

课程设计--基于卷积神经网络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`



face166.bmp



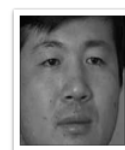
face167.bmp



face168.bmp



face169.bmp



face170.bmp



face171.bmp



face172.bmp



face173.bmp



face174.bmp



face175.bmp

- 女性数据 ./images/female.png



face201.bmp



face202.bmp



face203.bmp



face204.bmp



face205.bmp



face206.bmp



face207.bmp



face208.bmp



face209.bmp

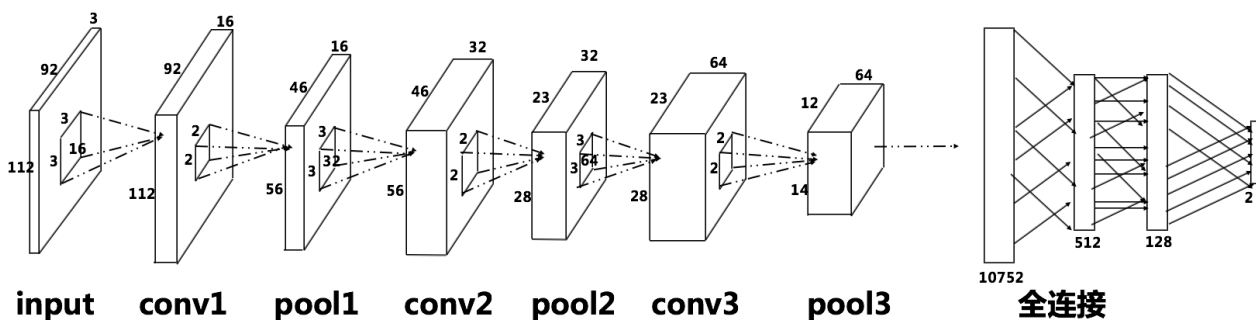


face210.bmp

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: $-1 \times 112 \times 92 \times 3$
 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: $-1 \times 56 \times 46 \times 16$
 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: $-1 \times 28 \times 23 \times 32$
 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: $-1 \times 14 \times 12 \times 64$
 全连接第一层权重矩阵: 10752×512
 全连接第二层权重矩阵: 512×128
 输出层与全连接隐藏层之间: 128×2

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

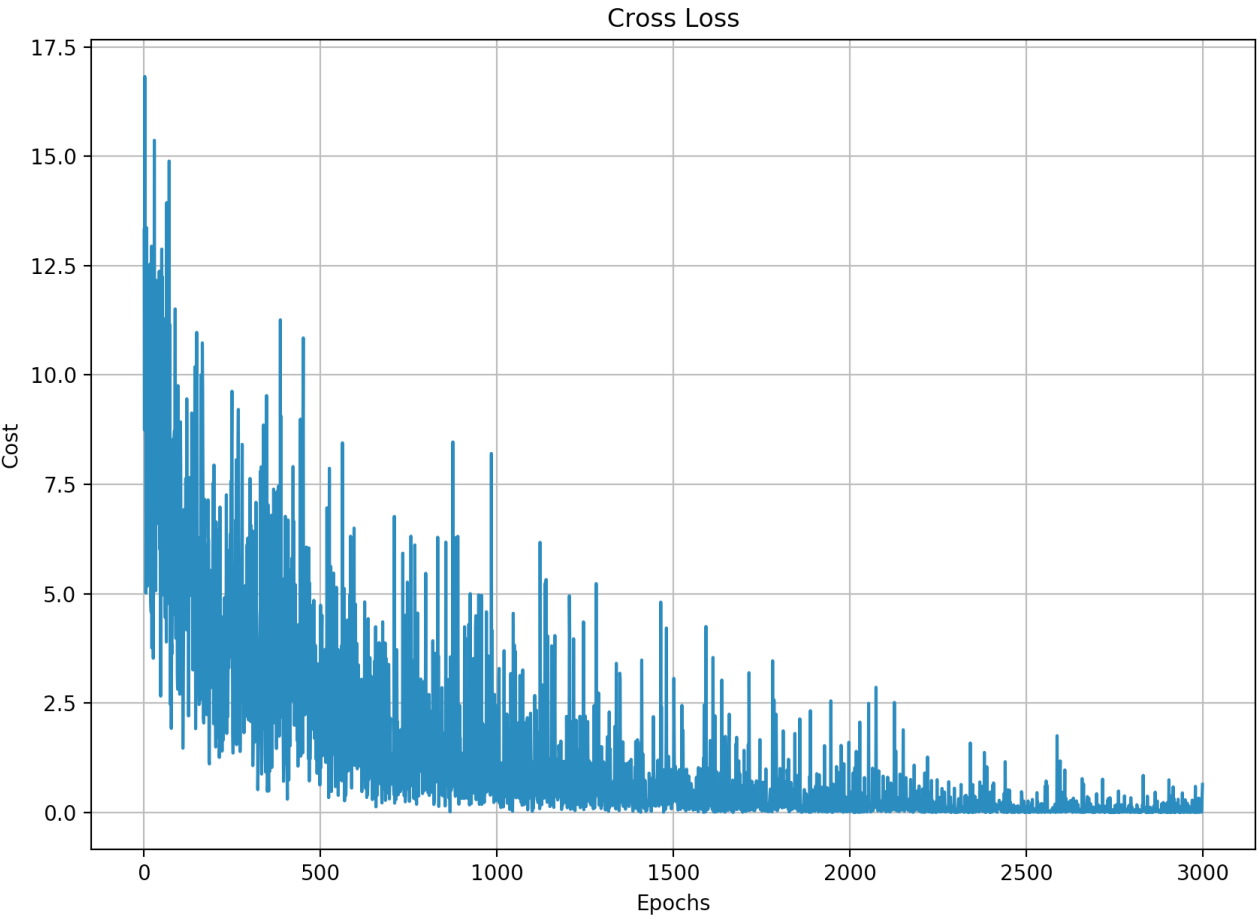
- 详见./model文件夹内保存了生成的模型

结果可视化

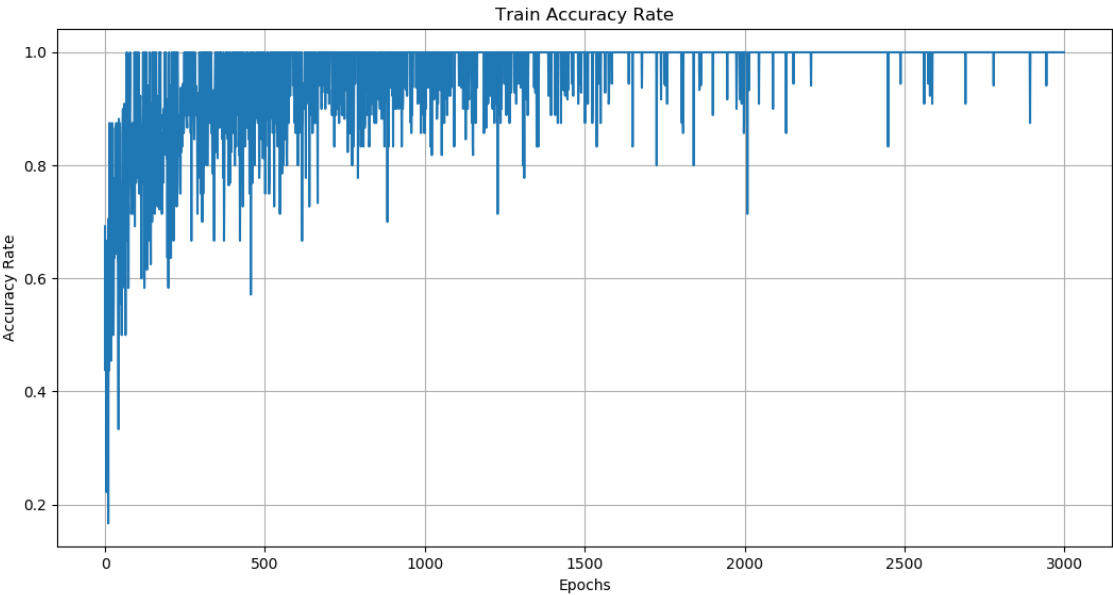
训练集上的准确率

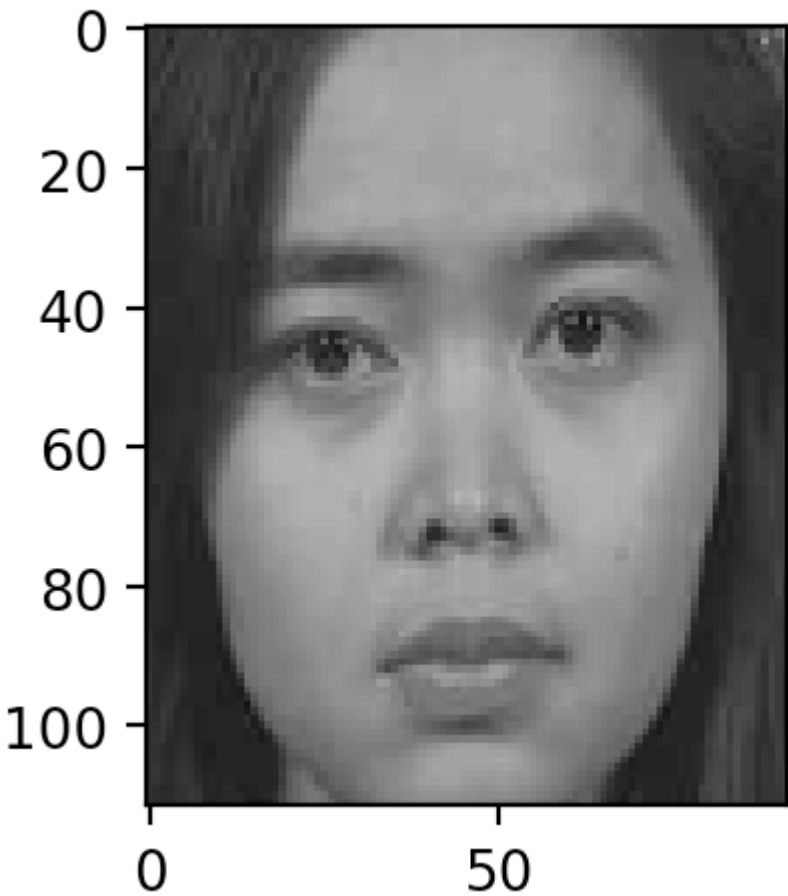
	precision	recall	f1-score	support
0	0.95	0.98	0.97	43
1	0.97	0.95	0.96	37
accuracy			0.96	80
macro avg	0.96	0.96	0.96	80
weighted avg	0.96	0.96	0.96	80

训练交叉熵代价



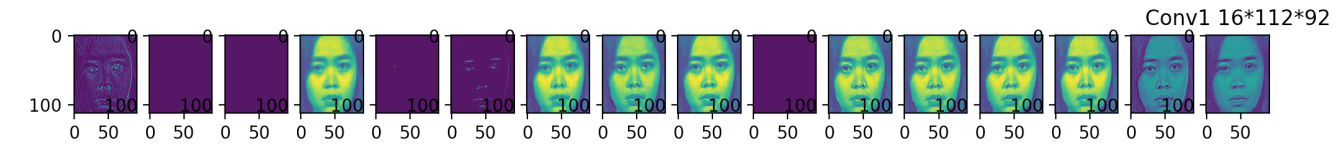
训练的准确率



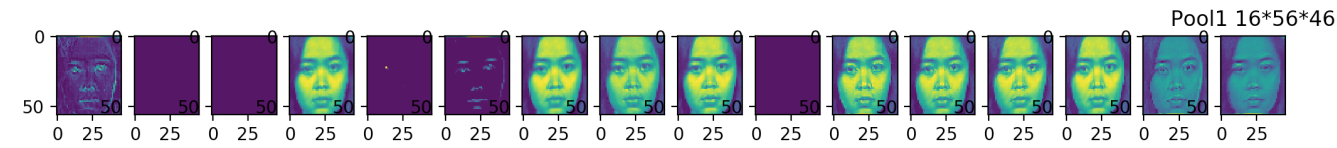


训练数据中的一个样本

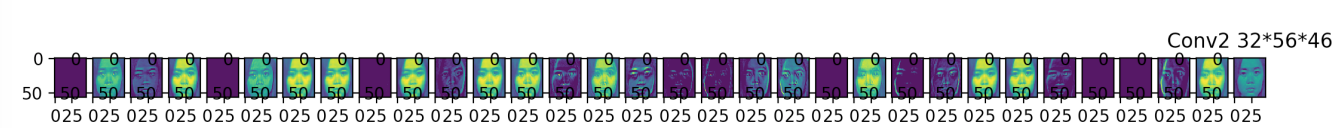
第一层卷积提取的特征



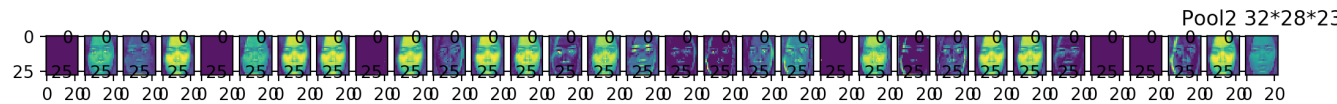
2x2池化后特征



第二层卷积提取的特征



2x2池化后特征



[illegible]

```
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)
```

```
permutation = np.random.permutation(labels.shape[0])
all_images = images[permutation,:]
all_labels = labels[permutation,:]
```

```
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shape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

```
#测试
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')
```