

# Mask R-CNN 详解

## 一、Mask R-CNN 是什么，可以做哪些任务？

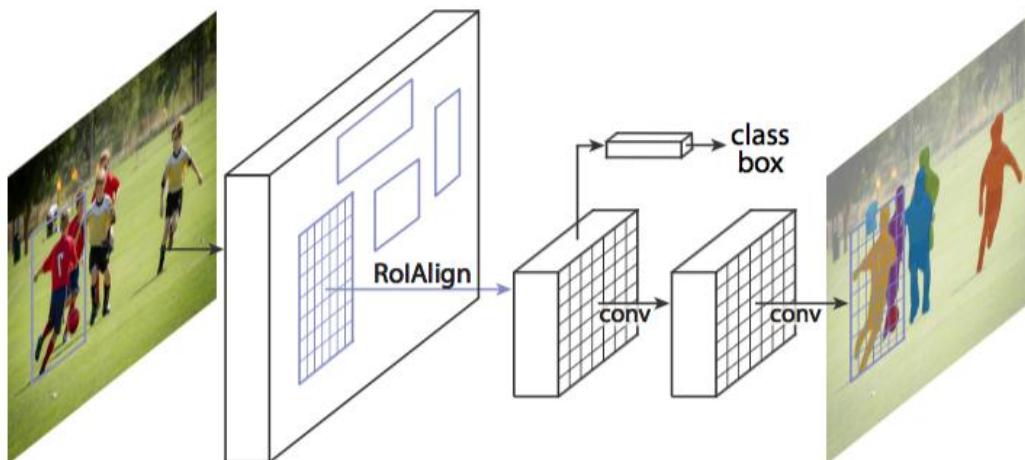


图 1 Mask R-CNN 整体架构

Mask R-CNN 是一个实例分割（Instance segmentation）算法，可以用来做“目标检测”、“目标实例分割”、“目标关键点检测”。

### 1. 实例分割（Instance segmentation）和语义分割（Semantic segmentation）的区别与联系

联系：语义分割和实例分割都是目标分割中的两个小的领域，都是用来对输入的图片做分割处理；

区别：

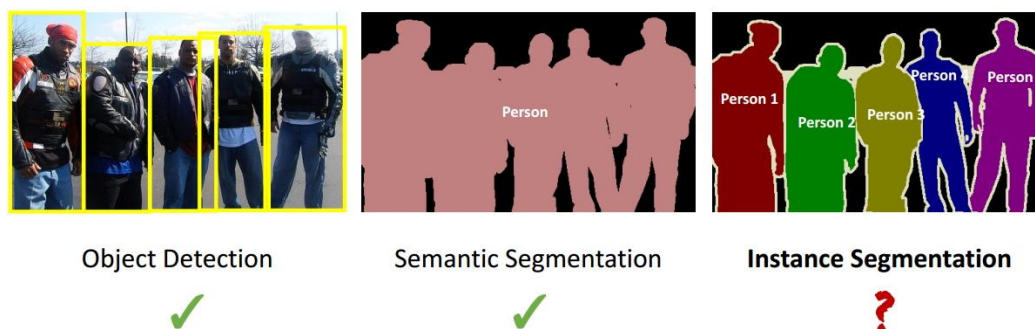


图 2 实例分割与语义分割区别

1.1 通常意义上的目标分割指的是语义分割，语义分割已经有很长的发展历史，已经取得了很好地进展，目前有很多的学者在做这方面的研究；然而实例分割是一个从目标分割领域独立出来的一个小领域，是最近几年才发展起来的，与前者相比，后者更加复杂，当前研究的学者也比较少，是一个有研究空间的热门领域，如图 1 所示，这是一个正在探索中的领域；

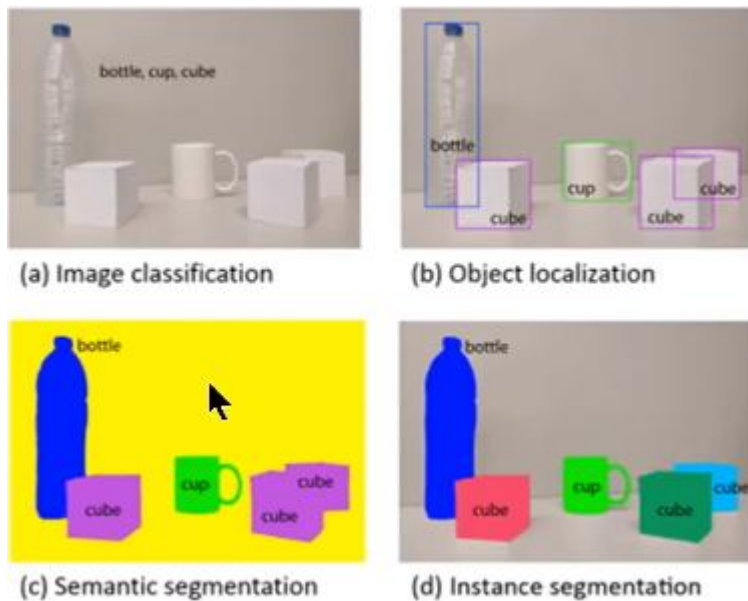


图 3 实例分割与语义分割区别

1.2 观察图 3 中的 c 和 d 图，c 图是对 a 图进行语义分割的结果，d 图是对 a 图进行实例分割的结果。两者最大的区别就是图中的"cube 对象"，在语义分割中给了它们相同的颜色，而在实例分割中却给了不同的颜色。即实例分割需要在语义分割的基础上对同类物体进行更精细的分割。

注：很多博客中都没有完全理解清楚这个问题，很多人将这个算法看做语义分割，其实它是一个实例分割算法。

## 2. Mask R-CNN 可以完成的任务

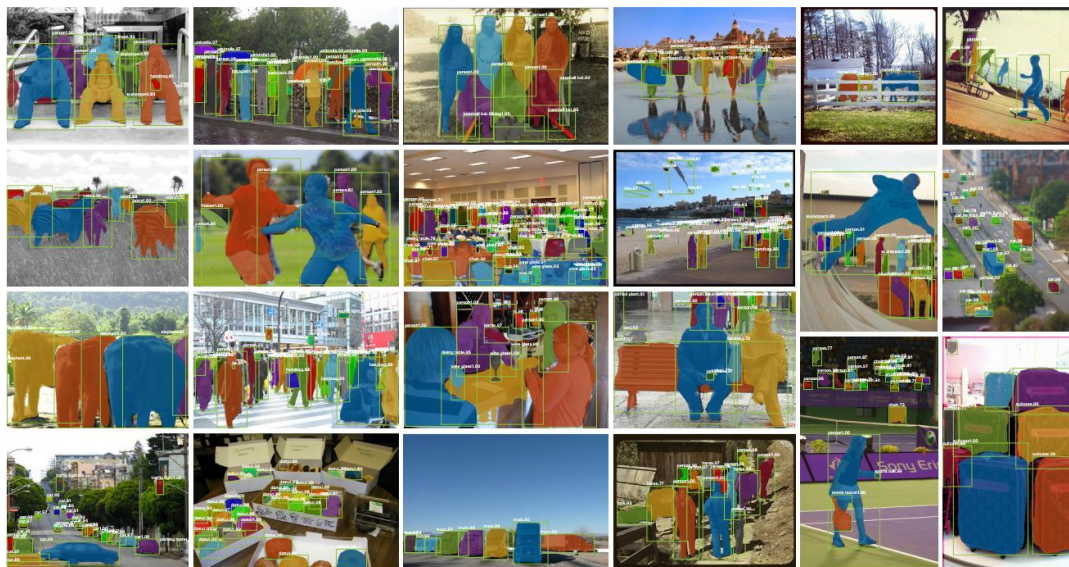


图 4 Mask R-CNN 进行目标检测与实例分割



图 5 Mask R-CNN 进行人体姿态识别

总之，Mask R-CNN 是一个非常灵活的框架，可以增加不同的分支完成不同的任务，可以完成目标分类、目标检测、语义分割、实例分割、人体姿势识别等多种任务，真不愧是一个好算法！

### 3. Mask R-CNN 预期达到的目标

高速

高准确率（高的分类准确率、高的检测准确率、高的实例分割准确率等）

简单直观

易于使用

### 4. 如何实现这些目标

**高速和高准确率：**为了实现这个目的，作者选用了经典的目标检测算法 Faster-rcnn 和经典的语义分割算法 FCN。Faster-rcnn 可以既快又准的完成目标检测的功能；FCN 可以精准的完成语义分割的功能，这两个算法都是对应领域中的经典之作。Mask R-CNN 比 Faster-rcnn 复杂，但是最终仍然可以达到 5fps 的速度，这和原始的 Faster-rcnn 的速度相当。由于发现了 ROI Pooling 中所存在的像素偏差问题，提出了对应的 ROIAlign 策略，加上 FCN 精准的像素 MASK，使得其可以获得高准确率。

**简单直观：**整个 Mask R-CNN 算法的思路很简单，就是在原始 Faster-rcnn 算法的基础上面增加了 FCN 来产生对应的 MASK 分支。即 Faster-rcnn + FCN，更细致的是 RPN + ROIAlign + Fast-rcnn + FCN。

**易于使用：**整个 Mask R-CNN 算法非常的灵活，可以用来完成多种任务，包括目标分类、目标检测、语义分割、实例分割、人体姿态识别等多个任务，这将其易于使用的特点展现的淋漓尽致。我很少见到有哪个算法有这么好的扩展性和易用性，值得我们学习和借鉴。除此之外，我们可以更换不同的 backbone architecture 和 Head Architecture 来获得不同性能的结果。

## 二、Mask R-CNN 框架解析

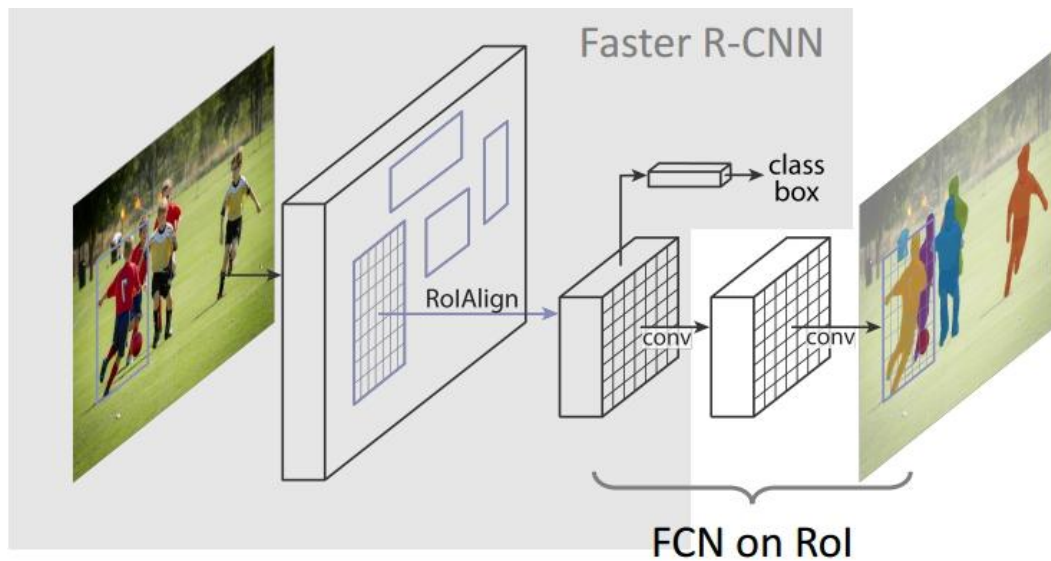


图 6 Mask R-CNN 算法框架

### 1. Mask R-CNN 算法步骤

- 1.1 首先，输入一幅你想处理的图片，然后进行对应的预处理操作，或者预处理后的图片；
- 1.2 然后，将其输入到一个预训练好的神经网络中（ResNeXt 等）获得对应的 feature map；
- 1.3 接着，对这个 feature map 中的每一点设定预定个的 ROI，从而获得多个候选 ROI；
- 1.4 接着，将这些候选的 ROI 送入 RPN 网络进行二值分类（前景或背景）和 BB 回归，过滤掉一部分候选的 ROI；
- 1.5 接着，对这些剩下的 ROI 进行 ROIAlign 操作（即先将原图和 feature map 的 pixel 对应起来，然后将 feature map 和固定的 feature 对应起来）；
- 1.6 最后，对这些 ROI 进行分类（N 类别分类）、BB 回归和 MASK 生成（在每一个 ROI 里面进行 FCN 操作）。

### 2. Mask R-CNN 架构分解

在这里，我将 Mask R-CNN 分解为如下的 3 个模块，Faster-rcnn、ROIAlign 和 FCN。然后分别对这 3 个模块进行讲解，这也是该算法的核心。

### 3. Faster-rcnn（该算法请参考该链接，我进行了详细的分析）



#### 4. FCN

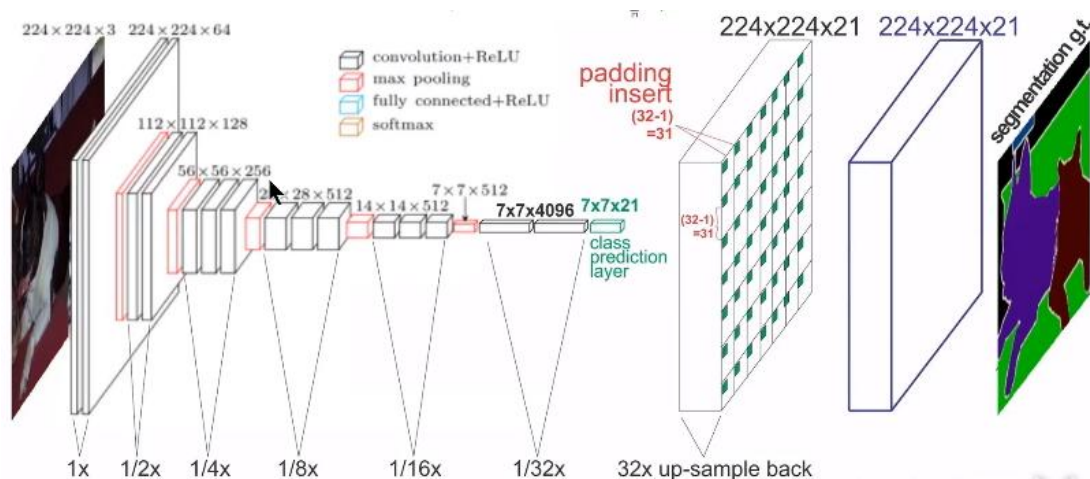


图 7 FCN 网络架构

FCN 算法是一个经典的语义分割算法，可以对图片中的目标进行准确的分割。其总体架构如上图所示，它是一个端到端的网络，主要的模块包括卷积和去卷积，即先对图像进行卷积和池化，使其 feature map 的大小不断减小；然后进行反卷积操作，即进行插值操作，不断的增大其 feature map，最后对每一个像素值进行分类。从而实现对输入图像的准确分割。具体的细节请参考该链接。

#### 5. ROI Pooling 和 ROIAlign 的分析与比较

- RoIPool breaks pixel-to-pixel translation-equivariance

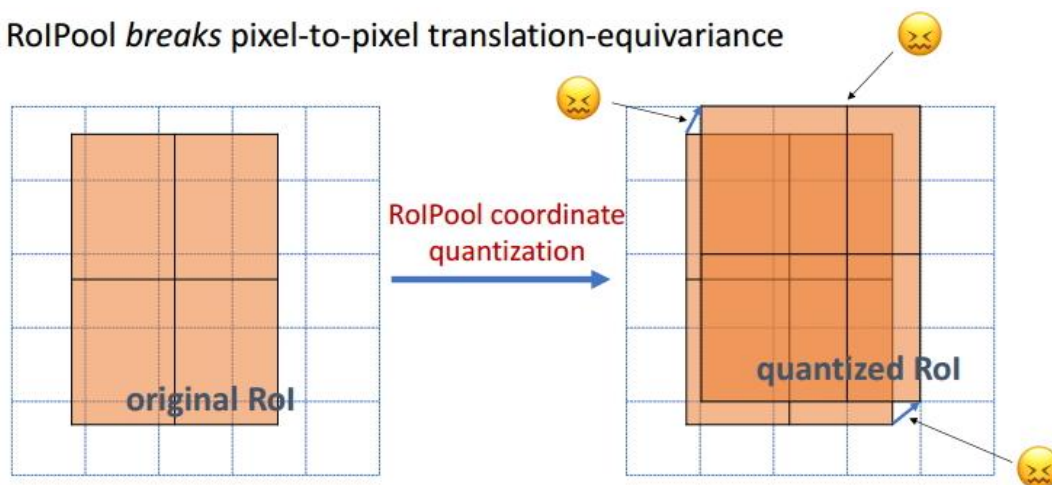


图 8 ROI Pooling 和 ROIAlign 的比较

如图 8 所示，ROI Pooling 和 ROIAlign 最大的区别是：前者使用了两次量化操作，而后者并没有采用量化操作，使用了线性插值算法，具体的解释如下所示。

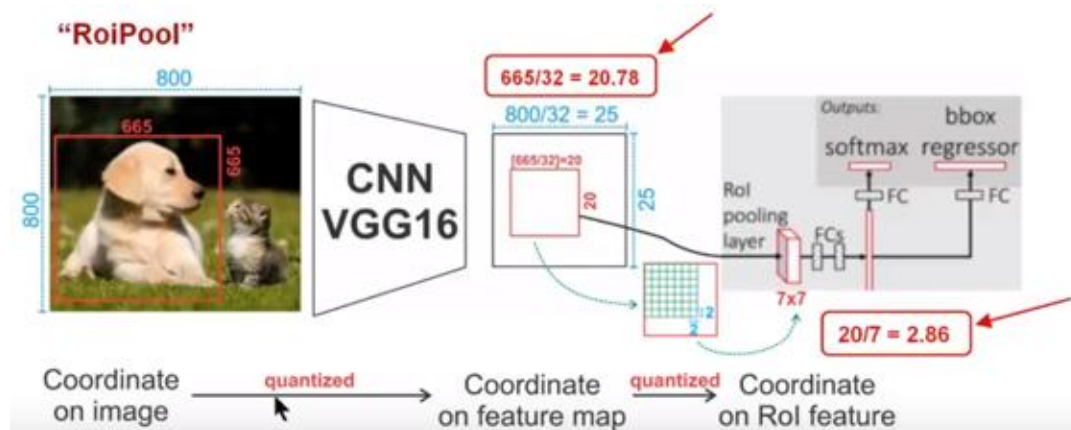


图 9 ROI Pooling 技术

如图 9 所示，为了得到固定大小（7x7）的 feature map，我们需要做两次量化操作：1）图像坐标 — feature map 坐标，2）feature map 坐标 — ROI feature 坐标。我们来说一下具体的细节，如图我们输入的是一张 800x800 的图像，在图像中有两个目标（猫和狗），狗的 BB 大小为 665x665，经过 VGG16 网络后，我们可以获得对应的 feature map，如果我们对卷积层进行 Padding 操作，我们的图片经过卷积层后保持原来的大小，但是由于池化层的存在，我们最终获得 feature map 会比原图缩小一定的比例，这和 Pooling 层的个数和大小有关。在该 VGG16 中，我们使用了 5 个池化操作，每个池化操作都是 2Pooling，因此我们最终获得 feature map 的大小为  $800/32 \times 800/32 = 25 \times 25$ （是整数），但是将狗的 BB 对应到 feature map 上面，我们得到的结果是  $665/32 \times 665/32 = 20.78 \times 20.78$ ，结果是浮点数，含有小数，但是我们的像素值可没有小数，那么作者就对其进行了量化操作（即取整操作），即其结果变为  $20 \times 20$ ，在这里引入了第一次的量化误差；然而我们的 feature map 中有不同大小的 ROI，但是我们后面的网络却要求我们有固定的输入，因此，我们需要将不同大小的 ROI 转化为固定的 ROI feature，在这里使用的是 7x7 的 ROI feature，那么我们需要将  $20 \times 20$  的 ROI 映射成  $7 \times 7$  的 ROI feature，其结果是  $20/7 \times 20/7 = 2.86 \times 2.86$ ，同样是浮点数，含有小数点，我们采取同样的操作对其进行取整吧，在这里引入了第二次量化误差。其实，这里引入的误差会导致图像中的像素和特征中的像素的偏差，即将 feature 空间的 ROI 对应到原图上面会出现很大的偏差。原因如下：比如用我们第二次引入的误差来分析，本来是 2.86，我们将其量化为 2，这期间引入了 0.86 的误差，看起来是一个很小的误差呀，但是你要记得这是在 feature 空间，我们的 feature 空间和图像空间是有比例关系的，在这里是 1:32，那么对应到原图上面的差距就是  $0.86 \times 32 = 27.52$ 。这个差距不小吧，这还是仅仅考虑了第二次的量化误差。这会大大影响整个检测算法的性能，因此是一个严重的问题。好的，应该解释清楚了吧，好累！

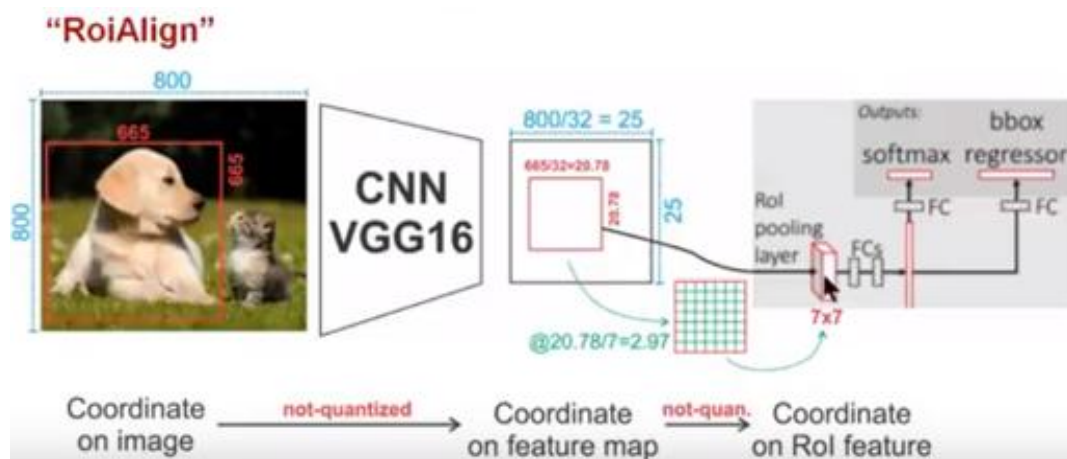


图 10 ROIAlign 技术

如图 10 所示，为了得到固定大小（7x7）的 feature map，ROIAlign 技术并没有使用量化操作，即我们不想引入量化误差，比如  $665 / 32 = 20.78$ ，我们就用 20.78，不用什么 20 来替代它，比如  $20.78 / 7 = 2.97$ ，我们就用 2.97，而不用 2 来替代它。这就是 ROIAlign 的初衷。那么我们如何处理这些浮点数呢，我们的解决思路是使用“双线性插值”算法。双线性插值是一种比较好的图像缩放算法，它充分的利用了原图中虚拟点（比如 20.56 这个浮点数，像素位置都是整数，没有浮点值）四周的四个真实存在的像素值来共同决定目标图中的一个像素值，即可以将 20.56 这个虚拟的位置点对应的像素值估计出来。厉害哈。如图 11 所示，蓝色的虚线框表示卷积后获得的 feature map，黑色实线框表示 ROI feature，最后需要输出的大小是 2x2，那么我们就利用双线性插值来估计这些蓝点（虚拟坐标点，又称双线性插值的网格点）处所对应的像素值，最后得到相应的输出。这些蓝点是 2x2Cell 中的随机采样的普通点，作者指出，这些采样点的个数和位置不会对性能产生很大的影响，你也可以用其它的方法获得。然后在每一个橘红色的区域里面进行 max pooling 或者 average pooling 操作，获得最终 2x2 的输出结果。我们的整个过程中没有用到量化操作，没有引入误差，即原图中的像素和 feature map 中的像素是完全对齐的，没有偏差，这不仅可以提高检测的精度，同时也会有利于实例分割。这么细心，做科研就应该关注细节，细节决定成败。

we propose an RoIAlign layer that removes the harsh quantization of RoIPool, properly aligning the extracted features with the input. Our proposed change is simple: we avoid any quantization of the RoI boundaries or bins (i.e., we use  $x=16$  instead of  $[x=16]$ ). We use bilinear interpolation [22] to compute the exact values of the input features at four regularly sampled locations in each RoI bin, and aggregate the result (using max or average), see Figure 3 for details. We note that the results are not sensitive to the exact sampling locations, or how many points are sampled, as long as no quantization is performed.

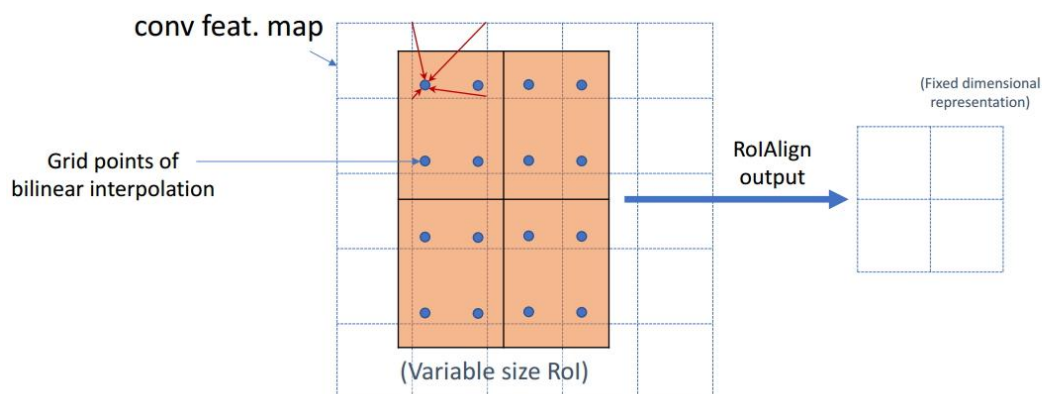


图 11 双线性插值

## 6. LOSS 计算与分析

由于增加了 mask 分支，每个 ROI 的 Loss 函数如下所示：

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

其中  $L_{cls}$  和  $L_{box}$  和 Faster r-cnn 中定义的相同。对于每一个 ROI，mask 分支有  $Km \times m$  维度的输出，其对  $K$  个大小为  $m \times m$  的 mask 进行编码，每一个 mask 有  $K$  个类别。我们使用了 per-pixel sigmoid，并且将  $L_{mask}$  定义为 the average binary cross-entropy loss。对应一个属于 GT 中的第  $k$  类的 ROI， $L_{mask}$  仅仅在第  $k$  个 mask 上面有定义（其它的  $k-1$  个 mask 输出对整个 Loss 没有贡献）。我们定义的  $L_{mask}$  允许网络为每一类生成一个 mask，而不用和其它类进行竞争；我们依赖于分类分支所预测的类别标签来选择输出的 mask。这样将分类和 mask 生成分解开来。这与利用 FCN 进行语义分割的有所不同，它通常使用一个 per-pixel sigmoid 和一个 multinomial cross-entropy loss，在这种情况下 mask 之间存在竞争关系；而由于我们使用了一个 per-pixel sigmoid 和一个 binary loss，不同的 mask 之间不存在竞争关系。经验表明，这可以提高实例分割的效果。

一个 mask 对一个目标的输入空间布局进行编码，与类别标签和 BB 偏置不同，它们通常需要通过 FC 层而导致其以短向量的形式输出。我们可以通过由卷积提供的像素和像素的对应关系来获得 mask 的空间结构信息。具体的来说，我们使用 FCN 从每一个 ROI 中预测出一个  $m \times m$  大小的 mask，这使得 mask 分支中的每个层能够明确的保持  $m \times m$  空间布局，而不将其折叠成缺少空间维度的向量表示。和以前用 fc 层做 mask 预测的方法不同的是，我们的实验表明我们的 mask 表示需要更少的参数，而且更加准确。这些像素到像素的行为需要我们的 ROI 特征，而我们的 ROI 特征通常是比较小的 feature map，其已经进行了对其操作，为了一致的较好的保持明确的单像素空间对应关系，我们提出了 ROIAlign 操作。



### 三、Mask R-CNN 细节分析

#### 1. Head Architecture

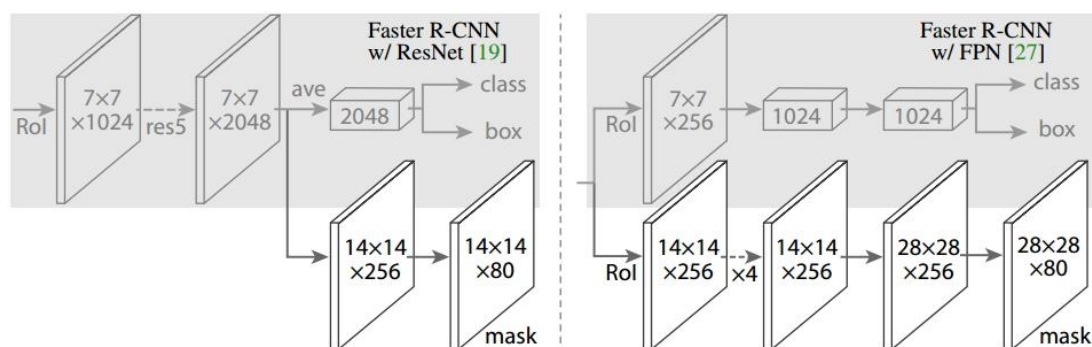


图 12 Head Architecture

如上图所示，为了产生对应的 Mask，文中提出了两种架构，即左边的 Faster R-CNN/ResNet 和右边的 Faster R-CNN/FPN。对于左边的架构，我们的 backbone 使用的是预训练好的 ResNet，使用了 ResNet 倒数第 4 层的网络。输入的 ROI 首先获得 7x7x1024 的 ROI feature，然后将其升维到 2048 个通道（这里修改了原始的 ResNet 网络架构），然后有两个分支，上面的分支负责分类和回归，下面的分支负责生成对应的 mask。由于前面进行了多次卷积和池化，减小了对应的分辨率，mask 分支开始利用反卷积进行分辨率的提升，同时减少通道的个数，变为 14x14x256，最后输出了 14x14x80 的 mask 模板。而右边使用到的 backbone 是 FPN 网络，这是一个新的网络，通过输入单一尺度的图片，最后可以对应的特征金字塔，如果想要了解它的细节，请参考该链接。得到证实的是，该网络可以在一定程度上提高检测的精度，当前很多的方法都用到了它。由于 FPN 网络已经包含了 res5，可以更加高效的使用特征，因此这里使用了较少的 filters。该架构也分为两个分支，作用于前者相同，但是分类分支和 mask 分支和前者相比有很大的区别。可能是因为 FPN 网络可以在不同尺度的特征上面获得许多有用信息，因此分类时使用了更少的滤波器。而 mask 分支中进行了多次卷积操作，首先将 ROI 变化为 14x14x256 的 feature，然后进行了 5 次相同的操作（不清楚这里的原理，期待着你的解释），然后进行反卷积操作，最后输出 28x28x80 的 mask。即输出了更大的 mask，与前者相比可以获得更细致的 mask。

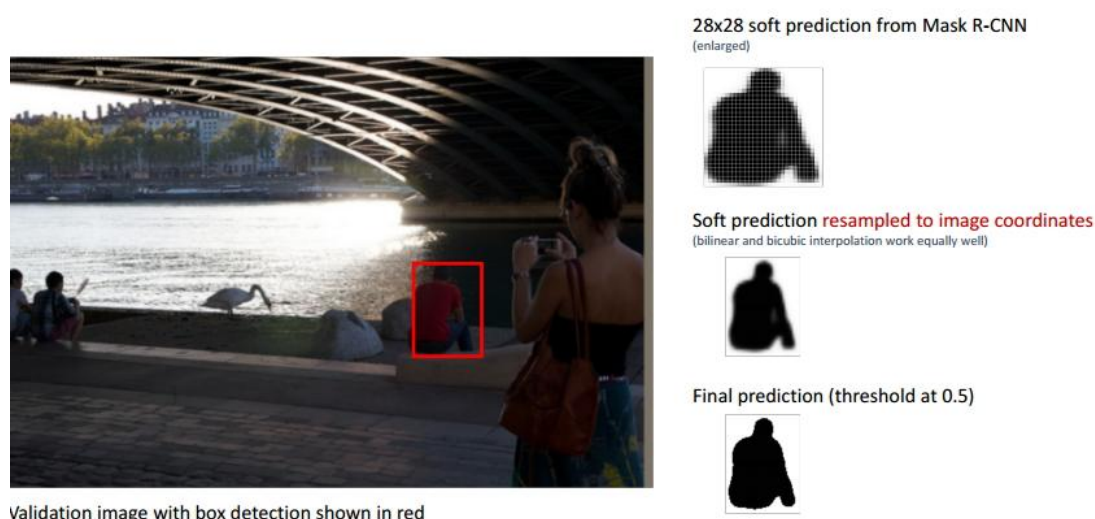


图 13 BB 输出的 mask 结果

如上图所示，图像中红色的 BB 表示检测到的目标，我们可以用肉眼可以观察到检测结果并不是很好，即整个 BB 稍微偏右，左边的一部分像素并没有包括在 BB 之内，但是右边显示的最终结果却很完美。

**2. Equivariance in Mask R-CNN**

Equivariance 指随着输入的变化输出也会发生变化。

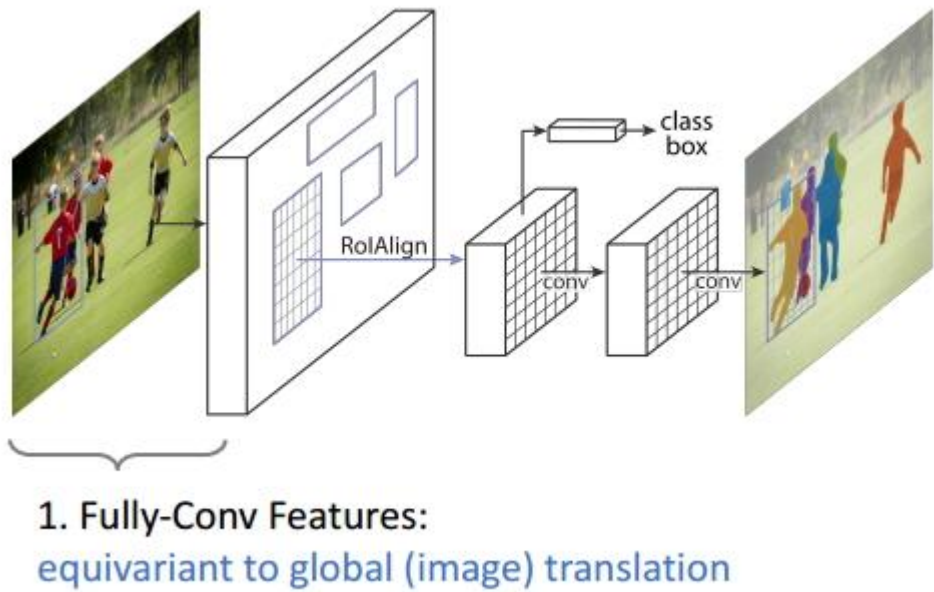


图 14 Equivariance 1

即全卷积特征（Faster R-CNN 网络）和图像的变换具有同变形，即随着图像的变换，全卷积的特征也会发生对应的变化；

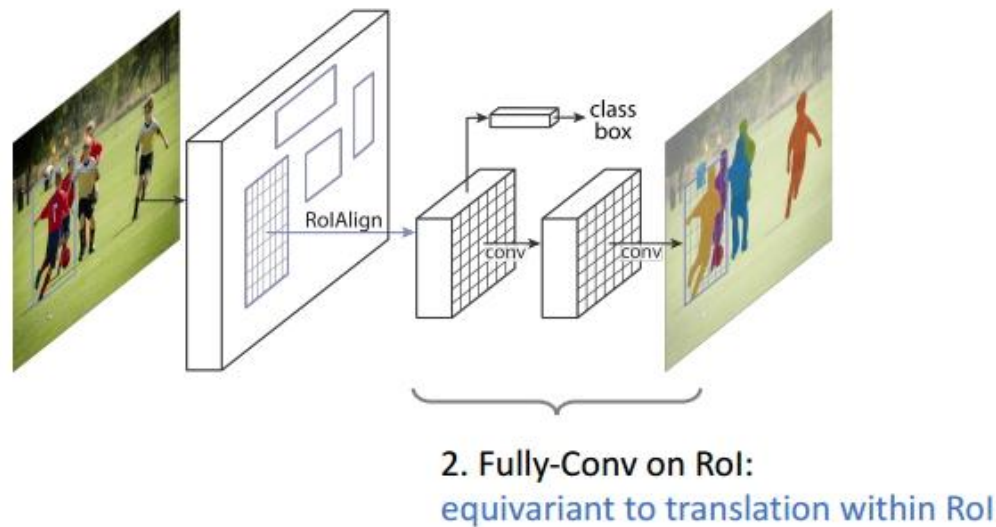
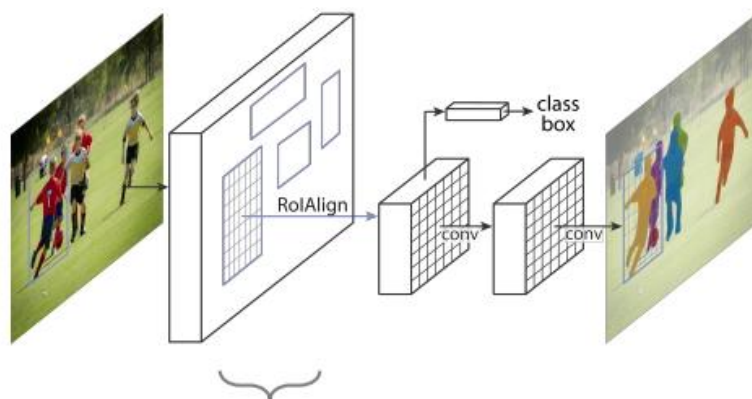


图 15 Equivariance2

在 ROI 上面的全卷积操作（FCN 网络）和在 ROI 中的变换具有同变性；



### 3. RoIAlign:

3a. maintain translation-equivariance before/after RoI

图 16 Equivariance3

ROIAlign 操作保持了 ROI 变换前后的同变性；

## Fully-Conv on RoI

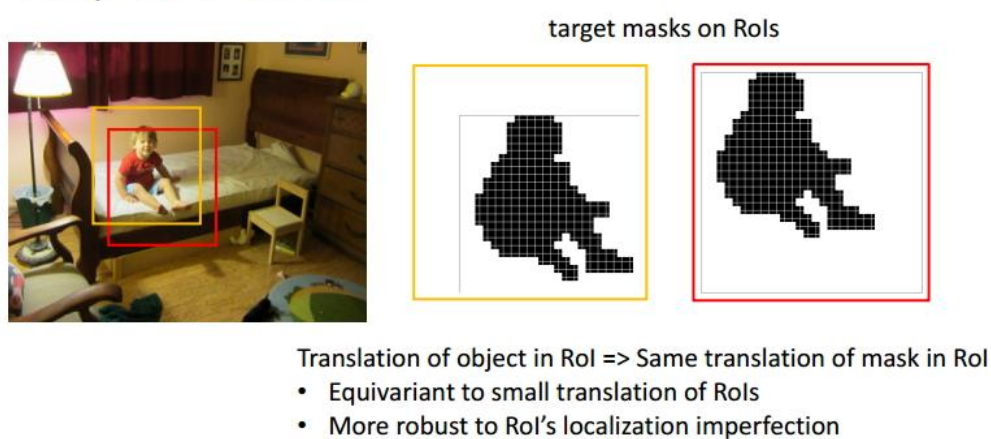


图 17 ROI 中的全卷积

## RoIAlign: Scale-Equivariance

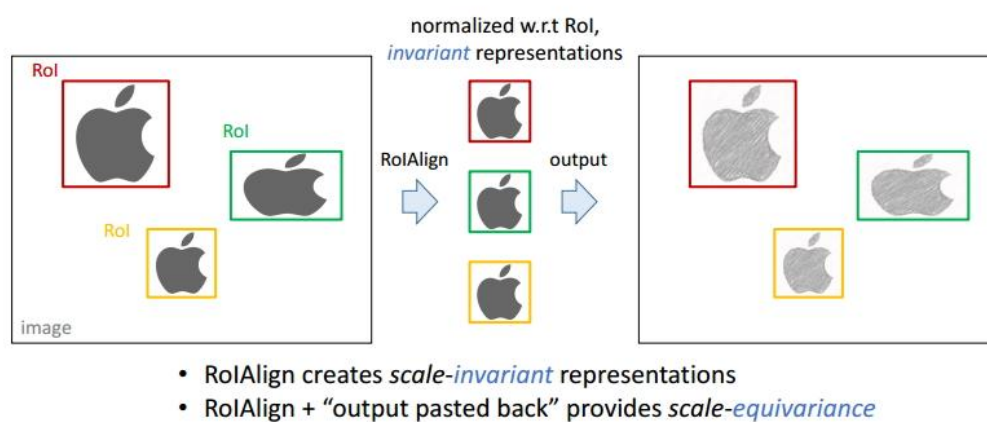


图 18 ROIAlign 的尺度同变性

- Translation-equivariant
  - FCN features
  - FCN mask head
  - RoIAlign (pixel-to-pixel behavior)
- Scale-equivariant (and aspect-ratio-equivariant)
  - RoIAlign (warping and normalization behavior) + paste-back
  - FPN features

图 19 Mask R-CNN 中的同变性总结

### 3. 算法实现细节

- Hyperparameter** : using existing Faster R-CNN
- Backbone architectures** : ResNet50, ResNet101, FPN (Feature Pyramid Networks)
- Input image** : resized into 800px for its shorter size
- GPU** : 8 GPU @2 images on training
- Training time** : 32 hours (ResNet50-FPN), 44 hours ResNet101-FPN), not end-to-end training.
- Testing time (ResNet101-FPN)**: 195ms per image on an Nvidia Tesla M40 GPU (plus 15ms CPU time resizing the outputs to the original resolution)
- Dataset** : MS COCO (80k train, 35k val, 5k test)

图 20 算法实现细节

观察上图，我们可以得到以下的信息：

Mask R-CNN 中的超参数都是用了 Faster r-cnn 中的值，机智，省时省力，效果还好，别人已经替你调节过啦，哈哈哈；

使用到的预训练网络包括 ResNet50、ResNet101、FPN，都是一些性能很好地网络，尤其是 FPN，后面会有分析；

对于过大的图片，它会将其裁剪成 800x800 大小，图像太大的话会大大的增加计算量的；利用 8 个 GPU 同时训练，开始的学习率是 0.01，经过 18k 次将其衰减为 0.001，ResNet50-FPN 网络训练了 32 小时，ResNet101-FPN 训练了 44 小时；

在 Nvidia Tesla M40 GPU 上面的测试时间是 195ms/张；

使用了 MS COCO 数据集，将 120k 的数据集划分为 80k 的训练集、35k 的验证集和 5k 的测试集；



## 四、性能比较

### 1. 定量结果分析

## Ablation: RoIPool vs. RoIAlign

baseline: ResNet-50-Conv5 backbone, stride=32

	mask AP			box AP		
	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sup>bb</sup>	AP <sub>50</sub> <sup>bb</sup>	AP <sub>75</sub> <sup>bb</sup>
<i>RoIPool</i>	23.6	46.5	21.6	28.2	52.7	26.9
<i>RoIAlign</i>	<b>30.9</b>	<b>51.8</b>	<b>32.1</b>	<b>34.0</b>	<b>55.3</b>	<b>36.4</b>
	+7.3	+ 5.3	<b>+10.5</b>	+5.8	+2.6	+9.5

- huge gain at high IoU, in case of big stride (32)

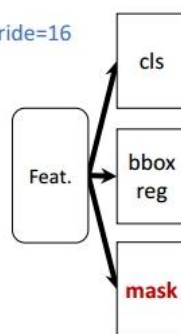
表 1 ROI Pool 和 ROIAlign 性能比较

由前面的分析，我们就可以定性的得到一个结论，ROIAlign 会使得目标检测的效果有很大的性能提升。根据上表，我们进行定量的分析，结果表明，ROIAlign 使得 mask 的 AP 值提升了 10.5 个百分点，使得 box 的 AP 值提升了 9.5 个百分点。

## Ablation: Multinomial vs. Binary Masks

baseline: ResNet-50-Conv4 backbone, stride=16

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



- cls head: did recognition



- mask head: no need to recognize again



表 2 Multinomial 和 Binary loss 比较

根据上表的分析，我们知道 Mask R-CNN 利用两个分支将分类和 mask 生成解耦出来，然后利用 Binary Loss 代替 Multinomial Loss，使得不同类别的 mask 之间消除了竞争。依赖于分类分支所预测的类别标签来选择输出对应的 mask。使得 mask 分支不需要进行重新分类工作，使得性能得到了提升。


## Ablation: MLP vs. FCN mask

baseline: ResNet-50-FPN backbone

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

• +2.1 point

• MLP: lose “place-coded” info, too abstract



• FCN: translation-equivariant




表 3 MLP 与 FCN mask 性能比较

如上表所示，MLP 即利用 FC 来生成对应的 mask，而 FCN 利用 Conv 来生成对应的 mask，仅仅从参数量上来讲，后者比前者少了很多，这样不仅会节约大量的内存空间，同时会加速整个训练过程（因此需要进行推理、更新的参数更少啦）。除此之外，由于 MLP 获得的特征比较抽象，使得最终的 mask 中丢失了一部分有用信息，我们可以直观地从右边看到差别。从定性角度来讲，FCN 使得 mask AP 值提升了 2.1 个百分点。

## Instance Segmentation Results on COCO

	backbone	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
MNC [7]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [20] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [20] +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	<b>35.7</b>	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	<b>37.1</b>	<b>60.0</b>	<b>39.4</b>	<b>16.9</b>	<b>39.9</b>	<b>53.5</b>

- **2 AP better** than SOTA w/ R101, without bells and whistles
- **200ms / img**

表 4 实例分割的结果

## Object Detection Results on COCO

	backbone	AP <sup>bb</sup>	AP <sub>50</sub> <sup>bb</sup>	AP <sub>75</sub> <sup>bb</sup>	AP <sub>S</sub> <sup>bb</sup>	AP <sub>M</sub> <sup>bb</sup>	AP <sub>L</sub> <sup>bb</sup>
Faster R-CNN+++ [15]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [22]	ResNet-101-FPN	<b>36.2</b>	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [32]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [31]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	<b>52.1</b>
Faster R-CNN, RoIAlign	ResNet-101-FPN	<b>37.3</b>	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	<b>39.8</b>	<b>62.3</b>	<b>43.4</b>	<b>22.1</b>	<b>43.2</b>	51.2

bbbox detection improved by:

- RoIAlign

表 5 目标检测的结果

观察目标检测的表格，我们可以发现使用了 ROIAIign 操作的 Faster R-CNN 算法性能得到了 0.9 个百分点，Mask R-CNN 比最好的 Faster R-CNN 高出了 2.6 个百分点。

2. 定性结果分析

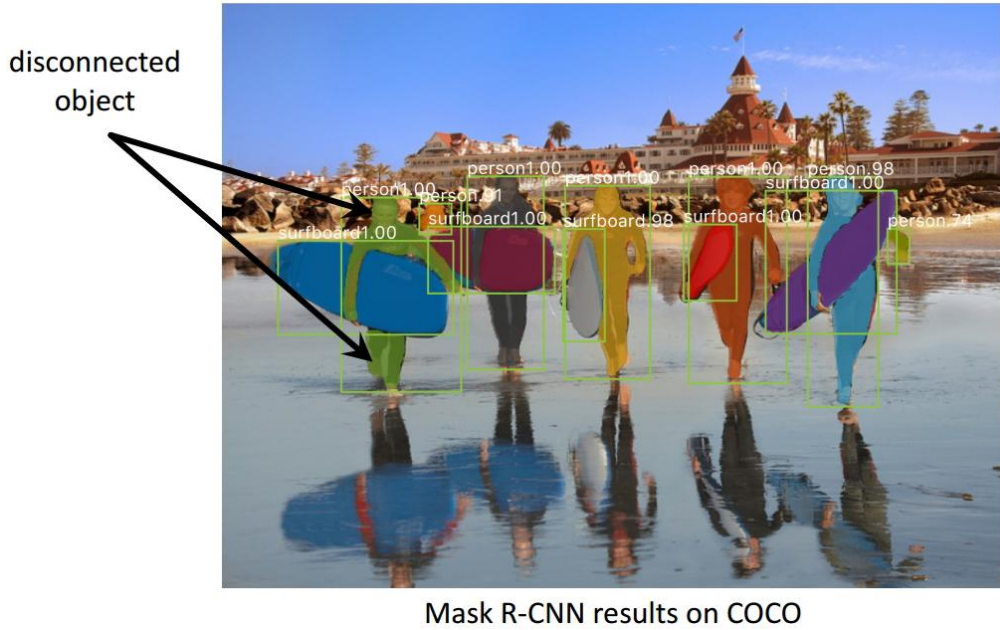


图 21 实例分割结果 1

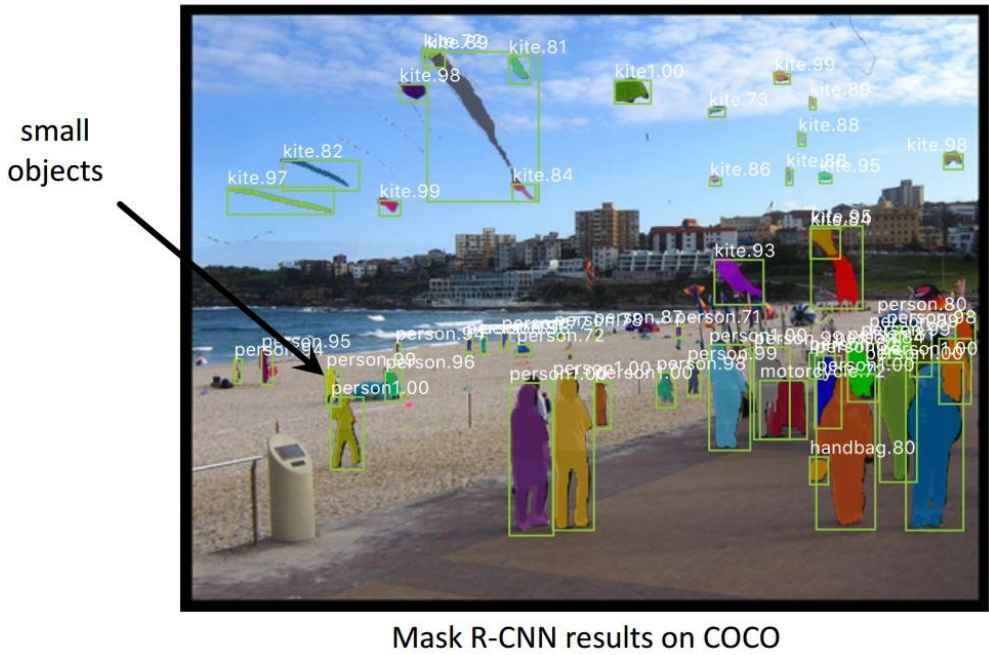


图 22 实例分割结果 2



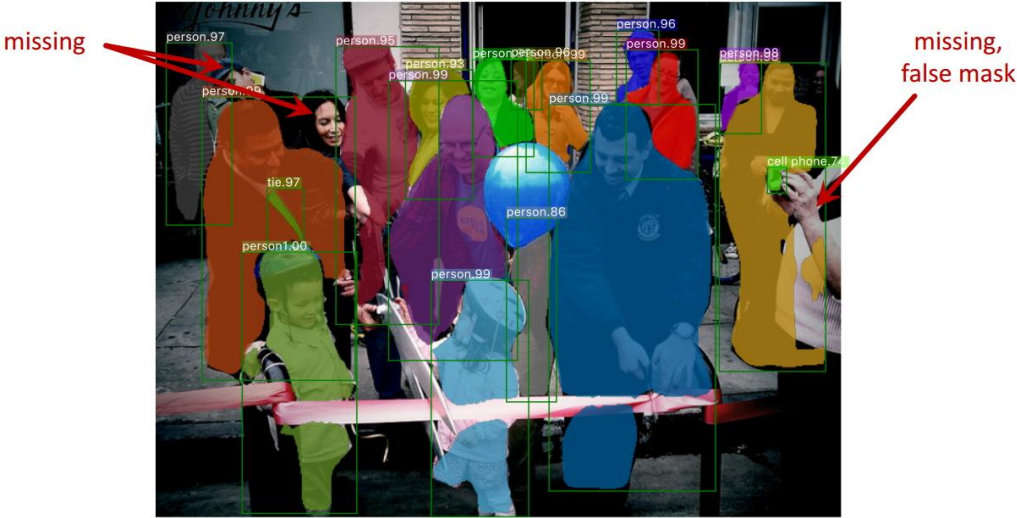
### Human pose estimation

	$AP^{kp}$	$AP_{50}^{kp}$	$AP_{75}^{kp}$	$AP_M^{kp}$	$AP_L^{kp}$
CMU-Pose+++ [6]	61.8	84.9	67.5	57.1	68.2
G-RMI [31] <sup>†</sup>	62.4	84.0	68.5	<b>59.1</b>	68.1
Mask R-CNN, keypoint-only	62.7	87.0	68.4	57.4	71.1
Mask R-CNN, keypoint & mask	<b>63.1</b>	<b>87.3</b>	<b>68.7</b>	57.8	<b>71.4</b>



图 23 人体姿势识别结果

### Failure case: detection/segmentation



Mask R-CNN results on COCO

图 24 失败检测案例 1



## Failure case: recognition

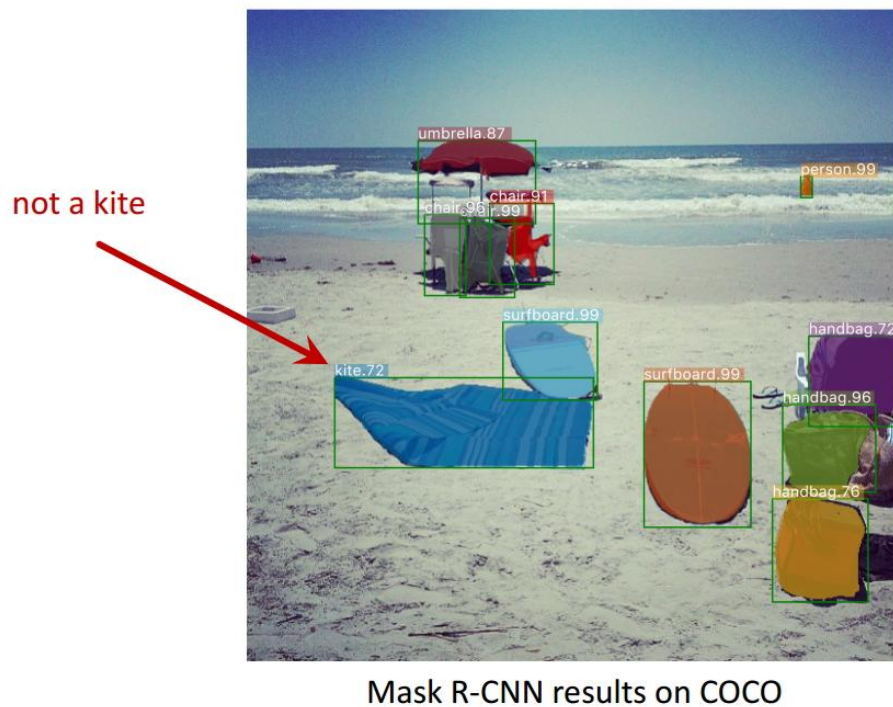


图 25 失败检测案例 2

## 五、总结

Mask R-CNN 论文的主要贡献包括以下几点：

分析了 ROI Pool 的不足，提升了 ROIAlign，提升了检测和实例分割的效果；  
将实例分割分解为分类和 mask 生成两个分支，依赖于分类分支所预测的类别标签来选择输出对应的 mask。同时利用 Binary Loss 代替 Multinomial Loss，消除了不同类别的 mask 之间的竞争，生成了准确的二值 mask；  
并行进行分类和 mask 生成任务，对模型进行了加速。

参考文献：

[1] 何铠明大神在 ICCV2017 上在的 Slides，视频链接

[2] Ardian Umam 对 Mask R-CNN 的讲解，视频链接



