

人工智能应用实践 开题报告

——人脸检测任务

学号：201720121206

班级：17 级人工智能班

姓名：吕程

目录

1	论文解读	1
1.1	MTCNN.....	1
1.1.1	概要.....	1
1.1.2	总体框架.....	1
1.1.3	NMS——非极大值抑制	2
1.1.4	CNN 网络结构	2
1.1.5	训练.....	3
1.2	FaceNet.....	4
1.2.1	概要.....	4
1.2.2	网络架构.....	5
1.2.3	目标函数.....	5
1.2.4	三元组的选择.....	6
1.2.5	网络模型.....	7
1.3	SRN	7
1.3.1	概要.....	7
1.3.2	网络结构.....	8
2	源码运行	11
2.1	代码来源	11
2.2	下载 LFW 数据集	11
2.3	下载 Google 预训练的网络模型.....	11
2.4	预训练模型准确率测试	12
2.5	比较两张图片的欧氏距离	12
2.6	制作自己的人脸数据库	13
2.6.1	准备数据集.....	13
2.6.2	制作人脸识别库	13
2.7	验证图片	14
2.8	使用摄像头实时识别	15
3	预计进展	16

1 论文解读

1.1 MTCNN

1.1.1 概要

在本文中，提出新的级联架构来整合多任务卷积神经网络学习的问题。该算法有三个阶段组成：第一阶段，浅层的 CNN 快速产生候选窗体；第二阶段，通过更复杂的 CNN 精炼候选窗体，丢弃大量的重叠窗体；第三阶段，使用更加强大的 CNN，实现候选窗体去留，同时显示五个面部关键点定位。

通过这个多任务学习框架，算法的性能可以显著提高。本文的主要贡献总结如下：

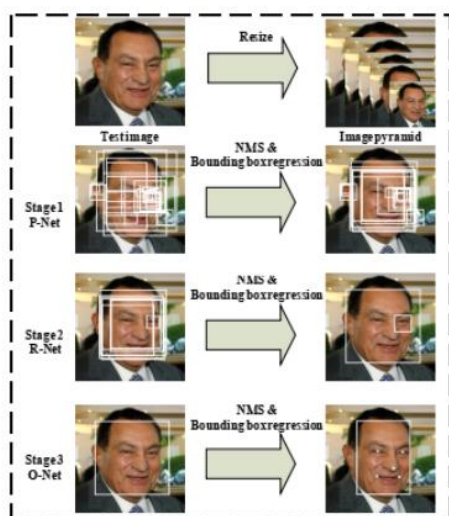
（1）提出了一种新的级联 CNNs 框架，用于联合人脸检测和对齐，并仔细设计了轻量级 CNN 架构以实现实时性能。

（2）提出一种有效的在线难样本挖掘方法来提高性能。

（3）对具有挑战性的基准进行了大量的实验，与在面部检测和面部对准任务中最先进的技术相比，该方法显示出显著的性能改进

1.1.2 总体框架

总体流程如下图所示：



给定一幅图像，我们首先调整它的大小以建立一个图像金字塔，这是以下三级级联框架的输入：

阶段 1：利用完全卷积网络（称为建议网络（P-Net））获得候选面部窗口及其边界框回归向量。然后基于估计的边界框回归向量校准候选窗口。之后，采用非最大抑制（NMS）来合并高度重叠的候选窗口。

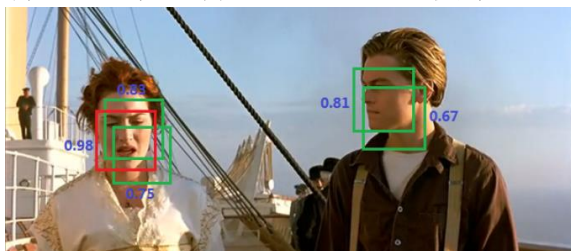
阶段 2：所有候选窗口都被送到另一个叫做 Refine Network（R-Net）的 CNN，它进一步拒绝了大量错误的候选窗口，用边界框回归进行校准，并进行非最大抑制（NMS）。

阶段 3：这个阶段与第二阶段相似，但在这个阶段我们的目标是通过更多的监督来识别人脸区域。特别是，该网络将输出五个面部目标的位置。

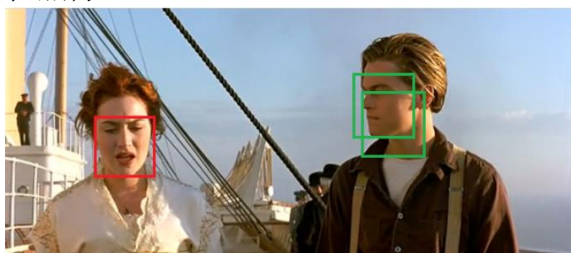
1.1.3 NMS——非极大值抑制

抑制的过程是一个迭代-遍历-消除的过程：

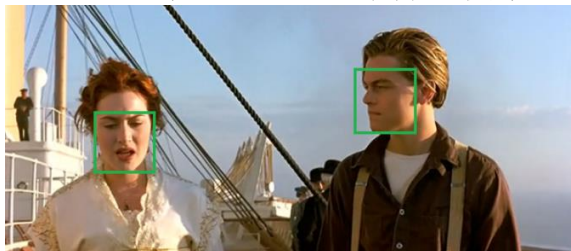
- 1、将所有的框的得分排序，选中最高分及其对应框



- 2、遍历其余的框，如果和当前最高分框的重叠面积（IOU）大于一定的阈值，就将框删除



- 3、从未处理的框中继续选一个得分最高的，重复上述过程



1.1.4 CNN 网络结构

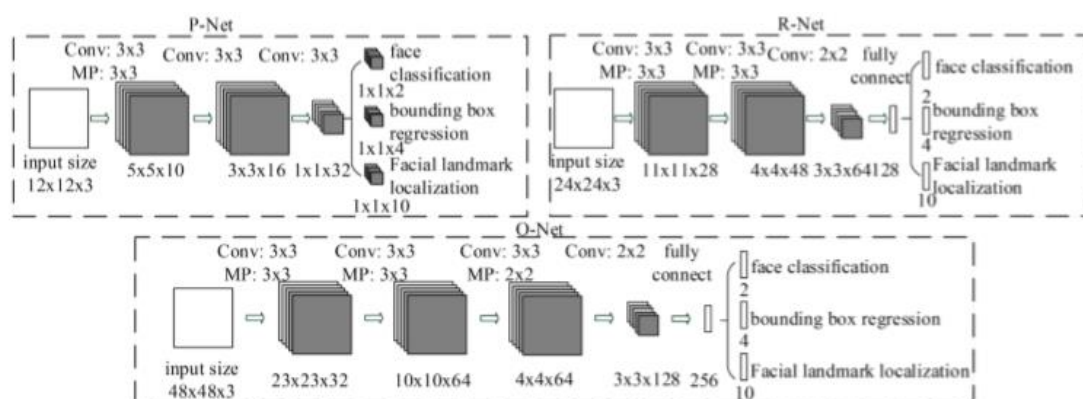
其中，多个 CNN 被设计用于人脸检测。但是，它的性能可能受到以下事实的限制：

(1) 卷积层中的一些滤波器缺少可能限制其区分能力的多样性。

(2) 与其他多类目标检测和分类任务相比，人脸检测是一项具有挑战性的二值分类任务，因此每层滤波器的数量可能会减少。为此，减少滤波器数量并将 5×5 滤波器更改为 3×3 滤波器，以减少计算量，同时增加深度以获得更好的性能。

通过这些改进，与之前的架构相比，可以在更少的运行时间内获得更好的性能（训练阶段的结果下表所示。为了公平比较，在每个组中使用相同的训练和验证数据）。CNN 网络结构下图所示。将 PReLU 应用于卷积和完全连接层（输出层除外）之后作为非线性激活函数

Group	CNN	300 Times Forward	Accuracy
Group1	12-Net [19]	0.038s	94.4%
Group1	P-Net	0.031s	94.6%
Group2	24-Net [19]	0.738s	95.1%
Group2	R-Net	0.458s	95.4%
Group3	48-Net [19]	3.577s	93.2%
Group3	O-Net	1.347s	95.4%



1.1.5 训练

利用三项任务来训练 CNN 检测器：面部/非面部分类，边界框回归和面部标志定位。

1) 脸部分类：学习目标表述为一个两级分类问题。对于每个样本 x_i ，使用交叉熵损失：

$$L_i^{det} = -(y_i^{det} * \log(p_i) + (1 - y_i^{det}) * (1 - \log(p_i)))$$

其中 g_i 是由网络产生的表示样本 x_i 是人脸的概率。符号 $y_i \in \{0,1\}$ 表示样本对应的真实标签。

2) 边界框回归：对于每个候选窗口，我们预测它与最近的真实标签值之间的偏移（即边界框的左边，顶边，高度和宽度）。学习目标被形容成为一个回归问题，使用每个样本 x_i 的欧几里德损失：

$$L_i^{box} = \|\hat{y}_i^{box} - y_i^{box}\|_2^2$$

其中 \hat{y}_i^{box} 是从卷积神经网络中获得的回归目标， y_i^{box} 是样本对应的真实坐标。

该真实坐标有四个坐标值，包括左上角坐标值、高度以及宽度值，因此 $y_i^{box} \in \mathbb{R}^4$ 。

3) 面部关键点定位：与边界框回归任务类似，将面部标志检测表示为回归问题，将欧几里德损失最小化：

$$L_i^{landmark} = \|\hat{y}_i^{landmark} - y_i^{landmark}\|_2^2$$

$\hat{y}_i^{landmark}$ 是从卷积神经网络中获得的面部标志的坐标， $y_i^{landmark}$ 是第 i 个样本对应的真实面部关键点坐标。该真实面部关键点坐标包括左眼、右眼、鼻子、左嘴角和右嘴角等坐标，因此 $y_i^{landmark} \in \mathbb{R}^{10}$ 。

4) 多源训练：由于在每个 CNN 中完成不同的任务，所以在学习过程中存在不同类型的训练图像，如人脸，非人脸和部分对齐人脸。在这种情况下，上面列出的三个损失函数将不会使用。例如，对于背景区域的样本，只计算 L_i^{det} ，另外两个损失设置为 0，这可以直接使用样本类型指标来实现。

5) 在线难样本挖掘：与原始分类器训练完成后进行传统难样本挖掘不同，在适应训练过程的人脸/非人脸分类任务中进行在线难样本挖掘。特别是，在每个小样本批次中，对从所有样本向前传播中计算出的损失进行排序，并选择最高的 70% 作为难样本。然后只计算这些反向传播中难样本的梯度。这意味着忽略了在训练期间加强检测器的帮助不大的简单样本。

1.2 FaceNet

1.2.1 概要

与其他的深度学习方法在人脸上的应用不同，FaceNet 并没有用传统的 softmax 的方式去进行分类学习，然后抽取其中某一层作为特征，而是直接进行端对端学习

一个从图像到欧式空间的编码方法，然后基于这个编码再做人脸识别、人脸验证和人脸聚类等。

FaceNet 算法有如下要点：

去掉了最后的 softmax，而是用元组计算距离的方式来进行模型的训练。使用这种方式学到的图像表示非常紧致，使用 128 位足矣。

元组的选择非常重要，选的好可以很快的收敛。

1.2.2 网络架构

FaceNet 使用深度卷积网络。讨论了两种不同的核心体系结构：Zeiler&Fergus [22]型网络和最近的 Inception [16]型网络。

给定模型细节，并将其视为黑匣子，方法的最重要部分在于整个系统的端到端学习。为此，采用三重损失，该三元组损失直接反映了在面部验证，识别和聚类中要实现的目标。争取一个嵌入 $f(x)$ ，从图像 x 到特征空间 R^d 中，以使相同身份的所有脸部之间的平方距离小，而与成像条件无关，而不同身份的一对图像之间的平方距离大。



Deep Architecture 就是卷积神经网络去掉 softmax 后的结构，经过 L2 的归一化，然后得到特征表示，基于这个特征表示计算三元组损失。

1.2.3 目标函数



所谓的三元组就是三个样例，如(anchor, pos, neg)，其中， x 和 p 是同一类， x 和 n 是不同类。那么学习的过程就是学到一种表示，对于尽可能多的三元组，使得 anchor 和 pos 的距离，小于 anchor 和 neg 的距离。即：

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

变换一下，得到目标函数：

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

目标函数的含义就是对于不满足条件的三元组，进行优化；对于满足条件的三元组，就 pass 先不管

1.2.4 三元组的选择

很少的数据就可以产生很多的三元组，如果三元组选的不得法，那么模型要很久很久才能收敛。因而，三元组的选择特别重要。

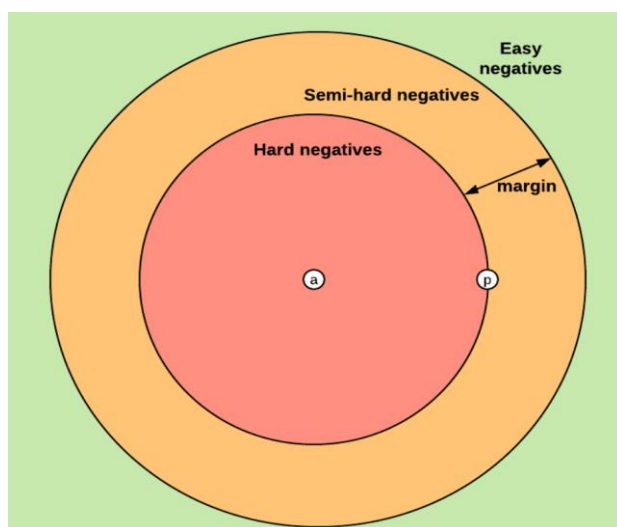
当然最暴力的方法就是对于每个样本，从所有样本中找出离他最近的反例和离它最远的正例，然后进行优化。这种方法有两个弊端：一是耗时，基本上选三元组要比训练还要耗时，二是容易受不好的数据的主导，导致得到的模型会很差。

所以，为了解决上述问题，论文中提出了两种策略：第一种是每 N 步线下在数据的子集上生成一些 triplet，第二种是在线生成 triplet，在每一个 mini-batch 中选择 hard pos/neg 样例。

为了使 mini-batch 中生成的 triplet 合理，生成 mini-batch 的时候，保证每个 mini-batch 中每个人平均有 40 张图片。然后随机加一些反例进去。在生成 triplet 的时候，找出所有的 anchor-pos 对，然后对每个 anchor-pos 对找出其 hard neg 样本。这里，并不是严格的去找 hard 的 anchor-pos 对，找出所有的 anchor-pos 对训练的收敛速度也很快。

除了上述策略外，还可能会选择一些 semi-hard 的样例，所谓的 semi-hard 即不考虑 alpha 因素，即：

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2$$



1.2.5 网络模型

论文使用了两种卷积模型：第一种是 Zeiler&Fergus 架构，22 层，140M 参数，1.6billion FLOPS(FLOPS 是什么？)，称之为 NN1。第二种是 GoogleNet 式的 Inception 模型。模型参数是第一个的 20 分之一，FLOPS 是第一个的五分之一。

基于 Inception 模型，减小模型大小，形成两个小模型：

NNS1：26M 参数，220M FLOPS。

NNS2：4.3M 参数，20M FLOPS。

NN3 与 NN4 和 NN2 结构一样，但输入变小了。

NN2 原始输入： 224×224

NN3 输入： 160×160

NN4 输入： 96×96

其中，NNS 模型可以在手机上运行。

1.3 SRN

1.3.1 概要

本文提出了一种新颖的单镜头人脸检测器，命名为选择性改进网络（SRN），它将新的两步分类和回归操作选择性地引入到基于锚的人脸检测器中，以同时减少误报并提高定位精度。特别是，SRN 由两个模块组成：选择性两步分类（STC）模块和选择性两步回归（STR）模块。STC 的目的是从低水平检测层中滤除大多数简单的负锚，以减少后续分类器的搜索空间，而 STR 则设计用于从高级检测层粗略调整锚

的位置和大小，以便为随后的回归量。此外，还设计了一个感受野增强（RFE）块，以提供更多样化的接收场，这有助于更好地捕捉某些极端姿势的面部。因此，所提出的SRN检测器在所有广泛使用的面部检测基准上实现了最先进的性能，包括AFW，PASCAL 面，FDDB 和 WIDER FACE 数据集。

显著解决两个问题：1、减少 False positive 2、捕获某些极端姿势的面部

文章贡献有三：

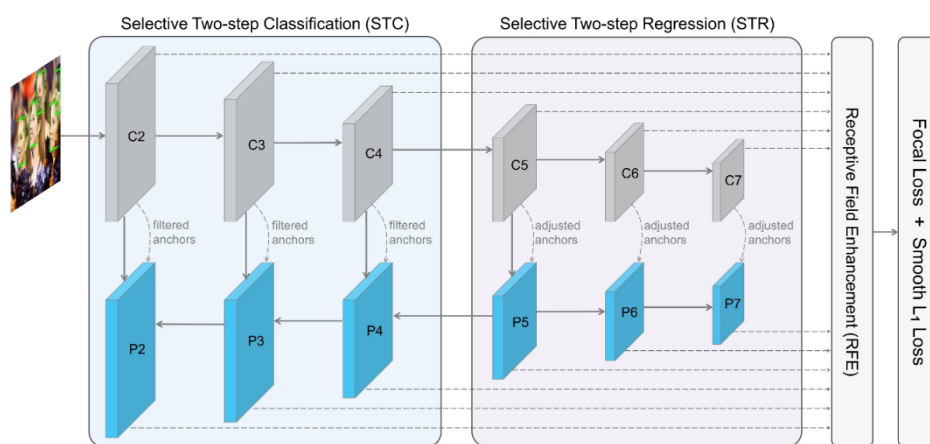
- 提出了一个 STC 模块，用于过滤掉来自低层的大多数简单负样本，以减少分类搜索空间。

- 设计了一个 STR 模块，可以从高级层粗略调整锚点的位置和大小，为后续的回归量提供更好的初始化。

- 引入 RFE 模块，为检测极端姿势面提供更多样化的感受野

1.3.2 网络结构

SRN 的整体框架如图所示：



采用具有 6 级特征金字塔结构的 ResNet-50 作为 SRN 的骨干网络。从这四个残余块中提取的特征图分别表示为 C2，C3，C4 和 C5。在 C5 之后，C6 和 C7 只是由两个简单的下采样 3×3 卷积层提取。自下而上和自上而下通路之间的横向结构。P2，P3，P4 和 P5 是从横向连接中提取的特征图，对应于分别具有相同空间大小的 C2，C3，C4 和 C5，而 P6 和 P7 仅由两个 3×3 卷积层进行下采样。

STC 模块选择 C2，C3，C4，P2，P3 和 P4 进行两步分类，而 STR 模块选择 C5，C6，C7，P5，P6 和 P7 进行两步回归，RFE 模块负责丰富用于预测物体分类和位置的特征的接收领域。

在每个金字塔等级，使用两个特定的锚点尺度（即 $2S$ 和 $2/2S$ ，其中 S 代表每个金字塔等级的总步幅）和一个纵横比（即 1.25）。总的来说，每个级别都有 $A=2$ 锚点，它们覆盖了相对于网络输入图像的各级别的比例范围 8-362 像素。

在深层架构的末尾添加混合损失，利用焦点损失和平滑 L1 损失的优点来推动模型更专注于更难训练示例并学习更好的回归结果

STC

两步分类是一种通过两步网络架构实现的级联分类，其中第一步使用预设的负阈值 0.99 来减少大多数简单的负锚定以减少后续步骤的搜索空间。对于基于锚的人脸检测器，有必要在图像上平铺大量小锚以检测小脸，这导致正样本和负样本之间的极端类不平衡。例如，在具有 1024×1024 输入分辨率的 SRN 结构中，如果在每个锚点处平铺 2 个锚点，则样本总数将达到 300k。其中，阳性样本的数量只有几十个或更少。为了减少分类器的搜索空间，必须进行两步分类以减少误报。

由于平铺在三个较高层（即 P5, P6 和 P7）上的锚仅占 11.1% 并且相关特征更加充分。因此，在这三个更高的金字塔等级中，分类任务相对容易。因此，在三个较高的金字塔等级上应用两步分类是不必要的，并且如果应用，将导致计算成本的增加。相反，三个较低的金字塔等级（即 P2, P3 和 P4）具有绝大多数样本（88.9%）并且缺乏足够的特征。迫切需要这些低金字塔等级进行两步分类，以减轻类不平衡问题并减少后续分类器的搜索空间。

STC 的损失函数为：

$$\mathcal{L}_{\text{STC}}(\{p_i\}, \{q_i\}) = \frac{1}{N_{s_1}} \sum_{i \in \Omega} \mathcal{L}_{\text{FL}}(p_i, l_i^*) + \frac{1}{N_{s_2}} \sum_{i \in \Phi} \mathcal{L}_{\text{FL}}(q_i, l_i^*),$$

STR

在三个较低的金字塔等级中应用两步回归会损害性能，原因有两个：1）三个较低的金字塔等级与大量的小锚相关联，以检测小脸。这些小脸的特征是非常粗糙的特征表示，因此这些小锚非常难以执行两步回归；2）在训练阶段，如果让网络过分关注低金字塔等级的困难回归任务，它将导致更大的回归损失并阻碍更重要的分类任务。

STR 背后的动机是充分利用三个较高金字塔等级上的大面积的详细特征来回归更准确的边界框位置，并使三个较低的金字塔等级更加注重分类任务。这种分而治之的策略使整个框架更加高效。

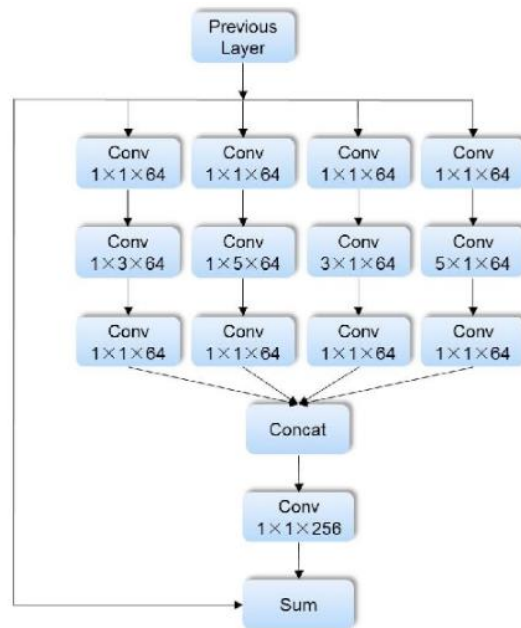
STR 的损失函数也由两部分组成：

$$\mathcal{L}_{\text{STR}}(\{x_i\}, \{t_i\}) = \sum_{i \in \Psi} [l_i^* = 1] \mathcal{L}_r(x_i, g_i^*) + \sum_{i \in \Phi} [l_i^* = 1] \mathcal{L}_r(t_i, g_i^*),$$

RFE

接收场的单一性影响具有不同纵横比的物体的检测。这个问题在人脸检测任务中似乎并不重要，因为在许多数据集中，人脸注释的纵横比约为 1: 1。尽管如此，统计数据显示，WIDER FACE 训练集中有相当一部分面部长宽比大于 2 或小于 0.5。因此，网络的接收场与面部的纵横比之间存在不匹配。

为了解决这个问题，提出了一个名为感知场增强（RFE）的模块，以在预测类和位置之前使特征的接收领域多样化。特别是，RFE 模块取代了类子网中的两个卷积层和 RetinaNet 的子网。RFE 的结构如下图所示。RFE 模块采用四分支结构，其灵感来自 Inception 模块。具体来说，首先，使用 1x1 卷积层将通道数减少到前一层四分之一。其次，使用 1xk 和 kx1 (k=3 和 5) 卷积层来提供矩形接收字段。通过另一个 1x1 卷积层，将来自四个分支的特征映射连接在一起。



2 源码运行

2.1 代码来源

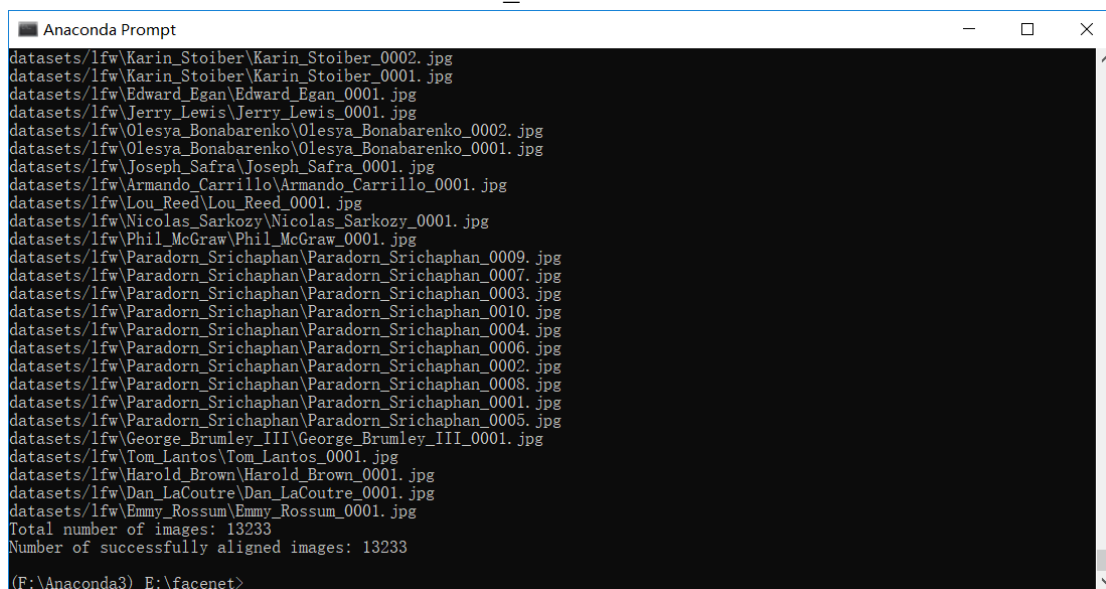
首先下载 FaceNet 源码到本地，然后配置好相应的环境

2.2 下载 LFW 数据集

LFW 是评估人脸识别算法效果的公开测试数据集，共有 13233 张图片，属于 5749 个不同人，其中图片的大小为 250×250 。

对 LFW 图片预处理

LFW 图片原图尺寸为 250×250 ，需要修改图片尺寸，使其大小符合预训练模型的输入尺寸，即 160×160 。在 `src/align/align_dataset_mtcnn.py` 文件里，采用 MTCNN 人脸检测算法对人脸进行检测，进一步人脸对齐，然后再把人脸图片尺寸修改为 160×160 的尺寸，保存到 `datasets/lfw_160` 目录下。



```
Anaconda Prompt
datasets\lfw\Karin_Stoiber\Karin_Stoiber_0002.jpg
datasets\lfw\Karin_Stoiber\Karin_Stoiber_0001.jpg
datasets\lfw\Edward_Egan\Edward_Egan_0001.jpg
datasets\lfw\Jerry_Lewis\Jerry_Lewis_0001.jpg
datasets\lfw\Olesya_Bonabarenko\Olesya_Bonabarenko_0002.jpg
datasets\lfw\Olesya_Bonabarenko\Olesya_Bonabarenko_0001.jpg
datasets\lfw\Joseph_Safra\Joseph_Safra_0001.jpg
datasets\lfw\Armando_Carrillo\Armando_Carrillo_0001.jpg
datasets\lfw\Lou_Reed\Lou_Reed_0001.jpg
datasets\lfw\Nicolas_Sarkozy\Nicolas_Sarkozy_0001.jpg
datasets\lfw\Phil_McGraw\Phil_McGraw_0001.jpg
datasets\lfw\Paradorn_Srichaphan\Paradorn_Srichaphan_0009.jpg
datasets\lfw\Paradorn_Srichaphan\Paradorn_Srichaphan_0007.jpg
datasets\lfw\Paradorn_Srichaphan\Paradorn_Srichaphan_0003.jpg
datasets\lfw\Paradorn_Srichaphan\Paradorn_Srichaphan_0010.jpg
datasets\lfw\Paradorn_Srichaphan\Paradorn_Srichaphan_0004.jpg
datasets\lfw\Paradorn_Srichaphan\Paradorn_Srichaphan_0006.jpg
datasets\lfw\Paradorn_Srichaphan\Paradorn_Srichaphan_0002.jpg
datasets\lfw\Paradorn_Srichaphan\Paradorn_Srichaphan_0008.jpg
datasets\lfw\Paradorn_Srichaphan\Paradorn_Srichaphan_0001.jpg
datasets\lfw\Paradorn_Srichaphan\Paradorn_Srichaphan_0005.jpg
datasets\lfw\George_Brumley_III\George_Brumley_III_0001.jpg
datasets\lfw\Tom_Lantos\Tom_Lantos_0001.jpg
datasets\lfw\Harold_Brown\Harold_Brown_0001.jpg
datasets\lfw\Dan_LaCoutre\Dan_LaCoutre_0001.jpg
datasets\lfw\Emmy_Rossum\Emmy_Rossum_0001.jpg
Total number of images: 13233
Number of successfully aligned images: 13233
(F:\Anaconda3) E:\facenet>
```

2.3 下载 Google 预训练的网络模型

官网提供了两个模型，这两个模型都是基于 Inception ResNet V1 网络架构下进行识别的，两个模型基于不同的训练集进行训练，这里我下载的是 VGGFace2 数据集的模型

Model name	LFW accuracy	Training dataset	Architecture
20180408-102900	0.9905	CASIA-WebFace	Inception ResNet v1
20180402-114759	0.9965	VGGFace2	Inception ResNet v1

2.4 预训练模型准确率测试

可以看到，准确度达到了 0.98500+-0.00658

```
C:\Windows\system32\cmd.exe
tensorflow.python.training.checkpoint_management) is deprecated and will be removed in a future version.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
WARNING:tensorflow:From src\validate_on_lfw.py:79: start_queue_runners (from tensorflow.python.training.queue_runner_im
l) is deprecated and will be removed in a future version.
Instructions for updating:
To construct input pipelines, use the 'tf.data' module.
Running forward pass on LFW images
.....
Accuracy: 0.98500+-0.00658
Validation rate: 0.90100+-0.02395 @ FAR=0.00067
Area Under Curve (AUC): 0.998
Equal Error Rate (EER): 0.016
2019-10-22 10:45:53.877650: W tensorflow/core/kernels/queue_base.cc:285] _0_FIFOQueueV2: Skipping cancelled dequeue atte
mpt with queue not closed
2019-10-22 10:45:53.883906: W tensorflow/core/kernels/queue_base.cc:285] _4_FIFOQueueV2_1: Skipping cancelled dequeue at
tempt with queue not closed
2019-10-22 10:45:53.891250: W tensorflow/core/kernels/queue_base.cc:277] _2_input_producer: Skipping cancelled enqueue a
ttempt with queue not closed
2019-10-22 10:45:53.896766: W tensorflow/core/kernels/queue_base.cc:285] _4_FIFOQueueV2_1: Skipping cancelled dequeue at
tempt with queue not closed
2019-10-22 10:45:53.901717: W tensorflow/core/kernels/queue_base.cc:285] _4_FIFOQueueV2_1: Skipping cancelled dequeue at
tempt with queue not closed
2019-10-22 10:45:53.908774: W tensorflow/core/kernels/queue_base.cc:285] _4_FIFOQueueV2_1: Skipping cancelled dequeue at
tempt with queue not closed
2019-10-22 10:45:53.914144: W tensorflow/core/kernels/queue_base.cc:285] _0_FIFOQueueV2: Skipping cancelled dequeue atte
mpt with queue not closed
2019-10-22 10:45:53.919339: W tensorflow/core/kernels/queue_base.cc:285] _0_FIFOQueueV2: Skipping cancelled dequeue atte
mpt with queue not closed
2019-10-22 10:45:53.926031: W tensorflow/core/kernels/queue_base.cc:285] _0_FIFOQueueV2: Skipping cancelled dequeue atte
```

2.5 比较两张图片的欧氏距离

使用欧氏距离来衡量两张人脸的相似程度，用来判别这两张图片是否为同一个人，；两张人脸图片越相似，空间距离越小，差别越大，则空间距离越大

```
C:\Windows\system32\cmd.exe
mpt with queue not closed
E:\facenet>python src\compare.py src\models\20180402-114759\20180402-114759.pb data\images\Anthony_Hopkins_0001.jpg data
/images/Anthony_Hopkins_0002.jpg --gpu_memory_fraction 0.25
Creating networks and loading parameters
2019-10-22 10:57:06.470592: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that thi
s TensorFlow binary was not compiled to use: AVX2
WARNING:tensorflow:From F:\Anaconda3\lib\site-packages\tensorflow\python\framework\op_def_library.py:263: colocate_with
(from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From E:\facenet\src\align\detect_face.py:213: div (from tensorflow.python.ops.math_ops) is deprecated
and will be removed in a future version.
Instructions for updating:
Deprecated in favor of operator or tf.math.divide.
Model filename: src\models\20180402-114759\20180402-114759.pb
WARNING:tensorflow:From E:\facenet\src\facenet.py:371: FastGFile.__init__ (from tensorflow.python.platform.gfile) is dep
recated and will be removed in a future version.
Instructions for updating:
Use tf.gfile.GFile.
Images:
0: data\images\Anthony_Hopkins_0001.jpg
1: data\images\Anthony_Hopkins_0002.jpg
Distance matrix
  0      1
0  0.0000  0.8396
1  0.8396  0.0000
E:\facenet>
```


2.6 制作自己的人脸数据库

2.6.1 准备数据集

通过百度图片爬取一些明星图片，然后上传了我自己的一些图片，构成一个小的数据集



再通过 `align_dataset_mtcnn.py` 进行人脸检测，并把人脸图片尺寸设定为 160×160 ，数据集里面的人脸图片数据集并不是可靠的，从网上下载的图片可能有些图片质量较差，或者有多张人脸的，然后导致人脸检测失败，所以还需要检测数据集里面的人脸图片，把不合格的图片给删掉。

2.6.2 制作人脸识别库

训练一个人脸识别库，生成 `.pkl` 文件，在生成 `.pkl` 文件的时候用到 `calssifier.py` 文件，里面用 `SVM` 来训练一个分类器。整体流程大致是：`CNN forward` 输出后经 `L2` 传入 `Embedding` 层，得到 `embedding output` 的特征进行传给 `SVM classifier` 来训练一个分类器。然后把训练好的分类器保存为 `pickle` 文件。在执行指令读取分类器的时候加载 `SVM` 分类器模型，自己所用的测试图片数据就会 `SVM` 分类器模型中的类别做对比判断。`.pkl` 文件保存的参数包括模型和类别，是可以直接读出来的，代码里面 `SVM` 有两种模式：`TRAIN`（用来训练 `SVM` 模型）和 `CLASSIFY`（用来加载 `SVM` 模型）。看代码会有更好的理解

然后使用 `.pkl` 文件来验证图片，看看效果：

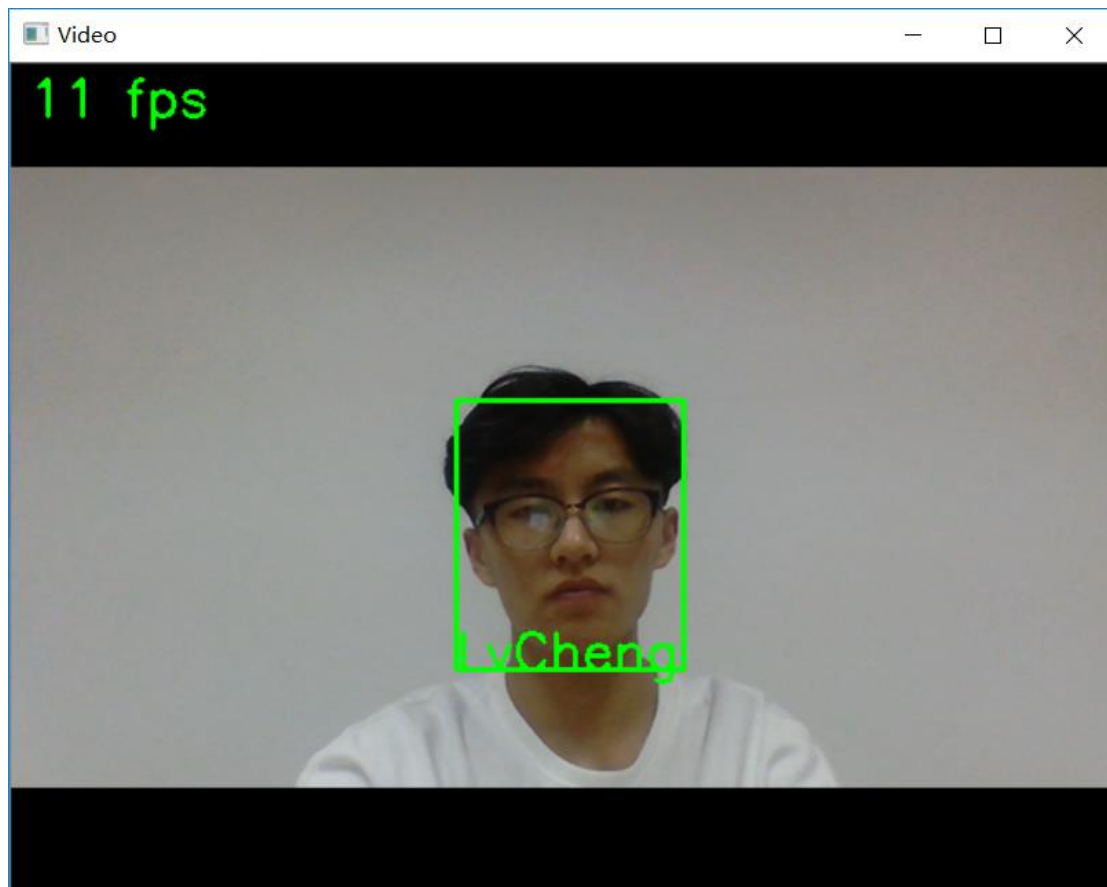
```
C:\Windows\system32\cmd.exe
154 yangmi: 0.841
155 yangmi: 0.835
156 zhaoliying: 0.877
157 zhaoliying: 0.790
158 zhaoliying: 0.685
159 zhaoliying: 0.376
160 zhaoliying: 0.698
161 zhaoliying: 0.875
162 zhaoliying: 0.898
163 zhaoliying: 0.887
164 zhaoliying: 0.800
165 zhaoliying: 0.815
166 zhaoliying: 0.842
167 zhaoliying: 0.733
168 zhaoliying: 0.816
169 zhaoliying: 0.687
170 zhaoliying: 0.658
171 zhaoliying: 0.770
172 zhaoliying: 0.845
173 zhaoliying: 0.740
174 zhaoliying: 0.573
175 zhaoliying: 0.579
176 zhaoliying: 0.890
177 zhaoliying: 0.907
178 zhaoliying: 0.851
179 zhaoliying: 0.824
180 zhaoliying: 0.786
Accuracy: 0.978
E:\facenet>
```

2.7 验证图片

网上下载一张图片，然后用 `predict.py` 代码来测试，结果显示能够正确识别



2.8 使用摄像头实时识别



模型显示可以准确识别！

3 预计进展

对代码理解的更加透彻，并且争取能够在此基础上加上一些自己的东西，人脸检测是我比较喜欢的一个方向，预计中期报告之后读更多的相关论文，争取能做一些改进，对模型加深印象，感觉目前还是了解的比较少，并没有深层次的了解网络架构，而且目前只读了两三篇论文，应该更深入地去学习人脸检测方面的东西，争取能自己实现比较好的变脸的效果。