**上海电力大学**

**本科毕业设计（论文）**



题 目： 基于深度学习的表情识别

院 系： 电子与信息工程学院

专业年级： 电子科学与技术2015级

学生姓名： 林正浩 学号： 20151306

指导教师： 贾振堂

2019年5月23日

【摘 要】

人类对人脸表情识别从上世纪70年代开始就有所研究，当时受限于电子计算机与互联网的发展，主要以建立面部肌肉模型的传统方法，对表情的识别判定点进行研究。随着数字图像处理、机器学习和模式识别技术等科技的发展，人工智能让人脸表情识别技术成为了一门崭新的学科，形成了新的研究方向，以自动人脸表情识别为目标。

本课题从结构最简单的CNN开始，描述了从最简单的CNN优化到最终模型结构的全过程，以及选择各项改善方法的理由。主要的优化点有：卷积核的选取，用Global Average Pooling层代替所有全连接层，池化层的选取，用Batch Normalization解决梯度消失问题，用特化模型的思路进一步提高正确率。

在本课题中，我将原数据集用Haar分类器一分二，分为了正面脸数据集和非正面脸数据集，分别在原本用全集训练好的模型上进行微调，得到两个特化模型。在实际使用时，仍然先用Haar分类器，判断实际样本属于正面脸还是非正面脸，若是正面脸则调用正面脸模型，若是非正面脸则调用非正面脸模型。经过实验，对于一个已经训练到饱和无法继续提升正确率的模型，这个做法能在原基础上对正确率有进一步的提升。

最终应用在演示程序中的模型在正面脸验证集上的正确率是82.56%。

关键词：深度学习；卷积神经网络；人脸表情识别；全局平均池化；FER2013

**【ABSTRACT】**

Human facial expression recognition has been studied since the 1970s. Due to the limitation by the development of electronic computers and the Internet, it mainly used the traditional method of establishing facial muscle models to study the recognition points of expressions. With the development of digital image processing, machine learning and pattern recognition technologies, Artificial Intelligence empowers facial expression recognition technology to be a brand new discipline, forming a new research direction, with automatic facial expression recognition as the goal.

Starting with the simplest CNN, this topic describes the whole process from the simplest CNN optimization to the final model structure, and the reasons for choosing various improvement methods. The main optimization points are: the selection of the convolution kernel, using Global Average Pooling Layer to replace all the fully connected layers, the selection of pooling layer, the application of Batch Normalization to solve the vanishing gradient problem, and the special model adopted to further improve the accuracy.

In this topic, I divide the original data set into two parts, using the Haar classifier, into a frontal face dataset and a non-frontal facet dataset. They are fine-tuned on the original model that was trained with the universal sets, and two specialization models are obtained. In actual use, the Haar classifier is still used first to determine whether the actual sample belongs to the front or non-frontal face. If it is a front face, the front face model is called, and if it is a non-positive face, the non-frontal face model is called. After experimentation, this method can further improve the correct rate on the original basis for a model that has been trained to saturate without further improvement in accuracy.

The accuracy of the model applied in the demo program on the front face validation set is 82.56%.

**Key words:** deep learning, CNN, facial expression recognition, GAP, FER2013

# 目录

[第一章 绪论 1](#_Toc9514748)

[1.1 课题研究背景 1](#_Toc9514749)

[1.2 课题主要研究内容 1](#_Toc9514750)

[1.3 本文的组织结构和内容安排 2](#_Toc9514751)

[第二章 神经网络相关技术与工具的介绍 3](#_Toc9514752)

[2.1 神经网络框架的选取 3](#_Toc9514753)

[2.2 工具及模块的选取 3](#_Toc9514754)

[2.3 神经网络相关技术的介绍 3](#_Toc9514755)

[第三章 数据集与模型的训练配置 5](#_Toc9514756)

[3.1 数据集的选取 5](#_Toc9514757)

[3.1.1 主流数据集的对比 5](#_Toc9514758)

[3.1.2 FER2013及FER+数据集介绍 7](#_Toc9514759)

[3.2 模型训练配置 7](#_Toc9514760)

[3.2.1 Compile函数与Fit函数的配置 7](#_Toc9514761)

[3.2.2 回调函数的配置 8](#_Toc9514762)

[3.3 模型的评估方法 9](#_Toc9514763)

[第四章 演示程序的设计与编写 11](#_Toc9514764)

[4.1 演示程序的构成 11](#_Toc9514765)

[4.1.1 绘制文字AddTextToImage函数 12](#_Toc9514767)

[4.1.2 绘制人脸定位方框facialExpressionRecognition函数 12](#_Toc9514768)

[4.1.3 摄像头循环部分的代码 12](#_Toc9514768)

[4.2 演示程序中的优化 12](#_Toc9514766)

[4.2.1 直方图均衡化 12](#_Toc9514767)

[4.2.2 使用基于FER+数据集的特化模型 12](#_Toc9514768)

[第五章 模型结构的研究和优化 14](#_Toc9514769)

[5.1 最简CNN 14](#_Toc9514770)

[5.2 加深CNN 15](#_Toc9514771)

[5.3 再加深CNN出现的神经元失活及其解决方案 17](#_Toc9514772)

[5.4 参照VGG-16继续优化 21](#_Toc9514773)

[5.5 进一步改善 26](#_Toc9514774)

[5.5.1 卷积核优化 26](#_Toc9514775)

[5.5.2 平均池化与最大池化 26](#_Toc9514776)

[5.5.3 用GAP层代替FC层 26](#_Toc9514777)

[5.6 特化模型 27](#_Toc9514778)

[5.6.1 特化模型的意义 27](#_Toc9514779)

[5.6.2 特化模型与既存思路的异同 27](#_Toc9514780)

[5.7 未采用的尝试方向 28](#_Toc9514781)

[5.7.1 用Facenet特征向量辅助识别 28](#_Toc9514782)

[5.7.2 性别特化模型 30](#_Toc9514783)

[第六章 最终模型及其优化效果的验证 31](#_Toc9514784)

[6.1 最终CNN模型的结构与性能 31](#_Toc9514785)

[6.2 特化模型的正确率评估 33](#_Toc9514786)

[6.3 最终模型在FER+上的表现 35](#_Toc9514787)

[6.4 各模型的正确率表现汇总 37](#_Toc9514788)

[第七章 总结与展望 38](#_Toc9514789)

[谢辞 39](#_Toc9514790)

[参考文献 40](#_Toc9514791)

# 第一章 绪论

## 1.1 课题研究背景

人的面部表情含有丰富的情感信息，从人的面部表情可以一定程度上分析获知人类对应的心理状态并加以利用。人的面部就像一种可以传输信息的媒介，包含了很大一部分的情绪信息。加州大学（UCLA）一位教授：艾伯特．梅拉比安（Albert Mehrabian）所做的研究曾提出，讯息中55%的意义来自于视觉的肢体语言、面部表情，38%的意义来自语气、声调等语音上的情感信息，只有7%的意义来自说话的实际文本内容。

此外，在神经网络研究的范围内，人脸表情识别这个课题与其他围绕人脸的课题之间都有很强的联系，好的模型、优化方法、改进点都可以通过迁移学习在各个人脸相关课题之间通用。人脸是人类最常用、最复杂、最重要的生物特征，即使都是人脸，也会因为人种、年龄、性别，乃至光照、装扮、角度而产生显著差异，人类因为长期的进化，对人脸演化出了极高的识别能力，在各种苛刻的条件下都能获取人脸的特征、表情、情绪乃至微表情等细致信息。对人脸包含的各种信息的提取是神经网络在计算机视觉问题应用中极为重要的一块课题。

人脸表情识别在现实场景中也有丰富的应用，大规模的应用比如人机交互、机器人制造、客户心理分析等场景。小规模的具体应用比如可以用于商店对来店客人的表情进行识别，分析其情绪，找出客人对店铺感到不满意的要素；还可以根据店铺内客人的情绪分布，推送合适的音乐，调整客户的不良情绪让他们能消费愉快；还可以判断住院患者的情绪，让医护人员可以更早地发现病人的不良心理状态，及时介入，确保更好的治疗效果。任何需要获取人员情绪的场景，都可以通过表情识别的方式来间接获取其情绪，这也是最便捷的方法。尤其是长时间、大量人脸、大范围的分布，通过人工识别和分析是不可能实现的，而用神经网络实现的自动识别就能很好地胜任这些人工所不能覆盖的应用场景。

2017年苹果出品的新款iPhone中也加入了用摄像头识别人脸表情，自动匹配出动态Emoji的功能，非常有趣且吸引人。从Emoji的广泛应用来看，也许用表情识别操控动态Emoji会成为智能设备的一项基本功能。

## 1.2 课题主要研究内容

人类对人脸表情识别从上世纪70年代开始就有所研究，当时受限于电子计算机与互联网的发展，主要以建立面部肌肉模型的传统方法，对表情的识别判定点进行研究。随着数字图像处理、机器学习和模式识别技术等科技的发展，人工智能让人脸表情识别技术成为了一门崭新的学科，形成了新的研究方向，以自动人脸表情识别为目标。

识别技术大体上可以分为基于面部模型的传统机器学习方法，和运用了神经网络深度学车技术的深度学习模型。传统的面部模型方法因为开始时间早，有着大量研究成果和理论，但他主要还是依赖人工经验，泛用性和准确度都不高。深度学习模型可以对人脸特征数据进行更有效的提取，在泛化能力和准确度上都有了划时代的提升，广泛用于各种分类课题，是当前最先进最重要的研究方法之一。

而同为神经网络范畴的研究也有许多不同的研究成果，近年来基于迁移学习的思路，发展了一系列诸如VGG-face，Facenet等泛用人脸识别模型。迁移模型是一种可以将某个训练好的模型迁移到另一个相关相似的课题上的技术，在相似的问题之间可以用相似的网络结构和权重分布，通过恰当地使用迁移学习可以解决样本不足、训练慢、收敛慢的问题。

对于本文中使用的FER2013数据集，目前我所知道的最高成绩是论文[5]中的75.42%的正确率，其中用到了BOVW, 迁移VGG-face，Local SVM等多种技术。

## 1.3 本文的组织结构和内容安排

本课题在优化模型在指定数据集上的正确率时，更多地考虑研究创新的模型优化方法。表情识别的优化方法往往也能类推到近似的模型。

第二章中对本课题用到的一些技术和工具进行了介绍。

第三章阐述了模型的超参数设置、训练与评估的内容，以及数据集的选取过程。简单介绍了用到的技术，以及模型的训练方法和评估方法。各模型在超参数上保持统一，第五章中提到的各种模型结构都会在同样的训练方法和评估方法下进行，控制变量，使正确率之间更有可比性。

第四章描述了演示程序的构成，介绍了在演示时针对实际演示环境的一些优化。

第五章描述了模型的优化方法，从结构最简单的CNN开始，描述了从最简单的CNN优化到最终模型结构的全过程，以及选择各项改善方法的理由。主要的优化点有：卷积核的选取，用Global Average Pooling层代替所有全连接层，池化层的选取，用Batch Normalization层解决梯度消失问题，用特化模型的思路进一步提高正确率。

第六章中，根据前一章提出的优化点，完成了最终的CNN特化模型，在FER2013和FER+数据集上正确率分别是65.28%和 80.02%，同时整理了文中出现的其他模型的正确率，以供对比。

第七章对全文得出的有价值结论进行了总结，提出未来工作的展望和方向。

本课题的主要重点放在第五章对模型优化方法的研究。

# 第二章 神经网络相关技术与工具的介绍

## 2.1 神经网络框架的选取

本文最终选用的开发框架是以Python语言编写，以Tensorflow为后端(Backend)的Keras框架。

Keras是一种可以用各种框架作为后端的深度学习库。其中提供了更加用户友好便于使用的高层API。相比直接使用Tensorflow框架的API，Keras在开发效率上有巨大优势，便于编写试验程序。

Keras现在已经整合到Tensorflow模块的内部，为了更好的兼容性，且不同版本的Keras混用会发生不可知的障害，本次都使用的是Tensorflow模块中内置的Keras模块。

因为Keras没有C++接口，本次程序统一用Python语言编写。

## 2.2 工具及模块的选取

程序用Anaconda对Python的扩展包进行环境管理，演示程序在VS2017中编写，用GIT进行版本管理。

简单介绍一下本程序中用到的主要模块：

sklearn.SVM：用支持向量机进行简单的想法验证。

plt：用于绘制图表。

numpy：numpy支持大量的维度数组与矩阵运算，Opencv等模块内部也是基于numpy的数据格式。在演示程序中用于处理矩阵形式的图像的格式。

PIL：拥有对图像进行基础操作的功能，几乎已经成为了已经是Python平台事实上的图像处理标准库，在演示程序中用于在目标图片上根据需要绘图。

cv2(Opencv)：非常便利的视觉模块，用于调用摄像头、图像预处理等。

pandas-相当于Python中的excel，用于建立多维数据表格，便于读写。

## 2.3 神经网络相关技术的介绍

人工神经网络：人工神经网络（Artificial Neural Networks，简写为ANNs）也简称为神经网络（NNs）或称作连接模型（Connection Model），它是一种模仿动物神经网络行为特征，进行分布式并行信息处理的算法数学模型。这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的 [12] 。

卷积神经网络：卷积神经网络（Convolutional Neural Networks, CNN）是一类包含卷积计算且具有深度结构的前馈神经网络（Feedforward Neural Networks），是深度学习（Deep Learning）的代表算法之一 [8-9]。卷积神经网络具有表征学习（Representation Learning）能力，能够按其阶层结构对输入信息进行平移不变分类（Shift-Invariant Classification），因此也被称为“平移不变人工神经网络（Shift-Invariant Artificial Neural Networks, SIANN）” [10]。卷积层内每个神经元都与前一层中位置接近的区域的多个神经元相连，区域的大小取决于卷积核的大小，在文献中被称为“感受野（Receptive Field）”。本课题主要以卷积神经网络为基础进行研究。

梯度消失：在神经网络中，当前面隐藏层的学习速率低于后面隐藏层的学习速率，即随着隐藏层数目的增加，分类准确率反而下降了。这种现象叫做消失的梯度问题。前面的隐藏层的学习速度要低于后面的隐藏层。这种现象普遍存在于神经网络之中， 叫做消失的梯度问题（Vanishing Gradient Problem）。更加一般地说，在深度神经网络中的梯度是不稳定的，在前面的层中或会消失，或会激增。这种不稳定性才是[深度神经网络](https://baike.baidu.com/item/%E6%B7%B1%E5%BA%A6%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C/6424200" \t "_blank)中基于梯度学习的根本问题 [11]  。

深度学习：深度学习实质上是通过构建具有很多隐层的机器学习模型和海量的训练数据，来学习更有用的特征，从而最终提升分类或预测的准确性。因此，“深度模型”是手段，“特征学习”是目的。区别于传统的浅层学习，深度学习的不同在于：1）强调了模型结构的深度，通常有5层、6层，甚至10多层的隐层节点；2）明确突出了特征学习的重要性。

# 第三章 数据集与模型的训练配置

## 3.1 数据集的选取

### 3.1.1 主流数据集的对比

表3-1 常用人脸表情数据集一览

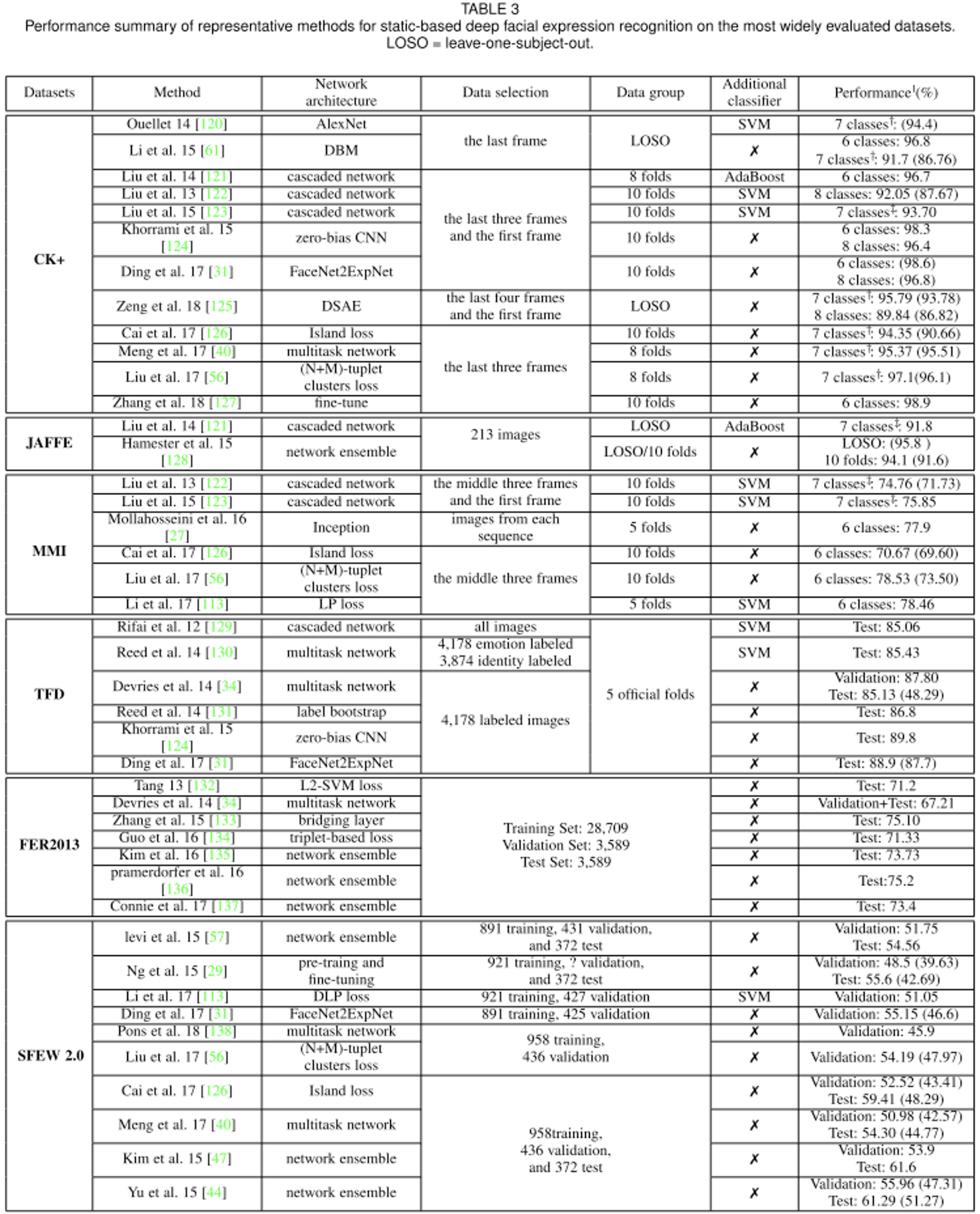


表3-1引自文[1], 其中列出的是常用的现存表情数据集，以及该数据集上在2018年4月23日为止获得的最高成绩。本文出于多方面考虑最终选用了FER2013数据集。

为了便于与现有的研究成果中的正确率进行对比，本次只从便于获得历史成绩的数据集中选取。从多篇论文来看，“FER2013”常常被用作评估模型性能的基准点。这个数据集是为了被用于ICML2013竞赛而建立的数据集，其中的表情分为以下几类: 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral。

因为模型的实际应用场景是东亚人，因此本来想优先选用纯日本人的数据集JAFFE，但很遗憾其中的数据只有213条，且只有几位女性模特做出不同表情，数据量非常不足，因此最终没有选用JAFFE。

表3-1中可以看到，各个数据集的最高成绩之间正确率差异巨大。有两方面原因，一是数据集的标注普遍存在错误，制约了最终正确率的最大值；二是比如CK+、JAFFE、TFD这些数据集，可以看到在这些数据集上的正确率很高，这是因为参加制作的人员数量太少，训练集部分和测试集部分是同一批人的人脸，因此拟合程度偏高，不能完全反映模型的实际效果。

上述数据集来说，大多存在数据量不够大，标签标注不完全的问题。而FER2013数据集中带标注的图片量达到28709，是其中最高的。

在1971 年心理学家Ekman和Friesen 就提出了人类的六种基本情感（不包括中性表情），分别是快乐(happiness)、厌恶(disgust)、惊讶(surprise)、愤怒(anger)、悲伤(sadness)和恐惧(fear)。FER2013的表情标签也是分成了这最常见的七类。

出于上述理由，最终本文的数据集我采用了FER2013数据集，以及基于FER2013改进的FER+数据集。

以下是未采用的其他数据集的简单介绍：

• CK+:包括123个subjects, 593 个 image sequence。该数据库由118名受试者录制，在这593个image sequence中，有327个sequence 有 emotion的 label。除了中性外包含7种表情：愤怒、蔑视、厌恶、恐惧、高兴、悲伤和惊讶。

• MMI：包括32个subjects，326个image sequence。213个sequence 有 emotion的 label。包含6中表情（相比较于CK+没有蔑视），MMI更具挑战性，因为很多人都戴有配饰。

• JAFFE：包含213副（每幅图像分辨率为256\*256）日本女性的脸部图像，包含7种表情。该数据库均为正面脸，且对原始图像进行了调整和修剪，光照均为正面光源，但光照强度有差异。

• TFD：该数据库是几个面部表情数据集的集合，TFD包含112234张图片(每张图片被调整到48\*48大小)，所有实验对象的眼睛都是相同的距离。其中4189张有标注，包含7种表情。

• FER2013：该数据库通过谷歌的图片API自动收集，数据库中所有图片都修正了标签，将图片调整到48\*48大小。包含28709张训练图像，3589张测试图像，包含7种表情。

• SFEW：该数据集是从AFEW数据集中抽取的有表情的静态帧，包含7类表情。训练集、验证集和测试集分别包含958、436和372sample。

### 3.1.2 FER2013及FER+数据集介绍

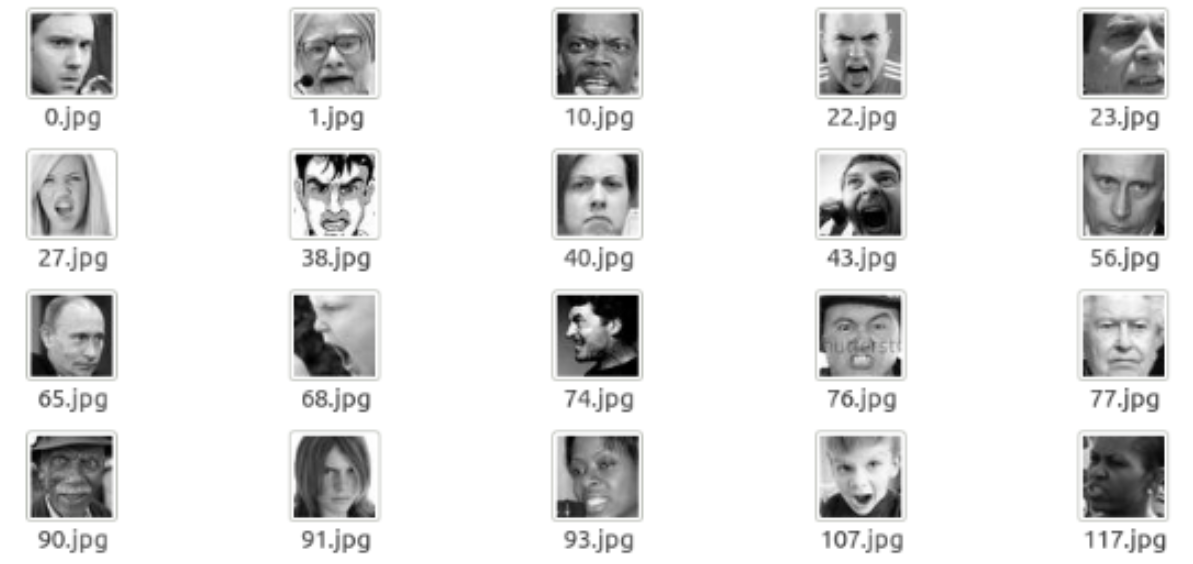


图3-1 FER2013数据集样本图片示例

FER2013人脸表情数据集是Kaggle出品的一款免费数据集，由35886张人脸表情图片组成，本身就分为训练集和验证集。因为大家用于评判模型的验证集都来自数据集官方规定的验证集，因此在验证集上的验证结果和他人的成绩更有可比性。

每张图片都是48x48尺寸的单通道矩阵图像。

标签分为0 anger 生气； 1 disgust 厌恶； 2 fear 恐惧； 3 happy 开心； 4 sad 伤心；5 surprised 惊讶； 6 normal 中性。



图3-2 FER+对于FER2013中错误标签的修正

FER+是Kaggle在FER2013发布后补充的修正标签集，其中图片还是和FER2013保持一致，但是标签变成了每种表情下的概率值。如图3-2所示，其中修正了FER2013中包含的大量标注错误，并且新增了contempt表情和unknown，但为了便于和FER2013对比，本课题中仅仅采用FER+中和FER2013相同的7个标签。

## 3.2 模型训练配置

### 3.2.1 Compile函数与Fit函数的配置

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

如上方代码中函数参数列表所示，优化器选择了Adam优化器，损失函数选择了交叉熵categorical\_crossentropy，评估标准设为accuracy。优化器以及损失函数都是神经网络训练中最常见最简单的配置。

Adam 在深度学习领域内是十分流行的算法，因为它能很快地实现优良的结果。经验性结果证明 Adam 算法在实践中性能优异，相对于其他种类的随机优化算法具有很大的优势。Adam最初由文[13]提出，在介绍这个算法时，原论文列举了将 Adam 优化算法应用在非凸优化问题中所获得的优势：直截了当地实现；高效的计算；所需内存少；梯度对角缩放的不变性（第二部分将给予证明）；适合解决含大规模数据和参数的优化问题；适用于非稳态（non-stationary）目标；适用于解决包含很高噪声或稀疏梯度的问题；超参数可以很直观地解释，并且基本上只需极少量的调参。

categorical\_crossentropy（交叉熵损失函数)可以用来评估当前训练得到的概率分布与真实分布的差异情况。如下式1-1:(其中y为期望的输出，a为神经元实际输出）

（1-1） （1-2）

（1-3）

式1-2和式1-3中的是交叉熵的导数，从上式可以看到，导数中没有σ′(z)这一项，权重的更新是受σ(z)−y这一项影响，即受误差的影响，所以当误差大的时候，权重更新就快，当误差小的时候，权重的更新就慢。这是一个很好的性质。

本课题研究主题主要在于各个网络结构之间的对比，因此尽量保持超参数一致。

history = model.fit(train\_x, Y\_train,

batch\_size=256, nb\_Epoch=1000,

verbose=1,

validation\_data=(test\_x, Y\_test),

callbacks=[checkpointer, learning\_rate\_reduction, early\_stopping])

如上方代码所示，batch设置为256，总Epoch设为1000。但实际上主要靠EarlyStopping停止训练。

### 3.2.2 回调函数的配置

本次用到了三个回调函数：ModelCheckpoint，ReduceLROnPlateau，EarlyStopping 。

具体配置代码如下：

checkpointer = ModelCheckpoint(filepath="../checkpoints/%s\_checkpoint-{Epoch:02d}e-val\_acc\_{val\_acc:.4f}.h5" %model\_name, monitor='val\_acc', save\_best\_only=True, verbose=1, period=1)

如上方代码所示，save\_best\_only被设为了True，检查点函数会在每个Epoch后自动检查是否是本轮训练最高成绩，如果是最高成绩，则自己根据Epoch序号和正确率命名并存档。

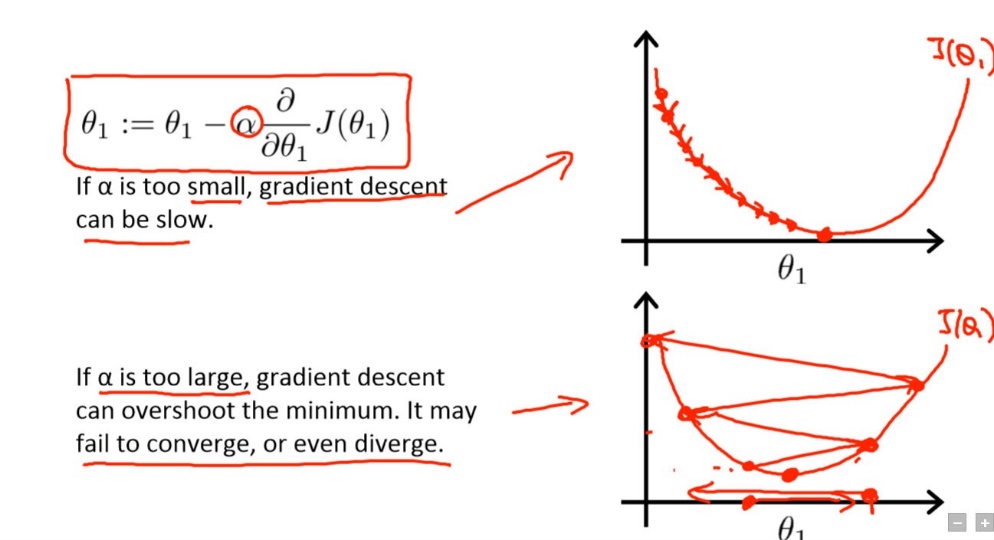


图3-3 学习率调整示意图

learning\_rate\_reduction = ReduceLROnPlateau(monitor='val\_acc', patience=3, verbose=1, factor=0.5, min\_lr=0.00001)

如上方代码所示，学习率调整函数的等待间隔patience=3，当间隔3个Epoch正确率还没有发生提升时，就将学习率乘以调整因子factor=0.5。因为没有设置初始学习率，因此初始学习率是Keras默认的0.01。学习率的下限设置为了min\_lr=0.00001。

在训练过程中降低学习率可以更好地逼近损失函数的极值，虽然较大的学习率可以让损失函数迅速变化，快速训练，但若一直保持较大的学习率，损失只会在底部附近振荡，而无法逼近底部。如图3-3所示，更小的学习率才能更接近损失函数的底部。

early\_stopping = EarlyStopping(monitor='val\_acc', patience=30, verbose=2)

如上方代码所示，早停函数的patience主要设置为30，在对最终模型进行训练时，为了避免错过最大值，追求最佳结果，保险起见将patience设为了100，但实际上从最终生成的checkpoint的间隔来看，没有两个checkpoint之间超过间隔30个Epoch，因此认为patience的改变没有影响到结果。

## 3.3 模型的评估方法

最高成绩已在训练过程中记入了checkpoint的文件名，以最高成绩的checkpoint的正确率为准。

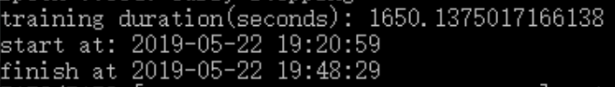


图3-4 训练时间输出示例

如图3-4所示，在每次训练后，训练时间和每一轮Epoch的时间都通过time.time()函数打印到了控制台，可以对比每个模型训练到饱和所需的时间。

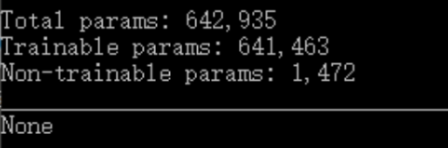


图3-5 模型参数量输出示例

在每次训练后，模型的大致结构都会用model.summary()函数打印到控制台，如图3-5所示，summary()函数同时会记录每个模型的总参数量，参数量会影响到这个模型的训练所需时间，参数量越大则需要更多的时间训练。可以借此对比各个模型之间在参数量上的优劣。

显示训练过程的函数用到了plt模块，是一种可以简单输出图表的模块，用plt模块在同一张图片上输出了模型在训练集上的损失、训练集上的正确率、验证集上的损失、验证集上的正确率。通过最终图表的形态，分析训练过程中过拟合、梯度消失等现象。下文中会附上涉及到的每个模型的训练过程图像。

显示训练过程的函数代码片段如下：

#显示训练过程、正确率与损失

def show\_train\_history(train\_history):

plt.plot(train\_history.history['acc'], 'r')#画出曲线

plt.plot(train\_history.history['val\_acc'], 'g')#画出曲线

plt.plot(train\_history.history['loss'], 'b')#画出曲线

plt.plot(train\_history.history['val\_loss'], 'k')#画出曲线

plt.title(model\_name)#设置本plot的标题

plt.ylabel('acc/loss')#设置y轴标记

plt.xlabel('Epoch')#设置x轴标记

plt.legend(['train\_acc', 'val\_acc', 'train\_loss', 'val\_loss'], loc='upper left')#设置曲线注释在左上角

plt.show()

# 第四章 演示程序的设计与编写

## 4.1 演示程序的构成

演示程序主要由初始化部分和摄像头循环部分这两大块组成。

再初始化时需要配置的项目有：工作路径、Haar识别器模型路径、表情识别模型路径、字体文件路径。

配置完毕之后进入摄像头主循环，在摄像头主循环中处理的内容主要有：

opencv模块返回摄像头的一帧图像🡪调用haar识别器返回图中存在的任意张人脸的坐标元组🡪根据坐标截取人脸转换成单通道灰度图🡪缩放图片尺寸到48×48🡪直方图均衡化🡪灰度值归一化🡪用numpy模块调整矩阵的格式🡪调用表情识别模型识别处理好的人脸🡪根据识别结果用PIL模块将识别到的标签绘制到图片指定坐标🡪输出图片到屏幕窗口🡪开始处理下一帧。

演示程序的程序结构流程图如下图4-1所示：

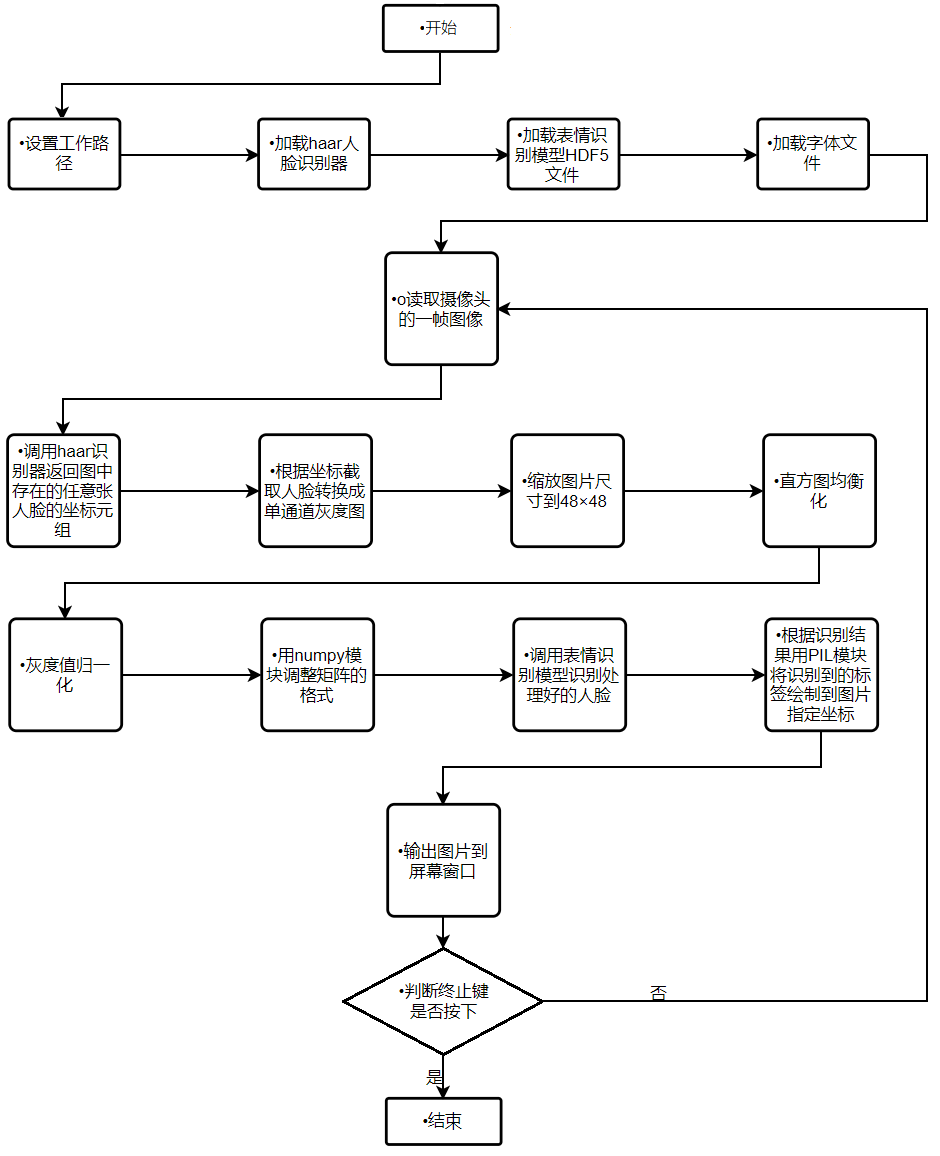


图4-1 演示程序-程序结构流程图

代码中最关键的部分有以下几部分：

### 4.1.1绘制文字AddTextToImage函数：

def AddTextToImage(image, text, left, top, textColor=(0, 0, 0), textSize=20)

这个函数会根据传入的图片和文字，在图片左上角绘制指定字体颜色的文字。

### 4.1.2绘制人脸定位方框facialExpressionRecognition函数：

def facialExpressionRecognition(image, facialExpression\_classifier, face\_classifier)

这个函数会调用表情识别模型facialExpression\_classifier和Haar人脸识别器face\_classifier，对图片中的人脸进行识别，将人脸的位置用方框绘制在图片上，将预测结果用AddTextToImage函数绘制在图片上。

### 4.1.3摄像头循环部分的代码

用Opencv模块不断取得摄像头的每一帧，取得后先用facialExpressionRecognition函数将我们需要的信息绘制在图片上，然后再输出到屏幕，这样循环着取得摄像头帧、处理、输出，就能实现摄像头动态识别的效果。

## 4.2 演示程序中的优化

### 4.2.1 直方图均衡化

直方图均衡化可以将偏暗或偏亮的图片调整成灰度适中的图片，实质上是对图像进行非线性拉伸，使灰度分布更均衡。

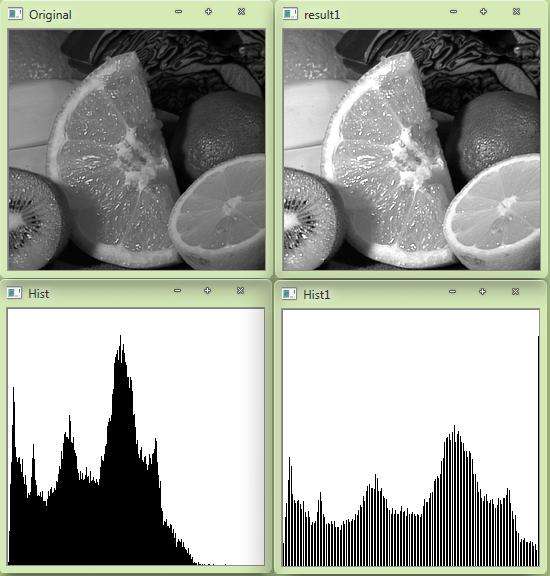


图4-2 直方图均衡化效果图

图4-2是直方图均衡化效果示意图，左边是未进行直方图均衡化的原图，灰度分布偏黑色，经过直方图均衡化的原图在灰度分布上更均匀，有利于识别。利用Opencv对灰度图进行直方图均衡化的代码如下：

gray\_face = cv2.equalizeHist(gray\_face) #直方图均衡化

因为FER2013本身各样本灰度分布就较平衡，且预处理后难以和他人的数据进行比较，因此没有在训练时加入预处理，而是在演示程序中对实际采样到的人脸进行预处理，使其分布更接近训练模型时使用的样本，从而达到更好的实际识别效果。

### 4.2.2 使用基于FER+数据集的特化模型

因为演示程序中是用Haar分类器识别人脸之后，再加以识别表情标签的，下文中的特化模型也是基于Haar分类器分为正面和非正面，因此只使用正面脸特化模型即可。特化模型可以实现比普通模型更好的综合识别率。

因为FER+是FER2013基础上的除错和扩展，因此最终效果数据上远远好于FER2013，在实际应用中的表现也当然远远好于FER2013，因此在演示程序中使用基于FER+的正面脸特化模型，来达到该应用场景中的最优识别效果。

最终应用在演示程序中的模型基于FER+，在正面脸验证集上该模型的正确率达到了82.56%。

# 第五章 模型结构的研究和优化

本章将从最简单的基础CNN开始，逐渐进行各种改善，对各项改善起到的作用进行研究，阐述逐步优化得出最终模型结构的全过程。

## 5.1 最简CNN

最简单的CNN结构如下表5-1所示，一个卷积层、一个平均池化层、一个平坦层、一个隐含全连接层，一个Softmax分类全连接层。中间插入了两个抗过拟合的Dropout层。

表5-1 最简CNN结构

|  |
| --- |
| **Input (48 x 48)** |
| 5x5 Convolutional Layer - 32 filters |
| 2x2 Average Pooling Layer |
| Flatten Layer |
| Dropout Layer – 50% |
| Full Connected Layer – 256 |
| Dropout Layer – 50% |
| Softmax - 7 |

本课题中除了最后分类时使用了Softmax激活函数，隐含层的激活函数全部采用了Relu函数，以防止Sigmoid等曲线非线性函数会出现的梯度消失问题。

这是最简单最基础的CNN结构，如预想的一样，对于人脸这样稍复杂的图像，这样一个最简单的CNN并不能起到很好的效果。图5-3为训练历史曲线，各颜色曲线含义如左上角所示，黑线为验证集上损失，蓝线为训练集上损失，绿线为验证集上正确率，红线为训练集上正确率。下文中的曲线含义都与此保持一致。模型训练结果见图5-1和图5-2，在验证集上的正确率只有49.41%，参数量为3,967,815。在第226个Epoch发生Early Stopping，总耗时362秒。

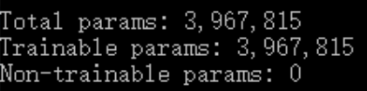


图5-1 最简CNN参数量

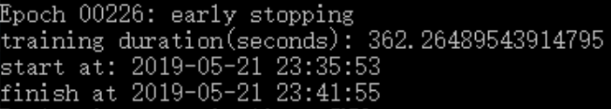


图5-2 最简CNN训练时间

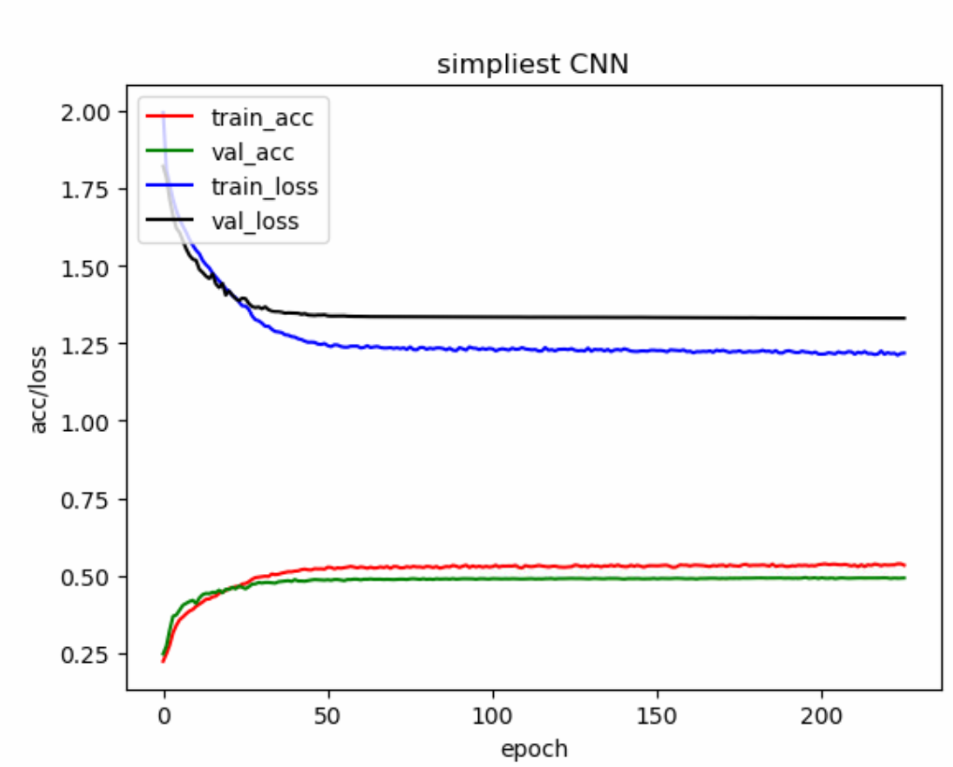


图5-3 最简CNN训练历史图表

## 5.2 加深CNN

本节从5.1中的最简CNN开始进行优化。最简单的优化思路就是堆叠卷积层和池化层，实现更深的网络，一般认为在出现梯度消失或梯度爆炸现象之前，更深的网络能有更好的表现。

表5-2 加深CNN结构

|  |
| --- |
| **Input (48 x 48)** |
| 5x5 Convolutional Layer - 32 filters |
| 2x2 Average Pooling Layer |
| 5x5 Convolutional Layer - 32 filters |
| 2x2 Average Pooling Layer |
| Dropout Layer – 50% |
| 5x5 Convolutional Layer - 32 filters |
| 2x2 Average Pooling Layer |
| 5x5 Convolutional Layer - 32 filters |
| 2x2 Average Pooling Layer |

续表5-2

|  |
| --- |
| Dropout Layer – 50% |
| Flatten Layer |
| Full Connected Layer – 256 |
| Dropout Layer – 50% |
| Softmax - 7 |

增加深度之后的CNN网络结构如上表5-2所示。模型训练结果见图5-4和图5-5，我将卷积层-池化层的组合堆叠了四次，在验证集上实现了56.52%的正确率，参数量为112,551。在第78个Epoch发生Early Stopping，总耗时156秒。正确率相比最简CNN有了7.11%的提升。同时，从图5-6上历史曲线可以看到，过拟合的程度相比最简CNN更小。

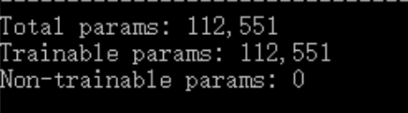


图5-4 加深CNN参数量

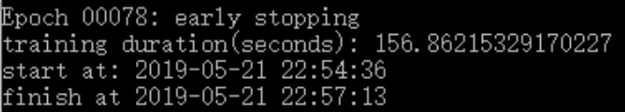


图5-5 加深CNN训练时间

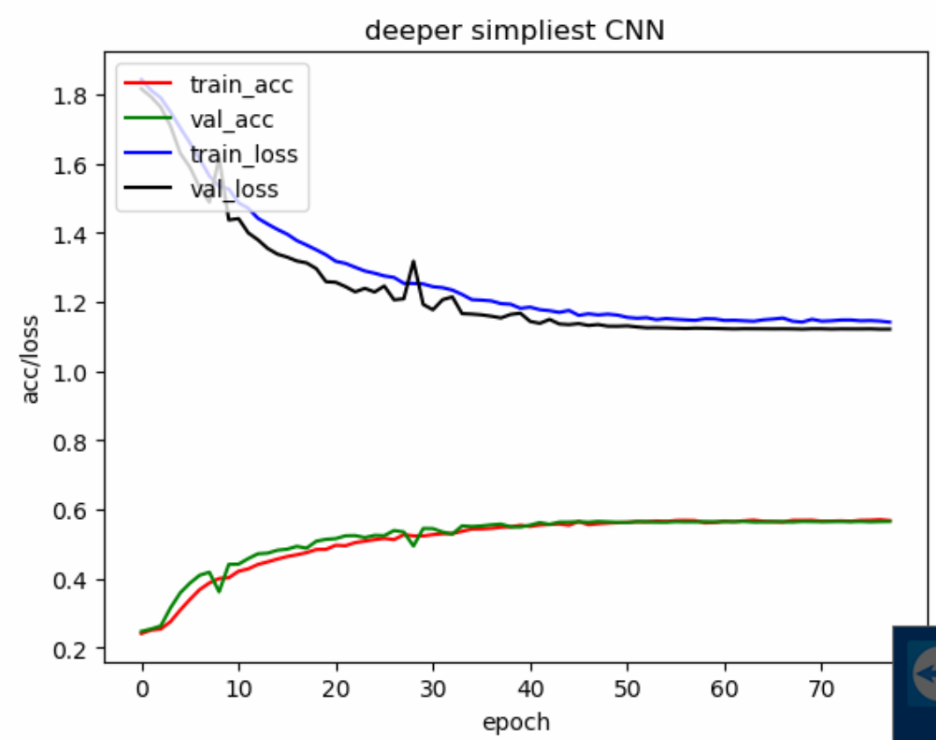


图5-6 加深CNN训练历史图表

## 5.3 再加深CNN出现的神经元失活及其解决方案

本节将从5.2节中的加深CNN开始进行优化。上一节的优化方法是增加卷积层和池化层组合的数量，首先我们延续上一节的思路，继续深化网络结构。

表5-3 再加深CNN结构

|  |
| --- |
| **Input (48 x 48)** |
| 5x5 Convolutional Layer - 32 filters |
| 5x5 Convolutional Layer – 32 filters |
| 2x2 Average Pooling Layer |
| Dropout Layer – 50% |
| 5x5 Convolutional Layer – 64 filters |
| 5x5 Convolutional Layer -64 filters |
| 2x2 Average Pooling Layer |
| Dropout Layer – 50% |
| 5x5 Convolutional Layer – 128 filters |
| 5x5 Convolutional Layer -128 filters |
| 2x2 Average Pooling Layer |
| Dropout Layer – 50% |
| 5x5 Convolutional Layer – 256 filters |
| 5x5 Convolutional Layer -256 filters |
| 2x2 Average Pooling Layer |
| Dropout Layer – 50% |
| Flatten Layer |
| Full Connected Layer – 256 |
| Dropout Layer – 50% |
| Softmax - 7 |

首先要提到一下池化层的作用和原理，池化层是在上一层图像的基础上，取一定大小的图像，根据指定的算法，将输入端的多个像素点合并为一个像素点，以缩小图像。池化层可以增大感受野，降维减少参数量，优化平移不变性。

因为原始图片较小，本课题中池化层采用的池化滤镜都是2x2的尺寸。即使如此，每次池化还是会使图片缩小一半，原图的长宽都是48像素，48>2的5次方，可以得出整个模型最多可以取5次2x2的池化。

基于这个结论，因为上一节的加深CNN中已经有了4个池化层，已经接近上限，因此不再增加池化层，只是继续增加卷积层。

而在继续增加卷积层的过程中，出现了问题，损失从第二个Epoch开始就完全没有下降。

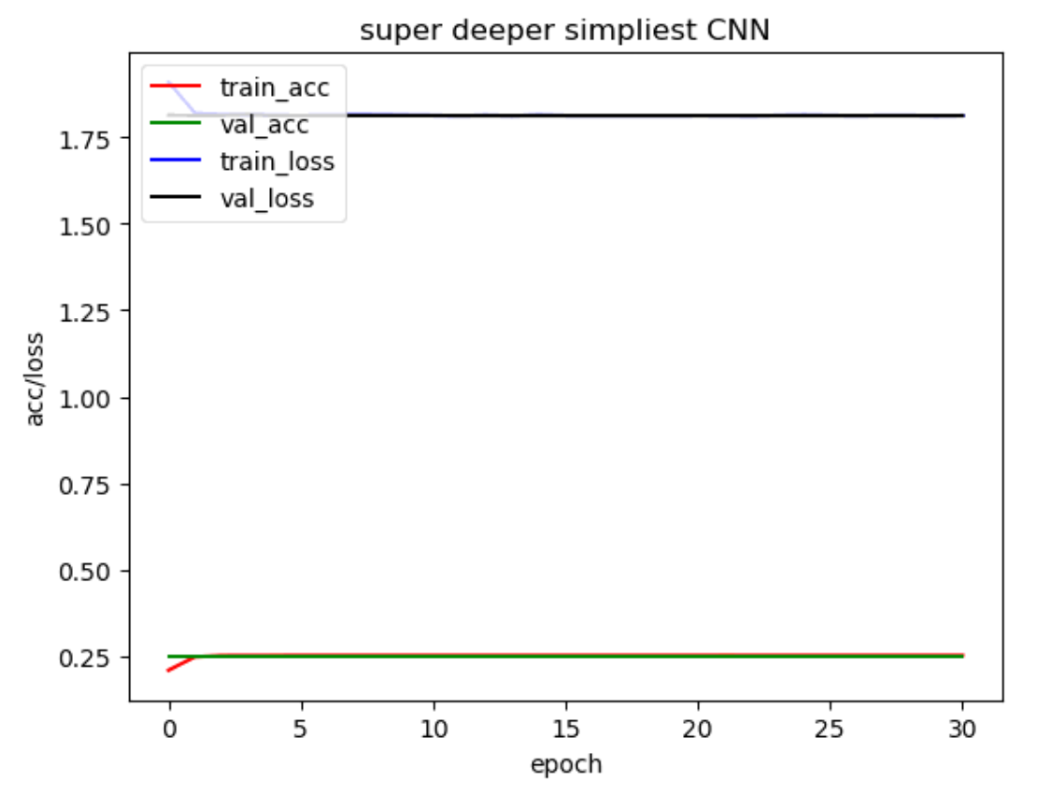


图5-7 再加深CNN发生无法训练的问题

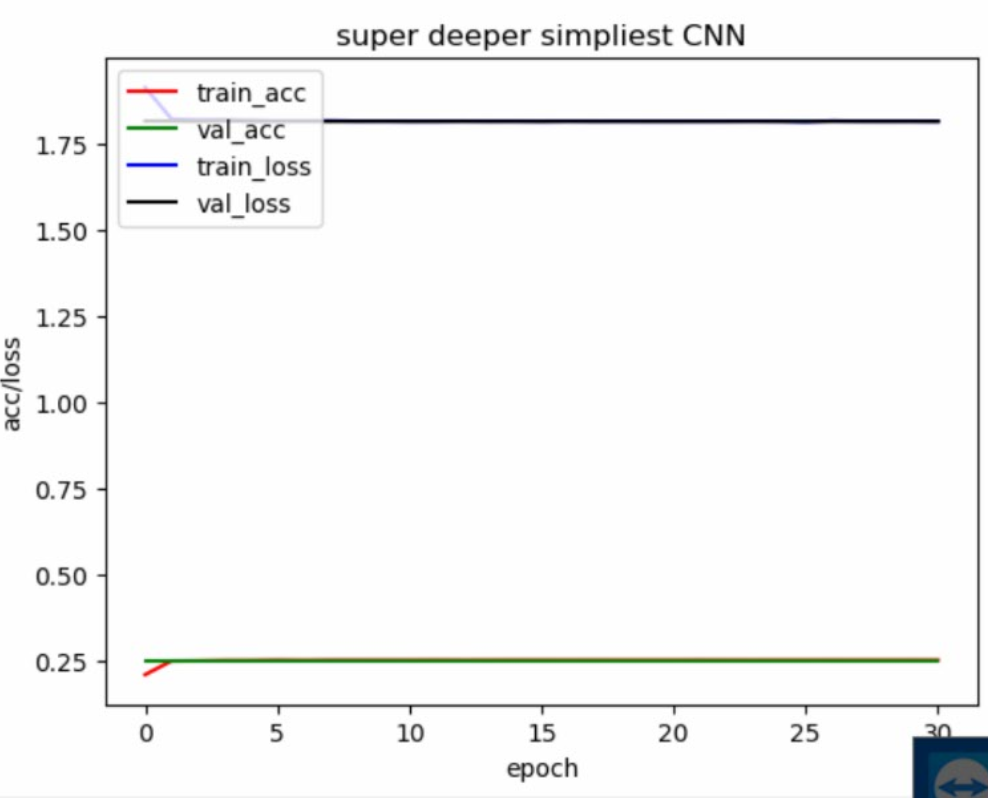


图5-8 删去一个池化层后，问题未解除

如图5-7所示，正确率从第二个Epoch就开始停滞，之后的训练完全没有效果。

我最初认为是因为池化次数太多，导致后面的图像过小，影响了分类。删除了一层池化层之后，训练效果如图5-8所示，仍然是第二个Epoch就停滞。

本课题中，为了防止梯度消失，一开始就全部选择了Relu函数作为激活函数，而且这次遇到的问题的现象是从第二个Epoch开始完全无法训练，而不是梯度极小收敛效果不佳，因此认为这不是梯度消失。从Relu函数的原理出发，Relu函数会将负半轴的输入全部转换成0输出，猜想当前网络是因为大量神经元在第一个Epoch的调参后，神经元输出落入负半轴，经过Relu函数后失活，发生了神经元坏死无法训练的现象。而失活的原因可能是初始学习率过大，导致许多神经元在第一个Epoch时输出变动太大神经元输出为负数导致失活。

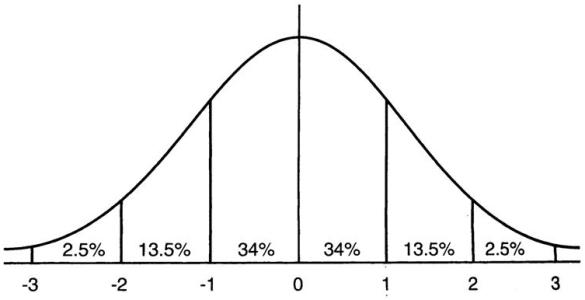


图5-9 Batch Normalization效果示意图

对这种Relu函数发生的神经元失活问题，我调查了一些方案，认为将Relu函数换成leaky-Relu函数或者插入Batch Normalization层，理论上可以消除问题。

表5-4 改进后的再加深CNN结构

|  |
| --- |
| **Input (48 x 48)** |
| 5x5 Convolutional Layer - 16 filters |
| Batch Normalization Layer |
| 5x5 Convolutional Layer – 32 filters |
| Batch Normalization Layer |
| 5x5 Convolutional Layer – 32 filters |
| Batch Normalization Layer |
| 2x2 Average Pooling Layer |
| Dropout Layer – 50% |
| 5x5 Convolutional Layer - 64 filters |
| Batch Normalization Layer |
| 5x5 Convolutional Layer – 64 filters |
| Batch Normalization Layer |

续表5-4

|  |
| --- |
| 5x5 Convolutional Layer – 64 filters |
| Batch Normalization Layer |
| 2x2 Average Pooling Layer |
| Dropout Layer – 50% |
| 5x5 Convolutional Layer - 128 filters |
| Batch Normalization Layer |
| 5x5 Convolutional Layer – 128 filters |
| Batch Normalization Layer |
| 5x5 Convolutional Layer – 128 filters |
| Batch Normalization Layer |
| 2x2 Average Pooling Layer |
| Dropout Layer – 50% |
| Flatten Layer |
| Full Connected Layer – 256 |
| Dropout Layer – 50% |
| Softmax - 7 |

在本课题中我采用了插入Batch Normalization层的方式。Batch Normalization的方法由文[4]提出，如图5-9所示，其效果简单来说，是让上一层的神经元的参数分布强制服从正态分布，对Sigmoid激活函数，它能让大部分神经元的参数都落在Sigmoid函数的敏感区，从而消除梯度消失。对于Relu函数，它也可以起到让大部分函数不落在Relu函数的空值区间的效果。如表5-4所示，我在每个卷积层后面都插入了一层Batch Normalization层。经过实验，插入Batch Normalization层之后梯度消失的现象消失。最终在验证集上正确率达到65.13%，但是过拟合较重，不适合继续训练。图5-10为训练历史曲线，各颜色曲线含义如左上角所示。模型训练结果见图5-11和图5-12，参数量为2,143,335。在第82个Epoch发生Early Stopping，总耗时742秒。

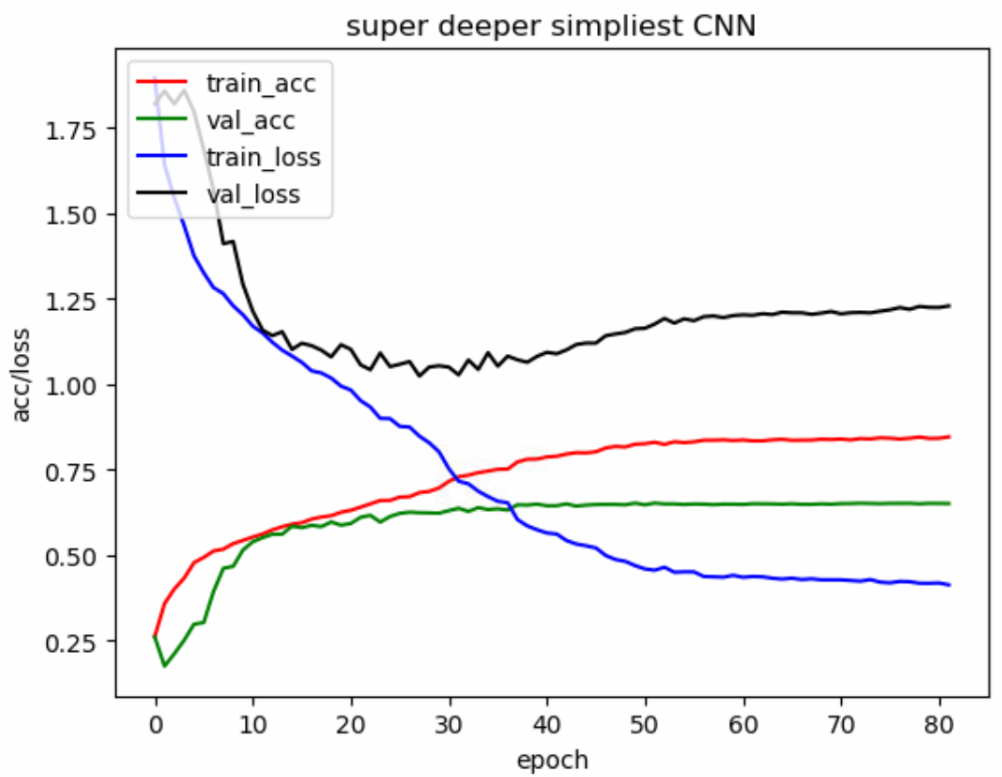


图5-10 再加深CNN训练历史图表

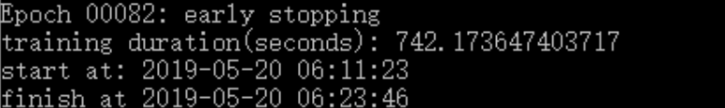


图5-11 再加深CNN训练时间

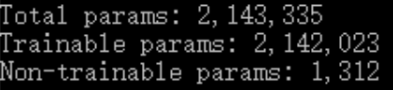
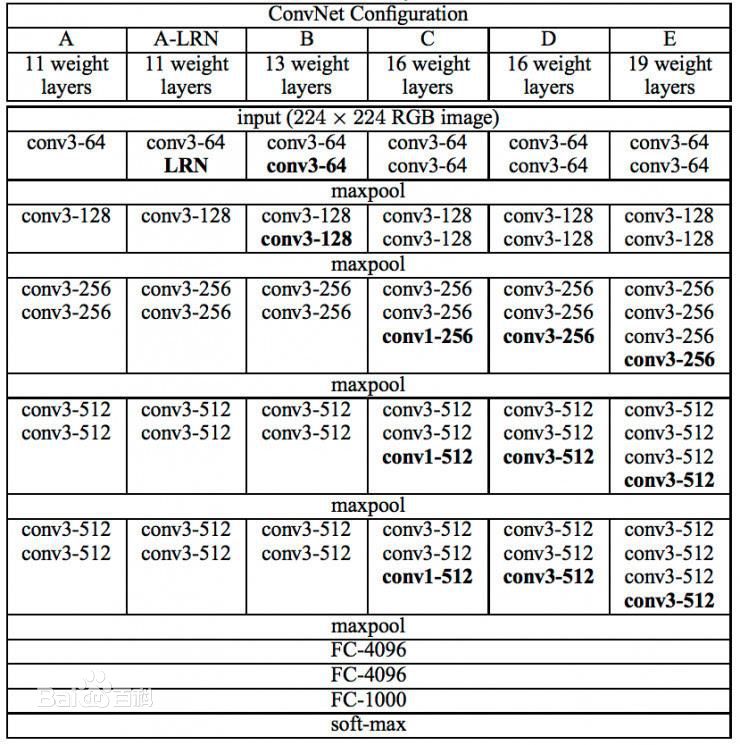


图5-12 再加深CNN参数量

## 5.4 参照VGG-16继续优化

为了在减少参数量、提高正确率、减少过拟合等性能上寻求进一步的突破，我参考了现有的网络VGG。VGG是一种常用于人脸相关的深度学习任务的模型，是2014年ILSVRC竞赛的第二名， VGG模型在迁移学习任务中的表现要优于GoogLeNet，可以认为它的结构拥有更好的任务泛化能力。

表5-5 VGG网络结构示意图



其实VGG的结构并不复杂，如上表5-5所示，主要由卷积层-最大值池化层的堆叠，和若干个全连接层组成。他从AlexNet发展而来，不同于AlexNet的特点是其全部采用了3x3的卷积核和2x2的池化核，VGG的作者认为小卷积核的表现更好，他用小的卷积核的堆叠实现了比大卷积核更好的效果，两个3x3的卷积堆叠获得的感受野相当一个5x5的卷积的感受野，而3个3x3卷积的堆叠获取到的感受野就相当于一个7x7的卷积的感受野。

模仿VGG对网络进行修改之后，网络结构如表5-6所示。VGG的输入是224x224的图像，相比FER2013的图像更大。之前已经发生过疑似因为图片太小信息不足发生的梯度消失现象，因此我试着将卷积滤镜数量等参数只取VGG的四分之一。

表5-6 仿VGG的CNN网络结构

|  |
| --- |
| **Input (48 x 48)** |
| 3x3 Convolutional Layer - 16 filters |

续表5-5

|  |
| --- |
| Batch Normalization Layer |
| 3x3 Convolutional Layer - 16 filters |
| Batch Normalization Layer |
| 2x2 Max Pooling Layer |
| Dropout Layer – 50% |
| 3x3 Convolutional Layer - 32 filters |
| Batch Normalization Layer |
| 3x3 Convolutional Layer - 32 filters |
| Batch Normalization Layer |
| 2x2 Max Pooling Layer |
| Dropout Layer – 50% |
| 3x3 Convolutional Layer - 64 filters |
| Batch Normalization Layer |
| 3x3 Convolutional Layer – 64 filters |
| Batch Normalization Layer |
| 3x3 Convolutional Layer – 64 filters |
| Batch Normalization Layer |
| 2x2 Max Pooling Layer |
| Dropout Layer – 50% |
| 3x3 Convolutional Layer - 128 filters |
| Batch Normalization Layer |
| 3x3 Convolutional Layer - 128 filters |
| Batch Normalization Layer |
| 3x3 Convolutional Layer - 128 filters |
| Batch Normalization Layer |
| 2x2 Max Pooling Layer |
| Dropout Layer – 50% |
| 3x3 Convolutional Layer - 256 filters |

续表5-5

|  |
| --- |
| Batch Normalization Layer |
| 3x3 Convolutional Layer - 256 filters |
| Batch Normalization Layer |
| 3x3 Convolutional Layer - 256 filters |
| Batch Normalization Layer |
| 2x2 Max Pooling Layer |
| Dropout Layer – 50% |
| Flatten Layer |
| Full Connected Layer – 256 |
| Dropout Layer – 50% |
| Full Connected Layer –128 |
| Dropout Layer – 50% |
| Full Connected Layer –64 |
| Dropout Layer – 50% |
| Softmax - 7 |

尝试进行训练后，如图5-13所示，训练极慢，收敛过早，发生了梯度消失。

梯度消失现象指的是，当我们在做反向传播，计算损失函数对权重的梯度时，随着越向后传播，梯度变得越来越小，这就意味着在网络的前面一些层的神经元，会比后面的训练的要慢很多，甚至不会变化。

最典型的容易发生梯度消失的激活函数是Sigmoid这样的曲线非线性函数，当神经元输出值落在Sigmoid的两头较平坦的区间时，神经元参数的改变，经过激活函数后对输出值的贡献很小，导致部分神经元几乎失活。且Sigmoid函数的导数最大值也仅有0.25，很容易导致梯度过小引发梯度消失。最常见的解决方法是将Sigmoid激活函数换成Relu激活函数。

本课题中为了防止梯度消失，一开始就用了Relu函数，且这次每一个卷积层后面都被接上了可以缓解梯度消失的Batch Normalization层，但是却出现了意外的梯度消失，我认为是因为原始图像太少，信息量不足，经过多个卷积层之后有效信息损失过大导致的。

因此我适量削减了两层卷积层，将原本3层的3x3卷积核的卷积层变成了2层。经过修改后，如图5-14历史曲线所示模型不再发生梯度消失，模型训练结果见图5-15和图5-6，最终成绩61.33%。参数量2,041,527。

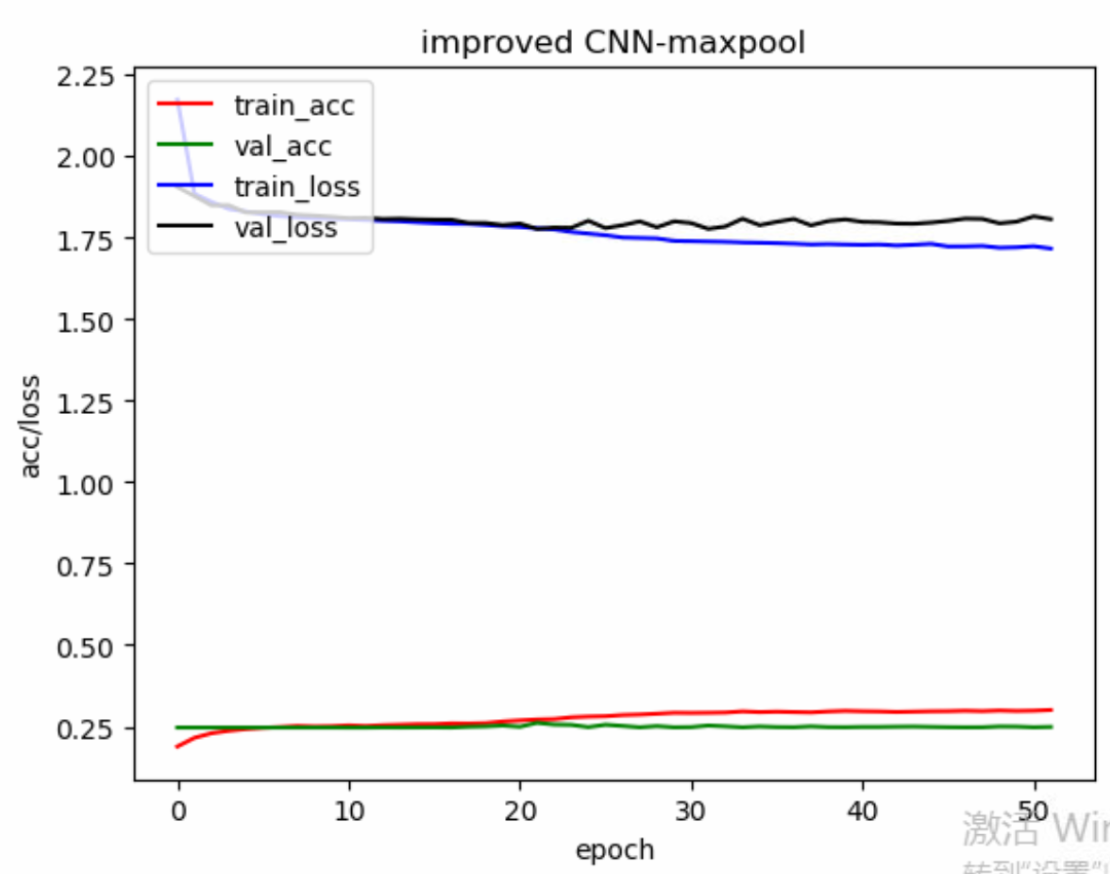


图5-13 仿VGG的CNN网络出现梯度消失

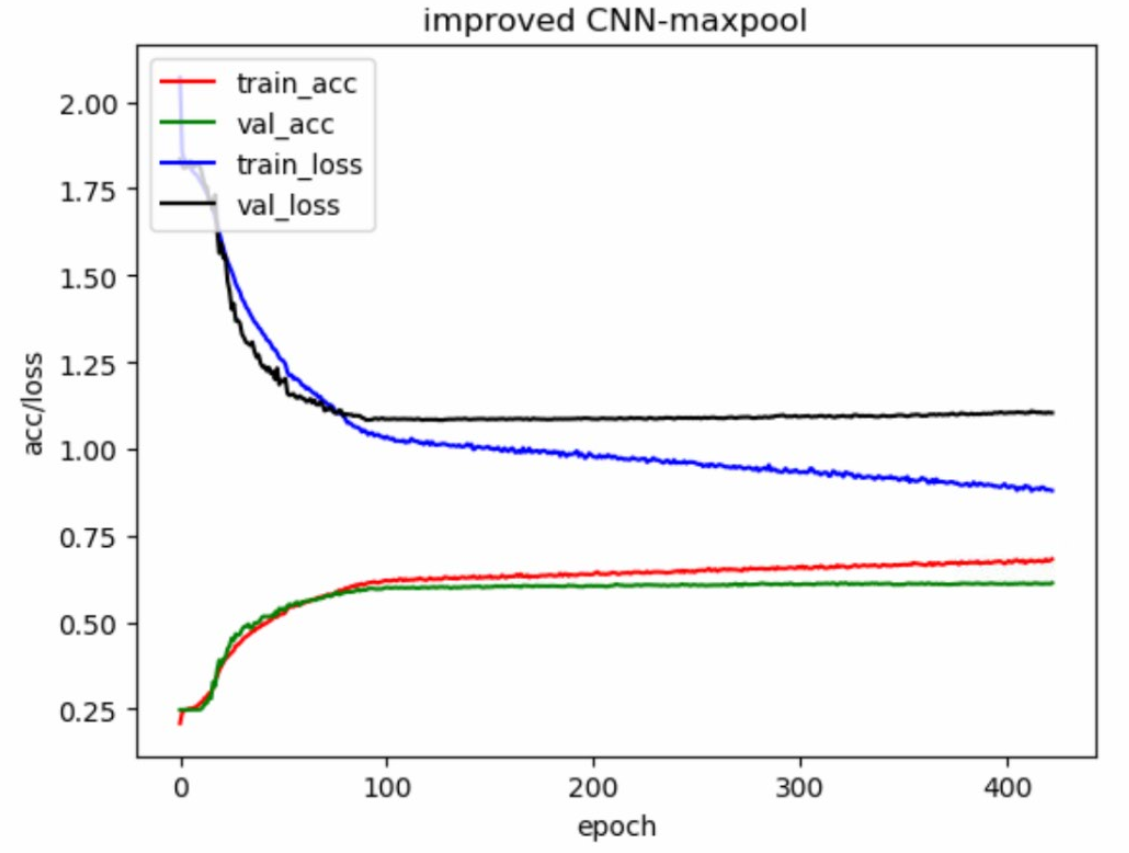


图5-14 修正后的仿VGG的CNN网络的训练历史图表

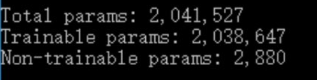


图5-15 修正后的仿VGG的CNN网络的参数量

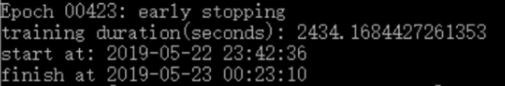


图5-16 修正后的仿VGG的CNN网络的训练时间

## 5.5 进一步改善

### 5.5.1 卷积核优化

模仿VGG对模型进行改进之后，虽然在过拟合上表现比之前的Super Deep CNN要好，但最终正确率却更低，因此本节继续对模型的性能改善进行试验和研究。

从上一节的仿VGG模型发生梯度消失可以看出，由于VGG模型建立时使用的是224x224的源图像，在深度的定量以及参数定量上，都不能直接适用于图片仅有48x48大小的FER2013数据集。

VGG的中心思想就是认为“5x5卷积核可以用2个3x3卷积核替代”。因为VGG从Alexnet发展而来，Alexnet的深度相比VGG较浅，VGG是将Alexnet中的大卷积核拆成了多层小卷积核。上一节，通过削减卷积层数量才消除了梯度消失的现象，因此为了减少卷积层的数量的同时尽量保持性能，现在对上述VGG的思路进行逆向操作，将多个小卷积核的卷积层合并成一个大卷积核的卷积层。

### 5.5.2 平均池化与最大池化

VGG中池化层使用的都是最大值池化Max Pool而不是平均池化Average Pool。

Max Pool和Average Pool同样都能优化平移不变性和旋转不变性。他们的一大区别在于，Max Pool更多地保留了边缘信息，而Average Pool更多地保留了背景信息。

因此本次样本图片偏小，我认为Max Pool会过早、过大地消去图片中的有效纹理信息，降低最终的正确率。

因此在最终模型中，我将Max Pool替换成了Average Pool。

### 5.5.3 用GAP层代替FC层

论文[2]的第3.2节中提出了用Global Average Pooling层替代全连接层的方法。去除全连接层可以极大减少参数量，因此在最终模型中采用了这个方法，将之前所有的全连接层，替换成一个7滤镜的卷积层和Global Average Pooling层的组合。这样最后的一层的数据是7个3\*3的特征图，Global Average Pooling会将每一张特征图所有像素点取均值，输出一个数据值。有两个优点：一是Global Average Pooling在特征图与最终的分类间转换更加简单自然，运算量更少；二是不像全连接层那样需要大量训练调优的参数，因此能降低空间参数，使模型更加健壮，抗过拟合效果更佳。如图5-17是GAP的工作原理示意图。

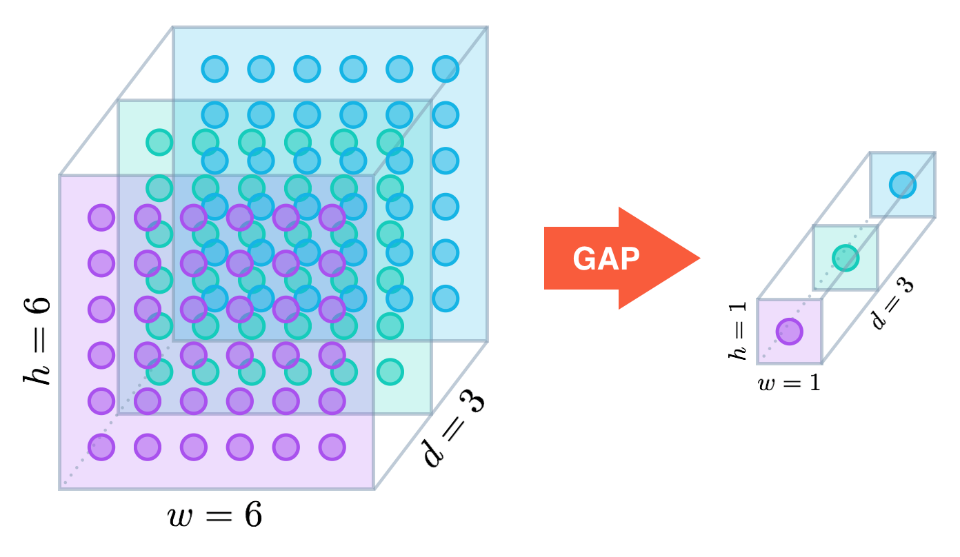


图5-17 GAP层的作用示意图

## 5.6 特化模型

### 5.6.1 特化模型的意义

特化模型在本文中，指代的是在原模型的基础上，从原本的数据集全集中，选取一部分有共同特征的数据，比如同为正面脸的数据，同为男性的数据，以这些特化的数据对原模型进行微调（Fine Tuning）。

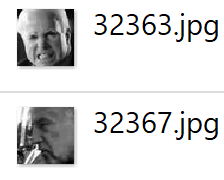


图5-18 FER2013中存在的正面脸和非正面脸样本

如图5-18所示，数据集中存在正面的样本和非正面的样本。数据集中大部分都是正面脸的样本，但也存在相当一部分非正面脸的样本。在本课题中，我将原数据集用Haar分类器一分二，分为了正面脸数据集和非正面脸数据集，分别在原本用全集训练好的模型上进行微调，得到两个特化模型。

在实际使用模型时，仍然先用Haar分类器，判断实际样本属于正面脸还是非正面脸，若是正面脸则调用正面脸模型，若是非正面脸则调用非正面脸模型。

经过实验，对于一个已经训练到饱和无法继续提升正确率的模型，这个做法能在原基础上对正确率有进一步的提升。

### 5.6.2 特化模型与既存思路的异同

A. 特化模型和一些大型任务中“先聚类后分类”的思路相似。

比如对于大量网络新闻的处理，往往会先提取一部分特征聚类成大概的主题，再对其中的内容做细致的分类。

但先聚类后分类的聚类往往是无监督的聚类，选取最相近的聚类进行分类，在5.7.1节中失败的利用Facenet进行的优化就是基于这个思路，想要借助特征向量聚类。特化模型是选取特定的成品模型，对原数据进行有利于特定识别主题的预分类，来代替无监督的聚类。

B. 特化模型和多输入单输出的思路相似。

在一些任务中，会将汽车图片和汽车的拍摄角度等同时输入模型，做出多输入单输出的模型。

但多输入单输出的模型一般需要全连接层对合并后的数据进行分类，而譬如本课题中实现的最终模型，为了追求更小的参数量，用GAP层代替了全连接层。全连接层参数量较大，并且容易过拟合。在模型中不希望加入全连接层以减少数据量的话，可以用特化模型作为多输入单输出的替代方案。

## 5.7 未采用的尝试方向

### 5.7.1 用Facenet特征向量辅助识别

先简单介绍一下Facenet，Facenet是谷歌研发的一种神经网络模型结构[3]，用于生成图像到欧式空间上点的映射，映射关系表现了人脸之间的相似度，最终输出的128维特征向量可以用于对不同的人脸进行识别。

这次尝试的思路是尝试将Facenet输出的128维特征向量用作表情识别模型的输入，以探究是否能够实现比输入原图更高的正确率。

本次尝试使用的是sklearn.svm支持向量机模块中的SVC函数。其优点在于仅仅对函数传入输入输出数据就可以实现基本的分类任务，对数据的格式要求比较低，代码简单，适合对想法进行最基本的验证

以下是代码片段：

clf=SVC(kernel='rbf',gamma='auto')

clf=train\_model(train\_x,train\_y)

进行了三种尝试：

• 将原始图片+Facenet 128维特征向量一起作为输入，输出7项表情分类。

• 将原始图片平展成2304个像素点作为输入，输出7项表情分类。

• 用Facenet 128维特征向量作为输入，输出7项表情分类。

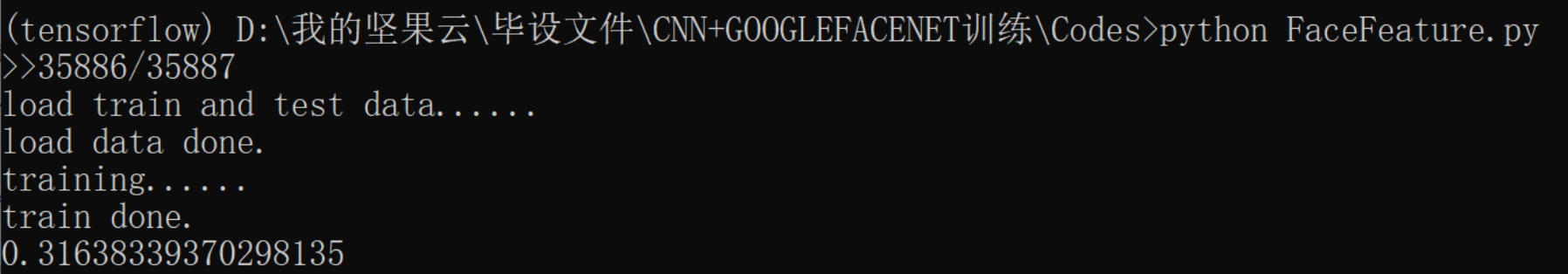


图5-19 将原始图片+Facenet 128维特征向量一起作为输入

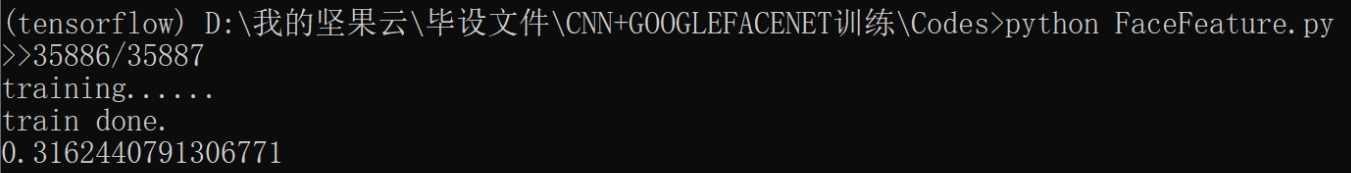


图5-20 将原始图片平展成2304个像素点作为输入

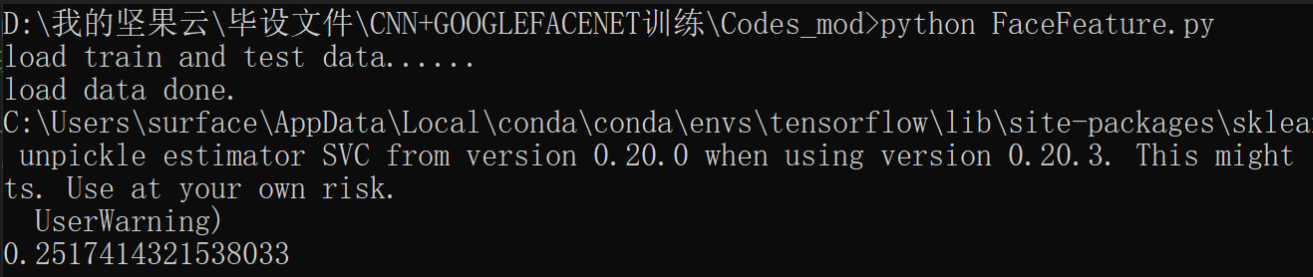


图5-21 用Facenet 128维特征向量作为输入

实验结果如图5-19，图5-20，图5-21所示，仅用Facenet 128维特征向量作为输入时，得出的分类正确率远低于以原始图片作为输入时的正确率，仅仅有25.17%。7项分类、随机猜想时正确率也有14.28%，可以认为仅以Facenet 128维特征向量作为输入的正确率低到接近随机猜想的水平。

将原始图片和Facenet 128维特征向量一起作为输入时，得出的分类正确率31.64%非常接近以原始图片作为输入时的正确率31.62%，仅仅相差0.02%。可以认为Facenet输出的128维特征向量没有从原图中提取出多少有价值的信息。

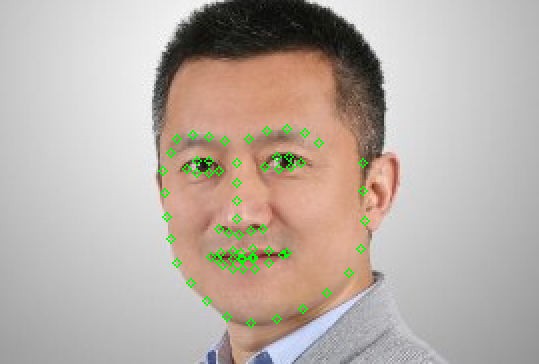


图5-22 用程序取得的人脸边缘特征点

失败的原因猜想认为是因为FER2013数据集的图片是48x48的低清晰度图片，人脸边缘信息本来就非常有限，Facenet提取特征向量时就非常困难。上图5-22是我用简单的测试程序输出的人脸边缘标识点，右图是FER2013数据集，可以看到在FER2013数据集上，人脸边缘特征之间仅仅有一两个像素的距离，我认为这极大地限制了Facenet提取出的特征向量的有效性。

综上所述，经过试验，我认为Facenet对FER2013数据集上的表情识别不会有太大的作用，因此没有对这个方向进行进一步的深入研究。但这次试验对本课题中提出的特化模型的思路有一定启发。

### 5.7.2 性别特化模型

在对模型进行正脸/非正脸的分化之前，我先尝试了对模型进行男性和女性的分化。

通过现成的性别分类器，我将数据集自动分为了男性部分和女性部分，作出了男性数据集和女性数据集，然后分别训练，得到了男性特化模型和女性特化模型。

但是很遗憾没有什么效果。最终两个特化模型的合计正确率在原模型的基础上还降低了0.05%，基本认为属于误差。

失败的原因我认为是在这个48x48的清晰度下，男性和女性之间的表情特征差异过于小，不足以辅助判断表情。

# 第六章 最终模型及其优化效果的验证

## 6.1 最终CNN模型的结构与性能

根据上一张所提出的优化点进行改进后，最终模型结构如下表6-1所示：

表6-1 最终CNN模型结构

|  |
| --- |
| **Input (48 x 48)** |
| 7x7 Convolutional Layer - 16 filters |
| Batch Normalization |
| 7x7 Convolutional Layer - 16 filters |
| Batch Normalization |
| 2x2 Average Pooling Layer |
| Dropout Layer – 50% |
| 5x5 Convolutional Layer - 32 filters |
| Batch Normalization |
| 5x5 Convolutional Layer - 32 filters |
| Batch Normalization |
| 2x2 Average Pooling Layer |
| Dropout Layer – 50% |
| 3x3 Convolutional Layer - 64 filters |
| Batch Normalization |
| 3x3 Convolutional Layer – 64 filters |
| Batch Normalization |
| 2x2 Average Pooling Layer |
| Dropout Layer – 50% |
| 3x3 Convolutional Layer - 128 filters |
| Batch Normalization |
| 3x3 Convolutional Layer – 128 filters |
| Batch Normalization |
| 2x2 Average Pooling Layer |

续表6-1

|  |
| --- |
| 3x3 Convolutional Layer - 256 filters |
| Dropout Layer – 50% |
| Batch Normalization |
| 3x3 Convolutional Layer – 7 filters |
| Batch Normalization |
| Global Average Pooling - Softmax |

在上一次模型的基础上，将3x3卷积核改成了从输入到输出尺寸递减的卷积核；将Max Pool换成了Average Pool；将全连接层全部去除换成了GAP层；用特化模型的方法按照正面脸/非正面脸分别Fine-Tuning。

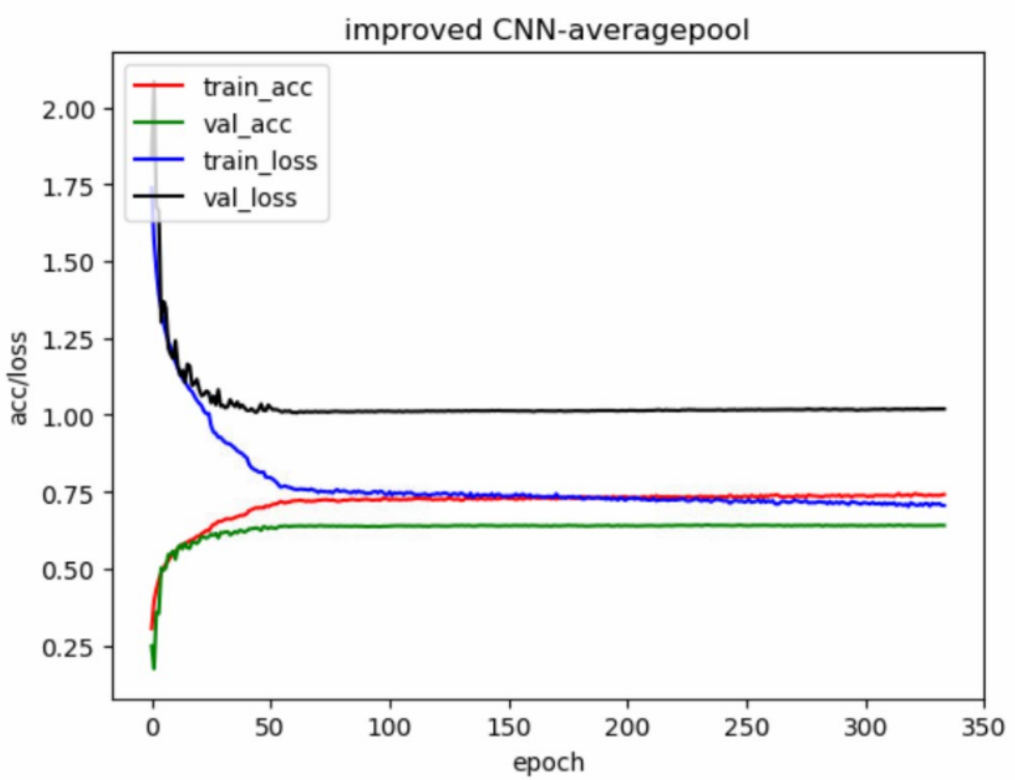


图6-1 最终CNN模型的训练历史图表

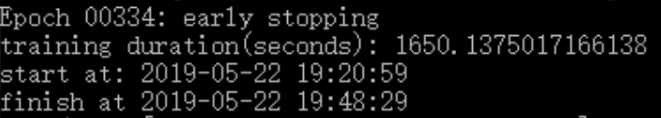


图6-2 最终CNN模型的训练时间

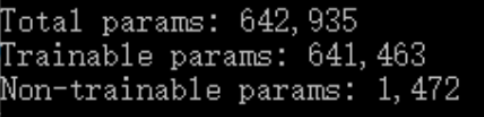


图6-3 最终CNN模型的参数量

在全集上训练得出的最终正确率是64.15%，模型训练结果见图6-2和图6-3，在仿VGG-16的模型基础上上升了2.82%。参数量只有642,935，得益于用GAP层替换了全部全连接层，参数量相比上文中5-4节中的2,041,527有了显著减少。

## 6.2 特化模型的正确率评估

下图6-4是将最终模型在非正面脸数据集上Fine Tuning直到收敛得到的图表，图6-5是在正面脸数据集上Fine Tuning到收敛的图表，图6-6直接用全集继续训练的图表。

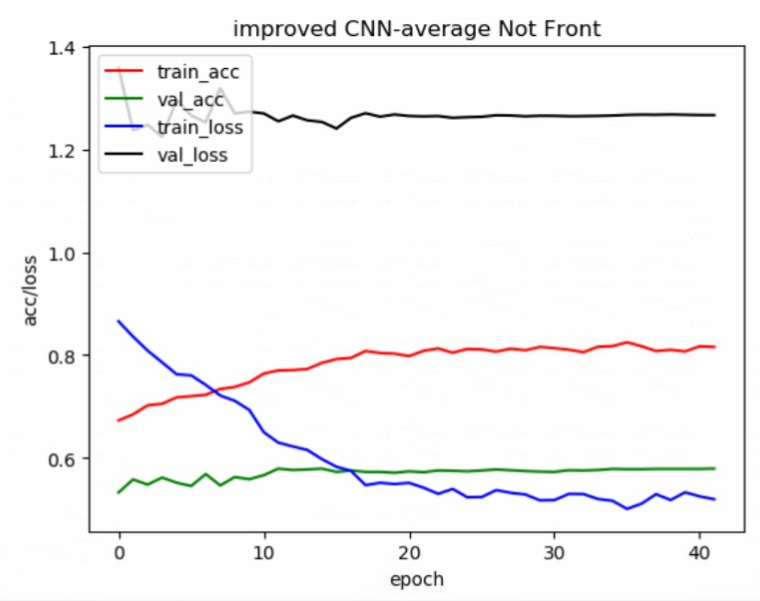


图6-4 最终CNN非正面脸特化模型的训练历史图表

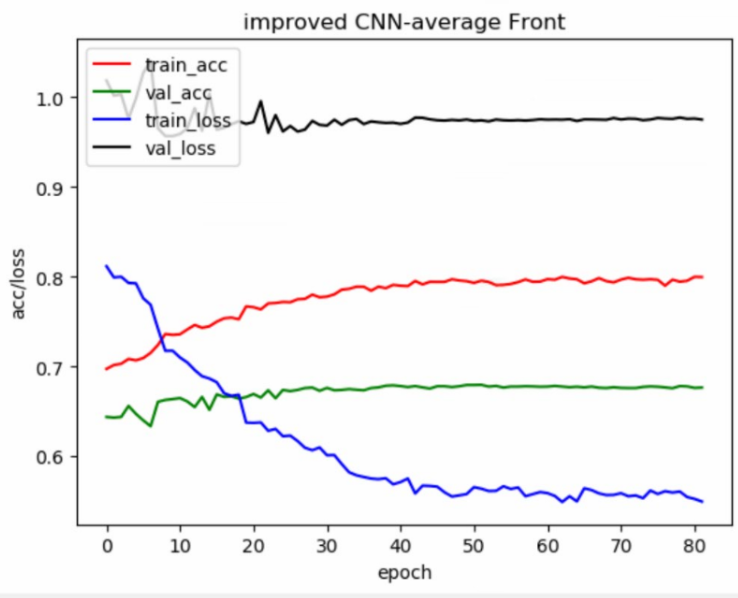


图6-5 最终CNN正面脸特化模型的训练历史图表

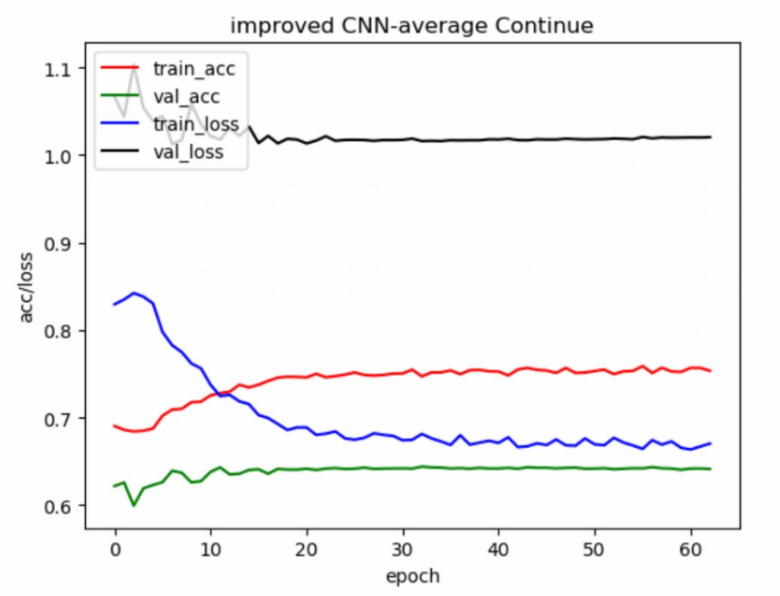


图6-6 最终CNN模型直接继续训练的训练历史图表

这三者的结果分别是：正面脸67.92%，非正面脸0.57.93%，继续训练64.29%。

完整的FER2013数据集总共有28709张图片，通过分类器分别筛选出21128张正面脸图片，以及7581张非正面脸图片。已知在正面脸上的正确率是67.92%，在非正面脸上的正确率是57.93%，可以用简单的计算得出综合正确率：

input：

accFront=0.6792

accNotFront=0.5793

numFront=21128

numNotFront=7581

numAll=numFront+numNotFront

accAll=accFront\*(numFront/numAll)+accNotFront\*(numNotFront/numAll)

print('after=', 100\*accAll)

output：

after= 65.28200529450694

由此得出最终合并正确率是65.28%，相比之前的成绩64.15%高了1.13%。而直接继续训练的最终最高正确率是64.29%，并没有在之前的基础上有太明显的提升。

如图4-8-9所示，特化前的模型在正面脸数据集上的正确率是66.5%，在非正面脸数据集上的正确率是57.6%。特化后在正面脸上的表现提升了1.42%，在非正面脸上的表现0.32%，综合提升了1.13%。

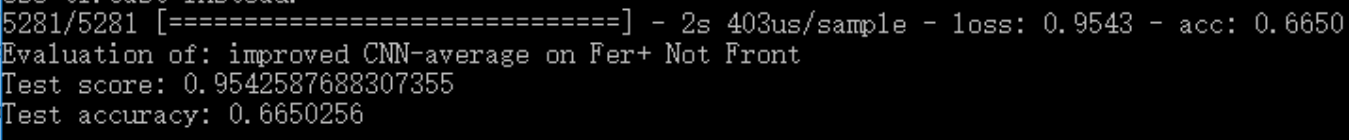


图6-7 最终CNN模型特化前在正面脸样本集上的正确率

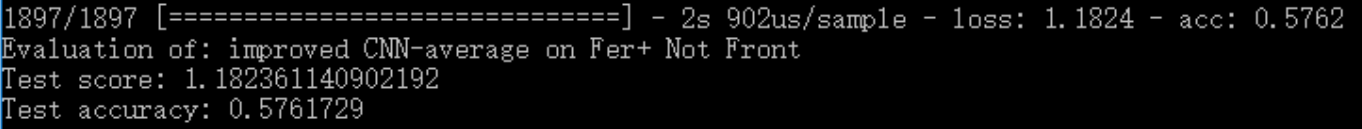


图6-8 最终CNN模型特化前在非正面脸样本集上的正确率

对于特化模型有效的原因，我认为是因为在网络规模不变的情况下，收缩了样本空间，原本的模型中掺杂着正面脸和非正面脸的共用特征，而收缩了样本空间之后，模型对有共同特征的一部分局部样本进行优化，样本之间的相似度提高了，对这部分样本来说模型中的“噪声”减少了，因此能一定程度上提高正确率。

## 6.3 最终模型在FER+上的表现

下图6-9和图6-10是将最终模型在FER+数据集上从零开始训练的结果，因为FER+中标注错误较少，因此在FER+上得出的正确率远远高于在FER2013上得出的正确率。最后得到的正确率为79.69%。

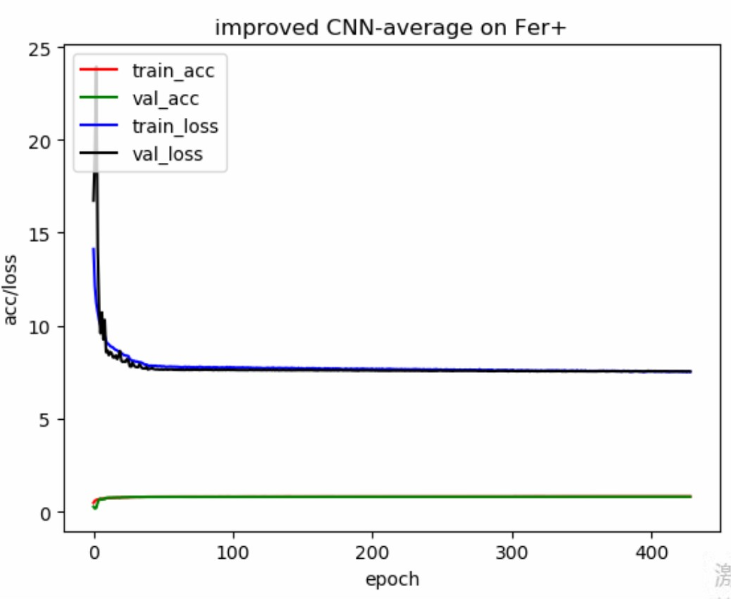


图6-9最终CNN模型在FER+数据集上的训练历史图表1

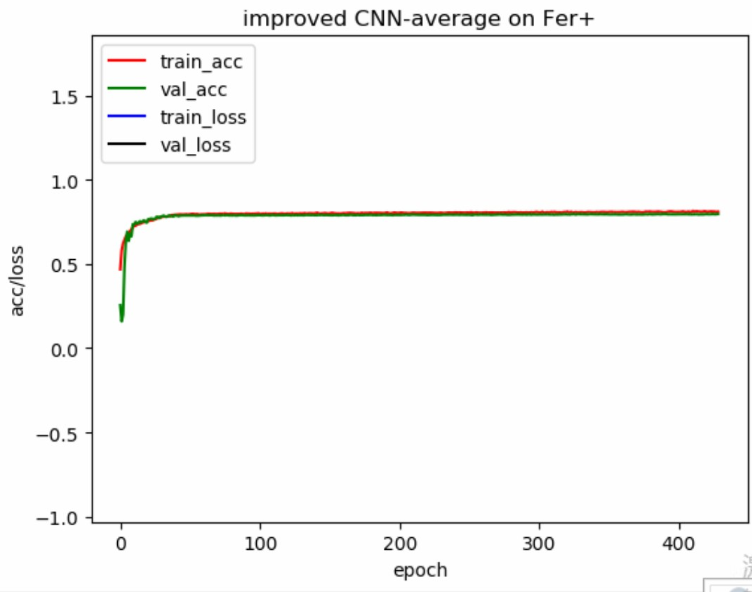


图6-10最终CNN模型在FER+数据集上的训练历史图表2

如下图6-11和图6-12，对FER+也进行模型特化。FER+上正面脸模型的正确率达到82.56%，非正面脸模型的正确率达到72.96%，用和上一节相同的方法，可以算出综合正确率为80.02%，在原基础上提升了0.32%。

演示程序中仅识别正面脸，因此82.56%这个最高成绩模型将被用于演示程序。

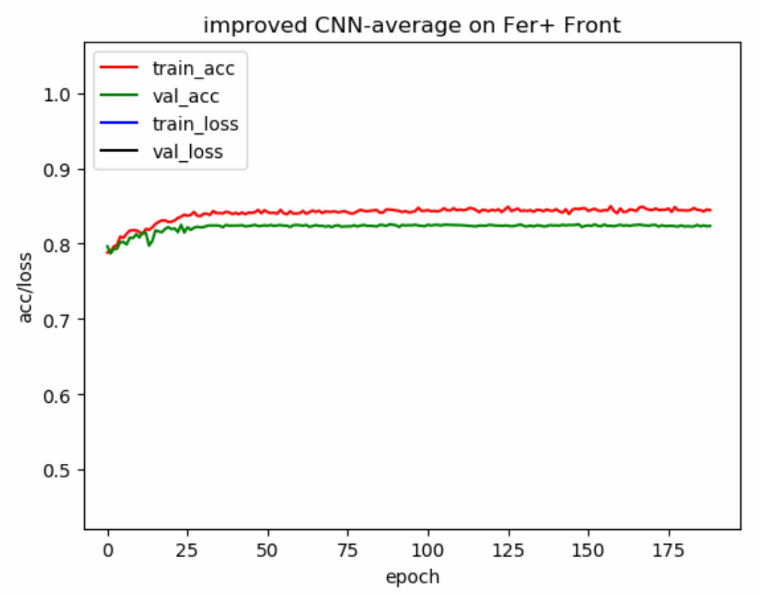


图6-11最终CNN模型在FER+数据集上的正面脸特化模型训练历史图表

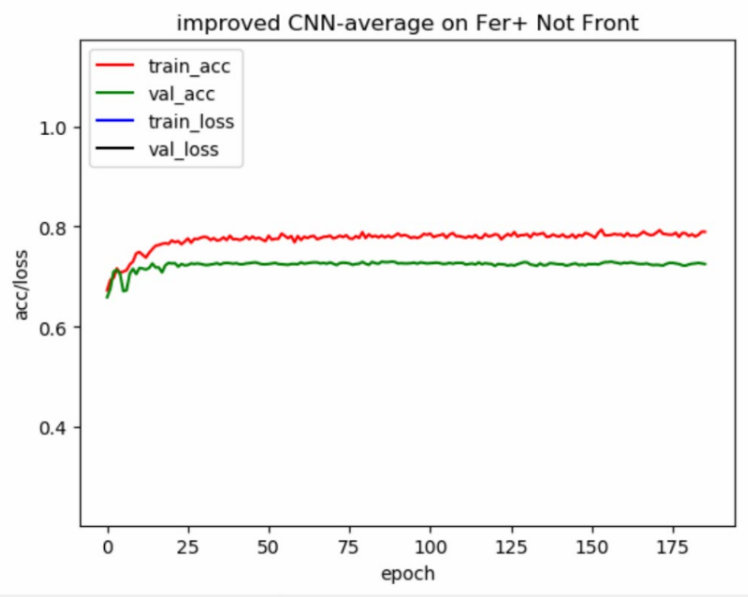


图6-12最终CNN模型在FER+数据集上的非正面脸特化模型训练历史图表

## 6.4 各模型的正确率表现汇总

下表6-2中汇总了本文中出现的各个模型的正确率和参数量，方便对比。

最终的CNN模型的主要特点有：用输入到输出尺寸递减的卷积核代替小卷积核以降低深度；用平均池化替代了最大池化；用 GAP层代替了全连接层；按照正面脸/非正面脸分别Fine-Tuning建立特化模型进一步提升了正确率。

和之前的CNN网络结构相比，最终的CNN模型在正确率、参数量、过拟合程度上都有优势。

表6-2 本文中出现的各个模型的正确率汇总

|  |  |  |  |
| --- | --- | --- | --- |
|  | **FER2013** | **FER+** | **参数量** |
| 5.1 最简CNN | 49.41% | 无 | 3,967,815 |
| 5.2 加深CNN | 56.52% | 无 | 112,551 |
| 5.3 再加深CNN | 65.13% | 无 | 2,143,335 |
| 5.4 仿VGG-16优化CNN | 61.33% | 无 | 2,041,527 |
| 6.1最终CNN模型 | 64.15% | 79.69% | 642,935 |
| 6.2最终模型的特化模型（综合正确率） | 65.28% | 80.02% | 642,935 |

# 第七章 总结与展望

本论文从结构最简的CNN开始，采用了一些近年流行的优化方法，逐步优化，阐述了每个优化点对模型的改善作用和原理。最终在FER2013数据集上实现了65.28%的正确率，在FER+数据集上实现了80.02%的正确率。

在优化过程中，两次出现了梯度消失的现象。实验表明加入Batch Normalization层和降低模型层数可以抑制梯度消失的发生。

在模仿VGG-16结构的过程中出现了梯度消失和性能不佳的现象，最终为了消除梯度消失和提高性能，放弃了VGG-16特有的全部小卷积核结构，反而达到了更好的效果，可以看出模型的参数选取会受到样本形态的影响，不能简单照搬。

用Global Average Pooling代替全连接层是近年来的流行做法，越来越多地被各种图像分类任务所采纳，本课题中应用了这一做法，的确显著减少了参数量和过拟合。

本课题中用Python语言完成了主要基于Opencv模块的演示程序，成功将训练好的模型用于现场演示程序，可以现场直接识别摄像头中的多张人脸，标出人脸对应的表情，实现了神经网络模型的实用化。

现在特化模型这个思路还比较原始，只是将数据集切分训练，训练出多个在局部样本上表现更优的特化模型，在使用中分别调用。

目前分别训练两个模型，相当于用了两倍的神经元，非常浪费资源。可以尝试将特化模型的思路融合进单一的神经网络，但不是做成简单的多输入单输出模型，而是试着在特化时追加适当量的专用神经元，这些专用的神经元将只接收有特定标签的数据的损失传播。

今后会将“把特化模型的思路融合进单一的神经网络”作为研究方向。

# 谢辞

通过本次毕业设计，我掌握了人工智能课题的一整套研究方法和必要的知识体系，相信今后我也能在这个方向上有所突破，非常感谢贾振堂老师的无私帮助和热心指导，和老师的交谈为我的思路启发许多。多亏贾振堂老师的帮助，我才能在人工智能这个初次接触的领域上顺利前进。

也感谢在百忙之中抽出时间来评审论文和参加答辩的老师。最后感谢上海电力大学给我提供了良好的学习环境。

# 参考文献

1. Li S, Deng W. Deep Facial Expression Recognition: A Survey[J/OL]. Eprint arXiv:1804.08348, 2018. <https://arxiv.org/abs/1804.08348>.
2. Lin M, Chen Q, Yan S. Network in network[J/OL]. Eprint arXiv: 1312.4400, 2013. <https://arxiv.org/abs/1312.4400>.
3. Schroff F, Kalenichenko D, Philbin J. Facenet: A unified embedding for face recognition and clustering[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 815-823.
4. Rasmus A, Berglund M, Honkala M, et al. Semi-supervised learning with ladder networks[C]//Advances in neural information processing systems. 2015: 3546-3554.
5. Georgescu M I, Ionescu R T, Popescu M. Local learning with deep and handcrafted features for facial expression recognition[J]. IEEE Access, 2019, 7: 64827-64836.
6. Barsoum E, Zhang C, Ferrer C C, et al. Training deep networks for facial expression recognition with crowd-sourced label distribution[C]//Proceedings of the 18th ACM International Conference on Multimodal Interaction. ACM, 2016: 279-283.
7. Connie T, Al-Shabi M, Cheah W P, et al. Facial expression recognition using a hybrid CNN–SIFT aggregator[C]//International Workshop on Multi-disciplinary Trends in Artificial Intelligence. Springer, Cham, 2017: 139-149.
8. Goodfellow I, Bengio Y, Courville A. Deep learning[M]. Cambridge : MIT press, 2016.
9. Gu J, Wang Z, Kuen J, et al. Recent advances in convolutional neural networks[J]. Pattern Recognition, 2018, 77: 354-377.
10. Zhang W, Tanida J, Itoh K, et al. Shift-invariant pattern recognition neural network and its optical architecture[C]//Proceedings of annual conference of the Japan Society of Applied Physics. 1988.
11. 周祥全, 张津. 深层网络中的梯度消失现象[J]. 科技展望, 2017(27): 284.
12. 何明编. 大学计算机基础[M]. 南京: 东南大学出版社, 2015.
13. Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. Eprint arXiv:1412.6980, 2014. <https://arxiv.org/abs/1412.6980>