



Mask R-CNN

Abstract

우리는 object instance segmentation에 대한 개념적으로 간단하고 유연하며 일반적인 프레임워크를 제시한다. 우리는 오브젝트 디텍션과 동시에 객체마다의 세크먼테이션을 진행합니다. Faster R-CNN에서 bounding box recognition을 위한 branch와 병렬적으로 mask 예측 branch를 추가하는식으로 발전된 Mask R-CNN을 사용합니다. Mask R-CNN은 학습하기 평이하며 Faster R-CNN보다 약간의 오버헤드만이 추가됩니다. (5fps) 더욱이 Mask R-CNN은 human pose 추정과 같은 다른 테스크에 적용 가능합니다. Mask R-CNN은 COCO 2016 challenge에서 모든 분야에서 좋은 성능을 보였으며 이와 같은 모델이, 기본이 되어 발전에 기여하기를 바랍니다.

Introduction

컴퓨터 비전은 분야는 object detection과 semantic segmentation을 짧은 기간동안 많은 개선을 시켰습니다. 이러한 발전은 Fast/Fast-RCNN, FCN 프레임워크를 통해서 object detection과 semantic segmentation에 기본 토대를 마련했습니다. 우리의 목적은 이를 통해서 instance segmentation을 가능케 하고자 하는 것입니다.

Instance segmentation은 object detection과 semantic segmentation을 동시에 진행하는 것입니다. object detection의 목표는 객체의 탐지와 클래스 분류이며 semantic segmentation은 픽셀별 분류입니다. 따라서 semantic segmentation은 객체 구분, 클래스 구분, 픽셀별 구분 모두를 수행합니다. 이전에 복잡한 방식에 따라 수행이 가능하지만 우리는 훨씬 빠르며 성능이 좋은 Mask R-CNN을 제시합니다.

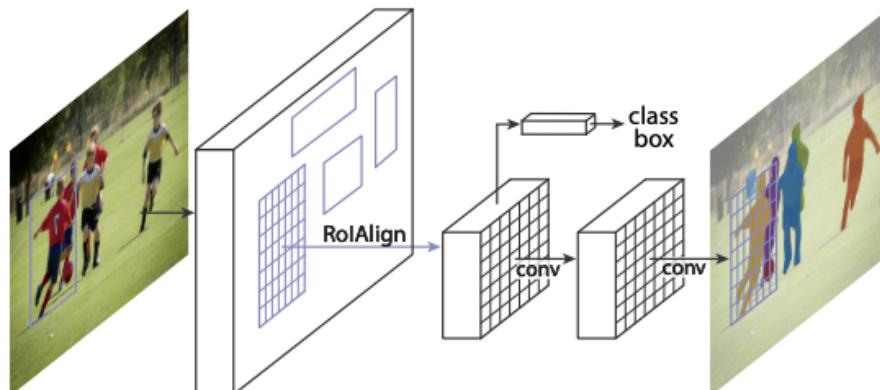


Figure 1. The Mask R-CNN framework for instance segmentation.

Mask R-CNN은 Faster R-CNN에서 Mask branch를 classification과 bounding box regression branch와 병렬로 추가한 것이며 Mask branch는 작은 FCN으로 구성되어집니다.

또한, Faster R-CNN은 Mask branch가 있는 것은 중요하며, 더욱이 Faster R-CNN은 pixel-to-pixel을 위한 alignment가 이루어지지 않습니다. 이를 해결하기 위해 우리는 정확히 공간적 정보를 보존하는 RoIAlign을 제시합니다. 이는 작은 변화로 보이지만 큰 영향을 가집니다.

RoIAlign은 10% ~ 50% 정도의 성능 향상을 일으켰습니다. 두 번째로 우리는 mask와 class prediction을 분리하는 것이 중요하다는 것을 알아냈습니다. 대조적으로 FCN에서는 픽셀별 segmentation과 classification을 동시에 진행하는데 이는 잘 작동되지 않음을 확인했습니다.

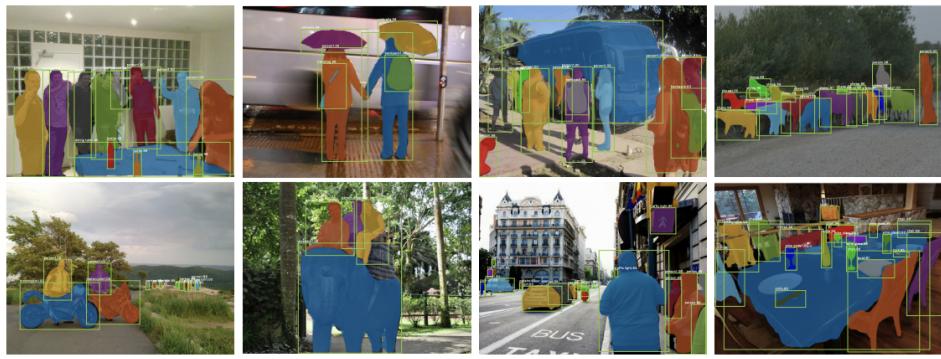


Figure 2. Mask R-CNN results on the COCO test set. These results are based on ResNet-101 [19], achieving a mask AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

긴말 안하고 Mask R-CNN은 짱입니다. 우리의 모델은 COCO 2016competition에서 우승했으며 GPU환경에서 이미지 한 장당 200ms의 속도로 처리합니다. 빠른 학습과 인퍼런스 그리고 좋은 성능을 보입니다.

그리고 COCO keypoint competition에서 약간의 수정된 Mask R-CNN을 통해 human pose estimation에서도 좋은 성능을 보였습니다.

Related Work

R-CNN은 bounding box object detection의 접근법으로 후보 지역을 선정하여 각각의 ROI마다 독립적으로 CNN을 지닙니다. 이는 이후 RoIPool을 통해 더 빠른 스피드와 높은 정확도를 가지게 됩니다. 그리고 더 발전해 Faster R-CNN은 Region Proposal Network를 통해 더 빠른 스피드를 가집니다. 이는 현재의 기본 framework가 되고 있습니다.

R-CNN의 영향으로 instance segmentation에 대한 많은 접근은 segment proposal에 기반합니다. 초기 방식인 DeepMask나 다른 방식은 bottom-up segment로 segment candidates를 선정하는 것을 학습하며 Fast R-CNN을 통해 분류합니다. 이러한 방식은 인식이 segmentation보다 선행되어지며 속도 저하가 생기게 됩니다. 또한, multi-stage cascade 방식은 bounding-box와 classification에서 오는 segment proposal 예측으로 인해 복잡합니다. 이러한 것을 대신해 우리는 mask와 class를 병렬적으로 예측하며 더욱 유연하고 간단합니다.

최근에 segmentation과 object detection이 결합되는 시도인 fully convolution instance segmentation이 이뤄지고 있습니다. 이는 convolution channel을 통해서 class, box, mask를 찾아냅니다. 그러나 FCIS는 오버랩된 인스턴스와 오류를 생성합니다. 다른 방식은 semantic segmentation에서 발전했습니다. 픽셀 별 분류 결과에서 시작하여 인스턴스끼리 분류합니다. 우리의 방식인 Mask R-CNN은 인스턴스 결과로부터 시작하는 방식을 사용합니다. 추후, 두 가지 방식이 통합되는 방식이 나올 것이라 생각합니다.

Mask R-CNN

Mask R-CNN은 개념적으로 간단합니다. Faster R-CNN은 아웃풋으로 오브젝트의 위치와 class label을 내지만, 우리는 여기에 세번째로 mask branch를 추가합니다. 그러나 mask는 세밀한 예측을 요구합니다. 다음으로, pixel-to-pixel alignment를 소개합니다.

Faster R-CNN

간략하게 설명하자면 Faster R-CNN은 two-stages로 이루어져 있습니다. 첫 번째 stage는 Region Proposal Network(RPN)으로 불리며 boxes를 제안합니다. 두번째 stage는 Fast R-CNN에서 왔으며 RoIPool을 이용하여 classification과 bounding box regression을 진행합니다. 합쳐를 양쪽의 stage에서 사용하는 것은 공유될 수 있으며 빠른 인퍼런스가 가능합니다.

Mask R-CNN

Mask R-CNN도 마찬가지로 two-stages로 이루어져 있습니다. 첫 번째 stage는 RPN이며 두 번째 stage는 병렬적으로 mask branch가 추가됩니다. 이전 연구에서 합쳐서 진행된 것과 달리 분리하여 진행하는 것은 큰 간략화를 가져다 주었습니다.

학습을 진행하며 우리는 multi-task loss로 $L = L_{cls} + L_{box} + L_{mask}$ 로 이루어집니다. maks branch의 아웃풋은 Km^2 차원으로 이루어집니다. m은 해상도를 뜻하며 K는 class 수를 뜻합니다. 우리는 픽셀마다의 sigmoid loss를 적용하기 위해서 binary cross-entropy loss의 픽셀 별 평균을 적용합니다. 또한 RoI는 정답 class와만 관련있기 때문에 L_{mask} 는 해당 class에서만 이루어집니다. 우리가 정의한 L_{mask} 는 다른 클래스와 경쟁 없이 모든 클래스에 대해 마스크를 생성할 수 있게 해줍니다. 우리는 그리고 mask의 output을 출력하기 위해 class label 예측값에 의존합니다. 이것이 mask와 clas의 예측을 분리한 이유입니다. 이것은 일반적인 FCNs 적용 방식과 다릅니다.

Mask Representation

마스크는 객체의 공간 정보를 표현합니다. 그래서 클래스 라벨 혹은 박스 위치와 달리 fc layer가 아닌 convolution으로 표현이 필요합니다. 특히 우리는 FCN을 사용해 각각의 ROI로부터 $m \times m$ 의 마스크를 예측합니다. 이는 각각의 레이어가 차원을 유지할 수 있도록 해줍니다. 이러한 방식은 더 적은 연산량을 요구할 뿐 아니라 더 좋은 성능을 보입니다. pixel-to-pixel에 대한 예측은 ROI features가 필요합니다.

RoIAlign

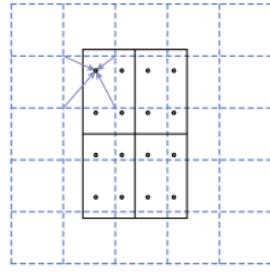


Figure 3. RoIAlign: The dashed grid represents a feature map, the solid lines an ROI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the ROI, its bins, or the sampling points.

RoIPool은 각각의 ROI로부터 작은 피쳐맵을 추출하기 위한 표준 방식입니다. RoIPool은 먼저 ROI에 해당하는 부분을 특정 값으로 변경하고 이를 spatial bins로 분할합니다. 그 후 각각의 bins은 맥스 풀링과 같은 방식으로 계산되어집니다. 이 때 특정 값으로 변경하는 과정에서 불연속적인 숫자의 방식을 사용하는데 이 과정에서 잘못된 할당이 발생하게 됩니다. 이러한 것은 분류에 큰 영향을 주지는 않지만, mask 예측에는 큰 오류로 일어납니다.

이를 해결하기 위해 우리는 RoIAlign을 제안합니다. 기존의 특정 값으로 변경하는 RoIPooling에 방식에서 탈피해서 우리는 bilinear interpolation을 사용해서 값을 정하게 됩니다. RoIAlign은 정확도에 많은 영향을 가져다 줍니다.

Network Architecture

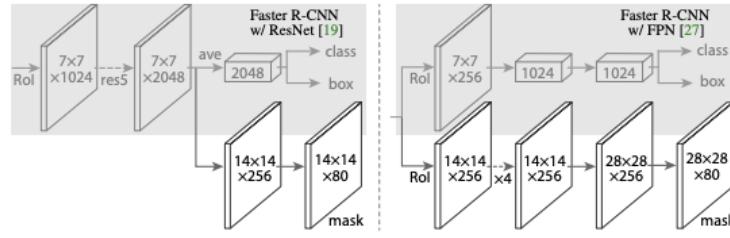


Figure 4. Head Architecture: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively, to which a mask branch is added. Numbers denote spatial resolution and channels. Arrows denote either conv, deconv, or fc layers as can be inferred from context (conv preserves spatial dimension while deconv increases it). All convs are 3×3 , except the output conv which is 1×1 , deconv are 2×2 with stride 2, and we use ReLU [31] in hidden layers. *Left*: ‘res5’ denotes ResNet’s fifth stage, which for simplicity we altered so that the first conv operates on a 7×7 ROI with stride 1 (instead of 14×14 / stride 2 as in [19]). *Right*: ‘ $\times 4$ ’ denotes a stack of four consecutive convs.

우리의 제안 방식을 일반화시키기 위해서 우리는 다음과 같은 두 가지를 다르게 실험합니다.

1. 전체 이미지를 convolution backbone architecture를 통해 피쳐화
2. bounding box recognition, mask prediction separately

backbone network로 Resnet과 FPN을 사용합니다. Resnet을 backbone으로 사용할 경우, res5 layer가 추가되는데 이는, FPN안에는 res5가 포함되어 있기 때문입니다. 이렇게 FPN으로 학습할 경우 조금 더 효율적입니다. 위에 보시다시피 구조는 간단하며, 이러한 구조는 복잡해질수록 성능이 올라갈 것입니다. 하지만, 이 논문의 초점은 그게 아니니 넘어갑니다.

Implementation Details



Figure 5. More results of Mask R-CNN on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).

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
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3
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
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1
Mask R-CNN	ResNeXt-101-FPN	37.1	60.0	39.4	16.9	39.9
						53.5

Table 1. **Instance segmentation mask AP** on COCO test-dev. MNC [10] and FCIS [26] are the winners of the COCO 2015 and 2016 segmentation challenges, respectively. Without bells and whistles, Mask R-CNN outperforms the more complex FCIS++, which includes multi-scale train/test, horizontal flip test, and OHEM [38]. All entries are *single-model* results.

우리는 하이퍼 파라미터의 설정을 Fast/Faster R-CNN과 같이 설정합니다.

학습은 ROI에서 IOU가 0.5이상일 경우 true 그 외는 negative로 설정합니다. 그리고 *L.mask*는 오직 positive에 대해서만 진행되어집니다. 그리고 1:3의 비율로 진행되어집니다. 이미지는 800size로 resize하여 이미지의 센터를 크롭하여 진행합니다.

테스트시에는 proposal number를 300(resnet), 1000(fpn)을 사용합니다. 이 proposal에 대해서 박스 예측을 진행하고 이후 nms를 적용하여 100개의 박스를 남깁니다. ROI에 대해서 k개의 class전부 mask를 예측하며 사용하는 것은 오직 classification branch에서 나온 것에 대해서입니다.

Experiments

학습을 위해 80000개의 이미지 검증을 위해 35000개의 이미지를 사용합니다.

net-depth-features	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.

Backbone network를 변경해가며 실험하였을 경우 FPN이 더 성능이 좋았으며 더 큰 모델일수록 성능이 좋습니다.

	AP	AP ₅₀	AP ₇₅
<i>softmax</i>	24.8	44.1	25.1
<i>sigmoid</i>	30.3	51.2	31.5
	+5.5	+7.1	+6.4

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

mask와 class branch를 결합하고 혹은 결합하지 않았을 때의 비교 실험입니다. 둘이 떨어뜨려 진행하였을 경우 (sigmoid)가 성능이 훨씬 좋은 것을 볼 수 있습니다.

	<th>bilinear?</th> <th>agg.</th> <th>AP</th> <th>AP₅₀</th> <th>AP₇₅</th>	bilinear?	agg.	AP	AP ₅₀	AP ₇₅
<i>RoIPool</i> [12]			max	26.9	48.8	26.4
<i>RoIWarp</i> [10]		✓	max ave	27.2 27.1	49.2 48.9	27.1
<i>RoIAlign</i>	✓	✓	max ave	30.2 30.3	51.0 51.2	31.8 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.

ROI layer의 종류에 따른 실험이며 ROIAlign을 사용했을 때 성능이 훨씬 좋은 것을 확인할 수 있다.

	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

Table 3. Object detection single-model results (bounding box AP), vs. state-of-the-art on test-dev. Mask R-CNN using ResNet-101-FPN outperforms the base variants of all previous state-of-the-art models (the mask output is ignored in these experiments). The gains of Mask R-CNN over [27] come from using ROIAlign (+1.1 AP^{bb}), multitask training (+0.9 AP^{bb}), and ResNeXt-101 (+1.6 AP^{bb}).

또한, segmentation뿐 아니라 object detection에서도 더욱 좋은 성능을 보인다.

Mask R-CNN for Human Pose Estimation



Figure 7. Keypoint detection results on COCO test using Mask R-CNN (ResNet-50-FPN), with person segmentation masks predicted from the same model. This model has a keypoint AP of 63.1 and runs at 5 fps.

우리는 또한 pose estimation이 가능하다. pose estimation까지 추가한다면 box, segment, keypoint를 예측하며 이는 5fps의 속도가 가능하다.