基于 YOLOv2 目标检测器的行人检测

赵惠

指导教师 郑艳

厦门大学

厦門大學

本 科 毕 业 论 文 （设 计）

（主修专业）

基于 YOLOv2 目标检测器的行人检测

Pedestrian Detection Based on YOLOv2 Object Detection

姓　　名：赵惠

学　　号：24320132202527

学　　院：软件学院

专　　业：软件工程

年　　级：2013 级

校内指导教师：郑艳　讲师

二〇一七 年 五 月 十四 日

厦门大学本科学位论文诚信承诺书

本人呈交的学位论文是在导师指导下独立完成的研究成果。本人在论文写作中参考其他个人或集体已经发表的研究成果，均在文中以适当方式明确标明，并符合相关法律规范及《厦门大学本科毕业论文（设计）规范》。

该学位论文为（　　）课题（组）的研究成果，获得（　　）课题（组）经费或实验室的资助，在（　　）实验室完成（请在以上括号内填写课题或课题组负责人或实验室名称，未有此项声明内容的，可以不作特别声明）。

另外，本人承诺辅修专业毕业论文（设计）（如有）的内容与主修专业不存在相同与相近情况。

学生声明（签名）：

年　　月　　日

致　谢

凤凰花开的季节，到了属于我的毕业季。四年的大学时光，转瞬即逝，留下的是一个心底永远的回忆。大一加入体育部，和体育部小伙伴们团结协作组织活动，一起作为西边烤场的常客；大二宿舍，教室，饭堂三点一线，潜心学习，有了满意的成绩；大三加入贝壳工作室，参与厦大迎新系统的开发，体会肩上的那份责任；大四广州实习，感受深度学习的魅力。这四年，紧张但充实，收获友谊，收获成长，少一分稚气，多一分稳重。感谢厦大，临别之际，向您表达最崇高的敬意！

值此论文完成之际，谨向所有关心和支持我的人们致以诚挚的谢意！

首先，我要感谢我的导师郑艳讲师。在毕业设计阶段，根据我的实习环境，给出毕业设计的研究方向，督促我认真完成毕业论文，对我的论文进行批阅，给提出指导性的意见和建议。在此，谨向郑艳老师致以最诚挚的感谢！

其次，我要感谢广州市维安科技股份有限公司，提供毕业设计用的硬件设备。同时感谢陈海玉同事在实习期间给我提供了住所，感谢王喆同事在实习期间给予我电脑硬件设备方面的帮助。

再者，我要感谢厦门大学软件学院全体老师的辛勤栽培。感谢我的舍友，体育部的成员，贝壳工作室的成员，大学四年，一起成长，感谢你们的陪伴。

最后，我要感谢我的家人，感谢一直以来的鼓励与支持。在未来的日子里，我会更加努力学习和工作，不辜负你们的期望。

摘　要

行人检测作为自动驾驶、智能监控、智能机器人等许多应用的底层基础，近年来，已经引起了超过一般目标检测的特别关注。随着卷积神经网络（Convolutional Neural Networks, CNNs）的在计算机视觉领域的不断发展，目标检测领域在较高精度下，已经可以达到实时的检测性能。比如，以回归的方法处理检测问题的 YOLOv2[1]，是先进的、实时的目标检测器。在一个 Titan X 上每秒可检测 40 帧分辨率为 544*544 的图片。行人一般具有比较小的尺度，行人各种姿势和不同遮挡水平对行人检测来说都是巨大的挑战。为了保证精度，代表行人检测先进技术的几个行人检测器，RPN+BF[2]，SAF R-CNN[3]，CompACT-Deep[4]检测一帧图片的时间约为 0.5 秒。目前最精确的行人检测器 F-DNN[5]检测速度约每秒 9 帧，这还远远达不到监控视频每秒至少检测 25 帧的实时性要求。

本论文旨在修改 YOLOv2 目标检测器，将他专门用于检测行人。在保证检测速度的同时，提高行人检测的精度。

关键词：行人检测；卷积神经网络；YOLOv2 目标检测器

Abstract

Pedestrian detection, as a basic foundation of real-world applications such as automatic driving, intelligent surveillance and intelligent robot, has attracted special attention beyond general object detection. With the continuous development of Convolutional Neural Networks in the field of computer vision, object detection has been able to achieve real-time detection performance with high accuracy. For example, YOLOv2[1] detector, which solve detection problem as regression problem, is a state-of-the-art, real-time object detection system. On a Titan X it processes 544*544 images at 40 frames per second (FPS). Small scales, difference postures and occlusion levels are huge challenges for pedestrian detection. In order to ensure accuracy, the state-of -the-art pedestrian detectors, such as RPN+BF[2], SAF R-CNN[3] and CompACT-Deep[4] detector, detect an image in about 0.5s. The most accurate pedestrian detector F-DNN[5] detect an image with a speed of about 9 FPS, which is still far less than the real-time requirement of at least 25 FPS to detect surveillance videos.

This paper aims to modify the YOLOv2 object detector, making it specially designed for pedestrian detection. The goal is to improve the accuracy of pedestrian detection while ensuring the detection speed.

Key words： Pedestrian Detection; Convolutional Neural Networks; YOLOv2 Object Detector

目 录

Contents

第一章 绪论

1.1 研究背景及意义

目标检测的任务，是在图片或视频中检测出目标，标明目标的类别和框出目标的位置，目标可以是多个类别。行人检测是一类特殊的目标检测，只在图片或视频中检测出行人，框出行人的位置，近年来受到了超越一般目标检测外的特别关注。行人检测是许多应用的重要组件。

1、车辆辅助驾驶与自动驾驶

近年来车辆越来越多，交通事故也频频发生。据世界卫生组织统计，“一名成年行人如果被时速低于 50 公里/小时的汽车碰撞，死亡几率在 20%以下，但如果被时速为 80 公里/小时的汽车碰撞，则死亡风险几乎为 60%”[6]。这说明控制车速可以有效降低交通事故的死亡风险。如果在车辆行驶的过程中实时检测出处于危险区域的行人，使司机有足够的时间采取措施，或是车辆自动采取措施，就能降低发生交通事故风险或者死亡风险，所以行人检测是车辆辅助驾驶与自动驾驶的关键组件之一。目前的车辆辅助驾驶和无人驾驶是学术界和工业界共同关注的焦点，但目前行人检测的精度和速度都还有待提高。

2、智能监控

在安防领域，目前的监控工作大多还是由人工完成，这需要大量的人员。工作枯燥但又必须时刻保持警惕，稍有不慎，带来的损失不容小觑。设想一下，如果在监狱、金库、珠宝店等在特定时间避免行人出入的地方，计算机能实时获取监控视频，精确地检测出行人，记录行人出现的时间。这样就可以节约监控的人力成本，提高安防水平。一旦发生盗窃案件，越狱事件等，也能快速定位到发生的时间，有助于快速破案。

3、视频结构化

视频结构化是当前的一个发展趋势，利用行人检测找出视频中的人，进一步对行人的肤色，着装，性别等进行判断，构造新型的结构化数据，存储到数据库里，方便事后取证。当然还可以结合机动车检测等，存储别的视频结构化信息，提供多种多样的服务。

4、智能机器人

智能机器人也是近年来最热门的研究方向，大部分智能机器人的主要服务对象是人，所以检测出人是它的基本技能，在此基础上才能更好地为人服务。

行人检测的应用场景不局限于上述的几种，随着时间推移和科技发展，会有更多各种各样的需求，因此行人检测近年来受到了广泛的关注。

1.2 行人检测研究现状

目前目标检测有两个基本的框架：R-CNN 系列的基于候选区域（Region Proposals）的目标检测框架，这种框架把检测问题当做分类问题解决，另一个是 YOLO 系列的目标检测框架，这种框架则把检测问题当做回归问题来解决。大多数的目标检测和行人检测研究都是基于把检测问题当做分类问题。

传统的目标检测利用穷举的策略进行区域选择，采用滑动窗口，设置不同的窗口大小和不同的长宽比对图像进行遍历，然后提取一些手工设计的特征如 SIFT、HOG 等，再给 SVM、Adaboost 等分类器进行分类。基于滑动窗口的区域选择方式选择出来的窗口太多且没有针对性，手工设计的特征对于多样性的变化不够健壮。于是出现了基于候选区域的一系列深度学习目标检测算法。

文献[7]提出了候选区域的概念，先从图像中找出可能包含目标的位置，可以用 Selective Search、Edge Boxes 等方法实现。然后对每个候选区域利用卷积神经网络提取出关键特征，输入到 SVM 进行分类。这种方法称为 R-CNN。候选区域的方式比滑动窗口的方式可以明显减少提取窗口的数量，而且保持较高的召回率。但是 R-CNN 检测速度太慢，假设对图像提取了 2000 个候选区域，之后每个候选区域就相当于是一张图像一样进行卷积网络提取特征和分类，那么检测实际上相当于对 1 张图像进行了 2000 次提取特征和分类的过程。这个问题在文献[8]中解决了。

文献[8]提出了 SPP-NET 网络，利用空间金字塔池化层（Spatial Pyramid Pooling Layer），对每个候选区域使用不同大小的金字塔映射，使不同尺寸的候选区域在进入全连接层时可以统一到固定的长度（卷积层的输入不限制输入图像尺寸，但全连接层限制图像的尺寸），从而使得一张图像只用提取一次卷积层特征，然后将候选区域在原图上的位置映射到卷积特征图上。SPP-NET 因此大大加快了检测速度，但是 R-CNN 和 SPP-NET 还有一个共同的问题，训练分为多个阶段，要微调网络，然后训练分类器，再训练边框回归器。这个问题在文献[9]中进

行了改善。

文献[9]提出了 Fast R-CNN 网络，它用感兴趣区域池化层（Region-of-Interest Pooling Layer）代替了 SPP-NET 中的金字塔映射，感兴趣区域池化层只需要对每个候选区域下采样到一个 7*7 的特征图即可。Fast R-CNN 用 Softmax 代替 SVM 分类，利用多任务损失函数将边框回归加入到了网络中，使得在不考虑候选区域选取过程的情况下，整个检测器的训练过程是端到端的。但是候选区域选取的过程本身就是一个耗时的过程，文献[10]对这部分做了改进。

文献[10]提出了 Faster R-CNN 网络，它采用了 RPN（Region Proposal Network）进行候选区域的提取，再将提取到的候选区域给 Fast R-CNN 进行分类。RPN 使用卷积神经网络直接产生候选区域，只在最后的卷积层上使用滑动窗口，用 Anchor 机制和边框回归得到多尺度多长宽比的候选区域。使用 RPN 可以得到更少而且质量更高的候选区域。Faster R-CNN 虽然在目标检测上取得了不错的成果，但对于行人检测来说，效果并不好。

中山大学的张立亮等人对 Faster R-CNN 为什么不能很好的检测行人进行研究，并提出了 RPN+BF 的方法进行行人检测[2]。他们认为 Faster R-CNN 只在 RPN 最后一层卷积层提取特征，特征图的分辨率不足以检测尺度比较小的行人，而且也缺少容易误判成行人的背景样本进行训练，于是提出 RPN+BF 的方法。该方法同样使用 RPN 提取候选区域，但是是在几个较低层、分辨率比较高的卷积特征图上提取特征，并将这些特征简单的连接起来。把得到的特征给下游的级联增强森林（Cascaded Boosted Forest）进行分类。该方法在多个行人检测基准数据集上取得了不错的效果，在 Caltech 数据集合理行人的评估设置上排名第二，检测时间约每秒两帧。

在Caltech数据集多个评估设置上排名第一的是融合深神经网络（FDNN）[5]。该方法利用SSD[11]生成行人的候选区域，SSD是把检测当做回归问题来做的一个目标检测算法，得到的候选区域尽可能包含全部的行人。然后利用基于软拒绝的网络融合方法（Soft-rejection Based Network Fusion Method），融合了多个深层神经网络分类器形成一个分类网络，减少SSD产生的大量的假阳例，给出最终的分类结果。他们还进一步提出了利用有语义分割作用的上下文聚集扩展卷积网络作为另一个分类器集成到网络的融合架构里，以提高检测精度。他们的方法检测时

间约每秒十帧，这个速度相比于检测精度排名靠前的几个行人检测算法来说是最快的了，但也达不到实时的检测效果。

1.3 研究内容

行人检测已经成为独立于目标检测的一类问题，他的发展较目标检测相对慢一些。精度较高的行人检测器还不能达到实时检测的要求，但目标检测器可以。

YOLOv2 是目前先进的实时目标检测系统之一，是 YOLO[12]目标检测器的改进版本。它把目标检测问题当作回归问题来解决，直接在原图上进行目标的检测与定位，在保持精度的情况下，极大地提高了检测速度。具体的，在 Titan X 上处理图像的速度在 40 至 90 FPS，在 PASCAL VOC 2007 数据集上 mAP（Mean Average Precision）为 78.6%，在 COCO test-dev 数据集上 mAP 为 44.0%。

本文旨在修改YOLOv2目标检测器，将它用于行人检测，经过实验和分析提高检测精度。

1.4 论文组织结构

本文一共包括五章，论文的具体内容安排如下：

第一章 绪论。本章主要阐述行人检测的研究现状，本文的研究内容及论文组织结构。

第二章 相关知识概述。本章主要介绍本篇论文开展的基础，包括神经网络，卷积神经网络，YOLOv2 目标检测器以及行人检测权威的几个数据库。

第三章 基于 YOLOv2 的行人检测。本章主要介绍将 YOLOv2 目标检测器改成行人检测器的几个修改方向，前期的数据准备和评估设置。

第四章 实验和结果分析。本章主要介绍实验环境，具体的实验设置和展示实验结果，并进行分析。

第五章 总结与展望。本章主要对全文进行总结以及对以后工作的展望。

第二章 相关知识概述

本文目标是修改 YOLOv2 目标检测器，使其专门用于行人检测，本章将对涉及到的技术和知识进行介绍。YOLOv2 目标检测器的基础是神经网络与卷积神经网络，所以本章先从产生背景，网络结构和使用方法三个方面对神经网络进行初步的介绍。接下来对新型的神经网络——卷积神经网络进行介绍，也包含三个方面：背景，网络结构和主要特点。再者对 YOLOv2 目标检测器的网络结构和具体细节进行较详细的介绍。最后介绍本文使用到的三个行人数据库，重点介绍目前图片数量最多，检测难度最大的 Caltech 行人数据库。

2.1 人工神经网络

1、产生背景

人工神经网络是机器学习的一个新的领域。传统的机器学习大致过程如图 2-1 所示：

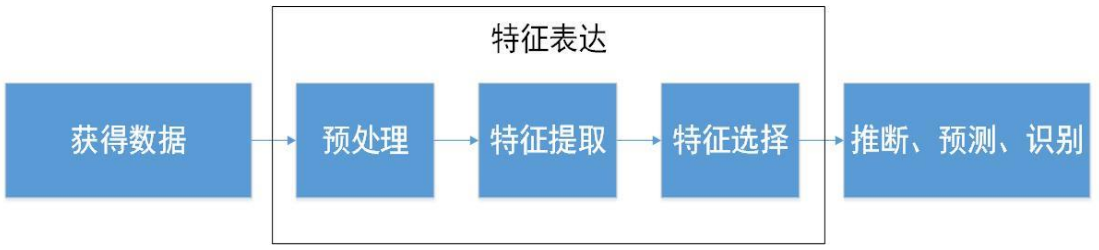

图2-1 传统机器学习过程图

传统的机器学习主要在最后一个阶段，即推理、预测或识别方面做了很多的研究，比如线性回归、SVM 等。而中间三个部分，合起来就是特征表达。特征表达的好与坏，直接关系到最终算法的准确性。这一部分一般是由人工完成的，依靠人工来提取特征。虽然也有一些人工设计的效果不错的特征，比如 SIFT，HoG 特征。但是由人工设计特征，不仅费时、费力、需要专业的知识，而且设计出来的特征也难以具有通用性，不够鲁棒。于是我们希望算法能够自动提取好的特征。人工神经网络就是一种可以自动提取特征的机器学习算法，它的灵感来源于人脑的视觉机理。

科学家们利用猫做实验来研究瞳孔区域与大脑皮层神经元的对应关系，发现

了一种称为“方向选择性细胞”的神经元细胞。当瞳孔发现了指向某个方向的物体边缘时，这种神经元活跃。视觉信息的处理过程或许是一个不断迭代、抽象的过程，如图 2-2 所示：

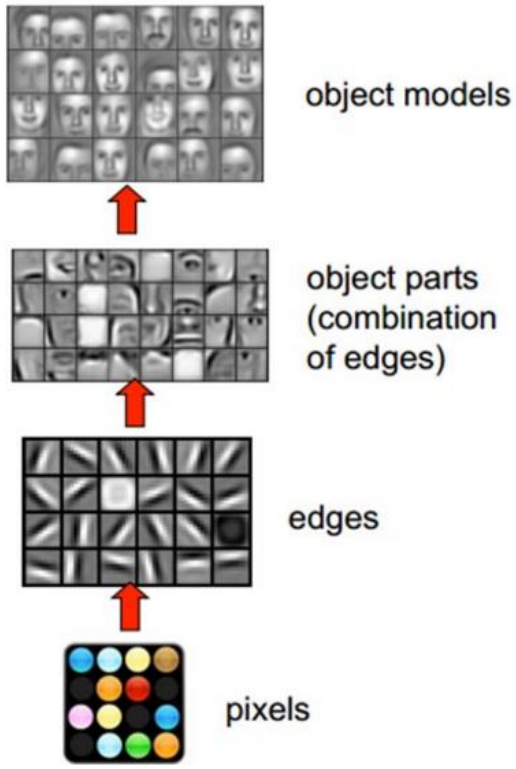

图2-2 大脑视觉信息处理过程[13]

由像素组合抽象成物体边缘，由物体边缘组合抽象成人脸各个部分，再由人脸各个部分组合抽象成各种人脸。人工神经网络就借鉴了这个过程。机器学习实际上就是在找从给定输入到目标输出的一个映射函数，人工神经网络其实就是用一个层次型的结构表示出了一个复杂的映射函数。

2、人工神经网络结构

神经元是神经网络的基本单元，它的设计灵感来自于生物学上神经元的信息传播机制。神经元有两种状态，兴奋和抑制。一般情况下，大多数神经元处于抑制状态，一旦某个神经元受到刺激，导致它的电位超过一个阈值，这个神经元会被激活，处于“兴奋”状态，进而向其他神经元传播化学物质，其实就是信息[14]。图 2-3 为生物神经元，生物神经元包含三个部分：树突、细胞体和轴突。

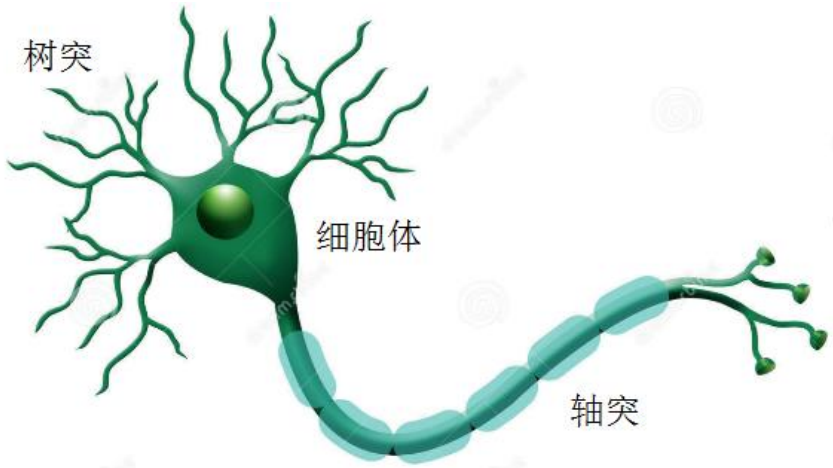

图2-3 生物神经元[14]

科学家将图 2-3 的生物神经元结构用一种简单的模型进行表示，构成了一种称为“M-P 神经元模型”的人工神经元模型，如图 2-4：

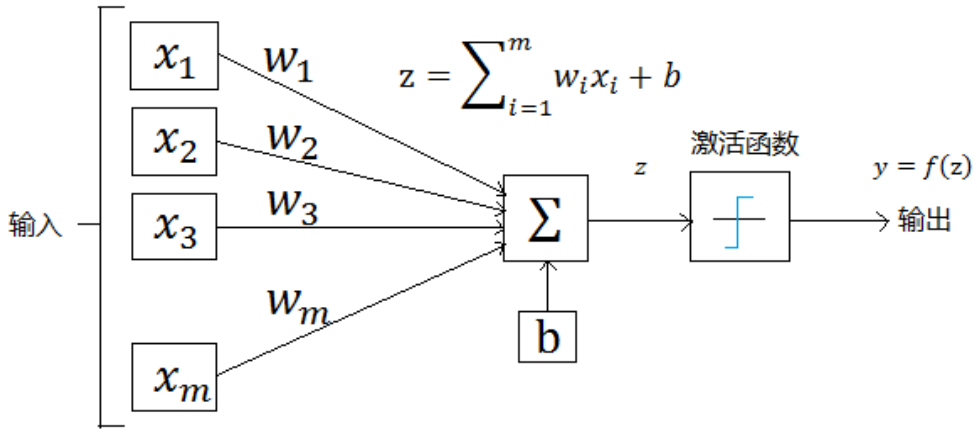

图 2-4 人工神经元模型

图 2-4 中 $x_1,x_2,\ldots x_m$ 是来自其他神经元的刺激，也就是这个神经元的输入，$w_1,w_2,\ldots,w_m$ 是权重，b 是偏置。这个神经元的细胞体部分对树突接收到的刺激做个汇总，把不同的刺激乘以他们的权重再加起来。因为不同的神经元对收到的不同的刺激反应可能不同，所以需要乘以相应的权重。比如一个神经元对颜色比较敏感，那么有关颜色刺激的权重就大一点。有了总刺激后，要看看这个总刺激有没达到使这个神经元兴奋的阈值，所以要加个偏置 b，最后超过阈值的刺激就为

z。然后神经元要决定怎么处理这个刺激，所以刺激 z 要经过一个激活函数 f，这个激活函数决定这个神经元的输出。最后这个神经元的输出 y 如下：

$$y = f\left(\sum_{i=1}^{m} w_i x_i + b\right) \qquad \text{（公式 2-1）}$$

神经元不同的连接就组成了不同的神经网络结构，权值和偏置就是网络的参数。全连接网络是指上层神经元和下层神经元全连接的神经网络。神经网络一般包含输入层，隐含层和输出层，可有多个隐含层。说一个神经网络含有几层，一般是指除了输入层之外所有层的层数。图 2-5 是一个 3 层的全连接神经网络示意图，含有 2 个隐含层。

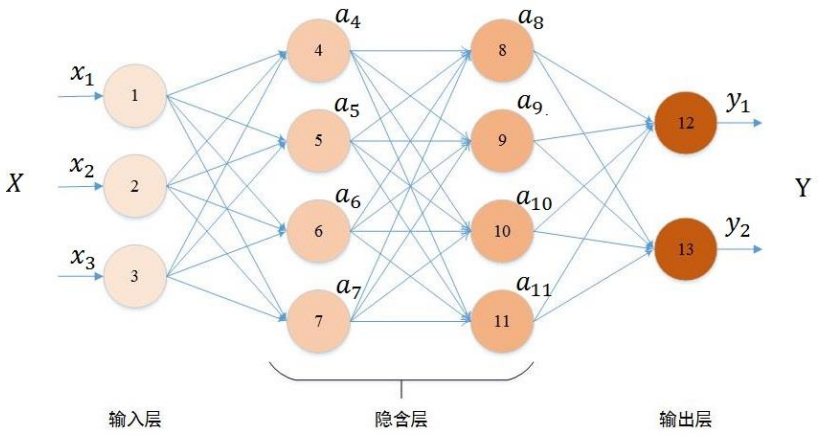

图 2-5 全连接网络示意图

3、人工神经网络的使用方法

统计学习方法利用大量的数据，在数据中发现规律和模式。这种数据驱动的方法分为如下三个步骤：

（1） 准备用于训练和测试的数据集

（2） 利用机器学习方法训练模型

这步又分为以下三步：

（a） 定义模型的假设空间

（b） 定义模型选择的准则

（c） 定义模型选择的算法

（3） 用测试集来评估分类器的好坏

神经网络模型是机器学习模型的一种，所以利用神经网络模型解决问题，也就是上述的三个步骤。

首先要定义模型的假设空间。假设空间其实就是一组函数（也称为一组假设，得分函数），我们要定义这组函数，使它包含最优的函数。如果函数的参数确定，就是一个函数；如果函数的参数不确定，就是一组函数。人工神经网络其实就是以分层形式表示的一个映射函数，他的参数就是网络的权值和偏置。神经元的连接方式确定了网络结构，网络结构就决定了网络的权值和偏置个数。也就确定了模型的假设空间。最后要从假设空间中选择最好的假设，其实就是确定网络参数的过程。

然后要定义模型选择的准则，也就是希望模型达到什么样的效果，即定义目标函数。比如手写字体识别，给定一张手写数字的图片，我们希望网络输出的结果与该图片真实表示的数字一致。所以我们训练的目标就是使得网络的输出与真实值越接近越好，这样我们就可以定义目标函数（也成称为损失函数）。这样模型的选择问题就变成了最优化问题。

最后，要确定模型选择的算法，即如何得到最优的参数组。神经网络常用的优化方法是反向传播算法（Backpropagation algorithm, 简称 BP 算法）。BP 算法的主要思想如下：

（1） 将训练集数据输入到人工神经网络输入层，经过隐含层，到达输出层，并输出结果，这是前向传播过程。

（2） 计算输出结果与真实值之间的误差，并将该误差从输出层向隐含层反向传播，直至传播到输入层，这是反向传播过程。

（3） 在反向传播过程中，根据误差调整各个参数的值

（4） 不断迭代上述过程，直至收敛。

所以利用神经网络模型解决问题，通常是需要根据实际问题来构造出网络结构，然后确定损失函数和优化方法，然后通过训练样本和优化方法来迭代找到最优参数组。

2.2 卷积神经网络

1、基础介绍

卷积神经网络是人工神经网络的一种，其网络结构更接近于生物神经网络，

主要用于计算机视觉和语音识别领域。它在人工神经网络的基础上，引入卷积运算来自动提取特征，这样得到的特征对于平移、尺度缩放、旋转、扭曲等其他形变有着高度的不变性，通用性也更好。卷积神经网络还利用空间位置关系，采用稀疏连接和权值共享的方式，减少需要训练的参数数量，提高反向传播训练效率。

2、网络结构

卷积神经网络模型的种类很多，但基本结构大致相同，如图 2-6 所示。

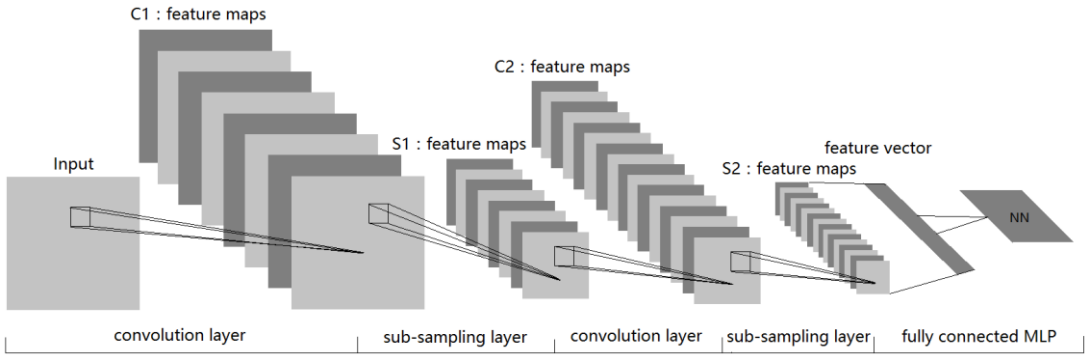

图 2-6 卷积神经网络示意图

卷积神经网络有多个隐含层，每层包含多个二维平面，每个平面包含多个神经元。在卷积神经网络中，这种二维平面称为特征地图（Feature Map），图 2-6 中的 C-层表示卷积层（Convolution Layer），S-层表示下采样层（Sub-sampling Layer）。典型的卷积神经网络由输入层、卷积层、下采样层、全连接的多层感知机（Fully Connected MLP）构成。经过简单预处理的输入图像作为输入层，常用的预处理操作为图像去均值（将输入数据各个维度都中心化为 0，避免数据过多偏差，影响训练效果）和调整图像大小等。输入层后接一系列交替排列的卷积层和下采样层。

卷积层的主要作用是利用卷积操作提取出图片中的特征，它是卷积神经网络中非常重要的层。卷积操作就是一块图像数据和滤波矩阵做内积的操作。滤波矩阵又叫卷积核（Convolution Kernel）或滤波器（Filter），是一组固定的权重。图 2-7 是卷积操作的示意图。

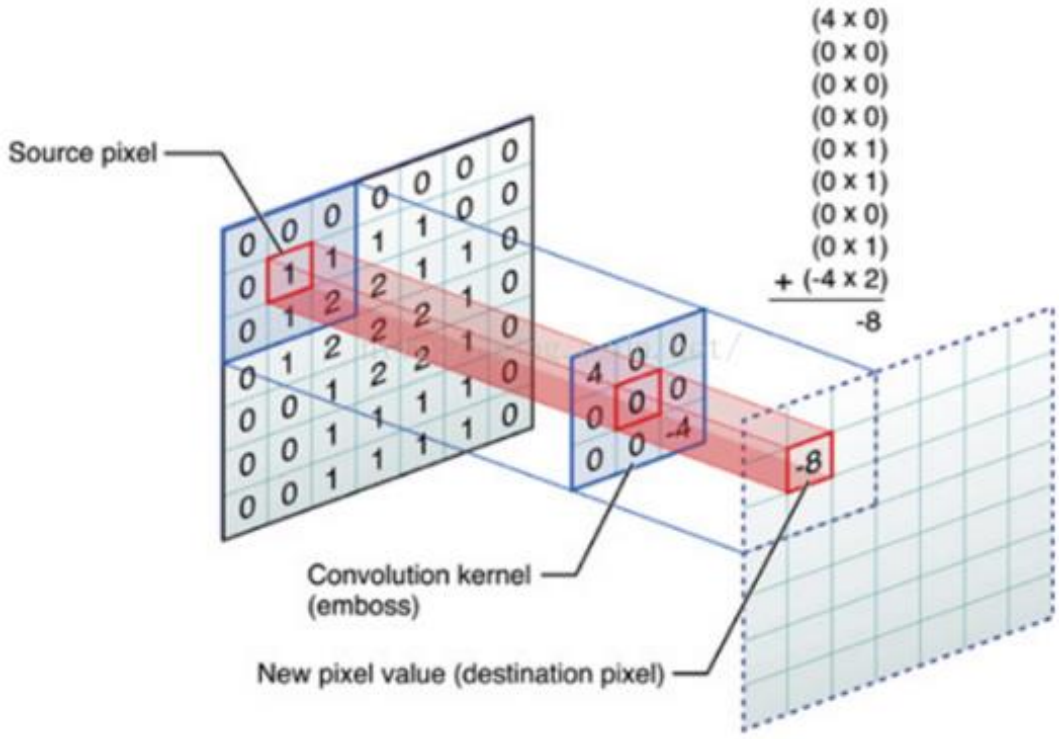

图 2-7 卷积操作示意图[15]

图 2-7 左侧为原始输入数据，中间为滤波矩阵，右侧为输出的新的二维数据。中间滤波矩阵与数据窗口做内积，即对应位置上数字先相乘再相加。数据窗口的大小和滤波矩阵的大小是一致的。对应的数据窗口是卷积神经网络中的一个重要概念——感受野（Reception Field）。在卷积神经网络中，滤波矩阵对感受野中的数据进行卷积计算，然后感受野不断平移滑动，直到计算完图像所有数据。这个过程涉及到如下几个参数：

（1） 深度（Depth）：滤波器的个数，决定输出几层特征地图

（2） 步长（Stride）：感受野滑动时跳过几个数据

（3） 填充值（Zero-padding）：在输入图像或者特征地图边缘填充若干圈 0，使得从图像初始位置可以刚好滑到图像末尾位置

与传统神经网络一致，对卷积操作的输出用一个激活函数来转换。

卷积层的具体计算公式如下：

$$u_j^l = \sum_{i \in M_j} x_i^l \otimes k_{ij}^l + b_j^l \qquad \text{（公式 2-2）}$$

$$x_j^{l+1} = y_j^l = f\left(u_j^l\right) \qquad \text{（公式 2-3）}$$

式中

u_j^l　第 l 层的第 j 个特征地图的卷积输出

x_i^l　第 l 层的第 i 个输入特征地图

k_{ij}^l　连接第 l 层的第 i 个输入特征地图和第 j 个特征地图的卷积核

b_j^l　第 l 层的第 j 个特征地图的偏置

$f()$　激活函数

y_j^l　第 l 层的第 j 个输出特征地图，也是第 l+1 层的第 j 个输入特征地图

卷积具体计算过程示意图如图 2-8，图中最左侧为第 l 层特征地图，该层有 3 个特征地图，i 的范围即为 1 至 3，每个卷积核的层数应和输入特征地图个数一致，即为 3 层。这里有两个卷积核，j 的范围为 1 至 2，第 l 层特征地图和 2 个卷积核做卷积运算的结果加上偏置，即得到 l 层的卷积输出u^l，卷积输出u^l经过激活函数后堆叠起来，即得到第 l 层输出的 j 个特征地图y^l。

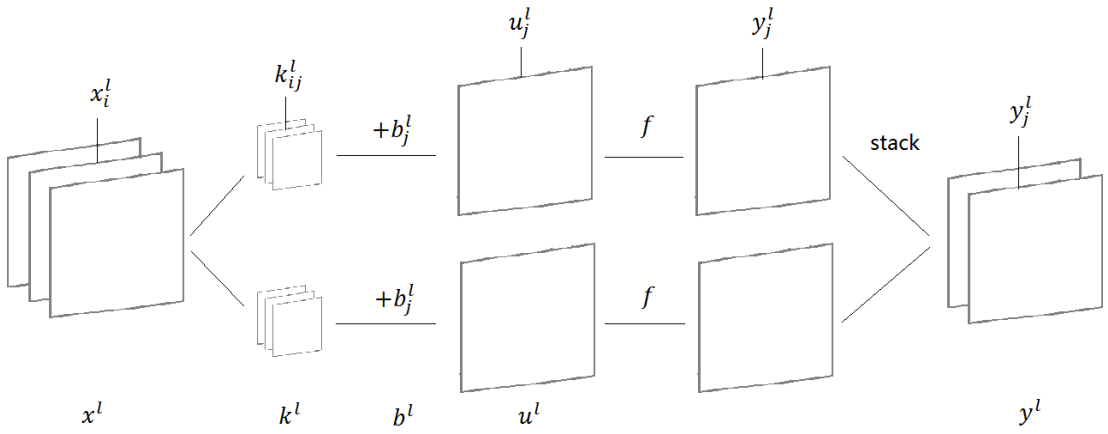

图 2-8 卷积具体计算过程示意图

下采样层的主要作用是对图片进行降维。常用的下采样操作是池化。池化层的输出结果是一个$K_x^l * K_y^l$的不重叠矩形区域的最大激活值或者平均激活值，对应叫做最大池化和平均池化。最大池化，就是取矩形区域中的最大值来代表这个区域；平均池化，就是取矩形区域中的平均值来代表这个区域。图 2-9 为最大池化和平均池化的示意图：

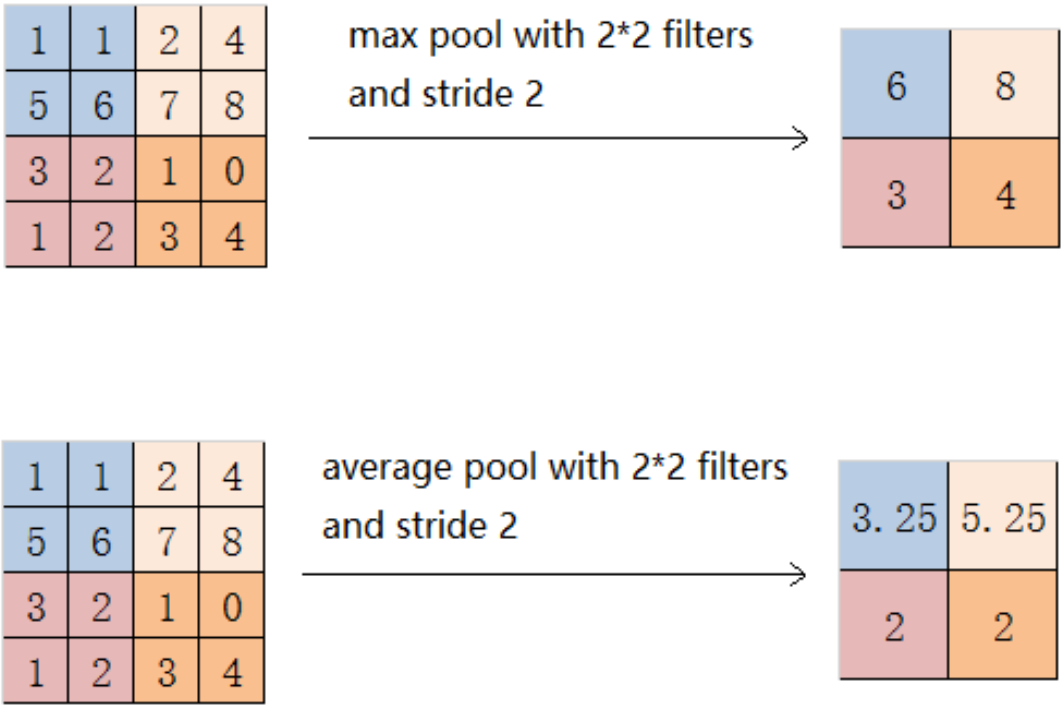

图 2-9 最大池化与平均池化示意图

一系列交替的卷积层和下采样层之后，将特征地图光栅化为特征向量，输入全连接的多层感知机进行分类等操作。

3、主要特点

卷积神经网络的主要特点如下：

（1） 局部感受野

局部感受野的概念来源于生物学中的视觉系统结构，视觉皮层的神经元并不是全局接收信息的，而是局部接收信息。传统的神经网络，隐含层的每一个神经元都与前一层所有神经元相连，如图 2-10 左侧。如果前一层是 1000*1000 的特征地图，隐含层有一百万个神经元，那之间的连接参数一共有 10^{12} 个。实际上，图像有空间位置关系，这种位置关系是局部的，人的视觉神经元也是局部接受信息的。卷积神经网络因此受启发，将原来全连接的方式改成局部连接，如图 2-10 右侧所示。每个隐含层的神经元，只与前一层局部感受野中的神经元相连，这个感受野大小为 10*10。这种连接方式一共才需要学习 10^8 个参数，比原来全连接方式需要学习的参数少了一万倍。

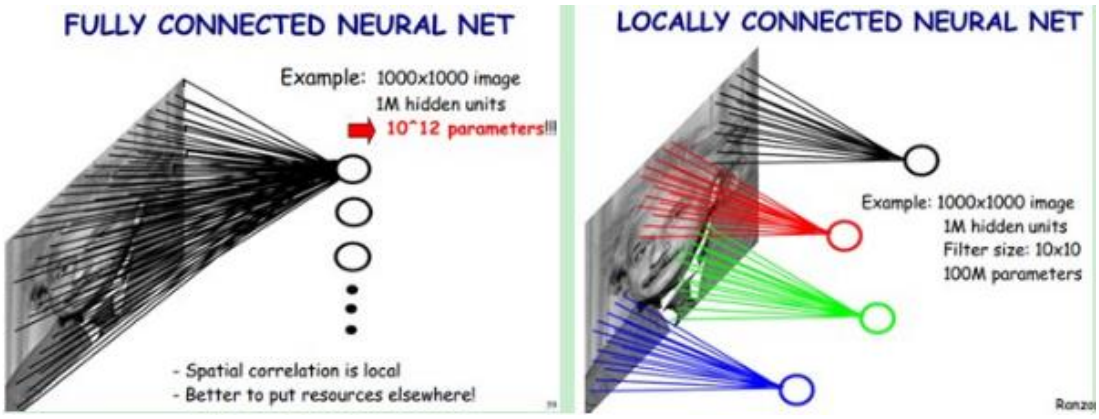

图 2-10 全连接与局部连接[16]

（2） 权值共享

图像不同区域可能具有相同的特征，比如垂直方向的边缘信息。这意味着在图像不同区域如果具有相同的特征，提取该特征的方法是一样的，这就是权值共享概念的来源。卷积网络中，需要提取图像中的某一种特征，就用固定的卷积核（固定的权值矩阵）在图像上滑动进行卷积计算即可。这样如果卷积核的大小为10*10，那么学习一种特征只需要 100 个参数，学习 1000 种特征也只需要十万个参数。局部感受野和权值共享大大减少了参数数量，使模型更简单，易于训练。

（3） 池化

经过卷积操作得到的特征维度还是很大。卷积神经网络利用池化操作，将图像区域的统计特性作为整个区域的特征，这样不仅减少了为特征的维度，还能防止过拟合。

2.3 YOLOv2 目标检测器

1、简介

YOLOv2目标检测器是由Joseph Chet Redmon等人提出的，它在YOLO目标检测器基础上做了一系列改进，在实时检测情况下，提高了检测的精度。YOLOv2可以接受不同的输入分辨率。表2-1是YOLOv2与其他目标检测框架的比较。Fast R-CNN，Faster R-CNN，SSD都是先进的目标检测框架。可以看到，低分辨率的情况下，YOLOv2是一个廉价，但相当准确的检测器。在高分辨率下，YOLOv2是最先进的检测器 [1]。

表2-1 PASCAL VOC 2007数据集上各目标检测框架时间与精度的比较[1]

检测框架	训练集	mAP	FPS
Fast R-CNN	2007+2012	70.0	0.5
Faster R-CNN VGG-16	2007+2012	73.2	7
Faster R-CNN ResNet	2007+2012	76.4	5
YOLO	2007+2012	63.4	45
SSD300	2007+2012	74.3	46
SSD500	2007+2012	76.8	19
YOLOv2 288*288	2007+2012	69.0	91
YOLOv2 352*352	2007+2012	73.7	81
YOLOv2 416*416	2007+2012	76.8	67
YOLOv2 480*480	2007+2012	77.8	59
YOLOv2 544*544	2007+2012	78.6	40

2、网络结构

YOLOv2和YOLO目标检测器的核心思想，都是将目标检测问题采用回归的方式解决。YOLOv2结构十分简单，它将整张图作为网络输入，经过若干卷积层和池化层，直接在输出层回归出边界框（Bounding Box, BBox）的位置和边界框所属类别。

YOLOv2 将输入图像分为 S*S 个网格，每个网格单元只负责检测中心落在该网格单元的物体。每个网格单元预测 B 个边界框以及这些边界框的置信分数（Confidence Score）。这个置信分数反映了这个模型对于这个网格单元中是否含有物体的预测，以及是这个物体的可能性是多少。这个置信分数被定义为：$\Pr(Object) * IOU_{pred}^{truth}$。如果这个网格中不含有任何物体，则这个置信分数为 0；否则这个置信分数等于预测的边界框（Predict Box）与真实边界框（Ground Truth）之间的 IOU（Intersection Over Union）。每个边界框除了要预测 4 个坐标值和 1 个置信分数，还要预测 C 个条件类别概率（Conditional Class Probability）：$\Pr(Class_i|Object)$。这个条件类别概率表示，该网格单元有物体时，该物体属于第 i 个类别时的条件概率。在测试时，每个边界框的置信分数和每个条件类别概

率相乘，就得到每个边界框属于特定类别的置信分数：

$$\Pr(Class_i|Object) * Pr(Object) * IOU_{pred}^{truth}$$

$$= Pr(Class_i) * IOU_{pred}^{truth} \quad \text{（公式 2-4）}$$

然后设置阈值，过滤掉得分低的边界框，对保留的边界框进行非极大抑制（Non-maximum suppression, NMS），就得到最终的检测结果。

将YOLOv2用于PASCAL VOC数据集时，使用S=13，B=5，C=20，即将图片分成7*7的网格，每个网格单元预测5个边界框，每个边界框除了预测4个坐标值和1个置信分数，还要预测20个条件类概率，所以每个网格单元总共需要预测B*（4+1+C）=125个值。图2-11为用于PASCAL VOC数据集的YOLOv2网络结构简易图。

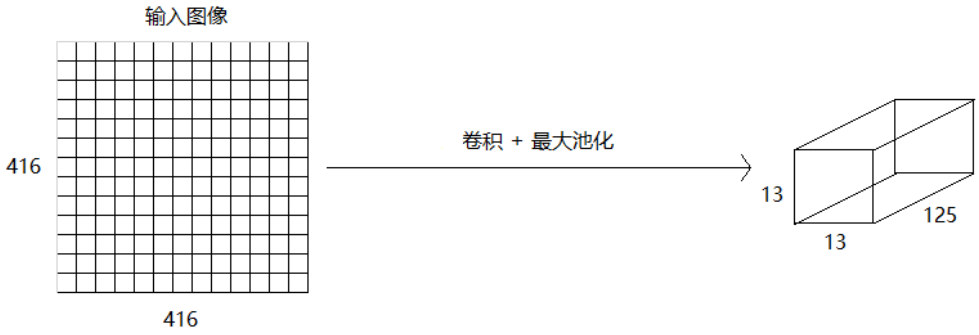

图2-11 用于PASCAL VOC 数据集的YOLOv2网络结构简易图

本文接下来提到的YOLOv2网络，没有特殊说明的话，均指用于PASCAL VOC数据集的YOLOv2网络。图2-12为每个网格单元预测值的示意图。

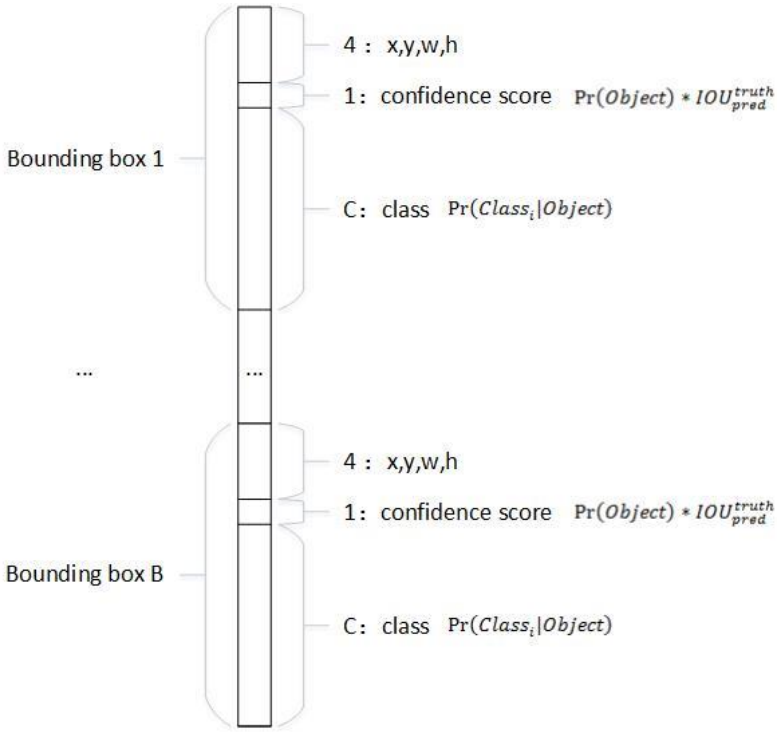

图2-12 每个网格单元预测值示意图

3、具体细节

损失函数（Loss Function）使用总均方误差（Sum-squared Error），便于优化，但直接使用会有问题。对于 PASCAL VOC 数据集来说，一个网格单元需要预测 B*4=5*4=20 维坐标和 B*C=5*20=100 维条件类别概率，把 20 维的坐标误差与 100 维类别误差视为同等重要不太合理。而且如果一个网格单元不含物体（一张图片中这种网格单元有很多），那么这些网格单元的边界框的置信分数为 0，这会导致网络不稳定甚至发散。所以在训练期间，优化下列多部分损失函数：

$$\lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B}1_{ij}^{obj}[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2]$$
$$+\lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right]$$
$$+\sum_{i=0}^{s^2}\sum_{j=0}^{B}1_{ij}^{obj}(c_i-\hat{c}_i)^2+\lambda_{noobj}\sum_{i=0}^{s^2}\sum_{j=0}^{B}1_{ij}^{noobj}(c_i-\hat{c}_i)^2$$
$$+\sum_{i=0}^{s^2}1_{i}^{obj}\sum_{c\in class_i}\left(P_i(c)-\hat{P}_i(c)\right)^2 \qquad \text{（公式 2-5）}$$

其中，

λ_{coord}：坐标预测损失的权重

λ_{noobj}：没有物体的边界框的置信分数损失的权重

1_i^{obj}：网格单元 i 是否含有物体（是否有物体的中心落在该网格中）

1_{ij}^{obj}：网格单元 i 的第 j 个边界框预测是否对这个物体负责

预测的边界框对与它IOU最大的真实边界框所表示的物体负责。公式2-5具体含义说明见表2-2：

表2-2 公式2-5具体说明

公式组成部分	说明
第一、二项	坐标预测损失，利用λ_{coord}来增加或减小这部分损失在总损失中的权重。由于小物体对坐标预测更敏感，所以将边界框的宽高取平方根代替原来的宽高，以减弱这个问题。
第三项	含物体边界框的置信分数预测损失
第四项	不含物体边界框的置信分数预测损失，利用λ_{noobj}来增加或减小这部分损失在总损失中的权重
第五项	类别预测损失

这个损失函数中，有物体的网格才对类别预测损失进行惩罚。只有当某个边界框预测器对某个真实边界框负责时，才会对边界框的坐标预测损失进行惩罚，而对哪个真实边界框负责就看它的预测值和真实边界框的IOU是不是那个网格单元中所有的真实边界框中最大的。

为了防止过拟合，YOLOv2在每层卷积层后使用批量归一化（Batch Normalization）。

YOLOv2在ImageNet上以224*224的分辨率预训练后，把整个网络在448*448分辨率上进行微调，使得卷积核能适应较大的分辨率输入。然后再修改网络用于检测，再进行微调。

YOLOv2借鉴了Faster R-CNN中的Anchor机制[10]来预测边界框。它在训练数

据集上采用k-means聚类，来选择比较好的Anchor Box个数和对应的尺寸。经过在PASCAL VOC数据集上实验，作者发现随着聚类个数增加，召回率也在增加，但是复杂度也在增加。权衡之后，选择的聚类个数为5.

使用Anchor机制会让模型不稳定，特别是最开始的几次迭代，大多数不稳定因素来自于预测边界框坐标的时候。YOLOv2预测相对于网格单元的位置坐标来解决这个问题。在PASCAL VOC数据集上，网络在最后一层输出特征图的每个网格单元上预测5个边界框，每个边界框预测t_x, t_y, t_w, t_h, t_o 5个坐标，关系如图2-13：

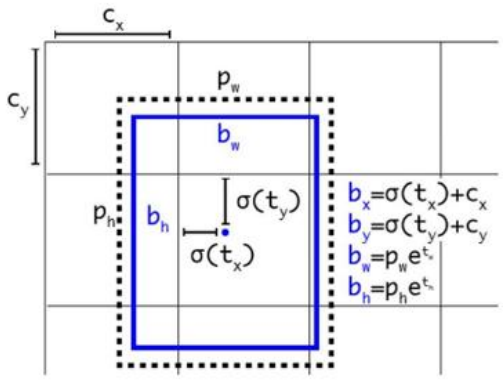

图2-13 预测值与先验Anchor之间的关系图[1]

如果一个网格单元相对于图像的左上角偏移量为$(c_x，c_y)$，并且先验边界框（Anchor）的宽度和高度为$p_w，p_h$，则预测边界框的坐标和置信分数应为：

$$b_x = \sigma(t_x) + c_x \quad （公式2-6）$$

$$b_y = \sigma(t_y) + c_y \quad （公式2-7）$$

$$b_w = p_w e^{t_w} \quad （公式2-8）$$

$$b_h = p_h e^{t_h} \quad （公式2-9）$$

$$Pr(object) * IOU(b，object) = \sigma(t_o) \quad （公式2-10）$$

在检测时，作者让图像输入分辨率为 416*416，以产生一个中心的网格单元。因为大物体一般出现在图像的中间，可以只用一个网格单元检测这个大物体。如果为偶数，则要用中间的 4 个网格单元进行预测。YOLOv2 网络将输入图像下采样了 32 倍，最后输出的特征图为 13*13。YOLOv2 通过添加一个转移层

（Passthrough Layer），连接深层和浅层的特征地图，以获得多尺度的适应性。

在训练期间，作者在几次迭代后就会调整网络。每经过 10 轮训练（10 个 epoch），就会随机选择新的图片尺寸，调整网络的大小，继续训练。可选网络尺寸为降采样参数 32 的倍数，最小为 320，最大为 608。训练时调整网络可以使网络适应不同输入分辨率，低分辨率下速度快，高分辨率下精度高。

网络先在 ImageNet 1000 分类数据集上训练，采用随机梯度下降和标准的数据增强方法。先在 224*224 输入图像上训练，然后在 448*448 输入图像上微调。之后修改网络用于检测，采用类似 YOLOv1 和 SSD 的数据增强方法。网络详细结构如图 2-14 所示，表 2-3 是图 2-14 的符号说明。

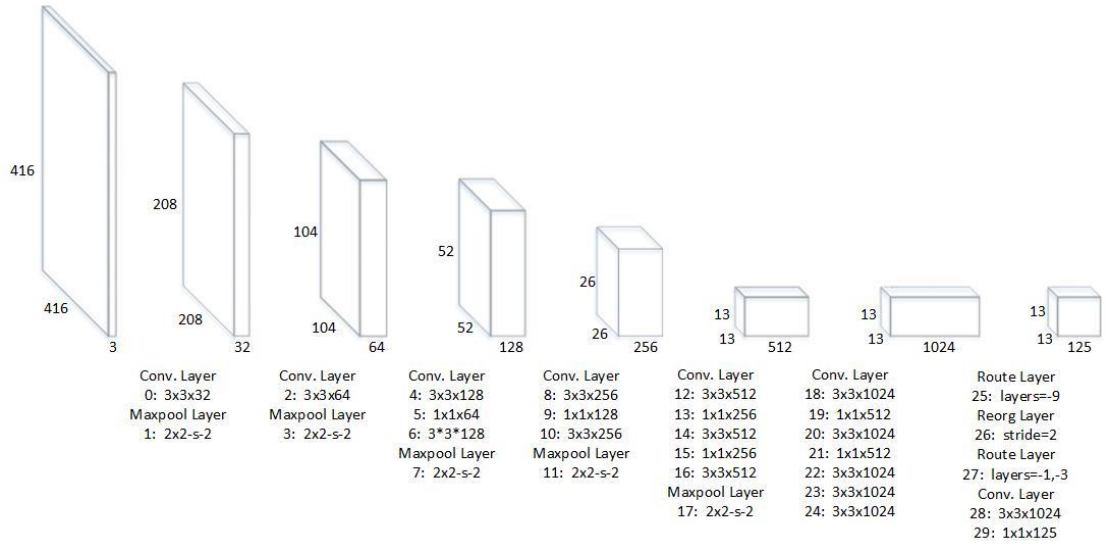

图2-14 用于PASCAL VOC数据集的YOLOv2网络结构

表2-3 图2-14符号说明

符号例子	说明
Conv. Layer 0: 3x3x32	第 0 层为卷积层，使用卷积核大小为 3*3，个数为 32，默认步长 1，默认填充值 1
Maxpool Layer 1: 2x2-s-2	第 1 层为最大池化层，使用卷积核大小为 2*2，步长为 2
Route Layer 25： layers=-9	定位到倒数第 9 层，即 16 层的输出特征图

表 2-3 图 2-14 符号说明（续）

符号例子	说明
Reorg Layer 26： stride=2	将特征图水平每两个像素取一个，垂直每两个像素取一个，原本一个特征图变为 4 个叠加起来
Route Layer 27： layers=-1,-3	将倒数第 1 层和倒数第 3 层输出的特征图按不同通道叠加起来。图中的两个 Route Layer 和一个 Reorg Layer 起到 Passthrough Layer 的作用

2.4 行人检测数据库

行人检测超越一般目标检测，受到了广泛的关注。为了更好的训练模型，且更公平进行各种模型之间的比较，行人检测领域也有专门的行人数据库，下列是本文涉及到的行人数据库的基本介绍。

1、ETH行人数据库

ETH数据库中的视频采用车载摄像头拍摄，分辨率为640*480，帧率为13至14 FPS，给出了行人标注信息[17]。

2、TUD-Brussels行人数据库

TUD-Brussels行人数据库的测试集采用手持摄像机和车载摄像机拍摄，有508对图像，分辨率为640x480，共有1326个行人[17]。

3、Caltech行人数据库

该数据库是目前规模较大的行人数据库，在城市环境中通过常规交通行驶的车辆拍摄，分辨率为640x480，30帧/秒，标注详细的行人信息。Caltech总共有10个数据集（set00-set10），每个数据集大约1GB，具有6到13个1分钟长的seq文件。Caltech中的每个Ground truth都被分配为表2-4中的三个标签之一。

表2-4 Caltech标签说明

标签	说明
Person	单个行人
People	标记个体麻烦或不可能标记的大群体
Person?	行人不明确或容易错分的行人

Caltech中大部分的行人属于中小尺度，见表2-5：

表2-5 Caltech行人尺度说明

尺度名称	尺度范围	尺度所占百分比
Near（大尺度）	大于等于 80 像素	16%
Medium（中等尺度）	大于 30 像素，小于 80 像素	69%
Far（小尺度）	小于等于 30 像素	15%

文献[18]给出了Caltech数据集的4个使用场景，见表2-6：

表2-6 Caltech数据集使用说明

阶段	名称	详细说明
开发阶段	场景 ext0	使用额外的数据训练，在 Caltech 数据集 set00-set05 上测试
模型确定后	场景 ext1	使用额外的数据训练，在 Caltech 数据集 set06-set10 上测试
开发阶段	场景 cal0	在 Caltech 数据集 set00-set05 上进行 6 折交叉验证，每次使用 5 个 set 进行训练，剩下的 1 个 set 进行测试，总的结果为 set00-set05 上合并后的结果
模型确定后	场景 cal1	在 Caltech 数据集 set00-set05 上训练，在 set06-set10 上测试

Caltech数据集以seq/vbb格式发布完整的图像视频/注释，并且将主流的几个行人数据库，如INRIA、ETH、TUD-Brussels和Daimler数据库也转化成了seq/vbb格式。Caltech数据集官网还提供了使用seq/vbb数据和评估算法的matlab例程，方便进行比较。

2.5 本章小结

本章主要对本文实验工作所涉及的相关知识进行介绍，包括神经网络、卷积神经网络、YOLOv2 目标检测器以及本文用到的三个行人数据库。在了解以上技术和知识情况下进行行人检测器的研究。

第三章 基于 YOLOv2 模型的行人检测

本章主要介绍了基于 YOLOv2 模型的行人检测研究开始前的准备工作。先根据 YOLOv2 目标检测器，罗列出修改成行人检测器的几个可能方向，然后准备数据集，将数据集转化成 YOLOv2 网络的输入格式，并组合成三折交叉验证。最后确定评估格式和实验评估所用的指标。

3.1 修改方向

本论文的目标是将用于PASCAL VOC数据集的YOLOv2目标检测器修改成行人检测器，然后逐步提高检测精度。大致有如下几个修改方向和选择。

1、类别数

行人检测器只需要检测出行人一个类别。在测试时，YOLOv2对每个检测出的边界框预测第2.3节提到的置信分数和条件类别概率，最后边界框的得分是这两个值的乘积，分数大于某个阈值，则判定为行人。类别有表3-1两种设置方式，一种是只设置行人类别，学习行人的特征，另一种是设置行人和背景类别，学习行人特征和背景特征。第二种设置两个类别，常用于以分类方式解决检测问题的方法中。

表3-1 可选类别设置方式

序号	类别个数	类别名称	样本	说明
1	1 个	行人	只有行人样本	只学习行人特征
2	2 个	行人；背景	行人和背景样本	学习行人和背景特征

2、样本

Caltech数据集都标注了详细的行人注释信息，包含行人、人群、遮挡、是否忽略等信息，这使得行人样本也有选择的余地。行人样本的选择如表3-2所示，第一种选择参考文献[2]，0.41是文献[18]经过统计得到的Caltech数据集上的行人标准宽高比，将行人真实边界框和检测结果调整成标准的行人宽高比，有利于提高检测精度。

表3-2 行人样本选择

序号	行人真实边界框条件	调整宽高比为 0.41
1	≥50 像素高，60%可见，标记为“Person”，ignore=0	是
2	标记为“Person”，ignore=0	否

背景样本的选择有表3-3两种，一种是随机在图片中产生背景样本，这种方法比较简单，但是不容易得到容易误判成行人的背景样本。另一种使用文献[2]中训练RPN网络得到的背景样本，这里的背景样本就包含比较容易误判成行人的背景样本。

表3-3 背景样本选择

序号	产生方式
1	每张图片随机产生一些背景样本
2	用文献[2]中训练 RPN 时得到的背景样本

3、数据集

本实验使用三个数据集：Caltech行人数据集（USA版本，只使用set00-set05），ETH行人数据集，Tud-Brussels行人数据集。对于ETH和Tud-Brussels数据集，使用的是Caltech官网发布的seq/vbb版本。由于ETH和Tud-Brussels数据集比较小，其中的视频取每帧图像作为训练或验证的图像。Caltech数据集比较大，所以设置了两种使用方式，见表3-4：

表3-4 Caltech数据集取图像帧策略

序号	Caltech 行人数据集取图像帧策略
1	每 3 帧取一帧图像
2	取每帧图像

有些图像中是没有符合条件的行人真实边界框，所以图片选用还有如表3-5所示的两种选择：

表3-5 图像选用策略

序号	图像选用策略
1	只使用包含符合条件行人真实边界框的图像
2	使用所有图像

4、Anchor

用于PASCAL VOC数据集的YOLOv2目标检测器使用的默认Anchor个数和尺寸大小，是在训练集上使用k-means聚类得到的，最后得出的结论是聚类中心，即Anchor的个数为5。聚类中心个数越多，召回率越高，但是模型比较复杂。文献[2]中也采用了Anchor机制，使用的Anchor个数为9，论文中提到使用的默认Anchor宽高比为0.41，从40像素高开始，缩放步长为1.3倍，代码实现却是从62.4像素高开始的。所以Anchor个数和尺寸选择有如表3-6所示的几种：

表3-6 Anchor使用选择

序号	Anchor 个数	Anchor 尺寸
1	9	从 40 像素高开始，缩放步长为 1.3 倍
2	9	从 62.4 像素高开始，缩放步长为 1.3 倍
3	9	从训练集上聚类产生 Anchor 尺寸
4	5	从 1 中取第 1,3,5,7,9 个 Anchor 尺寸
5	5	从 2 中取第 1,3,5,7,9 个 Anchor 尺寸
6	5	从 1 中取第 3,4,5,6,7,个 Anchor 尺寸
7	5	从 2 中取第 3,4,5,6,7,个 Anchor 尺寸
8	5	从训练集上聚类产生 Anchor 尺寸

5、学习率与最大迭代次数

学习率和最大迭代次数的选择需要经验，借鉴了用于PASCAL VOC数据集的YOLOv2目标检测器的学习率和最大迭代次数设置，设置了如表3-7的几种。阶段和缩放因子表示，迭代到指定次数后，学习率乘以对应的缩放因子，以此来控制训练过程中学习率的大小。

表3-7 学习率与最大迭代次数设置

序号	初始学习率（learning_rate）	最大迭代次数（max_batches）	阶段（steps）	缩放因子（scales）
1	0.0001	100000	100, 2000	10, .1
2	0.001	100000	50000	.1
3	0.0001	35000	1000, 20000	10, .1

6、损失函数权重

第2.3节提到YOLOv2目标检测器的损失函数，使用的是平方损失，并利用权重因子来控制位置预测损失与类别预测损失之间的权重。权重因子和预测的类别个数以及Anchor个数有关，决定最后的优化目标。

7、转移层

本文使用的三个数据集，行人尺寸都比较小，转移层连接多个更低层但分辨率较大的特征地图，有可能有利于小尺寸行人的检测。

8、网络大小

用于PASCAL VOC数据集的YOLOv2目标检测器网络的输入大小为416*416分辨率。PASCAL VOC数据集中图片大小是不一致的，416*416分辨率属于中偏上大小。第2.3.3节提到YOLOv2目标检测器在训练过程中会自动调整网络大小，以适应不同网络的输入，可选的网络大小值在320*320分辨率到608*608分辨率之间，为32的倍数。本文使用到的图片大小均为640*480分辨率，调大网络输入的分辨率有可能提高检测的精度，特别是尺度较小的行人。于是网络大小和可调整范围有如表3-8所示几种设置：

表3-8 网络输入尺寸与调整范围

序号	网络输入尺寸	网络调整范围
1	416*416	32 的 13 至 19 倍
2	416*416	32 的 15 至 20 倍
3	640*640	32 的 15 至 24 倍

3.2 数据准备

本文实验采用Caltech官网上公开发布的seq/vbb格式的Caltech行人数据集，ETH行人数据集和TuD-Brussels行人数据集作为实验数据。本节主要描述了将seq/vbb格式的数据转换成YOLOv2目标检测器使用的数据格式的过程以及所使用的例程。

转换步骤如下：

1、从Caltech官网下载seq/vbb格式的Caltech行人数据集（set00-set05），ETH行人数据集和TuD-Brussels行人数据集，图3-1，3-2，3-3分别为三个数据集的组成：

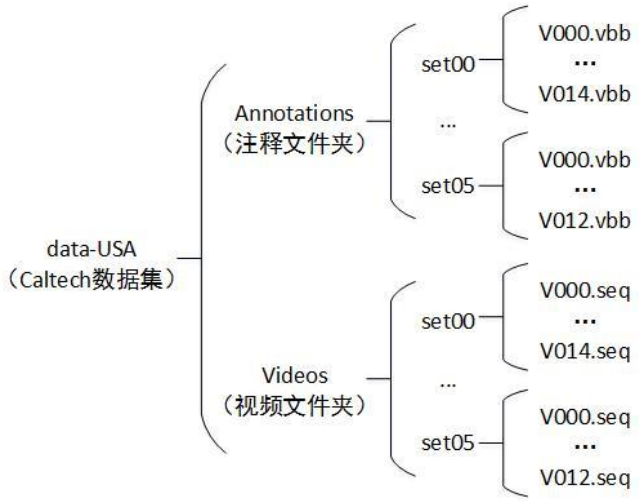

图3-1 Caltech数据集（USA版本）组成

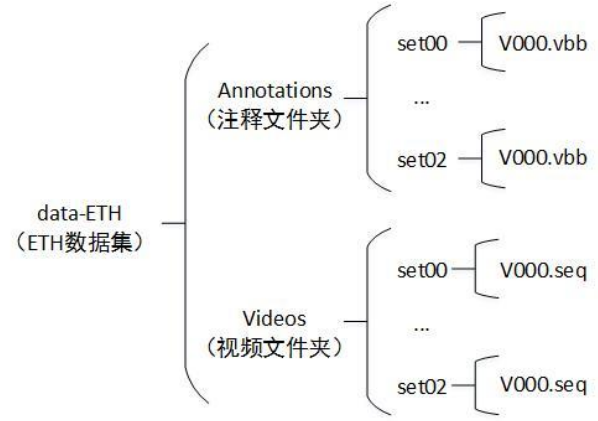

图3-2 ETH数据集组成

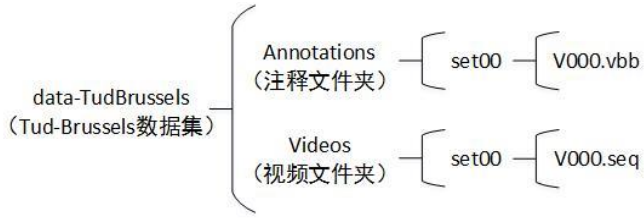

图3-3 TUD-Brussels数据集组成

2、利用dbExtract.m例程（Caltech官网提供的），按间隔帧数将seq/vbb格式数据转换成image/txt格式，每个txt文件每行表示一个物体，包含标签，左上角坐标，物体宽高，是否遮挡等信息，包含所有的标注信息。

3、利用get_labels.m例程，过滤掉第二步得到的txt文件中不需要使用的真实边界框（本实验只保留了至少50像素高，60%可见，标记为“person”类别，ignore=0的真实边界框），保持高度不变，调整宽度，将真实边界框的宽高比调整成0.41，然后转成YOLOv2目标检测器的格式。即，每个txt文件每行表示一个物体，包含类别号，相对于图片宽高的中心位置坐标，相对于图片宽高的物体宽高。

get_labels.m例程如下：

```
function get_labels()
    get_labels_for_yolov2('train','caltech');
    get_labels_for_yolov2('train','eth');
    get_labels_for_yolov2('train','tudbrussels');
end

%过滤真实边界框，转成 YOLOv2 需要的格式
function get_labels_for_yolov2(stage,datasetName)
imdb.name=stage;
imdb.width = 640;imdb.height = 480;%设置原图宽高
%找到第二步得到的 txt 文件路径（anno_path），和产生新的文件的存放路径（label_path）
if(strcmp(datasetName,'caltech'))
    anno_path = ['./datasets/caltech/'    imdb.name '/annotations'];
    label_path = strcat( ['./datasets/caltech/'    imdb.name] , '/labels' ) ;
elseif(strcmp(datasetName,'eth'))
    anno_path = './datasets/eth/annotations';
    label_path = './datasets/eth/labels';
elseif(strcmp(datasetName,'tudbrussels'))
    anno_path = './datasets/tudbrussels/annotations';
    label_path = './datasets/tudbrussels/labels';
end
```

```
if ~exist( label_path , 'dir' )
   mkdir( label_path );
end

%设置需要保留的真实边界框的属性
%只保留标记为 person 的边界框，标记为 people 的边界框记为 ignore，将边界框调整成宽高比为 0.41
pLoad={'lbls',{'person'},'ilbls',{'people'},'squarify',{3,.41}};
pLoad = [pLoad 'hRng',[50 inf], 'vRng',[0.6 1] ];%真实边界框大于 50 像素高，至少 60%可见
files=bbGt('getFiles',{anno_path});

   for i = 1:length(files)
      %得到过滤后的真实边界框，gts 为[nx5]的数组，每行为 1 个 GT，[左上角 x 坐标，左上角 y 坐标，GT 宽，GT 高，ignore]，ignore 表示是否忽略
      [~,gts]=bbGt('bbLoad',files{i},pLoad);
      fName=files{i};
      fName=strrep(fName,'annotations','labels') ;
      %把 GT 左上角坐标改为相对于原图大小的中心位置坐标
      x_ctr_relative = ( gts(:,1) + gts(:,3) / 2 ) / imdb.width ;
      y_ctr_relative = ( gts(:,2) + gts(:,4) / 2 ) / imdb.height ;
      %把 GT 宽高改为相对于原图大小的宽高
      w_relative = gts(:,3) / imdb.width;
      h_relative = gts(:,4) / imdb.height;
      %将结果保存到文件
      fid=fopen(fName,'w'); assert(fid>0);
      if( size(gts,1) ~= 0 )
            for i=1:size(gts,1)
                  %标记为 ignore 的 GT 不写入文件
                  if(gts(i,5) == 0)
                        fprintf(fid,['0 ' repmat('%f ',1,4) '\n'], x_ctr_relative(i) , y_ctr_relative(i), w_relative(i), h_relative(i)); %  常数 1 表示是行人真实边界框
                  end
            end
      end
      fclose(fid);
   end
end
```

4、利用getImagePath.m例程得到Caltech数据集所有图片文件的路径，利用getImagePath_eth_tud.m例程得到ETH和TuD-Brussels数据集所有图片文件路径。getImagePath.m例程如下：

```
function getImagePath()
    imagePath( 'train' , 'set' );
    imagePath( 'test' , 'set' );
    for i=0 : 5
        imagePath( 'train' , ['set0' num2str(i)] );
    end
end

function imagePath( image_set , set )
% get the image path for yolov2 to train or test
% INPUTS
% image_set = 'test' or 'train'
% set =
% 'set'     :   if image_set='test' means get set06-set10 images path else get set00-set05 images
path
% 'set0x' :get set0x images path
    %得到所有图片文件路径
    root_dir=[pwd '/datasets/caltech'];
    imdb.name = image_set;
    imdb.extension = '.jpg';
    imdb.image_dir = fullfile(root_dir, image_set, 'images');
    imgs = dir(fullfile(imdb.image_dir, [set '*' imdb.extension]));
    imdb.image_ids = cell(length(imgs), 1);
    for i = 1:length(imgs)
       imdb.image_ids{i} = imgs(i).name(1:end-4);
    end
    imdb.image_at = @(i) ...
    fullfile(imdb.image_dir, [imdb.image_ids{i} imdb.extension]);
    %设置文件名
    if strcmp( set , 'set' )
        set = image_set ;
    end
    %将路径写入对应文件
    filename = [root_dir '/' set '.txt'] ;
    fid=fopen( filename ,'wt');
    for i = 1:length(imdb.image_ids)
        fprintf(fid,'%s\n',imdb.image_at(i));
    end
    fclose(fid);
end
```

getImagePath_eth_tud.m例程如下：

```
function getImagePath_eth_tud(datasetName)
    %限制输入
    if ~strcmp(datasetName,'eth') && ~strcmp(datasetName,'tudbrussels')
        error(' datasetName = "eth" or "tudbrussels" ');
    end
    %得到所有图片文件路径
    root_dir=[pwd '/datasets/' datasetName];
    imdb.extension = '.png';
    imdb.image_dir = fullfile(root_dir, 'images');
    imgs = dir(fullfile(imdb.image_dir, [ '*' imdb.extension]));
    imdb.image_ids = cell(length(imgs), 1);
    for i = 1:length(imgs)
      imdb.image_ids{i} = imgs(i).name(1:end-4);
    end
    imdb.image_at = @(i) ...
    fullfile(imdb.image_dir, [imdb.image_ids{i} imdb.extension]);
    %将路径写入文件
    filename = [root_dir '/' datasetName '.txt'] ;
    fid=fopen( filename ,'wt');
    for i = 1:length(imdb.image_ids)
        fprintf(fid,'%s\n',imdb.image_at(i));
    end
    fclose(fid);
end
```

5、利用getHasGTImagePath.m例程，得到Caltech数据集包含保留真实边界框的图片文件路径。利用getHasGTImagePath_eth_tud.m例程，得到ETH数据集和Tud-Brussels数据集包含保留真实边界框的图片文件路径。

getHasGTImagePath.m例程如下：

```
function getHasGTImagePath()
    imagePath( 'train' , 'set' );
    for i=0 : 5
        imagePath( 'train' , ['set0' num2str(i)]);
    end
end

function imagePath( image_set , set )
    %得到路径
    root_dir=[pwd '/datasets/caltech'];
    imdb.name = image_set; %test or train
```

```
    imdb.extension = '.jpg';
    imdb.image_dir = fullfile(root_dir, image_set, 'images'); %得到图片所在目录
    imdb.label_dir = fullfile(root_dir, image_set, 'labels'); %得到注释所在目录
    imgs = dir(fullfile(imdb.image_dir, [set '*' imdb.extension]));
    imdb.image_ids = cell(length(imgs), 1);
    for i = 1:length(imgs)
       imdb.image_ids{i} = imgs(i).name(1:end-4);
    end
    imdb.image_at = @(i) ...
    fullfile(imdb.image_dir, [imdb.image_ids{i} imdb.extension]);
    imdb.label_at = @(i) ...
    fullfile(imdb.label_dir, [imdb.image_ids{i} '.txt']);
    %写入文件
    fid=fopen([root_dir '/GT' set '.txt'],'wt');
    for i = 1:length(imdb.image_ids)
        s=dir(imdb.label_at(i));
        if s.bytes~=0 %只有当 image 中含有符合条件的 Gt 时，才会该图片路径才会被
写入文件
            fprintf(fid,'%s\n',imdb.image_at(i));
        end
    end
    fclose(fid);
end
```

getHasGTImagePath_eth_tud.m例程如下：

```
function getHasGTImagePath_eth_tud(datasetName)

    %限制输入
    if ~strcmp(datasetName,'eth') && ~strcmp(datasetName,'tudbrussels')
        error(' datasetName = "eth" or "tudbrussels" ');
    end
    %得到路径
    root_dir=[pwd '/datasets/' datasetName];
    imdb.extension = '.png';
    imdb.image_dir = fullfile(root_dir,    'images'); %得到图片所在目录
    imdb.label_dir = fullfile(root_dir,    'labels'); %得到注释所在目录
    imgs = dir(fullfile(imdb.image_dir, [ '*' imdb.extension]));
    imdb.image_ids = cell(length(imgs), 1);
    for i = 1:length(imgs)
       imdb.image_ids{i} = imgs(i).name(1:end-4);
    end
```

```
    imdb.image_at = @(i) ...
    fullfile(imdb.image_dir, [imdb.image_ids{i} imdb.extension]);
    imdb.label_at = @(i) ...
    fullfile(imdb.label_dir, [imdb.image_ids{i} '.txt']);
    %写入文件
    fid=fopen([root_dir '/GT' datasetName '.txt'],'wt');
    for i = 1:length(imdb.image_ids)
        s=dir(imdb.label_at(i));
        if s.bytes~=0 %只有当 image 中含有符合条件的 Gt 时，才会该图片路径才会被
写入文件
            fprintf(fid,'%s\n',imdb.image_at(i));
        end
    end
    fclose(fid);
end
```

表3-9为三个数据集图片数量统计信息以及对应文件名称，本文实验中对Caltech数据集采用了两种间隔帧数取图像。

表3-9 Caltech、ETH和Tud-Brussels数据集图片数量统计信息以及对应名称

Caltech 数据集	**视频每 3 帧取一帧**	
	所有图片数量	**包含保留真实边界框的图片数量**
set00	8559（set00.txt）	4105（GTset00.txt）
set01	3619（set01.txt）	1724（GTset01.txt）
set02	7410（set02.txt）	492（GTset02.txt）
set03	7976（set03.txt）	1330（GTset03.txt）
set04	7328（set04.txt）	925（GTset04.txt）
set05	7890（set05.txt）	920（GTset05.txt）
总计	42782（train.txt）	9496（GTset.txt）
Caltech 数据集	**视频取每帧**	
	所有图片数量	**包含保留真实边界框的图片数量**
set00	25693（Fullset00.txt）	12303（FullGTset00.txt）
set01	10864（Fullset01.txt）	5173（FullGTset01.txt）
set02	22239（Fullset02.txt）	1473（FullGTset02.txt）
set03	23944（Fullset03.txt）	3986（FullGTset03.txt）
set04	21995（Fullset04.txt）	2784（FullGTset04.txt）
set05	23684（Fullset05.txt）	2744（FullGTset05.txt）
总计	128419（Fulltrain.txt）	28463（FullGTset.txt）
数据集	**视频取每帧**	
	所有图片数量	**包含保留真实边界框的图片数量**
ETH	1804（eth.txt）	1799（GTeth.txt）
Tud-Brussels	508（tudbrussels.txt）	320(GTtudbrussels.txt)

6、根据包含有效真实边界框的图片数量将所有图片分成三份，留做三折交叉验证。设置了两种三折交叉验证，两种设置的验证集是一样的，只是训练样本一个多，一个少。

当Caltech数据集每3帧取一帧时，三折交叉验证的设置如表3-10：

表3-10 三折交叉验证设置（Caltech数据集每3帧取一帧）

每 3 帧取一帧				
3-fold（train）				**总计**
GTset00				4105（one.txt）
GTset01	GTset02	GTset03	GTtudbrussels	3866（two.txt）
GTset04	GTset05	GTeth		3644（three.txt）
3-fold（Vaild）				**总计**
set00				8559（一）
set01	set02	set03	tudbrussels	19513（二）
set04	set05	eth		17022（三）
训练			**测试**	**名称**
基础		**合并**		
two.txt	three.txt	7510（-one.txt）	8559（一）	第一折交叉验证
one.txt	three.txt	7749（-two.txt）	19513（二）	第二折交叉验证
one.txt	two.txt	7971（-three.txt）	17022（三）	第三折交叉验证

当Caltech数据集取每帧时，三折交叉验证设置如表3-11：

表3-11 三折交叉验证设置（Caltech数据集取每帧）

取每帧				
3-fold（Train）				**总计**
FullGTset00				12303（fullone.txt）
FullGTset01	FullGTset02	FullGTset03	GTtudbrussels	10952（fulltwo.txt）
FullGTset04	FullGTset05	GTeth		7327（fullthree.txt）
3-fold（Vaild）				**总计**
set00				8559（一）
set01	set02	set03	tudbrussels	19513（二）
set04	set05	eth		17022（三）
训练			**测试**	**名称**
基础		**合并**		
fulltwo.txt	fullthree.txt	18279（-fullone.txt）	8559（一）	第一折交叉验证
fullone.txt	fullthree.txt	19630（-fulltwo.txt）	19513（二）	第二折交叉验证
fullone.txt	fulltwo.txt	23255（-fullthree.txt）	17022（三）	第三折交叉验证

7、利用createNotPerson.m例程，给每个图像产生背景样本和符合条件的行人样本，转成YOLOv2目标检测器输入的格式。背景样本是随机产生的。

createNotPerson.m的伪代码如下：

算法 3-1 生成负样本算法

Begin

（1） 得到要产生行人样本图片的路径

（2） 对于每张图片，执行以下步骤

（a）得到该图片符合条件的行人真实边界框

（b）在(10,25]随机生成该图片要产生的背景样本数量num

（c）当 num > 0

① 随机选择一个缩放尺度

② 根据缩放尺度得到对应的背景样本的宽高

③ 在左上角坐标可选范围内（不要让样本超出图像范围），随机得到背景样本左上角的坐标

④ If 是行人样本（生成的边界框与该图片所有行人样本的IOU都小于threshold，则为背景样本）

Continue;

End

⑤ 将合格的背景样本转换成YOLOv2的输入格式

⑥ num=num-1;

（d）合并生成的背景样本和符合条件的行人真实边界框

（e）输出到文件

End

3.3 评估设置

3.3.1 格式转换

测试时，YOLOv2 目标检测器先过滤掉分数低于 0.005 的边界框，然后进行阈值为 0.45 的非极大抑制，将剩余的边界框按照类别输出到文件。文件每行表示一个检测到的边界框，包含测试图片标识，得分，边界框左上角坐标和边界框

右下角坐标。Caltech 官网提供了 matlab 格式的算法评估例程，但要求算法结果整理成统一的格式，如图 3-4：

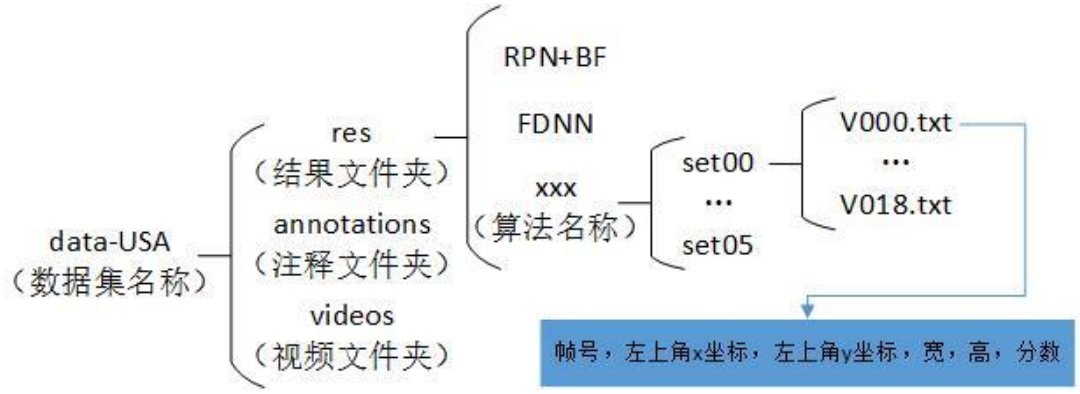

图 3-4 Caltech 评估格式例子

转换例程 prepareForEval_3fold.m 如下：

```
function prepareForEval_3fold()

    %得到需要验证的数据集名称
    basefilename = { 'USA' , 'ETH' ,'TudBrussels' };

    %设置算法名称
    methodname = 'np-code5-lr3-5000' ;

    %开始转换
    for i=1 : length(basefilename)
        prepareForEval_dataset(basefilename{1,i} , methodname);
    end

end

function prepareForEval_dataset(dataset,methodname)

    %加载检测结果
    filepath =[ '/home/vision/darknet/results/'    methodname '/' dataset '.txt'];
    [image_ids,prob,left_x,left_y,right_x,right_y]=textread(filepath ,'%s %f %f %f %f %f',-1);
    aboxes = [ left_x    left_y    right_x    right_y prob];

    %开始转换过程
    fprintf('Preparing the results for evaluation ...');
    result_dir = [ pwd '/external/code3.2.1/code3.2.1/data-' dataset '/res/' methodname];
```

```
    %产生空白结果文件
    createEmptyTestResultFiles(result_dir,dataset);

    %转换格式，由[左上角坐标，右下角坐标]转成[左上角坐标，宽，高]
    res_boxes = aboxes ;
    res_boxes( : ,3) = res_boxes( : ,3) - res_boxes( : ,1) ;
    res_boxes( : ,4) = res_boxes( : ,4) - res_boxes( : ,2) ;

    %写入到对应文件
    for i = 1 : size(res_boxes , 1)
        sstr = strsplit(image_ids{i,1}, '_');
        fid = fopen(fullfile(result_dir, sstr{1}, [sstr{2} '.txt']), 'a');
        fprintf(fid, '%d,%f,%f,%f,%f,%f\n', str2double(sstr{3}(2:end))+1, res_boxes(i , :));
        fclose(fid);
    end
    fprintf('Done.');
end

%创建空白结果文件
function createEmptyTestResultFiles(result_dir,dataset)

    %得到每个数据集 set 和 videos 的信息
    switch dataset
      case 'USA' % Caltech Pedestrian Datasets (all)
        setIds=0:5; subdir='USA'; skip=3; ext='jpg';
        vidIds={0:14 0:5 0:11 0:12 0:11 0:12};
      case 'TudBrussels' % TUD-Brussels dataset
        setIds=0; subdir='TudBrussels'; skip=1; ext='png';
        vidIds={0};
      case 'ETH' % ETH dataset
        setIds=0:2; subdir='ETH'; skip=1; ext='png';
        vidIds={0 0 0};
      otherwise,
        error('unknown dataset : %s',dataset);
    end

    %删除之前的检测结果
    DIRS=dir(result_dir); n=length(DIRS);
    for i=1:n
        if (DIRS(i).isdir && ~strcmp(DIRS(i).name,'.') && ~strcmp(DIRS(i).name,'..') )
            rmdir(fullfile(result_dir ,DIRS(i).name),'s');
        end
    end
```

```
    %创建结果文件夹
    if(~exist(result_dir,'dir'))
        mkdir(result_dir);
    end

    %生成空白文件
    for s=1 : length(setIds)
        setname=sprintf('set%02d',setIds(s));
        set_dir=[result_dir '/' setname];
        if(~exist(set_dir,'dir'))
            mkdir(set_dir);
        end
        for v=1 : length(vidIds{s})
            vidname=sprintf('V%03d',vidIds{s}(v));
            file=[set_dir '/' vidname '.txt'];
            fid = fopen( file ,'a+');
            fclose(fid);
        end
    end
end
```

3.3.2 评估指标

1、平均损失（Avgerage Loss）

本文所有实验采用的都是批量梯度下降（BGD），批大小为 64。每迭代一次，记录一个平均损失（Average Loss），平均损失计算如下：

$$al_t = 0.9 * al_{t-1} + 0.1 * loss_t \quad \text{（公式 3-1）}$$

其中：

al_t 表示第 t 次迭代后得到的平均损失

$loss_t$ 表示第 t 次迭代后的总损失

本文通过记录每间隔 100 次迭代的平均损失值，绘制平均损失值随 epoch 变化情况图，来记录模型训练损失的变化情况。

绘制不同实验设置的平均损失值随 epoch 或迭代次数变化情况的 plotAvgLoss.m 例程如下：

```
showEpoch=true;
rootpath='../log/';
n=1000;
clrs=zeros(n,3);
for i=1:n, clrs(i,:)=max(.3,mod([78 121 42]*(i+1),255)/255); end %初始化可选的颜色
algs = {%选择不同实验设置进行对比（去掉%号即可）
%                   'paper',                        118, clrs(1,:),'-'
%                   'code',                          118, clrs(2,:),'-'
%                   'code-lr2',                     118, clrs(3,:),'-'
%                   '2-code-lr2',                   121, clrs(4,:),'-'
%                   '3-code-lr2',                   125, clrs(5,:),'-'
%                   'full-code-lr2',                286, clrs(6,:),'-'
%                   '2-full-code-lr2',              307, clrs(7,:),'-'
%                   '3-full-code-lr2',              364, clrs(8,:),'-'
%                   'code5-lr3',                    118, clrs(9,:),'-'
%                   '2-code5-lr3',                  121, clrs(10,:),'-'
%                   '3-code5-lr3',                  125, clrs(11,:),'-'
%                   'np-code5-lr3',                 118, clrs(12,:),'-'
%                   'large-code5-lr3',              118, clrs(13,:),'-'
};
algs=cell2struct(algs',{'name','epoch','color','style'});

for i=1:length(algs)%逐个绘制不同实验设置的曲线
    filepath=[rootpath algs(i).name '.txt'];
    [ iter , avgloss ] = textread( filepath ,'%f\t%f' , -1);%读取数据
    if showEpoch
        iter = iter / algs(i).epoch;
    end
    plot(iter,avgloss,'Color',algs(i).color,'LineStyle',algs(i).style,'LineWidth',1,'displayname',s
printf(algs(i).name));%绘制
    hold on
end

%选择横坐标是迭代次数还是轮次
if showEpoch
    xlabel('epoches');
else
    xlabel('iterations');
end

ylabel('avg loss');
grid on ;
legend('show');
```

2、对数平均错误率（Log-average Miss Rate，简称 LAMR）

对数平均错误率在文献 [18]中提出，用来总结检测器的性能。在文献[2],[5]中都使用过，这个值越小越好。同一组实验设置，在特定几次迭代次数时检测一次结果，并利用 LAMR 指标评估在训练集和测试集上的性能。

绘制 LAMR 随 epoch 变化情况图的例程 plotLAMR.m 如下：

```
rootpath='../lamr/';
n=1000;
clrs=zeros(n,3);
for i=1:n, clrs(i,:)=max(.3,mod([78 121 42]*(i+1),255)/255); end %设置可选颜色
algs = {%选择不同实验设置进行对比（去掉%号即可）
%                   'paper',                        118, clrs(1,:),'-','first',7
%                   'code',                          118, clrs(2,:),'-','first',6
%                   'code-lr2',                    118, clrs(3,:),'-','first',10
%                   '2-code-lr2',                  121, clrs(4,:),'-','second',12
%                   '3-code-lr2',                  125, clrs(5,:),'-','third',12
%                   'full-code-lr2',              286, clrs(6,:),'-','first',15
%                   '2-full-code-lr2',           307, clrs(7,:),'-','second',17
%                   '3-full-code-lr2',           364, clrs(8,:),'-','third',17
%                   'code5-lr3',                    118, clrs(29,:),'-','first',11
%                   '2-code5-lr3',                 121, clrs(10,:),'-','second',13
%                   '3-code5-lr3',                 125, clrs(11,:),'-','third',13
%                   'np-code5-lr3',               118, clrs(12,:),'-','first',14
%                   'large-code5-lr3',          118, clrs(13,:),'-','first',17
};
algs=cell2struct(algs',{'name','epoch','color','style','fold','startChar'});
resPath='/home/vision/darknet/playYOLO-Caltech/history/';
j=1;
for i=1:length(algs) %逐个绘制不同实验设置的曲线
    %得到数据
    trainPath = [ rootpath algs(i).fold '/' algs(i).name '-train.txt'];
    vaildPath = [ rootpath algs(i).fold '/' algs(i).name '-vaild.txt'];
    [ trainName , trainLamr ] = textread( trainPath ,'%s %f' , -1);
    [ vaildName , vaildLamr ] = textread( vaildPath ,'%s %f' , -1);
    trainChar = char(trainName);
    vaildChar = char(vaildName);
    trainIter = trainChar(:,algs(i).startChar:end);
    vaildIter = vaildChar(:,algs(i).startChar:end);
    trainIter = str2num(trainIter);
    vaildIter = str2num(vaildIter);
    trainIter = trainIter / algs(i).epoch;
    vaildIter = vaildIter / algs(i).epoch;
```

```
    %分别绘制在训练集和测试集上的曲线
    plot(trainIter,trainLamr,'Color',algs(i).color,'LineStyle',algs(i).style,'LineWidth',1 ,'display
name',sprintf([algs(i).name '-train']));
    hold on
    plot(vaildIter,vaildLamr,'Color',algs(i).color,'LineStyle','--
','LineWidth',1,'displayname',sprintf([algs(i).name '-vaild']));
    hold on
end

xlabel('epoches');
ylabel('log-average miss rate');
grid on
legend('show');
```

3.4 本章小结

本章主要介绍了行人检测实验前的准备工作，包括修改方向、数据准备和评估设置三大部分。

第四章 实验和结果分析

本章介绍了基于 YOLOv2 目标检测器的行人检测研究实验过程。先搭建了实验环境，然后将 YOLOv2 目标检测器改成最基本的行人检测器，训练后，根据实验结果再设置相应实验设置，逐步尝试提高检测精度。

4.1 实验环境

本文实验基于YOLOv2目标检测器，采用Darknet框架，并且利用Matlab软件做数据准备与评估。整个实验是在Ubuntu 14.04系统上进行的，具体实验环境配置如表4-1：

表4-1 实验环境说明

操作系统	Ubuntu 14.04 LTS
内核信息	GNU/Linux_4.4.0-31-generic_x86_64
处理器	Intel® Xeon(R) CPU E3-1230 v5 @ 3.40GHz x 8
内存	7.7 GiB
显卡	GeForce GTX 1080
显卡驱动	NVIDIA-SMI 367.57
GCC	4.7.3
CUDA	7.5
OPENCV	2.4.8
MATLAB	8.4.0.150421 (R2014b)
Darknet	

4.2 实验和结果分析

YOLOv2 作者发布了在 PASCAL VOC 数据集和 COCO 数据集上训练好的 YOLOv2 网络，这两个网络都可以检测行人。表 4-2 给出了在两个数据集上训练好的 YOLOv2 网络在三折交叉验证中对数平均错误率（log-average miss rate，简称 LAMR）的结果，可以看到效果并不好，因为这两个数据集中的行人尺度都比

较大，而实验用的数据集行人尺度都较小。

表4-2 用于PASCAL VOC和COCO数据集的YOLOv2网络在三折交叉验证中的LAMR

验证集 / 训练集	第一折交叉验证	第二折交叉验证	第三折交叉验证
COCO	0.728930	0.733433	0.601146
VOC	0.706233	0.735046	0.586465

在用于 PASCAL VOC 数据集的 YOLOv2 网络上进行修改。

1、使用文献[2]中提到的 Anchor 尺度，还是使用文献[2]代码中的 Anchor 尺度？

先设置如表 4-3 所示的两组实验设置，看是使用文献[2]中提到的 Anchor 尺度，还是使用文献[2]代码中的 Anchor 尺度效果比较好。

表 4-3 实验设置 1,2

序号	1	2
配置名称	paper	code
训练集	-one.txt	-one.txt
Anchor	paper	code
初始学习率（learning_rate）	0.0001	0.0001
最大迭代次数（max_batches）	100000	100000
阶段（steps）	100,2000	100,2000
缩放因子（scales）	10,.1	10,.1
最后一层卷积层的 filter 个数	54	54
类别数（class）	1	1
类别名称	person	person
可调整网络倍数	32 的 10 至 19 倍	32 的 10 至 19 倍

表 4-3 中训练集“-one.txt”表示采用第一折交叉验证的训练集进行训练，Anchor 的“paper”表示使用文献[2]论文中提到的 Anchor 尺度和个数，“code”

表示使用文献[2]代码用的 Anchor 尺度和个数。训练网络需要 3G 多的显卡内存，单独训练一个网络，一次迭代需要 6 秒左右。如果同时训练两个网络，一次迭代需要 10 秒左右。由于时间有限，内存有限，这两组实验都迭代了一万多次就停止训练了。这两组实验的平均损失随迭代次数变化情况如图 4-1，可以看到，使用文献[2]代码中用的 Anchor 尺度和大小平均损失值稍微好一点点，没有太大差别，所以以下的实验设置都采用的是文献[2]代码中用的 Anchor 尺度。

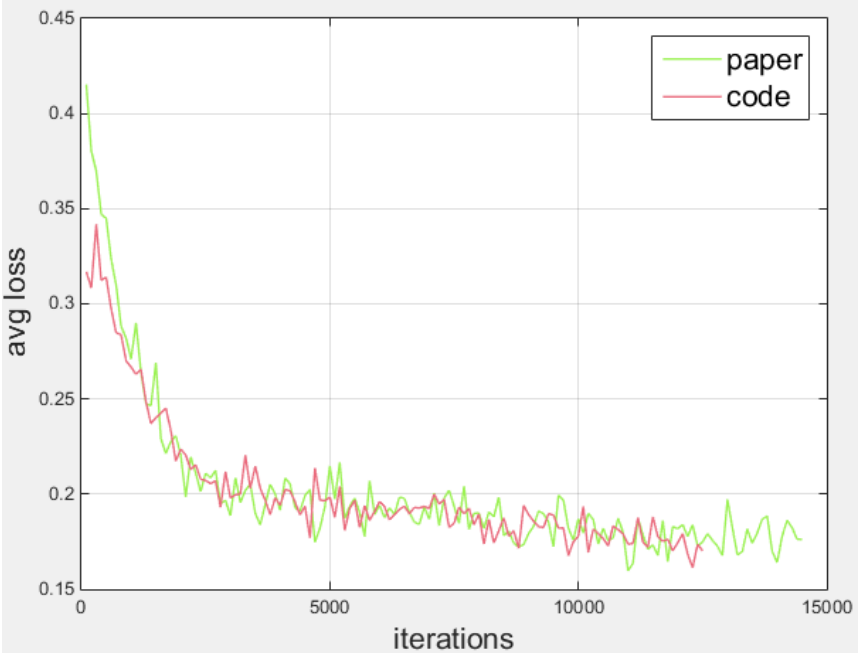

图 4-1 实验设置 1,2 的平均损失变化情况图

2、平均损失下降太慢，换组学习率是否会快一点？

图 4-1 中可以看到 2000 次迭代后，学习率降到 0.0001，平均损失值下降的速度太慢，于是换了一组学习率进行实验，见表 4-4：

表 4-4 实验设置 2,3

序号	2	3
配置名称	code	code-lr2
训练集	-one.txt	-one.txt
Anchor	code	code
初始学习率（learning_rate）	0.0001	0.001

表 4-4 实验设置 2,3（续）

最大迭代次数（max_batches）	100000	100000
阶段（steps）	100,2000	50000
缩放因子（scales）	10,.1	.1
最后一层卷积层的 filter 个数	54	54
类别数（class）	1	1
类别名称	person	person
可调整网络倍数	32 的 10 至 19 倍	32 的 10 至 19 倍

第 2,3 组实验设置的平均损失随 epoch 的变化情况如图 4-2。图中 100 个 epoches 左右损失值变成 0 是因为电脑内存不够，没有记录下来平均损失的信息，但总体来看，损失值下降得要快很多了。

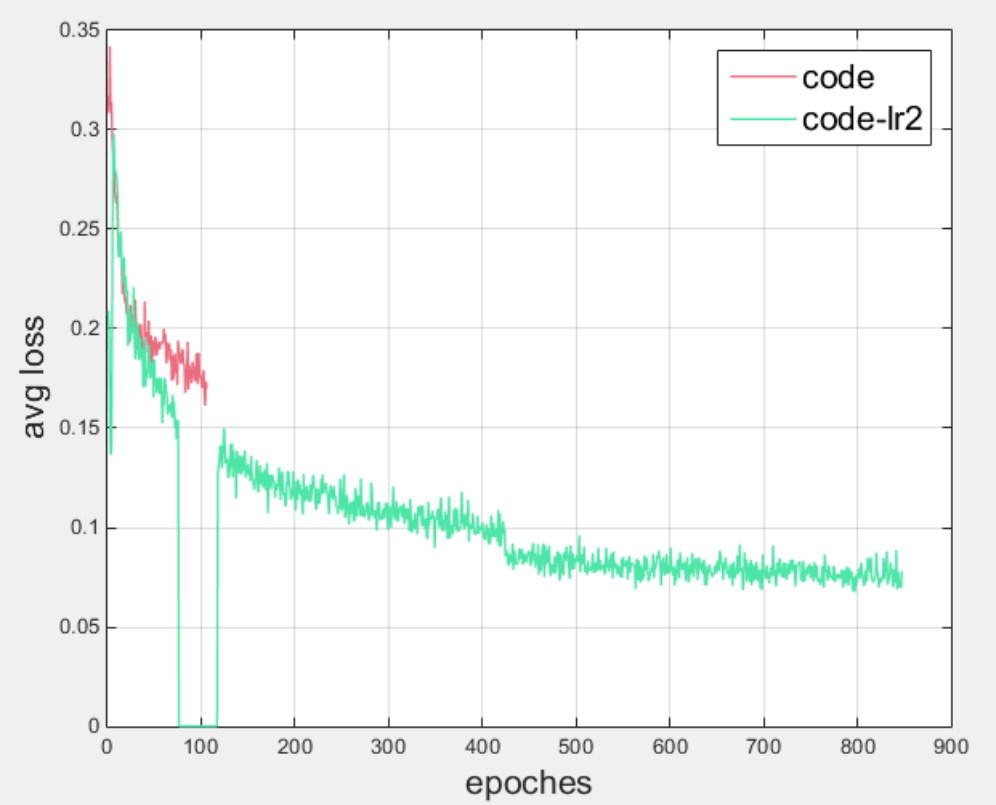

图 4-2 第 2,3 组实验设置的平均损失变化情况图

第 2,3 组实验设置的 LAMR 随 epoch 的变化情况如图 4-3：

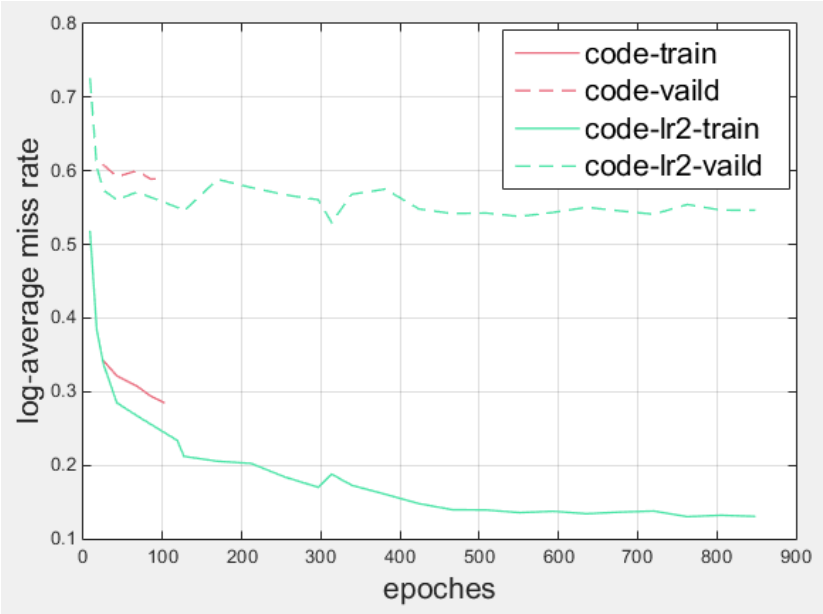

图 4-3 第 2,3 组实验设置的 LAMR 变化情况图

实线表示在训练集上的 LAMR 变化情况，虚线表示在验证集上的 LAMR 变化情况，可以看到，损失值下降得快一点的第 3 组实验，LAMR 也稍微好一点，所以之后的实验暂时用新的一组学习率。可是两组实验在训练集上的 LAMR 虽然还可以，都会随迭代次数的增加而减小，但在验证集上的 LAMR 要差很多，而且 50 个 epoches 之后基本没怎么变化。考虑是过拟合了，随后进行了 code-lr2 的另外两组三折交叉验证，结果见图 4-4，可以看到情况大致相同。

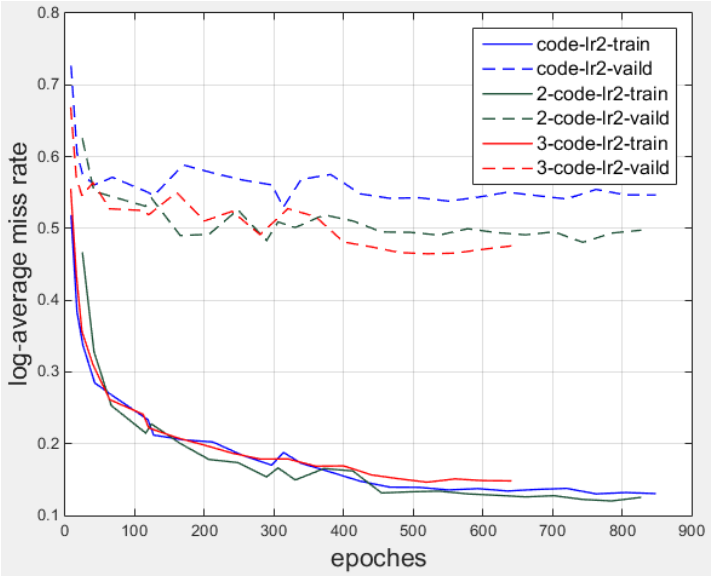

图 4-4 第 3 组实验设置的另外两组三折交叉验证的 LAMR 变化情况图

3、过拟合了，增加训练样本会好一些么？

训练过拟合了，原来使用的是 Caltech 每 3 帧取一帧的训练样本，改成使用取每帧的训练样本，增加训练样本试试。于是设置了第 6 组实验，见表 4-5。

表 4-5 第 3,6 组实验设置

序号	3	6
配置名称	code-lr2	full-code-lr2
训练集	-one.txt	-fullone.txt
Anchor	code	code
初始学习率（learning_rate）	0.001	0.001
最大迭代次数（max_batches）	100000	100000
阶段（steps）	50000	50000
缩放因子（scales）	.1	.1
最后一层卷积层的 filter 个数	54	54
类别数（class）	1	1
类别名称	person	person
可调整网络倍数	32 的 10 至 19 倍	32 的 10 至 19 倍

第 3,6 组实验设置的平均损失随 epoch 的变化情况如图 4-5。因为训练同样的迭代次数，而第 6 组实验，即 full-code-lr2 的训练样本多，一个 epoch 对应的迭代次数就多，所以折线就会比第 3 组实验短很多。从图 4-5 中我们发现，增多训练样本后，平均损失下降的速度快多了。

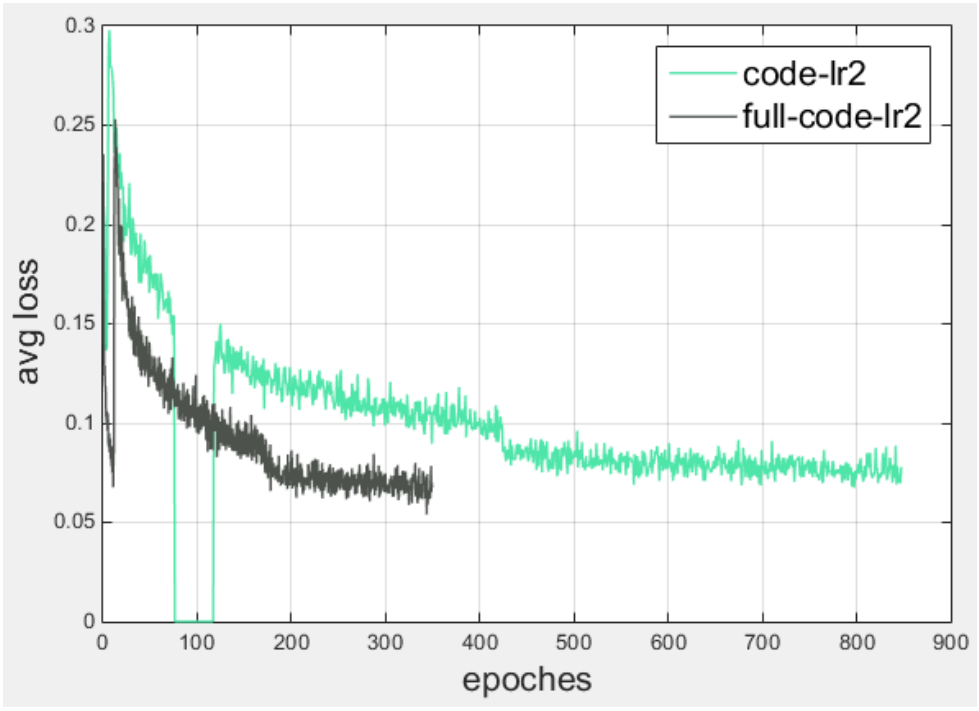

图 4-5 第 3,6 组实验设置的平均损失变化情况图

图 4-6 为第 3,6 组实验的 LAMR 随 epoch 的变化情况：

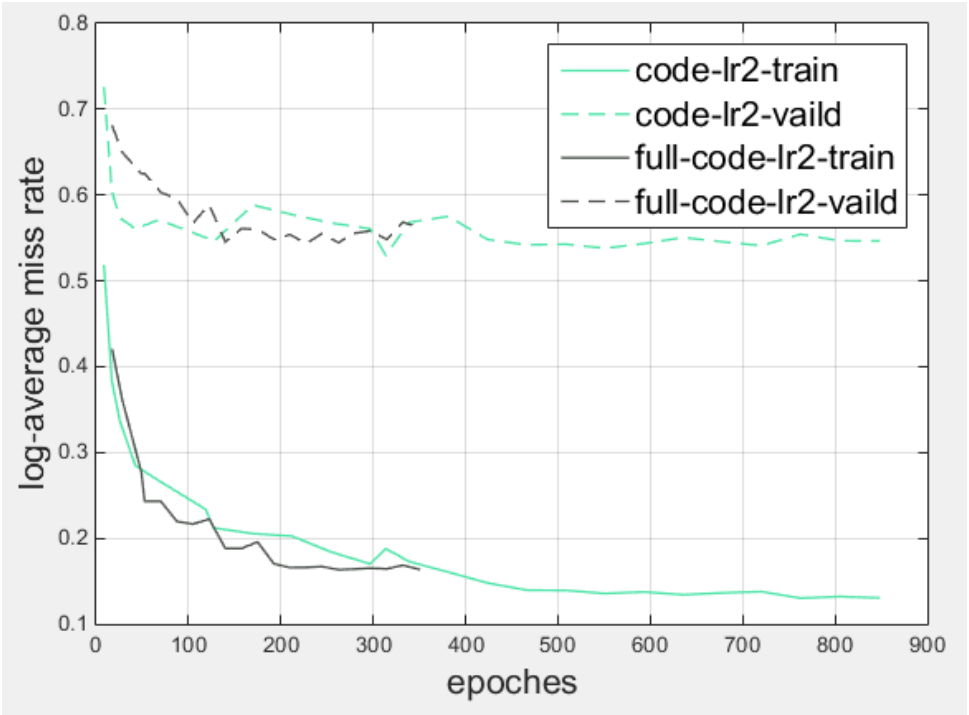

图 4-6 第 3,6 组实验设置的 LAMR 变化情况图

从图 4-6 上看，基本没什么变化，过拟合的情况还是存在，于是做了第 6 组实验的另外两组三折交叉验证，见图 4-7，4-8。情况大致相同，看来过拟合不是因为训练样本不够。

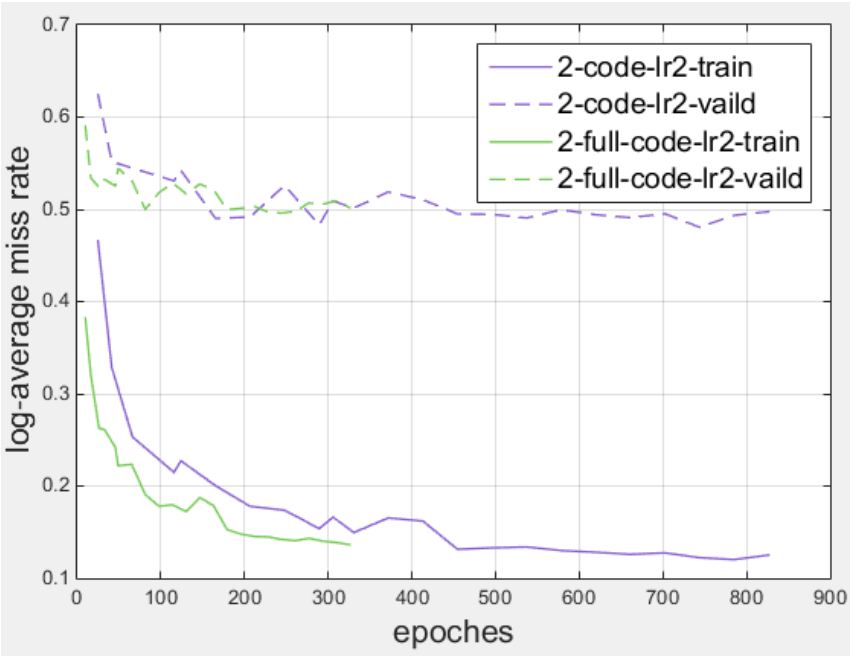

图 4-7 第 6 组实验设置第二折交叉验证的 LAMR 变化情况图

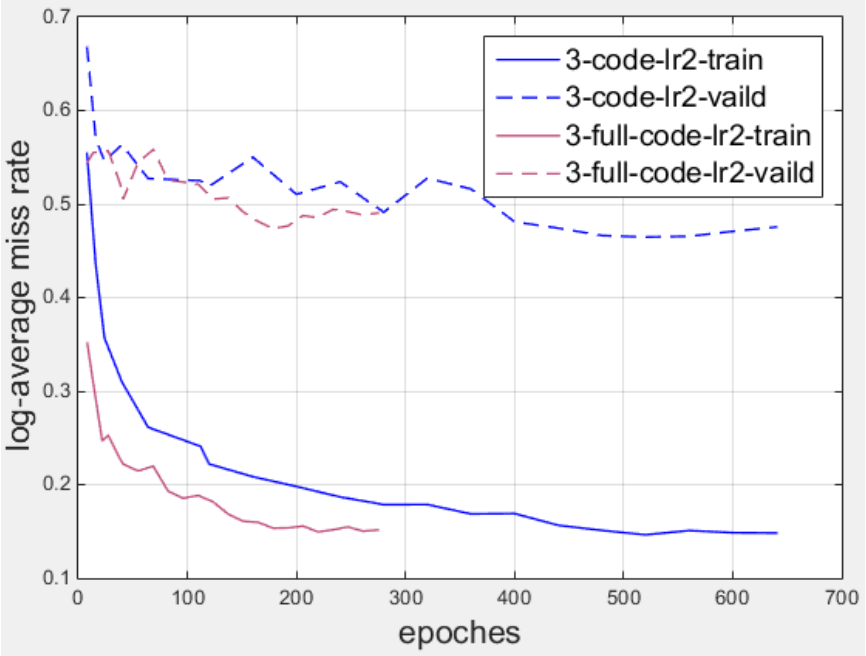

图 4-8 第 6 组实验设置第三折交叉验证的 LAMR 变化情况图

我们发现了一个奇怪的现象，见图 4-9。第 3 组和其另外两组三折交叉验证实验的平均损失，在开始的迭代中，总是有个迅速下降然后迅速上升的过程，模型不稳定。第 6 组和其另外两组三折交叉验证实验增加了样本数量，这个过程稍微提前了一点，但没有消失。

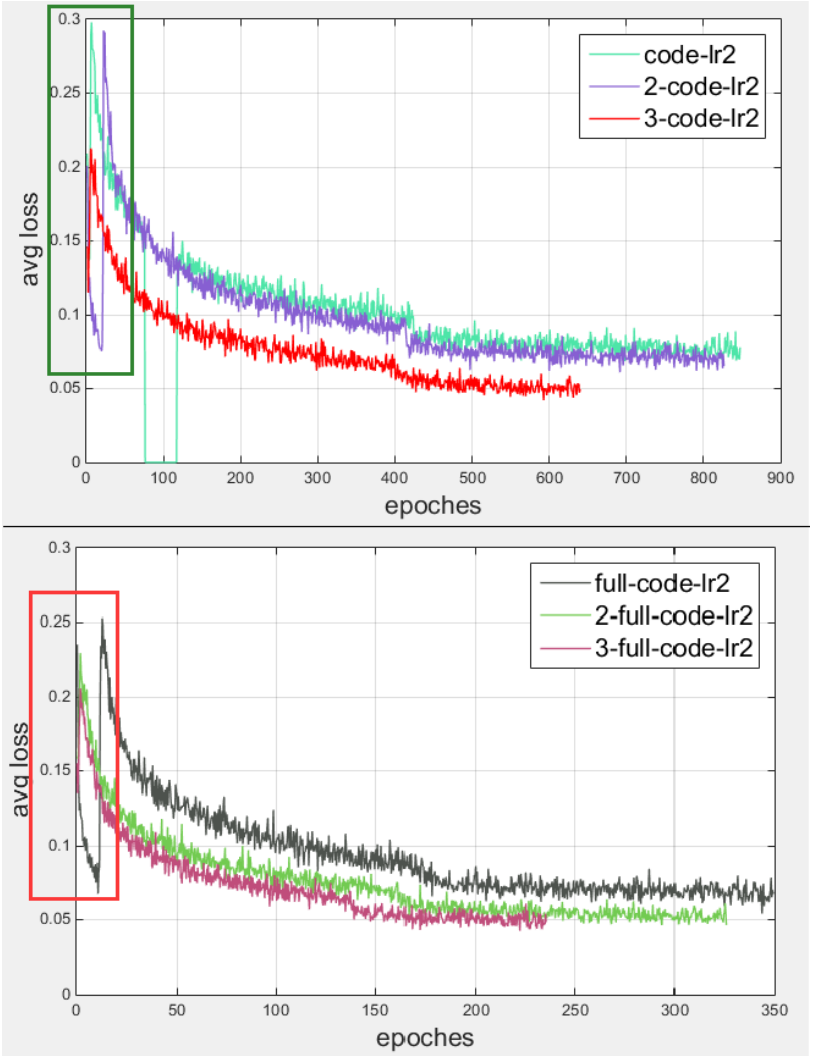

图 4-9 实验平均损失变化情况的奇怪现象

4、模型不稳定，降低前几次迭代的学习率试试？

文献[12]中提到，先以小的学习率开始，经过几次迭代后升高学习率，有利于模型稳定和损失下降快。文献[1]中提到 Anchor 的个数为 5 是精度和模型复杂度的一个权衡结果。之前用到的是文献[2]中的 Anchor，个数为 9 个，模型复杂度较高，可能也是过拟合的原因，于是我们再次换个学习率，并减少 Anchor 的

个数 ，设置了第 9 组实验，见表 4-6。由于增加训练样本数量，并没有改善过拟合的问题，为了节省时间，还是采用原来的训练样本，最大迭代次数改成 35,000。

表 4-6 第 3,9 组实验设置

序号	3	9
配置名称	code-lr2	code5-lr3
训练集	-one.txt	-one.txt
Anchor	code	code5
初始学习率（learning_rate）	0.001	0.0001
最大迭代次数（max_batches）	100000	35000
阶段（steps）	50000	1000,20000
缩放因子（scales）	.1	10,.1
最后一层卷积层的 filter 个数	54	30
类别数（class）	1	1
类别名称	person	person
可调整网络倍数	32 的 10 至 19 倍	32 的 10 至 19 倍

第 3，6，9 组实验设置的平均损失随 epoch 变化情况如图 4-10：

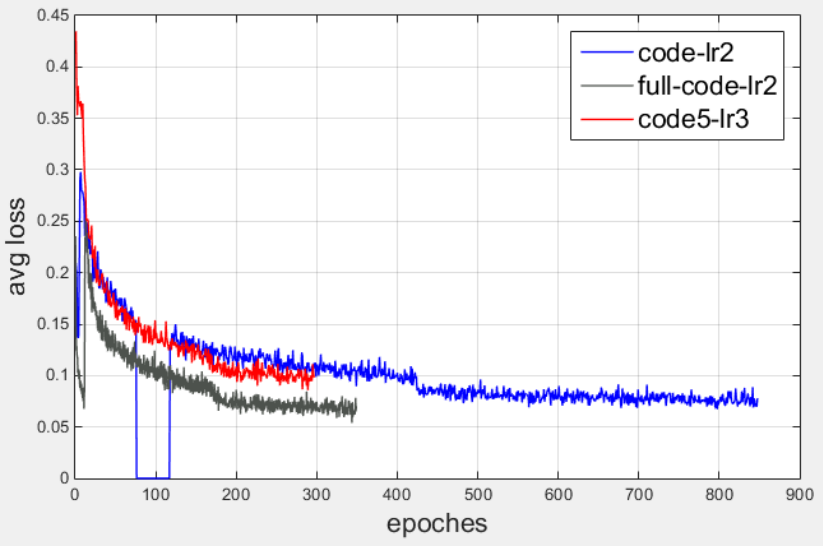

图 4-10 第 3,6,9 组实验设置的平均损失变化情况图

可以看到，模型已经比较稳定了，再进行了第 9 组实验设置的另外两组三折交叉验证，对比结果如图 4-11，4-12，效果大致相同。

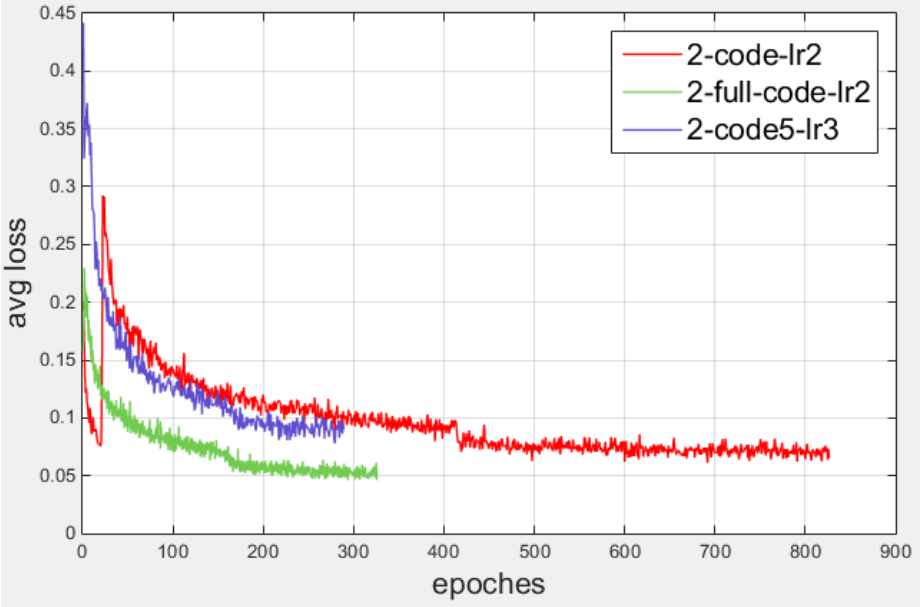

图 4-11 第 3,6,9 组实验设置第二折交叉验证的平均损失变化情况图

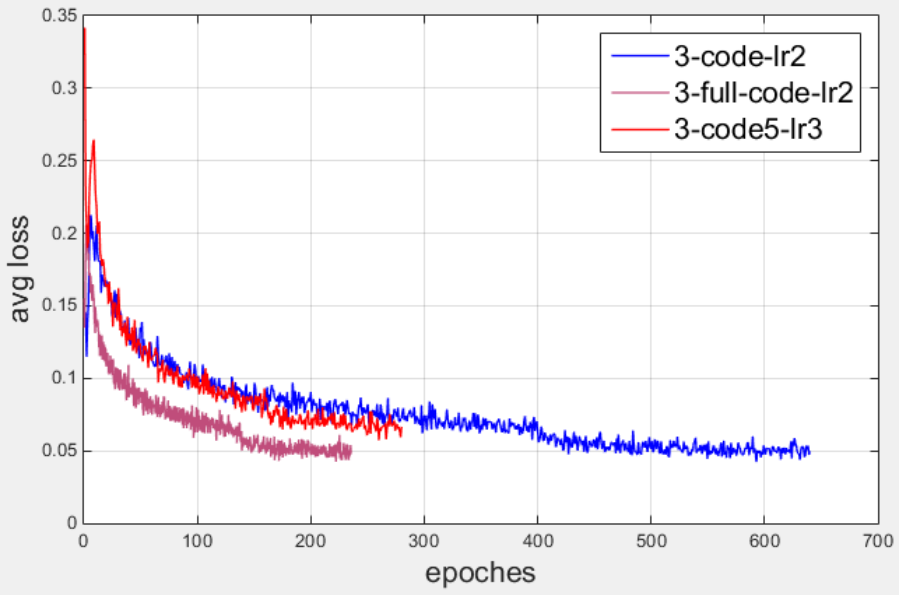

图 4-12 第 3,6,9 组实验设置第三折交叉验证的平均损失变化情况图

再看看 LAMR 的变化情况图，见图 4-13。还是没什么大的变化，所以过拟合也不是因为模型太复杂。

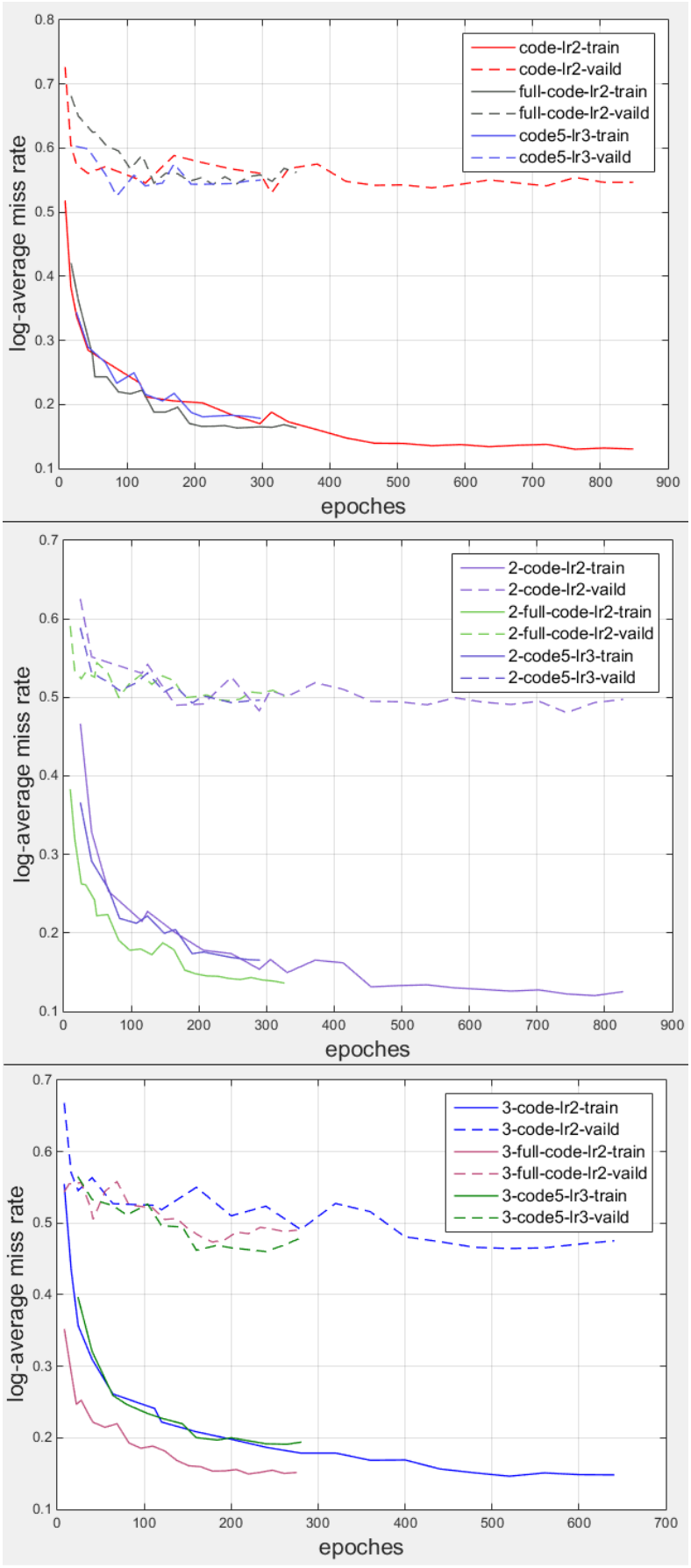

图 4-13 第 3,6,9 组实验设置及它们另外两折交叉验证的 LAMR 变化情况对比图

5、因为没有负样本，所以过拟合么？

之前的实验设置都设置的是一个类别，但是模型过拟合了。把检测问题当做分类问题来做时，一般还会设置背景类，学习背景特征，特别是易于和行人混淆的背景。在 YOLOv2 模型中，我们觉得不需要专门产生背景样本进行训练。因为 YOLOv2 模型会学习边界框内有无物体的概率和有物体的情况下属于某个类别的概率。由于只有一个类别，模型退化成，如果检测到有物体，该物体就是行人。如果增加了背景样本，在计算位置损失的时候就会把背景的坐标损失计算在内，这是不合理的。为了验证我的想法，设置了第 12 组实验，见表 4-7：

表 4-7 第 9,12 组实验设置

序号	9	12
配置名称	code5-lr3	np-code5-lr3
训练集	-one.txt	-one.txt
Anchor	code5	code5
初始学习率（learning_rate）	0.0001	0.0001
最大迭代次数（max_batches）	35000	35000
阶段（steps）	1000,20000	1000,20000
缩放因子（scales）	10,.1	10,.1
最后一层卷积层的 filter 个数	30	35
类别数（class）	1	2
类别名称	person	person；bg
可调整网络倍数	32 的 10 至 19 倍	32 的 10 至 19 倍

图 4-14 为第 9,12 组实验设置的 LAMR 变化情况图。可以看到增加背景样本，不管是在训练集还是测试集上，效果的确都比只用行人样本的差。

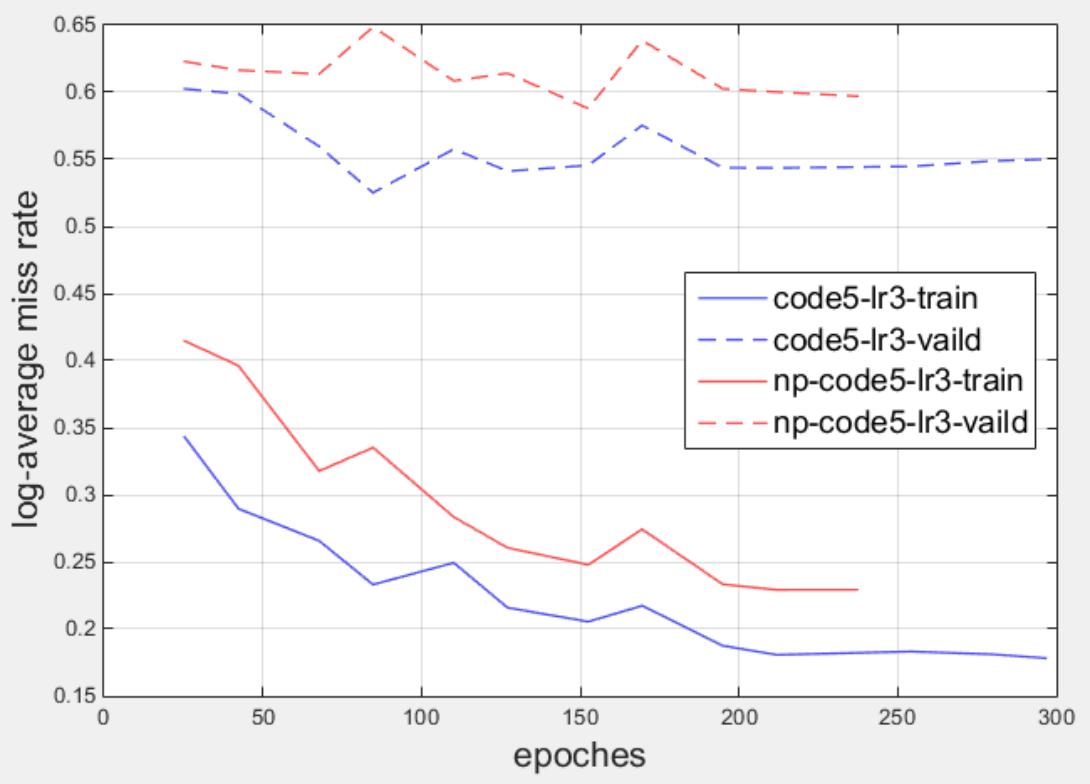

图 4-14 第 9,12 组实验设置的 LAMR 变化情况图

选取了实验过程中 LAMR 最小的实验设置 3-full-code-lr2 训练出来的 YOLOv2 网络与用 PASCAL VOC 数据集训练出来的 YOLOv2 网络进行检测结果对比，如图 4-15。图 4-15 中左侧原图来自于 Caltech 数据集的 set6 至 set10 测试集，行人真实边界框数据也来自于 Caltech 数据集官网，包括了完整的标注内容。图 4-15 中间是用 PASCAL VOC 数据集训练出来的 YOLOv2 网络的检测结果，蓝色框表示行人，可以看到漏检了许多尺度较小的行人。图 4-15 右侧是实验设置 3-full-code-lr2 训练出来的 YOLOv2 行人检测网络的检测结果。虽然网络过拟合了，但是对于长得像训练数据的图像，还是能较好的检测。

把 YOLOv2 目标检测器修改成行人检测器尚处于开发阶段，从验证集上的结果来看，检测精度离目前先进的行人检测器还有段距离，但速度已经达到实时的要求。实验设置 3-full-code-lr2 训练出来的 YOLOv2 模型在 GeForce GTX 1080 上检测 640*480 分辨率的图像速度为 50 FPS，这比目前排名第一的行人检测器 FDNN 快了 5 倍多。

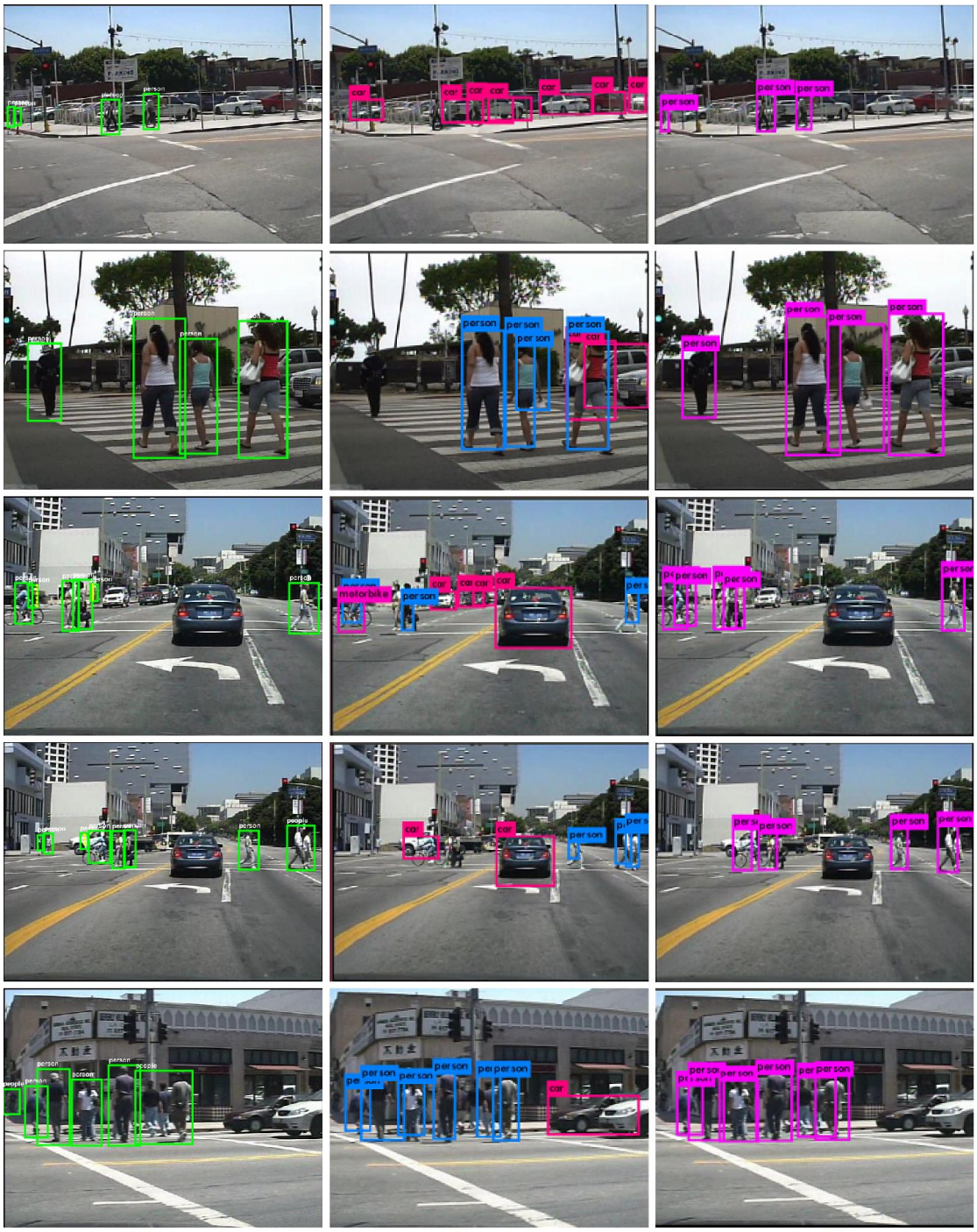

图 4-15 检测结果对比图

4.3 本章小结

本章介绍了实验的环境、将用于 PASACL VOC 的 YOLOv2 网络修改成行人检测网络的过程，以及尝试解决训练中问题的相关实验设置和结果分析。

第五章 总结与展望

5.1 论文总结

行人检测近年来已经受到学术界和工业界的重视，它是众多应用的底层基础。随着深度学习、卷积神经网络的发展，越来越多学者采用深度学习的方式来解决行人检测问题。

目前大多数行人检测器采用分类的方式解决检测问题，但是检测精度较高的行人检测器，检测速度都不快，不能达到工业界实时检测的性能要求。

本篇论文将目光着眼至目标检测领域的 YOLOv2 目标检测器。这是一个采用回归方式解决检测问题的目标检测器，模型结构简单，检测精度高，而且速度快。我们尝试将用于 PASCAL VOC 数据集的 YOLOv2 网络改成行人检测网络，并用行人数据集进行训练，根据训练结果一步一步尝试改进网络，提高行人检测的精度。

5.2 工作展望

本文成功的将用于 PASCAL VOC 数据集的 YOLOv2 网络改成检测行人的网络结构，但是网络的训练过拟合，经过若干尝试还是没能解决网络过拟合的问题。未来的研究工作主要从以下几个方面展开：

1、解决网络过拟合问题，可以尝试更改损失函数，损失权重的值。

2、提高对尺度较小行人的检测精度，可以尝试增大网络输入尺度，转移层连接低层但高分辨率的特征地图。

参考文献

[1] Redmon J, Farhadi A. YOLO9000: Better, Faster, Stronger[J]. 2016.

[2] Zhang L, Lin L, Liang X, et al. Is Faster R-CNN Doing Well for Pedestrian Detection?[M]// Computer Vision – ECCV 2016. 2016.

[3] Li J, Liang X, Shen S M, et al. Scale-aware Fast R-CNN for Pedestrian Detection[J]. Computer Science, 2016.

[4] Cai Z, Saberian M, Vasconcelos N. Learning Complexity-Aware Cascades for Deep Pedestrian Detection[C]// IEEE International Conference on Computer Vision. IEEE, 2015:3361-3369.

[5] Du X, Elkhamy M, Lee J, et al. Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection[J]. 2016.

[6] http://www.who.int/violence_injury_prevention/road_safety_status/2015/GSRRS2015_Summary_CH.pdf?ua=1

[7] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2012:1097-1105.

[8] He K, Zhang X, Ren S, et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2015, 37(9):1904.

[9] Girshick R. Fast R-CNN[J]. Computer Science, 2015.

[10] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2015, PP(99):1-1.

[11] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector[J]. 2015.

[12] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection[J]. 2016:779-788.

[13] http://blog.csdn.net/zouxy09/article/details/8775360

[14] http://www.cnblogs.com/maybe2030/p/5597716.html

[15] http://blog.csdn.net/v_july_v/article/details/51812459

[16] http://blog.csdn.net/zouxy09/article/details/8781543

[17] http://www.cnblogs.com/CarryPotMan/p/5343693.html

[18] Wojek C, Dollar P, Schiele B, et al. Pedestrian Detection: An Evaluation of the State of the Art[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2012, 34(4):743-761.