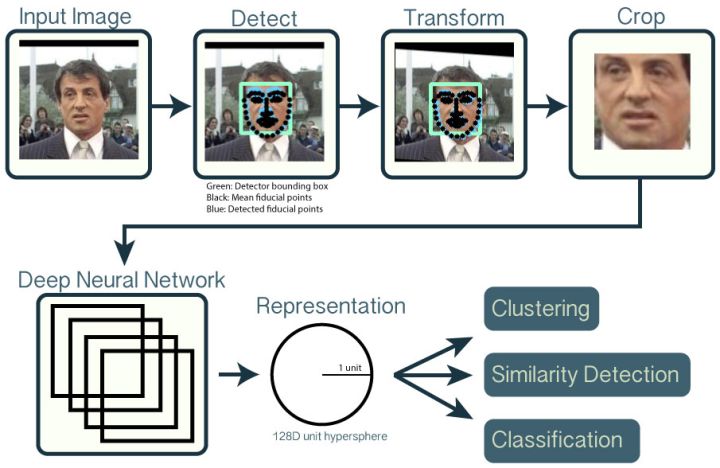
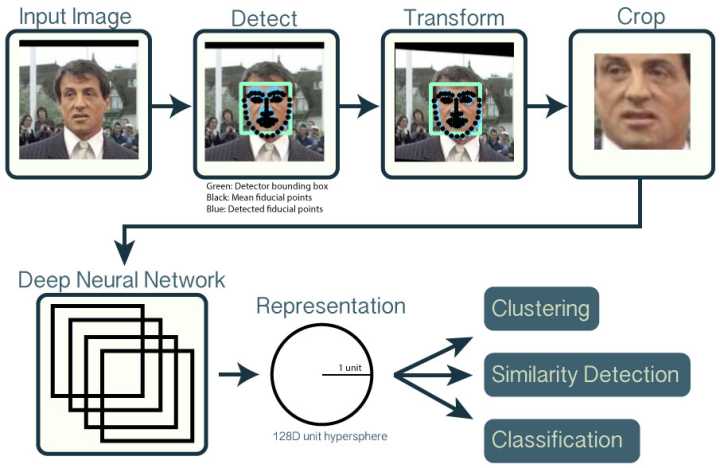
作者：何之源  
链接：https://www.zhihu.com/question/60759296/answer/180176680  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

我这里简单讲下[OpenFace](https://link.zhihu.com/?target=https://cmusatyalab.github.io/openface/" \t "_blank)中实现人脸识别的pipeline**，这个pipeline可以看做是使用深度卷积网络处理人脸问题的一个基本框架，很有学习价值**，它的结构如下图所示：



## 1、Input Image -> Detect

输入：原始的可能含有人脸的图像。

输出：人脸位置的bounding box。

这一步一般我们称之为“人脸检测”（Face Detection），在OpenFace中，使用的是dlib、OpenCV现有的人脸检测方法。此方法与深度学习无关，使用的特征是传统计算机视觉中的方法（一般是Hog、Haar等特征）。

对人脸检测这一步感兴趣的可以参考下列资料：

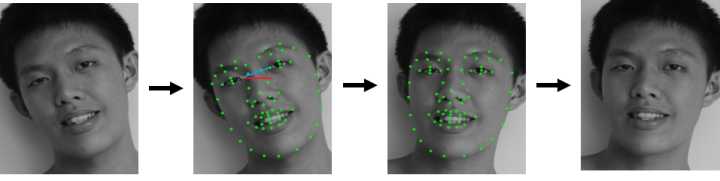
* dlib的实现：[http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html](https://link.zhihu.com/?target=http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html" \t "_blank)
* openCV的实现：[Face Detection using Haar Cascades](https://link.zhihu.com/?target=http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html" \t "_blank)

## 2、Detect -> Transform -> Crop

输入：原始图像 + 人脸位置bounding box

输出：“校准”过的只含有人脸的图像

对于输入的原始图像 + bounding box，这一步要做的事情就是要检测人脸中的关键点，然后根据这些关键点对人脸做对齐校准。所谓关键点，就是下图所示的绿色的点，通常是眼角的位置、鼻子的位置、脸的轮廓点等等。有了这些关键点后，我们就可以把人脸“校准”，或者说是“对齐”。解释就是原先人脸可能比较歪，这里根据关键点，使用仿射变换将人脸统一“摆正”，尽量去消除姿势不同带来的误差。这一步我们一般叫Face Alignment。



在OpenFace中，这一步同样使用的是传统方法，特点是比较快，对应的论文是：

* [https://pdfs.semanticscholar.org/d78b/6a5b0dcaa81b1faea5fb0000045a62513567.pdf](https://link.zhihu.com/?target=https://pdfs.semanticscholar.org/d78b/6a5b0dcaa81b1faea5fb0000045a62513567.pdf" \t "_blank)

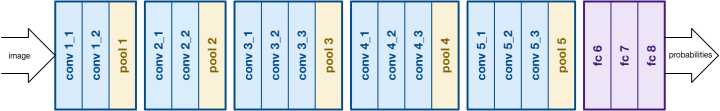
## 3、Crop -> Representation

输入：校准后的单张人脸图像

输出：一个向量表示。

这一步就是使用深度卷积网络，将输入的人脸图像，转换成一个向量的表示。在OpenFace中使用的向量是128x1的，也就是一个128维的向量。

我们可以先看一下VGG16的模型：



VGG16是深度学习中一个比较简单的基本模型。输入神经网络的是图像，经过一系列卷积后，全连接分类得到类别概率。



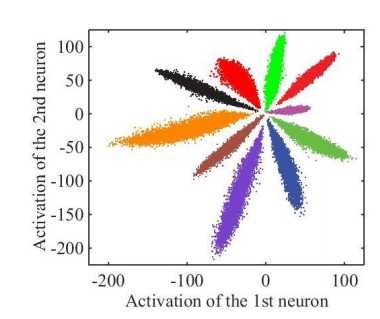
**在通常的图像应用中，我们可以去掉全连接层，用计算的特征（一般就是卷积层的最后一层，e.g. 图中的conv5\_3）来当作提取的特征进行计算。但如果对人脸识别问题同样采用这样的方法，即，使用卷积层最后一层做为人脸的“向量表示”，效果其实是不好的。**如何改进？我们之后再谈，这里先谈谈我们希望这种人脸的“向量表示”应该具有哪些性质。

在理想的状况下，我们希望“向量表示”之间的距离就可以直接反映人脸的相似度：

* 对于同一个人的人脸图像，对应的向量的欧几里得距离应该比较小。
* 对于不同人的人脸图像，对应的向量之间的欧几里得距离应该比较大。

这种表示实际上就可以看做某种“embedding”。**在原始的VGG16模型中，我们使用的是softmax损失，没有对每一类的向量表示之间的距离做出要求。**所以不能直接用作人脸表示。

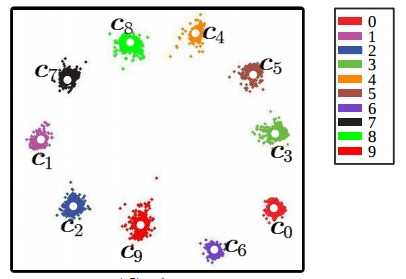
举个例子，使用CNN对MNIST进行分类，我们设计一个特殊的卷积网络，**让最后一层的向量变为2维**，此时可以画出每一类对应的2维向量表示的图（图中一种颜色对应一种类别）：



上图是我们直接使用softmax训练得到的结果，它就不符合我们希望特征具有的特点：

* 我们希望同一类对应的向量表示尽可能接近。但这里同一类（如紫色），可能具有很大的类间距离。
* 我们希望不同类对应的向量应该尽可能远。但在图中靠中心的位置，各个类别的距离都很近。

那么训练人脸特征表示的正确姿势是什么？其实有很多种方法。一种方法就是使用“center loss”。centor loss实际上是在softmax的loss上再加入一个损失，这个损失对每一类规定了一个“中心”点，每一类的特征应该离这个中心点比较近，而不同类的中心点离的比较远。加入center loss后，训练出的特征大致长这样：



这样的特征表示就比较符合我们的要求了。center loss的原始论文在这里：[http://ydwen.github.io/papers/WenECCV16.pdf](https://link.zhihu.com/?target=http://ydwen.github.io/papers/WenECCV16.pdf" \t "_blank) 。上面这两幅图同样是从这篇论文中截下来的。

顺带一提，除了center loss外。学习人脸特征表示的方法还有很多，如triplet loss（论文地址：[A Unified Embedding for Face Recognition and Clustering](https://link.zhihu.com/?target=https://arxiv.org/abs/1503.03832" \t "_blank)）。triplet loss直接这样的用三元组（A的图像1，A的图像2，B的图像）来训练网络。去掉了最后的分类层，强迫神经网络对相同的人脸图像（三元组中的同一人A）建立统一的表达。

## 4、实际应用

输入：人脸的向量表示。

有了人脸的向量表示后，剩下的问题就非常简单了。**因为这种表示具有相同人对应的向量的距离小，不同人对应的向量距离大的特点。**接下来一般的应用有以下几类：

* 人脸验证（Face Identification）。就是检测A、B是否是属于同一个人。只需要计算向量之间的距离，设定合适的报警阈值（threshold）即可。
* 人脸识别（Face Recognition）。这个应用是最多的，给定一张图片，检测数据库中与之最相似的人脸。显然可以被转换为一个求距离的最近邻问题。
* 人脸聚类（Face Clustering）。在数据库中对人脸进行聚类，直接K-Means即可。

## 5、后记

以上给大家介绍了OpenFace中处理人脸问题的pipeline。**需要特别指出的是，人脸相关的问题是一个比较大的方向，一篇文章显然是说不清楚的，这里只是基于OpenFace，对比较重要的方法还有名词做了一个解释。**在OpenFace中，为了速度的考虑，提取人脸特征之前的Face Detection和Face Alignment就是使用的传统方法。实际上也可以换用精度更高的深度学习相关方法，比如在中科院山世光老师开源的人脸识别引擎[seetaface/SeetaFaceEngine中](https://link.zhihu.com/?target=https://github.com/seetaface/SeetaFaceEngine" \t "_blank)，Face Alignment使用就是一个基于autoencoder网络的方法。另外，学习人脸特征同样有适合不同场景的不同方法，这些都是要进一步学习的。