

第七部分

实战 TensorFlow 人脸识别



扫描二维码

试看/购买《TensorFlow 快速入门与实战》视频课程

第七部分 目录

- 人脸识别问题概述
- 典型人脸相关数据集介绍
- 人脸检测算法介绍
- 人脸识别算法介绍
- 人脸检测工具介绍
- 解析 FaceNet 人脸识别模型
- 实战 FaceNet 人脸识别模型
- 测试与可视化分析

人脸识别问题概述

人脸识别概述

人脸识别，特指利用分析比较**人脸视觉特征信息**进行身份鉴别的计算机技术。

广义的人脸识别实际包括构建人脸识别系统的一系列相关技术，包括人脸图像采集、人脸定位、人脸识别预处理、身份确认以及身份查找等；而狭义的人脸识别特指通过人脸进行身份确认或者身份查找的技术或系统。

人脸识别是一项热门的计算机技术研究领域，它属于**生物特征识别技术**，是对生物体（一般特指人）本身的生物特征来区分生物体个体。生物特征识别技术所研究的生物特征包括**脸、指纹、手掌纹、虹膜、视网膜、声音（语音）、体形、个人习惯**（例如敲击键盘的力度和频率、签字）等，相应的识别技术就有人脸识别、指纹识别、掌纹识别、虹膜识别、视网膜识别、语音识别（用语音识别可以进行身份识别，也可以进行语音内容的识别，只有前者属于生物特征识别技术）、体形识别、键盘敲击识别、签字识别等。

人脸识别的技术困难

虽然人脸识别有很多其他识别无法比拟的优点，但是它本身也存在许多困难。人脸识别被认为是生物特征识别领域甚至人工智能领域最困难的研究课题之一。人脸识别的困难主要是人脸作为生物特征的特点所带来的。人脸在视觉上的特点是：

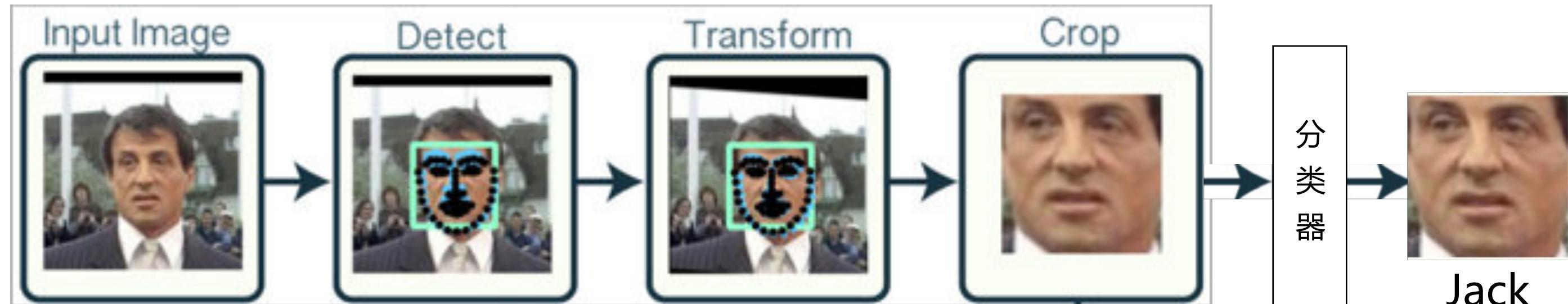
1. 不同个体之间的区别不大，**所有的人脸的结构都相似**，甚至人脸器官的结构外形都很相似。这样的特点对于利用人脸进行定位是有利的，但是对于利用人脸区分人类个体是不利的。
2. **人脸的外形很不稳定**，人可以通过脸部的变化产生很多表情，而在不同观察角度，人脸的视觉图像也相差很大，另外，人脸识别还受**光照条件**（例如白天和夜晚，室内和室外等）、人脸的很多**遮盖物**（例如口罩、墨镜、头发、胡须等）、**年龄**、**拍摄的姿态角度**等多方面因素的影响。

人脸识别典型流程

人脸识别的经典流程分为三个步骤：

1) 人脸检测 ; 2) 人脸对齐 ; 3) 人脸特征表示。

基于传统机器学习的人脸识别一般分为高维人工特征提取（例如：LBP, Gabor等）和降维两个步骤。在深度学习流行之后，我们可以从原始图像空间直接学习判别性的人脸表示，实现端到端的人脸识别模型。



深度学习“引爆”人脸识别

过去几年，深度学习正在彻底改变人脸识别领域。由于GPU的计算效率不断提高，谷歌的研究人员在 CVPR (Computer Vision and Pattern Recognition) 2015 上发表了一篇开创性的论文：FaceNet。FaceNet 是一个解决人脸识别和人脸聚类问题的全新深度神经网络架构，其在 LFW (Labeled Faces in the Wild) 人脸识别数据集上十折平均精度达到99.63%。同时，它为使用深度学习创建下一代人脸识别系统打下了坚实基础。



Schroff, F., Kalenichenko, D. and Philbin, J., 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).

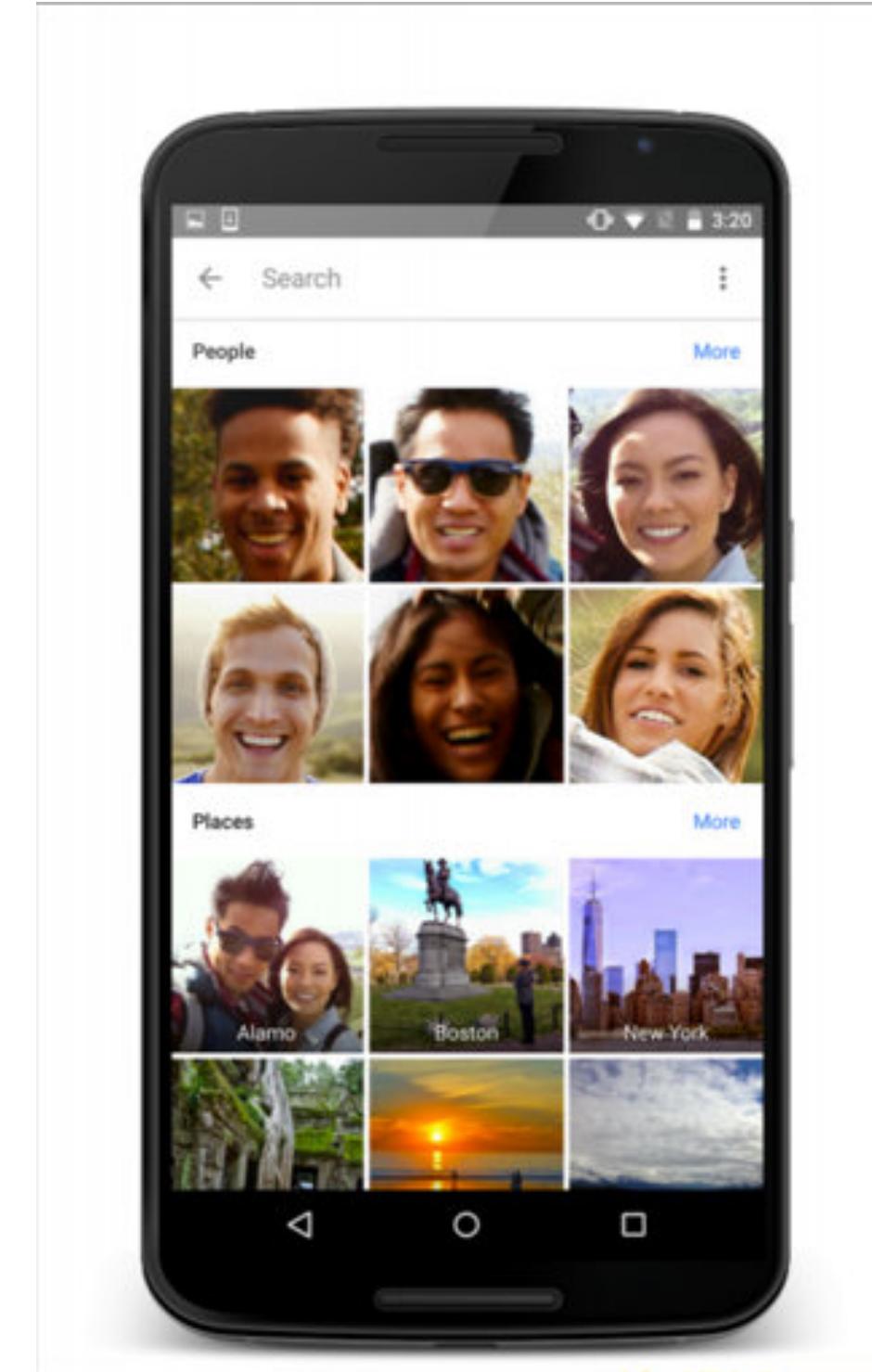
人脸识别应用 – 安防



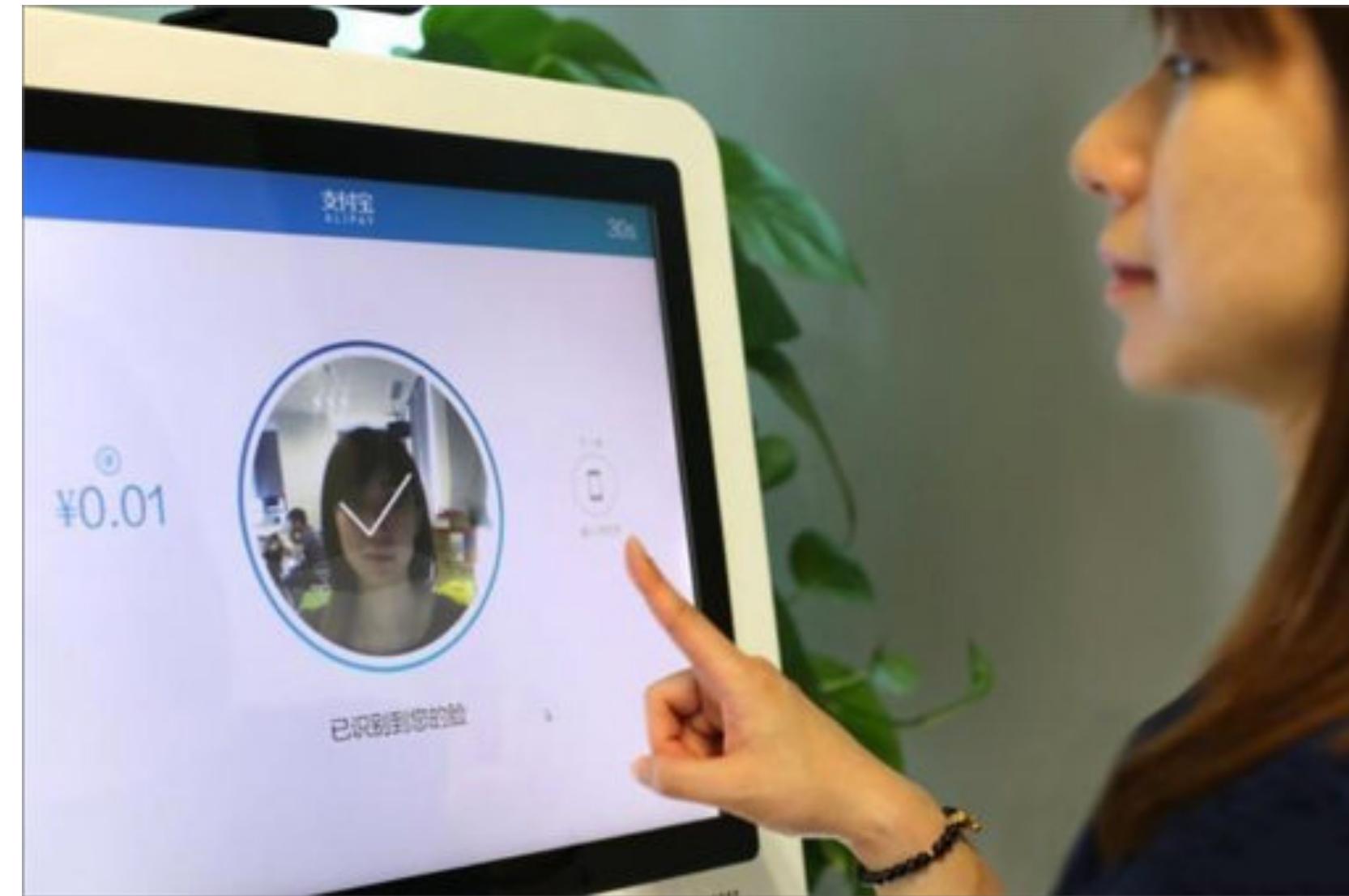
人脸识别应用 – 安检



人脸识别应用 – 个人相册管理



人脸识别应用 – 支付



人脸识别应用 – KYC



应用模式	典型具体应用	特点说明	
身份识别	出入境管理	过滤敏感人物（间谍、恐怖分子等）	国家安全 公共安全
	嫌疑人照片比对	公安系统用于确定犯罪嫌疑人身份	
	敏感人物智能监控	监控敏感人物（间谍、恐怖分子等）	
	网上追逃	在PDA等移动终端上进行现场比对	
	会议代表身份识别	防止非法人员进入会场带来危险因素	
	关键场所视频监控	如银行大厅，预警可能的不安全因素	
	家政服务机器人	能够识别家庭成员的智能机器人	
	自动系统登陆	自动识别用户身份，提供个性化界面	
	智能Agent	自动识别用户身份，提供个性化界面	
	真实感虚拟游戏	提供真实感的人物面像，增加交互性	
身份验证	护照、身份证件、驾照等各类证件查验	海关、港口、机要部门等查验持证人的身份是否合法	公共安全
	准考证查验	防止替考问题	教育
	机要部门物理门禁	避免钥匙和密码被窃取造成失窃	公共安全
	机要信息系统门禁	避免单纯的密码被窃取造成信息被窃	信息安全
	面像考勤系统	方便，快捷，杜绝代考勤问题	企业应用
	金融用户身份验证	避免单纯的密码被窃取造成财产损失	金融安全
	电子商务身份验证	安全可靠的身份验证手段	金融安全
	智能卡	安全可靠的授权	信息安全
	会议代表身份验证	防止非法人员进入会场带来危险因素	公共安全
	屏幕保护程序	方便快捷的允许合法用户打开屏保	人机交互

典型人脸相关数据集介绍

LFW (Labeled Face in the Wild)

2007年以来，LFW 数据库成为事实上真实条件下的人脸识别问题的测试基准。LFW数据集包括来源于因特网的 5749 人的 13233 张人脸图像，其中有 1680 人有两张或以上的图像。



LFW (Labeled Face in the Wild)

LFW 的标准测试协议包括 6000 对人脸的十折确认任务，每折包括 300 对正例和300 对反例，采用十折平均精度作为性能评价指标。自从LFW发布以来，性能被不断刷新。

2013年之前，主要技术路线为人造或基于学习的局部描述子+测度学习。

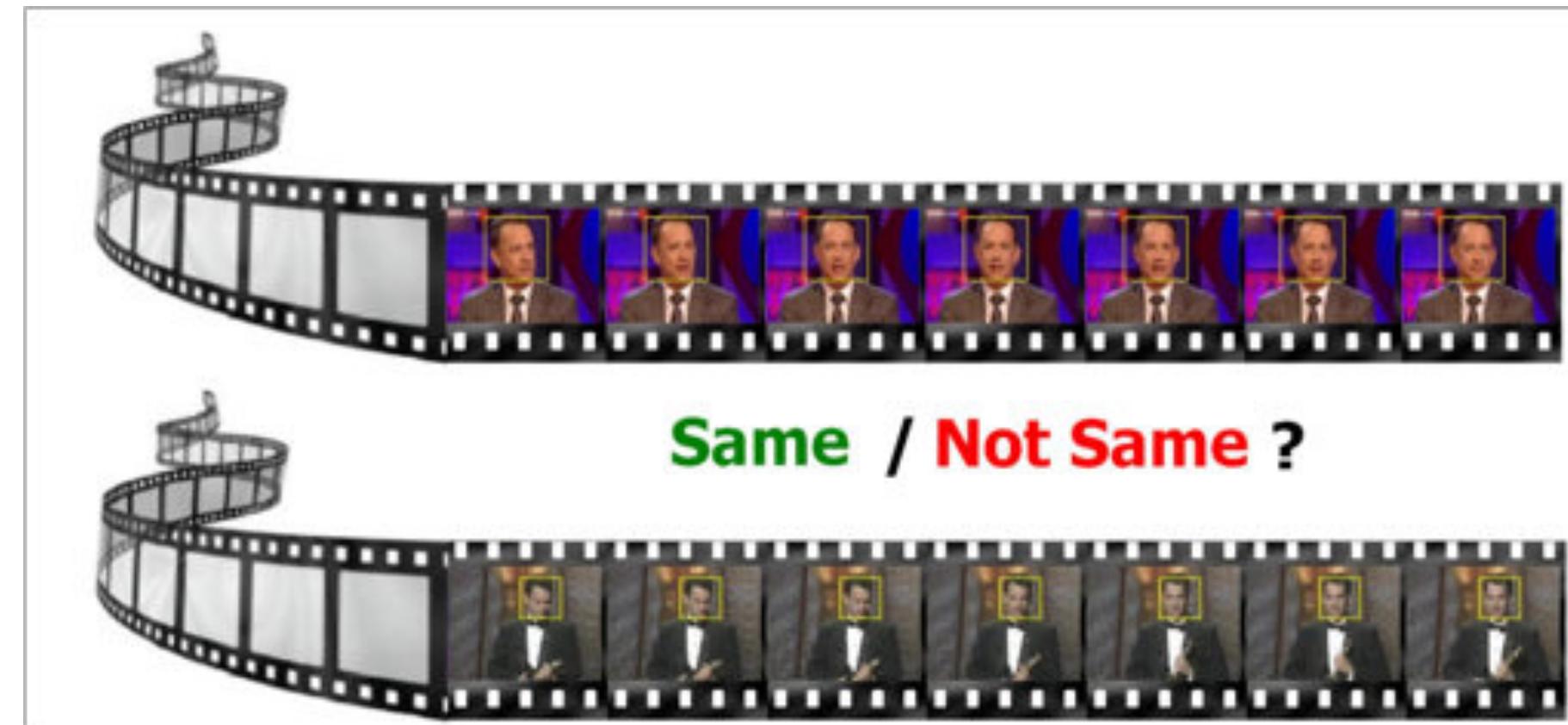
2014年之后，主要技术路线为深度学习。

2014年以来，深度学习+大数据（海量的有标注人脸数据）成为人脸识别领域的主流技术路线，其中两个重要的趋势为：

- 网络变大变深（ VGGFace16层，FaceNet22层）；
- 数据量不断增大（ DeepFace 400万，FaceNet2亿），大数据成为提升人脸识别性能的关键。

YouTube Faces DB

YouTube Faces 是一个面部视频数据库，旨在研究视频中非约束环境下的人脸识别问题。该数据集包含3,425个视频，1,595个不同的人。所有视频都是从YouTube下载的。每个主题平均有2.15个视频。剪辑最短的视频为48帧，最长的为6,070帧，视频平均长度为181.3帧。



CASIA-WebFace

CASIA-WebFace 数据库包含10k+人和约500K张图片，用于非约束环境下人脸识别的科学研究。数据库中的面部图像由中国科学院（CASIA）自动化研究所从互联网上爬取。官方允许该数据库进行教育或非商业用途的免费使用。但由于数据库中的图像可能受版权保护，官方没有在他们的网站上公开发布。如果是研究使用，可以在官方网站上申请访问权限。

NAME and TITLE	
JOB TITLE	
NATIONALITY	
SIGNATURE and DATE	
ORGANIZATION	
DEPARTMENT / LABORATORY	
ADDRESS	
EMAIL	
TELEPHONE	

人脸识别数据集

数据库	描述	用途
LFW	5k+人脸，超过10K张图片	标准的人脸识别数据集
YouTube Faces DB	1,595人 3,425段视频	非限制场景、视频
CASIA-WebFace	10k+人，约500K张图片	非限制场景
FaceScrub	530人，约100k张图片	非限制场景
MultiPIE	337人的不同姿态、表情、光照的人脸图像，共750k+人脸图像	限制场景人脸识别
MegaFace	690k不同的人的1000k人脸图像	新的人脸识别评测集合
CAS-PEAL	1040人的30k+张人脸图像，主要包含姿态、表情、光照变化	限制场景下人脸识别
Pubfig	200人的58k+人脸图像	非限制场景下的人脸识别

FDDB: Face Detection Data Set and Benchmark

人脸检测数据集和基准测试（ FDDB ）是一个面部区域数据集，用于研究非约束环境下人脸检测问题。 FDDB 从 LFW 数据集选取了 5171 张人脸和 2845 张图片，由马萨诸塞大学（ University of Massachusetts , UMASS ）维护。



WIDER FACE: A Face Detection Benchmark

WIDER FACE 数据集是由汤晓鸥团队发布的人脸检测基准数据集。其从公开数据集 WIDER 中选取了 32,203 个图像并标记了 393,703 个面部，其比例、姿势和遮挡度具有高度可变性。数据集共分为61个类。每一类别的训练、验证和测试集比例都是 4:1:5。基准测试的评估指标与PASCAL VOC数据集采用的相同。



Large-scale CelebFaces Attributes (CelebA) Dataset

CelebFaces Attributes Dataset (CelebA) 是一个大型人脸属性数据集，拥有超过 10,177 个名人的 202,599 张面部图像，每张图像都有 5 个特征点标注和 40 个属性注释。同时，此数据集中的图像还覆盖了巨大的姿势变化和杂乱背景。



人脸检测数据集

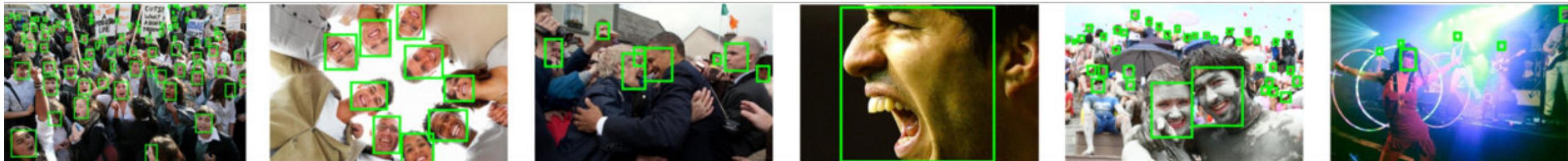
数据库	描述	用途
FDDB: Face Detection Data Set and Benchmark	2845图片，5171人脸	矩形/椭圆框
WIDER FACE	32303 图片，393703人脸	矩形框
Large-scale CelebFaces Attributes (CelebA)	10177 人，202599 人脸图像	人脸属性识别，人脸检测
Caltech10k Web Faces	10524 人脸	人脸点检测

人脸检测算法介绍

人脸检测算法简介

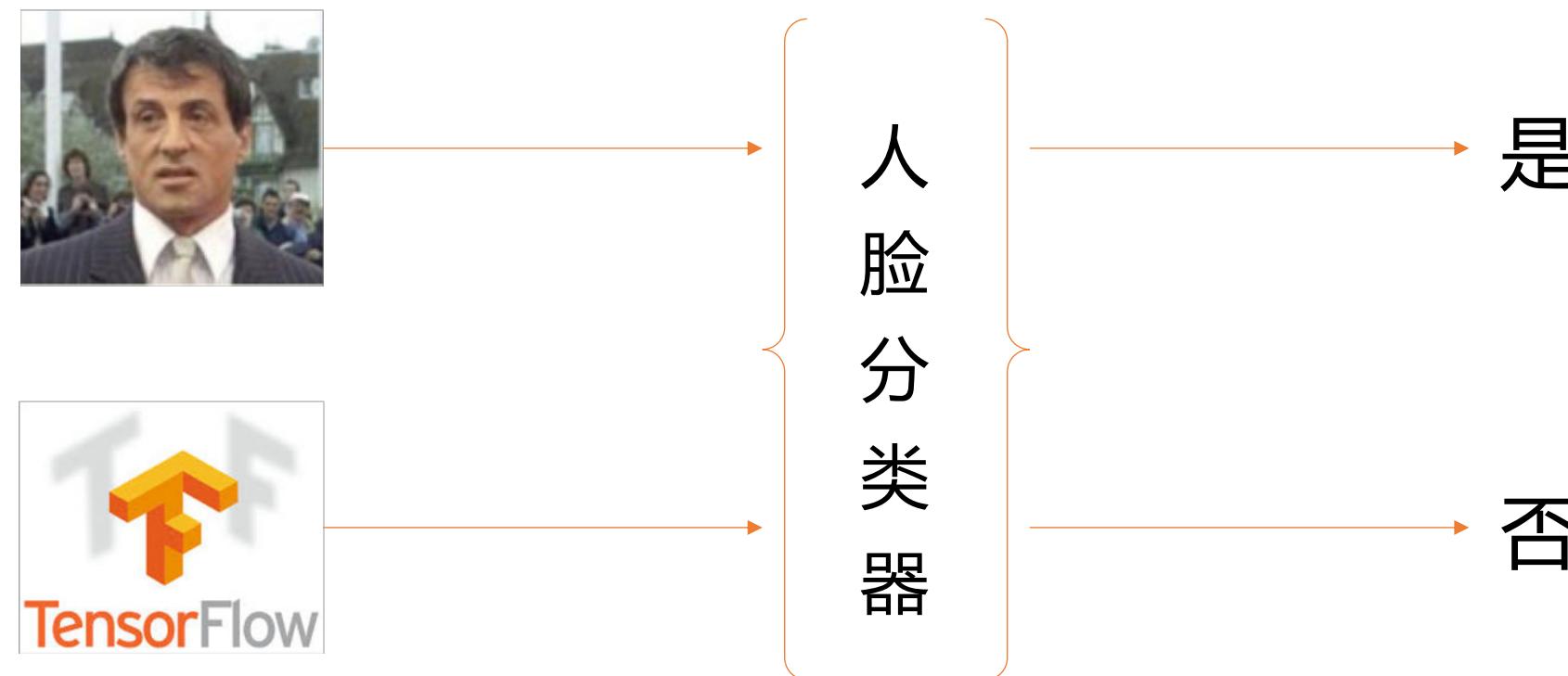
自动人脸检测技术是所有人脸影像分析衍生应用的**基础**，这些扩展应用细分有人脸识别、人脸验证、人脸跟踪、人脸属性识别，人脸行为分析、个人相册管理、机器人人机交互、社交平台的应用等。

人脸检测的目标是找出图像中所有的人脸对应的位置，算法的输出是人脸外接矩形在图像中的坐标，可能还包括姿态如倾斜角度等信息。虽然人脸的结构是确定的，由眉毛、眼睛、鼻子和嘴等部位组成，近似是一个刚体，但由于姿态和表情的变化，不同人的外观差异，光照，遮挡的影响，准确的检测处于各种条件下的人脸是一件相对困难的事情。



经典人脸检测算法流程

用大量的人脸和非人脸样本图像进行训练，得到一个解决二分类问题的分类器，也称为人脸检测模板。这个分类器接受固定大小的输入图片，判断这个输入图片是否为人脸，即解决是和否的问题。人脸二分类器的原理如下图所示：



人脸检测 – 研究进展

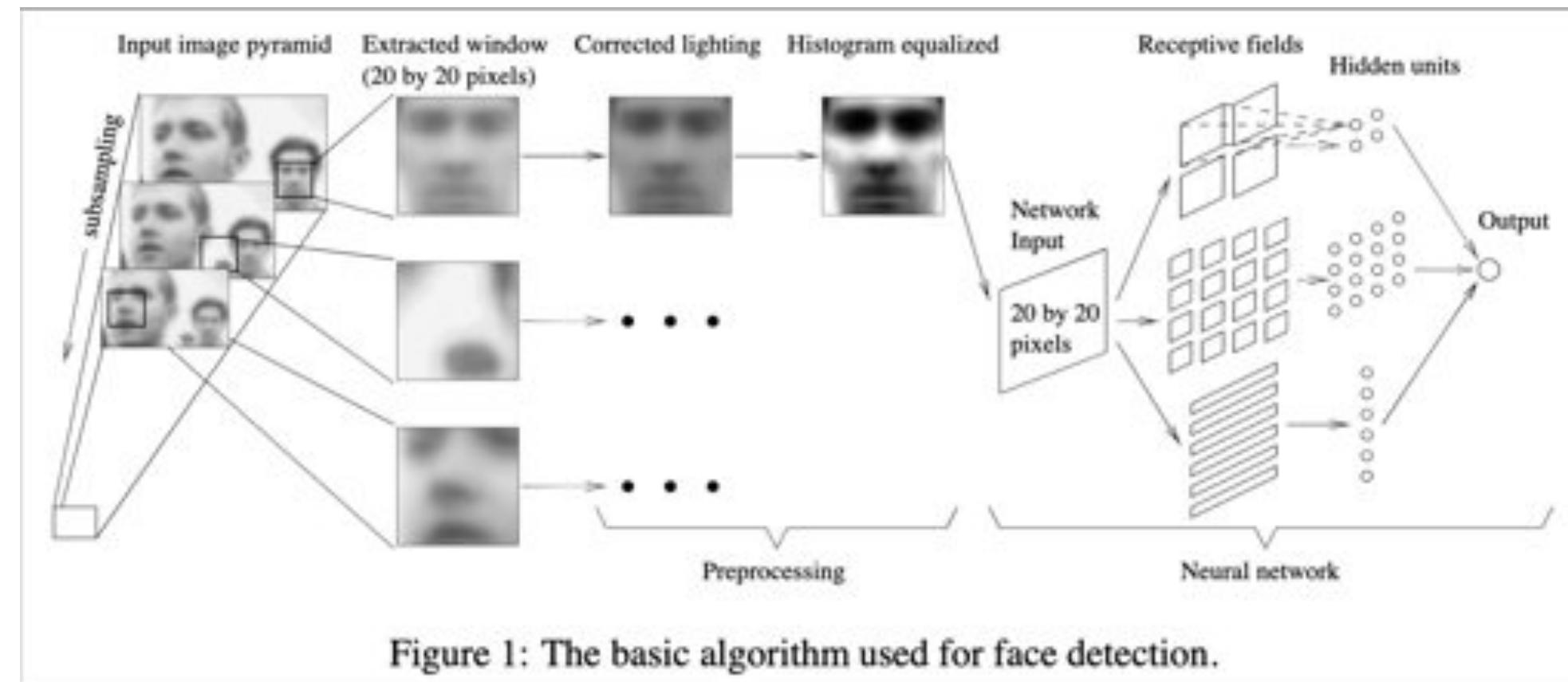
人脸检测算法研究分为3个发展阶段：

1. 基于模版匹配的算法
2. 基于 AdaBoost 的框架
3. 基于深度学习的算法

基于模板匹配的人脸检测算法

早期的人脸检测算法使用了模板匹配技术，即用一个人脸模板图像与被检测图像中的各个**位置进行匹配**，确定这个位置处是否有人脸，即针对图像中某个区域进行人脸-非人脸二分类的判别。

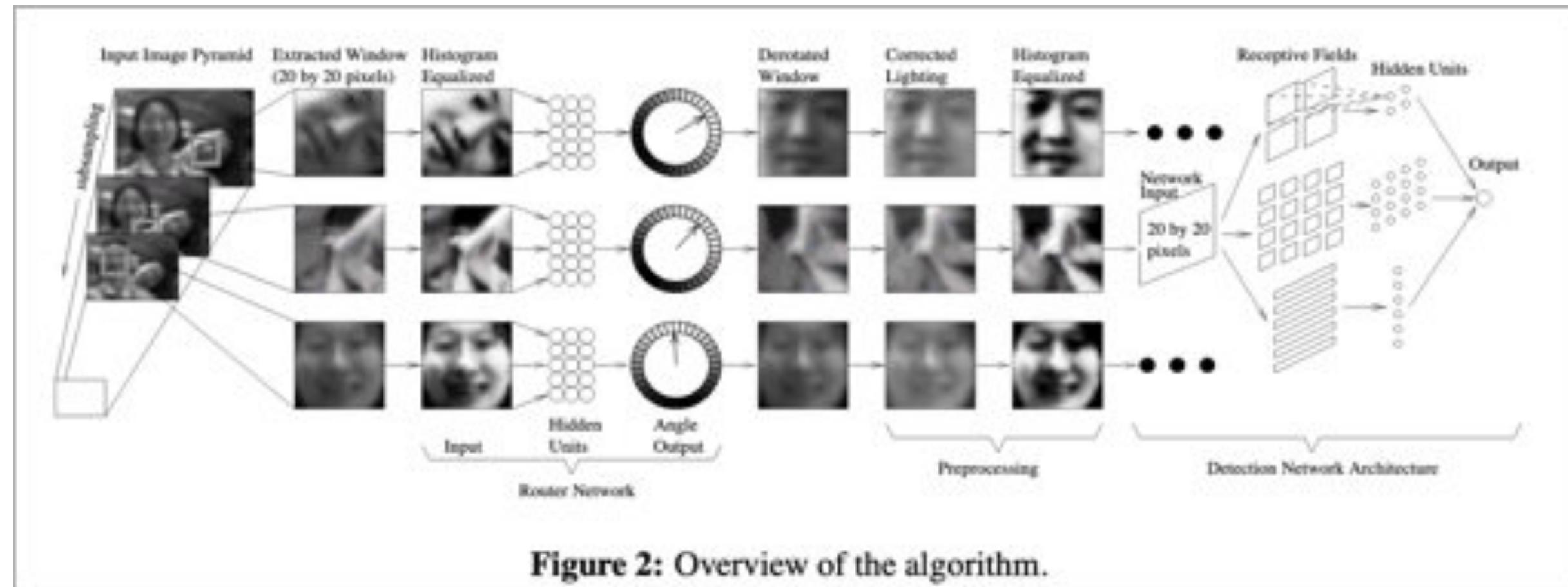
早期有代表性的成果是**Rowley**等人提出的方法[1]。他们用神经网络进行人脸检测，用 20×20 的人脸和非人脸图像训练了一个多层感知器模型。论文[1]中的方法用于解决近似正面的人脸检测问题，原理如下图所示：



[1] Henry A Rowley, Shumeet Baluja, Takeo Kanade. Neural network-based face detection. 1998, IEEE Transactions on Pattern Analysis and Machine Intelligence.

基于模板匹配的人脸检测算法

Rowley等人提出的方法[2] 解决了多角度人脸检测问题，整个系统由两个神经网络构成，第一个网络用于估计人脸的角度，第二个用于判断是否为人脸。角度估计器输出一个旋转角度，然后用整个角度对检测窗进行旋转，然后用第二个网络对旋转后的图像进行判断，确定是否为人脸。系统结构如下图所示：



基于AdaBoost框架的人脸检测算法

Boost 算法是基于 PAC (Probably Approximately Correct) 学习理论而建立的一套集成学习 (Ensemble Learning) 算法。俗话说 “三个臭皮匠，顶个诸葛亮” 。 Boost 的核心思想便是利用多个简单的弱分类器，构建出高准确率的强分类器。

在2001年，**Viola和Jones**设计了一种人脸检测框架 (VJ框架) [3]。它使用简单的 Haar-like 特征和级联的 AdaBoost 分类器构造检测器，检测速度较之前的方法有2个数量级的提高，并且保持了很好的精度。

在深度学习出现以前工业界的方案都是基于VJ算法。但VJ算法仍存在一些问题：

1. Haar-like特征是一种相对简单的特征，其**稳定性较低**；
2. 弱分类器采用简单的决策树，**容易过拟合**。因此，该算法对于解决正面的人脸效果好，对于人脸的遮挡，姿态，表情等特殊且复杂的情况效果一般；
3. 基于VJ-cascade的分类器设计，进入下一个stage后，之前的信息都丢了。分类器评价一个样本不会基于它在之前stage的表现，**鲁棒性差**。

基于深度学习的人脸检测算法

CNN 在图像分类问题上取得成功之后很快被用于人脸检测问题，在精度上大幅度超越之前的AdaBoost框架。在此之前，滑动窗口+卷积对窗口图像进行分类的计算量巨大，无法做到实时检测。当前，已经有一些快速、高效的基于深度学习的算法，我们介绍重点介绍 **Cascade CNN**[4] 和 **MTCNN**[5]。

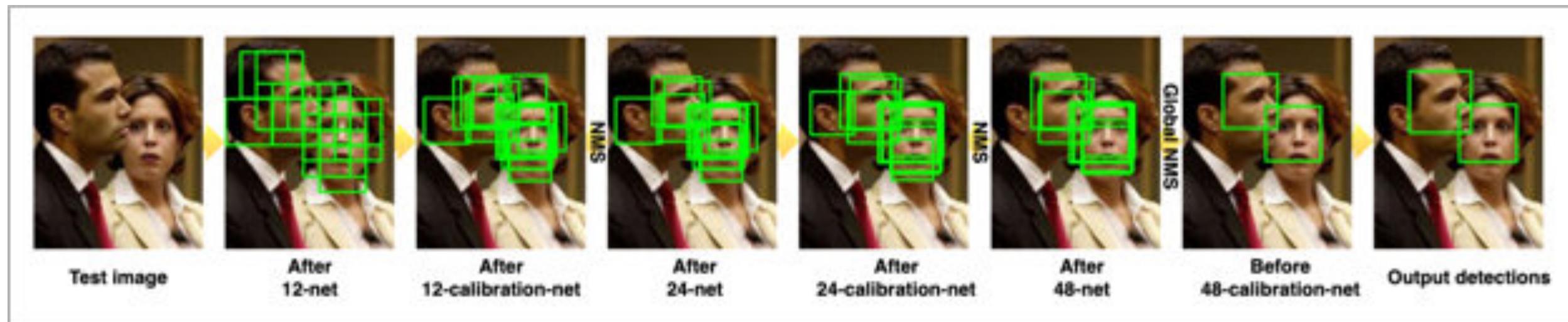
Cascade CNN可以认为是传统技术和深度网络相结合的一个代表，和VJ人脸检测器一样，其包含了多个分类器，这些分类器采用级联结构进行组织。不同的地方在于，Cascade CNN采用卷积网络作为每一级的分类器。

[4] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, Gang Hua. A convolutional neural network cascade for face detection. 2015, computer vision and pattern recognition

[5] Kaipeng Zhan, Zhanpeng Zhang, Zhifeng L, Yu Qiao. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. 2016, IEEE Signal Processing Letters.

Cascade CNN

Cascade CNN 检测流程：构建多尺度的人脸图像金字塔，**12-net**密集的扫描整幅图像（不同的尺寸），快速的剔除掉超过90%的检测窗口；剩下窗口送入 **12-calibration-net** 调整尺寸和位置，使其更靠近潜在的人脸图像附近。接着采用非极大值抑制（NMS）合并高度重叠的检测窗口，保留下来的候选检测窗口将会被归一化到 24x24 作为 **24-net** 的输入，进一步剔除掉剩下来的近90%的检测窗口。接着通过 **24-calibration-net** 矫正检测窗口，并应用NMS进一步合并减少检测窗口的数量。将通过之前所有层级的检测窗口对应的图像区域归一化到48x48送入 **48-net** 进行分类得到进一步过滤的人脸候选窗口。然后利用 NMS进行窗口合并，送入 **48-calibration-net** 矫正检测窗口作为最后的输出。

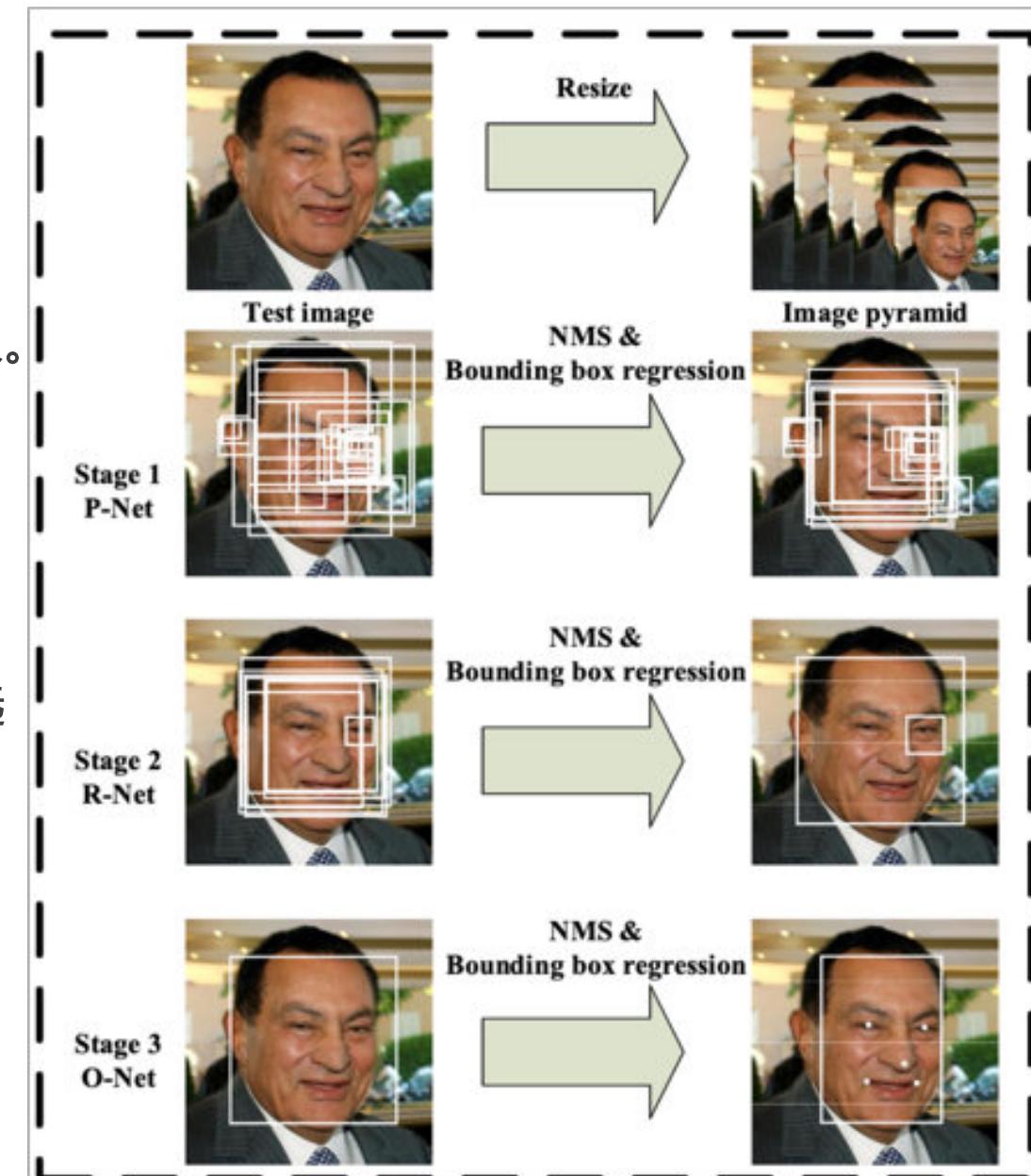


MTCNN

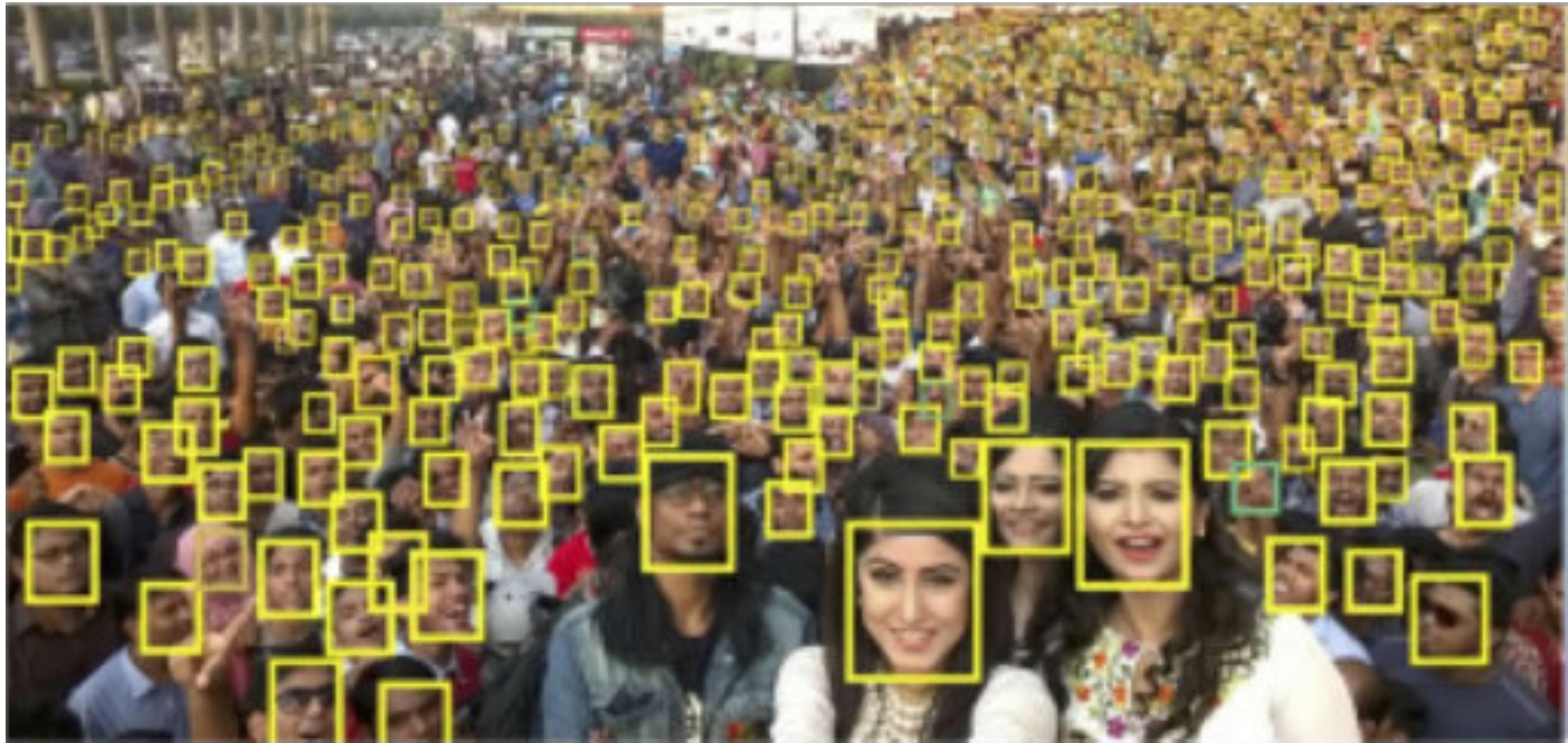
MTCNN的整体设计思路很好，它使用多任务方法将人脸检测和人脸对齐集成到了一个框架中实现，分三个流程: **P-Net, R-Net, O-Net**。

- 0) 首先按不同比例缩放照片，形成图片的特征金字塔作为 P-Net 输入。
- 1) P-Net 主要获得了人脸区域的候选窗口和边界框的回归向量。并用该边界框做回归，对候选窗口进行校准，然后通过NMS来合并高度重叠的候选框。
- 2) 然后将候选框输入 R-Net 网络训练，利用边界框的回归值微调候选窗体，再利用 NMS 去除重叠窗体。
- 3) O-Net 功能与 R-Net 作用类似，只是在去除重叠候选窗口的同时，显示五个人脸关键点定位。

MTCNN 同Cascade CNN一样也是基于cascade的框架，但是降低了模型整体复杂度，使其更好落地到工业界应用。



How many people here?

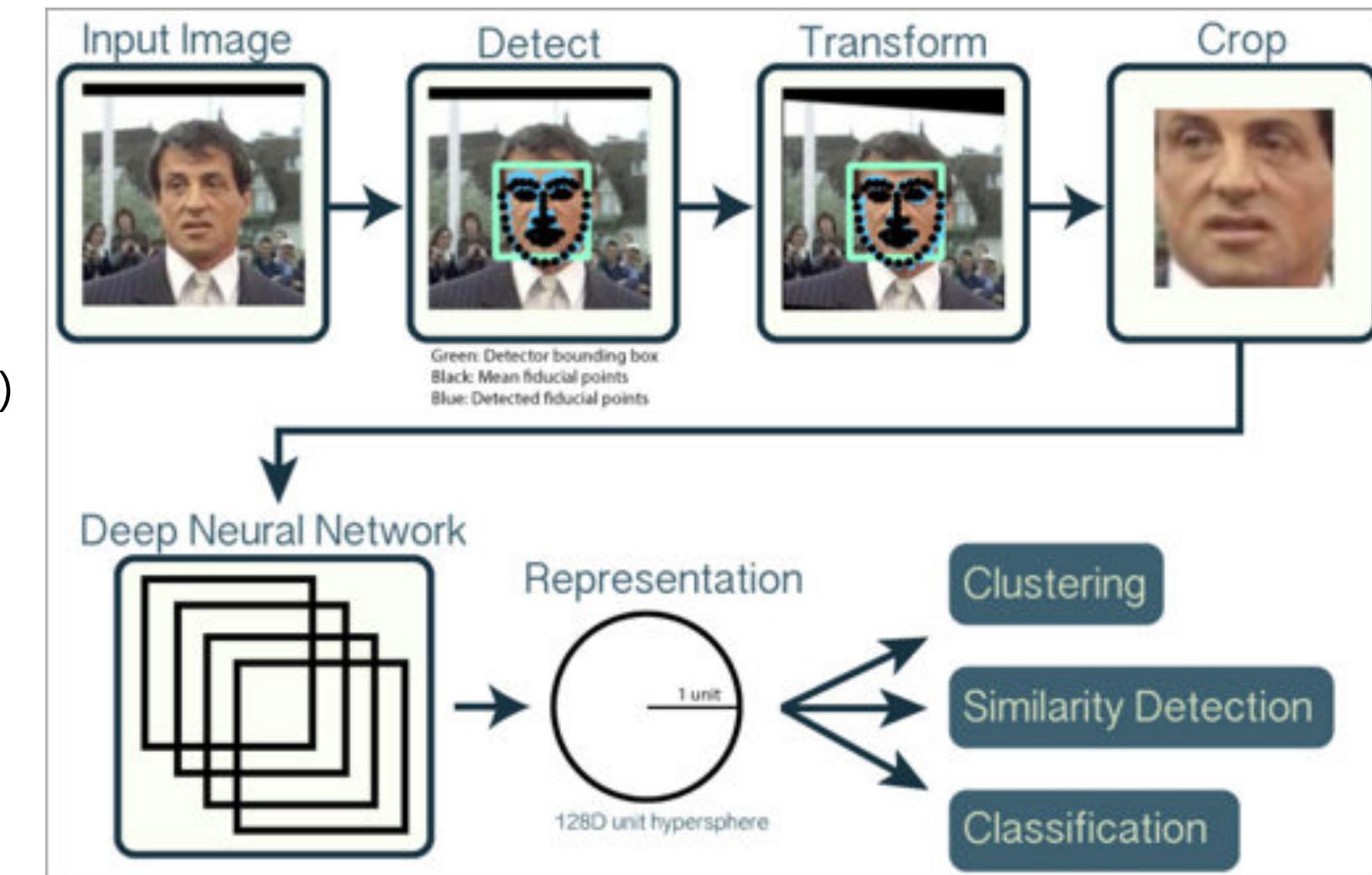


人脸识别算法介绍

人脸识别算法流程

人脸识别算法主要分为三个流程：

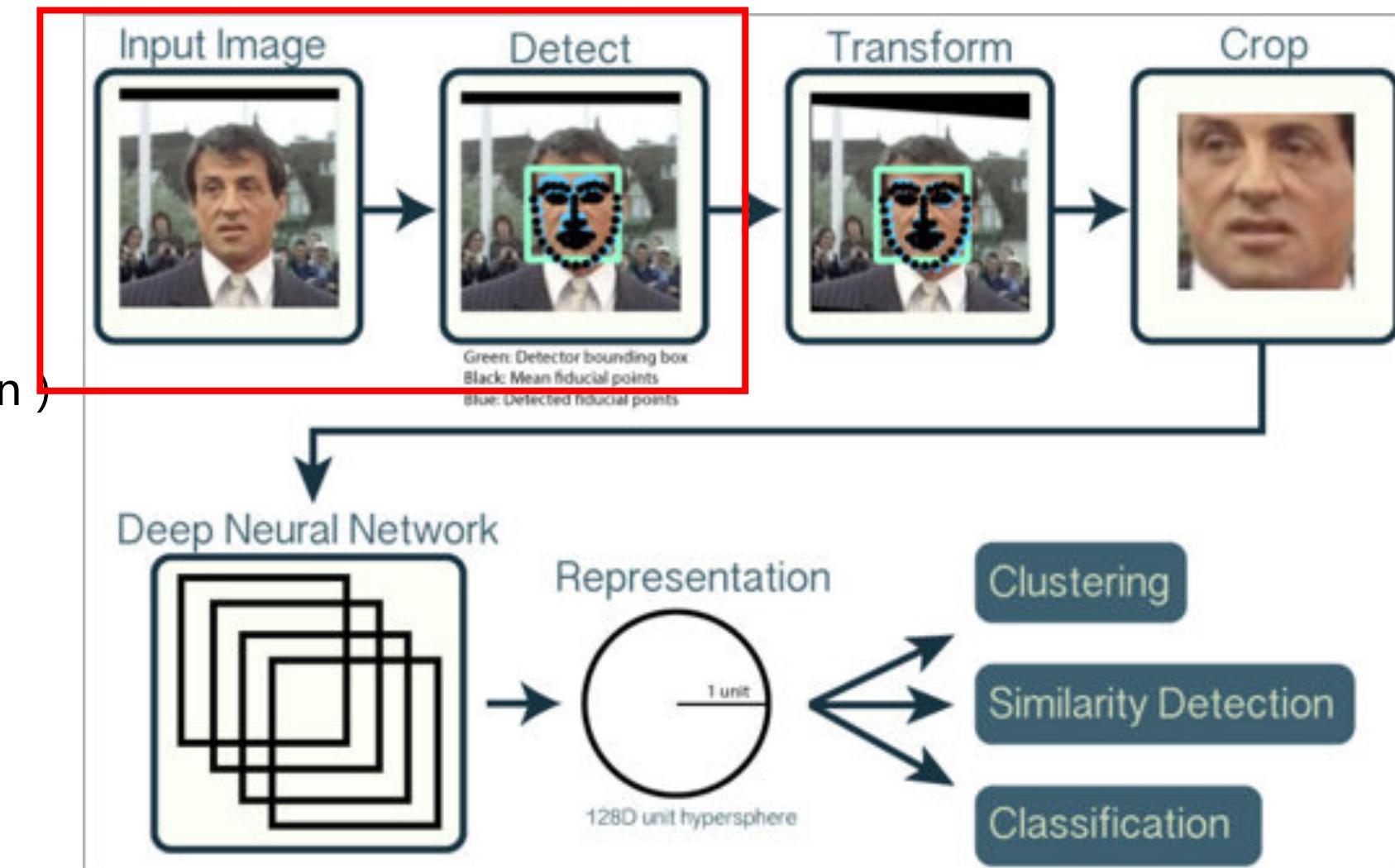
1. 人脸检测 (Face Detection)
2. 人脸对齐 (Face Alignment)
3. 人脸特征表征 (Feature Representation)



人脸识别算法流程

人脸识别算法主要分为三个流程：

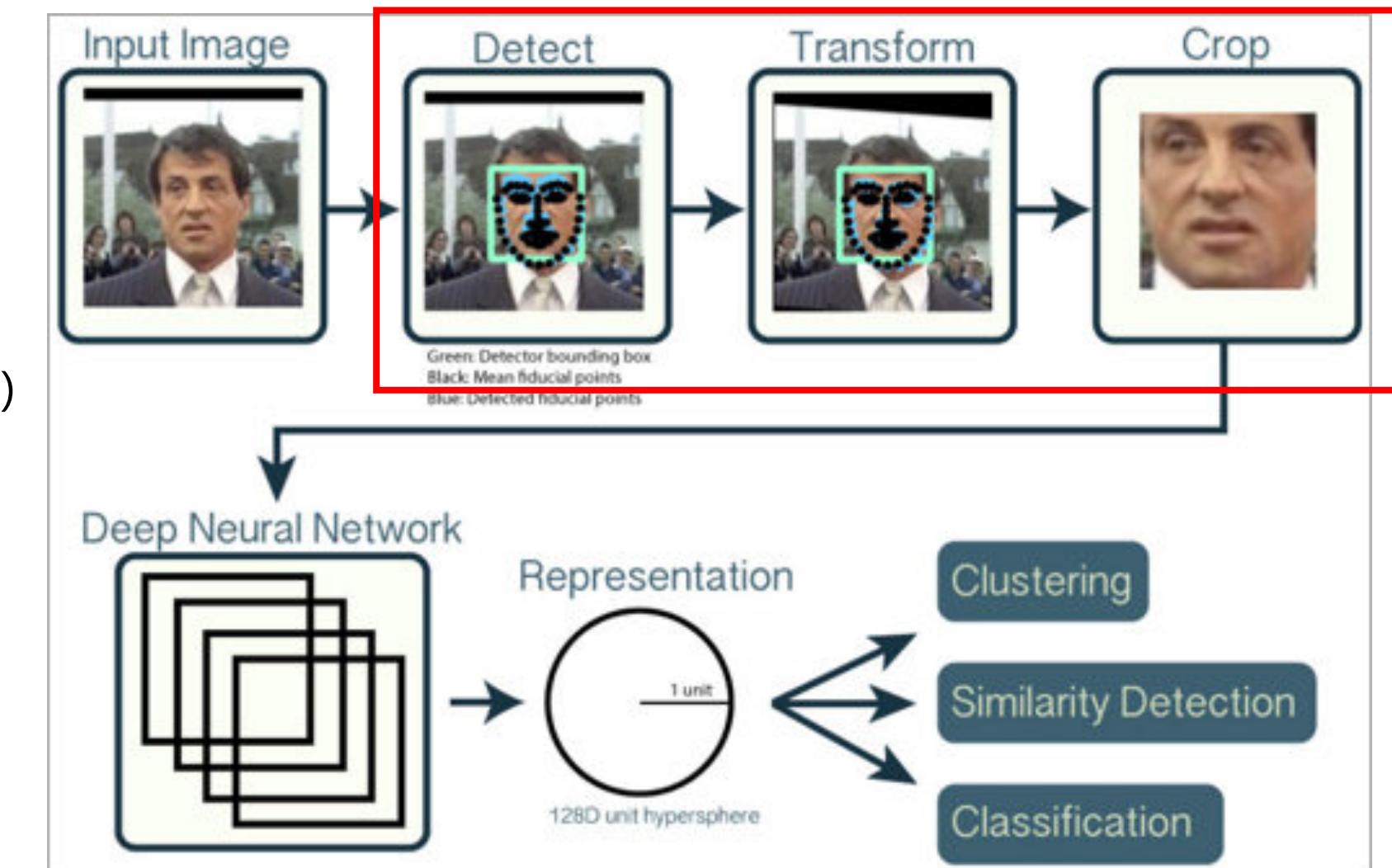
1. 人脸检测 (Face Detection)
2. 人脸对齐 (Face Alignment)
3. 人脸特征表征 (Feature Representation)



人脸识别算法流程

人脸识别算法主要分为三个流程：

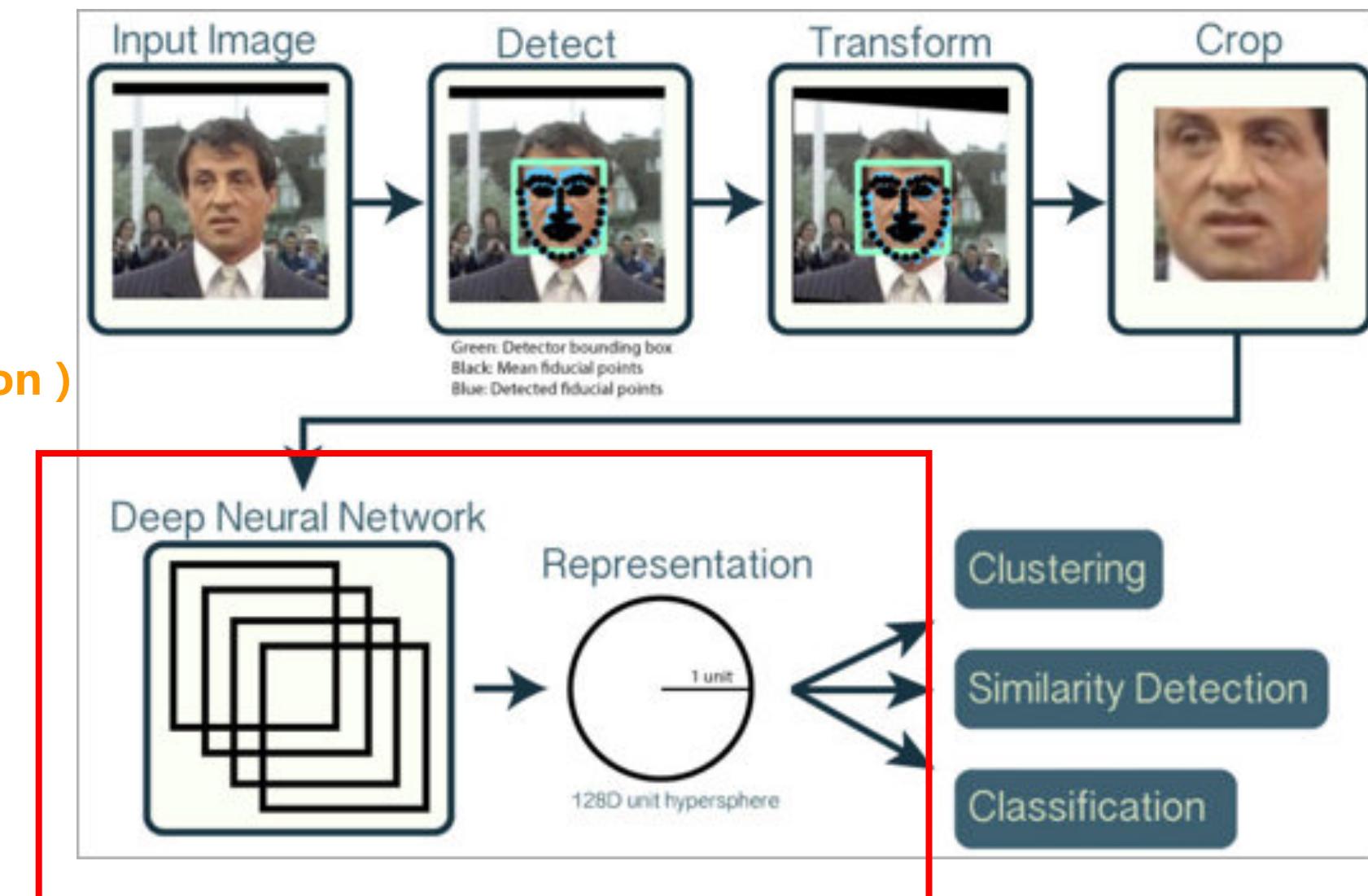
1. 人脸检测 (Face Detection)
2. 人脸对齐 (Face Alignment)
3. 人脸特征表征 (Feature Representation)



人脸识别算法流程

人脸识别算法主要分为三个流程：

1. 人脸检测 (Face Detection)
2. 人脸对齐 (Face Alignment)
3. 人脸特征表征 (Feature Representation)



人脸识别 - 研究进展

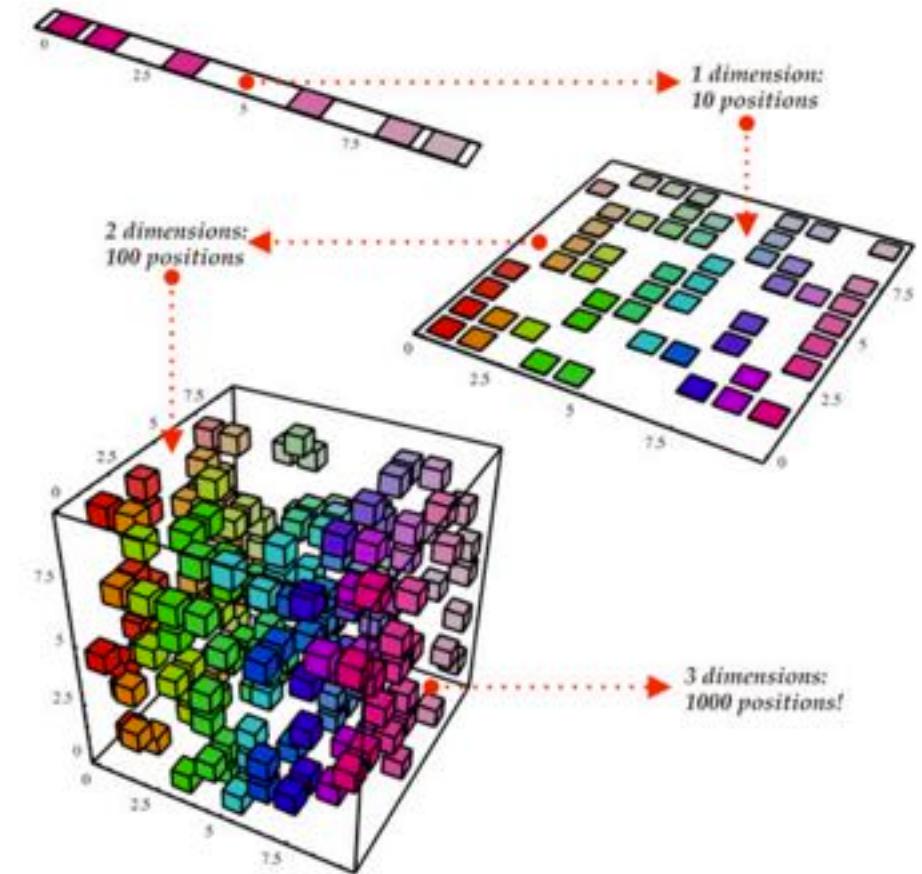
人脸识别算法的研究分为3个发展阶段：

1. 早期算法：基于几何特征、模版匹配、子空间等
2. 人工特征+分类器
3. 基于深度学习的算法

早期算法 – 线性降维

在研究早期，人们使用子空间算法：将人脸图像当作一个高维向量，将其投影到低维空间后，期望得到的低维向量对不同的人具有区分度。比如 Matthew 等使用 PCA[1] 降维得到特征脸（Eigenface）、Peter 等使用 LDA[2] 降维得到 Fisherface 来实现人脸识别。

但是，PCA 和 LDA 都是线性降维技术，显然人脸在高维空间中的分布是非线性的。因此，我们可以进一步使用非线性降维算法。如：流形[3]和核方法[4]。



[1] Matthew Turk,Alex Pentland. Eigenfaces for recognition. 1991, Journal of Cognitive Neuroscience.

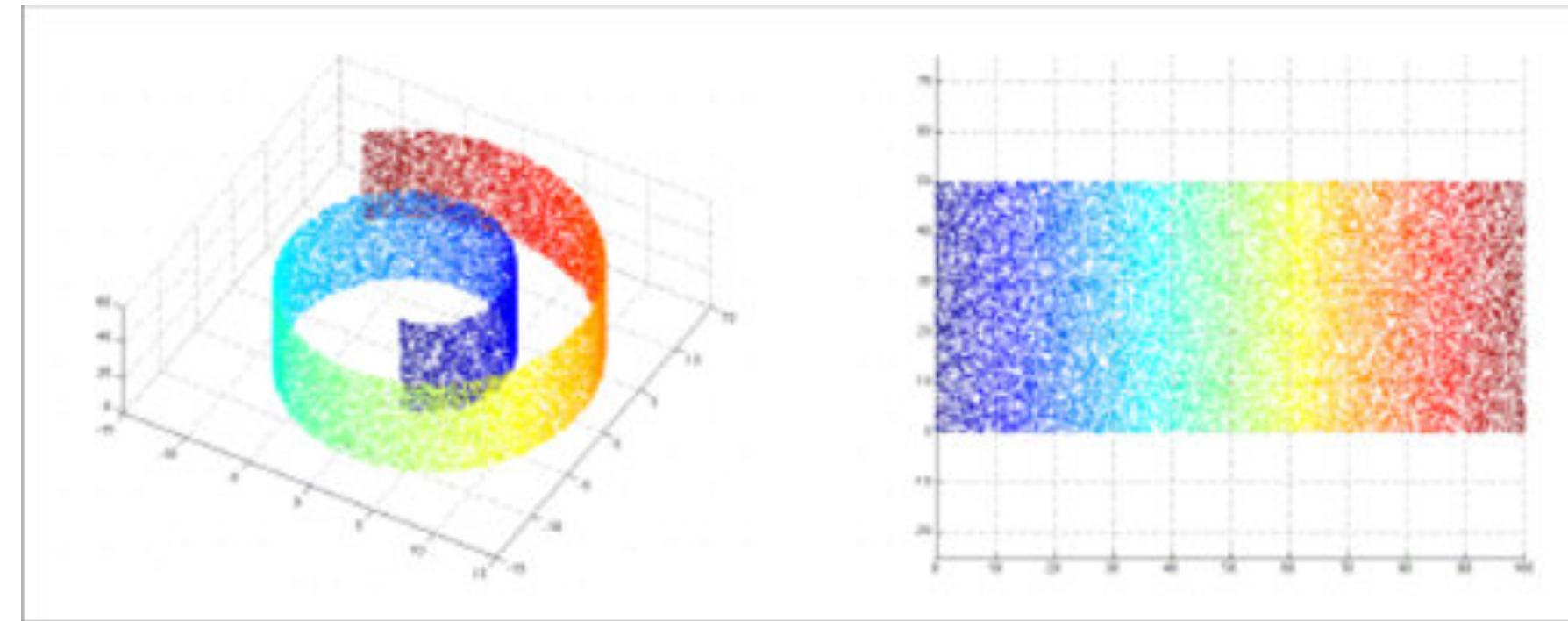
[2] Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. Peter N Belhumeur J P Hespanha David Kriegman. 1997 IEEE Transactions on Pattern Analysis and Machine Intelligence.

[3] He, Xiaofei, et al. Face recognition using Laplacianfaces. Pattern Analysis and Machine Intelligence, IEEE Transactions on 27.3 (2005): 328-340.

[4] Bartlett M S , Movellan J R , Sejnowski TJ. Face Recognition by Independent Component Analysis [J]. IEEE Trans. on Neural Network , 2002 , 13(6) : 1450-1464

早期算法 - 非线性降维

流形学习是一种非线性降维方法，它假设向量点在高维空间中的分布具有某些几何形状，然后在保持这些几何形状约束的前提下将向量投影到低维空间中，这种投影是通过非线性变换完成的。如下所示，我们将3维空间中的瑞士卷（Swiss Roll）数据集非线性降维到2维，并尽可能降低对数据分布的影响。



但是，这些通过直接降维来进行人脸识别的早期算法，普遍都存在严重依赖训练集和测试集场景的问题，且对光照、人脸的表情、姿态敏感，泛化能力不足，不具有太多的实用价值。

人工特征 + 分类器

第二阶段的人脸识别算法普遍采用了**人工特征 + 分类器**的模式。

分类模型经过多年的发展，已经有比较成熟的分类器，如逻辑回归、贝叶斯、支持向量机、神经网络等。

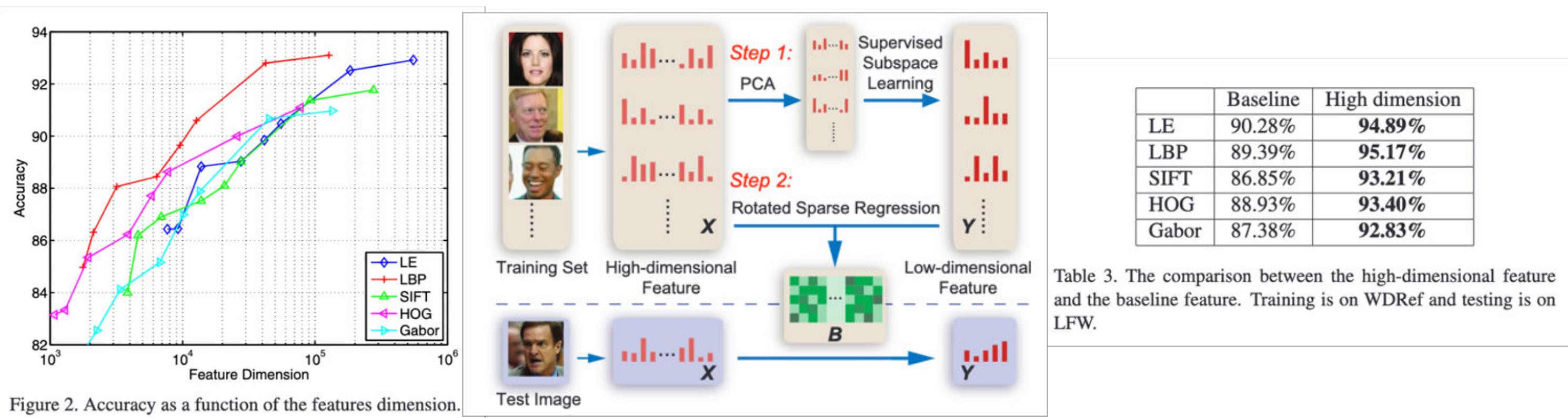
因此，算法的关键是特征工程，即如何设计处有效区分不同人的特征。

于是，计算机视觉领域很多描述图像的特征都先后被用于人脸识别问题，包括HOG、SIFT、Gabor、LBP等。它们中的典型代表是简单高效的LBP（局部二值模式）特征。LBP部分解决了光照敏感问题。

联合贝叶斯[5]是对贝叶斯人脸的改进方法，选用LBP和LE作为基础特征，将人脸图像的差异表示为：相同人因姿态、表情等导致的差异和不同人间的差异两个因素，用潜在变量组成的协方差，建立两张人脸的关联。联合贝叶斯在LFW上取得了**92.4%**的准确率。

MSRA “Feature Master”

2013年，MSRA 的 Dong Chen [6]等继续发力，发表了一篇关于如何使用高维度特征在人脸验证中的文章，作者主要以 LBP 为例，论述了高维特征和验证性能成正相关，即人脸维度越高，验证的准确度就越高。文中最好的方法在 LFW 上的准确率达到了**95.17%**，可谓是集特征工程技艺之大成。



基于深度学习的人脸识别

第三阶段是基于深度学习的算法。2012 年深度学习在 ILSVRC-2012 取得成功后，学术界意识到通过 CNN 学习得到的卷积核明显优于“人工特征+分类器”的方案。自此开始，大家便使用 CNN + 海量人脸图片来提取特征，以此替代人工特征的设计。

前期，大家在神经网络结构和输入数据设计方面做了大量尝试。

后期，大家主要集中在人脸损失函数的改进上，即约束 CNN 学习对区分不同人更有效的特征。至今，仍然不断有效果更好的损失函数设计。而人脸识别研究领域（从人脸检测到人脸识别）也彻底被深度学习“攻占”了。

我们主要介绍两个模型：Facebook 在 CVPR 2014 发布的 DeepFace [7] 和 Google 在 CVPR 2015 发布的 FaceNet [8]

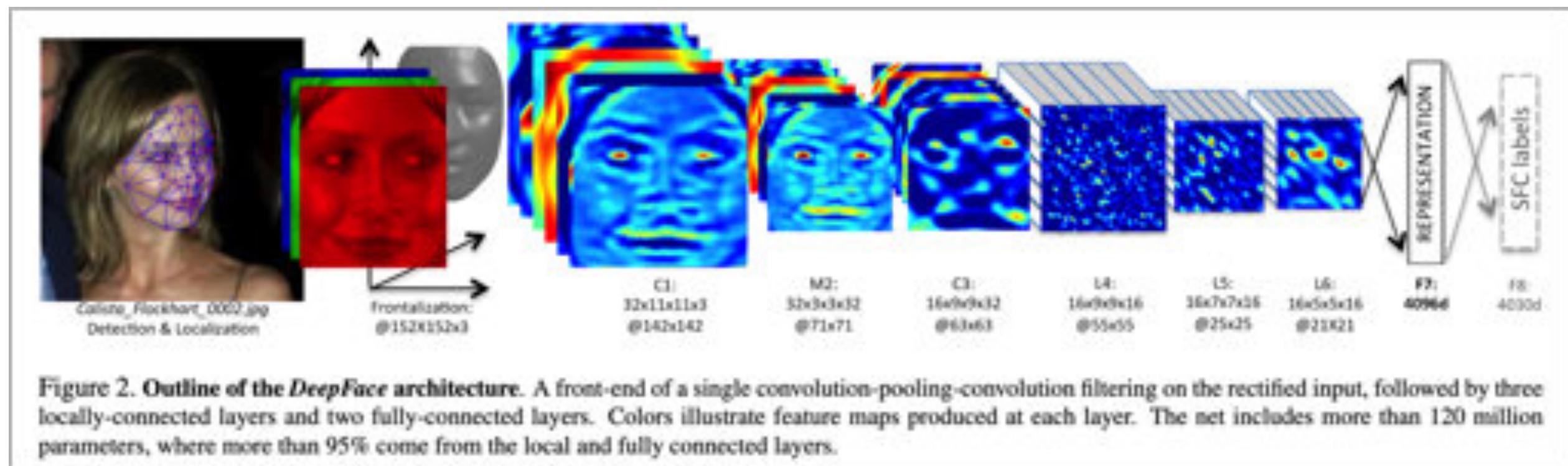
[7] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. 2014, computer vision and pattern recognition.

[8] Florian Schroff, Dmitry Kalenichenko, James Philbin. FaceNet: A unified embedding for face recognition and clustering. 2015, computer vision and pattern recognition.

Facebook DeepFace

DeepFace 是深度卷积神经网络在人脸识别领域的奠基之作。他们使用 3D 模型来解决人脸对齐问题，同时又使用了9层深度神经网络来做人脸特征表示。损失函数使用了 Softmax Loss，最后通过特征嵌入（Feature Embedding）得到固定长度的人脸特征向量。

DeepFace 在LFW上取得了**97.35%**的准确率，已经接近人类的水平。



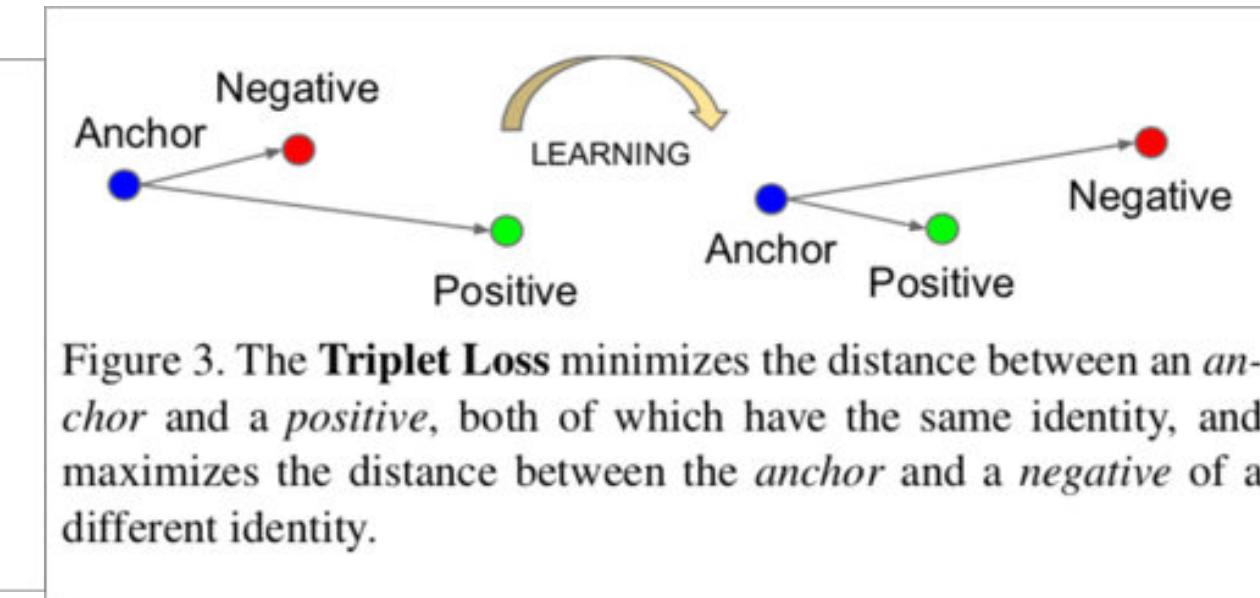
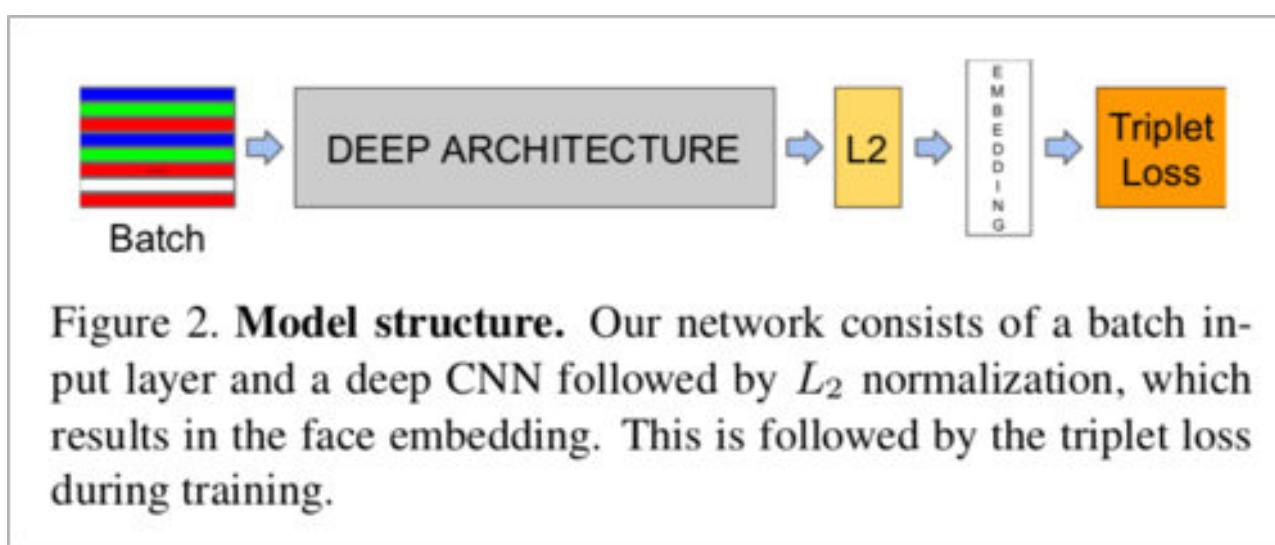
Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification.
2014, computer vision and pattern recognition.

Google FaceNet

次年，Google FaceNet 创新的提出了使用三元组损失函数（Triplet Loss）替代 Softmax Loss。

在一个超球空间上进行优化使类内距离更紧凑，类间距离更远，最后得到了一个紧凑的128维人脸特征，其网络使用 GoogLeNet 的 Inception 模型，模型参数量较小，精度更高。

FaceNet 在LFW上取得了**99.63%**的准确率。



人脸检测工具介绍

开放 API – 某AI开放平台

Request

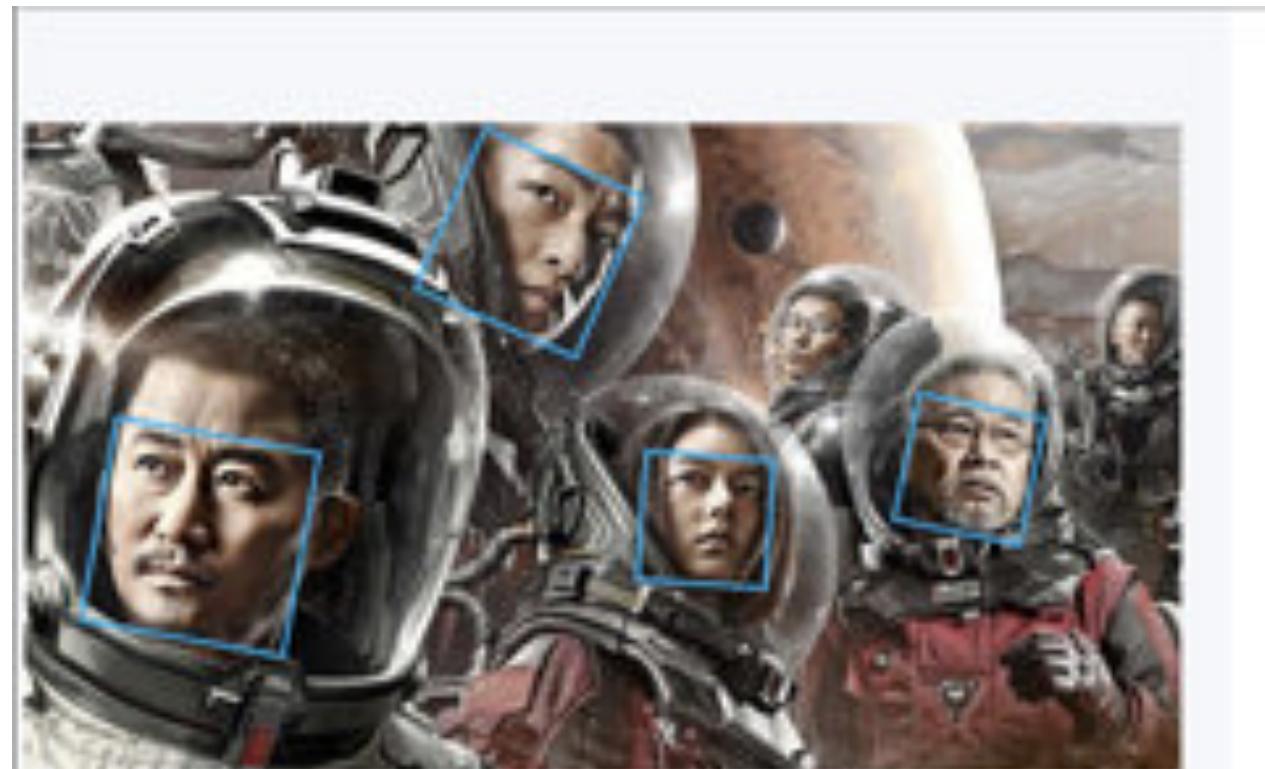
Response

[查看接口文档 >](#)

{

```
"face_num": 6,  
"Face_list": [  
    {  
        "face_token":  
            "a4683ca9187d63ea7fcbb53359",  
        "location": {  
            "left": 67.90,  
            "top": 207.2,  
            "width": 141,  
            "height": 133  
        },  
        "angle": {  
            "yaw": 9.64,  
            "pitch": -1.12,  
            "roll": 0.00  
        },  
        "face_probability": 0.9999  
    }]
```

开放 API – 某AI开放平台



已检测到图中 4 张人脸，并生成对应face_token，点击人脸图片查看结果信息。您可将以上信息传给其他API进行后续处理和分析。推荐使用人脸属性和人脸关键点API。



 本地上传

图片URL

检测



更多



OpenCV

开源计算机视觉库（Open Source Computer Vision Library, OpenCV）遵从BSD协议许可。

它具有C++，Python和Java接口，支持Windows，Linux，Mac OS，iOS和Android。

OpenCV专为提高计算效率而设计，专注于实时应用。该库以优化的C / C++编写，通过OpenCL可以启用多核处理和硬件加速模式。

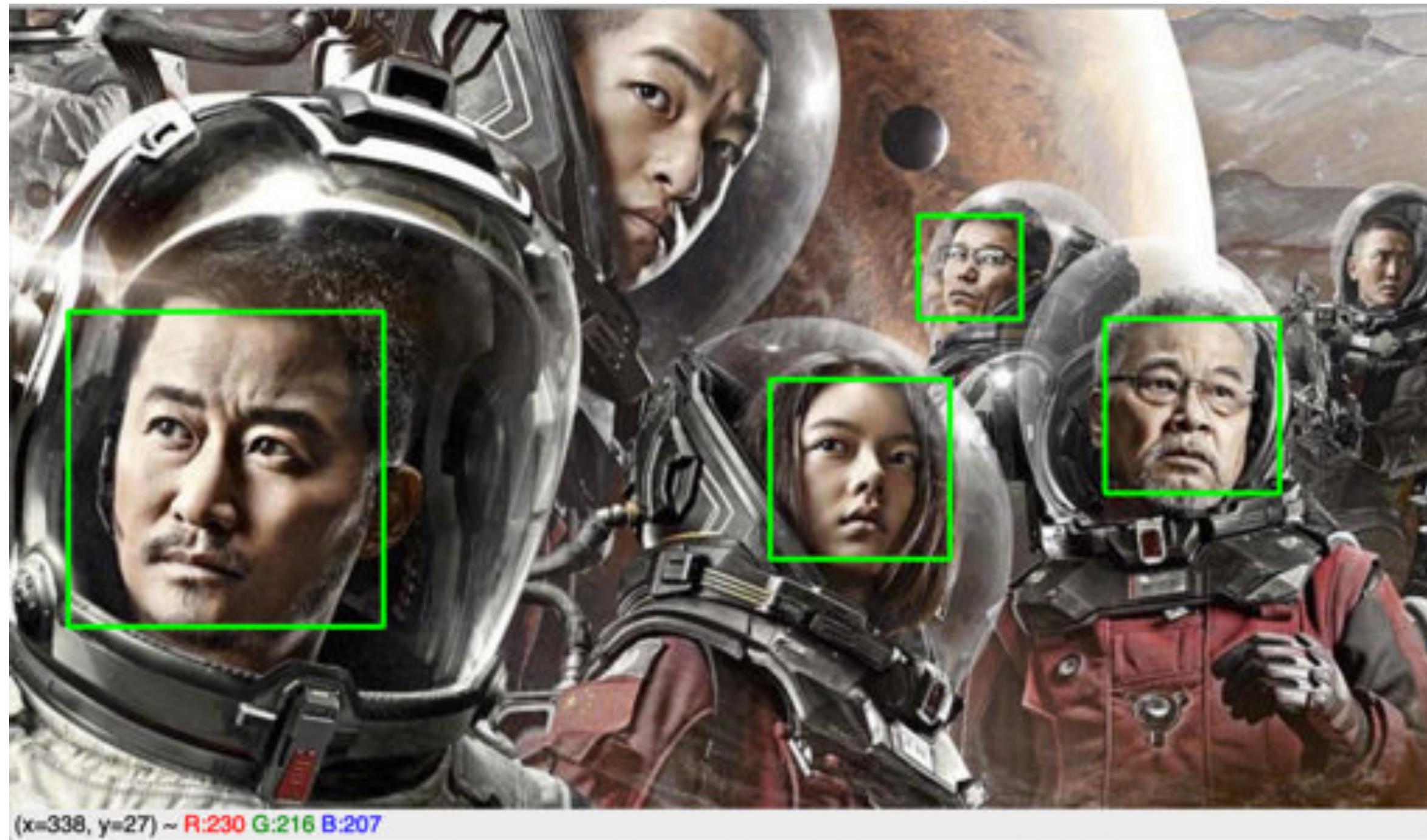
安装 OpenCV 的 Python 接口：

```
$ pip3 install python-opencv
```

使用 OpenCV 进行人脸检测

```
1 # -*- coding: utf-8 -*-
2
3 import cv2
4 import sys
5
6 # Get user supplied values
7 imagePath = sys.argv[1]
8 cascPath = "haarcascade_frontalface_default.xml"
9
10 # Create the haar cascade
11 faceCascade = cv2.CascadeClassifier(cascPath)
12
13 # Read the image
14 image = cv2.imread(imagePath)
15 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
16
17 # Detect faces in the image
18 faces = faceCascade.detectMultiScale(
19     gray,
20     scaleFactor=1.1,
21     minNeighbors=5,
22     minSize=(30, 30),
23     flags = cv2.CV_HAAR_SCALE_IMAGE # opencv3 需要注释此标记
24 )
25
26 print("Found {} faces!".format(len(faces)))
27
28 # Draw a rectangle around the faces
29 for (x, y, w, h) in faces:
30     cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
31
32 cv2.imshow("Faces found", image)
33 cv2.waitKey(0)
```

使用 OpenCV 进行人脸检测



使用 face_recognition 库进行人脸检测

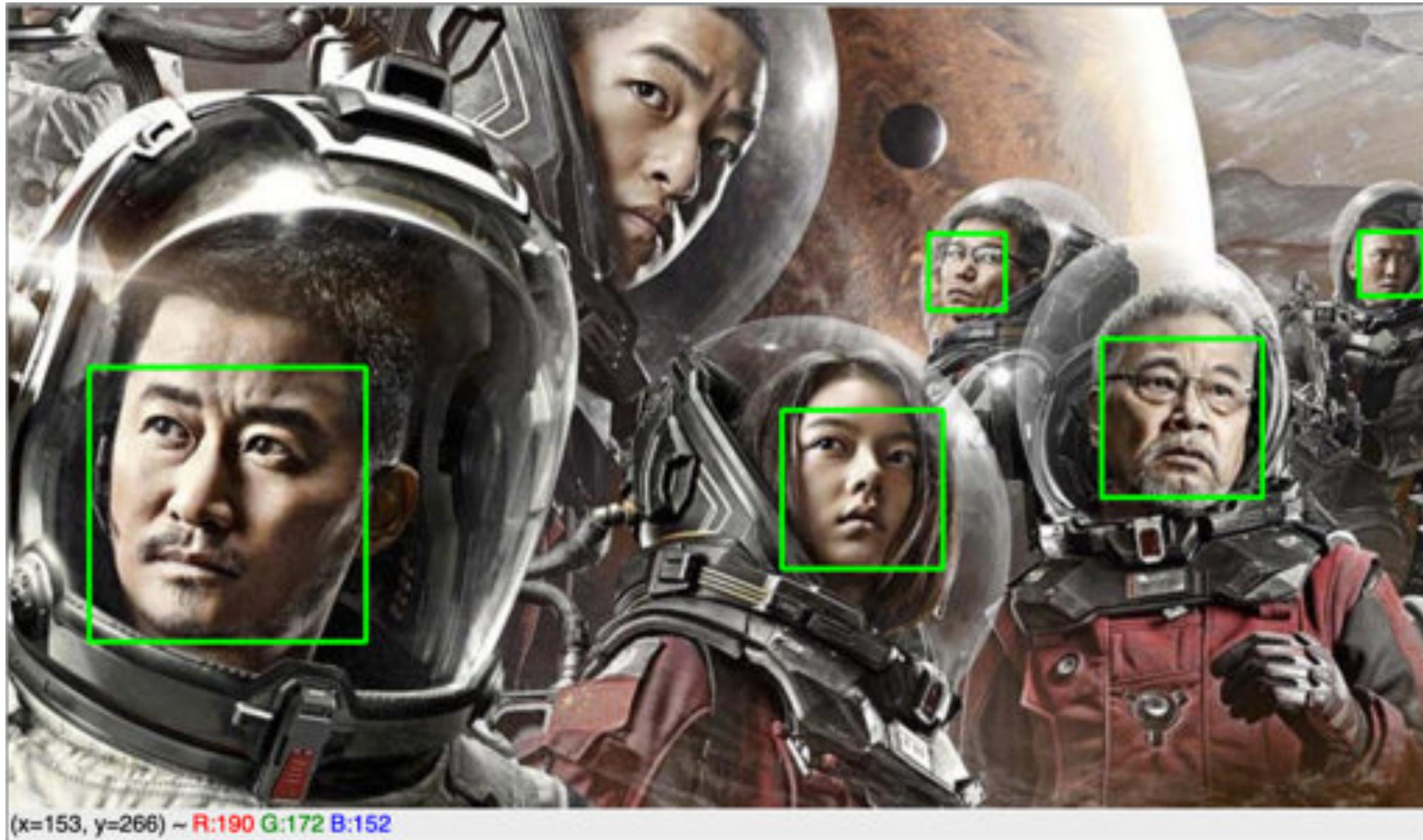
使用 Dlib 最先进的面部识别功能构建而成，具有深度学习功能。该模型在 LFW 上的准确率为99.38%。

安装 face_recognition 库：

```
$ pip3 install face_recognition
```

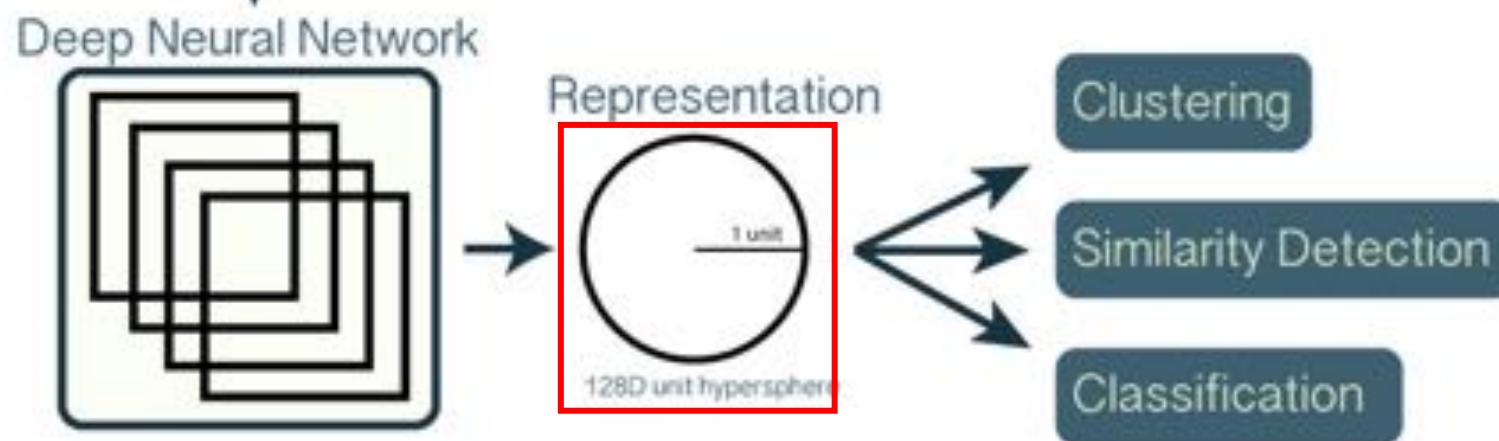
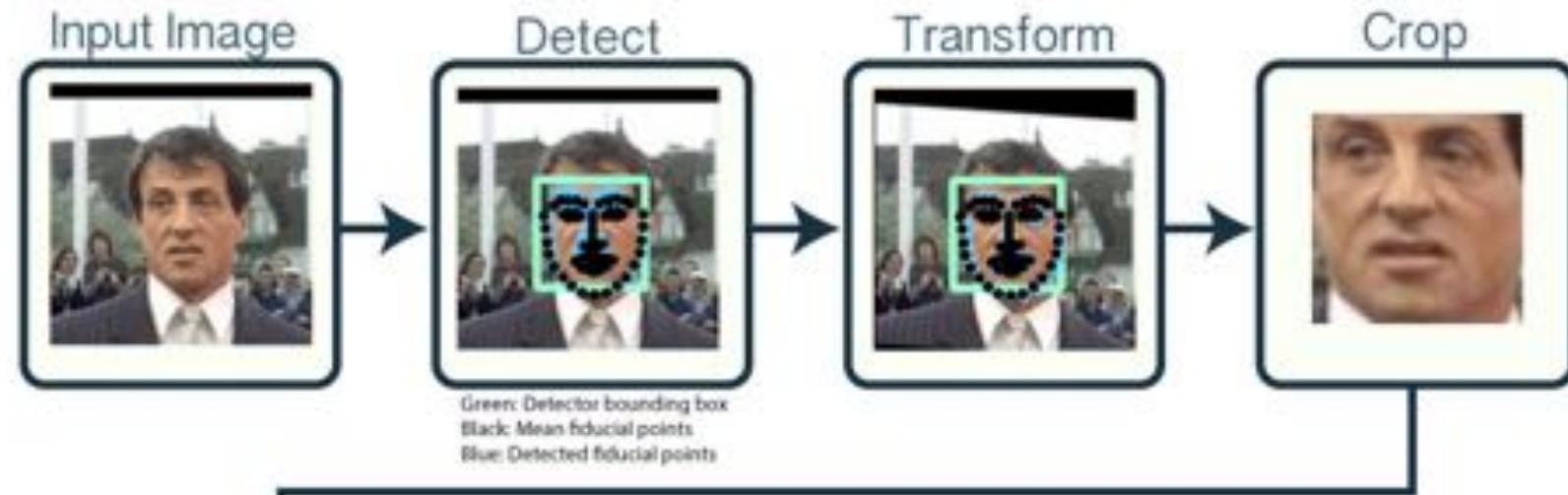
```
1 # -*- coding: utf-8 -*-
2
3 import cv2
4 import sys
5
6 import face_recognition
7
8
9 # Get user supplied values
10 imagePath = sys.argv[1]
11
12 # Load the image with face_recognition
13 image = face_recognition.load_image_file(imagePath)
14 # Detect faces in the image
15 face_locations = face_recognition.face_locations(image)
16
17 print("Found {} faces!".format(len(face_locations)))
18
19
20 # Read the image with openCV
21 image = cv2.imread(imagePath)
22
23 # Draw a rectangle around the faces
24 for (top, right, bottom, left) in face_locations:
25     cv2.rectangle(image, (left, top), (right, bottom), (0, 255, 0), 2)
26
27
28 cv2.imshow("Faces found", image)
29 cv2.waitKey(0)
```

使用 face_recognition 库进行人脸检测

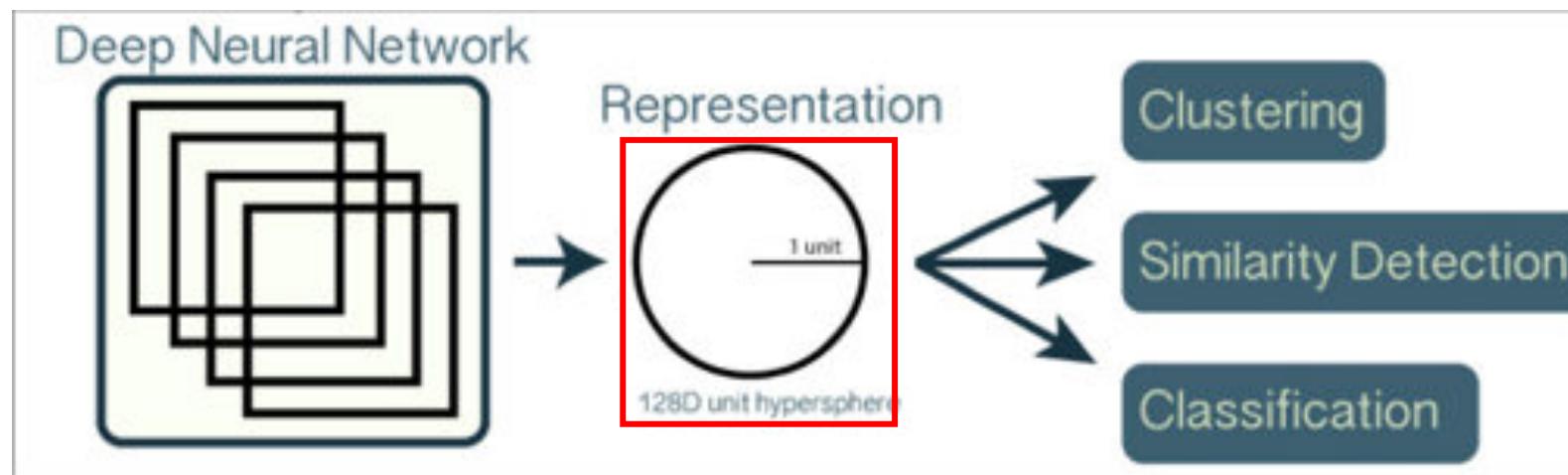
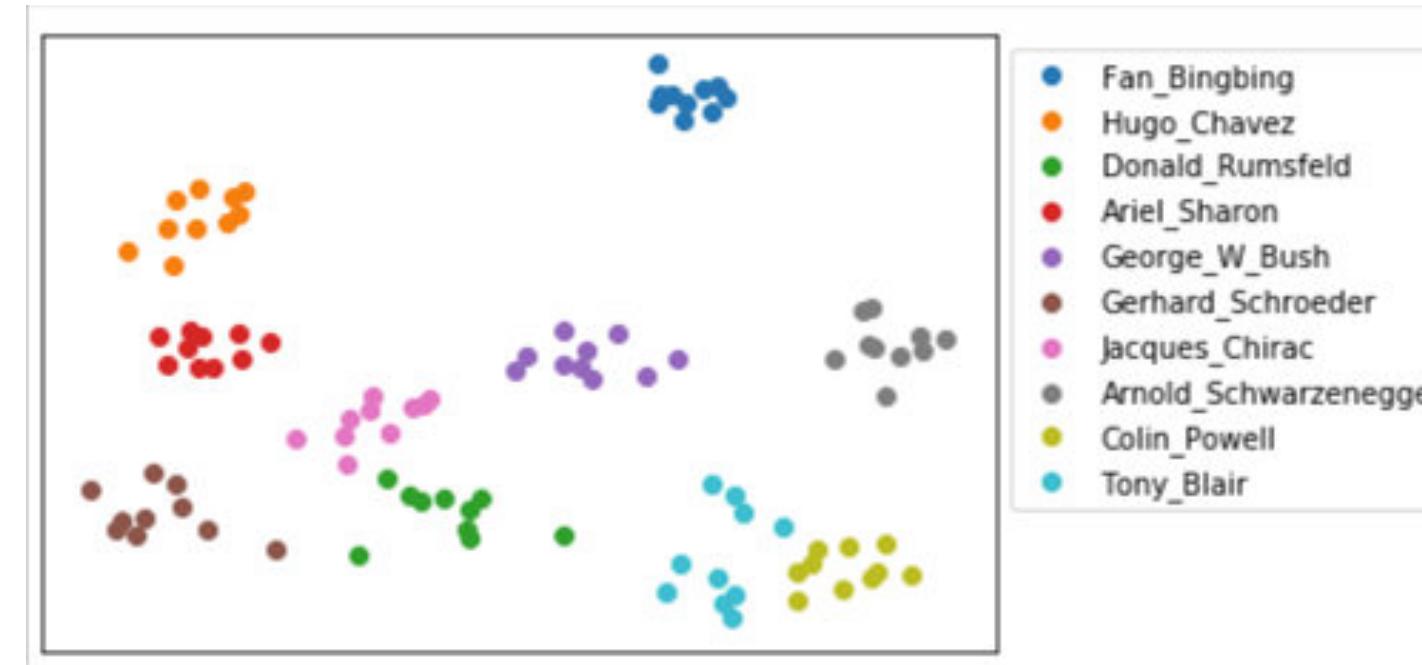


解析 FaceNet 人脸识别模型

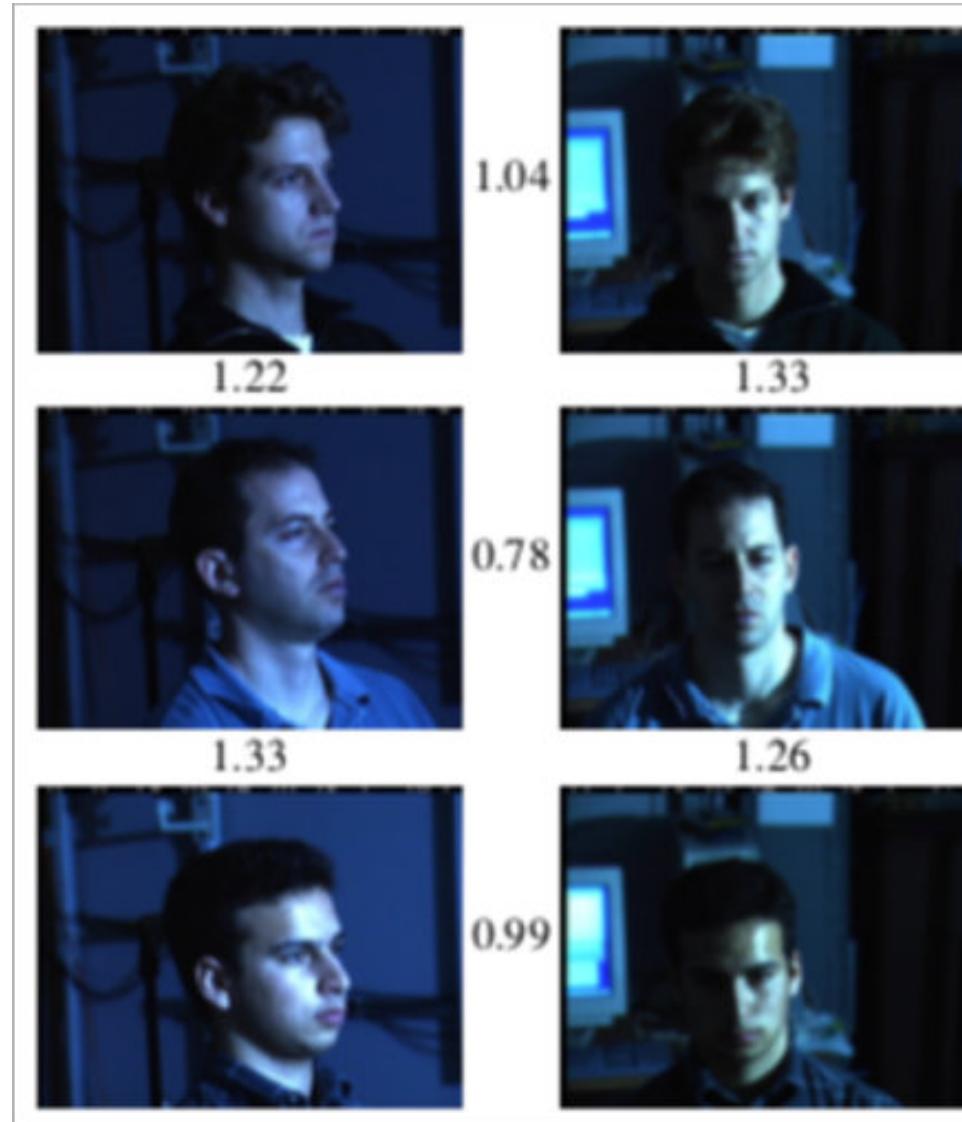
FaceNet 模型介绍



FaceNet 模型介绍



FaceNet 在复杂光照和姿态下表现出色



threshold = 1.1

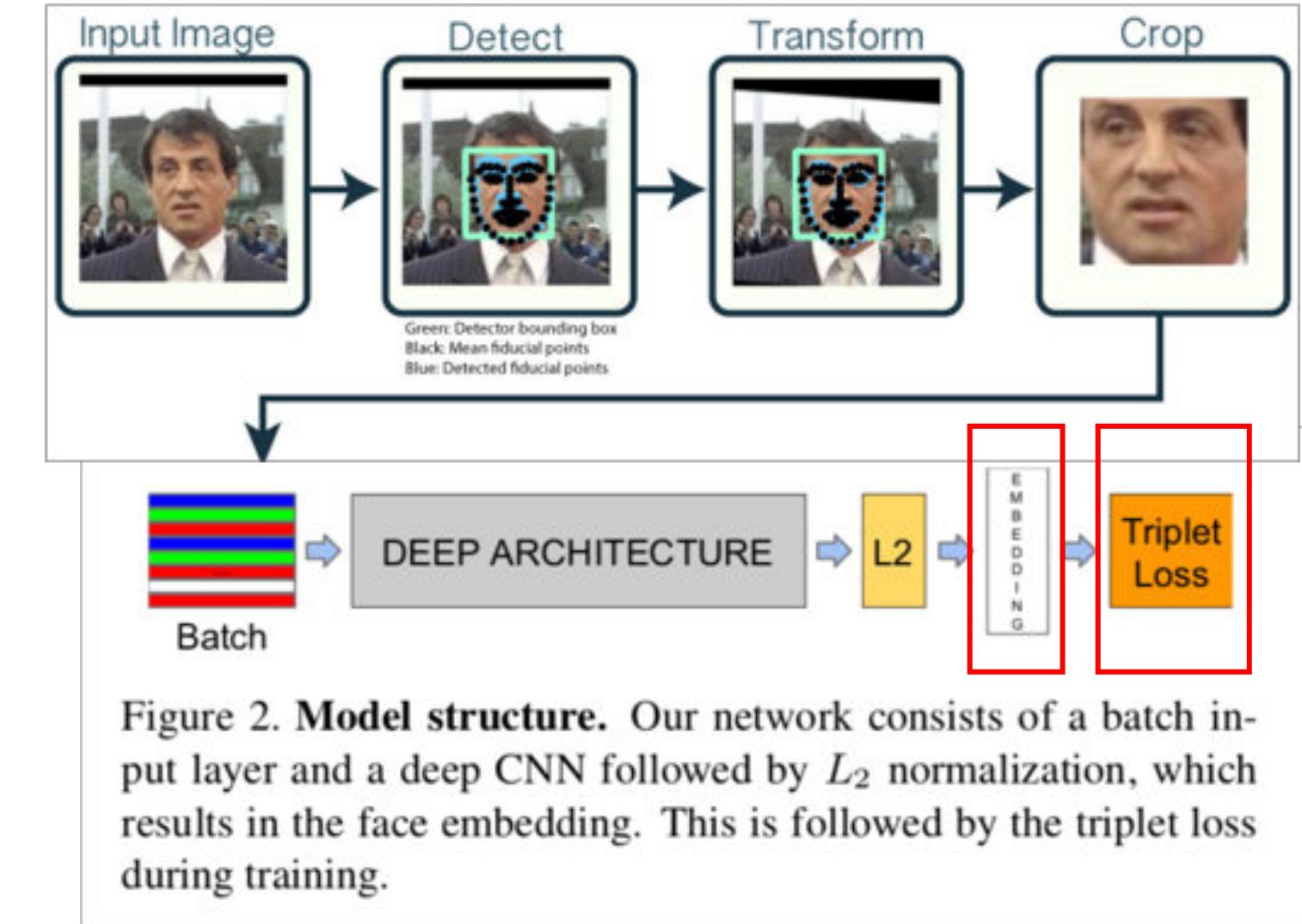


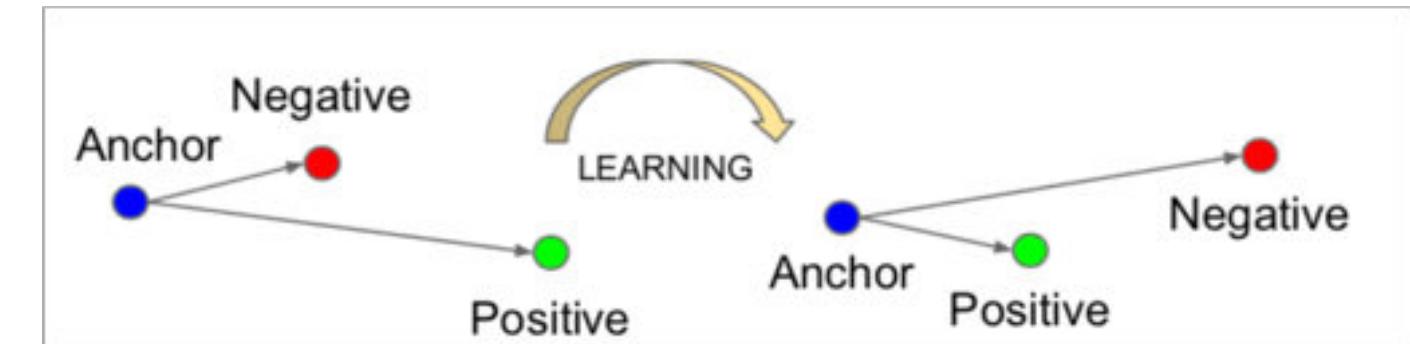
Figure 2. Model structure. Our network consists of a batch input layer and a deep CNN followed by L_2 normalization, which results in the face embedding. This is followed by the triplet loss during training.

模型创新点：Triplet Loss

Triplet Loss 核心思想：将人脸图像 x 嵌入到 d 维的欧几里得特征空间 $f(x) \in R^d$ 。训练使得同一人的不同人脸向量间距离小，不同人间向量距离大。计算欧几里得距离： $\|f(x)\|_2^2 = 1$ 。

选取以下3张人脸图像进行优化学习：

- 基本人脸图像 x_i^a (Anchor)
- 与 x_i^a 最像（距离最小）的错误人脸 x_i^n (Negative)
- 与 x_i^a 最不像（距离最大）的正确人脸 x_i^p (Positive)



$$\|x_i^a - x_i^p\|_2^2 + \alpha < \|x_i^a - x_i^n\|_2^2, \forall (x_i^a, x_i^p, x_i^n) \in T$$

其中， α 是 Positive 与 Negative 的边界， T 是训练集中所有可行三元组的集合（基数为 N ）。

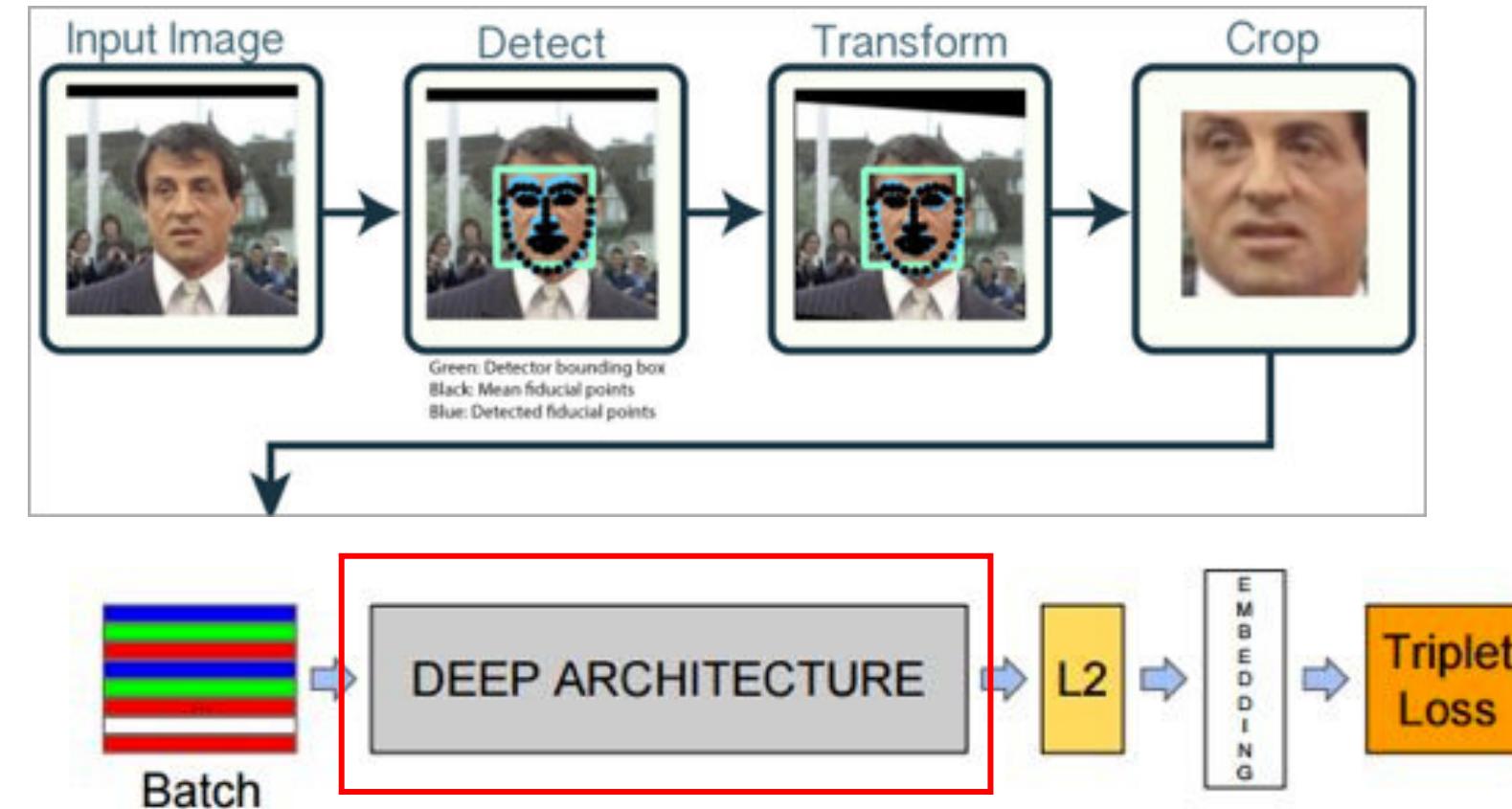
损失函数：

$$L = \sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$

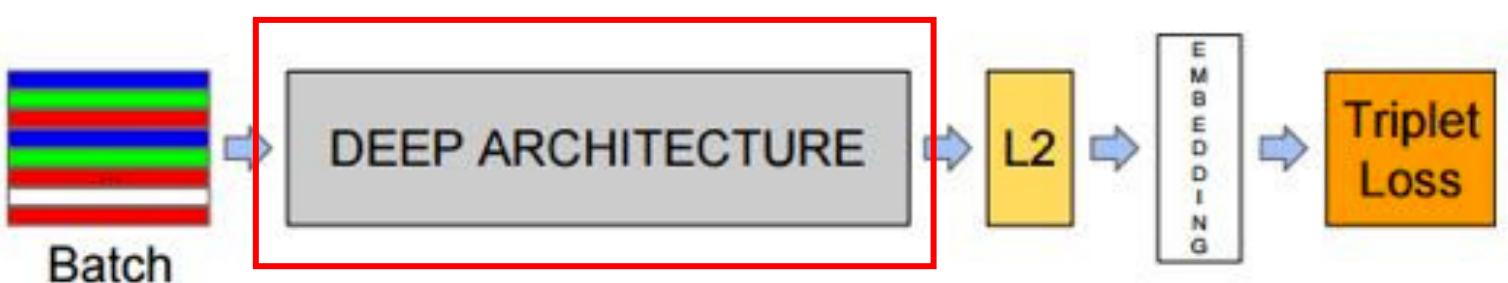
为了快速收敛，训练时需要选择合适的三元组。即：

对于给定 x_i^a ，求 x_i^p 和 x_i^n ，分别使 $\text{argmax}_{x_i^p} \|f(x_i^a) - f(x_i^p)\|_2^2$ 和 $\text{argmin}_{x_i^n} \|f(x_i^a) - f(x_i^n)\|_2^2$

FaceNet 模型架构



FaceNet – NN1



layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

Table 1. **NN1.** This table shows the structure of our Zeiler&Fergus [22] based model with 1×1 convolutions inspired by [9]. The input and output sizes are described in $\text{rows} \times \text{cols} \times \# \text{filters}$. The kernel is specified as $\text{rows} \times \text{cols}$, stride and the maxout [6] pooling size as $p = 2$.

FaceNet – NN2

type	output size	depth	#1x1	#3x3 reduce	#3x3	#5x5 reduce	#5x5	pool proj (p)	params	FLOPS
conv1 (7x7x3, 2)	112x112x64	1							98	119M
max pool + norm	56x56x64	0						m 3x3, 2		
inception (2)	56x56x192	2		64	192				115K	360M
norm + max pool	28x28x192	0						m 3x3, 2		
inception (3a)	28x28x256	2	64	96	128	16	32	m, 12p	364K	128M
inception (3b)	28x28x320	2	64	96	128	32	64	L ₂ , 64p	2.59K	179M
inception (3c)	14x14x640	2	0	128	256,2	32	64,2	m 3x3, 2	398K	108M
inception (4a)	14x14x640	2	256	96	192	32	64	L ₂ , 128p	545K	107M
inception (4b)	14x14x640	2	224	112	224	32	64	L ₂ , 128p	595K	117M
inception (4c)	14x14x640	2	192	128	256	32	64	L ₂ , 128p	654K	128M
inception (4d)	14x14x640	2	160	144	288	32	64	L ₂ , 128p	725K	142M
inception (4e)	7x7x1024	2	0	160	256,2	64	128,2	m 3x3, 2	717K	56M
inception (5a)	7x7x1024	2	384	192	384	48	128	L ₂ , 128p	1.6M	78M
inception (5b)	7x7x1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1x1x1024	0								
fully conn	1x1x128	1							131K	0.1M
L2 normalization	1x1x128	0								
total									7.5M	1.6B

Table 2. NN2. Details of the NN2 Inception incarnation. This model is almost identical to the one described in [16]. The two major differences are the use of L_2 pooling instead of max pooling (m), where specified. The pooling is always 3×3 (aside from the final average pooling) and in parallel to the coevolutional modules inside each Inception module. If there is a dimensionality reduction after the pooling it is denoted with p. 1x1, 3x3, and 5x5 pooling are then concatenated to get the final output.

FaceNet 计算量对结果影响

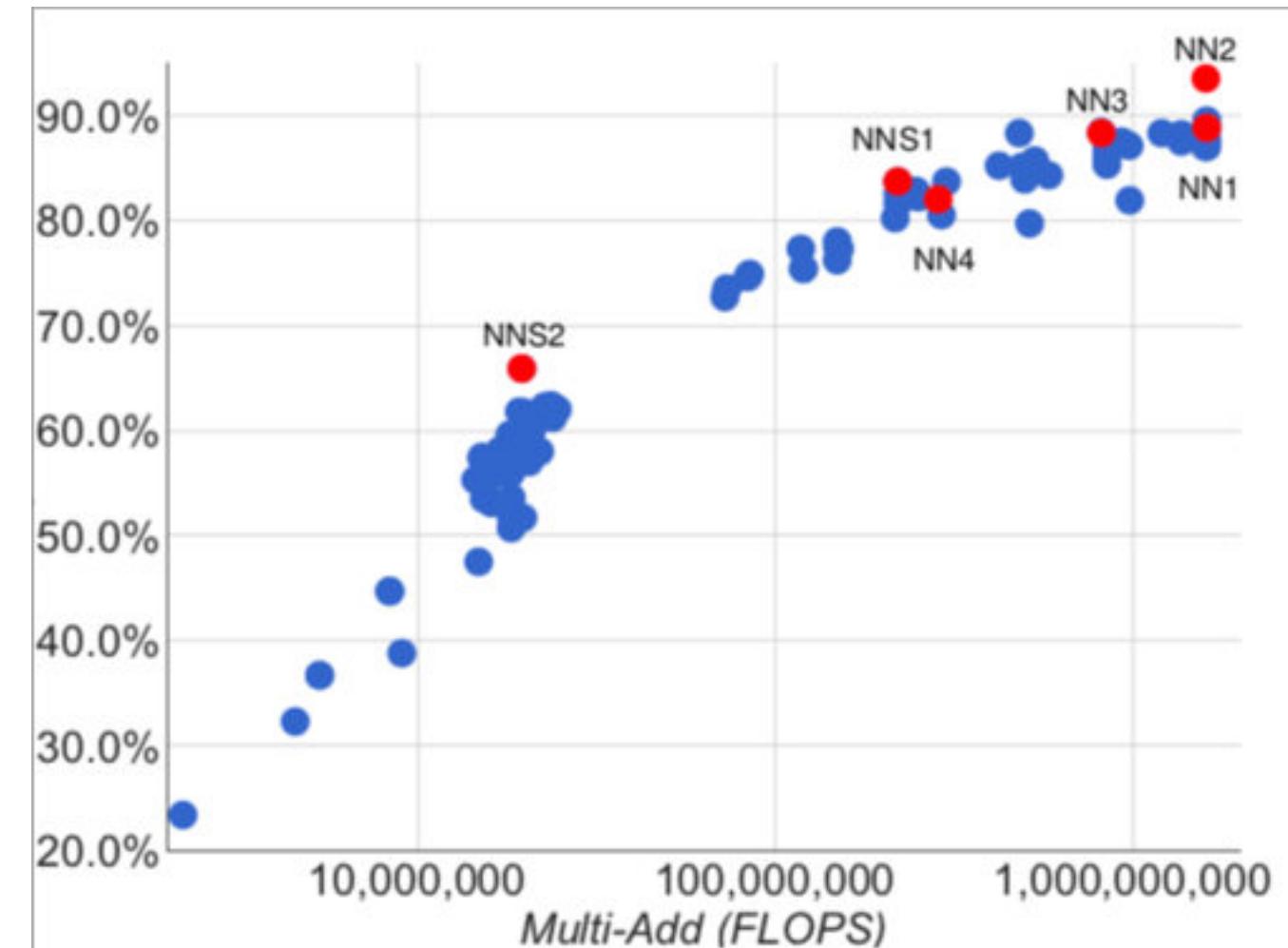


Figure 4. **FLOPS vs. Accuracy trade-off.** Shown is the trade-off between FLOPS and accuracy for a wide range of different model sizes and architectures. Highlighted are the four models that we focus on in our experiments.

FaceNet CNN架构对结果影响

architecture	VAL
NN1 (Zeiler&Fergus 220×220)	87.9% ± 1.9
NN2 (Inception 224×224)	89.4% ± 1.6
NN3 (Inception 160×160)	88.3% ± 1.7
NN4 (Inception 96×96)	82.0% ± 2.3
NNS1 (mini Inception 165×165)	82.4% ± 2.4
NNS2 (tiny Inception 140×116)	51.9% ± 2.9

Table 3. Network Architectures. This table compares the performance of our model architectures on the hold out test set (see section 4.1). Reported is the mean validation rate VAL at 10^{-3} false accept rate. Also shown is the standard error of the mean across the five test splits.

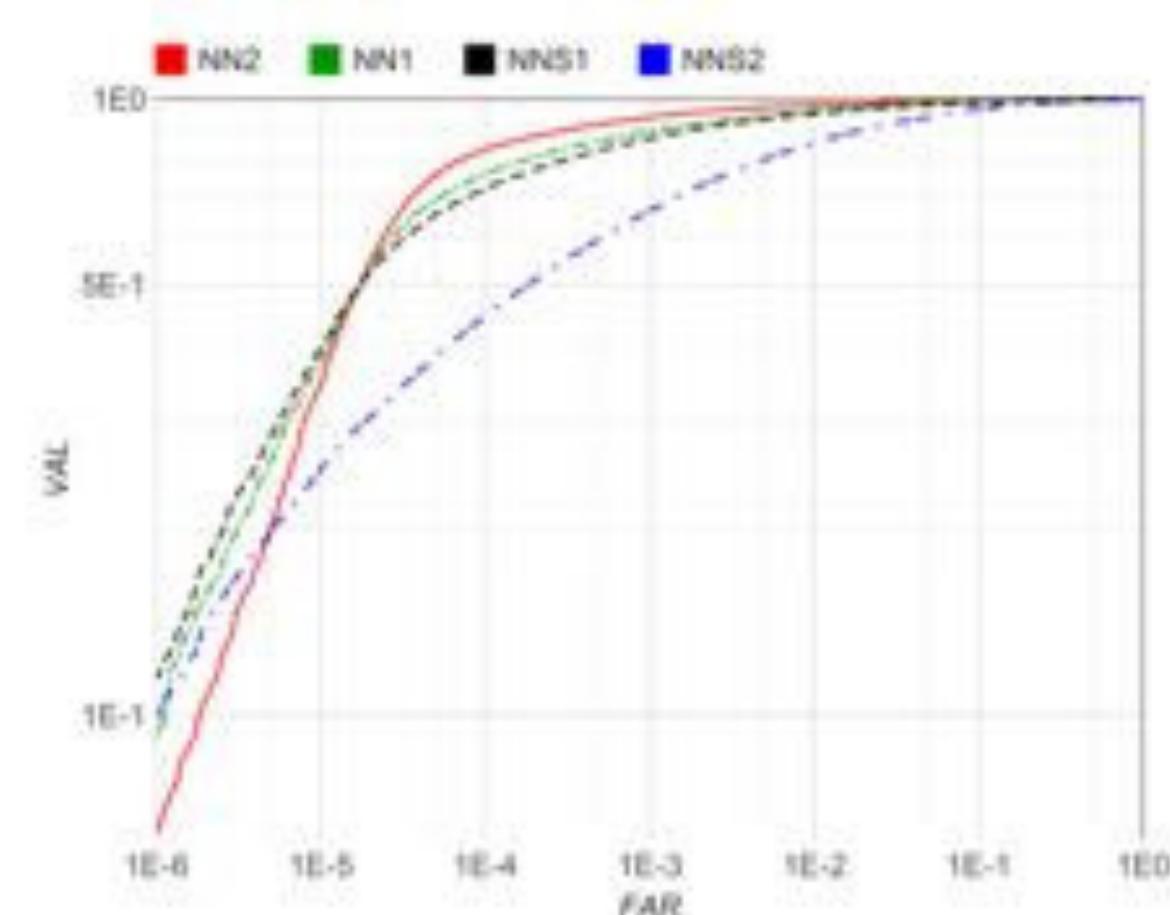


Figure 5. Network Architectures. This plot shows the complete ROC for the four different models on our personal photos test set from section 4.2. The sharp drop at 10^{-4} FAR can be explained by noise in the groundtruth labels. The models in order of performance are: **NN2**: 224×224 input Inception based model; **NN1**: Zeiler&Fergus based network with 1×1 convolutions; **NNS1**: small Inception style model with only 220M FLOPS; **NNS2**: tiny Inception model with only 20M FLOPS.

FaceNet 图像质量对结果影响

jpeg q	val-rate
10	67.3%
20	81.4%
30	83.9%
50	85.5%
70	86.1%
90	86.5%

#pixels	val-rate
1,600	37.8%
6,400	79.5%
14,400	84.5%
25,600	85.7%
65,536	86.4%

Table 4. Image Quality. The table on the left shows the effect on the validation rate at 10E-3 precision with varying JPEG quality. The one on the right shows how the image size in pixels effects the validation rate at 10E-3 precision. This experiment was done with NN1 on the first split of our test hold-out dataset.

FaceNet 特征空间维度对结果影响

#dims	VAL
64	86.8% ± 1.7
128	87.9% ± 1.9
256	87.7% ± 1.9
512	85.6% ± 2.0

Table 5. Embedding Dimensionality. This Table compares the effect of the embedding dimensionality of our model NN1 on our hold-out set from section 4.1. In addition to the VAL at 10E-3 we also show the standard error of the mean computed across five splits.

FaceNet 训练数据量对结果影响

#training images	VAL
2,600,000	76.3%
26,000,000	85.1%
52,000,000	85.1%
260,000,000	86.2%

Table 6. Training Data Size. This table compares the performance after 700h of training for a smaller model with 96x96 pixel inputs. The model architecture is similar to NN2, but without the 5x5 convolutions in the Inception modules.

FaceNet 模型的创新与突破

使用 Triplet Loss 替代 Softmax Loss

模型输出为 128 维人脸特征向量，而不是人脸分类结果（FaceID）

准确率创历史新高，LFW 上达 99.63%

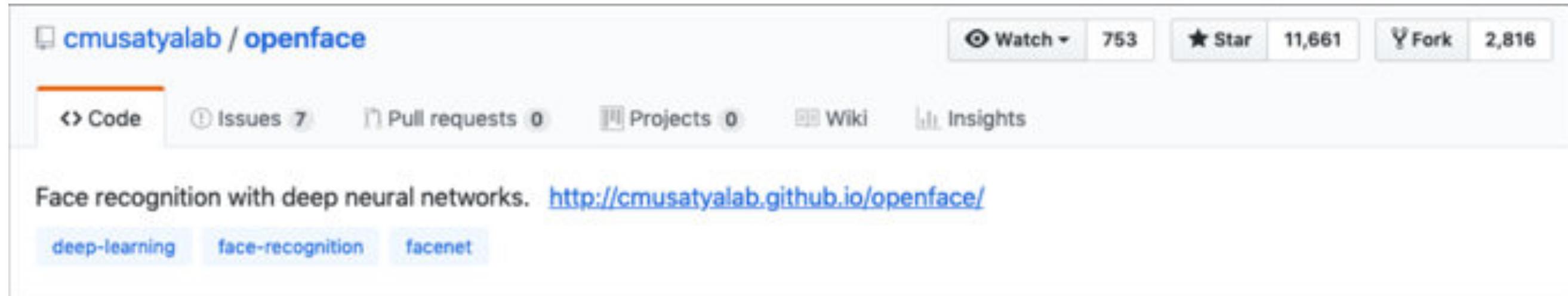
模型“瘦身”，减少了参数和计算量

通用性好，可应用到人脸验证、识别、分类、搜索和跟踪等多种场景。

实战 FaceNet 人脸识别模型

OpenFace 项目介绍

[OpenFace](#) 是用 Python 和 Torch 实现的基于深度神经网络的人脸识别模型 FaceNet。因为该项目是使用 Torch 开发的，所以也支持 CUDA。



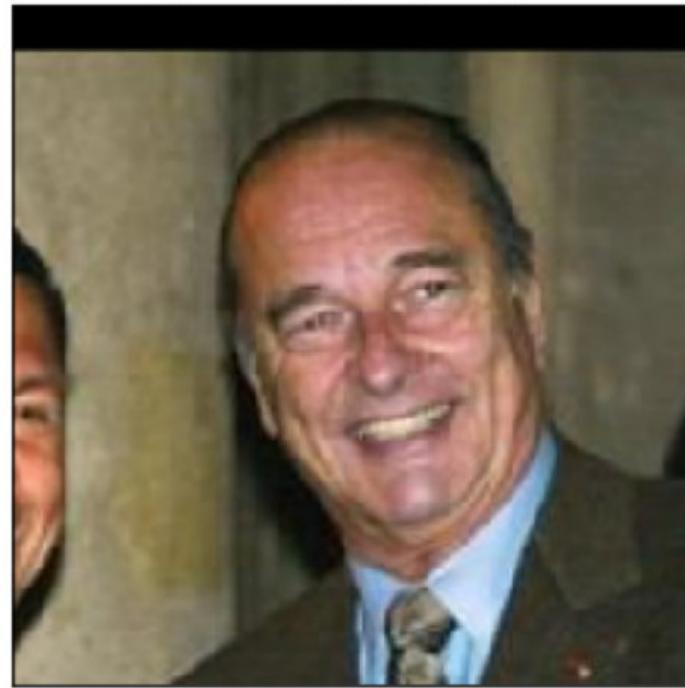
目前，许多开源爱好者受到 OpenFace 项目启发，又实现了 Keras 和 TensorFlow (low-level API) 版本的 FaceNet。

Try it

测试与可视化分析

人脸识别测试

Recognized as Jacques_Chirac



Recognized as Deng_chao



人脸识别测试

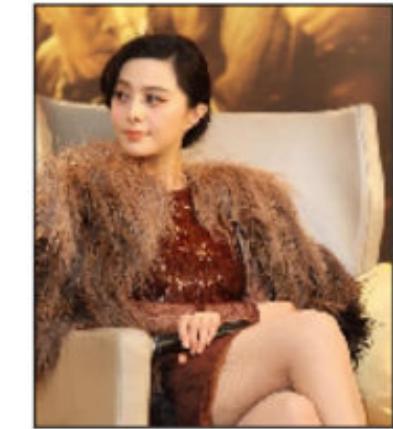
Distance = 0.12



Distance = 1.19



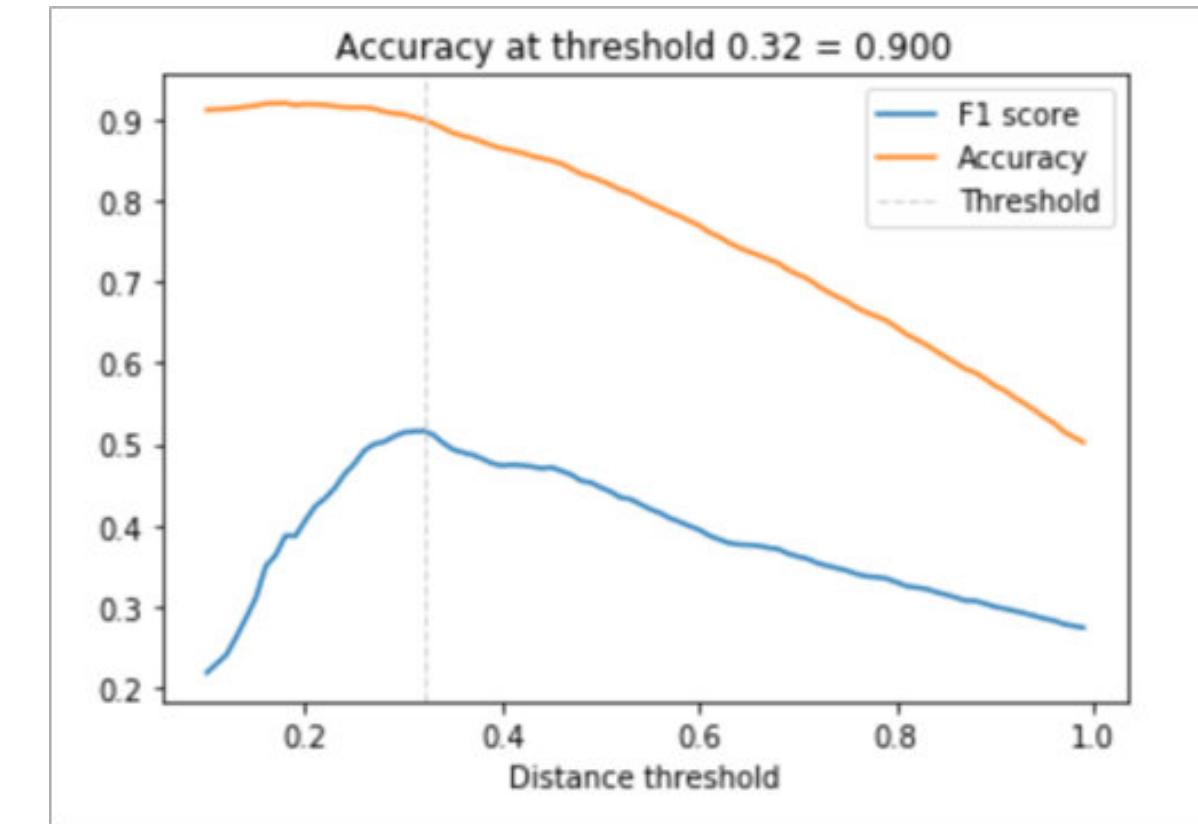
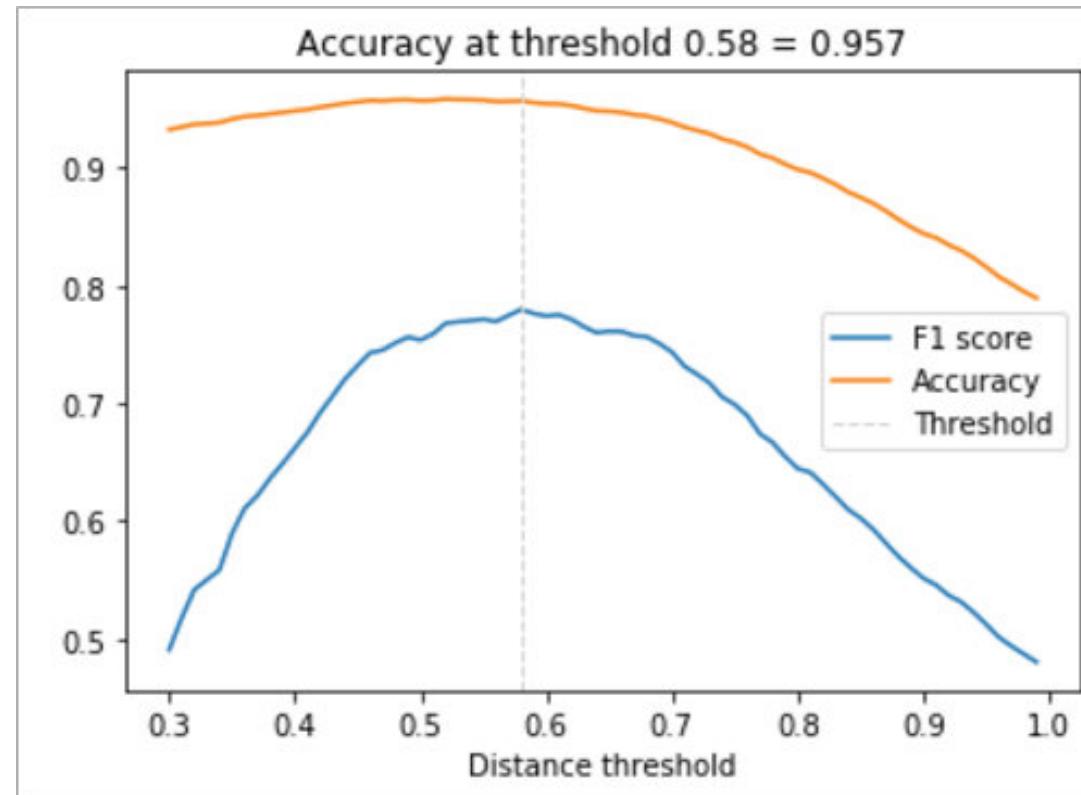
Distance = 0.22



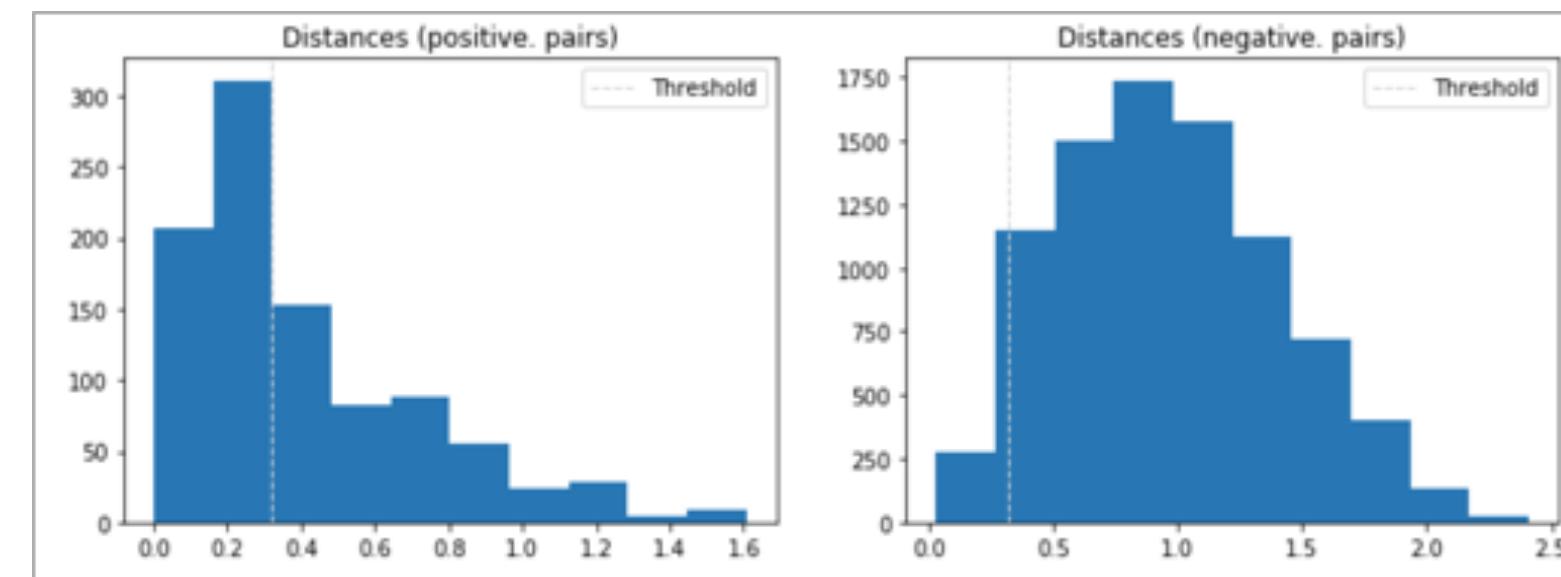
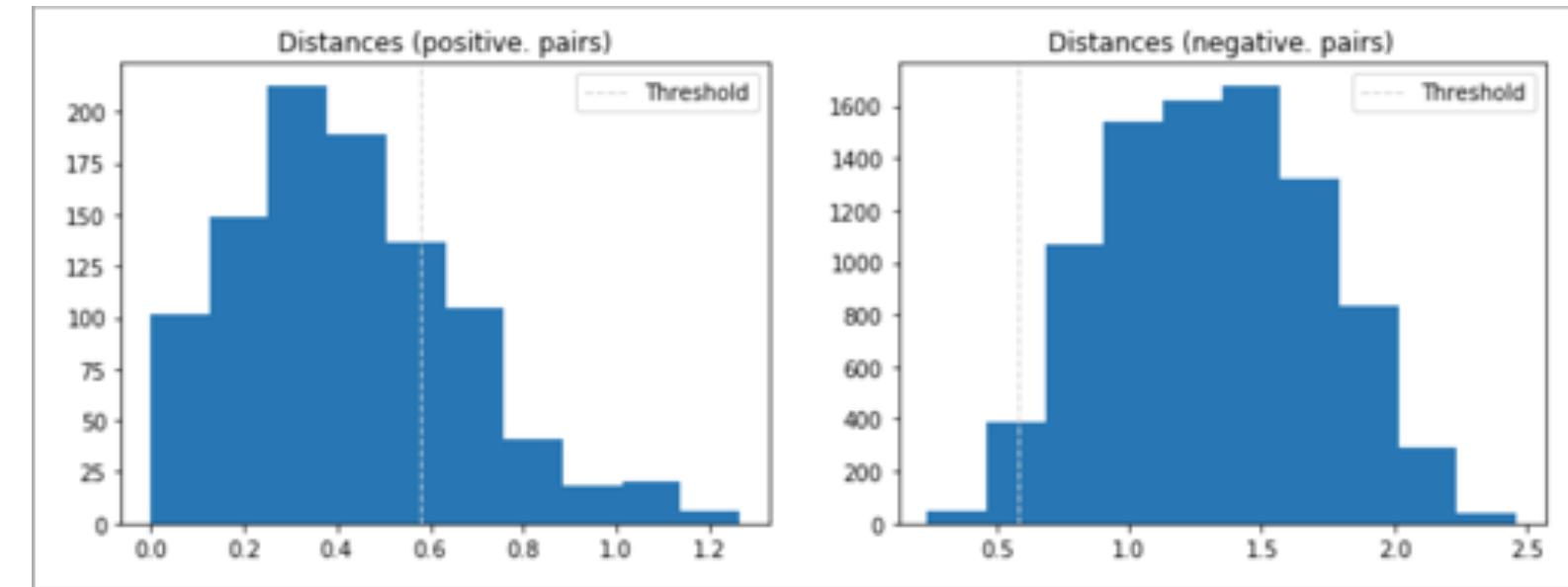
Distance = 2.60



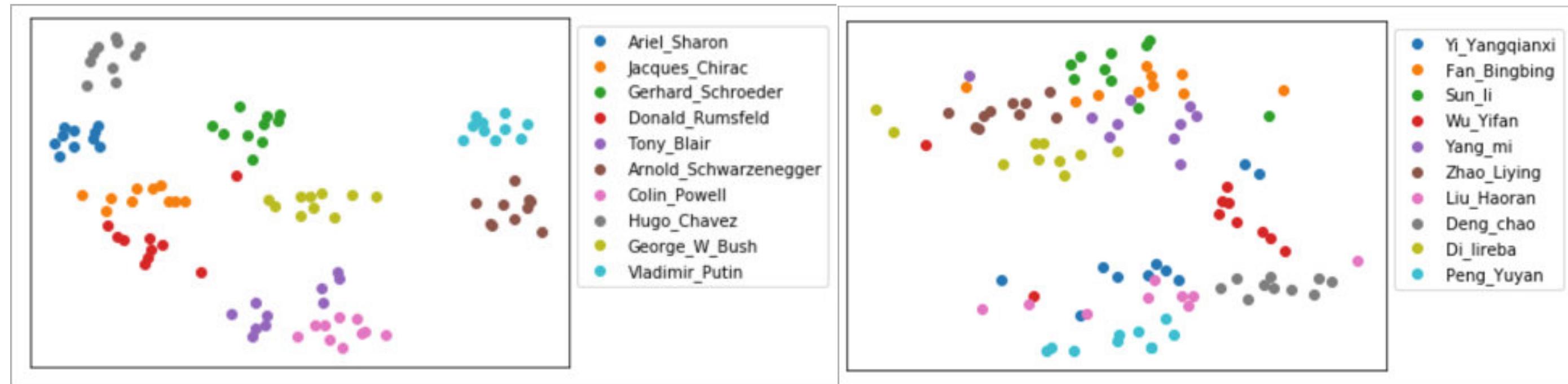
人脸识别对不同人种敏感



人脸识别对不同人种敏感



人脸识别对不同人种敏感



Try it



扫描二维码

试看/购买《TensorFlow 快速入门与实战》视频课程