

**本科生毕业论文（设计）**

****

**题 目 基于Yolo和LPRNet的车辆信息识别系统**

**学 院 计算机学院**

**专 业 计算机科学与技术**

**学生姓名 陈家瑞**

**学 号 2020141460290 年级 2020级**

**指导教师 辛卫**

**教务处制表**

**2024年 4 月 18 日**

**基于Yolo和LPRNet的车辆信息识别系统**

专业：计算机科学与技术

学生：陈家瑞 指导老师：辛卫

**[摘要]** 本项目旨在利用神经网络技术和计算机视觉技术实现对车辆的自动识别和信息提取。主要功能包括车辆和车牌检测与识别，以及车辆信息的统一存储。车辆检测与识别利用BIT-Vehicle Dataset数据集训练YOLOv5模型，对车辆视频或图像进行检测，以实现车辆识别。检测到车辆后，将车辆照片截取并保存，方便后续导入车牌检测与识别模型。车牌检测与识别使用CCPD Dataset数据集训练YOLOv5模型，对截取的车辆照片进行车牌检测，并利用LPRNET网络识别车牌号码。车牌颜色和车身颜色直接利用OpenCV通过划分颜色区间进行整体识别。最后，将车牌号码作为车辆信息的唯一标识，存储车辆种类、车身颜色、车牌号码、车牌颜色等信息。为用户提供便利，项目搭建了网页端平台，根据数据库存储的车辆信息搭建展示仓，包括车辆存储数量、网页点击率、车辆识别正确率等数据展示。网页端平台还提供识别图片和视频车辆的功能，用户可直接导入需要识别的图片和视频进行识别，并选择是否将识别的车辆信息存储到系统数据库中。

**[主题词]** 车辆识别 信息存储 神经网络

Vehicle information identification system based on

Yolo and LPRNet

Major: Computer Science and Technology

**Student:** Chen Jiarui **Adviser:** Xin Wei

**[Abstract]** This project aims to use neural network technology and computer vision technology to realize automatic identification and information extraction of vehicles. The main functions include vehicle and license plate detection and recognition, as well as unified storage of vehicle information. Vehicle detection and identification uses the BIT-Vehicle Dataset to train the YOLOv5 model to detect vehicle videos or images to achieve vehicle identification. After the vehicle is detected, the vehicle photo is intercepted and saved to facilitate subsequent import of the license plate detection and recognition model. License plate detection and recognition uses the CCPD Dataset data set to train the YOLOv5 model, detects the license plate on the intercepted vehicle photos, and uses the LPRNET network to identify the license plate number. The license plate color and car body color directly use OpenCV to perform overall recognition by dividing the color interval. Finally, the license plate number is used as the unique identifier of vehicle information, and information such as vehicle type, body color, license plate number, license plate color, etc. is stored. To provide convenience to users, the project has built a web platform and built a display warehouse based on the vehicle information stored in the database, including data display such as the number of vehicles stored, web page click rate, and vehicle identification accuracy. The web platform also provides the function of identifying vehicles in pictures and videos. Users can directly import the pictures and videos that need to be identified for identification, and choose whether to store the identified vehicle information in the system database.

**[Key Words]** Vehicle recognition information storage neural network

**目录**

[摘要 Ⅰ](#_Toc23862)

[Abstract Ⅱ](#_Toc23862)

[目录 Ⅲ](#_Toc23862)

[第一章 引言 1](#_Toc23862)

[1.1. 研究背景 1](#_Toc5932)

[1.2. 研究内容 1](#_Toc5932)

[1.2.1. 目标检测 1](#_Toc1895)

[1.2.2. 图片数字识别 1](#_Toc1895)

[1.2.3. 可视化界面的搭建与信息存储 1](#_Toc1895)

[1.3. 国内外研究进展 1](#_Toc5932)

[第二章 背景技术介绍 1](#_Toc23862)

[2.1. YOLOv5 1](#_Toc5932)

[2.2. LPRNET 1](#_Toc5932)

[2.3. NodeJS 1](#_Toc5932)

[2.4. Vue＆SpringBoot 1](#_Toc5932)

[第三章 系统分析与设计 1](#_Toc23862)

[3.1. 系统需求分析与建模 1](#_Toc5932)

[3.1.1. 系统需求概述 1](#_Toc1895)

[3.1.2. 需求清单 1](#_Toc1895)

[3.1.3. 系统数据分析 1](#_Toc1895)

[3.1.4. 系统功能分析 1](#_Toc1895)

[第四章 系统实现 1](#_Toc23862)

[4.1. 实现环境与工具的简要说明 1](#_Toc5932)

[4.1.1. 硬件环境 1](#_Toc1895)

[4.1.2. 软件环境 1](#_Toc1895)

[4.2. 主要程序模块实现 1](#_Toc5932)

[4.2.1. 车辆检测模型训练 1](#_Toc1895)

[4.2.2. 车牌检测模型训练 1](#_Toc1895)

[4.2.3. LPRNet模型训练 1](#_Toc1895)

[4.2.4. 端到端的车辆信息检测 1](#_Toc1895)

[4.2.5. NodeJs后端平台搭建 1](#_Toc1895)

[4.2.6. SpringBoot+Vue端搭建 1](#_Toc1895)

[第五章 工作总结与心得体会 1](#_Toc23862)

[5.1. 工作总结 1](#_Toc5932)

[5.2. 心得体会 1](#_Toc5932)

参考文献 2

致谢 2

[声明 2](#_Toc7507)

[学位论文使用授权书 2](#_Toc9446)

第一章 引言

1.1 研究背景

在智能交通系统中，车辆信息的检测是至关重要的核心任务之一。在复杂多变的自然场景下，迅速准确地检测车辆信息对于智能交通系统至关重要。随着车辆保有量的不断增加，与之相关的交通问题也变得日益严重。因此，实现对道路上车辆的检测和分类势在必行。此外，通过对这项技术的不断优化，可以为无人驾驶技术的发展以及智慧城市的建设提供一定的技术支持。

在此背景下，传统的车辆识别技术显得愈加局限，其在准确性和依赖人工干预方面的不足日益凸显。因此，研发一种高效、准确的车辆识别与信息提取系统已成为解决城市交通管理难题的迫切需求。近年来，深度学习技术的快速发展为车辆识别技术提供了新的契机。基于深度学习的目标检测算法在车辆识别领域取得了巨大进展，尤其是RCNN[1]、YOLO[2]等算法的出现使得自动化车辆识别成为可能。同时，计算机视觉技术的不断进步为车辆信息的提取提供了更多可能性，例如通过颜色、型号等特征进行车辆识别。

然而，尽管深度学习和计算机视觉技术在车辆识别领域取得了巨大进步，但仍然存在一些挑战需要克服。例如，对于复杂多变的场景和交通环境，车辆识别系统仍然面临着识别准确性和鲁棒性的问题，特别是在恶劣天气或光照条件下。此外，对于移动车辆的实时识别和跟踪也是一个挑战，需要系统能够在短时间内对车辆进行准确的检测和分类，以满足实时交通管理的需求。

对于车牌检测来说，我们可以认为它是车辆识别过程中的一个关键环节。车牌检测不仅能够提供车辆的身份信息，还是实施交通法规和管理措施的重要依据。随着技术的进步，车牌检测方法也从基于传统图像处理的算法逐渐转向采用深度学习模型，这些模型能够处理更复杂的图像背景和各种光照条件。目前，使用深度卷积神经网络（CNN[3]）进行车牌识别和检测已显示出极高的效率和准确率。例如，可以通过改进的卷积神经网络来增强特征提取的能力，从而在各种复杂环境下提高识别的准确性。此外，结合区域生成网络（RPN[4]）的方法如Faster-RCNN[5]，可以实现车牌的快速和精确定位。而在定位了车牌之后，我们还需要对车牌的字符进行识别，在这一步骤中，字符识别技术扮演着至关重要的角色。使用深度学习方法如卷积神经网络，可以对车牌上的字符进行精确分割和识别。例如，可以利用长短时记忆网络（LSTM[6]）或其他序列模型处理车牌字符的顺序，从而提高识别的准确性和可靠性。

进一步地，结合车辆检测和车牌识别的系统可以整合为一个统一的框架，通过端到端的训练方式，提升系统的整体性能。这种集成化的方法不仅可以提高处理速度，还可以降低错误识别的可能性。

想要实现端到端的车辆到车牌的识别，还有一个难题亟待解决，就是如何将识别到的车辆信息和车牌信息进行一一对应。如果车牌识别和车辆识别并行进行，会出现车牌和车辆信息无法对应的问题，在视频识别中这个问题尤为重要。在逐帧检测期间，可能一秒会产生检测出很多的车辆和车牌，这些信息如果不加以有效的处理和匹配，很容易导致数据的混乱。

为了确保车辆和车牌的识别准确率，我们需要在检测之后再次验证识别结果是否正确，这是我们必须关注的问题。因此，我们需要选择一个适合的模型，同时设计一套高效的验证机制，以提高车辆和车牌识别系统的准确性。

1.2 研究内容

1.2.1 目标检测模型

目标检测（Object Detection）的任务是找出图像中所有感兴趣的目标（物体），确定它们的类别和位置，是计算机视觉领域的核心问题之一。由于各类物体有不同的外观、形状和姿态，加上成像时光照、遮挡等因素的干扰，目标检测一直是计算机视觉领域最具有挑战性的问题。目标检测模型主要用于在图像或视频中检测和定位物体的位置，并识别物体所属的类别。它在计算机视觉领域广泛应用于诸如智能监控、自动驾驶、人脸识别、医学影像分析等方面。

在本项目中，我们需要定位图像或者视频中所有的车辆位置，所以我们需要选择一个适合的目标检测模型。近年来，随着计算机视觉和深度学习技术的不断发展，目标检测模型已经取得了显著的进步。为了在本项目中有效地定位图像或视频中的所有车辆位置，我们可以考虑使用一些先进的深度学习模型，如YOLO（You Only Look Once）、SSD[7]（Single Shot MultiBox Detector）和Faster R-CNN（Faster Regions with Convolutional Neural Network）。这些模型以其高效率和准确性而闻名，特别适用于实时视频处理和复杂场景下的车辆检测。

在本项目中，作者选择了YOLO（You Only Look Once）算法作为本项目的目标检测算法，YOLO（You Only Look Once）模型的主要优势在于它的极速处理能力和高度整合性，使其在实时目标检测领域表现卓越。与其他模型相比，YOLO将整个目标检测过程视为一个单一的回归问题，从而能在单次前向传递中直接预测出目标的位置和类别。这种方法不仅简化了传统的目标检测流程，避免了先生成候选区域再对其分类的复杂步骤，还大大提高了处理速度。因此，YOLO能够实现极高的帧率，适合应用于需要快速响应的场景。此外，随着版本的迭代，YOLO的精度也得到了显著提升，尤其是在处理小物体和复杂场景时的表现越来越好，使其不仅快速而且准确，满足了广泛的实际应用需求。

为了实现端到端的车辆到车牌的识别，就不能对一个图片和视频同时进行车辆和车牌的目标检测。作者考虑到既然YOLO模型可以识别视频和图片，那么能不能将视频和图片中识别出的车辆单独截取下来，作为车牌识别算法的输入，这样就能够完成端到端的识别，且完成车辆和车牌的统一定位和识别。这种方法通过将车辆检测与车牌识别分开处理，但仍然保持一个统一的工作流程，可以有效地提高整体识别的准确性和效率。

实际操作中，可以首先使用YOLO模型对视频或图片中的车辆进行快速且精确的定位和识别。一旦车辆被检测出，相应的图像区域（即车辆的图像部分）就可以被截取下来并单独处理。这些截取的车辆图像随后被用作车牌识别算法的输入，其中可以应用专门针对车牌特征优化的算法，如基于深度学习的字符识别技术，进一步识别出车牌上的文字和数字。

此种该设计有几个显著优势。首先，由于车辆图像已经被YOLO模型预筛选和定位，车牌识别算法可以在一个更加精确和限定的搜索范围内操作，这大大减少了背景噪声和不相关信息的干扰，提高了识别的精度。其次，这种方法使得处理流程更加模块化，便于优化和维护。每个模块可以独立进行优化和调整，而不会影响到其他部分，这对于应对不同的操作环境和需求尤为有利。

1.2.2 图片数字识别模型

图片数字识别模型主要用于识别图像中的数字，例如手写数字识别、车牌号码识别等。它在各种场景下都有广泛的应用，如自动识别验证码、数字化文档处理、手写数字识别等。数字识别技术的核心通常依赖于深度学习模型，特别是卷积神经网络（CNN）和循环神经网络（RNN），它们能够有效地从复杂背景中提取数字特征并进行准确识别。这些模型通过训练大量带有标记的图像数据，学会识别不同形式和风格的数字，从而能够应对实际应用中各种变化的挑战。

而在本项目中，作者选择了LPRNet[8]这一模型来处理车牌号码识别的任务，因为LPRNet设计用于直接从车牌图像中无需分割即可识别文字，这使得整个识别过程更加高效和准确。LPRNet结合了深度卷积神经网络和循环神经网络的特点，利用CNN层提取图像特征，随后通过RNN层处理这些特征以解析出连续的字符序列，最终通过CTC[9]（Connectionist Temporal Classification）层输出车牌上的完整文本。

这种端到端的方法不仅提升了车牌识别的速度，还因其不需要传统方法中的字符分割环节而降低了错误率。此外，LPRNet能够适应不同国家和地区的车牌格式，包括不同的字体、颜色和布局，这是因为其训练过程中包含了多样化的车牌样本。这种灵活性和鲁棒性使得LPRNet成为广泛使用的车牌识别解决方案之一。

在本项目中，作者将识别到的车牌进行裁剪，将裁剪到的图片导入到LPRNet的网络中，进行进一步的分析和识别。该过程自动执行，无需额外的人工干预，极大地提高了处理效率和减少了错误的可能性。通过这种方式，从视频监控或实时交通流中快速准确地提取车牌信息成为可能。

1.2.3 可视化界面的搭建与信息存储

可视化界面的搭建与信息存储主要用于展示和管理数据、与用户进行交互，并提供友好的用户体验。它在各种应用中都有广泛的应用，例如管理系统、数据分析工具、社交平台等。

为了能够在前端中直接调用后端的python检测模型，作者考虑是否能在前端中直接使用child\_process模块中的spawn子进程调用方法对python检测模型进行直接调用。但是由于ES5和ES6的不兼容关系，作者考虑到能否使用NodeJS独立的搭建一个后台，单独进行对python识别模型的调用。同时在NodeJS后台中开放express网络端口，让前端可以通过网络的socket方式（IP:Port）进行直接调用，同时NodeJs还会将处理到的数据返回到前端，实现一个流畅的数据交互过程。

为了实现这个方案，作者首先需要设置一个NodeJS服务器，这个服务器将使用Express框架来处理HTTP请求，并管理与Python识别模型的交互。通过使用child\_process模块中的spawn方法，NodeJS可以创建一个子进程来运行Python脚本。这种方法允许Python脚本作为一个独立进程运行，同时还可以通过标准输入输出来与NodeJS交换数据。

为了实现信息存储，作者采用了Mysql+Mybatis(Springboot)集成的框架进行后端数据管理。这种集成框架提供了一种高效且易于管理的方式来处理应用程序中的数据存取操作。用户可以在前端界面上选择是否将该识别的车辆数据放入数据库中进行存储，如果选择存储，系统将自动执行数据插入操作。这一过程通过预先配置好的MyBatis映射器实现，存入数据库后，总体数据将会在展示仓界面进行更新操作。

1.3 国内外研究现状

在2001年，Viola 和 Jones 在CVPR上发表了经典的《Rapid Object Detection using a Boosted Cascade of Simple Features》[10]和《Robust Real-Time Face Detection》[11]，提出了Viola-Jones检测器。VJ框架在AdaBoost算法的基础上，使用Haar-like小波特征和积分图技术来进行人脸检测。在深度学习出现以前，工业界的方案都是基于VJ算法。但VJ算法仍存在一些问题：Haar-like特征是一种相对简单的特征，其稳定性较低；弱分类器采用简单的决策树，容易过拟合。因此，该算法对于解决正面的 人脸效果好，对于人脸的遮挡，姿态，表情等特殊且复杂的情况，处理效果不理想；基于VJ-cascade的分类器设计，进入下一个分类器后，之前的信息都丢弃了，分类器评价一个样本不会基于样本在之前步骤中的表现，导致分类器的鲁棒性差。

基于手工特征提取的传统目标检测算法进展缓慢，且性能较差。然而，自2012年卷积神经网络（Convolutional Neural Networks, CNNs）的兴起之后，目标检测领域迎来了巨大的变革。基于CNN的目标检测算法主要分为两个技术路线：Anchor-Based[12]和Anchor-Free[13]方法。这两种方法的主要区别在于是否利用锚点（Anchor）来进行训练和预测；其中，Anchor-based方法可以进一步分为一阶段和二阶段检测算法。通常，二阶段目标检测算法在精度上优于一阶段方法，但后者在速度上更具优势。近年来，Anchor-Free算法也在不断地完善和发展。

在Anchor-based的一阶段方法中，它们直接在提取的特征基础上预测物体的分类和位置，无需生成RP。这类方法的任务流程简化为：特征提取 -> 分类/定位回归。常见的一阶段目标检测算法包括OverFeat[14]、YOLO系列、SSD和RetinaNet[15]等。

而在Anchor-based的二阶段方法中，首先生成区域建议（region proposal，简称RP），这些RP是可能包含目标对象的预选框。随后，通过卷积神经网络对这些RP进行样本分类和定位回归。二阶段目标检测算法的任务流程大致为：特征提取 -> 生成RP -> 分类/定位回归。其中常见的二阶段目标检测算法包括R-CNN、SPP-Net[16]、Fast R-CNN[17]、Faster R-CNN和R-FCN[18]等。

第二章 背景技术介绍

2.1 YOLOv5

YOLOv5（You Only Look Once version 5）是一种目标检测算法，其原理基于YOLO系列的基本思想，但在架构和细节上进行了改进和优化。与传统的目标检测算法不同，YOLOv5采用了单次检测的方法，即将整个图像作为输入，并在一次前向传递中直接预测边界框及其对应的类别和置信度。为了提高模型的适应性，YOLOv5使用了锚框（Anchor Boxes），这些是事先定义的边界框，用于在图像中提取目标，从而更好地适应不同大小和形状的目标。另外，YOLOv5采用了CSPDarknet53作为其主干网络，具有Cross Stage Partial连接（CSP连接），能够有效地提取图像特征。在特征提取和检测方面，YOLOv5将特征图传递给一系列卷积层和池化层，转换为边界框及其相应的类别概率。它还使用了一种称为"cross-stage partial connections"的连接方式，促进特征的传递和信息流动，提高了检测性能。损失函数方面，YOLOv5结合了目标位置的回归损失、目标类别的分类损失以及目标置信度的置信度损失，通过最小化这些损失函数来学习准确地检测和定位目标。训练过程包括数据加载、网络前向传播、损失计算和反向传播等步骤，一旦训练完成，模型就可以用于推理，即在新的图像上进行目标检测。

2.2 LPRNET

LPRNet（License Plate Recognition Net）是一种用于车牌识别的神经网络模型，主要用于自动识别车牌上的字符以及车牌的区域。它采用基于深度学习的端到端模型，直接从原始图像中提取特征并输出车牌上的字符序列，消除了传统方法中复杂的手工特征提取和后续处理步骤。利用卷积神经网络（CNN）来提取图像中的特征，帮助网络理解车牌上的字符以及它们的空间位置关系，其深度和宽度有助于模型学习复杂的图像模式。LPRNet的主要任务是从车牌图像中提取特征，并识别其中的字符，通过多个卷积层和池化层逐渐提取和压缩图像的特征，最后通过全连接层和Softmax层将特征映射到字符类别，并输出字符序列。为了识别不同尺度的字符和车牌，LPRNet可能会使用多尺度的特征图，经过不同层级的特征提取和池化后进行融合，获得更全面的特征表示。训练过程通常采用端到端的方式进行，在大量标记的车牌图像上进行训练，使网络学习到有效的特征提取和字符识别策略。在字符识别后可能需要进行一些后处理步骤，如字符序列的过滤和纠正，以确保识别的准确性和一致性。综上所述，LPRNet通过深度学习方法，利用卷积神经网络提取车牌图像的特征，并识别其中的字符序列，从而实现车牌识别的功能。

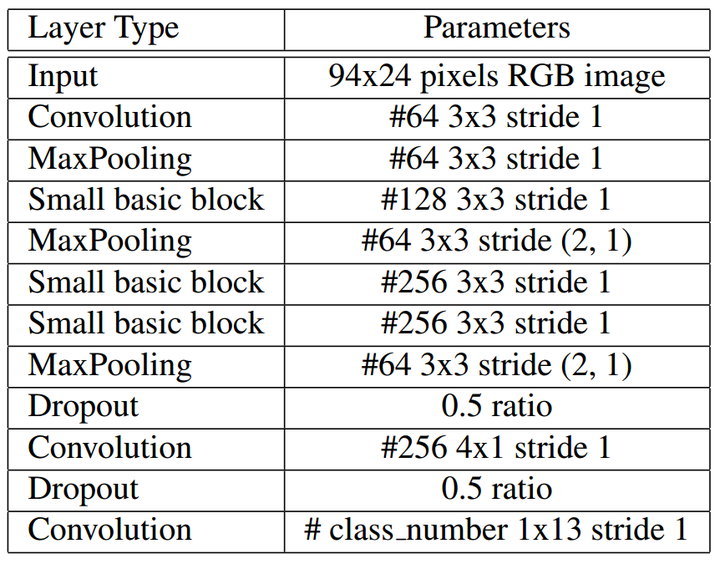


图2.1 LPRNet骨干网络构成

2.3 NodeJS

Node.js是一个基于Chrome V8引擎的JavaScript运行环境，主要用于构建高性能、可扩展的网络应用程序。它采用事件驱动的编程模型，基于事件循环机制，通过注册回调函数来处理各种事件，如I/O操作完成、HTTP请求等。通过单线程模型处理请求，所有的I/O操作都是异步的，不会阻塞主线程，同时利用底层的libuv库实现了非阻塞I/O，充分利用多核CPU。事件循环是实现异步非阻塞I/O的关键机制，负责监听事件队列中的事件，并调度执行相应的回调函数，使得Node.js能够高效地处理大量的并发请求。Node.js采用了CommonJS模块规范，允许开发者将代码模块化，提供了一系列内置模块和支持第三方模块的安装和使用。通过回调函数和Promise等机制来实现异步编程，编写非阻塞的代码，提高了程序的性能和吞吐量。借助Chrome V8引擎的优势，Node.js具有出色的性能表现，将JavaScript代码编译成本地机器码执行，采用了即时编译和垃圾回收等技术来优化性能。Node.js的原理基于事件驱动、单线程、非阻塞I/O等特性，通过事件循环和异步编程实现高效的网络应用程序开发。

2.4 Vue＆SpringBoot

Vue.js是一款流行的JavaScript前端框架，用于构建交互式的Web界面。它提供了响应式数据绑定机制，通过将数据和DOM进行绑定，当数据发生变化时，DOM会自动更新，使开发者更方便地管理和维护应用程序的状态。另外，Vue.js支持组件化开发，允许将页面拆分为独立的组件，每个组件负责特定的功能，提高了代码的可复用性和可维护性。它还利用虚拟DOM技术提高页面渲染的性能，通过将页面的状态保存在内存中的虚拟DOM树中，与实际DOM进行比较并进行更新。Vue.js提供了一系列的生命周期钩子函数，允许开发者在组件的不同阶段执行特定的操作，如数据初始化、DOM渲染、销毁等。

Spring Boot是一款用于构建Java后端应用程序的框架，基于Spring框架，简化了Spring应用程序的配置和开发过程。它采用依赖注入的方式管理应用程序中的组件和服务，通过将依赖关系定义在配置文件中或通过注解来实现，更灵活地管理应用程序的组件。Spring Boot提供了自动配置功能，根据应用程序的依赖关系和配置文件，自动配置应用程序的各种组件，减少了开发者的配置工作量。它集成了多种嵌入式Web服务器（如Tomcat、Jetty等），使开发者可以将应用程序打包成一个独立的可执行JAR文件，并运行在内置的Web服务器上。此外，Spring Boot还集成了多种ORM框架（如Hibernate、MyBatis等），方便与数据库进行交互，实现持久化操作，并提供了丰富的支持来构建RESTful风格的API，包括对HTTP请求的映射、参数验证、异常处理等功能。Vue.js用于构建交互式的前端界面，提供了响应式数据绑定和组件化开发等特性；而Spring Boot用于构建Java后端应用程序，提供了依赖注入、自动配置、嵌入式Web服务器等特性。

将Vue.js和Spring Boot结合使用可以构建出功能丰富、性能优越的全栈Web应用程序。

第三章 系统分析与设计

3.1 系统需求分析与建模

3.1.1 系统需求概述

该项目旨在确保智能交通系统能够高效准确地完成车辆检测与车牌识别任务。系统需支持视频流处理，能够检测并识别多种车辆类型，包括轿车、SUV、卡车等，同时保证在不同光照和天气条件下的鲁棒性。车牌识别系统应能处理复杂背景下的车牌图像，并准确识别车牌上的文字和数字。系统还需具备端到端的处理能力，从图像输入到输出完整的车辆和车牌信息，实现自动化的工作流程。

3.1.2 需求清单

|  |  |  |
| --- | --- | --- |
| 模块 | 功能 | 功能描述 |
| 车辆检测模块 | 实时车辆识别 | 系统应能够处理实时视频流，准确识别并定位图像或视频中的各种类型车辆，并进行自动裁剪。 |
| 车辆颜色识别 | 系统能够识别车辆的颜色，为进一步的车辆分析提供数据支持。 |
| 多类型车辆检测 | 能够识别轿车、SUV、卡车等多种车辆类型，适应不同车辆的外观特征。 |
| 重复车辆去重 | 对连续帧中检测到的车辆进行分析，避免同一辆车被重复识别。 |
| 车牌识别模块 | 自动化车牌定位 | 系统应能够准确识别出车辆图像中的车牌位置，并进行自动裁剪。 |
| 字符识别 | 对车牌上的字符进行识别，包括数字、字母及汉字，适应不同车牌颜色和字体。 |
| 车牌颜色识别 | 识别车牌的底色，辅助判断车牌类型 |
| 优化车牌图片 | 对车牌图片进行优化处理，提高识别准确率，如调整对比度、亮度等 |
| 重复号牌去重 | 确保同一车牌号码不会被多次记录，提高数据的准确性和可靠性。 |
| 用户界面模块 | 结果展示 | 提供直观的用户界面展示车辆检测和车牌识别的结果，包括车牌号码、车辆类型等信息。 |
| 数据交互 | 允许用户上传图像或视频文件，并与后端进行数据交互以获取识别结果。 |
| 界面友好性 | 设计简洁易用，确保用户能够快速理解如何操作，并对结果进行查询或存储。 |
| 数据处理模块 | 后端服务 | 使用SpringBoot+Mybatis框架实现后端逻辑，提供RESTful API支持。 |
| 数据存储 | 与MySQL数据库集成，实现车辆信息的存储、检索和更新。 |
| 系统整合 | 确保前端界面与后端服务的无缝集成，提供一致的用户体验和高效的数据处理能力。 |

表3.1 需求功能清单

3.1.3 系统数据分析



图3.2 E-R图

以下是系统各个数据实体及其所包含的数据项和数据项的相关说明。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名** | **字段类型** | **长度** | **是否为空** | **主键** | **备注** |
| carPlateNumber | varchar | 255 | NO | PRI | 车牌号码 |
| carPlateColor | varchar | 255 | NO |  | 车牌颜色 |
| carType | varchar | 255 | YES |  | 车辆类型 |
| carColor | varchar | 255 | YES |  | 车辆颜色 |

表3.3 车辆信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 长度 | 是否为空 | 主键 | 备注 |
| index | varchar | 255 | NO | PRI | 访问者类型 |
| number | int | 11 | NO |  | 访问总人数 |

表3.4 访问信息

3.1.4 系统功能分析

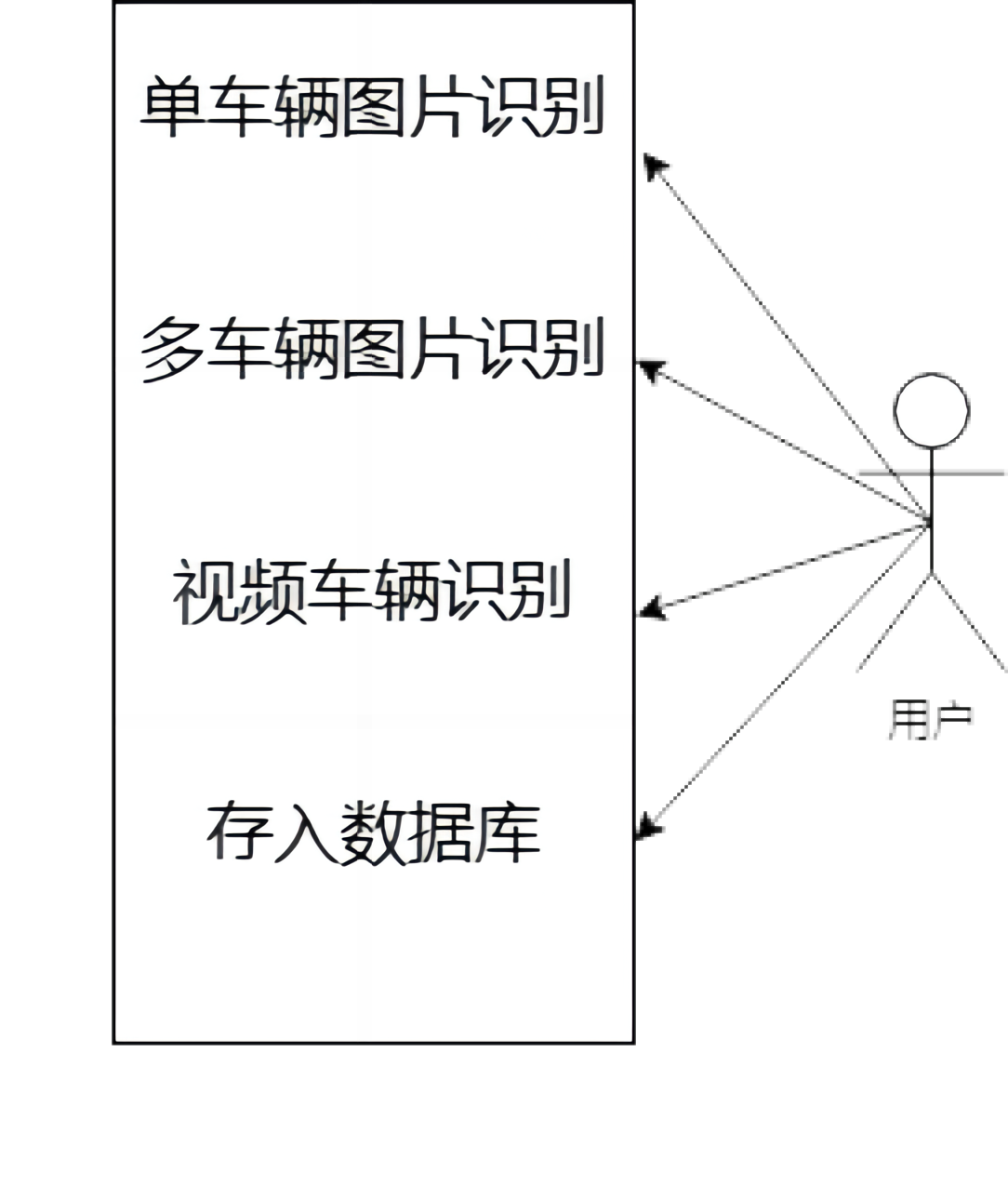


图3.5 系统用例图

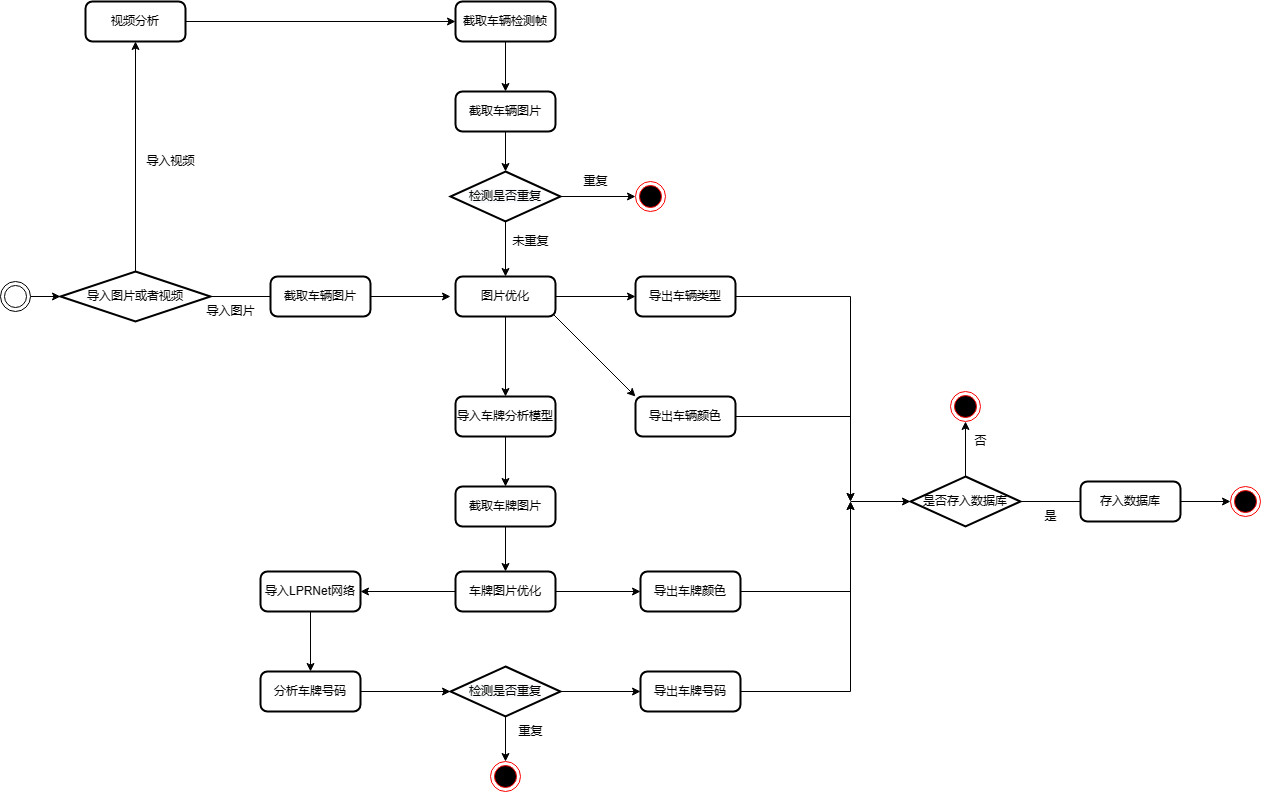


图3.6 系统活动图

第四章 系统实现

4.1 实现环境与工具的简要说明

4.1.1 硬件环境

|  |  |  |  |
| --- | --- | --- | --- |
| CPU | RAM | GPU | OS |
| Intel(R) Core(TM) i7-12700H 2.30GHz | 16.0 GB | NVIDIA RTX 3060 6G | Windows 11 |

表4.1 硬件环境

4.1.2 软件环境

|  |  |
| --- | --- |
| Python | 3.8.18 |
| Torch | 1.9.0+cu111 |
| Torchaudio | 0.9.0 |
| Torchvision | 0.10.0+cu111 |
| Numpy | 1.20.3 |
| NodeJS | 18.13.0 |
| Java | 8 |
| SpringBoot | 2.7.6 |
| Mybatis | 3.5.7 |
| Mysql | 5.7.28 |

表4.2 软件环境

## 4.2主要程序模块实现

4.2.1 车辆检测模型训练

①数据集收集和标注

为训练该模型，作者选择的是BITVehicle车辆检测数据集。BITVehicle车辆检测数据集：总共9850张图片，标注了6个类别：公共汽车(Bus)、微型客车(Microbus)、小型货车(Minivan)、轿车(Sedan)、SUV(SUV) 和卡车(Truck)。该数据集包括了两个部分，第一部分是高速上的车辆图片（格式为.jpg），第二部分是车辆图片对应的类别和方框的定位信息（格式为.xml）。



图4.3 数据集截图

在后续实验中发现如果按照该数据集进行车辆检测的训练，训练出来的模型对于高速车辆的检测来说正确率较高，但对于普通街区，复杂地形的车辆检测的正确率偏低。所以尽管模型在高速车辆检测方面表现出色，但在应对复杂街区环境中的车辆检测时存在局限性。这是由于训练数据集主要包含高速行驶的车辆图像，而缺乏足够的复杂环境下的车辆图像，导致模型未能学习到处理复杂环境所需的特征。

为了提高模型在复杂地形下的表现，作者还使用了标注工具LabelImg对部分街区的汽车图片采用YOLO标签格式进行手动标注。作者为了同时适应BITVehicle车辆检测数据集，也标注了6个类别：公共汽车(Bus)、微型客车(Microbus)、小型货车(Minivan)、轿车(Sedan)、SUV(SUV) 和卡车(Truck)（如图4.3所示）。

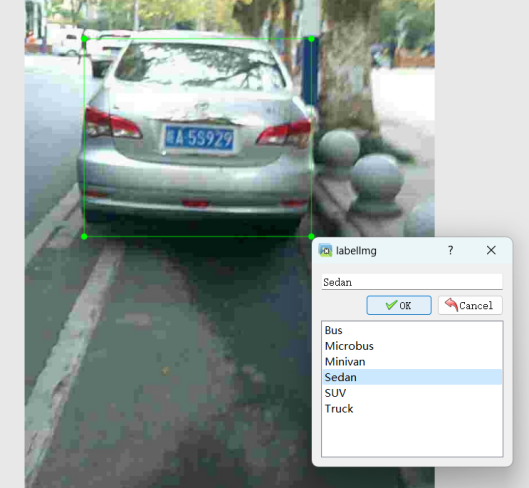


图4.4 LabelImg标注界面

②模型训练

将所有的xml格式的标注文件通过python程序转换成用于检测的txt格式后，再将大约10000张图片按照 4:1 的比例分为训练集和测试集。最终的数据集结构如下所示：其中，VOC2024文件夹保存了分割前的原始图片和标注文件；images文件夹存放了车辆图片，并且按照训练集（train）和测试集（val）进行了划分；labels文件夹存放了对应的车辆图片的txt格式标注文件，同样按照训练集（train）和测试集（val）划分。

├─images

│ ├─train

│ └─val

├─labels

│ ├─train

│ └─val

└─VOC2024

├─Annotations

├─JPEGImages

└─YOLOLabels

在配置完剩余的配置文件后，选择yolov5s.pt的预训练模型，选择epochs的次数为50次，键入以下命令让yolov5模型开始训练。python .\train.py --data .\data\car.yaml --cfg .\models\car.yaml --weights .\weights\yolov5s.pt --batch-size 8 --epochs 50

③训练结果

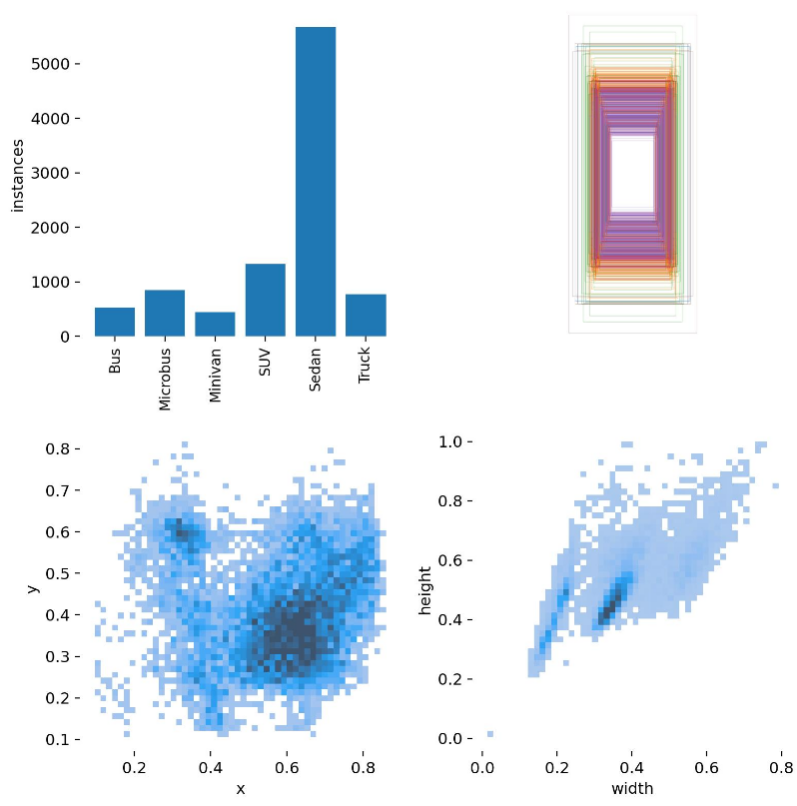


图4.5 标签分布概览

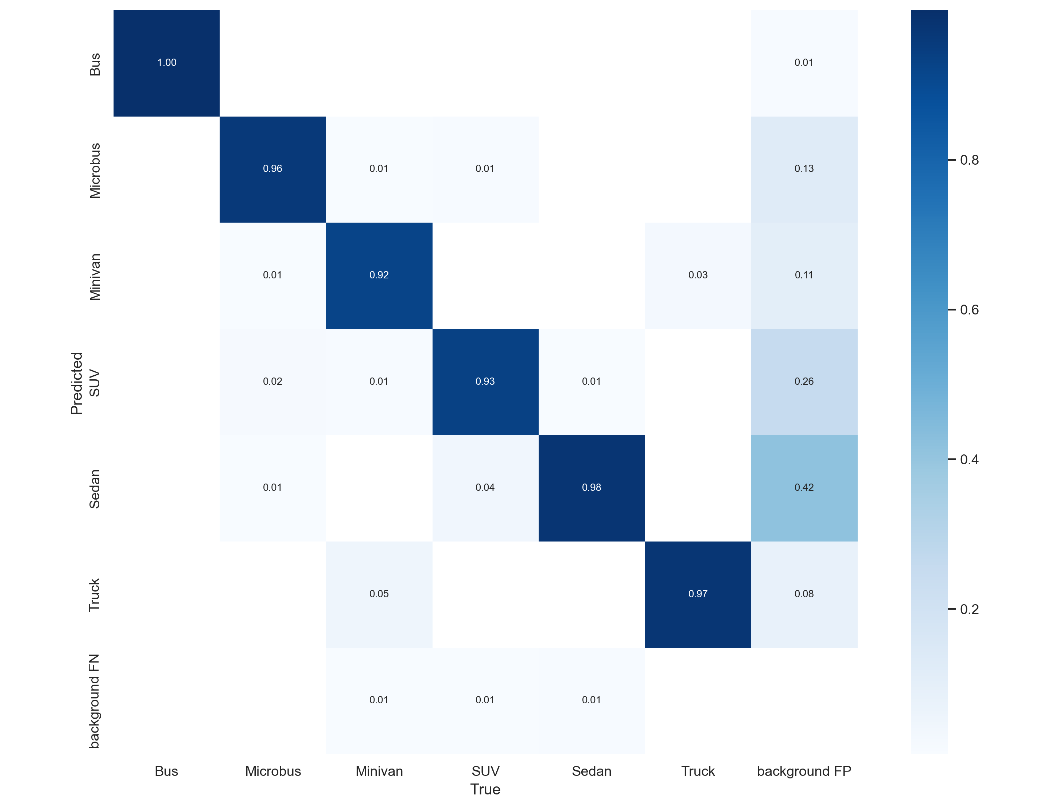


图4.6 混淆矩阵

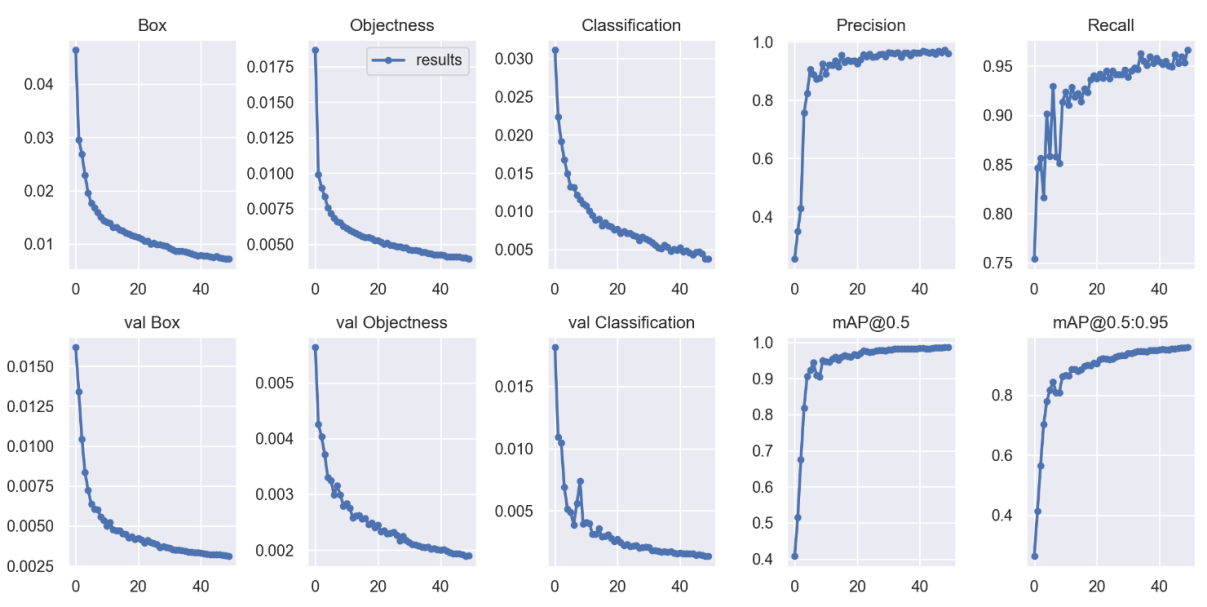


图4.7 训练过程中的损失，召回率，精准率和平均精度

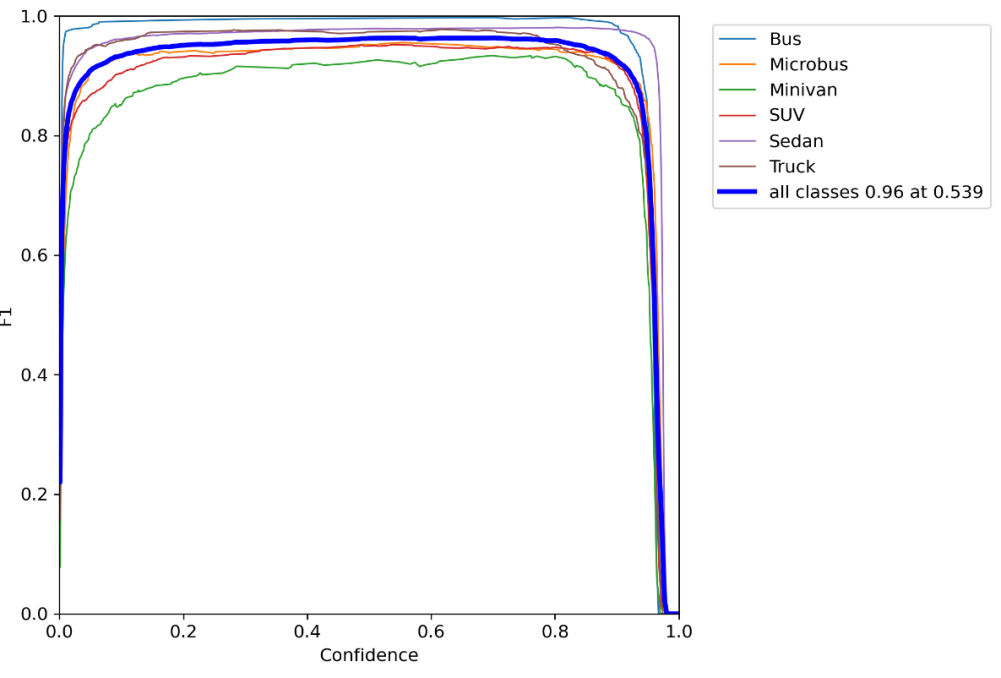


图4.8 模型F1曲线

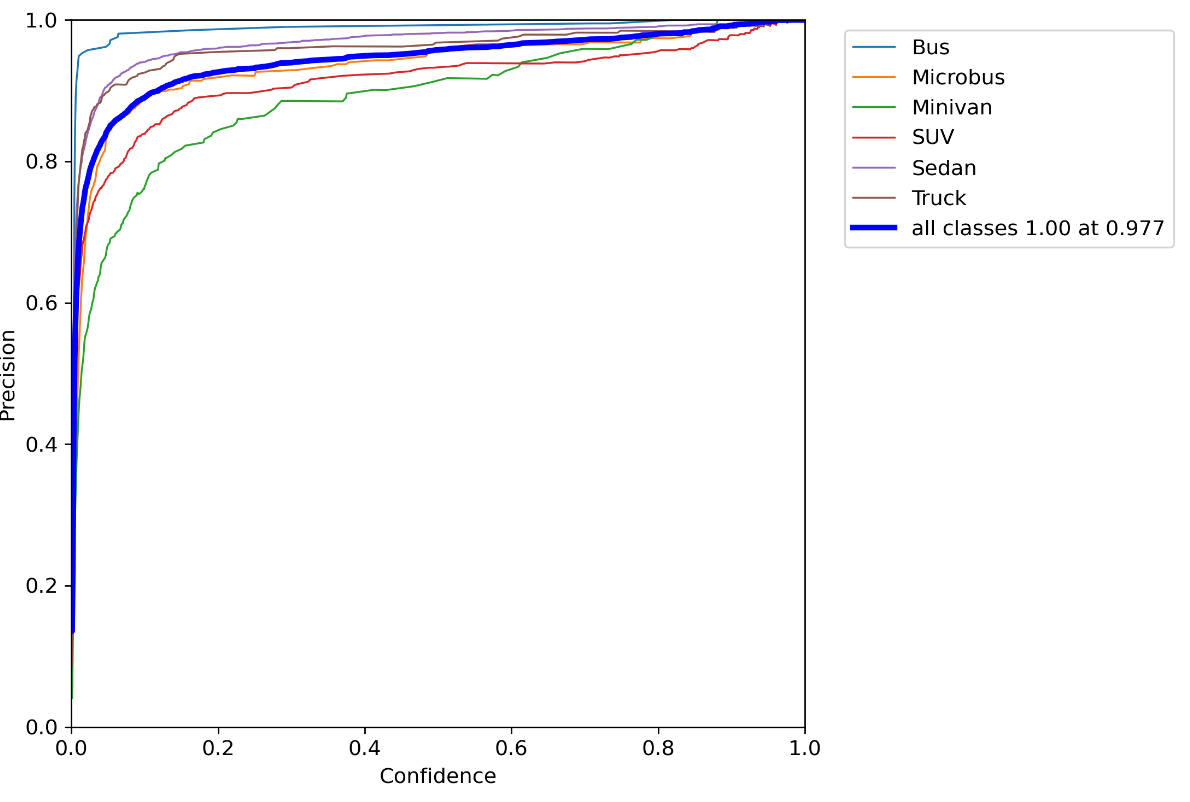


图4.9 模型P曲线

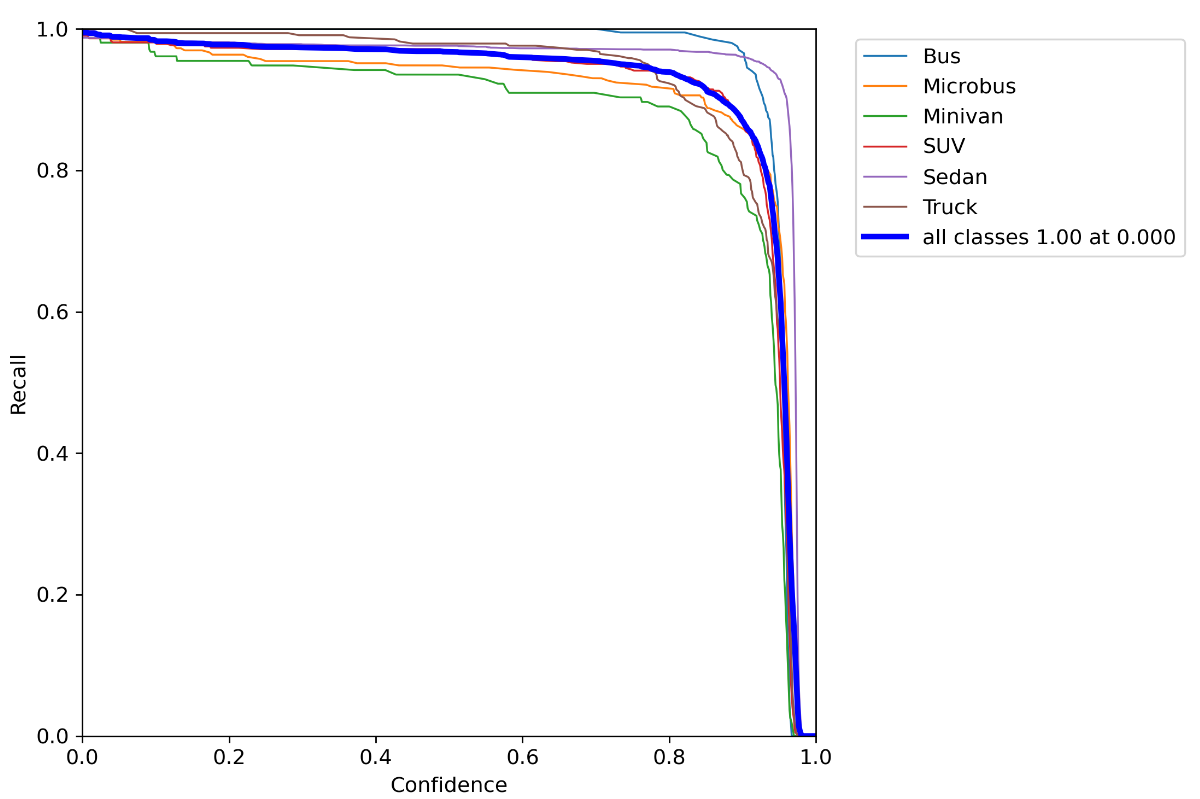


图4.10 模型R曲线

④训练结果分析

混淆矩阵展示了实际类别与模型预测类别之间的关系。从这个矩阵中可以观察到，大多数预测结果集中在对角线上，这表明模型在多数情况下能够正确地进行分类。同时，从训练过程中的损失、召回率、精确率以及平均精度的图表中可以看出，随着训练epoch数的增加，模型的性能整体呈现出改善的趋势。而模型训练的F1曲线，P曲线，R曲线，进一步验证了模型随训练进度的持续优化。F1曲线，代表精确率与召回率的平衡，随着训练的深入，其值逐渐增加，显示出模型在维持错误识别最低的同时提高正确识别的能力。P曲线，即精确率曲线，随着模型对各类别判断的准确性提高，其上升趋势明显。R曲线，表示召回率，亦随着模型对正样本的捕捉能力增强而提升。

4.2.2 车牌检测模型训练

①数据集收集和标注

车牌检测的模型我也选择的是YOLO模型，所以车牌检测的模型训练和车辆检测的模型训练总体差别不大。

为训练该模型，作者选择的是CCPD2019和CPDD2020的车牌检测数据集。CCPD是一个大型的、多样化的、经过仔细标注的中国城市车牌开源数据集。CCPD数据集主要分为CCPD2019数据集和CCPD2020(CCPD-Green)数据集。CCPD2019数据集车牌类型仅有普通车牌(蓝色车牌)，CCPD2020数据集车牌类型仅有新能源车牌(绿色车牌)。我们将两个年份的车牌放在一个文件夹中，得到大约10000张车牌图片和对应的标注文件。在CCPD数据集中，每张图片仅包含一张车牌，车牌的车牌省份主要为皖。



图4.11数据集截图

②模型训练

将所有的xml格式的标注文件通过python程序转换成用于检测的txt格式后，再将大约10000张图片按照 4:1 的比例分为训练集和测试集。

在配置完剩余的配置文件后，选择yolov5s.pt的预训练模型，选择epochs的次数为50次，键入以下命令让yolov5模型开始训练。python .\train.py --data .\data\plate.yaml --cfg .\models\plate.yaml --weights .\weights\yolov5s.pt --batch-size 8 --epochs 50

③训练结果

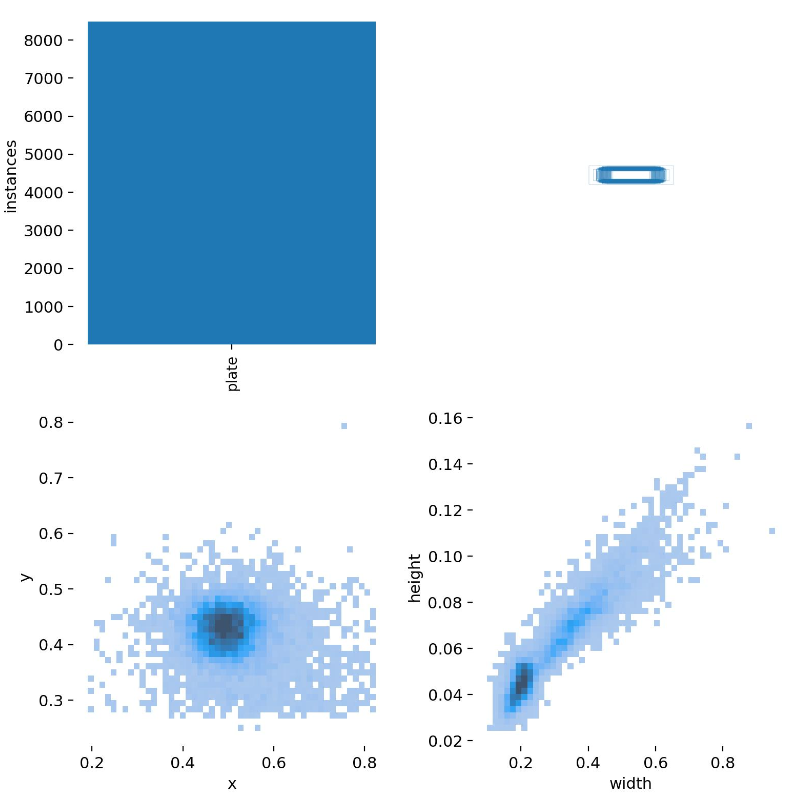


图4.12 标签分布概览

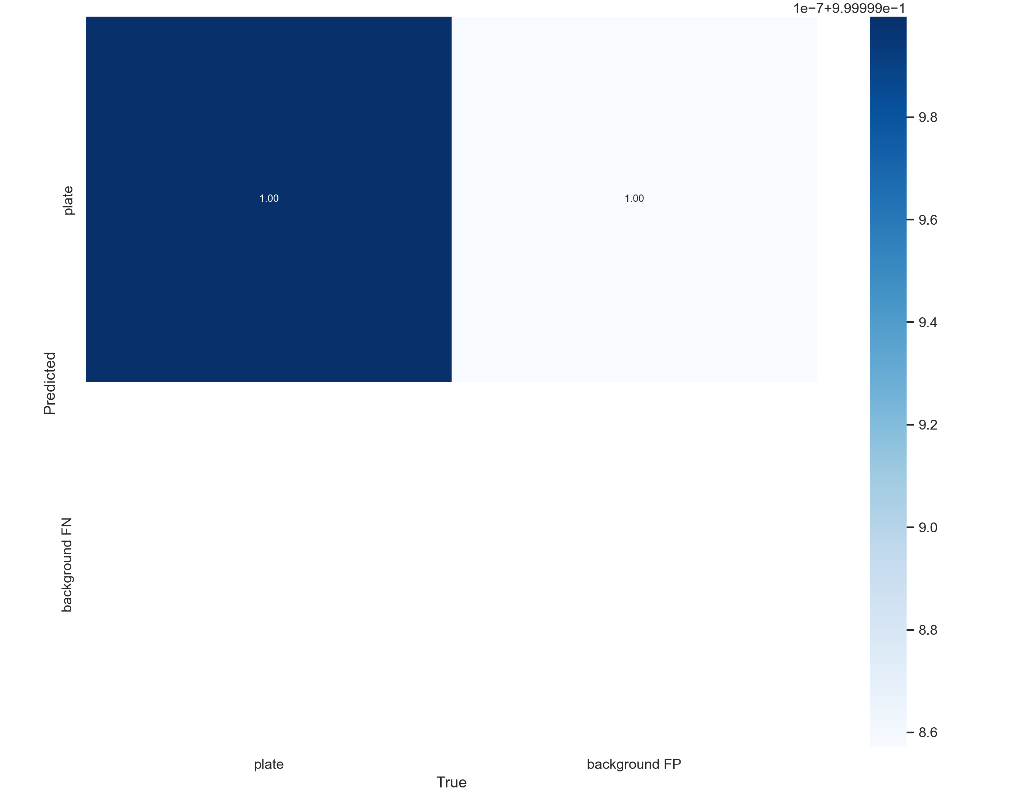


图4.13 混淆矩阵

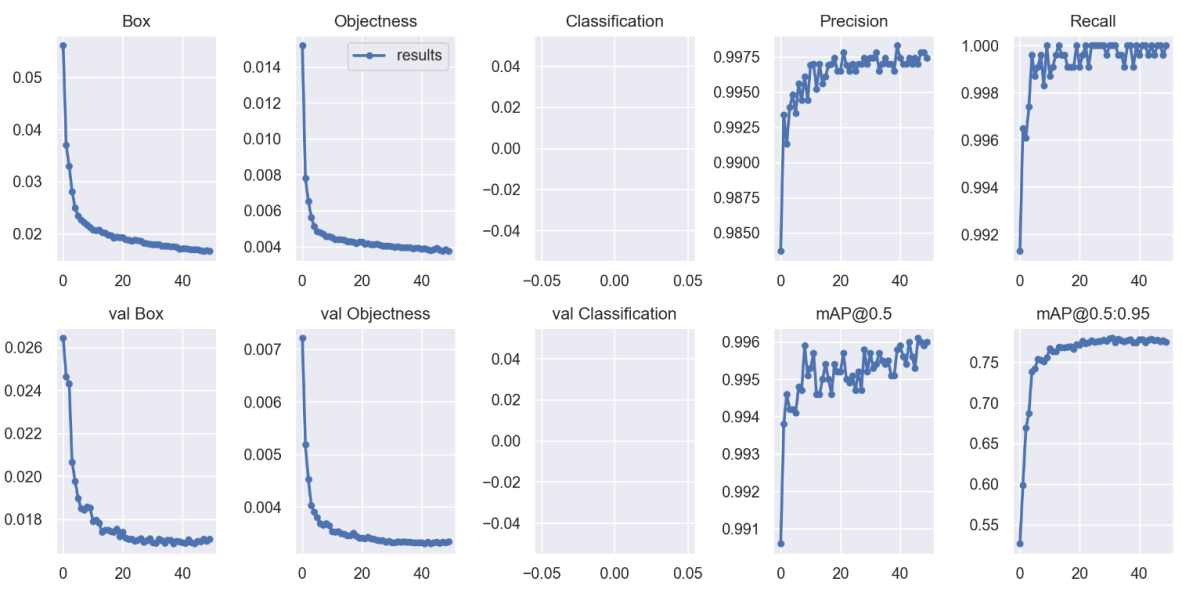


图4.14 训练过程中的损失，召回率，精准率和平均精度

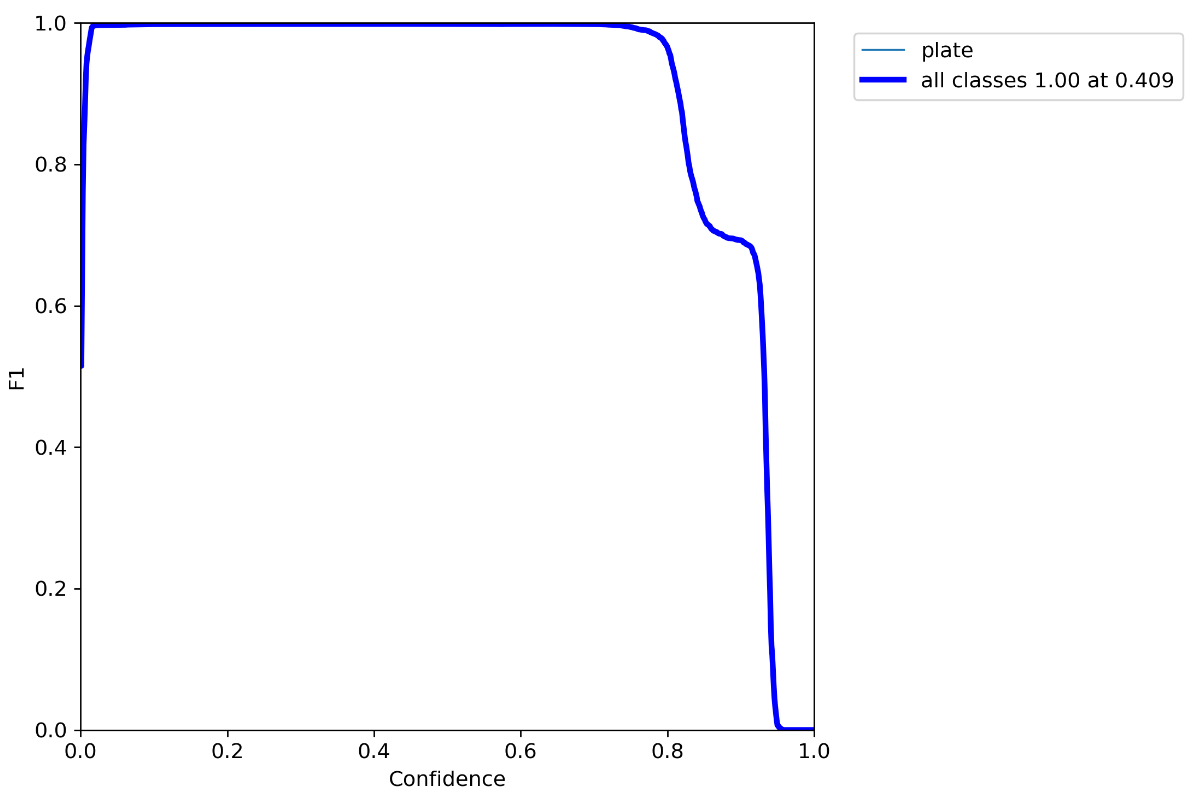


图4.15 模型F1曲线

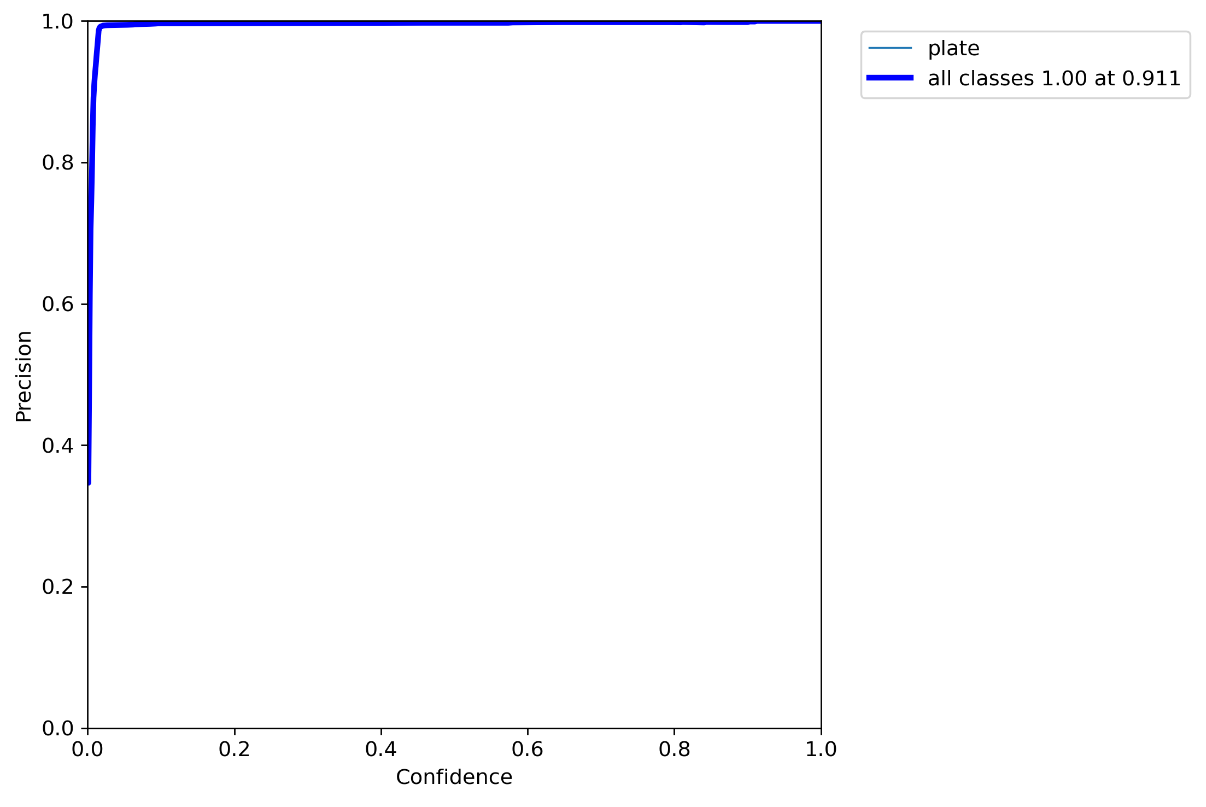


图4.16 模型P曲线



图4.17 模型R曲线

④训练结果分析

混淆矩阵展示了实际类别与模型预测类别之间的关系。但是因为只有plate一种类别，所以混淆矩阵的参照性并无太大意义。但是，从训练过程中的损失、召回率、精确率以及平均精度的图表中可以看出，随着训练epoch数的增加，模型的性能整体呈现出改善的趋势。而模型训练的F1曲线，P曲线，R曲线，亦说明随着模型的训练，模型的检测能力在提升。

4.2.3 LPRNet模型训练

LPRNet(License Plate Recognition via Deep Neural Networks)网络模型是专门为车牌号码识别所设计的网络模型，所以无需对该模型进行调整。在github的源码中，源码作者已经训练好了一个较为合适的参数，可以直接使用。在本项目中，我们直接选择该参数即可。这样可以显著减少项目开发时间和资源投入，同时确保我们能够快速部署一个高效且可靠的车牌识别系统。

4.2.4 端到端的车辆信息检测

①将图片或视频导入车辆检测网络

如果是前端对python程序进行调用，那么需要进行参数调用。当其他的parser.argument都调整好后，只需要对图片或者视频的路径进行参数调用即可。车辆检测网络会将识别到的车辆存放在detect目录中，包括车辆的原图片（对于车辆视频而言，存放的是检测到车辆的帧图片）和车辆检测信息（以txt格式存储）。



图4.18 前端进行参数调用示意代码

②对车辆图片进行裁剪和图片优化

裁剪车辆的大体步骤如下：

1)读取标注文件：首先打开一个包含车辆标注信息的文件，这通常是一个文本文件，每行包含了车辆在图片中的边界框（bounding box）坐标。

2)解析边界框信息：对于文件中的每一行，通过空格分割字符串来获取车辆的中心坐标（x\_center, y\_center）和边界框的宽度（width）和高度（height）。

3)计算原始边界框坐标：根据车辆的中心坐标和边界框的宽度和高度，计算出边界框的左上角（x1, y1）和右下角（x2, y2）坐标。

4)调整边界框大小：为了更精确地裁剪车辆，我们将边界框的宽度和高度减小了10%，从而生成一个新的边界框。

5)计算新的边界框坐标：使用调整后的宽度和高度，重新计算边界框的左上角和右下角坐标。

6)裁剪图片：根据计算出的坐标，可以使用图像处理库（如OpenCV或PIL）来裁剪原始图片，得到只包含车辆的图片。



图4.19 对检测出来的车辆进行裁剪

对于裁剪出来的车辆图片，作者对车辆图片进行图片优化，以提高图片的清晰度。大体步骤如下：

1)增强清晰度：创建一个ImageEnhance.Sharpness对象来增强图片的清晰度。enhance()方法接受一个增强因子，用于控制清晰度的增强程度。

2)应用增强：通过调用enhance()方法并传入enhancement\_factor，图片的清晰度得到增强。



图4.20 对裁剪出的车辆进行图片优化

③对裁剪的重复车辆进行去重

由于同一辆车可能在连续的帧中多次出现，因此需要进行去重处理，确保每辆车只被分析一次。这样可以在初始端就减少导入图片的数量，减少图片分析时间。通常通过比较车辆的位置、大小或在时间序列中的移动轨迹来实现。在本项目中，作者采用的是如下步骤比较确定两张图片是否为同一辆车。

1)读取图片：使用OpenCV的cv2.imread()函数读取两张车辆图片。

2)特征提取：使用特征检测和描述子提取算法--ORB从两张图片中提取特征点。ORB（Oriented FAST and Rotated BRIEF）算法是一种用于计算机视觉和图像处理的流行特征提取和匹配算法。ORB算法结合了两种不同的算法：FAST（Features from Accelerated Segment Test）和BRIEF（Binary Robust Independent Elementary Features）。ORB算法主要应用于图像中的特征点检测、方向分配和描述子生成，常用于图像匹配、目标跟踪和三维重建等领域。

3)特征匹配：使用匹配算法--暴力匹配来匹配两组特征点。暴力匹配算法（Brute-Force Matcher）是一种简单的特征匹配方法，用于在两组特征点之间寻找最佳匹配。这种方法的核心思想是将第一组中的每一对特征描述子与第二组中的所有描述子进行比较，以找到最佳匹配。暴力匹配算法由于其简单性，通常作为其他更复杂匹配算法的基准。

4)分析匹配结果：根据匹配的特征点数量来判断两张图片是否为同一辆车，当特征点大于100的时候我们认为这辆车是一辆重复检测过的车辆，从而不再进行存储和后续操作。



图4.21 判断是否为同一车辆核心代码



图4.22 相同车辆特征匹配（判断为相同车辆）

④对裁剪的车辆进行颜色分析和类型分析

在使用OpenCV对车辆图像进行颜色识别的过程中，首先通过cv2.imread()函数读取车辆的图像。随后，将BGR（Blue, Green, Red）格式存储的图像转换成HSV（Hue, Saturation, Value）颜色空间。转换到HSV颜色空间是因为HSV对于颜色的表述更接近人类视觉感知，使得颜色分割和识别变得更加有效。对于每种可能存在的车辆主要颜色（如红、黄、蓝、黑、白、灰），都会设定一个HSV阈值范围。使用cv2.inRange()函数，可以根据这些阈值创建颜色掩码，此掩码会在定义的颜色范围内的像素处显示为255，而在范围外的像素显示为0。然后，通过cv2.countNonZero()函数统计每个颜色掩码中的非零像素数，即可得到每种颜色的像素数量。最后，比较这些数量，将像素数最多的颜色认定为车辆的主导颜色。如果没有明显占优的颜色，则判定为“无法识别”。这种方法简单直观，但可能对光照条件敏感，经过实验证明，这种方法大致能够颜色识别的需求。

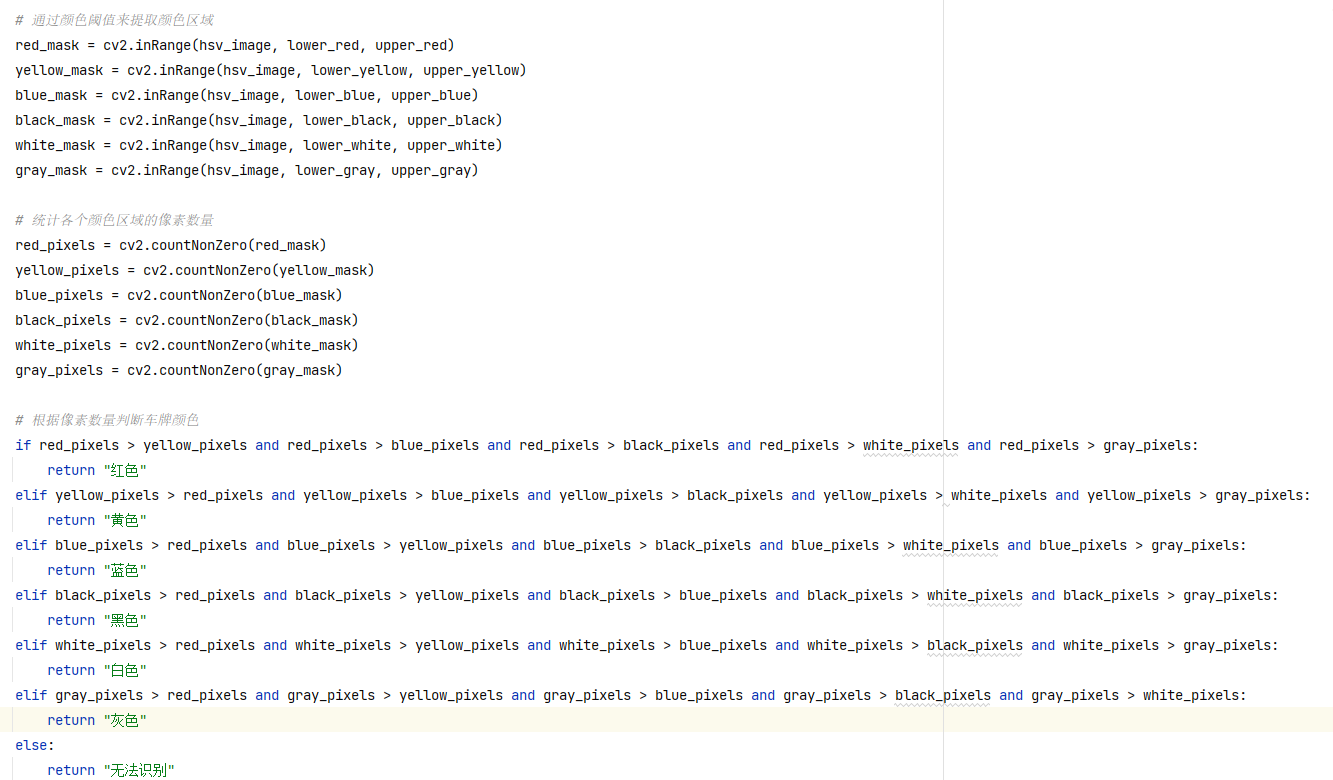


图4.23 车辆颜色分析

车辆类别分析的原理是读取一个文本文件并处理特定的行。首先打开指定的文件（车辆检测对应的文本文件，其中文件的内容包括检测出的所有车辆的类别，和检测框的四个点坐标），然后通过遍历文件的每一行，并跟踪当前的行号。当遇到与目标行号匹配的行时，函数会检查该行的第一个字符，并根据字符的不同返回预定义的车辆类型（如巴士、小型巴士、小型货车等）。如果第一个字符不匹配任何预定义的类型，则返回“未识别出类型”。一旦处理了目标行，函数将停止进一步的文件遍历并返回结果。这种方法适用于处理那些以特定格式编码车辆类型信息的文本文件，其中每行的第一个字符代表了该行所描述的车辆类型。

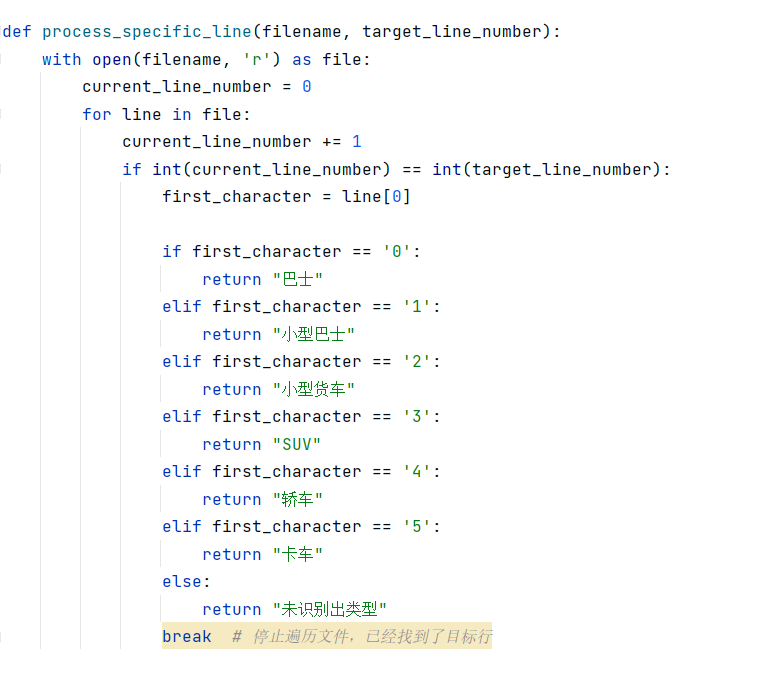


图4.24 车辆类别分析

⑤将裁剪出的车辆图片导入车牌检测网络

这一步和步骤①原理基本相同，唯一的区别就是，车辆检测的时候可以选择检测视频或者检测图片，而车牌检测一般只导入裁剪和优化过的车辆图片。这是因为为了完成端到端的检测，只能对对应的车辆进行单独的车牌检测，不能同时对视频或多车辆图片进行并行检测，导致检测信息无法一一对应。

⑥将检测出的车牌图片进行裁剪和图片优化

裁剪原理和图片优化原理同②，为了不让检测边框影响车牌整体颜色和避免边框对于边缘字符产生影响，作者将YOLO标记的边框取消，并且将检测到的车牌按比例放大2%，避免因为检测框问题让边缘字符无法全部截取成功。

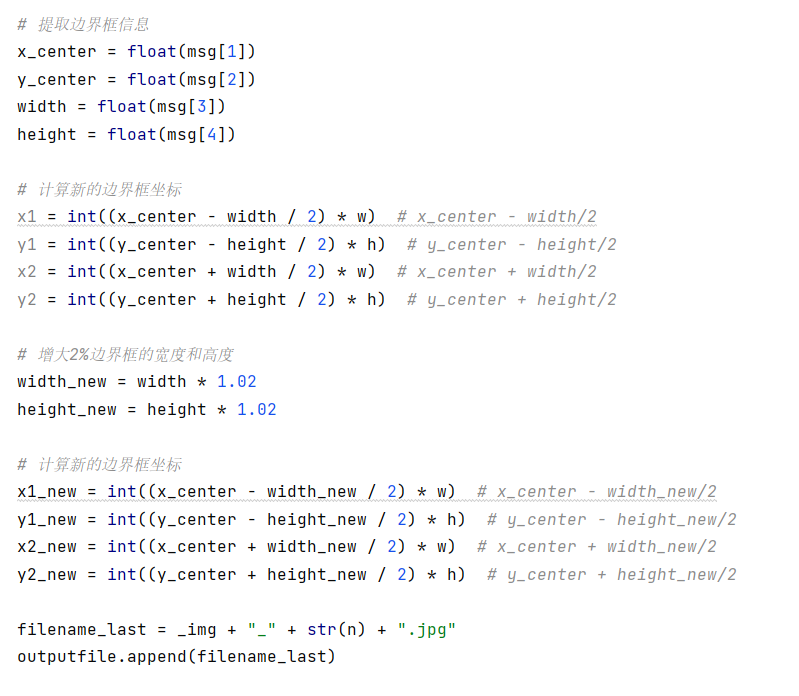


图4.25 对检测出来的车牌进行裁剪

⑦对裁剪的车牌进行颜色分析

颜色分析原理同④，但是为了让车牌的颜色识别正确率更高，我们在识别颜色之前考虑将车牌的白色和黑色部分的像素点全部清除掉了，这是因为我们不想因为车牌数字颜色影响程序对颜色的判断。但是其实这是没有必要的，因为中国车牌颜色主要是分成蓝底白色号牌（油车），绿底黑色号牌（电车），黄底黑色号牌（大型车辆）。而车牌的底色不存在黑色和白色，因此，如果我们仅比较HSV颜色空间中三种颜色（蓝、绿、黄）的阈值所生成的颜色掩码的像素数量，实际上不会受到车牌号码颜色（黑、白）的影响。



图4.26 车牌颜色分析

⑧将优化好的图片导入LPRNet网络中进行字符识别

主要实现了使用LPRNet模型对车牌图片进行字符识别的功能。首先，通过OpenCV库处理输入的车牌图像，将其大小调整为模型所需的94x24像素。接着，使用定义好的图像转换函数将图像的像素值进行标准化，并转换为适合神经网络处理的张量格式。此后，模型加载了预训练的LPRNet权重，并将处理好的图像数据送入模型进行预测。

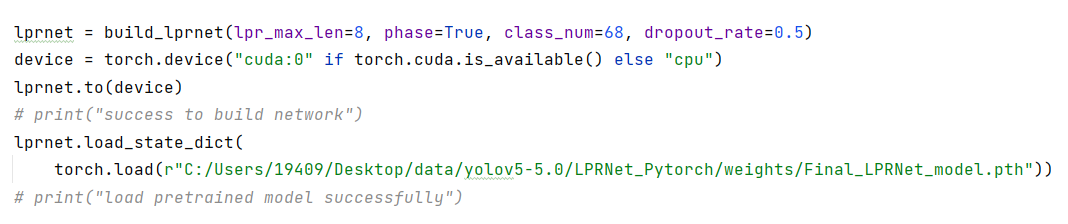


图4.27 创建LPRNet模型，并导入模型参数

模型的输出是一个概率分布数组，表示每个可能的字符在每个位置上的预测概率。模型通过选取每列概率最高的索引值来确定每个位置最可能的字符，并进行了一步去重处理，防止连续重复的字符影响识别结果，同时忽略了模型中设计的空白字符标记。得到的最终字符序列再通过一个映射过程，将索引转换回相应的字符（包括省份简称、数字和字母等），形成最终的车牌号码。

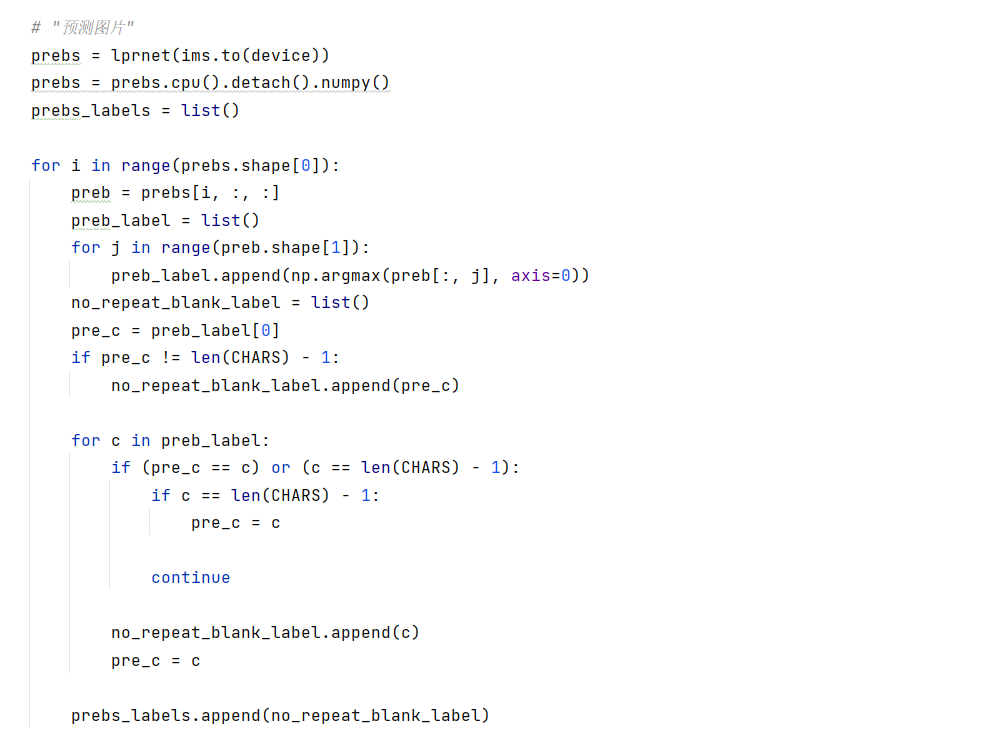


图4.28 预测号牌字符

⑨判断车牌信息是否正确并且判断车牌信息是否重复

这一个步骤主要是要判断识别出的车牌号码是否正确，车辆号牌信息作为车辆的唯一标识。如果车辆号牌信息不正确，那么我们识别出来的车辆信息将没有任何意义。目前所有的数字识别的模型都无法做到百分百正确率。包括在实验中作者也发现只有号牌足够清晰，识别出来得正确率才能保持到较高水准。所以原视频或者原图片应该尽量保证一定的清晰度，这样才能得到较为满意的识别信息。

一方面需要提升导入视频和图片的清晰度和较好的车辆形态，另一方面需要对识别出的车牌号码进行判断。中国车牌的构成是有一定格式的：蓝牌通常以省份的简称字母开头，后面跟一个字母（代表发牌地的地级市或地区），再跟5位数字/字母。黄牌结构类似与蓝牌。而绿牌通常以省份的简称字母开头，后面跟一个字母，然后是6位数字/字母组合。所以可以对检测出来的车牌号码可以和颜色相对应，并且检测是否满足对应的车牌的正则表达式。比如蓝牌和黄牌需要满足该正则表达式^[京津沪渝冀豫云辽黑湘皖鲁新苏浙赣鄂桂甘晋蒙陕吉闽贵粤青藏川宁琼使领][A-Z]([A-HJ-NP-Z\d]{4}[A-HJ-NP-Z\d挂学警港澳试超])$，而绿牌需要满足^[京津沪渝冀豫云辽黑湘皖鲁新苏浙赣鄂桂甘晋蒙陕吉闽贵粤青藏川宁琼使领][A-Z][DF][A-HJ-NP-Z\d]{5}$的正则表达式。

在检测完号牌信息是否正确后，还需要判断该号牌信息是否已经被检测过，如果是已经被检测过了的重复号牌信息，则不需要返回结果。这是因为虽然我们之前已经对相同车辆进行了判断和去重，但是实验中还是不能避免少量的相同车辆无法被检测出来。这里可以选择python中的set数据结构，可以较方便地检测车牌号码是否在同一个集合中。

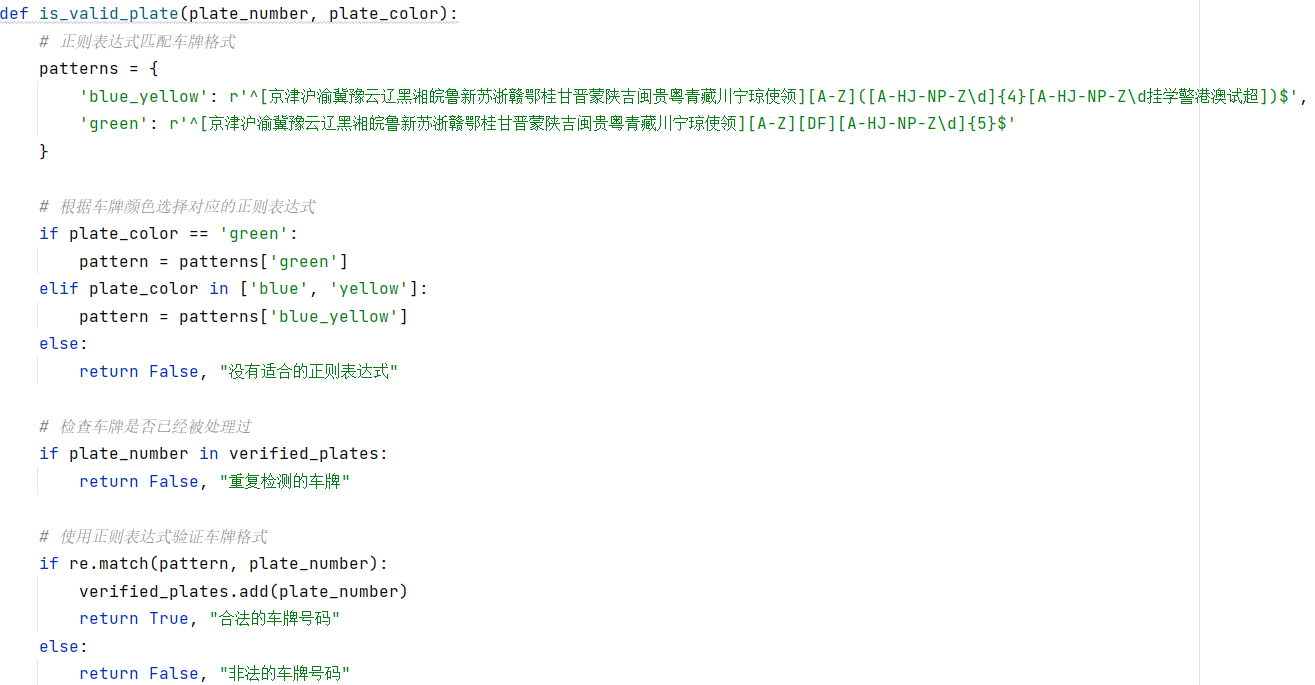


图4.29 判断号牌是否合法并去重

⑩返回车辆信息（车牌号码，车牌颜色，车辆类型，车辆颜色信息）

在完成车辆信息和车牌信息的识别和验证后，我们可以进入最终阶段——返回车辆的详细信息。我们分别在前面的步骤中得到了车辆的类型信息，车辆的颜色信息，车牌号码信息，车牌颜色信息的获取，实现了输入车辆图片或者车辆视频后，可以端到端地处理和分析整个车辆识别流程。此过程整合了从图像捕获、车辆检测、车牌识别到颜色分析和最终信息输出的各个环节，使得系统能够为用户提供一套完整的车辆信息识别方案。

4.2.5 NodeJs后端平台搭建

作者利用Node.js开发了一个后端平台，其核心功能是提供一个中间服务接口，支持车牌识别任务的前端应用和后端处理逻辑之间的通信。这包括为前端应用提供网络接口，以及启动处理图像和视频文件的后端Python脚本。为确保Python脚本在适当的环境中运行，我们设定了特定的工作目录（包括python解释器的特定环境，python脚本的特定位置）。通过使用Node.js的child\_process模块中的spawn函数，启动Python脚本，从而执行图像处理和车牌识别任务。

在识别过程中，detectCar函数负责启动一个用于车辆检测的Python脚本，而detectImage函数则启动一个车牌检测脚本。这些函数监听进程的标准输出和错误输出，一旦识别完成，它们会解析输出数据，捕捉车牌信息并将其格式化。这些函数使用正则表达式来提取输出中的必要信息，并处理多种潜在的输出格式。

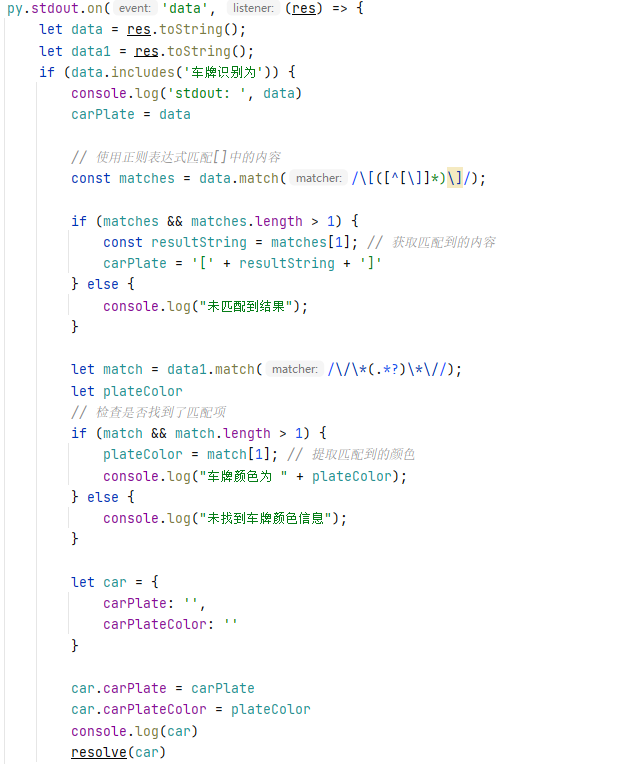


图4.30 detectImage部分代码



图4.31 detectCar部分代码

此外，该后端平台还通过定义异步函数来处理HTTP请求，包括接收图像（视频）数据，将其保存为文件，并响应客户端的请求。这使得后端可以作为前端应用的服务接口，接收用户提交的图像（视频），处理这些图像（视频），并将结果返回给用户。

4.2.6 SpringBoot+Vue端搭建

前端界面使用Vue框架开发，包括一个驾驶舱的搭建和图片（视频）车辆检测功能。

驾驶舱主要是使用了ECharts图表库，实现了动态的数据可视化展示。用户可以在系统中看到多种统计信息，如访问人数、车辆注册数量、识别车辆数及车辆识别的正确率等。此外，系统还提供了车辆类型、车辆颜色、车牌地域及车牌颜色的分析图表，通过这些图表可以直观地了解车辆分布和状态。



图4.32 驾驶舱功能图

图片（视频）车辆检测功能主要分成两个区域。左侧区域：提供图像（视频）上传功能。用户可以上传车辆图像（视频），并通过系统进行实时的车牌识别。识别结果包括车牌号码、车牌颜色、车辆类型和车辆颜色等信息，这些信息经过处理后展示在输入框中。右侧区域：展示从左侧识别的所有车辆信息，包括车牌号码和车辆类型等。用户可以选择将这些信息保存到数据库中，或进行进一步的查询和处理。

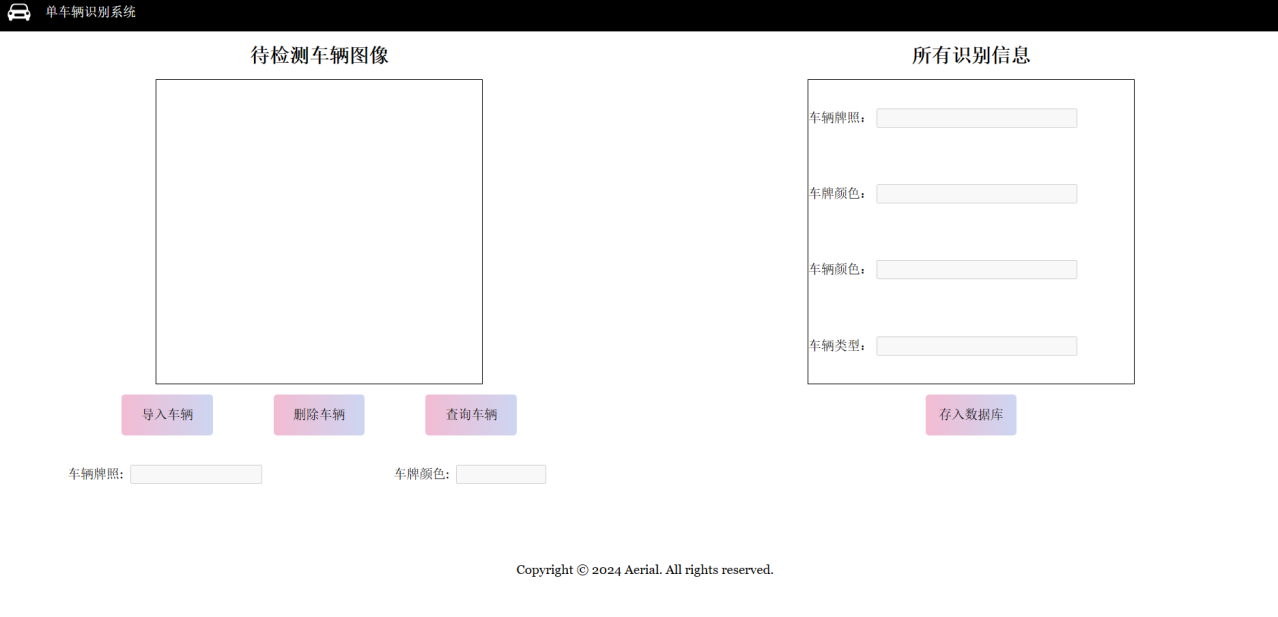


图4.33 车辆图片检测示意图



图4.34 车辆视频检测示意图

后端部分由SpringBoot + Mybatis + Mysql实现，负责处理数据存取、车辆信息管理等核心功能。后端接口包括获取车辆数量、通过车牌号查询车辆、获取所有车辆信息以及添加新车辆等功能。这些接口的实现使用了Spring的RESTful架构风格，确保了数据传输的高效和系统的易于维护性。



图4.35 Restful API风格Controller代码实现

整个系统支持车辆信息的动态展示和实时更新，还可以处理车辆图像（视频），进行车辆识别，自动提取和验证车牌信息，从而极大地自动化了车辆管理流程。用户界面设计简洁直观，确保了用户操作的便捷性，而后端的高效数据处理则保障了系统的性能和稳定性。

第五章 工作总结与心得体会

## 5.1工作总结

项目采用了YOLO（You Only Look Once）算法进行车辆检测，这是一种单次通过卷积神经网络（CNN）即可预测出图像中目标位置和类别的高效算法。YOLO通过减少计算量和优化算法结构，大幅提高了目标检测的速度，同时保持了较高的准确率。LPRNet用于车牌识别，它通过端到端的方式直接从图像中提取特征并识别字符，省去了传统OCR技术中的分割步骤，提高了识别速度和准确率。YOLO算法的快速处理能力特别适合需要实时反馈的应用场景，如交通监控和自动驾驶。它能够在极短的时间内完成图像中车辆的检测，为实时交通管理提供了技术支撑。LPRNet在设计时考虑了多种复杂情况，如不同的光照条件和复杂的背景，这使得它在实际应用中表现出良好的鲁棒性，即使在不理想的拍摄条件下也能保持较高的识别准确率。

项目实现了从车辆检测到车牌识别的端到端流程，这意味着系统可以自动处理输入的图像或视频，自动识别出车辆和车牌信息，无需人工干预，大大提高了工作效率。通过将车辆检测和车牌识别整合到一个系统中，项目不仅提高了处理速度，还通过统一的框架降低了错误识别的风险，增强了系统的可靠性。

前端界面采用Vue.js框架开发，它以响应式数据绑定和组件化开发为特点，提供了一个交互性强、用户友好的界面。这使得用户可以轻松地上传图像或视频，并获取识别结果。

后端采用SpringBoot和Mybatis，这是一个现代化的Java开发框架组合，它通过简化配置和提供自动配置功能，提高了开发效率。同时，Mybatis作为持久层框架，优化了与数据库的交互，确保了数据处理的高效性。

而模型缺点也有一些。比如模型的泛化能力有限，由于深度学习模型通常在特定的数据集上进行训练，如果数据集不够多样化，模型可能无法很好地泛化到未见过的场景。在复杂街区环境中，车辆的类型、大小、颜色以及背景的多样性都远高于单一数据集所提供的信息。因此，模型可能在这些复杂环境中的表现不如在训练数据集上的表现。而且过度依赖单一数据集可能导致模型对特定类型的车辆或场景过于敏感，而忽视了其他类型的车辆或场景。这限制了模型在多样化场景中的应用潜力。解决这一问题的方法包括构建一个更全面的数据集，涵盖更多的车辆类型、场景和条件，以及使用数据增强技术来人工增加数据集的多样性。还有一个问题是，在动态交通环境中，快速移动的车辆需要实时且连续的跟踪，这对算法的实时性提出了更高的要求。当前的模型可能在单帧图像的车辆检测上表现良好，但在连续帧之间的状态转移和跟踪上可能存在检测问题。

并且在测试过程中，作者还发现了一个较大的问题，可能需要之后去进行优化。在前端调用nodeJs的过程中，无法进行并发的对python程序进行调用。这就意味着前端调用程序的时候只能通过同步的方法进行串行调用，这会使检测速度大大降低。所幸的是作者在项目检测的过程中加入了去重的工作，将重复检测到的车辆和重复的车牌号都进行了舍弃，使得项目的该问题并没有那么严重。而且经过实验表明，检测速度能够达到基本要求。

5.2 心得体会

在完成这篇关于智能交通系统中车辆检测与车牌识别技术的论文过程中，我获得了丰富的个人成长和深刻的见解。这项研究不仅加深了我对深度学习技术，特别是YOLO和LPRNet模型在图像处理领域应用的理解，而且也锻炼了我的独立研究能力和技术实践技能。面对模型泛化能力和实时识别的挑战，我学会了自主寻找解决方案，通过大量的文献阅读、实验设计和算法调优，我提高了解决复杂问题的能力。

此外，这个过程中我也面临了项目管理的挑战，包括时间管理、资源协调和自我激励。我学会了如何在没有团队支持的情况下，依靠自己的毅力和创新思维来推进项目。通过不断的试错和迭代，我体会到了科研工作既有的不确定性和带来的成就感。

这次研究经历对我的职业生涯有着很大的影响。它不仅增强了我对机器学习和人工智能领域的热情，也让我认识到了作为一名研究者，持续学习和适应新技术的重要性。我为能够独立完成这样一个复杂的项目而感到自豪，同时，我也期待将这些经验应用到未来的研究和工作中，以解决更多的技术难题，并为智能交通系统的发展做出自己的贡献。

参考文献

1. Girshick, R. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).
2. Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
3. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25 (NIPS'12).
4. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).
5. Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137-1149.
6. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.
7. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In Proceedings of the European Conference on Computer Vision (ECCV).
8. Liu, H., & Li, W. (2018). LPRNet: License Plate Recognition via Deep Neural Networks. IEEE Transactions on Intelligent Transportation Systems, 19(1), 330-339.
9. Graves, A. (2012). Supervised Sequence Labelling with Recurrent Neural Networks. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), Advances in Neural Information Processing Systems 25 (NIPS'12).
10. Viola, P., & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
11. Viola, P., & Jones, M. (2004). Robust Real-Time Face Detection. International Journal of Computer Vision, 57(2), 137-154.
12. Zhou, X., Wang, X., & Lin, S. (2019). Anchor-Based Object Detection: A Survey. arXiv preprint arXiv:1903.09106.
13. Wang, J., Lu, X., Shen, J., Crandall, D. J., & Luo, J. (2019). Anchor-Free Object Detection: A Survey. arXiv preprint arXiv:1903.09256.
14. Sarode, S., & Chaudhari, N. (2018). OverFeat: Integrated Recognition, Localization and Detection Using Convolutional Networks. arXiv preprint arXiv:1804.03782.
15. Brisimi, T., & Gao, S. (2017). RetinaNet: Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).
16. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
17. Girshick, R. (2015). Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV).
18. Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object Detection via Region-based Fully Convolutional Networks. In Proceedings of the Advances in Neural Information Processing Systems (NIPS).

致谢

转眼间，校园生活即将结束。此篇论文完稿之际，要感谢众多师长和亲友，谢谢你们的期望与鼓励。此时此刻，我无法找到合适的言语来表达我内心深处最真挚的谢意。

首先，我要特别感谢我的导师辛卫老师，他的专业知识、严谨态度和无私精神一直是我学习的榜样。在整个研究过程中，辛卫老师提供了极具洞察力的建议和建设性的批评，这对我完成这项研究至关重要。

其次，我要感谢我的家人，他们一直是我学习和研究道路上的坚强后盾。感谢他们的理解、耐心和不断的鼓励，没有他们的支持，我无法完成这项工作。

在四川大学的四年时间里，我经历了知识的积累、思维的磨砺以及个人的成长。这段宝贵的时光，是我人生中不可或缺的一部分。

同时，我也要向所有同窗好友表示感激，感谢他们在学习和生活中的陪伴与支持。我们一起讨论问题，一起面对挑战，一起庆祝每一个小小的成就。这段友谊是我大学生活中最宝贵的财富之一。

此外，我还要感谢四川大学提供的优越学习环境和丰富的学术资源。校园内充满活力的学术氛围，以及各种学术讲座和研讨会，极大地拓宽了我的视野，丰富了我的学术经历。

在论文的最后，我也要感谢自己，感谢自己在这四年中不断努力，不懈追求学术和个人成长的每一个瞬间。这个过程中有挫折，有困惑，但更多的是成长和收获。

随着校园生活的结束，我将带着在四川大学收获的知识和经验，继续前行，迎接新的挑战。再次感谢所有给予我帮助和支持的人，是你们让这段旅程变得如此有意义。

声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得四川大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

本学位论文成果是本人在四川大学读书期间在导师指导下取得的，论文成果归四川大学所有，特此声明。

作者签名： 导师签名：

年 月 日

学位论文使用授权书

本学位论文作者完全了解四川大学有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或相关机构送交论文的原件、复印件和电子版，允许论文被查阅和借阅。本人授权四川大学将本学位论文的全部或部分内容编入有关数据库进行信息技术服务，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文，并用于学术活动。

（涉密学位论文在解密后适用于本授权书）

作者签名： 导师签名：

年 月 日