MMDetection

Open MMLab Detection Toolbox and Benchmark (17 Jun 2019)

Abstract

object detection 및 instance segmentation을 다루는 모델을 하나의 toolbox로 표현한 플랫폼 (Coco Challenge 2018 MMDet 팀 우승)

Introduction

1. 모듈화 - 사용자 제작이 간편
2. 여러 가지 형태로 제공되는 Framework가 있다.
3. 높은 효율성. Detectron, maskrcnn-benchmark 등 다른 플랫폼 대비 학습속도 ↑

Model Architecture

Support backbones (7개)

- ResNet(CVPR'2016), ResNeXt(CVPR'2017), VGG(ICLR'2015), ...

Supported methods (42개):

- RPN(NeurIPS'2015), Faster R-CNN(NeurIPS'2015), Cascade R-CNN(CVPR'2018), ...

General Modules and Methods (15개)

- Mixed Precision Training, M2Det, BN, ...

Architecture

- backbone, Neck, DenseHead, RoIExtractor, RoIHead(BBoxHead, MaskHead)

Training Pipeline

: 이미지 로드, 전처리, formatting, test-time augmentation 에 대한 pipeline 설정

link : https://arxiv.org/pdf/1906.07155.pdf

DetectoRS

Detecting Objects with Recursive Feature Pyramid and Switchable Atrous Convolution (30 Nov, 2020)

Abstract

Object Detection의 backbone단계에서 보고 판단하는 두가지 메커니즘을 수행하고자 함.

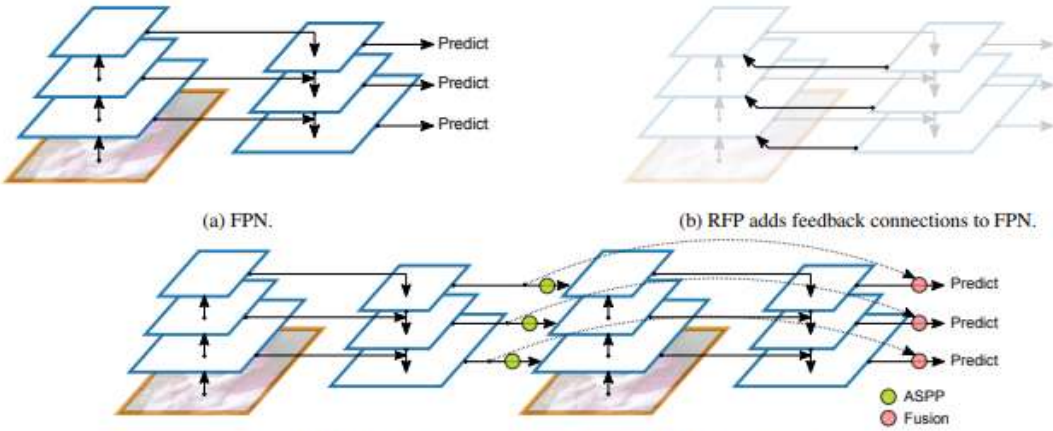
Macro level에서 두 가지 방식이 적용.

1. RFP 방식. RFP(Recursive Feature Pyramid)는 기존 Pyramid 모델 Architecture인 Top-down, Bottom-up 단계에 recursive 단계를 추가하여 초기 pooling 단계에서 중간 단계에 있는 이미지의 경우 Feature 추출이 정확성이 떨어지는 점을 보완하였음.

2. Switchable Atrous Convolution 방식. Switchable Atrous Convolution 방식이란 이미지 Convolution 상황에서 특정 Switchable Function을 적용하여 이미지가 이 함수를 지나쳤을 때, 결과값을 토대로 Convolution 방식을 결정하여 특징을 정함. 이렇게 하면 특징 추출 시 좁은 영역과 넓은 영역을 다른 방식으로 다루게 되므로 좀 더 연역적으로 판단했을 때 일반화된 특징을 잡을 수 있게 됨.

Introduction

인간의 Object detect 단계에서 1차적으로 종류를 파악하고 2차적으로 종류에 대해 생각하는 두 가지 사고가 적용되어 최종적인 퍼포먼스 결과가 나오는 걸 motive로 DetectoRS를 만들었으며 기존의 state-of-art object detector HTC의 퍼포먼스 성능을 넘어서는 결과를 확인하였음.



$$\mathbf{f}_i = \mathbf{F}_i(\mathbf{f}_{i+1}, \mathbf{x}_i), \quad \mathbf{x}_i = \mathbf{B}_i(\mathbf{x}_{i-1}),$$

$$\mathbf{f}_i = \mathbf{F}_i(\mathbf{f}_{i+1}, \mathbf{x}_i), \quad \mathbf{x}_i = \mathbf{B}_i(\mathbf{x}_{i-1}, \mathbf{R}_i(\mathbf{f}_i)), \tag{2}$$

which makes RFP a recursive operation. We unroll it to a sequential network, *i.e.*, $\forall i = 1, ..., S, t = 1, ...T$,

$$\mathbf{f}_i^t = \mathbf{F}_i^t(\mathbf{f}_{i+1}^t, \mathbf{x}_i^t), \ \mathbf{x}_i^t = \mathbf{B}_i^t(\mathbf{x}_{i-1}^t, \mathbf{R}_i^t(\mathbf{f}_i^{t-1})), \tag{3}$$

※

F : F-function in top-down pyramid layer

B : B-function in bottom-up pyramid layer

R : R-function send the result in F-function's result weight (small f)

Fusion Module을 사용. ResNet과 흡사한 방식으로 한 Rotate를 지난 top-down layers의 weight 값을 다음 rotate에 전달함과 동시에 값 자체를 다음 rotate가 끝날 결과와 연결하여 각각의 값들을 activation하여 addition한 결과물이 곧 predict 값이 된다.

Convolution 단계에서 SAC가 계속 사용되고 있는데, 이 SAC layer 앞뒤로 Global Context가 사용된다. 이 layer 층은 받은 값을 Global AvgPool 한 후 Conv (1x1) 층을 지나는 두 번의 연산을 수행하고 받은 값을 저장하여 두 가지 값을 addition하는 작용을 한다. Global Context만 사용했던 SENet은 큰 성능을 내지 못했고 이 층들 차이에 SAC를 적용한 경우 성능 향상이 두드러졌다.

또한 SAC는 단독적으로 사용되지 않고 ResNet backbone에 더해지는 형태로 값을 전달한다. (기본적으로 3x3 convolution layer를 활용한다.) 만일 초기 weight 값과 bias가 0으로 지정되어 있을 경우에는 ResNet과 ResNet + SAC의 성능은 비슷한 지표를 나타냈다.

link : https://arxiv.org/pdf/2006.02334v2.pdf