|  |  |
| --- | --- |
| dlib库人脸特征提取解构报告 | 文件类型：技术文档 |
| 文件编号： |
| 面向的部门： 研发中心 |
| 保密等级：高 |
| 作者： 刘凯 |
| 日期：2018-09-20 |
| 版本：1.0V |

dlib库人脸特征提取解构报告



目 录

[1 dlib库人脸特征提取算法 3](#_Toc528332722)

[1.1 ResNet简介 3](#_Toc528332723)

[1.2 网络结构 4](#_Toc528332724)

[2 算法实现细节 7](#_Toc528332725)

[2.1 人脸校正 7](#_Toc528332726)

[2.2 特征提取网络 9](#_Toc528332727)

[附录：详细网络结构 10](#_Toc528332728)

[参考文献 1](#_Toc528332729)

# dlib库人脸特征提取算法

dlib库里的人脸特征提取采用的是ResNet[1]的网络架构，并用loss\_metric作为损失函数，以减小特征空间中同类的距离，增加不同类的距离。本节从ResNet简介，dlib人脸特征提取的ResNet网络结构来解构dlib库的人脸特征提取算法。

* 1. ResNet简介

从经验上看，深度学习网络的层数越多，学习能力应该越强，但实验发现并非如此，随着深度的增加，深度网络会出现学习退化的问题(degradation problem)，如图1所示。这并非是因为过拟合，因为过拟合时，训练集性能不会变差。ResNet的提出很好地解决了这个问题，既能增加深度，提高学习能力，又能很好地避免退化问题。

ResNet的原理是在浅层的网络结构上增加一个恒等映射层，其网络结构如图2所示。从图中可以看出，一方面ResNet仅仅在传统网络上增加了一条短路连接(shortcut connection)，不增加参数存储和计算资源的消耗，可以用传统的框架训练；另一方面，因为增加了一条短路连接，当网络已经达到最优，残差为0时，ResNet相当于直接联通，性能不会损失，从而解决了退化问题。当然实际情况下，残差不可能完全为零，网络总是能学习到新的特征，从而提升性能。



图1：深度学习“退化”问题



图2：ResNet网络模块

具体而言，对于图2的残差网络模块，其输出为：



其中，为待学习的残差映射，可以是卷积层，全连接层等。式需要输入和输出的维度一致，当两者维度不一致时，需要用一个线性投影矩阵来匹配输出维度：



* 1. 网络结构

dlib库里用于提取人脸特征的网络，是修改了文献[1]里ResNet34网络而来。具体而言，网络主要采用了两种残差网络块，ares和ares\_down，如图3所示。其中，“3\*3 conv”表示卷积层，卷积核的维度为3\*3；“3\*3 conv，/2”表示3\*3的卷积层，但步幅(stride)为2，相对于进行了2倍下采样；“affine”表示仿射层，运算操作为输入和该层参数进行pointwise的乘法和加法，功能类似于批归一化层(BN, batch normalization)；“Relu”表示激活层采用线性修正单元(Rectified Linear Unit); “a\_pool, /2”表示平均池化层，步幅为2，相当于进行2倍下采样，以匹配输入和残差的维度。

ResNet包含了1个输入模块和4级(levels)的残差模块，如图4和图5所示。其中,“input, 150”表示输入150\*150的图片；“conv(32,7,2)”表示卷积层通道数为32，卷积核维度为7\*7，步幅为2；“m\_pool”表示max pooling；“ares,32”表示采用ares网络块，且其卷积层的通道数为32；类似的，“ares\_down,64”表示采用ares\_down网络块，且其卷积层的通道数为64。完整的ResNet网络结构如图6所示。其中，“a\_pool2”表示将每个通道都求平均的平均池化操作；“fc, 128”表示输出为128维的全连接层；最后，“loss\_metric”是dlib库用的损失函数层。这个loss函数旨在减小同类在特征空间的距离，增加不同类在特征空间的距离。

3\*3 conv

affine

Relu

3\*3 conv

affine

Relu

a) **ares**网络块

3\*3 conv，/2

affine

Relu

3\*3 conv

affine

Relu

a\_pool, /2

b) **ares\_down**网络块

图3：2种残差网络块

input,150

conv(32,7,2)

affine

a) **input block**

relu

m\_pool

ares,32

ares,32

ares,32

b) **level 1**

ares\_down,64

ares,64

ares,64

c) **level 2**

ares,64

图4：ResNet中间结构块

ares\_down,128

ares,128

ares,128

a) **level 3**

ares\_down,256

ares,256

ares,256

b) **level 4**

ares\_down,256

图5：ResNet中间结构块

level 1

level 2

level 3

level 4

input block

a\_pool2

fc, 128

loss\_metric

output feature

图6：ResNet完整网络结构

# 算法实现细节

dlib库的人脸识别算法包括两个模块：人脸校正和特征提取网络。人脸校正是根据检测的人脸框和人脸特征点(shape)，对人脸图像进行旋转和插值，使得人脸图像符合特征提取网络的输入(这里是的RGB图片)。特征提取网络则是输入校正好的人脸图片，历经1.2节介绍的特征提取网络，得到该人脸的128维特征向量。

* 1. 人脸校正

人脸校正模块的输入是人脸对齐模块输出的人脸特征点，即68点的人脸特征点坐标。68点人脸特征点如图7所示，表1为特征点部位和索引的对应关系。人脸校正模块的输出为大小的RGB格式的人脸图片，模块内要完成人脸图片的旋转和插值，具体实现分为四步，如图8所示。



图7: 68点人脸特征点示意图

计算偏移角度和人脸矩形框

根据角度旋转矩形框

双线性插值

人脸金字塔降采样

图8：人脸校正模块流程

表1：68点人脸特征点分布

|  |  |
| --- | --- |
| 部位 | 特征点索引 |
| 人脸轮廓 | 0~16 |
| 左右眉毛 | 17~26 |
| 鼻子 | 27~35 |
| 眼睛 | 36~47 |
| 嘴唇 | 48~67 |

第一步，计算偏移角度和人脸矩形框。给出标准的校正好的人脸shape坐标，选取68个特征点中的鼻子、眼睛和上嘴唇的特征点(对应的特征index为27~54和60~64共33个特征点)，根据这些特征点，计算校正的人脸shape和人脸对齐模块输出的人脸shape的转换矩阵，转换矩阵的计算采用的是文献[2]的公式(34)到(43)，然后根据计算的转换矩阵，计算人脸偏移的角度(逆时针)和新的人脸检测框(参考dlib库interpolation.h头文件里的chip\_details结构)。

第二步，人脸金字塔降采样。因为要求输出为大小的图片，为了得到比较准确的人脸图片，若人脸图片的矩形框远大于150时，要进行降采样，直至人脸图片框大小在150左右为止。这里降采样率为2。

第三步，旋转矩形框。根据第一步计算的偏移角度和人脸矩形框，计算旋转后的人脸矩形框，即四个角逆时针旋转后的坐标位置。设，偏移角为，则逆时针旋转后的点的坐标为：



第四步，双线性插值。将大小的人脸框上的每个点映射到上一步计算出来的人脸框上，选择离映射点最近的四个像素点进行双线性插值，最终得到的人脸图片。

* 1. 特征提取网络

网络模型在1.2节已经介绍过，具体的网络结构参考附录。网络主要包括卷积层(con)，affine层，池化层(pool)，relu层，全连接层(fc)等，具体的实现过程这里不再赘述，只是有几个细节需要注意。

(1) 附录中各层的顺序是从下往上的，即第131层为输入层，第0层为输出层。

(2) 校正后的人脸图片要减掉均值(122.782,117.001,104.298)，并归一化，即除以256后才输入网络中。

(3) 在con层后面跟的不是一般的bn层，而是affine层。这一层的计算过程如下。设con层输出为，affine层的参数，在对应维度执行元素乘和元素加运算：



其中是扩展后的张量。

(4) 在下采样的ares\_down模块中，实际上是将输入经过残差网络和平均池化网络后进行相加(即附录的add\_prev2层)，这是相加的两个数据的维度可能不一样，这时就取各个维度最大的值，作为输出的维度。例如输入经过残差网络后的输出维度为，经过平均池化网络后的输出维度为，若，且，则add\_prev2层输出的维度为，维度对应的部分相加，不对应的部分补零。

附录：详细网络结构

The net has 132 layers in it.

layer<0> loss\_metric (margin=0.04, distance\_threshold=0.6)

layer<1> fc\_no\_bias (num\_outputs=128)

layer<2> avg\_pool (nr=0, nc=0, stride\_y=1, stride\_x=1, padding\_y=0, padding\_x=0)

layer<3> relu

layer<4> add\_prev2

layer<5> avg\_pool (nr=2, nc=2, stride\_y=2, stride\_x=2, padding\_y=0, padding\_x=0)

layer<6> skip1

layer<7> tag2

layer<8> affine

layer<9> con (num\_filters=256, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<10> relu

layer<11> affine

layer<12> con (num\_filters=256, nr=3, nc=3, stride\_y=2, stride\_x=2, padding\_y=0, padding\_x=0)

layer<13> tag1

layer<14> relu

layer<15> add\_prev1

layer<16> affine

layer<17> con (num\_filters=256, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<18> relu

layer<19> affine

layer<20> con (num\_filters=256, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<21> tag1

layer<22> relu

layer<23> add\_prev1

layer<24> affine

layer<25> con (num\_filters=256, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<26> relu

layer<27> affine

layer<28> con (num\_filters=256, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<29> tag1

layer<30> relu

layer<31> add\_prev2

layer<32> avg\_pool (nr=2, nc=2, stride\_y=2, stride\_x=2, padding\_y=0, padding\_x=0)

layer<33> skip1

layer<34> tag2

layer<35> affine

layer<36> con (num\_filters=256, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<37> relu

layer<38> affine

layer<39> con (num\_filters=256, nr=3, nc=3, stride\_y=2, stride\_x=2, padding\_y=0, padding\_x=0)

layer<40> tag1

layer<41> relu

layer<42> add\_prev1

layer<43> affine

layer<44> con (num\_filters=128, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<45> relu

layer<46> affine

layer<47> con (num\_filters=128, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<48> tag1

layer<49> relu

layer<50> add\_prev1

layer<51> affine

layer<52> con (num\_filters=128, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<53> relu

layer<54> affine

layer<55> con (num\_filters=128, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<56> tag1

layer<57> relu

layer<58> add\_prev2

layer<59> avg\_pool (nr=2, nc=2, stride\_y=2, stride\_x=2, padding\_y=0, padding\_x=0)

layer<60> skip1

layer<61> tag2

layer<62> affine

layer<63> con (num\_filters=128, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<64> relu

layer<65> affine

layer<66> con (num\_filters=128, nr=3, nc=3, stride\_y=2, stride\_x=2, padding\_y=0, padding\_x=0)

layer<67> tag1

layer<68> relu

layer<69> add\_prev1

layer<70> affine

layer<71> con (num\_filters=64, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<72> relu

layer<73> affine

layer<74> con (num\_filters=64, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<75> tag1

layer<76> relu

layer<77> add\_prev1

layer<78> affine

layer<79> con (num\_filters=64, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<80> relu

layer<81> affine

layer<82> con (num\_filters=64, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<83> tag1

layer<84> relu

layer<85> add\_prev1

layer<86> affine

layer<87> con (num\_filters=64, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<88> relu

layer<89> affine

layer<90> con (num\_filters=64, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<91> tag1

layer<92> relu

layer<93> add\_prev2

layer<94> avg\_pool (nr=2, nc=2, stride\_y=2, stride\_x=2, padding\_y=0, padding\_x=0)

layer<95> skip1

layer<96> tag2

layer<97> affine

layer<98> con (num\_filters=64, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<99> relu

layer<100> affine

layer<101> con (num\_filters=64, nr=3, nc=3, stride\_y=2, stride\_x=2, padding\_y=0, padding\_x=0)

layer<102> tag1

layer<103> relu

layer<104> add\_prev1

layer<105> affine

layer<106> con (num\_filters=32, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<107> relu

layer<108> affine

layer<109> con (num\_filters=32, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<110> tag1

layer<111> relu

layer<112> add\_prev1

layer<113> affine

layer<114> con (num\_filters=32, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<115> relu

layer<116> affine

layer<117> con (num\_filters=32, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<118> tag1

layer<119> relu

layer<120> add\_prev1

layer<121> affine

layer<122> con (num\_filters=32, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<123> relu

layer<124> affine

layer<125> con (num\_filters=32, nr=3, nc=3, stride\_y=1, stride\_x=1, padding\_y=1, padding\_x=1)

layer<126> tag1

layer<127> max\_pool (nr=3, nc=3, stride\_y=2, stride\_x=2, padding\_y=0, padding\_x=0)

layer<128> relu

layer<129> affine

layer<130> con (num\_filters=32, nr=7, nc=7, stride\_y=2, stride\_x=2, padding\_y=0, padding\_x=0)

layer<131> input\_rgb\_image\_sized(122.782,117.001,104.298) nr=150 nc=150

参考文献

1. Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun. Deep Residual Learning for Image Recognition. CVPR 2016.
2. Shinji Umeyama. Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1991.