人脸识别

人脸识别系统通常被分成两大类：

①**人脸验证**：“这是不是本人”，需要通过刷身份证（或者能证明身份的有效证件）以及摄像头拍摄人脸照片，然后通过验证照片是否是你本人，比如车站人脸识别，手机人脸解锁功能，这些都是 1:1 的匹配问题。

②**人脸识别**：“这个人是谁”，只需要通过摄像头去拍摄你的人脸，检测拍摄的人脸是否存在于系统中，比如公司的刷脸考勤，这样就不要去刷 ID 卡了，这是 1:K 的匹配问题

通过下面项目的内容，将学到下面的东西：

①实现人脸识别中的三元组损失函数

②使用一个已经训练好的模型将人脸图像映射到 128-dimension encoding

③使用这些 encoding 来执行人脸验证和人脸识别

（Note：在此次项目中，使用一个训练好的模型，该模型使用了 channels_first，这个变准现在 DL 社区还没有统一的标准（channels_first or channels_last））

这样我们先导入包

```
from keras.models import Sequential
from keras.layers import Conv2D, ZeroPadding2D, Activation, Input, concatenate
from keras.models import Model
from keras.layers.normalization import BatchNormalization
from keras.layers.pooling import MaxPooling2D, AveragePooling2D
from keras.layers.merge import Concatenate
from keras.layers.core import Lambda, Flatten, Dense
from keras.engine.topology import Layer
from keras import backend as K
K.set_image_data_format("channels_first")   ###这里使用通道优先
import cv2
import os
import numpy as np
from numpy import genfromtxt
import pandas as pd
import tensorflow as tf
from fr_utils import *
from inception_blocks_v2 import *
```

我们能想到最简单的人脸验证，最简单的方法就是逐像素的比较两幅图像，选出图片之间误差小于阈值的，那么则可以判断是同一个人，可是，如果收到光照，明亮，人脸朝向，甚至是微小的差别，这样做的效果一定会很差，那么该怎么办呢？？与使用原始图像之间的距离不同的是让系统学习构建一个 f(img)，也就是构建一个 encoding，对该 encoding 的每个元素(这里比较 128)进行，可以更加准确的判断两幅图像是否属于同一个人

1、将人脸图像编码为 128 位的向量

1.1-使用卷积网络来进行编码

FaceNet 模型需要大量的数据和长时间的训练，因为，在深度学习中，我们常见的操作是加载其他人已经训练过的权值，这里提供了模型的实现方法，可以查看 inception_blocks_v2.py 文件来查看是如何实现 faceRecoModel(input_shape)的。

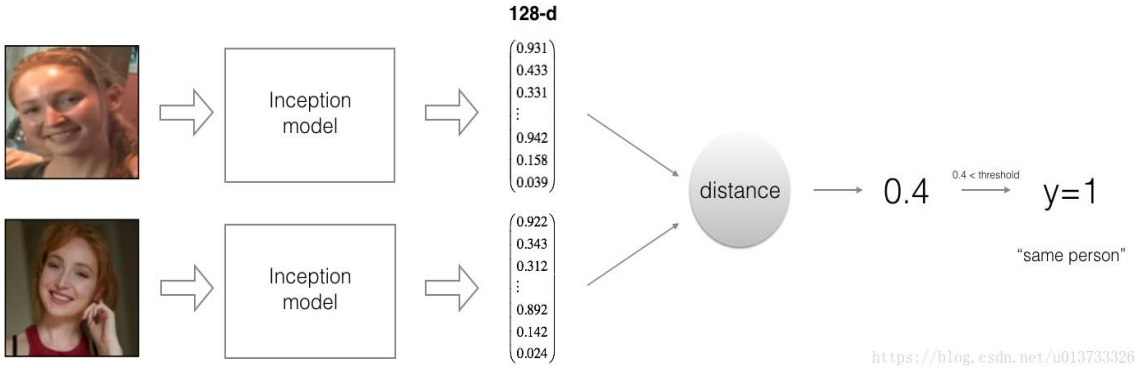
①Inception 网络使用了 96 * 96 的 RGB 图像作为输入数据，图像数量为 m，输入的数据维度为$(m,n_c,n_h,n_w) = (m,3,96,96)$

②输出为(m,128)表示 encoding 后的 m 个 128 维的向量.

下面我们来创建一个人脸识别的模型

```
# 获取模型
FRmodel = faceRecoModel(input_shape=(3, 96, 96))
```

FRmodel.count_params()###可以打印模型的总参数数量



通过计算两个编码距离与阈值之间的误差，可以确定两幅图片是否是一个人

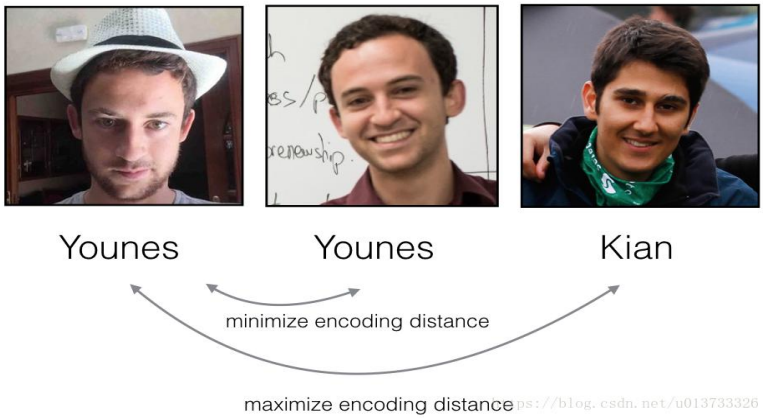
Encoding 是一个很好的方法：

①同一个人的两个图像的编码非常相似

②两个人不同的图像编码非常的不同

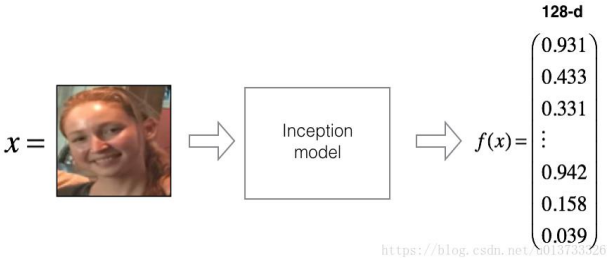
说了这么多，但是训练模型时拿什么损失函数呢，也就是什么样的损失函数能保证 encoding 是一个很好的方法呢？那就是三元损失函数

三元损失函数，会试图将同一个人的两个图像(A 和 P)的编码“拉近”，同时将两个不同的人的图像(A 和 N)的编码进一步“分离”，如下图(从左到右 Anchor, Positive, Negative)：



1.2-The Triplet Loss(三元组损失函数)

对于给定的图像 x，其编码为 f(x)，其中 f 为神经网络的计算函数，如下。



我们使用三元组图像(A, P, N)进行训练：

①A-是一个人的图像

②P-是相对于“Anchor”的同一个人的另外一张图像

③N-是相对于“Anchor”的不同的人的另外一张图像

所以我们希望编码 f（A）接近 f（P）,即$||f(A)-f(P)||^2$尽可能的小。而$||f(A)-f(N)||^2$尽可能的大，我们还要保证图像 A 与图像 P 的差值至少比图像 N 的差值相差α（这里α=0.2）

$||f(A)-f(P)||^2+\alpha \leq ||f(A)-f(N)||^2$

$||f(A)-f(P)||^2+\alpha-||f(A)-f(N)||^2 \leq 0$

这样就可以定义 Loss function 为：

$L(A,P,N) = max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2+\alpha, 0)$

那么，对于 m 组训练样本，cost function 为:

$$J = \sum_{i=1}^{m} L(A^{(i)}, P^{(i)}, N^{(i)})$$

那么我们来看一下代码实现：

```
# 1、下面定义 triplet_loss
def triplet_loss(y_true, y_pred, alpha = 0.2):
    """
    Implemention of the triplet loss as defined by formula
    :param y_true: true labels
    :param y_pred: python list containing three object:
                        anchor: the encodings for the anchor images, of shape(None, 128)
                        positive: the encodings for the positive images, of shape(None, 128)
                        negative: the encodings for the negative images, of shape(None, 128)
    :param alpha:
    :return:
    loss: real numbers, values of the loss
    """
    anchor, positive, negative = y_pred[0], y_pred[1], y_pred[2]

    ### shart code ###
    # Step 1: Compute the (encoding) distance between the anchor and the positive
    pos_dist = tf.reduce_sum(tf.subtract(anchor, positive), axis=-1)
    # Step 2: Computer the (encoding) distance betwween the anchor and the negative
    neg_dist = tf.reduce_sum(tf.subtract(anchor, negative), axis=-1)
    # Step 3: Subtract the two previous distances and add alpha
```

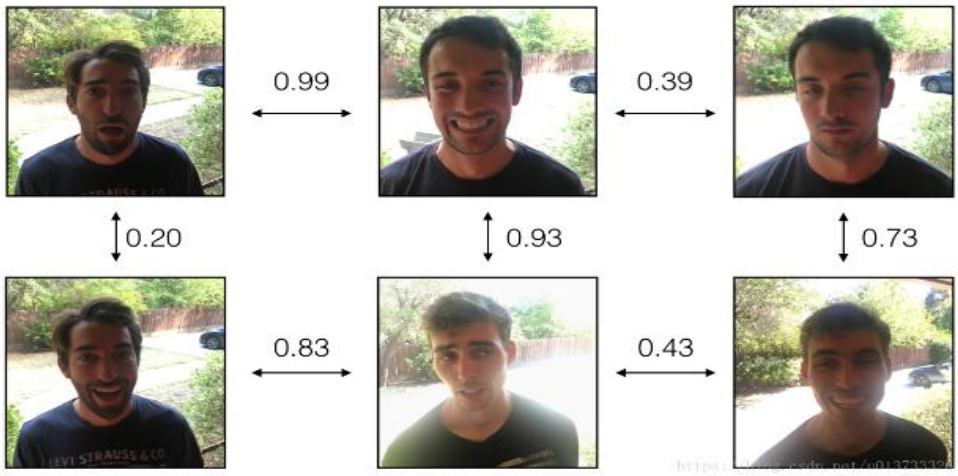
```
    basic_loss = tf.add(tf.subtract(pos_dist, neg_dist), alpha)
    # Step 4: Take the maximum of basic_loss and 0.0. Sum over the training examples
    loss = tf.reduce_sum(tf.maximum(basic_loss, 0.0))
    ### end code ###
    return loss
```

2、加载模型

FaceNet 是通过最小化三元组损失来训练的，但是由于训练需要大量的数据和时间，所以我们不会从头训练，相反，我会加载一个已经训练好了的模型，运行下列代码来加载模型，可能会需要几分钟的时间。

```
# 2、Loading the trained model
"""
FaceNet 是通过最小化三元组损失函数来训练的，但是由于训练需要大量的数据和时间，所以我们不会从头开始训练，
相反，我们会加载一个已经训练好的模型，运行下列代码来加载模型，只需要几分钟的时间
"""
import time
start_time = time.clock()

#编译模型
FRmodel.compile(optimizer="adam", loss=triplet_loss, metrics=["accuracy"])

#加载权值
load_weights_from_FaceNet(FRmodel)

end_time = time.clock()

minium = end_time - start_time
print("执行了："+str(int(minium/60)) + "分" + str(int(minium%60)))
```

上面代码还计算了加载模型所需要的时间，另外我们来看一下三个人之间编码的距离的例子。



这部分模型已经构建好，那么如何通过编码距离进行人脸验证和人脸识别呢？？

3、模型应用

之前对“欢乐家”添加了笑脸识别，现在要构建一个人脸验证系统，以便只允许来自指定列表的人员进入。为了通过门禁，每个人都必须在门口刷身份证以表明自己的身份，然后人脸识别系统将检查他们到底是谁。

3.1-人脸验证

对于人脸验证系统，首先需要构建一个数据库，里面包含了允许进入的人员的编码向量，我们使用 fr_uitls.img_to_encoding(image_path, model)函数来生成编码，它会根据图像来进行模型的前向传播计算人脸图像编码。

我们这里的数据库使用的是一个字典来表示，这个字典将每个人的名字映射到他们面部的 128 维编码上，如下（Note：这里要把 images 目录保存在当前目录下面，否则会报错）。

```
"""构建一个数据库，里面包含了允许进入的人员的编码向量，注意这里要把 images 目录保存在当前目录下面，否则会报错，至于为什么我也没搞清楚"""
database = {}
database["danielle"] = img_to_encoding("images\\danielle.png", FRmodel)
database["younes"] = img_to_encoding("images\\younes.jpg", FRmodel)
database["tian"] = img_to_encoding("images\\tian.jpg", FRmodel)
database["andrew"] = img_to_encoding("images\\andrew.jpg", FRmodel)
database["kian"] = img_to_encoding("images\\kian.jpg", FRmodel)
database["dan"] = img_to_encoding("images\\dan.jpg", FRmodel)
database["sebastiano"] = img_to_encoding("images\\sebastiano.jpg", FRmodel)
database["bertrand"] = img_to_encoding("images\\bertrand.jpg", FRmodel)
database["kevin"] = img_to_encoding("images\\kevin.jpg", FRmodel)
database["felix"] = img_to_encoding("images\\felix.jpg", FRmodel)
database["benoit"] = img_to_encoding("images\\benoit.jpg", FRmodel)
database["arnaud"] = img_to_encoding("images\\arnaud.jpg", FRmodel)
```

人脸验证的思想很简单，即当有人出现在的门前刷他们的身份证的时候，你可以在数据库中根据身份证上的名字查找他们的对应的编码，用它来检查站在门前的人是否与身份证上的名字匹配，代码如下。

```
def verify(image_path, identity, database, model):
    """
    Function that verifies if the person on the "image_path" image is "identity"
    :param image_path:  摄像头的照片
    :param identity:  字符类型，想要验证的人的名字
    :param database:  字典类型，包含了成员的名字信息与对应的编码
    :param model:  在 Keras 的模型的实例
    :return:
                dist:  摄像头的图片与数据库中的图片的编码的差距( use the L2 distance (np.linalg.norm( ))
                door_open:  是否开门
    """
```

```
    ### Start code here ###
    # Step 1: 计算图像的编码
    encoding = img_to_encoding(image_path, model)
    # Step 2: 计算与数据库中保存的编码差距
    dist = np.linalg.norm(encoding - database[identity])
    # Step 3: 判断是否开门
    if dist < 0.7:
        print("It's " + str(identity) + ",welcome home!")
        door_open = True
    else:
        print("It,s not " + str(identity) + ",please go away")
        door_open = False
    ### End code here ###
    return dist, door_open
```

我们来测试一下，现在 younes 在门外，相机已经拍下了照片并存放在了(“images/camera_0.jpg”)，现在我们来验证一下他是否能开门成功。


(younes)

```
# younes 来到门前，摄像头拍摄的照片存入 camera_0.jpg，刷 ID 验证是否是 younes 本人
dist, door_open = verify("images\\camera_0.jpg", "younes", database, FRmodel)
print(dist, door_open)
```

It's younes,welcome home!
0.6710074 True

还有一种情况，比如 Benoit 已经被禁止进入，也从数据库中删除了 Benoit 的信息，但是他偷了 Kian 的身份证并试图通过门禁，我们来看看他能不能进入呢？(“images/camera_2.jpg”)


(Benoit)

```
# Benoit 拿着 kian 的 ID 来，刷 kian 的 ID 卡想进入房子，结果应该是验证失败
dist1, door_open1 = verify("images\\camera_2.jpg", "kian", database, FRmodel)
print(dist1, door_open1)
```

It,s notkian,please go away
0.85800153 False

3.2-人脸识别

对于人脸验证系统有个弊端，那就是如果一个人的身份证不见了，那么他将无法进入房子，要是不用身份证那个多好啊，当然这是可以实现的，也就是这里所说的人脸识别系统，

这样就不用再带身份证了，一个被授权的人只要走到房子前面，前门就会自动为他们打开!
代码思路和实现如下：

```
"""上面是人脸验证：数据库中存入人员的预先 encoding，然后通过摄像头拍照，将图片进行 encoding
后，任何根据身份证的 ID 名来查询数据库中预先按存入的 encoding
    计算编码距离，然后根据阈值来验证门前的这个人是否是本人，还是很简单，
    但是现在有个人的身份证被偷了，或者丢了，进不去家门了，是不是这是一种很糟糕的情况，那
么人脸识别就来帮你解决这件麻烦事,那么人们就不需要带身份证，被授权的人只要站在门前，
    门就会自动打开，如下思路：
    step1：根据 image_path 计算图像的 encoding
    step2：从数据库中找到与目标编码具有最小差距的编码
            ①初始化 min_dist 变量为足够大的数字(比如 100)，它负责找到与输入编码最接近的编码。
            ②遍历数据库的名字 encoding，for (name, db_encoding) in database.items()
                    ·计算目标编码与当前数据库编码之间的 L2 距离
                    ·如果距离小于 min_dist,那么就更新名字与编码到 identity 与 min_dist 中"""
def who_is_it(image_path, database, model):
    encoding = img_to_encoding(image_path, model)
    min_dist = 100
    for (name, dic_enc) in database.items():
        dist = np.linalg.norm(encoding - dic_enc)
        if dist < min_dist:
            min_dist = dist
            identity = name
    if min_dist > 0.7:
        print("Not in the database.")
    else:
        print("it's " + str(identity) +", the distance is " + str(min_dist))
    return min_dist, identity
```

我们也来看一下效果怎么样，younes 和 bertrand 站在前门，相机给他拍了张照片(“images/camera_0.jpg”)和(“images/camera_1.jpg”)。让我们看看 who_it_is()算法是否识别 younes 和 bertrand

 (younes)  (bertrand)

```
min_dist, identity = who_is_it("images\\camera_0.jpg", database, FRmodel)
print(min_dist, identity)
min_dist1, identity1 = who_is_it("images\\camera_1.jpg", database, FRmodel)
print(min_dist1, identity1)
```

```
it's younes, the distance is 0.6710074
0.6710074 younes
it's bertrand, the distance is 0.46740144
```

0.46740144 bertrand

4、总结

①人脸验证解决了更容易的 1:1 匹配问题，人脸识别解决了更难的 1：k 匹配问题。
②Triplet loss 是训练神经网络学习人脸图像编码的一种有效的损失函数。
③相同的编码可用于验证和识别。测量两个图像编码之间的距离可以确定它们是否是同一个人的图片。