

haoji007的博客

机器学习，深度学习，遥感图像应用

目录视图

摘要视图

RSS 订阅

个人资料



haoji007

关注

发私信

访问：162693次

积分：2099

等级：BLOG > 5

排名：第19011名

原创：6篇

转载：404篇

译文：0篇

评论：16条

文章搜索

异步赠书：9月重磅新书升级，本本经典 程序员9月书讯 每周荐书：ES6、虚拟现实、物联网（评论送书）

【Caffe实践】基于CNN的性别、年龄识别

2016-07-31 21:24

1409人阅读

评论(0)

收藏

分类：

【Caffe技巧及有趣实验】（142）

CNN应用之性别、年龄识别

原文地址：<http://blog.csdn.net/hjimce/article/details/42255212>

作者：hjimce

一、相关理论

本篇博文主要讲解2015年一篇paper《Age and Gender Classification using Convolutional Neural Networks》，个人感觉这篇文献没啥难度，只要懂得Alexnet，实现这篇文献的算法，会比较容易。其实读完这篇paper之后，我一直在想paper的创新点在哪里？因为我实在没有看出paper的创新点在哪里，估计是自己水平太lower了，看文献没有抓到文献的创新点。难道是因为利用CNN做年龄和性别分类的paper很少吗？网上

关闭

文章分类

- 【深度学习 及 论文笔记】 (138)
- 【机器学习 及 论文笔记】 (42)
- 【Caffe技巧及有趣实验】 (143)
- 【深度学习----应用实例】 (12)
- 【实验数据集和数据库】 (14)
- 【ML优化方法】 (8)
- 【Python相关】 (63)
- 【Matlab相关】 (14)
- 【Ubuntu相关】 (11)
- 【其他】 (20)

阅读排行

- 【Caffe安装】caffe安装系列—... (34816)
- python数字图像处理（5）：图... (5077)
- 利用matlab求图像均值和方差... (4463)
- 用python简单处理图片（4）：... (3278)
- FaceDataset常用的人脸数据库 (2747)
- Matlab相关工具箱下载地址汇总 (2728)
- python数字图像处理（17）：... (2358)
- 【目标检测大集合】R-FCN、... (2184)
- python数字图像处理（6）：图... (2038)
- 图像处理中滤波（filtering）与... (2029)

搜索了一下，性别预测，以前很多都是用SVM算法，用CNN搞性别分类就只搜索到这一篇文章。个人感觉利用CNN进行图片分类已经不是什么新鲜事了，年龄、和性别预测，随便搞个CNN网络，然后开始训练跑起来，也可以获得不错的精度。

性别分类自然而然的是二分类问题，然而对于年龄怎么搞？年龄预测是回归问题吗？paper采用的方法是把年龄划分为多个年龄段，每个年龄段相当于一个类别，这样性别也就多分类问题了。所以我们不要觉得现在的一些APP，功能好像很牛逼，什么性别、年龄、衣服类型、是否佩戴眼镜等识别问题，其实这种识别对于CNN来说，松松搞定的事，当然你如果要达到非常高的识别精度，是另外一回事了，就

言归正传，下面开始讲解2015年paper《Age and Gender Classification using Convolutional Neural Networks》的网络结构，这篇文章没有什么新算法，只有调参，改变[卷积核大小等.....所以如果已经对Alexnet比较熟悉的，可能会觉得看起来没这篇paper的相关源码和训练数据，文献作者有给我们提供可以到Caffe zoo model：<https://github.com/BVLC/caffe/wiki/Model-Zoo> 或者文献的主页：http://www.openu.ac.il/home/hassner/projects/cnn_agegender/。下载相关训练好的模型，paper性别、年龄预测的应用场景比较复杂，都是一些非常糟糕的图片，比较模糊的图片等，所以如果我们想要直接利用paper训练好的模型，用到我们自己的项目上，可能精度会比较低，后面我将会具体讲一下利用paper的模型进行fine-tuning，以适应我们的应用，提高我们自己项目的识别精度。

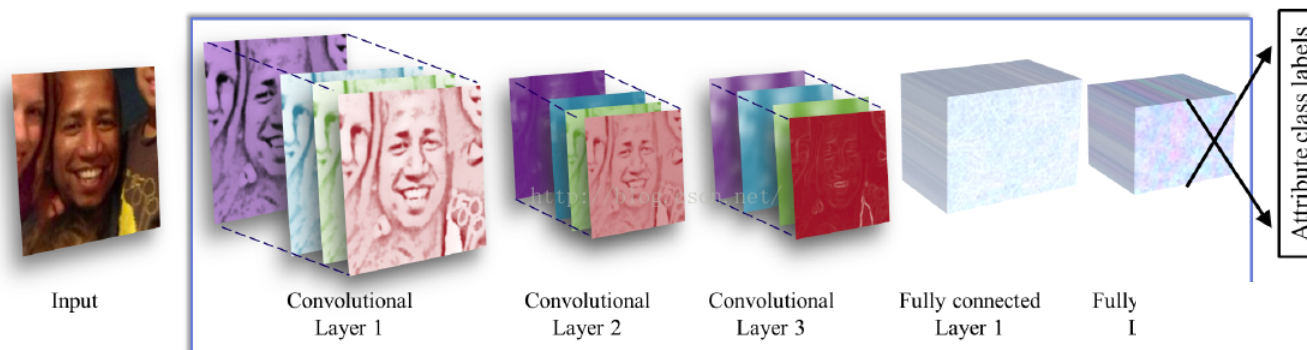
二、算法实现

评论排行

- 【Caffe安装】caffe安装系列—... (2)
- 【目标检测大集合】R-FCN、... (2)
- Caffe傻瓜系列(9): 训练和测试... (2)
- 深度学习 (十五) 基于级联卷... (1)
- 【论文笔记】A Convolutional ... (1)
- caffe上手：如何导出caffemode... (1)
- 人脸集数据库 (1)
- 人脸预处理工具FaceTools (1)
- 基于Caffe的人脸检测实现 (1)
- 深度学习 (三十) 贪婪深度字... (1)

因为paper的主页，有提供网络结构的源码，我将结合网络结构文件进行讲解。

1、网络结构



Paper所用的网络包含：3个卷积层，还有2个全连接层。这个算是层数比较多的网络模型了，这样可以避免过拟合。对于年龄的识别，paper仅仅有8个年龄段，相当于8分类模型；然后对于性别识别自然而是二分类问题了。

然后图像处理直接采用3通道彩色图像进行处理，图片6都统一缩放到256*256，然后进行裁剪，为227*227（训练过程随机裁剪，验证测试：+10%的正方形的人脸，中心裁剪），也就是说网络的输入是227*227的3通道彩色图像，总之基本上跟Alexnet一样。

网络模型：

(1)第一层：采用96个卷积核，每个卷积核参数个数为3*7*7，这个就相当于3个7*7大小的卷积核在每个通道进行卷积。激活函数采用ReLU，池化采用最大重叠池化，池化的size选择3*3，strides选择2。然后接着再来一个局部响应归一化层。什么叫局部响应归一化，自己可以查看一下文献：

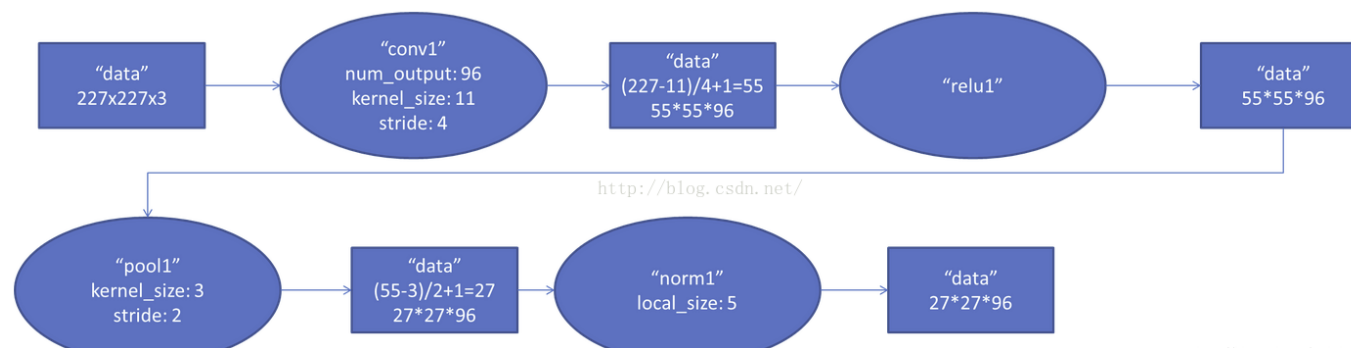
[关闭](#)

《ImageNet Classification with Deep Convolutional Neural Networks》，局部响应归一化可以提高网络的泛化能力。

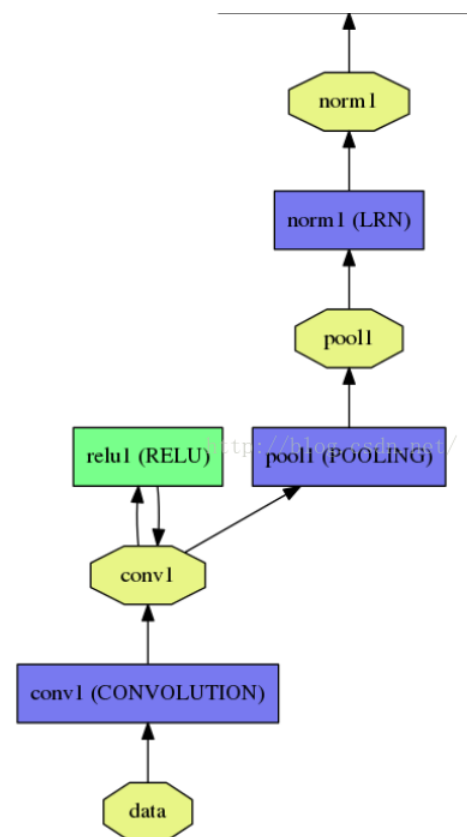
$$b_{x,y}^i = a_{x,y}^i / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right)^\beta$$

局部响应归一化，这个分成两种情况，一种是3D的归一化，也就是特征图之间对应像素点的一个归一化。还有一种是2D归一化，就是对特征图的每个像素的局部归一化。局部响应归一化其实这个可有可无，精度提高不了多少，如果你还不明白式子也没有关系。我们可以利用最新的算法：Batch Normalize，这个才牛逼啊。2015年，我觉得最牛逼的算法之一，不仅提高了训练速度，连精度也提高了。不过，通过7*7大小的卷积核，对227*227图片卷积，然后特征图的个数为96个，每个特征图都是11*11大小的，这个作者没有讲到卷积层的stride大小，不过我们大体可以推测出来，这个网络结构是模仿：ImageNet Classification with Deep Convolutional Neural Networks的网络结构的，连输入图片的大小也是一样的，这篇文献的第一层如下所示：

关闭



我们可以推测出，paper选择的卷积步长为4，这样经过卷积后，然后pad为2，这样经过卷积后图片的大小为： $(227-7)/4+1=56$ 。然后经过 $3*3$ ，且步长为2的大小，进行重叠池化，可以得到： $56/2=28*28$ 大小的图片，具体边界需要补齐。下面是原文的第一层结构示意图：



关闭

[python] [view plain](#) [copy](#)  

```
01. layers {  
02.   name: "conv1"  
03.   type: CONVOLUTION  
04.   bottom: "data"  
05.   top: "conv1"
```

```
06. blobs_lr: 1
07. blobs_lr: 2
08. weight_decay: 1
09. weight_decay: 0
10. convolution_param {
11.   num_output: 96
12.   kernel_size: 7
13.   stride: 4
14.   weight_filler {
15.     type: "gaussian"
16.     std: 0.01
17.   }
18.   bias_filler {
19.     type: "constant"
20.     value: 0
21.   }
22. }
23. }
24. layers {
25.   name: "relu1"
26.   type: RELU
27.   bottom: "conv1"
28.   top: "conv1"
29. }
30. layers {
31.   name: "pool1"
32.   type: POOLING
33.   bottom: "conv1"
34.   top: "pool1"
35.   pooling_param {
36.     pool: MAX
37.     kernel_size: 3
38.     stride: 2
39.   }
40. }
41. layers {
42.   name: "norm1"
43.   type: LRN
44.   bottom: "pool1"
```

关闭

```

45.     top: "norm1"
46.     lrn_param {
47.         local_size: 5
48.         alpha: 0.0001
49.         beta: 0.75
50.     }
51. }

```

(2)第二层：第二层的输入也就是96*28*28的单通道图片，因为我们上一步已经把三通道合在一起进行卷积了。第二层结构，选择256个滤波器，滤波器大小为5*5，步长为1，这个也可以参考AlexNet的结构。池化也是选择跟上面的一样的参数。

```

[python] view plain copy C {
01. layers {
02.     name: "conv2"
03.     type: CONVOLUTION
04.     bottom: "norm1"
05.     top: "conv2"
06.     blobs_lr: 1
07.     blobs_lr: 2
08.     weight_decay: 1
09.     weight_decay: 0
10.     convolution_param {
11.         num_output: 256
12.         pad: 2
13.         kernel_size: 5
14.         weight_filler {
15.             type: "gaussian"
16.             std: 0.01
17.         }
18.         bias_filler {
19.             type: "constant"

```

关闭

```
20.         value: 1
21.     }
22. }
23. }
24. layers {
25.     name: "relu2"
26.     type: RELU
27.     bottom: "conv2"
28.     top: "conv2"
29. }
30. layers {
31.     name: "pool2"
32.     type: POOLING
33.     bottom: "conv2"
34.     top: "pool2"
35.     pooling_param {
36.         pool: MAX
37.         kernel_size: 3
38.         stride: 2
39.     }
40. }
41. layers {
42.     name: "norm2"
43.     type: LRN
44.     bottom: "pool2"
45.     top: "norm2"
46.     lrn_param {
47.         local_size: 5
48.         alpha: 0.0001
49.         beta: 0.75
50.     }
51. }
```

关闭

(3)第三层：滤波器个数选择384，卷积核大小为3*3。

[python] [view plain](#) [copy](#)  


```
01. layers {
02.   name: "conv3"
03.   type: CONVOLUTION
04.   bottom: "norm2"
05.   top: "conv3"
06.   blobs_lr: 1
07.   blobs_lr: 2
08.   weight_decay: 1
09.   weight_decay: 0
10.   convolution_param {
11.     num_output: 384
12.     pad: 1
13.     kernel_size: 3
14.     weight_filler {
15.       type: "gaussian"
16.       std: 0.01
17.     }
18.     bias_filler {
19.       type: "constant"
20.       value: 0
21.     }
22.   }
23. }
24. layers {
25.   name: "relu3"
26.   type: RELU
27.   bottom: "conv3"
28.   top: "conv3"
29. }
30. layers {
31.   name: "pool5"
32.   type: POOLING
33.   bottom: "conv3"
34.   top: "pool5"
35.   pooling_param {
36.     pool: MAX
37.     kernel_size: 3
38.     stride: 2
39.   }
```

关闭

40. | }

(4)第四层：第一个全连接层，神经元个数选择512。

```

[python] view plain copy C ?
01. layers {
02.     name: "fc6"
03.     type: INNER_PRODUCT
04.     bottom: "pool5"
05.     top: "fc6"
06.     blobs_lr: 1
07.     blobs_lr: 2
08.     weight_decay: 1
09.     weight_decay: 0
10.     inner_product_param {
11.         num_output: 512
12.         weight_filler {
13.             type: "gaussian"
14.             std: 0.005
15.         }
16.         bias_filler {
17.             type: "constant"
18.             value: 1
19.         }
20.     }
21. }
22. layers {
23.     name: "relu6"
24.     type: RELU
25.     bottom: "fc6"
26.     top: "fc6"
27. }
28. layers {
29.     name: "drop6"
30.     type: DROPOUT
31.     bottom: "fc6"
32.     top: "fc6"
33.     dropout_param {

```

关闭

```
34.     dropout_ratio: 0.5
35.   }
36. }
```

(5)第五层：第二个全连接层，神经元个数也是选择512。

```
[python] view plain copy C ?
01. layers {
02.   name: "fc7"
03.   type: INNER_PRODUCT
04.   bottom: "fc6"
05.   top: "fc7"
06.   blobs_lr: 1
07.   blobs_lr: 2
08.   weight_decay: 1
09.   weight_decay: 0
10.   inner_product_param {
11.     num_output: 512
12.     weight_filler {
13.       type: "gaussian"
14.       std: 0.005
15.     }
16.     bias_filler {
17.       type: "constant"
18.       value: 1
19.     }
20.   }
21. }
22. layers {
23.   name: "relu7"
24.   type: RELU
25.   bottom: "fc7"
26.   top: "fc7"
27. }
28. layers {
29.   name: "drop7"
```



关闭

```

30.     type: DROPOUT
31.     bottom: "fc7"
32.     top: "fc7"
33.     dropout_param {
34.         dropout_ratio: 0.5
35.     }
36. }

```

(6)第六次：输出层，对于性别来说是二分类，输入神经元个数为2。

[python] [view plain](#) [copy](#)  

```

01. layers {
02.     name: "fc8"
03.     type: INNER_PRODUCT
04.     bottom: "fc7"
05.     top: "fc8"
06.     blobs_lr: 10
07.     blobs_lr: 20
08.     weight_decay: 1
09.     weight_decay: 0
10.     inner_product_param {
11.         num_output: 2
12.         weight_filler {
13.             type: "gaussian"
14.             std: 0.01
15.         }
16.         bias_filler {
17.             type: "constant"
18.             value: 0
19.         }
20.     }
21. }
22. layers {
23.     name: "accuracy"

```

关闭

```

24.     type: ACCURACY
25.     bottom: "fc8"
26.     bottom: "label"
27.     top: "accuracy"
28.     include: { phase: TEST }
29. }
30. layers {
31.     name: "loss"
32.     type: SOFTMAX_LOSS
33.     bottom: "fc8"
34.     bottom: "label"
35.     top: "loss"
36. }

```

网络方面，paper没有什么创新点，模仿AlexNet结构。

2、网络训练

(1)初始化参数：权重初始化方法采用标准差为0.01，均值为0的高斯正太分布。

(2)网络训练：采用dropout，来限制过拟合。Drop out比例采用0.5，还有就是数据扩充

充，数据扩充是通过输入256*256的图片，然后进行随机裁剪。

关闭

当然裁剪要以face中心为基础，进行裁剪。

(3)训练方法采用，随机梯度下降法，min-batch 大小选择50，学习率大小0.001，然后当迭代到10000次以后，把学习率调为0.0001。

(4)结果预测：预测方法采用输入一张256*256的图片，然后进行裁剪5张图片为227*227大小，其中四张图片的裁剪方法分别采用以256*256的图片的4个角为基点点，进行裁剪。然后最后一张，以人脸的中心为基点进行裁剪。然后对这5张图片进行预测，最后对预测结果进行平均。

三、实际应用

文献作者给我们提供，可以到Caffe zoo

model : <https://github.com/BVLC/caffe/wiki/Model-Zoo> 或者文献的主

页 : http://www.openup.ac.il/home/hassner/projects/cnn_agegender/。下载相关训练好的模

型，paper性别、年龄预测的应用场景比较复杂，都是一些非常糟糕的图片，比较模糊的图片等，所以如果我们想要直接利用paper训练好的模型，用到我们自己的项目上，可能精度会比较低。我测试了一下，直接使用paper给的模型，在我的数据_

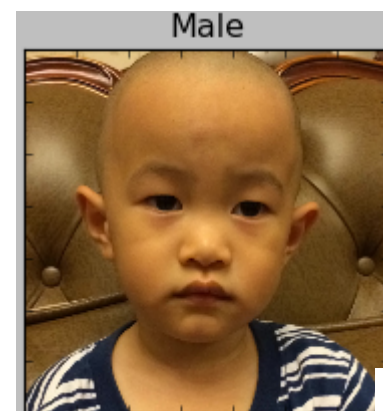
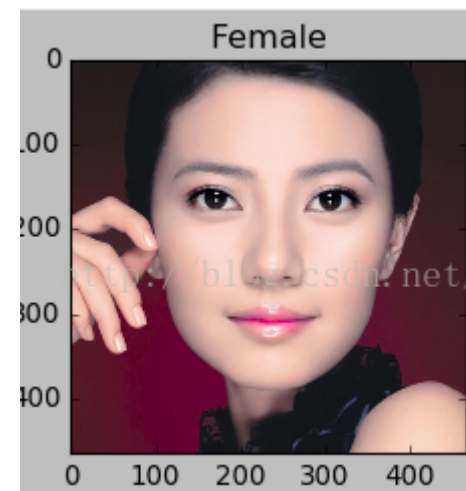
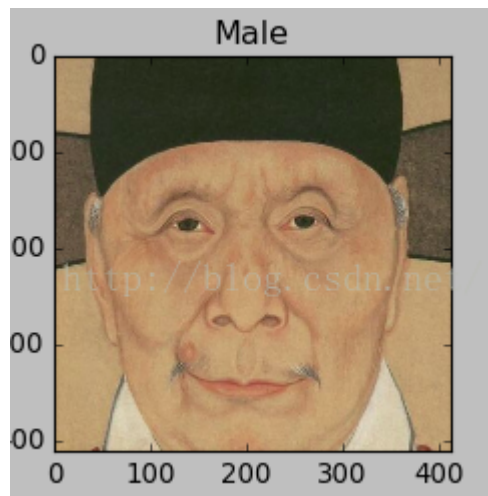
试，我的数据是中国人、，然后也比较清晰，直接用作者训练好的模型，精

右，这个精度对于我们实际的工程应用还差很远。后面就要发挥自己的调参

度提高上去，才能达到95%以上的精度，具体因为项目保密，所以不再啰嗦。最后预测

结果如下：

关闭



测试精度：

```
Male  
Male  
Male  
Male  
acu: 0.969849246231
```

总结：看完这篇文献，感觉没看到什么比较牛逼的创新点，只是把Alexnet网络改一改而已，个人感觉AlexNet的一些算法已经过时了，现在各种最新牛逼文献的算法一大

关闭

堆，随便找一个，调一调参，应该可以得到更高的精度，因为毕竟图片分类的算法更新太快了。年龄预测方面，因为自己的项目用不到，而且年龄预测这个东西，精度一向很低，很容易受光照、拍摄角度等因素影响，即便是我们人类，也很难精确判断一个人的年龄，有的人五十几岁了，但是看起来却很年轻.....

PS：赶紧研究深度学习算法去，现在大部分深度学习的文章，有的文献只是稍微改一下参数、改一下结构，然后发现精度state-of-art，于是发表paper，很容易就被录用了。

参考文献：

- 1、《Age and Gender Classification using Convolutional Neural Networks
- 2、《ImageNet Classification with Deep Convolutional Neural Networks》
- 3、http://www.openup.ac.il/home/hassner/projects/cnn_agegender/
- 4、<https://github.com/BVLC/caffe/wiki/Model-Zoo>

顶
0

踩
0

关闭

- [上一篇](#) 【Caffe实践】基于Caffe的人脸检测实现
- [下一篇](#) 【Caffe实践】基于CNN的性别、年龄识别的代码实现

相关文章推荐

- 使用caffe实现性别年龄预测
- Presto的服务治理与架构在京东的实践与应用--王哲...
- 【Caffe实践】基于Caffe的人脸检测实现
- 深入掌握Kubernetes应用实践--王渊命
- 【转载】在caffe上跑自己的数据
- Python基础知识汇总
- 基于caffe的性别、年龄识别
- Android核心技术详解
- 深度学习（十四）基于CNN的性别、年龄识别
- Retrofit 从入门封装到源码解析
- Ubuntu16.04桥接下如何配置固定IP
- 自然语言处理工具Word2Vec
- win7第一个caffe实验【识别人脸年龄和性别】
- Ubuntu16.04 Caffe 安装步骤记录（超详尽）
- caffe神经网络构建、参数设置
- 基于CNN的性别、年龄识别

查看评论

暂无评论

您还没有登录,请[登录](#)或[注册](#)

关闭

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

[网站客服](#) [杂志客服](#) [微博客服](#) webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 | 江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved

