摘 要

运动车辆是现实交通行为中的主要参与者，对于愈发智能化的交通技术来说，获取准确并且实时有效的车辆信息和轨迹信息，已经成为交通信息工程中的重要任务。基于目前来说，传统的交通信息采集及获取主要采用传感器来进行实现，这些传统的采集获取系统都具有一些很大的弊端。然而随着交通监控的普及，计算机视觉技术逐渐拥有了可以应用在交通情况下进行识别与跟踪的平台基础。在本文中，通过对传统和基于深度学习的多目标识别算法和常见两种多目标跟踪算法的研究，提出了 YOLO v5 网络结合 Deep-SORT 算法的解决方案。论文主要工作如下：

(1) 通过对基于卷积神经网络的识别算法和常用的跟踪算法进行分析，结合交通流视频的图像特点，完成了基于视频进行运动车辆识别与跟踪系统的需求分析。

(2) 针对交通视频流的准确性和实时性的需求，提出了 YOLO v5 网络结合 Deep-SORT 算法的解决方案。

(3) 通过自行采集的数据集，用于训练和验证基于 YOLO 网络的目标识别器。使用开源的车辆深度特征训练数据集，用于训练 Deep-SORT 跟踪算法的深度特征权重文件。

本文是针对西安南二环路中段车流量相对较多的地区，从西安南二环路中段直接采集一段城市道路的交通信息视频，并对该段视频数据集进行了间隔帧的提取，制作成相关数据集，用于训练和验证 YOLO v5 神经网络；结合检验的结果，使用一个开源车辆深度模型数据集训练车辆深度特征权重文件，使用 Deep-SORT 算法完成目标跟踪的实现，能够实现实时并且较为准确的运动车辆多目标识别和跟踪。

关键词：车辆识别，多目标跟踪，yolo，deep-sort

ABSTRACT

The moving vehicle is the main participant in the real traffic behavior. For the increasingly intelligent traffic technology, it has become an important task in the traffic information engineering to obtain accurate and real-time effective vehicle information and trajectory information. At present, the traditional traffic information collection and acquisition is mainly realized by sensors, and these traditional collection and acquisition systems all have some great drawbacks. However, with the popularity of traffic surveillance, computer vision technology gradually has a platform foundation that can be applied to identify and track traffic. In this paper, through the study of traditional and Deep learning-based target recognition algorithms and two common multi-target tracking algorithms, a solution of YOLO V5 network combined with deep-sort algorithm is proposed. The main work of this paper is as follows:

(1) Through the analysis of the recognition algorithm based on convolutional neural network and common tracking algorithm, combined with the image characteristics of traffic flow video, the demand analysis of the moving vehicle recognition and tracking system based on video is completed.

(2) To meet the requirements of accuracy and real-time performance of traffic video stream, a solution combining YOLO V5 network with deep-sort algorithm was proposed.

(3) The self-collected data set is used to train and verify the target recognizer based on YOLO network. An open-source vehicle depth feature training dataset is used to train the depth feature weight file of the deep-sort tracking algorithm.

In this paper, the traffic information video of urban road is directly collected from the middle section of Xi 'an South Second Ring Road where the traffic flow is relatively large, and the interval frame is extracted from the video data set to produce the relevant data set, which is used to train and verify the YOLO V5 neural network. Combined with the inspection results, an open source vehicle depth model data set is used to train the vehicle depth feature weight file, and the deep-sort algorithm is used to complete the target tracking, which can realize real-time and relatively accurate multi-target recognition and tracking of moving vehicles.

KEY WORDS: vehicle recognition, multi-object tracking, yolo, deep-sort

目 录

第一章 绪论

1.1 论文的研究背景及意义

随着我国经济总量的不断上升，越来越多的国人在出行方式上选择使用汽车出行，从而导致许多交通方面的问题比如容易产生拥堵以及车祸高发情况的出现，因此，研究一套成熟并且可实用性高的智能交通系统并且将其成功开发出来具有非常大的必要性。在智能交通系统的研究中，可以发现，由于对于道路目标识别的需求，道路上的车辆以及行人等主要的主体信息的采集获取及识别处理已经成为一个极其重要的课题。基于目前来说，以往过去的交通信息采集及获取的过程主要是基于传感器系统，包括波频采集系统、磁频采集系统[1]。这些传统的数据采集和获取系统都在设计上具有一些很大的缺点，这两种方法都很容易受到其他因素的干扰而导致结果不那么准确，并且其维护成本也较高。随着高级图像采集设备的出现，计算机视觉技术获得了大量的数据集从而得以飞速发展，同时该技术在智能交通系统领域也变得愈来愈普遍地被使用。在计算机视觉领域中，通常将获取当前图像或者视频中的目标位置称为目标识别，对于在完成目标识别的基础上对进行目标轨迹的提取称作是目标追踪。目标识别和跟踪被广泛的应用于各个领域[2]。

此外，在实际交通视频的情况中，由于车辆本身大部分是处于运动状态的，所以对于运动状态的车辆进行研究是十分有必要的。同时，大多数交通视频中车辆是并行出现的，是同时存在的，同时由于摄像头的不同会出现遮挡的现象，并且各个车辆之间的相似度也很高，不是很容易清晰的进行分辨不同，所以对于车辆的识别和跟踪技术来说。对实时性鲁棒性和准确度的要求都很高。而目前的相当一部分算法无法实现这般需求。因此，研究基于实际情况下多目标车辆的实时识别与跟踪算法对于道路交通信息的自动提取和运行状态管理具有很重要的意义。

本课题旨在于研究在城市中的实际交通视频中，如何使用合适的目标识别算法对处于运动状态的车辆进行识别，以便于达成能够高准确度并且快速地获取得到车辆目标信息的要求。此外需要在实现运动车辆目标识别的基础上，如何使用这种识别的结果针对跟踪算法进行初始化，从而能够可以在未来的后续工作内容中完全实现对车辆的跟踪并且能够保证其实时性。随着本篇论文中的这项技术的成功实现，我们可以通过这项技术，能够进一步地获取当前车辆的行驶状态信息，诸如车辆的当前位置和速度等信息。基于目前的主要交通问题和智能交通系统的研究情况来看，这些信息的获取将会对具体交通情况中车辆是否超速、是否违章进行有效的判定，并且可以实现对违法车辆的具体位置的确定和跟踪[3]，从而在交通领域大大地减少人力和物力开销，很大程度上能够实现交通领域的智能化。

1.2 国内外研究现状

目标识别及跟踪主要分为单目标识别和跟踪以及多目标识别和跟踪两个方面，本课题所研究的基于视频的运动车辆目标识别与跟踪是以多目标识别为基础，不断获取新的目标的位置信息从而获取目标的轨迹信息，实现多目标跟踪。本文主要研究视频环境下的车辆识别与跟踪，下面将会对车辆目标识别和目标跟踪两个领域的国内外的研究现状进行分析。

1.2.1 基于视频的车辆识别算法研究现状

在 20 世纪初的某个时候，随着数字计算机信息技术的出现初步得到了发展，作为其中非常重要并且必不可少的分支领域之一的计算机信息和视觉技术下的数字图像信息处理技术也已经开始大量出现。21 世纪以来，目标识别算法日趋成熟，成为了计算机视觉研究中的一个重要的研究分支。随着智能优化算法、深度学习等技术的发展，之后相关技术人员开始着重将该技术应用于计算机视觉领域，交通中的目标识别技术也在众多学者的努力下得到了大力的发展并广泛地应用于人们生活的方方面面[4]。

针对于我国及国外现代传统运动目标识别算法的分析，发现研究工作人员的主要研究建议基本上都是在充分地考虑了基于前文所介绍的一些运动识别算法的相关理论基础上，来开展如何引入一些新算子的实际应用操作，从而对于传统的算法进行了改善用于解决它们在于实际应用研究中的困难。例如，Kim[5]等人提出了一种利用 Harris 算子代替传统光流场算法中运动场概念的研究提议，使用这种计算方法可以很有效地减少传统的光流法的运算量，大大提升目标识别的效率；德国学者 Bernd Kitt 和 Benjiamin Ranft[6]等提出了一种通过将扩展卡尔曼滤波与光流场的计算相互结合的一种研究提议，该方法可用于自动监控系统中的识别算法，可同时对多目标进行识别；Minoh M 和 Kameda Y[7]提出了三帧差分法，该方法是基于帧差分法改进的；甘明刚[8]结合边缘识别和三帧差分进行目标识别，识别结果更加准确；林明秀[9]等提出一种首先将融合帧间差分和背景差分进行融合，之后再加入灰度相关性分析对目标状态进行判定的研究方法。

然而，针对于本课题，当待识别目标处于运动状态时，可能会有诸如遮挡等情况的出现，故而使得以上方法都存在一些问题与不足之处，导致上述算法不是特别适合这种情况下的目标识别。为解决上述的问题，本文引入基于神经网络的目标识别算法，不需要过多地考虑行车环境下目标与摄像头的相对运动状态以及遮挡等情况，直接对车辆进行目标识别。

1.2.2 基于视频的车辆跟踪算法研究现状

在上世纪 60 年代初，目标跟踪被学者们提出，但由于当时计算机在硬件方面的性能，尤其是在处理音频、图像和视频等大规模多媒体数据上运算能力较差而导致的不足，使得其发展速度远不及前文所描述的运动目标识别算法在当时那么迅速。等到 80 年代左右，随着计算机硬件技术的发展，视频跟踪技术才进入较为快速的发展阶段。较为成熟的 Mean shift

算法由 Comaniciu[10]等人提出，该方法中指出，向量总是向着一个稳态点即一个概率密度函数最大的点趋近并且表现出收敛的特性，当然这是在一定的特定场景下，可以利用这一特性用于目标跟踪算法的实现；王小峰[11]等提出了一种融合多维混合高斯模型与帧间差分算法，从而实现可自适应的背景建模方法的研究提议；在 21 世纪初的时候，搜索算法开始成为目标识别与跟踪算法的主要研究对象，如均值滤波、粒子滤波和 Kalman 滤波[12, 13]，研究人员开始考虑使用相关滤波方法对目标运动状态进行预测；Reid D[14]提出了一种利用在最优运动估计基础上的 Kalman 滤波目标跟踪算法，通过对目标位置进行相应预测，生成预测结果，之后对该预测结果进行一定的最优化操作，从而用于目标跟踪；Kalal Z[15]等人在 2012 年提出了 TLD（Tracking Learning Detection）跟踪算法，该方法主要是要将一段很长的目标跟踪过程进行分段简化，比如可以分解为检测和学习，最后还有跟踪的阶段，然后对不同帧之间的跟踪结果给出检测器的错误率并且不断进行跟踪；Henriques J F[16]等人提出了一种利用核相关滤波的研究提议，这种所提出来的算法在运算速度上表现极佳。

当待跟踪目标处于运动状态时，帧间差分、背景差分等方法将会不能够准确地将前后帧之间目标的连贯性等信息判断出来，所以本课题需要使用一种不同的方法来实现。在此论文中，我将在实现目标识别的基础上，引入深度学习概念，使用 Deep－SORT 算法，从而达到实现实时跟踪目标的目的。

1.3 论文主要研究内容及方法

1.3.1 主要研究内容

本篇论文的研究目标主要是在对城市交通车辆的运动视频实际应用场景下，基于该类型的视频进行城市交通运动中车辆的识别和跟踪，主要研究内容为：

(1) 完成了基于视频进行运动车辆识别与跟踪系统的整体构建。对基于视频进行运动车辆的识别与跟踪系统进行需求分析和整体架构设计，在车辆目标识别过程中，通过对比多种目标识别算法，给出合适的针对与本论文研究对象交通视频中的车辆的目标识别算法，并且进行相关的分析；在车辆目标跟踪过程中，通过对比多种目标跟踪算法，给出合适的针对与本论文研究对象交通视频中的车辆的目标跟踪算法，并且进行相关的分析。

(2) 针对交通视频中的准确性和实时性的需求，提出了 YOLO v5 网络结合 Deep-SORT 算法的解决方案。采用 YOLO 网络作为基线网络，利用自采数据集对网络进行训练，结合车辆跟踪算法，得到可用于对跟踪算法进行初始化的车辆检测结果。之后，结合前述算法所得到车辆检测结果与 Deep-Sort 实现车辆跟踪。

(3) 使用自行采集的数据集，测试集视频时长为 32s，训练集包括 1166 张图片，用于训练和验证基于 YOLO 网络的目标识别器；使用开源的车辆深度特征数据集训练模型，得到 Deep-SORT 跟踪算法的深度特征权重文件。然后在基于权重文件的基础上，最后测得所提

出方案的具有较为准确的优越性。

1.3.2 论文结构安排

全文总共分为五个部分，具体做了如下安排：

第一章，绪论。在本篇绪论中对所涉及的研究课题的历史发展背景和其对研究的重要性进行了简单的阐述，接着对基于视频的汽车识别和跟踪技术的研究现状问题进行分析，最后在基于前面的分析研究现状问题的基础上，给出本篇论文所需要使用的研究手段和方法，同时对于本篇论文的主要研究内容和框架安排也给出具体的说明。

第二章，车辆目标识别与跟踪算法架构分析。对于本论文要求研究的车辆对象，对当前时代中目标识别和多目标跟踪的主流相关算法进行比对和分析，基于此，将本篇论文所要使用的基本技术路线提出。

第三章，环境搭建与系统设计。对于本论文所需要的开发环境搭建过程进行阐述，并且在基于前文中算法架构分析的基础上，采用 YOLOv5 和 Deep-SORT 预训练权重分别对识别和跟踪功能进行测试，体现该系统的可行性，然后给出所要实现的技术路线的具体设计思路。

第四章，系统实现与结果分析。本章中采用自制作的采集自西安市南二环中段的车辆数据集，以 70%比例的单帧图像进行神经网络的训练，30%比例的单帧图像通过合成视频来作为测试，从而生成相关的训练模型权重文件，同时在基于自行提取的深度特征的基础上，来实现跟踪功能并且展现实验效果，同时对实现结果进行分析。

结论与展望，对于本文基于视频的识别和跟踪算法进行了成果总结和阐述，并且将此成果结合分析当今社会的实际应用情况和我国智能轨道交通在未来的主要应用以及发展的大方向特点进行了技术分析和理论展望

1.3.3 主要研究方法

首先，通过查阅相关书籍及文献，学习深度学习的相关知识，并且安装编程软件和相关函数库，完成设计前期准备。

接着，制作数据集，对输入的视频图像进行归一化尺度缩放，对神经网络进行训练和测试，实现车辆识别功能。将 YOLO v5 识别出的车辆目标识别框与来自上一帧的卡尔曼滤波预测框继续对比，分别计算两者之间的马氏距离和特征向量最小余弦距离，融合成一个的关联矩阵。对跟踪得到的预测框和当前帧识别框，使用匈牙利算法进行匹配，如果匹配失败，就认为距离上次匹配到目标过去的时间 T 超出了设定的临界值，那么就删除这条轨迹，否则轨迹继续保持；如果匹配成功，就认为该目标处于被识别到的状态，没有跟丢过，可以作为最终结果输出。该过程如图 1-1 所示：

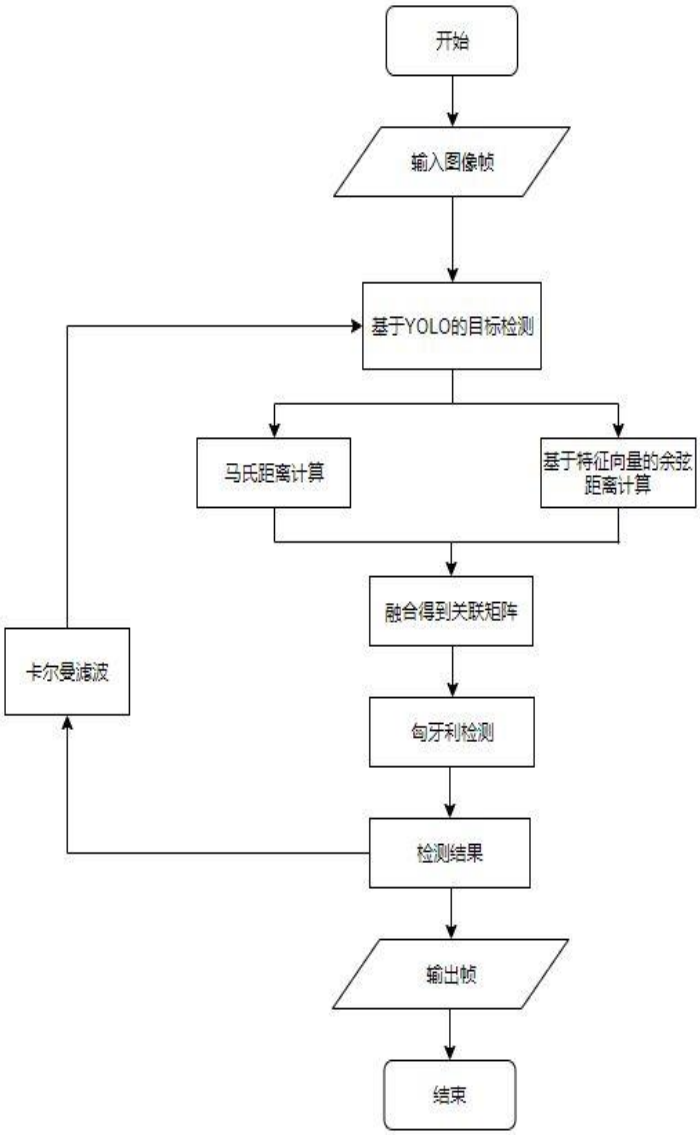

图 1-1 主要研究方法流程图

1.4 本章小结

本章首先简要阐明了本篇论文的研究背景及其社会意义，然后分别对当今目标识别与跟踪算法技术在国内外应用中的研究发展现状进行描述和分析，最后针对本文的主要研究内容、章节框架结构及其技术思路给出具体的介绍。

第二章 车辆目标识别与跟踪算法架构分析

本章的目的在于，结合当今社会背景及车辆目标识别与跟踪技术的研究现状进行需求分析，对于当今较为典型的基于区域卷积神经网络的目标识别算法以及目标跟踪算法进行分析，给出所要使用的基于视频的车辆识别与跟踪技术实现框架。

2.1 目标识别与跟踪需求分析

目标识别与跟踪技术在当今社会广泛应用于各个行业和各个领域，比如说比较热门的医学图像处理、人脸识别和姿态估计等领域。虽然目前几乎所有的通用算法都可以达成目标识别与跟踪的需求，但是对于本论文所针对的交通视频中车辆这一研究对象来说，其主要要求的性能和其他领域又有所不同。对于车辆来说，车辆的交通安全肯定是第一考虑的要素，从交通安全角度来看，基于视频的车辆目标识别与跟踪架构就有着很高的实时性和精确度方面的要求；其次，对于视频中运动的车辆来说，对于识别和跟踪算法的精确度和泛化能力要求是要高于静态物体的。

首先对于目前主要的几种目标识别算法来进行分析。目标识别算法的目标是在视频和图像中将所需要的目标信息进行提取和标定，主流的算法有帧差法、背景模型法和基于卷积的神经网络的目标识别算法，下面本论文将对此三种算法在本论文研究对象的基础上进行分析。

在帧差法[17]中，该算法主要是对不同的视频帧之间的图像进行对比和区分，通过设定出来对比阈值来进行背景和目标的区分，如果图像中一部分小于所设定的阈值，则其划分为背景，如果大于则划分为目标。这种算法虽然简单，但是对于较为复杂且目标众多的交通视频中的车辆容易出现误判的现象，无法保证精确性，故而排除此方法不进行采用。在背景模型法[18]中，此算法主要是将视频帧分为前后景，对于车辆的识别来说前景为运动车辆后景为其他环境背景，它还可以对后来帧的像素点进行分类进行自适应的不断更新背景模型，这对于一个静态简单的目标来说，达到了很好的识别效果，但是对于正在运动的汽车来说，无法处理因快速驾驶而引起的导致汽车在视频帧中发生变形的情况，不太适宜于复杂的道路交通情况。在基于卷积神经网络的目标识别算法中，其主要功能就是在目标初始化的技术基础上，通过特征提取和模型生成并且多次训练不断地更新目标信息来实时获取目标对于运动性很强同时也对于精确度要求很高的视频车辆这一研究对象来说，是十分适合的。因此，在接下来的文章中将主要讨论基于卷积神经网络的目标识别算法。

接着对于当下的主要几种目标跟踪算法来进行分析。目标跟踪任务，主要是在完成目标识别的基础上，基于有着明确的某视频序列初始帧的目标识别框的情况下，预测后续视频帧

中的这一目标识别框所存在的地方。结合上一章的内容，可以了解到主流跟踪算法有以稀疏表示为代表的基于生成式模型的算法，以相关滤波和深度学习为代表的基于判别式模型的算法。本论文接下来将对以上所述的三种方法进行分析比对。

稀疏表示方法[19]将输入信号用这组过完备字典线性表示，但是它处理遮挡的情况时，主要使用琐碎模板这一固定化的方式来解决，这将会对许多处于灰暗的情况下的遮挡存在不是很好的处理效果，这无法适应交通视频中车辆遮挡情况复杂的情况，故而将其排除。基于相关滤波的跟踪方法[20] 的基本操作是在基于使用一个确定的 Template 的情况下，然后对于下一帧图像被该 Template 做一个卷积操作，卷积之后所能够看到响应最大的区域就是所要寻找的预测的目标。但是所有相关滤波算法都存在边缘效应的问题，即当对图像进行滤波时，对于图片的边缘部分来说，卷积模板的一部分会位于图像边缘外面。这一效应将会大大降低目标跟踪的准确性，故而在此课题中不选用该算法。对于基于深度学习的目标跟踪算法来说，首先由于深度特征算法能够对每个目标都进行非常好的表达，虽然在该技术被引入的第一阶段，在目标跟踪技术领域的实际应用并不是非常合适，因为目标跟踪的任务仅仅包含初始帧的视频和图片数据，它就是能够被广泛地应用于进行目标深度学习这一技术的特殊性，因此目标跟踪缺乏了大量数据提供的神经网络自主学习。从近年的相关研究发展情况看，研究者们已开始考虑把在基于分类好的图像数据集如 COCO、VOC 等数据集所训练好的卷积神经网络中使用迁移式学习方法直接搬进目标追溯跟踪中去，基于深度学习的目标追溯方法才能够得到充分发展。SORT[21] 和 Deep-SORT[22] 是多目标跟踪中两个知名度比较高的算法。

所述算法速度和精度对比图如下图 2-1 所示：

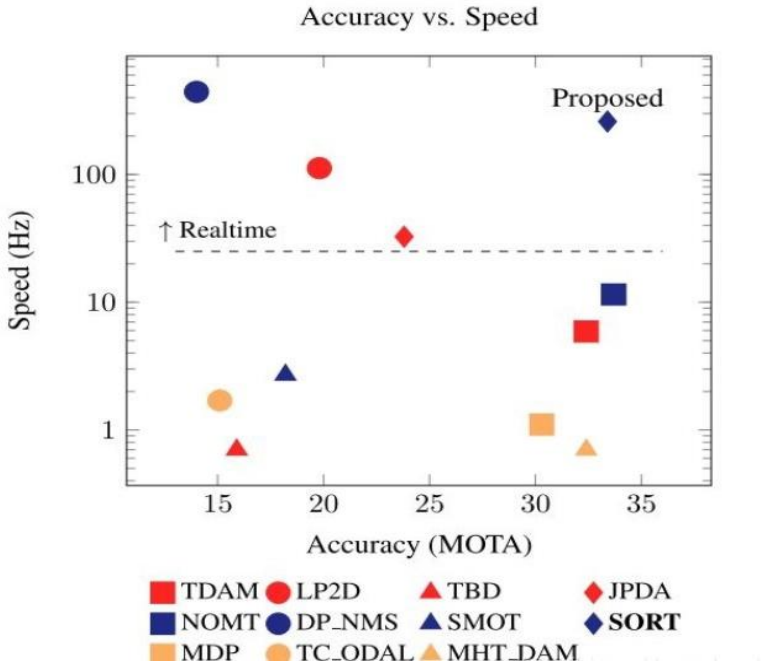

图 2-1 11 种主流跟踪算法速度精度对比图（2016）[21]

从上图 2-1 可以看出，SORT 算法无论是从速度还是精度上都完胜其他相关算法，Deep-SORT 算法属于是 SORT 算法的改进版本。因此在本篇论文中面向多目标跟踪将主要分析这两个算法。

2.2 基于深度学习的车辆识别算法分析

2.2.1 卷积神经网络介绍

卷积神经网络（Convolutional Neural Networks, CNN）是一类包含卷积计算且具有深度结构的前馈神经网络，是深度学习的代表算法之一[23]。近年来，基于深度学习的目标识别与识别方法应用在车辆的识别和分类中，效果比传统方法好很多。这一系列算法大致可以分为两大类：

(1) Two－Stage 目标识别算法，这类识别算法主要可以分为两个基本的阶段，第一个是该阶段首先会产生一个候选区域(proposal region)， 给出一个候选区域目标所在大概的位置情况，然后第二个阶段就需要对候选地点所在的大概位置情况进行具体分类（classifier）和在位置上的提前精修。较为典型的此类算法主要有 R－CNN、Fast R－CNN、Faster R－CNN 等；

(2) One－Stage 目标识别算法，这类识别算法不需要使用候选区域这一阶段，而是通过直接地产生所要检测的目标及其属于具体某一候选区域类别的概率和具体位置四角坐标的数量值，真正在传统意义上已经实现了对端到终点的目标识别，比较典型的识别算法主要有 YOLO（you only look once）、 SSD 等。初代的 YOLO 算法性能虽然不如 SSD 算法[24]，不过从下图 2-2 可以看出，YOLOv3 算法明显优于 SSD 算法，速度更快，准确性更高[25]。

Model	Train	Test	mAP	FLOPS	FPS	Cfg	Weights
SSD300	COCO trainval	test-dev	41.2	-	46		link
SSD500	COCO trainval	test-dev	46.5	-	19		link
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40	cfg	weights
Tiny YOLO	COCO trainval	-	-	7.07 Bn	200	cfg	weights
SSD321	COCO trainval	test-dev	45.4	-	16		link
DSSD321	COCO trainval	test-dev	46.1	-	12		link
R-FCN	COCO trainval	test-dev	51.9	-	12		link
SSD513	COCO trainval	test-dev	50.4	-	8		link
DSSD513	COCO trainval	test-dev	53.3	-	6		link
FPN FRCN	COCO trainval	test-dev	59.1	-	6		link
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14		link
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11		link
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5		link
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45	cfg	weights
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35	cfg	weights
YOLOv3-416	COCO trainval	test-dev	57.9	140.69 Bn	20	cfg	weights

图 2-2　YOLO v3 速度对比图

2.2.2 基于 R-CNN 算法的车辆识别

R-CNN 的全称是 Region-CNN ，它基于卷积神经网络(CNN) ，线性回归 ，和支持向量机 (SVM) 等算法，实现目标识别技术[26] 。对于一张简单的三维图片，R-CNN 将会生成大量的候选区域，接着对于每一个候选区域的特征进行特征向量的生成，最后我们可以直接得到一个带有特征值的矢量。然后再通过对于不同的特征类别给出一个不同的 SVM 分类器，判定所检测出来的目标属于该类别的可能性的大小。为了提升定位准确性，R-CNN 最后又训练了一个边界框回归模型，通过边框回归模型对框的准确位置进行修正[27] 。该过程如下图 2-3 所示：

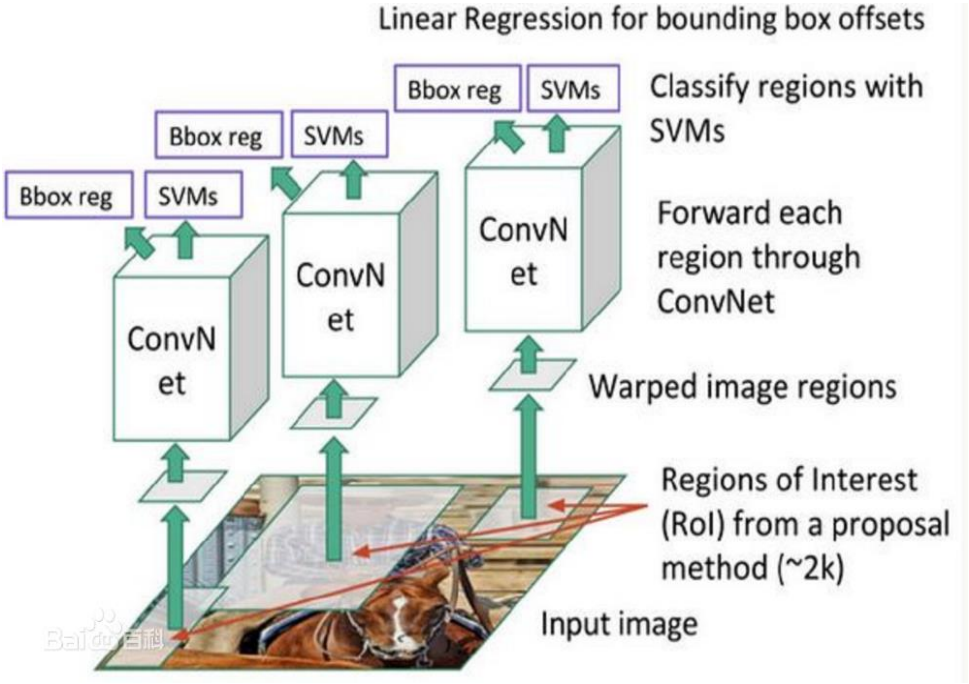

图 2-3 R-CNN 目标识别过程图[26]

2.2.3 基于 YOLO 算法的车辆识别

YOLO（You Only Look Once）[28] ，是一种一体化的实时目标识别与分类算法模型。它只通过一次卷积就能实现端到端的预测，而它的运算速度，是保证实时性的关键。

YOLO 一次性读入整张图片，经过一个 20 层的 GoogleNet ，然后经过四个带 ReLU 激活函数的卷积层，接着是两个全连接层，最后将维度 reshape 为 7*7*30。该过程如下图 2-4 所示。

输入图片被分割为 S=7 的 S*S 窗格单元（grid cell），每个窗格单元关联两个大小形状不同的边界框（Bounding Box），每个边界框有五个参数，分别是中心点的坐标 x，y ，边框的宽 w 和高 h ，以及置信度 (confidence) ，每个窗格单元有一组条件类别概率，其数量等于类别数 C 。当窗格单元中有物体时，窗格可被编码为一个

$$S*S(B*5+C) \tag{2-1}$$

S*S : 窗格单元的数目

B : 每个窗格单元负责的边界框的数量

C : 分类数

的张量。之所以 B 乘以 5 是因为每个边界框有 5 个参数，它包含了物体的位置信息。而公式中的 C 包含了物体的分类信息。

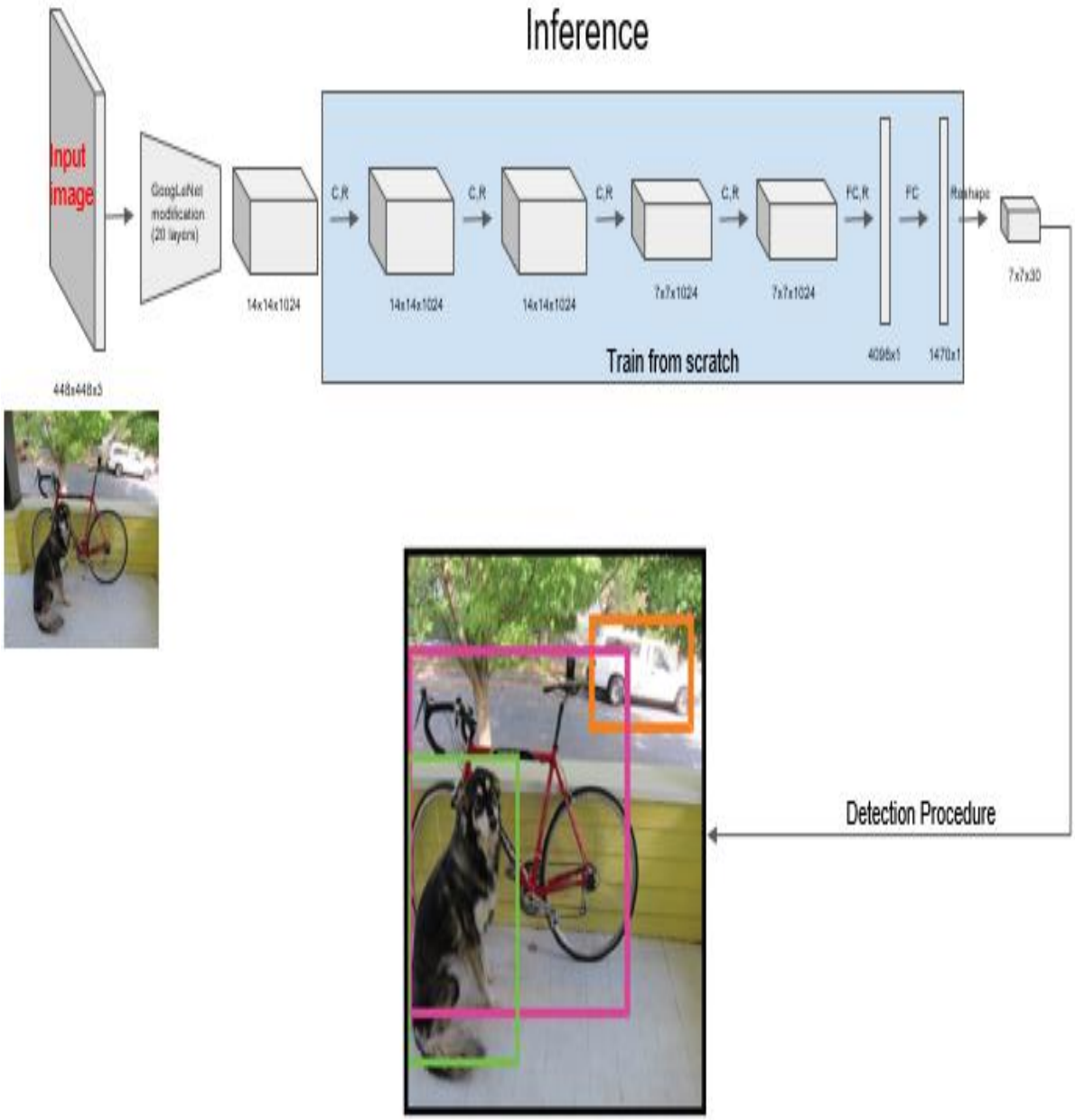

图 2-4 YOLO 模型结构[28]

在形式上，该算法将置信度 confidence 定义为

$$\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} \tag{2-2}$$

IOU 为真实结果与预测结果的交集与并集之比。

如果一个中心目标没有一个节点位置在这个中心网格之中，可以得到 Pr(Object)=0，也就是说该目标的置信度分数为零。否则，如果有节点存在中心网格之中，在网络训练的过程中，我们的目标应该是想让 Confidence 无限趋近于能够等于预测框与真实值之间联合部分的交集（IoU），即 Pr (Object) 应趋向 1，所以可以得到 Confidence=IoU。

YOLO 的 LOSS 函数定义如公式 2-3 所示：

loss function:

$$\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned} \tag{2-3}$$

综上所述，YOLO 基于 GoogleNet 架构，一次性读入整张图片，利用窗格单元与边界框一次性完成位置和分类特征的提取。

自 YOLO v3 提出之后，网络结构变得更深，它由原来的 19 层网络一下子跃迁到 53 层，其结构如图 2-5 所示。从图中我们可以看出，在网络架构中以一个步长分别为 2 的卷积层来代替了池化层。5 个这样的卷积层实现对输入分辨率缩小 32 倍的效果。当然，过深的网络深度必然会导致模型难以收敛。

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

图 2-5 YOLO v3 采用的网络结构

同时，2020 年 6 月 25 日，Ultralytics 发布了 YOLOV5 的第一个正式版本，被认为是现今最先进的对象识别技术，并在推理速度上是目前最强。其性能如下图 2-6 所示：

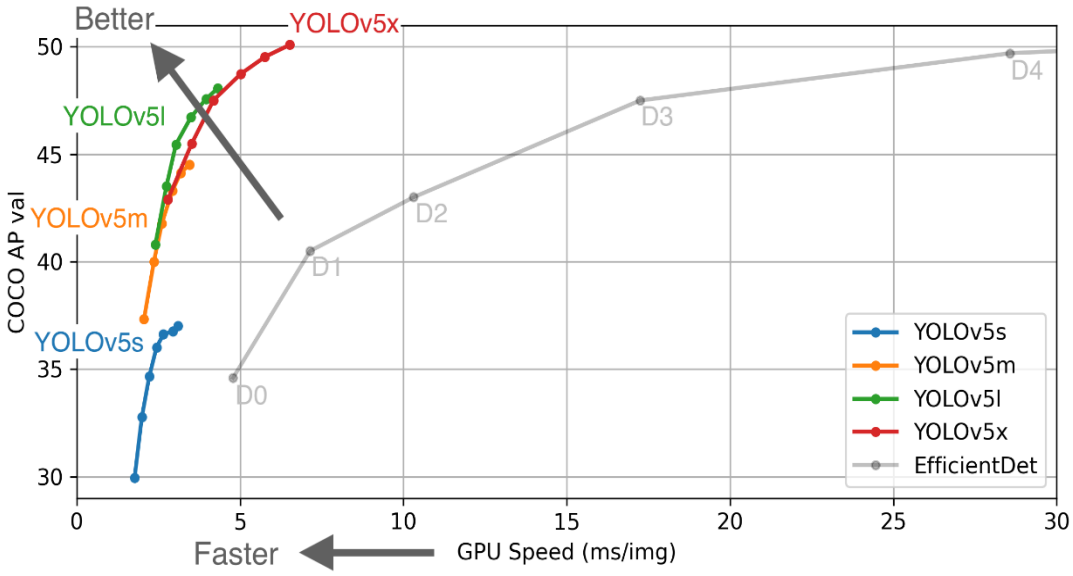

图 2-6 YOLO v5 性能图

YOLO v5 虽然在性能上提升不大，但是它的在灵活性与运行速度上却远强于 YOLO 前期的版本，在模型的快速部署上具有极强优势。

2.2.4 车辆识别算法比对分析

通过前文中对于两种主流卷积神经网络的车辆目标识别算法的介绍，可以看出，YOLO 识别算法大幅度地简化了传统的卷积神经网络识别算法，去掉了候选区域和提取的两个步骤，只使用一个网络即可完成目标识别，降低了计算量。这在实时性上来说，将会带来一个很大的优势，对于交通场景下，随时处于运动状态的车辆来说，实时性的强大将会带来很大的目标识别方面的优势。基于该应用要求，本论文选择基于 YOLO 的车辆识别算法来使用，作为下一步车辆跟踪的基础。

同时基于作者本人的试验开发环境条件，在 YOLO 系列算法中使用灵活性和速度领先且主要面向微端部署的 YOLO v5 无疑是本论文进行目标识别的首选算法。

2.3 车辆跟踪算法分析

2.3.1 目标跟踪介绍

目标跟踪技术首先需要输入第一帧的初始化目标框，在下一帧中，基于这些初始化好的目标框来产生众多候选框，然后使用具体的数据关联算法对这些候选框进行数据关联，最后在这些结果中找一个数据关联程度最高的候选框作为预测的目标，或者融合所得到的多个预测值，从而得到更为优良的预测目标。对于本论文中的交通视频车辆这一研究对象，基于其复杂的交通道路情况，只需要讨论多目标跟踪。多目标跟踪，顾名思义就是同时跟踪多个目标。在单目标跟踪中，通常来说是需要在一开始生成一个初始框，接着通过不同的预测算法比如卡尔曼滤波算法等，从而在随后的视频帧中针对该初始框内所有物体位置进行一个相应的预测，之后再通过相关的数据关联算法选取关联度最大的物体位置，从而达到跟踪的目的。而多目标跟踪算法，大部分都是不会在一开始生成初始框的，在多目标跟踪领域常见的跟踪策略为 TBD（Tracking-by-Detection）。本文中将主要介绍的 SORT[21] 和 Deep-SORT[22] 算法的多目标跟踪主要是由预测跟踪之后进行数据关联来实现。

2.3.2 SORT 算法

SORT[21] 全称为 Simple Online And Realtime Tracking， 对于现在的多目标追踪，更多需要依赖的就是其识别性能的优劣和好坏，也就是说通过改变一个识别器可以使效率提高 18.9% ， SORT 算法尽管只是把普通的算法如卡尔曼滤波（Kalman Filter）和匈牙利算法 (Hungarian algorithm) 结合到一起，却可以匹配 2016 年的 SOTA 算法，且速度可以达到 260Hz，比前者快了几乎 20 倍。该研究论文作者在实际中使用 SORT 算法对目标进行跟踪时没有发现任何所将要被跟踪的目标深度和外观上的基本特征，仅仅是使用了目标识别框的

位置和尺寸大小来对目标的运动估计和与数据之间的关联，也没有对其进行任何重新识别目标的算法，所以当目标跟丢后并且再次出现时，就不能够进行有效的跟踪，只能通过识别去重新更新 ID，这就不符合跟踪算法的常理了，需要改进，当然这篇文章主要就是追求速度，而不是过多的关注与识别错误的鲁棒性。实验中，作者使用了 CNN-based 的网络 Faster RCNN 和传统的行人识别 ACF 两个识别模型，另外，为了解决动作预测和数据关联，使用了两个十分高效的算法卡尔曼滤波和匈牙利算法。

对于预测模型，即用于表示和用于将目标的标签传播到下一帧。这里可以近似地把框架间位移看作都有一个线性匀速模型，从而可以独立于其他物体和相机运动。状态每个目标的模型如下：

$$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T \tag{2-4}$$

其中 u 和 v 分别代表目标的中心横纵坐标，s 和 r 表示目标的 BBox 的尺寸大小和比例，注意长宽比应该为一个常量。因此后面三个数据表示了所预测的下一帧，当目标识别结果和所预测出来的上一帧目标之间无关时，识别得到的边界框可以用来更新上一帧的目标状态，具体的过程中通过卡尔曼方法进行了一个优化和求解。如果未识别和确定目标之间的关联，则只要使用一个线性时间加速度模型。

对于视频帧的数据关联，作者采用匈牙利指派算法来对于视频帧中不同帧之间的数据的关联，使用 cost 矩阵表示其原有视频帧目标在当前视频帧中的预测位置和当前视频帧目标识别框之间的一个交并比，其中小于指定交并比阈值的指派结果是无效的。作者发现使用交并比能够解决目标的短时被遮挡问题，这是因为目标被遮挡时，识别到了遮挡物，没有识别到原有目标，假设把遮挡物和原有目标进行了关联。那么在遮挡结束后，因为在相近大小的目标 IoU 往往较大，因此很快就可以恢复正确的关联。这是建立在遮挡物面积大于目标的基础上的。如果连续 T_{lost} 视频帧没有实现与已追踪好的目标预测位置和识别框的交并比匹配，则认为目标消失。实验中设置 $T_{lost} = 1$的主要原因是在于匀速运动假设不合理，同时作者主要关注短时目标追踪。如果连续视频帧未能实现交并比与已经被追踪好的目标预测位置与识别框交并比相匹配，则可以认为该目标已经消失。在实验中设定的主要原因之一就在于匀速运动的假设并不合理，同时笔者主要集中精力对短时间内的目标跟踪。另外，尽早地删除已经遗弃或丢失的目标也将有助于改善追踪的效率。但是这样的情况下我们就可能导致视频跟踪系统中的 ID 会在节点上出现频繁的切换。该算法的研究人员在对目标进行跟踪时并未使用任何被识别的跟踪目标的物体外观和特征，而仅只是使用识别框的位置和尺寸来进行对目标的运动估计和与数据之间的关联，也没有对其进行任何重识别的算法，所以当目标跟丢时，就找不回来，只能通过识别去重新更新 ID ，这就不符合跟踪算法的常理了，需要改进，当然这篇文章主要就是追求速度，而不是过多的关注与识别错误的鲁棒性。

2.3.3 Deep-SORT 算法

在 Deep SORT[22] 中，作者使用其他的度量来代替关联，并使用卷积神经网络网络对大规模行人数据集进行了深度学习训练，并提取深度特征，从而增加卷积神经网络对遗失目标和障碍目标的处理能力。

该算法主要首先是使用一个八维度的空间函数来对于一个轨迹在特定时刻内的状态进行描述，$(u，v，r，h，x^*，y^*，r^*，h^*)$中心点的位置、纵横比、高度、以及图像上的坐标中所对应的运动速度等。然后对于我们的预测更新轨迹步骤而言，使用卡尔曼滤波器就是一个简单的实现方法。在这篇论文中，卡尔曼滤波器主要是采用匀速模型和非线性观测模型，观察变量$(u，v，r，h)$。然后对于目标的每一个轨迹进行处理，对于每一条轨迹都应该具有一个阈值 A ，它用来记录一条轨迹从上一次成功地与目标相匹配至当前某一个时刻的持续时间。但是当该阈值超过了提前假设的阈值 $A_{\max}$ 则被认为修正轨迹将会终止。然后在进行匹配时，对于未完成匹配的任何一种识别均被认为很有可能会产生一种新的轨道痕迹。将这种特殊情况下新生成的轨迹进行标注到特定状态' tentative ' ，然后通过观察在接下来的连续若干帧（如果在论文中应该是 3 帧）中判断它们是否连续地匹配成功，是的话则可以认为它们就是新的轨迹被重复产生，标注到是' confirmed ' ，否则就可以认为它们就是一个可以进行删除操作的轨迹，状态标注为' deleted '。

对于关联度匹配问题，需要将目标运动和表面特征信息相结合，然后融合这两个相似的测量指标。使用马氏距离来评测预测的 Kalman 状态和新来的状态：

$$d^{(1)}(i,j) = \left(d_j - y_i\right)^T S_i^{-1}\left(d_j - y_i\right) \tag{2-5}$$

表示第 j 个检测结果和第 i 条轨迹之间的非静止的匹配度。

其中S_i 是轨迹由 Kalman 滤波器在当前时刻预测得到的所进行观测的空间的协方差矩阵，y_i是轨迹在当前时刻的 prediction ，d_j 时第 j 个检测结果的状态$(u，v，r，h)$。在基于运动所具有的连续性这一特性的基础上，可以通过该马氏距离的计算公式对检测结果进行筛选，Deep-SORT 算法原论文中使用卡方分布的 0.95 分位点作为阈值 $t^{(1)} = 0.4877$，并且定义一个门限函数 2-6：

$$b_{i,j}^{(1)} = \prod\left[d^{(1)}(i,j) \le t^{(1)}\right] \tag{2-6}$$

在实际操作中，如相机运动时会造成马氏距离大量不能匹配，也就会使这个度量失效，所以马氏距离不是一个很好的关联度量。因此，该算法整合第二个度量标准，对每一个 BBox 识别框d_j我们计算一个表面特征描述子γ_j，$|\gamma_j| = 1$， 该算法创建一个 gallery 用来存放最新的$L_k = 100$ 个轨迹的描述子，即$R_k = \{\gamma k^{(i)}\}_{k=1}^{L_k}$，然后使用第 i 个轨迹和第 j 个轨迹的最

小余弦距离作为第二个衡量尺度。

$$d^{(2)}(i,j) = \min\left\{1 - \gamma_j^T \gamma_k^{(i)} \middle| \gamma_k \in R_1\right\} \tag{2-7}$$

当然，我们也可以用一个门限函数来表示

$$b_{i,j}^{(2)} = \prod\left[d^{(2)}(i,j) \le t^{(2)}\right] \tag{2-8}$$

接着，我们把这两个尺度相融合为：

$$c_{i,j} = \lambda\, d^{(1)}(i,\ j) + (1-\lambda)\, d^{(2)}(i,\ j) \tag{2-9}$$

$$b_{i,j} = \prod_{m=1}^{2} b_{i,j}^{(m)}$$

对于深度特征提取，该算法建立了一个深度卷积神经网络去提取目标的特征信息，并使用 L2 标准化把特征投影到一个统一的超球面。该模型结构如下表 2-1 所示：

表 2-1 Deep-SORT 深度模型结构表

Name	**Parch Size/Stride**	**Output Size**
Conv 1	3 x 3/1	32 x 128 x 64
Conv 2	3 x 3/1	32 x 128 x 64
Max Pool 3	3 x 3/2	32 x 64 x 32
Residual 4	3 x 3/1	32 x 64 x 32
Residual 5	3 x 3/1	32 x 64 x 32
Residual 6	3 x 3/2	64 x 32 x 16
Residual 7	3 x 3/1	64 x 32 x 16
Residual 8	3 x 3/2	128 x 16 x 8
Residual 9	3 x 3/1	128 x 16 x 8
Dense 10		128
Batch and l2 normalization		128

总的来说，效果还是很明显的，对于一个连续视频帧或者较短时间期内的视频帧的进行卡尔曼预测和匈牙利算法匹配来说，距离度量这一方法具有很好的结果，而对于丢失时间比较长的轨迹而言，使用深度特征表观信息进行匹配度度量也起到了比较好的作用。并且，使用 CNN 提取的特征进行匹配，大大减少了 SORT 中的标识符号的切换，经过研究人员实验

表明它们已经减少了大约 45%,在高速率的视频流中也已经达到了良好的水平。

2.3.4 车辆跟踪算法比对分析

通过前文中对于两种主流面向多目标跟踪的车辆目标跟踪算法的介绍，可以看出，Deep-SORT 跟踪算法相对于 SORT 跟踪算法而言，它主要通过建立一个 CNN 去提取目标的特征信息作为被跟踪目标的外观特征，当目标跟丢时可以找的回来，不像是 SORT 跟踪算法只能通过识别去重新更新 ID。这一操作上的区别大幅度地简化了 SORT 跟踪算法的 ID 切换，增加了算法的准确性和鲁棒性，同时避免了 ID 切换这一内存开销，也从很大程度上加快了跟踪算法的速度。这在准确度和实时性上来说，将会带来一个很大的优势，对于交通场景下，随时处于运动状态的车辆来说，正如前文对于目标跟踪的需求分析所述，实时性和准确度的强大将会带来很大的目标识别方面的优势。基于该应用要求，本论文选择基于 Deep-SORT 的车辆跟踪算法来使用从而完成论文目标。

2.4 本章小结

在本章中，通过对于实际情况交通环境、车辆运动状态，以及多种多样的车辆识别和典型的目标跟踪算法进行分析，可以看出如果想要在交通场景下的视频流中对车辆对象进行目标识别和跟踪，其对于精确度和实时性都有着一定的要求，所以需要选用一个准确度和实时性都较好的技术路线来进行实现。然后通过对多种基于卷积神经网络的识别算法和两种主流的多目标跟踪算法进行对比分析后，确定了基于 YOLO v5 和 Deep-SORT 算法的车辆识别和跟踪技术框架，并且也给出了选择这两种算法的原因和优势所在。

第三章 环境搭建与系统设计

基于第二章多种算法对比分析得出的结果而选定的技术路线，本章将旨在于对本论文课题所需要的实验环境和基本的搭建过程进行简要介绍，同时将基于搭建好的环境对 YOLO v5 目标识别器系统和 Deep-SORT 目标跟踪器给出具体的设计思路，为后续的基于视频的车辆识别与跟踪系统的实现做好准备。

3.1 实验环境与搭建

3.1.1 实验环境

CPU: Intel(R) Core(TM) i7-8750H @2.20GHz 2.21GHz

GPU: NVIDIA GeForce GTX 1050 Ti 4G

RAM: 8.00GB(7.88GB 可用)

OS: Win10 x64

Anaconda 3(64 Bit)

Python 3.6.6(64 Bir)

Pytorch + YOLO v5 + Deep-SORT

3.1.2 搭建过程简要介绍

(1) 安装 Pytorch

首先使用打开命令行输入 conda create -n pytorch python=3.6 创建名叫 pytorch，Python 版本为 3.6 的新环境，同时使用 activate pytorch 激活名叫 pytorch 的环境。接着去 pytorch 官网下载 pytorch，鉴于即将使用的框架为 YOLO v5 4.0，可以搭配 pytorch 最新版本 1.8.1。在官网上选择合适的环境，即可获得相关的下载指令。

之后从原先的 conda 指令中将-c 之后的部分去掉，conda 中的-c 就相当于 pip 中的-i，都代表指定下载源，所以我们要去掉，这样才是使用我们上面添加的国内镜像源。等待到下载成功之后，启动 Python，导入 pytorch 库，并且使用如下代码查看 pytorch 的版本和 gpu 版本是否可用。

下图 3-1 是实际安装过程中官网下载指令截图：

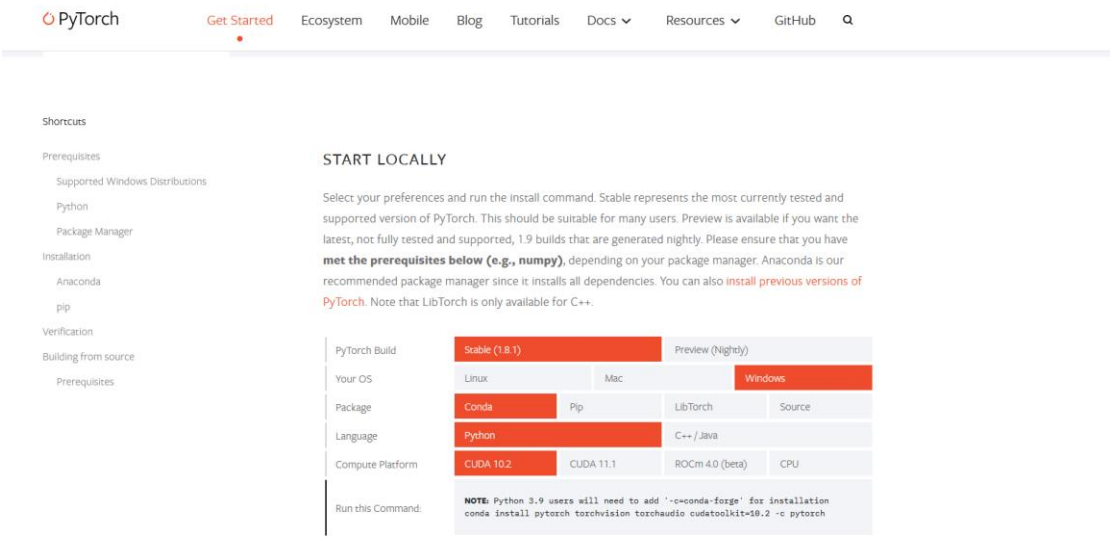

图 3-1 pytorch 官网安装图

#查看版本

print(torch.__version__)

#查看 gpu 是否可用

torch.cuda.is_available()

#返回设备 gpu 个数

torch.cuda.device_count()

(2) 安装 YOLO v5

首先来到 github 下载代码库 https://github.com/ultralytics/yolov5。由于前期安装过 git，故而可以直接输下面的代码 git clone https://github.com/ultralytics/yolov5 进行克隆。接着通过命令行来到 yolov5 文件夹下输入 pip install -qr requirements.txt 就可以把所需要的依赖库都安装好。接着下载官网上的预训练权重，下载好之后通过 cd 命令来到 yolov5/ 文件夹下使用 python detect.py 代码进行运行测试。这样就是通过调用自己电脑的摄像头，用 yolov5s.pt 的模型来进行对象识别。

(3) 安装 Deep-SORT

直接输下面的代码 git https://github.com/ZQPei/deep_sort_pytorch 进行克隆。通过下载预训练 cpkt.pt 权重文件，然后基于前期下载好的 YOLO 预训练权重进行识别该 Deep-SORT 算法是否可用。

3.2 基于 YOLO v5 的目标识别器系统设计

在基于前文完成基本环境的搭建之后，首先要做的是对训练基于 YOLO v5 的目标识别器系统的数据集的选择安排、采集和处理。这里将在第四章中的 4.1 小节进行详细叙述。接

着在基于处理好的数据集的基础上，需要通过该数据集对于 YOLO v5 模型进行训练，具体训练过程将在第四章中的 4.2 小节进行详细叙述。这里将主要在基于具体功能模块设计思路进行阐述，以便为后续实现和效果分析内容进行准备。

基于前文的叙述，可以得知道，Deep-SORT 模型需要以目标的质心和宽高来作为识别的基础，所以首先需要对能够实现计算边界值功能的函数进行设计。其中，bbox_left 通过对于识别结果 xyxy 数组的第一项和第三项（代表横向坐标的边界值）进行取最小值，bbox_top 通过对于识别结果 xyxy 数组的第二项和第四项（代表纵向坐标的边界值）进行取最小值，从而分别获取 xyxy 的左下角点的横向和纵向坐标。bbox_w 和 bbox_h 分别计算宽和高，x_c = (bbox_left + bbox_w / 2)和 y_c = (bbox_top + bbox_h / 2)分别计算中心点的横纵坐标。从而达成该函数功能的设计。

接着完成一些前期的生成目标识别框功能的设计，对于不同的标签利用调色板来获取不同的颜色，设计函数 compute_color_for_labels(label)，以 color = [int((p * (label ** 2 - label + 1)) % 255) for p in palette]来获取。然后对于识别出来的目标边框进行绘制，首先利用 annotations 里已经标定好的四角坐标，将其读入。接着生成标签和颜色，利用 cv2.rectangle 来生成矩形边框。

然后，开始正式设计目标识别功能模块。首先，需要对识别设备、要载入的模型、要读入的数据进行初始化，为了给予该基于 YOLO v5 的目标识别器系统更多的使用方式，在这里调用 parser 的 argument 来进行权重和读入数据等初始化参数的传入。在完成基本的初始化之后，使用 torch.load(weights， map_location=device) 和 pred = model(img，augment=opt.augment)[0]对训练好的模型权重进行载入并且生成预测值 pred，获取标签名称和颜色。之后开始正式运行，先对于 pred 进行非极大值抑制，遍历数据每一帧图像，直到 det 即识别目标不再存在的情况下，重新定义边框，输出结果，利用权重文件中对类型的标识符 c 来获取每一个识别识别目标属于的类型，将其类型打到标签上。

最后，为后续基于 Deep-SORT 的目标跟踪器系统设计把需要的 deepsort 函数进行预留，同时对于保存和训练显示过程功能进行设计，利用 cv2.imshow() 这一 API 调用摄像头显示过程，利用 cv2.VideoWriter 函数完成视频数据存储这一功能的实现。

到此，基于 YOLO 的目标识别器系统设计完成，并且为后续的基于 Deep-SORT 的目标跟踪器系统设计预留了跟踪的函数。在进行目标识别测试时，可采用将预留跟踪 Deepsort 函数和参数进行注释的方式下进行测试即可。

3.3 基于 Deep-SORT 的目标跟踪器系统设计

在本 3.3 小节，将主要基于前 3.2 小节中所介绍的基于 YOLO 的目标识别器系统设计的基础上，完成基于 Deep-SORT 的目标跟踪器系统的设计。

基于本章所主要叙述内容，对于数据集的选取，采集和处理在这里先将按前文一样按下

不表，意在与先完成功能设计，如需要测试可采用 Deep-SORT 原论文中的 cpkt.t7 预训练权重文件进行测试。

对于 Deep-SORT 目标跟踪系统设计，具体流程内容如下图 3-2 所示：

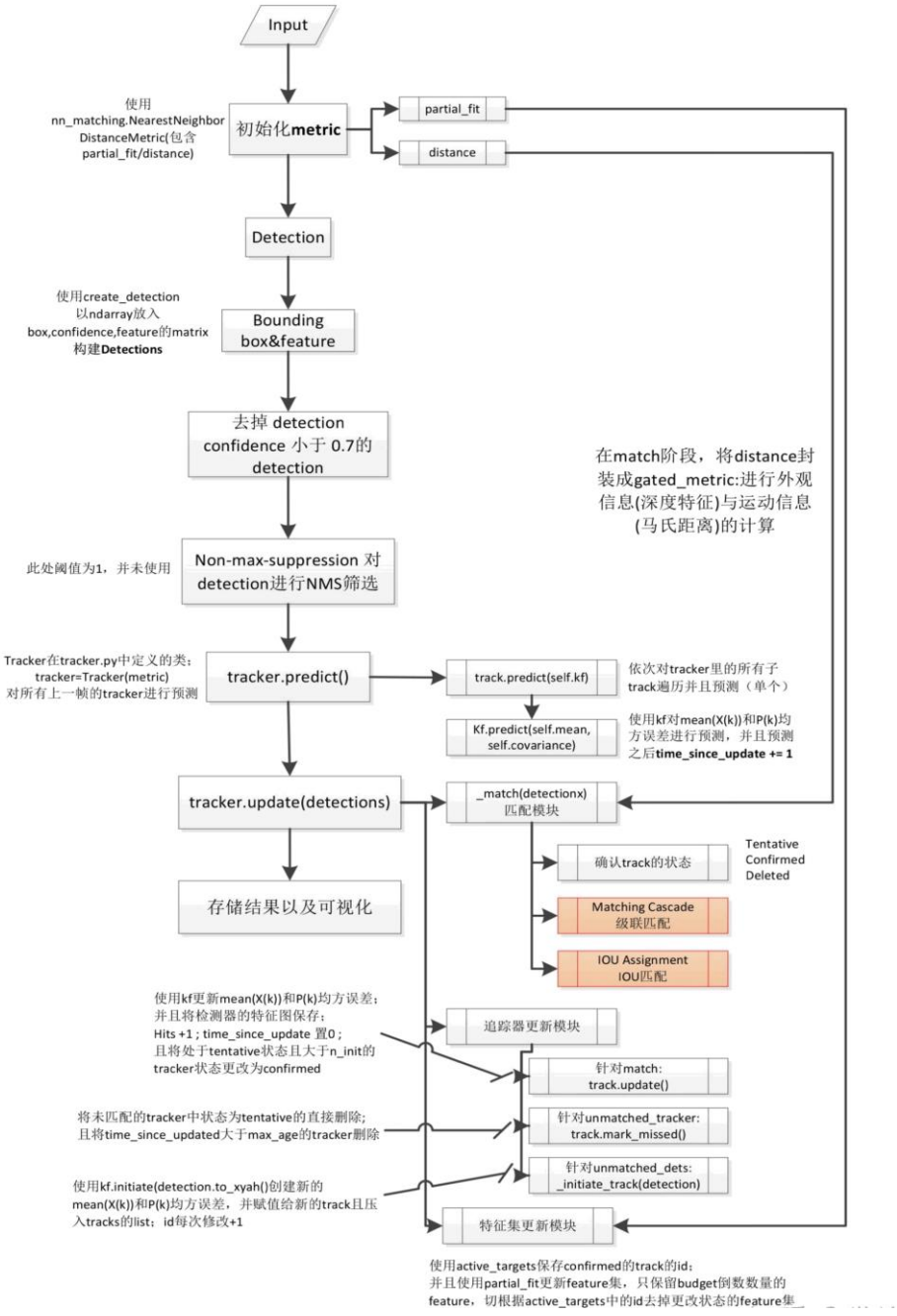

图 3-2 Deep-SORT 目标跟踪器设计具体流程图

Deep-SORT 目标跟踪器系统主要根据按视频帧的顺序来进行处理，其对于每一个视频帧的具体处理流程详细描述如下：

首先，读取当前视频帧目标 ID 所检测出来框的位置以及每个识别框中图像目标的深度

特征（此处，我们必须在处理实际使用情况时自行提取）。接着通过置信度过滤目标框架时，即删除目标框架中低置信度者的特征。接着对识别帧进行非最大抑制以消除多个目标检测框在单个目标上的位置。

然后将进行预测步骤，使用 kalman 滤波器预测目标在当前视频帧中具体的位置信息，执行卡尔曼滤波公式 3-1 和 3-2：

$$x(k) = Ax(k-1) \tag{3-1}$$

$$p(k) = Qp(k-1) \tag{3-2}$$

其中，$x(k-1)$为上一帧的目标的状态信息（即卡尔曼滤波模块代码中的 mean），为上一帧中的目标的信息[center x， center y， aspect ration， height，0，0，0，0]; $p(k-1)$为目标的估计误差（代码中的 covariance），A 为状态转移矩阵，Q 为系统误差。等到完成预测之后，每一个追踪器都要被设置 self.time_since_update += 1。

在完成了预测之后，需要很好地保证了追踪的准确有效性，所以我们就需要为此设计一个可以实现更新的功能，用于对 kalman 跟踪器的参数和特征集这一要求进行实时的调整。另外，还需要对于消失的老目标和刚发生的新跟踪目标之间相互匹配度进行判断，首先我们需要通过实现将所检测得到的结果跟跟踪预测得到的结果相互匹配，从而识别出已经确认的状态跟踪器和不是已经确认的状态跟踪器。

对于已经经过确认的级联状态匹配跟踪器，进行了二次级联状态匹配。对于在相同保存时间内识别消失在多个帧的跟踪器，计算当前视频帧新保存识别的每一个跟踪目标正弦深度距离特征和各个在跟踪器现场中已经重新保存的每个跟踪器目标深度距离特征集之间的一个余下正弦深度距离矩阵。假如当前帧有 11 个识别目标，已有 10 个追踪器，每个追踪器已保留前面 6 帧目标的深度特征，则计算得到的 cost_matrix 大小为 10*11，计算过程为首先对每一个追踪器的 6 个特征，计算与当前帧 11 个新识别目标特征之间的（1 - 余弦距离），得到 6*11 矩阵，对每个识别块求最小余弦距离，得到 1*11 矩阵，存入 cost_matrix 对应行，表示对当前追踪器而言与当前各识别块之间最小的余弦距离。具体代码如下：

```
cost_matrix = self.metric.distance(features， targets)
    #distance 函数
    cost_matrix = np.zeros((len(targets)， len(features)))
        for i， target in enumerate(targets):
            cost_matrix[i， :] = self._metric(self.samples[target]， features)
    #_metric 函数
            distances = _cosine_distance(x， y)
            return distances.min(axis=0)
```

return cost_matrix

接着，在计算特征的 cost_matrix 的基础上，计算 kalman 滤波预测位置与识别框之间的马氏距离，具体过程为，先将各识别框由[x，y，w，h]转化为[center x，center y， aspect ration，height]，对每一个追踪器，也就是 cost_matrix 中的每一行，计算预测结果与识别结果之间的马氏距离。然后将 cost_matrix 中大于 max_distance 的元素置为 cost_matrix > max_distance，接着使用匈牙利算法输入 cost_matrix 从而进行指派。等到指派完成后，对未匹配的识别、未匹配的追踪、匹配对，即 cost_matrix 中阈值大于 max_distance 的匹配对也送入未匹配识别和未匹配追踪中去，进行分类。

而对于在没有能够成功地进行级联匹配上的跟踪器和不能够对其状态信息进行确认的跟踪与未能够实现级联匹配成功的识别之间基于交并比而进行的匹配，具体在这个过程中主要是依靠对于 cost _ matrix 这一矩阵来进行计算，矩阵的第 i 行代码就是表示第 i 个跟踪器和其所相应的识别结果之间的不交并比例(1- IoU)。对于 cost _ matrix 中大于 max _ distance 的元素设定为 max _ distance ，然后我们使用匈牙利算法以 cost _ matrix 矩阵的值作为输入方式进行了指派，在这种指派工作完成后依然可以统计出匹配成功者，未能匹配成功的跟踪器，无法匹配成功的轨迹。之后对于成功匹配上的识别框，需要进行一下参数更新，该参数更新的过程的具体公式如下公式 3-3，3-4，3-5 所示：

$$Q_t = E[\omega_t \omega_t^T] > 0 \qquad \mathrm{R_t} = \mathrm{E}[\mathrm{v_t v_t^T}] > 0 \tag{3-3}$$

$$\hat{x}_t^- = A_{t-1}\hat{x}_{t-1} \tag{3-4}$$

$$\hat{X}_t = \hat{X}_t^- + k_t\left(Y_t - C_t\hat{X}_t^-\right) \tag{3-5}$$

在完成了参数的更新之后，把这个特征值插入到跟踪器的特征集之中，把其相对应的参数重新进行了初始化。使用代码 self.time_since_update = 0 将时间重置为 0，当追踪器状态不为待定状态（Tentative）时，确认追踪器。

未匹配的追踪器表示虽然预测到了新的位置，但是下一帧识别框匹配不上的识别框。对于未匹配的追踪器有一下几种情况：对于仍然在待定状态之下的追踪器，将会直接被进行删除操作；对于已经是确认（confirm）状态的追踪器，虽然连续的多个视频帧中都预测了目标，但是其中间过程中任何一个视频帧都不能够实现与结果的数据关联，此时直接设置追踪器为待删除状态。

而对于未匹配的识别，则把它进行了初始化，这样就成为了一个全局跟踪器。因为对于没有匹配上的目标检测结果来说，就是表明出现了一个全新的卡尔曼滤波器需要对目标进行跟踪检测的目标结果，此时可以创建一个全新的卡尔曼滤波器对象，根据其初始识别的位置对新的卡尔曼滤波器中 mean 和 covariance，从而使得我们可以重新初始化一个全新的跟踪器。之后即可删除待删除状态的追踪器，并且更新留下来的追踪器的特征集。

最后完成对于 DeepSort 类自身构造函数的设计，以模型路径，最大马氏距离，最小置信度，最大 IOU 距离，是否使用 cuda 作为成员，从而在目标识别实现之后实现对该函数的调用。

3.4 本章小结

本章承接第二章中对车辆识别和车辆跟踪算法的分析，基于 YOLO v5 + Deep-SORT 技术框架，首先给出了具体实验期间所需要的基本软件硬件环境，给出了安装环境步骤，并且进行了基于视频的车辆识别和跟踪系统的设计思路的阐述。对于基于 YOLO v5 的目标识别系统，主要采用源码中的识别部分进行改编，基于基本的读入数据和权重文件，运行识别获取 ID 从而生成标签和识别标识边框以及显示及输出功能的设计来进行阐述；对于基于 Deep-SORT 的目标跟踪器系统设计，主要基于 SORT 论文源码中的流程进行分析，在添加深度特征的基础上，基于前期编写的 DeepSort 类来书写具体的构造函数，以及调用获取深度特征和调用更新跟踪器的接口函数。

第四章 系统实现与结果分析

本章将会主要旨在于基于第三章基于视频的车辆识别和跟踪系统设计思路的基础上，从数据集的采集和处理，到基于 YOLO v5 识别器和基于 Deep-SORT 跟踪器的实现，给出该系统整体的具体实现过程。同时，也将会对识别和跟踪的结果情况进行阐述，同时对于实现的效果利用主流的评价效果参数进行分析和性能评价。基于此前的内容及本章内容，彻底阐明一个实用的基于视频的车辆识别和跟踪系统的内容，并且验证其实用性。

4.1 数据集采集与处理

本课题所采用的数据集主要包括两个部分，第一部分为采集自西安市南二环路中段的交通情况数据集，主要将用于进行 YOLO v5 神经网络的训练和测试；第二部分为来自 CSDN 论坛[29]的公开数据集，主要将用于对 Deep-SORT 的深度外观特征。

首先阐述第一部分数据集，该数据集采集地点来自西安市南二环中段天桥上，使用专业摄像机进行捕捉，形成连续的图片数据集。该数据集 FPS（Frame Per Second）大概为 6-8fps，每张图片为 2288 x 1712 像素，1.08MB 大小，具体情况如下图 4-1 所示。

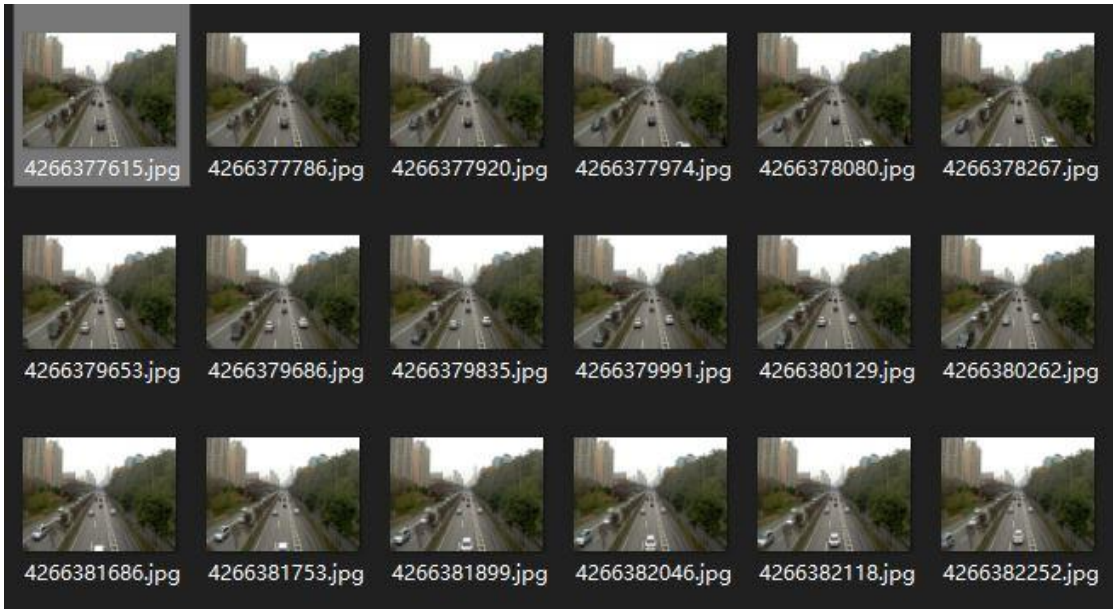

图 4-1 第一部分数据集图片情况图

将采集到的车辆数据集，使用 labelimg 进行处理，对车辆划分为 car， taxi， minivan，truck， motorcycle 五个类，以方便更好地对车辆目标进行识别，之后完成标注，得到对应的 XML 文档，即 Pascal VOC 标准格式的 annotations。该格式中第一列为标签名，第二列为中心点 x 的相对坐标，第三列为中心点 y 的相对坐标，第四列为宽的相对坐标，第五列为

高的相对坐标，在这里对应第三章中 YOLO v5 目标识别器中对 *xyxy 的读入。该内容如下图 4-2 所示。

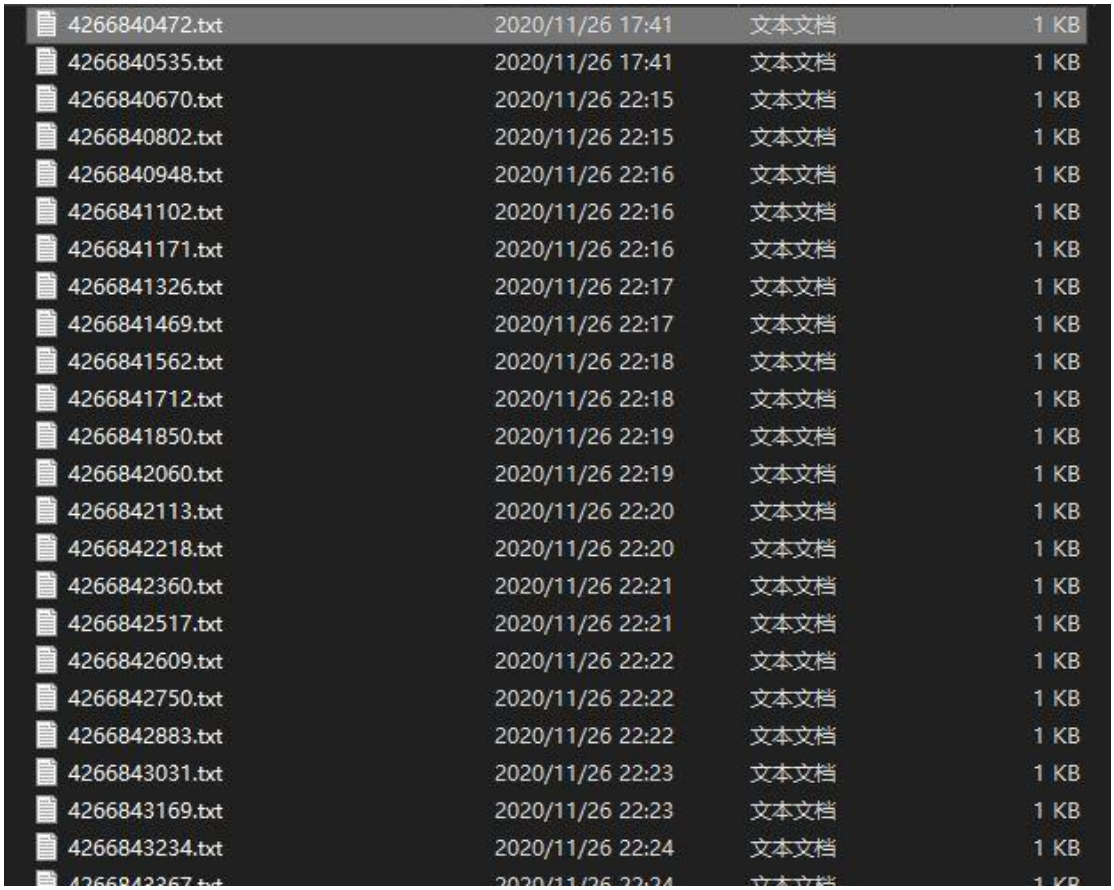

图 4-2 第一部分数据集标注情况图

对于该部分数据集，根据训练、识别需求对它进行划分，分为训练集、验证集、测试集，其中训练验证集占总数据集的 80%，测试集占总数据集的 20%，训练验证集中的 80%为训练集，20%为验证集。从而得到可以训练的数据集以后，使用该数据集对 YOLO v5 目标识别器进行训练。

接着阐述第二部分数据集，众所周知，Deep-SORT 的深度外观模型是在人体的重识别数据集上训练得到的，在用于人的多目标跟踪效果好，用于车辆就不一定适用，所以要训练适用于车辆的深度外观模型。该部分数据集主要用于训练车辆的深度外观模型，以便 SORT 算法更好地完成跟踪功能。在此使用 CSDN 论坛上的开源数据集进行训练，该数据集主要基于监控视频对 769 辆不同的车辆，进行不同角度（大于等于 12 个角度）的图像采集，固定图片为 128 x 256 宽高，图片命名格式为，修改图片文件名：[0:4]与图片上一级目录同，[4:6]相机 ID，[6:11]跟踪 ID，[11-15]图片序号。从而保证完成对于车辆的深度特征完整的提取。具体情况如下图 4-3 所示。

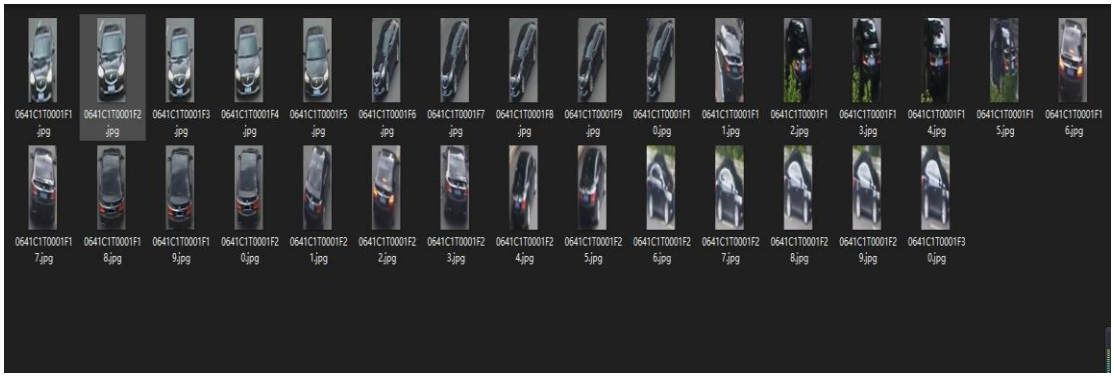

图 4-3 第二部分数据集情况图

之后即可基于该部分数据集对 Deep-SORT 进行训练，从而达成进行跟踪车辆的功能目的。

4.2 基于 YOLO v5 识别器实现与效果分析

4.2.1 实现过程

在前文 3.2 中基于 YOLO v5 识别器的设计和 4.1 中对用于 YOLO v5 目标识别器的数据集的基础上，需要完成数据的 yaml 文件的配置，从 yolov5/data/下复制一份 yaml 文件到数据集 cars 文件下，打开之后修改对应的图片路径，以及 nc（标签个数）、name（标签名）。

同时，基于 YOLO 原本使用的 coco 数据集的普遍性和庞大性，并且包含一定的车辆数据，本课题在此考虑可以使用迁移学习法进行车辆识别器的训练。迁移学习方法能够对深度和特征进行迁移，能够很好地改善和提升模型的广泛性能，对于庞大的数据集来说也是如此，并且随着一个参数被固定的一个层数 n 的变化而增长，对于两个相似度小的任务来说，它们之间转移距离的增加速度比两个相似度大的两个任务之间转移距离的增加速度要快，两组数据越相似，深度特征迁移训练的效果越差。

使用迁移学习法的实现过程中接着需要对想要进行迁移训练的模型文件修改 nc 个数。在此，基于图 2-6 中各个 YOLO v5 预训练模型性能对比图来看，精确度和速度都表现不差且部署较为轻量级的 yolov5m.pt 文件实为本课题的首选权重文件。

在 yolov5 源码中给出了 train.py 源文件中，主要相关的 augment 有如下几种，其所代表的含义在此一一给出：

--img 统一输入图像规模

--batch 每次网络训练输入图像的数量，最低性能为 1

--epochs 训练次数

--data 数据 yaml 文件的相对位置

--cfg 模型 yaml 文件的位置

--weights 预训练模型位置

--device 训练的设备(CPU or GPU)

在本课题的 GPU 为 NVIDIA GeForce GTX 1050 Ti 4G 的客观条件下，对原数据集进行一定的压缩处理，选用—img 640，--batch 4，作为输入图像规模。经过 50 次训练，历时 7.25 小时完成训练，其效果分析内容将在 4.2.2 中给出。

在完成基于 YOLO v5 的目标识别器的训练之后，以测试数据集的单帧图片为原图片，使用 PowerPoint 进行视频合成，从而合成视频进行目标识别测试。具体效果将在 4.2.2 中给出。

4.2.2 效果分析

在训练的过程中，对于整体过程可以通过 wandb 库进行可视化操作。训练过程中使用 GPU 进行训练，训练时 CPU 和 GPU 利用率情况如图 4-4 和图 4-5 所示。下面可以从图 4-4YOLO 训练过程中 CPU 情况图和图 4-5 YOLO 训练过程中 GPU 情况图中看出，CPU 利用率达到 44%，总体情况稳定，GPU 利用率达到 16%，内存占用 2.9/4.0GB，总体情况稳定。

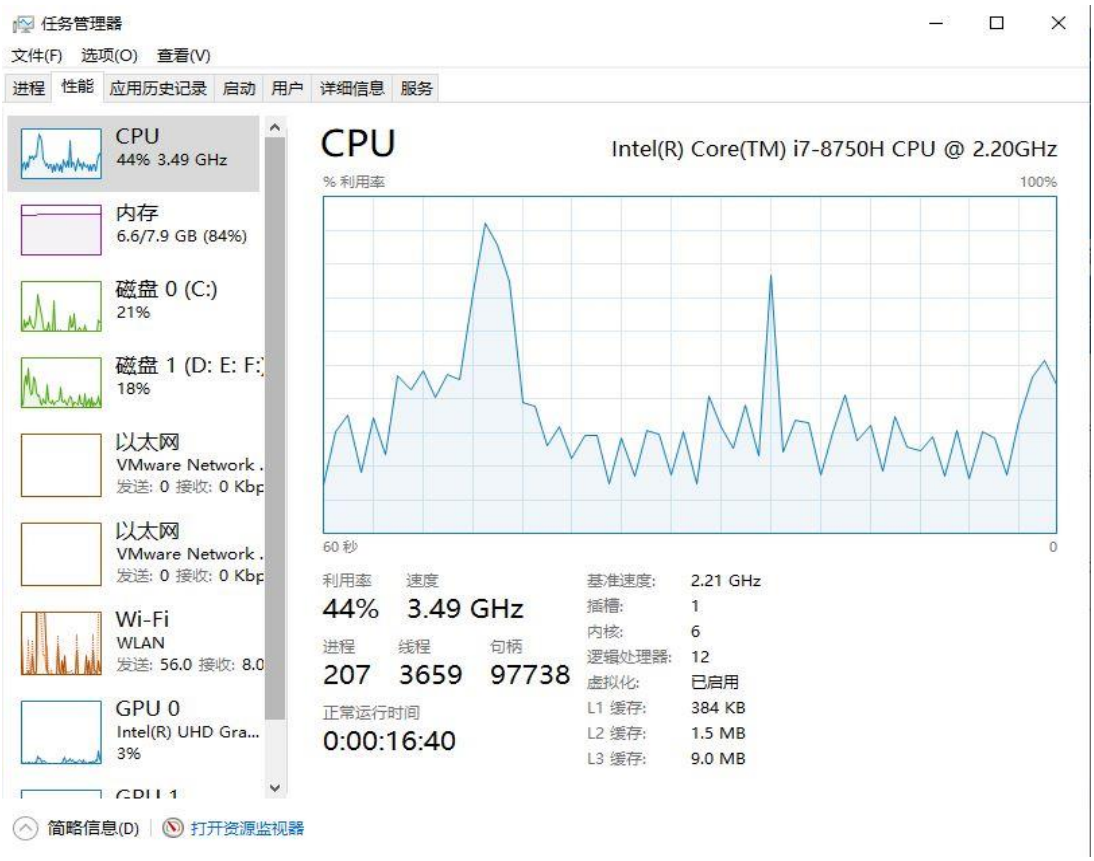

图 4-4 YOLO 训练过程中 CPU 情况

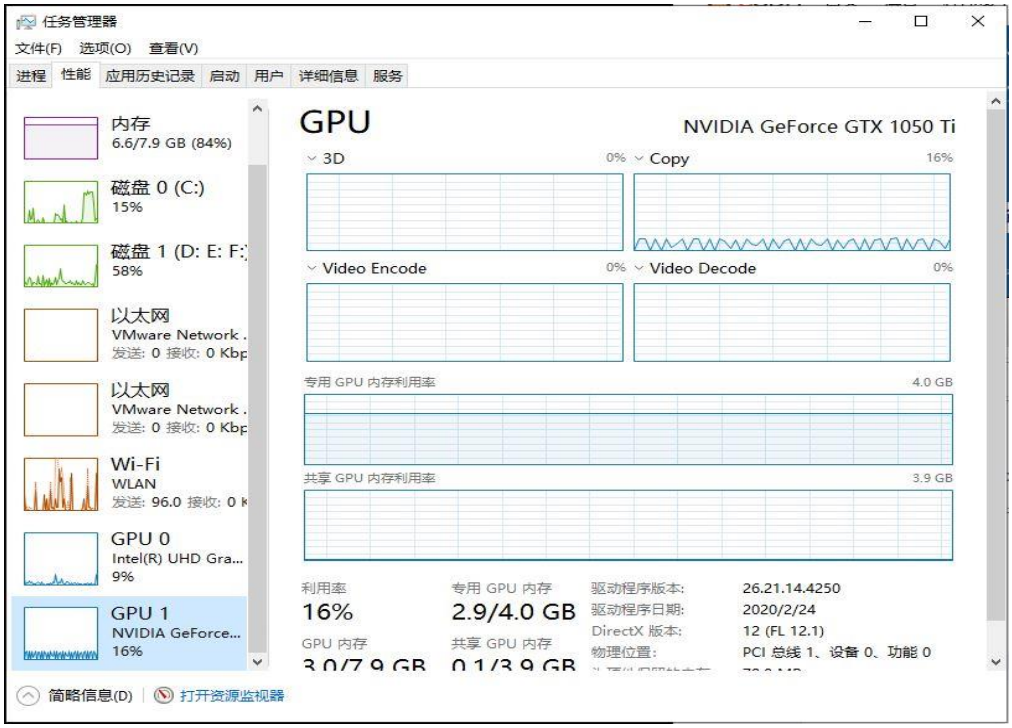

图 4-5 YOLO 训练过程中 GPU 情况

对于 YOLO 网络结构来说，主要的评价指标有如下几种：

（1） Precision

$$Precision = \frac{TP}{TP+FP} \tag{4-1}$$

精确率 Precision 是用来表示分类器对某一类别预测结果的准确率的一项指标，对所有类别的预测准确率求和取均值后可以得到整体得到精确率。其中 TP 是指 True Positive 区域，即与实际情况（Ground truth）区域的交并比>=0.5 的区域，FP 是指 False Positive 区域，即 IoU < 0.5 的区域。

（2） Recall

$$Recall = \frac{TP}{TP+FN} \tag{4-2}$$

召回率 Recall 是用于衡量正例标签中正例成功进行预测的比例的指标。其中 TP 同上，FN 是指 False Negative 区域，即遗漏的 Ground truth 区域。

（3） mAP (mean Average Precision)

平均精确率 AP 是指在不同的召回速率 Recall 情况下，在平均 PR 曲线下的面积，是 Precision 在 Recall 之下的积分。mAP 是一种泛指对不同的类别之间进行衡量的一个平均精度，即所有类别的一般平均精度要求及其除以全部的类别，基本上每一个数据集中各类别的数量通常都是平均或相等分布的，mAP 也就代表了每一个数据集中各类的一般平均精度的

一个平均值。

对于本课题所在的多目标识别领域，主要以 mAP 作为评价指标。

在训练过程中借助可视化工具可以看到训练效果如下图 4-6 所示（以第二次训练为例）。

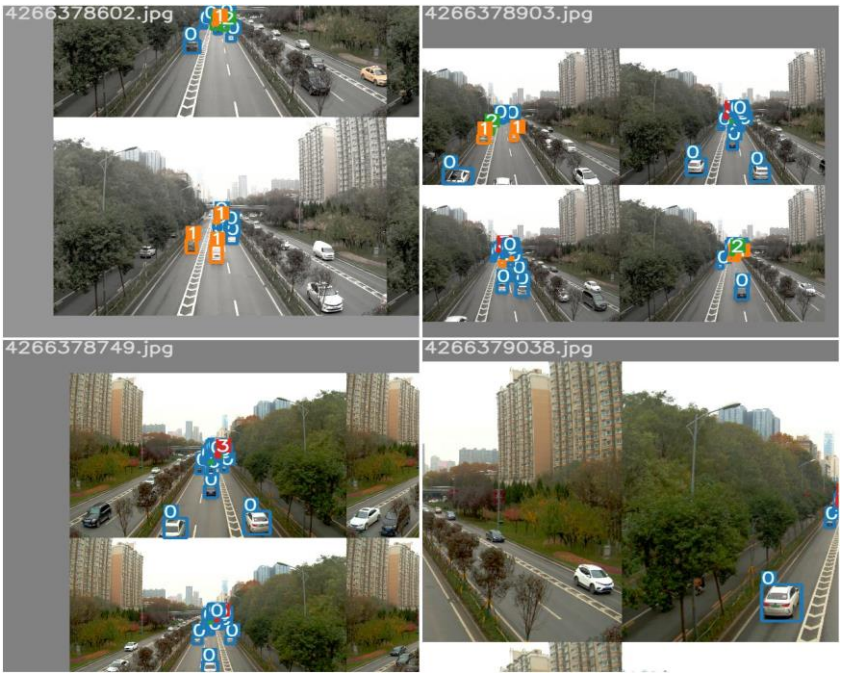

图 4-6 第二次训练效果图

在完成训练之后，需要使用验证集对模型进行验证，在验证过程中借助可视化工具可以看到验证效果如下图 4-7 所示（以第二次验证为例），肉眼可见，置信度较高。

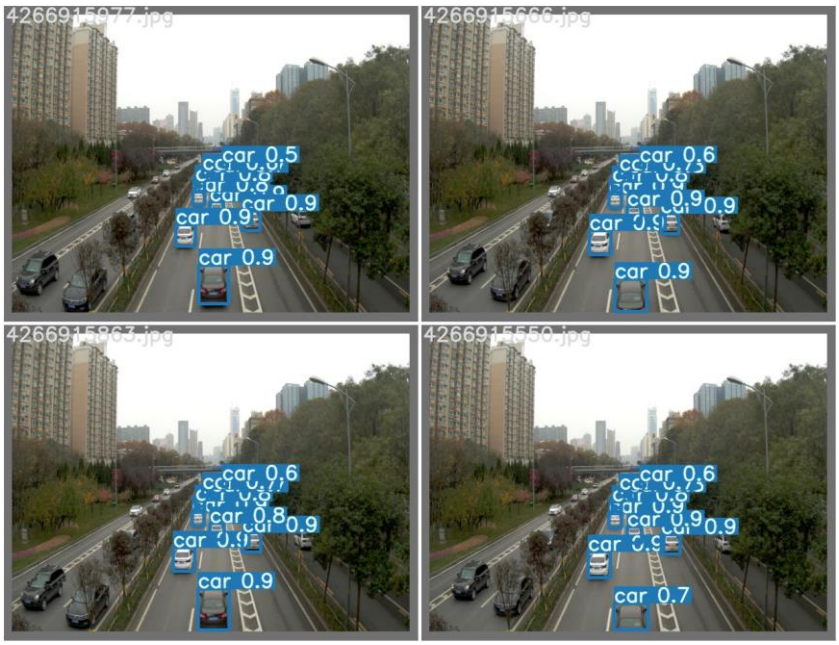

图 4-7 第二次验证效果图

在完成训练之后，我们可以得到 PRC（Precision-Recall Curve）图如下图 4-8 所示。

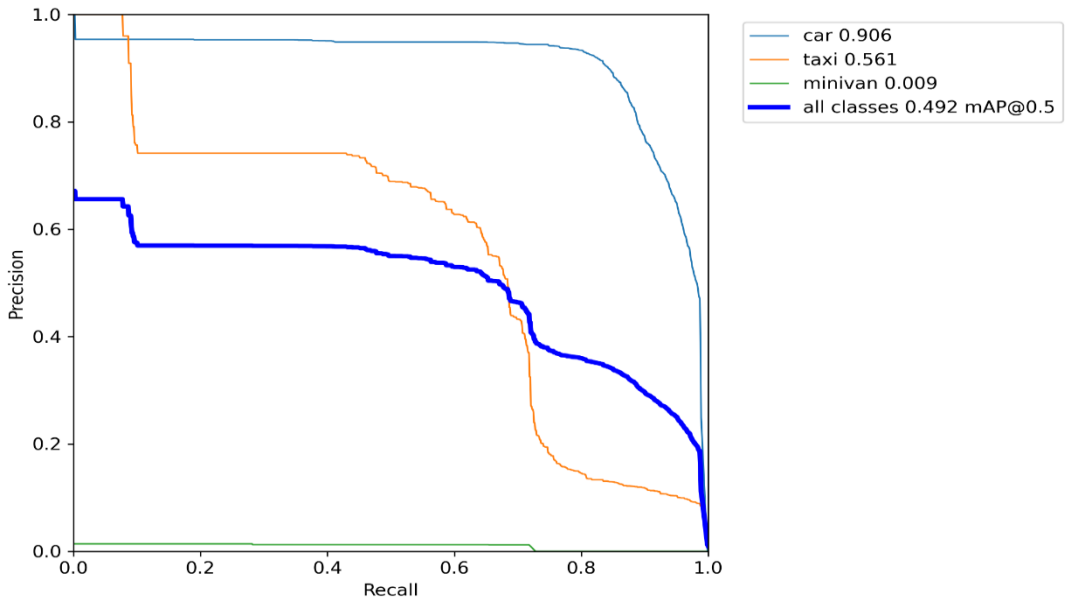

图 4-8 Precision-Recall 曲线图

同时也可以得到训练过程的 mAP 值，该值可从下图 4-9 训练结果图中看到变化趋势。

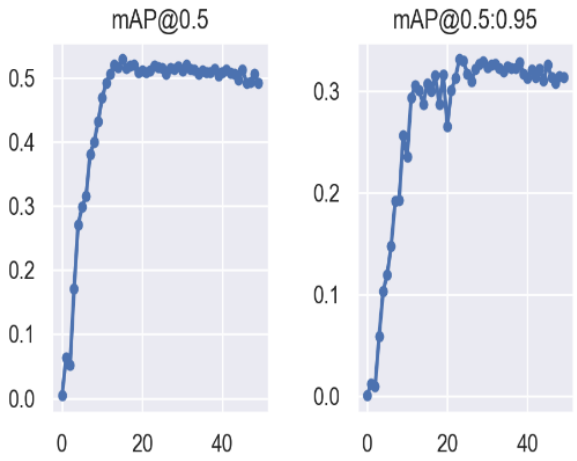

图 4-9 训练结果图

可以得出，训练最终结果为 mAP@0.5 为 0.5567，mAP@0.5：0.95 为 0.4919，可以看出，mAP@0.5 值相对于官方 YOLO v5 项目的 mAP@0.5 值还是相对要高的，可见训练效果可观，鉴于数据集的简单性和实验开发环境的情况来看，使用训练次数为 50 次所就能达到的训练效果，此权重文件可以作为本课题目标识别功能的权重文件进行使用，满足对于交通视频情况下的要求。之后通过对测试集合成的视频进行测试，效果可观，具体效果如下图 4-10 所示。

图 4-10 测试集视频测试效果图

4.3 基于 Deep-SORT 跟踪器实现与效果分析

4.3.1 实现过程

由 Deep-SORT 原论文可知，Deep-SORT 的深度外观模型是在人的重识别数据集上训练得到的，在用于人的多目标跟踪效果很好，但恐怕用于车辆就不一定适用，所以在此，本课题要训练适用于车辆的深度外观模型，从而满足本课题对于车辆目标识别的准确度这一关键特性的要求。

在 4.1 节中准备好的数据集的基础上，首先需要将该数据集全部调整成为与原论文行人重识别数据一样的格式，修改图片文件名格式为[0:4]与图片上一级目录同，[4:6]相机 ID，[6:11]跟踪 ID，[11-15]图片序号，并且将图片改成 128 宽成 256 高。

接着将使用论文中给出的 train_mars.py 文件对于该数据集进行训练，使用 TensorFlow 框架训练，损失模型（--loss_mode）采用默认的 cosine-softmax，在训练过程中每 5 分钟自动保存一组文件，之后生成如下图 4-11 所示的文件内容，该过程保存的文件如选中的文件所示。

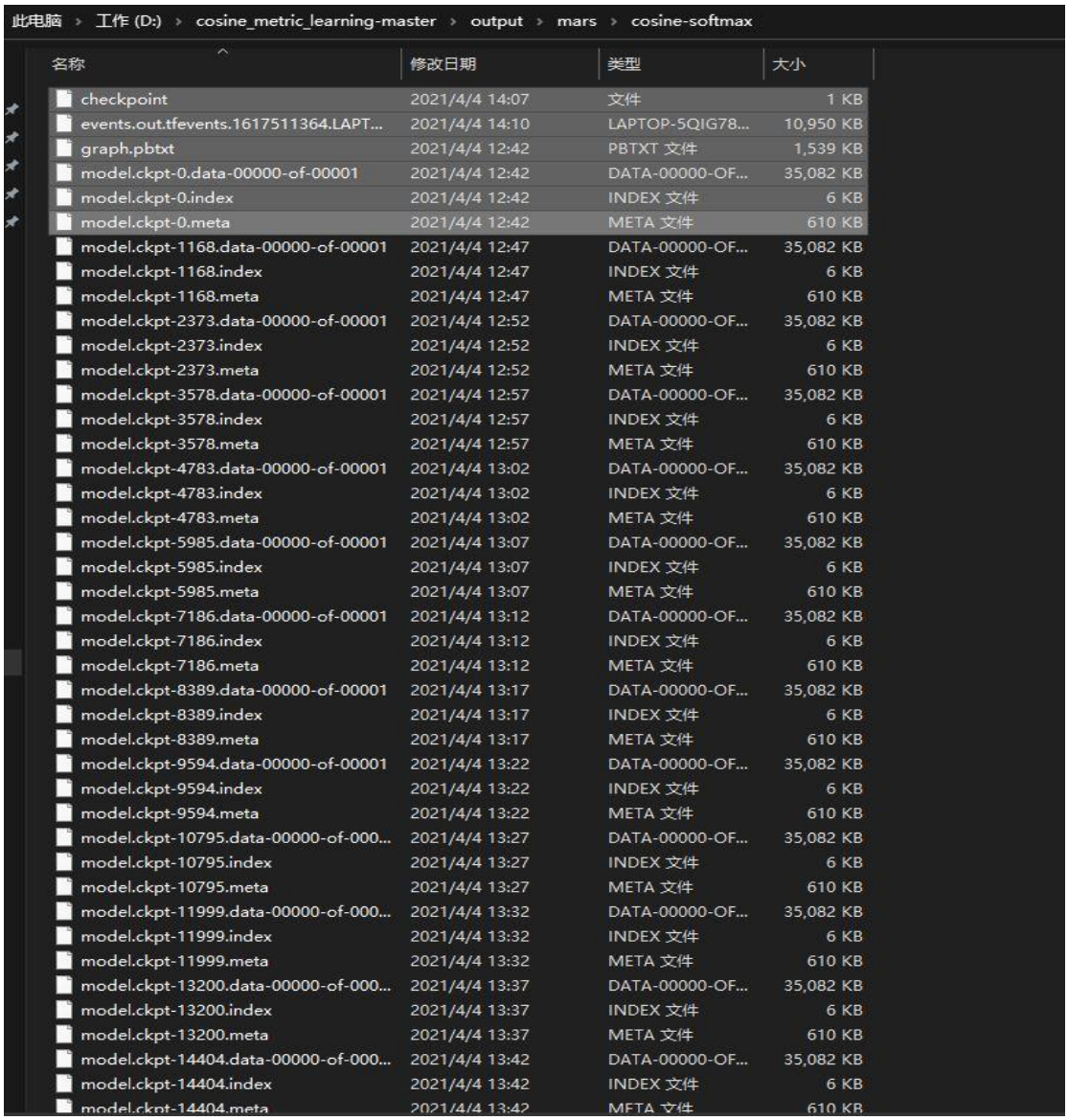

图 4-11 跟踪器深度特征训练得到的文件图

对于该图中所生成的文件进行如下说明，Tensorflow 训练后的模型可以保存 checkpoint 文件或 pb 文件。checkpoint 文件是结构与权重分离的四个文件，便于训练；pb 文件则是 graph_def 的序列化文件，类似于 caffe model，便于发布和离线预测。官方提供 freeze_grpah.py 脚本来将 ckpt 文件转为 pb 文件。Checkpoint 保存断点文件列表，可以用来迅速查找最近一次的断点文件；meta 文件是 MetaGraphDef 序列化的二进制文件，保存了网络结构相关的数据，包括 graph_def 和 saver_def 等；index 文件为数据文件提供索引，存储的核心内容是以 tensor name 为键以 BundleEntry 为值的表格 entries，BundleEntry 主要内容是权值的类型、形状、偏移、校验和等信息。Index 文件由 data block/index block/Footer 等组成，构建时主要涉及 BundleWriter、TableBuilder、BlockBuilder 几个类，除了 BundleEntry 的序列化，还涉及了 tensor name 的编码及优化（比如丢弃重复的前缀）和 data block 的 snappy 压缩。数据（data）文件保存所有变量的值，即网络权值。

在训练过程中使用 tensorboard 这一可视化工具，使用该命令打开浏览器，对训练过程进行实时监控，具体情况展示在下 4.2.2 小节中。待到经过超出 20000 次训练之后，观察到 loss 稳定后，停止训练，此时查看 checkpoint 中最后截止时生成的 ckpt 文件，使用 finalize 命令和 freeze 命令生成最终的深度模型权重.pb 文件，接着即可使用该权重模型进行跟踪。

之后对于 deepsort.yaml 数据配置文件进行更改，使之对应训练好的权重，基于前期设计好的结合 YOLO v5 目标识别器生成好的识别框和 Deep-SORT 目标跟踪器进行对测试视频的跟踪测试，其效果展示在 4.2.2 之中。至此，全部的基于视频的车辆识别与跟踪系统的设计和实现均完成。

4.3.2 效果分析

首先对于提取的车辆深度特征效果进行分析，该深度特征在完成训练时的 weight_loss 和 triplet_loss 以及 total_loss 函数值均保持稳定，分别稳定在 1.8, 0.58, 4 左右，说明训练效果良好。具体 weight_loss 和 triplet_loss 以及 total_loss 损失函数变化情况如下图 4-12, 4-13, 4-14 所示。

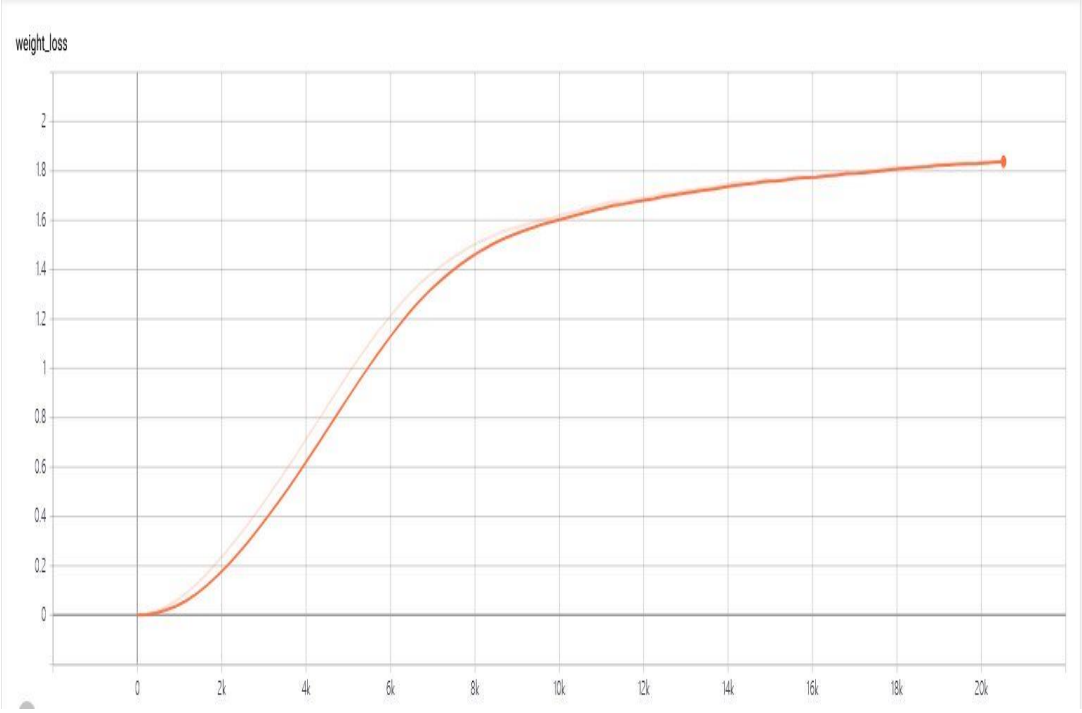

图 4-12 跟踪器深度特征训练 weight_loss 函数变化

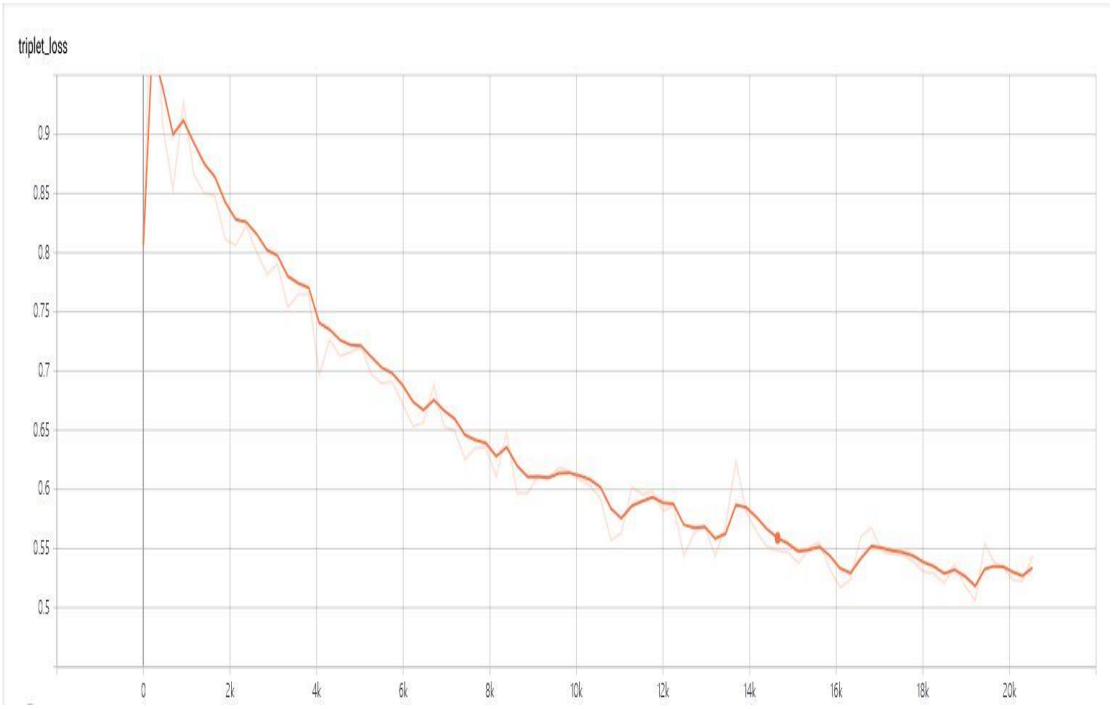

图 4-13 跟踪器深度特征训练 triplet_loss 函数变化

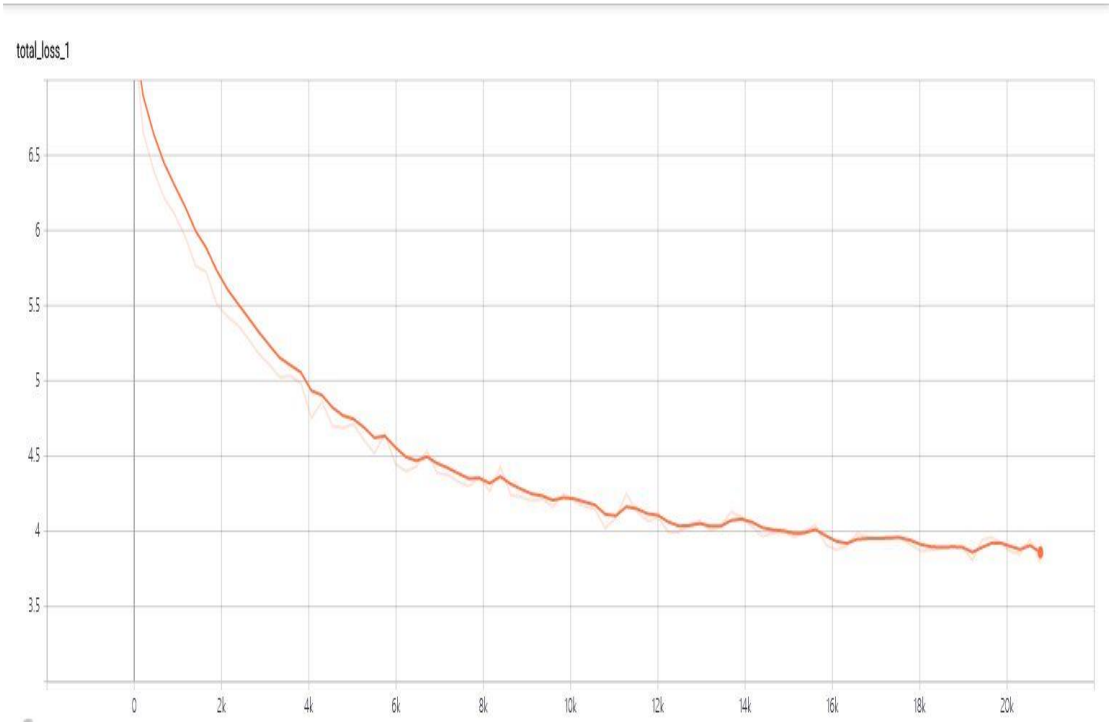

图 4-14 跟踪器深度特征训练 total_loss 函数变化

同时，也可以得到分类的精确度变化过程的图像如图 4-15 所示。我们可以清晰地看出，最后精确度无限趋近于 1，可见精确度也是较高的程度。

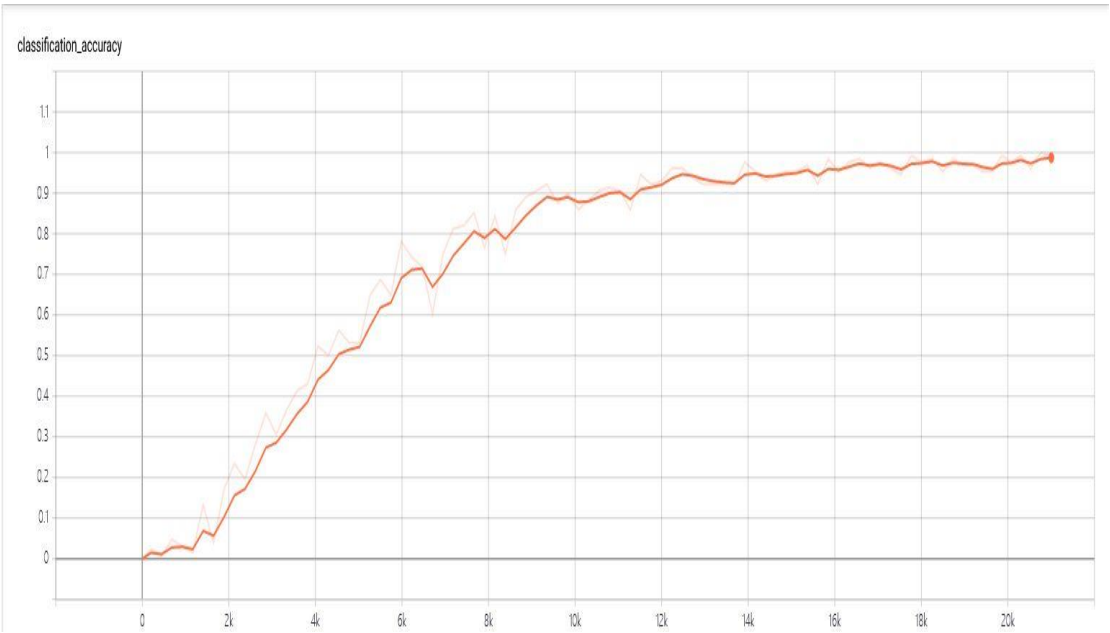

图 4-15 跟踪器深度特征训练 classification_accuracy 函数变化

最后给出整个本课题所设计和实现的基于视频的车辆识别与跟踪系统进行整体系统测试。该测试基于前期训练好的 YOLO v5 模型车辆识别权重文件 best.pt 和 Deep-SORT 车辆跟踪权重文件 mars.pt 文件，进行测试集视频车辆的识别与跟踪。该效果如下图 4-16 所示。

图 4-16 基于视频的车辆识别与跟踪系统效果图

图中的方框表示的是基于目标识别生成的目标识别框，阿拉伯数字标号表示的是 Deep-

SORT 算法中的 ID。可以看出，对于所有带有目标识别框的车辆都可以进行分配 ID，完成跟踪的效果。

4.4 本章小结

本章在基于第三章中对整体系统功能完成设计的基础上，对整个基于视频的车辆识别与跟踪系统进行了实现。首先给出了数据集的采集来源和处理过程，得到了标准化的两部分数据集，分别应用于 YOLO v5 识别和 Deep-SORT 深度特征训练过程。接着基于数据集，给出了具体的训练过程，在完成训练的基础上进行了目标识别和识别的效果分析，通过对于相关的指标进行分析，可以看出，识别的准确性有一定的保证。最后再在完成识别的基础上，基于车辆的深度特征，完成对车辆轨迹的预测和数据关联，实现对车辆的跟踪，并且对于深度特征的训练结果进行了分析，可从评价指标看出，该深度特征权重十分可靠，接着对于整体系统效果进行分析，可以看出，所有被识识别到的车辆可以被 Deep-SORT 算法跟踪，准确度上满足实际应用的需求。

结论与展望

本课题通过分析在交通视频流的场景情况下车辆行驶的特点，以及交通安全对于准确度等的需求，引入了基于 YOLO v5 目标识别算法作为车辆跟踪算法的初始化信息框，从而保证了基于 Deep-SORT 跟踪算法的车辆跟踪算法准确率。本论文主要内容包括：

（1）分析了当今基于视频的车辆识别与跟踪课题的课题背景以及国内外研究现状，确定了该课题的研究内容和研究意义所在。

（2）通过对于不同类型的目标识别算法和目标跟踪算法进行了整体的研究和分析，以及基于前期对于交通场景下的情况的分析，确定了对于准确度和实时性均表现良好的深度学习算法作为主要的工具，采用 YOLO v5 目标识别算法和 Deep-SORT 目标跟踪算法的技术路线。

（3）在确定好技术路线的基础上，对于整个系统的不同模块进行设计，分别设计目标识别模块的功能和目标跟踪模块的功能。

（4）采用城市交通数据集，对目标识别神经网络进行训练，同时采用车辆外观数据集，对目标跟踪所需要的深度特征神经网络进行训练，基于训练好的权重，对目标识别系统和整体的跟踪系统进行分别实现，并且分别给出效果分析。验证达成了本课题设计的系统能够有效地交通场景下车辆识别与跟踪的目的。

论文对于基于交通场景下的车辆和跟踪进行了深入的研究，并且取得了一定的效果，但是对于复杂且灵活的交通实际场景，本论文所提出的技术路线和设计的系统还需要进一步的研究和完善，主要有如下几个方面：

（1）本课题所采用的数据集为实际采集的交通数据集，数量十分有限，共 1700 余张照片，尽管使用 yolov5m.pt 文件作为迁移学习训练的基础，但是训练效果仍然十分具有局限性。数据集仅限于对白天且为阴天情况下的交通视频流，没有考虑到晴天雨雪天等其他天气条件下以及夜间情况下的情况。所以，对于具体的复杂的不同的交通场景，进行车辆跟踪仍然需要大量的数据作为实验基础。

（2）本课题所采用的数据集在实时性上仍然具有一定的局限性，这对于整个系统来说，存在着一定的缺陷，该数据集实际只有达到 6-8 FPS，这将会导致识别视频在播放时存在一定的不流畅，所以可以看出存在着一定的掉帧现象，同时目标识别的训练结果来说，mAP 值受到了数据集这一方面很大的影响。对于该问题，需要进一步考虑使用更加优秀的设备来采集更加具有实时性的图片数据集。

（3）本课题所采用的实验设备较为落后，在训练的过程中存在着一定的计算误差和时间效率限制，可以考虑再进一步使用专业的台式机设备作为训练的设备，解决此方面的问题。

致 谢

时间过得非常快，四年大学本科生生涯即将结束。在这四年的时间中，我经历了许多，从大一进入时的呐喊，到大四时的朝花夕拾。长安四载，所获良多，不仅仅限于书本上的基础知识和编写代码的能力和经验，更多的是收获成长，有了对自己的了解，也有了对生活的理解。

本论文的顺利完成，离不开我的导师惠飞教授的悉心的指导。是他帮助我消除了选题的迷茫、修正了草案的缺陷、克服了设计的困难、提高了论文格式的规范性。在大学期间的学习阶段中，惠老师经常会很有耐心和宽容心的对所教授的课程给予我们指导，让我们热爱上计算机科学这门学科；此外，在这半年以来的毕业设计交流与指导当中，导师惠飞教授认真的工作作风、一丝不苟的教学习惯使我受益良多，每次会议都会面对我们的问题进行悉心指导和解决，并且给予我们支持和鼓励。同时，我的学业毕业设计的完成离不开长安大学在设备、场所和经费上的支持，给予了我们 211 学校这么好的一个平台。在此，我对我的导师惠飞以及培育我的长安大学表示衷心的感谢与敬佩，祝愿惠飞老师身体健康，家庭美满，事业顺利，祝愿长安大学蒸蒸日上。

同时，我还要感谢在毕业设计中帮我打开思路与耐心指导的靳少杰博士学姐，是她在训练 YOLO v5 的过程中让我少走了很多弯路，给予了我数据集采集和制作过程的帮助，并且给我讲明了训练过程中的注意事项，在论文撰写中，也给予了我很大程度上的建议，帮助我顺利完成毕业设计与论文撰写。祝学姐科研顺利。

感谢四年来所有幸认识的好同学好兄弟们，感谢遇见，感谢四年中的互相帮助和一起走过，祝未来可期。

感谢父母和家人，感谢您们无私的包容和陪伴，给予我勇气，助我顺利完成学业。祝身体安康。

最后，在此由衷地感谢审阅本文的专家教授，感谢您们能够百忙之中来审阅，请不吝批评指正，感谢您们的指导。祝您们身体健康，事业顺利，生活幸福。

参考文献

[1] Foote R S. Automatic vehicle identification: Tests and applications in the late 1970's[J]. Vehicular Technology IEEE Transactions on, 1980, 29(2): 226-229.

[2] 杨戈，刘宏。视觉跟踪算法综述 [J]. 智能系统学报，2010，5(02)：95-105.

[3] 武晓洁。基于视频的车辆识别与跟踪方法研究[D]. 长安大学，2018.

[4] 龚健雅，季顺平。从摄影测量到计算机视觉。武汉大学学报（信息科学版） [J]. 2017；42(11)：21-5+118.

[5] J K, S Y G, J K D. Moving Object Detection under Free-Moving Camera. IEEE International Conference on Image Processing 2010 [C]. p. 4669-72.

[6] B K, B R, H L. Detection and tracking of independently moving objects in urban environments. International IEEEC on reference on Intelligent Transportation Systems IEEE 2010 [C]. p. 1396-401.

[7] Kameda Y M M. A Human Motion Estimation Method Using 3-Successive Video Frames. International Conference on Virtual Systems and Multimedia 1996 [C].p.135-40.

[8] 甘明刚，陈杰，刘劲，王亚楠。一种基于三帧差分和边缘信息的运动目标识别方法。电子与信息学报 [J]. 2010；32(04)：136-9.

[9] 王静静，林明秀，魏颖。基于灰度相关的帧间差分和背景差分相融合的实时目标识别[C]. 2009年中国智能自动化会议论文集（第六分册）【中南大学学报（增刊）】2009.

[10] J C, P C, P F. Improved particle filter for non linear problems. IEEE Proceedings-Radar, Sonar and Navigation[J].2002;146(1):2-7.

[11] F D, Z L, D K, a le. Adapting the Sample Size in Particle Filters Through KLD-Sampling. International Journal of Robotics Research [J].2003;22(12):985-1003.

[12] C B, Y W, H L, H J. Real Time Robust L1 Tracker Using Accelerated Proximal Gradient Approach. IEEE Conference on Computer Vision and Pattern Recognition 2012[C]. p. 1830-7.

[13] Q W, F C, L X W, H Y M. Object Tracking via Partial Least Squares Analysis. IEEE Transactions on Image Processing [J].2012;21(10):4454-65.

[14] D R. An Algorithm For Tracking Multiple Targets. IEEE Transactions on Auto-matic Control [J].2013;24(6):843-54.

[15] Kalal. Z, Mikolajczyk, Matas. J K. Tracking-learning-detection. IEEE Transaction on Pattern Analysis and Machine Intelligence [J].2012;34(7):1409-22.

[16] F H J, C R, P M, a le. High-Speed Tracking with Kernelized Correlation Filters. IEEE Transactions on Pattern Analysis &Machine Intelligence [J].2015;37(3):583-96.

[17] Lipton A J, Fujiyoshi H, Patil R S. Moving Target Classification and Tracking from Real-time Video[C]. IEEE Workshop on Applications of Computer Vision. IEEE Computer Society, 1998: 8.

[18] Solehah S, Yaakob S N, Kadim Z. Moving object extraction in PTZ camera using the integration of background subtraction and local histogram processing[C]. IEEE Symposium on Computer Applications & Industrial Electronics. IEEE, 2013: 167-172.

[19] 韦超现。基于视觉传达的多帧图像特征目标跟踪仿真[J]. 计算机仿真，2021，38(01)：

404-407+420.

[20] 田丹，臧守雨，涂斌斌。具有空间调整和稀疏约束的相关滤波跟踪算法 [J/OL]. 图学学报：1-7 [2021-03-31].

[21] Bewley, Alex and Ge, Zongyuan and Ott, Lionel and Ramos, Fabio and Upcroft, Ben. Simple Online and Realtime Tracking. [C]//2016 IEEE International Conference on Image Processing (ICIP).

[22] Bewley A,Ge Z,Ott L,et al. Simple online and realtime tracking with a deep association metric. [C]//2017 IEEE International Conference on Image Processing (ICIP) , 2017: 3645－3649.

[23] Goodfellow, I., Bengio, Y., Courville, A.. Deep learning (Vol. 1). Cambridge：MIT press，2016：326-366

[24] Liu W,Anguelov D,Erhan D,et al . SSD:Single Shot MultiBox Detector[C]//European Conference on Computer Vision, 2016.

[25] Redmon, Joseph, Farhadi, Ali. YOLOv3: An Incremental Improvement[J]. Computer Vision and Pattern Recognition,arXiv:1804. 02767, 2017.

[26] 何之源。21个项目玩转深度学习。北京：电子工业出版社，2018年：88-90.

[27] 目标识别（Object Detection）算法汇集。CSDN。2018年08月06日。

[28] Redmon, Joseph and Farhadi.YOLOv3:An Incremental Improvement[D].arXiv.2018.

[29] https://download.csdn.net/download/chen_yuazzy/10601135.