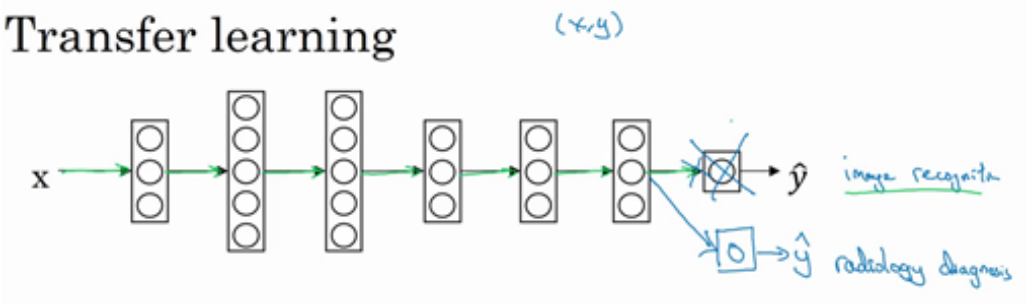
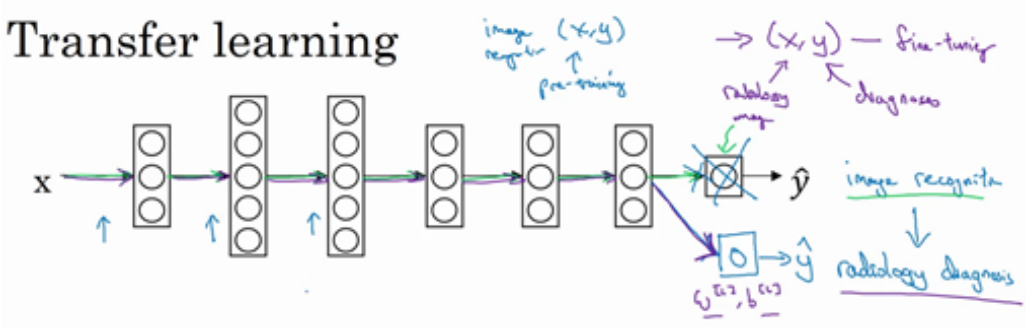
2.7 迁移学习（Transfer learning）

深度学习中，最强大的理念之一就是，有的时候**神经网络可以从一个任务中习得知识，并将这些知识应用到另一个独立的任务中。**所以例如，也许你已经训练好一个神经网络，能够识别像猫这样的对象，然后使用那些知识，或者部分习得的知识去帮助您更好地阅读 x 射线扫描图，这就是所谓的迁移学习。

我们来看看，假设你已经训练好一个图像识别神经网络，所以你首先用一个神经网络，并在(x, y)对上训练，其中x是图像，y是某些对象，图像是猫、狗、鸟或其他东西。如果你把这个神经网络拿来，然后让它适应或者说迁移，在不同任务中学到的知识，比如放射科诊断，就是说阅读X射线扫描图。你可以做的是把神经网络最后的输出层拿走，就把它删掉，还有进入到最后一层的权重删掉，然后为最后一层重新赋予随机权重，然后让它在放射诊断数据上训练。



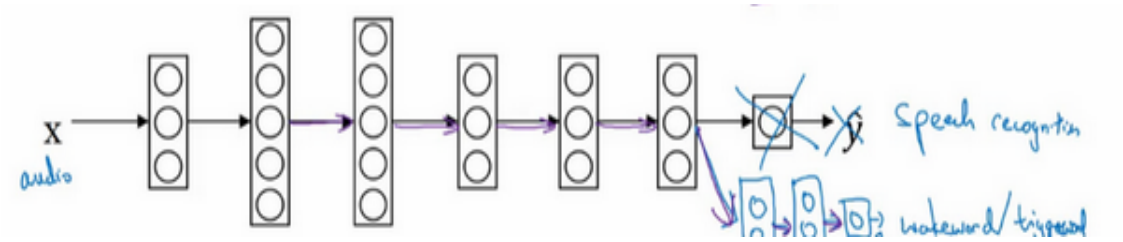
具体来说，在第一阶段训练过程中，当你进行图像识别任务训练时，你可以训练神经网络的所有常用参数，所有的权重，所有的层，然后你就得到了一个能够做图像识别预测的网络。在训练了这个神经网络后，要实现迁移学习，你现在要做的是，把数据集换成新的(x, y)对，现在这些变成放射科图像，而y是你想要预测的诊断，你要做的是初始化最后一层的权重，让我们称之为$w^{[L]}$和$b^{[L]}$随机初始化。



现在，我们在这个新数据集上重新训练网络，在新的放射科数据集上训练网络。要用放

射科数据集重新训练神经网络有几种做法。你可能，如果你的放射科数据集很小，你可能只需要重新训练最后一层的权重，就是$w^{[L]}$和$b^{[L]}$，并保持其他参数不变。如果你有足够多的数据，你可以重新训练神经网络中剩下的所有层。经验规则是，如果你有一个小数据集，就只训练输出层前的最后一层，或者也许是最后一两层。但是如果你有很多数据，那么也许你可以重新训练网络中的所有参数。如果你重新训练神经网络中的所有参数，那么这个在图像识别数据的初期训练阶段，有时称为预训练（pre-training），因为你在用图像识别数据去预先初始化，或者预训练神经网络的权重。然后，如果你以后更新所有权重，然后在放射科数据上训练，有时这个过程叫微调（fine tuning）。如果你在深度学习文献中看到预训练和微调，你就知道它们说的是这个意思，预训练和微调的权重来源于迁移学习。

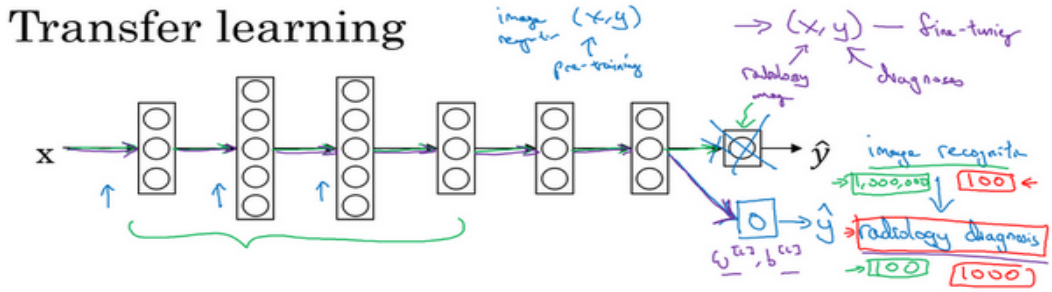
在这个例子中你做的是，把图像识别中学到的知识应用或迁移到放射科诊断上来，为什么这样做有效果呢？有很多低层次特征，比如说边缘检测、曲线检测、阳性对象检测（**positive objects**），从非常大的图像识别数据库中习得这些能力可能有助于你的学习算法在放射科诊断中做得更好，算法学到了很多结构信息，图像形状的信息，其中一些知识可能会很有用，所以学会了图像识别，它就可能学到足够多的信息，可以了解不同图像的组成部分是怎样的，学到线条、点、曲线这些知识，也许对象的一小部分，这些知识有可能帮助你的放射科诊断网络学习更快一些，或者需要更少的学习数据。



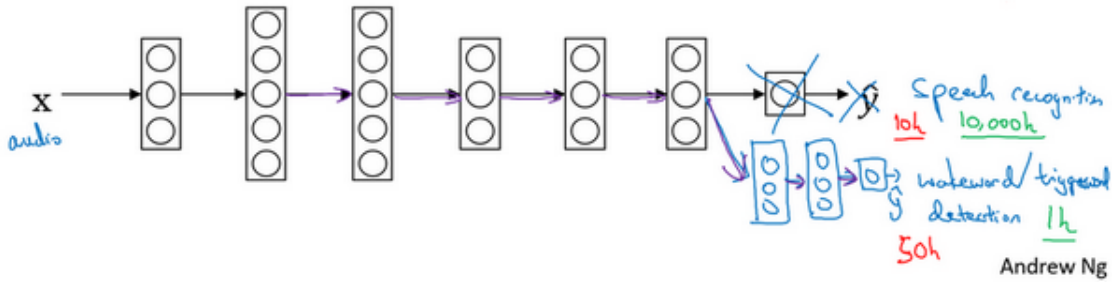
这里是另一个例子，假设你已经训练出一个语音识别系统，现在x是音频或音频片段输入，而y是听写文本，所以你已经训练了语音识别系统，让它输出听写文本。现在我们说你想搭建一个“唤醒词”或“触发词”检测系统，所谓唤醒词或触发词就是我们说的一句话，可以唤醒家里的语音控制设备，比如你说“**Alexa**”可以唤醒一个亚马逊 **Echo** 设备，或用“**OK Google**”来唤醒 **Google** 设备，用"**Hey Siri**"来唤醒苹果设备，用"你好百度"唤醒一个百度设备。要做到这点，你可能需要去掉神经网络的最后一层，然后加入新的输出节点，但有时你可以不只加入一个新节点，或者甚至往你的神经网络加入几个新层，然后把唤醒词检测问题的标签y喂进去训练。再次，这取决于你有多少数据，你可能只需要重新训练网络的新层，也许你需要重新训练神经网络中更多的层。

那么迁移学习什么时候是有意义的呢？迁移学习起作用的场合是，在迁移来源问题中

你有很多数据，但迁移目标问题你没有那么多数据。例如，假设图像识别任务中你有 1 百万个样本，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，也许你只有一百个样本，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。



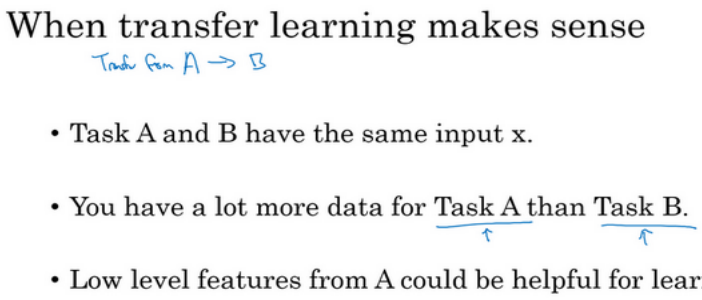
对于语音识别，也许你已经用 10,000 小时数据训练过你的语言识别系统，所以你从这 10,000 小时数据学到了很多人类声音的特征，这数据量其实很多了。但对于触发字检测，也许你只有 1 小时数据，所以这数据太小，不能用来拟合很多参数。所以在这种情况下，预先学到很多人类声音的特征人类语言的组成部分等等知识，可以帮你建立一个很好的唤醒字检测器，即使你的数据集相对较小。对于唤醒词任务来说，至少数据集要小得多。



所以在这两种情况下，你从数据量很多的问题迁移到数据量相对小的问题。然后反过来的话，迁移学习可能就没有意义了。比如，你用 100 张图训练图像识别系统，然后有 100 甚至 1000 张图用于训练放射科诊断系统，人们可能会想，为了提升放射科诊断的性能，假设你真的希望这个放射科诊断系统做得好，那么用放射科图像训练可能比使用猫和狗的图像更有价值，所以这里（100 甚至 1000 张图用于训练放射科诊断系统）的每个样本价值比这里（100 张图训练图像识别系统）要大得多，至少就建立性能良好的放射科系统而言是这样。所以，如果你的放射科数据更多，那么你这 100 张猫猫狗狗或者随机物体的图片肯定不会有太大帮助，因为来自猫狗识别任务中，每一张图的价值肯定不如一张X射线扫描图有价值，对于建立良好的放射科诊断系统而言是这样。

所以，这是其中一个例子，说明迁移学习可能不会有害，但也别指望这么做可以带来有

意义的增益。同样，如果你用 10 小时数据训练出一个语音识别系统。然后你实际上有 10 个小时甚至更多，比如说 50 个小时唤醒字检测的数据，你知道迁移学习有可能会有帮助，也可能不会，也许把这 10 小时数据迁移学习不会有太大坏处，但是你也别指望会得到有意义的增益。



所以总结一下，什么时候迁移学习是有意义的？如果你想从任务A学习并迁移一些知识到任务B，那么当任务A和任务B都有同样的输入x时，迁移学习是有意义的。在第一个例子中，A和B的输入都是图像，在第二个例子中，两者输入都是音频。当任务A的数据比任务B多得多时，迁移学习意义更大。所有这些假设的前提都是，你希望提高任务B的性能，因为任务B每个数据更有价值，对任务B来说通常任务A的数据量必须大得多，才有帮助，因为任务A里单个样本的价值没有比任务B单个样本价值大。然后如果你觉得任务A的低层次特征，可以帮助任务B的学习，那迁移学习更有意义一些。

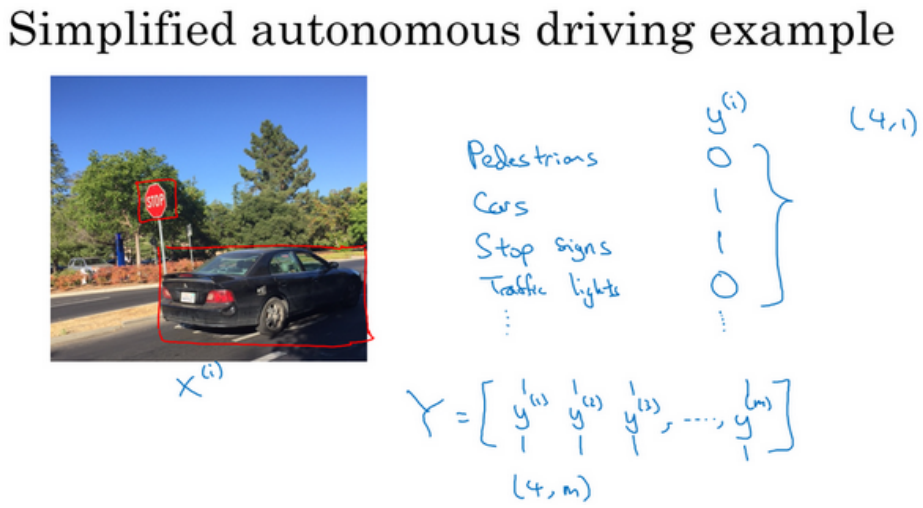
所以总结一下，迁移学习最有用的场合是，如果你尝试优化任务B的性能，通常这个任务数据相对较少，例如，在放射科中你知道很难收集很多X射线扫描图来搭建一个性能良好的放射科诊断系统，所以在这种情况下，你可能会找一个相关但不同的任务，如图像识别，其中你可能用 1 百万张图片训练过了，并从中学到很多低层次特征，所以那也许能帮助网络在任务B在放射科任务上做得更好，尽管任务B没有这么多数据。迁移学习什么时候是有意义的？它确实可以显著提高你的学习任务的性能，但我有时候也见过有些场合使用迁移学习时，任务A实际上数据量比任务B要少，这种情况下增益可能不多。

好，这就是迁移学习，你从一个任务中学习，然后尝试迁移到另一个不同任务中。从多个任务中学习还有另外一个版本，就是所谓的多任务学习，当你尝试从多个任务中并行学习，而不是串行学习，在训练了一个任务之后试图迁移到另一个任务，所以在下一个视频中，让我们来讨论多任务学习。

2.8 多任务学习（Multi-task learning）

在迁移学习中，你的步骤是串行的，你从任务A里学习只是然后迁移到任务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能帮到其他所有任务。

我们来看一个例子，假设你在研发无人驾驶车辆，那么你的无人驾驶车可能需要同时检测不同的物体，比如检测行人、车辆、停车标志，还有交通灯各种其他东西。比如在左边这个例子中，图像里有个停车标志，然后图像中有辆车，但没有行人，也没有交通灯。

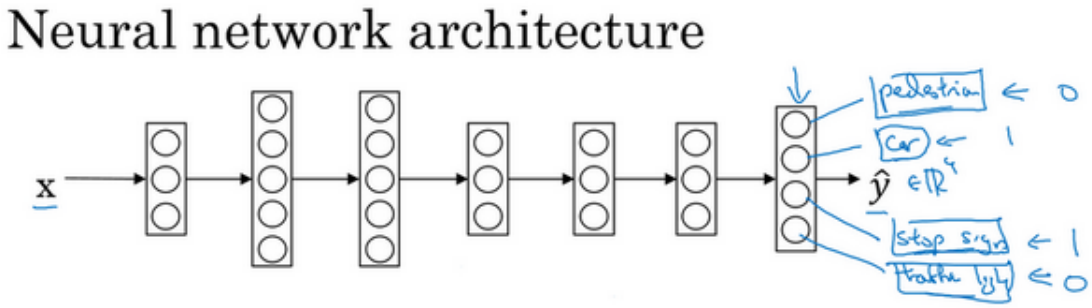


如果这是输入图像$x^{(i)}$，那么这里不再是一个标签 $y^{(i)}$，而是有 4 个标签。在这个例子中，没有行人，有一辆车，有一个停车标志，没有交通灯。然后如果你尝试检测其他物体，

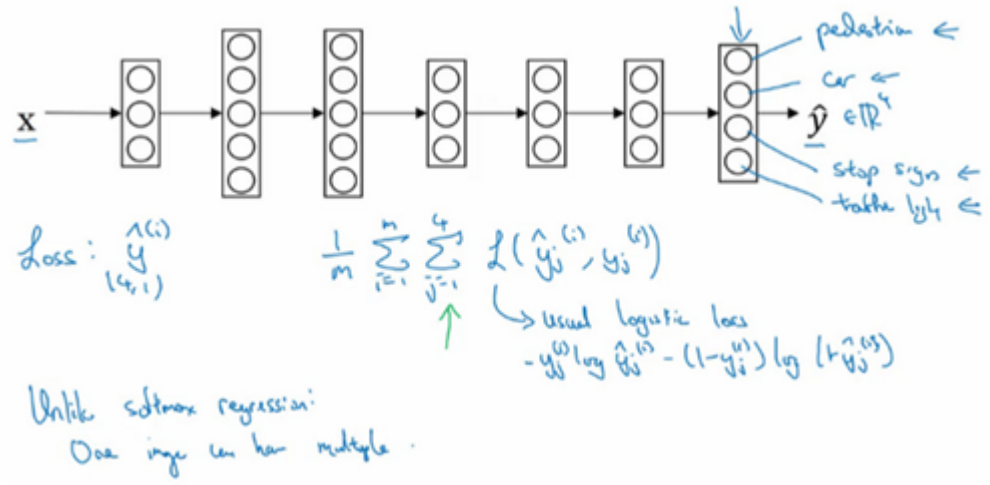
也许 $y^{(i)}$的维数会更高，现在我们就先用 4 个吧，所以 $y^{(i)}$是个 4×1 向量。如果你从整体来看这个训练集标签和以前类似，我们将训练集的标签水平堆叠起来，像这样$y^{(1)}$一直到$y^{(m)}$：

$$Y = \begin{bmatrix} | & | & | & \cdots & | \\ y^{(1)} & y^{(2)} & y^{(3)} & \cdots & y^{(m)} \\ | & | & | & \cdots & | \end{bmatrix}$$

不过现在$y^{(i)}$是 4×1 向量，所以这些都是竖向的列向量，所以这个矩阵Y现在变成$4 \times m$矩阵。而之前，当y是单实数时，这就是$1 \times m$矩阵。



那么你现在可以做的是训练一个神经网络，来预测这些y值，你就得到这样的神经网络，输入x，现在输出是一个四维向量y。请注意，这里输出我画了四个节点，所以第一个节点就是我们想预测图中有没有行人，然后第二个输出节点预测的是有没有车，这里预测有没有停车标志，这里预测有没有交通灯，所以这里$\hat{y}$是四维的。



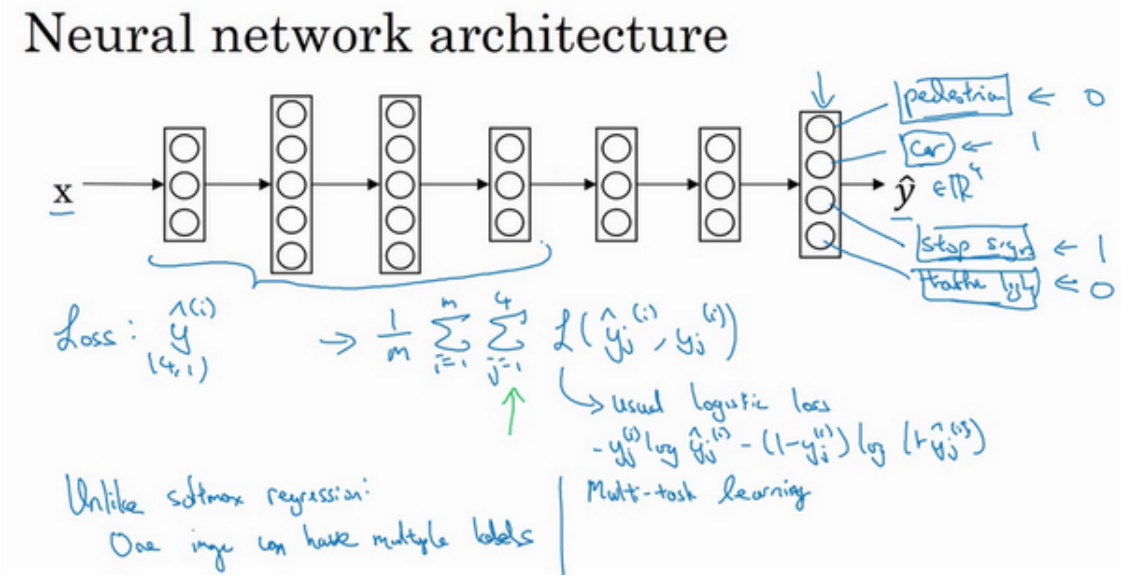
要训练这个神经网络，你现在需要定义神经网络的损失函数，对于一个输出$\hat{y}$，是个 4 维向量，对于整个训练集的平均损失：

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{4} L(\hat{y}_j^{(i)}, y_j^{(i)})$$

$\sum_{j=1}^{4} L(\hat{y}_j^{(i)}, y_j^{(i)})$这些单个预测的损失，所以这就是对四个分量的求和，行人、车、停车标志、交通灯，而这个标志L指的是 **logistic 损失**，我们就这么写：

$$L(\hat{y}_j^{(i)}, y_j^{(i)}) = -y_j^{(i)} \log \hat{y}_j^{(i)} - (1 - y_j^{(i)}) log(1 - \hat{y}_j^{(i)})$$

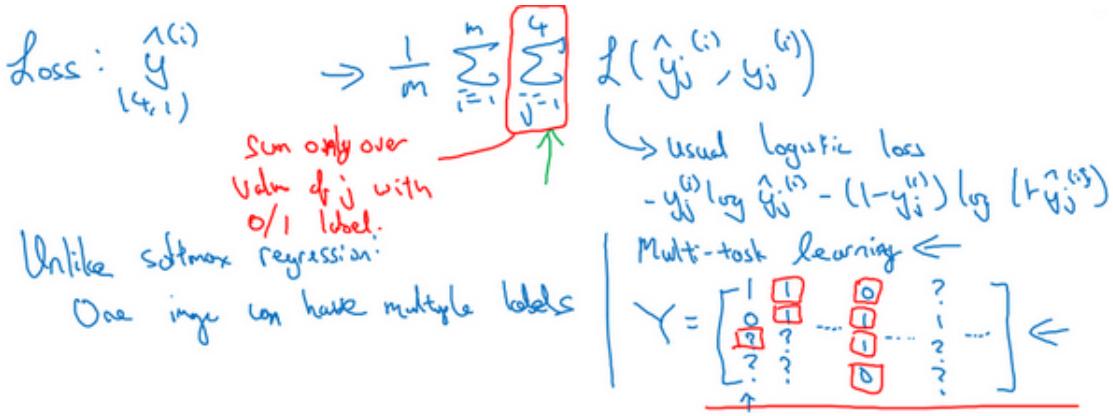
整个训练集的平均损失和之前分类猫的例子主要区别在于，现在你要对$j = 1$到4求和，这与 **softmax** 回归的主要区别在于，与 **softmax** 回归不同，**softmax** 将单个标签分配给单个样本。



而这张图可以有很多不同的标签，所以不是说每张图都只是一张行人图片，汽车图片、停车标志图片或者交通灯图片。你要知道每张照片是否有行人、或汽车、停车标志或交通灯，多个物体可能同时出现在一张图里。实际上，在上一张幻灯片中，那张图同时有车和停车标志，但没有行人和交通灯，所以你不是只给图片一个标签，而是需要遍历不同类型，然后看看每个类型，那类物体有没有出现在图中。所以我就说在这个场合，一张图可以有多个标签。如果你训练了一个神经网络，试图最小化这个成本函数，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，然后解决四个问题，系统试图告诉你，每张图里面有没有这四个物体。另外你也可以训练四个不同的神经网络，而不是训练一个网络做四件事情。但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四件事情会比训练四个完全独立的神经网络分别做四件事性能要更好，这就是多任务学习的力量。

另一个细节，到目前为止，我是这么描述算法的，好像每张图都有全部标签。事实证明，多任务学习也可以处理图像只有部分物体被标记的情况。所以第一个训练样本，我们说有人，给数据贴标签的人告诉你里面有一个行人，没有车，但他们没有标记是否有停车标志，或者是否有交通灯。也许第二个例子中，有行人，有车。但是，当标记人看着那张图片时，他们

没有加标签，没有标记是否有停车标志，是否有交通灯等等。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，这就是对j从 1 到 4 求和，你就只对带 0 和 1 标签的j值求和，所以当有问号的时候，你就在求和时忽略那个项，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，它就是有意义的。

When multi-task learning makes sense

- Training on a set of tasks that could benefit from having shared lower-level features.
- Usually: Amount of data you have for each task is quite similar.

- Can train a big enough neural network to do well on all the tasks.

第一，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志，因为这些都是道路上的特征。

第二，这个准则没有那么绝对，所以不一定是对的。但我从很多成功的多任务学习案例中看到，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 1 百万个样本，任务B只有 1000 个样本，那么你从这 1 百万

个样本学到的知识，真的可以帮你增强对更小数据集任务B的训练。那么多任务学习又怎么样呢？在多任务学习中，你通常有更多任务而不仅仅是两个，所以也许你有，以前我们有 4 个任务，但比如说你要完成 100 个任务，而你要做多任务学习，尝试同时识别 100 种不同类型的物体。你可能会发现，每个任务大概有 1000 个样本。所以如果你专注加强单个任务的性能，比如我们专注加强第 100 个任务的表现，我们用$A100$表示，如果你试图单独去做这个最后的任务，你只有 1000 个样本去训练这个任务，这是 100 项任务之一，而通过在其他 99 项任务的训练，这些加起来可以一共有 99000 个样本，这可能大幅提升算法性能，可以提供很多知识来增强这个任务的性能。不然对于任务$A100$，只有 1000 个样本的训练集，效果可能会很差。如果有对称性，这其他 99 个任务，也许能提供一些数据或提供一些知识来帮到这 100 个任务中的每一个任务。所以第二点不是绝对正确的准则，但我通常会看的是如果你专注于单项任务，如果想要从多任务学习得到很大性能提升，那么其他任务加起来必须要有比单个任务大得多的数据量。要满足这个条件，其中一种方法是，比如右边这个例子这样，或者如果每个任务中的数据量很相近，但关键在于，如果对于单个任务你已经有 1000 个样本了，那么对于所有其他任务，你最好有超过 1000 个样本，这样其他任务的知识才能帮你改善这个任务的性能。

最后多任务学习往往在以下场合更有意义，当你可以训练一个足够大的神经网络，同时做好所有的工作，所以多任务学习的替代方法是为每个任务训练一个单独的神经网络。所以不是训练单个神经网络同时处理行人、汽车、停车标志和交通灯检测。你可以训练一个用于行人检测的神经网络，一个用于汽车检测的神经网络，一个用于停车标志检测的神经网络和一个用于交通信号灯检测的神经网络。那么研究员 **Rich Carona** 几年前发现的是什么呢？多任务学习会降低性能的唯一情况，和训练单个神经网络相比性能更低的情况就是你的神经网络还不够大。但如果你可以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会降低性能，我们都希望它可以提升性能，比单独训练神经网络来单独完成各个任务性能要更好。

所以这就是多任务学习，在实践中，多任务学习的使用频率要低于迁移学习。我看到很多迁移学习的应用，你需要解决一个问题，但你的训练数据很少，所以你需要找一个数据很多的相关问题来预先学习，并将知识迁移到这个新问题上。但多任务学习比较少见，就是你需要同时处理很多任务，都要做好，你可以同时训练所有这些任务，也许计算机视觉是一个例子。在物体检测中，我们看到更多使用多任务学习的应用，其中一个神经网络尝试检测一大堆物体，比分别训练不同的神经网络检测物体更好。但我说，平均来说，目前迁移学习使

用频率更高，比多任务学习频率要高，但两者都可以成为你的强力工具。

所以总结一下，多任务学习能让你训练一个神经网络来执行许多任务，这可以给你更高的性能，比单独完成各个任务更高的性能。但要注意，实际上迁移学习比多任务学习使用频率更高。我看到很多任务都是，如果你想解决一个机器学习问题，但你的数据集相对较小，那么迁移学习真的能帮到你，就是如果你找到一个相关问题，其中数据量要大得多，你就能以它为基础训练你的神经网络，然后迁移到这个数据量很少的任务上来。

今天我们学到了很多和迁移学习有关的问题，还有一些迁移学习和多任务学习的应用。但多任务学习，我觉得使用频率比迁移学习要少得多，也许其中一个例外是计算机视觉，物体检测。在那些任务中，人们经常训练一个神经网络同时检测很多不同物体，这比训练单独的神经网络来检测视觉物体要更好。但平均而言，我认为即使迁移学习和多任务学习工作方式类似。实际上，我看到用迁移学习比多任务学习要更多，我觉得这是因为你很难找到那么多相似且数据量对等的任务可以用单一神经网络训练。再次，在计算机视觉领域，物体检测这个例子是最显著的例外情况。

所以这就是多任务学习，多任务学习和迁移学习都是你的工具包中的重要工具。最后，我想继续讨论端到端深度学习，所以我们来看下一个视频来讨论端到端学习。