

第一课

[主页](#) / [视频\(YouTube\)](#) / [视频\(bilibili\)](#) / [论坛第一课板块](#) / [论坛](#)

欢迎!

确保已经设置好了GPU环境并且可以运行Jupyter Notebook

[00_notebook_tutorial.ipynb](#)

快捷键:

- `Shift + Enter`: 运行单元里的代码, 选中下个单元
- `Up Arrow + Down Arrow`: 在单元间切换
- `b`: 创建新单元
- `0 + 0`: 重启内核

[2:45]

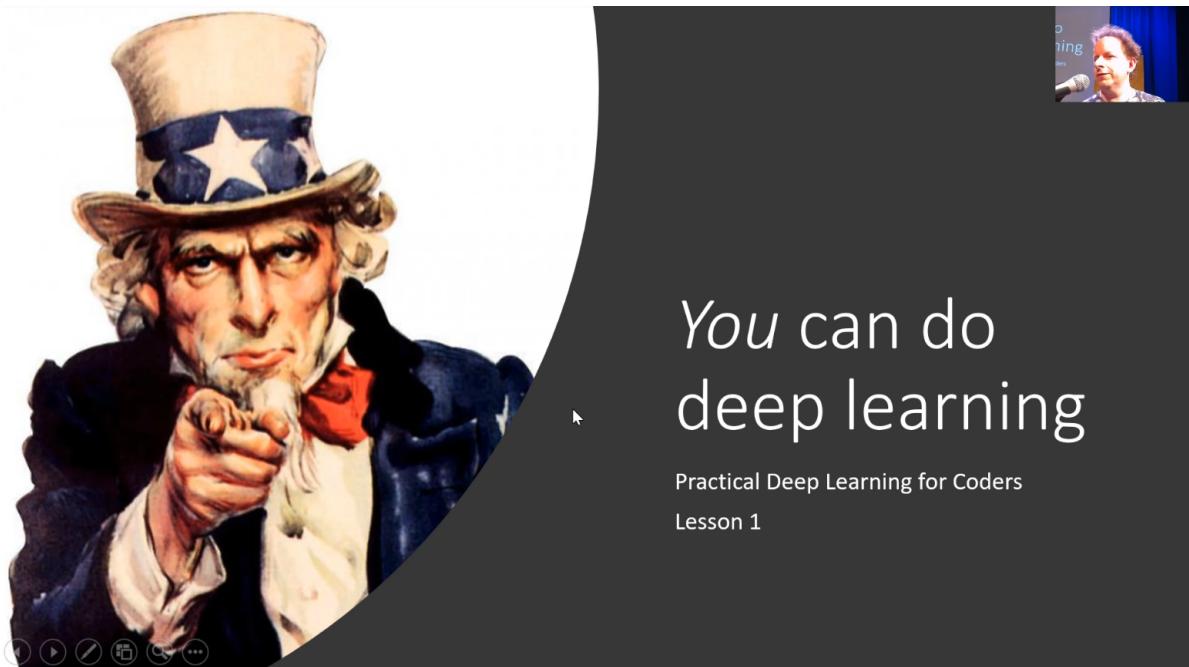
Jupyter Notebook 对于数据科学工作者来说是一个有趣的工具, 你不仅可以获得静态的信息, 还可以进行交互实验.

基于过去三年的经验, 这样使用notebook和学习材料最有效:

直接从头到尾看视频.

- 不必试图跟上课程内容, 课程节奏的被设计地比较快, 这样你可以快速地对各方面有一个初步的概念, 并且了解到这些方面是怎样结合在一起的
- 看完第一遍后, 再回过头来, 重新看一遍视频, 在需要的地方暂停, 按照视频里讲的自己做一遍. 确保你可以做出和我一样的结果, 然后在这个基础上用你自己的方式做些拓展.
- 第一遍看视频时不要运行代码, 中间不要停下来, 不必试图理解所有内容.

你可以做世界顶尖水平的深度学习 [[4:31](#)]



You can do deep learning

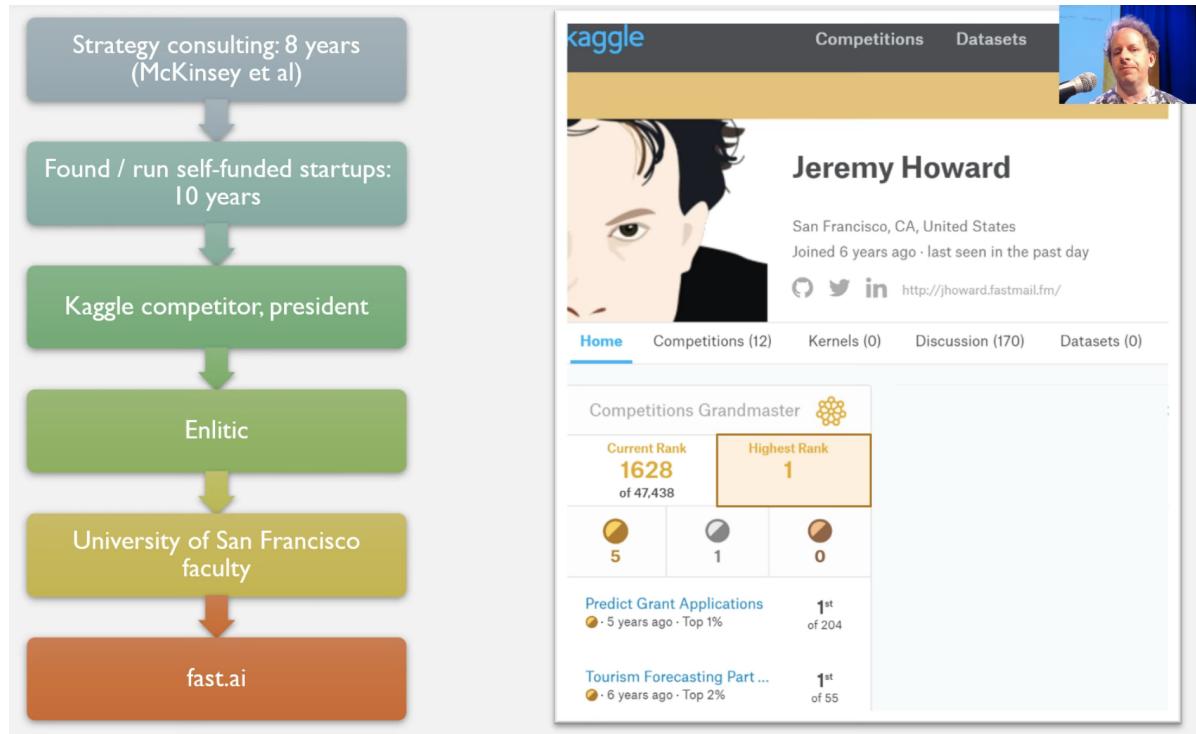
Practical Deep Learning for Coders

Lesson 1

在这里获取课程资源:

- <http://course-v3.fast.ai/>
- <https://forums.fast.ai/>

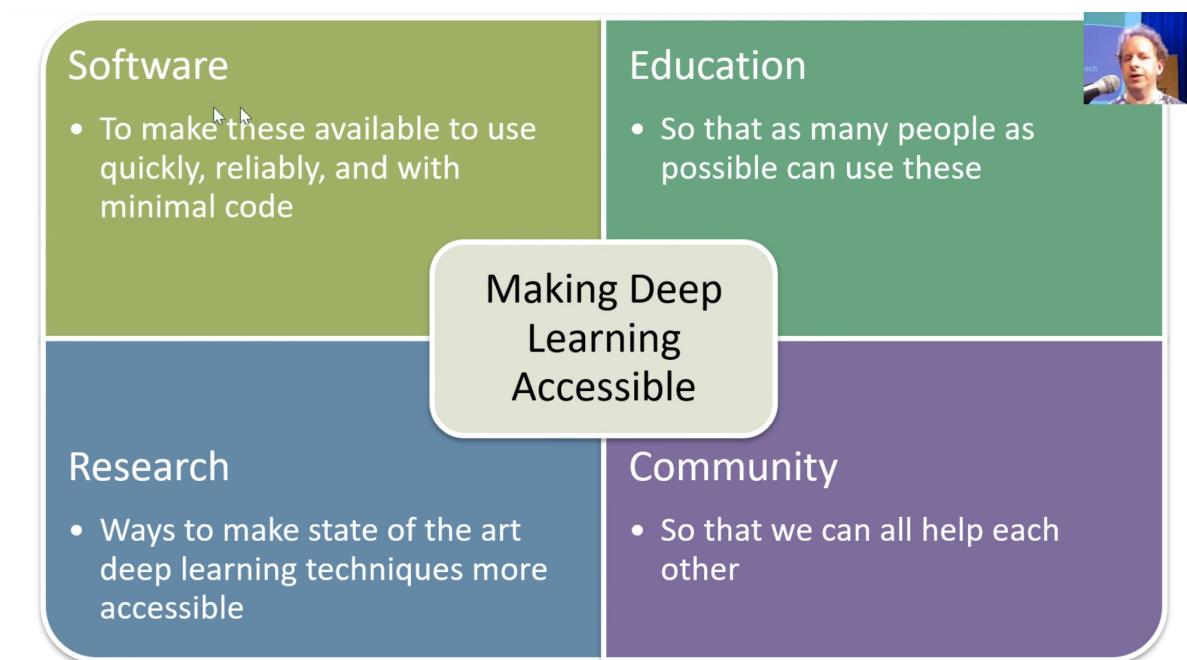
为什么听Jeremy讲课 [5:27]



我是谁，为什么要听我讲课。可能你们不应该花时间听我讲课，但我会尝试证明你们应该这样。我已经使用机器学习超过25年了。开始时，我做管理咨询，实际上我是McKinsey（麦肯锡）公司的第一位分析专家，做常规的咨询。然后创建了几家创业公司，做了很长时间。最后，成了Kaggle的总裁。但是，最让我自豪的事情是，得到Kaggle竞赛第一名。我觉得这可以证明你可以训练模型来做预测和做数据科学里其它的重要的东西。我还创建一家叫Enlitic的公司，这是第一家医疗深度学习公司。现在，我

在旧金山大学。我和Rachel是fast.ai的联合创始人。我一直在使用机器学习，尽管我现在在旧金山大学，但我觉得我不是一个学术型的老师，我更注重用这些工具来做有用的事情。

使用机器学习做有用的事情 [6:48]



通过fastai，我们在努力帮助人们用深度学习来做有用的事情。我们创建软件来让深度学习更容易在一个更高的封装层次使用。我们做教育，就像你现在正在学习的课程。我们做研究，我们花了大量的时间，来研究怎样让深度学习更容易在一个很高的封装层次使用，最终你会在软件中看到。我们通过论坛建立一个社区，大家可以找到彼此，一起合作。这就是我们在做的事。

[7:26]

What can I do after 7 lessons? Create a world class model to...



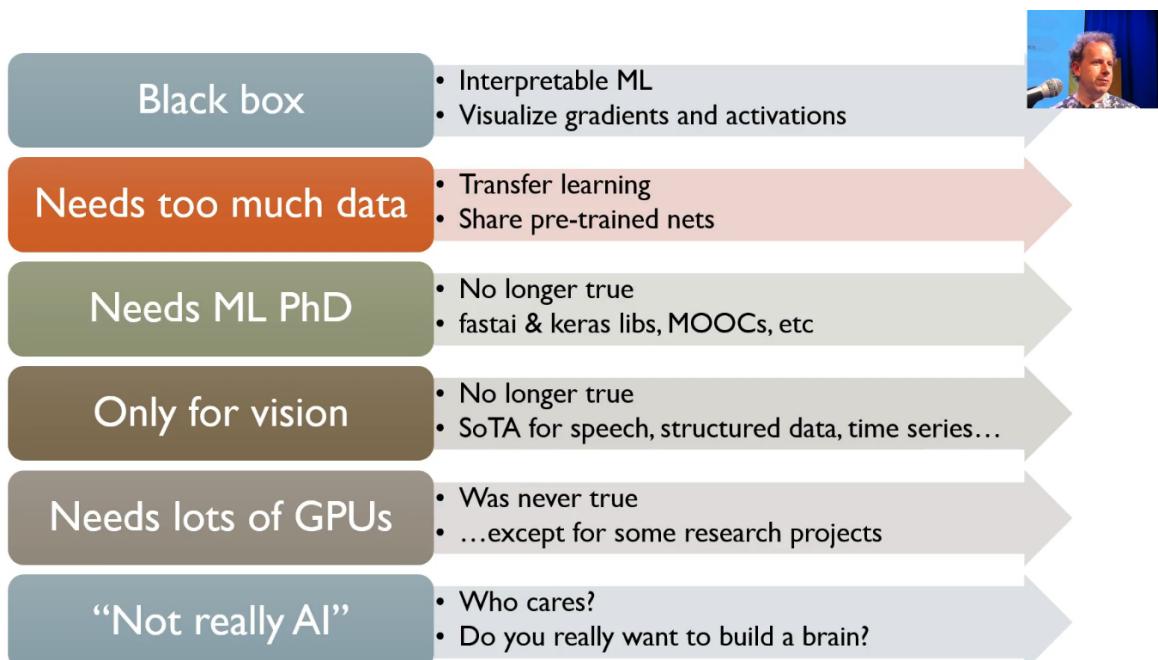
这个课程包含7节课，每节大概2个小时，我们希望你能花8到10个小时来做作业。也就是总共70到80个小时。每个人花在上面的时间差别很大，我认识有的人全职学习fastai，有的人在这两部分课程上花一年时间，有的人用双倍速看视频，没有做过作业，当然这样最后没太多收获。所以有很多种不同的方式，如果你能坚持7周，每周花10个小时在课程上，最终你可以做到：

1. 为你选择的图片构建一个世界水平的分类器
2. 对任何你感兴趣的数据集做文本分类
3. 做商业预测，比如销售预测
4. 构建像Netflix一样的推荐系统

所有这些并非玩具示例，而是可以在Kaggle竞赛赢得前十的实战项目，这样的成绩足以超过学术社区。

前提条件是一年的编程经验和高中数学知识。我们有数千学生做到了这一点。

关于深度学习的一些无意义或者不正确的说法 [9:05]



- 它不是一个黑盒子，它的工作是可以解释的
- 大部分实际的应用不需要太多数据
- 想掌握深度学习，并不需要你有一个博士学位，当然如果你有的话，也不会妨碍你从事深度学习
- 可以被广泛应用在很多不同的领域，并不单单是视觉领域
- 你不需要太多硬件，36美分一小时的服务器足够求解大部分问题，并取得世界级的结果
- 它确实不能帮助你构建一个有意识的大脑，这并不是我们关心的，我们专注于解决有意义的现实问题

[10:24]

Cricket vs. Baseball

with just 30 training images

```
In [25]: # 1. A few correct Labels at random  
plot_val_with_title(rand_by_correct(True), "Correctly classified")  
Correctly classified
```



Nikhil Balaji

棒球 VS. 板球 - Nikhil 提供的一个例子，学完第一课后，你将能够完成这个题目

由上至下的学习方式 [11:02]

A screenshot of a Jupyter Notebook interface. On the left, there is a dark gray sidebar with a white circle containing the text: "You will learn **how** it works before you learn **why** it works". The main area shows a video feed of a man speaking into a microphone. Below the video, the notebook cells are visible:

```
In [8]: learn = ConvLearner(data, tvm.resnet34, metrics=accuracy)  
In [8]: learn.fit_one_cycle(5)  
Total time: 01:22  
epoch  train loss  valid loss  accuracy  
0      1.260069  0.331716  0.909774 (00:17)  
1      0.533219  0.257838  0.917977 (00:16)  
2      0.362374  0.233951  0.926863 (00:16)  
3      0.245795  0.203109  0.937115 (00:16)  
4      0.283346  0.204246  0.939166 (00:16)  
In [9]: learn.save('stage-1')  
In [32]: interp = ClassificationInterpretation.from_learner(learn)  
interp.most_confused(min_val=3)  
Out[32]: [('Ragdoll', 'Birman', 5),  
          ('american_pit_bull_terrier', 'staffordshire_bull_terrier', 5),  
          ('Egyptian_Mau', 'Bengal', 4),  
          ('staffordshire_bull_terrier', 'american_pit_bull_terrier', 4)]
```

我们从阅读代码开始，这不同于大部分学校课程。今天我们将学习构建有用的程序。今天的课程结束后，你并不会学会所有的理论，有很多东西你可能并不清楚它们是怎样工作的，以及它们为什么是有效的。这没有关系，在接下来的7周里，你将会学会这些。现在，着手写代码而不是关注理论知识，这是更有效的学习方式。

宠物是什么品种 [12:26]

[lesson1-pets.ipynb](#)

+ 运行一个单元

每一个notebook都以这三行代码开始：

```
1 | %reload_ext autoreload
2 | %autoreload 2
3 | %matplotlib inline
```

% 开头的代码是对Jupyter Notebook的特殊指令，不是python代码，它们被称作“魔法”

- 如果程序执行时有人修改了依赖库的代码，自动重新加载库
- 如果有人要求画出一些内容，就把它们画在Jupyter Notebook上

接下来的两行引入了fastai库：

```
1 | from fastai import *
2 | from fastai.vision import *
```

什么是fastai库？ <http://docs.fast.ai/>

我们的所有工作都是基于fastai库或者PyTorch，fastai库本身也是基于PyTorch的。 PyTorch 是一个很流行的库，用户在快速增长。几年前我们使用TensorFlow，但是我们发现PyTorch可以做更多，做起来更快，所以我们改用了PyTorch。

目前fastai库支持四种应用：

1. 计算机视觉
2. 自然语言文本
3. 表格数据
4. 协同过滤

[15:45]

import * - 你们被告知永远不要这样做。

对于很多库来说，不在生产环境使用 import * 是有充分的理由的。但对MATLAB这样的库来说，是另外一回事。所有的东西都已经准备好了。你不必多次引入各种库。这是一个有趣的现象，我们走了两个极端。在科学计算界里有一种方式，在软件工程界有另外一种方式。这两种方式都有充足的理由。

对于fastai库，我们支持两种方式。当你想用Jupyter Notebook快速做些尝试时，你不希望总是回到文件开头添加一些引用。你希望有很多代码能自动完成，对于很多实验性的工作，import * 是合适的。当你构建生产环境程序时，你可以遵守PEP8这样的软件工程规范。这是一种不同的编程风格。数据科学编程并非没有规范，只不过是这里的规则不同于软件工程。在训练模型时，最重要的事情是能够快速尝试，所以你将看到很多不同的流程、风格和工具。这是有原因的，慢慢地，你将了解到这些原因。

另外，fastai库是用非常有意思的模块化方法设计的，使用import *带来的问题会比你预期的少，它被专门设计，使你能够非常方便地加载和使用里面的方法。

数据 [17:56]

课程中，我们主要从这两个地方获取数据：

1. 学术数据集

- 学术数据集是很重要的。它们很有意义。学者们花费了大量时间搜集整理一个数据集，用来测试各种不同的算法在数据集上的表现。他们设计一个数据集，然后用不同的方式挑战，寻求能够获得突破，得到更好的结果。
- 我们将从一个名为宠物数据的学术数据集开始

2. Kaggle竞赛数据集

这两种数据集都是我们关注的，它们提供了有力的基线，可以很好地评价你做得如何。使用来自竞赛的Kaggle数据集，你可以把结果提交到Kaggle，看下你在竞赛中的成绩。如果你能进入前10%，那就说明你做得相当好。

对于学术数据集，学者们在论文里写下了他们在这个数据集上取得的最好结果。这就是我们将要做的。我们创建争取能够在Kaggle竞赛里取得高排名的模型，最好是前十，而不仅是前10%，或者能达到或超过学术论文里发表的最佳结果。在你使用一个学术数据库时，把它放在参考文献里。你不必立即阅读这论文，当你希望了解到数据集的更多信息的时候，比如它为什么被创建、是如何被创建的，你可以从中了解到细节。

宠物数据集要求我们区分出37种不同的猫和狗的品种。这是一个困难的任务。事实上，在之前的课程里，我们使用另外一个数据集，它只要求分辨出图片里是狗还是猫。所以你有一半的机会猜对，并且猫和狗的区别很大。而不同品种的猫狗之间的差别并不大。

为什么我们更换了数据集？现在深度学习很简单，运行地很快。区分猫和狗这个问题对深度学习来说太简单了，尽管几年前这被认为是非常困难的，达到80%的准确率就是最好水平。我们的模型可以在不做任何优化的情况下直接输出完全正确的结果，这样我们就不能讲授一些复杂的技术，所以今年我们选择了一个更难的问题。

[20:51]

学术上，区分相似分类被叫做细粒度分类（fine grained classification）。

untar_data

我们要做的第一件事情是下载和解压数据，我们将使用 `untar_data` 这个函数，它会自动下载并解压数据。AWS给我们提供了许多空间和带宽，数据集可以被很快的下载下来。

```
1 | path = untar_data(URLs.PETS); path
```

help

怎样可以知道 `untar_data` 是什么？你可以输入 `help`，这样你可以看到这个方法来源于那个模块（因为我们使用了 `import *`，所以你不必知道所属的模块）、它会做什么、以及一些你以前不知道的事情，或许你是一个有经验的开发者，但可能并不清楚究竟应该传递什么参数。你可能很熟悉这些名字：`url`、`fname`、`dest`，但你可能没怎么见过 `Union[pathlib.Path, str]`。这是参数的类型，如果你熟悉类型编程语言的话，你可能会经常见到它们，但Python开发者可能对它不怎么熟悉。只有你知道了每一个传入的参数的类型，你才能知道怎样使用一个函数，所以我们在帮助里说明了类型信息。

对于这个函数，`url` 是一个字符串，`fname` 是一个 `path` 或者一个字符串，默认是 `None` (`Union` 表示“或者”）。`dest` 是一个 `path` 或者一个字符串

```
1 | help(untar_data)
```

```
1 | Help on function untar_data in module fastai.datasets:  
2 |  
3 |     untar_data(url:str, fname:Union[pathlib.Path, str]=None,  
4 |         dest:Union[pathlib.Path, str]=None)  
5 |             Download `url` if doesn't exist to `fname` and un-tgz to folder `dest`
```

从代码里可以看到，我们不需要传递文件名 `fname` 和目录 `dest` 参数，程序会根据 `url` 自动生成这些参数。稍后我们会学习如何获取关于这个功能的详细文档。

对于课程里的所有数据集，我们都定义了对应的常量。在这个 `URLs` 类里，你可以看到程序是怎样下载数据的。

`untar_data` 会下载数据集到一个方便使用的目录，把数据集解压，返回文件路径。

```
1 | path = untar_data(URLs.PETS); path
```

```
1 | PosixPath('/data1/jhoward/git/course-v3/nbs/dl1/data/oxford-iiit-pet')
```

在Jupyter Notebook里，你可以仅仅写一个变量（分号只是一个python语句的结尾），它就可以被打印出来。你也可以用 `print(path)`，但我们尽可能用最简便的方式，所以只写了一个变量。这里输出的就是数据的路径。

下次再执行这段代码时，因为数据已经被下载过了，它就不会再被重复下载。因为它已经被解压过了，也就不会再被重复解压。所有的功能都被设计得很简便，很自动化。

[23:50]

在Python里，有些语法不是很方便交互。比如，对一个路径对象，想查看路径下的文件，需要不少代码。所以，我们为python对象拓展了一些方法。其中一个就是为path添加了一个`ls()`方法

```
1 | path.ls()
```

```
1 | ['annotations', 'images']
```

这是目录下的文件。我们刚刚下载下来的东西。

Python 3 pathlib [24:25]

```
1 | path_anno = path/'annotations'  
2 | path_img = path/'images'
```

如果你不是一个经验丰富的Python开发者，你可能会不太熟悉斜杠的这种用法。这是Python 3里一个非常方便的函数。它是在[pathlib](#)定义的。Path对象比字符串好用得多。使用它可以这样创建子目录。无论你用的是Windows、Linux还是Mac，它的行为都是一样的。`path_img`是数据集里图片的路径。

[24:57]

如果你想用一个新的数据集做深度学习，第一件事情大概就是看看数据集里有什么。我们可以看到数据集里有 `annotations` 和 `images` 两个目录，来看下`images` 目录里有什么？

get_image_files [25:15]

`get_image_files` 可以取到一个包含所有图片路径的数组。

```
1 | fnames = get_image_files(path_img)  
2 | fnames[:5]
```

```
1 [PosixPath('/data1/jhoward/git/course-v3/nbs/d11/data/oxford-iiit-
2 pet/images/american_bulldog_146.jpg'),
3 PosixPath('/data1/jhoward/git/course-v3/nbs/d11/data/oxford-iiit-
4 pet/images/german_shorthaired_137.jpg'),
5 PosixPath('/data1/jhoward/git/course-v3/nbs/d11/data/oxford-iiit-
pet/images/japanese_chin_139.jpg'),
6 PosixPath('/data1/jhoward/git/course-v3/nbs/d11/data/oxford-iiit-
pet/images/great_pyrenees_121.jpg'),
7 PosixPath('/data1/jhoward/git/course-v3/nbs/d11/data/oxford-iiit-
pet/images/Bombay_151.jpg')]
```

把所有的文件放在一个文件夹下，这是处理计算机视觉数据集的常见方式。接下来有意思的是我们获取标签的方式。。在机器学习中，标签是指我们想要预测的东西。如果我们浏览下这些文件，可以看出标签是文件名的一部分。文件名的格式是 目录/标签_编号.拓展名。我们要想办法获取到文件名中 标签部分的列表，这样就可以得到标签。这些就是用来构建一个深度学习模型的所有东西：

- 图片文件
- 标签

在fastai里，这被设计得很简单。有一个叫做 `ImageDataBunch` 的对象，一个`ImageDataBunch` 代表你创建一个模型所需要的所有数据，使用工厂方法可以很方便得创建一个包含训练集和验证集的 `ImageDataBunch`，训练集和验证集里都包含图片和标签。

在这个案例里，我们需要从文件名提取标签。我们会使用 `from_name_re`。`re` 是python里做正则表达式的模块，正则表达对提取文本是非常有用的。这是提取标签的正则表达式：

```
1 np.random.seed(2)
2 pat = r'/(.+)_\d+.jpg$'
```

对于这个工厂方法，我们可以传入这些参数

- `path_img`: 存放图片的目录
- `fnames`: 存放文件名的列表
- `pat`: 从文件名中提取标签的正则表达式
- `ds_tfm`: 变形，我们稍后再讲
- `size`: 你想处理的图片的尺寸

图片需要按固定的尺寸来处理，这看起来有点怪。这是目前深度学习技术的一个缺点。为了处理地更快，GPU需要对同时处理的一组数据执行相同的指令。如果图片的形状尺寸不同，这就做不到了。所以我们需要让所有图片有相同的形状尺寸。在课程的第一部分，我们会把图片做成正方形。第二部分，我们会学习使用矩形。这两者有些奇特的区别。很多的人在很多计算机视觉模型里都是用的正方形的方法。224 x 224，是一个极其常见的尺寸，大部分模型都使用这个尺寸，如果你直接使用这个尺寸，大部分情况下很容易得到不错的结果。原因我们晚些再介绍。这是我想教给大家的一个小技巧，它通常很有效。你们尽管使用这个尺寸，在大部分情况下它都是有效的。

```
1 data = ImageDataBunch.from_name_re(path_img, fnames, pat,
2 ds_tfms=get_transforms(), size=224)
3 data.normalize(imagenet_stats)
```

[29:16]

`ImageDataBunch.from_name_re` 将会返回一个dataBunch对象。在fastai里，所有你用来建模的东西都是一个DataBunch对象。DataBunch对象基本上会包含2个或者3个数据集：训练数据、验证数据，有时会有测试数据。每一个都包含图片和标签，或者文本和标签，或者表格和标签，等等。它们都被放在一个地方(比如 `data`)。

一个需要多说明一点的是标准化 (normalization) 。在几乎所有机器学习任务里，你需要让数据有相同的“尺寸”，就是说有相同的平均值和标准差。所以fastai里有一个标准化函数，我们可以用这样的方式来标准化数据。

[30:25]

提问：如果图片尺寸不是224，这个函数会做些什么？

我们稍后会学习这部分内容。变形操作会对图片做一些处理，其中一个就是把图片尺寸变成224。

data.show_batch

我们来看看这些图片。这是data bunch里的一些图片。`data.show_batch`可以显示data bunch里的内容。可以看出，这些图片都被恰当地缩放或者裁剪过。这种方法叫中心裁剪，它抓取出图片的中间部分，然后调整图片尺寸。我们将会学习这个方法的更多细节，它是相当重要的。它基本上是裁剪和调整尺寸两种方法的组合。

```
1 | data.show_batch(rows=3, figsize=(7,6))
```



我们也会用它来做数据增强 (data augmentation) 。对于裁剪多少和在哪里裁剪之类的问题，是有些随机性的。

基本思路是裁剪、缩放和设置边距。做数据增强时，根据不同情况有很多不同的方式，我们稍后将学习这些内容。

[31:51]

提问: 标准化图片是什么意思?

后面我们会学习更多关于标准化图片的知识。简单来讲，一个像素有红绿蓝三个通道，每个值介于0到255之间，有的通道会太亮，有的会太暗，有的变化比较大，有的没什么变化。如果我们让三个通道的均值都是0，标准差都是1，会有利于我们训练深度学习模型。

如果数据没有标准化，难以训练出好的模型。如果你的模型效果不好，需要检查下是否标准化了数据。

[33:00]

提问: GPU内存是2的指数，相比于224来说，是不是256对于利用GPU更有效？

简单来说，模型的最后一层的尺寸是 7×7 ，所以我们希望的输入是7乘以2的指数。

[33:27]

我们将学习所有这些细节。但重要的是，我希望能尽快开始训练模型。

查看数据是重要的

一个优秀的实践者的一个重要能力就是能够查看数据。使用 `data.show_batch` 方法来看看数据是很重要的。当你查看你拿到的数据时，你会发现这些情况会很常见：图片有奇怪的黑色边框，在图片中的一些物品上有文字，有些图片被旋转过。一定要查看下这些数据。

另外一件我们要做的是查看标签。所有可能的标签都是一种分类，使用 `DataBunch`，你可以打印 `data.classes`

```
1 | print(data.classes)
2 | len(data.classes), data.c
```

```
1 | ['american_bulldog', 'german_shorthaired', 'japanese_chin', 'great_pyrenees',
  | 'Bombay', 'Bengal', 'keeshond', 'shiba_inu', 'Sphynx', 'boxer',
  | 'english_cocker_spaniel', 'american_pit_bull_terrrier', 'birman',
  | 'basset_hound', 'british_shorthair', 'leonberger', 'abyssinian',
  | 'wheaten_terrier', 'scottish_terrier', 'maine_coon', 'saint_bernard',
  | 'newfoundland', 'yorkshire_terrrier', 'persian', 'havanese', 'pug',
  | 'miniature_pinscher', 'russian_blue', 'staffordshire_bull_terrrier', 'beagle',
  | 'siamese', 'samoyed', 'chihuahua', 'egyptian_mau', 'ragdoll', 'pomeranian',
  | 'english_setter']
2 |
3 | (37, 37)
```

这是我们使用正则表达式找出的所有可能的标签。之前我们讲过有37中可能的分类，检查下 `len(data.classes)`，确实是37。DataBunch也有一个叫做 `c` 的属性。我们稍后再学习这些技术细节，现在你可以把它理解为类别的数量。对于回归和多标签分类问题，这不是很准确，但对于目前这样的解释够用了。`data.c` 是一个很重要的信息，这点要记住，它基本上表示类别的数量，至少对于分类问题是这样的。

训练 [35:07]

信不信由你，我们现在已经准备好训练模型了。在 fastai 里我们使用“learner”来训练模型。

- **DataBunch:** 一个 fastai 里广泛使用的概念，代表你的数据。对具体的应用，有对应的子类，比如 `ImageBunch`
- **Learner:** 一个 fastai 里广泛使用的概念，代表学习拟合一个模型的操作。在各种具体的应用中有很多对应的子类，用来简化使用，比如有一个 `convnet learner`，可以用来创建一个卷积神经网络。

```
1 | learn = create_cnn(data, models.resnet34, metrics=error_rate)
```

目前，创建一个卷积神经网络的learner，只需要知道两个参数：

data：你的数据，一个data bunch；

arch：模型的结构。有很多不同的方式构建一个卷积神经网络

现在，你需要知道有一种叫做ResNet的模型，它几乎总是非常有效的。你只需要选择ResNet的大小，ResNet有ResNet34和ResNet50两种大小。当我们开始做一个任务时，我会先用这个小些的模型，它训练地更快。想成为一个好的实践者，现在只需要知道有两种架构效果很好：ResNet34和ResNet50。先试下小的，看看效果是否足够好。

这是我们要创建一个卷积神经网络learner所需要知道的所有信息。

另外一个传入的参数是metrics（度量），metrics是训练时逐个地打印出来的东西。传入error_rate就是让它打印出错误率。

[37:25]

Training: resnet34

Now we will start training our model. We will use a [convolutional neural network](#) backbone and a fully connected head with a single hidden layer as a classifier. Don't know what these things mean? Not to worry, we will dive deeper in the coming lessons. For the moment you need to know that we are building a model which will take images as input and will output the predicted probability for each of the categories (in this case, it will have 37 outputs).

We will train for 5 epochs (5 cycles through all our data).

```
learn = ConvLearner(data, models.resnet34, metrics=error_rate)

Downloading: "https://download.pytorch.org/models/resnet34-333f7ec4.pth" to /home/hiromi.suenaga/.torch/models/resnet34-333f7ec4.pth
100%|██████████| 87306240/87306240 [00:09<00:00, 9590563.05it/s]
```

```
learn.fit_one_cycle(4)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

我第一次在一个新安装的环境里运行这些代码，它会下载ResNet34预训练权重。也就是说，这是一个针对特定任务训练过的特定模型。这个任务是训练模型看50万张各种物品的图片，这些物品属于1000个种类，这个图片数据集叫ImageNet。所以我们可以下载这些预训练过的参数，不必从一个一无所知的模型，而是从一个已经能够识别ImageNet里1000种类别物品的模型开始。并非所有37种品种都在ImageNet里，但里面确实有几种猫和几种狗。所以这个预训练模型知道一些猫和狗长什么样子，并且知道很多动物长什么样子，认识很多照片。所以我们是从一个空模型开始，而是基于一个已经懂得识别一些图片的模型。预训练模型会在第一次被使用时被自动下载，以后就不会再下载了，而是直接使用先前下载的那个。

迁移学习 (Transfer learning) [38:54]

这部分很重要。我们将学习很多有关迁移学习的内容。这是整个课程的重点。迁移学习研究的是怎样使用一个已经能很好地完成一些任务的模型来完成新的任务。我们使用一个预训练的模型，然后调整它，不再使用ImageNet数据来预测一千种分类，而是使用宠物数据集来预测37种品种。这样的话，相对于一次常规的训练，你训练一个模型仅需百分之一的时间和数据，甚至更少。有可能会少于千分之一。记得我在展示的Nikhil的去年第一课项目的幻灯片吗，他只用了30张图片。ImageNet里没有板球和篮球的图片，但结果ImageNet还是很擅长识别世界里的各种事物，仅仅30个打篮球和板球的例子就足够构建一个几乎完美的分类器。

过拟合 [40:05]

你怎么知道这个模型可以广泛识别出人们打板球和篮球的图片。或许它只是会识别出这30张图片，或许这只是作弊。这被称作“过拟合”。我们将在课程中讲解和多关于过拟合的内容。过拟合是指你的模型并没有学会识别图片，比如区分板球和篮球，而是仅仅能识别这几张特定图片里的板球运动员和这几张特定图片里的篮球运动员。我们必须确认我们没有过拟合。使用验证数据集可以检查有没有过拟合。验证集是你的模型没有使用过的一组图片。基于验证集的度量值（比如错误率）被自动地打印出来。当我们创建data bunch时，它会自动创建一个验证集。我们将学习很多种创建和使用验证集的方法。因为我们尝试集成所有最佳实践，你几乎无法不使用验证集。因为如果你不使用验证集，你就不知道你是否过拟合。所以我们总是打印出验证集的度量。我们总是保证模型不接触到验证集。这些都是已经实现了的，这些方法都已经被集成在data bunch对象中。

拟合模型 [41:40]

现在我们有了一个ConvLearner，我们可以开始拟合它。你可以使用一个叫 `fit` 的方法。但实践中，你应该总是使用 `fit_one_cycle` 这个方法。简单来讲，one cycle learning 是4月份发表的 [一篇论文](#)，它明显比以前的方法更快更准确。再重复一遍，我不想教大家怎样用2017年的方式做深度学习。在2018年，最好的拟合模型的方法是使用one cycle。

现在我们遍历整个数据集四次，我们把数据展示给模型看四次来训练它。每次它看到一个图片，它会变得更好些。但这会花费时间，并且这意味着会过拟合。如果它看太多次同一个图片，它只会学会识别这个图片，而不是区分宠物品种。接下来的课程里我们将学习如何调整遍历次数，现在我们选择使用4次，来看看程序是怎样运行的，你可以看到，遍历4次后，错误率是6%，这花费了1分56秒。

```
1 | learn.fit_one_cycle(4)
```

```
1 | Total time: 01:10
2 | epoch  train loss  valid loss  error_rate
3 | 1      1.175709   0.318438   0.099800   (00:18)
4 | 2      0.492309   0.229078   0.075183   (00:17)
5 | 3      0.336315   0.211106   0.067199   (00:17)
6 | 4      0.233666   0.191813   0.057219   (00:17)
```

94%的情况下，我们可以在37个猫狗的品种中选择出正确的那个，我认为这个结果很不错。要衡量这个结果好到什么程度，或许我们应该回过头来看看论文。记住，我说过使用学术或者kaggle数据集的好处是我们可以把我们的方案和Kaggle或者学术界里的最好成绩做对比。宠物品种数据集最初出现在2012年，如果浏览这些文章，你可以在论文里找到一个实验的章节。在实验章节里，你可以找到准确率的部分。他们实现了很多不同的模型，就像你在文章里读到的一样，这些模型是专门针对宠物识别的，他们学习了宠物的头长什么样，身体长什么样，宠物的图形一般长什么样。然后把这些组合在一起。他们使用这些复杂的代码和算法得到了59%的准确率。所以在2012，这项专门针对宠物的分析得到了59%的准确率。他们是牛津的顶尖研究者。现在，在2018，使用简单的三行代码，我们得到了94%的准确率（也就是6%的错误率）。可见我们使用深度学习取得了多大的进步，以及，使用PyTorch和fastai，这是多么容易做到。

[46:43]

我们仅仅训练了一个模型。我们还不是非常清楚这是如何做到的，但我们知道我们使用三四行代码，我们做到了远远超过2012年顶尖水平的准确率。对于识别不同品种的猫狗来说，6%的错误率听起来令人印象深刻。我们还不太了解它是怎样工作的，但我们将学习这些。这就够了。

过去的学生最后悔的事：

“I personally fell into the habit of watching the lectures too much and googling definitions / concepts / etc too much, without running the code. At first I thought that I should read the code quickly and then spend time researching the theory behind it...

“In retrospect, I should have spent the majority of my time on the actual code in the notebooks instead, in terms of running it and seeing that goes into it and what comes out of it”

所以请运行代码。真正地去运行代码 [47:54]

实践的最重要的技巧是学习和理解给程序输入了什么，程序输出了什么。

The screenshot shows a news article from ZDNet. The title is "Fast.ai's software could radically democratize AI". Below the title, there is a short summary: "San Francisco open source software outfit Fast.ai today unveiled the 1.0 version of its machine learning programming library, after two years in development. Built on top of the open-source PyTorch library, it makes it drop-dead easy to get started with deep learning. Creator Jeremy Howard tells ZDNet he hopes it will spread machine learning far beyond the select few practitioners who dominate the field." To the right of the summary, there is a small video thumbnail showing a man speaking into a microphone. Below the summary, there is a quote in a blue box: "'Google's Google Cloud was the first to announce support for fastai.'". Further down, another quote in a blue box says: "Amazon Web Services has announced support, and will be making it available in its 'AWS Deep Learning AMIs' and its 'Amazon SageMaker.' Microsoft has also today announced support in its Azure cloud service. Microsoft's CTO for AI, Joseph Sirosh, said that Microsoft is 'happy to see Fast.AI helping democratize deep learning at scale and leveraging the power of the cloud.'" There are also some social media sharing options and a small photo of a person.

Fastai库很新，但它得到了很多关注。它让很多事情简单了很多，也让做一些新东西变得可能。真正理解fastai程序要花不少精力。最好的方式是使用[fastai 文档](#)。

Keras[49:25]



Dogs vs. Cats

fastai v1 for PyTorch: Faster, more accurate, easier deep learning
Written: 02 Oct 2018 by Jeremy Howard



	fastai resnet34*	fastai resnet50	Keras
Lines of code (excluding imports)	5	5	31
Stage 1 error	0.70%	0.65%	2.05%
Stage 2 error	0.50%	0.50%	0.80%
Test time augmentation (TTA) error	0.30%	0.40%	N/A*
Stage 1 time	4:56	9:30	8:30
Stage 2 time	6:44	12:48	17:38

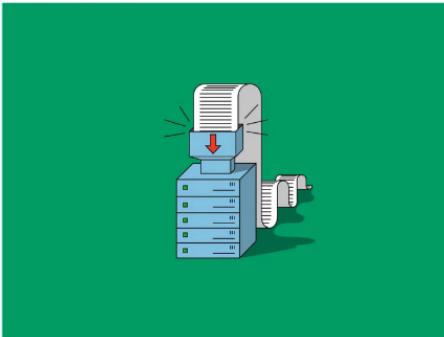
* Keras does not provide resnet 34 or TTA

fastai和其他软件比起来怎么样?唯一的和fastai类似的致力于简化深度学习的主流软件是Keras。Keras是很棒的软件，在使用fastai之前，我们在之前的课程里使用Keras。它基于Tensorflow。之前它是简化深度学习软件的典范。现在，使用fastai做深度学习更容易。如果对比去年的猫狗大战的课程练习，fastai可以得到更高的准确率(验证集上的错误率少于Keras的一半)，训练时间少于一半，代码行数是1/6。代码行数比你认为的更重要，31行Keras代码意味着你要做大量的决定，设置很多参数，做很多设置。这些是为了获得最佳的结果你必须了解的。fastai的5行代码，尽可能多地为你做了这些设置。通常，fastai为你选择了最优的默认值。你将发现这是一个非常有用的库，不仅是在学习深度学习上，也可以用它来做开发和研究。

[50:53]

GREGORY BARBER BUSINESS 09.07.16 01:00 PM

AI CAN RECOGNIZE IMAGES. BUT CAN IT UNDERSTAND THIS HEADLINE?



IN 2012, ARTIFICIAL intelligence researchers revealed a big improvement in computers' ability to recognize images by feeding a neural network millions of labeled images from a database called ImageNet. It ushered in an exciting phase for computer vision, as it became clear that a model trained using ImageNet could help tackle all sorts of image-recognition

SHARE  

...recent research from [fast.ai](#), [OpenAI](#), and the [Allen Institute for AI](#) suggests a potential breakthrough, with more robust language models that can help researchers tackle a range of unsolved problems. Sebastian Ruder, a researcher behind one of the new models, calls it his field's "ImageNet moment."

用它能做到什么？我们在fastai做的研究都是使用这个库，我们最近的一项研究就是一个例子，这项研究被Wired报道，这项突破被称为自然语言处理的ImageNet时刻，我们刷新了文本分类的记录，OpenAI基于我们的论文使用更多的机器和更多的数据来执行不同的任务，取得了很大进展。这是过去六个月我和我的同事Sebastian Ruder所做的事情中的一个例子，这是基于fastai库构建的应用的一个例

子，你将在三节课的时间里学习使用这个新的模型。你将能独自取得和论文一致的结果。

[51:50]



A screenshot of a GitHub Engineering blog post. The title is "Towards Natural Language Semantic Code Search". Below the title are two user profiles: "hamelsmu" and "hohsiangwu", followed by the date "September 18, 2018".

"The semantic code search demo is only the tip of the iceberg, as folks in sales, marketing, fraud are currently leveraging the power of fastai to bring transformative change to their business areas."

-- Github Senior ML Scientist Hamel Husain

另外一个例子，Hamel Husain，他是我们的一个学员，他构建了一个自然语言语义代码搜索系统，你可以在Github上找到这个系统，你向它输入英文语句，它可以找出你想要的代码。它也是使用fastai库构建的，你们也将在接下来的七周里学习用来实现它的技术。

[52:27]

学习这些内容和相关内容的最好的地方是我们的论坛，课程每个部分在里面都有对应的板块，也有讨论研究论文应用的深度学习通用板块。

尽管今天我们只讨论了用于图像分类的一个任务的几行代码，并没有用到太多数学和理论，但是在接下来的七周和第二部分课程里，我们会越来越深入。

这将给你怎样的未来? [53:05]



这是Sarah Hooker。她几年前参加了我们的第一次课程。参加课程前她有两年的编程经验。她创建了一个叫做Delta Analytics的非盈利组织，这个组织帮助建立了这样一个令人惊奇的系统，他们在Kanyan热带雨林的树上安装里旧手机，用来监听电锯声，他们使用深度学习来识别是不是有人在使用电锯，他们建立了一个向护林员告警，阻止非法砍伐热带雨林树木的系统。这是她在参加我们的课程做课题时做的一些事情。

Evaluating Feature Importance Estimates

Sara Hooker^{1,2} Dumitru Erhan¹ Pieter-Jan Kindermans^{1,2} Been Kim¹

Abstract

(DNN) is particularly challenging because there is typically a high number of input features and we are unable to concisely determine the representation learnt by the model. For example, the model input for a computer vision task is often a digital image and an input feature can be defined as a pixel or a group of connected pixels. Thus, a single image in the dataset can easily be associated with a quarter of a million input features. The aim of feature importance estimators is to quantify the importance of each of these input features to the model prediction for that image. The set of estimates for all pixels in the image is often treated as a sorted ranking of importance, or they are visualized as a natural image "heatmap" to assist humans in understanding parts of the image that the model pays attention to.

Despite the challenges involved, a substantial body of research on input importance estimation and numerous estimators have been proposed (Baehrens et al., 2010; Bach et al., 2015; Zintereit et al., 2017; Selvaraju et al., 2017;

Sara Hooker

Algorithms and Theory → Machine Intelligence
Machine Perception

她现在是谷歌大脑的研究员，发表了一些论文，现在她将去非洲建立谷歌大脑的第一个非洲深度学习研究中心。她在这个课程上投入了很多，不仅完成所有的作业，并且读Ian Goodfellow的书，做了很多其他事情。从她身上可以看到，一个没有计算机科学和数学背景的人也可以成为世界顶尖机器学习研究者，做非常有价值的工作。

[54:49]

Deep Learning Explorations

BLOG LEARNING ABOUT DEEP LEARNING NEURAL GALLERY EDUCATIONAL APPS CHAT

CLARA: A NEURAL NET MUSIC GENERATOR

By mcleavey Posted August 29, 2018 In Highlights, Music

OpenAI

Christine McLeavey Payne

另外一个例子是参加我们最近课程的 Christine Payne。她现在在OpenAI，你可以找到[她发表的文章](#)，听到她的自动创作室内音乐程序的音乐作品。



Vive la Musique Francaise!

BY JANOS GEREBEN, May 27, 2014

French classical music is not an endangered genre in the Bay Area: San Francisco Symphony alone has just concluded subscription programs featuring Debussy's *Images* and coming up this week, Charles Dutoit conducts Poulenc's *Gloria* and Fauré's *Requiem*.

Still, you can never have enough of a good thing, so consider the June 20-21 **First Festival of French Classical Music**, presented by Alliance Française of Silicon Valley.

To be held in the Mountain View Community School of Music and Art's Tateuchi Recital Hall, the concerts present 28 Bay Area musicians in performance of music by Maurice Ravel, Ernest Chausson, Gabriel Fauré, Andre Jolivet, Darius Milhaud, Francois Poulenc, and Charles-Valentin Alkan; also songs made famous by Edith Piaf.



"Une soirée parisienne" performers:
 Roman Fukshansky, clarinet; Moni Simeonov, violin; **Christine McLeavey Payne**, piano; and Jonah Kim, cello

DANCE JAZZ OPERA

Enter Email 

GET THE SFCV WEEKLY

FIND EVENTS

All Dates
 All Region
 All Types

Add your event here

FIND EVENTS



SF Symphony Plays Dvorak & Prokofiev OCT 11-13 THU-SAT



她是一个古典钢琴演奏家。但她并不是一个普通的古典钢琴演奏家，她有斯坦福医学硕士学位，学过神经科学，是DE Show的高性能计算专家，普林斯顿的毕业演说者。一个令人嫉妒的、做什么都很擅长的人。看到一个钢琴专家通过学习fastai课程，成为了一个OpenAI fellow是一件很酷的事。

有意思的是，我们的另外一个学员最近采访了她，他正在做有关顶级AI研究者的博客专题，在采访中Christine说她得到的最重要的建议中的一个是我建议的，这个建议是：

找到一个项目，把它做好，做得非常棒 [56:20](#)

在这个课程中，我们将讨论很多关于你做一个项目，把它做得非常棒的内容。

[\[56:36\]](#)

讲这些并不是想让你们去从事AI或者进入谷歌大脑项目。我希望你们回到你们的工作种或者你们热衷的项目里，在那里应用这些技能。

MIT发布了一份深度学习课程，在他们的公告里着重介绍了医学图像的例子。我们的一个学员Alex是一个放射科医生，他说这是一个过拟合的模型。我有资格这样说，因为我也是一个放射学者，这不是一个胸透该有的样子。这是深度学习的学习者应该做的事，这是为什么我知道你的模型运行得如何的原因。Alex结合他在放射学和深度学习的知识，仅凭两张图片就能准确地评估MIT的模型。这是我希望你们中的大多数人做的事情，像Alex那样，把你们的专业领域知识和你们将学到深度学习实践技能结合在一起。很多放射科医生已经学习了这个课程，成立了审议小组和美国放射学会（American Council of Radiology）实践小组。现在在美国放射学会有了数据科学学会，Alex是其中一个做了很多组织工作的人。我希望你们能做Alex一样的事情，引领深度学习在你们行业的应用，应用到那些有社会影响的项目里，无论是什么项目。

[58:22]

Melissa Fabros

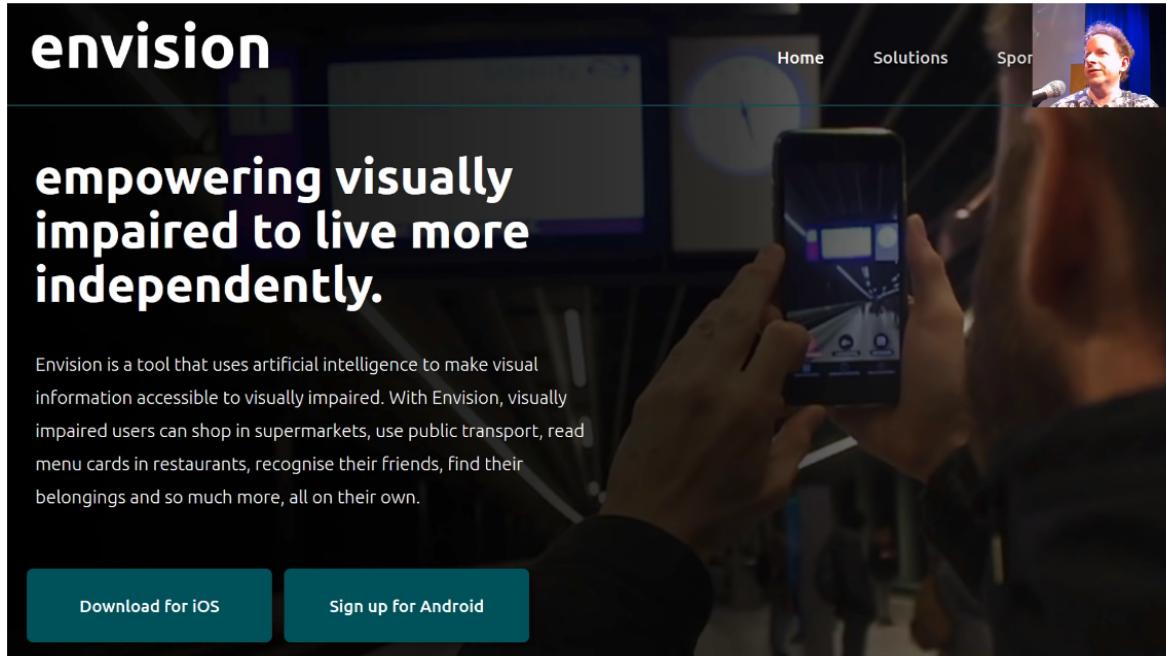
Elizabeth
Mother of 7, farmer, Kenya

CrowdFlower Names Kiva Engineer First Round Winner of \$1 Million AI For Everyone Challenge

另一个非常好的例子。这是Melissa Fabros，她曾是一个研究性别化语言和相关课题的英文文学的博士。Rachel在她以前的工作中教过Melissa编程。后来她参加了fastai课程。她帮助Kiva，一个有影响力的小额贷款机构，构建一个人脸识别系统。为什么这很必要？我们花时间介绍这个，是因为大部分AI研究者是男性白人，大部分计算机视觉程序仅能有效识别男性白人。印象中，IBM系统对男性白人的识别准确率是99.8%，对黑皮肤女性的识别准确率是65%。错误率相差三四十倍。这对Kiva来说很重要，

对这个贷款平台来说，黑人女性可能是他们最主要的客户。在Melissa学完我们的课程后，再次改变行业，投入了巨大热情在她的研究中，并且因为她为Kiva做的工作在AI竞赛中赢得了一百万美元奖金。

[59:53]



envision

empowering visually impaired to live more independently.

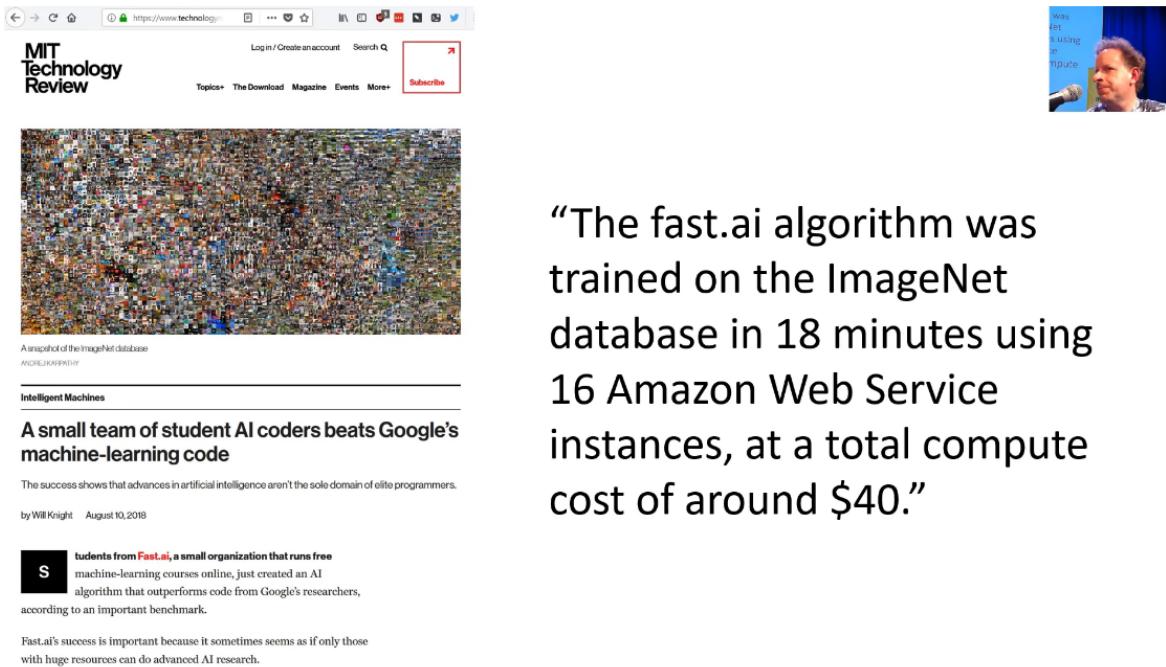
Envision is a tool that uses artificial intelligence to make visual information accessible to visually impaired. With Envision, visually impaired users can shop in supermarkets, use public transport, read menu cards in restaurants, recognise their friends, find their belongings and so much more, all on their own.

Download for iOS

Sign up for Android

参加了我们的课程后，Karthik意识到在他当时公司里的工作不是他想要的。他希望能帮助盲人了解他们周围的世界。他成立了一家叫做envision的公司。你可以下载这个app，把你的手机对准一些东西，这个app会告诉你它看到了什么。我把这种app告诉了一位盲人女士，她表示这对于失明的人是一个非常有用的工具。

[1:00:24]



MIT Technology Review

A snapshot of the ImageNet database

Intelligent Machines

A small team of student AI coders beats Google's machine-learning code

The success shows that advances in artificial intelligence aren't the sole domain of elite programmers.

by Will Knight August 10, 2018

Students from [Fast.ai](#), a small organization that runs free machine-learning courses online, just created an AI algorithm that outperforms code from Google's researchers, according to an important benchmark.

Fast.ai's success is important because it sometimes seems as if only those with huge resources can do advanced AI research.

The fast.ai algorithm was trained on the ImageNet database in 18 minutes using 16 Amazon Web Service instances, at a total compute cost of around \$40."

七周里你学习的内容和这个fastai软件将使你能够达到这个领域的领先水平，这可能会令人惊奇。我帮助了我们一些学员同事组成的团队在训练ImageNet的速度上打破了世界记录。我们使用的是标准的AWS主机，花费了40美元，我们使用了fastai库，在这个课程里我们将学习这个技术。这个课程真的可以为你提供很大帮助。不要因为最开始的内容很简单就放弃，我们会逐步深入。

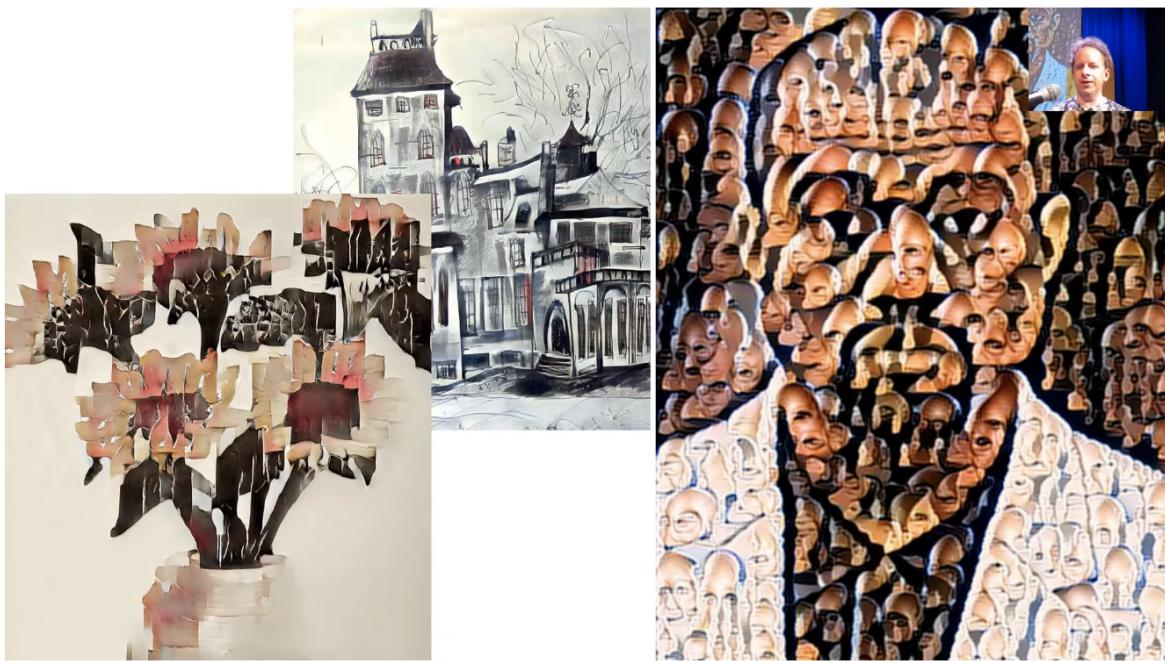
[1:01:17]



Helena S
@glagolista



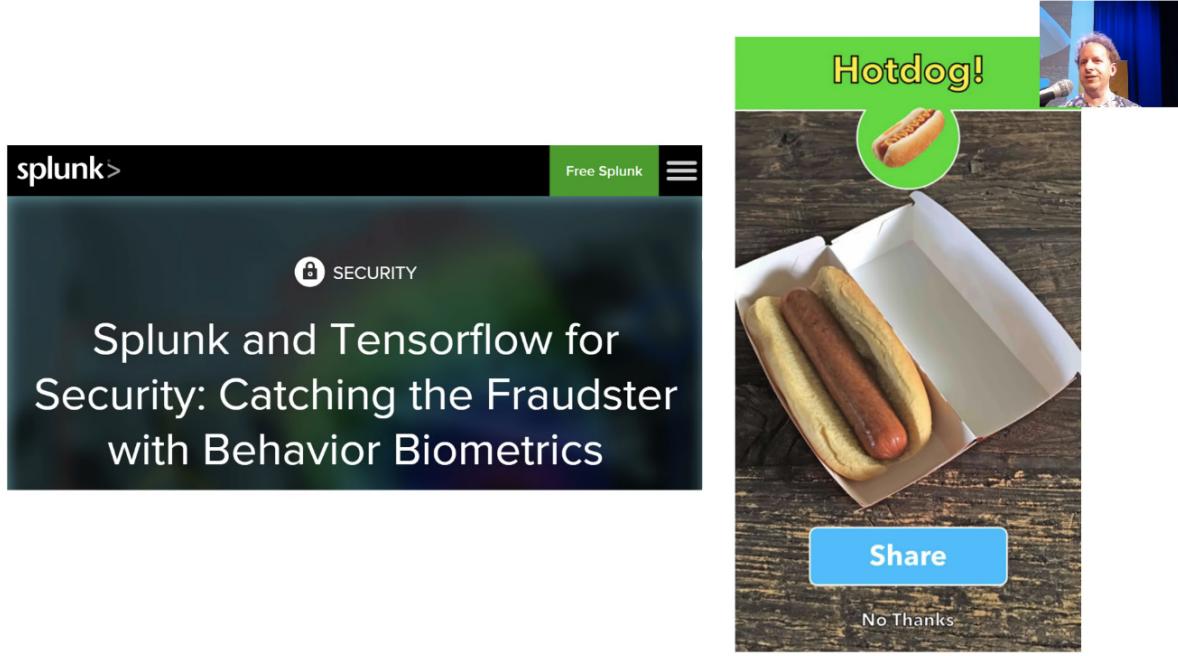
你也可以用它做一些其他感兴趣的事情。Helena Sarin，你可以查看她的Twitter[@glagolista](#)。这些作品是她用生成对抗模型把她的画组合在一起得到的令人惊奇的结果，这是一种新的艺术。我觉得这很酷。她不是专业的艺术家，她是一个专业的软件开发者，但她一直在制作这样漂亮的画。开始时，没有多少人传播讨论她的作品，现在一些高规格的文章认为她创造了一种新的艺术。



同样的，Brad Kenstler 做到了使用Patrick Stewart的头来画出一幅Kanye的画。这个技术你将会在课程里学到，它叫做“风格迁移”。他做了些微调，就画出了以前没有人尝试过的东西。这些作品帮助他得到了一份AWS深度学习专家的工作。

[1:02:41]

另外一位在Splunk担任软件工程师的学员，设计了一个算法，使得Splunk能有效地识别欺诈，我们稍后将详细介绍这个算法。



如果你看过HBO的剧集《硅谷》，里面的“是不是热狗”app是一个真实存在的app，你可以下载到它。它是Tim Anglade的课程项目。有很多你可以实现的很酷的事情。它获得了艾美奖提名。目前现在只有一个学员获得了艾美奖提名，请让这个数量变得更多。

Experiment	LM Perplexity per Word	Micro F1 on GermEval2017 task 1 timestamp 1	Micro F1 on GermEval2017 task 1 timestamp 2
ULMFIT-sp30k: WikIDEBTW17	157	0.765	0.781
ULMFIT-sp30k: BTW17	14	0.758	0.743
ULMFIT-vanilla: WikIDEBTW17	?	?	?
Naderalvojoud et al. (2017) SWN2-RNN	-	0.749	0.736
Sayyed et al. (2017) xgboost	-	0.733	0.750
TUVienKBS coarse 1	-	-	-
uhHLT fine 3	-	-	-

[1:03:30]

另外，论坛的帖子可以产生出这些很酷的东西。So Francisco 曾经和我一样是一个无聊的麦肯锡咨询顾问。我们俩都有这样可耻的经历，但我们现在还不错的。他发布了这个帖子，说我们正在学习构建不同语言的自然语言处理程序，让我们尝试大量不同语言吧。他开始做这件事情，把它叫做语言模型动物园。后来这赢得了波兰的一项学术竞赛，并发表了一篇学术论文。在泰语、德语都取得了领先水平。学生们不断刷新多种不同语言的记录，这都是学生们一起通过论坛完成的。

请去使用论坛。不要因为觉得你在论坛上看到的很多人都学习了很长时间而羞于发言。你可能因为感觉你是唯一的新手而感到害羞，实际上并不是，大家都是新手。你可以大胆地说“好的，你们都在德文语言模型上得到了优秀的结果。我启动不了服务器，我点击notebook但出现了报错，我应该怎样做？”。请在提问里提供足够的信息，人们会帮助你的。（[如何请求帮助](#)）

如果你有话想说，不要害怕发言。如果大家在讨论农作物产量分析，你恰好是一个农场主，你认为你可以发表一些意见，请大胆说出来，即使你不太确定这是不是大家关心的问题，这没有关系，尽管参与进来。记住，论坛里其它新人也都害怕发言，我们开始时都什么都不知道。尽管试着使用论坛。

[1:05:59]

提问：为什么使用ResNet而不是Inception?

有很多可供选择的模型，公平来说，没有最好的。但是如果你看看Stanford DAWN Bench这样的图片分类基准，你会发现前四名都是ResNet。ResNet足够好，所以我们使用了它。

Rank	Time to 93% Accuracy	Model	Hardware	Framework
1 Sep 2018	0:18:06	ResNet-50 <i>fast.ai/DIUX (Yaroslav Bulatov, Andrew Shaw, Jeremy Howard)</i> source	16 p3.16xlarge (AWS)	PyTorch 0.4.1
2 Sep 2018	0:18:53	Resnet 50 <i>Andrew Shaw, Yaroslav Bulatov, Jeremy Howard</i> source	64 * V100 (8 machines - AWS p3.16xlarge)	ncluster / Pytorch 0.5.0a0+0e8088d
3 Sep 2018	0:29:43	Resnet 50 <i>Andrew Shaw, Yaroslav Bulatov, Jeremy Howard</i> source	32 * V100 (4 machines - AWS p3.16xlarge)	ncluster / Pytorch 0.5.0a0+0e8088d
4 Apr 2018	0:30:43	ResNet50 <i>Google</i> source	Half of a TPUv2 Pod	TensorFlow 1.8.0-rc1
5 Apr 2018	1:06:32	AmoebaNet-D N6F256 <i>Google</i> source	1/4 of a TPUv2 Pod	TensorFlow 1.8.0-rc1

你需要另外一种架构的最可能的场景是边缘计算，比如说你需要在手机上运行一个模型。我认为大多数时候，最好的方法是在服务器运行模型，让手机应用和服务器通信获取结果。这更简单、更灵活。如果你真的需要在低性能的设备上运行模型，有一些特别的架构是用来做这个的。这个问题问的是Inception。这是另外一种架构，对内存要求更高。弹性不好。我们希望能介绍一种即使不做太多调试也能良好工作的模型。ResNet在很多情况下都能取得不错的成绩。我认为选择它更好。

[1:07:58]

我们得到了这个训练完的模型，事实上，它就是创建了一批权重。如果你做过类似线性回归和逻辑回归之类的事情，你会对系数比较熟悉。我们就是找到了一些有效的系数和参数。这花了1分56秒。如果你想尝试一些其他的事情，过一会儿再回头继续使用这个模型，我们最好保存下这些权重。你可以直接使用`learn.save`，传入一个名称。它将会把权重放在数据所在目录的模型子目录里，如果你保存了不同的数据集的不同的模型或者data bunch，它们会被分开存放，不用担心会被覆盖。

```
1 | learn.save('stage-1')
```

结果 [1:08:54]

想看看结果如何，我们可以使用这个分类任务解释类（ClassificationInterpretation）。我们使用`from_learner`这个工厂方法，我们传入一个`learn`对象。一个`learn`对象包含了两部分信息：

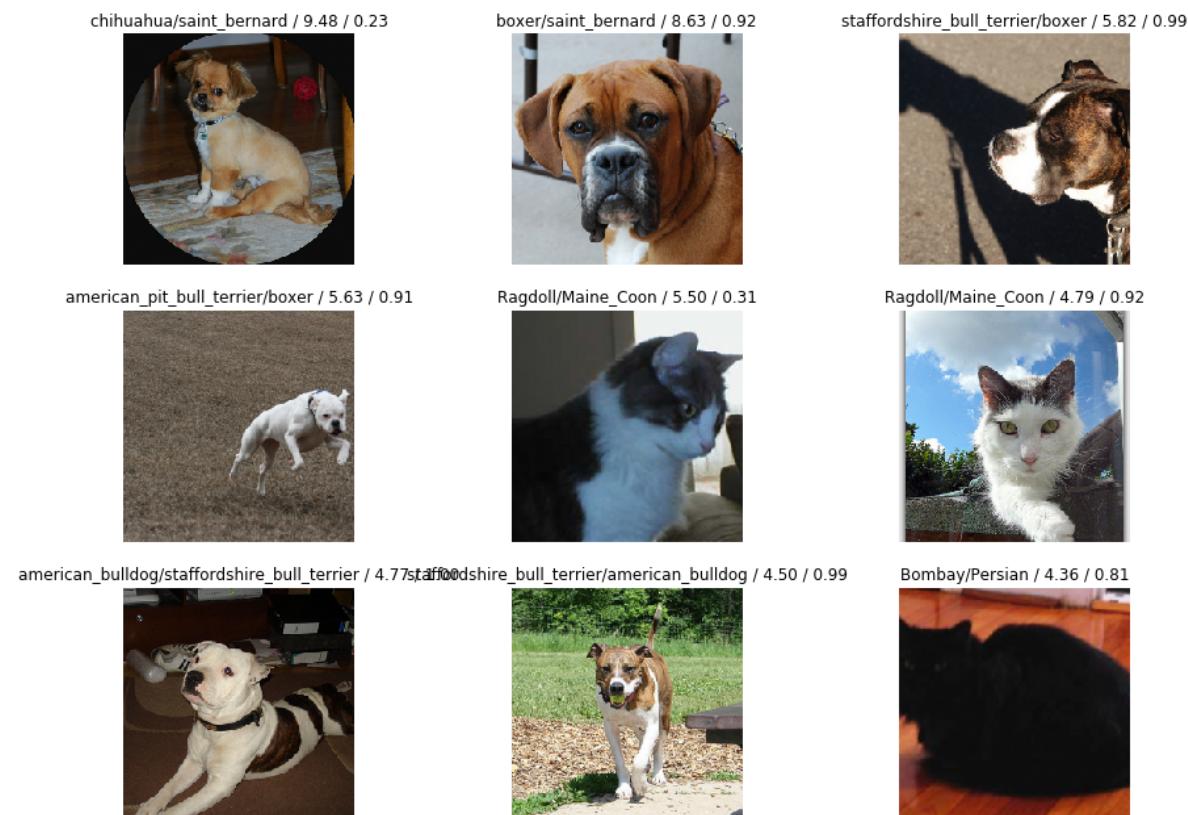
1. 数据是什么
2. 模型是什么。现在它不仅是一个结构，它已经是一个训练完成的模型

这是用来解释这个模型所需要的所有信息。

```
1 | interp = ClassificationInterpretation.from_learner(learn)
```

我们调用一下`plot_top_losses`, 这大概是一个最有用的方法。我们稍后将学习很多关于损失函数这个概念的内容, 简单来讲, 损失函数是用来评价你的预测的效果。比如说, 你为一个猫的图片预测了一个类别, 并且非常确信, 但事实上你又预测错了, 这时就会有一个比较高的损失, 因为你对错误答案很确信。这就是损失值比较高的含义。通过画出最高的损失值, 我们就可以看到我们最严重的错误是什么, 或者对哪个错误答案最确信。

```
1 | interp.plot_top_losses(9, figsize=(15,11))  
2 |
```



它打印出了四项内容。它们代表什么。或许我们需要看下文档。

我们已经介绍过 `help` 方法, `help` 仅仅是打印出一个简短的概述。如果你想了解怎样完成一个任务, 你需要使用 `doc` 方法。

```
In [19]: doc(interp.plot_top_losses)
```

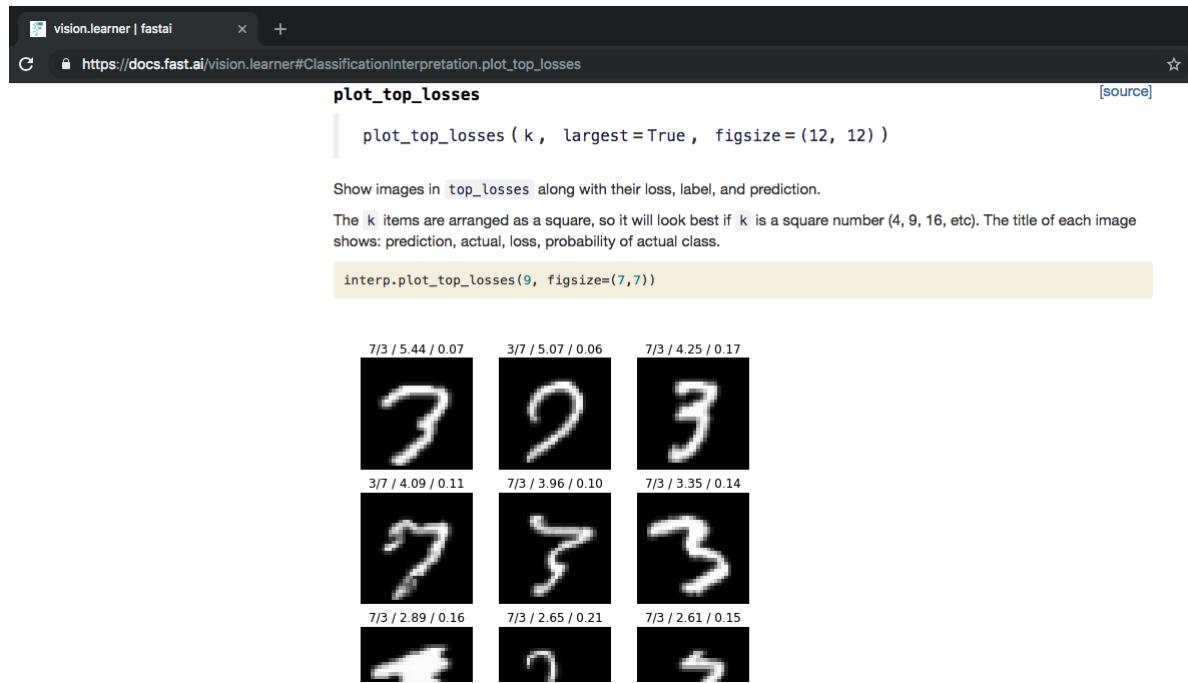
```
In [20]: interp.plot_confusion_matrix(figsize=(12,12), dpi=60)
```

plot top losses

```
plot_top_losses(k, largest=True, figsize=(12, 12))
```

Show images in `top_losses` along with their loss, label, and prediction. [source]
[Show in docs](#)

`doc` 提供了和 `help` 一样的信息，但是它包含了一项很重要的内容，那就是 `show in docs`。如果你点击它，它会弹出这些方法/类的文档。



开始显示的信息和使用`doc`方法看到的是一样的，都是这个方法需要的参数。后面提供了更多的信息：

每个图片的标题显示了：预测值、实际值、损失度、实际值的概率

文档里包含可执行的代码。当你想搞明白怎样使用这个时，这是很有用的。另外一个我想说的是，如果你是一个有经验的Python开发者，你会发现我们的代码可读性很好，我们尽量使代码比较短，一般少于半屏。如果你点击 `[source]`，你可以直接跳转到源代码。

```
82     def plot_top_losses(self, k, largest=True, figsize=(12,12)):
83         "Show images in 'top_losses' along with their prediction, actual, loss, and probability of actual class."
84         tl_val,tl_idx = self.top_losses(k,largest)
85         classes = self.data.classes
86         rows = math.ceil(math.sqrt(k))
87         fig,axes = plt.subplots(rows,rows,figsize=figsize)
88         for i,idx in enumerate(tl_idx):
89             t=self.data.valid_ds[idx]
90             t[0].show(ax=axes.flat[i], title=
91                         f'{classes[self.pred_class[idx]]}/{classes[t[1]]} / {self.losses[idx]:.2f} / {self.probs[idx][t[1]]:.2f}'")
```

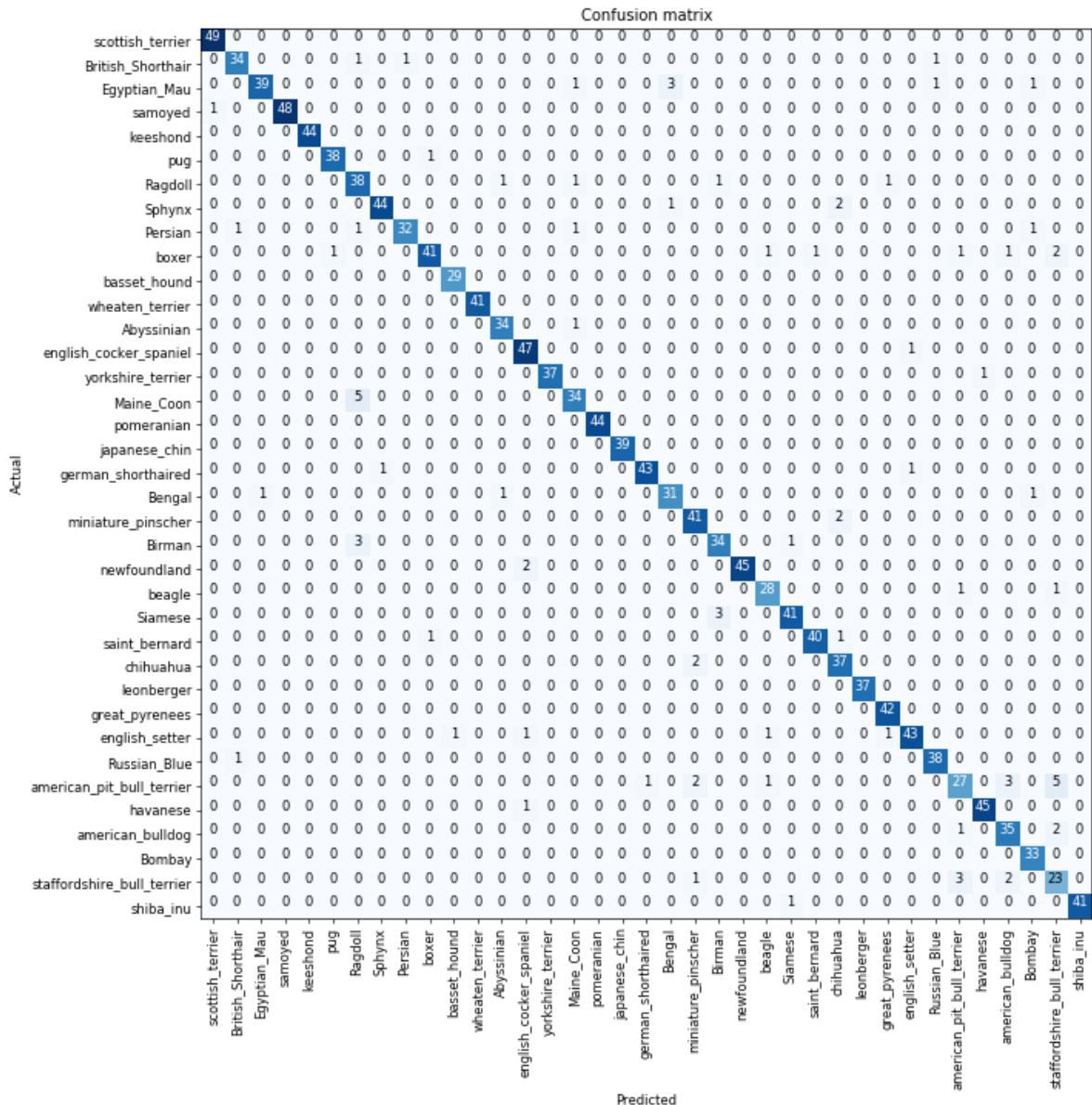
这就是`plot_top_loss`方法的代码，这也是学习如何使用fastai库的好方法。因为几乎每一行代码，都调用了fastai库里的东西。不要不敢读源代码。

[1:12:48]

这是我们查看损失度最大的数据的方法，这大概是最重要的图片分类解释工具，它可以让我们看到我们做错了什么。在这个例子里，如果你是一个宠物专家，你可能会发现出错的是一些非常难分辨的品种，这时你可以说“哦，我知道为什么它们会被认错”。所以这是一个很有用的工具。

混淆矩阵 1:13:21

另外一个有用的工具是混淆矩阵，它显示了每个品种被预测成各个品种的次数。遗憾的是，在这个例子中，因为模型预测得很准确，对角线显示了结果很好。



你可以看到里面颜色比较深的几个，想看清它们是哪两个类别的组合确实不太容易。所以我建议在类别比较多时，不要使用混淆矩阵。你可以调用“most_confused”。这是我觉得fastai里命名最好的一个方法，我对此很自豪。

Most_confused [1:13:52]

```
1 | interp.most_confused(min_val=2)
```

```
1 | [('american_pit_bull_terminer', 'staffordshire_bull_terminer', 5),
2 | ('Birman', 'Ragdoll', 5),
3 | ('english_setter', 'english_cocker_spaniel', 4),
4 | ('staffordshire_bull_terminer', 'american_pit_bull_terminer', 4),
5 | ('boxer', 'american_bulldog', 4),
6 | ('Ragdoll', 'Birman', 3),
7 | ('miniature_pinscher', 'chihuahua', 3),
8 | ('Siamese', 'Birman', 3)]
```

`most_confused` 会抓出混淆矩阵里被预测错次数最多的类别组合。这个例子里 `('american_pit_bull_terminer', 'staffordshire_bull_terminer', 7)` 表示：

- 真实品种 `american_pit_bull_terminer`
- 预测值 `staffordshire_bull_terminer`

- 这个组合出现了7次

这是一个很有用的方法，你看看这些结果，根据你的领域经验，看看得出这样的结果是不是合理的。

解冻，微调，学习率 [1:14:38]

让我们来优化下我们的模型。怎样才能做到呢？我们可以使用微调（fine-tuning）。我们执行了四个 epochs，它执行得很快。它执行得快是因为我们使用了一个技巧。卷积网络包含很多层。我们晚些将详细介绍什么是层，现在只需要知道它代表进行了大量计算。我们在那些我们已经训练好的模型后面加入了一些额外的层。我们保持原有的模型不变，这样就能训练得很快。如果你想构建的模型和已有的模型相似，这个微调的方法很好用。在这个例子里，数据和 ImageNet 是相似的。

但我们真正想做的是回过头来训练整个模型。这是为什么我们经常使用这个两阶段过程。首先，我们调用 ConvLearner 的 `fit` 或者 `fit_one_cycle` 方法，它只会微调我们加在末尾的一些层，这一步训练得很快，也不会过拟合。为了得到更好地模型，你需要调用 `unfreeze`。`unfreeze` 让它训练整个模型，然后我们再调用一遍 `fit_one_cycle`。

```
1 | learn.unfreeze()
2 | learn.fit_one_cycle(1)
```

```
1 |
2 | Total time: 00:20
3 | epoch  train_loss  valid_loss  error_rate
4 | 1      1.045145    0.505527    0.159681    (00:20)
```

噢。错误率变差了。为什么呢。为了理解为什么，我们要学习更多底层的原理。我们先从获得一个直观的理解开始，看下这个图片。

[1:16:28]

Visualizing and Understanding Convolutional Networks

Matthew D. Zeiler

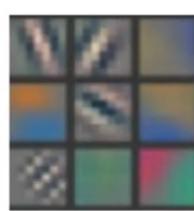
ZEILER@CS.NYU.EDU

Dept. of Computer Science, Courant Institute, New York University

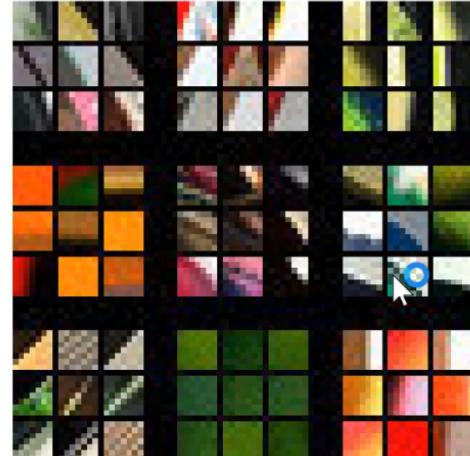
Rob Fergus

FERGUS@CS.NYU.EDU

Dept. of Computer Science, Courant Institute, New York University

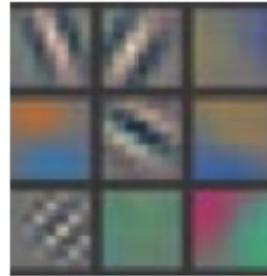


Layer 1

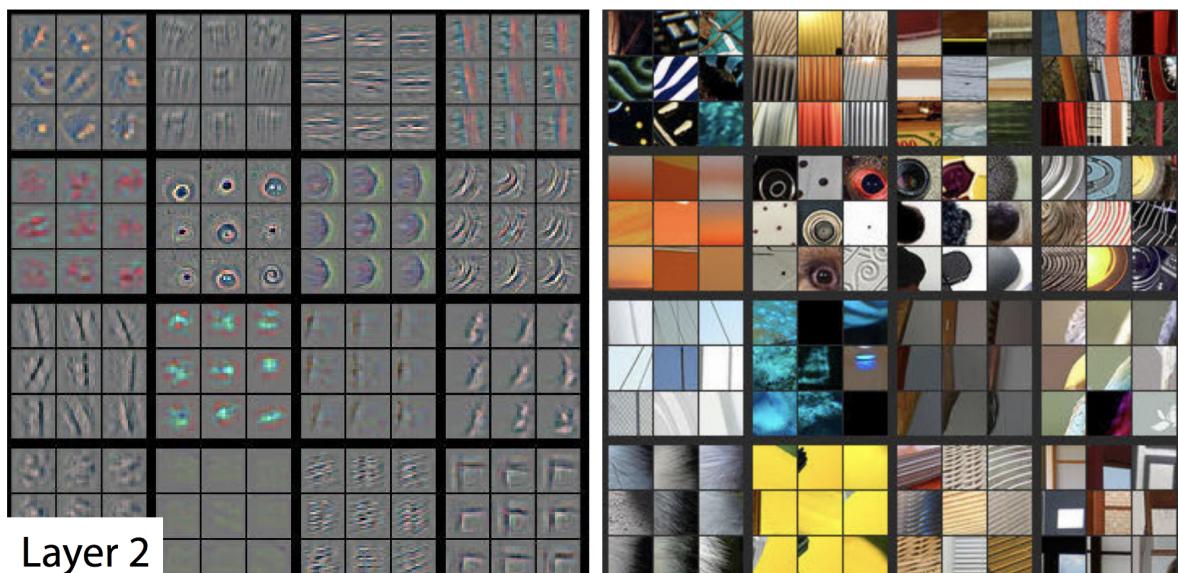


这些图片来自于Matt Zeiler和他的博士论文导师Rob Fergus的一篇了不起的论文，Matt Zeiler现在是Clarify的CEO，Clarify是一家很成功的计算机视觉创业公司。他们写了一篇论文来展示如何可视化卷积神经网络的层。稍后我们将在数学层面上介绍什么是卷积神经网络的层，现在我们可以这样理解，表示像素颜色的红绿蓝的值是0到255的数字，对它们做一些计算（第一层），输出了一些结果，把这些结果输入到第二层，再把第二层的结果输入到第三层，然后第四层。可以多达上千层。ResNet34有34层layers，ResNet50有50层，我们看下第一层。如果你了解卷积的话，可以知道它是一个非常简单的运算。第一层会输出什么，我们可以把这些系数和参数画出来，在第一层里有很多，我们没有画出全部，我们随机选择9个。

[1:17:45]

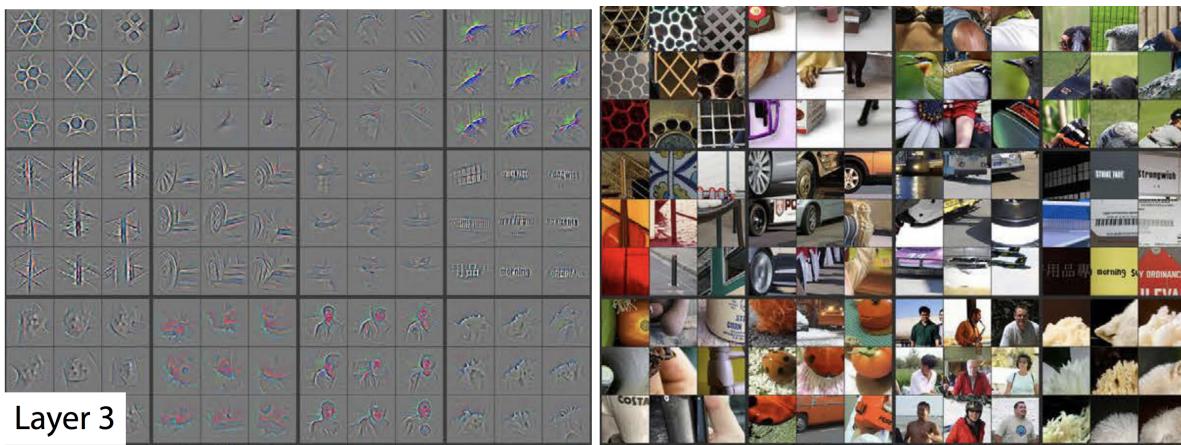


这是第一层的真实系数的9个例子。它们检查相互邻近的一组像素。第一个认出了有对角线的一组像素，第二个认出了一个反方向的对角线，第三个认出了从黄色到蓝色的渐变，等等。这是些很小很简单过滤器（filters）。这是ImageNet预训练卷积神经网络的第一层。

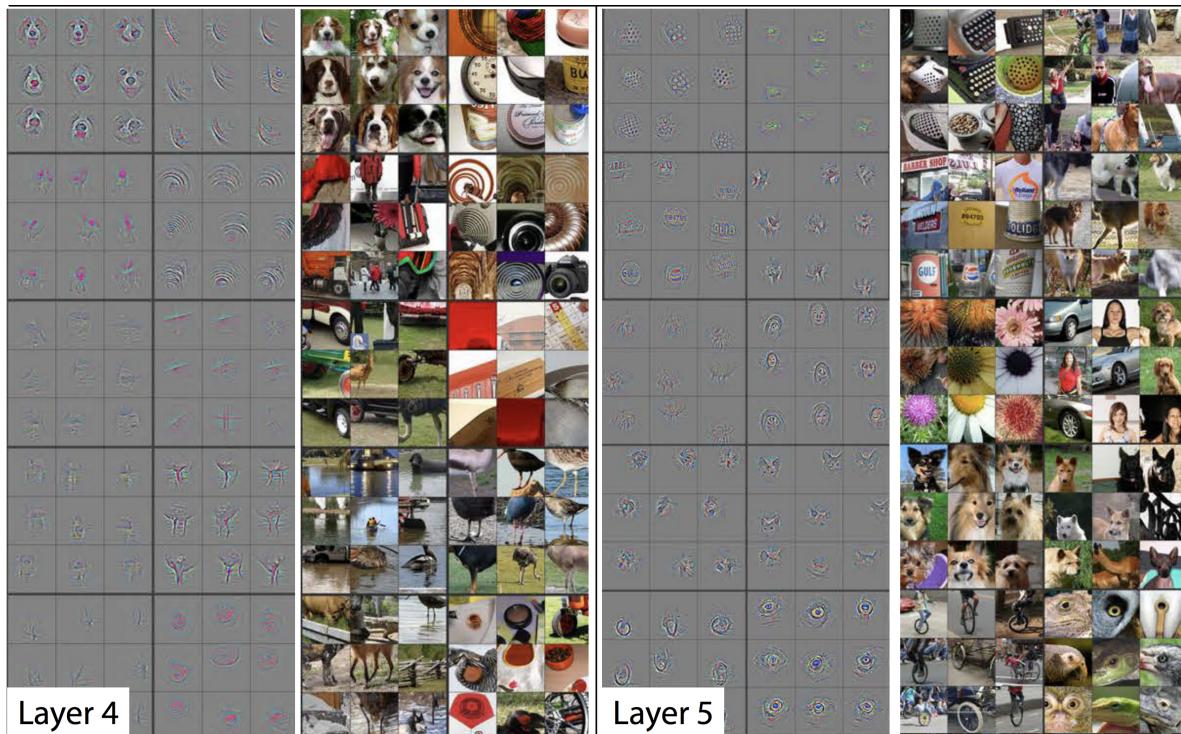


第二层使用这些过滤器的结果来做运算。右下角是一个能看出第二层特征的例子。就像你能看到的那样，它学会创建了一些能找出类似位于左上方的角的东西。有些学会了找出右侧的曲线或者小圆圈等等。在第一层里，它们只能认出到一条线。在第二层里，它们可以认出由两条线构成的图形或者重复的几条线。看下右边，这九个是可以激活这些过滤器的真实照片。换句话说，右下角的过滤器能够找出类似窗户角之类的东西了。

这能让你有很直观的理解。第一层发现了一些非常简单的斜线和直线，第二层找到了很简单的形状，第三层可以找到它们的组合。



现在我们可以找到重复的二维图形，可以找到连接在一起的物品，或者文字/窗户，所以它看起来能找到重复的水平的图形。还有些能找到毛绒的或者花朵一样的东西的形状或者地理图形。所以第三层能识别所有第二层识别的东西和第二层里东西的组合。



第四层能识别所有第三层识别的东西和它们的组合。在第四层里，它们可以认出狗脸或者鸟腿。

第五层，它们可以认出鸟眼和蜥蜴，或者一些品种的狗的脸。这样你可以看到一步步的到第34层，可以认出特定的猫狗的品种。这就是它的工作原理。

当我们第一次训练完（微调）一个预训练模型后，我们保留了你刚刚看到的这些所有的层，我们只是在已有的众多复杂特征的基础上再训练了额外的几层。现在我们回过头来，改变所有的东西。我们将以现有的做基础，但我们要看看能不能做得更好。

想让第一层的能力更好看起来不太容易。区分猫狗品种时和识别ImageNet数据时对对角线的定义没有什么不同。我们不怎么需要改变第一层。最后的几层是识别狗脸的之类的东西的，是我们最需要改变的。你需要有这样的直观感觉：神经网络的层代表了不同的语义复杂度层级。

[1:22:06]

这就是我们对这个模型的微调没有作用的原因，默认情况下，它以相同的速度训练所有层，就是说它用相同的方式更新代表对角线和渐变的参数和代表眼睛样子这种具体细节的参数。我们要改变这个情况。

改变它之前，我们先把它恢复成原来的样子。我们刚刚破坏了这个模型，它比开始时更差了。运行这个代码

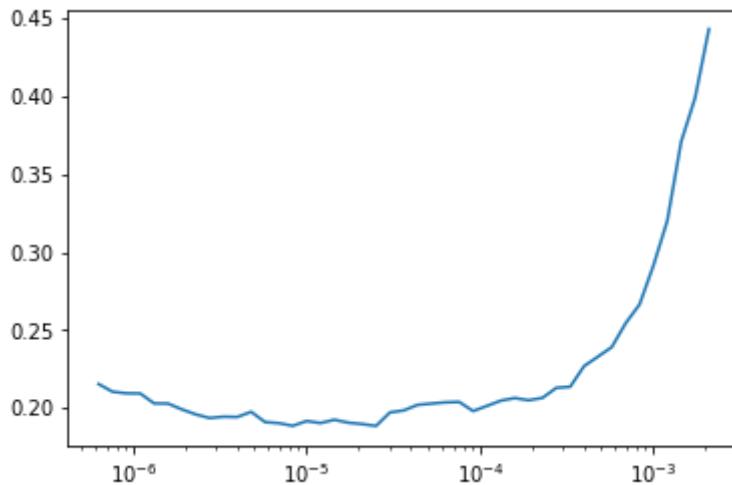
```
1 | learn.load('stage-1')
```

这让模型变成我们之前保存的那个。我们加载回它，现在我们使用的模型是被我们破坏之前的那个了。

学习率探测器（Learning rate finder） [1:22:58]

我们运行下learning rate finder。我们会在下周深入学习它，现在只需要知道它能帮助我们用最快的方式训练模型。

```
1 | learn.lr_find()  
2 | learn.recorder.plot()
```



这会画出learning rate finder的结果，这里展示的是最重要的参数——学习率。学习率代表更新模型参数的速度。x轴代表学习率，学习率逐渐增加。y轴代表损失度。你可以看到，学习率超过 10^{-4} 后，损失度变差了。这和实际发生的一样。我们可以验证它，按下 `shift + tab`，学习率默认是0.003，你可以看到损失度变差了。我们想做些微调，不能使用这么高的学习率。按照学习率探测器的结果，我试着选择 `1e-6`。但是不能用这个来训练所有的层，后面的几层在我们加快速度前也运行得很好。所以我们传入一个学习率的区间到 `learn.fit_one_cycle`。我们这样做：

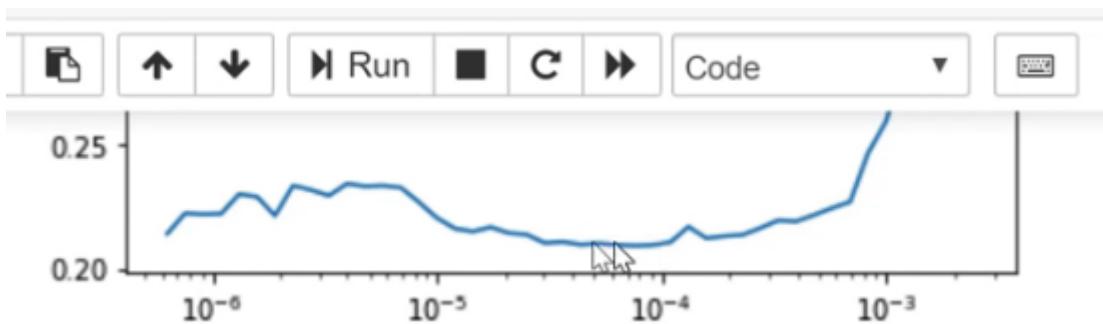
```
1 | learn.unfreeze()  
2 | learn.fit_one_cycle(2, max_lr=slice(1e-6,1e-4))
```

```
1 | Total time: 00:41  
2 | epoch  train_loss  valid_loss  error_rate  
3 | 1      0.226494    0.173675    0.057219    (00:20)  
4 | 2      0.197376    0.170252    0.053227    (00:20)
```

使用Python里的 `slice` 关键字，它接收一个起始值和结束值。这样会按照 $1e-6$ 的学习率训练第一层，按照 $1e-4$ 的学习率训练最后一层，按照层数等分，每层使用对应的学习率。

解冻后怎样选择学习率 [1:25:23]

一个简单的方法是在解冻后（也就是训练所有层时），传入一个 `max_lr` 参数，给它传入一个 `slice`，`slice`的第二个参数的值用第一阶段学习率的十分之一。第一阶段用的是 $1e-3$ ，所以现在使用 $1e-4$ 。`slice`的第一个参数是从learning rate finder得到，用结果变差之前的值。你可以看到学习率是在这个位置开始变差的：



我选择至少比它小10倍的值。

这样一来，错误率变好了些。我基本可以说对大部分情况，这两步足够得到一个世界级的模型。你并不会赢得Kaggle竞赛，某种程度上是因为现在很多fastai的学员在参加Kaggle比赛，这是他们做的第一件事。但是在真实的工作任务中，你会得到比大多数人更好的结果。

ResNet50 [1:26:55]

我们可以通过使用更多层来提升模型。我们下周会学习这个。其实就是使用ResNet50来替代ResNet34。你也可以这周就运行这部分代码。你可以看到结果和之前的结果一样，并没有什么进步。

```

1
2 data = ImageDataBunch.from_name_re(path_img, fnames, pat,
3   ds_tfms=get_transforms(), size=320, bs=bs//2)
4 data.normalize(imagenet_stats)
```

```

1
2 learn = ConvLearner(data, models.resnet50, metrics=error_rate)
```

如果你尝试执行这个，你有可能会得到一个GPU内存溢出的报错。因为ResNet50比ResNet34更大，它包含更多参数，需要占用更多内存。不同于通常的电脑内存，它使用的是GPU内存。如果你使用默认的Salamander、AWS，你有16G的GPU内存。通常我用的显卡有11G内存，便宜些的显卡有8G。这是你可能会用到的显卡的内存情况。如果你的内存小于8G，它会让你失望。

如果你运行这个代码，非常有可能遇到内存溢出问题，因为程序试图做很多事——更新很多参数，数量超出了你的GPU内存的能力。这很容易解决。`ImageDataBunch` 构造函数的最后一个参数是 `bs`，代表batch size。就是一次训练多少图片。如果你遇到内存溢出，就把它设小些。

使用较小的batch size会让训练花的时间更久些，仅此而已。这是你这周需要尝试的一个事情。

```
1 | learn.fit_one_cycle(8, max_lr=slice(1e-3))
```

```

1 Total time: 07:08
2 epoch  train_loss  valid_loss  error_rate
3 1      0.926640    0.320040    0.076555    (00:52)
4 2      0.394781    0.205191    0.063568    (00:52)
5 3      0.307754    0.203281    0.069036    (00:53)
6 4      0.244182    0.160488    0.054682    (00:53)
7 5      0.185785    0.153520    0.049214    (00:53)
8 6      0.157732    0.149660    0.047163    (00:53)
9 7      0.107212    0.136898    0.043062    (00:53)
10 8     0.097324    0.136638    0.042379    (00:54)
```

训练一会儿，错误率降到了4.2%，这令人惊奇。我们在第一次课程里，对于猫狗大战的问题，我们得到了大概3%的错误率，这个问题是有50%的机会猜对的，猫和狗完全不同，很容易区分。所以在区分品种这样的各个图片仅有细微差别的问题上能取得4.2%的错误率是一个非常了不起的成绩。

再次解释结果 [1:29:41](#)

```
1 | interp = ClassificationInterpretation.from_learner(learn)
2 | interp.most_confused(min_val=2)
```

```
1 | [('Ragdoll', 'Birman', 7),
2 | ('american_pit_bull_terrier', 'staffordshire_bull_terrier', 6),
3 | ('Egyptian_Mau', 'Bengal', 6),
4 | ('Maine_Coon', 'Bengal', 3),
5 | ('staffordshire_bull_terrier', 'american_pit_bull_terrier', 3)]
```

你可以调用most_confused方法，看下分类错的品种。你得到的结果里的每个品种的数量可能不同，但是这些品种应该是一致的。我经常看到Ragdoll和Birman经常被搞混。我以前也没有听说过这两种猫。所以我查了一下，结果在theCatSite.com上找到了一个题目是“这是一只Birman还是Ragdoll？”的帖子，里面有很多动物专家在激烈争论这到底是什么品种。

theCatSite HOME FORUMS ARTICLES REVIEWS GALLERY MEMBERS Q. SEARCH

Is this a Birman or Ragdoll kitten?

Discussion in 'Describing Cats - What Does My Cat Look Like?' started by esteevius, Jan 14, 2015.

Jan 14, 2015 #1

 esteevius Thread Starter TCS Member Kitten

8 1 Jan 14, 2015 New Mexico

I adopted my boy, Etson at the animal shelter on November 9th, 2014. He eats more than my other kitten Gando and I sometimes have to stop him. At first I thought he was Siamese because I have never really researched cat types.. but someone was visiting and said he was definitely not Siamese, but looked like a Ragdoll. I decided to look into cat types.

The two types I believe he resembles the most are the Ragdoll and the Birman. The animal shelter only had him listed as a Domestic shorthair, but his fur is quite longer and really soft. He also has white back legs, which Birman tend to not have. The nose however looks more Birman to me, but I'm really having a hard time deciding. He's grown quite a bit since I got him, considering he's only almost 3 months old. I'm just wondering if I should start looking for a bigger litter box early.. haha.

(The shadowing on his youngest and first picture looks like he has coloring on his rear legs, but he does not.)



pitbull和staffordshire bull terrier这两种狗也很相似，大概他们之间的区别是英国犬协会指导手册里的定义不同。人能区分出的一个区别大概是它们中的一个可能有个红鼻子。这些是领域专家才知道的事情，处理这些问题会帮助你成为一个领域专家。现在我比以前更加了解哪些品种更难区分。解释模型要从这两方面进行。

作业 [\[1:30:58\]](#)

这周我希望你做的是运行这个notebook，把它运行一遍。另外我希望你能创建你自己的图像数据集，Francisco已经整理了一份使用Google Images下载数据的指南，你可以创建自己的数据集来运行下。最后，我想演示下怎样用几种不同的方式创建标签，因为你获取数据集的地方可能没有提供基于正则的命名方式，可能会有五花八门的格式。所以讲一下如何处理。我使用MNIST做例子。MNIST是一个手写数字数据集。我希望演示下创建数据集的不同方式。

```
1 | path = untar_data(URLs.MNIST_SAMPLE); path
```

```
1 | path.ls()
```

```
1 | ['train', 'valid', 'labels.csv', 'models']
```

这里已经有了一个训练集和验证集。创建数据集的人已经决定好了你应该用什么做验证集。

场景1：标签是文件夹名

```
1 | (path/'train').ls()
```

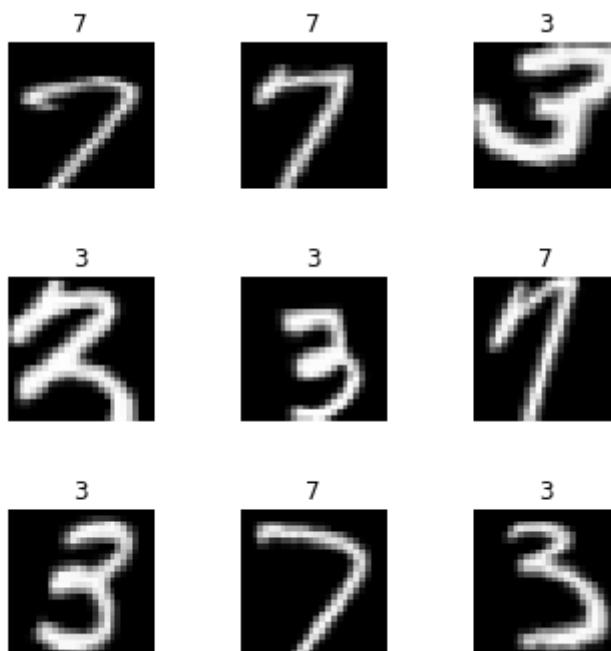
```
1 | ['3', '7']
```

这里有一个叫做“3”和“7”的文件夹。这是一种常用的提供标签的方式。所有3的图片都被放在这个叫做“3”的文件夹里。所有7的图片都被放在这个叫做“7”的文件夹里。这被叫做“ImageNet风格数据集”，ImageNet的数据就是这样存放的。如果你的数据是这种标签就是文件名的话，你可以使用 `from_folder` 方法。

```
1 | tfms = get_transforms(do_flip=False)
2 | data = ImageDataBunch.from_folder(path, ds_tfms=tfms, size=26)
```

这会为你创建一个`ImageDataBunch`，你可以看到它为图片创建了标签：

```
1 | data.show_batch(rows=3, figsize=(5,5))
```



场景2: CSV 文件 [1:33:17]

另外一种可能的情况是，标签存放在一个CSV文件里。在这个MNIST例子里也有，文件内容是这样的。

```
1 df = pd.read_csv(path/'labels.csv')
2 df.head()
```

	name	label
0	train/3/7463.png	0
1	train/3/21102.png	0
2	train/3/31559.png	0
3	train/3/46882.png	0
4	train/3/26209.png	0

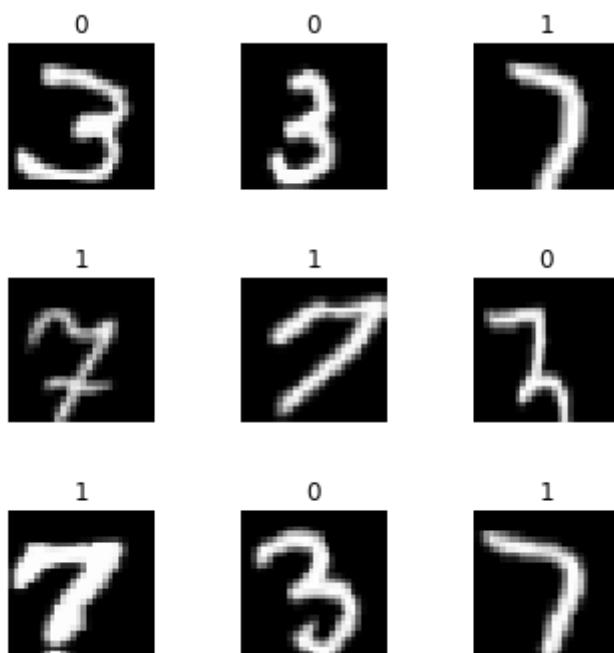
每个文件名有一个对应的标签。这里标签内容不是3或者7，而是0或1，它们代表文件是不是7。这是另外一种可能的情况。这时你可以使用 `from_csv`：

```
1 data = ImageDataBunch.from_csv(path, ds_tfms=tfms, size=28)
```

如果文件名是 `labels.csv`，你可以不用传入文件名。

```
1 data.show_batch(rows=3, figsize=(5,5))
2 data.classes
```

```
1 [0, 1]
```



场景3: 使用正则表达式

```
1 | fn_paths = [path/name for name in df['name']]; fn_paths[:2]  
  
1 | [PosixPath('/home/jhoward/.fastai/data/mnist_sample/train/3/7463.png'),  
2 | PosixPath('/home/jhoward/.fastai/data/mnist_sample/train/3/21102.png')]
```

标签名还是文件夹名。但是我可以使用正则表达式来提取标签。这种方式我们已经见过了：

```
1 | pat = r"/(\d)/\d+\.\png$"  
2 | data = ImageDataBunch.from_name_re(path, fn_paths, pat=pat, ds_tfms=tfms,  
3 | size=24)  
4 | data.classes  
  
1 | ['3', '7']
```

场景4: 更复杂的情况 [1:34:21]

你可以创建一个可以从文件名或者路径提取标签的函数。这时，你要调用 `from_name_func`：

```
1 | data = ImageDataBunch.from_name_func(path, fn_paths, ds_tfms=tfms, size=24,  
2 |         label_func = lambda x: '3' if '/3/' in str(x) else '7')  
3 | data.classes
```

场景5：你需要更灵活的方法

如果你需要更灵活的方法，你可以创建一个标签数组，然后使用 `from_lists`，把这个数组传入。

```
1 | labels = [('3' if '/3/' in str(x) else '7') for x in fn_paths]  
2 | labels[:5]  
  
1 | data = ImageDataBunch.from_lists(path, fn_paths, labels=labels, ds_tfms=tfms,  
2 | size=24)  
3 | data.classes
```

这里有很多不同的创建标签的方法。这一周里，试试使用它们。

你可能想知道怎样使用它们。哪里有这样的信息。我给你们看个很酷的东西。你们知道怎样获取文档：

```
1 | doc(ImageDataBunch.from_name_re)
```

[\[在文档中显示\]](#)

from_name_re

[source]

```
from_name_re ( path : PathOrStr , fnames : FilePathList , pat : str ,
    valid_pct : int = 0.2 , test : str = None , kwargs )
```

Creates an `ImageDataBunch` from `fnames`, calling a regular expression (containing one *re group*) on the file names to get the labels, putting aside `valid_pct` for the validation. In the same way as `ImageDataBunch.from_csv`, an optional `test` folder contains unlabelled data.

Our previously created dataframe contains the labels in the filenames so we can leverage it to test this new method. `ImageDataBunch.from_name_re` needs the exact path of each file so we will append the data path to each filename before creating our `ImageDataBunch` object.

```
fn_paths = [path/name for name in df['name']]; fn_paths[:2]

[PosixPath('/home/ubuntu/.fastai/data/mnist_sample/train/3/7463.png'),
 PosixPath('/home/ubuntu/.fastai/data/mnist_sample/train/3/21102.png')]

pat = r"/(\d)/\d+\.\png$"
data = ImageDataBunch.from_name_re(path, fn_paths, pat=pat, ds_tfms=tfms, size=24)
```

```
data.classes
```

```
['3', '7']
```

from_name_func

[source]

我演示的代码都是今天上午从文档里拷过来的。你可以看到这就是我刚刚使用的代码。fastai的文档不仅告诉你要做什么，还会告诉你如何一步步去做。这可能是最酷的事了，如果你访问[fastai/fastai docs](#)，点击[docs/src](#)，可以看到我们所有的文档都是Jupyter Notebooks格式。你可以用git克隆这个仓库，你可以运行文档里的每行代码。

这对我来说是一个实验的极好的例子。你在文档里的读到的所有内容都有可以运行的使用实际数据集的例子，数据集已经包含在仓库里。你可以在浏览器里尝试使用每个函数，看看它们的输入输出是什么。

[1:37:27]

提问：这个库会默认使用多GPU并行计算吗？

这个库会默认使用多CPU，但默认只用一个GPU。我们在第二部分才介绍多GPU。这很简单，你可以在论坛上找到这方面的内容。现在，大多数人用不上这个。

Question: 这个库支持类似核磁共振/层析成像扫描的3D数据吗？

会的。已经有一个讨论这个问题的帖子。在MOOC发布时，它大概会支持3D。

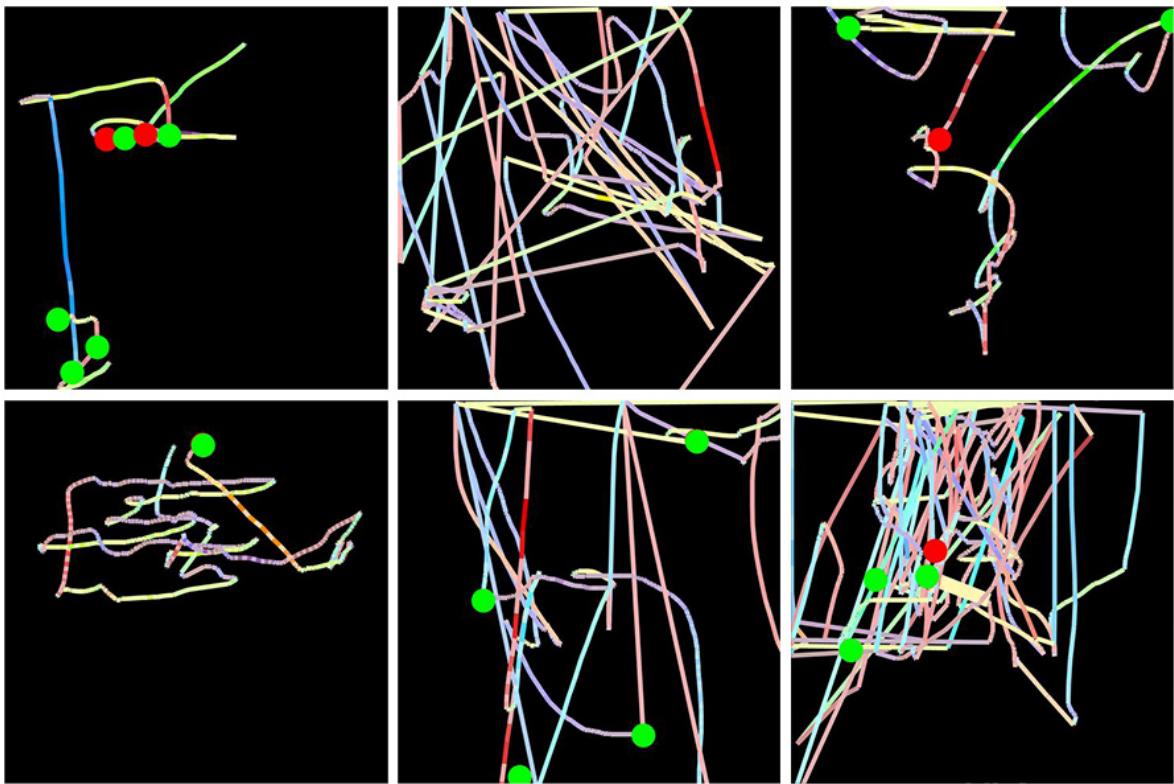
Splunk反欺诈软件 [1:38:10]

[blog](#)

最后，我给你们看一个有意思的东西，你们可以做类似的练习。

我之前提到过一个在Splunk工作的学员做了一个新的反欺诈程序，Splunk是一个纳斯达克上市的成

功公司。这是他在fastai第一部分课程的课程项目：



他取到安装了Splunk analytics的用户的远程数据 (telemetry)，观察鼠标的移动，把鼠标移动做成图像。他把速度转成颜色，鼠标点击转成圆点。他使用我们刚刚看到的代码，用我们刚刚学习的方式创建了一个CNN来训练他的欺诈检测模型。他把一些本来不是图形的东西转换成图形，得到了一个欺诈分析领域非常好的结果。

这就是创造性思维的价值。如果你想研究声音，实际上很多研究声音的人创建一份声谱图，把它输入到一个卷积网络中。所以使用这个网络你可以做很多很酷的事情。

这周，配置好你的GPU，尝试运行第一个notebook，确保你能运行第一课的notebook，把里面的内容运行一遍。然后看看你能不能在你自己的数据集上做一遍。在论坛上告诉大家你的成功的结果，哪怕很小的成功。对于你遇到的困难，如果尝试一两个小时后还是解决不了，就求助我们。下周见。