3.2 特征点检测（Landmark detection）

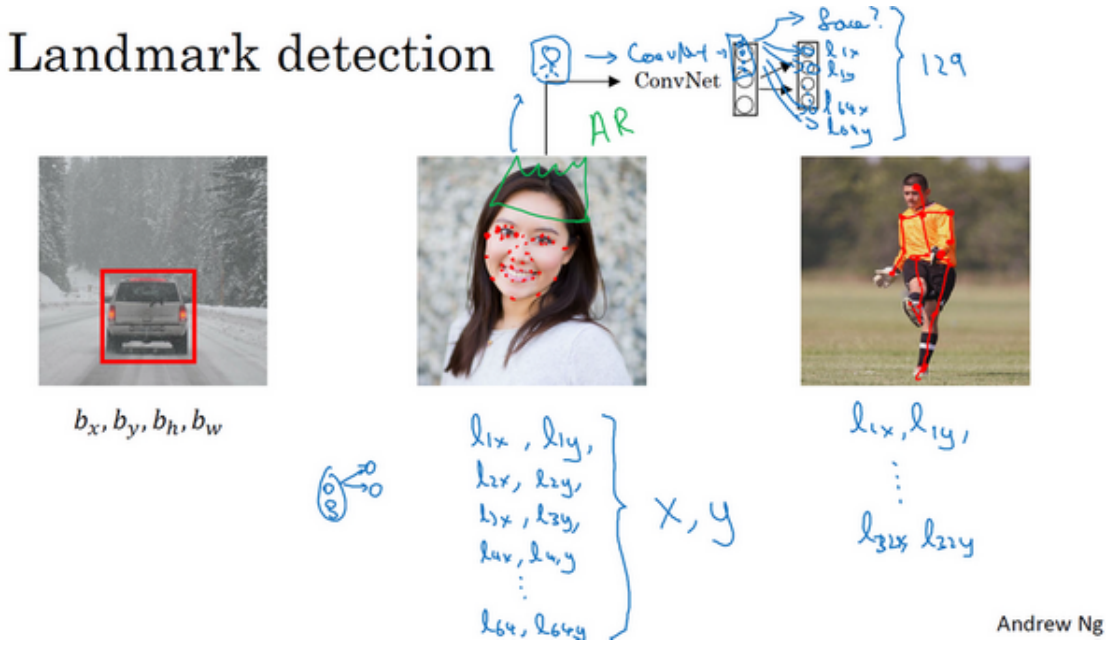
上节课，我们讲了如何利用神经网络进行对象定位，即通过输出四个参数值b_x、b_y、b_h和b_w给出图片中对象的边界框。更概括地说，神经网络可以通过输出图片上特征点的(x, y)坐标来实现对目标特征的识别，我们看几个例子。

假设你正在构建一个人脸识别应用，出于某种原因，你希望算法可以给出眼角的具体位置。眼角坐标为(x, y)，你可以让神经网络的最后一层多输出两个数字l_x和l_y，作为眼角的坐标值。如果你想知道两只眼睛的四个眼角的具体位置，那么从左到右，依次用四个特征点来表示这四个眼角。对神经网络稍做些修改，输出第一个特征点（l_{1x}，l_{1y}），第二个特征点（l_{2x}，l_{2y}），依此类推，这四个脸部特征点的位置就可以通过神经网络输出了。

（编者注：图中的模特是吴恩达老师的夫人 **Carol Reiley**）

也许除了这四个特征点，你还想得到更多的特征点输出值，这些（图中眼眶上的红色特征点）都是眼睛的特征点，你还可以根据嘴部的关键点输出值来确定嘴的形状，从而判断人物是在微笑还是皱眉，也可以提取鼻子周围的关键特征点。为了便于说明，**你可以设定特征点的个数，假设脸部有 64 个特征点，有些点甚至可以帮助你定义脸部轮廓或下颌轮廓。选定特征点个数，并生成包含这些特征点的标签训练集，然后利用神经网络输出脸部关键特征点的位置。**

具体做法是，准备一个卷积网络和一些特征集，将人脸图片输入卷积网络，输出 1 或 0，1 表示有人脸，0 表示没有人脸，然后输出（l_{1x}，l_{1y}）……直到（l_{64x}，l_{64y}）。这里我用l代表一个特征，这里有 129 个输出单元，其中 1 表示图片中有人脸，因为有 64 个特征，64×2=128，所以最终输出 128+1=129 个单元，由此实现对图片的人脸检测和定位。这只是一个识别脸部表情的基本构造模块，如果你玩过 **Snapchat** 或其它娱乐类应用，你应该对 **AR**（增强现实）过滤器多少有些了解，**Snapchat** 过滤器实现了在脸上画皇冠和其他一些特殊效果。检测脸部特征也是计算机图形效果的一个关键构造模块，比如实现脸部扭曲，头戴皇冠等等。当然为了构建这样的网络，你需要准备一个标签训练集，也就是图片x和标签y的集合，这些点都是人为辛苦标注的。



最后一个例子，如果你对人体姿态检测感兴趣，你还可以定义一些关键特征点，如胸部的中点，左肩，左肘，腰等等。然后通过神经网络标注人物姿态的关键特征点，再输出这些标注过的特征点，就相当于输出了人物的姿态动作。当然，要实现这个功能，你需要设定这

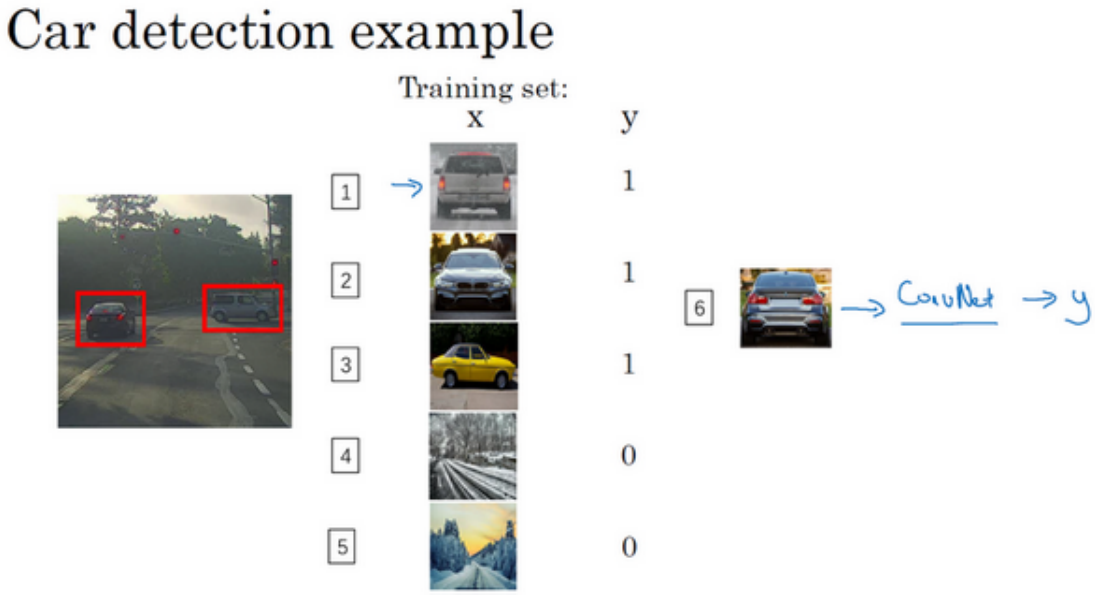
些关键特征点，从胸部中心点(l_{1x}，l_{1y})一直往下，直到(l_{32x}，l_{32y})。

一旦了解如何用二维坐标系定义人物姿态，操作起来就相当简单了，批量添加输出单元，用以输出要识别的各个特征点的(x, y)坐标值。要明确一点，特征点 1 的特性在所有图片中必须保持一致，就好比，特征点 1 始终是右眼的外眼角，特征点 2 是右眼的内眼角，特征点 3 是左眼内眼角，特征点 4 是左眼外眼角等等。所以标签在所有图片中必须保持一致，假如你雇用他人或自己标记了一个足够大的数据集，那么神经网络便可以输出上述所有特征点，你可以利用它们实现其他有趣的效果，比如判断人物的动作姿态，识别图片中的人物表情等等。

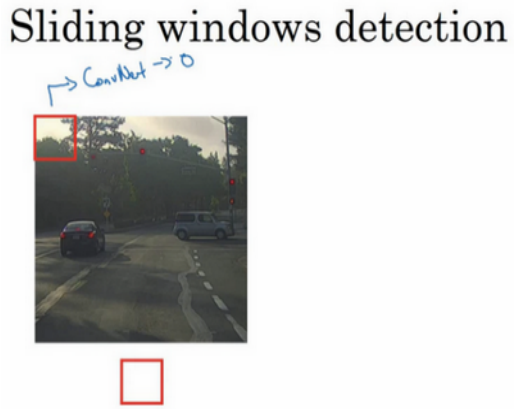
以上就是特征点检测的内容，下节课我们将利用这些构造模块来构建对象检测算法。

3.3 目标检测（Object detection）

学过了对象定位和特征点检测，今天我们来构建一个对象检测算法。这节课，我们将学习如何通过卷积网络进行对象检测，采用的是基于滑动窗口的目标检测算法。



假如你想构建一个汽车检测算法，步骤是，首先创建一个标签训练集，也就是x和y表示适当剪切的汽车图片样本，这张图片（编号 1）x是一个正样本，因为它是一辆汽车图片，这几张图片（编号 2、3）也有汽车，但这两张（编号 4、5）没有汽车。**出于我们对这个训练集的期望，你一开始可以使用适当剪切的图片，就是整张图片x几乎都被汽车占据，**你可以照张照片，然后剪切，剪掉汽车以外的部分，使汽车居于中间位置，并基本占据整张图片。有了这个标签训练集，你就可以开始训练卷积网络了，输入这些适当剪切过的图片（编号 6），卷积网络输出y，0 或 1 表示图片中有汽车或没有汽车。训练完这个卷积网络，就可以用它来实现**滑动窗口目标检测**，具体步骤如下。



假设这是一张测试图片，首先选定一个特定大小的窗口，比如图片下方这个窗口，将这

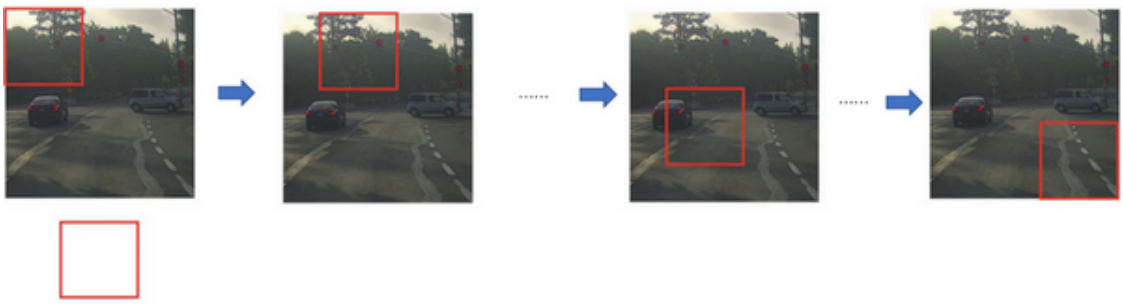
个红色小方块输入卷积神经网络，卷积网络开始进行预测，即判断红色方框内有没有汽车。

滑动窗口目标检测算法接下来会继续处理第二个图像，即红色方框稍向右滑动之后的区域，并输入给卷积网络，因此输入给卷积网络的只有红色方框内的区域，再次运行卷积网络，然后处理第三个图像，依次重复操作，直到这个窗口滑过图像的每一个角落。

为了滑动得更快，我这里选用的步幅比较大，思路是以固定步幅移动窗口，遍历图像的每个区域，把这些剪切后的小图像输入卷积网络，对每个位置按 0 或 1 进行分类，这就是所谓的图像滑动窗口操作。

重复上述操作，不过这次我们选择一个更大的窗口，截取更大的区域，并输入给卷积神经网络处理，你可以根据卷积网络对输入大小调整这个区域，然后输入给卷积网络，输出 0 或 1。

再以某个固定步幅滑动窗口，重复以上操作，遍历整个图像，输出结果。



然后第三次重复操作，这次选用更大的窗口。

如果你这样做，不论汽车在图片的什么位置，总有一个窗口可以检测到它。

比如，将这个窗口（编号 1）输入卷积网络，希望卷积网络对该输入区域的输出结果为 1，说明网络检测到图上有辆车。

这种算法叫作滑动窗口目标检测，因为我们以某个步幅滑动这些方框窗口遍历整张图片，对这些方形区域进行分类，判断里面有没有汽车。

滑动窗口目标检测算法也有很明显的缺点，就是计算成本，因为你在图片中剪切出太多小方块，卷积网络要一个个地处理。如果你选用的步幅很大，显然会减少输入卷积网络的窗口个数，但是粗糙间隔尺寸可能会影响性能。反之，如果采用小粒度或小步幅，传递给卷积网络的小窗口会特别多，这意味着超高的计算成本。

所以在神经网络兴起之前，人们通常采用更简单的分类器进行对象检测，比如通过采用手工处理工程特征的简单的线性分类器来执行对象检测。至于误差，因为每个分类器的计算成本都很低，它只是一个线性函数，所以滑动窗口目标检测算法表现良好，是个不错的算法。然而，卷积网络运行单个分类人物的成本却高得多，像这样滑动窗口太慢。除非采用超细粒度或极小步幅，否则无法准确定位图片中的对象。

不过，庆幸的是，计算成本问题已经有了很好的解决方案，大大提高了卷积层上应用滑动窗口目标检测器的效率，关于它的具体实现，我们下节课再讲。