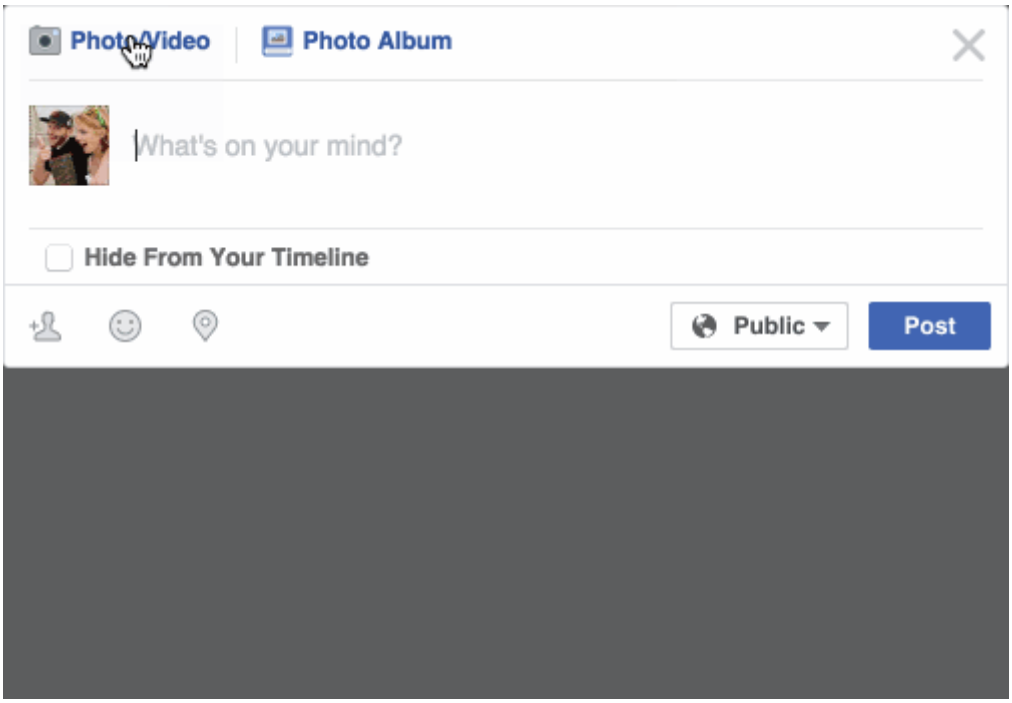
Issue special-edition | 专题

机器学习原来这么有趣！（四）

亚当·盖特吉 12 个月前

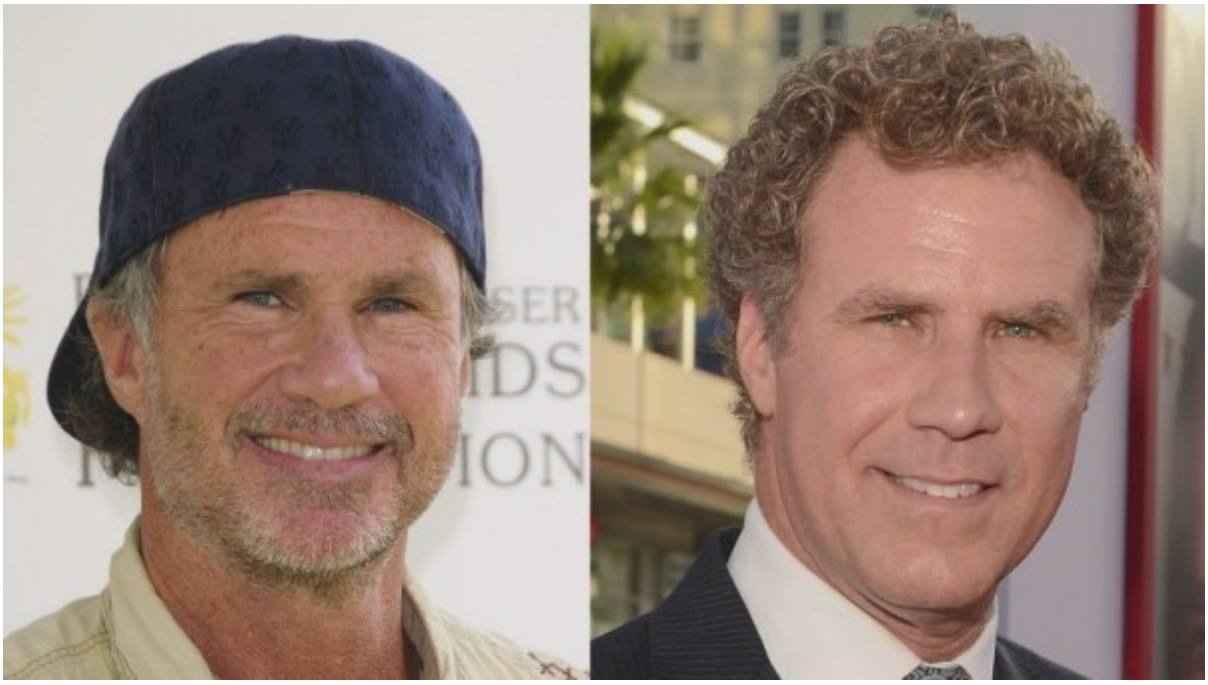
你有没有发现 Facebook 研发出了一种能够在你的照片中识别出你朋友的神奇能力？ 之前，你需要手动点击你朋友的照片，输入他们的名字，然后加上标签。现在，只要你一上传照片，Facebook 就会神奇地标注出你的每一个朋友：



Facebook 会基于你之前的标注，自动标注出你照片中的人。我不知道该说它有用呢还是邪乎呢！

这种技术被称为人脸识别。你的朋友被标记了几次之后，Facebook 的算法就能够识别你朋友的脸。 这是一项非常惊人的黑科技——Facebook 的人脸识别准确率达到了 98%，几乎与人类做得一样好！

让我们来了解一下现代人脸识别是如何工作的！ 但是，识别你的朋友这太容易了。 我们可以最大化扩展这项技术，来解决一个更具挑战性的问题——区分威尔·法瑞尔（Will Ferrell，著名演员）和查德·史密斯（Chad Smith，著名摇滚音乐家）！



其中一个人是威尔·法瑞尔，另一个是查德·史密斯。我保证他们不是同一个人！

如何用机器学习解决复杂问题？

到目前为止，在前三章，我们用机器学习来解决了一些一步就能完成的问题——估计房子的价格、基于现有数据生成新数据[1]、判别图像当中是否包含某个物品。 所有这些问题都可以通过下列步骤解决：选择一个机器学习算法，输入数据，然后获得结果。

但是，人脸识别是由一系列的几个相关问题组成的：

1. 首先，找到一张图片中的所有人脸。
2. 第二，对于每一张脸来说，无论光线明暗或面朝别处，它依旧能够识别出是同一个人的脸。
3. 第三，能够在每一张脸上找出可用于与他人区分的独特之处，比如说眼睛有多大，脸有多长等等。
4. 最后，将这张脸的特点与已知的所有人脸进行比较，以确定这个人的姓名。

作为人类，你的大脑总是在一瞬间自动做出了这些判断。实际上，人类在识别人脸这方面做得**太好了**，以至于他们会在日常物品中同样去「找脸」：

计算机还不具备这种高级泛化能力（**至少现在还不行……**），所以我们必须手把手一步一步教它们怎么做。

我们需要构建一个**流水线**（pipeline）来识别人脸，它可以把上一个步骤的结果发送给下一个步骤。换句话说，我们会将好几个机器学习算法连接到一起：

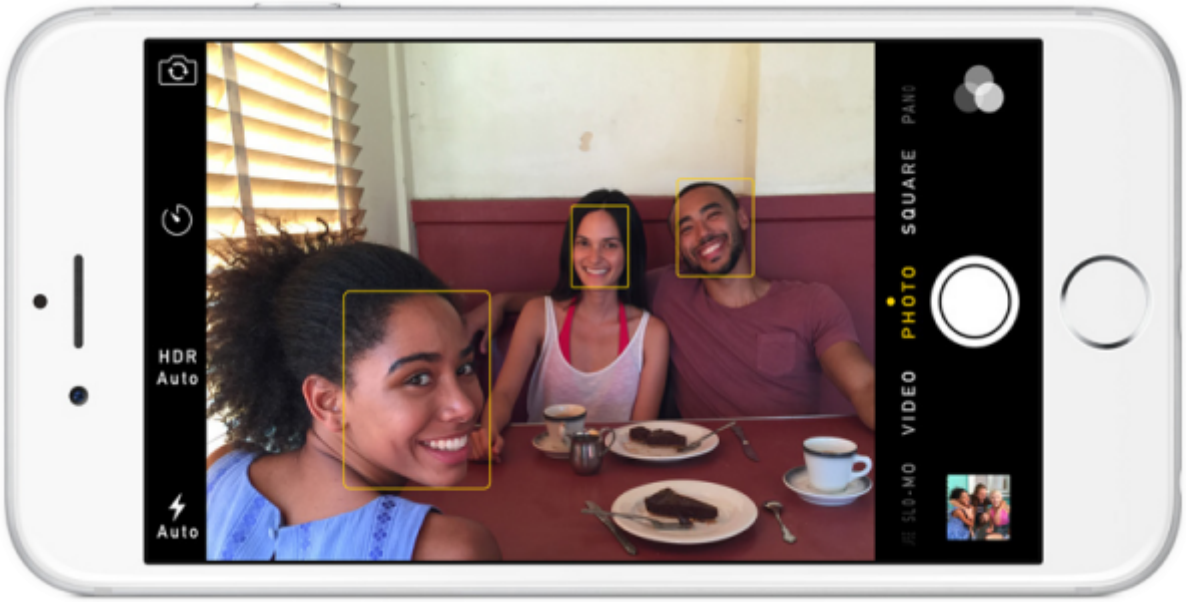
一个基础的人脸识别的流水线是怎样工作的。

人脸识别——分步讲解

让我们一步一步地解决这个问题。 对于每个步骤，我们将学习一个不同的机器学习算法。我并不会完全解释每一个的算法，否则这篇文章就变成了一本教科书。但你会学到每个步骤的精髓，以及如何在 Python 中使用 OpenFace 和 dlib 来构建一个你自己的面部识别系统。

我们流水线的第一步是**人脸检测**。显然，在我们区分人脸之前，我们必须要照片中找到他们才行！

如果你在过去 10 年里使用过相机，你可能已经见过正在运行中的人脸检测功能：

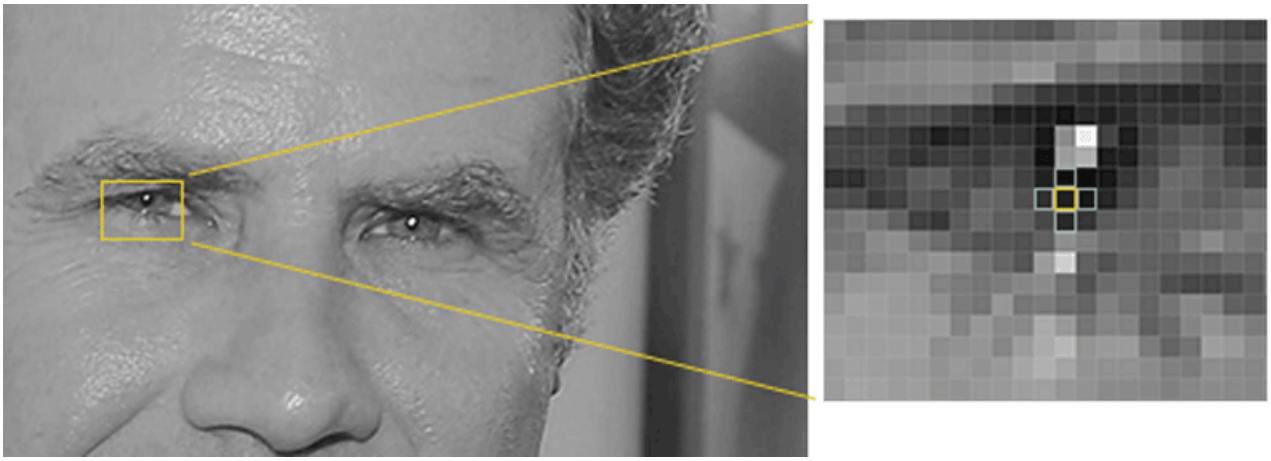


人脸检测是相机很好的一个功能。 当相机自动选中人脸时，它可以确保在拍摄时对焦到所有人脸。 不过，我们使用它另有其因——我们需要找到想要传递到流水线下一步的图像区域。

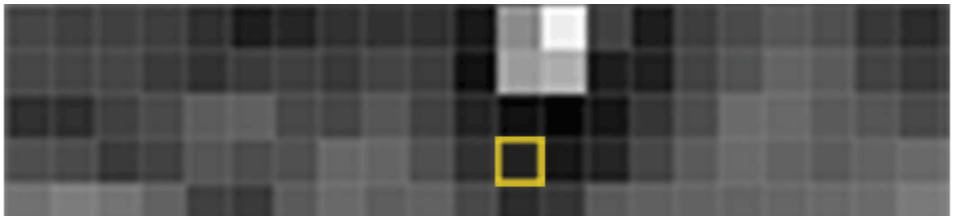
2000 年初的时候， 当保罗·比奥拉（Paul Viola）和迈克尔·琼斯（Michael Jones） 发明了一种能够快速在廉价相机上运行的人脸检测方法之后[2] ，人脸检测在成为了主流。然而现在，更可靠的解决方案出现了。 我们将使用 2005 年发明的一种称为方向梯度直方图（Histogram of Oriented Gradients）的方法，简称 **HOG**。

要在一张图片中找到脸，我们首先将图像转换为黑白，因为我们并不需要颜色数据来找到脸：

然后，我们将查看图片中的每一个像素。 对于单个像素，我们也要查看它周围的其他像素：



我们的目标是找出并比较当前像素与直接围绕它的像素的深度。 然后我们要画一个箭头来代表图像变暗的方向：



看这个像素和它周围的像素，图像向右上方变暗。

如果你对图片中的**每一个像素**重复这个过程，最终每个像素会被一个箭头取代。这些箭头被称为**梯度**（gradients），它们能显示出图像上从明亮到黑暗的流动过程：

用梯度来代替像素这事看起来没有明确目的，但其实背后的理由很充分。如果我们直接分析像素，同一个人明暗不同的两张照片将具有完全不同的像素值。但是如果只考虑亮度变化**方向**（direction）的话，明暗图像将会有同样的结果。这使得问题变得更容易解决！

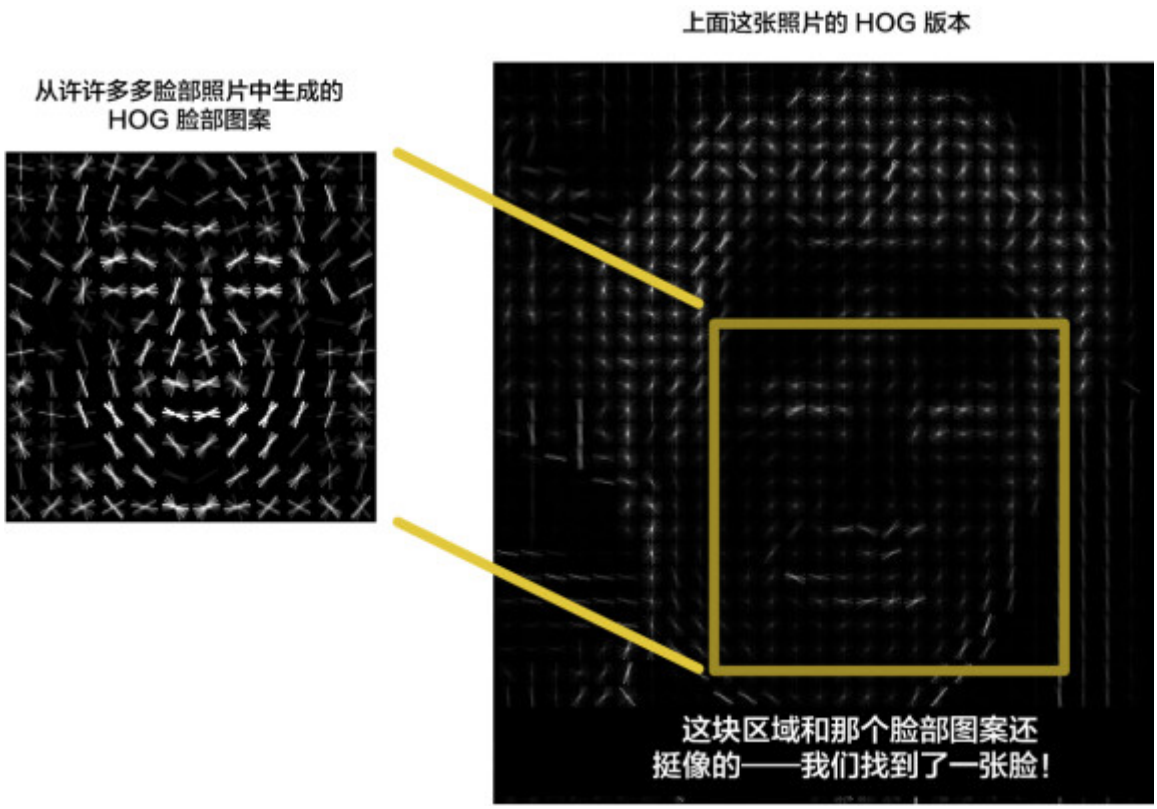
但是保存每个像素的梯度太过细节化了，我们最终很有可能「一叶障目不见泰山」。如果能从更高的角度上观察基本的明暗流动，我们就可以看出图像的基本规律，这会比之前更好。

为了做到这一点，我们将图像分割成一些 16×16 像素的小方块。在每个小方块中，我们将计算出每个主方向上有多少个梯度（有多少指向上，指向右上，指向右等）。然后我们将用指向性最强那个方向的箭头来代替原来的那个小方块。

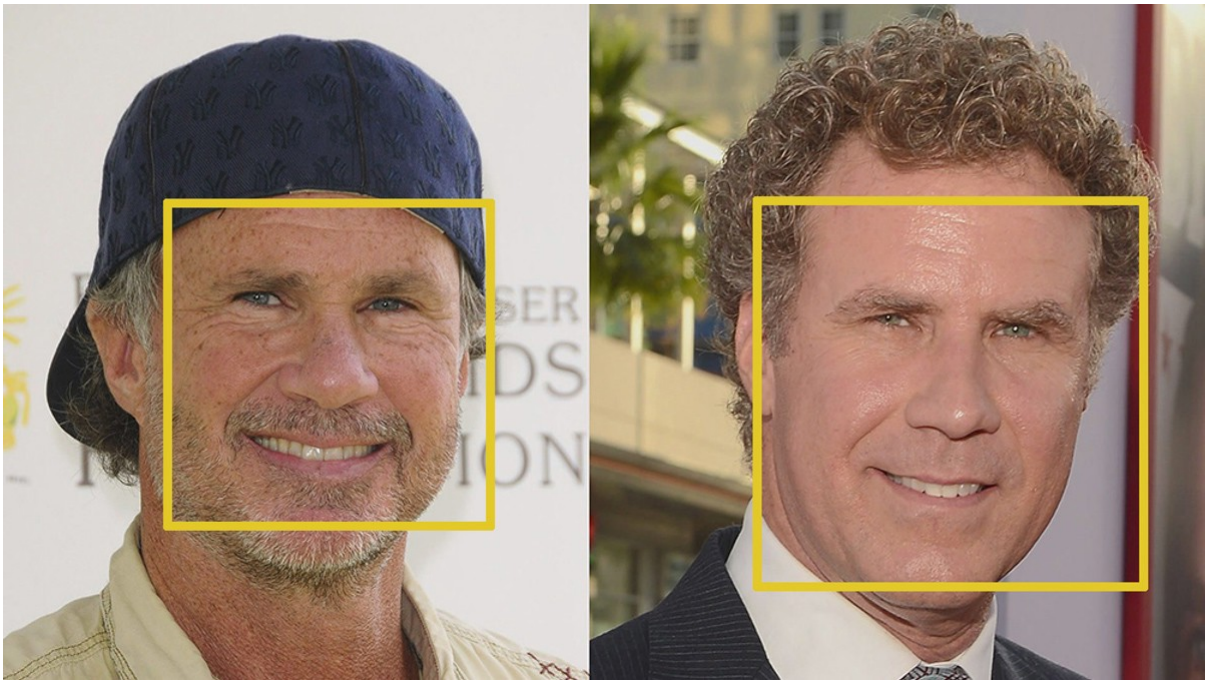
最终的结果是，我们把原始图像转换成了一个非常简单的表达形式，这种表达形式可以用一种简单的方式来捕获面部的基本结构：

原始图像被表示成了 HOG 形式，以捕获图像的主要特征，无论图像明暗度如何。

为了在这个 HOG 图像中找到脸部，我们要所需要做的，就是找到我们的图像中，与已知的一些 HOG 图案中，看起来最相似的部分。这些 HOG 图案都是从其他面部训练数据中提取出来的：



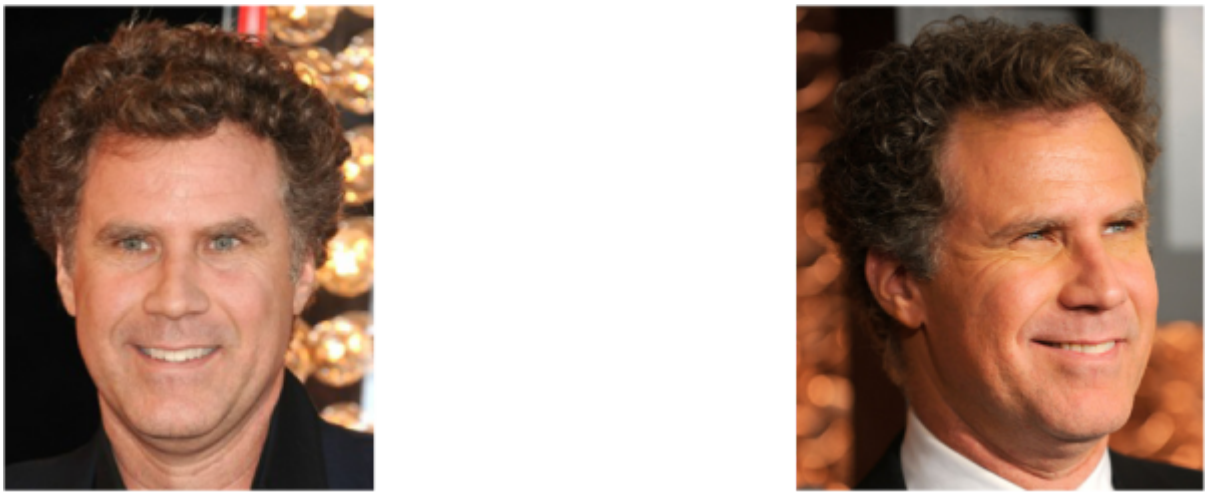
使用这种技术，我们现在可以轻松地在任何图片中找到脸部：



如果你想用 Python 和 dlib 亲手试试看，这些代码显示了如何生成和查看 HOG 图像。

第二步：脸部的不同姿势

哇，我们把图片中的脸部分离出来了。 但现在，我们要处理的问题就是，对于电脑来说，面朝不同方向的同一张脸，是不同的东西：

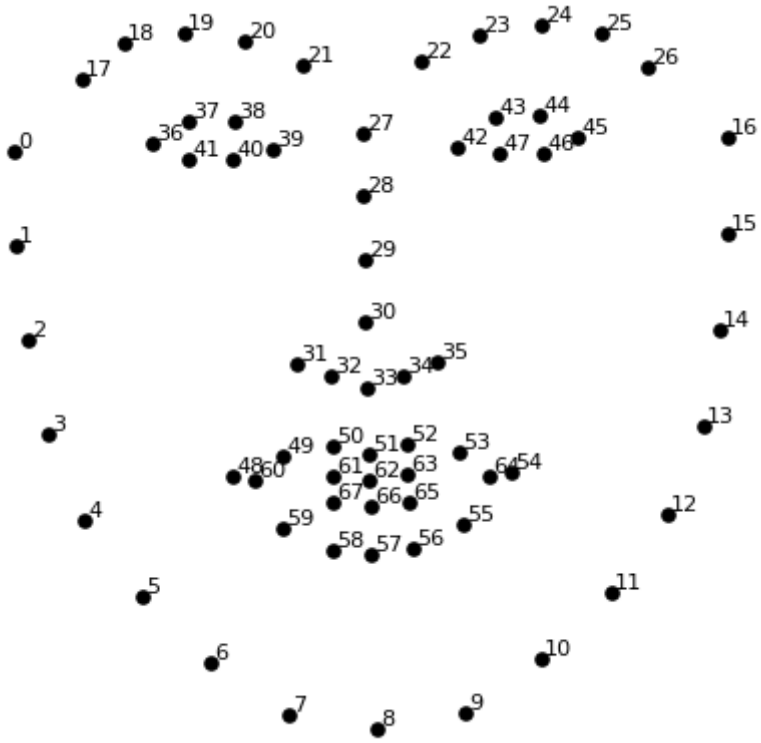


人类可以很轻松地识别出到两个图片都是威尔·法瑞尔，但电脑会认为这两张图片是两个完全不同的人。

为了解决这一点，我们将试图扭曲每个图片，使得眼睛和嘴唇总是在图像中的样本位置（sample place）。 这将使我们在接下来的步骤中，更容易比较脸部之间的不同。

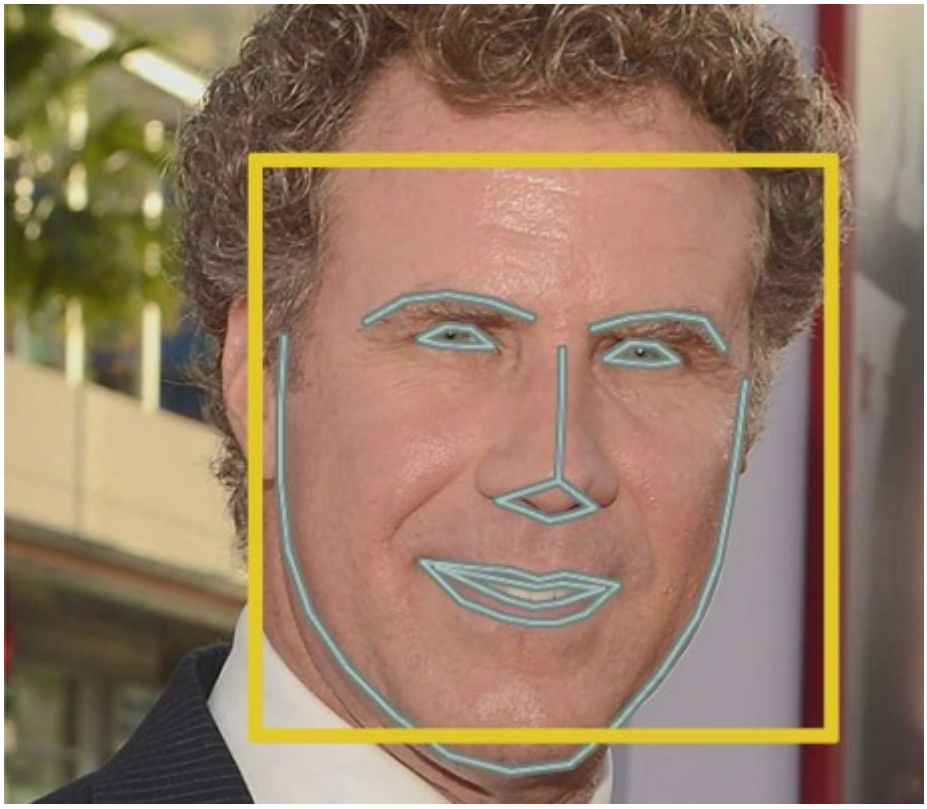
为此，我们将使用一种称为**面部特征点估计**（face landmark estimation）的算法。 很多方法都可以做到这一点，但这次我们会使用由 瓦希德·卡奇米（Vahid Kazemi）和约瑟菲娜·沙利文（Josephine Sullivan）在 2014 年发明的方法。

这一算法的基本思路是找到 68 个人脸上普遍存在的特定点（称为**特征点**， landmarks）——包括下巴的顶部、每只眼睛的外部轮廓、每条眉毛的内部轮廓等。接下来我们训练一个机器学习算法，让它能够在任何脸部找到这 68 个特定的点：



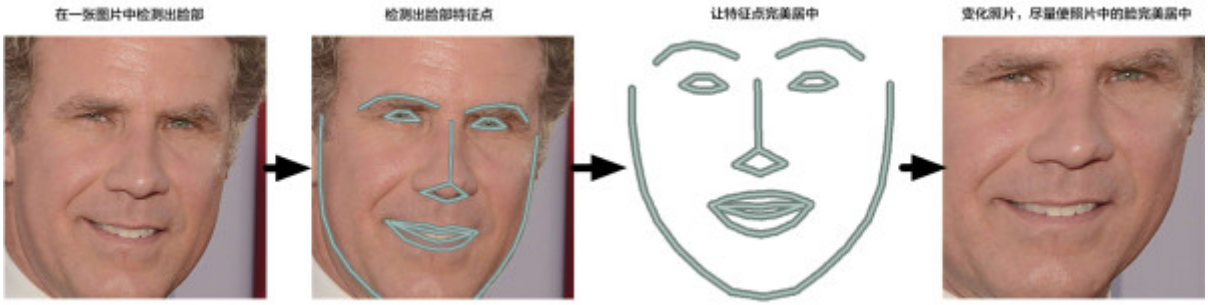
我们将在每一张脸上定位的 68 个特征点。这张图片的作者是在 OpenFace 工作的卡内基梅隆大学 Ph.D. 布兰东·阿莫斯（Brandon Amos）。

这是在测试图片上定位 68 个特征点的结果：



友情提示：你也可以使用这一技术来实现自己的 Snapchat 实时 3D 脸部过滤器！

现在，我们知道了眼睛和嘴巴在哪儿，我们将图像进行旋转、缩放和错切，使得眼睛和嘴巴尽可能靠近中心。我们不会做任何花哨的三维扭曲，因为这会让图像失真。我们只会使用那些能够保持图片相对平行的基本图像变换，例如旋转和缩放（称为仿射变换）：



现在无论人脸朝向哪边，我们都能将眼睛和嘴巴向中间挪动到大致相同的位置。这将使我们的下一步更加准确。

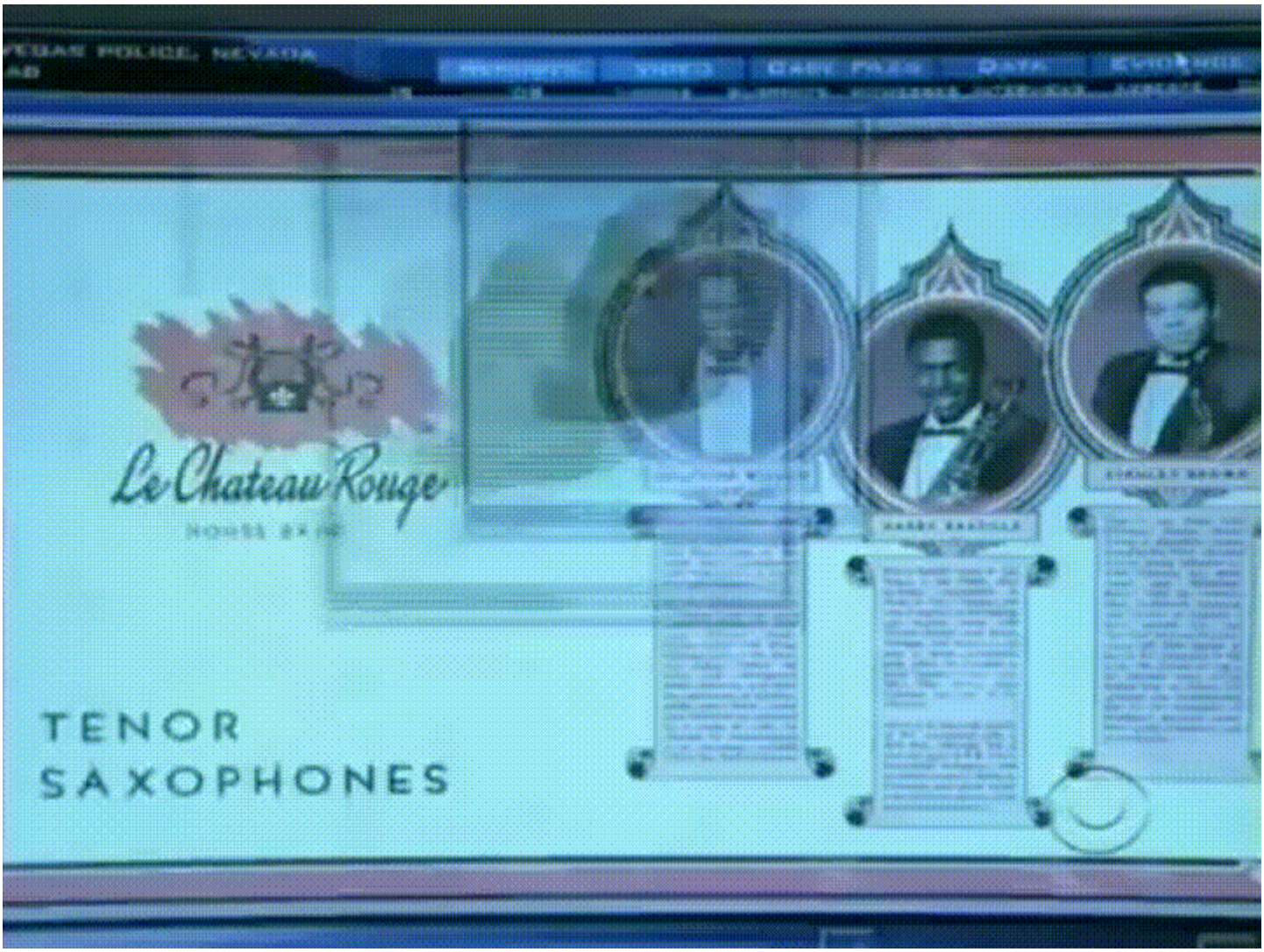
如果你想用 Python 和 dlib 亲手试试看这一步的话，这里有一些代码帮你寻找脸部特征点并用这些特征点完成图像变形。

现在我们要面临最核心的问题了——如何区分不同的人脸。这才是这件事的有趣之处！

最简单的人脸识别方法，就是直接把我们在第二步中发现的未知人脸，与我们已经标注了的人脸图片作比较。当我们发现未知的面孔与一个以前标注过的面孔看起来及其相似的时候，它肯定是同一个人。似乎这主意看起来不错，对吧？

实际上这种方法有一个巨大的问题。像 Facebook 这种拥有数十亿用户和数万亿张照片的网站，是不可能去循环比较每张先前标记的脸的，这浪费的时间太长了。他们需要在毫秒内识别人脸，而不是几个小时。

我们需要的方法是一种从每张人脸上提取一些基本的测量数值。然后，我们可以用同样的方式测量未知的面孔，并找到最接近测量数值的那张已知的脸。例如，我们可以测量每个耳朵的大小、眼睛之间的间距、鼻子的长度等。如果你曾经看过像《犯罪现场调查》这样的电视剧，你就知道我在说什么了。



测量人脸的最可靠的方法

好的，那么我们应该测量面部的哪些数值，来建立我们的已知脸部数据库呢？耳朵的大小？鼻子的长度？眼睛的颜色？还有什么？

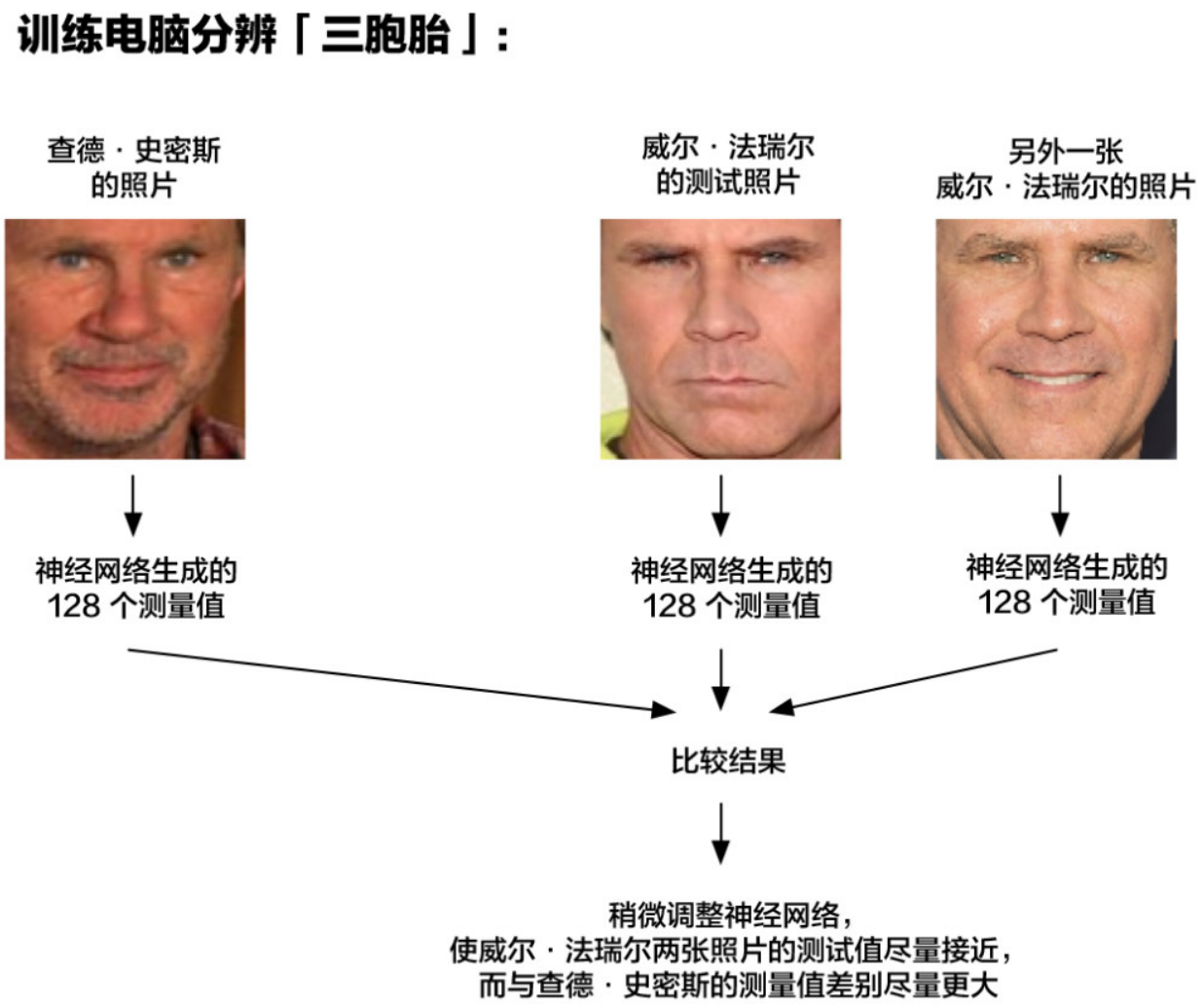
事实证明，对于我们人类来说一些显而易见的测量值（比如眼睛颜色），对计算机来说没什么意义。研究人员发现，最准确的方法是让计算机自己找出它要收集的测量值。深度学习比人类更懂面部哪些部分的测量值比较重要。

所以，解决方案是训练一个深度卷积神经网络（就像我们在第三章做的那样）。但是，并不是让它去识别图片中的物体，这一次我们的训练是要让它为脸部生成 128 个测量值。

每次训练要观察三个不同的脸部图像：

1. 加载一张已知的人的面部训练图像
2. 加载同一个人的另一张照片
3. 加载另外一个人的照片

然后，算法查看它自己为这三个图片生成的测量值。再然后，稍微调整神经网络，以确保第一张和第二张生成的测量值接近，而第二张和第三张生成的测量值略有不同。



在为几千个人的数百万图像重复该步骤数百万次之后，神经网络学习了如何可靠地为每个人生成 128 个测量值。对于同一个人的任何十张不同的照片，它都应该给出大致相同的测量值。

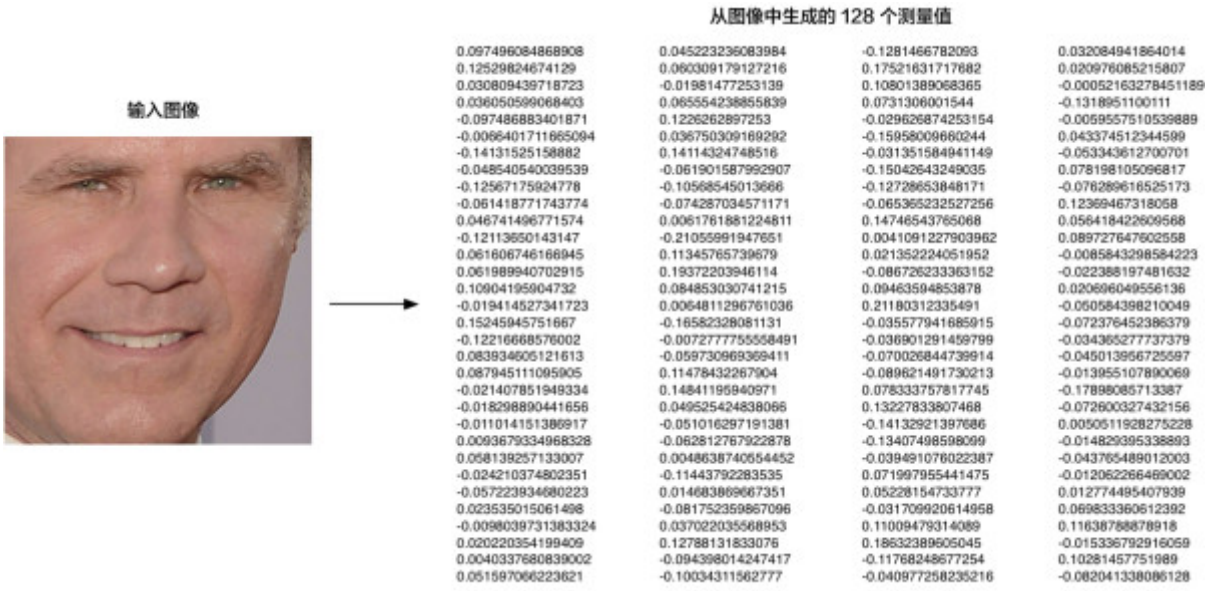
机器学习专业人士把每张脸的 128 个测量值称为一个**嵌入**（embedding）。将复杂的原始数据（如图片）缩减为可由计算机生成的一个数列的方法，在机器学习（特别是语言翻译）中出现了很多次。我们正在使用的这种脸部提取方法是由 Google 的研究人员在 2015 年发明的，但也有许多类似方法存在。

给我们的脸部图像编码

这个通过训练卷积神经网络来输出脸部嵌入的过程，需要大量的数据和强大的计算能力。即使使用昂贵的 Nvidia Telsa 显卡，你也需要大约 24 小时的连续训练，才能获得良好的准确性。

布了几个训练过可以直接使用的网络。谢谢布兰东·阿莫斯和他的团队！

所以我们需要做的，就是通过他们预训练的网络来处理我们的脸部图像，以获得 128 个测量值。这是我们测试图像的一些测量值：



那么，这 128 个数字到底测量了脸部的哪些部分？我们当然不知道，但是这对我们并不重要。我们关心的是，当看到同一个人两张不同的图片时，我们的网络需要能得到几乎相同的数值。

如果你想自己尝试这个步骤，OpenFace 提供了一个 lua 脚本，它可以生成一个文件夹中所有图像的嵌入，并将它们写入 csv 文件。点此查看如何运行。

第四步：从编码中找出人的名字

最后这一步实际上是整个过程中最简单的一步。我们要做的就是找到数据库中，与我们的测试图像的测量值最接近的那个人。

你可以通过任何基本的机器学习分类算法来达成这一目标。我们并不需要太花哨的深度学习技巧。我们将使用一个简单的线性 SVM 分类器，但实际上还有很多其他的分类算法可以使用。

我们需要做的是训练一个分类器，它可以从一个新的测试图像中获取测量结果，并找出最匹配的那个人。分类器运行一次只需要几毫秒，分类器的结果就是人的名字！

所以让我们试一下我们的系统。首先，我使用威尔·法瑞尔、查德·史密斯和吉米·法伦（Jimmy Falon）三人每人 20 张照片的嵌入来训练分类器：

嗯……就是这些训练数据！

接下来，我在这个分类器上运行了威尔·法瑞尔和查德·史密斯在吉米·法伦的节目上互相模仿的那个视频的每一帧：

结果成功了！不同角度的脸部，甚至是侧脸，它都能捕捉到！

你自己做一遍

让我们回顾一下我们的步骤：

1. 使用 HOG 算法给图片编码，以创建图片的简化版本。使用这个简化的图像，找到其中

3. 把上一步得到的面部图像放入神经网络中，神经网络知道如何找到 128 个特征测量值。保存这 128 个测量值。
4. 看看我们过去已经测量过的所有脸部，找出哪个人的测量值和我们要测量的面部最接近。这就是你要找的人！

现在你知道这一切都是如何运行的了，这里是如何使用 OpenFace 在你自己的电脑上运行整个人脸识别系统的说明：

开始之前

确保你已经安装了 python、OpenFace 和 dlib。你也可以在这里手动安装，或者使用一个已经设定好的 docker image：

```
docker pull bamos/openface
docker run -p 9000:9000 -p 8000:8000 -t -i bamos/openface /bin/bash
cd /root/openface
```

友情提示：如果你正在 OSX 上使用 Docker，你可以这样使你的 OSX /Users/ 文件夹在 docker image 中可见：

```
docker run -v /Users:/host/Users -p 9000:9000 -p 8000:8000 -t -i bamos/openface /bin/bash
cd /root/openface
```

然后你就能访问你在 docker image 中 /host/Users/... 的 OSX 文件

```
ls /host/Users/
```

在 openface 文件中建立一个名为 ./training-images/ 的文件夹。

```
mkdir training-images
```

第二步

为你想识别的每个人建立一个子文件夹。例如：

```
mkdir ./training-images/will-ferrell/
mkdir ./training-images/chad-smith/
mkdir ./training-images/jimmy-fallon/
```

第三步

将每个人的所有图像复制进对应的子文件夹。确保每张图像上只出现一张脸。不需要裁剪脸部周围的区域。OpenFace 会自己裁剪。

第四步

从 openface 的根目录中运行这个 openface 脚本。

首先，进行姿势检测和校准：

```
./util/align-dlib.py ./training-images/ align outerEyesAndNose ./aligned-images/ --size 96
```

这将创建一个名为 ./aligned-images/ 的子文件夹，里面是每一个测试图像裁剪过、并且对

其次，从对齐的图像中生成特征文件：

```
./batch-represent/main.lua -outDir ./generated-embeddings/ -data ./aligned-images/
```

运行完后，这个 ./generated-embeddings/ 子文件夹会包含一个带有每张图像嵌入的 csv 文件。

第三，训练你的面部检测模型：

```
./demos/classifier.py train ./generated-embeddings/
```

这将生成一个名为 ./generated-embeddings/classifier.pkl 的新文件，其中包含了你用来识别新面孔的 SVM 模型。

到这一步为止，你应该有了一个可用的人脸识别器！

第五步：识别面孔！

获取一个未知脸孔的新照片，然后像这样把它传递入分类器脚本中：

```
./demos/classifier.py infer ./generated-embeddings/classifier.pkl your_test_image.jpg
```

你应该会得到像这样的一个预测：

```
=== /test-images/will-ferrel-1.jpg ===
```

至此，你已经完成了一个预测了。你也可以修改 ./demos/classifier.py 这个 python 脚本，来让它匹配其他人的脸。

重要提示：

- 如果你得到的结果不够理想，试着在第三步为每个人添加更多照片（特别是不同姿势的照片）。
- 即使完全不知道这个面孔是谁，现在这个脚本仍然会给出预测。在真实应用中，低可信度（low confidence）的预测可能会被直接舍弃，因为很有可能它们就是错的。

作者：Adam Geitgey

原文：https://medium.com/@ageitgey/machine-learning-is-fun-part–4-modern-face-recognition-with-deep-learning-c3cffc121d78#.ugpgrc4eq

译文：https://zhuanlan.zhihu.com/p/24567586

译者：巡洋舰科技——赵95

校对：林沁

亚当·盖特吉

软件工程师，Groupon 工程部总监，带领团队维护 Groupon 网页端。热爱计算机与机器学习。推特帐号 @ageitgey 。

发表评论

电子邮件地址不会被公开。 必填项已用*标注

评论

姓名 *

电子邮件 *

站点

一个图灵小测试 *

× 9 = 72

发表评论

下一篇

机器学习原来这么有趣（五）