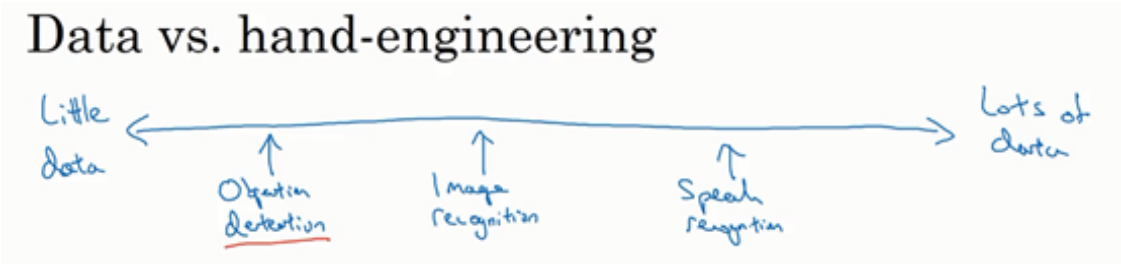
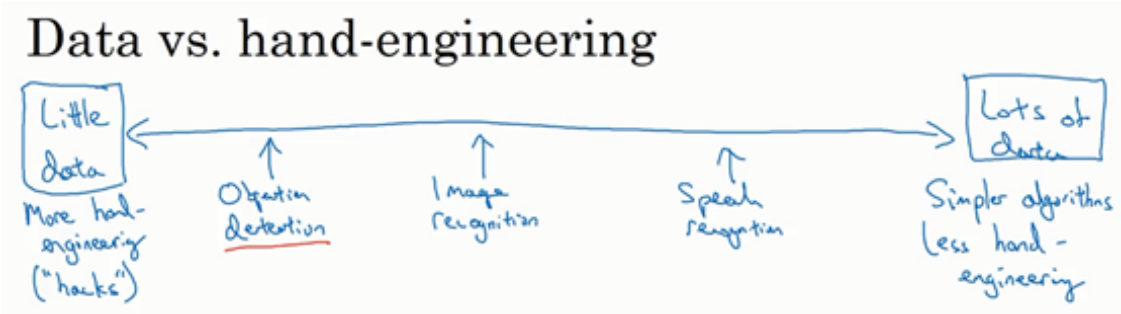
2.11 计算机视觉现状（The state of computer vision）

深度学习已经成功地应用于计算机视觉、自然语言处理、语音识别、在线广告、物流还有其他许多问题。在计算机视觉的现状下，深度学习应用于计算机视觉应用有一些独特之处。在这个视频中，我将和你们分享一些我对深度学习在计算机视觉方面应用的认识，希望能帮助你们更好地理解计算机视觉作品（此处指计算机视觉或者数据竞赛中的模型）以及其中的想法，以及如何自己构建这些计算机视觉系统。



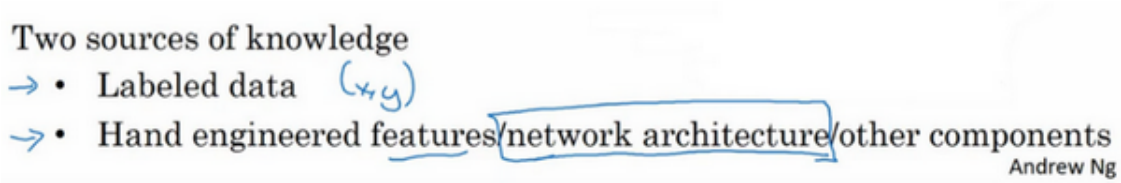
你可以认为大部分机器学习问题是介于少量数据和大量数据范围之间的。举个例子，我认为今天我们有相当数量的语音识别数据，至少相对于这个问题的复杂性而言。虽然现在图像识别或图像分类方面有相当大的数据集，因为图像识别是一个复杂的问题，通过分析像素并识别出它是什么，感觉即使在线数据集非常大，如超过一百万张图片，我们仍然希望我们能有更多的数据。还有一些问题，比如物体检测，我们拥有的数据更少。提醒一下，图像识别其实是如何看图片的问题，并且告诉你这张图是不是猫，而对象检测则是看一幅图，你画一个框，告诉你图片里的物体，比如汽车等等。因为获取边框的成本比标记对象的成本更高，所以我们进行对象检测的数据往往比图像识别数据要少，对象检测是我们下周要讨论的内容。



所以，观察一下机器学习数据范围图谱，你会发现当你有很多数据时，人们倾向于使用更简单的算法和更少的手工工程，因为我们不需要为这个问题精心设计特征。当你有大量的数据时，只要有一个大型的神经网络，甚至一个更简单的架构，可以是一个神经网络，就可

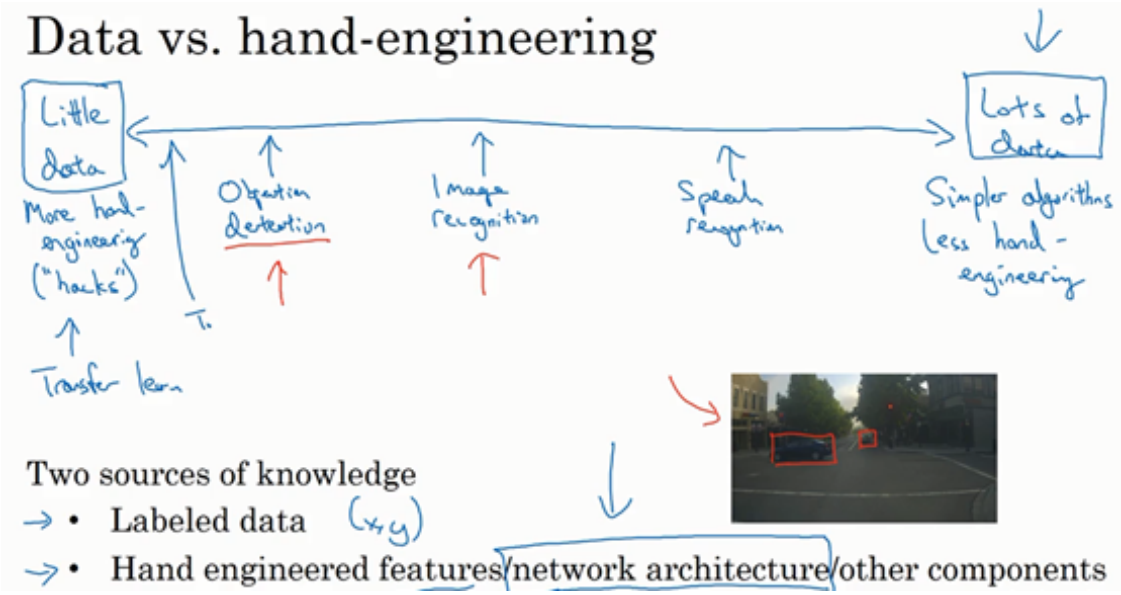
以去学习它想学习的东西。

相反当你没有那么多的数据时，那时你会看到人们从事更多的是手工工程，低调点说就是你有很多小技巧可用（整理者注：在机器学习或者深度学习中，一般更崇尚更少的人工处理，而手工工程更多依赖人工处理，注意领会 **Andrew NG** 的意思）。但我认为当你没有太多数据时，手工工程实际上是获得良好表现的最佳方式。



所以当我看机器学习应用时，我们认为通常我们的学习算法有两种知识来源，**一个来源是被标记的数据，就像(x, y)应用在监督学习。第二个知识来源是手工工程，有很多方法去建立一个手工工程系统，它可以是源于精心设计的特征，手工精心设计的网络体系结构或者是系统的其他组件。所以当你没有太多标签数据时，你只需要更多地考虑手工工程。**

所以我认为计算机视觉是在试图学习一个非常复杂的功能，我们经常感觉我们没有足够的数据，即使获得了更多数据，我们还是经常觉得还是没有足够的数据来满足需求。这就是为什么计算机视觉，从过去甚至到现在都更多地依赖于手工工程。我认为这也是计算机视觉领域发展相当复杂网络架构地原因，因为在缺乏更多数据的情况下，获得良好表现的方式还是花更多时间进行架构设计，或者说在网络架构设计上浪费（贬义褒用，即需要花费更多时间的意思）更多时间。



如果你认为我是在贬低手工工程，那并不是我的意思，当你没有足够的数据时，手工工

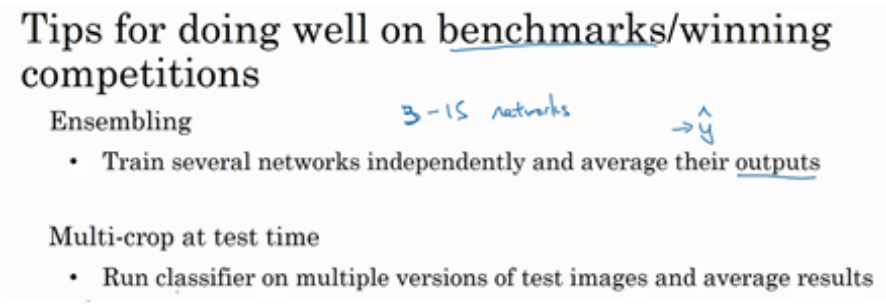
程是一项非常困难，非常需要技巧的任务，它需要很好的洞察力，那些对手工工程有深刻见解的人将会得到更好的表现。当你没有足够的数据时，手工工程对一个项目来说贡献就很大。当你有很多数据的时候我就不会花时间去做手工工程，我会花时间去建立学习系统。但我认为从历史而言，计算机视觉领域还只是使用了非常小的数据集，因此从历史上来看计算机视觉还是依赖于大量的手工工程。甚至在过去的几年里，计算机视觉任务的数据量急剧增加，我认为这导致了手工工程量大幅减少，但是在计算机视觉上仍然有很多的网络架构使用手工工程，这就是为什么你会在计算机视觉中看到非常复杂的超参数选择，比你在其他领域中要复杂的多。实际上，因为你通常有比图像识别数据集更小的对象检测数据集，当我们谈论对象检测时，其实这是下周的任务，你会看到算法变得更加复杂，而且有更多特殊的组件。

幸运的是，当你有少量的数据时，有一件事对你很有帮助，那就是迁移学习。我想说的是，在之前的幻灯片中，**Tigger**、**Misty** 或者二者都不是的检测问题中，我们有这么少的数据，迁移学习会有很大帮助。这是另一套技术，当你有相对较少的数据时就可以用很多相似的数据。

如果你看一下计算机视觉方面的作品，看看那里的创意，你会发现人们真的是踌躇满志，他们在基准测试中和竞赛中表现出色。对计算机视觉研究者来说，如果你在基准上做得很好了，那就更容易发表论文了，所以有许多人致力于这些基准上，把它做得很好。积极的一面是，它有助于整个社区找出最有效得算法。但是你在论文上也看到，人们所做的事情让你在数据基准上表现出色，但你不会真正部署在一个实际得应用程序用在生产或一个系统上。

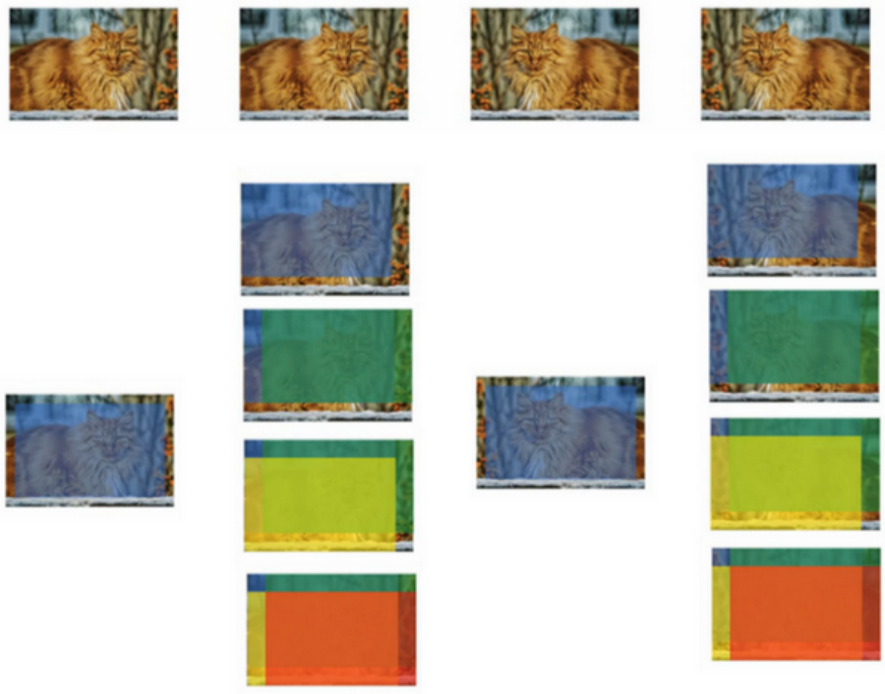
（整理着注：Benchmark 基准测试，**Benchmark** 是一个评价方式，在整个计算机领域有着长期的应用。维基百科上解释：“As computer architecture advanced, it became more difficult to compare the performance of various computer systems simply by looking at their specifications.Therefore, tests were developed that allowed comparison of different architectures.”**Benchmark** 在计算机领域应用最成功的就是性能测试，主要测试负载的执行时间、传输速度、吞吐量、资源占用率等。）

下面是一些有助于在基准测试中表现出色的小技巧，这些都是我自己从来没使用过的东西，如果我把一个系统投入生产，那就是为客户服务。

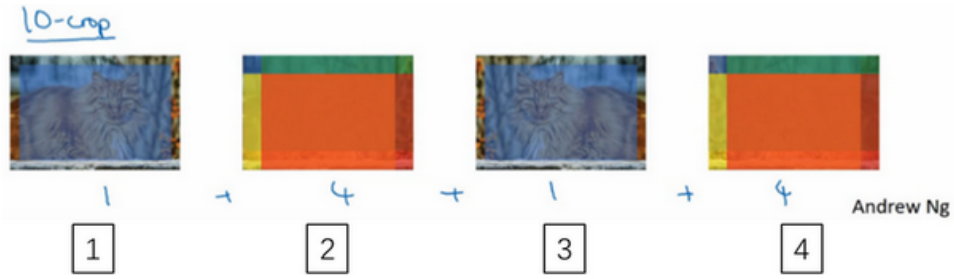


其中一个是集成，这就意味着在你想好了你想要的神经网络之后，可以独立训练几个神经网络，并平均它们的输出。比如说随机初始化三个、五个或者七个神经网络，然后训练所有这些网络，然后平均它们的输出。另外对他们的输出$\hat{y}$进行平均计算是很重要的，不要平均他们的权重，这是行不通的。看看你的 7 个神经网络，它们有 7 个不同的预测，然后平均他们，这可能会让你在基准上提高 1%，2%或者更好。这会让你做得更好，也许有时会达到 1%或 2%，这真的能帮助你赢得比赛。但因为集成意味着要对每张图片进行测试，你可能需要在从 3 到 15 个不同的网络中运行一个图像，这是很典型的，因为这 3 到 15 个网络可能会让你的运行时间变慢，甚至更多时间，所以技巧之一的集成是人们在基准测试中表现出色和赢得比赛的利器，但我认为这几乎不用于生产服务于客户的，我想除非你有一个巨大的计算预算而且不介意在每个用户图像数据上花费大量的计算。

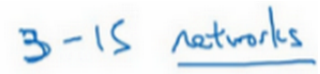
你在论文中可以看到在测试时，对进准测试有帮助的另一个技巧就是 Multi-crop at test time，我的意思是你已经看到了如何进行数据增强，**Multi-crop** 是一种将数据增强应用到你的测试图像中的一种形式。



举个例子，让我们看看猫的图片，然后把它复制四遍，包括它的两个镜像版本。有一种叫作 **10-crop** 的技术（**crop** 理解为裁剪的意思），它基本上说，假设你取这个中心区域，裁剪，然后通过你的分类器去运行它，然后取左上角区域，运行你的分类器，右上角用绿色表示，左下方用黄色表示，右下方用橙色表示，通过你的分类器来运行它，然后对镜像图像做同样的事情对吧？所以取中心的 **crop**，然后取四个角落的 **crop**。



这是这里（编号 1）和这里（编号 3）就是中心 **crop**，这里（编号 2）和这里（编号 4）就是四个角落的 **crop**。如果把这些加起来，就会有 10 种不同的图像的 **crop**，因此命名为 **10-crop**。所以你要做的就是，通过你的分类器来运行这十张图片，然后对结果进行平均。如果你有足够的计算预算，你可以这么做，也许他们需要 10 个 **crops**，你可以使用更多，这可能会让你在生产系统中获得更好的性能。如果是生产的话，我的意思还是实际部署用户的系统。但这是另一种技术，它在基准测试上的应用，要比实际生产系统中好得多。



集成的一个大问题是你需要保持所有这些不同的神经网络，这就占用了更多的计算机内存。对于 **multi-crop**，我想你只保留一个网络，所以它不会占用太多的内存，但它仍然会让你的运行时间变慢。

这些是你看到的小技巧，研究论文也可以参考这些，但我个人并不倾向于在构建生产系统时使用这些方法，尽管它们在基准测试和竞赛上做得很好。

由于计算机视觉问题建立在小数据集之上，其他人已经完成了大量的网络架构的手工工程。一个神经网络在某个计算机视觉问题上很有效，但令人惊讶的是它通常也会解决其他计算机视觉问题。

所以，要想建立一个实用的系统，你最好先从其他人的神经网络架构入手。如果可能的话，你可以使用开源的一些应用，因为开放的源码实现可能已经找到了所有繁琐的细节，比如学习率衰减方式或者超参数。

最后，其他人可能已经在几路 **GPU** 上花了几个星期的时间来训练一个模型，训练超过一百万张图片，所以通过使用其他人的预先训练得模型，然后在数据集上进行微调，你可以

在应用程序上运行得更快。当然如果你有电脑资源并且有意愿，我不会阻止你从头开始训练你自己的网络。事实上，如果你想发明你自己的计算机视觉算法，这可能是你必须要做的。

这就是本周的学习，我希望看到大量的计算机视觉架构能够帮助你理解什么是有效的。在本周的编程练习中，你实际上会学习另一种编程框架，并使用它来实现 **ResNets**。所以我希望你们喜欢这个编程练习，我期待下周还能见到你们。

参考文献：

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun - Deep Residual Learning for Image Recognition (2015)
- Francois Chollet's github repository: https://github.com/fchollet/deep-learning-models/blob/master/resnet50.py