



上海大学1

毕业设计（论文）

题目：基于SQuAD数据集的问题回答模型

*装订线*

学 院：机电工程与自动化学院

专 业：电气工程及其自动化

学 号：15122876

学生姓名：雷清淇

指导教师：周维民

起讫日期：

上海大学学士学位论文

**基于SQuAD数据集的问题回答模型**

姓 名： 雷清淇

导 师： 周维民

学科专业：自动化

Q&A Model on SQuAD Data Set

**Q&A Model on SQuAD Data Set**

Candidate：

Supervisor：

Major：Electrical Engineering and Automation

**School of Mechatronical Engineering and Automation, Shanghai University**

**January, 2018**

目 录

[目 录 I](#_Toc515027988)

[摘 要 III](#_Toc515027989)

[ABSTRACT IV](#_Toc515027990)

[第一章 绪论 1](#_Toc515027991)

[1.1 课题背景及意义 1](#_Toc515027992)

[1.1.1 课题背景 1](#_Toc515027993)

[1.1.2 课题意义 1](#_Toc515027994)

[1.2 课题研究现状 2](#_Toc515027995)

[1.1.3 国外研究现状 2](#_Toc515027996)

[1.1.4 国内研究现状 3](#_Toc515027997)

[1.3 课题应用前景 4](#_Toc515027998)

[1.4 论文主要研究内容 5](#_Toc515027999)

[第二章 程序设计竞赛证照系统人脸检测的方法 7](#_Toc515028000)

[2.1 基础理论原理 7](#_Toc515028001)

[2.1.1 反向传播算法 7](#_Toc515028002)

[2.1.2 卷积神经网络 9](#_Toc515028003)

[2.2 目标检测方法 13](#_Toc515028004)

[2.2.1 概述 13](#_Toc515028005)

[2.2.2 R-CNN 13](#_Toc515028006)

[2.2.3 Fast R-CNN 14](#_Toc515028007)

[2.2.4 Faster 16](#_Toc515028008)

[2.3 深度学习框架 18](#_Toc515028009)

[2.3.1 Caffe 18](#_Toc515028010)

[2.3.2 TensorFlow 19](#_Toc515028014)

[第三章 程序设计竞赛证照系统人脸检测的实现 21](#_Toc515028015)

[3.1 Faster R-CNN人脸检测的实现 21](#_Toc515028016)

[3.1.1 环境配置 21](#_Toc515028017)

[3.1.2 准备数据 25](#_Toc515028019)

[3.1.3 训练模型 26](#_Toc515028020)

[3.1.4 测试模型 27](#_Toc515028021)

[3.1.5 性能指标 28](#_Toc515028022)

[3.2 OpenCV人脸检测的实现 30](#_Toc515028023)

[3.2.1 概述 30](#_Toc515028024)

[3.2.2 测试模型 31](#_Toc515028025)

[3.3 效果对比 31](#_Toc515028028)

[第四章 程序设计竞赛证照系统网页开发的实现 35](#_Toc515028029)

[4.1 Django 35](#_Toc515028030)

[4.1.1 概述 35](#_Toc515028031)

[4.1.2 MTV模式 35](#_Toc515028032)

[4.1.3 Booststrap 37](#_Toc515028033)

[4.2 搭建系统 37](#_Toc515028039)

[4.2.1 Python语言 37](#_Toc515028040)

[4.2.2 开发步骤 38](#_Toc515028041)

[4.3 人脸检测模型接口 40](#_Toc515028042)

[4.3.1 逻辑处理 40](#_Toc515028043)

[4.3.2 模型嵌入 41](#_Toc515028044)

[第五章 结论与展望 43](#_Toc515028045)

[5.1 总结 43](#_Toc515028046)

[5.2 展望 44](#_Toc515028047)

[致 谢 45](#_Toc515028048)

[参考文献 46](#_Toc515028049)

摘 要

一直以来各大比赛都会存在冒名顶替的不良现象，这严重影响到了社会的公平和公正。严查此现象在过去会耗费大量的人力、物力和财力，但如今随着计算机技术的日趋进步，人工智能领域的人脸检测技术使杜绝这一现象的发生成为可能。作为图像内容智能化分析关键技术之一的检测算法近年来在深度学习卷积神经网络的推动作用下发展迅速，取得了各种有价值的成果。

目前在准确率、鲁棒性和实时性这三方面均有不错表现的目标检测方法是快速区域卷积神经网络Faster R-CNN，它可以作为一种检测人脸的特定类别算法。该方法首先使用卷积神经网络提取原始输入图像的特征，然后在特征图上获取可能包含目标的候选区域，最后对这些区域进行分类和回归，实现所有计算没有重复的端到端学习。本课题针对ACM国际大学生程序设计竞赛赛事的举办，设计了一款基于该竞赛的证照系统，这个系统在Django框架开发的网页平台上嵌入了人脸检测模型，能够识别出参赛队员上传的证照是否符合要求，并将符合要求的证照作为标准照为后续比赛过程中的实时人脸识别提供参照像。

整个系统的功能实现都是在Ubuntu16.04操作系统上进行的：首先在完成Faster R-CNN配置编译、按照公开数据集Pascal VOC2007制作部分数据的基础上运用Caffe提供的底层代码内核框架，训练了一种基于目标检测算法Faster R-CNN的二分类人脸检测模型并调用OpenCV人脸特征数据进行对比实验验证本课题训练模型的有效性，然后在开源Web框架Django下初步完成参赛图像的上传和展示功能，最后通过调用临时写入本地的参赛图像将人脸检测模型接入Django框架的逻辑处理层中，完成程序设计竞赛证照系统的实际应用。

**关键词：**人脸检测；证照系统；卷积神经网络；Faster R-CNN；Django框架

ABSTRACT

There has been a bad phenomenon of impostor in all major competitions from ancient times to the present, which seriously affects the fairness and impartiality of the society and will take a lot of manpower, material and financial resources to work it out in the past. But now with the increasing progress of computer technology, face detection technology in the field of artificial intelligence computer vision has made it possible to prevent the occurrence of the phenomenon. In recent years, as one of the key technologies of intelligent analysis of image content, the detection algorithm has developed rapidly and achieved all kinds of valuable results under the impetus of deep learning convolution neural network.

At present, the face detection method with good accuracy, robustness and real time in three aspects is fast region convolution neural network Faster R-CNN, which can be used as a specific class algorithm for face detection. Firstly, a CNN is used to extract features on original input image, then Faster R-CNN extracts region proposals from feature map that are likely to contain objects, finally, each region is classified and regressive, which makes the whole object detection pipeline become an end-to-end process with not repeated calculations. This paper designs a license system for the ACM International Collegiate Programming Contest, which embeds a face detection model on the web platform developed by the Django framework. The system can not only identify whether the pictures uploaded by competitors are satisfied with the requirements, but also can take them as standard pictures, which provides reference for real-time face recognition in subsequent competitions.

The whole system is implemented on the Ubuntu16.04 operating system. First of all, I have focused on the configuration and compilation environment of Faster R-CNN and data making method of open data set Pascal VOC2007. After that, I trained a two-category face detection model of Faster R-CNN on the basis of Caffe providing the underlying code kernel framework and called the OpenCV face feature data to conduct comparative experiments to verify the validity of the Faster R-CNN training model. What’s more, I initially completed the uploading and displaying functions of the participating images under the open source Web framework Django. Finally, I applied the model to the logical processing layer under the Django framework through temporarily calling the image of the competition, which made it possible to complete the practical application of the license system for program design competition.

**Keywords:** Face Detection; License System; Convolutional Neural Network; Faster R-CNN; Django Framework

# 绪论

## 课题背景及意义

### 课题背景

ACM国际大学生程序设计竞赛是由国际计算机协会主办的在国际上公认水平最高、规模最大、影响最深的计算机专业竞赛[1] 。这项竞赛至今已有近40年的发展历史，它不仅可以培养在校学生的编程能力、团队精神和创新思维，而且可以作为权威的参考依据提供给企业选拔人才。随着社会用人机制的不断完善，许多学生由于就业竞争的压力和荣誉利益的驱使，试图以冒名顶替的方式参加比赛获取不劳而获的成果，这严重影响到了社会的发展。因此，竞争比赛的公平公正性越来越受到主办单位的重视。

本课题旨在解决以上问题，利用人脸检测技术设计了有效防止部分赛站冒名参赛现象的程序设计竞赛证照系统，该系统能够识别出上传的证照是否符合要求并且作为标准照为后续比赛过程中的实时人脸识别提供参照图像，同时也可以打印到后续参赛队员的获奖证书上。

### 课题意义

此系统相比于普通证照系统最大的不同是运用了人工智能计算机视觉领域的人脸检测技术，它可以灵活地迁移到各种防作弊系统中很好地对图像人脸内容进行检测。人脸检测是根据一定的机器学习算法提取给定图像中的特征来定位人脸区域的过程，它作为程序设计竞赛证照系统的核心技术，采用的快速区域卷积神经网络方法不仅可以获得人脸检测模型而且可以训练出各种不同目标的检测模型，扩大了系统应用的广泛性和适应性。

如今，大量研究机构在人脸检测的研究方面取得了很大的进展，人脸检测作为计算机视觉的一个相对独立的研究领域，除了应用于人脸识别，在基于内容检索、图像处理、数码相机、视频会议、智能人机交互[2] 等领域有着重要的理论意义和应用价值。

## 课题研究现状

人脸检测起初产生于人脸识别系统，但不同于人脸识别这个概念：人脸检测是在一张图像中判断是否有人脸，如果有就框出人脸所在的区域；而人脸识别是给定一张未知的脸，根据一定的算法将这张脸和数据库中某个已知的脸对应起来并从中提取这张未知脸对应的身份信息。人脸检测是人脸识别的前提条件和必要环节[3] [4] ，也就是说首先通过人脸检测框选出图像中的人脸，然后采用人脸识别算法判断被框选的人脸是谁。人脸检测所获取的精度和速度不仅决定了人脸识别系统的性能，而且在人脸验证、人脸检索等方面也有重要的应用价值。其中人脸验证是“1对1”的，判断两张未知图像的相似程度，如果相似程度超过阈值，则表示这两张未知图像中出现的是同一个人；人脸检索是指从图片库里找出给定图像中包含的所有人脸，是“多对多”的。

### 国外研究现状

人脸检测从起源至今历经了早期算法、AdaBoost框架和深度学习[5] 这三个阶段：早期算法阶段最具代表性的是利用固定人脸图像匹配输入测试图像的人脸检测方法；AdaBoost框架阶段最具代表性的是若干个弱分类器组成强分类器的集成学习算法；深度学习阶段最具代表性的是基于卷积神经网络的优化算法。

人脸检测最早开始于上世纪60年代末，成为许多先进国家的研究热点，主要以美国、日本和欧洲等发达国家为代表。比如美国在1998年实施的“人工智能实验室”计划在各个大学设立AI实验室对人脸检测进行了详细的研究；日本在偶像组合“岚”举办的演出上采取刷脸入场措施，有效防止了黄牛的炒票行为；芬兰在2013年推出了史上第一款应用人脸检测技术的“最快速支付系统”，实现了用刷脸替代现金、银行卡等传统支付手段的快捷支付。

人脸检测发展迅速，在国外取得了重大的突破：谷歌于2014年凭借深度神经网络模型结构GoogleNet[6] 获得了基于ImageNet[7] 数据集的ILSVRC[8] 冠军、研发了不需要操作者就能自动启动、行驶和停下来的驾驶汽车、开发了刷爆朋友圈的围棋智能程序AlphaGo；微软开发了根据用户上传的照片就能分析出其性别和面部年龄的人脸系统、研发了赢得ImageNet 2015计算机识别挑战赛中图像定位、图像分类和图像检测这三个项目冠军的深度神经网络结构模型[9] [10] 、推出了使用者仅仅只需要刷一下脸就能实时被正在运行的Windows10设备识别的“Windows Hello”；苹果推出了能够利用人脸检测算法将被锁屏的手机打开并且定制3D表情的手机iPhoneX；美国联邦调查局构建出了方便办案人员在损失最小的情况下以最快速度抓捕到犯人的人脸数据库；日本丰桥科学技术大学开发了一种通过观察用户的行动就能做出响应的机器人；澳大利亚边境保护局联合政府在2017年推出的“游客刷脸入境”政策使得游客再也不必携带随时都有可能丢失的护照并且不用排着长长的队伍入境澳大利亚，只需要前期在手机上提交面部信息，入境之前刷一下脸就能轻松通关。

随着人工智能的飞速发展，很多异想天开的事情已经不再是遥不可及了，国外人脸检测技术尤其是美国目前已经发展到相当不错的阶段了，他们对于计算机视觉领域的贡献推动着时代朝着智能化、便捷化的方向前进。

### 国内研究现状

与其他国家相比，我国在人脸检测领域的科学钻研起步较晚，但发展迅速，也取得了一定的研究成果。这些成果分为学术研究和商业用途这两个方面，国内清华大学的苏光大教授、南京大学的周志华教授、香港中文大学的汤晓鸥教授在计算机视觉人脸检测领域的学术研究起到了关键性的作用，具有一定的代表性，其中公安部门利用苏光大教授研究成功的人脸识别系统侦破了许多重要的公安刑事案件，这不仅为国家的安全做出了贡献而且保障了人民的生命安全；周志华教授在人工智能机器学习领域贡献了许多著名作品，并且成为了获得亚太数据挖掘大会卓越贡献奖有史以来最年轻的一位获奖者；汤晓鸥教授保持了人脸识别学术界比赛的高记录，其人脸识别算法在原创的基础上首次超越人眼对外物的识别极限。

我国人脸检测研究的商业用途主要来源于百度、阿里巴巴、腾讯等著名互联网企业。百度在国内相对较早开展人脸检测研究，其网页“百度开发者中心”能够以很快的速度检测到人脸属性并在许多场景中提供很高精度的人脸识别服务；其产品“百度识图”能够根据使用者上传的图片判断人脸情况从而搜索出相似的图片以及在整个网中搜寻出曾经出现的相似人像。阿里巴巴旗下的支付宝在使用传统密码登陆、支付的情况下增加了刷脸的功能，其通过云峰基金控股的依图科技可以在低于1秒时间内迅速从超过几亿张图像的数据库中查询并返回精确结果，还可以给公安部门提供车辆牌照查询的系统。腾讯优图在被认为不仅难度最高而且数据量很大的数据库识别中拔得头筹，刷新了世界纪录。另外，北京旷视科技有限公司推出的Face++[11] 平台包含了人脸检测、特征点检测、人脸识别、人脸分析等一系列技术功能，其中人脸分析技术能够准确地分析出人的性别、面部年龄、肤色以及种族等信息，并且其嵌入进的美颜相机能够很好地分析出用户需要美化的区域从而进行美化处理。小米的人脸检测“黑科技”也跟上了人工智能的大潮，在腾讯优图、Face++之后取得了FDDB数据集上人脸识别的第一名。更加令人震惊的是作为中国科学院唯一人脸识别代表的云从科技[12] 在人脸检测的应用方面做出了很大的贡献：为中国大部分银行提供了远程人脸认证等缩短自助业务办理时间的终端柜台解决方案系统、为车辆服务行业提供了刷脸切换用户更改驾车习惯的车辆人脸识别系统、为手机电脑等电子企业提供了避免不法分子利用面具、视频等手段进行欺诈行骗的人脸检测技术、为公安部门提供了出入境不用大量监控视频进行人工排查的动态人脸布控体系。

虽然国内人脸检测的研究和应用都晚于美国等发达国家，但是随着国家重点推进的涌动资金流入，各大学校研究所和中小企业纷纷加入到发展前景乐观和生活迫切需求的人工智能行列中来，并且到目前为止，他们都取得了或多或少的成就，甚至在某些方面超越了国外人脸检测的成果。

## 课题应用前景

随着市场需求的不断加速，人脸检测的应用场景势必将被不断挖掘，目前已经在安防领域和商业领域取得了举足轻重的作用。

在安防领域，人脸检测应用场景光明，不管是机场还是火车站都离不开这项技术，因为机器只需要乘客手动将二代身份证放在指定区域就能快速地判定是否通行，这不仅减少了车站的拥堵状况而且杜绝了冒用他人身份证的情况发生。人脸检测技术目前虽然还没完全普及应用到人类生活的方方面面，但不可置疑的是，它已经对我们的生活产生了翻天覆地的影响，我相信随着社会的不断发展，刷脸验票技术未来将会广泛地普及到更多的应用场景：汽车站、游玩景区、电影院、银行、企业等。

在商业领域，各大APP通过人脸检测技术实现刷脸登陆和支付功能，各种美颜相机根据人脸检测技术定位人脸进行美化，我相信在不久的将来一定是百花齐放的场景：人力资源面试官利用人脸检测技术快速实现求职者基本信息的查询、广告商根据人们在广告牌前停留的时间和面部表情进行广告的合理投放、会议通过视觉机器人全自动聚焦于演讲者的展示、用户利用商家的无人机进行实时现场刷脸签收快递。

人脸检测技术正以很快的速度打开市场、加速普及，具有重要的科研价值和广泛的应用前景：应用软件的安全认证、机场车站的安检通道、恐怖主义的安全防备、走失儿童的有效寻觅、人脸表情的系统识别、基于内容的系统检索以及三维人脸的模型研究，可以说它涉及应用到了公共安全、社会执法、移动互联和娱乐领域等各个方面。它的出现因其人工智能的特质引起了国内外的广泛关注，其研究近年发展迅速，在学术界和工业界取得了巨大的成果，虽然人脸检测技术目前还面临着一些技术上难以攻克的问题，但足以向应用化的方向发展。因此，在不断更新换代的今天，磨练技术才会趋于实用，正视挑战，人脸检测技术将会大放异彩，取得新一轮的发展。

## 论文主要研究内容

本文研究分析了程序设计竞赛证照系统的人脸检测和网页开发这两个部分，核心是人脸检测部分。

全文共分五章，安排如下：

第一章：绪论。毕业设计课题来源、背景及研究意义，人脸检测在国内外的研究现状和应用前景，此次课题研究内容的总结。

第二章：程序设计竞赛证照系统人脸检测的方法。首先分析了人脸检测涉及到的反向传播、卷积神经网络等基础理论，然后介绍了目标检测方法R-CNN[13] 和Fast R-CNN[14] 并重点分析了Faster R-CNN[15] 和它的RPN[16] 层，最后在Caffe[17] 和TensorFlow[18] 框架的比较下选择Caffe作为人脸检测模型训练的框架。

第三章：程序设计竞赛证照系统人脸检测的实现。通过Faster R-CNN和OpenCV[19] 这两种方法实现。Faster R-CNN实现人脸检测需要在环境搭建、数据预处理、修改参数等一系列操作之后训练并测试模型；OpenCV实现人脸检测只需要使用官方已经训练好的特征数据模型进行测试。对两者作对比实验并进行分析，验证Faster R-CNN模型的有效性。

第四章：程序设计竞赛证照系统网页开发的实现。首先介绍了Django[20] 的定义、发展及其特点，然后分析了它的MTV[21] 框架构造、运行机制和优势特点，之后详细描述了项目的环境配置和实现流程，并指出了满足任务要求的具体实现方法，最后交代了人脸检测部分与网页部分的接口。

第五章：总结与展望。对毕业设计中完成的工作进行了总结，并对以后要做的事情进行了展望。

# 神经网络

针对程序设计竞赛证照系统人脸检测这个部分，本章给出了逻辑回归（Logistic regression）,深度神经网络的基本推导过程、各种激活函数的介绍，以及一个可以进行猫与非猫识别的深度神经网络实现代码。

## 监督学习和神经网络

目前为止，由神经网络模型创造的价值基本上都是基于监督式学习（Supervised Learning）的。监督式学习与非监督式学习本质区别就是是否已知训练样本的输出y。在实际应用中，机器学习解决的大部分问题都属于监督式学习，神经网络模型也大都属于监督式学习。

数据类型一般分为两种：结构化数据和非结构化数据。结构化数据通常指的是有实际意义的数据。例如某一商品房的中的大小，价格，卧室数量等。这些数据都具有实际的物理意义，比较容易理解。而非结构化数据通常指的是比较抽象的数据，例如音频，图像或者文字段落。以前，计算机对于非结构化数据比较难以处理，而人类对非结构化数据却能够处理的比较好，例如我们第一眼很容易就识别出一张图片里是否有猫，但对于计算机来说并不那么简单。现在，值得庆幸的是，由于深度学习和神经网络的发展，计算机在处理非结构化数据方面效果越来越好，甚至在某些方面优于人类。总的来说，神经网络与深度学习无论对结构化数据还是非结构化数据都能处理得越来越好，并逐渐创造出巨大的实用价值。

## 为什么现在深度学习的发展日新月异

深度学习和神经网络背后的技术思想已经出现数十年了，那么为什么直到现在才开始发挥作用呢？

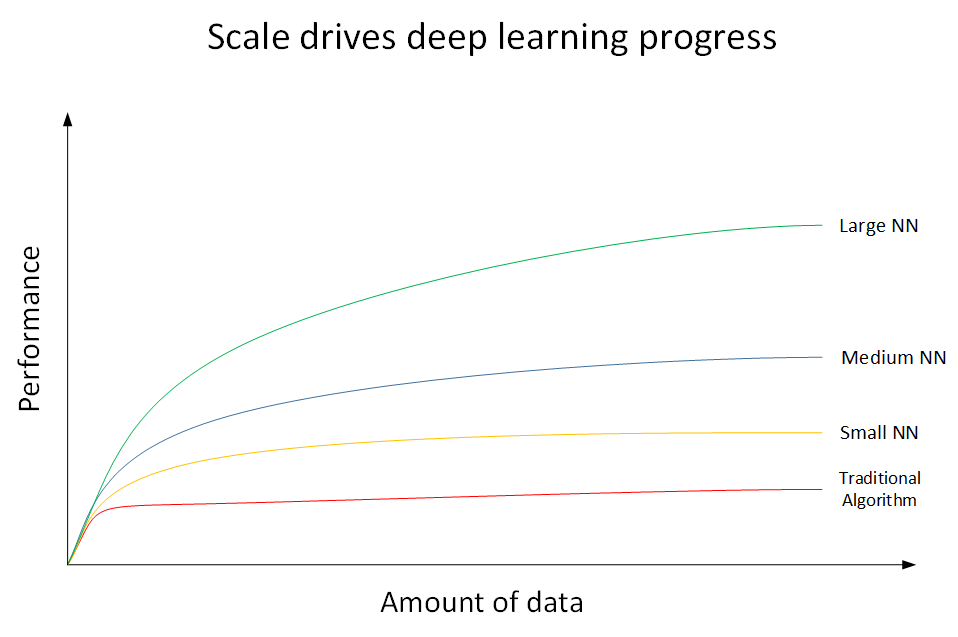


图2- 2 模型效果比较图

上图共有4条曲线。红色曲线代表了传统机器学习算法的表现，例如是支持向量机，逻辑回归，决策树等。当数据量比较小的时候，传统学习模型的表现是比较好的。但是当数据量很大的时候，性能基本不会变得更好。黄色曲线代表了规模较小的神经网络模型（Small NN）。它在数据量较大时候的性能优于传统的机器学习算法。蓝色曲线代表了规模中等的神经网络模型（Media NN），它在在数据量更大的时候的表现比Small NN更好。绿色曲线代表更大规模的神经网络（Large NN），即深度学习模型。在数据量很大的时候，深度学习模型的表现仍然是最好的，而且基本上保持了较快上升的趋势。值得一提的是，近些年来，由于数字计算机的普及，人类进入了大数据时代，每时每分，互联网上的数据是海量的、庞大的。如何对大数据建立稳健准确的学习模型变得尤为重要。传统机器学习算法在数据量较大的时候，性能一般，很难再有提升。然而，深度学习模型由于网络复杂，对大数据的处理和分析非常有效。所以，近些年来，在处理海量数据和建立复杂准确的学习模型方面，深度学习有着非常不错的表现。然而，在数据量不大的时候，例如上图中左边区域，深度学习模型不一定优于传统机器学习算法，性能差异可能并不大。

所以说，现在深度学习如此强大的原因归结为三个因素：Data，Computation，Algorithms。

其中，数据量的几何级数增加，加上GPU算力突飞猛进、计算机运算能力的大大提升，使得深度学习能够应用得更加广泛。另外，算法上的创新和改进让深度学习的性能和速度也大大提升。举个算法改进的例子，之前神经网络神经元的激活函数是Sigmoid函数，后来改成了ReLU函数，在实际应用中采用ReLU函数确实要比Sigmoid函数快很多。

## 逻辑回归

逻辑回归模型一般用来解决二分类，二分类就是输出y只有{0,1}两个离散值（也有{-1,1}的情况）。我们以一个图像识别问题为例，判断图片中是否有猫存在，0代表没有猫，1代表有猫。

上图所示，cat图片的尺寸为。将图片转化为一维的特征向量。则转化后的输入特征向量维度为。此特征向量是列向量，维度一般记为。训练样本共有张图片，那么整个训练样本x组成了矩阵，维度是。

在逻辑回归中，我们会引入参数和。权重的维度是，b是一个常数项（在python程序中，矩阵减去一个常数相当于所有矩阵元素减去那个常数）。逻辑回归中有线性变换和非线性变换，线性变换为：

线性输出区间为整个实数范围（-∞，+∞），而逻辑回归要求输出范围在[0,1]之间，可以引入非线性函数Sigmoid，此变换矩阵维度不变。

接下来，我们引入损失函数（），它的目的就是要衡量预测一个样本输出与其真实结果的接近程度。因为一个样本的输出y是一个自然数，损失函数计算结果也是一个自然数，

这个Loss Function是针对单个样本的。但是我们在训练时，不会一次只训练一个样本，一般我们把m个样本的输入矩阵叠加到一个矩阵x中，对于个样本，可以定义代价函数（），这里使用右上标表示某个样本，输出是一个自然数：

代价函数用来评价模型的好坏，其值越小说明模型和当前参数可以更好地预测。代价函数目标是计算出使用当前和b时，个训练样本的输出与真实值相差多大。我们的最小化，让尽可能地接近于零。怎么最小化，我们知道一个函数在一个点的导数代表着函数在当前点上函数值上升或者下降的快慢，而接下来的梯度下降就是不断使用的导数来最小化。

## 梯度下降

梯度下降算法是先随机选择一组参数w和b值，然后每次迭代的过程中分别沿着w和b的梯度（偏导数）的反方向前进一小步，不断修正w和b。每次迭代更新w和b后，都能让更接近全局最小值。w和b的修正表达式为：

接下来梯度下降推导使用dw代替损失函数对w的偏导数，并且dw矩阵维度和w矩阵维度相同，以此类推。

### 1个样本的梯度下降

对于1样本，需要求损失函数（Loss Function）关于w和b的偏导数： 我们使用表示损失函数对的偏导数，以此类推：

可以看到，损失函数对线性变换的结果z的偏导数是预测结果减去真实值，知道了，可以很容易对w和b求导：

梯度下降就是w和b减去学习率和w与b的偏导数，从而让损失函数计算出来的值变小，让预测值接近于真实值：

### m个样本的梯度下降

因为图片输入是m个样本一起输入，对于m个样本，需要对每一个样本求损失函数（Loss Function），我们使用右上标并加括号表示某一个样本，表示第i个样本的输入数据，以此类推：

关于w和b的偏导数并求平均：

因为b是一个自然数，所以db也要是一个数字，这里把维度为(1,m)的dz第一行的元素相加求平均得到db。而w维度是，需要xd。

经过一次迭代后，梯度下降，对w和b进行更新：

这样经过n次迭代，整个深度学习算法中学习部分就完成了，深度学习就是通过梯度下降不断对网络中参数更新，从而使代价函数计算出的数值不断降低让预测值接近于真实值。

## 激活函数

神经网络的隐藏层和输出层都需要激活函数，不同激活函数有各自用处。

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **函数** | **图像** | **导数** |
| Sigmoid |  |  |  |
| Tanh |  |  |  |
| ReLU |  |  |  |
| Leaky ReLU |  |  |  |

在实际使用中，sigmoid函数使用在二分类的输出层，sigmod函数和tanh函数的缺陷是当z很大或者很小的时候，激活函数的斜率很小，因此，梯度下降算法会运行的比较慢。同时，由于需要指数运算，计算比Relu更费时间。

ReLU避免了斜率很小的这个缺陷，在2010年左右很多深度学习研究者还认为sigmoid函数比ReLU效果更好，但是随着近几年计算速度（主要是GPU的计算速度）的提高，研究者在进行了大量数据的训练后发现神经网络使用ReLU，计算速度更快，效果使用sigmoid相似。对于隐藏层，选择ReLU作为激活函数能够保证z大于零时梯度始终为1，从而提高神经网络梯度下降算法运算速度。当z小于零，存在梯度为0的缺点，但在实际应用中，这个缺点影响不大。出现了Leaky ReLU函数，能够保证z小于零是梯度不为0。

实际应用中，隐藏层通常会使用ReLU函数或者Leaky ReLU函数，其实，具体选择哪个函数作为激活函数没有一个固定的准确的答案，应该要根据具体实际问题进行验证。

## 神经网络

逻辑回归可以理解为一个神经元，神经网络包含多个神经元。如下图所示，分别列举了逻辑回归、1个隐藏层的神经网络、2个隐藏层的神经网络和5个隐藏层的神经网络它们的模型结构。深层神经网络其实就是包含更多的隐藏层浅层神经网络。

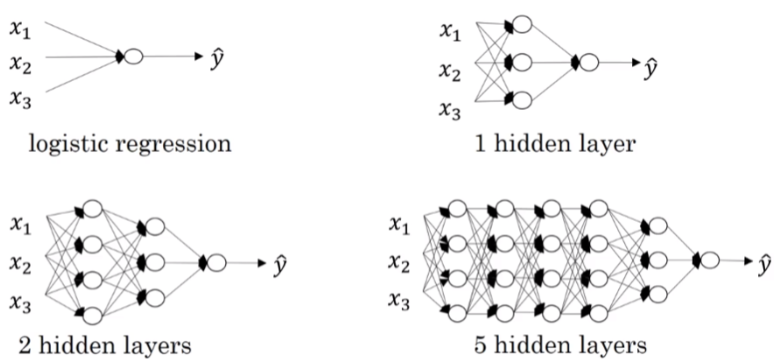


图2- 3 逻辑回归和神经网络

神经网络的层数只参考隐藏层个数和输出层，图2-3中，逻辑回归又叫1层神经网络，10层隐藏层，2层输出层的神经网络叫做12层神经网络，以此类推。总层数用L表示，输入层是第0层，输出层是第L层。神经网络的形状使用一个数组n来表示， 表示第三层有5个神经元。对于神经网络，用大写和来表示第l层参数，第0层（输入层）没有参数W和b。对于m个训练样本，各层参数维度：

输入

参数

参数

输出

在Python程序中，由于Numpy的广播性质，维度会被当成矩阵进行运算，和的维度和和维度相同。参数W不需要想逻辑回归中的w转秩。

### 神经网络中向前传播和向后传播

神经网络中，使用表示第l层线性变换结果，表示第l层激活函数输出结果。向前传播中第l层输入是第l-1层的输出，利用输入的计算出和。向后传播中第l层输入是第l+1层的输出，利用输入的计算出和。

向前传播比较简单，最后一层激活函数使用sigmoid函数，其他隐藏层都是用ReLU函数：

反向传播如图2-4所示利用上一层算出的， 先计算这一层的：

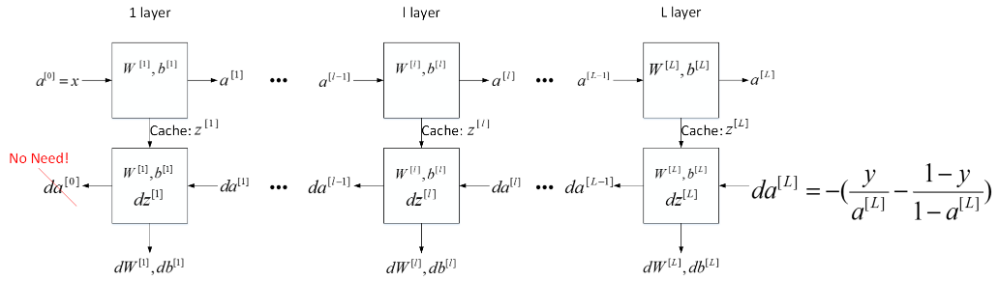


图2- 4 向前传播和向后传播

在使用，计算出，，，不断向后计算。完整代码如下：

， 损失函数的导数

，最后一层损失函数对b的偏导

## 深度学习框架

使用开源框架，自己可以不必实现神经网络的每一个细节，只需要著重于网路的设计和训练。使用Keras，一层神经网络或者所有的反向传播只需要一行代码。训练速度可能比自己实现的更快，因为框架底层都是使用C/C++来编写的，并不是Python，这自然比纯Python的神经网络运行速度快。框架的价值也可以从耗费资金来看，Tensorflow是Google开发的，应该有超过一百个软件工程师和算法工程师从事其的开发工作，Google软件工程师薪水普遍在数十万美元一年，也就是说这几年Google投入到Tensorflow上的钱就要用亿美元来计算，耗资之大从侧面表现了这些开源框架的强大。

### Tensorflow

2015年11月10日，Google宣布推出全新的机器学习开源工具TensorFlow。 TensorFlow 最初是由 Google 机器智能研究部门的 Google Brain 团队开发，基于Google 2011年开发的深度学习基础架构DistBelief构建起来的。Tensorflow是现在最流行的深度学习框架，它被大部分公司使用，获得了极大的成功，但它也有缺点，如下：

过于复杂的系统设计，Tensorflow在GitHub代码仓库的代码量超过100万行。

接口设计过于晦涩难懂。在设计TensorFlow时，创造了图、会话、命名空间、PlaceHolder等诸多抽象概念，对普通用户来说难以理解。

直接使用Tensorflow生产力低下，Google官方和其他开发者尝试构建更容易使用的接口，包括Keras、Sonnet、TFLearn、TensorLayer、Slim、Fold、PrettyLayer等数不胜数的第三方框架每隔几个月就会在新闻中出现一次。

凭借着Google强大的推广能力，TensorFlow已经成为当今最炙手可热的深度学习框架。但在2018年和2019年上半年，Facebook开源的PyTorch大受欢迎，渐渐成为和Tensorflow一样流行、成熟的深度学习框架。

### Pytorch

2017年1月，Facebook人工智能研究院团队在GitHub上开源了PyTorch，PyTorch随之成为Tensorflow的主要竞争对手。PyTorch的设计追求最少的封装，尽量避免重复造轮子。PyTorch的设计遵循tensor→variable(autograd)→nn.Module 三个由低到高的抽象层次，分别代表高维数组（张量）、自动求导（变量）和神经网络（层/模块），而且这三个抽象之间联系紧密，可以同时进行修改和操作。PyTorch是当前难得的简洁优雅且高效快速的框架。它的源码只有TensorFlow的十分之一左右，更少的抽象、更直观的设计使得PyTorch的源码十分易于阅读。它的灵活性不以速度为代价，其速度与Tensorflow相差不大。

### Keras

Keras应该是深度学习框架之中最容易上手的一个。Keras并不能称为一个深度学习框架，它更像一个深度学习接口，它构建于第三方框架之上。学习Keras十分容易，搭建一层神经网络只需要一行代码。但是在使用Keras的大多数时间里，用户主要是在调用接口，很难真正学习到深度学习的内容。

但是对于已经熟悉深度学习基本理论知识的人，Keras是开发速度最快的框架。Keras是Tensorflow的一部分，安装好Tensorflow后，就可以使用Keras了。



# 神经网络的优化

## 训练集/验证集/测试集

一般地，我们将所有的样本数据分成三个部分：Train/Dev/Test sets。Train sets用来训练你的算法模型；Dev sets用来验证不同算法的表现情况，从中选择最好的算法模型；Test sets用来测试最好算法的实际表现。之前人们通常设置Train sets和Test sets的数量比例为70%和30%。如果有Dev sets，则设置比例为60%、20%、20%，分别对应Train/Dev/Test sets。这种比例分配在样本数量不是很大的情况下，例如100,1000,10000，是比较科学的。但是如果数据量很大的时候，例如100万，这种比例分配就不太合适了。科学的做法是要将Dev sets和Test sets的比例设置得很低。因为Dev sets的目标是用来比较验证不同算法的优劣，从而选择更好的算法模型就行了。因此，通常不需要所有样本的20%这么多的数据来进行验证。对于100万的样本，往往只需要10000个样本来做验证就够了。对于大数据样本，Train/Dev/Test sets的比例通常可以设置为98%/1%/1%，或者99%/0.5%/0.5%。

## 过拟合和欠拟合

数据和模型的拟合情况是很重要的一个问题。判断欠拟合和过拟合主要通过训练样本和测试样本的损失或者正确率。减少欠拟合的主要方法是增加神经网络的复杂程度，训练时间延长，而减少过拟合主要有降低神经网络复杂度、正则化(Regularization)、Dropout，增加训练样本数。

|  |  |  |
| --- | --- | --- |
|  | 图例 | 训练时误差可能的情况 |
| 欠拟合 |  | Train set error: 15%  Dev set error: 16% |
| 拟合 |  | Train set error: 15%  Dev set error: 10% |
| 过拟合 |  | Train set error: 1%  Dev set error: 11% |

### 正则化

如果出现了过拟合，可以使用正则化(Regularization)来解决。虽软扩大训练样本数量也是一种方法，但是通常获得更多的训练样本比较困难。

数据和模型的拟合情况是很重要的一个问题。判断欠拟合和过拟合主要通过训练样本和测试样本的损失或者正确率。减少欠拟合的主要方法是增加神经网络的复杂程度，训练时间延长，而减少过拟合主要有降低神经网络复杂度、正则化Regularization、Dropout，增加训练样本数。对成本函数(Cost function)加入L2正则化：

是计算所有元素平方和再开放。由于加入了正则化项，梯度下降算法中的计算表达式需要做如下修改：

L2 Regularization也被称作weight decay。这是因为，由于加上了正则项，有个增量，在更新的时候，会多减去这个增量，使得比没有正则项的值要小一些。不断迭代更新，不断地减小。近似为零，意味着该神经网络模型中的某些神经元实际的作用很小，可以忽略。从效果上来看，其实是将某些神经元给忽略掉了。这样原本过于复杂的神经网络模型就变得不那么复杂了，而变得非常简单化了。这样过拟合的情况就被减少了。

### Dropout

除了L2 Regularization，Dropout也可以防止过拟合。Dropout是指在深度学习网络的训练过程中，对于每层的神经元，按照一定的概率将其暂时从网络中丢弃。也就是说，每次训练时，每一层都有部分神经元不工作，起到简化复杂网络模型的效果，从而避免发生过拟合。Dropout有不同的实现方法，接下来介绍一种常用的方法：Inverted dropout。假设对于第l层神经元，设定保留神经元比例概率keep\_prob=0.8，即该层有20%的神经元停止工作。

对于m个样本，单次迭代训练时，随机删除掉隐藏层一定数量的神经元；然后，在删除后的剩下的神经元上正向和反向更新权重w和常数项b；接着，下一次迭代中，再恢复之前删除的神经元，重新随机删除一定数量的神经元，进行正向和反向更新w和b。不断重复上述过程，直至迭代训练完成。从效果上来说，与L2 regularization是类似的，都是对权重w进行“惩罚”，减小了w的值。

值得注意的是，使用dropout训练结束后，在测试和实际应用模型时，不需要进行dropout和随机删减神经元，所有的神经元都在工作。

使用dropout的时候，可以通过绘制cost function来进行debug，看看dropout是否正确执行。一般做法是，将所有层的keep\_prob全设置为1，再绘制cost function，即涵盖所有神经元，看J是否单调下降。下一次迭代训练时，再将keep\_prob设置为其它值。

### 数据增加和早期停止

除了L2 regularization 和dropout regularization，还有其他减少过拟合的方法。

一种方法是增加训练样本数量。但是通常成本较高，难以获得额外的训练样本。但是，我们可以对已有的训练样本进行一些处理来“制造”出更多的样本，称为data augmentation。

还有另外一种防止过拟合的方法：early stopping。一个神经网络模型随着迭代训练次数增加，train set error一般是单调减小的，而dev set error 先减小，之后又增大。因此，迭代训练次数不是越多越好，可以通过train set error和dev set error随着迭代次数的变化趋势，选择合适的迭代次数，即early stopping。

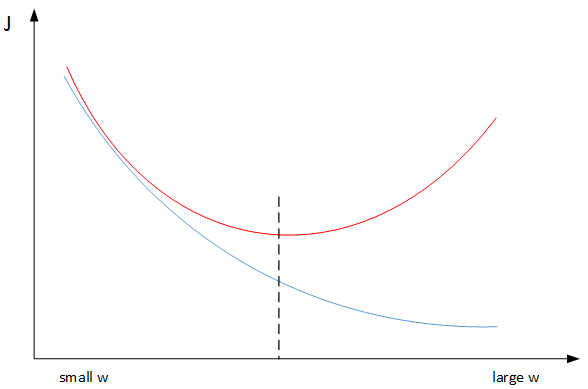


图3- 1 过拟合训练和验证集的error

然而，Early stopping有其自身缺点。通常来说，机器学习训练模型有两个目标：一是优化cost function，尽量减小J；二是防止过拟合。这两个目标彼此对立的，即减小J的同时可能会造成过拟合，反之亦然。我们把这二者之间的关系称为正交化orthogonalization。该节课开始部分就讲过，在深度学习中，我们可以同时减小Bias和Variance，构建最佳神经网络模型。但是，Early stopping的做法通过减少得带训练次数来防止过拟合，这样J就不会足够小。也就是说，early stopping将上述两个目标融合在一起，同时优化，但可能没有“分而治之”的效果好。

与early stopping相比，L2 regularization可以实现“分而治之”的效果：迭代训练足够多，减小J，而且也能有效防止过拟合。而L2 regularization的缺点之一是最优的正则化参数\lambdaλ的选择比较复杂。对这一点来说，early stopping比较简单。总的来说，L2 regularization更加常用一些。

## 梯度爆炸和梯度消失

在神经网络尤其是深度神经网络中存在可能存在这样一个问题：梯度消失和梯度爆炸。意思是当训练一个 层数非常多的神经网络时，计算得到的梯度可能非常小或非常大，甚至是指数级别的减小或增大。这样会让训练过程变得非常困难。

一般，在隐藏层，我们使用ReLU函数，ReLU函数在输入为正数部分输出为线性，即。那么我们忽略各层常数项b的影响，令b全部为零。这是前面推到出来的公式：

如果各层的元素都稍大于1，例如1.5，那么dZ就会很大。A是当前层的输入，上一层的输出，大数值的W也会让A变大，这就导致dW的变大，从而造成梯度爆炸。

也就是说，如果各层权重W都大于1或者都小于1，那么各层激活函数的输出将随着层数l的增加，呈指数型增大或减小。当层数很大时，出现数值爆炸或消失。同样，这种情况也会引起梯度呈现同样的指数型增大或减小的变化。L非常大时，例如L=150，则梯度会非常大或非常小，引起每次更新的步进长度过大或者过小，这让训练过程十分困难。

## 参数初始化

我们可以使用一些方法来避免梯度消失和梯度爆炸，方法是对权重w进行一些初始化处理。对于ReLU函数，将随机生成的W值乘以：，相应python伪代码：

## Mini-batch 梯度下降

前面神经网络训练过程是对所有m个样本，称为batch。如果m很大，例如达到百万数量级，训练速度往往会很慢，因为每次迭代都要对所有样本进行进行求和运算和矩阵运算。我们将这种梯度下降算法称为Batch Gradient Descent。为了解决这一问题，我们可以把m个训练样本分成若干个子集，称为mini-batches，这样每个子集包含的数据量就小了，然后每次在单一子集上进行神经网络训练，速度就会大大提高。这种梯度下降算法叫做Mini-batch Gradient Descent。

Mini-batches Gradient Descent的实现过程是先将总的训练样本分成T个子集（mini-batches），然后对每个mini-batch进行神经网络训练，包括Forward Propagation，Compute Cost Function，Backward Propagation，循环至T个mini-batch都训练完毕。

经过T次循环之后，所有m个训练样本都进行了梯度下降计算。这个过程，我们称之为经历了一个epoch。对于Batch Gradient Descent而言，一个epoch只进行一次梯度下降算法；而Mini-Batches Gradient Descent，一个epoch会进行T次梯度下降算法。

值得一提的是，对于Mini-Batches Gradient Descent，可以进行多次epoch训练。而且，每次epoch，最好是将总体训练数据重新打乱、重新分成T组mini-batches，这样有利于训练出最佳的神经网络模型。

一般来说，如果总体样本数量m不太大时，例如m<2000，建议直接使用Batch gradient descent。如果总体样本数量m很大时，建议将样本分成许多mini-batches。推荐常用的mini-batch size为64,128,256,512。这些都是2的幂。之所以这样设置的原因是计算机存储数据一般是2的幂，这样设置可以提高运算速度。

我们可以这样理解：对于训练样本数特别多的训练样本，可以一次使用一部分，并不需要一次就使用全部的训练数据，这样依然可以有很好的训练效果，关键是让训练变得可行。因为数据量特别大的化，可能训练花费很多资源，是不可行的。

## 梯度下降优化

### Momentum

动量梯度下降算法，其速度要比传统的梯度下降算法快很多。做法是在每次训练时，对梯度进行指数加权平均处理，然后用得到的梯度值更新权重W和常数项b。

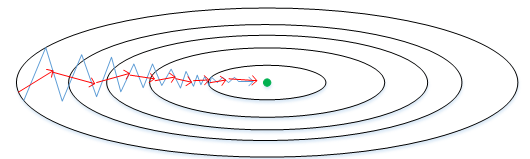


图3- 2 梯度下降过程图

原始的梯度下降算法如上图蓝色折线所示。在梯度下降过程中，梯度下降的振荡较大，尤其对于W、b之间数值范围差别较大的情况。此时每一点处的梯度只与当前方向有关，产生类似折线的效果，前进缓慢。而如果对梯度进行指数加权平均，这样使当前梯度不仅与当前方向有关，还与之前的方向有关，这样处理让梯度前进方向更加平滑，减少振荡，能够更快地到达最小值处。计算过程如下：

一般设置，即当前层占0.1的比例，其他层占0.9的比例。

### RmSprop

RMSprop是另外一种优化梯度下降速度的算法。每次迭代训练过程中，W和常数项b的更新表达式为：

为了避免RMSprop算法中分母为零，通常可以在分母增加一个极小的常数。

在图3-2中，为了便于分析，令水平方向为W的方向，垂直方向为b的方向。可以看到梯度下降在垂直方向上振荡较大，在水平方向上振荡较小，表示在b方向上梯度较大，即db较大，而在W方向上梯度较小，即dW较小。因此较小，较大。在更新W和b的表达式中，变化值较大，较小。也就使得W变化得多一些，b变化得少一些。即加快了W方向的速度，减小了b方向的速度，减小振荡，实现快速梯度下降算法，

### Adam

Adam（Adaptive Moment Estimation）算法结合了动量下降算法和RMSprop算法。每t次迭代训练过程中，W和常数项b的更新表达式为：

Adam算法包含了几个超参数，分别是。其中通常设置为0.9，通常设置为0.999，通常设置为。实际应用中，Adam算法是最常用的梯度下降优化算法，它结合了动量梯度下降和RMSprop各自的优点，使得神经网络训练速度大大提高。

## Batch Normalization

Sergey Ioffe和Christian Szegedy两位学者提出了Batch Normalization方法。Batch Normalization不仅可以让调试超参数更加简单，而且可以让神经网络模型更加“健壮”。也就是说较好模型可接受的超参数范围更大一些，包容性更强，使得更容易去训练一个深度神经网络。

在神经网络中，第l层隐藏层的输入是第l-1层隐藏层的输出,对进行标准化处理，可以提高W和b的训练速度和准确度。实际应用中，一般是对进行标准化处理而不是。对于第l层隐藏层的输入做如下标准化处理，i表示样本，忽略上标[l-1]：

是

# 卷积神经网络

# 循环神经网络

# 程序设计竞赛证照系统人脸检测的实现

本章给出了Faster R-CNN在Ubuntu16.04下的详细配置编译过程，并根据Pascal VOC2007数据集格式制作训练数据，在demo程序能够正常目标检测的基础上进行配置文件参数的修改，最终在Caffe框架下训练得到了基于Faster R-CNN方法的人脸检测模型并调用日志文件实现网络损失的可视化分析。为了验证训练模型的有效性，引入OpenCV已经训练好的人脸检测数据，对两者的检测结果进行对比分析，最终选择基于Faster R-CNN方法训练的人脸检测模型作为程序设计竞赛证照系统的人脸检测部分算法。

## Faster R-CNN人脸检测的实现

### 环境配置

环境配置主要涉及到Ubuntu16.04操作系统、显卡驱动、Anaconda2、OpenCV、CUDA、Caffe的安装以及Faster R-CNN的编译。

1. Ubuntu16.04操作系统的安装
2. 前期准备
3. Ubuntu16.04镜像文件：ubuntu-16.04.3-desktop-amd64.iso。
4. Ultraso软件：uiso9\_cn.exe。
5. 压缩卷。
6. 制作U盘启动盘。
7. 安装ubuntu16.04系统
8. 修改BIOS相关设置。
9. 选择启动器：保持 U 盘插入电脑的状态，重启电脑。
10. 分区界面
11. 按照表3- 1对Ubuntu16.04操作系统进行分区。
12. 使用默认安装引导。

表3- 1 Ubuntu安装分区表

|  |  |  |  |
| --- | --- | --- | --- |
| 挂载点 | 大小  （MB） | 类型  （分区） | 作用  （日志） |
| / | 20000 | 主 | EXT4 |
| 无 | 2048 | 逻辑 | 交换 |
| /boot | 500 | 逻辑 | EXT4 |
| /home | 剩余 | 逻辑 | EXT4 |

二、系统更新与显卡驱动安装

1. 系统更新
2. 设置→系统设置→软件和更新。
3. Ubuntu软件→souhu源（速度较快）。
4. 设置→关于这台计算机→安装更新系统。
5. 显卡驱动的安装
   1. 设置→系统设置→软件和更新。
   2. 附加驱动→NVIDIA专有驱动→应用更改。
6. Anaconda2的安装
7. 准备

Anaconda2安装脚本：Anaconda2-4.1.1-Linux-x86\_64.sh。

1. 安装Anaconda2
2. 安装：bash Anaconda2-4.1.1-Linux-x86\_64.sh
3. 激活：source ~/.bashrc
4. 测试：python

出现图3- 1的Anaconda，则安装成功。

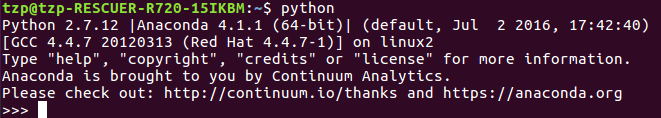


图3- 1 Ubuntu16.04下Anaconda2安装成功标志图

1. OpenCV3.0的安装
2. 解压并编译OpenCV3.0
3. 解压：unzip opencv-3.0.0-alpha.zip
4. 打开：cd opencv-3.0.0-alpha/
5. 编译：

cmake .

sudo make

sudo make install

1. 测试OpenCV3.0
2. 例子：cd opencv/samples
3. 编译：

cd samples

sudo cmake .

sudo make -j $(nproc)

1. 测试：

cd cpp/

./cpp-example-facedetect lena.jpg

检测到下图3- 2中的人脸框，则OpenCV3.0安装成功。

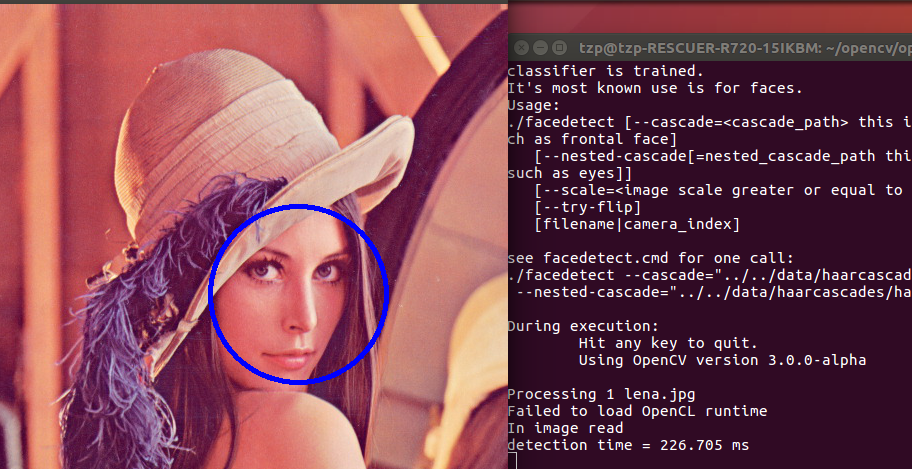


图3- 2 OpenCV3.0安装成功标志图

1. CUDA的安装

CUDA是一种通用并行计算架构，能够通过GPU快速解决繁琐复杂的计算项目，减少训练和测试时间。

1. 安装CUDA
2. 准备：cuda\_8.0.61\_375.26\_linux.run。
3. 安装：sudo sh cuda\_8.0.61\_375.26\_linux.run
4. 为了在命令行运行，添加环境变量：

sudo gedit ~/.bashrc

export PATH=/usr/local/cuda-8.0/bin${PATH:+:${PATH}}

1. 测试CUDA
2. 打开：cd /usr/local/cuda-8.0/samples/1\_Utilities/deviceQuery
3. 编译：sudo make
4. 执行：sudo ./deviceQuery

出现图3- 3的GPU信息，则Cuda安装成功。



图3- 3 Cuda安装成功标志图

1. 添加卷积神经网络加速库cuDNN
2. 下载并解压cudnn5.1.tgz
3. 配置相关文件并修改权限
4. Caffe的安装
5. 安装Caffe
6. 安装依赖包。
7. 下载：git clone git://github.com/BVLC/caffe.git
8. 修改配置文件。
9. 编译：

cd ~/caffe

make all

make test

make runtest

make pycaffe

1. 验证Caffe

python

import caffe

如果不提示错误，则安装成功。

1. Faster R-CNN的编译
2. 安装py-faster-rcnn
3. 下载：



1. 编译py-faster-rcnn：

cd /home/tzp/py-faster-rcnn/lib/

make

1. 编译caffe-fast-rcnn：

cd /home/tzp/py-faster-rcnn/caffe-fast-rcnn/

make

make pycaffe

1. 测试py-faster-rcnn

cd /home/tzp/py-faster-rcnn/

./tools/demo.py –gpu 0

如果检测到图片中的物体，则测试成功。



### 准备数据

根据Pascal VOC2007数据集的目录结构、数据类型来制作用于本实验的人脸图像训练数据。

1. 创建目录结构

在/py-faster-rcnn/data下按照公开数据集Pascal VOC2007创建如下目录结构：

VOCdevkit2007

VOC2007

ImageSets

Annotations

JPEGImages

Main

图3- 4 Pascal VOC2007数据集目录结构图

JPEGImages存放“.jpg”训练集，Annotations存放“xml”画框训练集，Main存放“txt”一定比例的训练集名称。

1. 数据预处理

选取FDDB数据集子集图片作为训练样本。

1. 重命名样本图片

格式：000001、000002、000003…000011…

1. 制作xml文件

利用图片标注工具LabelImg对样本图片中的人脸进行画框操作，画框完成后，将训练图片转化成xml文件存放到Annotations里。

1. 制作txt文件

制作test.txt(测试)、trainval.txt(训练+验证)、train.txt(训练)、val.txt(验证)这四个文件。

表3- 2 Pascal VOC标准数据集

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据集 | 训练数据  （张） | 验证数据  （张） | 测试数据  （张） | 类别  (个) | 图片尺寸  （长宽） | 平均物体数  （个/张） |
| VOC2007 | 2501 | 2510 | 4952 | 20 | 469387 | 2.473 |
| VOC2012 | 5717 | 2823 | 10991 | 20 | 469387 | 2.380 |

观察Pascal VOC标准数据集可得：test.txt和trainval.txt所含样本数几乎相等且各占总样本的百分之五十，train.txt和val.txt同理。

### 训练模型

1. 数据集

将公开数据集Pascal VOC2007保存至/py-faster-rcnn/data，使用预处理的人脸数据替代VOC2007中相应数据。

1. 下载ImageNet模型

表3- 3 ImageNet网络分类

|  |  |  |  |
| --- | --- | --- | --- |
| ImageNet |  | 类型 | 显存要求 |
| VGG16.v2.caffemodel |  | 大 | 3G |
| VGG\_CNN\_M\_1024.v2.caffemodel |  | 中 | 6G |
| ZF.v2.caffemodel |  | 小 | 11G |

本人电脑GTX1060，选择中型网络作为人脸检测的预训练模型，初始化网络参数。

1. 训练方式

表3- 4 Faster R-CNN训练方式

|  |  |  |
| --- | --- | --- |
| 训练方式 | 训练时间指标 | 准确率指标 |
| faster\_rcnn\_alt\_opt（交替训练） | 长 | 较高 |
| faster\_rcnn\_end2end（联合训练） | 短 | 较高 |

选择训练时间相对较短的联合训练方式。

1. 修改配置文件参数
2. 修改分类标签：Faster R-CNN物体检测20类前景+1类背景，本课题人脸检测1类前景+1类背景。删除20类前景，添加标签“face”。 

图3- 5 pascal\_voc.py分类标签

1. 修改网络模型描述文件train.prototxt。
2. 修改训练迭代次数。
3. 开始训练：

./experiments/scripts/faster\_rcnn\_end2 end.sh 0 VGG\_CNN\_M\_1024 pascal \_voc

训练后的模型会保存在output中，下图3- 6分别是每迭代10000次保存的模型，迭代次数可以在超参数文件solver.prototxt中进行修改。

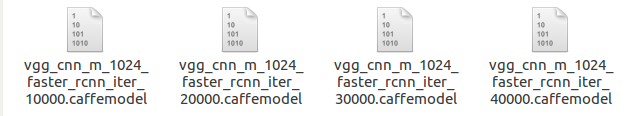


图3- 6训练产生的模型

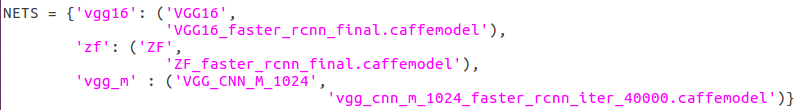
### 测试模型

1. 修改配置文件参数
2. 修改网络模型描述文件test.prototxt
3. 修改py-faster-rcnn/tools下的demo.py

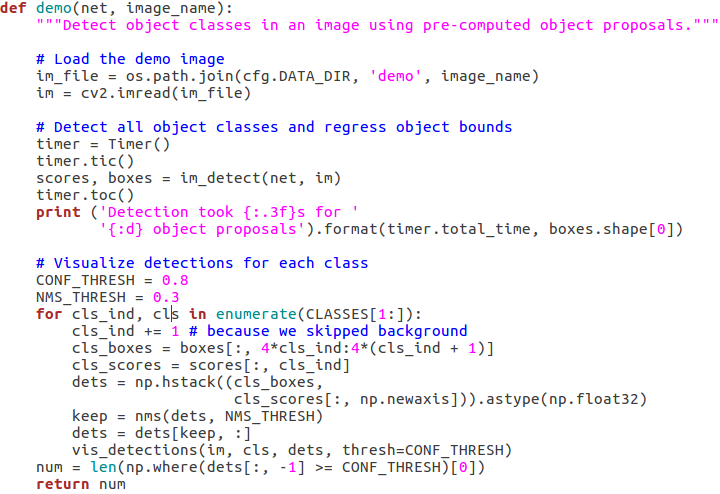
* 删除20个分类标签，添加face标签。



* 以字典的形式添加训练的模型vgg\_cnn\_m\_1024\_faster\_rcnn\_iter\_4000 0.caffemodel



* 计算检测到的人脸个数。



1. 运行测试命令：python demo.py --net vgg\_m --gpu 0

### 性能指标

1. 损失可视化

模型训练会产生相应的log文件，里面记录了训练模型过程中的重要数据，通过该文件可以获得损失的变化过程。

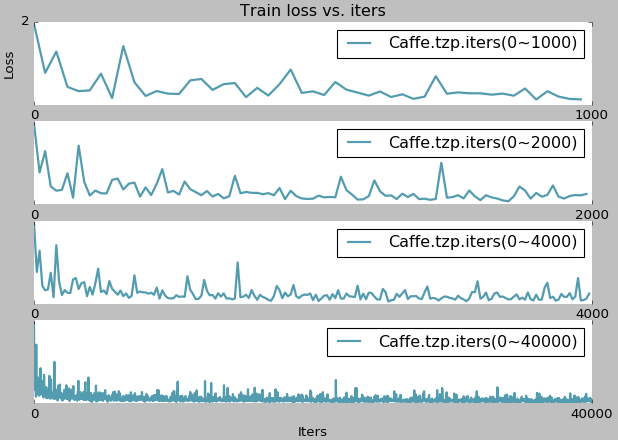


图3- 7训练总损失变化

图3- 7是不同训练迭代次数的损失变化过程对比，根据此图可以得出：随着训练迭代次数的增加损失逐渐趋向于0，并且在一定范围内迭代次数越多，损失趋向于0的程度越好。

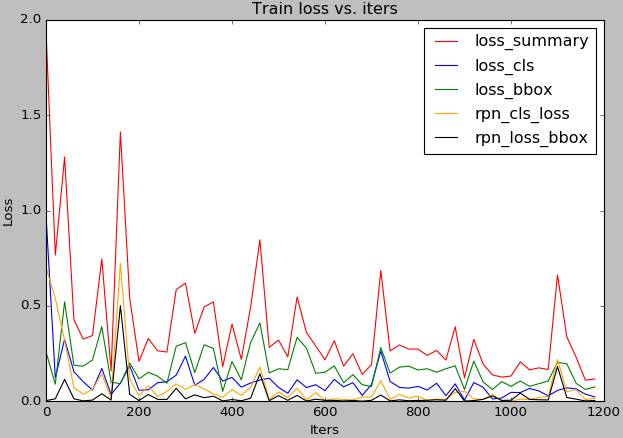


图3- 8区域生成网络和全连接网络损失对比

图3- 8是区域生成网络对应的分类、回归以及最终全连接网络的分类、回归损失对比过程，从图中可以看出：所有损失的变化趋势几乎相同，并且区域生成网络对应的损失比较小，最终全连接网络对应的损失比较大。

1. 精确率和召回率
2. 相关概念：

TP（True Positive）表示将测试样本中的人脸预测成人脸。

FP（False Positive）表示将测试样本的非人脸预测成人脸，为误检。

TN（True Negative）表示将测试样本的非人脸预测成非人脸。

FN（False Negative）表示将测试样本的人脸预测成非人脸，为漏检。

1. 精确率：针对预测样本，评价结果的质量。
2. 召回率：针对测试样本，评价结果的完整性。

评价模型的好坏不能只单独依靠精确率和召回率之中的任意一个性能指标，全面评价分模型的标准是准确率和召回率的共同作用。如果模型只检测特征明显的人脸样本，而忽略所有疑似人脸的样本，那么检测的精确率就会很高，召回率就会很低；如果模型将所有可能的人脸都划分为人脸，将剩余的人脸归类为非分类，那么检测的精确率就会很低，召回率就会很高。

通过face\_pr.pkl文件，可以获取训练模型的平均精确率大小和精确率、召回率的变化过程。

1. 平均精确率的大小：'ap': 0.90766430695321865
2. 精确率、召回率的变化过程：

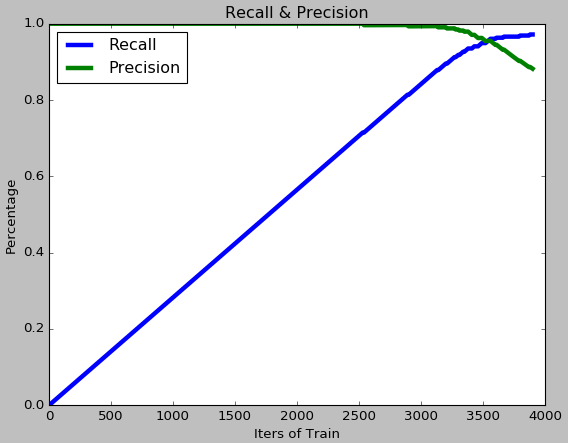


图3- 9精确率、召回率的变化过程

0<迭代次数<3200：随着训练迭代次数的增加，模型的召回率几乎以固定的速率增大，模型的精确率始终保持在1.0左右。

3200<迭代次数<4000：随着训练迭代次数的增加，模型的召回率逐渐趋于1.0，模型的精确率开始减小。

## OpenCV人脸检测的实现

### 概述

OpenCV(Open Source Computer Vision Library[19] )是一款研发于 Intel微处理器实验室的视觉交互研究组的可以获取源代码、用以处理图像的计算机视觉库。它是以C和C++编写完成的，提供了Python、Matlab等计算机编程语言的接口，使用方便、非常高效。 OpenCV具有很多特性：其代码是完全开源的，不仅可以很快地让初学者上手而且能让使用者快速地解决编程中遇到的难题；其矩阵运算能力和图像处理能力是非常强大的，使得OpenCV广泛地应用于计算机视觉领域；它可以跨平台运行在Windows、Linux、Mac OS等操作系统中，受到了学术界和工业界研究人员的钟爱。

### 测试模型

OpenCV人脸检测的实现调用haarcascade\_frontalface\_default.xml。图3- 10是批量检测每张输入图片所包含的人脸个数、检测时间以及图片总数结果。

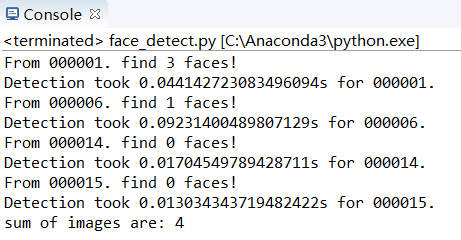


图3- 10输出每张图片检测到的人脸数及图片总数

1. 遍历程序设计竞赛参赛队员上传到本地的所有参赛图片。
2. 利用函数cv2.cvtColor()将每一张图片转换成灰度图，降低计算强度。这个函数有两个参数，第一个是传入的原始图片，第二个是颜色空间转换参数cv2.COLOR\_BGR2GRAY。
3. 调用人脸检测文件haarcascade\_frontalface\_default.xml作为人脸分类器。
4. 调用函数faces = face\_cascade.detectMultiScale(

gray, scaleFactor = 1.15, minNeighbors = 4) 探测人脸。

gray指第2)步转换的灰度图，scaleFactor指扫描的后一次搜索窗口大小是前一次搜索窗口的百分之十五。minNeighbors指最终呈现出来的检测目标候选框个数大于或等于4，也就是说检测候选框个数小于4的目标会被排除，最终不会被呈现。



## 效果对比

在大量非重叠性测试实验中，我选取了一些具有代表性的参赛照片作为模型测试数据，分别采用基于OpenCV和Faster R-CNN的方法进行检测并展开进一步的分析。

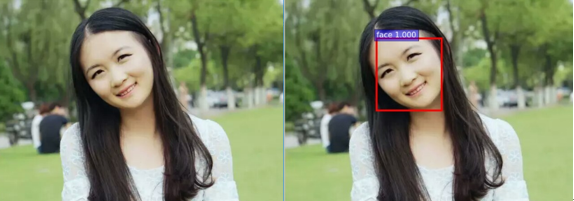


图3- 11基于OpenCV（左）和Faster R-CNN（右）人脸检测方法模型测试结果

从图3- 11中可以看出：OpenCV存在检测不到人脸的情况，初步设想造成此现象的原因是参数选择不当。为了探究出现这一现象的原因，设置minNeighbors = 0，目的是在不设定任何限制的情况下观察原始候选框的结果。结果如图3- 12所示：只检测出了一个置信度很低并且受其他候选框干扰的完整人脸。因此，排除参数选择不当的情况。

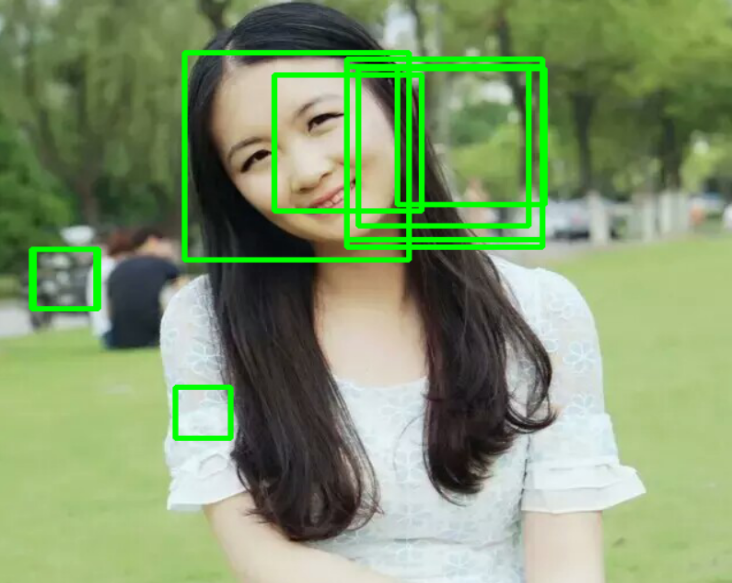


图3- 12设定minNeighbors为0的基于OpenCV人脸检测方法模型测试结果



图3- 13基于OpenCV人脸检测方法模型测试结果



图3- 14基于Faster R-CNN人脸检测方法模型测试结果

根据图3- 13、图3- 14的测试结果可以观察出如下现象：在一定偏差范围内，基于OpenCV人脸检测方法的模型只能检测出几乎垂直于底边轮廓的正面人脸而不能检测出偏移角度较大的人脸；基于Faster R-CNN人脸检测方法的模型不仅可以检测出几乎垂直于底边轮廓的正面人脸，而且能够检测出偏移角度较大的人脸。为了进一步证实上述现象的准确性，我选取了大量人脸数据进行两者的对比测试实验，并从实验结果中随机抽取8项数据展开分析。

表3- 5基于OpenCV、Faster R-CNN人脸检测方法模型测试对比数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试数据  编号 | 实际目  标数 | OpenCV  检测目标数 | Faster R-CNN  检测目标数 | 未检测目标是否  垂直于底边 |
| 000067 | 2 | 2 | 2 | 是 |
| 000002 | 2 | 1 | 2 | 否 |
| 000203 | 4 | 3 | 4 | 否 |
| 000498 | 3 | 1 | 3 | 否 |
| 000207 | 4 | 4 | 4 | 是 |
| 000349 | 8 | 6 | 8 | 否 |
| 000195 | 4 | 3 | 4 | 否 |
| 000045 | 3 | 2 | 3 | 否 |

从表3- 5的数据中可以得出如下结论：所有绝对垂直于底边轮廓的目标都能被基于OpenCV、Faster R-CNN人脸检测方法的模型检测出来，而不严格垂直于底边轮廓的目标存在不被基于OpenCV人脸检测方法的模型检测出来的情况，这就进一步证实了上述现象的正确性。

造成此现象的原因可能是：在复杂情况下，OpenCV人脸检测数据不能很好地排除噪声干扰的问题，而基于Faster R-CNN人脸检测方法的训练模型无论是在噪声干扰小的情况下还是在复杂环境下都具有很好的表现。为了充分探究这一现象背后的原理，我将大量红底背景的结婚合照和蓝底背景的证件照旋转任意角度作为测试数据进行实验，从实验结果中发现：OpenCV模型仍然不能检测出偏移角度较大的人脸。因此，排除噪声干扰的问题。

若想了解清楚这一特殊现象背后的原理，仍然需要在未来投入大量的工作开展更为深入的研究，原因有如下两点：以上实验开展过程中只考虑到“角度”这个单一影响因素而没有涉及到“像素”、“光线”、“遮挡物”等影响因素；OpenCV人脸检测数据的训练样本只有垂直于底边轮廓的正面人脸。总之，通过OpenCV人脸检测数据我们能很好地验证本实验Faster R-CNN人脸检测模型的有效性。

# 程序设计竞赛证照系统网页开发的实现

程序设计竞赛证照系统网页开发在Django框架下实现，这个部分包含两个页面，主页面用作上传参赛队员图片，次页面用来展示参赛队员所上传的图片。本章简单介绍了Django的定义、发展和特点，重点描述了Django的MTV框架构造、运行机制及其相比于MVC的优势，详细阐述了开发Django项目的环境配置和流程步骤，具体给出了将人脸检测模型和网页部分连接起来使两者成为统一整体的措施。系统后端基于Python语法进行逻辑控制，前端基于Html语言、采用django-bootstrap3进行网页的简单样式部署。

## Django

### 概述

Django是用 Python语言编写的一款开源Web框架[20] ，由于其代码的开源性、开发的高效快速性以及其封装的高度集成性，很多企业都用基于Django框架下的网页。它起源于广泛的实践需求而不是简单的科研项目，这样将Web中遇到的问题沉淀下来，实现的代码造就了Django强大的实用价值，并且其开源社区每天都在现有的基础上取得进步，这就更促进了Django的更新和完善并且提高了使用者的工作效率。如今，Django框架发展迅速、应用广泛、方便高效，不仅提供了一个简易的开发服务器，可以使用脚本文件manage.py进行开启[39] ，而且提供了一个几乎不用写任何代码的完整自助后台管理界面，可以在不用写SQL语句只用Python代码的情况下完成数据库对象的关联。

### MTV模式

Django的框架构造由Model(模型)、Templates(模板) 和View(视图)这三个部分组成，简称MTV[21] 。这与传统的框架构造组成部分Model(模型)、View(视图)和Controller(控制器)有所区别：Django的View在views.py中，主要作为控制器来完成业务逻辑操作，并在适当的情况下调用Mode和Templates，而传统的View负责与用户的交互，定义了html网页文件，决定系统显示的内容以及内容显示的形式，传统Controller的功能是接受用户输入、调用模型并且和视图一起完成业务逻辑操作。在Django中，html存放于Templates文件，决定了数据以怎样的形式显示给用户。Django的Model在models.py文件中，主要负责数据的处理，使之与数据库对应，功能与传统的Model类似，下表4- 1概括了MTV的层次和职责。

表4- 1 MTV层次和职责表

|  |  |  |
| --- | --- | --- |
| 名称 | 层次 | 职责 |
| M（模型） | 数据存取层 | 负责对象与数据库的关系映射 |
| T（模板） | 表现层 | 决定数据以怎样的形式显示给用户 |
| V（视图） | 业务逻辑层 | 完成业务逻辑操作 |

MTV这种软件开发方式的优势在于：数据访问模型、用户接口模板、完成业务逻辑操作的视图是合理分开的并且各种组件松散结合使得每个设定的应用可以独立更改而不受其它部分的影响。当我们在模型中更改数据库表名时，省去了从数据库中查找大量记录之后进行替换的时间；当我们在系统中更改其中一个应用的url时，不会干扰到其他应用程序的实现；当我们在模板中更改html前端样式时，不用接触到后端的功能程序代码。总之，在MTV这种开发模式下，各个部分各司其责，分工明确。图4- 1是Django 的MTV结构。



图4- 1 Django的MTV开发模式

当用户在浏览器中发送了一个http请求之后，网站的服务器接收这个http请求，Django根据URL匹配到相应的视图函数中对http进行处理，URL分发器能向不同的视图函数分发对应的URL页面请求，使视图函数分别对其进行处理。然后，视图函数通过模型将数据存储到数据库中并调用模板中的html，视图函数处理结束之后以html文档的形式返回http响应给浏览器，将页面展现给用户。同理，当需要向页面展示数据库存储的数据时，只需要视图函数通过模型从数据库中提取数据并将提取的数据返回给模板中相应的html中进行处理，最终以html文档的形式返回http响应给浏览器，将页面展现给用户。

### Booststrap

Bootstrap框架来自Twitter，是最受欢迎的HTML、CSS和JS框架，用于开发响应式构造、移动设备优先的web项目[40] 。它的web开发工具包不仅供给了许多款式简洁的组件、功效齐备的体系和方便快捷的插件而且可以跨浏览器设计开发从而使实现的页面简练美观并且使设计开发效率大大提高。本文项目在网页样式的部署方面采用了django-bootstrap3，它属于Bootstrap库，这组前端框架工具库可以设置样式给Web应用程序并且使用简洁灵活、操作方便。



## 搭建系统

### Python语言

Python是一种简单易学、面向对象、应用广泛、符合人类思维的解释性编程语言，适合多种类型应用程序的编程开发，有着简单脚本语言和解释性语言的易用特性[39] 。它支持各种开源库的免费调用使其广泛应用于人工智能、科学计算、游戏开发、系统运维等领域。Python语言具备许多优势：

1. 免费开源：无论是高级的开发人员还是一无所知的爱好者都可以免费使用每一个版本的python软件，并且可以查看它相关部分的源码。
2. 简单易学：其凝练的语法定义、简洁的结构创建、精短的关键字设置既可以帮助初学者快速入门又可以提高编程人员的工作效率。
3. 可移植性：用Python语言编写的程序代码几乎不需要较大的改动就可以在各种操作系统中使用而且还可以适用于不同的框架之中。
4. 可嵌入性：已有的C/C++程序需要某个特殊功能，但这个功能需要花费很长的时间进行底层编写。此时，可以将Python代码嵌入进C/C++代码的相应位置，实现这个功能。
5. 可扩展性：在某些特殊情况下，我们往往不希望一些关键代码被人查看，此时可以利用C或C++编写这部分代码，再用Python使用它们。

### 开发步骤

1. 安装Django1.11.2
2. 安装：pip install Django==1.11.2
3. 测试（三种方法）：
4. 屏幕提示Successfully
5. python -m django –version
6. python

import django

1. 创建项目

django-admin startproject djangoproject

djangoproject是项目的名称，该项目包含 manage.py和djangoproject。 manage.py是与项目进行交互命令行工具集的入口，它是将接收的命令交给Django相关部分去运行的项目管理器[41] 。djangoproject是项目的容器，保存一些项目最基本且最重要的配置：\_\_init\_\_.py、settings.py、urls.py和wsgi.py。\_\_init\_\_.py是声明模块的文件，表示djangoproject是一个模块文件；settings.py是对项目进行系统化管理并集所有子模块一体的动态交互式文件；urls.py是URL的配置文件；wsgi.py是Python的服务器网关接口，是整个项目与外部服务器通信的关键[42] 。

1. 配置数据库

进入djangoproject目录，修改settings.py:设置数据库为sqlite3。sqlite3是Django自带的轻量级数据库，用C语言编写，自给自足、完全独立、不需要服务器、零配置、支持SQL、可以直接访问其存储数据。

1. 创建应用
2. 创建APP：python manage.py startapp img\_db
3. 修改settings.py，在INSTALLED\_APPS列表里添加'img\_db',

img\_db是应用的名称。img\_db中存放数据迁移模块migrations、应用后台管理系统的配置文件admin.py、当前应用配置文件apps.py、数据模型models.py、自动化测试 tests.py和执行响应代码views.py。

1. 创建模型

进入img\_db目录，修改models.py:

class IMG(models.Model):

img = models.ImageField(upload\_to='image')

name = models.CharField(max\_length=24)

创建一个名为IMG的类，继承models.Model，IMG类包含两个属性：img和name，本地存储图像的文件名称是image，设置name的最大长度为24。

1. 创建视图，编写业务处理逻辑

进入img\_db目录，编写views.py，附录给出代码。

views.py视图函数默认接收request对象，并返回相应的数据对象。可以创建多个函数，并且每个函数有相应的URL映射。

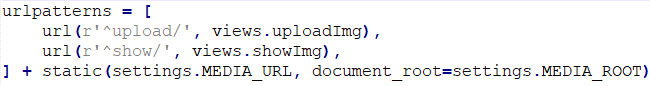
1. 创建模板，编写html

进入img\_db目录，创建Templates，在该目录下编写html。

1. 更新数据库
2. 初始化：python manage.py makemigrations img\_db

输入这条命令执行之后，会产生一个img\_db的迁移文件，通常文件名是0001\_initial.py，这能使数据库与models.py形成关系映射，存储相应的数据。

1. 更新：python manage.py migrate
2. 添加URL映射
3. 进入djangoproject目录，修改urls.py:
4. from django.conf.urls import url, include
5. 在列表urlpatterns里添加：url(r'^', include('img\_db.urls')
6. 进入img\_db目录，创建urls.py:



1. 样式部署

在使用bootstrap3对html文件设置样式之前，需要引入Bootstrap样式：



在这之后，就可以通过bootstrap3的规则对网页进行样式设置了。

1. 运行web服务
2. 运行：



1. 登录：



## 人脸检测模型接口

### 逻辑处理

1. 实现要求：
2. 参赛队员上传的文件必须是图片而且图片的类型只能是“.jpg”或者“.png”。
3. 图片的内容只能是参赛者本人而不可以是风景照。
4. 图片名称的命名规则必须以“学校\_姓名”命名。
5. 实现方式
6. 判断用户是否上传了文件：if request.method == 'POST':

如果有，则进行第2）步；如果没有，则向uploadimg.html返回'请上传参赛图片'。

1. 判断文件类型：if image.name.endswith(img\_map[i]):

将“.jpg”和“.png”以字符串的形式存放于列表img\_map，然后在for循环中用函数endswith判断用户上传文件的类型。如果类型是“.jpg”或者“.png”，则将这个文件用函数image.read()写入到本地作为临时文件，进行第3）步；如果类型不满足要求，则提示'上传失败，请上传.jpg或者.png的图片！'。

1. 判断人脸个数：if face\_num == 1:

读取2）中的临时文件并将其传入人脸检测算法，判断输入文件中检测的人脸个数face\_num。如果判断成立，则证明图片的内容是参赛者本人而不是风景照或者多人合照，进行第4）步；如果判断不成立，则返回“上传失败，图片不是本人”。

1. 图片名称判定：if '\_' in image.name:

如果判断不成立，则返回‘上传失败，图片名称格式不对，请以“学校\_姓名”命名！’；如果判断成立，系统才会将上传的文件保存到数据库里，同时会提示“上传成功”，以上过程可以用下图4- 2 upload逻辑流程展示。



图4- 2 upload逻辑流程图

参赛队员图片展示的次页面是视图函数showImg，通过imgs = IMG.objects. all()从数据库中取出图像信息，然后用模板showimg.html进行渲染，最终展示参赛图片信息。

### 模型嵌入

在Django框架下，获取参赛队员上传图片文件的方式是：在应用所在的views.py中通过函数request.FILES.get('parameter')获得图片文件，'parameter'是这个函数的参数，它和.html中表单输入的名称对应。获取到图片文件之后，将它临时存放到本地，这个步骤是模型接口的难点，因为我们获取到参赛队员上传的图片文件是不能直接通过常用读取图片的方式进行读取的，需要进行格式的转换，为了解决这个问题，我尝试输出这个图片文件的类型，但是console中始终没有打印出来，因为无法搞清楚这个图片文件的类型，就无法进行相应格式的转换，所以经过了两天的毫无进展之后，我决定放弃这个方法另找解决的办法。在这个过程中一直干扰我的是：虽然我在models.py指定了一个文件夹(upload\_to='img')但是参赛队员上传的图片文件存储路径默认是os.path.join(MEDIA\_ROOT, 'img')，并且要满足逻辑判断条件之后才会进行相应的存储。由于“判断在前存储在后”，所以我就一直没有往“先存储再判断”这个方向靠，即先将图片文件临时存放到本地然后再调用读取本地图片的函数来解决人脸检测接口连接的这个问题。我甚至尝试过直接跳过读取图片这个步骤，直接将获取到的图片文件放入人脸检测算法中，但是系统总是在这一行代码中提示错误。最终，我将获取到的图片文件image.read()写入到本地作为临时文件，然后调用常用读取图片的函数，将获取到的图片传入到人脸检测算法中，解决了人脸检测接口连接的这个问题。

# 结论与展望

## 总结

本文的研究工作来源于ACM国际大学生程序设计竞赛赛事举办，随着人工智能计算机技术的日趋成熟，引入程序设计竞赛证照系统是大势所趋，并且可以一改往届赛事的局面：防止了部分赛站冒名参赛的现象、保证了获奖证书的真实准确性。灵活运用人脸检测技术将有可能突破举办比赛存在的局限并有效提高布置效率、为人脸识别提供有力的参考数据。

本文主要研究内容如下：

1. 程序设计竞赛证照系统由人脸检测和网页开发这两个部分组成，本文重点分析了计算机视觉领域人脸检测技术的研究现状和发展趋势，对目标检测方法Faster R-CNN和人脸检测方法OpenCV中的相关模型进行了分析比较，并在Django搭建的网页平台基础上提出了程序设计竞赛证照系统任务要求的整体设计方案，包括参赛文件类型、格式、内容等具体要求。
2. 根据程序设计竞赛证照系统对参赛文件内容的实际需求，本文采用了基于Caffe底层代码实现的目标检测方法Faster R-CNN完成人脸检测模型的训练，对Faster R-CNN环境搭建、配置编译、数据预处理、模型训练、模型测试进行了详细的实验。
3. 网页开发采用结构为“MTV”的开源Web框架Django。对开发Django项目的运行机制、环境配置和流程步骤进行了详细介绍并为其与人脸检测模型的成功连接提出了具体实施办法。
4. 通过对OpenCV特征数据和在Caffe框架下训练好的基于目标检测方法Faster R-CNN的人脸检测模型相继进行了一系列非重复性的对比测试实验，大量精准良好的结果均证明两者的测试时间都能满足预期人脸检测要求，但基于Faster R-CNN方法利用程序设计竞赛证照系统参赛数据训练出的人脸检测模型测试效果更优。

## 展望

本人的程序设计竞赛证照系统通过一系列具有理论意义的实施方案和科学可靠的实验操作数据均满足了预先提出的所有任务要求，但仍然存在着许多需要改进和进行优化研究的地方。对今后的研究工作展望如下：本文的证照系统主要是针对ACM国际大学生程序设计竞赛赛事举办的，由于不同情况下赛事举办的要求有所区别、涉及到的内容也有所变化，所以要提高证照系统的适应性、推广性以及健壮性。另外，本系统只涉及到内网的访问而没有进行外网的部署，也就是说整个系统还处于无法使外网访问的状态，只能在同一个局域网下进行系统的访问，如果要使外网访问，那么就要将这个项目运行到一个公网固定IP的服务器上，所以今后要在这方面进行研究。此外，随着人工智能计算机视觉领域的飞速发展，本项目根据目标检测方法Faster R-CNN提供的源码在Caffe框架下训练的人脸检测模型在未来很有可能被更好的方法替代，如果在这个方面止步不前，本文的程序设计竞赛证照系统将会被时代抛弃。因此，只有跟着人脸检测前沿步伐前进并对其进行学习研究才能够不断更新，保证这个系统的实时性、鲁棒性、准确性。

致 谢

光阴似箭，日月如梭，我的大学时光即将结束，在这段日子里，我收获到了知识的快乐、老师的帮助、朋友的关心和同学的友情。

首先，我要感谢上海大学自动化系的周维民老师，因为在他的悉心教导下，我很好地完成了毕业项目的设计以及毕业论文的撰写。我在毕业设计期间，遇到了许多经过很长时间而无法解决的难题，周老师不论工作有多繁忙，总是能抽出时间帮我整理思路、指引方向，为我提供了一些改善性的帮助，并且对我的研究方向做出了许多建设性的指导意见，使我从中获益匪浅。在此，我诚挚地向周老师表达深深的敬意和由衷的感谢。

我还要感谢在生活上给予我支持、在精神上给予我鼓励的父母，是他们辛勤的哺育、无私的奉献给了我巨大的信心和力量，使我能在学校全心全意地投入到知识的海洋里，认真仔细、刻苦钻研。

最后，感谢参与论文评阅、指导的专家们！感谢所有陪伴我、关心我、帮助过我的人们！

参考文献

1. 陈志,李梦泽,马嫣,章韵,陈春玲.基于ACM程序设计竞赛的常规教学改革[J].电气电子教学学报,2011,33(06):18-20.
2. 黄福珍，苏剑波.人脸检测[M].上海：上海交通大学出版社.2006: 1-5.
3. 张重生. 刷脸背后[M]. 北京:电子工业出版社, 2017: 8,94-112,226.
4. 田源,于凤芹.人脸检测方法综述[J].计算机安全,2009(05):76-79.
5. 游清清,谌海云,骆俊,王小怡.人脸检测技术综述[J].无线互联科技,2017(10):137-140.
6. Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[J]. 2014:1-9.
7. Deng J, Dong W, Socher R, et al. ImageNet: A large-scale hierarchical image database[J]. Proc of IEEE Computer Vision & Pattern Recognition, 2009:248-255.
8. ImageNet. Large Scale Visual Recognition Challenge 2014(ILSVRC2014) [EB/OL]. <http://image-net.org/challenges/LSVRC/>.
9. He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[C]// Computer Vision and Pattern Recognition. IEEE, 2016:770-778.
10. 赵永科. 深度学习:21天实战Caffe[M]. 北京:电子工业出版社, 2016: 50-78,82-99,101-139,193-227,275.
11. Face++. Face Detection[EB/OL]. <https://www.faceplusplus.com/>.
12. 云从科技. 典型案例[EB/OL]. <http://www.cloudwalk.cn/caseList.html>.
13. Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]. IEEE,2014.
14. Girshick R. Fast R-CNN[J]. Computer Science, 2015.
15. Ren S, He K, Girshick R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, 39(6):1137.
16. 王远. 基于区域卷积神经网络的物体检测方法的研究与改进[D].哈尔滨工业大学,2017.
17. Jia, Yangqing, Shelhamer, et al. Caffe: Convolutional Architecture for Fast Feature Embedding[J]. 2014:675-678.
18. 杨云，杜飞. 深度学习实战[M].北京：清华大学出版社.2018: 82-86，196-207.
19. Bradski G, Kaehler A. Learning OpenCV: Computer Vision in C++ with the OpenCV Library[M]. O'Reilly Media, Inc. 2013.
20. 袁国忠. Python编程：从入门到实践[M].北京：人民邮电出版社.2016: 355-369，409-414.
21. 何敏煌. Python新手使用Django架站的16堂课[M]. 北京:清华大学出版社, 2017: 41-73,112-137.
22. 张铃, 张钹. 神经网络中BP算法的分析[J]. 模式识别与人工智能, 1994(3):191-195.
23. 刘曙光, 郑崇勋, 刘明远. 前馈神经网络中的反向传播算法及其改进:进展与展望[J]. 计算机科学, 1996, 23(1):76-79.
24. 范哲.反向传播算法浅析[J].科学技术创新,2017(23):132-133.
25. Ian Goodfellow，Yoshua Bengio，Aaron Courville.深度学习[M].北京：人民邮电出版社.2017：87-94，175-185，281-317.
26. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. Journal of Machine Learning Research, 2014, 15(1):1929-1958.
27. Uijlings J R R, Sande K E A V D, Gevers T, et al. Selective Search for Object Recognition[J]. International Journal of Computer Vision, 2013, 104(2):154-171.
28. Osuna E, Freund R, Girosi F. Training svm: An application to face detection[C]// 1997.
29. Liu W, Wen Y, Yu Z, et al. Large-Margin Softmax Loss for Convolutional Neural Networks[J]. 2016:507-516.
30. Subburaman V B, Marcel S. Fast Bounding Box Estimation based Face Detection[J]. Idiap, 2010.
31. 机器之心. 从RCNN到SSD这应该是最全的一份目标检测算法盘点[EB/OL]. <https://mp.weixin.qq.com/s/5zE78EU_NdV5ZeW5t1yV7A>.
32. Shang W, Sohn K, Almeida D, et al. Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units[J]. 2016:2217-2225.
33. Hubel D H, Wiesel T N. Receptive fields of single neurones in the cat's striate cortex[J]. Journal of Physiology, 1959, 148(3):574.
34. Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.
35. 乐国庆. 基于车载视觉系统的目标检测优化算法研究[D]. 北京交通大学, 2017:16-21.
36. 杨晓旭，高巍，顾颋. 基于卷积神经网络Caffe框架的图像分类[J].电子技术与软件工程,2017,(24):73-73.
37. 乐毅. 深度学习——Caffe之经典模型详解与实战[M].北京: 电子工业出版社.2016：11-29，36-52，58-106，239-244.
38. 郑泽宇，顾思宇.Tensorflow:实战Google深度学习框架[M].北京: 电子工业出版社.2017：25-169,232-241.
39. 陈勇. 基于Django框架的实时课堂投票系统设计与实现[D]. 大连理工大学, 2014.
40. Bootstrap. 起步[EB/OL]. <http://v3.bootcss.com/>.
41. 王冉阳.基于Django和Python的Web开发[J].电脑编程技巧与维护,2009(02):56-58.
42. 周玥. 基于 Django 框架的校园预约打印网站设计与实现[D].吉林大学,2013.