CapsNet是如何克服CNN的缺点的

胶囊网络（CapsNet）是一种新的热门的神经网络架构。它可能对深度学习带来深远的影响，特别是对计算机视觉领域。等一下！计算机视觉不是差不多已经被解决了吗？我们不是已经看到了多种卷积神经网络（CNN）的神奇案例？它们不是已经在计算机视觉任务（例如分类、定位、物体检测、语义分割或实例分割，见图1）上实现超越人类的水平了吗？

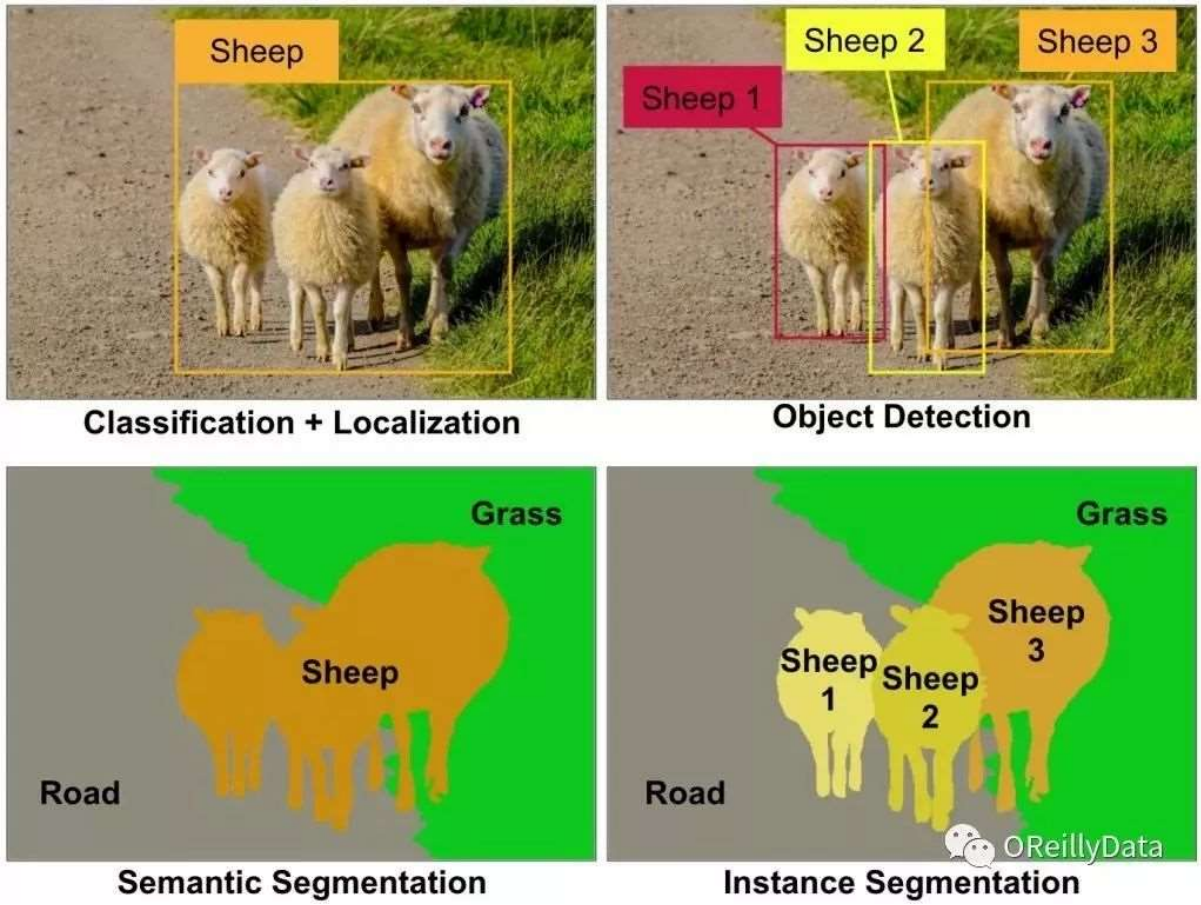

图1 一些主要的计算机视觉任务。当前，每种任务都需要一个不一样的CNN架构。比如分类里的ResNet，物体检测里的YOLO，实例分割里的Mask R-CNN等。图片由Aurélien Géron提供。

恩，是的。我们已经见到了很多神奇的CNN，但是，

- 它们需要非常多的图片进行训练（或重复使用了已用海量数据训练过的神经网络的一部分）。 而CapsNet使用少得多的训练数据就能泛化。
- CNN们并不能很好地应对模糊性，但CapsNet可以。所以它能在非常拥挤的场景里也表

现得很好（尽管它目前还需要解决背景图的问题）。

- CNN会在池化层理丢失大量的信息，从而降低了空间分辨率（见图2），这就导致对于输入的微小变化，其输出几乎是不变的。在诸如语义分割这样的场景里，这会是一个问题，因为细节信息必须要在网络里被保留。现在，这一问题已经被通过在CNN里构建复杂的架构来恢复这些损失的信息所解决。在CapNet里面，细节的姿态信息（比如对象的准确位置、旋转、厚度、倾斜度、尺寸等）会在网络里被保存下来，不用先丢失再恢复。输入上微小的变化会带来输出上的小变化，信息被保存。这被称为“等变的”。这就让CapsNet能使用一个简单和统一的架构来应对不同的视觉任务。

- 最后，CNN需要额外的组件来自动识别每个小部分属于哪个物体（例如这条腿属于这只羊）。而CapsNet则可以给你这些部分的层级结构。

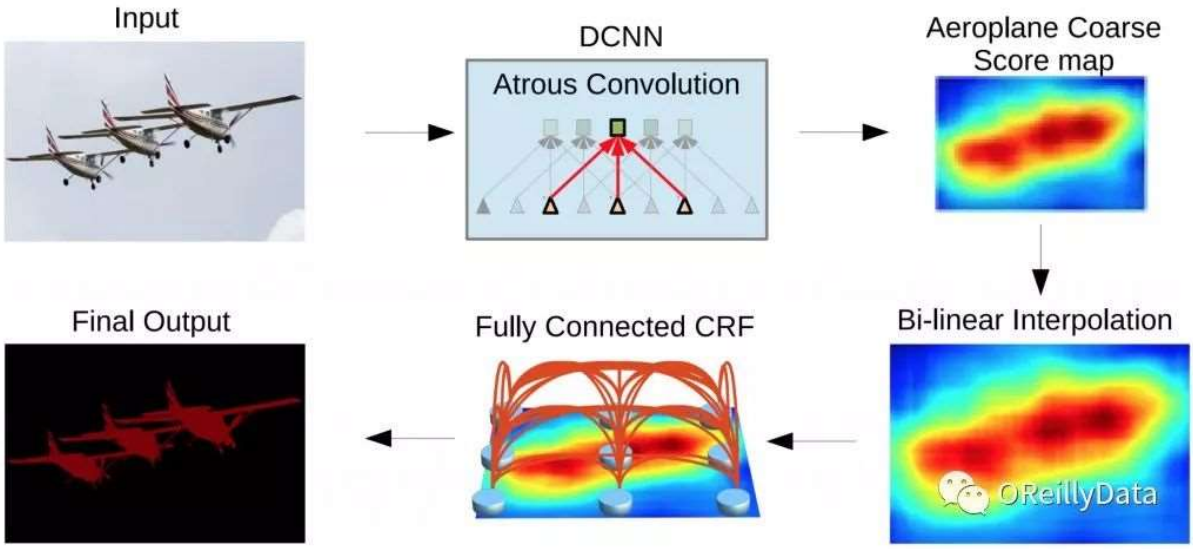

图2 DeepLab2的图像分割管道，来自Liang-Chieh Chen等。注意这里面CNN的输出（右上角图）是非常得粗糙。这就需要一些额外的步骤来恢复丢失的细节。图片来自论文《DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs》，获作者友情授权使用。可以查看S. Chilamkurthy的这篇很棒的博文来了解语义分割的神经网络的架构有多么复杂多样。

CapsNet是在2011年在Geoffrey Hinton等人的一篇名为《Transforming Autoencoders》的论文中首次出现。但仅在几个月前（即2017年11月），Sara Sabour、Nicholas Frosst和Geoffrey Hinton发表了一篇名为《Dynamic Routing between Capsules》的论文，其中介绍了CapsNet架构。

该架构在MNIST（著名的手写数字图像数据集）上达到了最先进的性能，并且在MultiMNIST数据集（一种有重叠的不同数字组的手写数字的变体）上获得了比CNN好得多的结果。见图

3。

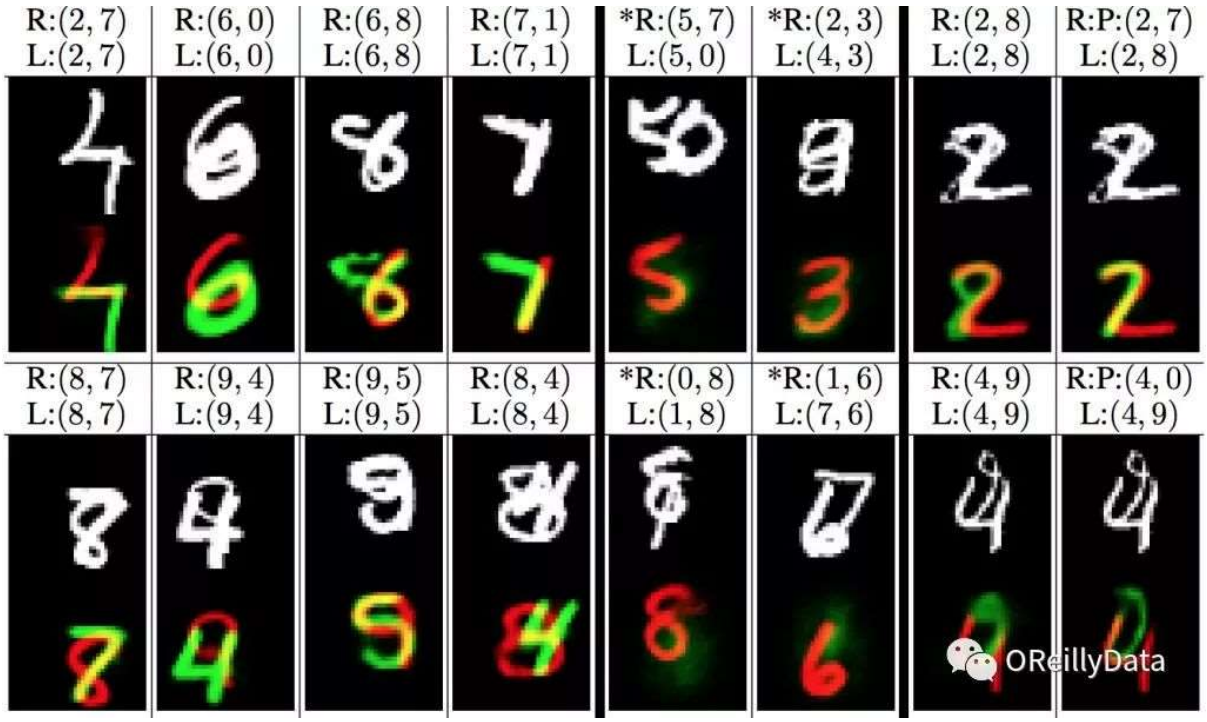

图3 MultiMNIST图片（白色）和由CapsNet重构后的结果（红色和绿色）。“R”=重构后的结果；“L”=标注。例如，对第一个样例（左上角）的预测是准确的，重构的结果也对。但是在第五个例子里，预测（5,7）是错的，不是（5,0）。因此，5被正确地重构了，但是0没有。图片来自论文《Dynamic routing between capsules》，作者友情授权使用。

尽管有上述这些优点，但CapsNet还远未到完美的程度。首先，它们在更大的图片上（例如CIFAR10或ImageNet数据集里）还没有CNN的表现好。另外，CapsNet的计算量很大，同时它还不能区分靠的很近的相同的物体（这被称为“拥挤问题”，人类也有这个问题）。

但是这里的关键点是胶囊网络是非常有希望的，看起来只要做一些修改就能让胶囊网络充分释放它们的潜能。毕竟现代CNN在1998年就被发明了，但也要经过几次改进，直到2012年的ImageNet大赛上才达到业界领先水平。

那么，到底CapsNet是什么?

简而言之，一个胶囊网络是由胶囊而不是由神经元构成。一个胶囊是一小群神经元，它们可以学习在一个图片的一定区域内检查一个特定的对象（比如，一个矩形）。

它的输出是一个向量（例如，一个8维的向量）。每个向量的长度代表了物体是否存在的估计概率[1]，它的方向（例如在8维空间里）记录了物体的姿态参数（比如，精确的位置、旋转等）。如果物体有稍微的变化（比如，移动、旋转、尺寸变化等），胶囊将也会输出一个长度相同但是方向稍微变化的向量。因此胶囊是等变的。

和常规神经网络很类似，CapsNet也是由多层构成（见图4）。处于最底层的胶囊被称为向量胶囊：它们每个都只用图片的一小部分区域作为输入（称为感知域），然后试图去探测某个特殊的模式（例如，一个矩形）是否存在，以及姿态如何。在更高层的胶囊（称为路由胶囊）则是探测更大和更复杂的物体，比如船等。

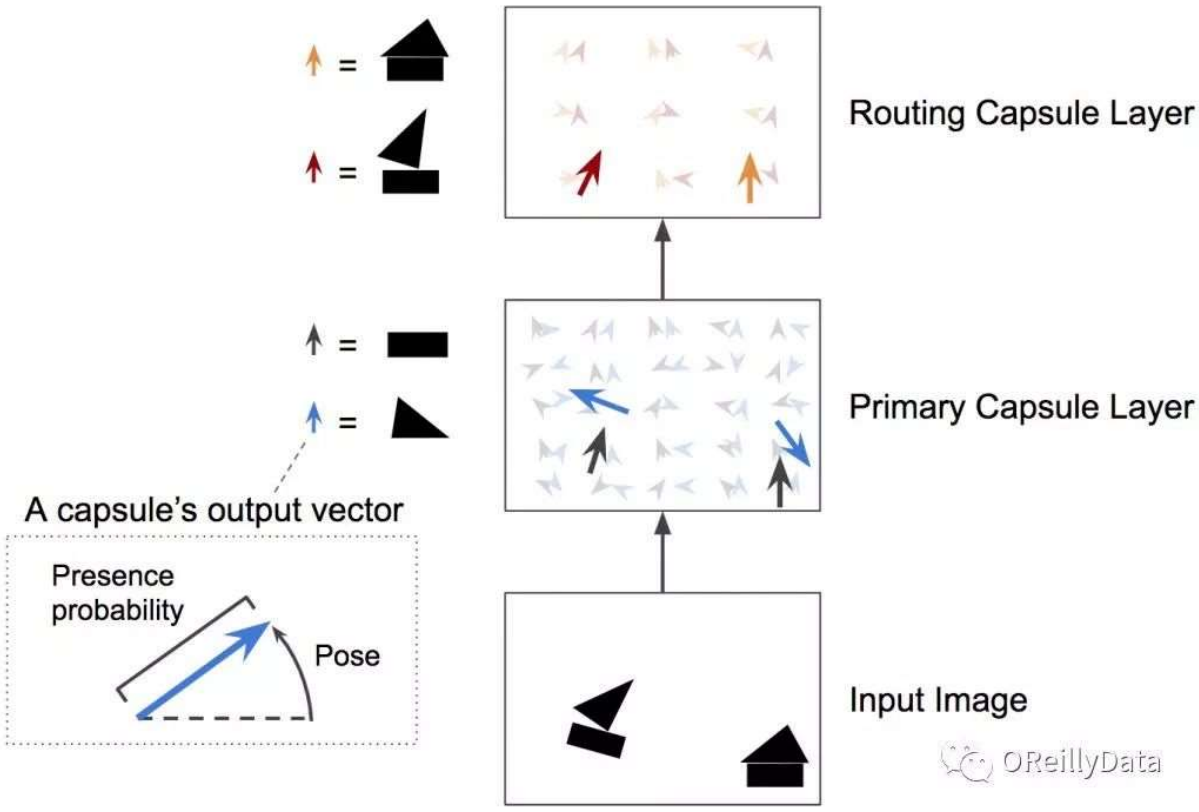

图4 一个两层的CapsNet。在本例里，向量胶囊层有两个5x5胶囊的特征图。其中第二个胶囊层有两组3x3胶囊的特征图。每个胶囊输出一个向量。每个箭头代表不同的胶囊的输出。蓝色的箭头代表那些检测三角形的胶囊的输出；黑色箭头代表试图检测矩形的胶囊的输出。图片由Aurélien Géron提供。

向量胶囊层是用几个常规卷积层实现。例如在这篇论文里，他们使用两个卷积层输出256个包含标量的6×6特征图。然后把这个输出变形成32个包含8维向量的6×6特征图。最后，他们使用一个新奇的压缩函数来确保这些向量的长度都是在0到1（来代表概率）之间。就这样，它们就产生了向量胶囊的输出。

下一层的胶囊也试图去检测物体和它们的姿态，但是它们的工作机制很不一样。它们使用的是一种叫按一致性路由的算法。这里是CapsNet的主要魅力所在。先让我们看一个例子。

假设我们只有两个向量胶囊：一个识别矩形的胶囊和一个识别三角形的胶囊，而且假定他们都能检测到相应的形状。矩形和三角形可以是房子或是船的一部分（见图5）。根据矩形的姿态，它们都是稍微向右旋转了一点，那么房子和船也都会向右旋转了一点。

根据三角形的姿态，这个房子几乎完全是上下颠倒的，而船则是稍微向右旋转了一点。注意在这里，整体的形状和整体与部分的关系都是在训练中学习的。现在可以看到矩形和三角形在船上的姿态是一致的，而在房子上的姿态则非常不一致。因此，很有可能这里的矩形和三角形是同一条船上的一部分，而房子上则不是。

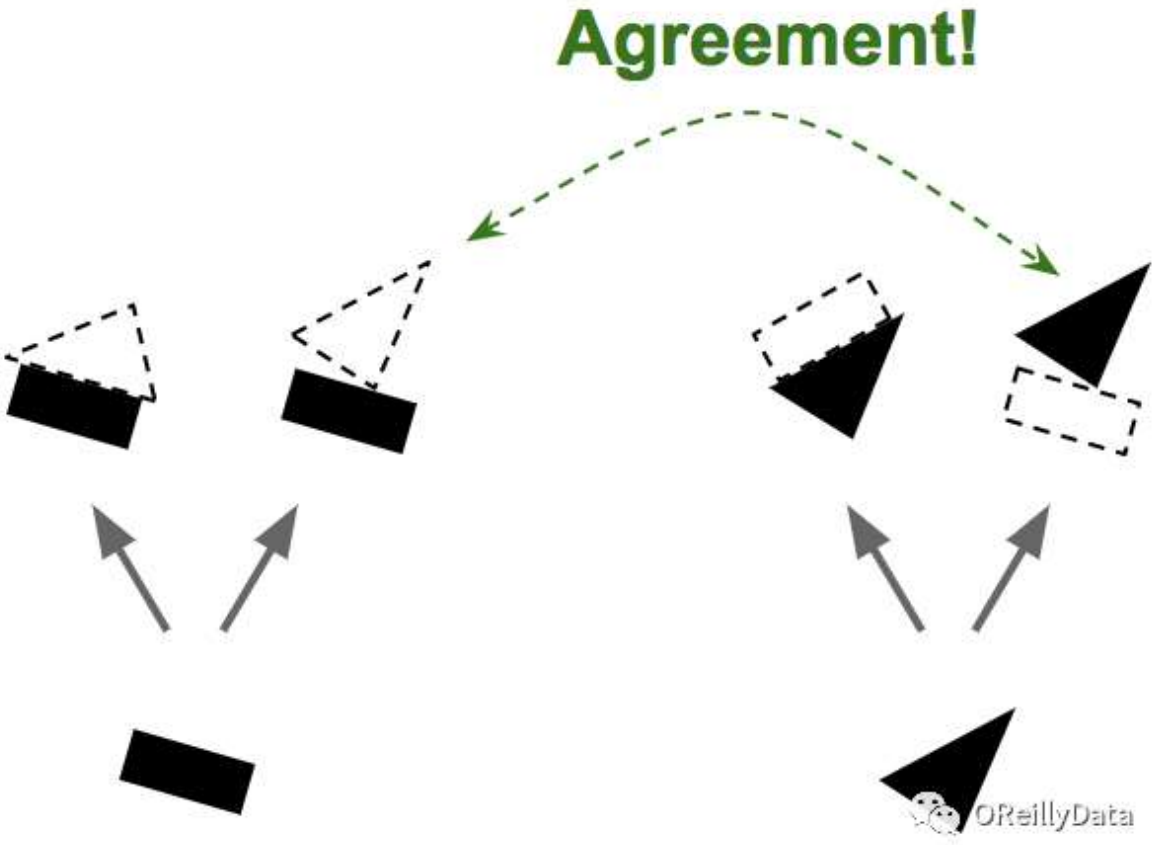

图5 按一致性路由。第一步——基于对象的部分是否存在和姿态来预测对象是否存在及其姿态，接着查看预测间的一致性。图片由Aurélien Géron提供。

因为我们现在已经很有信心地知道矩形和三角形是船的一部分，所以把矩形和三角形的输出更多指向船的胶囊而更少指向房子的胶囊就顺理成章了。用这个方法，船的胶囊将会获得更多的有用输入信号，而房子的胶囊则接收更少的噪音。对每一个连接，按一致性路由算法会维护一个路由权重（见图6）：它对于一致的会增加权重，而对于不一致的则降低权重。

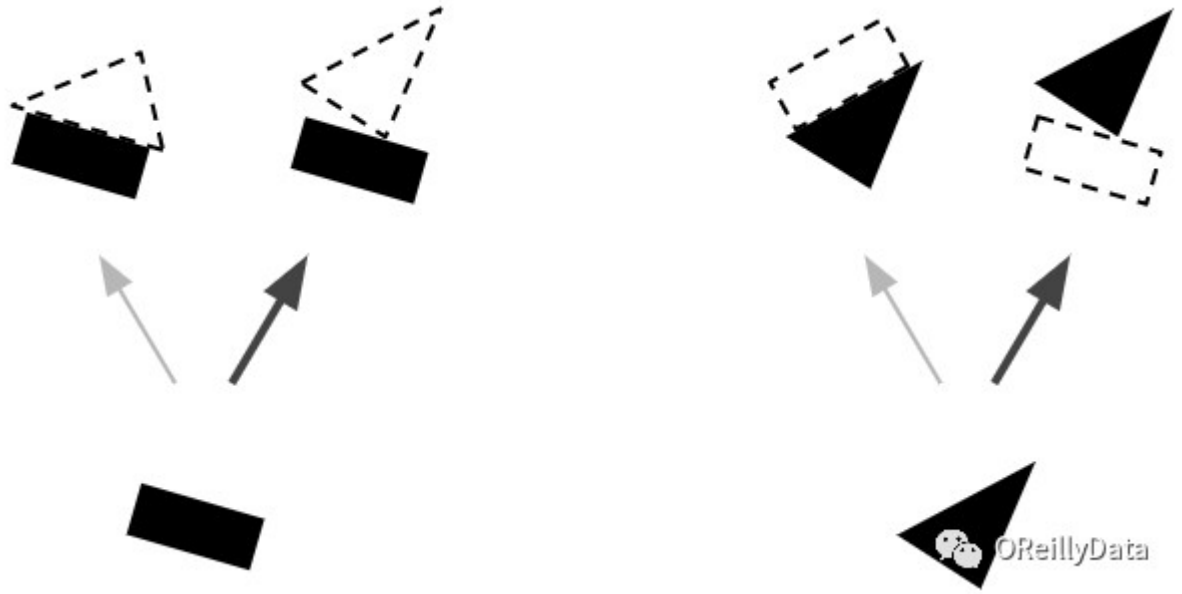

图6 按一致性路由。第二步——更新路由权重。图片由Aurélien Géron提供。

这个按一致性路由算法会涉及到一些循环：一致性检测和路由权重更新（值得注意的是，这一步骤会在每次预测时发生，不只是一次，也不只是在训练时）。这一算法对于拥挤的场景特别有用。例如图7里的场景是比较模糊的，因为你能看到中间有一个上下颠倒的房子，但这样会让上面的三角形和下面的矩形无法得到解释。

按一致性路由算法更有可能收敛到一个更好的解释：下面是一条船，而上面是一个房子。这样模糊性就“可以被解释了”：下面的矩形最好是被看成一条船的一部分，同时这样也解释了下面的三角形。一旦下面的两个部分被解释好了，剩下的部分就很容易地被解释成一个房子。

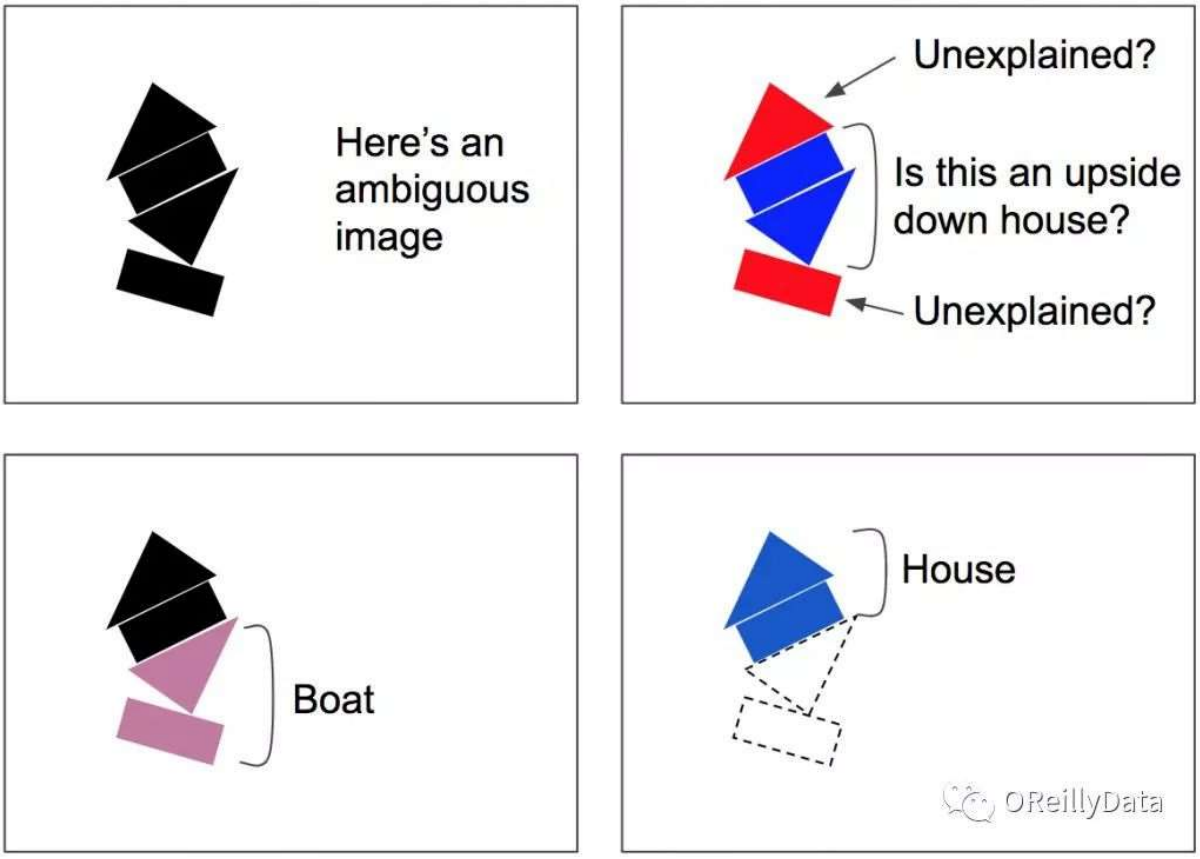

图7 按一致性路由可以分解拥挤的场景，例如这里的图片。因为它可能会被错误地解释为中间是一个上下颠倒的房子和两个无法解释的部分。相反的，下面的矩形会被路由给船，而这也就把下面的三角形带进了船。一旦这个船被解释清楚，那么就能很容易地解释上面的两个部分是一个房子了。图片由Aurélien Géron提供。

就是这些！你现在知道CapsNet背后的关键原理了！如果你想了解更多的细节，可以查看我的两个关于CapsNet的视频（一个是关于它的架构，另外一个是如何实现它）和带有我的注释的这个用TensorFlow实现的胶囊网络（Jupyter Notebook文件）。请随意对视频和GitHub上的文件进行评论，或是通过我的Twitter账号@aureliengeron联络我。希望你觉得这个博文对你有帮助。

[1] 这是由S. Sabour、N. Frosst和G. Hinton发表的论文《Dynamic routing with capsules》里提出的最初的架构。但是在文章发表后他们又提出了一个更通用的架构。其中物体出现的概率以及姿态参数都在输出向量里按不同的方式编码了。不过核心的观点还是没有改变。

Aurélien Géron