河南理工大学

《深度学习基础与实践》

课程设计报告

（2020—2021 学年第 二 学期）

教师签名：

日期：

目录

基于卷积神经网络的性别识别

摘要

本文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.在本文中,训练数据为112923张男女图片。利用卷积神经网络提取特征并训练模型更新参数,经过迭代更新使得模型对男女性别识别的准确率达到90%以上。

本文主要内容即为

一，介绍卷积神经网络的主要原理

二，介绍卷积神经网络的特点和由此特点衍生出来的用途

三，应用：开始构建模型并进行训练，此条内容为本文的主体。从而进一步可细分为：

1，从data目录读取数据，female存放女性图片，male存放男性图片，并将数据训练集与测试集的比例划分为8：2

2，初始化训练参数，定义各种辅助函数，定义池化层，卷积层，全连接层

3，训练

4，通过交叉熵代价函数（损失函数）计算准确率

5，用测试集的数据进行验证，正确率大于90%就保存模型

通过本文，可以实现对人脸进行男女性别识别。

关键词：卷积神经网络，深度学习，卷积层，池化层，全连接层，反向传播

正文：

一：原理与概念

1.1 关于人工智能与深度学习

1、人工智能

人工智能（Artificial intelligence）简称AI。人工智能是计算机科学的一个分支，它企图了解智能的本质，并生产出一种新的能以人类智能相似的方式

做出反应的智能机器，是研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统的一门新的技术科学。

人工智能目前分为弱人工智能和强人工智能和超人工智能。

1）弱人工智能：弱人工智能（ArtificialNarrow Intelligence /ANI),只专注于完成某个特定的任务，例如语音识别、图象识别和翻译等，是擅长于单个方面的人工智能。它们只是用于解决特定的具体类的任务问题而存在，大都是统计数据，以此从中归纳出模型。由于弱人工智能智能处理较为单一的问题，且发展程度并没有达到模拟人脑思维的程度，所以弱人工智能仍然属于“工具”的范畴，与传统的“产品”在本质上并无区别。

2) 强人工智能：强人工智能（Artificial Generallnteligence /AGI),属于人类级别的人工智能，在各方面都能和人类比肩，它能够进行思考、计划、解决问题、抽象思维、理解复杂理念、快速学习和从经验中学习等操作，并且和人类一样得心应手。

3）超人工智能：超人工智能（Artificial Superintelligence/ASI），在几乎所有领域都比最聪明的人类大脑都聪明许多，包括科学创新、通识和社交技能。在超人工智能阶段，人工智能已经跨过“奇点”，其计算和思维能力已经远超人脑。此时的人工智能已经不是人类可以理解和想象。人工智能将打破人脑受到的维度限制，其所观察和思考的内容，人脑已经无法理解，人工智能将形成一个新的社会。

目前我们仍处于弱人工智能阶段。

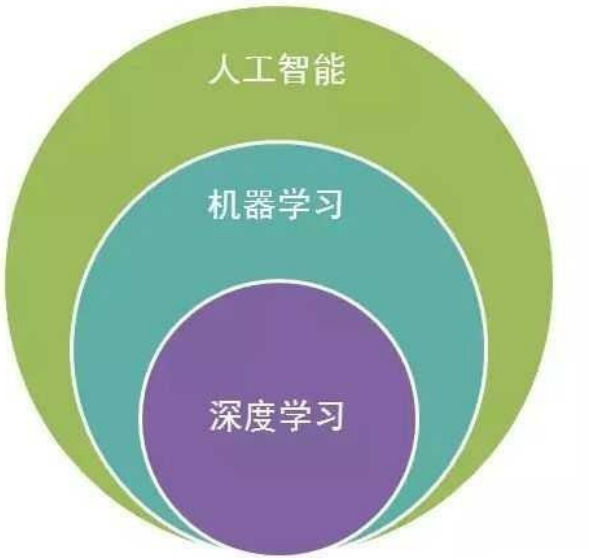
2、机器学习

机器学习(MachineLearning)简称 ML。机器学习属于人工智能的一个分支，也是人工智能的和核心。机器学习理论主要是设计和分析一些让计算机可以自动”学习“的算法。

3、深度学习

深度学习（DeepLearning）简称 DL。最初的深度学习是利用深度神经网络来解决特征表达的一种学习过程。深度神经网络本身并不是一个全新的概念，可大致理解为包含多个隐含层的神经网络结构。为了提高深层神经网络的训练效果，人们对神经元的连接方法和激活函数等方面做出相应的调整。深度学习是机器学习研究中的一个新的领域，其动机在于建立、模拟人脑进行分析学习的神经网络，它模仿人脑的机制来解释数据，如图象、声音、文本。

总结: 机器学习是一种实现人工智能的方法，深度学习是一种实现机器学习的技术,如下图



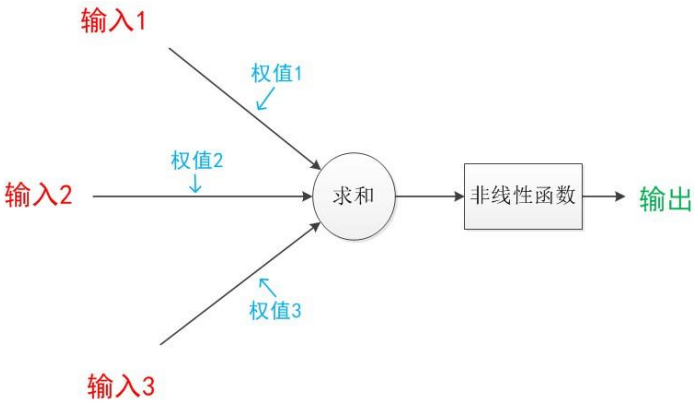
1.2 关于神经网络

神经网络是一门重要的机器学习技术。它是目前最为火热的研究方向--深度学习的基础。神经网络是一种模拟人脑的神经网络以期能够实现类人工智能的机器学习技术。人脑中的神经网络是一个非常复杂的组织。成人的大脑中估计有 1000 亿个神经元之多。

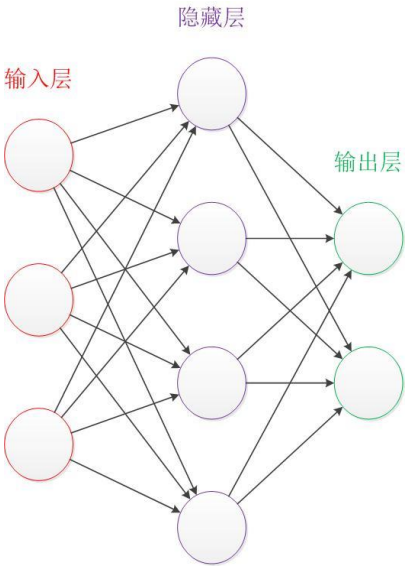
神经元模型是一个包含输入，输出与计算功能的模型。输入可以类比为神经元的树突，而输出可以类比为神经元的轴突，计算则可以类比为细胞核。

下图是一个典型的神经元模型：包含有 3 个输入，1 个输出，以及 2 个计算功能。

注意中间的箭头线。这些线称为“连接”。每个上有一个“权值”。

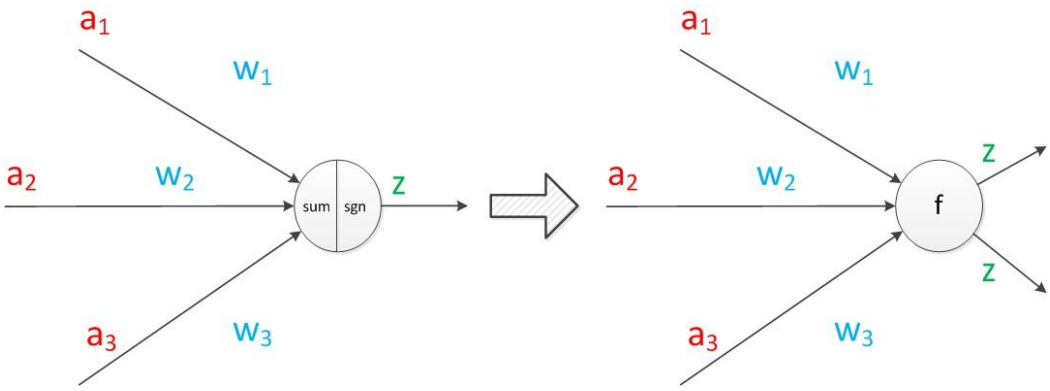


让我们来看一个经典的神经网络。这是一个包含三个层次的神经网络。红色的是输入层，绿色的是输出层，紫色的是中间层（也叫隐藏层）。输入层有 3 个输入单元，隐藏层有 4 个单元，输出层有 2 个单元。后文中，我们统一使用这种颜色来表达神经网络的结构。

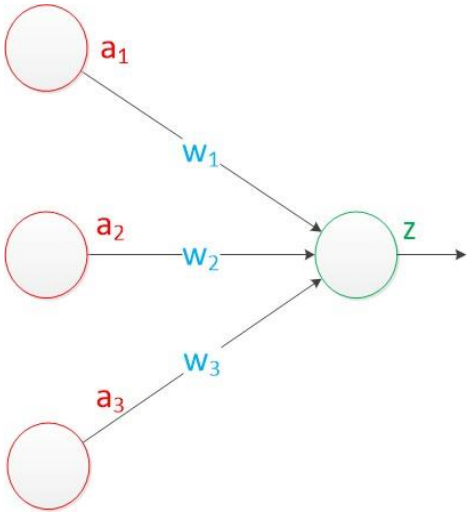


在开始介绍前，有一些知识可以先记在心里：

1. 设计一个神经网络时，输入层与输出层的节点数往往是固定的，中间层则可以自由指定；
2. 神经网络结构图中的拓扑与箭头代表着预测过程时数据的流向，跟训练时的数据流有一定的区别；
3. 结构图里的关键不是圆圈（代表“神经元”），而是连接线（代表“神经元”之间的连接）。每个连接线对应一个不同的权重（其值称为权值），这是需要训练得到的。



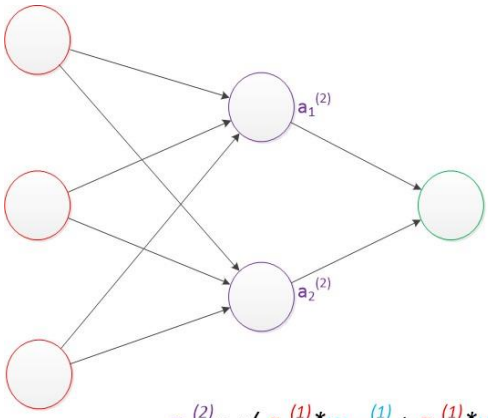
单层神经网络（感知器）：



在“感知器”中，有两个层次。分别是输入层和输出层。输入层里的“输入单元”只负责传输数据，不做计算。输出层里的“输出单元”则需要对前面一层的输入进行计算。

我们把需要计算的层次称之为“计算层”，并把拥有一个计算层的网络称之为“单层神经网络”。有一些文献会按照网络拥有的层数来命名，例如把“感知器”称为两层神经网络。但在本文里，我们根据计算层的数量来命名。

两层神经网络（多层感知器）



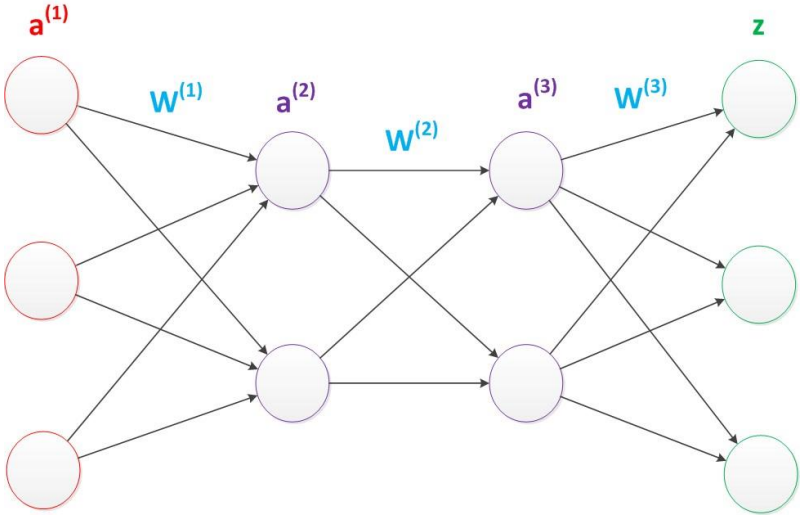
$$a_1^{(2)} = g(a_1^{(1)} * w_{1,1}^{(1)} + a_2^{(1)} * w_{1,2}^{(1)} + a_3^{(1)} * w_{1,3}^{(1)})$$

$$a_2^{(2)} = g(a_1^{(1)} * w_{2,1}^{(1)} + a_2^{(1)} * w_{2,2}^{(1)} + a_3^{(1)} * w_{2,3}^{(1)})$$

与单层神经网络不同。理论证明，两层神经网络可以无限逼近任意连续函数。

这是什么意思呢？也就是说，面对复杂的非线性分类任务，两层（带一个隐藏层）神经网络可以分类的很好。

多层神经网络（深度学习）



多层神经网络中，输出也是按照一层一层的方式来计算。从最外面的层开始，算出所有单元的值以后，再继续计算更深一层。只有当前层所有单元的值都计算完毕以后，才会算下一层。有点像计算向前不断推进的感觉。所以这个过程叫做“正向传播”。

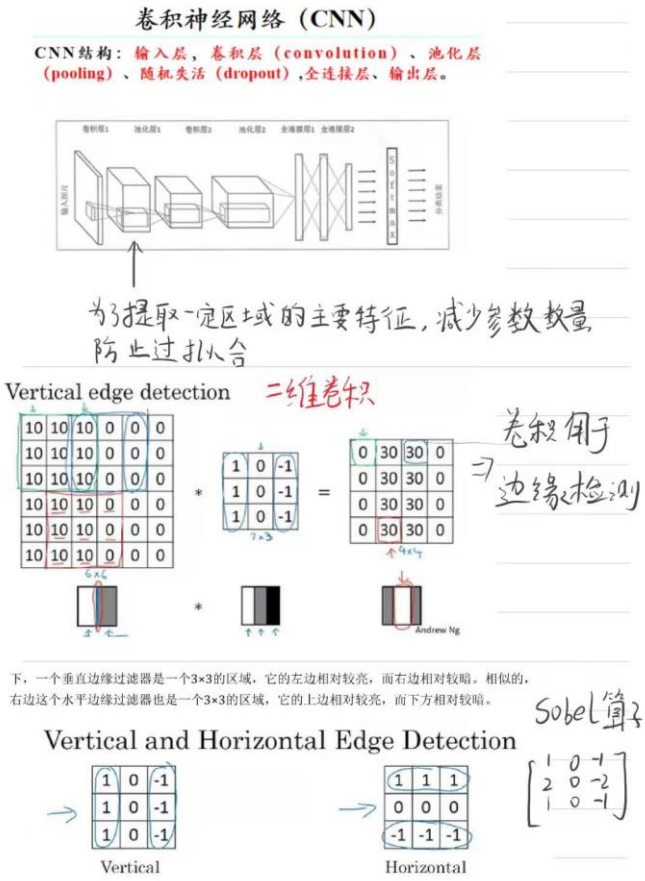
增加更多的层次有什么好处？更深入的表示特征，以及更强的函数模拟能力。更深入的表示特征可以这样理解，随着网络的层数增加，每一层对于前一层次的抽象表示更深入。在神经网络中，每一层神经元学习到的是前一层神经元值的更抽象的表示。例如第一个隐藏层学习到的是“边缘”的特征，第二个隐藏层学习到的是由“边缘”组成的“形状”的特征，第三个隐藏层学习到的是由“形状”组成的“图案”的特征，最后的隐藏层学习到的是由“图案”组成的“目标”的特征。通过抽取更抽象的特征来对事物进行区分，从而获得更好的区分与分类能力。

1.3 卷积神经网络

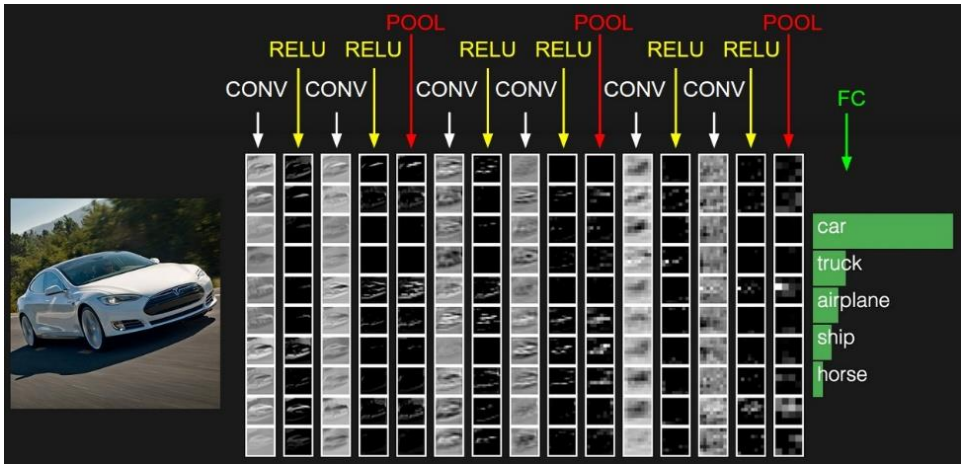
上面了解完完神经网络之后，我们再来了解卷积神经网络。卷积神经网络的最主要特征就是卷积操作

什么是卷积

对图像（不同的数据窗口数据）和滤波矩阵（一组固定的权重：因为每个神经元的多个权重固定，所以又可以看做一个恒定的滤波器 filter）做内积（逐个元素相乘再求和）的操作就是所谓的『卷积』操作，也是卷积神经网络的名字来源



卷积神经网络是一种多层的监督学习神经网络，隐含层的卷积层和池采样层是实现卷积神经网络特征提取功能的核心模块。该网络模型通过采用梯度下降法最小化损失函数对网络中的权重参数逐层反向调节，通过频繁的迭代训练提高网络的精度。卷积神经网络的低隐层是由卷积层和最大池采样层交替组成，高层是全连接层对应传统多层感知器的隐含层和逻辑回归分类器。第一个全连接层的输入是由卷积层和子采样层进行特征提取得到的特征图像。最后一层输出层是一个分类器，可以采用逻辑回归，Softmax 回归甚至是支持向量机对输入图像进行分类。



上图中 CNN 要做的事情是：给定一张图片，是车还是马未知，是什么车也未知，现在需要模型判断这张图片里具体是一个什么东西，总之输出一个结果：如果是车 那是什么车？

最左边是数据输入层，对数据做一些处理，比如去均值（把输入数据各个维度都中心化为 0，避免数据过多偏差，影响训练效果）、归一化（把所有的数据都归一到同样的范围）、PCA/白化等等。CNN 只对训练集做“去均值”这一步。

中间是：

CONV：卷积计算层，线性乘积 求和。

RELU：激励层，上文 2.2 节中有提到：ReLU 是激活函数的一种。

POOL：池化层，简言之，即取区域平均或最大。

最右边是：

FC：全连接层

这几个部分中，卷积计算层是 CNN 的核心。

卷积神经网络的特征：

CNN 的特征

1）具有一些传统技术所没有的优点：良好的容错能力、并行处理能力和自学习能力，可处理环境信息复杂，背景知识不清楚，推理规则不明确情况下的问题，允许样品有较大的缺损、畸变，运行速度快，自适应性能好，具有较高的分辨率。它是通过结构重组和减少权值将特征抽取功能融合进多层感知器，省略识别前复杂的图像特征抽取过程。

2）泛化能力要显著优于其它方法，卷积神经网络已被应用于模式分类，物体检测和物体识别等方面。利用卷积神经网络建立模式分类器，将卷积神经网络作为通用的模式分类器，直接用于灰度图像。

3）是一个前馈式神经网络，能从一个二维图像中提取其拓扑结构，采用反向传播算法来优化网络结构，求解网络中的未知参数。

4）一类特别设计用来处理二维数据的多层神经网络。CNN 被认为是第一个真正成功的采用多层层次结构网络的具有鲁棒性的深度学习方法。CNN 通过挖掘数据中的空间上的相关性，来减少网络中的可训练参数的数量，达到改进前向传播网络的反向传播算法效率，因为 CNN 需要非常少的数据预处理工作，所以也被认为是一种深度学习的方法。在 CNN 中，图像中的小块区域（也叫做“局部感知区域”）被当做层次结构中的底层的输入数据，信息通过前向传播经过网络中的各个层，

在每一层中都由过滤器构成，以便能够获得观测数据的一些显著特征。因为局部感知区域能够获得一些基础的特征，比如图像中的边界和角落等，这种方法能够提供一定程度对位移、拉伸和旋转的相对不变性。

5）CNN 中层次之间的紧密联系和空间信息使得其特别适用于图像的处理和理解，并且能够自动的从图像抽取出丰富的相关特性。

6）CNN 通过结合局部感知区域、共享权重、空间或者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，优化网络结构，并且保证一定程度上的位移和变形的不变性。

7）CNN 是一种深度的监督学习下的机器学习模型，具有极强的适应性，善于挖掘数据局部特征，提取全局训练特征和分类，它的权值共享结构网络使之更类似于生物神经网络，在模式识别各个领域都取得了很好的成果。

8） CNN 可以用来识别位移、缩放及其它形式扭曲不变性的二维或三维图像。CNN 的特征提取层参数是通过训练数据学习得到的，所以其避免了人工特征提取，而是从训练数据中进行学习;其次同一特征图的神经元共享权值,减少了网络参数，这也是卷积网络相对于全连接网络的一大优势。共享局部权值这一特殊结构更接近于真实的生物神经网络使 CNN 在图像处理、语音识别领域有着独特的优越性，另一方面权值共享同时降低了网络的复杂性，且多维输入信号（语音、图像）可以直接输入网络的特点避免了特征提取和分类过程中数据重排的过程。

9）CNN 的分类模型与传统模型的不同点在于其可以直接将一幅二维图像输入模型中，接着在输出端即给出分类结果。其优势在于不需复杂的预处理，将特征抽取，模式分类完全放入一个黑匣子中，通过不断的优化来获得网络所需参数，在输出层给出所需分类，网络核心就是网络的结构设计与网络的求解。这种求解结构比以往多种算法性能更高。

10）隐层的参数个数和隐层的神经元个数无关，只和滤波器的大小和滤波器种类的多少有关。隐层的神经元个数,它和原图像,也就是输入的大小(神经元个数)、滤波器的大小和滤波器在图像中的滑动步长都有关。

卷积神经网络的注意事项：
卷积神经网络注意事项

1）数据集的大小和分块

数据驱动的模型一般依赖于数据集的大小，CNN 和其他经验模型一样，能够适用于任意大小的数据集，但用于训练的数据集应该足够大， 能够覆盖问题域中所有已知可能出现的问题，

设计 CNN 的时候，数据集应该包含三个子集：训练集、测试集、验证集

训练集：包含问题域中的所有数据，并在训练阶段用来调整网络的权重

测试集：在训练的过程中用于测试网络对训练集中未出现的数据的分类性能，根据网络在测试集上的性能情况，网络的结构可能需要作出调整，或者增加训练循环次数。

验证集：验证集中的数据统一应该包含在测试集和训练集中没有出现过的数据，用于在网络确定之后能够更好的测试和衡量网络的性能

Looney 等人建议，数据集中 65%的用于训练，25%的用于测试，10%用于验证

2）数据预处理

为了加速训练算法的收敛速度，一般都会采用一些数据预处理技术，其中包括：去除噪声、输入数据降维、删除无关数据等。

数据的平衡化在分类问题中异常重要，一般认为训练集中的数据应该相对于标签类别近似于平均分布，也就是每一个类别标签所对应的数据集在训练集中是基本相等的，以避免网络过于倾向于表现某些分类的特点。

为了平衡数据集，应该移除一些过度富余的分类中的数据，并相应补充一些相对样例稀少的分类中的数据。

还有一个方法就是复制一部分这些样例稀少分类中的数据，并在这些数据中加入随机噪声。

3）数据规则化

将数据规则化到统一的区间（如[0,1]）中具有很重要的优点：防止数据中存在较大数值的数据造成数值较小的数据对于训练效果减弱甚至无效化，一个常用的方法是将输入和输出数据按比例调整到一个和激活函数相对应的区间。

4）网络权值初始化

CNN 的初始化主要是初始化卷积层和输出层的卷积核（权值）和偏置

网络权值初始化就是将网络中的所有连接权重赋予一个初始值，如果初始权重向量处在误差曲面的一个相对平缓的区域的时候，网络训练的收敛速度可能会很缓慢，一般情况下网络的连接权重和阈值被初始化在一个具有 0 均值的相对小的区间内均匀分布。

5）**收敛条件：**有几个条件可以作为停止训练的判定条件，训练误差、误差梯度、交叉验证等。一般来说，训练集的误差会随着网络训练的进行而逐步降低。

二，研究与设计的背景以及模型构建的具体过程

2.1 研究或设计背景

卷积神经网络广泛应用于计算机视觉领域并用于图像分类，现在有越来越多人脸识别的应用需求。我们要对人脸进行分类，首先就要对图像进行识别并提取特征

图像识别已成为当下的主流，每天都有成千上万的公司和数百万的消费者在使用这项技术。 图像识别由深度学习提供动力，

特别是卷积神经网络（CNN），这是一种神经网络体系结构，可模拟视觉皮层如何分解并分析图像数据。

CNN 和神经网络图像识别是计算机视觉深度学习的核心组成部分，它具有许多应用场景，包括电子商务，游戏，汽车，制造业和教育

对于图像分类，若应用普通的神经网络，由于图像的像素点很多，神经网络需要处理的数据量也很庞大。

卷积神经网络的卷积操作可以提取图像特征，简化数据处理量

2.2 算法与原理

- 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.

2.3 设计思路与训练过程

1）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,:]
```

2)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
```

3）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

4）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)

# 源码的位置在 tensorflow/python/ops 下 nn_impl.py 和 nn_ops.py
# 这个函数接收两个参数,x 是图像的像素, w 是卷积核
# x 张量的维度[batch, height, width, channels]
# w 卷积核的维度[height, width, channels, channels_multiplier]
# tf.nn.conv2d()是一个二维卷积函数,
# stirdes 是卷积核移动的步长,4 个 1 表示,在 x 张量维度的四个参数上移动步
```

长

```
# padding 参数'SAME',表示对原始输入像素进行填充,卷积后映射的 2D 图像与原图大小相等
# 填充,是指在原图像素值矩阵周围填充 0 像素点
# 如果不进行填充,假设 原图为 32x32 的图像,卷积和大小为 5x5 ,卷积后映射图像大小 为 28x28
def conv2d(images,weight):
    return
tf.nn.conv2d(images,weight,strides=[1,1,1,1],padding='SAME')
```

#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。卷积核的移动步长不一定能够整除图片像素的宽度，所以在有些图片的边框位置有些像素不能被卷积。这种不越过边缘的取样就叫做 valid padding，卷积后的图像面积小于原图像。为了让卷积核覆盖到所有的像素，可以对边缘位置进行 0 像素填充，然后在进行卷积。这种越过边缘的取样是 same padding。如过移动步长为 1，那么得到和原图一样大小的图像。如果步长很大，超过了卷积核长度，那么 same padding，得到的特征图也会小于原来的图像。

```
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])
```

5）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()
```

6）验证并保存模型

测试集验证

```
#测试
arg2_r =
sess.run(arg2,feed_dict={images_input:train_data.test_images,labels_i
nput:train_data.test_labels})
arg1_r =
sess.run(arg1,feed_dict={images_input:train_data.test_images,labels_i
nput: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')
```

使用已训练好的模型参考：gender_model_use.py

结果：迭代 3000 次，模型的准确率达到 93%

三，实验结果分析

```
Epoch : 1 ,  Cost : 1.7629557
Epoch : 2 ,  Cost : 0.8955871
Epoch : 3 ,  Cost : 0.6002768
Epoch : 4 ,  Cost : 0.4222347
*
*
*
*（因为篇幅有限，此处省略训练次数）
*
*
Epoch : 97 ,  Cost : 0.0289681
```

Epoch : 98 , Cost : 0.0271511

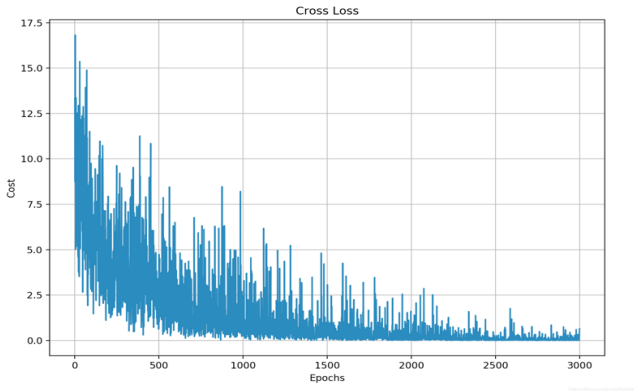
Epoch : 99 , Cost : 0.0131940

Epoch : 100 , Cost : 0.0418167

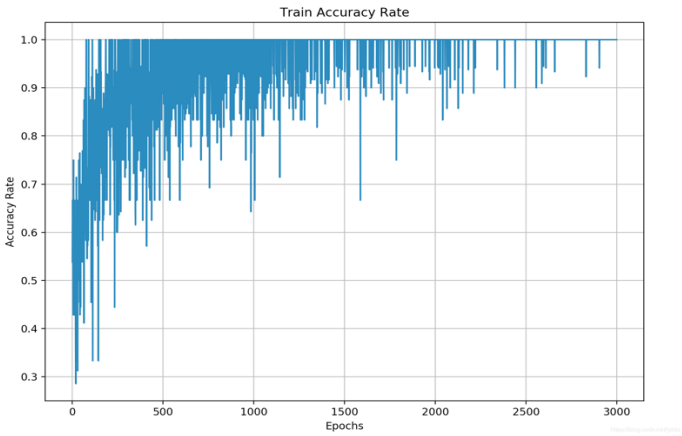
training finished

	precision	recall	f1-score	support
0	0.88	0.97	0.92	36
1	0.97	0.89	0.93	44
avg / total	0.93	0.93	0.93	80

3.1 训练交叉熵代价

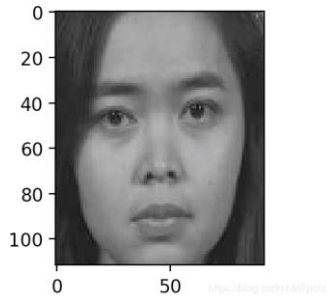


3.2 训练的准确率

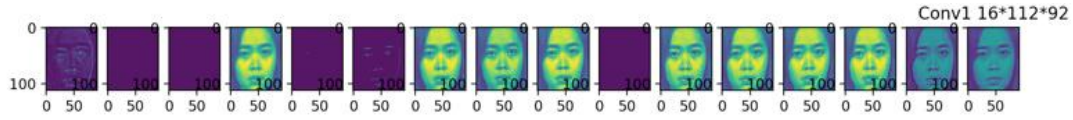


3.3 特征可视化

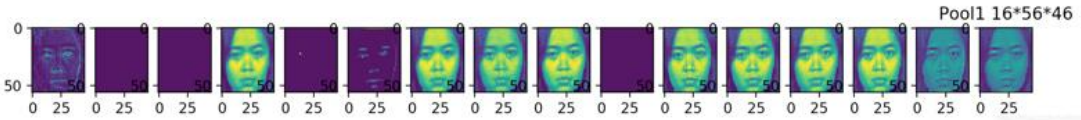
训练数据中的一个样本



第一层卷积提取的特征

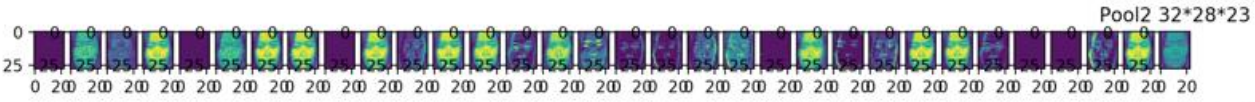


2x2 池化后特征



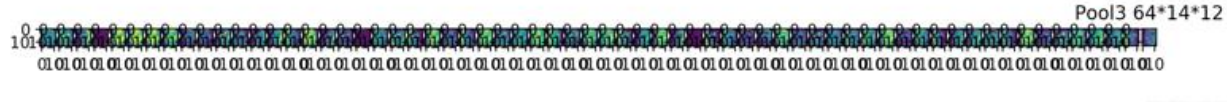
第二层卷积提取的特征

2x2 池化后特征



第三层卷积提取的特征

2x2 池化后特征



在这里我用了 3 个卷积层，三个池化层，一个全连接层对图像进行处理
1 输入层为输入的灰度图像尺寸: -1 x 112 x 92 x 3
2 第一个卷积层,卷积核的大小,深度和数量 (3, 3, 3, 16)
3 池化后的特征张量尺寸: -1 x 56 x 46 x 16
4 第二个卷积层,卷积核的大小,深度和数量 (3, 3, 16, 32)
5 池化后的特征张量尺寸: -1 x 28 x 23 x 32
6 第三个卷积层,卷积核的大小,深度和数量 (3, 3, 32, 64)
7 池化后的特征张量尺寸: -1 x 14 x 12 x 64
8 全连接第一层权重矩阵: 10752 x 512
9 全连接第二层权重矩阵: 512 x 128

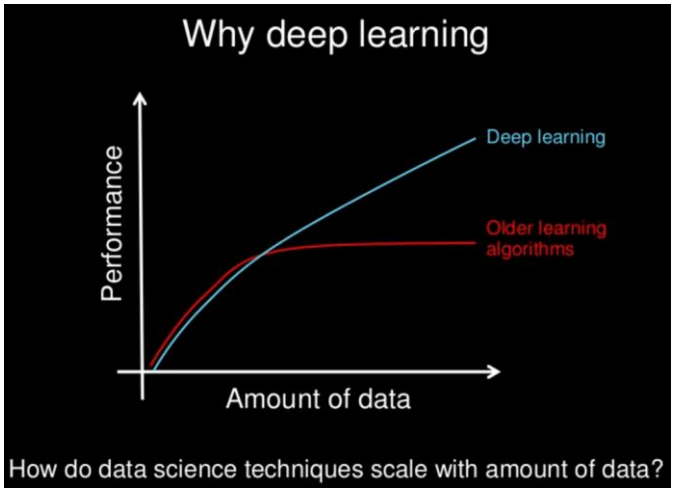
四，课程设计总结与心得

4.1 关于课程设计：

此次模型的训练非常耗时，从 epoch1 到 epoch100，最终损失函数下降到了 0.0418，
学习框架经过很多轮轮学习损失函数已经达到很小的程度,但是考虑到电脑的性能因素，如果性能更强的话可以多进行几轮。可以提高准确率
为了进一步提高准确率，可以网络参数初始化操作。，先给它一个初始参数以便在一遍遍更新权重参数时更准确。

性能提升：

深度学习和其它现代的非线性机器学习模型在大数据集上的效果更好,尤其是深度学习



深度学习算法往往在数据量大的时候效果好，所以本文用到了 11 多万张图片，数据量大一般会使你的模型训练的越准确

激活函数选择很重要，要根据问题来选择。在 ReLU 之前流行 sigmoid 和 tanh，然后是输出层的 softmax、线性和 sigmoid 函数。除此之外，我不建议尝试其它的选择。

这三种函数都试一试，记得把输入数据归一化到它们的值域范围。

显然，你需要根据输出内容的形式选择转移函数。

比方说，将二值分类的 sigmoid 函数改为回归问题的线性函数，然后对输出值进行再处理。同时，可能需要调整合适的损失函数。

参考文献

邱锡鹏—复旦大学《神经网络与深度学习》
吴恩达—深度学习课程
TensorFlow机器学习库。
极客教程的深度学习专栏的《卷积神经网络(CNN)简介》
https://geek-docs.com/deep-learning/cnn/cnn-introduce.html
《深度学习轻松学：核心算法与视觉实践》
《ResNet解析》