What is Mask R-CNN?

This article covers five parts of Mask R-CNN:

By Abhijeet Pujara

Jan 05, 2024 06:08 PM · 6 min. read ·
View original

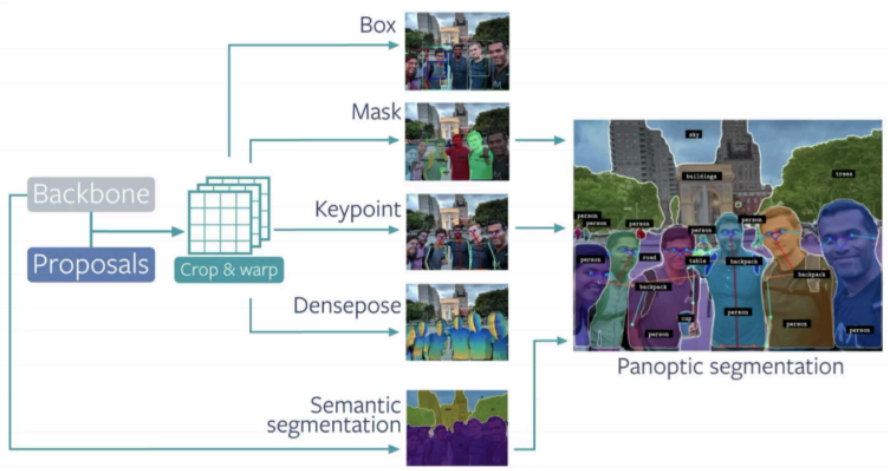


Image source

This article covers five parts of Mask R-CNN:

1. What is Mask R-CNN?
2. Instance and Semantic Segmentation
3. How does Mask R-CNN work?
4. Mask R-CNN With Python
5. Applications, Advantages and Limitations of Mask R-CNN

In this post, I'm assuming that you are comfortable with basic deep learning tasks and models specific to computer vision, such as convolutional neural networks (CNN), image classification, etc. Besides the traditional dog vs. cat classifier that most of us would have built on our deep learning journey, there is a whole lot more we can do with the very same idea of a neural network.

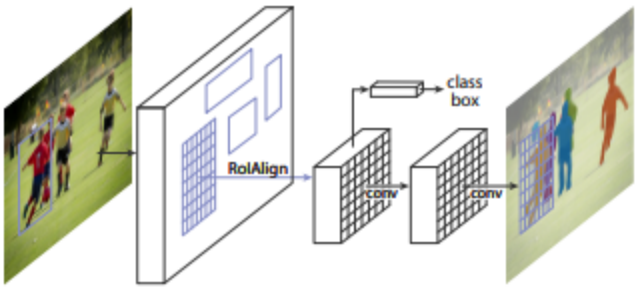
Anyway, in this post, we'll dive into some of the more recent developments in computer vision with deep learning and eventually build up to a model called "Mask R-CNN."

The Mask R-CNN framework is built on top of Faster R-CNN to solve instance segmentation tasks. It is very similar to Faster R-CNN, except there is another layer to predict segmented. The stage of region proposal generation is the same in both architectures; the second stage, which works in parallel, predicts the class and generates a bounding box as well as outputs a binary mask for each RoI. So, let's first quickly understand how faster R-CNN works.

Most importantly, faster R-CNN was not designed for pixel-to-pixel alignment between network inputs and outputs. This is evident in how RoIPool, the *de facto* core operation for attending to instances, performs coarse spatial quantization

for feature extraction. To fix the misalignment, Mask R-CNN utilises a simple, quantization-free layer called RoIAlign, that faithfully preserves exact spatial locations.

Secondly, Mask R-CNN *decouples* mask and class prediction: it predicts a binary mask for each class independently, without competition among classes, and relies on the network's RoI classification branch to predict the category. In contrast, an FCN usually performs per-pixel multi-class categorization, which couples' segmentation and classification.



The first step to understanding how Mask R-CNN works requires an understanding of the concept of image segmentation.

The computer vision task Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects). This segmentation is used to locate objects and boundaries (lines, curves, etc.). There are two main types of image segmentation that fall under Mask R-CNN:

1. Semantic Segmentation
2. Instance Segmentation

Semantic Segmentation

Semantic segmentation classifies each pixel into a fixed set of categories without differentiating object instances. In other words, semantic segmentation deals with the identification and classification of similar objects as a single class at the pixel level.

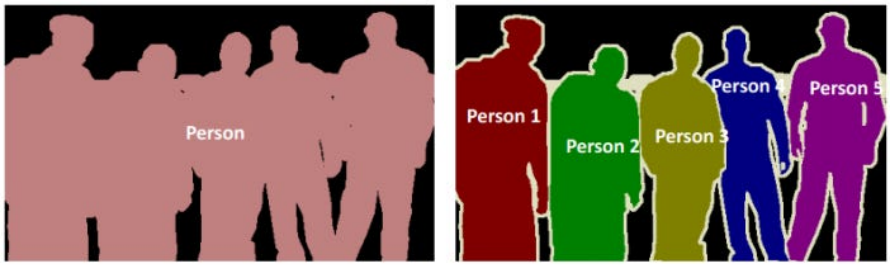
An important thing to note is that semantic segmentation does not highlight individual instances of a class differently. For example, if there were three cows in an image, the model would highlight the area they occupied, but it would not be able to distinguish one cow from another. If we want to add this functionality, we need to extend the task and introduce another term to complicate the already enormously large vocabulary of deep learning—instance segmentation.

Ok, it wasn't really all that bad, was it? The term is pretty much self-explanatory. Our goal is to segment or separate each "instance" of a class in an image. This should help you visualize what we are trying to achieve:

Instance Segmentation

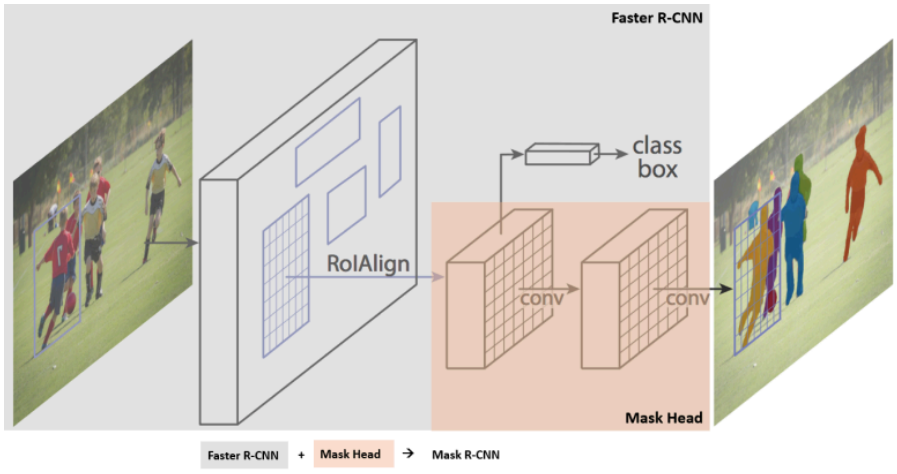
Instance segmentation, or instance recognition, deals with the correct detection of all objects in

an image while also precisely segmenting each instance. In other words, this type of segmentation goes further to give a clear distinction between each object classified as having similar instances. As shown in the example image, for instance, in segmentation, all objects are persons, but this segmentation process separates each person as a single entity. Semantic segmentation is otherwise known as foreground segmentation because it accentuates the subjects of the image instead of the background.



Semantic Segmentation Instance Segmentation

How does Mask R-CNN work?



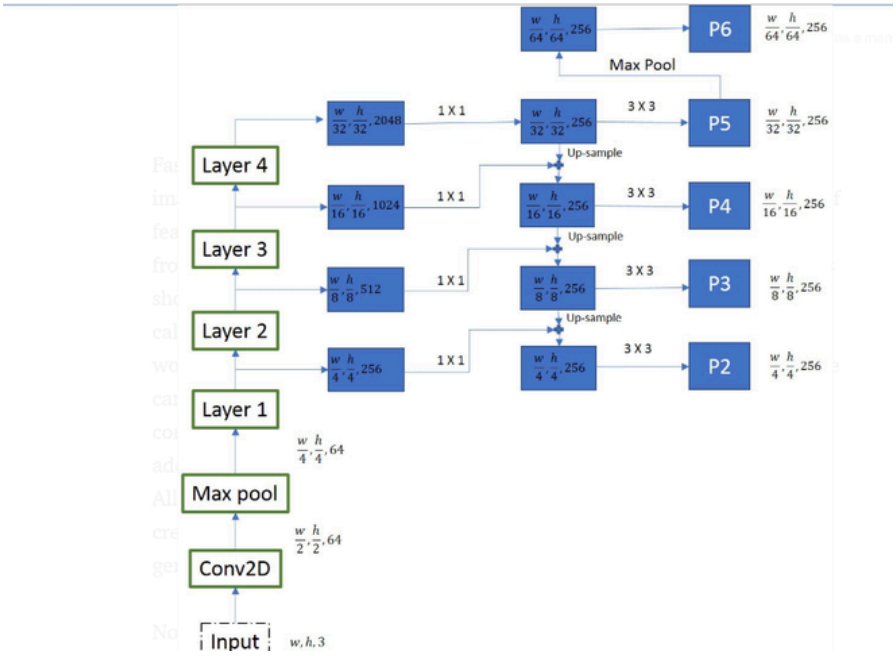
Mask R-CNN was built using Faster R-CNN. While Faster R-CNN has two outputs for each candidate

object: a class label and a bounding box offset, Mask R-CNN has the addition of a third branch that outputs the object mask. The additional mask output is distinct from the class and box outputs, requiring the extraction of a much finer spatial layout of an object.

It consists of:

Backbone Network

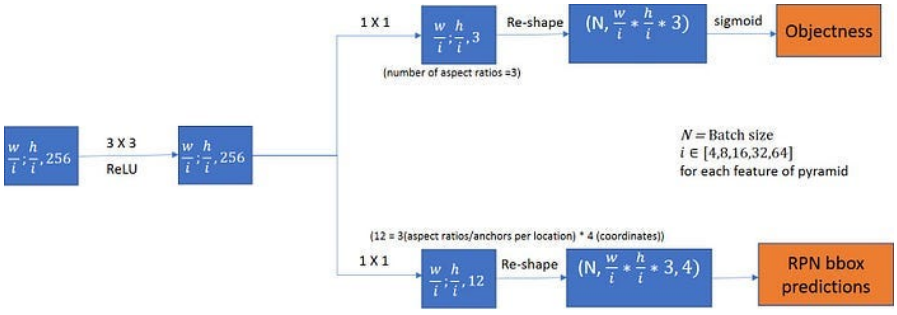
The authors of Mask R-CNN experimented with two kinds of backbone networks. The first is the standard ResNet architecture (ResNet-C4), and the other is ResNet with a feature pyramid network. The standard ResNet architecture was similar to that of Faster R-CNN, but ResNet-FPN has proposed some modifications. This consists of a multi-layer RoI generation. This multi-layer feature pyramid network generates RoIs of different scales, which improves the accuracy of the previous ResNet architecture.



Mask R-CNN backbone architecture

Region Proposal Network (RPN)

The RPN is responsible for generating region proposals or candidate bounding boxes that might contain objects within the image. It operates on the feature map produced by the backbone network and proposes potential regions of interest.



Region Proposal Network (RPN).

ROIAlign

After the RPN generates region proposals, the ROIAlign (Region of Interest Align) layer is

introduced. This step helps to overcome the misalignment issue in ROI pooling.

ROIAlign plays a crucial role in accurately extracting features from the input feature map for each region proposal, ensuring precise pixel-wise segmentation in instance segmentation tasks.

The primary purpose of ROIAlign is to align the features within a region of interest (ROI) with the spatial grid of the output feature map. This alignment is crucial to prevent information loss that can occur when quantizing the ROI's spatial coordinates to the nearest integer (as done in ROI pooling).

ROIAlign operation.

The ROIAlign process involves the following steps:

1. **Input Feature Map**: The process begins with the input feature map, which is typically obtained from the backbone network. This feature map contains high-level semantic information about the entire image.
2. **Region Proposals**: The Region Proposal Network (RPN) generates region proposals (candidate bounding boxes) that might contain objects of interest within the image.
3. **Dividing into Grids**: Each region proposal is divided into a fixed number of equal-sized spatial bins, or grids. These grids are used to extract features from the input feature map corresponding to the region of interest.
4. **Bilinear Interpolation**: Unlike ROI pooling, which quantizes the spatial coordinates of the

grids to the nearest integer, ROIAlign uses bilinear interpolation to calculate the pooling contributions for each grid. This interpolation ensures a more precise alignment of the features within the ROI.

5. **Output Features**: The features obtained from the input feature map, aligned with each grid in the output feature map, are used as the representative features for each region proposal. These aligned features capture fine-grained spatial information, which is crucial for accurate segmentation.

By using bilinear interpolation during the pooling process, ROIAlign significantly improves the accuracy of feature extraction for each region proposal, mitigating misalignment issues.

This precise alignment enables Mask R-CNN to generate more accurate segmentation masks, especially for small objects or regions that require fine details to be preserved. As a result, ROIAlign contributes to the strong performance of Mask R-CNN in instance segmentation tasks.

Mask R-CNN With Python

```
import sys
import random
import math
import numpy as np
import skimage.io
import matplotlib
import matplotlib.pyplot as plt

# Root directory of the project
ROOT_DIR = os.path.abspath("../")

# Import Mask RCNN
sys.path.append(ROOT_DIR)  # To find local version of the library
from mrcnn import utils
import mrcnn.model as modellib
from mrcnn import visualize
# Import COCO config
sys.path.append(os.path.join(ROOT_DIR, "samples/coco/"))  # To find local version
import coco

%matplotlib inline

# Directory to save logs and trained model
MODEL_DIR = os.path.join(ROOT_DIR, "logs")

# Local path to trained weights file
COCO_MODEL_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")
# Download COCO trained weights from Releases if needed
if not os.path.exists(COCO_MODEL_PATH):
    utils.download_trained_weights(COCO_MODEL_PATH)

# Directory of images to run detection on
IMAGE_DIR = os.path.join(ROOT_DIR, "images")
```

Applications of Mask R-CNN

Due to its additional capability to generate segmented masks, it is used in many computer vision applications, such as:

Advantages of Mask R-CNN

Mask R-CNN Limitations

Conclusion

Mask R-CNN merges object detection and instance segmentation, providing the capability to not only detect objects but also to precisely delineate their boundaries at the pixel level. By using a feature pyramid network (FPN) and the region of interest align (ROIAlign), Mask R-CNN achieves strong performance and accuracy.

Mask R-CNN does have certain limitations, such as its computational complexity and memory usage during training and inference. It may encounter challenges in accurately segmenting very small objects or handling heavily occluded scenes. Acquiring a substantial amount of annotated training data can also be demanding, and fine-tuning the model for specific domains may require careful parameter tuning.

> *Mask R-CNN Paper*