《软件技术基础》项目文档

> © 李浩东 3190104890 《基于卷积神经网络（CNN）的人脸识别》
>
> 浙江大学 控制科学与工程学院

文档目录

- 引言
- 算法介绍
- 数据集
- CNN建模
- 模型评估
- 项目开发时间线

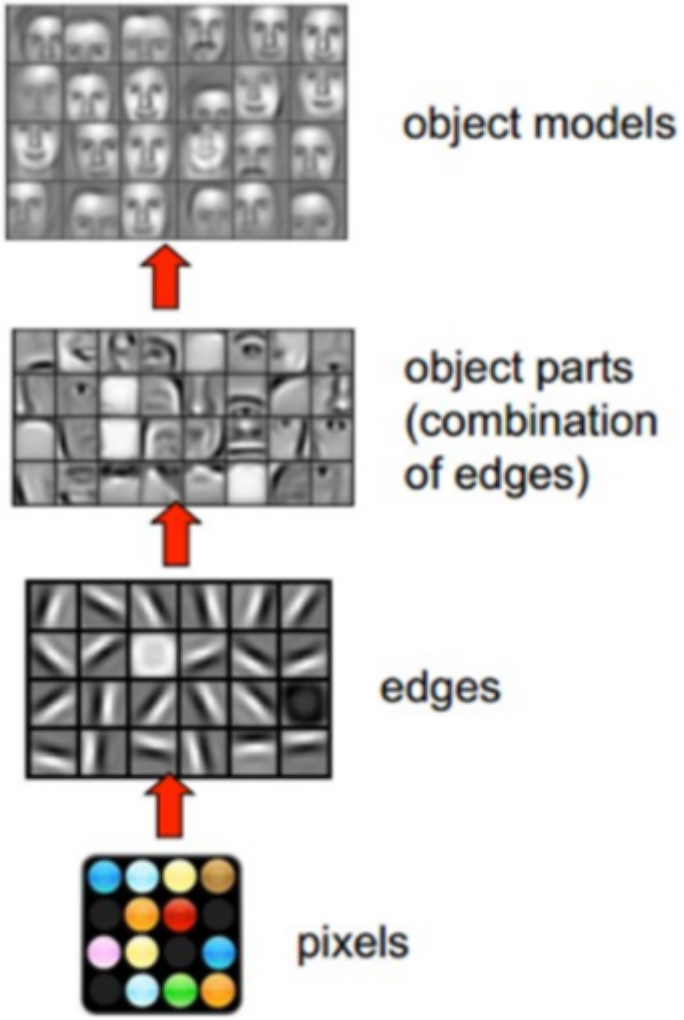
引言

- 脸部辨识系统（Facial recognition system）
    - 脸部辨识系统（Facial recognition system），又称人脸识别。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。
- 背景分析
    - 人脸识别系统的研究始于20世纪60年代，80年代后随着计算机技术和光学成像技术的发展得到提高，而真正进入初级的应用阶段则在90年后期，并且以美国、德国和日本的技术实现为主；人脸识别系统成功的关键在于是否拥有尖端的核心算法，并使识别结果具有实用化的识别率和识别速度；“人脸识别系统”集成了人工智能、机器识别、机器学习、模型理论、专家系统、视频图像处理等多种专业技术，同时需结合中间值处理的理论与实现，是生物特征识别的最新应用，其核心技术的实现，展现了弱人工智能向强人工智能的转化。
    - 截至2017年底，中国已在新疆部署了人工智能脸部辨识系统。访问该地区的记者发现，在几个城市每百米左右安装一监控摄像头。联邦调查局亦未经授权擅自扫描数百万张民众驾照。2019年11月，全球首个人脸识别导航智能停车场于中国广州K11启用，当车主接受人脸注册后，停车场可以提供车位实景导航服务，方便车主取车。
- 可行性
    - 在开发该人脸识别软件之前，我们查询了前人所写过的诸多论文以及源程序，在开发之时，结合了资料中的算法并揉进了自己的一些思想，使程序可以对人脸图片进行简易识别。
    - 技术可行性
        - 图像的处理方法很多，我们可以根据需要，有选择地使用各种方法。
          在确定脸部区域上，通常使用的方法有肤色提取。肤色提取，则对脸部区域的获取则比较准确，成功率达到95%以上，并且速度快，减少很多工作。
          图像的亮度变化，由于图像的亮度在不同环境的当中，必然受到不同光线的影响，图像

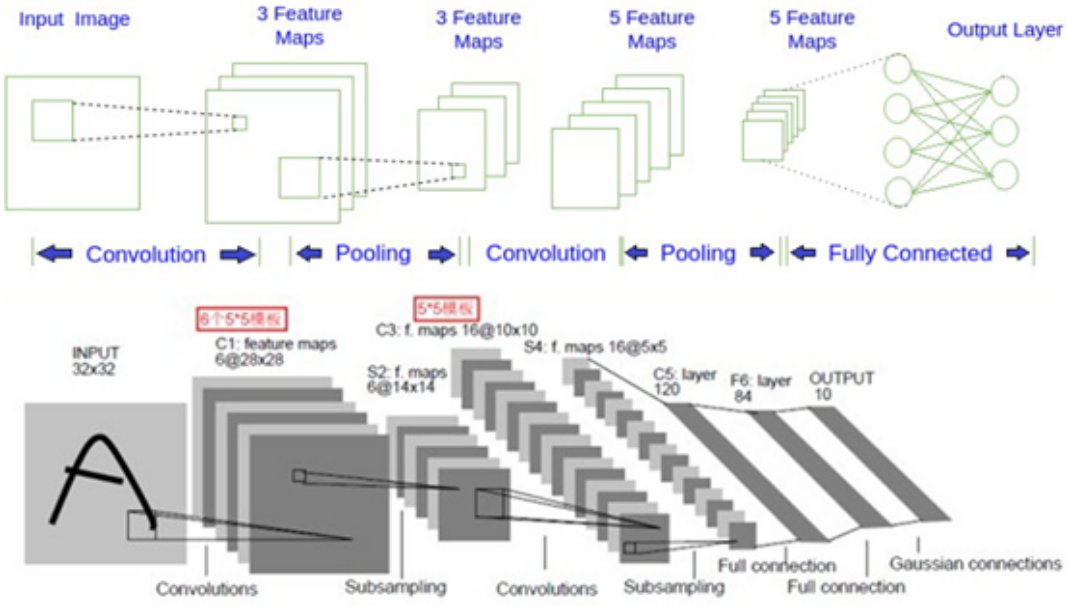
就变得太暗或太亮，我们就要对它的亮度进行调整，主要采取的措施是对图像进行光线补偿。
        - 高斯平滑：在图像的采集过程中,由于各种因素的影响,图像中往往会出现一些不规则的随机噪声,如数据在传输、存储时发生的数据丢失和损坏等,这些都会影响图像的质量，因此需要将图片进行平滑操作以此来消除噪声。
        - 灰度变换：进行灰度处理，我们要保证图像信息尽可能少的丢失。同样在进行灰度变换前，我们也要对图像的信息进行统计，找出一个比较合理的灰度值，才能进行灰度变换。
        - 灰度均衡：灰度变换后，就要进行灰度均衡，可以根据灰度分布来进行灰度均衡。
        - 对比度增强：将所要处理的区域和周围图像区域进一步拉开他们的对比度，使它们更加明显，主要通过像素的聚集来实现。
    - 操作可行性
        - 该人脸识别软件需要如下的运行环境：CPU：500M及以上；内存：64 M及以上。安装有Windows 98、Windows Me、Windows 2000、Windows NT等操作系统中的其中一种。另还装有摄像头可进行随机拍照和识别。
        - 因此，从操作可行性来看，只要系统用户的硬件软件设备满足以上条件，即可用该人脸识别软件进行人脸的识别。
- 需求分析
    - 人脸识别的优势在于其自然性和不被测个体察觉的特点。
    - 所谓自然性，是指该识别方式同人类（甚至其他生物）进行个体识别时所利用的生物特征相同。例如人脸识别，人类也是通过观察比较人脸区分和确认身份的，另外具有自然性的识别还有语音识别、体形识别等，而指纹识别、虹膜识别等都不具有自然性，因为人类或者其他生物并不通过此类生物特征区别个体。
    - 不被察觉的特点对于一种识别方法也很重要，这会使该识别方法不令人反感，并且因为不容易引起人的注意而不容易被欺骗。人脸识别具有这方面的特点，它完全利用可见光获取人脸图像信息，而不同于指纹识别或者虹膜识别，需要利用电子压力传感器采集指纹，或者利用红外线采集虹膜图像，这些特殊的采集方式很容易被人察觉，从而更有可能被伪装欺骗。
- 实际应用
    - 门禁系统：受安全保护的地区可以通过人脸识别辨识试图进入者的身份，比如监狱、看守所、小区、学校等。
    - 摄像监视系统：在例如银行、机场、体育场、商场、超级市场等公共场所对人群进行监视，以达到身份识别的目的。例如在机场安装监视系统以防止恐怖分子登机。
    - 网络应用：利用人脸识别辅助信用卡网络支付，以防止非信用卡的拥有者使用信用卡，社保支付防止冒领等。
    - 学生考勤系统：香港及澳门的中、小学已开始将智能卡配合人脸识别来为学生进行每天的出席点名记录。
    - 相机：新型的数码相机已内建人脸识别功能以辅助拍摄人物时对焦。
      智能手机：解锁手机、识别使用者，如Android 4.0以上，iPhone X。
    - 人证核验一体机：核验持证人和证件照是不是同一个人，主要用在酒店前台、税务局、医院等

算法分析

- 深度学习与视觉原理
    - 深度学习是指多层神经网络上运用各种机器学习算法解决图像，文本等各种问题的算法集合。
    - 深度学习的核心是特征学习，旨在通过分层网络获取分层次的特征信息，从而解决以往需要人工设计特征的重要难题。
    - 人类的视觉原理如下：从原始信号摄入开始（瞳孔摄入像素 Pixels），接着做初步处理（大脑皮层某些细胞发现边缘和方向），然后抽象（大脑判定，眼前的物体的形状，是圆形的），然后进一步抽象（大脑进一步判定该物体是只气球）。



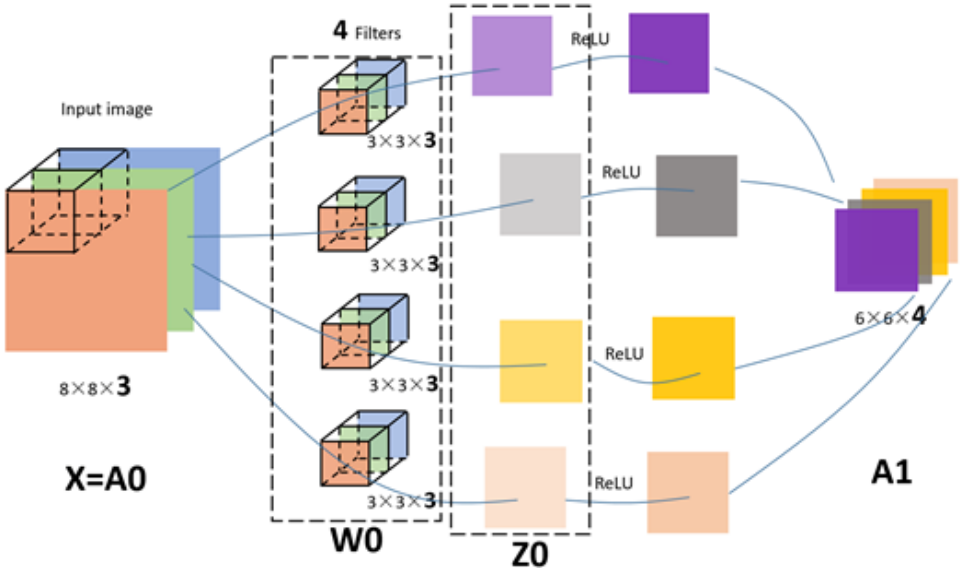
- 卷积神经网络（CNN）
  - 卷积神经网络是一种多层神经网络，擅长处理图像特别是大图像的相关机器学习问题。CNN通过卷积来模拟特征区分，并且通过卷积的权值共享及池化，来降低网络参数的数量级，最后通过传统神经网络完成分类等任务。



  - 输入层：
    - 该层要做的处理主要是对原始图像数据进行预处理，其中包括去均值与归一化等。
    - 去均值即把输入数据各个维度都中心化为0，其目的就是把样本的中心拉回到坐标系原点上。

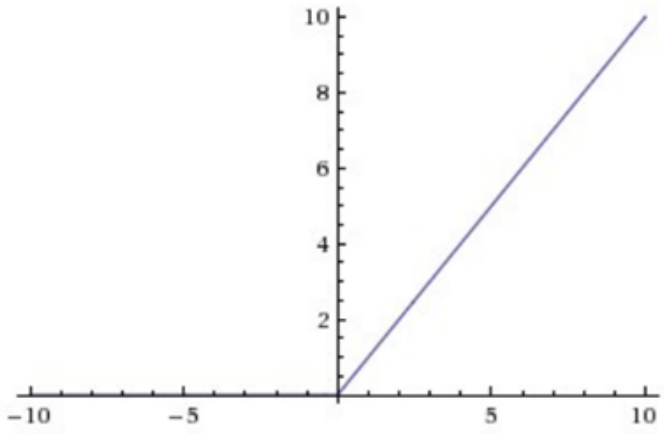
- 归一化是指幅度归一化到同样的范围，即减少各维度数据取值范围的差异而带来的干扰。

- 卷积层：
  - 这一层就是卷积神经网络最重要的一个层次，也是“卷积神经网络”的名字来源。在这个卷积层，有两个关键操作：局部关联与窗口滑动。局部关联将每个神经元看作一个滤波器，窗口滑动则是使用滤波器对局部数据计算。



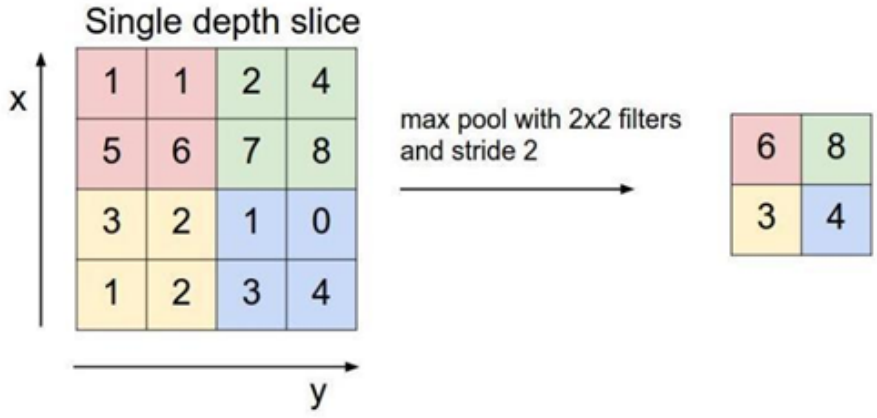
  - 如图，一个三通道的输入在四个神经元（滤波器）求卷积后得到的结果经过激励函数之后就能得到输出，每个滤波器的卷积结果都变成了A1中的一层。对于多通道的输入，我们的卷积核也是多通道的。通过卷积将一维的输入变为四维，每一维代表一个特征。卷积层对神经网络的意义在于提取特征。通过设计不同的卷积核我们可以对输入提取不同的特征，便于后期模型的训练。

- 激励层：
  - 激励层通过激活函数将卷积层的输出结果作非线性映射。这样做的好处在于能够有效增强卷积神经网络的非线性，进而使得模型更加贴近实际。这也是从生物神经元的研究中得到的启发。常用的激励函数有RELU，Maxout，tanh等。



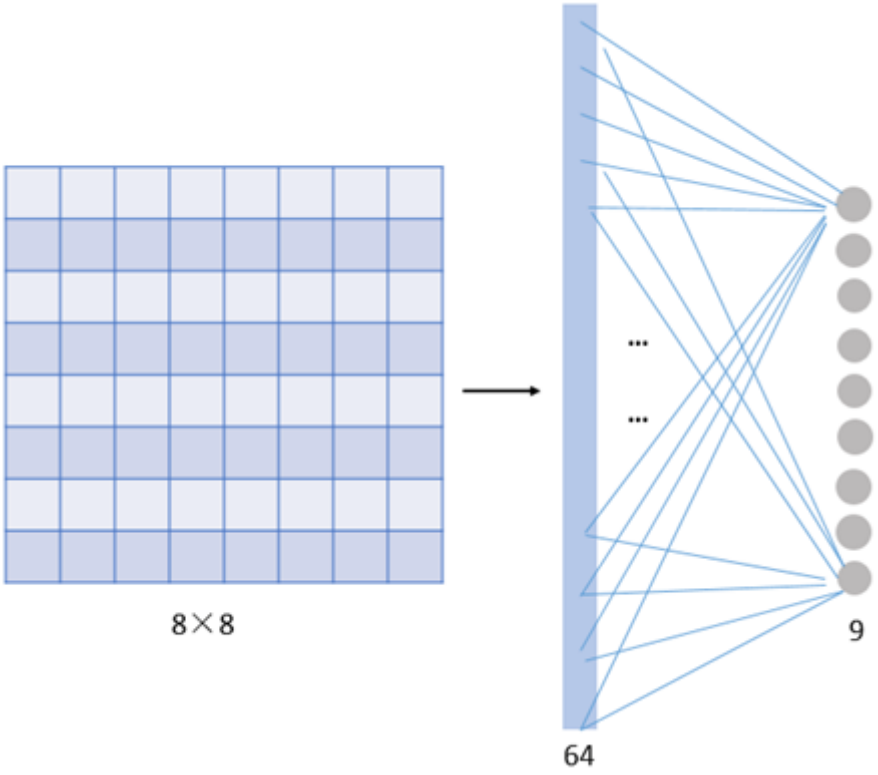
  - 图为RELU激活函数。这也是最常用的激活函数，特点是收敛快，求梯度简单，但较脆弱。当输入小于0时，激活函数输出为0；当输入大于0时，激活函数输出为本身。

- 池化层：
  - 池化层位于连续的卷积层中间，本身没有可训练的参数。池化层的目的在于对卷积层提取的特征进行降维，减少整个神经网络的参数。池化层主要依赖于特征不变性，降维时去掉的特征只是一些无关紧要的特征，而留下的则是具有尺度不变性的特征，是最能表达图像的特征。常用的池化方法有最大池、平均池，加和池等。



- 如图是一个最大池，通过2x2步长为2的过滤器将输入矩阵中对应区域的最大值找出来，可以得到降维后的矩阵。

- 全连接层：
  - 全连接层将两层之间所有神经元都有权重连接，通常全连接层在卷积神经网络尾部，与传统的神经网络神经元的连接方式是一样的。

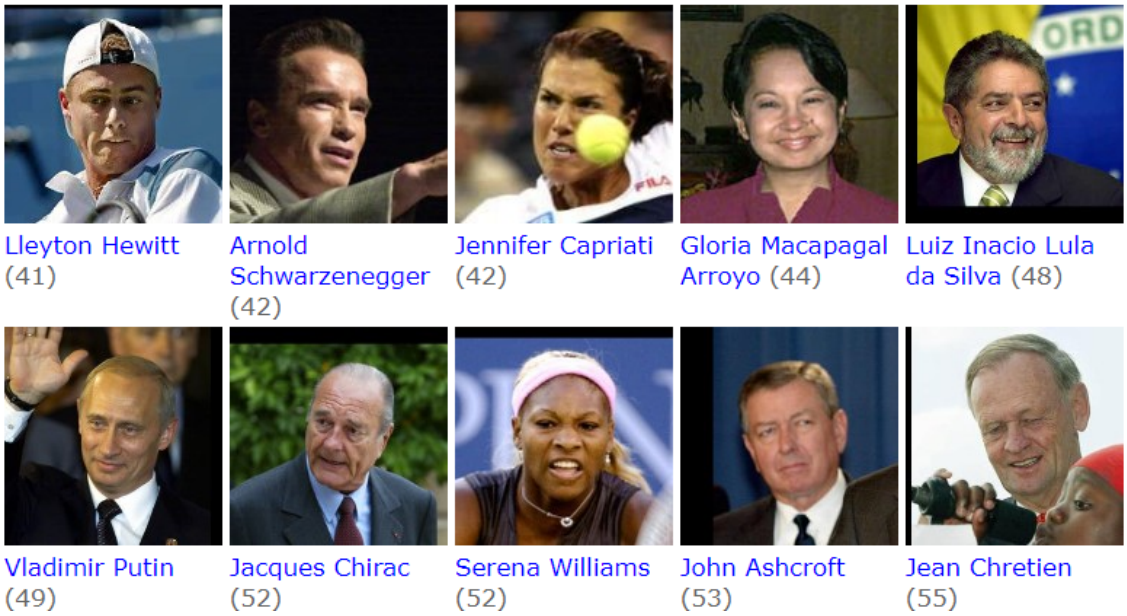


  - 经过几轮卷积和池化操作，可以认为图像中的信息已经被抽象成了信息含量更高的特征。我们可以将卷积和池化看成自动图像提取的过程，在特征提取完成后，仍然需要使用全连接层来完成分类任务。

- 基于人脸的生物识别
  - 物体定位：预测包含主要人脸的图像区域，以便识别区域中的人脸。
  - 物体识别：针对分割好的目标进行分类。
  - 目标分割：该模块主要是将处理后的人脸图片进行分割，将眼睛、鼻子、嘴巴标记出来，以便进行特征提取。
  - 关键点检测：从图像中检测目标人脸上某关键点的位置，例如眼睛、鼻子、嘴巴等。进行特征值检测，并与后台数据库中的值进行比较来完成识别功能。

数据集

- 下载链接：http://vis-www.cs.umass.edu/lfw/lfw.tgz
- 为了达到人脸识别的效果而不停留在人脸检测，同时适当减少运算时间，笔者决定仅选用lfw.tgz中的数据量较为充足的十个人的人脸图像数据进行训练，最终训练效果的评判标准为对于测试集数据的识别准确度，他们是：



- 训练集与测试集
  - 笔者采用`x_train, x_test, y_train, y_test`构建测试集与训练集
    - `x_train`存放用于训练的数据（人脸图片）的路径
    - `y_train`存放所有训练数据的label（即不同图片对应的人名）
    - `x_test`存放用于测试训练结果的数据的路径
    - `y_test`存放所有测试数据的label
  - 源码如下：

```
pathUnzipChoose = "./data/unzip/mine"
faceInfoMatch = [] # 人名对
facePathMatch = [] # 路径对
idNum = [] # ID

def readImage():

    for i in range(10):
        idNum.append(str(i))

    files = os.listdir(pathUnzipChoose) # 遍历文件夹
    index = 0
    for fileName in files:
        faceInfoMatch.append([str(index),fileName]) # 姓名与数字相匹配
        tempDir = pathUnzipChoose + "/" + fileName
        tempfiles = os.listdir(tempDir) # 遍历
        for tempFileNames in tempfiles:
            tempFilePath = pathUnzipChoose + "/" + fileName + "/" + tempFileNames
            facePathMatch.append([idNum[index], tempFilePath]) # 路径与数字（即姓名）匹配
        index += 1

    imageIndex = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0] # 记录每个数据集中的图片数量
```

```
23
24      x_train = []
25      y_train = []
26      x_test = []
27      y_test = []
28
29      for x in facePathMatch:
30          nameIndex = int(x[0])
31          if imageIndex[nameIndex] < 37: # 1~36 为训练集数据
32              x_train.append(x[1]) # 存放路径
33              y_train.append(idNum[nameIndex]) # 存放路径对应照片的
    label（姓名）
34              imageIndex[nameIndex] += 1 # 数量+1
35          else:
36              x_test.append(x[1]) # 37及以后视为测试及数据
37              y_test.append(idNum[nameIndex]) # 存放路径对应照片的
    label（姓名）
38      print("image data x_train, y_train, x_test, y_test done!")
39      return x_train, y_train, x_test, y_test
```

- 对于每一个数据集，笔者将1~36作为训练集，将剩下的（37及以后）作为测试集

- 训练集与测试集的图片读取，源码如下：

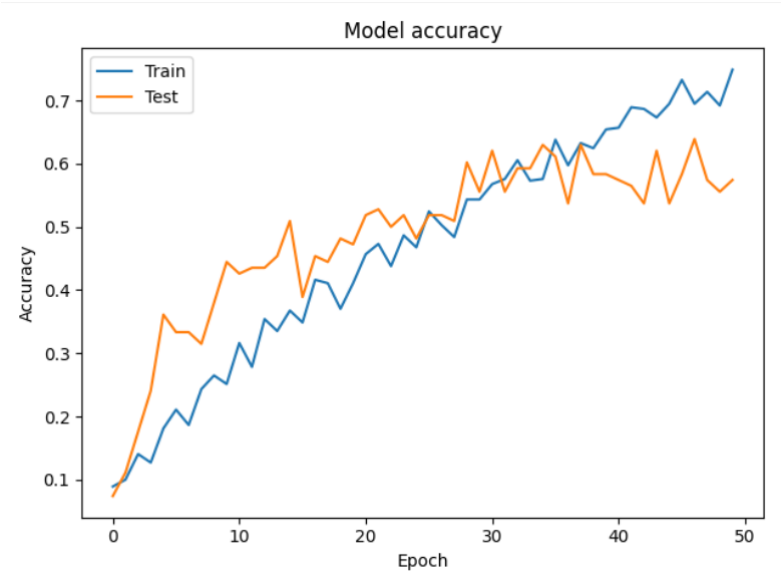
```
1  # 根据路径读入所有的图片，同时根据全局规定的图片尺寸要求进行裁剪,并按照模型输入要求
   reshape
2  def returnImage(x_train, x_test):
3      X_train = []
4      for i in range(len(x_train)):
5          printStringTrain = "x_train reading image data, current
   position: " + str(i)
6          print(printStringTrain)
7          X_train.append(preprocess(mp.imread(x_train[i]), img_h, img_w))
   # 读取图片，preprocess为图像预处理
8      X_train = np.array(X_train) # 数组化处理
9      print("X_train.shape is:  ", X_train.shape) # (370,250,250)
10     X_train = X_train.reshape(X_train.shape[0], X_train[0].shape[0],
   X_train[0].shape[1], channel).astype('float32')
11     # 与模型相匹配啦!
12     # 需要reshape一下下!
13
14     X_test = [] # 同理
15     for i in range(len(x_test)):
16         printStringTest = "x_test reading image data, current position:
   " + str(i)
17         print(printStringTest)
18         X_test.append(preprocess(mp.imread(x_test[i]),img_h, img_w))
19     X_test = np.array(X_test)
20     X_test =
   X_test.reshape(X_test.shape[0],X_test[0].shape[0],X_test[0].shape[1],cha
   nnel).astype('float32')
21     print("image read done!")
22     return X_train, X_test
```

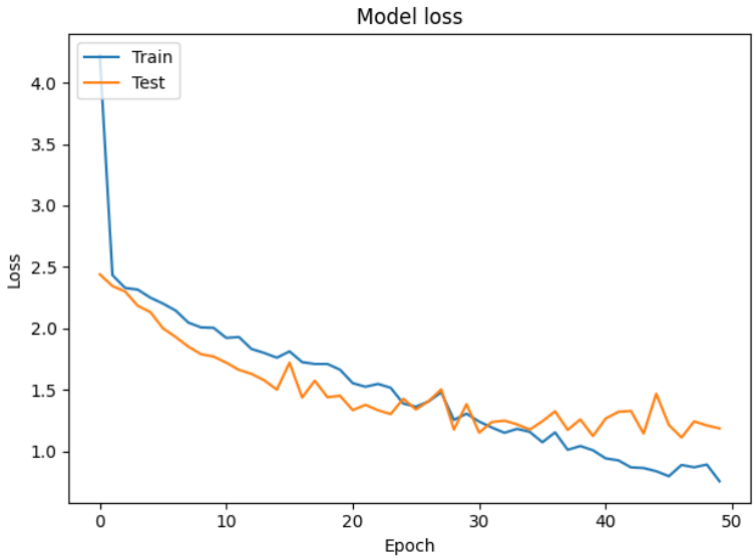
CNN建模

```
    # 2. 定义模型结构
    # 迭代次数：第一次设置为30，后为了优化训练效果更改为100，后改为50
    i = 50 # spoch number
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(5, 5), activation='relu', input_shape=(img_h, img_w, channel))) # 卷积
    model.add(MaxPool2D(pool_size=(2, 2), strides=(2, 2))) # 池化
    model.add(Conv2D(64, kernel_size=(5, 5), activation='relu'))
    model.add(MaxPool2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Conv2D(128, kernel_size=(5, 5), activation='relu'))
    model.add(MaxPool2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Conv2D(256, kernel_size=(5, 5), activation='relu'))
    model.add(MaxPool2D(pool_size=(2, 2), strides=(2, 2)))
    '''
    model.add(Conv2D(512, kernel_size=(5,5), activation='relu'))
    model.add(MaxPool2D(pool_size=(3,3), strides=(2,2)))
    model.add(Dropout(0.2))'''

    model.add(Flatten())
    model.add(Dense(1000, activation='relu')) # 全连接
    model.add(Dropout(0.5))
    model.add(Dense(classes, activation='softmax'))
```

模型评估

- 模型评估：由图可得，收敛效果明显，训练达到预期！





项目开发时间线

2021.1.8

- 实现tgz文件解压缩
  - 下载链接：http://vis-www.cs.umass.edu/lfw/lfw.tgz
- 初步了解人脸识别流程：
  - 解压缩
  - 信息整理（训练集与测试集）
  - 读取数据（训练集与测试集）
  - 搭建神经网络
  - 训练
  - 测试
- 主要流程的代码体现

```
def main(): # 主函数
    unzip() # 解压缩
    docuFaceInfo() # 记录信息
    x_train, y_train, x_test, y_test = readImage() # 测试集与训练集的分类整理（路径与对应人名编号）
    x_train_img, x_test_img = returnImage(x_train, x_test) # 图片读取
    trainModel(x_train_img, y_train, x_test_img, y_test) # CNN训练

# 调用函数
if __name__ == '__main__':
    main()
```

- 实现数据集信息记录`faceInfo.txt`

2021.1.9

- 安装`tensorflow`以及`keras`库：
  - 第一次安装失败，因为python版本为32bit

- 第二次安装成功，但是由于操作失误，python直接安装在F盘（而不是F盘的某一个特定文件夹下），于是重装python，之后`pip search tensorflow`以及`pip search keras`均没有结果，于是重新安装，语句如下：

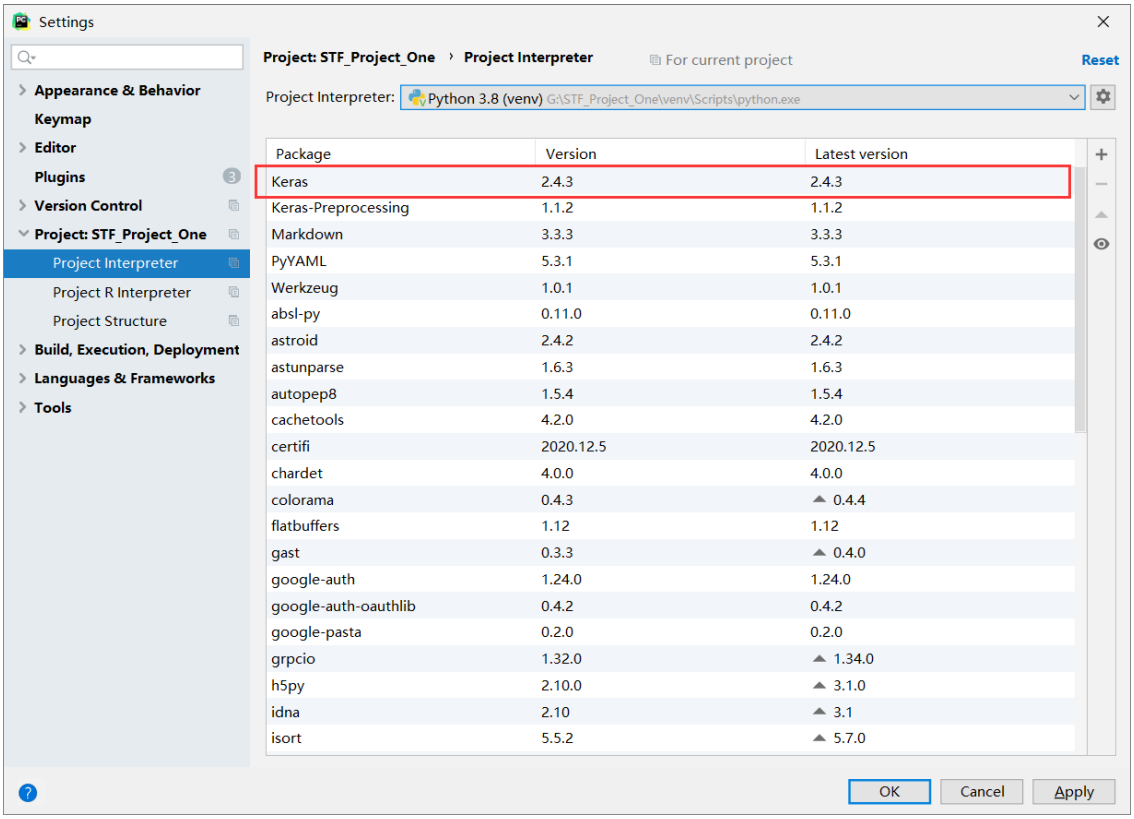
```
python -m pip install --upgrade pip
pip install tensorflow
pip install keras
```

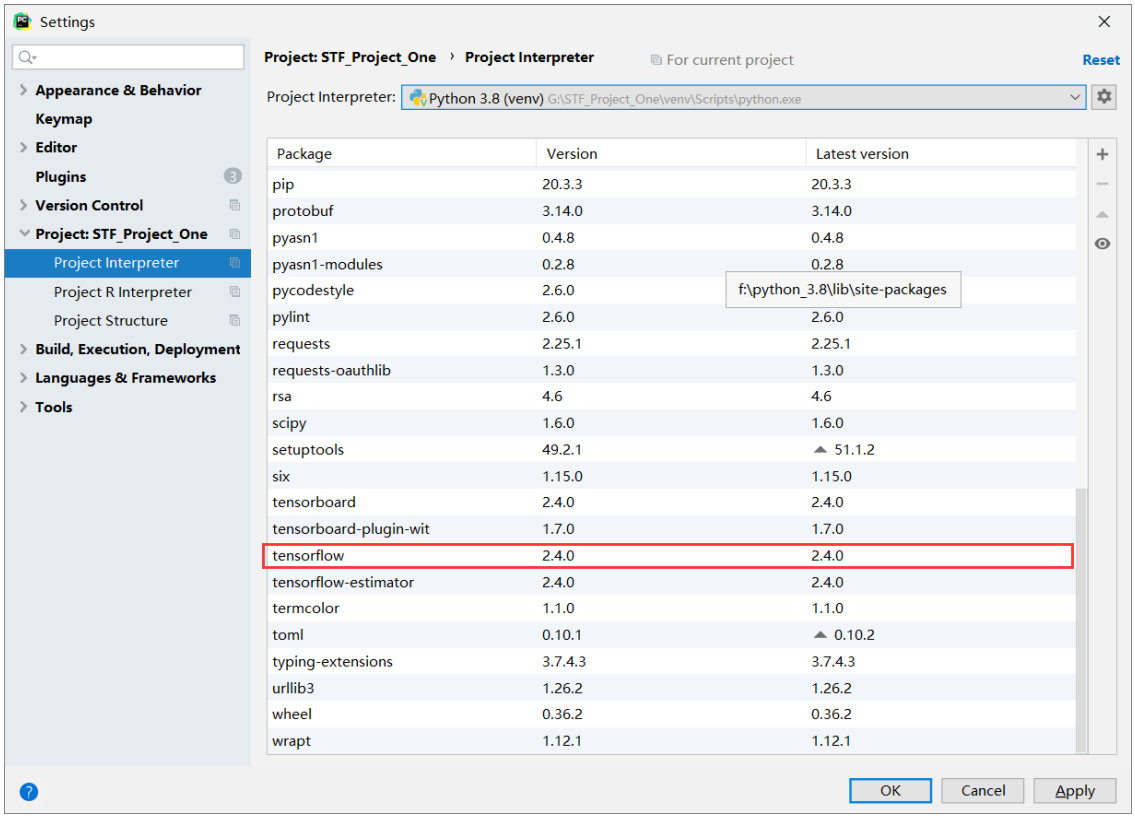
- 安装后测试如下：

```
C:\WINDOWS\system32>pip3 show tensorflow
Name: tensorflow
Version: 2.4.0
Summary: TensorFlow is an open source machine learning framework for everyone.
Home-page: https://www.tensorflow.org/
Author: Google Inc.
Author-email: packages@tensorflow.org
License: Apache 2.0
Location: f:\python_3.8\lib\site-packages
Requires: opt-einsum, typing-extensions, grpcio, gast, tensorflow-estimator, h5py, google-pas
ta, tensorboard, astunparse, keras-preprocessing, wrapt, absl-py, numpy, flatbuffers, six, wh
eel, protobuf, termcolor
Required-by:

C:\WINDOWS\system32>pip3 show keras
Name: Keras
Version: 2.4.3
Summary: Deep Learning for humans
Home-page: https://github.com/keras-team/keras
Author: Francois Chollet
Author-email: francois.chollet@gmail.com
License: MIT
Location: f:\python_3.8\lib\site-packages
Requires: pyyaml, h5py, scipy, numpy
Required-by:

C:\WINDOWS\system32>
```

安装成功！

- PyCharm配置：





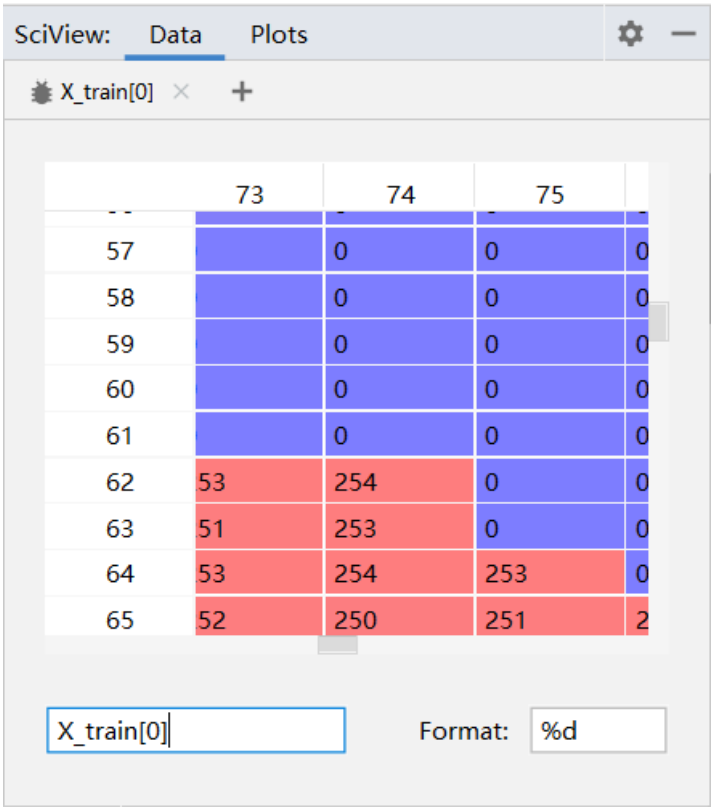
可以看见Project Interpreter中已经出现`tensorflow`以及`keras`，均为最新版本，配置成功！

- 实现`x_train, x_test, y_train, y_test`的构建
- 实现图片数据读取（保存于`X_train`以及`X_test`）（`X`为大写）

> 表示形式为数组（array）

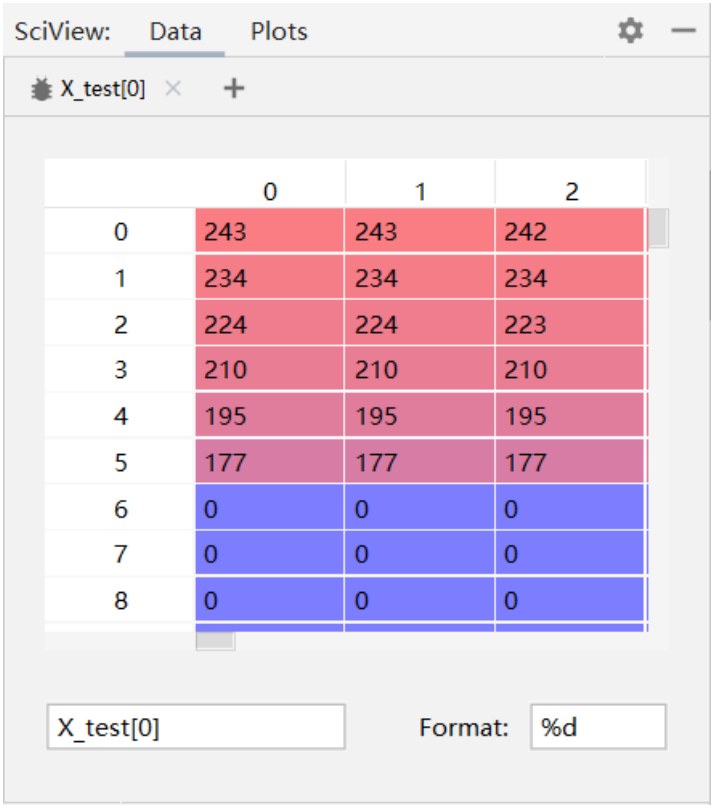
  - `X_train`存放`x_train`中不同路径下对应图片的数组

```
x_train reading image data, current position: 61
x_train reading image data, current position: 62
x_train reading image data, current position: 63
x_train reading image data, current position: 64
x_train reading image data, current position: 65
x_train reading image data, current position: 66
x_train reading image data, current position: 67
x_train reading image data, current position: 68
x_train reading image data, current position: 69
x_train reading image data, current position: 70
x_train reading image data, current position: 71
x_train reading image data, current position: 72
x_train reading image data, current position: 73
x_train reading image data, current position: 74
x_train reading image data, current position: 75
```



- `X_test`存放`x_test`中不同路径下对应图片的数组

```
x_test reading image data, current position: 72
x_test reading image data, current position: 73
x_test reading image data, current position: 74
x_test reading image data, current position: 75
x_test reading image data, current position: 76
x_test reading image data, current position: 77
x_test reading image data, current position: 78
x_test reading image data, current position: 79
x_test reading image data, current position: 80
x_test reading image data, current position: 81
x_test reading image data, current position: 82
x_test reading image data, current position: 83
x_test reading image data, current position: 84
x_test reading image data, current position: 85
x_test reading image data, current position: 86
x_test reading image data, current position: 87
```



	0	1	2
0	243	243	242
1	234	234	234
2	224	224	223
3	210	210	210
4	195	195	195
5	177	177	177
6	0	0	0
7	0	0	0
8	0	0	0

- 图片分别为`X_train`与`X_test`中第一个元素的读取结果

2021.1.10

- 实现对于图像信息的归一化处理

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
```

处理后的`x_train`的形状（shape）为:

```
x_train.shape is:  (370, 250, 250, 1)
```

- 实现卷积神经网络（CNN）模型的构建
- 实现编译

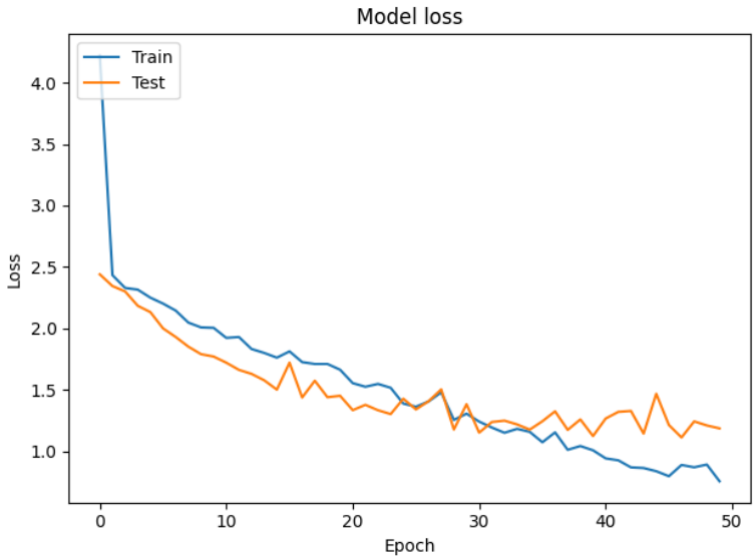
```
    # 3. 编译
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
```

- 实现训练与绘制训练

```
Epoch 1/50
2021-01-12 11:25:45.727808: W
tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of
991494144 exceeds 10% of free system memory.
2021-01-12 11:25:47.103902: W
tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of
464027648 exceeds 10% of free system memory.
```

```
2021-01-12 11:25:53.941881: W
tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of
464027648 exceeds 10% of free system memory.
2021-01-12 11:25:57.922458: W
tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of
495747072 exceeds 10% of free system memory.
2021-01-12 11:25:57.922816: W
tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of
991494144 exceeds 10% of free system memory.
3/3 [==============================] - 67s 22s/step - loss: 3.9685 -
accuracy: 0.0895 - val_loss: 2.4397 - val_accuracy: 0.0741
Epoch 2/50
3/3 [==============================] - 37s 13s/step - loss: 2.4262 -
accuracy: 0.1042 - val_loss: 2.3443 - val_accuracy: 0.1111
# 此处省略 3~47 迭代输出
Epoch 48/50
3/3 [==============================] - 36s 13s/step - loss: 0.8352 -
accuracy: 0.7129 - val_loss: 1.2419 - val_accuracy: 0.5741
Epoch 49/50
3/3 [==============================] - 36s 13s/step - loss: 0.8504 -
accuracy: 0.6949 - val_loss: 1.2088 - val_accuracy: 0.5556
Epoch 50/50
3/3 [==============================] - 37s 13s/step - loss: 0.7513 -
accuracy: 0.7483 - val_loss: 1.1850 - val_accuracy: 0.5741
4/4 [==============================] - 2s 529ms/step - loss: 1.1850 -
accuracy: 0.5741
acc 0.5740740895271301
```

- 实现训练效果评估（使用测试集数据，调参后处理结果）



由图可得，收敛效果明显，训练达到预期！

2021.1.11 与 2021.1.12

- 调参

> 详见：调参记录.pdf

2021.1.13 与 2021.1.14

- 文档撰写