基于卷积神经网络的人脸识别

目录

1. 综合实验题目

1.1 选题

CelebFaces 人脸识别。

1.2 选题依据

人脸识别是计算机视觉和深度学习领域的研究热点之一。它是一种基于人脸特征信息的生物识别技术。与其他识别方法相比，人脸识别以其直接、友好、方便的特点得到了广泛的研究和应用。此外，我们还可以进一步分析人脸识别的结果，获得更多关于人的性别、表情、年龄等方面的丰富信息，从而拓展人脸识别的应用前景。例如，近年来，人脸识别在登录、身份验证、网上支付、公共治安等领域的应用极大地改变了我们的生活。由于人脸识别的特殊优势，它在生物特征识别中具有重要的地位。

因此，我认为选择人脸识别是一次有意义的学习过程。

1.3 实验目标

设计出一种能够识别出 CelebFaces 数据集中任一指定属性的模型，并有较高的准确率和鲁棒性。

1.4 背景

随着大数据和深度学习的发展，神经网络在计算机视觉和自然语言处理等领域得到了广泛的关注和并取得突破性进展。从人脸识别的发展历程来看，深度学习在人脸识别中的作用十分明显。香港中文大学的 Sun Yi 等人提出了卷积神经网络（CNN）在人脸识别中的应用，采用 20 万训练数据，在 LFW 上第一次得到超过人类水平的识别精度，这是人脸识别发展历史上的一座里程碑。

因此，本实验采用卷积神经网络用于对 CelebFaces 数据集属性的识别。

2. 题目分析

2.1 数据来源

采用的数据集为 CelebFaces Attributes (CelebA) Dataset，下载于 https://www.kaggle.com/jessicali9530/celeba-dataset 。

原始数据来源于香港中文大学多媒体实验室：http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

2.2 数据格式

CelebFaces 属性数据集（CelebA）是一个大型人脸属性数据集，拥有超过 20 万的名人图像，每个图像都有 40 个属性注释。此数据集中的图像覆盖了大的姿势变化和背景杂乱。CelebA 拥有大量的多样性，大批量和丰富的注释，包括

- 202,599 个各种名人的脸部图像数量。

- 10,177 个独特身份，但没有给出身份名称。
- 每个图像 40 个二进制属性注释。

图 样本图片

实验过程中用到的数据文件组成：

- img_align_celeba.zip：包含所有面部图像，并经过裁剪和对齐。
- list_attr_celeba.csv：含有每个图像的属性标签。有 40 个属性。“1”表示含有该属性特征，而“-1”表示不含有。

2.3 适合的算法类型

在人脸识别技术研究的领域中，目前主要有几种研究的方向，如：

一种是根据人脸特征统计学的识别方法，其主要有特征脸的方法以及隐马尔科夫模型（HMM，Hidden Markov Model）方法等。

另一种人脸识别方法是关于连接机制的，主要有人工神经网络（ANN，Artificial Neural Network)方法和支持向量机(SVM,Support Vector Machine）方法等。

还有一个就是综合多种识别方式的方法。

目前使用较为广泛且有效的是人工神经网络中的卷积神经网络（CNN)。CNN 主要用来识别位移、缩放及其他形式的图形。由于 CNN 的特征检测层通过训练数据进行学习，所以在使用 CNN 时，避免了显示的特征抽取，而隐式地从训练数据中进行学习；再者由于同一特征映射面上的神经元权值相同，所以网络可以并行学习，这也是卷积网络相对于神经元彼此相连网络的一大优势。卷积神经网络以其局部权值共享的特殊结构在语音识别和图像处理方面有着独特的优越性，权值共享降低了网络的复杂性，特别是多维输入向量的图像可以直接输入网络这一特

点避免了特征提取和分类过程中数据重建的复杂度。

因此，综合来看，卷积神经网络更为合适。

3. 实验过程

3.1 数据预处理

原始图片大小为[218, 178, 3]。对于卷积神经网络来说，特别是较深的网络结构，图像的尺寸不宜过大，否则在训练时会因为参数过多，而发生 OOM（Out of Memory，内存溢出）问题。因此我们需要适当的减小图片的尺寸，压缩图像信息。这样既能减少参数规模，也能提高训练效率。

我们使用 PIL (Python Image Library)库进行一些图像处理。首先，我们将图像转化成灰度图，这样可以将 3 个颜色通道压缩成 1 个颜色通道，图像的尺寸减少了两倍。

```
img shape: h: 218, w: 178, c: 3
Text(0.5, 1.0, 'Raw Img')
```

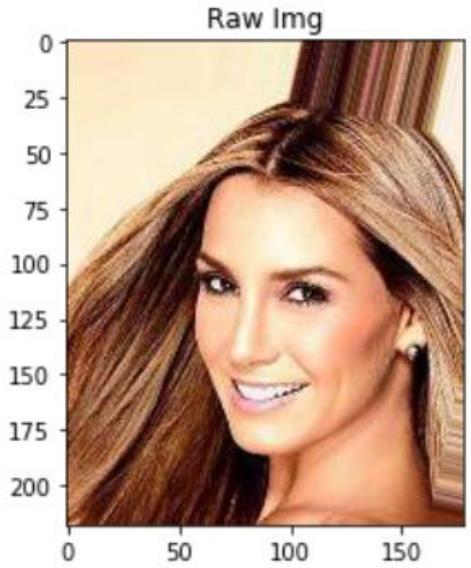

图 原始图片样例

```
new shape: h: 218, w: 178
Text(0.5, 1.0, 'Resized Gray Img')
```

图 灰度图样例

然后，我们将图像的高和宽成比例缩小，得到理想的输入图像。

```
img shape: h: 218, w: 178, c: 3
Text(0.5, 1.0, 'Raw Img')
```

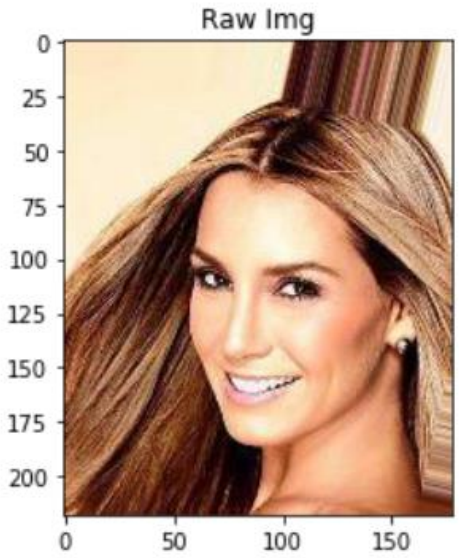

图 原始图片样例

```
new shape: h: 128, w: 104
Text(0.5, 1.0, 'Resized Gray Img')
```

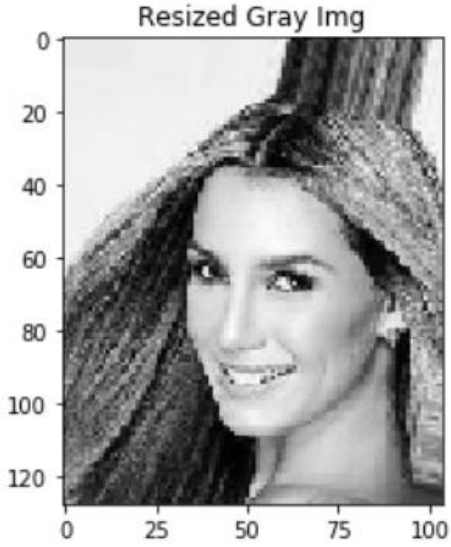

图 resize 后的灰度图样例

另外，假设数据集不够用时（只是假设，CelebA 数据集不存在这种问题），我们可以通过一些图形学方法扩充我们的数据集。例如通过翻转或旋转扩充已有的图片。数据集规模如果较小往往会产生过拟合问题（Overfitting）。

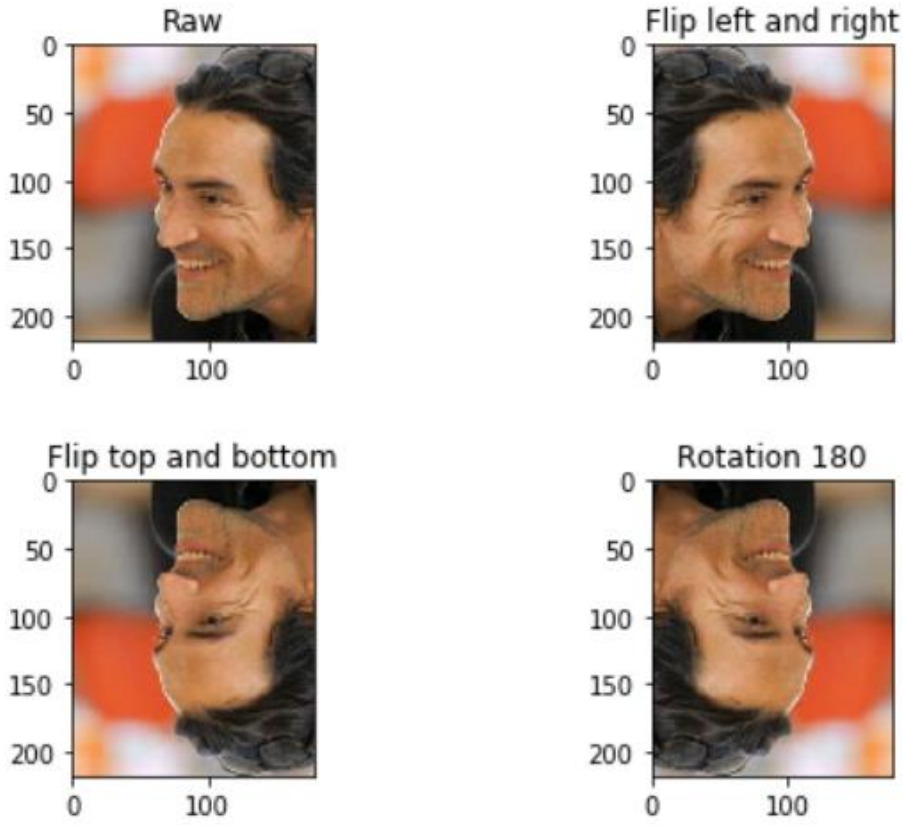

图 翻转和旋转图样例

在得到处理后的图片之后，将它们存储在数组中。同时，将指定的某一属性的值经过 one hot 编码后作为标签存储在数组中。之后，打乱数据集，然后划分出训练集和测试集（比例为 9:1）。

此后，这些数据就可以输入到模型中进行训练和测试。

3.2 算法的选择过程

因为此实验要做的是能够识别图像 40 个属性中任意指定的某个属性，所以使用的算法必须在图像识别领域中具有良好的泛型能力与准确度。传统的图像识别方法需要手动提取对应的特征，这就意味着，如果采用传统方法，要达到实验目的则需要手动提取 40 个属性对应的特征。这在短期内显然无法完成。而且，如果使用的是背景复杂或者经过旋转或翻转的数据，特征需要单独提取。所以，我们需要一种在较复杂的属性特征情况下能够有良好鲁棒性的模型。

卷积神经网络（CNN）是一种监督学习模型，能够自动地挖掘数据局部特征，提取全局训练特征和分类。CNN 通过结合人脸图像空间的局部感知区域、共享权重、在空间或时间上的降采样来充分利用数据本身包含的局部性等特征，优化模型结构，保证了一定的位移不变性。

因此，CNN 非常适合此次实验的算法需求。具体的模型如下图所示。

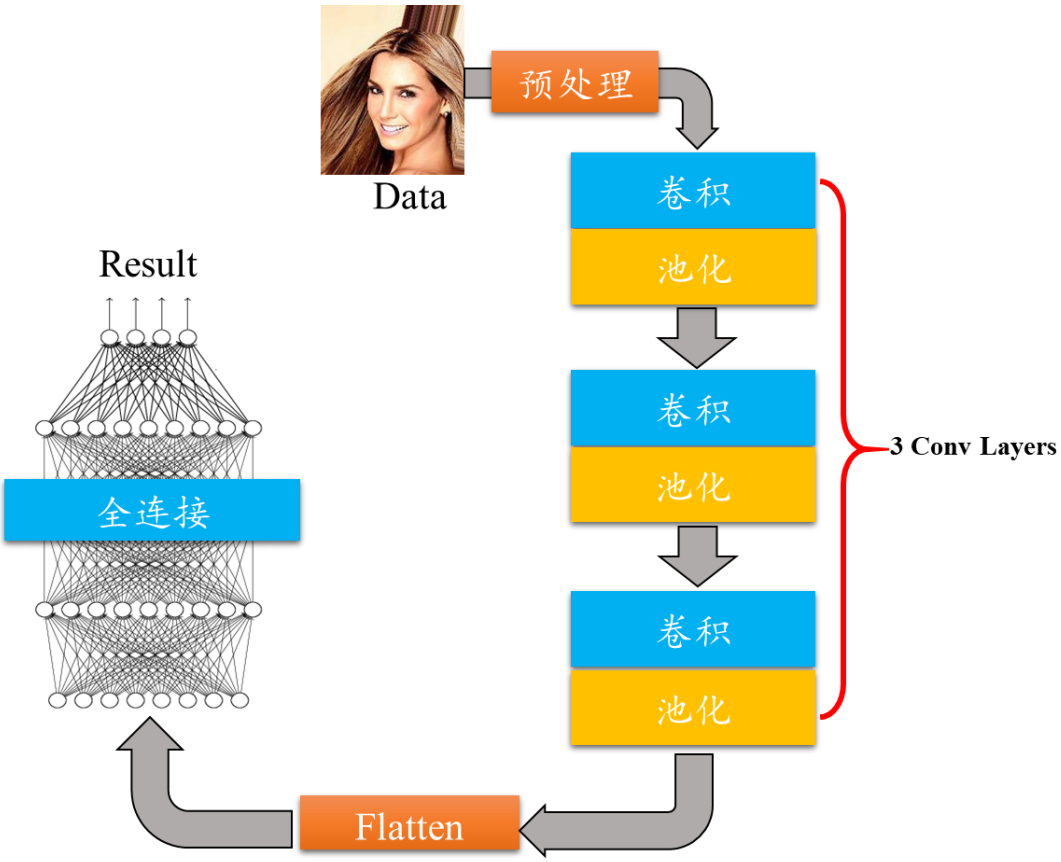

图 模型结构图

3.3 算法参数调优过程

3.3.1 原始图像 vs. 缩小后的灰度图

最初直接输入的是原始图像的矩阵（大小为[218, 178, 3]），因为实验机器的内存支持不了，然后训练过程中就会发生内存错误。如下图所示。

```
MemoryError                               Traceback (most recent call la
st)
<ipython-input-4-f0c5b4df89e6> in <module>
     22     celeb_labels.append(labels)
     23
---> 24 celeb_images = np.asarray(celeb_images, dtype=np.float32)
     25 print(np.shape(celeb_images))
     26 celeb_labels = np.asarray(celeb_labels, dtype=np.float32)

~\Anaconda3\envs\tf_gpu_v1\lib\site-packages\numpy\core\numeric.py in
asarray(a, dtype, order)
    536
    537     """
--> 538     return array(a, dtype, copy=False, order=order)
    539
    540
```

图 内存不足错误

将图像转成灰度图并按比例进行缩小（大小为[128,104,1]），模型正常运行。如下图。

```
epoch 12: training loss: 0.158681, training accuracy 0.94
epoch 12: training loss: 0.0909782, training accuracy 0.96
epoch 12: training loss: 0.114548, training accuracy 0.97
epoch 12: training loss: 0.117793, training accuracy 0.96
epoch 12: training loss: 0.131567, training accuracy 0.93
epoch 12: training loss: 0.134363, training accuracy 0.97
epoch 12: training loss: 0.109571, training accuracy 0.98
epoch 12: training loss: 0.152619, training accuracy 0.95
epoch 12: training loss: 0.15043, training accuracy 0.94
epoch 12: training loss: 0.157327, training accuracy 0.92
epoch 12: training loss: 0.162611, training accuracy 0.94
```

图 正常运行

3.3.2 batch size 的调整

一开始 batch size 过小（20 左右），对于较深的网络结构，准确率会非常不稳定，呈现跳跃现象。如图。

```
epoch 6: training loss: 0.0791107, training accuracy 1
epoch 6: training loss: 0.628506, training accuracy 0.782609
epoch 6: training loss: 0.0901792, training accuracy 0.956522
epoch 6: training loss: 0.111778, training accuracy 1
epoch 6: training loss: 0.0974144, training accuracy 0.956522
epoch 6: training loss: 0.0267809, training accuracy 1
epoch 6: training loss: 0.225426, training accuracy 0.913043
epoch 6: training loss: 0.276072, training accuracy 0.913043
epoch 6: training loss: 0.242672, training accuracy 0.869565
epoch 6: training loss: 0.0844678, training accuracy 1
epoch 6: training loss: 0.0524578, training accuracy 1
epoch 6: training loss: 0.391249, training accuracy 0.826087
```

图 准确率跳跃问题

而且也容易产生过拟合问题，如图。

```
epoch 19: training loss: 0.00317582, training accuracy 1
epoch 19: training loss: 0.00151663, training accuracy 1
epoch 19: training loss: 0.197822, training accuracy 0.913043
epoch 19: training loss: 0.00272031, training accuracy 1
epoch 19: training loss: 0.00917499, training accuracy 1
epoch 19: training loss: 0.0661391, training accuracy 0.956522
epoch 19: training loss: 6.54781e-05, training accuracy 1
epoch 19: training loss: 0.00022182, training accuracy 1
epoch 19: training loss: 0.00937784, training accuracy 1
epoch 19: training loss: 0.00226649, training accuracy 1
```

图 过拟合问题

将 batch size 设得过大（200），则在卷积的过程中会发生内存溢出错误，如图。

```
ResourceExhaustedError: OOM when allocating tensor with shape[200,128,
32,26] and type float on /job:localhost/replica:0/task:0/device:GPU:0
by allocator GPU_0_bfc
         [[{{node adam_optimizer/gradients/pool3/MaxPool_grad/MaxPoolG
rad}}]]
Hint: If you want to see a list of allocated tensors when OOM happens,
add report_tensor_allocations_upon_oom to RunOptions for current alloc
ation info.
```

图 OOM 错误（内存溢出）

将 batch size 设置为合适值（100）后，准确率变得较为稳定，且不再过拟合，如图。

```
epoch 19: training loss: 0.0487557, training accuracy 0.99
epoch 19: training loss: 0.0825783, training accuracy 0.97
epoch 19: training loss: 0.107094, training accuracy 0.96
epoch 19: training loss: 0.0820005, training accuracy 0.96
epoch 19: training loss: 0.0611393, training accuracy 0.97
epoch 19: training loss: 0.0457008, training accuracy 0.98
epoch 19: training loss: 0.0422713, training accuracy 0.99
epoch 19: training loss: 0.0665385, training accuracy 0.97
epoch 19: training loss: 0.0618207, training accuracy 0.97
epoch 19: training loss: 0.0311086, training accuracy 0.99
```

图 恢复正常

3.3.3 dropout 的设置

未加入 dropout 时，在训练中后期会出现过拟合现象，会经常出现准确率为 1，如下图所示。

```
epoch 15: training loss: 0.0450016, training accuracy 0.98
epoch 15: training loss: 0.0568439, training accuracy 0.98
epoch 15: training loss: 0.0366034, training accuracy 0.98
epoch 16: training loss: 0.0419463, training accuracy 0.99
epoch 16: training loss: 0.0262781, training accuracy 1
epoch 16: training loss: 0.0184958, training accuracy 1
epoch 16: training loss: 0.049205, training accuracy 0.97
epoch 16: training loss: 0.0343679, training accuracy 0.99
```

图 过拟合现象

加入 dropout（keep_prob 设为 0.5）后，回归正常，如下图。

```
epoch 15: training loss: 0.116911, training accuracy 0.98
epoch 15: training loss: 0.102713, training accuracy 0.96
epoch 15: training loss: 0.0729282, training accuracy 0.97
epoch 16: training loss: 0.142101, training accuracy 0.94
epoch 16: training loss: 0.119657, training accuracy 0.97
epoch 16: training loss: 0.12636, training accuracy 0.95
epoch 16: training loss: 0.0737885, training accuracy 0.98
epoch 16: training loss: 0.0776683, training accuracy 0.99
```

图 正常现象

4. 结果分析

4.1 实验参数（部分）

一些主要的实验参数如下表所示。

表 主要的实验参数信息

参数名	值

原图大小	h: 218, w: 178, c: 3
预处理后的图片大小	h: 128, w: 104, c: 1
训练集与测试集比例	9:1
Batch size	100
卷积核大小	5x5
Epoch	20
Dropout（keep_prob）	0.5
优化器	Adam
学习率	1e-4

4.2 实验结果

模型训练时 loss 的收敛速度非常快，训练集的准确率最后也稳定在 0.95 左右。在如下图所示。

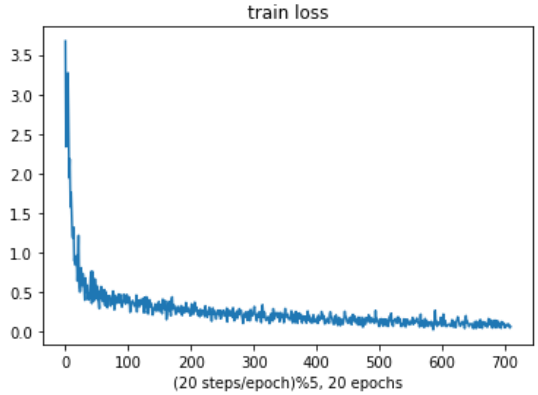

图 训练时的 loss 变化

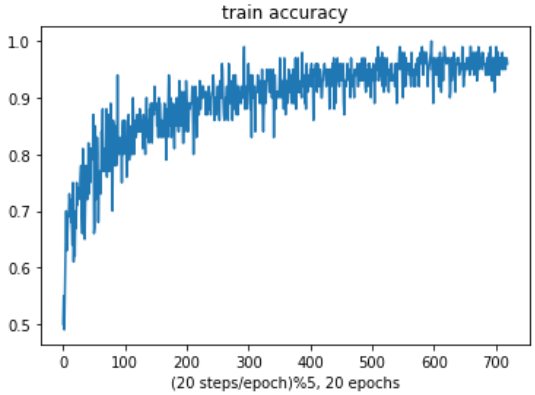

图 训练时的准确率变化

在测试集的平均表现也较好，为 0.931，如下图所示。

```
In [9]:
# test
test_acc=[]
total_batch = int(test_images.shape[0] / batch_size)
for i in range(total_batch):
    batch_test_x, batch_test_y = test_images[i * batch_size: (i + 1) * batch_size], test_labels[
                                                                                    i * batch_size: (i + 1) * batch_size]
    acc=sess.run(accuracy, feed_dict={x: batch_test_x, y_: batch_test_y, keep_prob: 1.0})
    test_acc.append(acc)
print("test accuracy: %g"%np.mean(test_acc))

test accuracy: 0.931
```

图 测试时的准确率

使用模型分别测试了 3 个属性：性别（Male 列），眼镜（Eyeglasses 列），笑容（Smiling 列）。对于它们在数据集中的分布各不相同。测试集的分类准确率如下表所示。

表 测试属性准确率分布

识别属性	分类准确率
性别（Male 列）	0.931
眼镜（Eyeglasses 列）	0.962
笑容（Smiling 列）	0.891
平均值	0.928

4.3 结果分析

之所以能够取得收敛速度较快、准确率较高，我认为原因有以下几点：

1. 模型设计合理且复杂度合适。实验采用了 3 层卷积加上 2 层全连接，对于这个数据集的图片的复杂程度来说，已经足够。
2. 数据集规模和质量满足训练模型的需求。原数据约有 20 万张图片以及对应的标签信息。实验中只用了 2 万张就取得了较好的实验效果。
3. 图片经过了缩小且变成了灰度图。如果未使用缩小后的灰度图作为训练数据，则实验对机器的性能和模型都会有更高的要求。经过处理后的数据既缩小了模型的参数量，又提升了训练效率，间接提升了相同模型条件下的准确率。
4. 训练时 batch size 合适。在调整了几次 batch size 后，发现其对模型的训练影响很大，当 batch size 过小时，训练变得不稳定，且容易出现过拟合现象；当它过大时，CNN 的参数量会爆发式增长而产生内存溢出问题。
5. Dropout 的加入。在未设定 Dropout 前，出现了过拟合问题，设定后（keep_prob=0.5）问题消失。

5. 结论

对于人脸识别，本实验使用卷积神经网络对 CelebFaces 数据集中任一指定属性进行识别。首先通过对图片的分析，决定采用对图片进行缩小和转成灰度图的预处理方式。预处理后，按 9:1 的比例划分训练集和测试集，对标签采用 one hot 编码。训练模型时，通过使用 dropout 和适合的 batch size 来防止过拟合或内存溢出等问题。经过以上的改进后，模型的收敛速度明显加快，训练效果也更加稳定。最后，测试的 3 个属性均取得了良好的效果，平均准确率为 0.928。

另外，我觉得这个数据集可以结合 GAN 做一件更有趣的事：根据语义生成带相关属性的人脸。