Project name: Object Detection and Instance Segmentation Using Mask R-CNN

Team member: Li Yueyi(ID:yl7544), Fang Qi(ID:qf281)

Part 1

Introduction to the model

Our research target is an instance segmentation technique, named Mask R-CNN. Mask R-CNN is a technique to detect objects in an image by generating a segmentation mask for each instance.

Mask R-CNN follows the idea of Faster R-CNN, the feature extraction uses the ResNet-FPN architecture, and an additional Mask prediction branch is added. Mask R-CNN can both portray the pixel-level frame outside each instance and display the figure of different individuals in the same category.

The foremost motivation of designing Mask R-CNN is to both realize pixel-level classfication and to recognize every single individual of each class at the same time. In other words, it combines the strengths of FCN and Faster R-CNN.

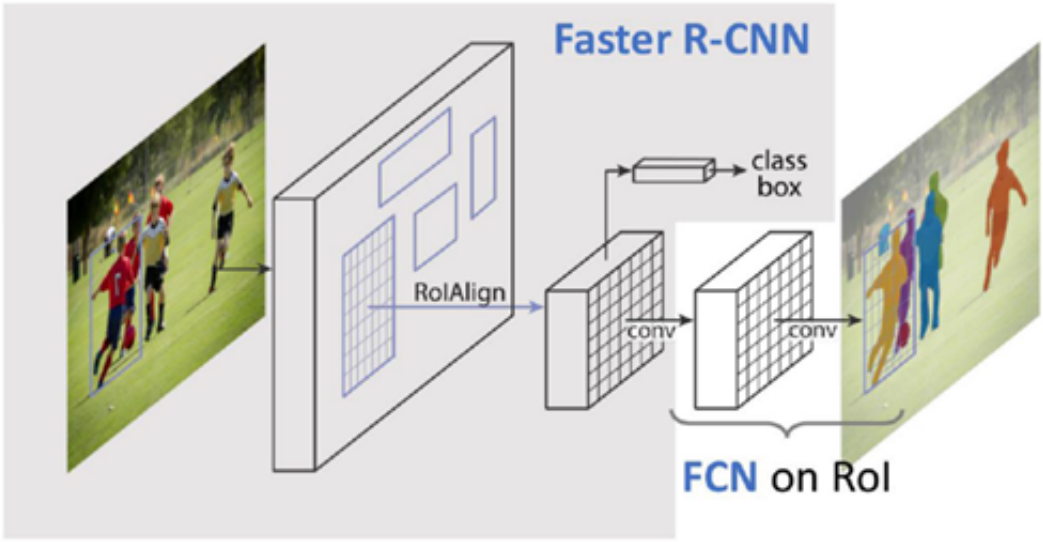


Figure 1: R-CNN= Faster R-CNN with FCN on RoIs

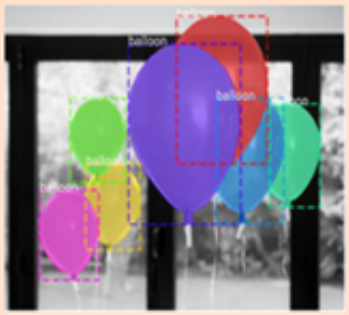
FCN: Fully Convolutional Net	Faster R-CNN	Mask R-CNN(Most powerful technique)
pixel-level classification	non-pixel level classification	pixel-level classification
cannot obtain individual instances of each class	can recognize each individual of all classes	obtaining individual instances of each class
		

Table 1: The advantages of Mask R-CNN over Faster R-CNN and FCN

Details of the model

Here are the steps of Mask R-CNN's procedure. 1). Input the target pictures, and the CNN would extract the features of the target to obtain a feature map. Meanwhile, it goes through the RPN to get the candidate frame. 2). For different candidate frames, we do ROI-Align to get a fixed size feature map. 3). Fully connected layers are used to do the current classification and the regression of the bounding box. 4). A branch parallel to the boudning box regression uses FCN to predict and segment the mask of ROI to get pixel-level frame along the figure of instances.

What makes Mask R-CNN more powerful than Faster R-CNN is the additional branch parallel to the bounding box regression. This branch uses FCN to analyze the segmentation mask of each ROI. Using this branch, we can realize pixel-level classification.

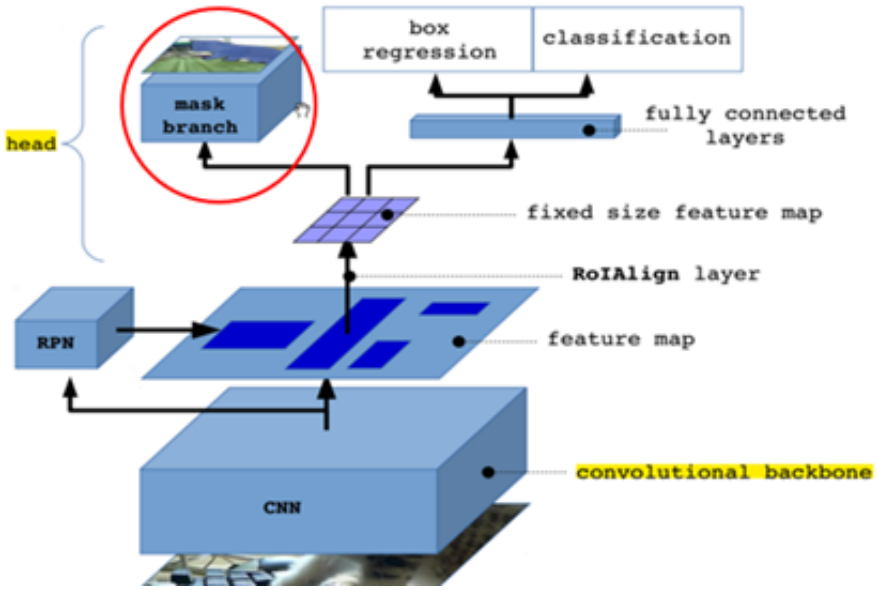


Figure 2: Diagram of how Mask R-CNN works

What is more, to detect instances more accurately, mask rcnn applied RoIAlign to replce RoIPooling when getting a fixed size feature map. Faster R-CNN gets the region of interest by RoI pooling, in which the quantization of the coordinate can cause error in position. To minimize the error, Mask R-CNN avoid quantization to segment instances more precisely.

The mask branch is the FCN(Fully Convolution Net) applied to Region of interest, so we can couple pixel-level classification and segmentation on Mask R-CNN.

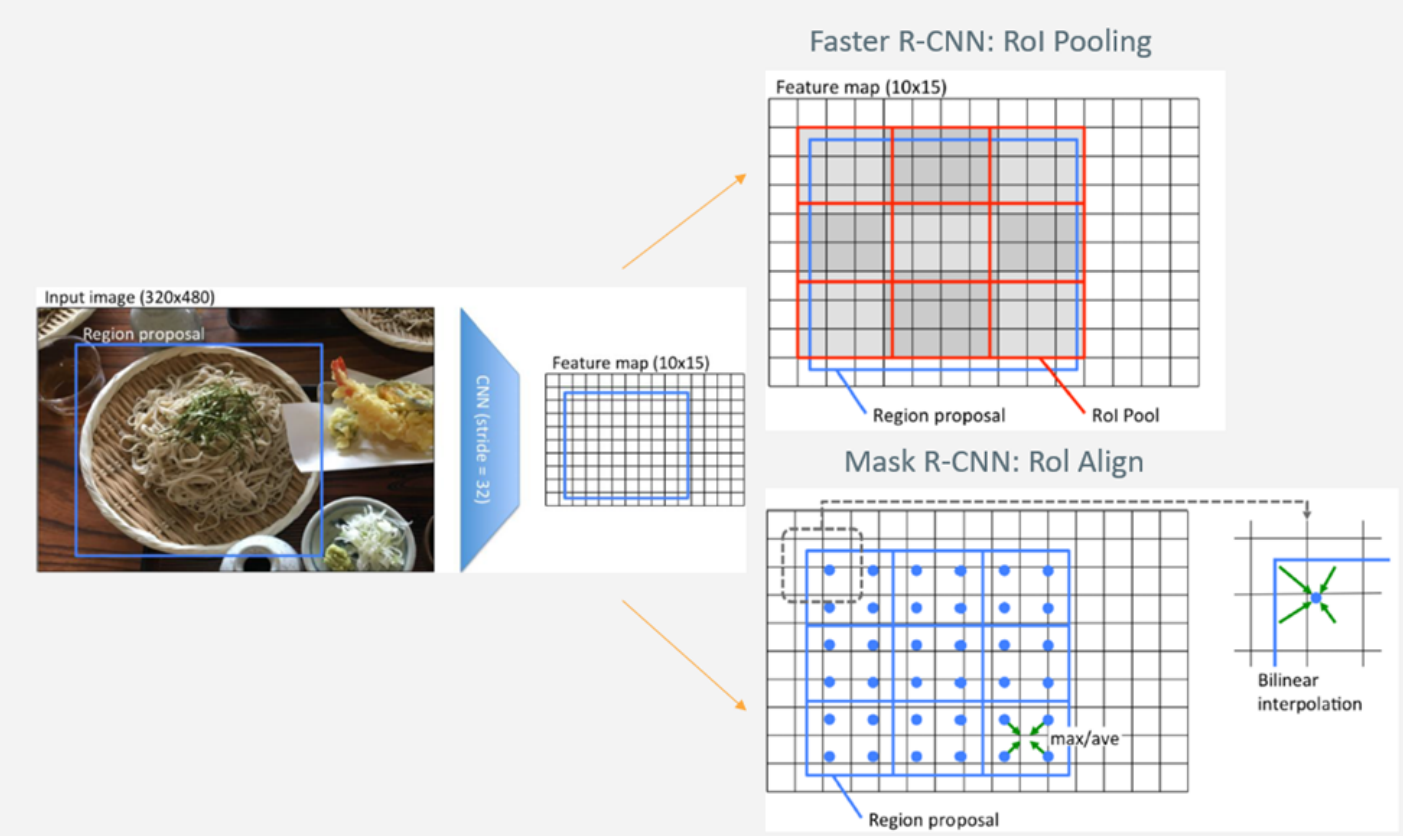


Figure 3: RoI Pooling vs. RoI Align

Details about RoI Align

For each ROI, For each ROI, the coordinates remain floating-point numbers after mapping, and on this basis, they are divided into k times k small areas, which we call bins. At this time, the coordinates still remain floating-point numbers. In each bin, divide the bin into four identical small regions. For each small region, use bilinear interpolation to get the pixel-value. Finally, use Max pooling for four small regions of each bin to get the pixel value.

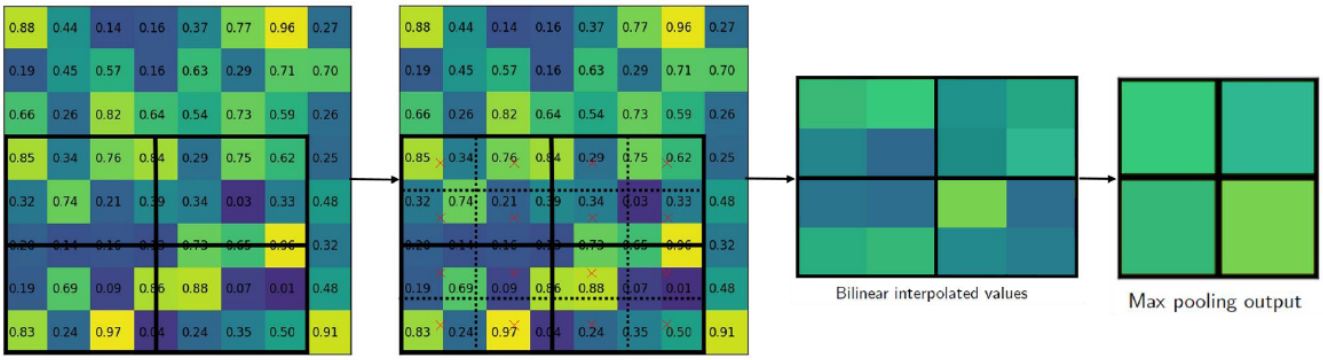


Figure 4: The procedure of RoI Align

The effect of replacing RoI Pooling with RoI Align is to segment the region of interest more precisely, in other words, to improve the value of average precision. We can further verify this conclusion in the upcoming extension work.

Part 2

Existing code

We use the Detectron Mask R-CNN as our baseline code. Detectron is Facebook AI Research's software system that implements state-of-the-art object detection algorithms, including Mask R-CNN. It is written in Python and powered by the Caffe2 deep learning framework. We use this pretrained Mask R-CNN model on the coco dataset.

In the existing code, this traffic picture is the test image, after model testing, we could see the result here, the labels、scores、boxes and mask are shown on the image.

Detectron Mask R-CNN Demo

This is a Mask R-CNN (https://arxiv.org/abs/1703.06870) colab notebook using facebookresearch/Detectron (https://github.com/facebookresearch/Detectron).

For other deep-learning Colab notebooks, visit tugstugi/dl-colab-notebooks (https://github.com/tugstugi/dl-colab-notebooks).

Install Detectron

In [ ]:

```
import os
from os.path import exists, join, basename, splitext

git_repo_url = 'https://github.com/facebookresearch/Detectron.git'
project_name = splitext(basename(git_repo_url))[0]
if not exists(project_name):
  # clone
  !git clone -q --depth 1 $git_repo_url
  # dependencies
  !cd $project_name && pip install -q -r requirements.txt
  # build
  !cd $project_name && make
# test Detectron
!python $project_name/detectron/tests/test_spatial_narrow_as_op.py

import sys
sys.path.append(project_name)
import time
import matplotlib
import matplotlib.pylab as plt
plt.rcParams["axes.grid"] = False

from IPython.display import clear_output
```

Evaluate on a test image

First, download a test image from the internet:

In [ ]:

```
IMAGE_URL = 'https://raw.githubusercontent.com/tugstugi/dl-colab-notebooks/master/resources/traffic_camera.jpg'

image_file = basename(IMAGE_URL)
image_file_ext = splitext(image_file)[1][1:]
images_dir = 'images'
!mkdir -p $images_dir && rm -rf $images_dir/*
!wget -q -O $images_dir/$image_file $IMAGE_URL

plt.figure(figsize=(10, 5))
plt.imshow(matplotlib.image.imread(join(images_dir, image_file)))
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x7ff314403128>
```

According to Detectron/MODEL_ZOO.md (https://github.com/facebookresearch/Detectron/blob/master/MODEL_ZOO.md#end-to-end-faster--mask-r-cnn-baselines), the Mask R-CNN model `X-101-64x4d-FPN` has Box mAP 42.4 and Mask mAP 37.5 on the COCO dataset. We will use this model and evaluate on the above downloaded image:

In [ ]:

```
MODEL_CFG = 'e2e_mask_rcnn_X-101-64x4d-FPN_1x.yaml'
PRETRAINED_MODEL_URL = 'https://dl.fbaipublicfiles.com/detectron/36494496/12_2017_baselines/e2e_mask_rcnn_X-101-64x4d-FPN_1x.yaml.07_50_11.fkwVtEvg/output/train/coco_2014_train%3Acoco_2014_valminusminival/generalized_rcnn/model_final.pkl'
# X-152-32x8d-FPN-IN5k -> OUT OF MEMORY ON COLAB!
# MODEL_CFG = 'e2e_mask_rcnn_X-152-32x8d-FPN-IN5k_1.44x.yaml'
# PRETRAINED_MODEL_URL = 'https://dl.fbaipublicfiles.com/detectron/37129812/12_2017_baselines/e2e_mask_rcnn_X-152-32x8d-FPN-IN5k_1.44x.yaml.09_35_36.8pzTQKYK/output/train/coco_2014_train%3Acoco_2014_valminusminival/generalized_rcnn/model_final.pkl'

!cd $project_name && python tools/infer_simple.py \
    --cfg configs/12_2017_baselines/$MODEL_CFG \
    --wts $PRETRAINED_MODEL_URL \
    --thresh 0.7 \
    --output-dir ../output/ --output-ext png --always-out \
    --image-ext $image_file_ext \
    ../$images_dir
```

Now visualize the result:

In [ ]:

```
plt.figure(figsize=(50, 50))
plt.imshow(matplotlib.image.imread('output/%s.png' % image_file))
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x7ff313ed6710>
```


car 0.76
car 0.89
car 0.71
car 0.90
car 0.89
car 0.97