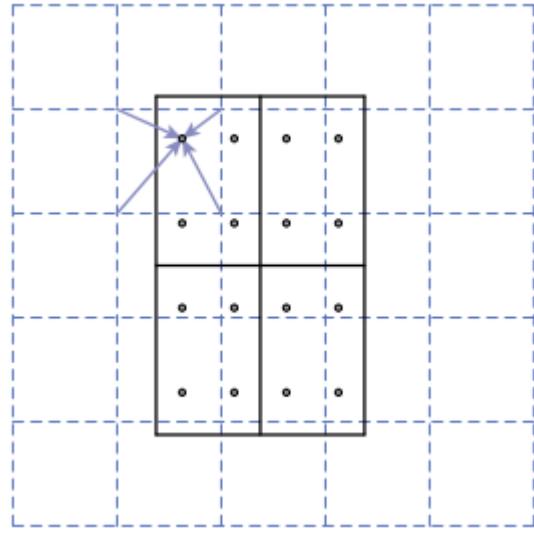
Semantic Segmentation의 경우, 픽셀마다 class와 mask를 예측해야 하므로 계산량이 많지만 Mask R-CNN은 class 예측과 독립적으로 이뤄지므로 연산량이 줄어듦

Mask Representation

- 각 ROI마다 FCN(Fully Convolution Network)을 적용해 공간 정보를 잃어버리지 않고 mxm 크기의 mask를 예측 가능
- convolution 연산에 의해 공간적 정보 손실을 최소화 가능
- FC layer를 사용하는 것보다 파라미터의 수가 훨씬 줄고 정확도는 높게 나옴

RoIAlign

- ROI pooling
 - 각 ROI에서 small feature map을 추출하기 위한 표준 연산
 - 다른 사이즈의 Region Proposal이 들어와도 max pooling을 이용해 output size를 동일하게 만듦
 - ROI를 feature map으로 quantization하게 되는데, 이 과정에서 ROI와 추출된 feature 사이에 오정렬을 초래하며 픽셀 단위로 예측하는 mask에 큰 악영향을 끼침
⇒ 추출한 feature를 input에 적절하게 정렬하는 RoIAlign layer 제안
- RoIAlign
 - bilinear interpolation 연산을 사용해 각 ROI bin의 샘플링된 4개의 위치에서 input feature의 정확한 값을 계산



■ bilinear interpolation

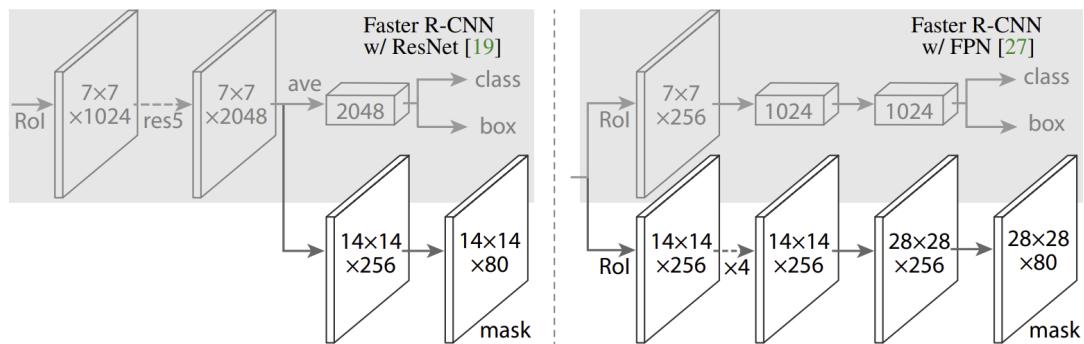
	<p>① 선형 보간법을 통해 I 와 J 에서의 Pixel 값 구한 뒤 ② I 와 J 의 좌표 정보와 Pixel 값을 이용해서 선형 보간법 통해 P의 Pixel 값을 구함 ③ 선형 보간법에는 다음의 식을 사용</p> $b = \frac{d_1}{d_1 + d_2} y_2 + \frac{d_2}{d_1 + d_2} y_1$
(1)	$f(I) = \frac{d_1}{d_1 + d_2} f(B) + \frac{d_2}{d_1 + d_2} f(A)$ $f(J) = \frac{d_1}{d_1 + d_2} f(D) + \frac{d_2}{d_1 + d_2} f(C)$
(2)	$f(P) = \frac{d_3}{d_3 + d_4} f(J) + \frac{d_4}{d_3 + d_4} f(I)$ $= \frac{1}{(d_1 + d_2)(d_3 + d_4)} \{ d_1 d_3 f(D) + d_2 d_3 f(C) + d_1 d_4 f(B) + d_2 d_4 f(A) \}$ $= \frac{1}{(d_1 + d_2)(d_3 + d_4)} \{ (y_1 - y)(x_1 - x)f(A) + (y_1 - y)(x - x_0)f(B) + (y - y_0)(x_1 - x)f(C) + (y - y_0)(x - x_0)f(D) \}$

- 결과를 max or average
- Faster R-CNN에서는 양자화가 사용되지 않음 (모델의 크기를 줄이지 못 함)

- 정수가 아닌 실수 값이 나올 때 bilinear interpolation

Network Architecture

- BackBone
 - 이미지의 feature를 추출하기 위해 사용
 - ResNet과 ResNeXt의 50, 101 layer와 FPN을 backbone으로 사용
- Head
 - Bounding Box Recognition (Classification and Regression)과 Mask Prediction을 위해 사용됨
 - Faster R-CNN의 Head(Classification and Regression)에 Mask branch를 추가
 - backbone(ResNet, FPN)에 따라 Head의 구조가 달라짐



Implementation Details

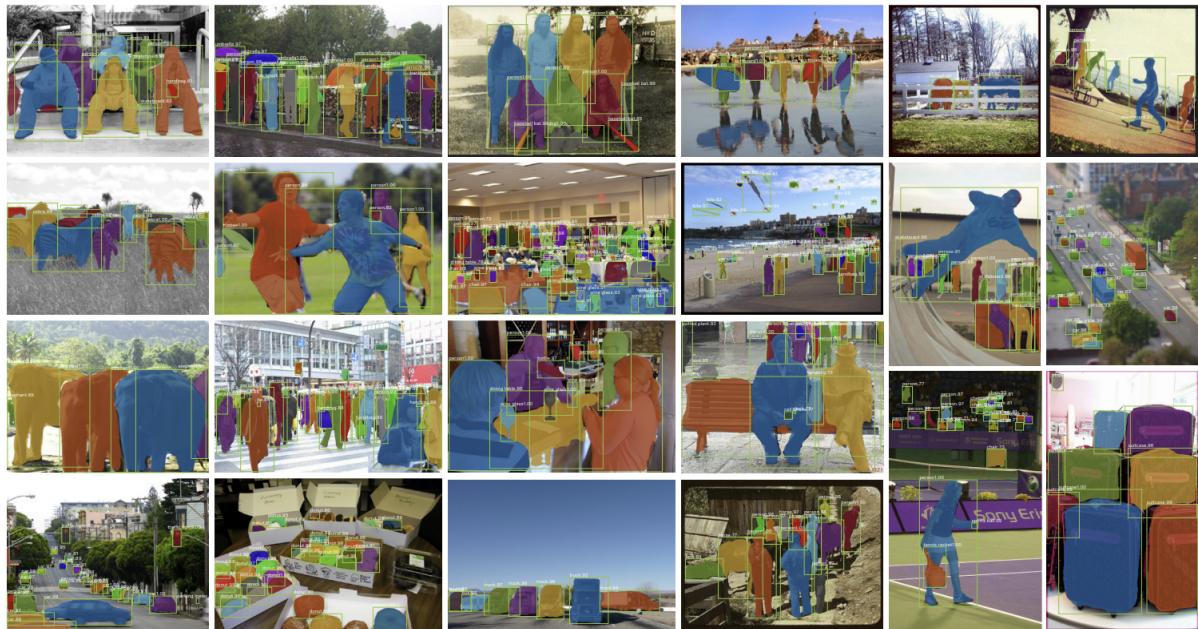
- Training
 - RoI에 의해 정해진다. (Ground-truth box와의 IoU가 0.5 이상인 것)
 - 이미지의 크기는 800픽셀
 - 미니 배치에 2장의 이미지 사용됨
 - 각 이미지에는 N개의 샘플링된 RoI가 있음. (ResNet N = 64, FPN N = 256)
 - GPU : 8개, mini batch : 16, epoch : 160,000, lr : 0.02, weight decay 0.0001, momentum 0.9
 - ResNeXt만 mini batch 당 1장의 이미지, lr은 0.01
- inference
 - ResNet은 proposal 300, FPN은 100

- box prediction 후 NMS(Non-Maximum Suppression) 적용
- Mask Branch는 score 점수가 높은 상위 100개 box에서 적용됨. 이는 속도와 정확도 향상에 도움
- mask Branch는 각 ROI마다 K개의 Mask를 예측 (K : classification branch에서 예측된 class들)

4. Experiments

Main Result

모든 기존의 SOTA 보다 뛰어난 성능을 보임



	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	37.1	60.0	39.4	16.9	39.9	53.5

<i>net-depth-features</i>	AP	AP ₅₀	AP ₇₅
ResNet-50-C4	30.3	51.2	31.5
ResNet-101-C4	32.7	54.2	34.3
ResNet-50-FPN	33.6	55.2	35.3
ResNet-101-FPN	35.4	57.3	37.5
ResNeXt-101-FPN	36.7	59.5	38.9

(a) **Backbone Architecture:** Better backbones bring expected gains; deeper networks do better, FPN outperforms C4 features, and ResNeXt improves on ResNet.

	AP	AP ₅₀	AP ₇₅	AP ^{bb}	AP ₅₀ ^{bb}	AP ₇₅ ^{bb}
<i>RoIPool</i>	23.6	46.5	21.6	28.2	52.7	26.9
<i>RoIAlign</i>	30.9	51.8	32.1	34.0	55.3	36.4
	+7.3	+5.3	+10.5	+5.8	+2.6	+9.5

(d) **RoIAlign** (ResNet-50-C5, stride 32): Mask-level and box-level AP using *large-stride* features. Misalignments are more severe than with stride-16 features (Table 2c), resulting in big accuracy gaps.

	AP	AP ₅₀	AP ₇₅
<i>softmax</i>	24.8	44.1	25.1
<i>sigmoid</i>	30.3	51.2	31.5

(b) **Multinomial vs. Independent Masks** (ResNet-50-C4): *Decoupling* via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

	align?	bilinear?	agg.	AP	AP ₅₀	AP ₇₅
<i>RoIPool</i> [12]			max	26.9	48.8	26.4
<i>RoIWarp</i> [10]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
<i>RoIAlign</i>	✓	✓	max	30.2	51.0	31.8
	✓	✓	ave	30.3	51.2	31.5

(c) **RoIAlign** (ResNet-50-C4): Mask results with various ROI layers. Our RoIAlign layer improves AP by ~3 points and AP₇₅ by ~5 points. Using proper alignment is the only factor that contributes to the large gap between ROI layers.

	mask branch		AP	AP ₅₀	AP ₇₅
MLP	fc: 1024 → 1024 → 80 · 28 ²		31.5	53.7	32.8
MLP	fc: 1024 → 1024 → 1024 → 80 · 28 ²		31.5	54.0	32.6
FCN	conv: 256 → 256 → 256 → 256 → 256 → 80		33.6	55.2	35.3

(e) **Mask Branch** (ResNet-50-FPN): Fully convolutional networks (FCN) vs. multi-layer perceptrons (MLP, fully-connected) for mask prediction. FCNs improve results as they take advantage of explicitly encoding spatial layout.

Bouding Box Detection Result

	backbone	AP ^{bb}	AP ₅₀ ^{bb}	AP ₇₅ ^{bb}	AP _S ^{bb}	AP _M ^{bb}	AP _L ^{bb}
Faster R-CNN+++ [19]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [27]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [21]	Inception-ResNet-v2 [41]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [39]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	39.8	62.3	43.4	22.1	43.2	51.2

Timing

- Inference

- Nvidia Tesla M40 GPU에서 이미지 1개당 195ms 소요

- Training

- ResNet-50-FPN backbone, COCO trainval135k로 학습 시 32시간 소요
- ResNet-101-FPN은 44시간 소요

5. Mask R-CNN for Human Pose Estimation

Mask R-CNN을 인간 pose estimation으로 확장가능

- 키포인트의 위치를 원-핫 마스크로 모델링하고 마스크 R-CNN을 채택해 K개의 키포인트 유형마다 하나씩 K 마스크를 예측함
⇒ Mask R-CNN의 유연성



Implementation Details

- 인스턴스의 각 K 키포인트에 대해 훈련 대상 : 단일 픽셀만 전경으로 레이블되는 원-핫 mxm 이진 마스크
- 훈련되는 동안 각각의 가시적인 truth 키포인트에 대해 m2-way softmax를 출력에 대한 cross entropy 손실을 최소화함
- K 키 포인트도 독립적으로 처리됨