课程设计--基于卷积神经网路的人脸性别识别分类

学院：电子科学与工程学院 专业：电子科学与技术 姓名：陈可馨 学号：201718JH02002

# 背景介绍

卷积神经网络

卷积神经网络最早是为了解决图像识别的问题,现在也用在时间序列数据和文本数据处理当中,卷积神经网 络对于数据特征的提取不用额外进行,在对网络的训练的过程当中,网络会自动提取主要的特征. 卷积神经网络直接用原始图像的全部像素作为输入,但是内部为非全连接结构.因为图像数据在空间上是有 组织结构的,每一个像素在空间上和周围的像素是有关系的,和相距很远的像素基本上是没什么联系的,每个 神经元只需要接受局部的像素作为输入,再将局部信息汇总就能得到全局信息. 权值共享和池化两个操作使 网络模型的参数大幅的减少,提高了模型的训练效率.

卷积神经网络主要特点

权值共享: 在卷积层中可以有多个卷积核,每个卷积核与原始图像进行卷积运算后会映射出一个新的2D图 像,新图像的每个像素都来自同一个卷积核.这就是权值共享.

池化: 降采样,对卷积(滤波)后,经过激活函数处理后的图像,保留像素块中灰度值最高的像素点(保留最主要 的特征),比如进行 2X2的最大池化,把一个2x2的像素块降为1x1的像素块.

# 课程设计

课程设计主要是实现了根据人脸识别性别的卷积神经网络,并对卷积过程中的提取特征进行了可视化.

## 实验数据sample

男性数据 ./images/male.png



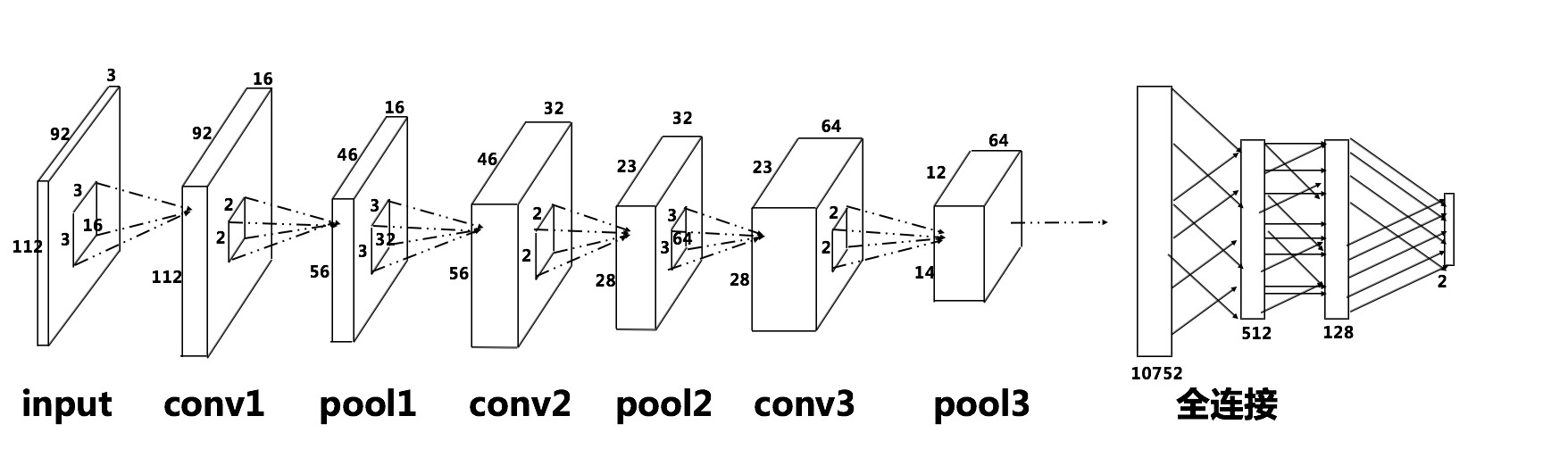
女性数据 ./images/female.png



## 功能介绍

课程设计主要是实现了根据人脸识别性别的卷积神经网络,并对卷积过程中的提取特征进行了可视化。 这实际上 是一个数据挖掘中的分类器问题和计算机视觉联系，最终目标是实现对图片中人脸为男/女进行预测。 通过对 训练集数据进行模型训练，采用cnn卷积神经网络，对模型进行交叉验证，模型具有较好的准确率，各分类下f- score值均表现良好。

## 网络结构



输入层为输入的灰度图像尺寸: -1 x 112 x 92 x 3

第一个卷积层,卷积核的大小,深度和数量 (3, 3, 3, 16)

池化后的特征张量尺寸: -1 x 56 x 46 x 16

第二个卷积层,卷积核的大小,深度和数量 (3, 3, 16, 32)

池化后的特征张量尺寸: -1 x 28 x 23 x 32

第三个卷积层,卷积核的大小,深度和数量 (3, 3, 32, 64)

池化后的特征张量尺寸: -1 x 14 x 12 x 64

全连接第一层权重矩阵: 10752 x 512

全连接第二层权重矩阵: 512 x 128

输出层与全连接隐藏层之间: 128 x 2

运行方法

运行环境

Python3+Anaconda包管理

运行次序

gender\_train\_data.py 完成对训练图片的读取 依赖包：os,numpy,opencv gender\_training.py 训练集上完成模型的建立+在测试集上打印结果报告到命令行 依赖包： tensorflow, gender\_train\_data, matplotlib,sklern gender\_model\_use.py运用模型进行人脸预测 依赖包：tensorflow,numpy, gender\_train\_data, matplotlib, os

## 运行结果

**模型保存**

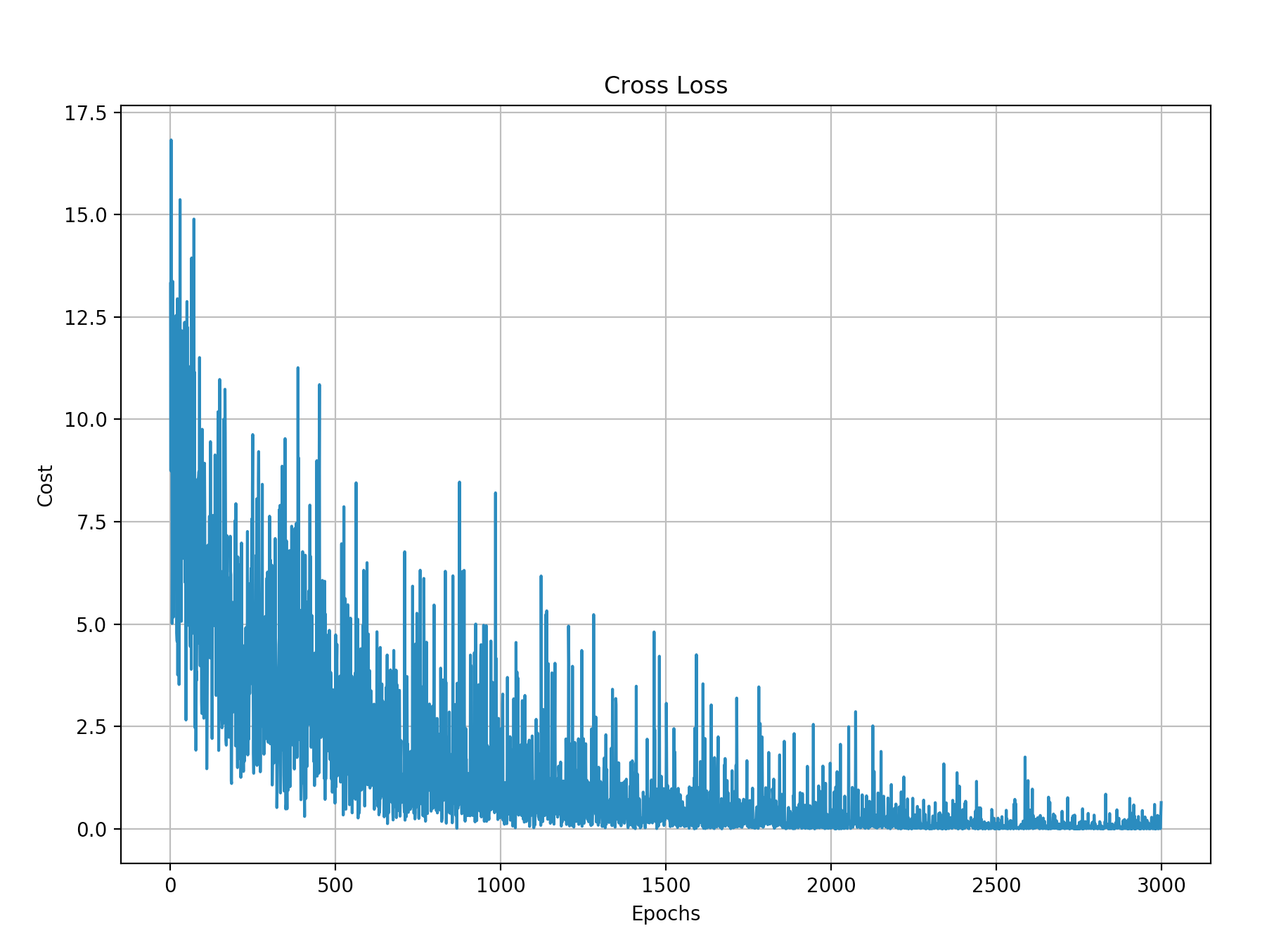
详见./model文件夹内保存了生成的模型

**结果可视化**

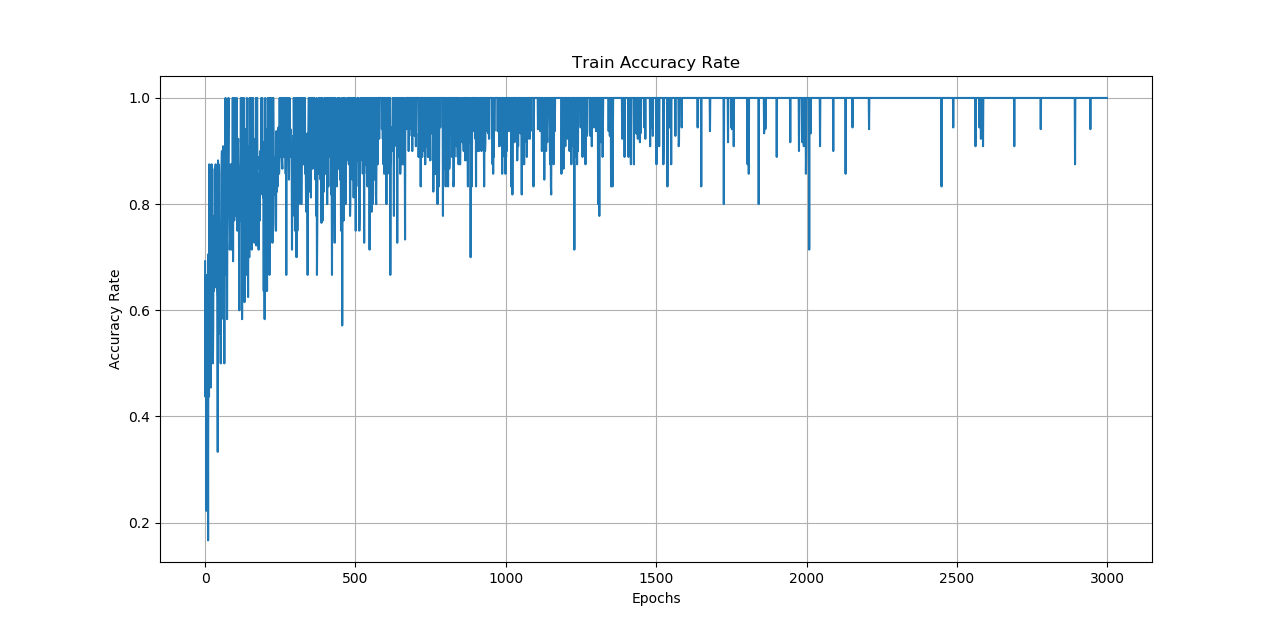
训练集上的准确率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | precision  0.95 | recall  0.98 | f1-score  0.97 | support  43 |
| 1 | 0.97 | 0.95 | 0.96 | 37 |
| accuracy |  |  | 0.96 | 80 |
| macro avg | 0.96 | 0.96 | 0.96 | 80 |
| weighted avg | 0.96 | 0.96 | 0.96 | 80 |

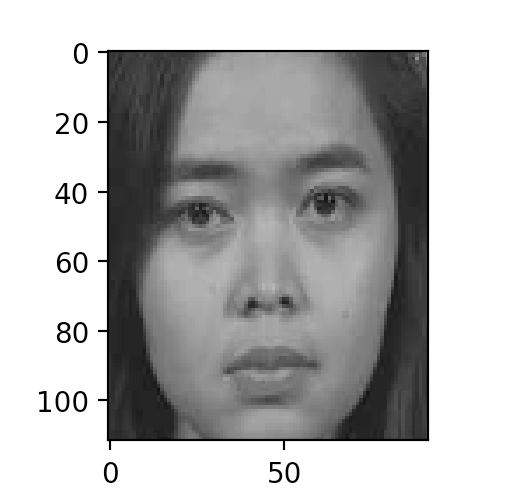
训练交叉熵代价



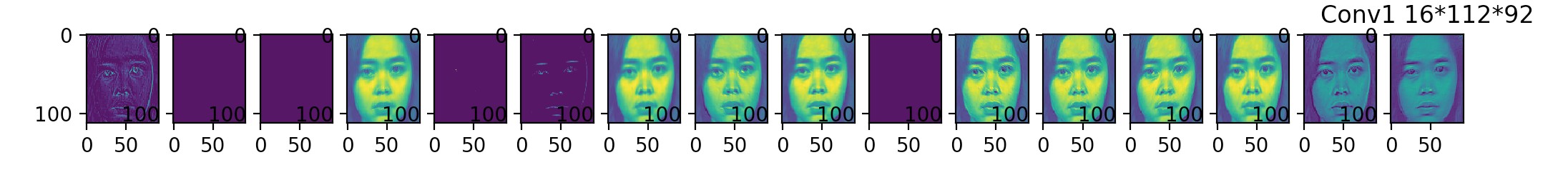
训练的准确率



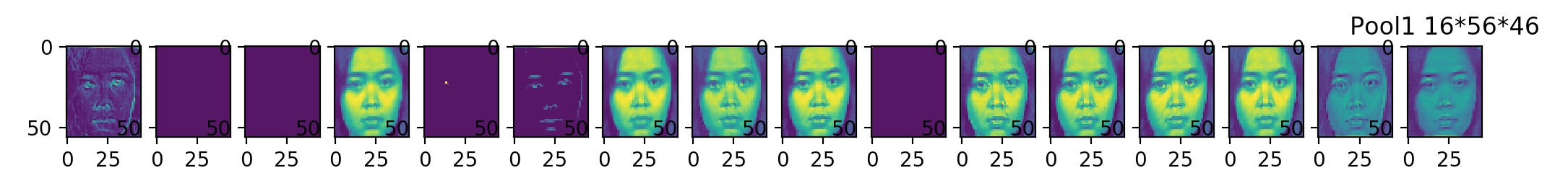
训练数据中的一个样本



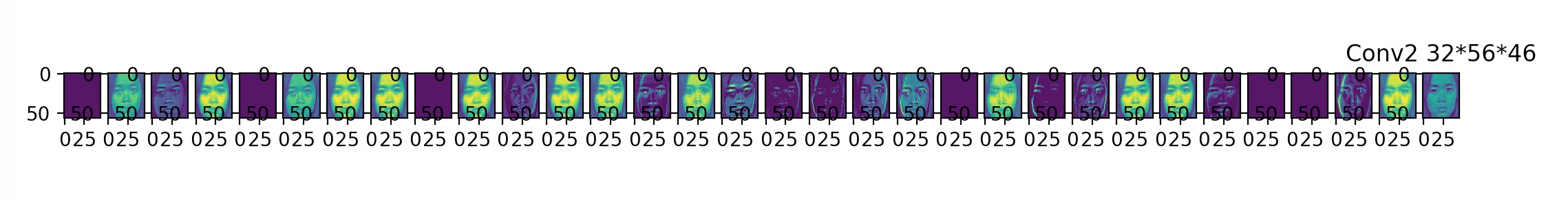
第一层卷积提取的特征



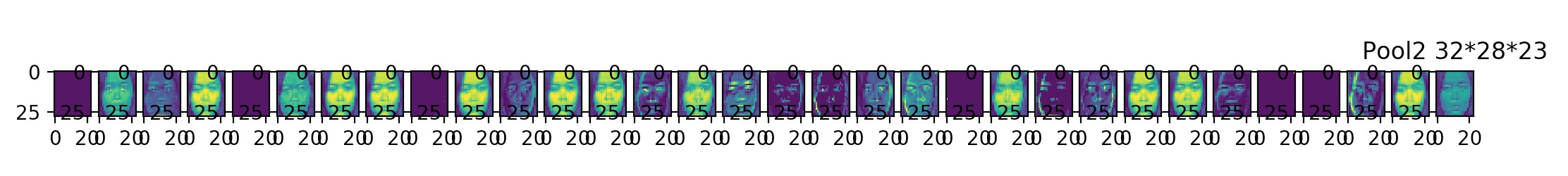
2x2池化后特征



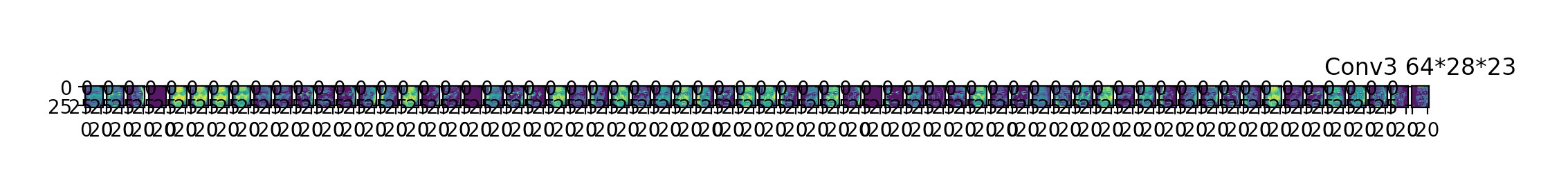
第二层卷积提取的特征



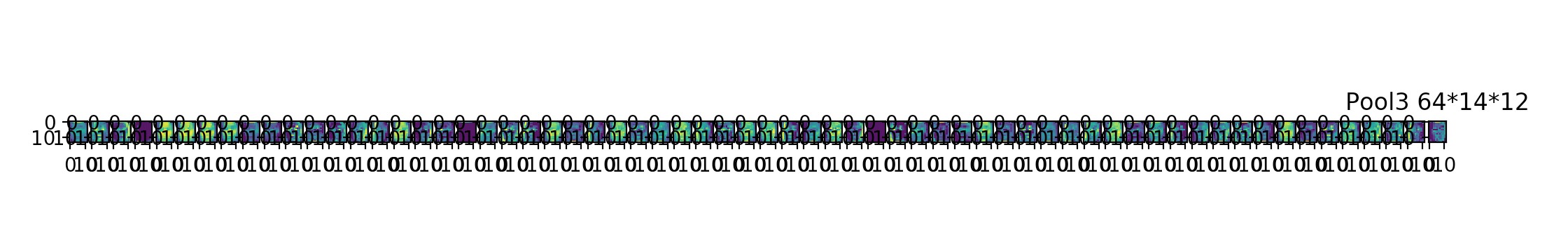
2x2池化后特征



第三层卷积提取的特征



2x2池化后特征



# --- 附录：核心代码解析 ---

## 卷积网络的训练数据（112\*92\*3图形）

从data目录读取数据，famale存放女性图片，male存放男性图片

def read\_img(list,flag=0):

for i in range(len(list)-1):

if os.path.isfile(list[i]): images.append(cv2.imread(list[i]).flatten()) labels.append(flag)

read\_img(get\_img\_list('male'),[0,1]) read\_img(get\_img\_list('female'),[1,0])

images = np.array(images) labels = np.array(labels)

重新打乱

permutation = np.random.permutation(labels.shape[0]) all\_images = images[permutation,:]

all\_labels = labels[permutation,:]

训练集与测试集比例 8：2

train\_total = all\_images.shape[0] train\_nums= int(all\_images.shape[0]\*0.8) test\_nums = all\_images.shape[0]-train\_nums

#训练集

images = all\_images[0:train\_nums,:] labels = all\_labels[0:train\_nums,:]

#测试集

test\_images = all\_images[train\_nums:train\_total,:] test\_labels = all\_labels[train\_nums:train\_total,:]

## 训练参数

train\_epochs=3000

# 训练轮数

batch\_size= random.randint(6,18) # 每次训练数据,随机

drop\_prob = 0.4 # 正则化,丢弃比例

learning\_rate=0.00001 # 学习效率

辅助函数

# 权重初始化(卷积核初始化)

# tf.truncated\_normal()不同于tf.random\_normal(),返回的值中不会偏离均值两倍的标准差

# 参数shpae为一个列表对象,例如[5, 5, 1, 32]对应

# 5,5 表示卷积核的大小, 1代表通道channel,对彩色图片做卷积是3,单色灰度为1

# 最后一个数字32,卷积核的个数,(也就是卷基层提取的特征数量)

def weight\_init(shape):

weight = tf.truncated\_normal(shape,stddev=0.1,dtype=tf.float32) return tf.Variable(weight)

#偏执初始化

def bias\_init(shape):

bias = tf.random\_normal(shape,dtype=tf.float32) return tf.Variable(bias)

#全连接矩阵初始化

def fch\_init(layer1,layer2,const=1):

min = -const \* (6.0 / (layer1 + layer2)); max = -min;

weight = tf.random\_uniform([layer1, layer2], minval=min, maxval=max, dtype=tf.float32)

return tf.Variable(weight)

def conv2d(images,weight):

return tf.nn.conv2d(images,weight,strides=[1,1,1,1],padding='SAME')

Padding

#池化 卷积核在提取特征时的动作成为padding，它有两种方式：SAME和VALID。这里采用SAME def max\_pool2x2(images,tname):

return tf.nn.max\_pool(images,ksize=[1,2,2,1],strides= [1,2,2,1],padding='SAME',name=tname)

#images\_input 为输入的图片，labels\_input为输入的标签

images\_input = tf.placeholder(tf.float32,[None,112\*92\*3],name='input\_images') labels\_input = tf.placeholder(tf.float32,[None,2],name='input\_labels')

#把图像转换为112\*92\*3的形状

x\_input = tf.reshape(images\_input,[-1,112,92,3])

## 训练

第一层卷积+池化

# 卷积核3\*3\*3 16个

第一层卷积

w1 = weight\_init([3,3,3,16])

b1 = bias\_init([16])

conv\_1 = conv2d(x\_input,w1)+b1

relu\_1 = tf.nn.relu(conv\_1,name='relu\_1') max\_pool\_1 = max\_pool2x2(relu\_1,'max\_pool\_1')

第二层卷积+池化

# 卷积核3\*3\*16 32个 第二层卷积 w2 = weight\_init([3,3,16,32]) b2 = bias\_init([32])

conv\_2 = conv2d(max\_pool\_1,w2) + b2 relu\_2 = tf.nn.relu(conv\_2,name='relu\_2')

max\_pool\_2 = max\_pool2x2(relu\_2,'max\_pool\_2')

第三层卷积+池化

w3 = weight\_init([3,3,32,64]) b3 = bias\_init([64])

conv\_3 = conv2d(max\_pool\_2,w3)+b3

relu\_3 = tf.nn.relu(conv\_3,name='relu\_3') max\_pool\_3 = max\_pool2x2(relu\_3,'max\_pool\_3')

全连接第一层

#把第三层的卷积结果平铺成一维向量

f\_input = tf.reshape(max\_pool\_3,[-1,14\*12\*64])

#全连接第一层 31\*31\*32,512 f\_w1= fch\_init(14\*12\*64,512) f\_b1 = bias\_init([512])

f\_r1 = tf.matmul(f\_input,f\_w1) + f\_b1

#激活函数，relu随机丢掉一些权重提供泛华能力 f\_relu\_r1 = tf.nn.relu(f\_r1)

# 为了防止网络出现过拟合的情况,对全连接隐藏层进行 Dropout(正则化)处理,在训练过程中随机的 丢弃部分

# 节点的数据来防止过拟合.Dropout同把节点数据设置为0来丢弃一些特征值,仅在训练过程中,

# 预测的时候,仍使用全数据特征

# 传入丢弃节点数据的比例

f\_dropout\_r1 = tf.nn.dropout(f\_relu\_r1,drop\_prob)

全连接第二层

f\_w2 = fch\_init(512,128) f\_b2 = bias\_init([128])

f\_r2 = tf.matmul(f\_dropout\_r1,f\_w2) + f\_b2 f\_relu\_r2 = tf.nn.relu(f\_r2)

f\_dropout\_r2 = tf.nn.dropout(f\_relu\_r2,drop\_prob)

全连接输出层

f\_w3 = fch\_init(128,2) f\_b3 = bias\_init([2])

f\_r3 = tf.matmul(f\_dropout\_r2,f\_w3) + f\_b3 最后输出结果，可能是这样的[[0.0001,0.99999] ，那个位置的结果大就属于哪个分类 f\_softmax = tf.nn.softmax(f\_r3,name='f\_softmax')

损失函数

#交叉熵代价函数

cross\_entry = tf.reduce\_mean(tf.reduce\_sum(-labels\_input\*tf.log(f\_softmax)))

#优化器，自动执行梯度下降算法

optimizer = tf.train.AdamOptimizer(learning\_rate).minimize(cross\_entry)

计算准确率&损失

arg1 = tf.argmax(labels\_input,1) arg2 = tf.argmax(f\_softmax,1)

#每个样本的预测结果是一个(1,2)的vector

cos = tf.equal(arg1,arg2)

# tf.cast把bool值转换为浮点数

acc = tf.reduce\_mean(tf.cast(cos,dtype=tf.float32))

启动会话开始训练

init = tf.global\_variables\_initializer() sess = tf.Session()

sess.run(init) Cost = [] Accuracy=[]

for i in range(train\_epochs): idx=random.randint(0,len(train\_data.images)-20) batch= random.randint(6,18)

train\_input = train\_data.images[idx:(idx+batch)] train\_labels = train\_data.labels[idx:(idx+batch)] result,acc1,cross\_entry\_r,cos1,f\_softmax1,relu\_1\_r=

sess.run([optimizer,acc,cross\_entry,cos,f\_softmax,relu\_1],feed\_dict=

{images\_input:train\_input,labels\_input:train\_labels}) print acc1

Cost.append(cross\_entry\_r) Accuracy.append(acc1)

# 代价函数曲线

fig1,ax1 = plt.subplots(figsize=(10,7)) plt.plot(Cost)

ax1.set\_xlabel('Epochs') ax1.set\_ylabel('Cost') plt.title('Cross Loss') plt.grid()

plt.show()

# 准确率曲线

fig7,ax7 = plt.subplots(figsize=(10,7)) plt.plot(Accuracy) ax7.set\_xlabel('Epochs') ax7.set\_ylabel('Accuracy Rate') plt.title('Train Accuracy Rate') plt.grid()

plt.show()

测试集验证

arg2\_r = sess.run(arg2,feed\_dict=

{images\_input:train\_data.test\_images,labels\_input:train\_data.test\_labels}) arg1\_r = sess.run(arg1,feed\_dict=

{images\_input:train\_data.test\_images,labels\_input:train\_data.test\_labels})

#使用混淆矩阵，打印报告

print (classification\_report(arg1\_r, arg2\_r))

验证通过，保存模型

#保存模型

saver = tf.train.Saver()

saver.save(sess, './model/my-gender-v1.0')