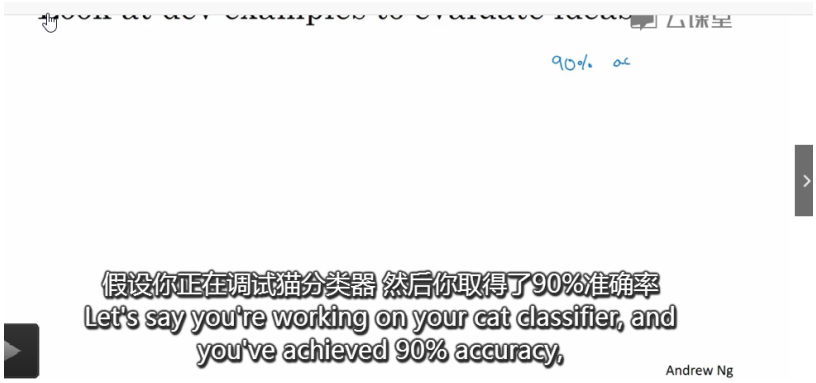
第三篇第二周

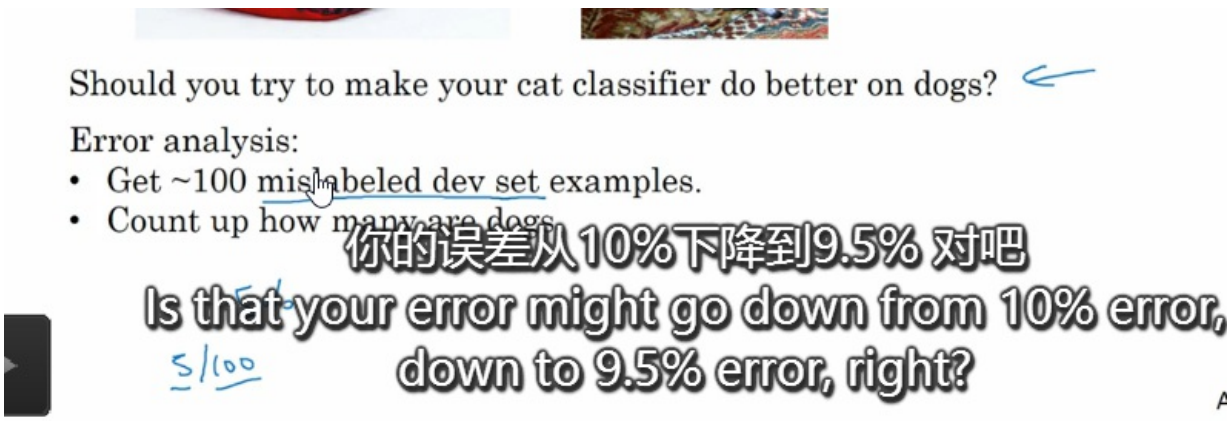
1、进行误差分析

如果你希望让学习算法能够胜任人类能做的任务，但是你的学习算法还没有达到人类的表现，那么人工检查一下你的算法犯的错误，也许可以让你了解接下来做应该什么，这个过程称为误差分析。

例子：

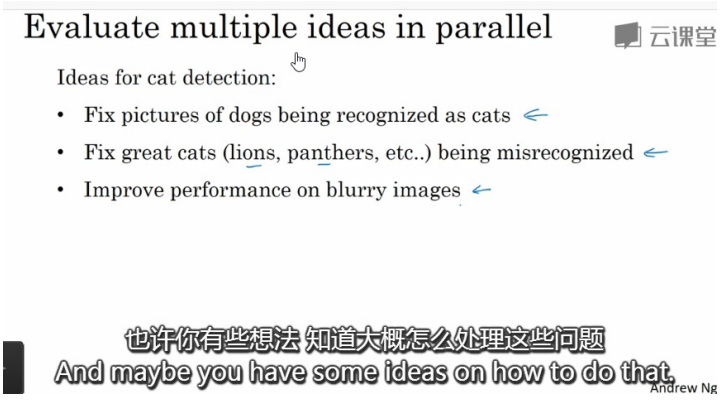


这离你的目标还很远，仔细一看，发现有些把狗分类成猫了，所以可以：



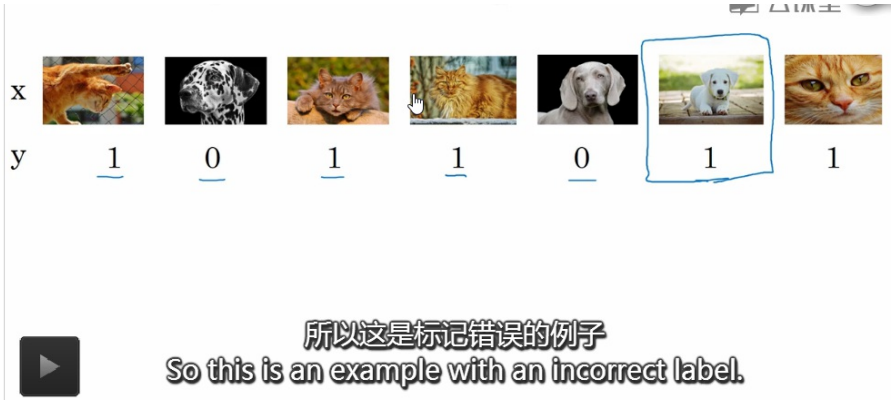
如果你发现100张分类错误的图片中，有50张是狗，所以有50%都是狗的图片，现在花费时间去解决狗的问题效果可能就很好了，如果你解决了这个问题，那么你的误差可能从10%下降到5%。那么你可能觉得让误差减半的方向值得一试，可以集中精力减少错误标记的狗图的问题。

你还可能有很多想法：



要进行误差分析来评估上面的三个想法。进行误差分析，你需要找到一组错误例子，可能在你的开发集里或者测试集里，观察错误标记的例子，看看假阳性和假阴性，统计属于不同错误类型的错误数量，在这个过程中，你可能会得到启发，归纳出新的错误类型，总之，统计不同错误标记类型占总数的百分比，可以帮助你发现那些问题需要优先解决，或者给你构思新优化方向的灵感。

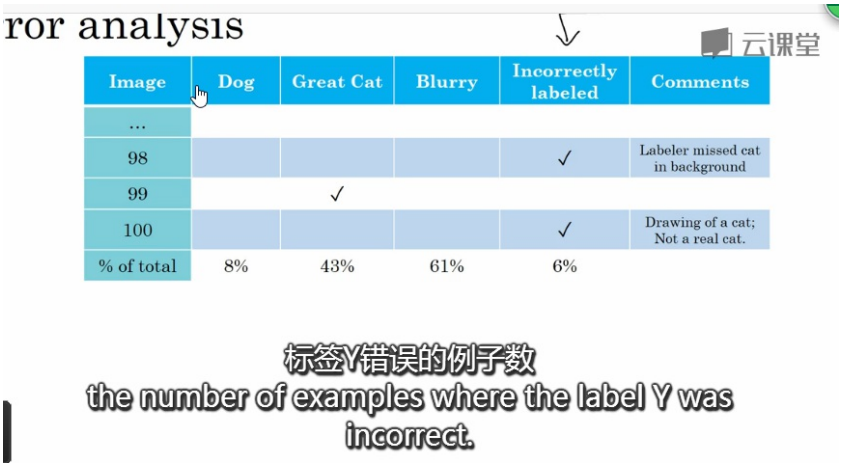
2、清楚标注错误的数据



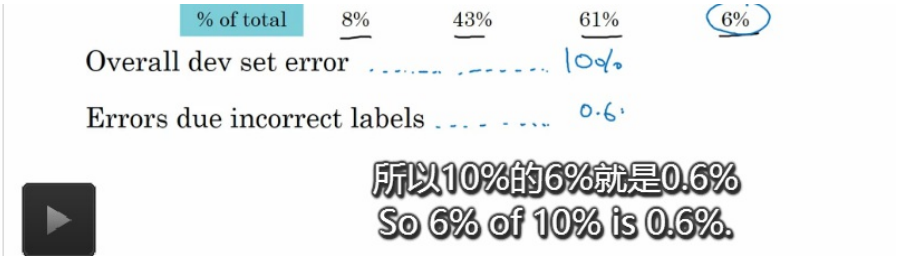
只要数据总量足够大，放着不管也是可以的，因为深度学习算法具有一定的鲁棒性，深度学习算法对随机误差很鲁棒，但是对系统性的问题就没有那么鲁棒了，所以比如说，做标记的人一直把白色的狗标记成猫，那就成问题了，因为你的分类器，学习之后，会把所有白色的狗分类成猫，但是随机误差或者近似随机的误差，对于大多数深度学习算法来说，不成问题，

问题2：

一般建议你在误差分析时，添加一个额外的列，这样你也可以统计：



如果这些标注错误，严重的影响了你在开发集上评估算法的能力，那么就应该花时间去修正错误的标签，但是，如果他们没有严重影响到你用开发集评估成本偏差的能力，那么可能就不应该花宝贵的时间去处理：



其他原因占比是9.4%：

这样子就相当于没有严重影响，这种情况下，有9.4%的误差需要集中精力去修正，而标记出错的错误仅仅只是0.6%，总体错误的一小部分而已，所以如果你一定要这么做，你也可以手工修正这些错误标签；

现在假设误差降到了2%，还是0.6%标注错误的，占比为百分之30，这时就要改了，

3、快速搭建你的第一个第一个系统，并进行迭代

如果你想要搭建全新的机器学习系统，然后开始迭代，你应该快速设立开发集，测试集还有就是评价指标，这样就决定了你的目标所在，如果你的目标定错了，之后改也是可以的，然后马上搭建一个机器学习系统原型，然后找到训练集训练一下，看看效果，开始理解你的算法表现如何，在开发集和测试集，你的评估指标上表现如何，当你建立了第一个系统之后，你就可以马上用之前的偏差方差分析，还有就是之前最后几个视频所介绍的误差分析，来确认下一步优先做什么，特别是如果误差分析让你了解了大部分误差来源，那么你就可以集中精力去研究这些技术，建立一个初始系统的所有意义在于，它可以是一个快速或者肮脏的实现，其实就是有一个学习过的系统，有一个训练过得系统，让你确定偏差和方差的范围，就可以知道下一步优先做什么，让你能够进行误差分析，可以观察一些错误，然后想出所有能走的方向，哪些是实际上最有希望的方向。

回顾一下：应该快速建立第一个系统，然后迭代，不过如果你在这个应用领域有很多经验，这个建议适用程度要低一些，还有一种情况适用程度更低，当这个领域有很多可以借鉴的学术文献时，处理的问题和你需要解决的完全相同，比如说，人脸识别就有很多学术文献，如果你尝试搭建一个人脸识别设备，那么可以从现有的大量学术文献为基础出发，一开始就搭建比较复杂的系统，但是如果你第一次处理某个新的问题，那不鼓励你想太多或者把第一个系统做的太复杂，采用上面第一段的做法就可以了，慢慢改善系统。

4、在不同的划分上进行训练或者测试

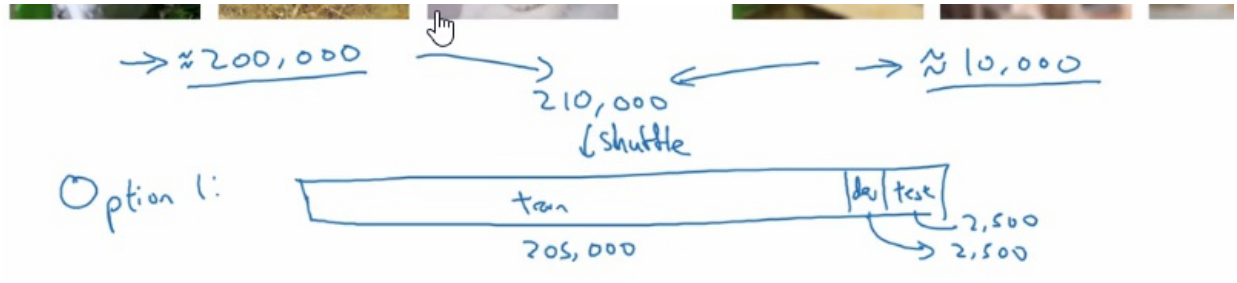
如何处理训练集和测试集分布存在差异的问题：

用户从应用中上传的图片是不是猫
that your users upload from the mobile app is a cat or not.

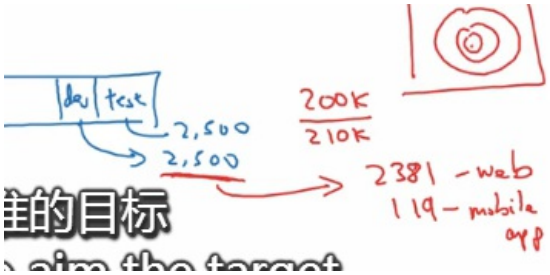
你有两个数据来源，一个是你真正关心的数据分布，来自应用上传的数据，比如右边应用，这些照片一般很业余，取景不太好，或者有些模糊，另外一些数据，比如左边的，可能来源于网络，比较清晰，像素比较高，我们真正关心的是，处理左边的图片时，效果好不好：

你的训练集就太小了
giving you a relatively small training set.

不想用右边的数据，因为这样的话训练集太小，用左边的数据的话似乎有帮助，但是问题在于，这20万张图片并不完全来自你想要的分布，你可以做的一件事情就是将两组数据合并起来，这样你就有21万张图片，然后你可以把它随机分到训练集，开发集和测试集上去：



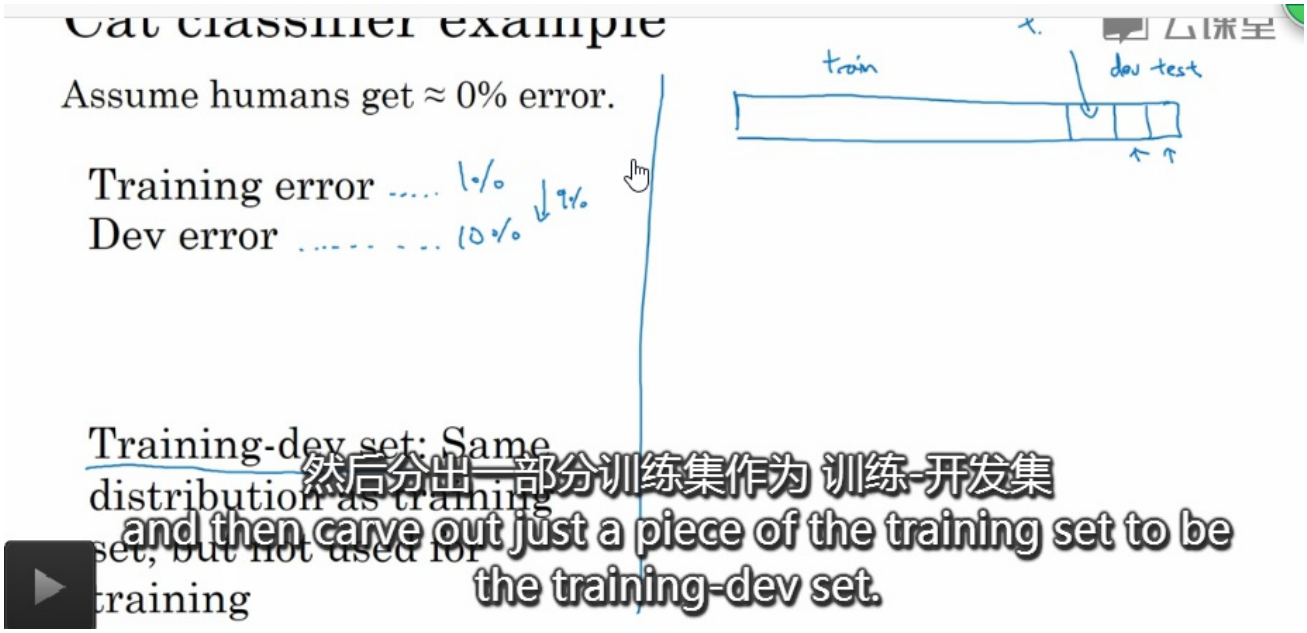
这样做有好的一面也有坏的一面，好的一面在于你的训练集开发集和测试集来自同一分布，这样更好管理，但是坏处在于，坏处也不小，就是如果你观察开发集，也就是那2500张图片，其中很多图片都来自网页下载的图片，并不是你真正关心的数据，你真正需要处理的是来自手机上的图片：开发集占的比例是：



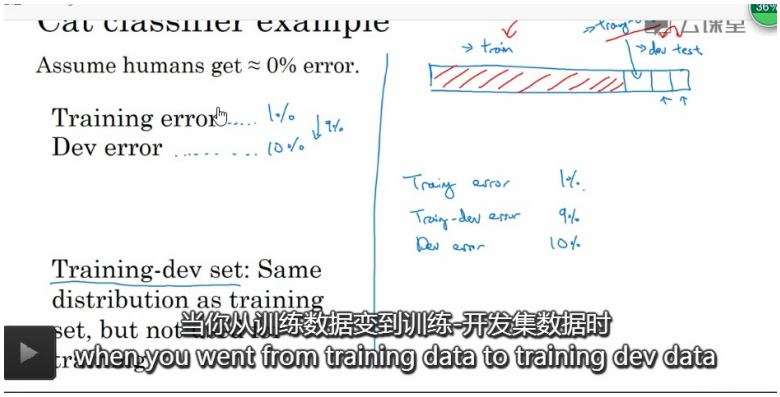
要记住，设立开发集的目的是，告诉你的团队去瞄准的目标，而你瞄准目标的方式你的大部分精力，都用在优先来自网页下载的图片，这其实不是你想要的。

你还可以这样做，开发集和测试集的各2500张图片全部来自手机拍摄，剩下的5000张图片全部合并到训练集上去，

5、不匹配数据划分的偏差和方差：



训练误差和测试误差差别巨大，可能原因是两者分布不同还有就是可能确实是方差比较大，为了弄清楚情况，我们可以从训练集划分出一块数据叫做训练-开发集，假如训练误差1%，训练-开发误差9%，开发集误差10%：



你可以得出结论：当你从训练数据变到训练-开发数据时，误差真的上升了很多，这就告诉你，算法存在方差问题，因为训练-开发集的误差，是在和训练集来自同一分布的数据集中测到的，虽然训练集上表现良好，但是无法泛化到相同分布的训练-开发集上去。

另外一个例子：训练误差1%，训练-开发误差1.5%，开发集误差10%，现在你的方差问题就很小了，所以这是数据不匹配的问题，数据不匹配，这是说明训练集和测试集分布不一样。

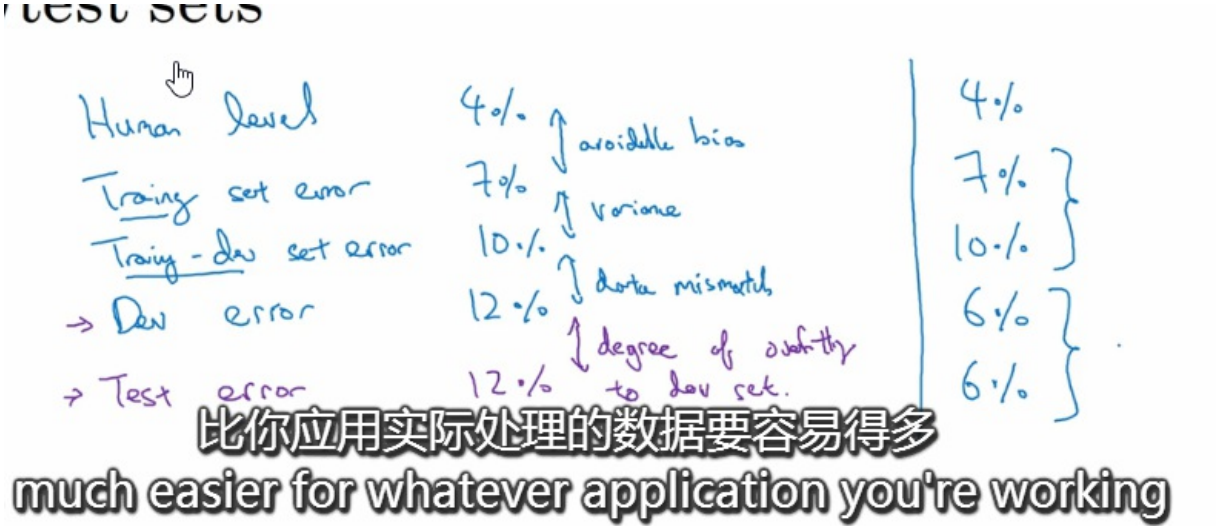
还有另外一种情况：训练误差10%，训练-开发误差11%，开发集误差12%，要知道人的误差接近0%（还是猫的分类问题）,那么这是存在高偏差问题。

还有一种情况是：训练误差10%，训练-开发误差11%，开发集误差20%，那么其实有两个问题，第一，可避免偏差相当高，第二，方差似乎很小但是数据不匹配问题很大。

一般原则：

我们要看的关键数据是：人类水平误差，你的训练集误差，训练-开发集误差（这个分布和训练集一样），开发集误差，通过这些，你就可以判断偏差问题，方差问题，数据不匹配问题等等。

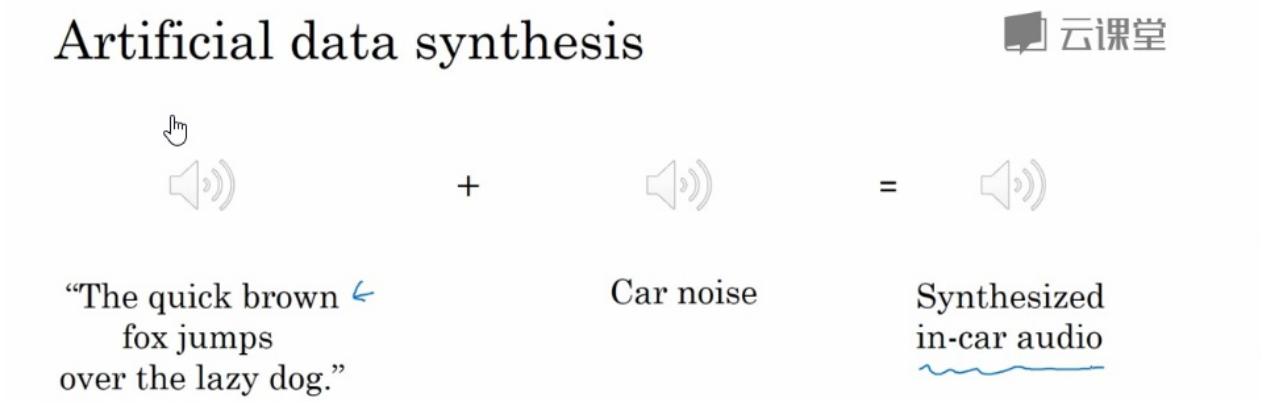
还有一种情况，就是你的训练集比开发集和测试集难识别得多，就会出现4,7,10,6,6也就是右边的情况：



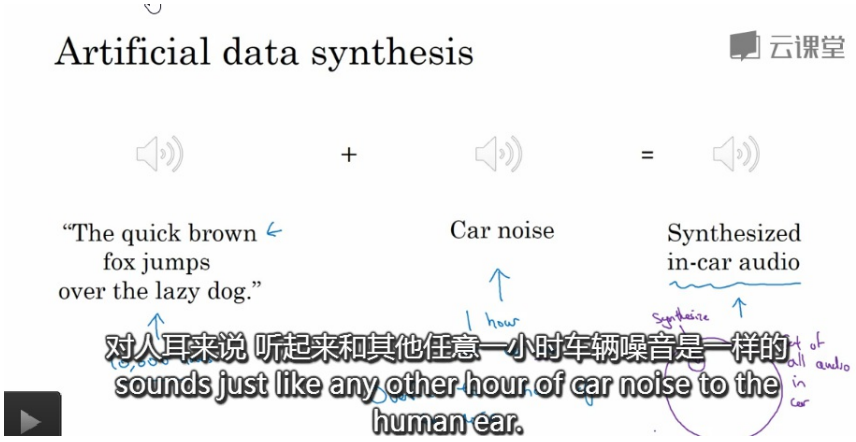
6、定位数据不匹配的问题

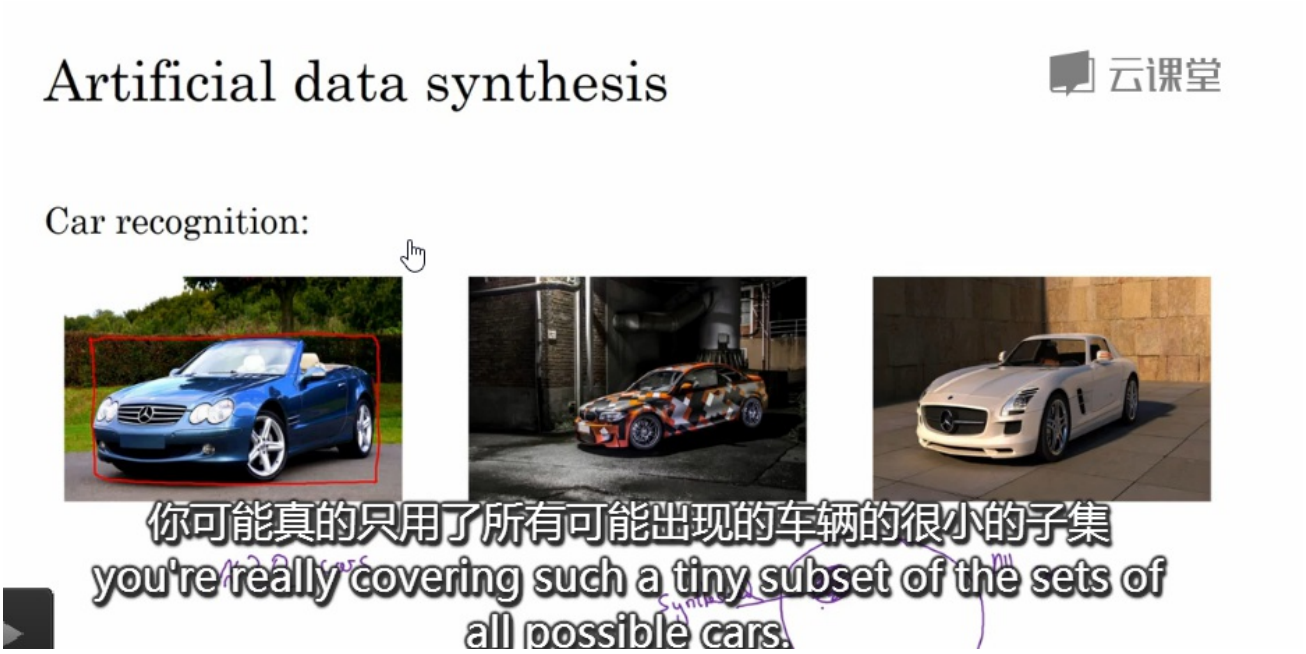
如果遇到数据不匹配怎么办：

1、如果我发现严重的数据不匹配问题，可以做误差分析，尝试了解训练集和开发集的具体差异，但是你只能人工去看开发集而不是测试集，解决方法之一就是收集更多类似于开发集和测试集的数据，对于语音识别，可以采用人工合成数据：



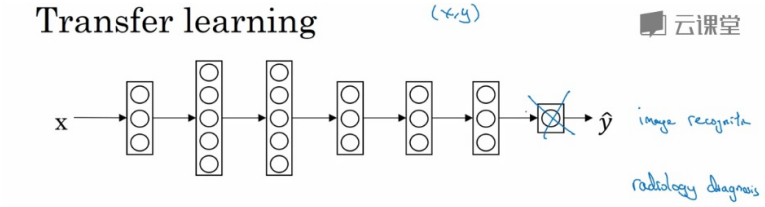
人工数据合成有一个潜在的问题，左边是10000个小时的安静状态下的录音，右边是1小时的车声，那么就可能过度拟合了车声：





7、迁移学习

假设你已经训练好一个图像识别的神经网络，所以你先用一个神经网络，并在(X,Y)对上训练，X是图像，Y是某些对象，图像是猫狗鸟和其他东西，如果你把这个神经网络拿来，然后让他适应或者说迁移，在不同任务中学到的知识，比如放射科诊断，或者说阅读X射线扫描图，你可以做的是，吧神经网络最后的输出层拿走：



然后为最后一层重新赋予随机权重，让它在放射诊断数据上训练，具体来说，在第一阶段训练过程中，当你进行图像识别任务训练时，你可以训练神经网络的所有常用参数，所有的权重，所有的层，然后你就得到一个能够图像识别预测的网络，在训练了这个神经网络之后，要实现迁移学习，你要做的是把数据集换成新的X,Y对，X也就是放射诊断图片，而Y是你要预测的诊断，你要做的是初始化最后一层的权重，假设称为W[l]，b[l]，随机初始化，现在，我在这个新的数据集上重新训练神经网络，你可以只是重新训练最后一层改变的那层的参数（数据量小的时候），你也可以重新训练所有的参数（数据量很多的时候）。

那么迁移学习什么时候是有意义的？

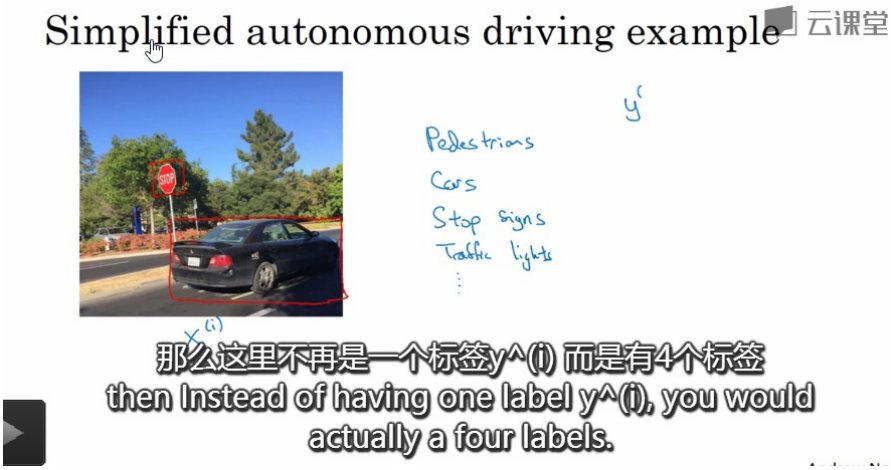
迁移学习起作用的场合是：迁移来源问题你有很多数据，但是迁移目标问题你没那么多数据，假设图像识别任务中你有100万个数据，所以这里数据相当多，可以学习低层次特征，可以在神经网络的前面几层学到如何识别很多有用的特征，但是对于放射科任务，也许你只有100个样本，所以你的放射学诊断问题数据很少，也许只有100次X射线扫描，所以你从图像识别训练中学到的很多知识可以迁移，并且真正帮你加强放射科识别任务的性能，即使你的放射科数据很少，对于语音识别，也许你已经用10000个小时数据，训练过你的语音识别系统，所以你从这10000个小时中学到了很多人类声音的特征，这数据量其实很多了，但对于触发字检测，也许你有1小时数据，所以这数据太小，不能用来拟合很多参数，所以在这种情况下，预先学到了很多人类声音的信息，人类语言的组成成分等等很多信息，可以帮你建立一个很好的唤醒字检测系统，即使你的数据集相对较小，只有当从数据量多的迁移到数据量少的才有意义，反过来就没有什么意义了。

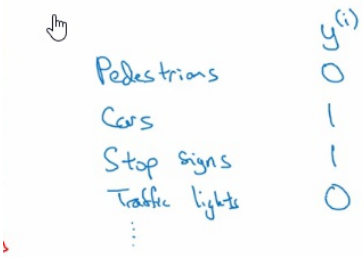
8、多任务学习

在迁移学习中，你的步骤是串行的，你从任务A中学到知识，然后迁移到任务B，在多任务学习中，你是同时开始学习的，试图让单个神经网络同时做几件事情，然后希望这里每个任务都可以帮到其他所有任务。

例子：

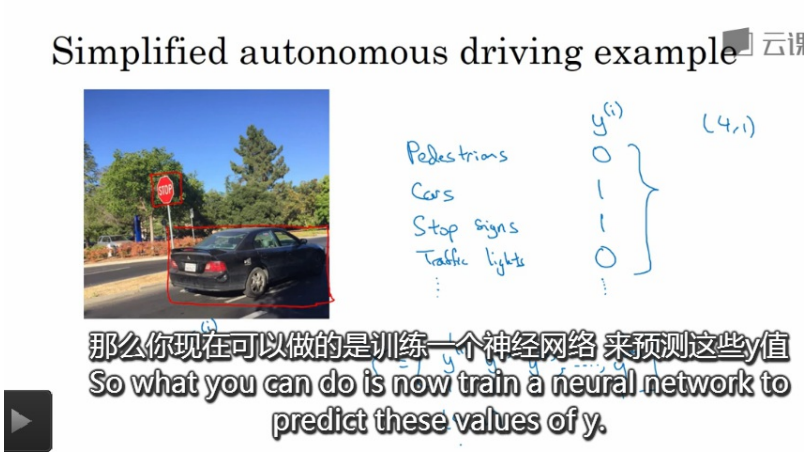
比如这个例子中，图像里有一个停车标志，还有一辆车，但是没有行人和交通灯，如果这是输入图片X(i)：



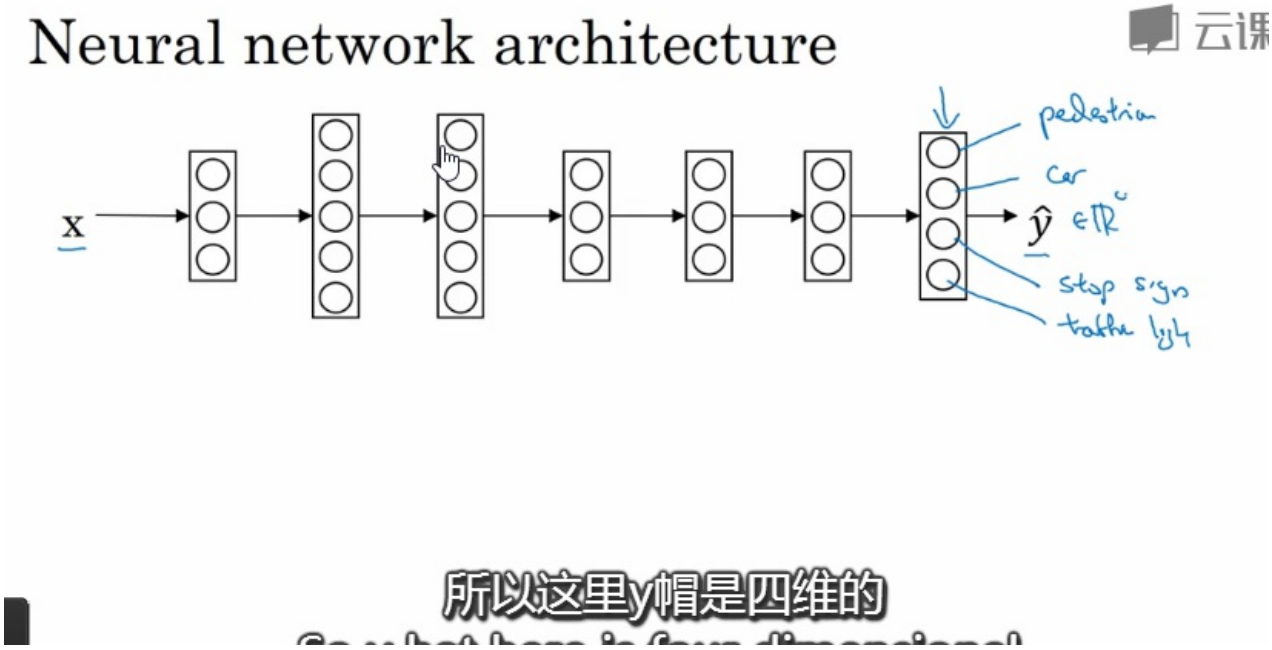


个停车标志 没有交通灯

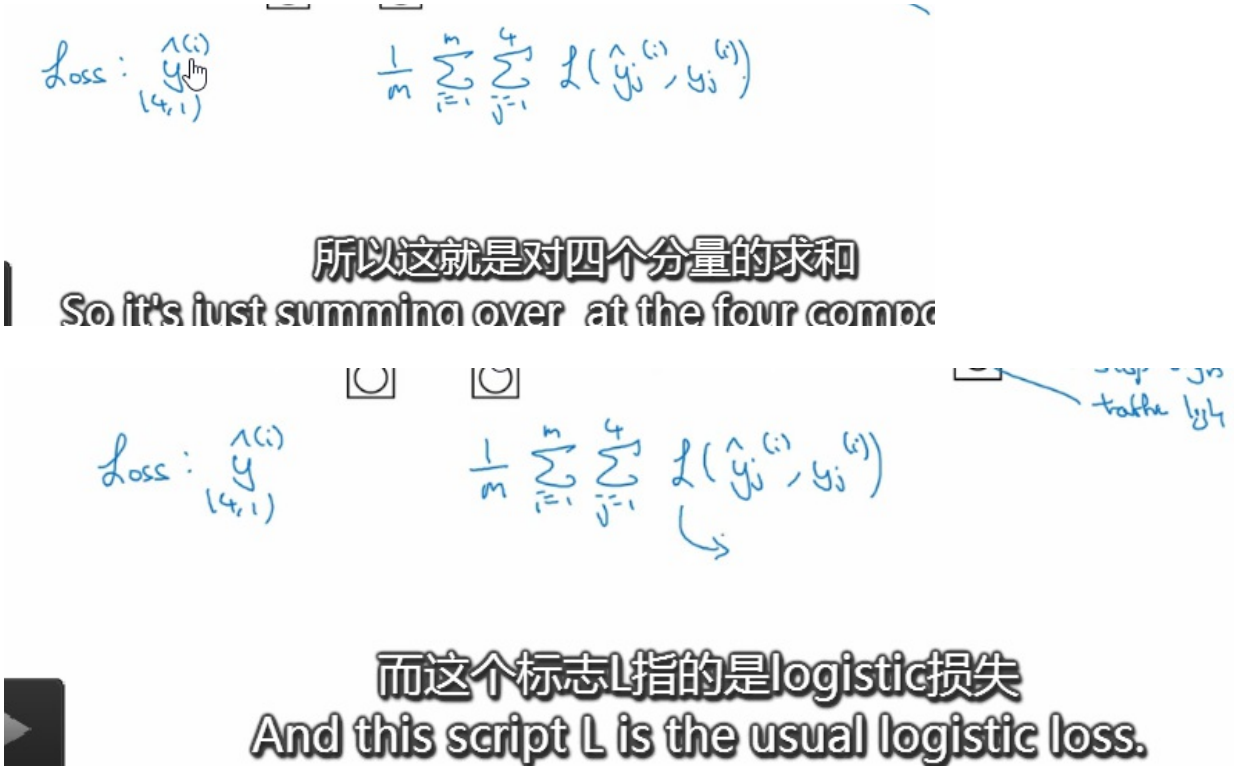
然后如果你尝试检测其他物体，也许y[i]的维数更高，现在我们就先用这四个吧，所以Y[i]是4*1的向量，如果你从整体来看这个训练集标签，现在你可以做的是训练一个神经网络，来预测这些Y值：

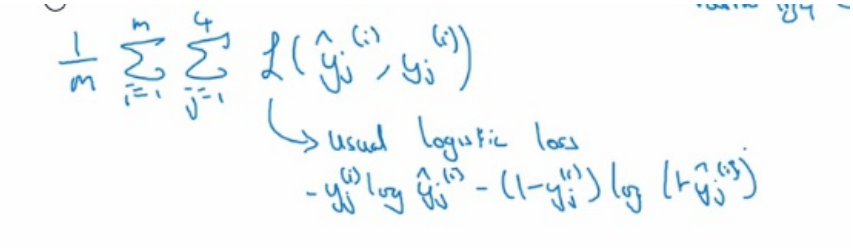


那么你的神经网络是输入X，输出得到4维向量Y，输出是4个节点：



要训练这个神经网络，就要先定义它的损失函数，对于一个输出Y^hat，是个4维向量，对于整个训练集的平均损失：

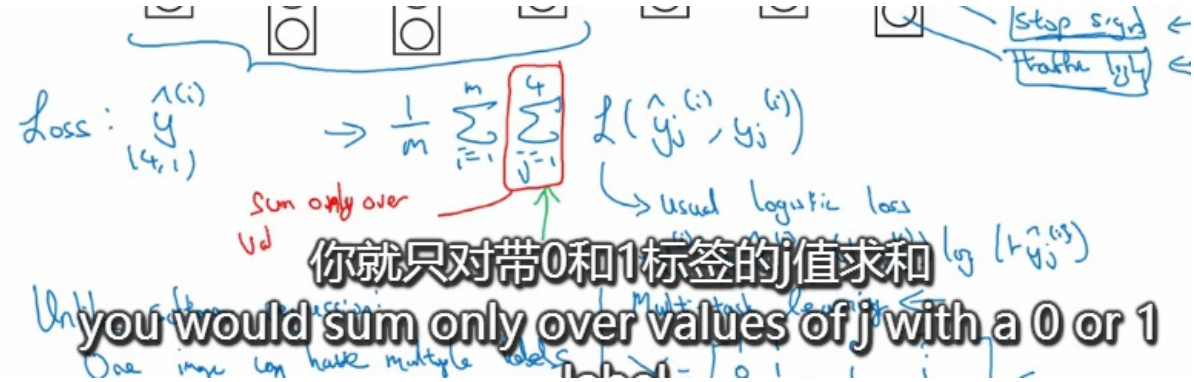




与softmax回归不同，softmax是将单个标签分给单个样本，而这张图有很多不同的标签，如果你试图最小化上面那个成本函数，你做的就是多任务学习，因为你现在做的是建立单个神经网络，

观察每张图片，然后解决4个问题，系统试图告诉你，每张图片里有没有这四个物体，另外你也可以训练4个不同的神经网络，而不是训练一个网络做4件事情，但是神经网络一些早期特征，在识别不同物体时都会用到，然后你发现，训练一个神经网络做4件事，会比训练四个完全独立的神经网络分别做4件事效果会更好，这就是多任务学习的力量。

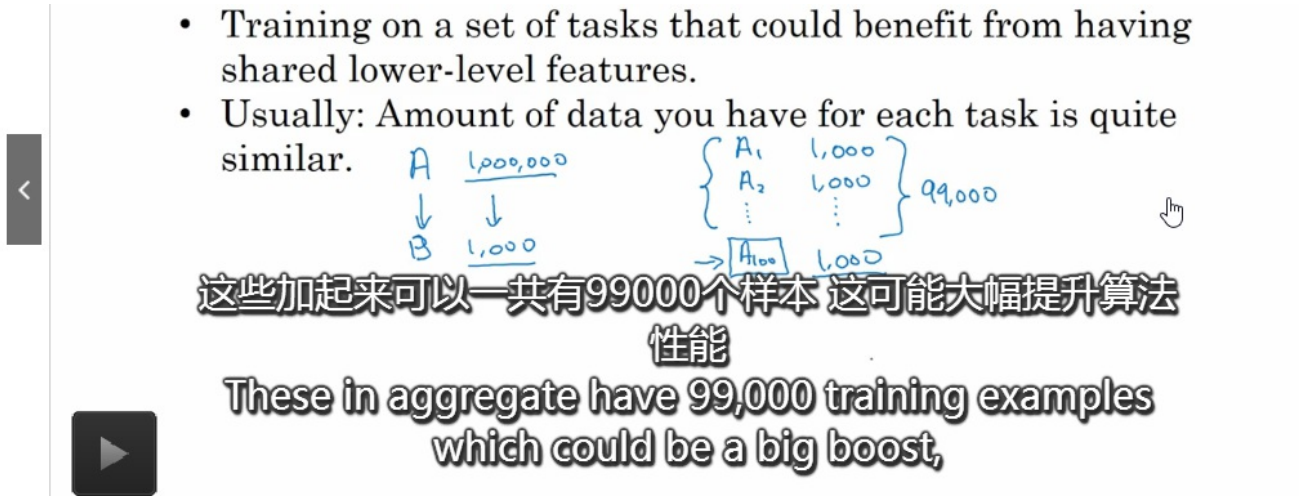
到目前为止，好像每一种图片都有全部标签，事实证明，多任务学习，也可以处理图片只有部分物体被标记的情况，所以第一个训练样本，我们可以说有人，给数据贴标签的人告诉你这里有一个行人，没有车，但是他们没有标记是否有停车标志和交通灯，也许有些样本都有标记，但也许有些样本他们只标记了有没有车，然后还有一些是问号，即使是这样的数据集，你也可以在上面训练算法，同时做四个任务，即使一些图像，只有一小部分标签，其他是问号，或者不管是什么，然后你训练算法的方式，即使这里有些标签是问号或者没有 标记：



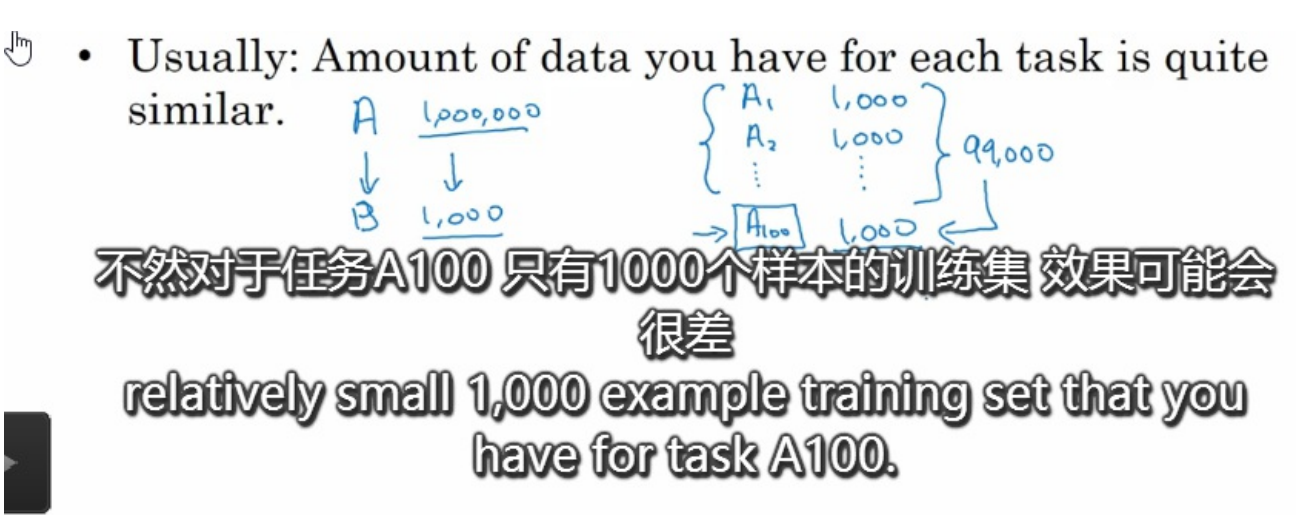
你可以在求和时忽略这些带问号的项，这样只对有标签的值求和，于是你就能利用这样的数据集，那么多任务学习什么时候有意义呢，当三件事为真时，他就是有意义的：

1、如果你训练的一组任务，可以共用低层次特征，对于无人驾驶的例子，同时识别交通灯汽车和行人是有道理的，这些物体有相似的特征，也许能帮你识别停车标记，因为这些都是道路上的特征。

2、这个准则没有那么绝对，所以不一定是对的，如果每个任务的数据量很接近，你还记得迁移学习时，你从A任务学到知识，然后迁移到B任务，所以如果任务A有100万个样本？任务B只有100个样本，那么你从这100万个样本学到的知识，真的可以帮助你增强对更小数据集任务B的训练,多任务学习可以把每个任务的数据集合在一起训练：



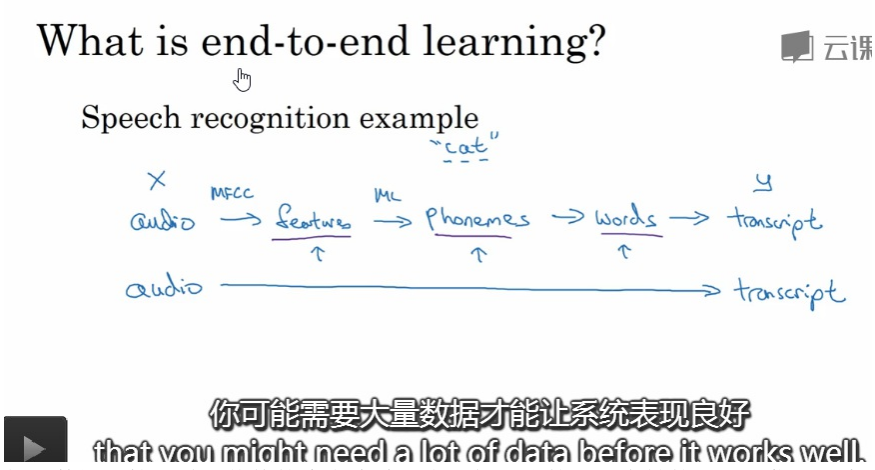
这样就可以提供很多知识来增强某个任务的性能比如A100这个任务：



多任务学习会降低性能的唯一情况和训练单个神经网络性能更低的情况：就是你的神经网络还不够大，如果你训练一个足够大的神经网络，那么多任务学习，肯定不会或者很少降低性能，我们都希望它可以提高性能，比单独完成各个任务性能更好。

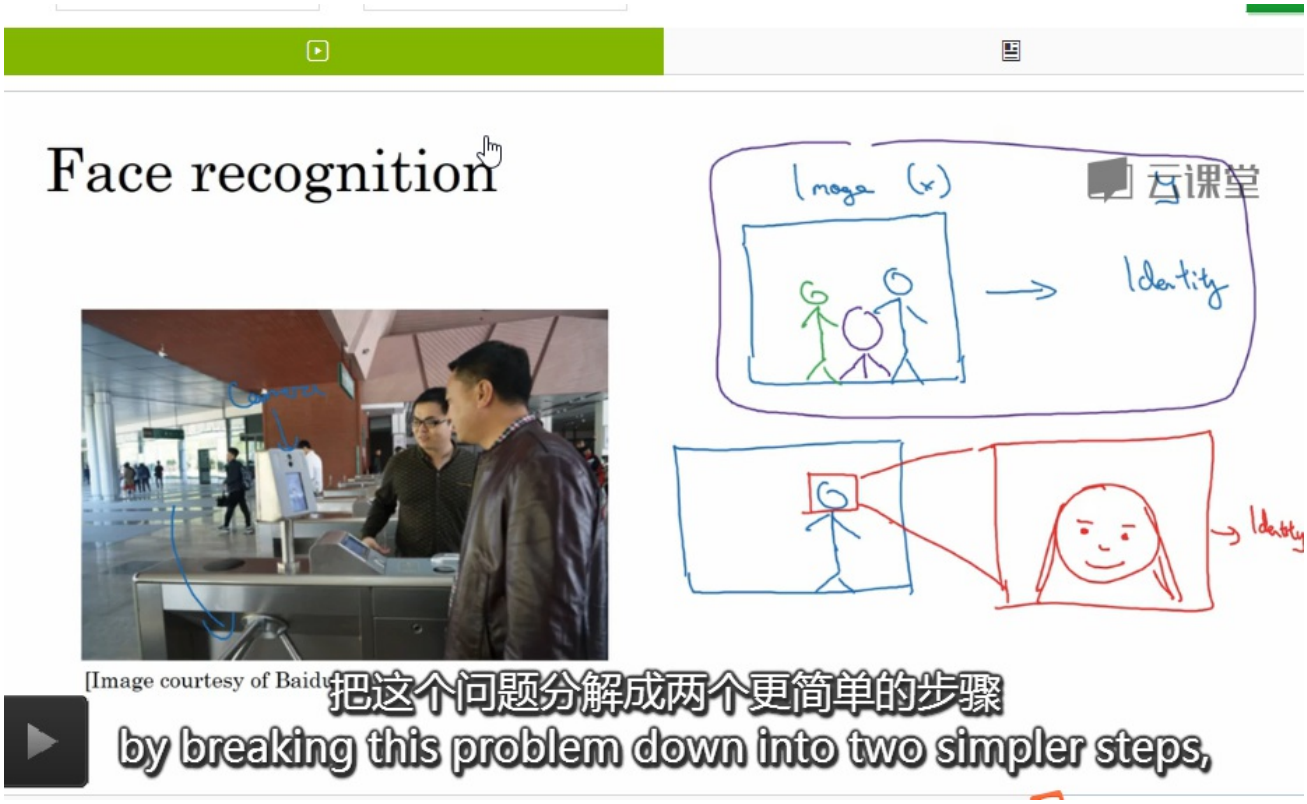
9、端到端的深度学习：

缺点是：

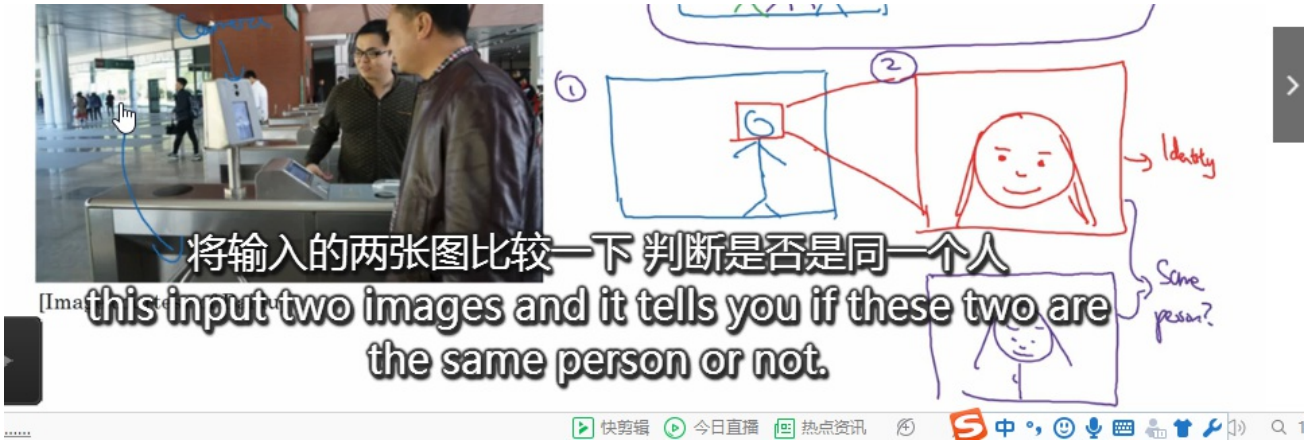


如果数据量较小时，传统的流水线效果也是很不错的，通常做的更好，你需要大量的数据，才能让端到端的方法真正发挥耀眼的光芒。

人脸识别常用的方法是，可以先运行一个软件把人脸提取出来然后喂到网络上去的：



第一步就是把脸取出来，第二步就是看看这脸是谁。第二步其实就是输入两张图片：



为什么两步法更好，第一是你解决了两个问题，每个问题实际上要简单的多，但第二，两个子任务的训练数据都很多，具体来说，有很多数据可以用于人脸识别训练，对于这里的任务一来说，任务就是观察一张图片，找出人脸所在的位置，把人脸图片框出来，所以有很多数据，有很多标签数据X,Y，X是图片，Y时表示人脸的位置，你可以建立一个神经网络，可以很好的处理任务一，然后任务二，也有很多数据可用，所以输入一张裁剪的很紧凑的图片。

10、是否要使用端到端的深度学习：

优点：

1、让数据说话，只要你有足够的数据，训练足够大的神经网络，可能更能捕获数据中的任何统计信息，而不是被迫引入人的成见，

2、所需手工设计组件更少。

缺点：

1、需要大量的数据。

2、它排除了可能有用的手工设置的组件。

学习算法有两个主要的知识来源，一个是数据，另一个是手工设计的任何东西，人工设计的组件可能非常有用的，但是它也可能伤害到算法的表现。