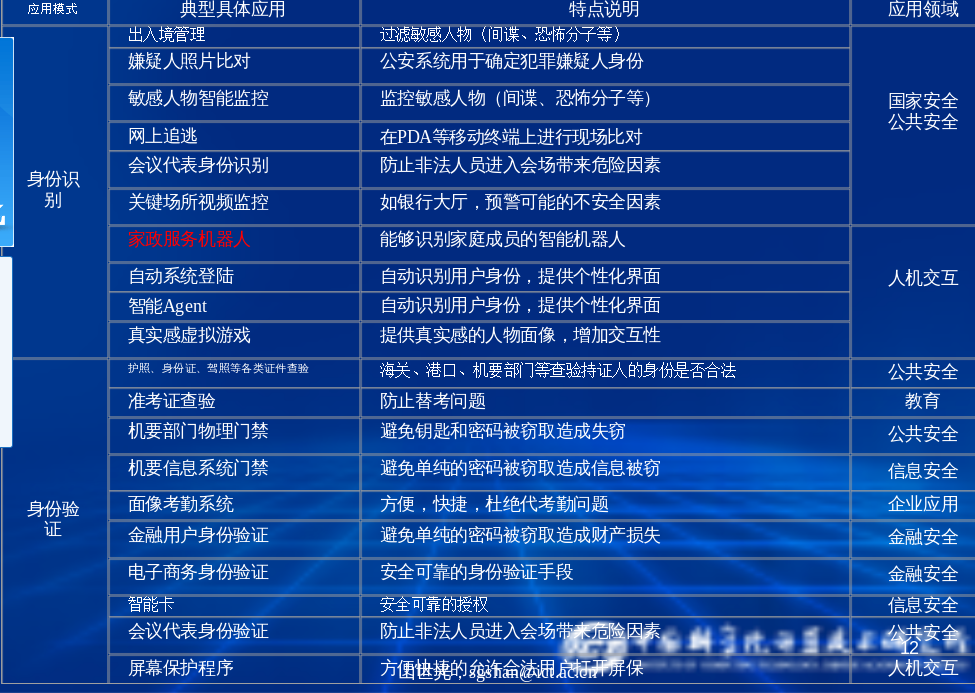
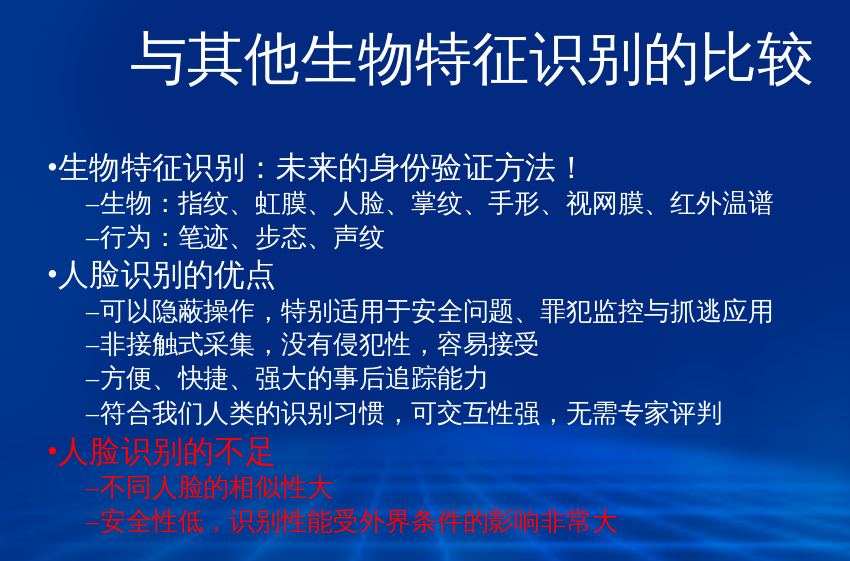
**人脸识别**

# 一、杂记

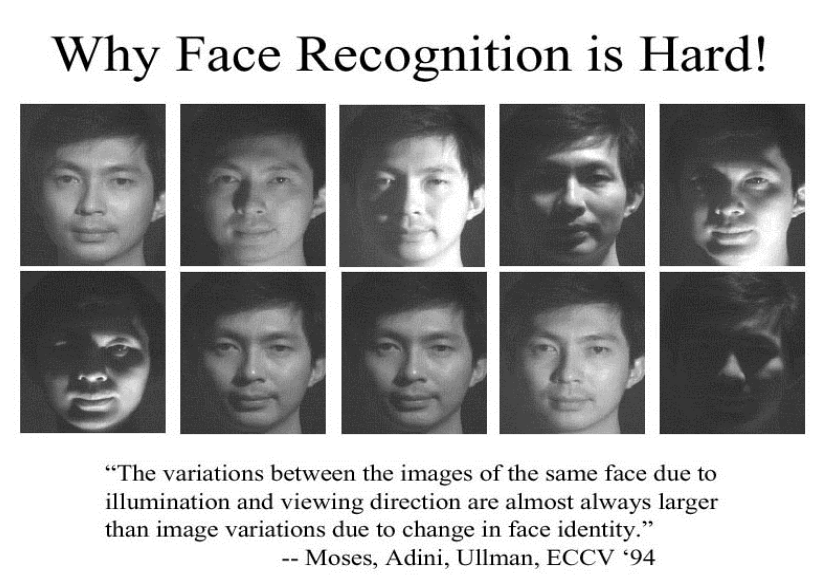
按照任务目的不同，将人脸分析分成了人脸检测，人脸关键点检测，人脸识别，人脸属性识别等任务。

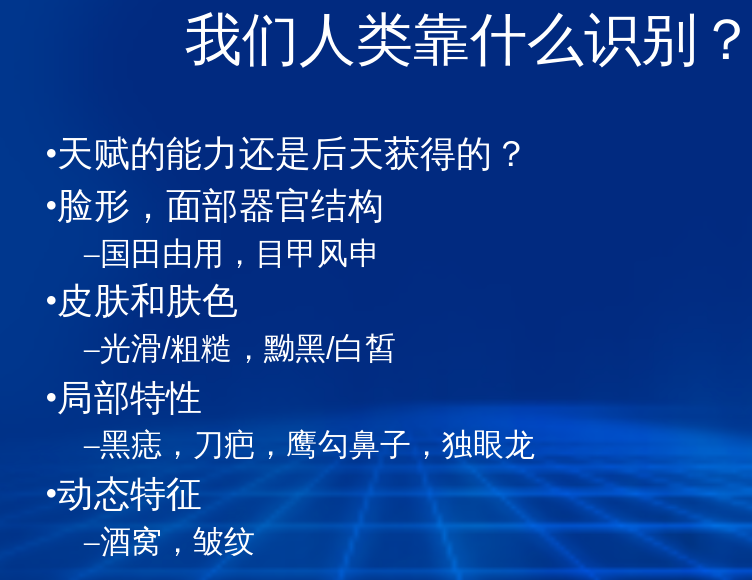
人脸识别应用：

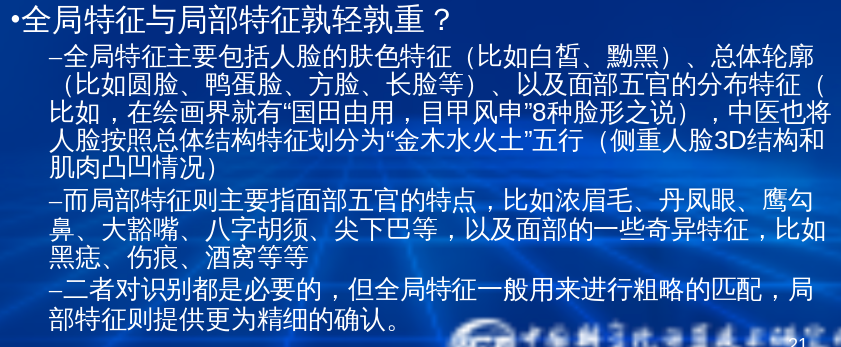






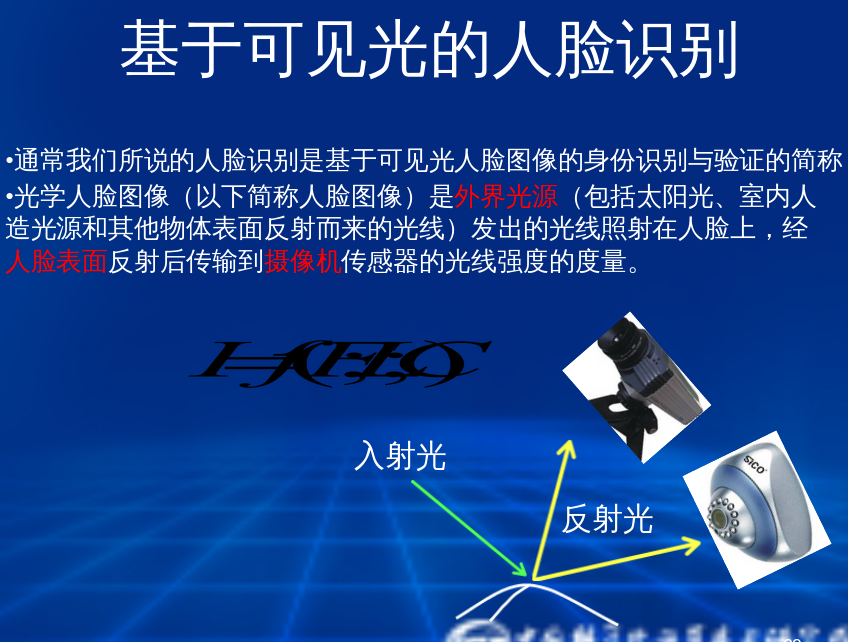


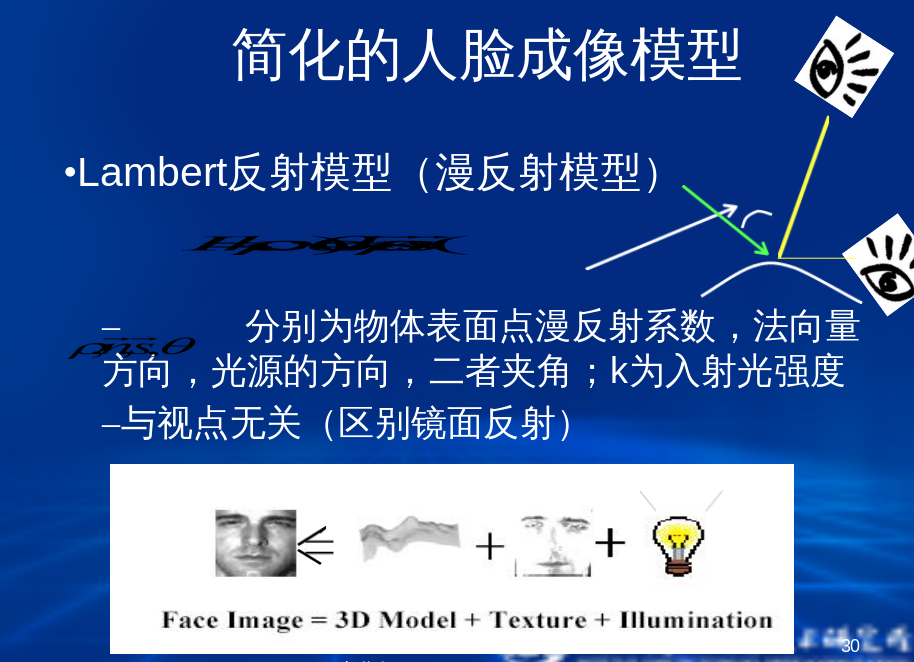


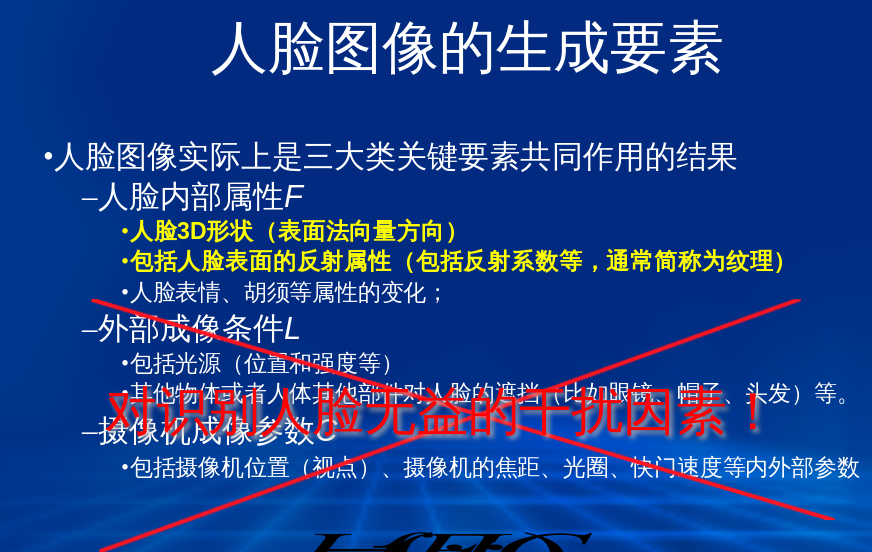


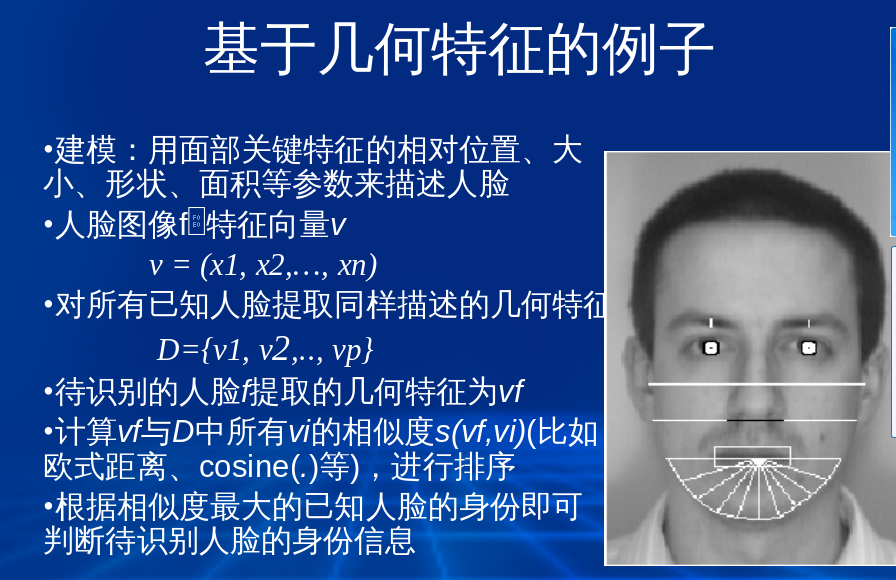












**1、传统的人脸识别方法**

1）主动形状模型（active shape model，ASM）

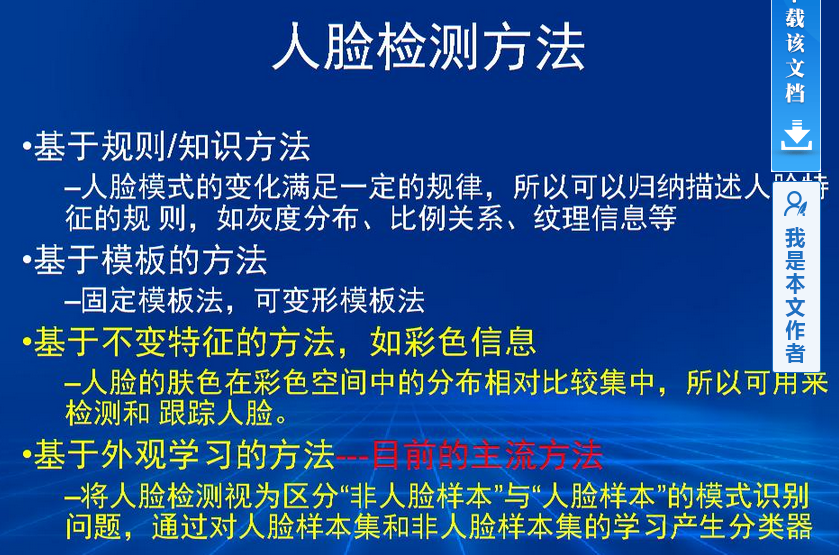
2）主动表观模型（active appearance models，AAM）

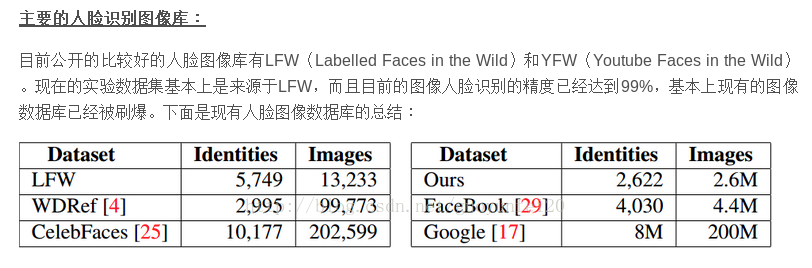
3）基于局部的方法：局部描述子Gabor，局部二值模式（local binary pattern，LBP）

4）基于全局的方法，包括特征脸方法（Eigenface），线性判别分析法（linear discriminant analysis，LDA），局部保持投影算法（locality preserving projection，LPP）

5）3D人脸识别

但是由于受到光照、姿态及表情变化、遮挡、海量数据等影响，其识别精度受到制约。





**2、典型应用**

1）基于卷积神经网络的人脸识别方法

2）深度非线性人脸形状提取方法

3）基于深度学习的人脸姿态鲁棒性建模

4）有约束环境中的全自动人脸识别

5）基于深度学习的视频监控下的人脸识别

6）基于深度学习的低分辨率人脸识别

7）其他基于深度学习的人脸相关信息的识别

# 二、基于深度学习的人脸识别方法

本文综述的人脸识别方法包括以下几个筛选标准：

1、在上表中识别精度超过0.95（超过人类的识别准确度）；

2、公布了方法（部分结果为商业公司提交，方法并未公布，比如Tencent-BestImage）；

3、使用深度学习方法

4、近两年的结果。

本文综述的方法包括：

1）face++（0.9950 ）

2）DeepFace（0.9735 ）

3）FR+FCN（0.9645 ）

4）DeepID（0.9745 ）

5）FaceNet（0.9963 ）

6）baidu的方法（0.9977 ）

7）pose+shape+expression augmentation（0.9807）

8）CNN-3DMM estimation(0.9235 ，准确率没那么高)。

参考 <https://zhuanlan.zhihu.com/p/24816781>

## 1、FaceNet

### 背景

论文地址：FaceNet: [A Unified Embedding for Face Recognition and Clustering](https://arxiv.org/abs/1503.03832)

代码地址：[GitHub（非官方）](https://github.com/cmusatyalab/openface)

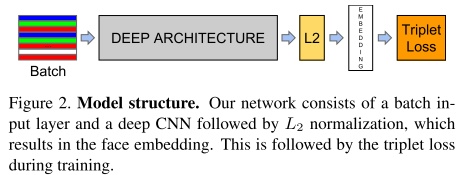
谷歌人脸检测算法，发表于 CVPR 2015，利用相同人脸在不同角度等姿态的照片下有高内聚性，不同人脸有低耦合性，提出使用 cnn + triplet mining 方法，在 LFW 数据集上准确度达到 99.63%，在 youtube 人脸数据集上准确度 95.12%，比以往准确度提升了将近 30%。

近年来，人脸识别技术取得了飞速的进展，但是人脸验证和识别在自然条件中应用仍然存在困难。本文中，作者开发了一个新的人脸识别系统：FaceNet，可以直接将人脸图像映射到欧几里得空间，空间距离的长度代表了人脸图像的相似性。只要该映射空间生成，人脸识别，验证和聚类等任务就可以轻松完成。文章的方法是基于深度卷积神经网络。FaceNet在LFW数据集上，准确率为0.9963，在YouTube Faces DB数据集上，准确率为0.9512。

FaceNet是一个通用的系统，可以用于人脸验证（是否是同一人？），识别（这个人是谁？）和聚类（寻找类似的人？）。FaceNet采用的方法是通过卷积神经网络学习将图像映射到欧几里得空间。空间距离直接和图片相似度相关：同一个人的不同图像在空间距离很小，不同人的图像在空间中有较大的距离。只要该映射确定下来，相关的人脸识别任务就变得很简单。

当前存在的基于深度神经网络的人脸识别模型使用了分类层（classification layer）：中间层为人脸图像的向量映射，然后以分类层作为输出层。这类方法的弊端是不直接和效率低。

与当前方法不同，FaceNet直接使用基于triplets的LMNN（最大边界近邻分类）的loss函数训练神经网络，网络直接输出为128维度的向量空间。我们选取的triplets（三联子）包含两个匹配脸部缩略图和一个非匹配的脸部缩略图，loss函数目标是通过距离边界区分正负类，如图1-1所示。



脸部缩略图为紧密裁剪的脸部区域，没有使用2d，3d对齐以及放大转换等预处理。

本文中，作者探索了两类深度卷积神经网络。第一类为Zeiler&Fergus研究中使用的神经网络，我们在网络后面加了多个1\*1\*d卷积层；第二类为Inception网络。模型结构的末端使用triplet loss来直接分类。triplet loss 的启发是传统loss函数趋向于将有一类特征的人脸图像映射到同一个空间。而triplet loss尝试将一个个体的人脸图像和其它人脸图像分开。

### 方法

通过 CNN 将人脸映射到欧式空间的特征向量上，计算不同图片人脸特征的距离，通过相同个体的人脸的距离，总是小于不同个体的人脸这一先验知识训练网络。

测试时只需要计算人脸特征，然后计算距离使用阈值即可判定两张人脸照片是否属于相同的个体。

### 实现

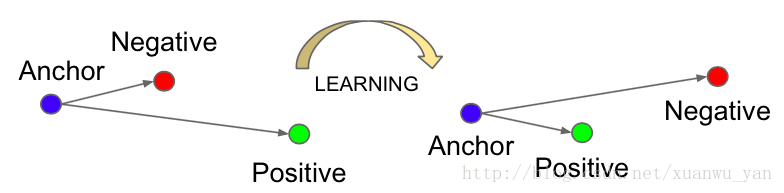
1）对图片使用 CNN 网络提取特征向量。

A）存在特征向量的维度选择问题，维度越小计算越快，但是太小的话很难区分不同图片；维度越大越容易区分不同图片，但是太大训练模型不易收敛，且测试时计算慢，占用空间大。作者实验证明 128 维的特征能够较好的平衡这个问题。

B）CNN 模型的选择，高精度的模型往往参数多，计算量大。移动设备上最好使用体积小，精度略低的模型；服务器上可以使用高精度，高计算量的模型。

2）使用三元损失函数（三联子（triplets）loss）

A）之前的工作有人使用的是二元损失函数，二元损失函数的目标是把相同个体的人脸特征映射到空间中的相同点，而三元损失函数目标是映射到相同的区域，使得类内距离小于类间距离。



选择一张图片作为anchor，Positive是跟anchor同一个人的，Negative是跟anchor不一样的人的，那我们希望通过学习使得anchor与positive的距离近一些，与negative的距离远一些，我这里直接将这样的三张图片的组合称为三元组。

B）假设 x 是输入的图片，f(x)∈Rd 是 CNN 表示后的特征向量，三元函数为

2018-04-03 15-35-58 的屏幕截图

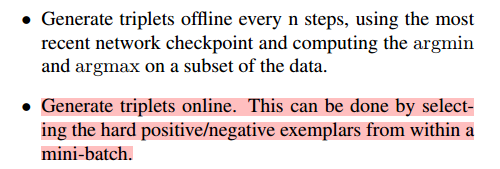
最终损失函数：



其中，ɑ为positive/negtive的边界。

对于整个数据集来说，这样的三元组的数量是非常多的，假设总共有N张图片，每个人有K张图，那么数量级大概是N\*N\*K, 而且这些组合当中有一些是非常容易满足上述条件的，那么对于优化的意义不大，所以应该通过更加合理的方式来选择这样的三元组。

如论文中所说，我们应该选择违背上述条件最严重的组合（有点像SVM吧。），是否违背上述条件是要通过计算图片之间的embedding的欧式距离得到，但是embedding又是不断在优化更新，也就是说违背条件最严重的组合是有可能不断变化的，如果每次更新都重新选择一次，那么对训练效率会有很大影响。文中提到了两种替代的方案:



一是每经过特定的迭代次数，选用最新的checkpoint，在一个训练数据子集上面选择一次三元组。

二是一种在线方法，每训练一次minibatch，选择一次三元组。

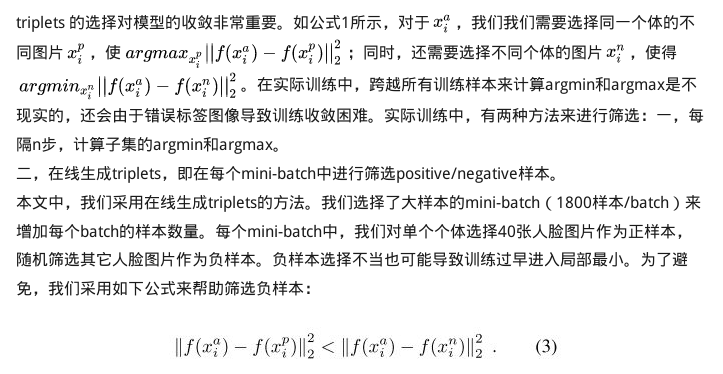
而facenet中选用的就是第二种，这样训练的思路就很清晰了，在一个minibatch中，我们根据当时的embedding，选择一次三元组，在这些三元组上计算triplet-loss, 再对embedding进行更新，不断重复，直到收敛或训练到指定迭代次数。那接下来就看一下facenet中是怎么具体操作的。

C）可以看到上面的公式中需要三个输入人像，如何选择这一个三元组训练呢？为了保证训练收敛速度，我们就选择距离最远的相同人像，和距离最近的不同人像来训练好了。于是作者在每个 mini-batch 中进行上述选择。

D）卷积网络。选择模型是常见的问题，作者针对 ZF/GoogLeNet 做了不同的测试。

E）最终验证。通过计算不同图片的特征向量的距离，使用阈值后得到结果。

### triplets筛选



参考文章

<https://blog.csdn.net/u011918382/article/details/79006782>

### 总结

1）提取特征直接计算距离，比之前的使用 PCA + SVM 更加简单，训练的损失函数直接针对实际误差，end-to-end 方式训练都能提高精度。

2）只要有个人脸的 bounding box 就行。

其他实验结果参考 <https://zhuanlan.zhihu.com/p/24837264>