学 校 代 号 10532　　学　　号 S1802W0223

分　类　号 TP391.4　　密　　级 公开

湖南大學
HUNAN UNIVERSITY

工程硕士学位论文

基于多特征融合的驾驶员疲劳驾驶检测方法研究与实现

学位申请人姓名 汪华军

培　养　单　位 机械与运载工程学院

导师姓名及职称 邓元望教授、王炽高级工程师

学　科　专　业 动力工程

研　究　方　向 智能汽车技术

论 文 提 交 日 期 2021 年 3 月 20 日

学校代号：10532

学　　号：S1802W0223

密　　级：公开

湖南大学工程硕士学位论文

基于多特征融合的驾驶员疲劳驾驶检测方法研究与实现

学位申请人姓名：　汪华军

导师姓名及职称：　邓元望教授、王炽高级工程师

培　养　单　位：　机械与运载工程学院

专　业　名　称：　动力工程

论文提交日期：　2021 年 3 月 20 日

论文答辩日期：　2021 年 5 月 17 日

答辩委员会主席：　程军圣教授

Research and Implementation of Driver Fatigue Detection Method Based on Multi-feature Fusion

by

WANG Huajun

B.E.(Wuhan University of Science and Technology) 2018

A thesis submitted in partial satisfaction of the

Requirements for the degree of

Master of Science

in

Power Engineering

in the

Graduate school

of

Hunan University

Supervisor

Professor Deng Yuanwang

Senior Engineer Wang Chi

May, 2021

摘　要

疲劳驾驶检测是各科研机构的重点研究领域，在众多研究成果中，基于驾驶员面部特征的检测是当前疲劳驾驶检测领域的重点研究内容。但是该方法存在检测精度受光照影响比较大、实时性和准确性难以权衡、单一疲劳驾驶判断因素的误差比较大等问题。因此，本文提出一种能够有效提高弱光条件下的检测精度、多特征融合判断、很好权衡实时性和准确性的驾驶员疲劳驾驶检测算法。研究内容如下：

(1)针对驾驶员疲劳驾驶的情况主要出现在夜间，此时光线比较暗，驾驶员的眼部与嘴部等特征不明显，影响其状态的识别精度。因此，在图像进入人脸检测与提取之前，先对图像进行处理，提高图像质量，从而提高后续眼睛睁闭状态和嘴巴张开状态的识别精度。

(2)采用RetinaFace算法完成对驾驶员的人脸检测并提取人脸区域图像，采用人脸关键点定位算法，准确获取人脸关键点坐标信息，为驾驶员的头部姿态估计提供相应的关键点坐标。

(3)分析卷积神经网络模型搭建的基本流程，以SSD网络模型为基础，结合本文的应用场景，设计了一种用于辨别驾驶员眼睛状态与嘴巴状态的卷积神经网络模型。然后使用自制的数据集，对此模型进行训练，得到最合适的学习率与批处理大小等关键参数值。

(4)融合疲劳驾驶检测常采用的参数，比如PERCLOS参数、打哈欠参数和头部姿态异常参数，然后结合本文的实际情况，给出适合本疲劳驾驶检测方法的参数，比如疲劳驾驶时眼部参数阈值、嘴部参数阈值和头部姿态异常阈值，紧接着融合这些疲劳驾驶判断依据信息来建立疲劳驾驶识别模型，能有效提高系统的鲁棒性。

经过相应数据集测试，本文的疲劳驾驶检测系统有一定的实际应用价值。

关键词：疲劳驾驶；低光增强；人脸检测；卷积神经网络；头部姿态估计

Abstract

Fatigue driving detection is a singificant research field in various scientific research institutions. Among the many research results, the detection based on driver's facial features is the significant research in the field of current fatigue driving detection. Nonetheless, this method has some problems, such as the great influence of light on detection accuracy, the difficult balance between real-time performance and accuracy, and the greater errors of single fatigue driving judgment factor. Therefore, this paper proposes a driver fatigue detection algorithm that can effectively improve the detection accuracy under weak light conditions, multi-feature fusion judgment, and good balance between real-time performance and accuracy. The research contents are as follows:

(1)The fatigue driving of the driver mainly occurs at night. when the light is relatively dark, the features of the driver's eyes and mouth are unconspicuous, which influences the recognition accuracy. So before the image enters the face detection and extraction, it should be processed to increase the quality, So as to improve the subsequent eyes open state and mouth open state recognition accuracy.

(2)RetinaFace algorithm is used to accomplish the facial detection of drivers and abstract facial area images. Then I introduce the face key point localization algorithm, uesd to accurately obtain the face contour, eyes, mouth, nose and other 68 key areas of the face coordinate information, providing the driver's head posture estimation for coordinate of key points.

(3)The basic process of convolutional neural network model construction is analyzed. Then, a convolutional neural network model is designed to distinguish the state of driver's eyes and mouth based on the SSD network model and combined with the application scenarios in this paper.Then the model is trained with a self-made data set to obtain the most appropriate key parameters such as learning rate and batch processing size.

(4)Fusion of parameters commonly used in fatigue driving detection, such as PERCLOS parameters, yawn parameters, and abnormal head posture parameters, are combined with the actual situation in this article to give the parameters suitable for the fatigue driving detection method, such as eyes during fatigue driving. Integrating these fatigue driving judgment basis information to establish a fatigue driving recognition model can effectively improve the robustness of the system.

After proper dataset testing, the fatigue driving detection system in this paper has certain practical application value.

Key words: Fatigue driving; Low light enhancement; Face detection; Convolutional neural network; Head pose estimation

目　录

学位论文版权使用授权书 I
摘　要 II
Abstract II
第1章 绪论 1
1.1 选题背景与研究意义 1
1.2 疲劳驾驶研究现状 3
1.2.1 基于机动车行为特征的疲劳驾驶检测 3
1.2.2 基于驾驶员生理特征的疲劳驾驶检测 4
1.2.3 基于驾驶员面部特征的疲劳驾驶检测 5
1.3 本文研究的主要内容 5
1.4 论文章节安排 6
第2章 低光图像增强算法 8
2.1 光照对图像的影响 8
2.2 传统低光图像增强算法 8
2.2.1 灰度变换 8
2.2.2 图像去噪 9
2.2.3 Gamma变换 11
2.2.4 直方图均衡化 11
2.3 主流低光图像增强算法 12
2.3.1 LIME算法 13
2.3.2 MBLLEN算法 13
2.3.3 KinD算法 14
2.3.4 Zero-DCE算法 15
2.4 图像增强算法对比与选取 16
2.5 本章小结 17
第3章 驾驶员面部检测与头部姿态估计 18
3.1 基于RetinaFace算法的驾驶员人脸检测 18
3.2 人脸关键点定位 20
3.3 头部姿态估计 22
3.4 实验设计与结果分析 25
3.4.1 实验数据处理与平台搭建 25

3.4.2 人脸检测结果与分析 26
3.4.3 头部姿态估计结果与分析 27
3.5 本章总结 28
第4章 基于卷积神经网络的眼部与嘴部状态识别 29
4.1 卷积神经网络概述 29
4.1.1 局部连接 29
4.1.2 权值共享 30
4.2 卷积神经网络的组成 31
4.3 卷积神经网络的训练 34
4.4 卷积神经网络模型的搭建 38
4.5 数据集的制作 40
4.6 卷积神经网络模型的训练与优化 41
4.6.1 学习率的选取 41
4.6.2 批处理大小的选择 44
4.7 实验设计与结果分析 44
4.7.1 实验数据处理与平台搭建 44
4.7.2 实验结果分析 45
4.8 本章小结 46
第5章 多特征融合疲劳驾驶检测 47
5.1 疲劳驾驶参数提取 47
5.1.1 眼部疲劳驾驶参数提取 47
5.1.2 嘴部疲劳驾驶参数提取 49
5.1.3 头部姿态疲劳驾驶参数提取 49
5.2 实验设计与结果分析 51
5.2.1 实验数据处理与平台搭建 51
5.2.2 鲁棒性测试 52
5.2.3 准确性测试 53
5.3 本章小结 54
总结与展望 55
1. 工作总结 55
2. 工作展望 56
参考文献 57
致谢 61
附录A 攻读硕士学位期间参加的科研项目 62
附录B 核心代码 63

第1章 绪论

1.1 选题背景与研究意义

伴随着社会经济的飞速发展，越来越多的家庭都拥有了汽车，汽车也成为了当今社会最主要的交通工具。根据我国交通管理部统计，截止2020年末，全国注册登记的机动车有3.75亿辆，其中，汽车约为2.7亿辆，占机动车总量的70%以上，汽车驾驶的人数约为4.1亿。根据商务部统计，预计在2021年6月，中国汽车保有量将超越美国，居于世界第一，其详细数据如图1.1所示。在巨大的车辆基数下，汽车在给人们的日常出行带来便利时，也带来了许多的问题，比如交通事故频发、道路拥塞、环境污染等，当前这些问题已经严重危害着人们的财产和生命安全，亟需解决。

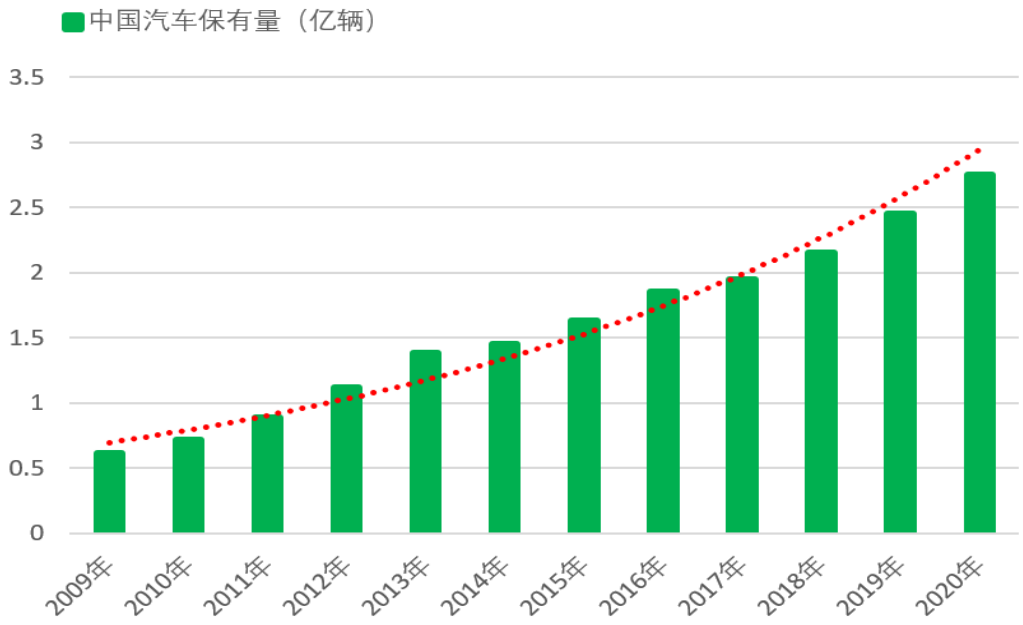


图1.1汽车保有量趋势图

世界卫生组织统计数据显示，人类在意外死亡事故原因中，因道路交通事故而死亡的原因占比约50%，每年大约有140万人因此死亡。我国每年大约有15万人死于各种各样的交通事故，其致死率约为27.3%，远远高于美国、英国等发达国家。其原因五花八门，有违章驾驶、超载驾驶、超速驾驶、酒后驾驶、疲劳驾驶等，占比如图1.2所示。其数据显示，在交通事故死亡的原因中，疲劳驾驶占比高达12%。并且，我国交通部的相关调查数据显示：疲劳驾驶是引发重大和特大交通事故发生的首要原因，其占比高达50%。此外，美国汽车交通安全部门的相关调查数据显示：疲劳驾驶在美国的交通事故死亡原因中占比高达21%。通过以上数据可知，

疲劳驾驶严重威胁人们的生命安全，是交通事故产生的首要原因。所以，一个有效的疲劳驾驶检测预警设备尤为重要。

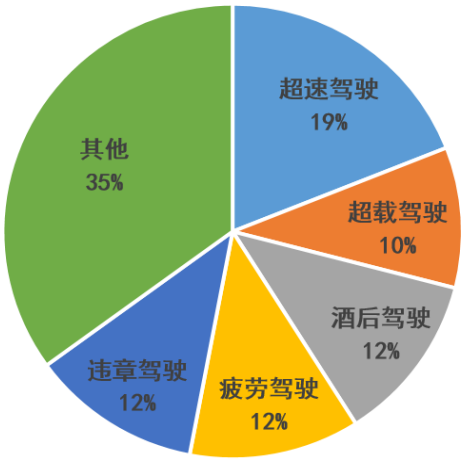


图1.2 道路交通事故成因比例图

一般来讲，造成驾驶员疲劳的现象主要有驾驶员自身状态和驾驶室内的环境两个因素。驾驶员自身状态指的是驾驶员连续驾驶车辆未停车休息或者睡眠时间没有达到正常标准。驾驶室内的环境主要影响驾驶员的心情与精神，造成驾驶员疲劳，主要包括噪声、振动以及温度。具体如图1.3所示。接下来将详细介绍驾驶室内环境对驾驶员疲劳程度的影响。

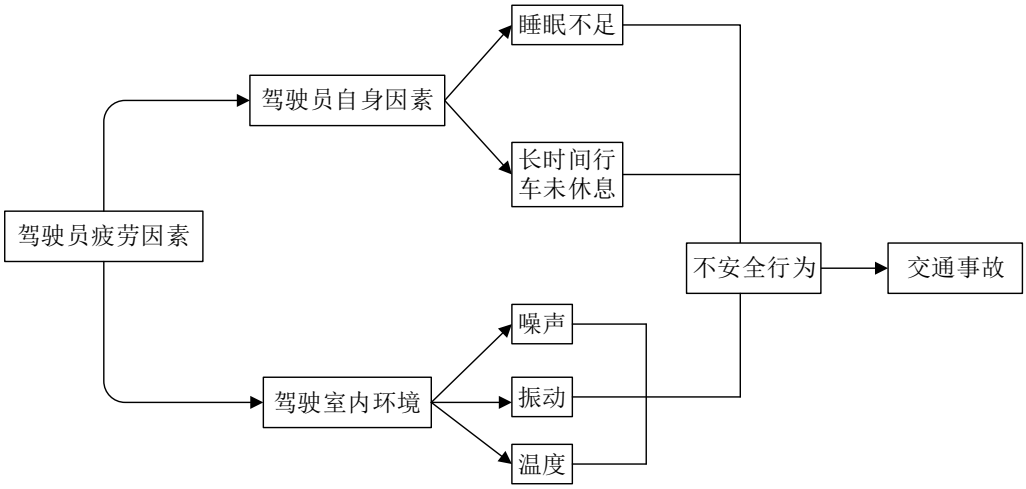


图1.3 驾驶员疲劳驾驶因素

(1)驾驶室内噪声对驾驶员的影响

驾驶室内的噪声主要来源于发动机和传动机构。当噪声超过90分贝时，人们就会产生头晕、心痛、心情急躁等现象；当噪声超过120分贝时，人们就会产生晕眩、呕吐、恐惧以及暂时性耳聋等现象。因此，如果驾驶员长时间在噪声的环境中行车，就会加速驾驶员疲劳，使驾驶员的注意力难以集中并且出现异常的动作或者不应有的失误，从而影响行车安全。

(2)驾驶室内振动对驾驶员的影响

驾驶室内振动主要指的是车辆在行驶过程中，驾驶室在横向、纵向以及垂直方

向上发生的机械运动。实验结果证明，长时间处于振动环境，人们会出现手腕乏力、手指麻木、腰酸背痛以及失眠等现象。因此，如果驾驶员长时间在振动的环境中行车，就会使驾驶员头、颈以及背感到不舒适，从而加速驾驶员的疲劳程度，使驾驶员的驾驶技能明显下降，从而影响行车安全。

(3)驾驶室内温度对驾驶员的影响

人们长时间位于高温度的条件下，往往会感到浑身不舒服、乏力无精神。理论研究表明，驾驶室内的温度应该保持在18~24℃，当温度高于或者低于这个温度时，就会加速驾驶员的疲劳程度。温度过高时，驾驶员的记忆力下降和驾驶员体验感变差，容易出现操作失误现象；温度过低时，驾驶员的肌肉活动能力下降，非常容易引起驾驶员疲劳。

由于疲劳驾驶时，驾驶员的行为特征比较明显，这一现象是可以通过相应的技术手段进行预测，并且给出驾驶员相应的预警，为驾驶员提供有效的保护。从降低交通事故率和技术实现可行性角度考虑，研发一种能够准确识别驾驶员疲劳驾驶并提供报警的系统是可行并且非常有必要的。

1.2 疲劳驾驶研究现状

疲劳驾驶检测的研究一直以来都是世界各国科研机构研究的重点领域，驾驶员在疲劳驾驶时，生理特征会有比较明显的改变，主要体现在眼电波、心电波、心率、眼电波等的改变；心理特征也会有比较明显的变化，主要体现在驾驶员面部特征的变化，比如频繁眨眼、打哈欠、点头等。因此针对疲劳驾驶的检测，目前国内外相关研究人员主要提出三种了解决方案，如图1.4所示，接下来将分别介绍这三种解决方案。

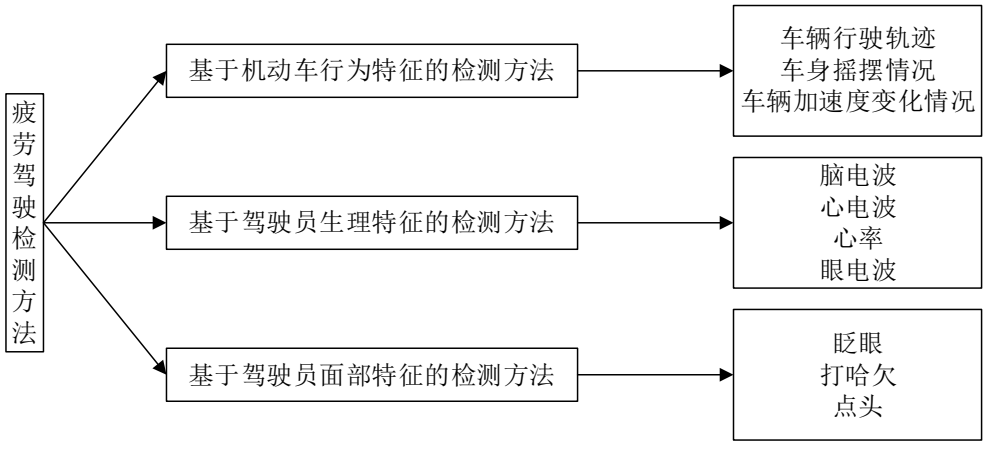


图1.4 疲劳驾驶检测方法

1.2.1 基于机动车行为特征的疲劳驾驶检测

驾驶员疲劳驾驶时，其操作车辆的行为特征与正常驾驶情况下区别明显，比如

驾驶员手握方向盘的力度会出现变化、车辆的速度控制不稳定、车身会左右摇晃并偏离正常行车道等。因此，我们可以根据驾驶员正常驾驶和疲劳驾驶时，不同驾驶信息来判断驾驶员是否疲劳驾驶。MS Wang[5]等人根据汽车的速度信息和驾驶员手握方向盘的力度信息以及方向盘的角度信息,来判断驾驶员是否疲劳驾驶，同时对上述的判断因子进行单独对比之后发现，评估驾驶员疲劳驾驶的最有效的因子是车辆的侧向加速度信息。沙春发等人通过压力传感器采集驾驶员在疲劳驾驶与正常驾驶状态的手握方向盘的力度，对其握力的五个特性参数进行分析，发现绝大部分驾驶员在不同的精神状态下，这些参数有显著特征，但是由于人为变化因素明显，可以作为疲劳驾驶的辅助检测手段。

车辆在正常行驶情况下，车身不会出现频繁的左右摇摆现象以及频繁的压线现象。因此，可以通过车载摄像头，实时采集车辆在行驶过程中出现的压线情况，对驾驶员是否疲劳驾驶进行判断。B Cheng[6]等人在车辆的行驶过程中，使用车载摄像头采集车身与车道线之间的距离信息，来判断车辆是否频繁出现压线现象，以此推断驾驶员是否疲劳驾驶。

这种检测方法，不需要外加复杂的传感器就能完成对驾驶员是否疲劳驾驶的预测，数据处理也相对简单，但是这种方法受外界因素与人为因素影响大，比如在有的路况下并没有车道线、驾驶员的操作习惯差别大等，误判概率大。因此，这种方法目前主要在疲劳驾驶中起到辅助检测的作用，并没有单独运用到疲劳驾驶检测上。

1.2.2 基于驾驶员生理特征的疲劳驾驶检测

驾驶员疲劳驾驶时，其心理特征与正常驾驶情况下区别明显，比如心电信号、脑电信号、体表温度、眼电信号和心率等变化明显。因此，测量驾驶员在疲劳驾驶状态和正常驾驶状态的心理特征的数据，就能够制定一个有效的疲劳驾驶判定标准，来对疲劳驾驶进行预测报警。

脑电信号是通过一定规则放置在头皮上的电极采集的脑电波活动的信号，相应的临床实践数据显示，脑电信号含有许多的生理信息数据。因此，可以通过采集驾驶员的脑电信号来预测疲劳驾驶。Picot[7]等人使用单通道脑电信号的alpha功率带(8-12Hz)来在线检测驾驶员是否疲劳驾驶，经过大量实测数据，此种方法对疲劳驾驶检测的精确度大概为85%。赵晓华[10]等人通过测量表征脑电信号的信息来检测驾驶员是否疲劳驾驶，经过大量的实测数据，此种方法对疲劳驾驶检测的精确度高达95%。通过以上例举的两个人研究，可以表明通过脑电信号检测疲劳驾驶完全可行。

心电图能够有效反应人们在不同场景下心电特征的变化。因此，国内外很多学者投入大量精力研究驾驶员在疲劳驾驶的时候，心电图是如何变化的，并通过大

量数据论证，得出了如果驾驶员的每分钟心跳次数低于正常情况下每分钟的心跳次数的20%时，就可以判定此时驾驶员疲劳驾驶。吴群[2]等人根据基于深度学习的心电图特征识别技术，建造了一个用于判断驾驶员是否疲劳驾驶的模型，使用该模型可以近似模拟出驾驶员从正常驾驶状态到疲劳驾驶状态的心电信号波形的特征，通过对比不同状态下的心电信号，可以准确对疲劳驾驶进行检测。

目前的技术手段能够准确检测人体生理信号的变化，并且通过大量数据论证也发现了驾驶员在正常驾驶和疲劳驾驶时，其心理特征变化明显。因此，根据相应的数据能够准确预测驾驶员是否疲劳驾驶，具有非常高的准确率和说服力。但是由于这种检测方法是接触式的，驾驶员需要配套相应的数据采集设备才能够完成相应的疲劳驾驶检测，驾驶员的驾驶体验感不好并且还会影响正常的驾驶，有一定的安全风险。此外，信号检测和处理设备比较昂贵。所以，这种疲劳驾驶检测方法的商业使用用途不大，难以普及。

1.2.3 基于驾驶员面部特征的疲劳驾驶检测

驾驶员疲劳驾驶时，其面部特征与正常驾驶情况下区别明显，比如驾驶员会频繁打哈欠、闭上眼睛的时间会增长以及头部姿态异常等。并且随着计算机软件与硬件的飞速发展，进一步促进了人工智能领域的发展，目前将人工智能技术运用到疲劳驾驶检测领域已经取得重大成功，也是疲劳检测领域非常热门的研究方向。就是通过摄像头等图像传感器采集到驾驶员的面部图像信息，使用深度学习技术，分析驾驶员眼睛状态、嘴巴状态以及头部左右上下晃动的幅度等来判断驾驶员是否出现疲劳驾驶的现象。

孙伟[12]等人提出了一种改进的BFA算法，其基本思路是设计了自回归积分平台(ARIMA)、径向基函数神经网络(RBFNN)和卡尔曼滤波(KF)三种预测因子，并将其加入到BFA中，改进后的BFA在对驾驶员是否疲劳驾驶预测精度明显高于传统的BFA。童兵亮[13]等人以驾驶员正常说话时嘴巴状态、嘴巴闭合状态以及打哈欠时嘴巴状态为数据源，使用BP神经网络训练出相应的模型，根据嘴巴张开的高度与宽度的比值来判断驾驶员是否疲劳驾驶。沈英超[14]等人首先使用MTCNN算法得到驾驶员的面部区域，然后使用一种基于多任务的深度学习框架来定位人眼区域，最后使用卷积神经网络来预测驾驶员是否疲劳驾驶。

1.3 本文研究的主要内容

疲劳驾驶是国内外科研机构研究的热点领域，由相应文献论证可知，基于机动车行为特征的检测方法会受到很多外部影响干扰，导致识别精度低；基于驾驶员生理特征的疲劳驾驶检测需要驾驶员佩戴相应的测试设备，属于接触式检测方法，其缺点比较明显，比如驾驶员驾驶体验感差、相应的测试设备价格昂贵以及会影

响驾驶等；而基于驾驶员面部特征的检测方法由于人工智能的飞速发展，已经不存在技术难题并且安装成本低、对驾驶员的眼睛状态和嘴巴状态识别精度高。综上所述，基于驾驶员面部特征的检测方法优势明显，所以本论文主要对这种方法展开深入研究。由于驾驶员出现疲劳驾驶的情况大部分出现在夜间，而夜间光线比较暗，如何提高这种情况的完场面部特征提取非常重要。并且，基于单一因素的疲劳检测误差比较大。因此，本文融合驾驶时眼部状态、嘴部状态和头部姿态三个因素完成对是否疲劳驾驶进行预测，提高识别精度和鲁棒性。论文主要工作如下：

(1)查阅相关文献完成对人脸识别、低光图像增强、面部特征提取与识别的理论、卷积神经网络、疲劳驾驶参数等知识的调研，为后续工作做相应的准备。

(2)使用摄像头等图像传感器完成驾驶员图像数据的采集，然后使用低光图像增强算法完成在光线较暗情况下的图像处理。

(3)使用RetinaFace算法完成驾驶员人脸区域的提取，紧接着使用人脸关键点定位算法，获取眼睛、嘴巴、面部轮廓等人脸关键坐标信息，为后续的处理做准备。

(4)使用卷积神经网络模型完成驾驶员的嘴巴状态与眼睛状态辨别。结合理论实际，给出嘴巴、眼睛以及头部姿态是否疲劳的阈值参数，最后融合这些疲劳参数完成对驾驶员是否疲劳驾驶进行预测。

(5)采用Python语言编程，基于Opencv、Dlib、Numpy、Imutils、pytorch等函数库、wxPython界面设计工具等设计出一款用于疲劳驾驶的软件。

1.4 论文章节安排

论文共有六章，每章内容如下：

第1章：绪论。本章介绍了疲劳驾驶研究的意义，详细地阐述了疲劳驾驶领域研究的现状，分析了目前主流疲劳驾驶检测方法的优劣性，提出了本文应采取的技术路线，详细地给出了具体的研究技术。

第2章：低光图像增强算法。本章首先了介绍了低光条件下，对驾驶员面部特征提取的难度。因为，驾驶员疲劳驾驶的情况主要出现在夜间或者正午，此时光照对识别精度的影响比较大。所以，紧接着介绍了传统的低光图像增强算法与近几年典型的低光图像增强算法，并比较它们的优缺点，最终确定了符合本文应用场景的低光图像增强算法。

第3章：驾驶员面部检测与头部姿态估计。本章首先介绍了RetinaFace算法，完成对驾驶员人脸的定位。紧接着，简要概述了人脸关键点定位算法，然后使用这种算法准确获取了脸部轮廓、眼睛、嘴巴、鼻子等人脸68个关键区域的坐标信息。然后介绍了头部姿态估计所涉及的理论知识。最后根据人脸的关键点坐标完成对驾驶员的头部姿态估计。通过相应实验，验证了在弱光条件下对驾驶员人脸特征

提取的效果比较好。

第4章：使用卷积神经网络模型完成驾驶员的嘴巴状态与眼睛状态辨别。本章首先简要概述了卷积神经网络模型的基本组成，然后分析卷积神经网络模型搭建基本流程，紧接着以SSD网络模型为基础，设计了一种用于辨别驾驶员眼睛状态与嘴巴状态的卷积神经网络模型，然后使用自制的数据集，对此模型进行训练，得到最合适的学习率与批处理大小等关键参数值。最后通过实验对比，得出了该卷积网络模型在准确率和处理速度上都具有一定的优势。

第5章：多特征融合疲劳驾驶检测。本章首先介绍了基于面部特征的疲劳驾驶检测常采用的参数，比如PERCLOS参数、打哈欠参数和头部姿态异常参数。然后结合本文的实际情况，给出了适合本疲劳驾驶检测方法的参数，比如疲劳驾驶时眼部参数阈值、嘴部参数阈值和头部姿态异常阈值。紧接着融合这些疲劳驾驶判断依据信息来建立疲劳驾驶识别模型，能有效提高系统的鲁棒性。最后以YawDD数据集以及摄像头实时采集的数据为测试集，验证本文的疲劳驾驶检测系统的可行性。

第6章：总结与展望。简要阐述了本课题所做的工作和成果，针对本研究存在的不足之处，给出了在未来可以继续优化提高的地方。

第2章 低光图像增强算法

车辆在行驶过程中，驾驶员在早、中、晚时面部所受到的光照变化比较大，加大了对驾驶员人脸目标检测与面部特征提取的难度。并且，驾驶员疲劳驾驶主要发生在夜间或者正午行车，在夜间行车时，光线比较黑暗，如何补光提高人脸图像的质量至关重要；在正午行车时，光线比较强烈，如何适当消除光照提高人脸图像的质量也非常重要，这些因素关乎本文的疲劳驾驶检测方法的判断精度。因此，本章将论证如何对图像进行预处理，来提高图像质量，为后续提高本文疲劳驾驶检测方法的判断精度埋下伏笔。

2.1 光照对图像的影响

随着计算机软硬件技术的蓬勃发展，越来越多的科研机构扎根于人工智能领域，推动了人脸识别技术的发展，并且逐步实现商业化。但是，经过相应的理论研究，人脸的表情变化、姿态变化以及光照变化对人脸识别的精度造成了严重影响，它们是制约人脸识别技术发展的主要原因。根据美国军方的人脸数据库(Face Recognition Technology)的测试结果，表明了光照[16]是束缚人脸识别技术得到商业化应用的主要因素。例如，在复杂光照或者暗光条件下，会导致人脸图像光照分布不均匀或者信息丢失。

2.2 传统低光图像增强算法

2.2.1 灰度变换

在弱光或者光照强烈的条件下，摄像头等图像传感器所采集的图像像素分布不均匀，较暗图像的像素大多分布在灰度值较小的区域，较亮图像的像素大多分布在灰度值较大的区域。而对于正常光照图像，其像素分布均匀。所以，在弱光或者光照强烈的条件下，图像像素差异范围比较大，图像质量比较差。因此，我们可以使用灰度变换来使图像的像素均匀分布，缩小图像像素差异范围，改善图像质量。灰度变换的本质是就是按照一定的规则修改图像的每个像素点的值，使图像的像素分布均匀。接下来将介绍目前常用的两种灰度变换理论基础与优缺点。

(1)线性变换

通俗来讲，图像的灰度线性变换就是将图像中的像素值按照灰度映射函数来调整源图像的像素值，从而达到增强图像的目的。灰度映射计算公式如下：

$$g(x,y)=k\times f(x,y)+d \tag{2.1}$$

式中，k是比例系数，$f(x,y)$是原始图像灰度值，$g(x,y)$是变换后图像灰度值，d是截距。当$|k|>1$时，图像的整体对比度明显增大，整体上增强了图像的显示效果；当$|k|=1$时，可以通过改变d的值来轻微增强或者减弱图像的对比图，以起到改善图像显示的效果；当$|k|<1$时，图像的整体对比度明显减弱，图像整体看起来也越灰暗。

(2)非线性增强

灰度非线性变化主要用来处理对图像的局部像素进行增强的情况。即当图像的像素主要集中在中间灰度部分，而其他部分像素比较少，只需要扩展像素集中的部分。对数变换计算公式如下：

$$g(i,j)=\alpha\times\log(1+\beta\times r)\qquad r\in[0,1] \tag{2.2}$$

式中，α、β是比例系数，$g(i,j)$是经过处理后的图像灰度值，r是原来的图像灰度值。

幂次变换计算公式如下：

$$g(i,j)=\alpha\times[f(i,j)]^{\beta} \tag{2.3}$$

式中，α、β是比例系数，$g(i,j)$是处理后图像的灰度值，$f(i,j)$是原来图像的灰度值。

使用灰度线性变换和灰度非线性变换对同一照片进行处理，其处理结果如图2.1所示。从图中可以看出，灰度线性变换和灰度非线性变换都能对图像进行一定程度的增强，但是同时也增加了图像的噪点问题，因此，需要对使用灰度变换增强的图像进行降噪处理。

a)原图　　b)线性增强后图像　　c)非线性增强后图像

图2.1 灰度变换

2.2.2 图像去噪

数字图像信号在形成过程、传输过程以及记录过程中，信号强度会遇到衰落下降并且也会受到环境因素的干扰，影响图像画质质量，因此需要通过合理方法消除这些造成图像画质不清晰的因素。通常情况下，影响图像画质质量的因素主要是椒盐噪声和高斯噪声。椒盐噪声是随机在图像较亮的区域出现黑点或者在较暗

的区域出现白点的噪声，产生的原因主要是数字信号突然受到了强烈的干扰。对于椒盐噪声，可以采用中值滤波进行处理。高斯噪声是指噪声的分布非常类似于高斯图像，因此得名，产生的原因是主要是数字信号在采集的过程中受到了不良光照或者高温。对于高斯噪声，可以采用线性滤波进行处理。接下来，将介绍目前比较常见的图像降噪方法

(1)中值滤波

中值滤波是统计滤波的一种典型应用。通俗来讲，就是把图像中3×3像素框内的像素值按照从小到大顺序排序，然后取其中值代替像素框中所有像素值。如图2.2所示。一般情况下主要用来处理椒盐噪声，并且计算简单，时间复杂度比其他滤波要低。

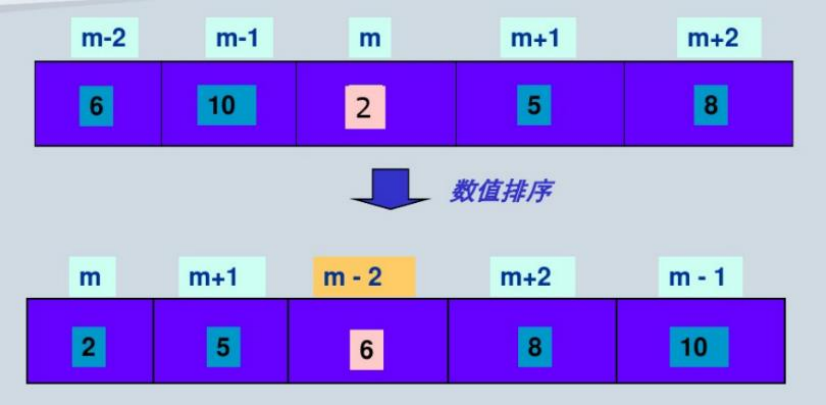


图2.2 中值滤波

(2)高斯滤波

高斯滤波是一种线性的图像处理方式，能够有效地抑制图像的噪声，达到改善图像质量的效果，主要用来降低高斯噪声。通俗来讲，高斯滤波就是计算每个离散点上的高斯函数值，然后扫描图像中的每一个像素对其做一定范围领域内的加权平均，这样做可以保证图像的连续性，进而有效降低了高斯噪声。但是高斯滤波计算较复杂，时间复杂度比较高。

(3)均值滤波

均值滤波是一种线性的图像处理方式，这种方式能够有效地处理掉图像中与周围有明显差别的像素点。通俗来讲，均值滤波就是首先计算当前像素点周围的n×n个像素点的平均值，然后使用计算出的平均值替换掉这个像素点的值，最后使用该方法依次遍历计算图像中的每一个像素点的值。通过上述介绍，其数学计算公式简单，相应算法的空间复杂度和时间复杂度也比较低，因此在处理速度上比较有优势。但是均值滤波的缺点也比较明显，就是对图像进行处理时，会破坏图像的细节部分，并且在降低噪声的同时也引入了不少其他的噪声，并不能明显提高图像的清晰度。当驾驶员在开车时，低光、强光或者突然增强的光照是影响驾驶员图像质量的主要因素，综上所述，对于处理此种情况我们优选中值滤波。

图2.3是中值滤波和均值滤波对低光条件的图像处理情况，可以明显看出，中值滤滤波对低光条件下的图像处理情况优于高斯滤波。

a)原图　　b)均值滤波　　c)中值滤波

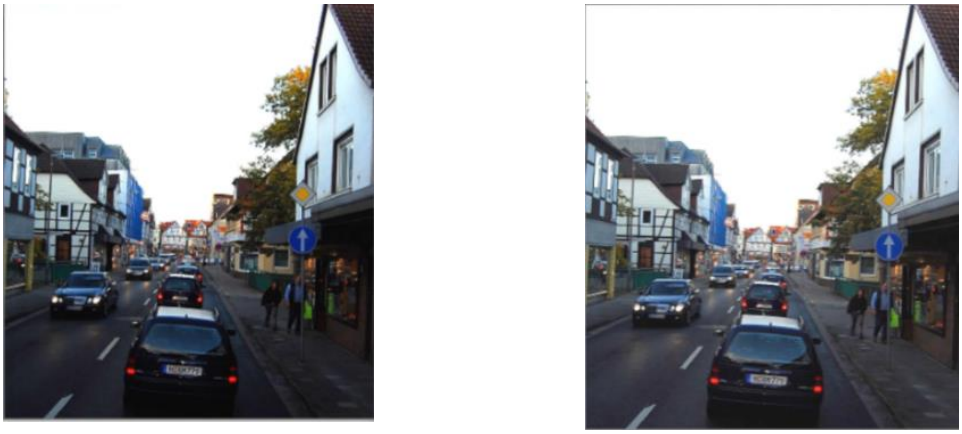
图2.3 图像降噪

2.2.3 Gamma变换

当驾驶员在开车时，由于转弯或者夜间行车时对面车辆光线照射分步均匀，导致摄像头等图像采集传感器采集的图像可能出现某一部分灰度过高或者灰度过低，使得图像的清晰度比较低，影响后续疲劳驾驶判断的识别度。因此可以使用Gamma变换对图像进行修正，对灰度值过高的图像区域进行适当降低，对灰度值过低的图像区域进行适当增强，从而提高图像的清晰度。计算公式如下：

$$s = cr^{\gamma} \qquad r \in [0,1] \tag{2.4}$$

式中，c是比例系数，r是原始图像灰度值，s是变换后图像灰度值。其中γ为临界值，当γ的值越小，对灰度值越低的图像区域的增强作用就越明显；当γ的值越大，对灰度值越高的图像区域的减弱作用就越明显。图2.4是Gamma变换后对图像的处理效果，可以看出对低光条件下的图像有明显的增强效果。

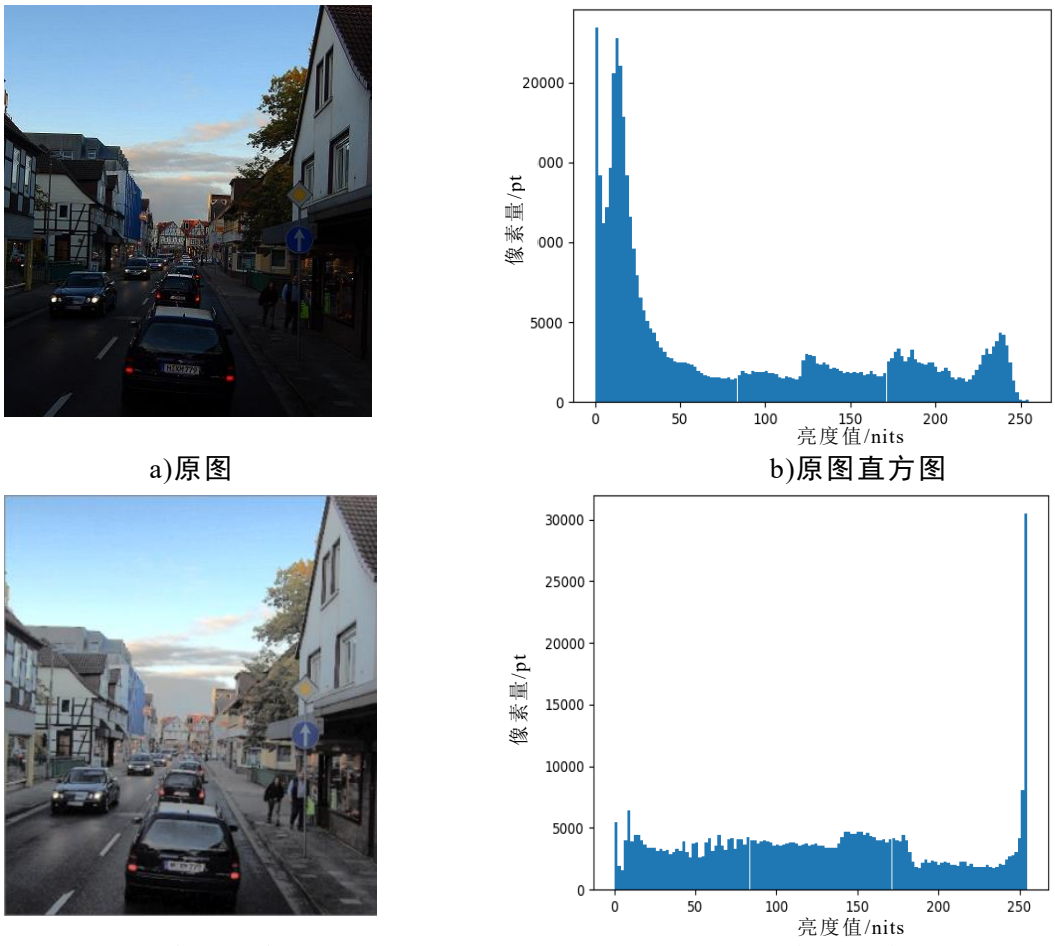


a)原图　　b)Gamma变换

图2.4 Gamma变换

2.2.4 直方图均衡化

在弱光或者强光条件下，图像的像素点分布不均匀，主要集中在某一窄小的部分，使得图像的清晰度比较低。直方图均衡化就是为了解决图像像素点分布不均匀的现象，本质上就是对图像中像素点出现的频率进行重新分配，使变换后的图像中每个像素点出现的频率大致相同[30]。如图2.5所示，原来图像直方图图形化的峰顶部分像素出现的频率会适当降低，而谷底部分像素出现的频率会适当提高，输出的图像直方图是比较平整的图形，从而增强图像对比度。其优点是能够很好地处理太亮或者太暗的图像，但是缺点就是对数据不加选择，使得图像某些细节消失。



a)原图　　b)原图直方图

c)直方图均衡化　　d)均衡化直方图

图2.5 直方图均衡化对比

2.3 主流低光图像增强算法

由上文介绍可知，目前常见的图像增强算法，都是通过相应算子对整副图像进行处理，是一个全局的过程，是对图像中所有的像素进行同一操作的过程，并没有考虑到图像每个区域的亮度差异，这种方法可能会导致图像中的某些亮度比较高的区域过度曝光，从而降低图像的清晰度。所以，为了摒弃传统图像增强算法的弊端，目前在图像增强领域的主流研究方向是基于Retinex理论模型的图像增强算法和基于深度学习的图像增强算法。

Retinex图像增强算法假设如下：

$$S(x,y)=R(x,y)\times L(x,y) \tag{2.5}$$

式中，$S(x,y)$表示原始图像，$R(x,y)$表示反射率图像，$L(x,y)$表示光照图像。其核心思想就是从原始图像$S(x,y)$中估计出光照$L(x,y)$，从而分解出$R(x,y)$,到达消除光照不均匀产生的影响，起到改善图像清晰度的效果。但是估计值依赖于参数选择，而参数选择是手动调整的。因此，基于Retinex理论模型的图像增强算法也存在缺陷，很难达到非常好的效果，但是整体而言，图像增强效果要优于传统的图像增强算法。

对于基于深度学习的图像增强算法，一般是针对低光图像的数据集，通过学习得出相应的神经网络模型，实现了端对端的从低光图像到正常图像的过程。近几年来，出现了许多优秀的神经网络模型，比如：LLCNN、LLNet、Retinex-Net等，它们的泛化能力强效率高，但是对底层硬件的要求比较高，并且在实现弱光增强的时候，并没有实现降噪功能。接下来将介绍近几年比较好的基于Retinex理论模型的图像增强算法和基于深度学习的图像增强算法。

2.3.1 LIME算法

LIME[17]算法是由Guo等人于2017年发表在IEEE上，解决了在弱光或者强光条件下，出现图像对比度比较低的问题，是一种高效并且简单的低光图像增强算法。该算法的基本思路是，首先输入一张低光照的图像，选取红色像素通道中的最大值、绿色像素通道中的最大值和蓝色像素通道中的最大值，然后根据这些最大值来分别估计三个通道上像素的光照情况，并以此为依据来初始化图像的光照图，紧接着通过施加一个结构先验来细化这个初始图像的光照图，最后根据Retinex基础理论模型来增强细化后的图像，生成高质量的图像。本质上，LIME算法是在Retinex理论模型的基础上演变而来，但是与以往的此类算法相比，这种算法仅仅只估计了红、绿、三个像素通道的最大值，这极大了缩短了计算复杂度。本算法的最大贡献是为基于Retinex理论模型的算法提出了一种加速优化的方法，并且也能够很好地处理现实生活中大部分低光图像，同时处理速度也比较快。但是，比较明显的缺点是仅仅只估计了每个像素通道的最大值，经过此算法处理后的图像失真明显，图像的色彩不够完美。

2.3.2 MBLLEN算法

MBLLEN[18]算法是由Lv等人于2018年发表在BMCV上，对简单神经网络或者单分支不能同时进行对比度增强的弊端进行改进，提出了一种多分支的弱光图像增强算法。该算法的基本思路是首先通过卷积神经网络的卷积层采集图像在不同层次下的图像特征，然后使用多个子网同时对采集到的图像特征进行增强，最后将所有分支的增强图像融合成最终的增强图像，达到了从多尺度来提高图像的质

量。网络模型如图2.6所示，由三个部分组成:融合模块(FM)、特征提取模块(FEM)和增强模块(EM)。FEM模块主要由一些的卷积层、单向10层网络、ReLU激活函数等构成，主要作用是提取不同尺度的不同图像特征；EM模块主要由一些反卷积和卷积组成的编解码架构等构成，主要作用是对所有的子网图像从多个维度进行增强处理。FM模块主要由3通道的1×1的卷积构成，主要作用是将EM模块的所有输出图像进行融合,形成最终的增强图像。由于该方法是从多尺度对图像进行增强，因此增强后的细节更加清晰，能够获得更加自然温和的增强效果。但是，在实际处理过程中，很多情况下，可能会出现低光图像会过度曝光的现象，很难在实的场景下应用。

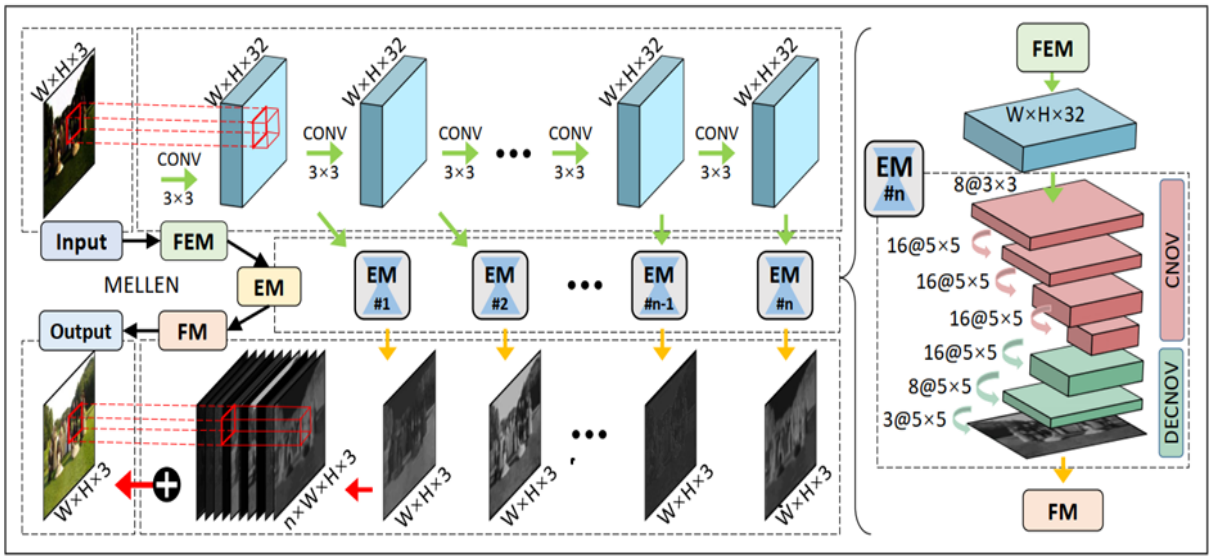


图2.6 MBLLEN网络模型

2.3.3 KinD算法

KinD[19]算法是由Zhang等人于2019年发表在ACM上，针对目前提高弱光图像的暗区会放大图像噪声和颜色失调的现象，提出了一种解决此类问题的算法。该算法的基本思想是首先将原始图像分为了光照图像和反射图像两部分，然后分别使用神经网络对光照图像进行调光，对反射图像进行退化消除，最后提供灵活的映射模型，对图像进行优化，最终来提高图像的质量。网络模型如图2.7所示，由三个部分组成：Decomposition-Net、Adjustment-Net和Restoration-Net。Decomposition-Net借鉴Retinex理论，主要作用是把弱光图像分解为光照图像(illumination maps)和反射图像(reflectance maps)两部分；Adjustment-Net主要由几个卷积层构成，主要作用是通过参数α，灵活调节光照；Restoration-Net主要由带有残差连接的编解码构成，主要作用是将Decomposition network得到的reflectance maps和adjustment network的输出的图像结合起来，形成最终的增强图像。由于，此算法对弱光图像增强后放大的图像噪声和颜色失调有很好的改善效果。因此增强后的图像更加明亮清晰，色彩也基本能够复原。但是美中不足的是，该算法的

运行速度比较慢，达不到实时性对图像进行处理的能力，速度有很大的改进空间。

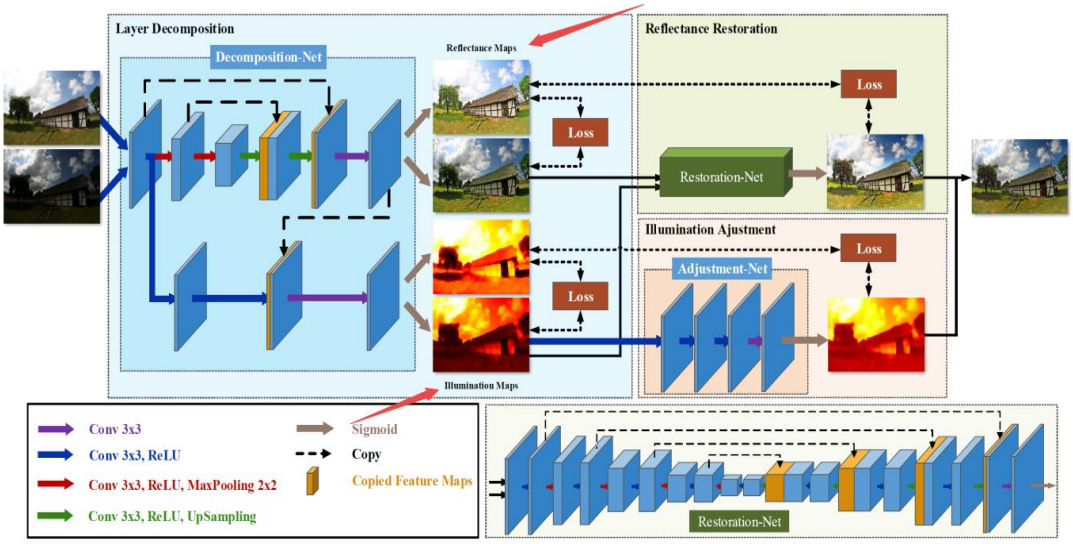


图2.7 KinD算法模型

2.3.4 Zero-DCE算法

Zero-DCE[20]算法是由Guo等人于2020发表在CVPR上，针对当前的绝大部分低光图像增强算法需要使用paired data或者unpaired data作为训练数据的弊端，提出了一种不依赖于配对或者非配对的训练数据的低光图像增强算法。该算法的基本思想是将低光图像增强的问题转换为低光图像作为输入，曲线作为输出的image-specific的问题，这种曲线对输入的图像的像素进行动态调整，并且通过设置一系列的non-reference的损失函数，使得神经网络在没有任何参考图像的情况下能够进行端到端的训练，最终整体上提高图像的质量。

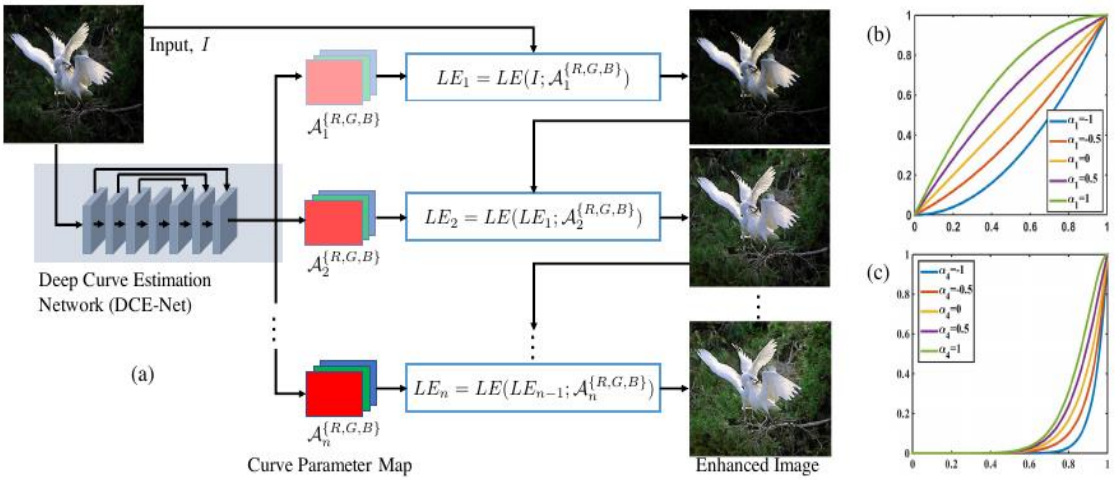


图2.8 Zero-DCE算法模型

网络模型如图2.8所示，由两个部分组成：Deep Cure Estimation Network(DCE-net)和Light-Enhancement Cure(LE-cure)组成。DCE-net主要由对称级联的七个卷积层的CNN和ReLU激活函数等构成，主要作用是根据给定的输入图像来估计一组最

适合的光增强曲线(LE-cure); LE-cure能够通过自适应迭代调整曲线的参数，完全依赖于输入图像，并且每一组的映射曲线简单、可以微分、可以训练。由于此算法不依赖配对或者非配对的训练数据，使用无监督自主学习，运行速度快，能够达到实时性对图像数据进行处理的能力。并且,通过设计一系列巧妙的损失函数，对数据进行训练，不断进行迭代，找到最适合的图像增强曲线。因此，经过此算法对低光图像进行增强后，显示效果明显提高，色彩基本也能够复原。

2.4 图像增强算法对比与选取

本文选取了近几年在低光图像增强领域非常优秀的几种算法，根据每种算法的理论基础，对比了其优缺点，结果如表2.1所示。从表中我们可以看出，与其他的图像增强算法相比，Zero-DCE算法优势比较明显。由于本文的疲劳驾驶检测系统对图像处理速度和图像质量要求都比较高，因此接下来将从实际测试的角度，对图片的实际处理效果和处理速度进行测试，验证本文应该使用哪种算法最为合适。

表2.1 低光图像增强算法对比

方法	优点	缺点
LIME	方法简单有效，能够适应很多不同的场合	增强图像失真明显，图像的色彩不够完美
MBLLEN	多尺度图像增强，增强后细节更加清晰，自然温和	可能会出现过度曝光的现象，很难在实际的场景下应用
KinD	对弱光图像增强后放大的图像噪声和颜色失调有很好的改善效果，色彩也基本能够复原	算法的运行速度比较慢，达不到实时性对图像进行处理的能力
Zero-DCE	无监督自主学习，运行速度快，图像增强后色彩基本能复原，处理效果好	符合本文的应用场景

使用四种算法对同一张低光图像进行处理，处理后的结果如图2.9所示，从图中可以明显看出，使用KinD算法和Zero-DCE算法处理之后的图片效果比较好。在对图片进行处理时，也记录了使用不同算法处理这张照片所消耗的时间，结果如表2.2所示，从表中可以明显看出，使用Zero-DCE算法所消耗的时间最短。从理论分析与实际测试结果都可以得出，Zero-DCE算法更加符合本论文的设计。因此，本文以Zero-DCE算法为基础，完成了驾驶员在夜间驾驶时，对摄像头采集到的图片进行预处理任务。

a)原图

b)LIME处理后

c)MBLLEN处理后

d)KinD处理后

e)Zero-DCE处理后

图2.9 四种图像低光增强算法处理对比图

表2.2 四种算法对同一张照片处理耗时

算法种类	处理上述照片耗时(s)
LIME	0.0321
MBLLEN	0.0124
KinD	0.0049
Zero-DCE	0.0029

2.5 本章小结

本章主要对低光图像增强算法进行了相应的介绍，首先介绍了传统的低光图像增强算法，具体包括：灰度变换、Gamma变换、直方图均衡化等。阐述了其原理以及其弊端，并给出了相应的实验结果。紧接着，本文介绍了近三年来在低光图像增强领域非常优秀的几种算法，并对其进行理论分析和实际测试结果，最终确定了符合本文应用场景的低光图像增强算法。

第3章 驾驶员面部检测与头部姿态估计

在经过第2章对车载摄像头采集的驾驶员面部信息的图片进行低光增强处理之后，图片质量已经得到了明显提升。由于本论文的疲劳驾驶检测方法结合了驾驶员眼睛状态、头部姿态以及嘴巴状态等判断因素，所以需要采用RetinaFace算法获取这些信息。然后使用人脸关键点定位算法准确获取了脸部轮廓、眼睛、嘴巴、鼻子等人脸68个关键区域的坐标信息，为头部姿态估计算法提供相应的坐标数据。

3.1 基于RetinaFace算法的驾驶员人脸检测

RetinaFace[23]算法是由Deng等人于2019年5月份提出来的基于one-stage的人脸检测算法，其处理速度和识别精度处于领先地位，是当前最优秀的人脸检测算法。其网络模型如图3.1所示，主要由主干网络(BackBone)、特征金字塔(FPN)、SSH(Single Stage Headless)、分类(Class Head)、预测框(Box Head)、关键点特征提取(Landmark Head)以及非极大抑制(NMS)等组成。

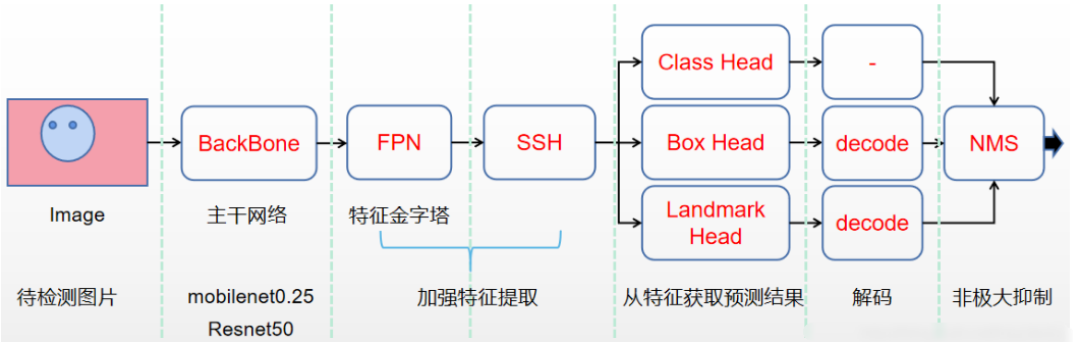


图3.1 RetinaFace算法网络模型

(1)主干网络

RetinaFace算法对图片进行检测时，首先会通过两种主干特征网络对数据进行训练，分别是Resent50和MobilenetV1-0.25。区别在于，Resent50模型精度和识别率更高，但是运行速度慢，对硬件要求更高；而MobilenetV1-0.25模型是谷歌为移动和嵌入式设备提出的一种轻量级的神经网络模型，具有运行速度快，对硬件要求低，可以对视频数据进行实时检测等优点，与普通的卷积相比，其参数数量比较少，计算运行成本比较低。由于本设计需要对驾驶员的面部特性数据进行实时采集，所以应该选择MobilenetV1-0.25模型完成对数据的训练。

(2)特征金字塔网络

特征金字塔(FPN)网络会对主干网络处理后的三个有效特征层进行构建，其构建的基本原理是，首先使用1×1的卷积对三个有效特征层进行通道数调整，然后利

用上采样和特征融合来进行特征加强提取。与图像金字塔和常规卷积操作相比，其主要优点在于在基本不增加原有模型的计算量的基础下，对小物体的检测的性能提升大幅度增加。

(3)SSH网络

SSH(Single Stage Headless)网络进一步加强了对特征层的特征提取。其核心思想是，使用三个并行结构的卷积：第一个是3×3的卷积，第二个是用两次3×3的卷积代替5×5的卷积，第三个是用3次3×3的卷积代替5×5的卷积。总而言之，就是多次使用3×3的卷积堆叠出5×5的卷积和7×7的卷积的效果，其优点是对特征层的特征提取一次循环就可搞定，速度效率更高。

(4)预测结果获取

通过SSH网络进行加强特征提取后，获得了三个有效的特征层：SSH1、SSH2和SSH3。紧接着，我们需要根据这三个特征层获得预测结果。预测结果包含三个，分别是分类预测(Face classification)、框回归预测(Face box regression)、人脸关键点回归预测(Facial landmark regression)。首先通过分类预测判断先验框中是否存在人脸，然后通过框回归预测对先验框进行调整，最后通过人脸关键点回归预测获取先验框中的人脸关键点。

(5)预测结果解码

通过上一步我们获取了三个有效的特征层，这三个有效的特征层把整幅图像划分成了不同大小的网格，然后需要根据编码公式反推出解码公式，完成对预测框和人脸关键点回归预测进行位置调整。

(6)非极大抑制

通过上一步处理后，发现可能会有多个框指向同一个物体。因此，需要使用非极大抑制(NFS)剔除同一个物体中的重合度较高的框，即筛选出一定区域内属于同一中物体最大的框。

综上所述，RetinaFace算法是一种强大优秀的人脸检测算法，人脸检测精度和运行速度都符合本设计要求。结果如图3.2所示。

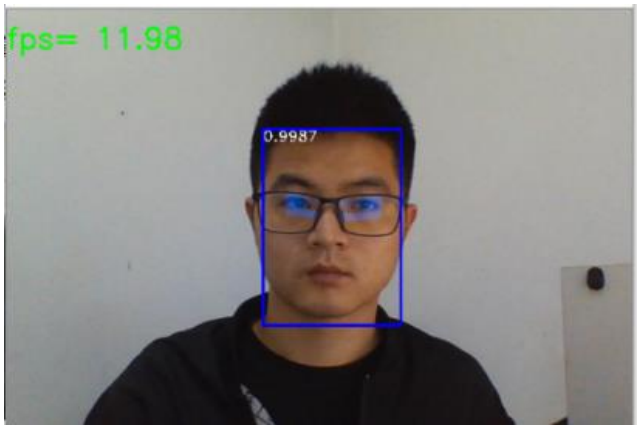


图3.2 人脸检测结果

3.2 人脸关键点定位

人脸关键点定位算法能够准确获取脸部轮廓、眼睛、嘴巴、鼻子等人脸68个关键区域的坐标信息。本文采用了基于回归树(Ensemble of Regression Tress)的人脸关键点定位算法[27]，该方法模型相对较小，占用内存小，运行速度快，检测精准。模型如图3.3所示。

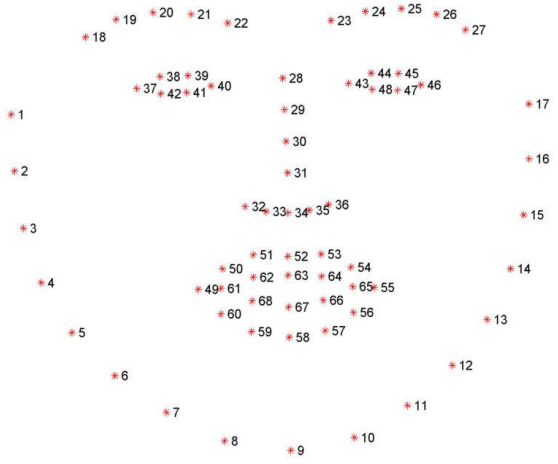


图3.3 人脸关键点定位模型

基于回归树的人脸特征点定位方法与以往的级联形状回归算法有所不同，它主要使用迭代的决策树算法，建立了一个级联的残差回归树模型，即每一个节点里都会有一个预测值，对人脸的形状进行迭代更新得到最准确的人脸面部图像。在迭代的过程中，残差回归量存储在每一个级联的残差回归树上的每一个叶子节点里，当输入落到这个节点上时，就会更新残差值，起到回归的作用，最后加上所有的经过级联的残差回归树上叶子节点的更新值，就能够准确获取脸部轮廓、眼睛、嘴巴、鼻子等人脸68个关键区域的坐标信息。具体算法如下：

(1)级联回归器

假设 Xi 为输入图像 I 的第i个人脸面部关键区域的坐标，向量 $s=(x_1^T, x_2^T, \cdots, x_n^T)^T$ 表示 I 的n个关键区域的坐标，s 表示人脸形状。计算公式如下：

$$\hat{s}^{(t+1)} = \hat{s}^{(t)} + r_t(I, \hat{s}^{(t)}) \tag{3.1}$$

式中，t 表示级联序号，$\hat{s}^{(t)}$ 表示第t级时人脸形状的估计，I 表示人脸特征图像，r_t 表示当前级的回归器，需要根据人脸的脸型 $\hat{s}^{(t)}$ 特征与人脸图像特征 I 来推断出一个新的向量，以此来提高人脸检测的准确率。

(2)各级联回归器的训练

使用梯度增强算法(Gradient Boosting)来对回归器r_t进行训练，这样做可以有效降低目标的真实值与估计值之间差值平方和的均值误差。假使训练的数据集是 (I_1,S_1)，…，(I_n,S_n) 其中 I_i 表示第i张人脸的训练图像，S_i 表示第 i 张人脸脸型图像。回归器初始化计算公式如下：

$$f_0(I,S^{(t)}) = \arg\min \sum_{i=1}^{N} \| DS_i^{(t)} - Y \|^2 \tag{3.2}$$

$$\Delta S_i^{(t)} = S_i - S_i^{(t)} \tag{3.3}$$

求解T级回归器计算公式如下：

$$r_{ik} = DS_i^{(t)} - f_{k-1}(I_i S_i^{(t)}) \tag{3.4}$$

更新$f_k(I_i, S_i^{(t)})$，计算公式如下：

$$f_k(I_i, S_i^{(t)}) = f_{k-1}(I_i, S_i^{(t)}) + v g_k(I, S_i^{(t)}) \tag{3.5}$$

式中，k表示迭代次数，v表示学习效率，$g_k(I, S_i^{(t)})$表示弱回归方程，训练完成之后的回归器为：

$$r_t(I, S^{(t)}) = f_k(I_i, S_i^{(t)}) \tag{3.6}$$

(3)形状不变分割检验

使用两个像素之间的强度差阈值来对回归树上的任意分叉节点进行决策的。但在平均形状的坐标系中定义时，我们假设测试过程中所使用的像素点位于u和v的位置。为了使人脸图像在任意位置和平均形状上的关键点具有相同的位置，需要根据当前的形状来对图像进行一定程度的扭曲变形，过程如下：

假设k_u是平均形状中最吻合u的人脸关键区域的索引指数，并定义其与u的偏移量的补偿为：

$$\delta x_u = u - \bar{X} k_u \tag{3.7}$$

在图像I_i和u中定义相似的位置u'，计算公式如下：

$$u' = \mathrm{x}_{i,ku} + \frac{1}{s_i} R_{\mathrm{i}}^T \delta x_u \tag{3.8}$$

式中，s_i表示S_i经过相似变换到$\bar{s}$的缩放比例，R_i表示S_i经过相似变换到$\bar{s}$的旋转矩阵。最小化计算公式如下：

$$\sum_{j=1}^{v} \| \bar{x}_j - (s_i R_i x_{i,j} + t_i) \|^2 \tag{3.9}$$

形状不变的节点分割计算公式如下：

$$h(I_i, S_i^{(t)}, \theta) = \begin{cases} 0, & I_i(u') - I_i(v') > \partial \\ 1, & \text{其他} \end{cases} \tag{3.10}$$

式中，$\theta = (\partial, u, v)$是决定节点分割的参数，$\partial$是像素点之间的强度差。

(4)特征选择

每一个节点的决策是通过像素的亮度值的差异阈值来决定的，与基于单个像素的强度差阈值相比，它的处理方式更加完美，因为它对全局光照的变化相对不敏感，不需要全局地改变像素相关的强大，测试更加简单。但是，让人遗憾的是，由于是基于一对像素点的，需要通过大量的图像遍历才能够获得比较好的参数

$\theta's$。考虑到图像数据的结构，这一限制可以通过以下指数先验进行优化，其计算公式如下：

$$P(u,v)\infty e^{-\lambda||u-v||} \tag{3.11}$$

式中，λ是比例系数。基于回归树的人脸特征点检测方法对人脸特征点的标记结果如图3.4所示。

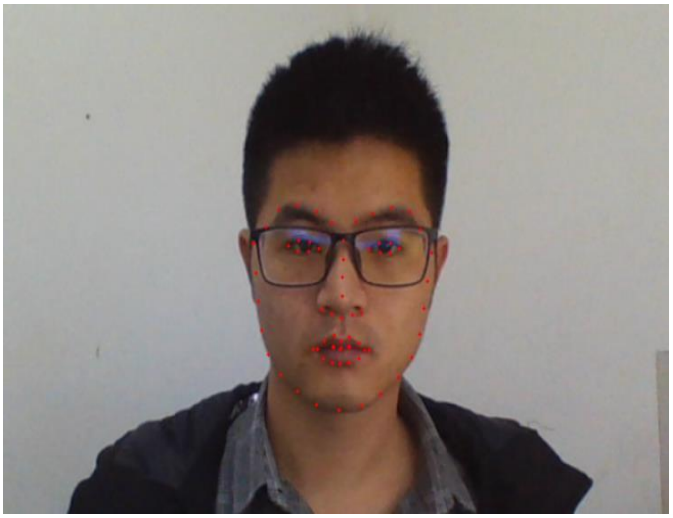


图3.4 人脸关键点检测

3.3 头部姿态估计

当驾驶员处于疲劳驾驶状态时，会出现无规则的、频繁的低头或者抬头以及视线会长时间地偏离正前方等头部异常状态的现象。因此，可以通过根据驾驶员的头部偏转角来判断此时驾驶员是否处于疲劳驾驶状态。

因为在运动的过程中，人的头部的形状和大小都不会发生变化，所以可以把头部假设为一个刚体。根据这个假设，人的头部在三维空间中只要沿着某一个定点进行旋转，不管旋转多少次，只要这个定点没有发生变化，那么其位置都可以使用三维空间中的某一个轴的一次旋转来表示。目前，常见的表示三维空间旋转的方式有很多种，其中欧拉角的表示方式比较简单直观，所以本设计选择欧拉角来表示三维空间的旋转。

对于三维空间中的任意一点，都可以使用三个欧拉角对其进行准确定位，即与x轴相关的俯仰角(pitch)，与y轴相关的偏航角(yaw)和与z轴相关的滚转角(roll)，如图3.6所示。对于绝大部分的正常人而言，其头部各个方向的旋转角度是有限制的。一般情况下，人头部水平左右旋转角度范围大概为-40.9º～36.3º，水平侧向旋转角度的范围大概为-79.8º～75.3º，前后旋转角度范围大概是-60.4º～69.6º。本设计选择了一种比较经典头部姿态估计算法(Head Pose Estimation)，基本步骤是：(1)二维空间的关键点检测；(2)三维空间的人脸匹配模型；(3)求解出三维空间点和对应二维空间点之间的转化关系；(4)求解出欧拉角。二维空间的关键点检测采用的

前文中的基于回归树的人脸关键点检测技术。因此，只需要把二维空间的关键点坐标值对应地映射到三维空间上即可。接下来，将介绍二维空间点如何转化到三维空间上去。

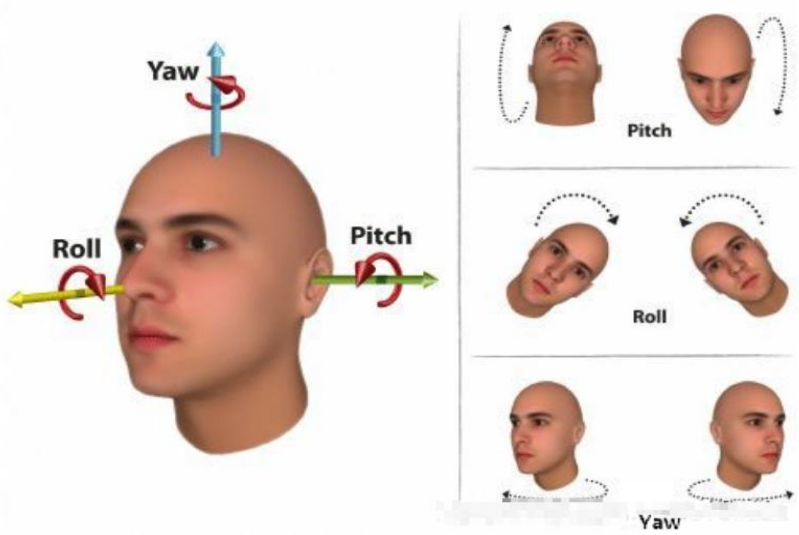


图3.6 头部欧拉角

对于一个物体而言，其相对于相机的位置能够使用平移矩阵(T)和旋转矩阵(R)来描述[32]。对图像进行相应处理时，经常会涉及到四个坐标系的处理：世界坐标系、相机坐标系、图像坐标系和像素坐标系，如图3.7所示。

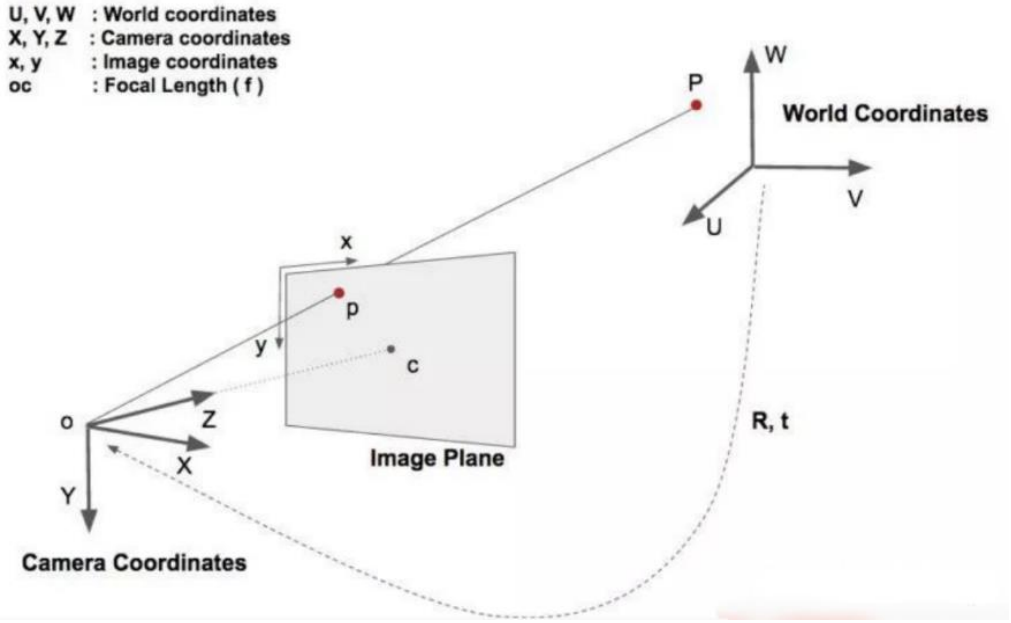


图3.7 不同坐标系

世界坐标系、相机坐标系、图像坐标系和像素坐标系的转换关系的基本原理是，首先对于不同人的头部拟合出对应的世界坐标系的3D模型，然后根据平移矩阵和旋转矩阵将世界坐标系中的3D模型转换成相机坐标系的3D模型，紧接着使用摄像机的焦距、光学中心等固定参数将相机坐标系的3D模型转换成图像坐标系的模型2D，最后通过相机的内参矩阵K完成图像坐标系到图像像素坐标系的转化。

假设我们知道世界坐标系中P点坐标为(Xw,Yw,Zw)，当我们使用摄像头对头部数据进行采集时，可以知道平移矩阵T是3×1的矢量，旋转矩阵R为3×3的矩阵，可以使用以下公式来计算P点在相机坐标系的坐标(Xw,Yw,Zw)：

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = R\begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} + T \rightarrow \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} RT \\ \vec{0}\ 1 \end{bmatrix}\begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{3.13}$$

但知道世界坐标系和相机坐标系足够多的对应点后，可以根据线性代数知识，可以求解出T和R的值。

相机坐标系到图像坐标系的转换，是三维图像到二维图像的转换，可以使用透视投影关系完成，计算公式如下：

$$Z_C\begin{bmatrix} \mathrm{x} \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} \tag{3.14}$$

图像坐标系和像素坐标系都位于成像层面上，但是图像坐标系的单位是毫米，而像素坐标系的单位是像素，它们之间的单位不同，转换公式如下：

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1/d_x & 0 & u_0 \\ 0 & 1/d_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.15}$$

式中，d_x表示每一行代表多少nm，d_{y}表示每一列代表多少nm。联立化简以上公式，就可以知道世界坐标上的点如何转化到像素坐标系上，转换公式如下：

$$Z_C\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} R & T \\ \vec{0} & 1 \end{bmatrix}\begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{3.16}$$

式中，等式后第一个矩阵是相机内参数，即相机焦距、像素大小等；第二个矩阵是相机外参数，即相机位置、旋转方向等。由以上公式可知，只需要知道图片中某个点在世界坐标系与像素坐标系中的位置，以及相机内参数矩阵就可以求出平移矩阵和旋转矩阵。其中，相机内参数矩阵可以通过标定得到。得到旋转矩阵之后，就可以求得欧拉角，欧拉角计算公式如下：

$$R = \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{pmatrix} \tag{3.17}$$

$$\begin{cases} \Psi = a\tan(-r_{12}, r_{22}) \\ \phi = a\tan(r_{02}, \sqrt{r_{11}^2 + r_{22}^2}) \\ \varphi = a\tan(-r_{01}, r_{00}) \end{cases} \tag{3.18}$$

式中，R表示旋转矩阵，Ψ表示滚转角，ϕ表示偏航角，φ表示俯仰角。

3.4 实验设计与结果分析

3.4.1 实验数据处理与平台搭建

本文使用的数据集是以YawDD视频数据集为基础，每隔一定时间对视频中光线较暗的情况下的驾驶员照片进行采集，分别采集了600张光线较暗情况的图像和600张正常光线情况下的图像。由于YawDD没有光线过暗情况下的图像，因此，自制采集了600张过暗光线下的图片。所使用的测试平台配置如表3.2所示，使用版本为3.7python程序设计语言、版本为1.2的torch深度学习框架以及pycharm的编辑器对本章的算法进行模拟测试。本章算法逻辑框图如图3.8所示。

表3.2 测试平台参数

类型	型号	参数
操作系统	Windows	Windows10教育版
CPU	Inter(R) Core(TM)i5-6300HQ	4核
显卡	NVDIA GeForce GT 950	4GB
内存	DDR4	8GB

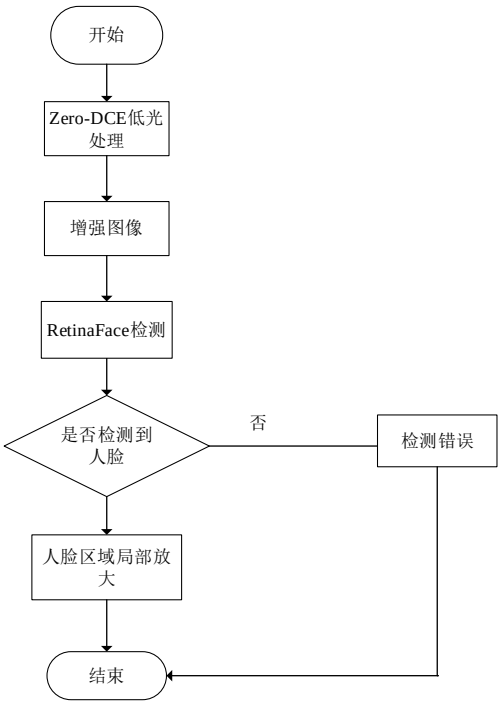


图3.8 本章检测算法逻辑框图

3.4.2 人脸检测结果与分析

为了验证使用第二章的低光图像增强算法是否对人脸检测效果的提升作用，首先对原始图像进行Zero-DCE低光增强算法处理，然后在使用基于RetinaFace的人脸检测算法完成对人脸区域的跟踪检测与局部放大。然后，与仅仅只使用RetinaFace人脸检测算法进行相应对比，对比结果如果3.9所示。从图中，可以明显看到，经过Zero-DCE低光增强算法对低光图像进行处理之后，人脸的眼睛、嘴巴等关键点特征更加明显，能够有效提升我们对驾驶员是否睁眼、打哈欠的判断精度。最后，为了验证经过低光增强处理后对人脸识别的准确度的提升效果，对原始数据集和先经过低光增强处理后的数据集分别进行人脸检测定位测试，结果如表3.1所示。根据表3.1的结果可知，测试数据集经过Zero-DCE低光增强算法处理之后，在暗光和弱光条件下人脸检测准确率有明显的提高。

a)原图

b)低光图像处理

c)直接RetinaFace人脸检测

d)低光处理后人脸检测

图3.9 图像处理与人脸检测

表3.1 原始数据与低光处理后数据在不同环境下人脸检测效果

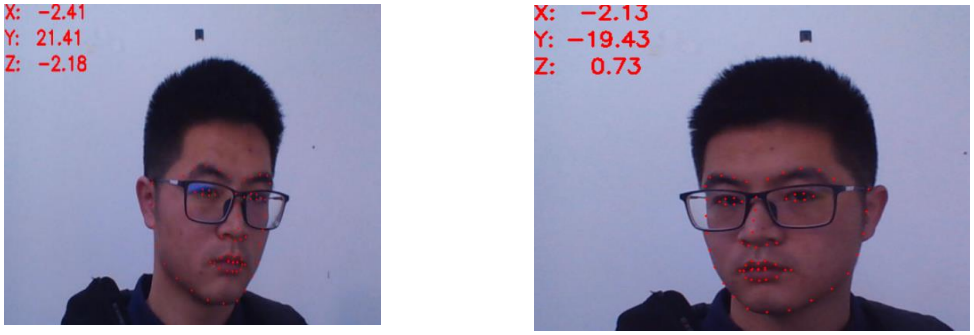
数据样本种类	光照环境	样本数量	人脸检测到数量	准确率
原始数据	暗光	600	539	89.7%
	弱光	600	563	93.8%
	正常光	600	589	98.1%
低光处理数据	暗光	600	575	95.8%
	弱光	600	580	96.7%
	正常光	600	592	98.7%

3.4.3 头部姿态估计结果与分析

当驾驶员出现疲劳驾驶时，可能会出现打瞌睡的现象，此时头部姿态变化会比较大，会出现频繁点头或者头部倾斜的动作。因此，可以根据一般人打瞌睡的表现来对疲劳驾驶进行预估，其判断依据主要集中在俯仰角和滚转角的角度变化。根据生活常识，驾驶员在打瞌睡时，头部的水平角度基本不会有动作产生。因此，头部姿态估计不考虑偏航角的角度变化。测试结果如图3.10所示，图a)展示头部俯仰角变化，当头部向上昂起时，角度为5.94°，当头部向下低时，角度为-18.27°，偏航角和滚转角的角度基本不变；图b)展示了头部在滚转角的变化，当头部向左下倾斜时，角度为12.30°，当头部向右下倾斜时，角度为10.41°，偏航角和俯仰角的角度基本不变；图c)展示了头部在偏航角的变化，当头部水平向左移动时，角度为21.41°，当头部水平向右移动时，角度为-19.43°，俯仰角和滚转角的角度基本不变。

a)俯仰角姿态测试

b)滚转角姿态测试



c)偏航角姿态测试

图3.10 头部姿态估计

3.5 本章总结

本章主要实现了对驾驶员人脸的检测与提取和驾驶员头部姿态估计。首先介绍基于RetinaFace的人脸检测算法，完成对驾驶员的人脸检测；紧接着介绍人脸关键点定位算法，然后使用这种算法准确获取脸部轮廓、眼睛、嘴巴、鼻子等人脸68个关键区域的坐标信息；然后介绍头部姿态估计所涉及的理论知识；最后使用以YawDD视频数据集为基础自制的数据集完成人脸检测的结果分析，实验结果表明，本文提出的算法在弱光条件下对驾驶员人脸检测准确率以及面部特征提取效果都比较好，紧接着根据人脸关键点坐标完成对驾驶员的头部姿态估计，来预测驾驶员是否疲劳驾驶。

第4章 基于卷积神经网络的眼部与嘴部状态识别

近几年来，伴随着计算机硬件与互联网技术的蓬勃发展，在日常生活中已经有许多深度学习技术应用的实际案例，比如人脸识别、无人驾驶、无人超市、语音识别、智能家居等。在深度学习领域中，卷积神经网络模型起着举足轻重的作用。因此，本章首先介绍了卷积神经网络的原理，然后结合本文的实验工况，对SSD卷积神经网络进行适当改进，紧接着对本文的卷积神经网络进行测试训练，得出合适的模型，最后对第3章提取到的驾驶员人脸图像的眼睛状态和嘴巴状态进行识别。与人工特征提取方法对眼睛状态与嘴巴状态判断方法相比，此方法自动完成对眼睛特征和嘴巴特征的学习，能够有效降低人工特征提取所产生的误差。

4.1 卷积神经网络概述

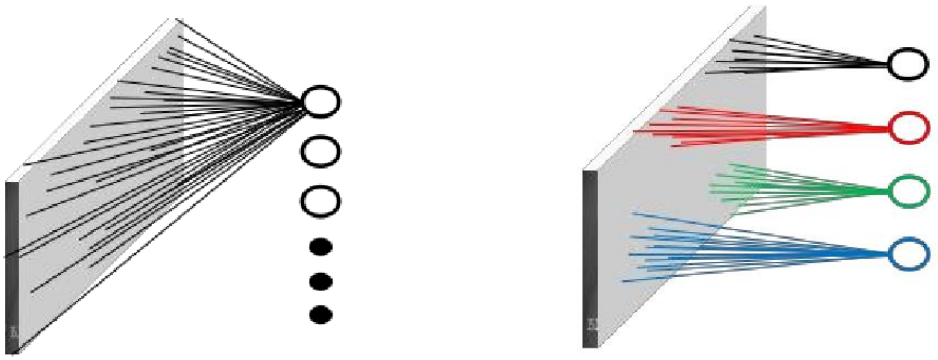
卷积神经网络模型[38](Convolutional Neural Network，CNN)是在1998年由纽约大学的Yann LeCun提出来的。由于其算法计算量偏大，受限于当时的计算机硬件水平，真正地广泛应用于图像识别、目标检测与跟踪、视频和图像字幕等领域是近十年才开始的。与其他的神经网络模型相比，卷积神经网络模型得到广泛应用的原因在于其采用了局部连接(Local Connectivity)和权值共享(Parameter Sharing)的方式。并且与传统的神经网络相比，卷积神经网络模型没有复杂的特征提取与数据重建的过程，可以直接使用图像作为网络模型的输入，能够直接提取纹理、颜色、形状等信息，在二维图像的处理过程之中具有显著的优势。接下来将详细介绍局部连接和权值共享。

4.1.1 局部连接

一般来讲，卷积神经网络中的局部连接指的是每个神经元只和输入神经元的一部分进行连接，而不是和输入神经元所有部分进行连接。理论依据来源是Hubel和Wiesel对大脑处理数据的机制研究，其研究结果表明：人们的视觉系统在感知周围事物时，不同的细胞组织对不同的输入部分具有不同的感受能力，比如某些细胞对事物的形状特征数据感兴趣但是对颜色数据不感兴趣。其次，由于图像像素在空间上的联系是与像素之间的距离密切相关。因此，与人们的视觉系统进行类比，没有必要把每个神经元同上一层的所有神经元进行连接，只需要与上一层的局部神经元进行连接即可。

全连接和局部连接的原理图如图4.1所示。假设图像的像素是1000×1000，神经元的数目为10^4。如果使用图4.1左图所示的全连接层进行数据训练，则需要的参数

数量为$1000\times1000\times10^4=10^{10}$；但是由于图像像素的空间联系是局部的，可以使用图4.1右图所示的局部连接成进行数据训练，假设图像的每个卷积层视觉感受区域只需要和10×10的局部图像进行连接，则需要的参数数量为$10\times10\times10^4=10^6$。从计算结果可以看出，全连接的参数时局部连接参数的4倍。因此，与全连接相比，局部连接能够显著减少参数的训练量，提高模型的训练效率与网络的学习适应能力。



a)全连接　　b)局部连接

图4.1 全连接与局部连接神经网络

4.1.2 权值共享

卷积神经网络的训练参数经过局部连接后，参数数量在很大程度上得到了减少，但是为了处理大规模数据与满足实时性要求，需要进一步减少模型的训练参数。因此，对于每一层的卷积核我们可以让它具有相同的权重矩阵和偏置项，即权值共享。原理如图4.2所示，相应计算公式如下：

$$h1=w\times x[1:3] \tag{4.1}$$

$$h2=w\times x[3:5] \tag{4.2}$$

$$h3=w\times x[5:7] \tag{4.3}$$

式中，x表示输入层，$x=[x1,x2,x3,x4,x5,x6,x7]$；h表示隐藏层，$h=[h1,h2,h3]$；w表示权重向量，$w=[w1,w2,w3]=[1,0,-1]$，权重h被$h1$，$h2$，$h3$共享。因此，权值共享能够进一步减少训练参数，并且可以检测到图像上不同位置同一类型的特征。

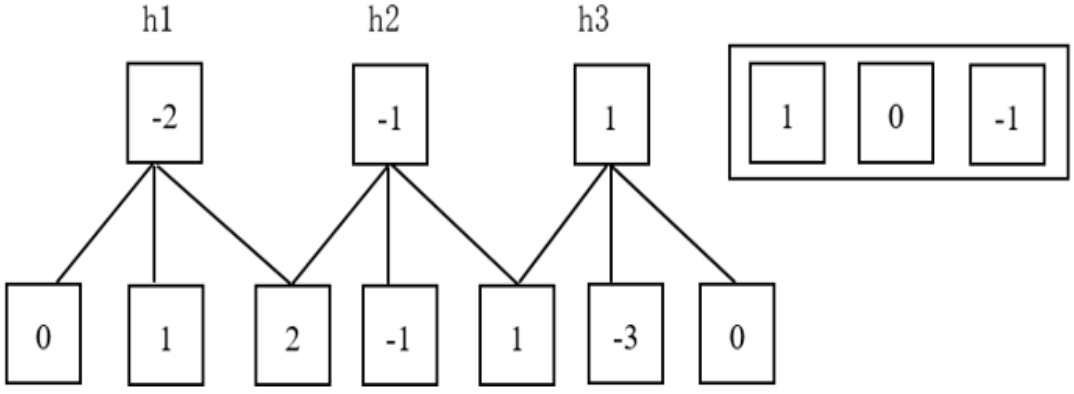


图4.2 权值共享原理图

4.2 卷积神经网络的组成

如图4.3所示，卷积神经网络由卷积层、激活函数层、池化层、全连接分类层等组成。接下来将详细介绍卷积神经网络的每一层结构。

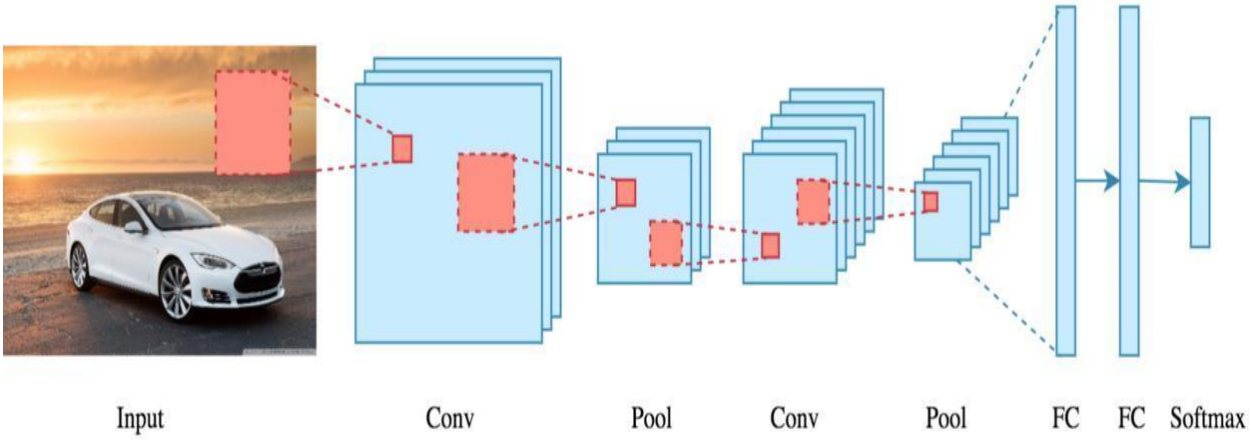


图4.3 卷积神经网络

(1)卷积层

卷积层主要目的是从输入图像中提取特征并且能够在不同层上提取不同的特征，主要作用是减少网络模型的训练参数。它是由若干个卷积核构成的，不同的卷积核能够完成不同的特征提取功能，比如锐化、检测竖边、检测周边、模糊等。卷积核本质上就是一个多维数组，根据不同的图像特征提取需要，可以设置不同的多维数组。通常上，多维数组维数一般设置为1×1、3×3、5×5和7×7等较小的奇数。如图4.4所示，一个3×3卷积核的工作原理是：卷积核在原图上进行滑动，每次向右或者向下移动一个单元，即滑动的步长(stride)设置为1，每次滑动，都可以提取图像的一小块区域特征。3×3的数组中的数字可以事先设定，也可以通过反向传播算法学习(back propagation learning)。

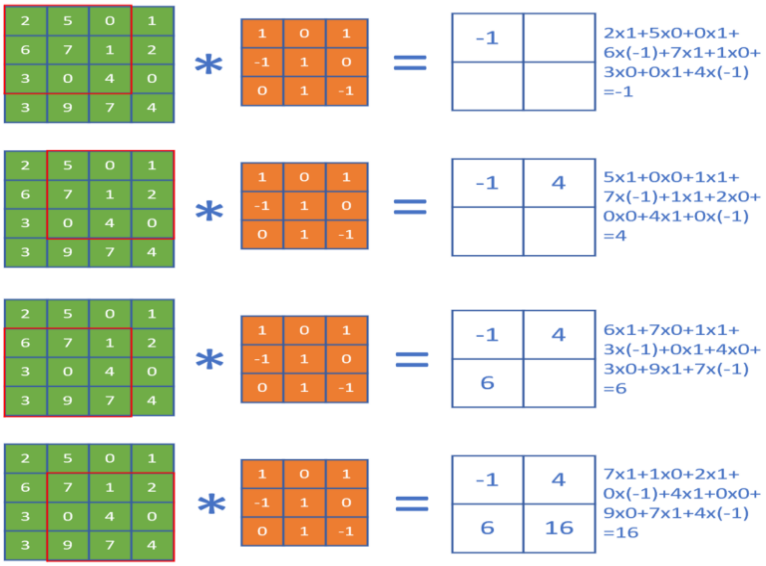


图4.4 3×3卷积核工作原理

相应计算公式如下：

$$a_{i,j} = f(\sum_{m=0}^{2}\sum_{n=0}^{2} w_{m,n}x_{i+m,j+n} + w_b) \tag{4.4}$$

式中，$w_{m,n}$表示权重矩阵，w_b表示偏置列向量，X表示像素值，f表示激活函数，$a_{i,j}$表示计算后得到的像素值。并且从图中可以看出，在对输入图像进行卷积操作时，存在一个明显的缺陷，即图像的边缘的像素点使用次数较少，在图像识别中边缘信息会弱化，使预测精度减低。因此，在对图像进行卷积时，会先对图像加上灰条，即在图像的原始数据周围填充n层数据。

(2)激活函数

激活函数主要作用是当使用神经网络模型处理音频、图像、语音等复杂数据时，仅仅依靠线性回归模型并不能够解决此类复杂问题，需要借助激活函数加入非线性因素。比如在处理图4.5所示的分类问题，如果仅仅使用线性函数，那么不管怎么样也不能很好把图中的三角形和圆形区分开来，只能使用图中红线所示的曲线才能把它们区分开来，即对于复杂的神经网络模型，必须添加激活函数来增加模型的泛化能力。在卷积神经网络中，常见的激活函数包括两大类，一类是挤压性激活函数，另一类是半线性激活函数。挤压性激活函数的特点是，当输入值域绝对值较大时，其输出在两端是饱和的，都具有S形的函数曲线以及压缩输入值域的作用，常用于简单的神经网络模型，最具代表性的是 Sigmoid函数。半线性激活函数的特点是，能够使用线性修正和正则化对神经元的活跃程度(即输出正值)进行相应的更改调试，常用于深度神经网络模型，具体包括ReLU函数、Leaky ReLU函数和softplus函数。本文主要介绍常用的Sigmoid函数与ReLU函数。

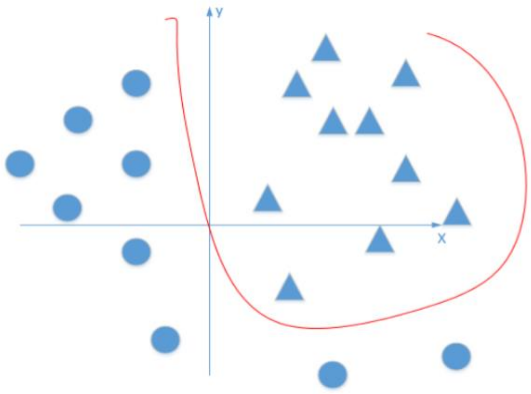


图4.5 圆形与三角形分类

Sigmoid函数的计算公式如下：

$$a = \frac{1}{1+e^{-z}} \tag{4.5}$$

函数图像如图4.6所示，从图中可以看出，当输入值大于6时，都会被压缩到1；当输入值小于-6时，都会被压缩到0。这样会导致一个比较严重的问题，就是会出

现梯度饱和的现象。从图中又可以看出，当输入值大于6或者小于-6，输出部分接近0，此时整个网络将无法训练。

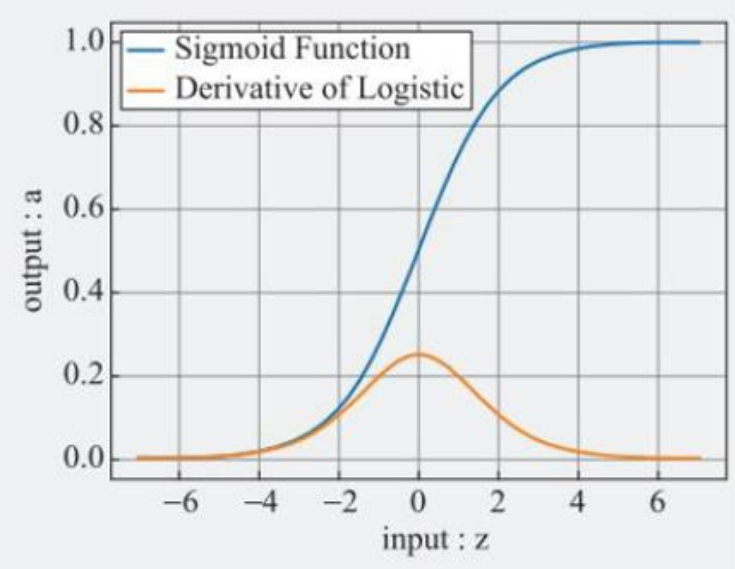


图4.6 Sigmoid函数和梯度函数

因此，为了解决当自变量进入某个区间后，梯度的变化会变得非常小的问题，人们把半线性激活函数引入到神经网络中。接下来将重点介绍*ReLU*半线性激活函数，计算公式如下：

$$\mathrm{Re}\,LU(z)=\max(0,z)=\begin{cases}1, z\geq 0\\ 0, z<0\end{cases} \tag{4.6}$$

函数图像如图4.7所示，从图中可以看出，当$z\geq 0$时，函数梯度为1；当$z<0$时，函数梯度为0。因此，*ReLU*函数能够避免梯度饱和现象的出现。

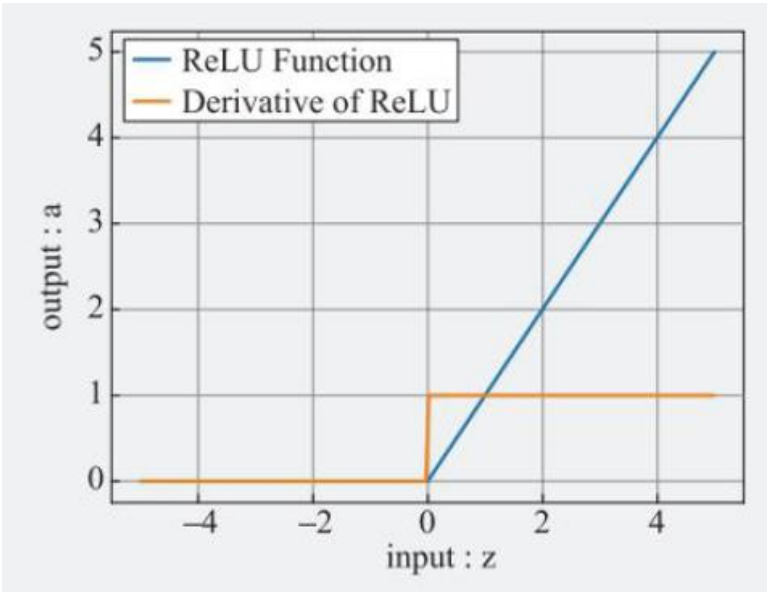


图4.7 *ReLU*函数和梯度函数

(3)池化层

池化层的作用主要是在保留图像的局部特征的前提下，压缩图像的数据和参

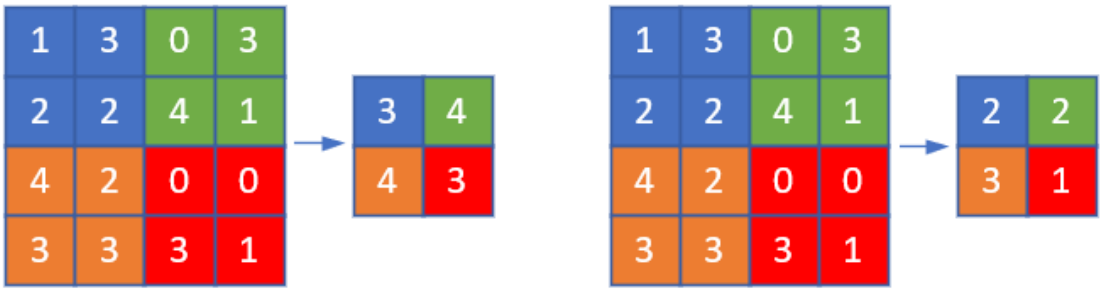
数的量，减小过拟合。常见有最大值池化和均值池化。最大值池化是取当前池化视野中所有元素的最大值，输出到一下层特征图中，其优点是它能够有效保存图像的边缘和纹理特征，提取最明显的特征，计算公式如下：

$$Y_{m,n}^{d} = \max_{i \subseteq R_{m,n}^{d}} x_i \tag{4.7}$$

式中，x表示像素值，$Y_{m,n}^{d}$表示处理后某一范围的像素值。均值池化是取当前视野中所有元素的平均值，输出到下一层特征图中，优点是可以减少均值估计的偏移，提高模型的鲁棒性，计算公式如下：

$$Y_{m,n}^{d} = \frac{1}{|R_{m,n}^{d}|} \sum_{i \subseteq R_{m,n}^{d}} x_i \tag{4.8}$$

原理如图4.8所示。



a) 最大池化　　b) 平均池化

图4.8 最大值池化和均值池化

(4)全连接层

全连接层通俗上讲就是将图像经过卷积层、激活函数、池化层处理后获得的若干特征层连接成一个向量，在实际的使用过程中，可以使用操作实现，主要是起到分类的作用。简单来说，就是当要去识别某个物体时，物体所在的位置并不重要，因为同一物体在不同的位置其特征层的值是相同的，因此只需要把这些特征层的值整合成一个值，值越大，就说明这个物体存在的可能性越高；值越小，就说明这个物体存在的可能性越低。因此，全连接层极大的提高了神经网络的鲁棒性。但是，在实际使用的过程中，全连接层参数复杂冗余，降低了模型的运行效率。所以，为了提供卷积神经网络的整体效率，近些年来，科研人员使用全局平均池化(global average pooling)替代全连接层，使用这种结构的卷积神经网络模型的典型代表有GoogLeNet和ResNet。在实际测试情况下，这种模型的预测效果性能也很不错，并且速度得到了很大的提升。因此，这也是未来卷积神经网络研究的热点。

4.3 卷积神经网络的训练

卷积神经网络的训练有前向传播过程和反向传播过程。前向传播过程指的是输入图像的向量经过卷积层、池化层等处理后得到目标输出，如果目标输出的结果与我们的期望结果相符，则输出，否则进入反向传播过程。反向传播过程指的是当目标输出的结果与我们的期望结果不相符时，我们需要计算出网络层中神经元的误差，然后把误差传回网络中，求出误差梯度后对权值进行更新。训练过程如图4.9所示。接下来将详细介绍与权值更新有关的反向传播算法与梯度下降算法。

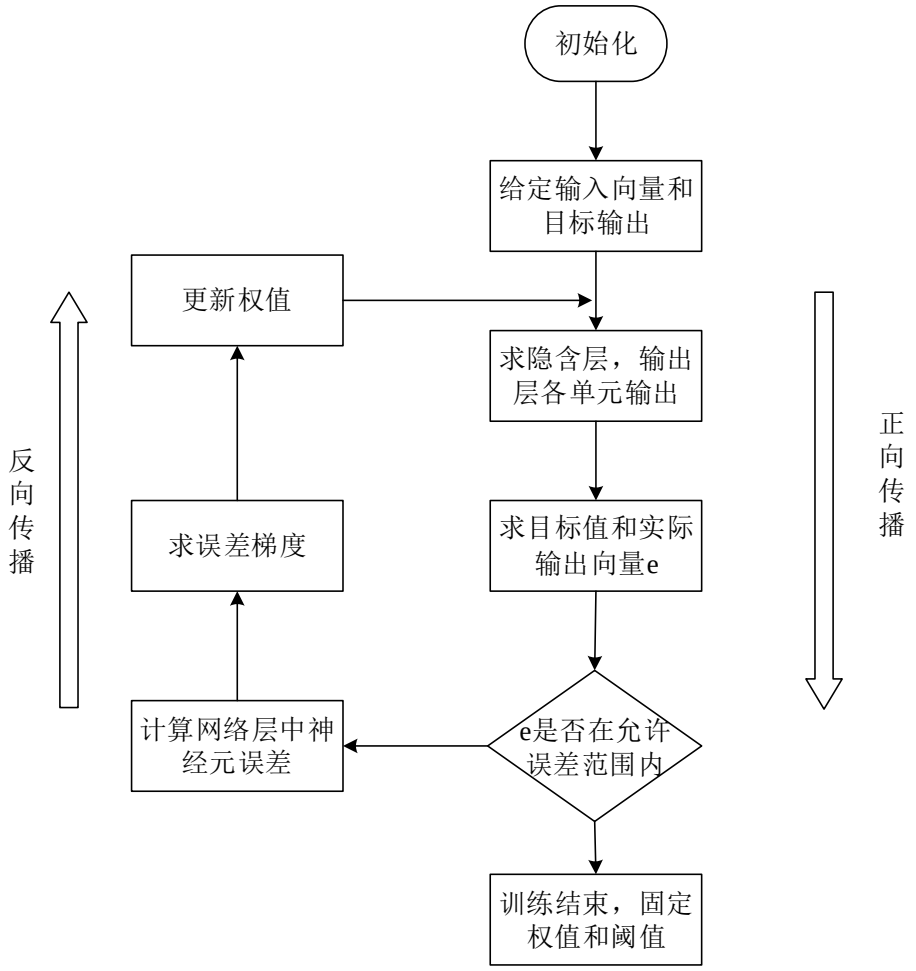


图4.9 卷积神经网络训练过程

(1)反向传播算法

反向传播算法指的是当目标输出的结果与我们的期望结果不相符时，我们需要计算出网络层中每一层神经元的误差，然后把误差传回网络中，求出误差梯度后对每一层的权值进行更新。首先依据卷积神经网络的 δ 误差是损失函数对当前层未激活输出 z^l 的倒数定义，紧接着假设考虑的是二维卷积并且在已经知道了 l+1 层的误差，通过微积分的求导法则，我们能够求出第 l 层的 δ 误差，计算公式如下：

$$\delta^l(x,y)=\frac{\partial C}{\partial z^l(x,y)}=\sum_{x'}\sum_{y'}\frac{\partial C}{\partial z^{l+1}(x',y')}\frac{\partial z^{l+1}(x',y')}{\partial z^l(x,y)}=\sum_{x'}\sum_{y'}\delta^{l+1}(x',y')\frac{\partial z^{l+1}(x',y')}{\partial z^l(x,y)} \quad (4.9)$$

式中，$\delta^l(x,y)$ 表示 l 层的 δ 误差，$z^l(x,y)$ 表示 l 层经过激活函数之前的输出，$z^{l+1}(x',y')$ 表示 l+1 层经过激活函数之前的输出,坐标 (x',y') 表示第 l+1 层中前向传播

中受到第l层坐标(x,y)影响的点，因为这些点的不止一个，所以需要把这些点求和累加。紧接着，根据前向传播的关系式:

$$z^{l+1}(x',y')=\sum_{a}\sum_{b}\sigma(z^{l}(x'+a,y'+b))w(a,b)+b^{l+1} \tag{4.10}$$

式中，w表示权重矩阵，b表示偏置列向量。将上述表达式进一步展开：

$$\delta^{l}(x,y)=\sum_{x'}\sum_{y'}\delta^{l+1}(x',y')\frac{\sum_{a}\sum_{b}\sigma(z^{l}(x'+a,y'+b))w(a,b)+b^{l+1}}{\partial z^{l}(x,y)} \tag{4.11}$$

进一步化简得：

$$\delta^{l}(x,y)=\sum_{x'}\sum_{y'}\delta^{l+1}(x',y')w(a,b)\sigma'(z^{l}(x,y)) \tag{4.12}$$

可以得出两个条件$x'+a=x$与$y'+a=y$，将这个条件带入到上述公式中，可以得到下面计算式子：

$$\delta^{l}(x,y)=\sum_{a}\sum_{b}\delta^{l+1}(x-a,y-b)w(a,b)\sigma'(z^{l}(x,y)) \tag{4.13}$$

在令$a'=-a$，$b'=-b$，最终可以得出结论：

$$\delta^{l}=\delta^{l+1}*ROT180(w^{l+1})\otimes\sigma'(z^{l}) \tag{4.14}$$

式中，$ROT180$表示矩阵旋转180°，⊗表示Hadmardche乘积，即逐元素相乘。因此，可以根据l+1层的δ误差推算出l层的δ误差。紧接着，在已知第l层δ误差的情况下，根据卷积神经网络的δ误差，得到下面的表达式：

$$\frac{\partial C}{\partial w^{l}}=\delta^{l}\times\sigma(z^{l-1}) \tag{4.15}$$

$$\frac{\partial C}{\partial b^{l}}=\sum_{x}\sum_{y}\delta^{l} \tag{4.16}$$

式中,C表示误差函数。然后根据梯度下降对参数进行更新，得到下面的表达式：

$$\mathrm{w}^{l}=\mathrm{w}^{l}-\frac{\eta}{\mathrm{batch\_}size}\sum\frac{\partial C}{\partial \mathrm{w}^{l}} \tag{4.17}$$

$$\mathrm{b}^{l}=b^{l}-\frac{\eta}{\mathrm{batch\_}size}\sum\frac{\partial C}{\partial b^{l}} \tag{4.18}$$

式中，batch_size表示批处理大小，η表示学习率。因此，可以根据(4.17)和(4.18)对参数w和b进行更新，重复上面的过程，逐步减少代价函数，当减少到一定数值之后，w和b的值会达到稳定，即停止更新，此时卷积神经网络也就可以求解出来。

(2)梯度下降算法

根据上文的方向传播算法公式推导，已经能够得到权重矩阵与偏置向量的计

算公式，此时需要使用梯度下降算法快速找到权重矩阵和偏置向量的局部最小值，来快速优化模型。根据数学定义，梯度的方向就是函数变化率最大和函数值增长最大的方向，所以当我们需要计算出函数的一个最小值，只需要沿着梯度的反方向进行寻找即可。

接下来，以函数表达式为 $f(x_1,x_2)$，函数图像如图4.10所示的二维图像为例，演示梯度下降算法的过程与原理。图中红色区域做标记的地方表示起始位置，黑色做标记的线表示梯度逐渐下降的过程。此函数在 x_0 处的泰勒展开式为：

$$f(x_0)\approx f(x_0)+\nabla f(x_0)(x-x_0) \tag{4.19}$$

进一步，可以根据负梯度-∇f 的向量方向，推出从 x_0 处迭代到 x_1 处的计算公式：

$$\frac{x_1-x_0}{|x_1-x_0|}=-\frac{\nabla f(x_0)}{|\nabla f(x_0)|} \tag{4.20}$$

对表达式进行调整优化，得到公式：

$$x_1-x_0=-\frac{\nabla f(x_0)}{|\nabla f(x_0)|} \tag{4.21}$$

进一步优化得：

$$x_1=x_0-\frac{\nabla f(x_0)}{|\nabla f(x_0)|}|x_1-x_0| \tag{4.22}$$

最后，令 $\lambda=\frac{|x_1-x_0|}{|\nabla f(x_0)|}$，整个迭代的关系式就可以简化成下面的公式：

$$x_1=x_0-\nabla f(x_0)\lambda \tag{4.23}$$

式中，λ 就是我们说的学习率，$|x_1-x_0|$ 就是步长，$|\nabla f(x_0)|$ 就是梯度向量模。在梯度下降公式中，学习率和步长是人为设置的变量，其合理性关乎是否可以找到最合适的损失函数。如图4.11所示，步长过大，始终落不到最低点；步长过小，收敛速度很慢。

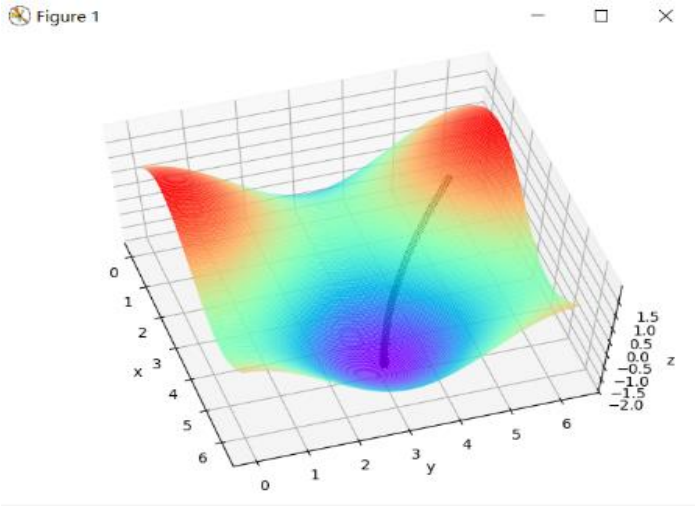
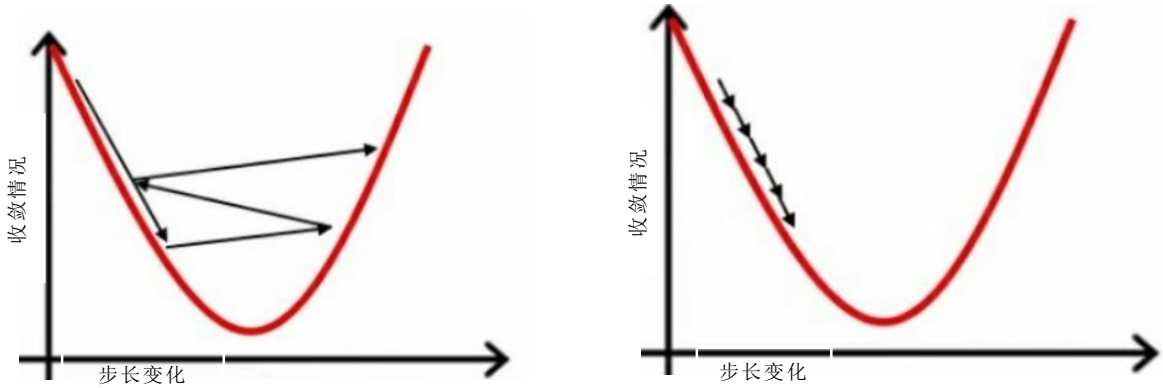


图4.10 梯度下降算法



a) 步长过大　　b) 步长过小

图4.11 梯度下降算法与步长的关系

4.4 卷积神经网络模型的搭建

在第3章提取到驾驶员的人脸图像后，需要辨别驾驶员的眼睛状态和嘴巴状态，然后根据这些特征来判断驾驶员是否疲劳驾驶。目前，卷积神经网络模型主要分为非端到端模型和端到端模型。非端到端模型通俗来讲就是先在图像中提取可能含有目标的候选框，然后将这些候选框输入到主干神经网络中，让主干神经网络判断候选框中是否真正的存在物体，适用于人脸识别领域。端到端模型通俗来讲就是输入原始图像，最后输出结果，这种方法能够缩减了人工预处理所需要的时间，增加了模型的整体契合度。对于本文情况，已经在图片中提取了人脸区域，能够确保眼部和嘴部区域的存在。因此，本网络模型应该设计成一种端到端的网络模型。

由上文介绍可知，卷积神经网络模型必须包含的模块有：卷积层、激活函数、池化层和全连接层。由于全连接参数冗余，降低了模型的运行效率，因此可以通过使用其他方法替换全连接层。本文借鉴2016年由Wei等人提出SSD[40]算法，这种算法是以YOLO[41]和fast R-CNN[42]为基础，继承了fast R-CNN准确和YOLO速度快的特点，使用回归预测和分类预测的特征检测网络替代全连接层，减少整个卷积神经网络的参数，提高运行速度。SSD网络框架如图4.12所示，从图中可以看出，其大致可以分为两部分：一部分是主干神经网络部分，完成对图像的特征提取；另一部分是多尺度特征检测网络部分，使用池化操作缩小主干网络提取的特征图，紧接着根据不同的卷积层在不同层上能够提取不同的特征图性质，然后利用这些特征图的差异性将预测物体进行分类操作以及对目标物体的边界框位置进行适当移动。进一步来讲，SSD算法就是把主干网络提取的特征层，划分成38×38、19×19、10×10、5×5、3×3和1×1的网格，然后以每一个网格为中心画出若干个先验框，之后根据先验框里面有没有需要的物体，有的话就把它标记出来。但是有些框是重合的，需要使用非极大抑制的方法找到需要的框，并标出需要的种类。

从上述原理分析可知，SSD目标检测算法由于事先并不知道图片所含有物体的大小、数量和类型等信息，所以为了提高模型的精度，主干神经网络的层数比较多，先验框的数量划分也比较精细，数量达到了8732个。而本文的应用场景是比较单一，输入的图片恒为人脸图片，所以此模型对本应用场景过于复杂，应对其进行相应更改和优化，缩减主干特征提取网络的层数和缩减多尺度特征检测网络的划分的网格大小和数量，以适应本文章需求。

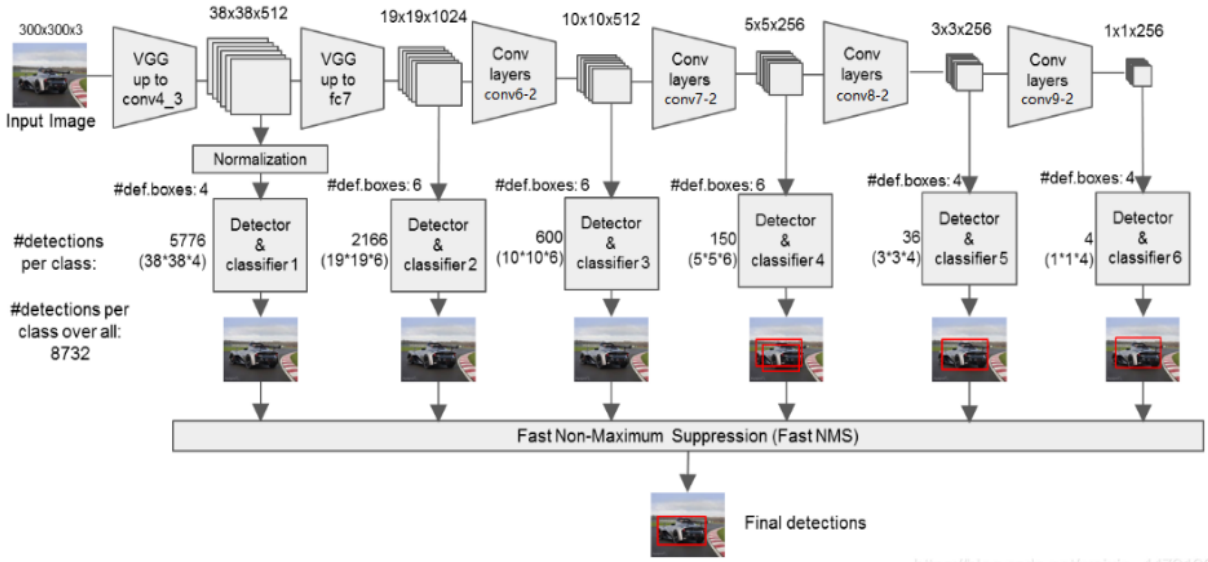


图4.12 SSD网络框架

在真实的驾驶环境中，摄像头所采集的驾驶员的图片尺寸会有所不同，所以在第3章我们对图片进行了相应的预处理，传入本文神经网络模型的图片是300×300大小驾驶员的人脸图像。通过测试，我们发现把人脸图像划分成8×8的网格能够比较好的包含眼部和嘴部区域，如图4.13所示。由于不同的人的面部、眼睛和嘴巴区域的大小都有所差别，并且驾驶员在驾驶时，距离摄像头的距离远近可能也会有所差别。因此，为了提高本网络模型的鲁棒性，应适当的扩大或者缩小网格的大小。综上，本文最终把主干特征提取网络提取到的特征划分为10×10、8×8和5×5的网格大小。具体的网络模型结构如图4.14所示。

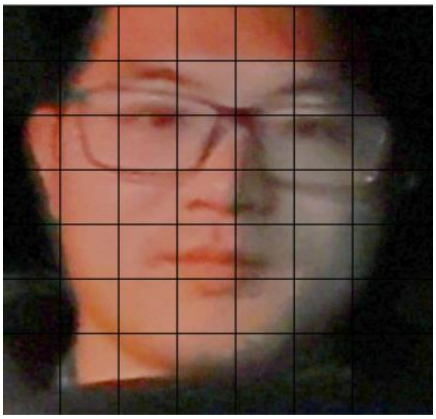


图4.13 人脸网格切分

从图4.8可知，输入图像的尺寸为300×300×3，工作流程是：(1)经过两次3×3的卷积，输出结果为300×300×64，在通过2×2的最大池化层对数据进行压缩后，输出结果为150×150×64；(2)经过两次3×3的卷积，输出结果为150×150×128，在通过2×2的最大池化层对数据进行压缩后，输出结果为75×75×128；(3)经过三次3×3的卷积，输出结果为75×75×256，在通过2×2的最大池化层对数据进行压缩后，输出结果为38×38×256；(4)经过三次3×3的卷积，输出结果为38×38×512，在经过2×2的最大池化，输出的结果为19×19×512，并把这些特征层划分成10×10的网格，每一个网格中默认先验框的个数设置为4，传入到回顾预测与分类预测网络中；(5)经过三次3×3的卷积，输出结果为19×19×1024，在经过2×2的最大池化，输出的结果为10×10×1024，并把这些特征层划分成8×8的网格，每一个网格中默认先验框的个数设置为6，传入到回顾预测与分类预测网络中；(6)经过一次1×1的卷积和一次3×3卷积，输出结果为10×10×1024，并把这些特征层划分成5×5的网格，每一个网格中默认先验框的个数设置为6，传入到回顾预测与分类预测网络中。最后经过非极大抑制，输出分类结果。

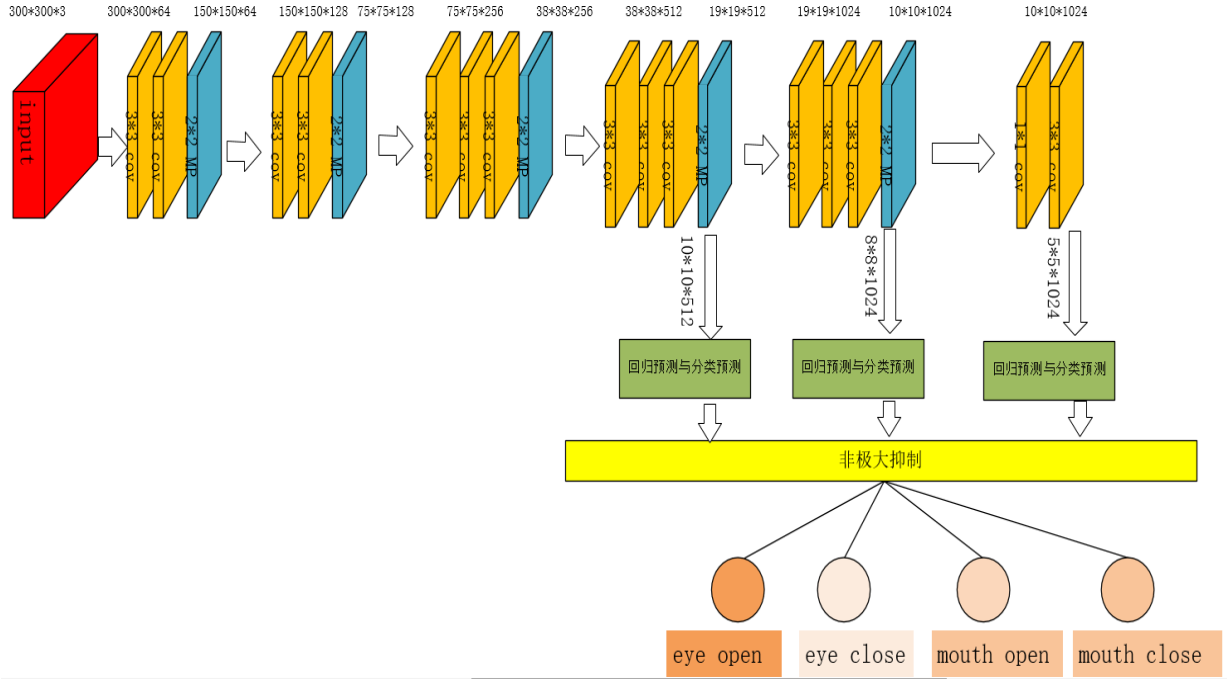


图4.14 本文神经网络模型

4.5 数据集的制作

实验所使用的的数据集是以YawDD视频数据集为基础制作而成。YawDD视频数据集采集了若干名女生和男生在过暗、弱光和正常光的条件下实际驾驶汽车的视频数据，该视频数据具体包括：正常驾驶时，眼睛睁开、嘴巴闭合或者轻微张开的状态；在疲劳驾驶时，眼睛闭合、嘴巴打哈欠的转态。每个一定时间采集正常驾驶和疲劳驾驶情况下的驾驶员图片，具体包括5000张眼睛睁开状态的照片、

5000张眼睛闭合状态的照片、5000张嘴巴正常转态的照片和5000张嘴巴打哈欠状态的照片，然后把每部分等分为训练集与测试集两部分。然后把采集到20000张照片经过低光图像处理和人脸区域提取，如图4.15所示。目标检测与识别任务是一个有监督的学习任务，因此必须要在实验的数据集中标记出感兴趣的物体所在的位置信息和类别信息。本文做采用的标记工具是LabelImg，操作简单。

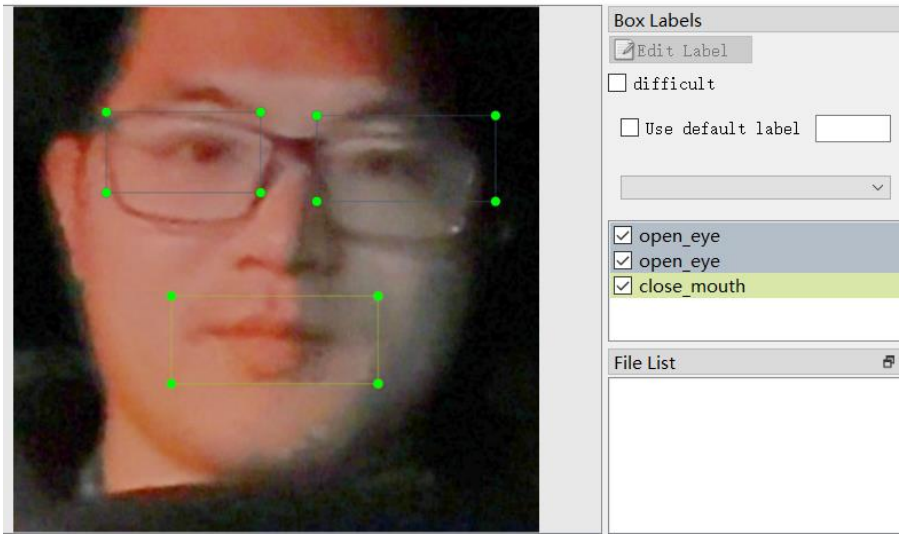


图4.15 感兴趣区域标记

4.6 卷积神经网络模型的训练与优化

目前，绝大多数卷积神经网络的模型采用批量随机梯度下降算法来得到最优化的损失函数以及模型参数值，其公式如下：

$$\theta_{t+1} = \theta_t - \frac{\lambda}{m}\sum_{i}^{m}\nabla J(\theta) \tag{4.24}$$

式中，θ是初始值，λ表示学习率(learning rate)，m表示批处理大小(batchsize)，$J(\theta)$对θ的偏导表示$J(\theta)$变化最大的方向。从公式可以得知，学习率和批处理大小的值影响着模型是否能够收敛以及收敛速度的快慢，因此这两个因子对模型性能效果起着至关重要的作用。具体来说，学习率影响模型的收敛状态，批处理大小影响模型的泛化性能。并且，这两个参数又是分子和分母的关系，彼此互相影响。有一句在业内人士流传很广的话：如果在所有超参中，只需要调整一个参数，那么就是学习率。可见，学习率对网络模型的性能影响是多么重要。但是在神经网络模型的训练过程中，选取一个适当的学习率值比较困难，因为学习率过小或者过大都不行，比如学习率过小会导致模型需要训练很长时间才能够收敛，而学习率过大会导致模型找不到最合适的损失函数，极端的情况会出现模型发散的现象。

4.6.1 学习率的选取

选择合适学习率时，首先需要固定批处理大小。批处理大小过大，会导致模型的泛化能力下降；批处理大小过小，模型的稳定性不足、数据的训练时间过长。并且，批处理大小的值与硬件性能紧密相关。因此，结合本文所使用的笔记本性能，初步选择批处理大小为8。

紧接着是固定epoch的大小，epoch相当于是停止程序的一个控制参数，本身上并不会影响网络模型的性能。一般而言，epoch越大，对数据训练的程度的越大，准确率在一定范围内也会有所提高。但是当epoch达到一定值后，准确率并不会提高、loss值也并不会下降，此时由于训练次数的增加，却需要耗费大量的时间成本，得不偿失。因此，应该设置合理的epoch。由于本文的数据量为10000，如果把epoch设置为50次，次数总共的训练次数为10000×50=500000，能够满足本文章的要求。

学习率的调整需要满足以下公式：

$$\sum_{i=1}^{\infty}\varepsilon_i=\infty \tag{4.25}$$

$$\sum_{i=1}^{\infty}\varepsilon_i^2<\infty \tag{4.26}$$

式中，ε_i表示初始学习率。

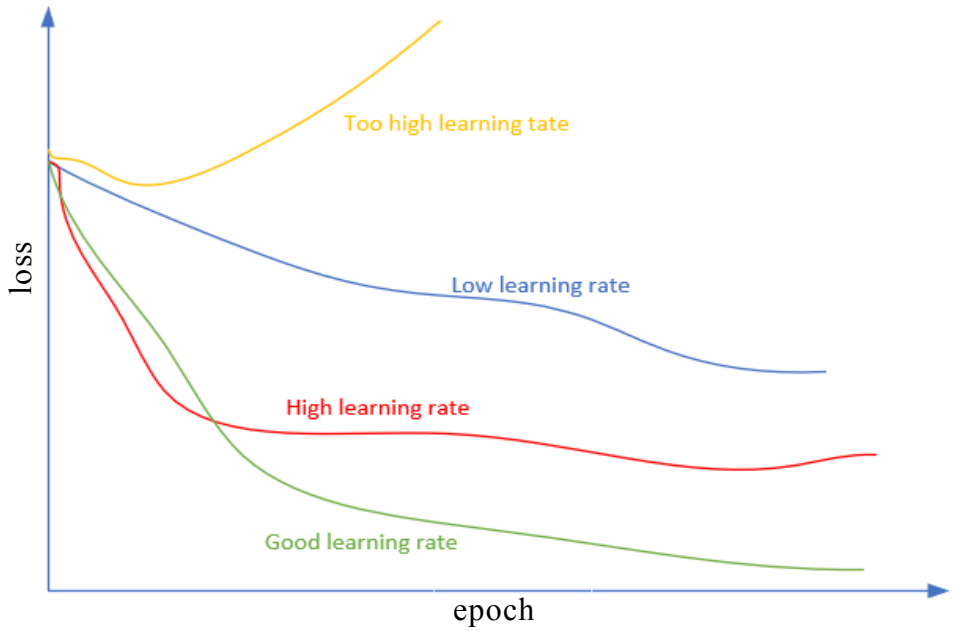


图4.16 学习率对训练结果的影响

图4.16显示了不同的学习率对训练结果影响，从图可以明显看出，黄色表示当学习率过大，loss值会逐渐增大，网络发散；蓝色表示当学习率过小，loss值下降速度比较慢，网络需要训练很长时间才能收敛；红色表示学习率稍大，loss开始下降速度快，会越过极小值点，并且最优解附近跳；绿色表示学习率设置合理，能找到极小值点。

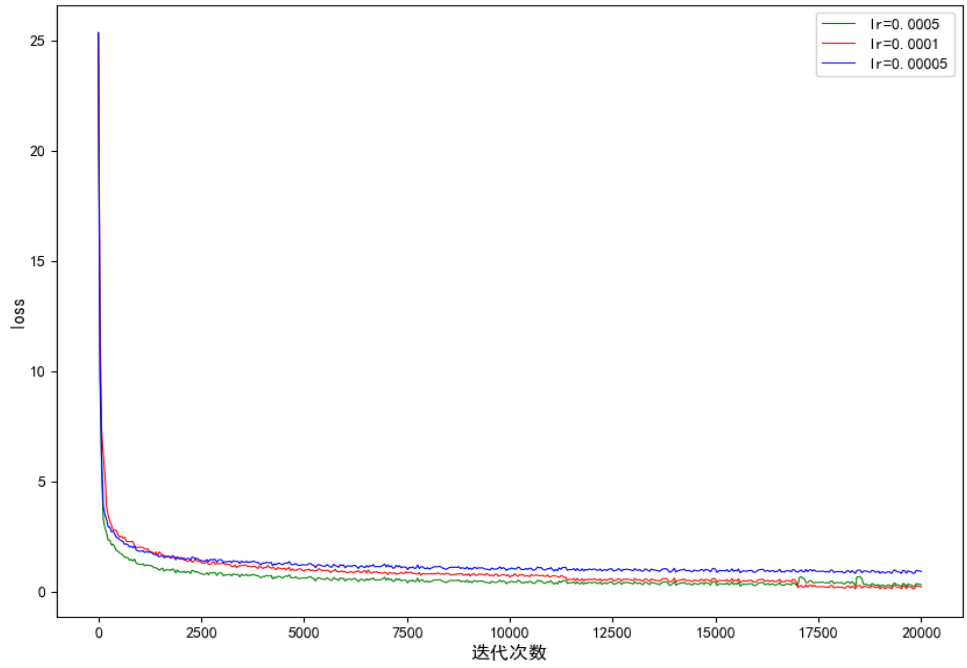


图4.17 不同学习率的loss对比

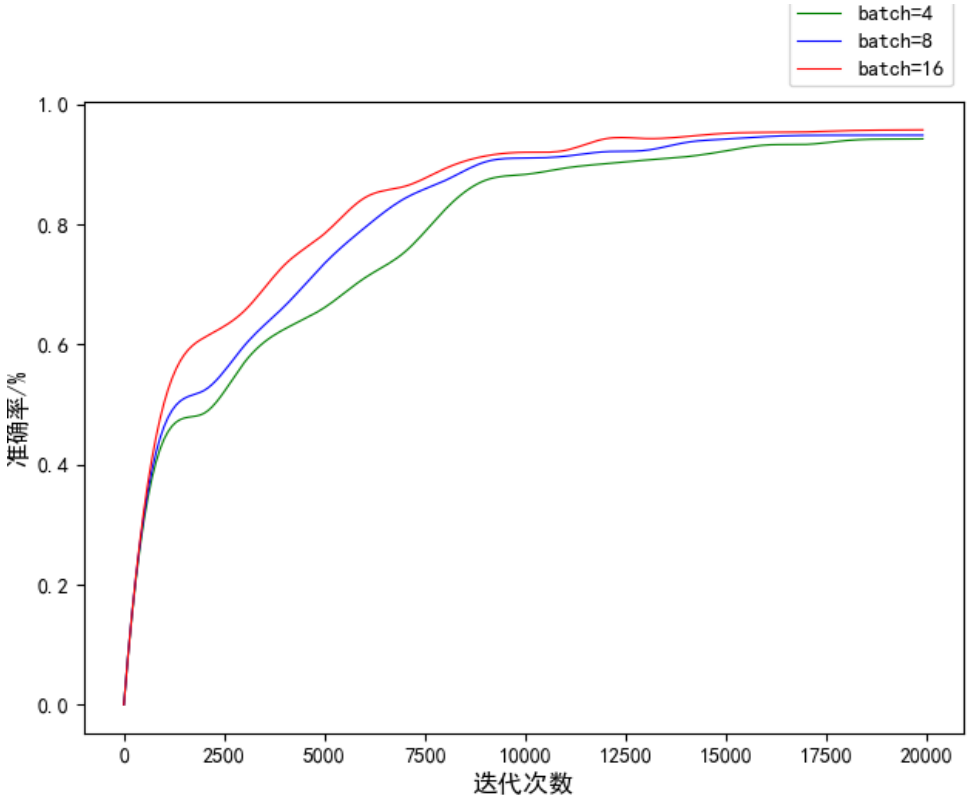


图4.18 不同学习率与准确率的对比

为了防止在训练时，网络不收敛，因此学习率不应设置过大。根据日常经验，本文把学习率大小设置为5×10^{-5}、1×10^{-4}和5×10^{-4}等值，然后找到最合适本文工况的学习率。不同学习率下，迭代次数与loss值的实验结果如图4.17所示，从图中可以明显看出：当学习率设置为5×10^{-5}时，模型收敛速度比较慢；当学习率设置为5×10^{-4}时，收速度快，但是收敛之后，抖动比较明显；当学习率设置为

1×10^{-4}时，模型收敛速度适中并且不会出现抖动。不同学习率下，迭代次数与准确率的实验结果如图4.18所示，从图中可以看出：当学习率1×10^{-4}时，准确率最高。综上所述，本文把学习率设置为1×10^{-4}。

4.6.2 批处理大小的选择

批处理大小对模型性能的影响没有学习率强烈，但是进一步提升模型性能的时候，批处理大小也会成为比较关键的因素。人工智能领域的研究者普遍发现，当批处理大小比较大时，能够减少训练所需要的时间，使计算更加稳定，模型的训练曲线更加平滑，并且在微调时，能够获得更好的结果。但是，当批处理大小的值过大时，又会使模型的性能下降。因此，为了提升模型的整体性能，本文固定其他参数，结合本实验所使用的硬件设备，将批处理大小设置为4、8和16，然后找到最合适的批处理大小。实验结果如图4.19所示，当批处理大小设置为16时，准确率最高。因此，本文把批处理大小设置为16。

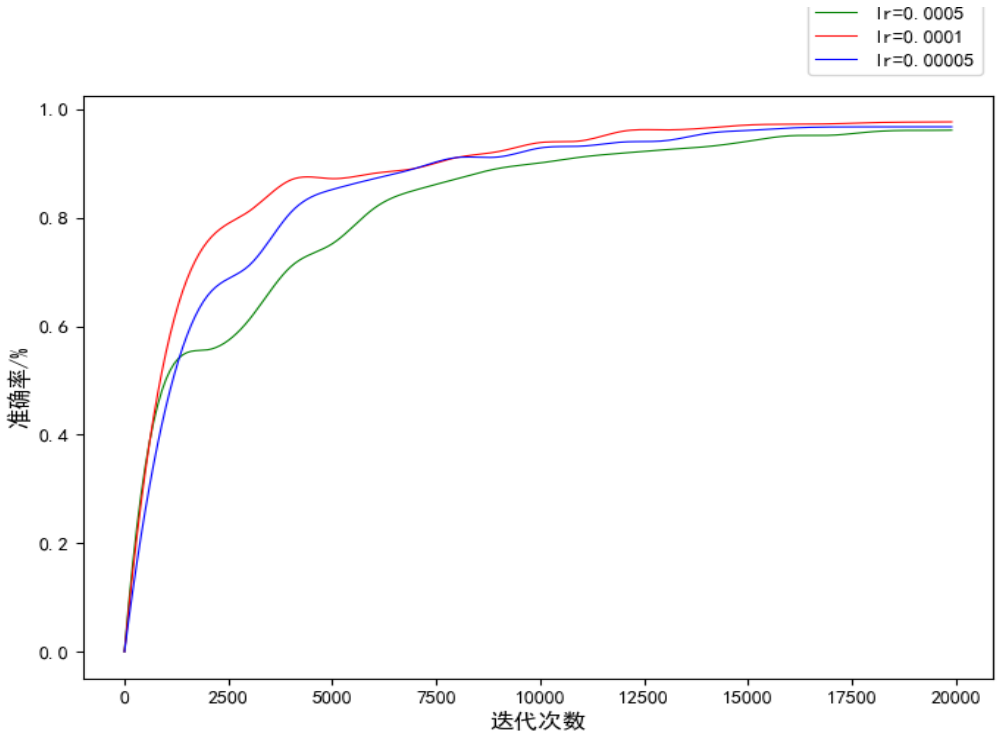


图4.19 不同批处理大小与准确率的对比

4.7 实验设计与结果分析

4.7.1 实验数据处理与平台搭建

本文以前文制作的数据集为基础，完成对VGG-16和SSD网络模型训练，获得最佳的模型数据，然后比较其准确率和数据处理速度。所使用的测试平台配置如表4.1所示，使用3.7版本的python语言、pycharm的编辑器以及1.2版本的torch深度学习框架进行模拟测试。

表4.1 测试平台参数

类型	型号	参数
操作系统	Windows	Windows10教育版
CPU	Inter(R) Core(TM)i5-6300HQ	4核
显卡	NVDIA GeForce GT 950	4GB
内存	DDR4	8GB

4.7.2 实验结果分析

为了验证本文所提出的卷积神经网络模型的性能，因此与 VGG-16和SSD这两种经典的卷积神经网络模型处理结果进行对比。VGG-16的测试结果如表4.2所示，从表中数据可以得知，对测试数据进行处理时，平均处理速度为143.23s，平均识别准确率为89.76%。

表4.2 VGG-16测试结果

状态	测试集图片数量(张)	正确识别数量(张)	处理速度(s)	准确率
眼睛闭合	5000	4459	142.25	89.18%
眼睛睁开	5000	4425	143.15	88.5%
嘴巴闭合	5000	4520	144.25	90.4%
嘴巴张开	5000	4550	143.28	91%

SSD的测试结果如表4.3所示，根据表中数据，当我们对测试数据进行处理时，平均处理速度为214.98s，平均识别准确率为96.37%。

表4.3 SSD测试结果

状态	测试集图片数量(张)	正确识别数量(张)	处理速度(s)	准确率
眼睛闭合	5000	4839	212.25	96.78%
眼睛睁开	5000	4825	215.15	96.5%
嘴巴闭合	5000	4810	218.25	96.2%
嘴巴张开	5000	4800	214.28	96%

本文所提出的卷积神经网络模型测试结果如表4.4所示，从表中数据可以得知，对测试数据进行处理时，平均处理速度为161.72s，平均识别准确率为95.35%。

表4.3 本文模型测试结果

状态	测试集图片数量(张)	正确识别数量(张)	处理速度(s)	准确率
眼睛闭合	5000	4755	161.28	95.1%
眼睛睁开	5000	4760	163.16	95.2%
嘴巴闭合	5000	4790	160.24	95.8%
嘴巴张开	5000	4765	162.18	95.3%

为了方便比较，将三种网络模型测试得到的平均处理处理速度与平均识别准确率的图形绘制出来，如图4.20所示。从图中，我们可以明显看出，本文网络模型对眼睛状态和嘴巴状态的识别准确率高于VGG-16网络模型，略低于SSD网络模型，对数据的处理速度上略低于VGG-16[38]，但是高于SSD网络模型。因此，本文提出的卷积神经网络模型在准确率和处理速度上都具有一定的优势。

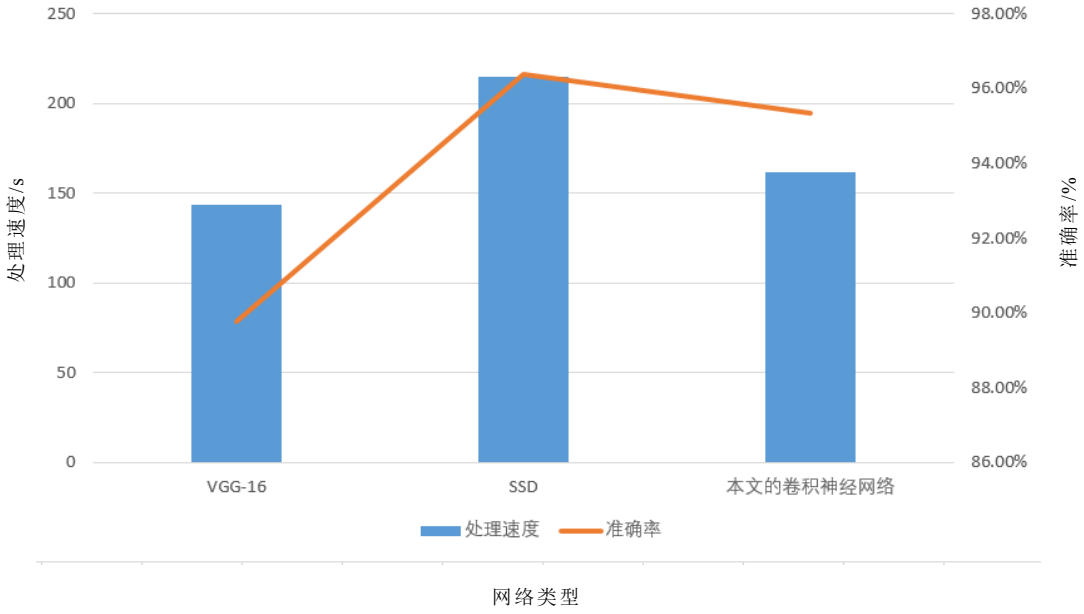


图4.20 三种网络模型平均处理结果对比

4.8 本章小结

本章首先简单概述了与卷积神经网络有关的知识，然后分析卷积神经网络模型搭建的基本流程，紧接着以SSD网络模型为基础，结合本文的应用场景，设计了一种用于辨别驾驶员眼睛状态与嘴巴状态的卷积神经网络模型。然后使用自制的数据集，对此模型进行训练，得到最合适的学习率与批处理大小等关键参数值。最后通过实验对比，得出了该卷积网络模型在准确率和处理速度上都具有一定的优势。

第5章 多特征融合疲劳驾驶检测

由于驾驶员连续驾驶车辆未停车休息或者睡眠时间没有达到正常标准等原因，造成驾驶员驾驶技能下降，我们把这种现象称之为疲劳驾驶。当驾驶员疲劳驾驶时，生理机能和心理机能将会有显著的变化，比如频繁的眨眼、长时间闭眼、打哈欠、点头等。因此，根据这些特征我们能够有效判断驾驶员是否疲劳驾驶。但是，如果仅仅根据单一因素来进行判断，误差会比较大，存在一定的局限性。比如当我们仅仅依据驾驶员的眼睛状态这一单一因素进行疲劳驾驶判断时，如果驾驶员佩戴深色眼睛，那么这种检测方法将会失效。因此，本章提取了第3章的头部姿态信息以及第4章通过卷积神经网络得出的驾驶员眼睛状态信息和嘴巴状态信息，比如闭眼帧数、睁眼帧数、嘴巴张开帧数和嘴巴闭合帧数，然后利用这些信息来综合判断驾驶员是否疲劳驾驶，提高系统的鲁棒性。

5.1 疲劳驾驶参数提取

5.1.1 眼部疲劳驾驶参数提取

根据相应的理论研究发现，眼睛在正常情况下和疲劳情况下，每分钟眨眼的次数是不一样的，在正常情况下每分钟大概眨眼15~20次，而眼睛在疲劳情况下，每分钟眨眼次数大概只有正常情况的60%。根据这一现象，研究者发现驾驶员疲劳驾驶时，眼睛的闭合时间与疲劳程度等级有紧密的关系。由卡内基梅隆研究所提出的PERCLOS驾驶员疲劳评测方法是最有效的、最可靠的、实时的方法。一般而言，PERCLOS包含3个判断标准：EM、P70和P80。EM表示当瞳孔面积有超过50%被眼睛遮住时，就认为眼睛处于闭合状态；P70表示当瞳孔面积有超过70%被眼睛遮住时，就认为眼睛处于闭合状态；P80表示当瞳孔面积有超过70%被眼睛遮住时，就认为眼睛处于闭合状态通过相应的实验表明，P80检测结果最准确，最适合作为疲劳驾驶检测的判断标准。因此，本文选择P80作为驾驶员眼部是否闭合的标准。一次眨眼的P80计算原理如图5.1所示。计算公式如下：

$$f = \frac{t_3 - t_2}{t_4 - t_1} \tag{5.1}$$

式中，$t_3 - t_2$表示眼睛睁开不到20%的时间，$t_4 - t_1$表示眼睛完全睁开到完全闭合再到完全睁开的时间，f表示眼睛睁开不到20%的时间与眼睛完全睁开到完全闭合再到完全睁开的时间的比值，即我们的P80判断标准。PERCLOS[52]的理论研究结果表明，驾驶员是否疲劳驾驶的P80判断标准阈值是0.15，即当f小于等于0.15时，

此时驾驶员处于正常驾驶状态，当 f 大于0.15时，此时驾驶员处于疲劳驾驶状态。

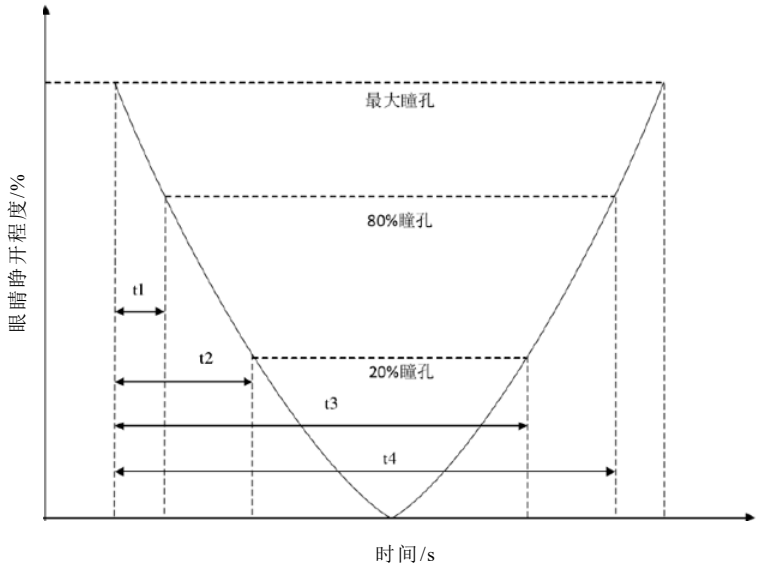
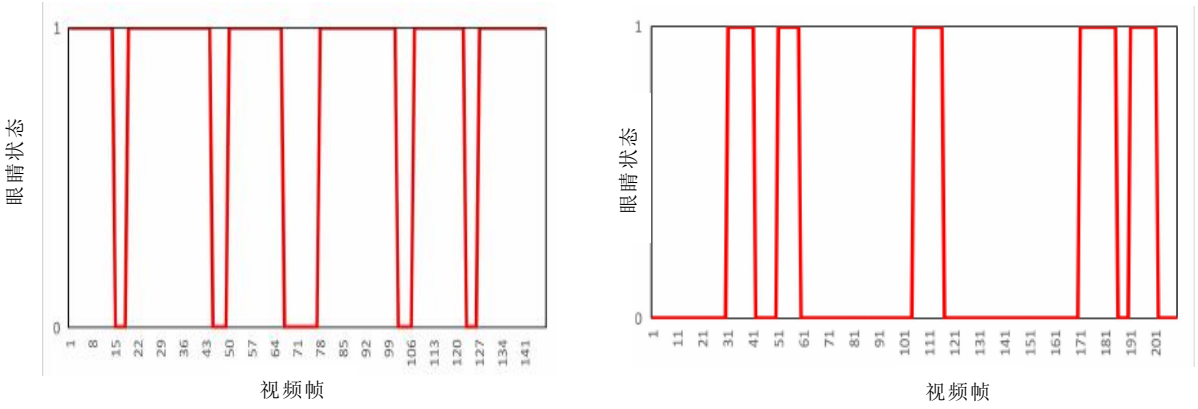


图5.1 P80原理图

以此判断标准为基础，结合本文的实际情况，给出适合的眼部疲劳判断标准，计算公式如下：

$$f_{eye} = \frac{n}{N} \tag{5.2}$$

式中，n表示眼睛闭合帧数，N表示视频序列总帧数，f_{eye}表示眼睛闭合帧数占总帧数的比值。当驾驶员疲劳驾驶程度越严重时，意味着在一分钟之内采集到的眼睛闭合的帧数越多，f_{eye}的值也就越大。借鉴P80疲劳驾驶判断标准，可以将眼部是否疲劳的阈值参数设置0.15。图5.2表示测试视频序列里驾驶员在正常驾驶和疲劳驾驶情况下睁眼与闭眼的数据统计情况，其中数字1表示眼睛睁开的状态，数字0表示眼睛闭合状态。从图中可以明显看出，疲劳驾驶状态下眼睛闭合的序列数量是远远大于正常驾驶状态下眼睛闭合的序列数量，疲劳驾驶状态下眼睛睁开的序列数量是远远小于正常驾驶状态下眼睛闭合的序列数量。



a)正常驾驶状态　　b)疲劳驾驶状态

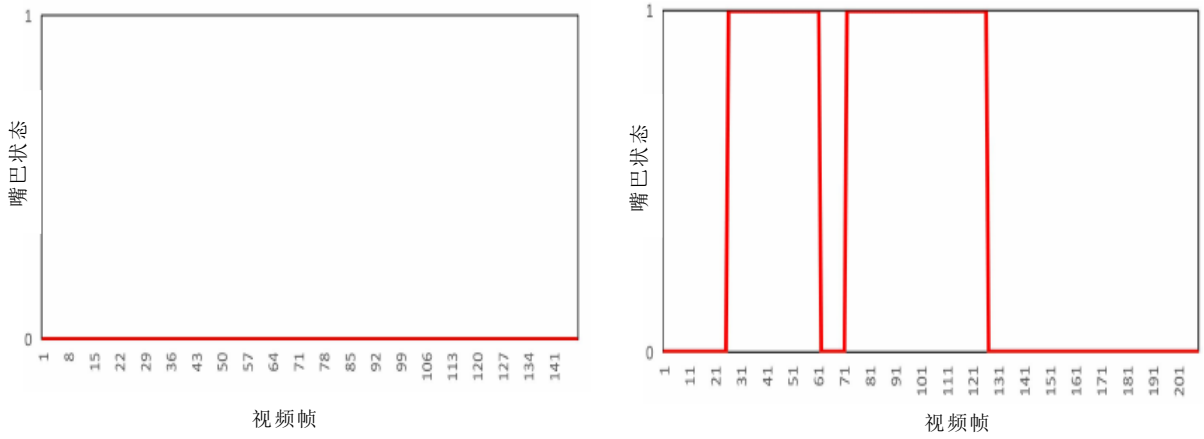
图5.2 驾驶员正常驾驶与疲劳驾驶时眼睛变化状态

5.1.2 嘴部疲劳驾驶参数提取

经过相应的理论研究，打哈欠是疲劳驾驶时驾驶员显著的面部变化特征之一。因此，我们可以把嘴巴的张开程度作为疲劳驾驶的判断依据之一。在汽车行驶过程中，驾驶员嘴巴状态一般可以分为三种情况：闭合状态、说话或者唱歌时嘴巴轻微张开状态、嘴巴张开持续时间比较长且幅度比较大的打哈欠状态。此外根据我们的日常生活经验，可以知道打哈欠时嘴巴的状态与正常说话时的状态是不相同的，比如嘴巴张开幅度大小。因此，我们可以根据嘴巴张开幅度大小来判断此时是否是打哈欠状态。为了便于实现嘴部疲劳驾驶检测，本文把打哈欠状态归于嘴巴张开状态，把说话或者唱歌时嘴巴状态归于嘴巴闭合状态。类比于PERCLOS的判断标准，可以使用以下公式作为嘴部疲劳驾驶判断依据：

$$f_{mouth} = \frac{n}{N} \tag{5.3}$$

式中，n表示一分钟之内嘴巴张开的帧数，N表示一分钟之内的视频序列的总帧数，f_{mouth}表示一分钟之内嘴巴张开的帧数与总帧数的比值。当驾驶员疲劳驾驶程度越严重时，意味着驾驶员可能打哈欠的次数也越多，相应的 f_{mouth} 的值也就越大。在大多数情况下，人们一次打哈欠持续的时间大约是3~6s。本文选取的单位检测时间是一分钟，据此，可以将嘴部是否疲劳的阈值参数设置为0.12。图5.3表示测试视频序列里驾驶员在正常驾驶和疲劳驾驶情况下嘴巴张开与闭合的数据统计情况，其中数字1表示嘴巴张开的状态，数字0表示嘴巴闭合的状态。从图中可以明显看出，疲劳驾驶状态下嘴巴张开的序列数量是远远大于正常驾驶状态下嘴巴张开的序列数量，疲劳驾驶状态下嘴巴闭合的序列数量是远远小于正常驾驶状态下嘴巴闭合的状态序列数量。



a)正常驾驶状态　　b)疲劳驾驶状态

图5.3 驾驶员正常驾驶与疲劳驾驶时嘴巴变化状态

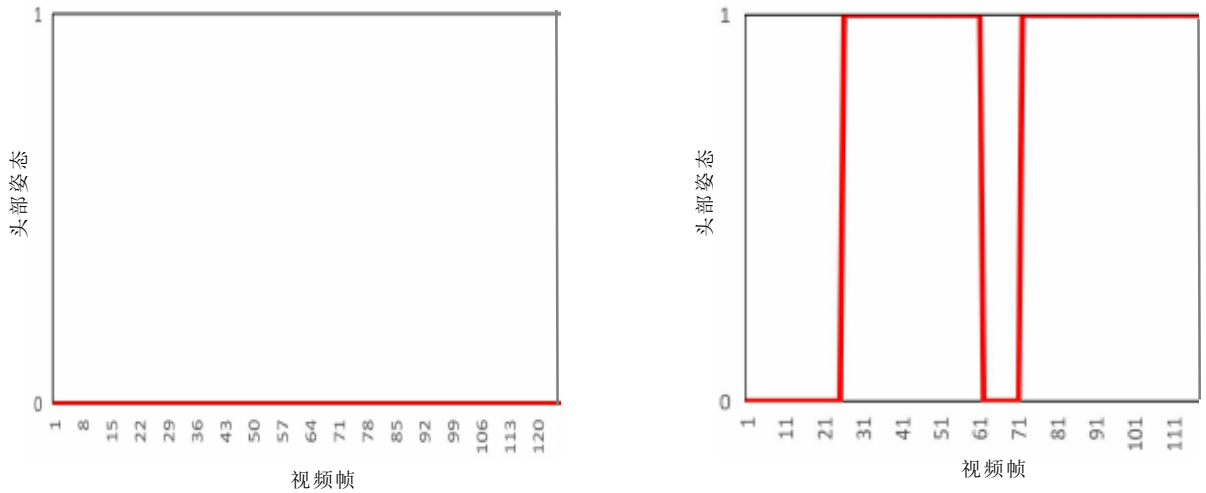
5.1.3 头部姿态疲劳驾驶参数提取

当驾驶员正常驾驶时，绝大部分情况头部会在水平方向上左右移动，少部分情

况会在前后方向上移动。但是，当驾驶员出现疲劳驾驶情况时，此时驾驶员的颈部肌肉松弛乏力，头部姿势很难保持正常水平，出频繁出现前倾后仰或者左右倾斜的情况，在头部姿态的欧拉模型中，对应着俯仰角(pitch)和滚转角(roll)的角度变化。通过第三章的头部姿态估计方法，我们可以得到驾驶员在驾驶过程中的头部姿态的俯仰角(pitch)和滚转角(roll)角度值。相应实验表明，当|pitch|≥20^{o}或者|roll|≥20^{o}时，可以认为此时驾驶员的头部姿势处于异常情况。类比于PERCLOS的判断标准，可以使用以下公式作为头部姿态疲劳驾驶判断依据：

$$f_{head}=\frac{n}{N} \tag{5.4}$$

式中，n表示一分钟之内头部姿态异常的帧数，N表示一分钟之内的视频序列的总帧数，f_{head}表示一分钟之内嘴巴张开的帧数与总帧数的比值。当驾驶员疲劳驾驶程度越严重时，意味着驾驶员可能出现头部姿态异常可的次数也越多，相应的f_{head}的值也就越大。在大多数情况下，驾驶员一次头部姿态异常持续的时间大约是6~10s。本文选取的单位检测时间是一分钟，据此，可以将头部姿态是否疲劳的阈值参数设置为0.15。图5.4表示测试视频序列里驾驶员在正常驾驶和疲劳驾驶情况下头部姿态正常与异常的数据统计情况，其中数字1表示头部姿态异常的状态，数字0表示头部姿态正常的状态。从图中可以明显看出，疲劳驾驶状态下头部姿态异常的序列数量是远远大于正常驾驶状态下头部姿态异的序列数量，疲劳驾驶状态下头部姿态正常的序列数量是远远小于正常驾驶状态下头部姿态正常的状态序列数量。



a)正常驾驶状态　　b)疲劳驾驶状态

图5.4 驾驶员不同驾驶状态时头部姿态变化

综上所述，本文最终将眼睛、嘴巴与头部姿态的疲劳判断阈值设置如表5.1所示。最后结合第二章、第三章、第四章和第五章内容，本论文的驾驶员疲劳驾驶检测方法的逻辑流程框图如图5.5所示。

表5.1 疲劳驾驶参数阈值设定

是否疲劳	f_{eye}	f_{mouth}	f_{head}
疲劳	>0.15	>0.12	>0.15
正常	<=0.15	<=0.12	<=0.15

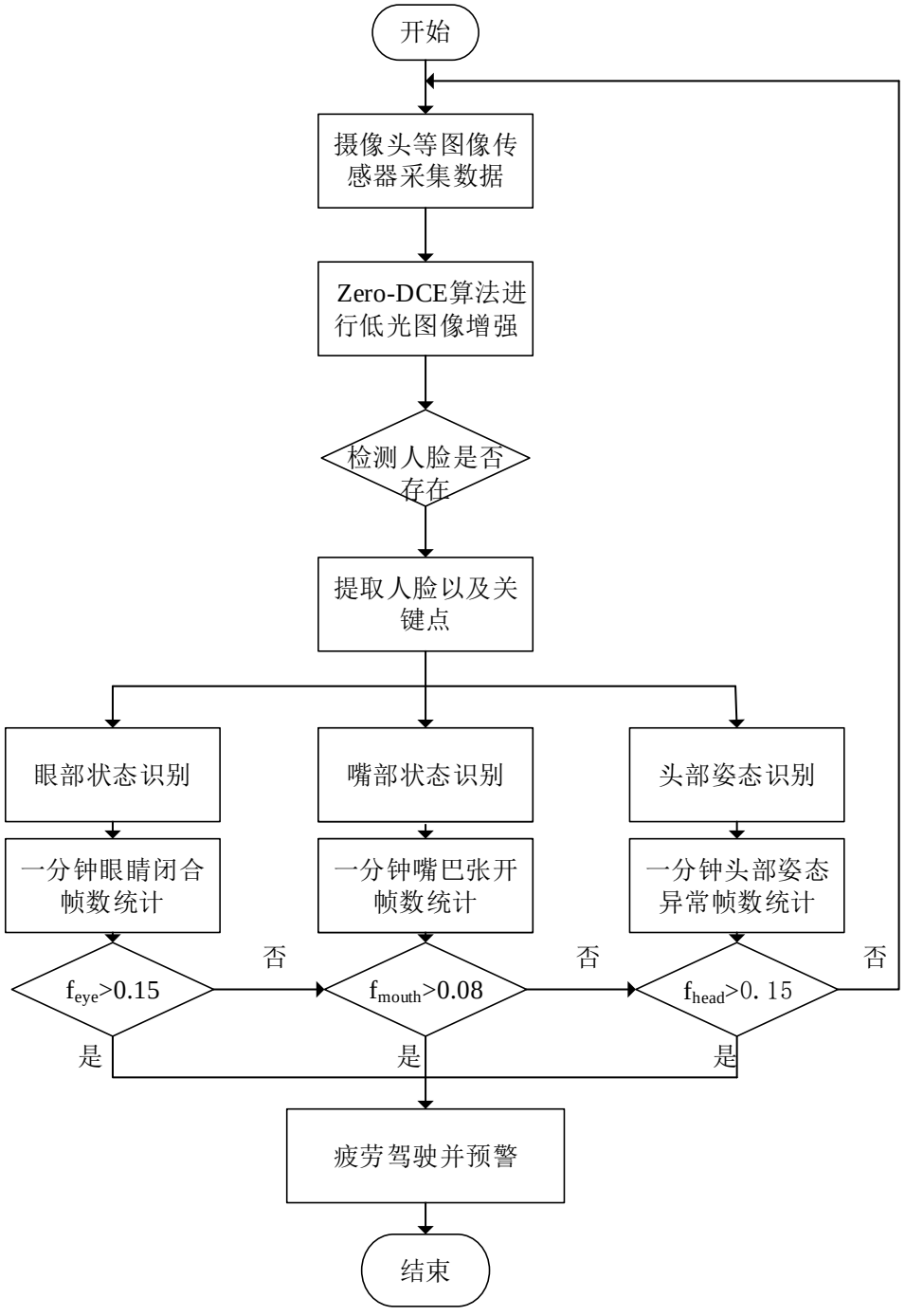


图5.5 逻辑框图

5.2 实验设计与结果分析

5.2.1 实验数据处理与平台搭建

为了测试本文的疲劳驾驶检测方法的可行性，本文选取了YawDD数据中的10段不同的视频，具体包括正常驾驶视频片段、打哈欠驾驶视频片段、头部姿态异常驾驶视频片段和闭眼次数过多视频片段。所使用的测试平台配置如表5.2所示，使用版本为3.7python程序设计语言、版本为1.2的torch深度学习框架以及pycharm的编辑器对疲劳驾驶检测算法进行模拟测试。

表5.2 测试平台参数

类型	型号	参数
操作系统	Windows	Windows10教育版
CPU	Inter(R) Core(TM)i5-6300HQ	4核
显卡	NVDIA GeForce GT 950	4GB
内存	DDR4	8GB
硬盘	固态硬盘	256GB

5.2.2 鲁棒性测试

(1)佩戴眼镜测试

有的驾驶员佩戴眼镜，有的驾驶员没有佩戴眼镜，因此需要对本文疲劳驾驶检测系统进行是否戴眼镜测试，测试结果如图5.6所示。从图中可以看出，当驾驶员佩戴眼镜时，本文的疲劳驾驶检测系统也能够有效的对眼睛状态、嘴巴状态进行有效判别。

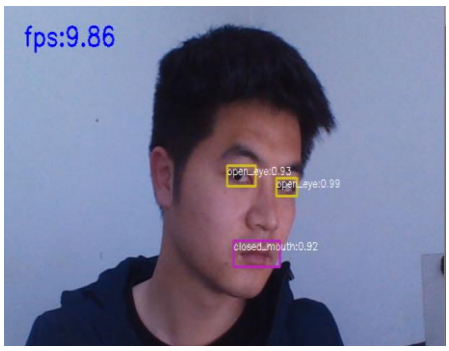
a)佩戴眼镜测试

b)不佩戴眼镜测试

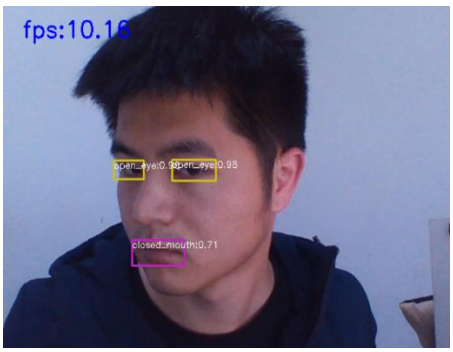
图5.6 是否佩戴眼镜测试

(2)不同头部姿态测试

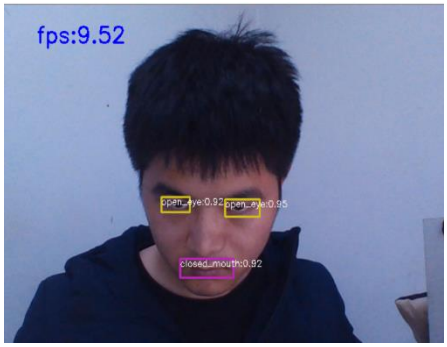
驾驶员在驾驶车辆时，会出现左偏、右偏、低头以及抬头等情况。因此，需要在不同的头部姿态下进行测试，测试结果如图5.7所示。从图中我们可以看出，当驾驶员的头部姿态处于不同情况时，本文的疲劳驾驶检测系统也能够有效的对眼睛状态、嘴巴状态进行有效判别。



a)右偏测试



b)左偏测试



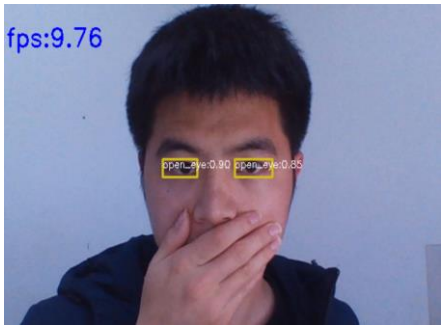
c)低头测试

d)抬头测试

图5.7 不同头部姿态测试

(3)面部遮挡测试

驾驶员在驾驶车辆时，有时会出现面部遮挡的情况，可能会影响本文的疲劳驾驶检测系统的判断精度。因此，接下来进行面部遮挡测试，测试结果如图5.8所示，从图中我们可以看出，当驾驶员的面部在一定范围内受到遮挡时，本文的疲劳驾驶检测系统也能够有效的对眼睛状态、嘴巴状态进行有效判别。



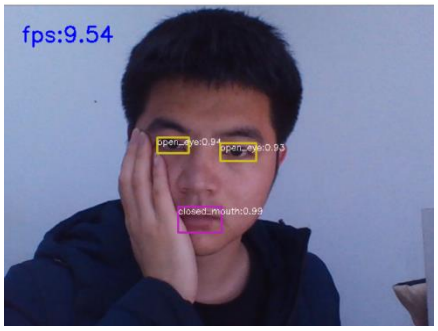


图5.8 面部遮挡测试

综上所述，本文的疲劳驾驶检测方法对外部环境的干扰有较强适应性，在不同的场景中能够对眼睛状态、嘴巴状态进行有效判别，鲁棒性比较好。

5.2.3 准确性测试

采用第4章已经建立的数据集进行准确性测试，表5.3是测试得到的结果数据。从表中可以看出，十段视频数据只有一段视频数据产生误判，把正常驾驶状态判断为疲劳驾驶状态。从表中的也可以看出，本论文的驾驶员疲劳驾驶检测方法对

嘴部状态和头部姿态判断较为准确，但是对眼部姿态判断有所误差，可能是由于驾驶员此时头部发生了前后方向的移动，导致没能有效的识别眼部状态，导致了误判。但是，整体而言，本论文的驾驶员疲劳驾驶检测算法的疲劳状态判断效果还不错。

表5.3 数据集测试结果

视频编号	闭眼帧数	睁眼帧数	闭嘴帧数	张嘴帧数	头部正常帧数	头部异常帧数	总帧数	f_{eye}	f_{mouth}	f_{head}	检测疲劳状态	实际疲劳状态
1	4	30	34	0	34	0	34	0.11	0	0	正常	正常
2	6	24	30	0	30	0	30	0.2	0	0	疲劳	正常
3	5	28	33	0	33	0	33	0.15	0	0	疲劳	正常
4	6	35	38	3	41	0	41	0.14	0.07	0	正常	正常
5	4	36	30	10	38	2	40	0.10	0.25	0.05	疲劳	疲劳
6	8	50	52	6	54	4	58	0.13	0.10	0.06	疲劳	疲劳
7	3	27	30	0	20	10	30	0.10	0	0.30	疲劳	疲劳
8	4	34	38	0	30	8	38	0.10	0	0.21	疲劳	疲劳
9	5	35	38	2	40	0	40	0.125	0.05	0	正常	正常
10	8	52	60	0	56	4	60	0.13	0	0.06	正常	正常

本文的疲劳驾驶检测系统的鲁棒性和准确性测试结果表明，本论文的驾驶员疲劳驾驶检测算法有一定的实际应用的价值。

5.3 本章小结

本章介绍了基于面部特征的疲劳驾驶检测常采用的参数，比如PERCLOS 参数、打哈欠参数和头部姿态异常参数。结合本文的实际情况，给出了适合本疲劳驾驶检测方法的参数，比如疲劳驾驶时眼部参数阈值、嘴部参数阈值和头部姿态异常阈值。然后融合这些疲劳驾驶判断依据信息来建立疲劳驾驶识别模型，能够有效提高系统的鲁棒性。最后以YawDD数据集以及摄像头实时采集的数据为测试集，验证本文的疲劳驾驶检测系统的可行性。

总结与展望

1. 工作总结

为了降低由疲劳驾驶产生的交通事故，设计一款疲劳驾驶检测系统显得尤为重要。本文通过梳理目前主流的疲劳监视检测方法并对比分析每种方法的优劣性，认为基于驾驶员面部特征的检测算法可实用性更好。针对目前在弱光条件下，疲劳驾驶检测系统检测精度不高的情况，本文使用低光图像增强算法解决低光环境下图像曝光度低的问题；针对传统的疲劳驾驶检测方法识别精度低、鲁棒性差的情况，本文使用卷积神经网络模型来提高其精度；针对目前单一疲劳驾驶判断因素的误差比较大，有一定的局限性的情况，本文融合多种疲劳驾驶判断因素，有效提高其判断精度。本文的具体工作内容如下所示：

(1)针对驾驶员疲劳驾驶的情况主要出现在夜间，此时光线比较暗，驾驶员的眼部与嘴部等特征不明显，影响其状态的识别精度。因此，在图像进入人脸检测与提取之前，先对图像进行相应的处理，提高图像质量，从而提高后续眼睛睁闭状态和嘴巴张开状态的识别精度。本文对比了传统的低光图像方法和近几年典型的图像增强算法，对比其优缺点，结合本文的应用的场景，最终使用Zero-DCE算法完成图像的低光增强处理，该模型在处理效果和实时性优势明显。

(2)采用RetinaFace算法完成对驾驶员的人脸检测并提取人脸区域图像，然后使用人脸关键点定位算法，准确获取了脸部轮廓、眼睛、嘴巴、鼻子等人脸68个关键区域的坐标信息，为驾驶员的头部姿态估计提供相应的关键点坐标。测试结果表明，本文采用的方法对驾驶员面部区域提取和关键点定位具有较高的准确性和速度。

(3)分析卷积神经网络模型搭建的基本流程，紧接着以SSD网络模型为基础，结合本文的应用场景，设计了一种用于辨别驾驶员眼睛状态与嘴巴状态的卷积神经网络模型。然后使用自制的数据集，对此模型进行训练，得到最合适的学习率与批处理大小等关键参数值。最后通过实验对比，得出了该卷积网络模型在准确率和处理速度上都具有一定的优势。

(4)基于面部特征的疲劳驾驶检测常采用的参数，比如PERCLOS 参数、打哈欠参数和头部姿态异常参数，然后结合本文的实际情况，给出了适合本疲劳驾驶检测方法的参数，比如疲劳驾驶时眼部参数阈值、嘴部参数阈值和头部姿态异常阈值。紧接着融合这些疲劳驾驶判断依据信息来建立疲劳驾驶识别模型，这样做能够有效提高系统的鲁棒性。经过相应数据集的测试，本文的疲劳驾驶检测系统有

一定的实际应用价值。

2. 工作展望

目前，疲劳驾驶检测领域仍处于研究阶段，设计并实现一个价格便宜、准确性高、判断速度快和鲁棒性的疲劳驾驶检测系统具有比较大的挑战性。虽然，本文设计的疲劳驾驶方法取得了一定的成果，但是还是存在很多的局限性，具体有以下几点：

(1)本文在Window系统上进行实验，Python语言编写。所以其可移植性比较差、对硬件的要求比较高，不利于开发平台的扩展，不利于其商业化。

(2)本文只是以驾驶员的面部特征和头部姿态作为疲劳驾驶的判断依据，如果能够融合车辆特征，比如车辆偏移、方向盘转动、车速等信息，能够进一步提供准确率和鲁棒性。

(3)本文的测试与验证是在室内模拟进行的，其驾驶员环境远没有真实的驾驶环境复杂。因此，本文的测试结果可能有一定的偏差，有待改进。

(4)本文的眼部、嘴部和头部姿态的疲劳阈值都是固定值，但是这些值并不是适合每一个驾驶员，会带来一定的误差。因此，未来可以研究阈值自动调整的算法。

参考文献

[1] 国家统计局. 中华人民共和国2019年国民经济和社会发展统计公报[J]. 中国统计, 2019, 2(1): 16-18.

[2] 吴群. 基于心电信号的驾驶疲劳检测方法研究[D]. 杭州: 浙江大学, 2008.

[3] 房瑞雪, 赵晓华, 荣建等. 基于脑电信号的驾驶疲劳研究[J]. 公路交通科技, 2009, 0(S1): 124-126.

[4] 宋战兵. 浅谈驾驶疲劳检测方法研究现状[J]. 科技风, 2017(9): 282-282.

[5] Wang M S, Jeong N T, Kim K S, et al. Drowsy behavior detection based on driving information[J]. International Journal of Automotive Technology, 2016, 17(1): 165-173.

[6] Cheng B, Zhang W, Lin Y, et al. Driver drowsiness detection based on multisource information[J]. Human Factors and Ergonomics in Manufacturing and Service Industries, 2012, 22(5): 450-467.

[7] Maddirala A K, Shaik R A. Separation of sources from single-channel EEG signals using independent component analysis[J]. IEEE Transactions on Instrumentation and measurement, 2017, 67(2): 382-393.

[8] 李庆臣. 基于面部特征的疲劳驾驶检测系统设计[D]. 郑州: 郑州大学, 2019.

[9] 毛须伟, 景文博, 王晓曼, 等. 一种基于眼部状态的疲劳驾驶检测方法[J]. 长春理工大学学报:自然科学版, 2016, 39(002): 125-130.

[10] 赵晓华, 杜洪吉, 荣建. 基于ROC曲线的疲劳驾驶判别方法研究[J]. 交通信息与安全, 2014, 32(5): 88-94.

[11] Zhang C, Wang H, Fu R. Automated detection of driver fatigue based on entropy and complexity measures[J]. IEEE Transactions on Intelligent Transportation Systems, 2013, 15(1): 168-177.

[12] 徐成, 孙伟, 戴争辉, 等. 一种面向入侵检测的BM模式匹配改进算法[J]. 计算机应用研究, 2006, 023(011): 89-91.

[13] 童兵亮. 基于嘴部状态的疲劳驾驶和精神分散状态监测方法研究[D]. 长春: 吉林大学, 2004.

[14] 沈英超. 基于眼部特征的疲劳驾驶检测系统的研究与实现[D]. 桂林: 桂林电子科技大学, 2019.

[15] Anitha C, Venkatesha M K, Adiga B S. A two fold expert system for yawning

detection[J]. Procedia Computer Science, 2016, 92(18): 63-71.

[16] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images[J]. Handbook of Systemic Autoimmune Diseases, 2009, 1(4): 2-18.

[17] Guo X, Li Y, Ling H. LIME: Low-light image enhancement via illumination map estimation[J]. Transactions on Image Processing, 2016, 26(2): 982–993.

[18] Lv F, Lu F, Wu J, at al. MBLLEN: Low-light image video enhancement using cnns[C]//BMVC. 2018: 220.

[19] Zhang Y, Zhang J, Guo X. Kindling the darkness: A practical low-light image enhancer[C]//Proceedings of the 27th ACM International Conference on Multimedia. 2019: 1632-1640.

[20] Guo C, Li C, Guo J, et al. Zero-Reference deep curve estimation for low-light image enhancement[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2020: 1780-1789.

[21] Wei C, Wang W, Yang W, et al. Deep retinex decomposition for low-light Enhancement[J]. Transactions on Image Processing, 2018, 31(5): 2072-2086.

[22] Jiang Y, Gong X, Liu D, et al. EnlightenGAN: Deep light enhancement without paired supervision[J]. Transactions on Image Processing, 2019, 2(30): 2340-2349.

[23] Deng J, Guo J, Zhou Y, et al. RetinaFace: Single-stage dense face localisation in the wild[J]. Computer Vision and Pattern Recognition, 2019, 34(8): 2025-2028.

[24] Cootes T F, Taylor C J, Cooper D H, et al. Active shape models-their training and application[J]. Computer Vision and Image Understanding, 1995, 61(1): 38-59.

[25] Dollár P, Welinder P, Perona P. Cascaded pose regression[C]//Computer Society Conference on Computer Vision and Pattern Recognition. 2010: 102-106.

[26] Sun Y, Wang X, Tang X. Deep convolutional network cascade for facial point detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2013: 3476-348.

[27] Kazemi V, Sullivan J. One millisecond face alignment with an ensemble of regression trees[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 1867-1874.

[28] Wang Y, Lucey S, Cohn J F. Enforcing convexity for improved alignment with constrained local models[C]//Conference on Computer Vision and Pattern

Recognition. 2008: 3621-3628.

[29] Saragih J M, Lucey S, Cohn J F. Deformable model fitting by regularized landmark mean-shift[J]. International journal of computer vision, 2011, 91(2): 200-215.

[30] Papandreou G, Maragos P. Adaptive and constrained algorithms for inverse compositional active appearance model fitting[C]//Conference on Computer Vision and Pattern Recognition. IEEE, 2008: 1539-1546.

[31] Matthews I, Baker S. Active appearance models revisited[J]. International Journal of Computer Vision, 2004, 60(2): 135-164.

[32] Amberg B, Blake A, Vetter T. On compositional image alignment, with an application to active appearance models[C]//Conference on Computer Vision and Pattern Recognition. IEEE, 2009: 1714-1721.

[33] 贾鹂宇, 张朝晖, 赵小燕, 闫晓炜. 基于人工智能视频处理的课堂学生状态分析[J]. 现代教育技术, 2019, 29(12): 82-88.

[34] 张乐. 驾驶员头部姿态估计方法的研究[D]. 武汉: 武汉理工大学, 2018.

[35] Rohit F, Kulathumani V, Kavi R, et al. Real-time drowsiness detection using wearable, lightweight brain sensing headbands[J]. IET Intelligent Transport Systems, 2017, 11(5): 255-263.

[36] Hubel D H, Wiesel T N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex[J]. Journal of Physiology, 1962, 160(1): 106-154.

[37] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks.International Conference on Neural Information Processing Systems[J]. Curran Associates Inc, 2012, 25(2): 1097-1105.

[38] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. Computer Science, 2014, 36(8): 1042-1050.

[39] He K, Zhang X, Ren S, et al. Identity mappings in deep residual networks[J]. European Conference on Computer Vision, 2016, 52(15): 2045-2051.

[40] Liu W, Anguelov D, Erhan D, et al. SSD: Single shot multiBox detector[C]// European Conference on Computer Vision. Springer, Cham, 2016: 122-128.

[41] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.

[42] Girshick R. Fast R-CNN[J]. Computer Science, 2015, 16(7): 1032-1040.

[43] 周云鹏. 基于面部视觉多特征融合的驾驶员疲劳检测方法研究[D]. 长沙:

湖南大学, 2015.

[44] 周云鹏, 朱青, 王耀南, 等. 面部多特征融合的驾驶员疲劳检测方法[J]. 电子测量与仪器学报, 2014, 28(10): 1140-1148.

[45] Wierwille W W, Antin J F, Dingus T A, et al. Visual attentional demand of an in-car navigation display system[C]//Vision in Vehicles II Second International Conference on Vision in Vehicles. 1988: 2010-2016.

[46] Bishop R. A survey of intelligent vehicle applications worldwide. Intelligent Vehicles Symposium//Proceedings of the IEEE. 2002: 266-269.

[47] Mao M, Du L. Research on drive fatigue detection using wavelet transform[C]//International Conference on Vehicular Electronics and Safety. IEEE, 2007: 1-4.

[48] Peng Y, Dong Y, Cheng D. Design and implementation of a driver's eye state recognition algorithm based on PERCLOS[J]. Chinese Journal of Electronics, 2014, 23(4): 669-672.

[49] Nair V, Hinton G E. Rectified linear units improve restricted boltzmann machines[C]//In Proceedings of International Conference on Machine Learning. 2010: 807-814.

[50] Wang M S, Jeong N T, Kim K S, et al. Drowsy behavior detection based on driving information[J]. International Journal of Automotive Technology, 2016, 17(1): 165-173.

[51] Sheng Y, Xiaohua L, Weihua Z, et al. Eyes state detection method based on lbp[J]. Application Research of Computers, 2015, 32(6): 1897-1901.

[52] Zhang W, Murphey Y L, Wang T, et al. Driver yawning detection based on deep convolutional neural learning and robust nose tracking[C]//International Joint Conference on Neural Networks. IEEE, 2015: 1-8.

[53] Sagonas C, Tzimiropoulos G, Zafeiriou S, et al. A semi-automatic methodology for facial landmark annotation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2013: 896-903.

[54] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]// Computer Vision and Pattern Recognition. 2016: 770-778.

[55] Wang H, Fu R. Automated detection of driver fatigue based on entropy and complexity measures[J]. IEEE Transactions on Intelligent Transportation Systems, 2013, 15(1): 168-177.

[56] Bishop R. A survey of intelligent vehicle applications worldwide[C]// Proceedings of the IEEE Intelligent Vehicles Symposium. IEEE, 2000: 25-30.

致谢

一年一度，似长江去浪，转瞬之间，求学生涯即将结束。此时此刻，有即将步入社会的喜悦之情，有对老师和同学不舍之情。但这就是生活，有相逢时喜悦，就有离别时伤感。在今后的人生旅途中，时刻铭记老师的教诲，乘风破浪，披荆斩棘，不畏艰险。

首先，诚挚地感谢我的恩师邓元望教授以及课题组的朱浩老师、鄂加强老师。在学业上，邓老师因材施教，启发性的指导每个学生在各自感兴趣的领域耕耘发展，并且提供一个非常好的学习交流场所；在生活上，邓老师经常给予力所能及的帮助，提倡劳逸结合的教学理念。这种良好的师生关系，增添了我研究生生涯的可能性，让我的人生多了更多的可能性。非常庆幸能来湖大读研并结识良师。

其次，感谢与我并肩奋斗的师兄弟们。不管遇到什么困难，我们总是积极面对、互相探讨、彼此勉励、共同进步。正时由于他们的陪伴，才让枯燥的科研生活焕发出新的活力。在未来，虽然彼此各奔东西，但是这份友情永存心中。

然后，感谢与我彻夜畅谈的室友、感谢在湖大认识的每一个人、感谢身边的所有朋友。三年求学生涯旅程，正是有你们相伴，才会充满盎然生机。

最后，感谢我的父母。不管我做出什么决定，他们总是义无反顾地支持我，站在我身边，给予物质上与精神上的帮助。我今天所拥有的一切都离不开我的父母。在这里，祝愿您们身体健康。

汪华军

2021年3月

附录A 攻读硕士学位期间参加的科研项目

1.高校知识产权转化项目：城市智慧储运专项

附录B 核心代码

(1)低光增强代码：

```
# 第一步：使用dlib.get_frontal_face_detector() 获得脸部位置检测器
detector = dlib.get_frontal_face_detector()
face_save_path='E:/cvpicture/lowlight_enhance_distracted_driver_detection/low_light_enhance/data/result/collect_face/'
# 图片保存命令计数
count_enhance = 0
count_face = 0
def detect_face(enhance_img, gray):
    dets = detector(gray,2)
    global count_face
    face_img = None
    for face in dets:
        #左下角（x1,y1）,右上角（x2,y2）
        x1, y1, x2, y2 = face.left(), face.top(), face.right(),face.bottom()
        #对人脸进行画框
        #cv2.rectangle(enhance_img, (x1,y1), (x2,y2), (0,255,0),2)
        # 适当放大区域，提高显示效果
        x1 = (int)(0.9*x1)
        y1 = (int)(0.9 * y1)
        x2 = (int)(1.1 * x2)
        y2 = (int)(1.1 * y2)
        face_img = enhance_img[y1:y2, x1:x2]
        face_img = cv2.resize(face_img, (300, 300)).astype(np.float32)
        cv2.imwrite(face_save_path+str(count_face)+'.jpg', face_img*255)
        count_face = count_face + 1
    return enhance_img, face_img

def transfer_32bit_to_8bit(img):
    min_32bit = np.min(img)
    max_32bit = np.max(img)
```

```
    img_8bit = np.array(np.rint(255.0*(img - min_32bit) / (max_32bit - min_32bit)),dtype=np.uint8)
    return img_8bit

def lowlight(image_path):
    global count_enhance
    os.environ['CUDA_VISIBLE_DEVICES']='0'
    data_lowlight = cv2.imread(image_path)
     # 类型转换，把PIL类型转换成numpy
    data_lowlight = (np.asarray(data_lowlight)/255.0)
    # 把numpy转换成torch
    data_lowlight = torch.from_numpy(data_lowlight).float()
    # 长宽通道变换
    data_lowlight = data_lowlight.permute(2,0,1)
    data_lowlight = data_lowlight.cuda().unsqueeze(0)
    DCE_net = model.enhance_net_nopool().cuda()
    DCE_net.load_state_dict(torch.load('E:/cvpicture/low-light_enhance_distracted_driver_detection/low_light_enhance/snapshots/Epoch99.pth'))
    start = time.time()
    _,enhanced_image,_ = DCE_net(data_lowlight)
    end_time = (time.time() - start)
    print(end_time)
    image_path = os.getcwd()
    result_path = '{}\\data\\result\\enhance_image\\'.format(image_path)
    if not os.path.exists(result_path):
        os.makedirs(result_path)

    # 把torch类型转换成numpty类型
    enhanced_image = enhanced_image.cpu()
    output = enhanced_image.detach().numpy()
    # 取照片的尺寸
    img = np.shape(output)[0:4]
    # 创建空的numpy保存图片
    result = np.empty(shape=(img[2], img[3], 3))
```

```
    # [1,3,480,640]变换成[3,480,640]
    image_another = output[0,:,:,:]
    for i in range(3):
        result[:,:,i] = image_another[i,:,:]
    #print('result11111=={}'.format(result.dtype))
    result = result.astype(np.float32)
    #print('result22222=={}'.format(result.dtype))
    gray = cv2.cvtColor(result, cv2.COLOR_BGR2GRAY)

    # 32位灰度图像变成8位灰度图像
    gray = transfer_32bit_to_8bit(gray)
    #print('gray=={}'.format(gray.dtype))
    #人脸检测
    result, face_img = detect_face(result, gray)
    '''
    cv2.imshow('wang', result)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    '''
    cv2.imwrite(result_path+str(count_enhance)+'.jpg',result*255)
    count_enhance = count_enhance+1
    return result, face_img
if __name__ == '__main__':
# test_images
    with torch.no_grad():
        filePath = 'data/test_data/'
        file_list = os.listdir(filePath)
        for file_name in file_list:
            test_list = glob.glob(filePath+file_name+r"/*")
            for image in test_list:
                # image = image
                _, face_img = lowlight(image)
```

(2)疲劳驾驶判断代码：

```
    #初始化网络
```

```
net=SSD()
net=torch.nn.DataParallel(net)
net.train(mode=False)
net.load_state_dict(torch.load('./weights/ssd300_VOC_100000.pth',map_location=la
mbda storage,loc: storage))
if torch.cuda.is_available():
    net = net.cuda()
    cudnn.benchmark = True

img_mean=(104.0,117.0,123.0)

#调用摄像头
cap=cv2.VideoCapture(0)
max_fps=0

#保存检测结果的List
#眼睛和嘴巴都是，张开为‘1’，闭合为‘0’
list_B=np.ones(15)#眼睛状态List,建议根据fps修改
list_Y=np.zeros(50)#嘴巴状态list，建议根据fps修改
list_Y1=np.ones(5)#如果在list_Y中存在list_Y1，则判定一次打哈欠，同上，长度
建议修改
blink_count=0#眨眼计数
yawn_count=0
blink_start=time.time()#炸眼时间
yawn_start=time.time()#打哈欠时间
blink_freq=0.5
yawn_freq=0
#开始检测，按‘q’退出
while(True):
    flag_B=True#是否闭眼的flag
    flag_Y=False
    num_rec=0#检测到的眼睛的数量
    start=time.time()#计时
    ret,img=cap.read()#读取图片
```

```
#检测
x=cv2.resize(img,(300,300)).astype(np.float32)
x-=img_mean
x=x.astype(np.float32)
x=x[:,:,::-1].copy()
x=torch.from_numpy(x).permute(2,0,1)
xx=Variable(x.unsqueeze(0))
if torch.cuda.is_available():
    xx=xx.cuda()
y=net(xx)
softmax=nn.Softmax(dim=-1)
detect=Detect(config.class_num,0,200,0.01,0.45)
priors=utils.default_prior_box()

loc,conf=y
loc=torch.cat([o.view(o.size(0),-1)for o in loc],1)
conf=torch.cat([o.view(o.size(0),-1)for o in conf],1)

detections=detect(
    loc.view(loc.size(0),-1,4),
    softmax(conf.view(conf.size(0),-1,config.class_num)),
    torch.cat([o.view(-1,4) for o in priors],0)
).data
labels=VOC_CLASSES
top_k=10

#将检测结果放置于图片上
scale=torch.Tensor(img.shape[1::-1]).repeat(2)
for i in range(detections.size(1)):
    j=0
    while detections[0,i,j,0]>=0.4:
        score=detections[0,i,j,0]
        label_name=labels[i-1]
        if label_name=='closed_eye':
            flag_B=False
```

```
            if label_name=='open_mouth':
                flag_Y=True
            display_txt='%s:%.2f'%(label_name,score)
            pt=(detections[0,i,j,1:]*scale).cpu().numpy()
            coords=(pt[0],pt[1]),pt[2]-pt[0]+1,pt[3]-pt[1]+1
            color=colors_tableau[i]
            cv2.rectangle(img,(pt[0],pt[1]),(pt[2],pt[3]),color,2)

    cv2.putText(img,display_txt,(int(pt[0]),int(pt[1])+10),cv2.FONT_HERSHEY_SI
MPLEX, 0.4, (255,255,255), 1, 8)
            j+=1
            num_rec+=1
    if num_rec>0:
        if flag_B:
            #print(' 1:eye-open')
            list_B=np.append(list_B,1)#睁眼为‘1’
        else:
            #print(' 0:eye-closed')
            list_B=np.append(list_B,0)#闭眼为‘0’
        list_B=np.delete(list_B,0)
        if flag_Y:
            list_Y=np.append(list_Y,1)
        else:
            list_Y=np.append(list_Y,0)
        list_Y=np.delete(list_Y,0)
    else:
        print('nothing detected')
    #print(list)
    #实时计算PERCLOS
    perclos=1-np.average(list_B)
    print('perclos={:f}'.format(perclos))
    if list_B[13]==1 and list_B[14]==0:
        #如果上一帧为’1‘，此帧为’0‘则判定为眨眼
        print('----------眨眼----------')
        blink_count+=1
```

```
blink_T=time.time()-blink_start
if blink_T>10:
    #每10秒计算一次眨眼频率
    blink_freq=blink_count/blink_T
    blink_start=time.time()
    blink_count=0
    print('blink_freq={:f}'.format(blink_freq))
#检测打哈欠
#if Yawn(list_Y,list_Y1):
if (list_Y[len(list_Y)-len(list_Y1):]==list_Y1).all():
    print('----------------------打哈欠----------------------')
    yawn_count+=1
    list_Y=np.zeros(50)
#计算打哈欠频率
yawn_T=time.time()-yawn_start
if yawn_T>60:
    yawn_freq=yawn_count/yawn_T
    yawn_start=time.time()
    yawn_count=0
    print('yawn_freq={:f}'.fomat(yawn_freq))

#此处为判断疲劳部分
if(perclos>0.12):
    print('疲劳')
elif(blink_freq<0.15):
    print('疲劳')
    blink_freq=0.5#如果因为眨眼频率判断疲劳，则初始化眨眼频率
elif(yawn_freq>0.15):
    print("疲劳")
    yawn_freq=0#初始化，同上
else:
    print('清醒')
T=time.time()-start
fps=1/T#实时在视频上显示fps
if fps>max_fps:
```

```
            max_fps=fps
        fps_txt='fps:%.2f'%(fps)
        cv2.putText(img,fps_txt,(50,50),cv2.FONT_HERSHEY_SIMPLEX, 1, (255,0,0),
2)
        cv2.imshow('network model in this paper',img)
        if cv2.waitKey(100) & 0xff == ord('q'):
            break
#print("-------end-------")
cap.release()
cv2.destroyAllWindows()
#print(max_fps)
```

(3)模型训练代码：

```
net=SSD()
net=torch.nn.DataParallel(net)
net.train(mode=False)
net.load_state_dict(torch.load('./weights/ssd300_VOC_100000.pth',map_location=la
mbda storage,loc: storage))
if torch.cuda.is_available():
    net = net.cuda()
    cudnn.benchmark = True

devkit_path='./dataset/'
annopath=os.path.join(devkit_path,'Annotations', '%s.xml')
ftest=open(devkit_path+'ImageSets/Main/test.txt','r')
img_mean=(104.0,117.0,123.0)

def parse_rec(filename):
    '''获取图片中所有的label和坐标'''
    tree=ET.parse(filename)
    objects=[]
    for obj in tree.findall('object'):
        obj_struct={}
        obj_struct['name']=obj.find('name').text
        bbox=obj.find('bndbox')
```

```
        obj_struct['bbox']=[int(bbox.find('xmin').text)-1,
                            int(bbox.find('ymin').text)-1,
                            int(bbox.find('xmax').text)-1,
                            int(bbox.find('ymax').text)-1]
        objects.append(obj_struct)

    return objects

def IoU(obj_R,obj_P):
    #计算交并比
    cood_r=obj_R['bbox']
    cood_p=obj_P['bbox']
    ixmin=max(cood_r[0],cood_p[0])
    iymin=max(cood_r[1],cood_p[1])
    ixmax=min(cood_r[2],cood_p[2])
    iymax=min(cood_r[3],cood_p[3])
    iw=max(ixmax-ixmin,0.)
    ih=max(iymax-iymin,0.)
    inters=iw*ih*1.0
    uni=((cood_r[2]-cood_r[0])*(cood_r[3]-cood_r[1])+
          (cood_p[2]-cood_p[0])*(cood_p[3]-cood_p[1])-
         inters)
    overlaps=inters/uni
    return overlaps

count=0
time_start=time.time()
accu_num=0
real_num=0

for line in ftest:
    name=line.strip()
    print(name)
    obj_real=parse_rec(devkit_path+'Annotations/'+name+'.xml')
    real_num+=len(obj_real)
```

```
    img=cv2.imread(devkit_path+'JPEGImages/'+name+'.jpg',cv2.IMREAD_COLOR
)
    x=cv2.resize(img,(300,300)).astype(np.float32)
    x-=img_mean
    x=x.astype(np.float32)
    x=x[:,:,::-1].copy()
    x=torch.from_numpy(x).permute(2,0,1)
    xx=Variable(x.unsqueeze(0))
    if torch.cuda.is_available():
        xx=xx.cuda()
    y=net(xx)
    softmax=nn.Softmax(dim=-1)
    detect=Detect(config.class_num,0,200,0.01,0.45)
    priors=utils.default_prior_box()

    loc,conf=y
    loc=torch.cat([o.view(o.size(0),-1)for o in loc],1)
    conf=torch.cat([o.view(o.size(0),-1)for o in conf],1)

    detections=detect(
        loc.view(loc.size(0),-1,4),
        softmax(conf.view(conf.size(0),-1,config.class_num)),
        torch.cat([o.view(-1,4) for o in priors],0)
    ).data
    labels=VOC_CLASSES
    top_k=10

    scale=torch.Tensor(img.shape[1::-1]).repeat(2)
    obj_pre=[]
    for i in range(detections.size(1)):
        j=0

        while detections[0,i,j,0]>=0.4:
            score=detections[0,i,j,0]
            obj={}
```

```
            obj['name']=labels[i-1]
            pt=(detections[0,i,j,1:]*scale).cpu().numpy()
            obj['bbox']=[int(pt[0]),
                         int(pt[1]),
                        int(pt[2]),
                        int(pt[3])]
            obj_pre.append(obj)

            label_name=labels[i-1]
            display_txt='%s:%.2f'%(label_name,score)
            coords=(pt[0],pt[1]),pt[2]-pt[0]+1,pt[3]-pt[1]+1
            color=colors_tableau[i]
            cv2.rectangle(img,(pt[0],pt[1]),(pt[2],pt[3]),color,2)

    cv2.putText(img,display_txt,(int(pt[0]),int(pt[1])+10),cv2.FONT_HERSHEY_SI
MPLEX, 0.4, (255,255,255), 1, 8)
            j+=1
    #把测试过的图片写入磁盘
    #cv2.imwrite('./tested/'+name+'.jpg',img)
    #print('Pic:'+name+" writed!")

    for obj_R in obj_real:
        for obj_P in obj_pre:
            if IoU(obj_R,obj_P)>0.5:#阈值暂设为0.5
                if obj_R['name']==obj_P['name']:
                    accu_num+=1
    count+=1
print("-------end-------")
elapsed=(time.time()-time_start)
print('共{:d}张图片\n用时：{:f} s\nfps={:f}\n准确率：{:f}'
      .format(count,elapsed,count/elapsed,accu_num/real_num))
```