1. 误差分析

通过人工检查机器学习模型得出的结果的一些错误，有助于深入了解下一步要进行的工作。这个过程被称作错误分析（Error Analysis）。

例如，我们训练一个猫图片识别器，最终得到了90%的精确度，即有10%的错误率，分类器错误将一些看上去像猫的狗识别为猫了。这时，立即盲目地去研究一个能够精确识别出狗的算法不一定是最好的选择，因为我们不知道这样做会对提高分类器的准确率有多大的帮助。

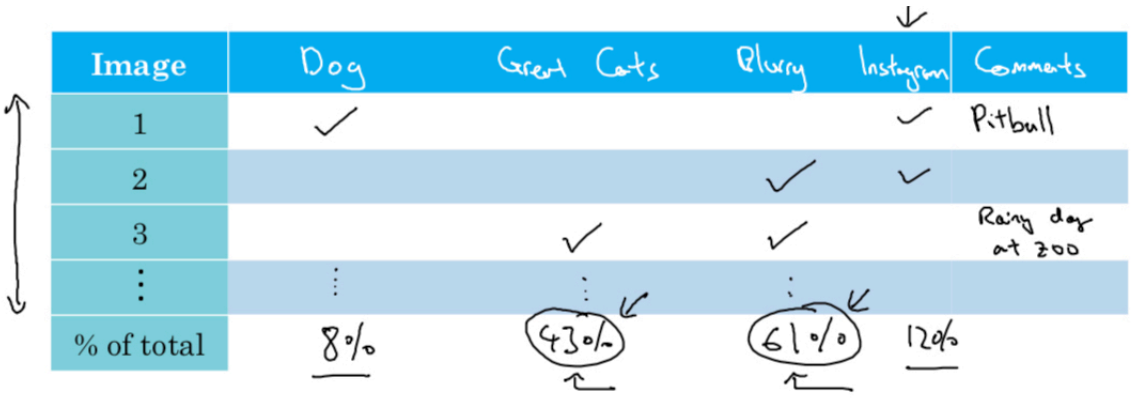
这时，可以做如下错误分析：

- 从开发集中收集大约100个错误标记的样本；
- 统计其中有多少是狗。

根据统计结果：

1. 假设100个样本中，有5例样本是狗，那么我们对数据集的错误标记做努力去改进模型的精度，那么可以提升的上限就是5%，即仅仅可以达到9.5%的错误率，这有时称为**性能上限**。那么这种情况下，可能这样耗时的努力方向就不是很值得的一件事。
2. 另外一种假设是100个数据中，有50多个样例都是狗，那么这种情况下，我们去改进数据集的错误标记，就是一个比较值得的改进方向，可以将模型的精度提升至95%。

在对输出结果中分类错误的样本进行人工分析时，可以建立一个表格来记录每一个分类错误的具体信息，例如某些图像是模糊的，或者是把狗识别成了猫等，并统计属于不同错误类型的错误数量，这样，分类结果会更加清晰。



总结一下，进行错误分析时，你应该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，通过统计不同错误标记类型占总数的百分比，有助于发现哪些问题亟需解决，或者提供构思新优化方向的灵感。

2. 修正错误标记

我们用mislabeled examples来表示学习算法输出了错误的Y值。而在做误差分析时，有时会注意到数据中有些样本被人为地错误标记（incorrectly labeled）了，该怎么做?

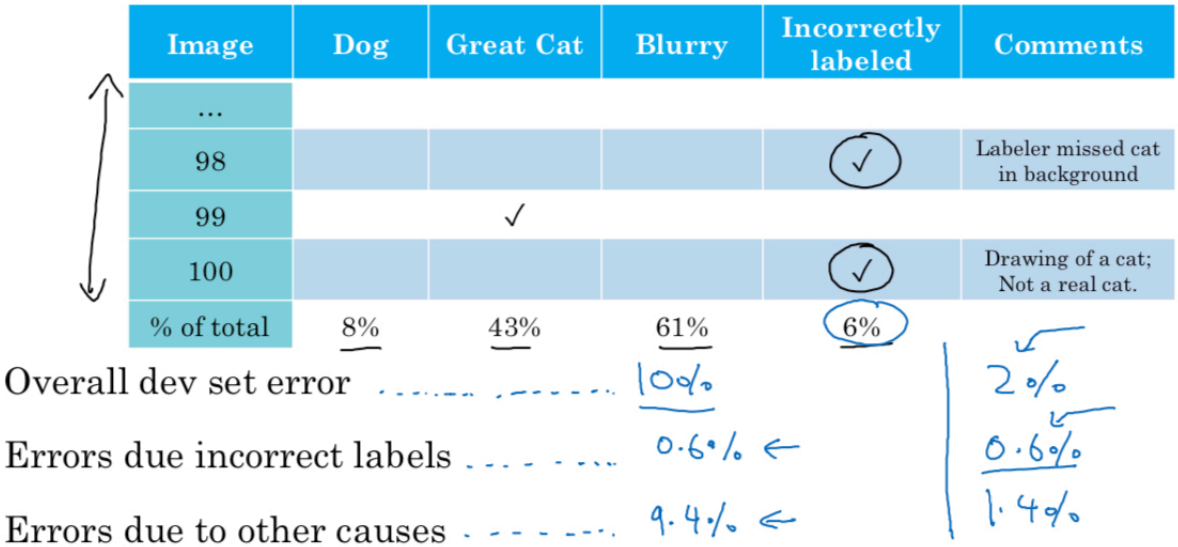
训练集错误标记

如果是在训练集中，由于深度学习算法对于随机误差具有很好的**鲁棒性（Robust）**，只要这些出错的样本数量较小，且分布近似随机，就不必花时间一一修正。

虽然深度学习算法对随机误差有很好的鲁棒性，但对系统误差就不是这样了。比如，标记人员一直将白色的狗标记为猫，那么最终会导致分类器出错。

开发/测试集错误标记

如果在开发集和测试集中出现了错误标记的问题，我们可以在误差分析的过程中，增加错误标记这一原因，再对错误的数据进行分析，得出修正这些标记错误的价值。



上图左边，因为错误标记占比很小，则可忽略开发/测试集中的错误标记；而右边错误标记占所有错误的30%，因为有必要修正错误标记。

当你决定在开发和测试集上手动检查标签并进行修正时，有一些额外的方针和原则需要考虑：

- 在开发集和测试集上同时使用同样的修正手段，以保证开发集和测试集来自相同的分布；
- 同时检查判断正确和判断错误的样本；
- 在修正开发集和测试集时，鉴于训练集的分布不必和开发/测试集完全相同，可以不去修正训练集。

3. 快速搭建系统再迭代

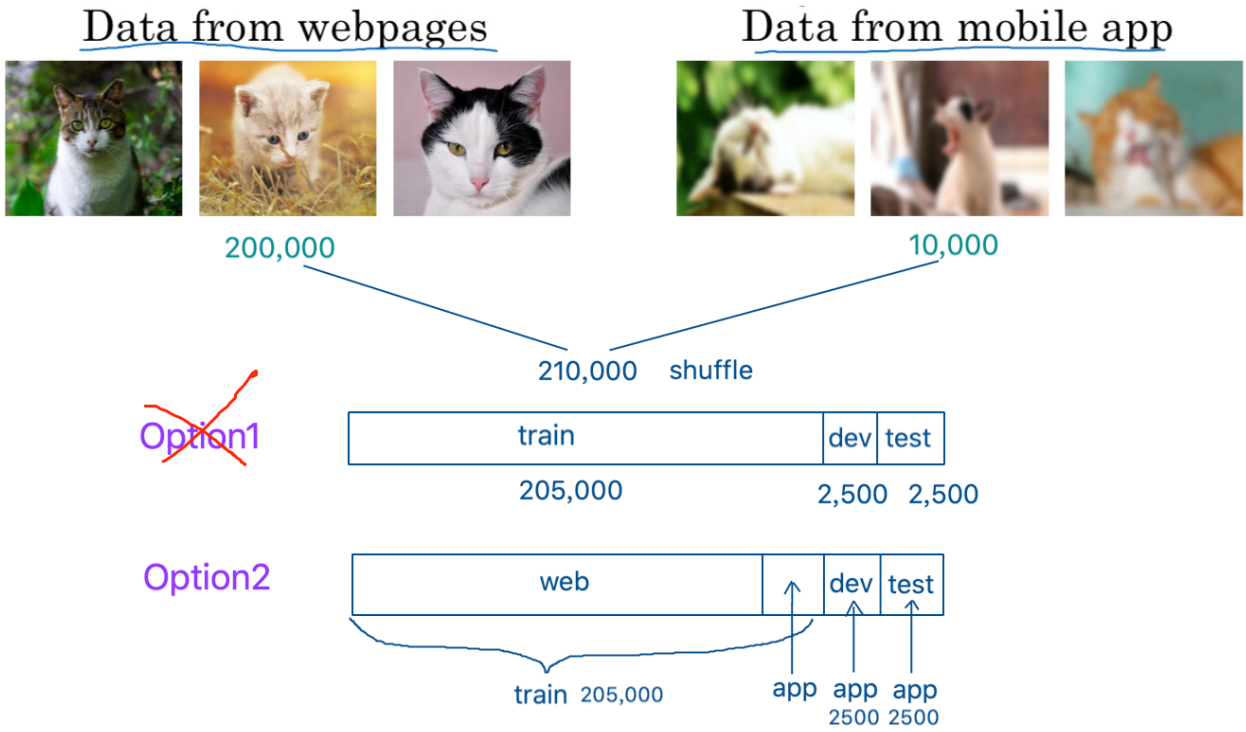
如果想搭建一个全新的机器学习系统，建议根据以下步骤快速搭建好第一个系统，然后开始迭代：

1. 设置好训练、验证、测试集及衡量指标，确定目标；
2. 快速训练出一个初步的系统，用训练集来拟合参数，用验证集调参，用测试集评估；
3. 通过偏差/方差分析以及错误分析等方法，决定下一步优先处理的方向。

4. 在不同的分布上训练和测试

在深度学习时代，因为需求的数据量非常大，现在很多的团队，使用的训练数据都是和开发集和测试集来自不同的分布。

下面是一些处理训练集和测试集存在差异的最佳做法，还是以猫分类问题为例：



方法一

将两组数据合并到一起，随机打散后分配到训练、开发和测试集中，这样做：

- 好处：三个集合中的数据同分布；
- 坏处：我们设置开发集的目的是瞄准目标，而现在我们绝大部分目标是为了优化从网上获取的高清照片，而不是我们的真正目标。

因此，这种处理方式不是一个好的方法。

方法二

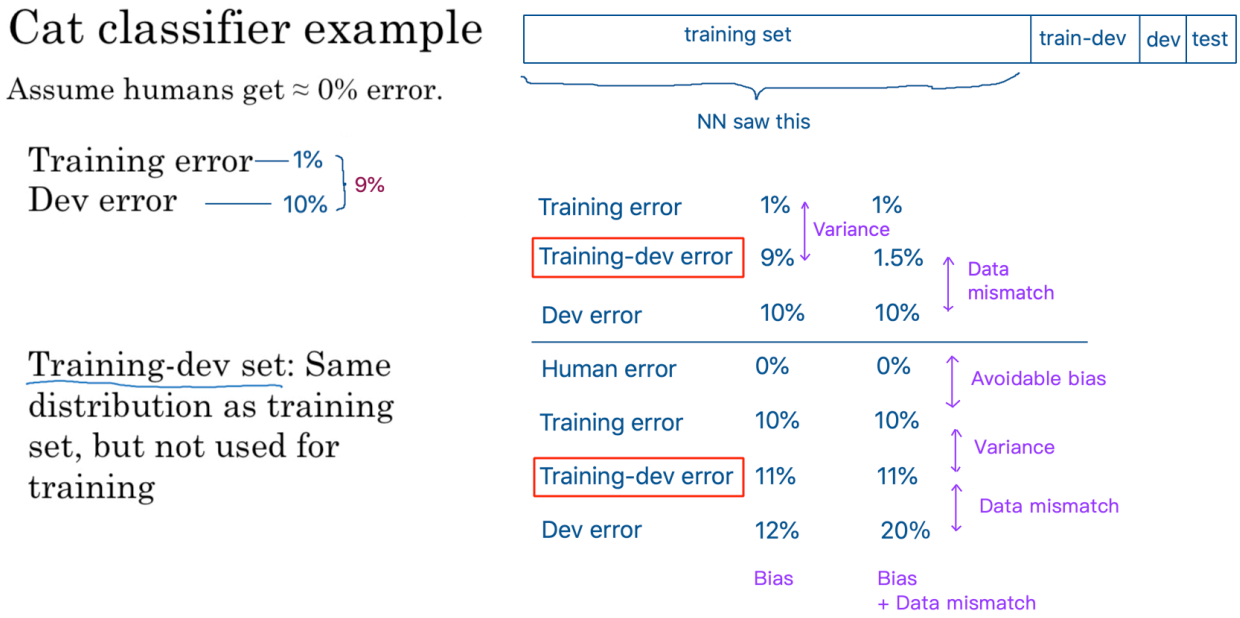
训练集均是来自网上下载的20万张高清图片，当然也可以加上5000张手机非高清图片；而开发集和测试集都是手机非高清图片：

- 好处：开发集全部来自手机图片，瞄准目标；
- 坏处：训练集和开发测试集来自不同分布

从长期来看，这样的分布能够给我们带来更好的系统性能。

5. 不同分布上的偏差和方差

通过估计学习算法的偏差和方差，可以帮助我们确定接下来应该优先努力的方向。但是当我们的训练集和开发、测试集来自不同的分布时，分析偏差和方差的方式就有一定的不同。



由于训练集和开发、测试集来自不同分布，因此，模型在训练集和开发集上的误差差很难说是模型存在方差问题，还是因为数据偏差导致。

这时可以设置**训练开发集（Training-dev set）**，其中的数据与训练集同分布，但不用于训练。

通过分析模型在不同数据集上的误差，可以分析得到模型到底是存在Avoidable bias、Variance还是Data mismatch问题。

6. 解决数据分布不匹配方法

通过上一节的误差分析可以得知，模型最终在开发和测试集上的误差可能是由于数据分布不匹配导致的，这种情况该如何解决?

- 进行人工误差分析，尝试去了解训练集和开发测试集的具体差异在哪里，如：背景噪音等；
- 尝试把训练数据变得更像开发集，或者收集更多的类似开发集和测试集的数据，如增加噪音。

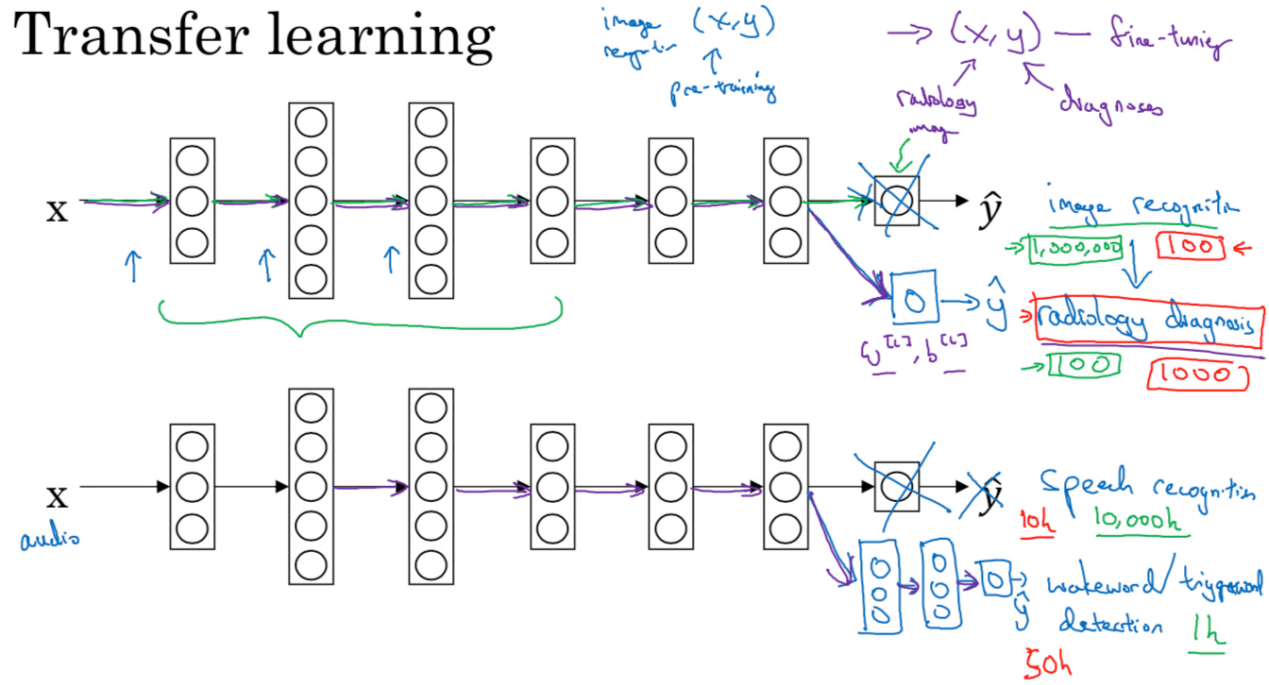
如果你打算将训练数据调整得更像验证集，可以使用的一种技术是**人工合成数据（Artificial Data Synthesis）**。以语音识别问题为例，实际应用场合（开发/测试集）是包含背景噪声的，而作为训练样本的音频很可能是清晰而没有背景噪声的。为了让训练集和开发/测试集分布一致，我们可以给训练集人工添加背景噪音，合成类似的实际场景的声音。

人工合成数据能够使数据集匹配，从而提升模型的效果。但需要注意的是，不能给每段语音都增加同一段背景噪声，因为这样模型会对这段背景噪音出现过拟合现象，使得效果不佳。

7. 迁移学习

迁移学习（Transfer Learning）是通过将已训练好的神经网络模型的一部分网络结构应用到另一个模型， 将一个神经网络从某个任务中学到的知识和经验运用到另一个任务中，以显著提高学习任务的性能。

例如，我们将为识别猫构建的神经网络迁移运用到放射科诊断中。因为猫识别器的神经网络已经学习到了有关图像的结构和性质等方面的知识，所以只要先删除神经网络中原有的输出层，加入新的输出层并随机初始化权重（$W^{[L]}$、$b^{[L]}$），随后用新的训练集进行训练，就完成了以上的迁移学习。



如果新的数据集很小，可能只需要重新训练输出层的最后一层权重，即$W^{[L]}$、$b^{[L]}$，并保持其他参数不变；而如果有足够多的数据，可以只保留网络结构，重新训练网络中的所有层参数。这时初始权重由之前的模型训练得到，这个过程称为**预训练（Pre-Training）**，之后的权重更新过程称为**微调（Fine-Tuning）**。

迁移学习有效的情况：

1. 任务A和任务B有相同的输入；
2. 任务A所拥有的数据量要远远大于任务B（对于更有价值的任务B，任务A所拥有的数据要比B大得多）；
3. 任务A的低层特征学习对任务B有一定的帮助。

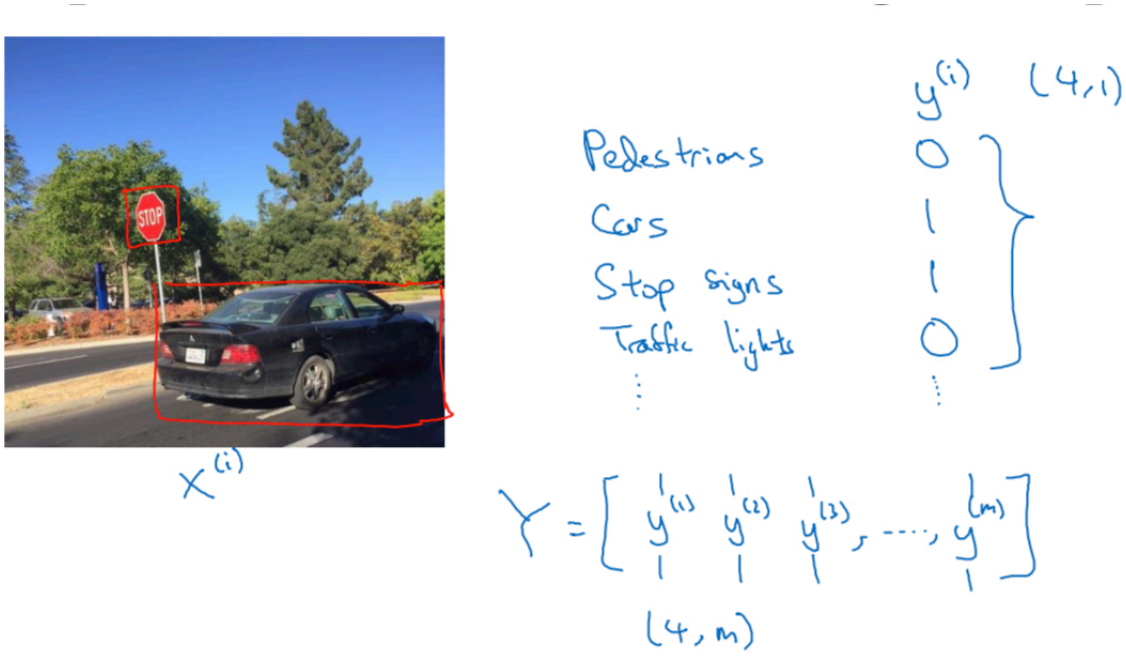
8. 多任务学习

迁移学习中的步骤是串行的；而**多任务学习（Multi-Task Learning）**使用单个神经网络模型，利用共享表示采用并行训练同时学习多个任务。多任务学习的基本假设是多个任务之间具有相关性，并且任务之间可以利用相关性相互促进。例如，属性分类中，抹口红和戴耳环有一定的相关性，单独训练的时候是无法利用这些信息，多任务学习则可利用任务相关性联合提高多个属性分类的精度。

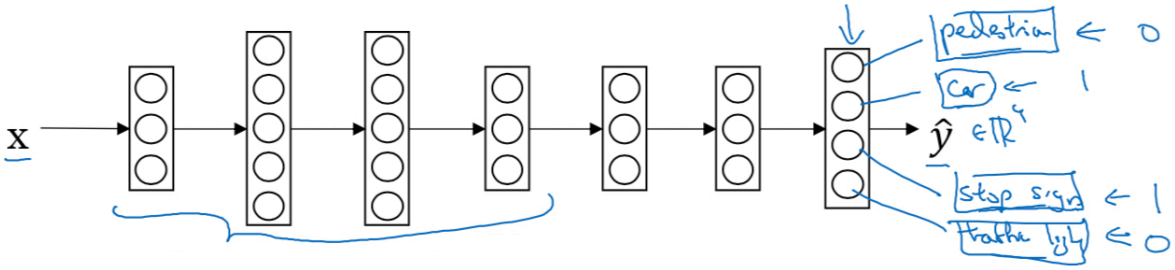
自动驾驶的例子

假设在自动驾驶任务中，我们需要同时检测的物体很多，比如行人、汽车、交通标志、信号灯等。

对于想在的任务，我们的目标值变成了一个向量的形式，向量中的每一个值代表检测到是否有如行人、汽车、交通标志和信号灯等。



多任务的神经网络结构如下图所示：



$$\text{Loss}\quad \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{4}L(\hat{y}_j^{(i)},\ y_j^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{4}(y_j^{(i)}\log\hat{y}_j^{(i)} + (1-y_j^{(i)})\log(1-\hat{y}_j^{(i)}))$$

对于这样的问题，我们就是在做多任务学习，因为我们建立单个神经网络，来解决多个问题。

多任务学习和Softmax回归看上去有些类似，容易混淆。它们的区别是，Softmax回归的输出向量y中只有一个元素为1；而多任务学习的输出向量y中可以有多个元素为1。

特定的一些问题，例如上面的例子中，数据集中可能只标注了部分信息，如其中一张只标注了人，汽车和信号灯的标识没有进行标注。那么对于这样的数据集，我们依旧可以用多任务学习来训练模型。当然要注意这里的loss function求和时，只对带0，1标签的j进行求和。

$$Y = \begin{bmatrix} 1 & 0 & ? & ? \\ 0 & 1 & 0 & 1 \\ 0 & ? & 1 & 0 \\ ? & 1 & 1 & 0 \end{bmatrix}$$

在下述场合进行多任务学习是有意义的：

1. 训练的一组任务可以共用低层次特征；
2. 通常，每个任务的数据量接近；
3. 能够训练一个足够大的神经网络，以同时做好所有的工作。多任务学习会降低性能的唯一情况（即和为每个任务训练单个神经网络相比性能更低的情况）是神经网络还不够大。

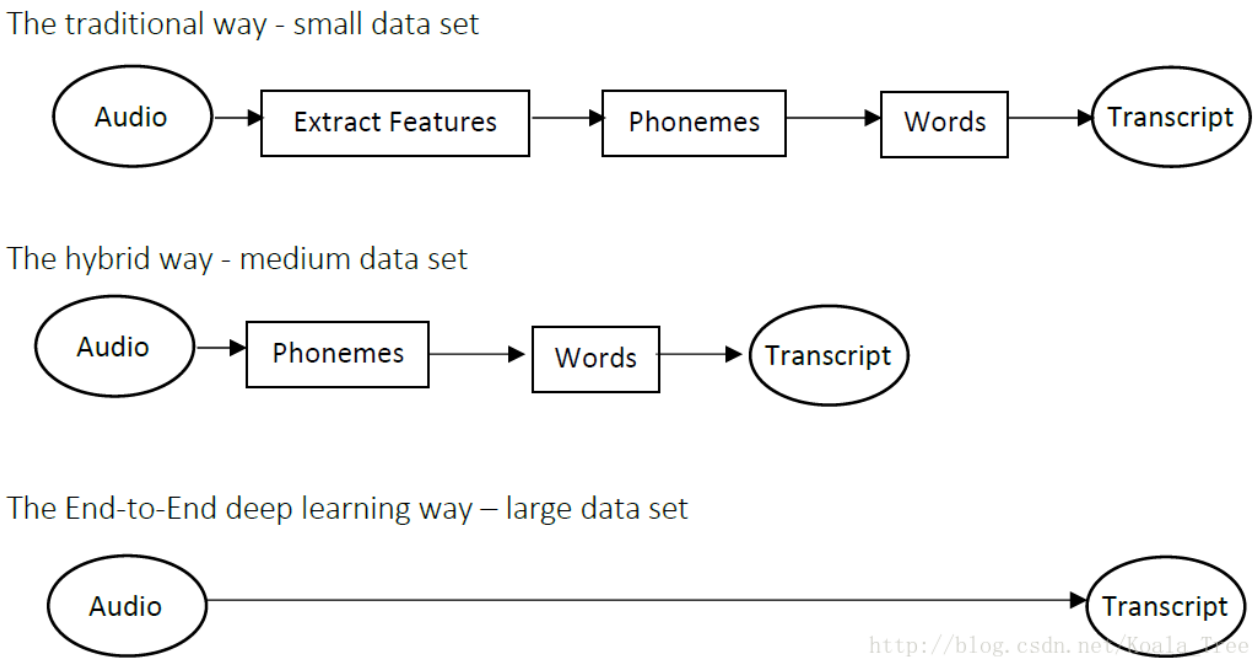
在实践中，多任务学习的使用频率要远低于迁移学习，计算机视觉中的物体识别是个多任务学习的例子。

9. 端到端学习

在传统的机器学习分块模型中，每一个模块处理一种输入，然后其输出作为下一个模块的输入，构成一条流水线。而**端到端深度学习（End-to-end Deep Learning）**只用一个单一的神经网络模型来实现所有的功能。它将所有模块混合在一起，只关心输入和输出。

如果数据量较少，传统机器学习分块模型所构成的流水线效果会很不错。但如果训练样本足够大，并且训练出的神经网络模型足够复杂，那么端到端深度学习模型的性能会比传统机器学习分块模型更好。

而如果数据集规模适中，还是可以使用流水线方法，但是可以混合端到端深度学习，通过神经网络绕过某些模块，直接输出某些特征。



端到端学习的优点：

1. 只要有足够多的数据，剩下的全部交给一个足够大的神经网络。比起传统的机器学习分块模型，可能更能捕获数据中的任何统计信息，而不需要用人类固有的认知（或者说，成见）来进行分析；
2. 所需手工设计的组件更少，简化设计工作流程。

缺点：

1. 需要大量的数据；
2. 排除了可能有用的人工设计组件。

根据以上分析，决定一个问题是否应用端到端学习的关键是：是否有足够的数据，支持能够直接学习从x映射到y并且足够复杂的函数？