杭州電子科技大学

本 科 毕 业 设 计

（2020 届）

题　　目	基于深度学习的汽车车型自动识别
学　　院	电子信息学院
专　　业	电子信息工程
班　　级	16041818
学　　号	16073212
学生姓名	滕佳禄
指导教师	项铁铭
完成日期	2020 年 6 月

诚 信 承 诺

我谨在此承诺：本人所写的毕业论文《基于深度学习的汽车车型自动识别》均系本人独立完成，没有抄袭行为，凡涉及其他作者的观点和材料，均作了注释，若有不实，后果由本人承担。

承诺人（签名）：

年　　月　　日

摘　要

近年来中国迎来了信息现代化的快速发展，交通服务行业也逐步向信息智能化方向转型，在转变过程中车型识技术起着至关重要的作用。本文研究基于深度学习的汽车车型自动识别方法，选择获得 2012 年 ImageNet 图片分类大赛冠军的模型 AlexNet，对自制的车型数据集进行训练，在测试集上的识别准确率达到了 86.9%，对该模型进行改进后，准确率提高到 88.4%，基本能够达到应用于真实场景的要求。

为了方便实现 AlexNet 模型，本文将该模型的双 GPU 并行结构改为单 GPU，并且调整了模型的部分结构和参数，以平衡失去一个 GPU 所带来的不利影响。该模型的网络结构包含 5 层卷积层和 3 层全连接层，这 8 层结构依次排列，上一层的输出作为下一层的输入，最后一层的输出结果为包含 6 个节点的向量，代表预测出的 6 种不同车型的概率。卷积神经网络与手工提取特征的机器学习方法相比有很多优势，例如不需要人工提取特征、具有更强的泛化能力等。

本文训练模型时所使用的数据集均来自汽车网站内可供下载的车型图片，共计 7530 张，其中 6144 张被制作为训练集，剩余部分用作测试集。为了提高加载数据集和训练时的速度，将图片的像素值提取出来，多张图片合成一个二进制文件，所有图片最终被制作成 15 个数据集文件。

AlexNet 模型的实现基于深度学习框架 Pytorch，该框架目前广泛应用于深度学习各个领域的研究中。框架支持自动求导和反向传播，并且内置各种卷积函数、损失函数等一系列 API，使 AlexNet 模型的实现降低了难度。最后将训练好的模型集成在一个窗口应用程序中，识别车型图片时只需打开文件夹选择图片，在应用程序中点击识别按钮便可自动识别出车型。这样既方便操作，又可移植到手机或其他设备上，直接应用在真实场景中。

关键词：车型识别；深度学习；卷积神经网络；AlexNet 模型

ABSTRACT

In recent years, China has ushered in the rapid development of information modernization, and the transportation service industry has also gradually transformed to the direction of information intelligence. In the transformation process, vehicle model recognition technology plays a vital role. This paper studies the automatic recognition method of car models based on deep learning, selects the model AlexNet that won the 2012 ImageNet image classification competition, trains the self-made model data set, and the recognition accuracy rate on the test set reaches 86.9%. After the improvement, the accuracy rate was increased to 88.4%, which can basically meet the requirements of applying to real scenes.

In order to facilitate the realization of the AlexNet model, this paper changes the dual GPU parallel structure of the model to a single GPU, and adjusts some of the model structure and parameters to balance the adverse effects of losing a GPU. The 8-layer structure is arranged in sequence, the output of the upper layer is used as the input of the next layer, and the output result of the last layer is a vector containing 6 nodes. Convolutional neural networks have many advantages over machine learning methods that manually extract features, such as no need to manually extract features, and a stronger generalization ability.

The data set used for training the model in this article are all from the downloadable car images on the car website, a total of 7530, of which 6144 were made as the training set, and the rest is used as the test set. In order to improve the speed of loading the data set and training, the pixel values of the pictures are extracted, and multiple pictures are combined into a binary file, and all the pictures are finally made into 15 data set files.

The implementation of the AlexNet model is based on the deep learning framework Pytorch, which is currently widely used in research in various fields of deep learning. The framework supports automatic derivation and back propagation, which makes the implementation of the AlexNet model less difficult. Finally, integrate the trained model in a window application. This is not only easy to operate, but also can be transplanted to mobile phones or other devices, directly applied in real scenes.

Key words： vehicle identification; deep learning; convolutional neural network; AlexNet model

目　录

1 引言

近几年在我国经济和科技快速发展的环境下，现代制造业和汽车工业得到了迅速成长，汽车的产能和销量已经保持了多年的持续增长。据国家统计局[1]数据显示，2019 年年末全国私人汽车保有量达到 22635 万辆，较上一年增加 1905 万辆。另外据中国汽车工业协会[2]的数据显示，2019 年中国汽车产销分别完成 2572.1 万辆和 2576.9 万辆，产销量继续蝉联全球第一。

汽车的大量普及虽然为人们的出行和生活带来了极大地便利，但也造成了很多不利的影响，诸如交通拥挤事故频发，道路堵塞引起的医疗急救、火警救援不及时等一系列交通问题日益凸显，人们的日常出行和个人安全受到了很大的影响。近年来人工智能、深度学习、大数据等技术的发展取得了突破性的进展，促使交通服务行业也得到了升级。其中，车型识别是最为重要和基本的组成部分和研究内容。

车型识别技术能够应用在众多场景中，例如交通流量检测、高速收费和违规检测、刑侦办案等。通过车型识别技术，交通流量智能检测系统可以自动统计通过监测点的不同类型的车辆数量。在高速收费系统中，车型识别技术与车辆其他信息相结合，可以进一步提高识别准确度，提高违规车辆的定位与追踪能力。在公安和刑侦系统中，车型识别技术能帮助公安部门识别出嫌疑车辆，有效协助犯罪案件的侦破，缩短案件侦破的时间。

近几年在深度学习技术的铺垫下，计算机视觉呈现出迅速发展的趋势。通过对大量标注数据进行训练，得到的模型能够达到比手工设计的特征模式更高的准确率，相比基于机器学习的传统方法，训练出的模型具备更强的泛化能力，同时对各种场景下的适应性也变得更强。基于以上观点，研究基于深度学习的汽车车型自动识别技术对于交通行业和公安部门来说非常有必要。

本文采用卷积神经网络对车型进行识别，该模型由获得 2012 年 ImageNet 大赛冠军的模型 AlexNet 调整而来，并且在本文所使用的自制汽车数据集上达到了非常高的准确率，这对车型识别技术在其他领域广泛应用提供了有力的支撑。

2 概述

2.1 研究现状

2.1.1 车型识别研究现状

车型识别技术主要包括车辆信息采集和处理、车型识别以及数据管理等方面。车型识别技术在过去很长一段时间没有太大的发展，直到最近几年才出现了突破，其识别方法主要根据负责识别功能的模块是基于硬件还是软件进行分类，按照这个方式基本可分为两类：基于硬件的物理参数模式识别和基于软件的计算机视觉图像识别。

对于第一类，基于物理参数模式识别的方法需要硬件环境的支撑，该类方法能够没有遗漏的捕获所有经过的车辆，并且通常不会有太大的差错。但其缺点也很明显，硬件设备的部署和维护代价很高，且易受设备所处工作环境的影响，不太适用于普通城市道路。目前此类方法的实际应用有红外线探测法、地感线圈检测法和动态电压检测法等。国内外相关人员对此方法进行了广泛研究，樊海泉[3]等人通过研究发现，不同车型由于轮廓和内部结构不同，会对地磁设备发出的磁场产生一定的影响，用这种方法能够识别出通过地磁设备上方的不同车型，再将地磁数据应用在模式识别的匹配算法上，取得了不错的效果。蔡智湘[4]等人通过二维红外检测技术获取车辆外形的长宽高等几何参数，然后结合模糊逻辑来自动识别车型。张伟[5]从小波变换的角度出发，对频率变化曲线进行小波变换，以此来提取车型的局部特征，再与特征匹配相结合进行车型识别，达到了较高的识别准确率。

另一类车型识别系统不需要专业硬件的支持，仅仅通过图像处理与特征提取来实现。该方法投入成本低、数据来源广泛、不影响交通系统的正常运作，得到了广泛使用。不过，这种方法依靠提取的单个模板特征进行特征匹配，这些特征依赖于人们对大量经验的总结，在特征提取时不仅耗时费力，还需要对图片进行复杂的预处理操作，这样做可能会过于关注对车辆识别没有帮助的细节信息，反而浪费了更多的运算成本。

近几年，基于深度学习[6]的汽车车型识别技术快速发展，许多学者进行了相关研究。贾瑞[7]使用 AlexNet 模型对斯坦福大学公开的汽车数据集进行识别，获得了比较好的效果。石维康[8]采用改进的特征金字塔 SSD 模型加上汽车部件检测模型，配合 AlexNet 网络在斯坦福数据集上取得了比以往更高的识别准确率。黄强强[9]将 AlexNet 模型与颜色识别结合起来进行车型识别，并且与仅用特征提取和颜色识别得到的效果进做了对比。楚翔宇[10]采用基于区域的卷积神经网络(RCNN)和改进后的 fastRCNN 进行车型识别。这些研究共同推动了深度学习在车型识别任务中朝着

更光明的方向前进。

2.1.2 深度学习研究现状

深度学习的概念最早在 2006 年由在人工智能领域耕耘多年的 Hinton 教授以及他的学生共同发表的一篇论文中首次提出。近年来互联网产业迅速发展，各行各业在网络中产生的数据迅速膨胀，这些海量的数据对深度学习的发展起到了极大的推动作用，大量研究人员和科技公司开始将精力投入进来。目前深度学习技术广泛应用在各种具体场景中，其中最有成效的三个领域分别是图像识别、语音识别和自然语言处理。

在这三个领域中，图像识别是深度学习算法最早涉足并且迅速获得成就的领域。在 2012 年举办的基于 ImageNet 数据集的 ILSVRC 大赛中，Krizhevsky[11]等人提出了一个多层卷积神经网络模型 AlexNet。该模型在识别 1000 类图像准确率上较以前基于特征提取的方法有了巨大的提高，一举赢得了当年的比赛。在 2013 年第二届的比赛中，成绩排在前 20 名的队伍所使用的技术全部都为深度学习。2014 年来自谷歌团队的网络模型 GoogLeNet[12]使得分类错误率降到了 6.7%。2015 年，卷积神经网络进一步朝着层级结构更加复杂的方向发展，微软亚洲研究院的团队设计了具有更深层次的网络，该网络拥有 152 层神经元，并将其取名为 ResNet[13]，它比之前所出现的网络能够提取更多的特征，但通过减小卷积核尺寸和丢弃部分神经元，使得算法复杂度没有太大增加。由此，深度神经网络已经和人眼识别物体的能力相近了。

在图像领域还有一个热门方向：风格转换和图像生成。这个领域最大的特点就是图像的无中生有，其做法是使用提供的图像通过学习后生成一些本来不存在的图像。拿风格转换为例，假如有两张图片，一张是随手拍的照片，另一张是法国著名画家莫奈绘制的一幅画，这幅画具有莫奈画作活泼开放的风格。现在通过风格转换，生成一张新的图像，这张图像是以莫奈风格绘制出的那张照片。

图像生成则是根据一段文字生成能够描述这段文字的图片。这个领域目前正在积极探索，比如有这样一句话：远处有一条河，近处有两只羊。那么程序就会自动生成一张描述这段文字的图片，其真实性与照片拍出来的效果相差无几。在这个领域有个热门的研究方向叫做生成对抗网络（GAN），该模型由生成器和鉴别器两部分组成，生成器用来来生成虚假物，鉴别器用来鉴别该虚假物是真实的还是虚假的。生成对抗网络中有个典型的模型叫做 CycleGAN，它实现了图片的一些非常有趣的转换，例如将真实的照片转换为某位画家风格的绘画，将具有某个季节特点的风景图片转换为另一个季节特点的图片，又或者将包含某种动物的图片转变为另一种动物。

2.2 研究内容与组织结构

本文的研究内容是根据提供的汽车图片来识别车型，并能将汽车分为大型车(包含公交车，客运车，货车三类)，中型车(包含面包车，皮卡车两类)和小型车(轿车)这六类。

针对本文所研究的内容，做了相应的工作。

第一，制作车型数据集，本文没有选择斯坦福大学的车型数据集，自己制作了质量更高的包含 6 类车型的数据集。该数据集共计 7530 张图片，每类车型数量均超过 1000 张，其中 6144 张用作训练集，剩下的图片制作成测试集，用来验证识别准确率。这些图片均来自汽车网站内公开的图片，车辆均为街道或公路正常拍摄且包含整个轮廓。

第二，识别模型采用对 AlexNet 网络结构进行调整后的模型，在 Pytorch 框架搭建的模型上训练，之后将训练出的含有大量参数的模型应用在测试集上，统计不同训练轮数在测试集上的识别准确率的情况。

第三，改进网络模型，对某些卷积层的卷积核尺寸和全连接层中节点的个数进行修改，期望能够高识别的准确率。而后开发窗口应用程序，调用训练好的模型对真实场景中拍摄的车型图片进行识别。

本文各章节具体内容按照所做工作的顺序进行编排，下面分别介绍每章的具体内容：

第一章为引言，介绍课题研究的背景和意义。

第二章概述，主要介绍课题的研究现状，涉及目前为止车型识别的研究现状和近年来深度学习的发展情况，最后概述论文的组织结构。

第三章介绍深度学习和卷积神经网络的理论基础，其重点在于卷积神经网络的原理和基本结构，以及整个网络中每个组成部分的原理与实现细节。

第四章介绍本文所使用的具体卷积神经网络模型 AlexNet。主要介绍该模型提出的背景，模型特点以及具体的网络结构，同时阐述整个网络内部的运行过程、运行中图形尺寸的变换和最终输出值的含义。

第五章使用开源深度学习框架 Pytorch 实现 AlexNet 模型，主要阐述搭建运行环境、加载数据集、实现模型、训练模型等步骤。

第六章分析车型识别结果，训练时发现不同训练轮数对识别准确率有一定的影响，并且得到了不同训练轮数下训练出的模型在测试集上的准确率。

第七章对模型进行改进，对某些卷积层的结构和部分参数进行调整，比如卷积核尺寸和全连接层节点个数，期望能够提高识别准确率，最后总结不同改进方案对准确率的影响，并对产生影响的原因进行分析。

3 深度学习与卷积神经网络

3.1 深度学习概述

深度学习是机器学习方法中的一种，最早于2006年提出，使用给定的训练方法将准备好的样本数据训练为具有多层网络的结构，从而能够更精准的反映数据所表现出的特征，具有更强的特征学习能力。

通常来说，神经网络中权重一般都默认设置随机的初始参数，这使得网络最有可能在局部最小值区域收敛。但深度学习可以优化网络的初始权重，再通过无监督学习来调整权重，这样可尽量避免收敛于局部最小值。

深度学习的具体结构通常是由一个非线性函数连接的深层网络，这个网络初始时不具有特征表征功能，通过不断逐层学习的方式逐渐完善，直到完整学习到稳定的数据特征，这种方式通常称之为深层学习。与深层学习对应的是浅层学习，它由人工在具体应用场景的基础上设计而来，这种特征模式具有单一结构，在分类任务和可视化时没有深度学习的优势。由以上描述，可将深度学习的思想概述为设计一个多层级结构，每一层之间前后连接，经过多层网络提取出输入数据的各个维度上的特征，最后映射到输出空间中。这种算法的原理与人脑识别物体的过程非常相似，先提取目标物体的边缘、色彩、材质等特征信息，再对提取到的各种特征信息进行更高层次的抽象化表示，最终在大脑负责视觉的区域形成更加复杂的视觉形状，以此来识别出该目标物。

3.2 卷积神经网络

3.2.1 卷积神经网络概述

深度学习近几年逐渐成为图像分类领域的主流研究方法，在众多不同深度神经网络结构中，卷积神经网络是应用最广泛、效果最好的网络结构。

卷积神经网络的概念由生物视觉的感受方式衍生出来，研究人员最初发现猫的视觉皮层上有一种细胞，它们负责收集外界的光信号，并且进行处理。到上世纪90年代，LeCun[14]等人发表论文，构建了一种由多个卷积层连接的网络结构，后来的二十多年，不断有人针对不同应用场景，对网络结构进行扩展和修改。

卷积神经网络相对于前馈神经网络进行了改进和提升，但两者的基本模型和原理仍然相同，它们的实现过程都是通过前向传播来计算输出值，然后将输出的预测值与真实值进行比较，计算出误差，最后利用反向传播算法对每层网络的参数进行修正，直到满足停止条件。

卷积积神经网络的整体结构一般比较固定，通常由输入层卷、卷积层、激活层、

全连接层、输出层等部分先后连接而成，通过对卷积层和全连接层的各种组合，或者调整卷积核个数和尺寸等参数，设计成各种不同结构的具有多层级的网络模型。在 LeCun 的论文中，他们提出了一个用来对手写的阿拉伯数字进行识别的网络模型，将其取名为 LeNet-5，该模型被后来被认为是最早应用在实际场景中的卷积神经网络，其网络结构包含了所有关键的部分，结构图如下：

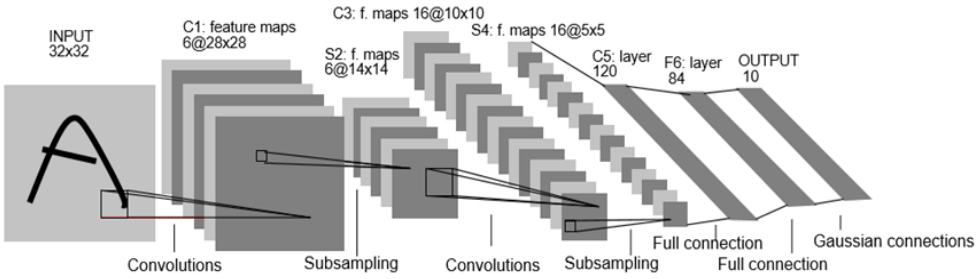


图 3.1　LeNet5 网络结构

卷积神经网络中的多层结构能够提取出原始图像中的特征，并且可以同时提取出多个不同维度、相互无关联的特征，这表明网络可以像人的眼睛一样从图像中识别出视觉上展现出来的物体的特征规律。然而，当时计算机硬件性能不足，运算能力较差，又没有足够的样本数据供网络训练，导致 LeNet-5 在数字识别上的效果不能复制到稍微复杂的应用场景中。

2012 年开始在 ImageNet 数据集上举办图像分类大赛之后，参赛者设计了更多性能卓越的网络，用来对大型图片库进行识别，比如 AlexNet、VGGNet、GoogLeNet、ResNet 等模型，从这以后卷积神经网络在许多领域大放光彩。

3.2.2 卷积层

卷积层是卷积神经网络中起关键作用的结构，它的主要作用是提取输入图像中能够表征图像类别的特征，正因为有了卷积这个核心运算，图片识别再也不用手动设置特征结构和提取特征，研究人员不再关注特征提取方面的技术，转而将精力放在了卷积层级结构和卷积核尺寸的设计。

每个卷积层通常会提取很多个特征，这些特征的提取依靠该卷积层中不同的卷积核，因此很多卷积神经网络中的卷积层包含几十甚至上百个卷积核。图 3.2 展示了卷积运算的过程，左边是一个尺寸为 6*6 的图像，在该图像上进行卷积操作，得到一个结果为 4*4 的图像，其中卷积核的大小为 3*3，步长为 1。

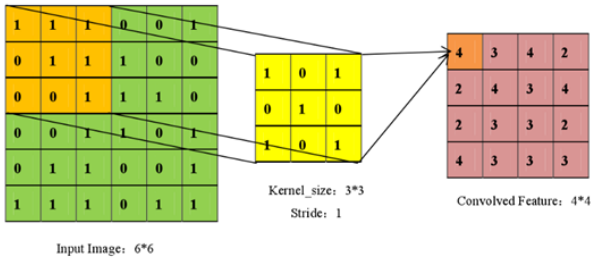


图 3.2　卷积运算示意图

3.2.3 池化层

卷积层提取特征以后，生成的特征图一般跟原图像尺寸相近，但是特征图的数量却增加了很多，这样不仅增加了后序处理数据的运算量，还由于特征图表现了太多的细节，导致容易出现过拟合，池化操作就是用来改善这两个问题，经过池化后图像尺寸通常会缩小，并且特征映射对输入图像的平移操作比较敏感，池化后降低了平移效应的精度。

池化操作的关键在于窗口的选取，通常都选择中等尺寸 2*2，计算窗口内池化后的值的算法有很多，应用比较广泛的有最大池化和平均池化。最大池化取窗口中的最大值，平均池化选择窗口所有值的平均值。对于池化窗口为 2*2，步长为 2 的池化操作，运算过后可将原图的尺寸缩小一半，降低后序操作的运算量。图 3.3 展示了两种池化操作的过程：

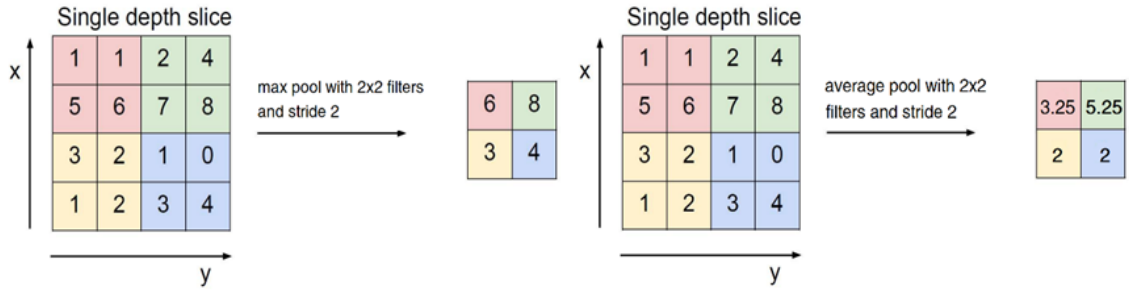


图 3.3　池化操作示意图

3.2.4 全连接层

在多个卷积层和池化层组成的结构之后，神经网络最后总是接入到全连接层，全连接层的个数不是固定的，通常有一层或者好几层。全连接层中的节点不像卷积层那样呈现二维结构，它们以一维向量的形式与上一层保持全连接的状态。全连接层与卷积层都做为网络中的层级关系而存在，唯一不同的地方在于卷积层与上一层的部分区域相连，虽然结构形式有一些不同，但它们在进行计算时的函数表达式的结构是相同的。

全连接层中节点的数量取决于具体应用场景，上一层的每个节点不是单独存在的，也不是随意连接的，他们作为下一层的输入，与下一层的所有节点相连，当前层只接受来自前一层节点的数据。最后一层的作用比较特殊，作为整个网络的输出，它们的输出值直接对应预测结果，节点个数与最终的分类个数相同，每个节点的相对大小关系就是落在该类别的可能性，以此作为模型的误差，该误差是反向传播更新参数的依据。

3.2.5 激活函数

在具有多层结构的神经网络中，相邻层之间不是直接相连的，通常会有一个变换来进行映射，这个过程可以用一个函数来表示，这个函数是必须的，如果不用的

话，相邻两层之间就是线性的关系，失去了多层结构的意义，这样的函数有一个专业的词汇，叫做激活函数。激活函数有很多，常见的用于神经网络中的有以下几种。

1）Sigmoid 函数

Sigmoid 是常用的非线性激活函数，表达式的分母含有指数运算，具体的函数形式和图像如下：

$$f(z)=\frac{1}{1+e^{-z}} \tag{3-1}$$

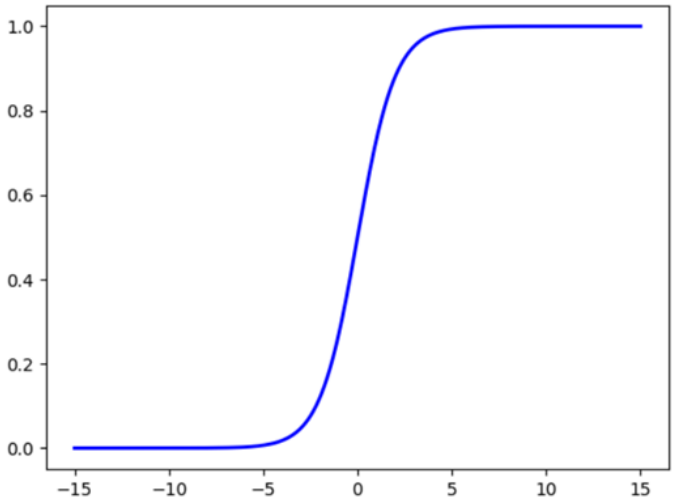


图 3.4　Sigmoid 函数图像

Sigmoid 函数有一个典型的特点，它的输入是实数空间内的所有值，输出是介于 0 到 1 之间的值，当输入为极小或极大的值，根据函数的特点，输出会无限趋于 0 和 1。这种变换将所有数据映射到 0-1 的范围内，这种映射关系有利于表达神经元激活和未激活两种状态。

不过 Sigmoid 函数也有一些不可忽视的缺点。第一，该函数很有可能会出现梯度消失的现象，从图 3.4 可以看出，当输入值出现极小值或极大值时，函数曲线非常平，它的导数会无限趋于 0，在反向传播中需要利用梯度下降来更新权重，导数为 0 时就失去了作用，该参数在网络中不能得到更新，网络就丧失了学习能力。第二，该函数在计算时需要进行指数运算，计算机往往使用泰勒展开来迭代计算指数函数，这样会导致运算速度变慢，对于规模非常大的网络，这无疑是提高模型运算效率的瓶颈。

2）tanh 函数

Tanh 函数同样含有指数运算，它的表达式和图像如下：

$$f(z)=\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \tag{3-2}$$

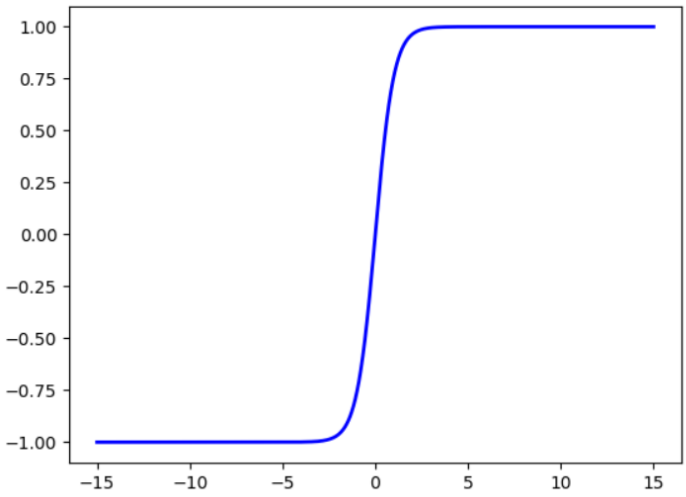


图 3.5　tanh 函数图像

Tanh 函数和 Sigmoid 函数非常相似，不同点在于 Tanh 函数的输出介于-1 到 1 之间，除此之外，它和 Sigmoid 函数没有其他性质上的差异，二者具有相同的特点和缺点。

3）ReLU 函数

ReLU 函数是目前使用最广泛的激活函数，它的表达式非常简单，输入小于 0 时为 0，大于 0 时是其本身，函数图像如下：

$$f(z) = \max(0, z) \tag{3-3}$$

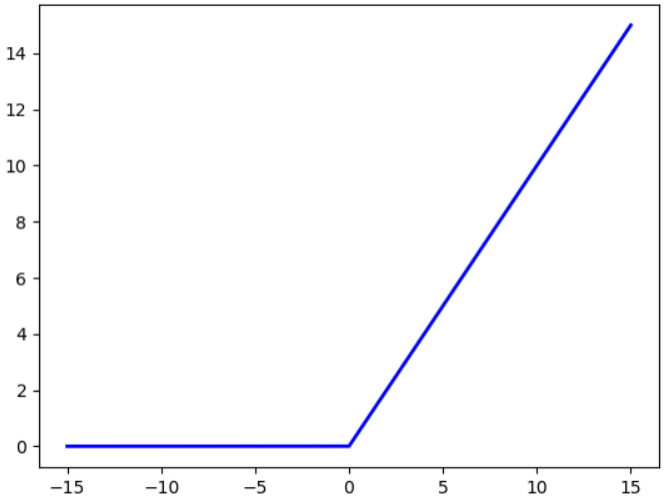


图 3.6　ReLU 函数图像

ReLU 函数虽然简单，但在近几年关于卷积神经网络的模型和论文中被广泛使用，在激活函数中起着非常重要的作用，它有几个典型的特点。首先，收敛速度远快于前面提到的两个函数。其次，在输入大于 0 的部分解决了梯度消失的问题。再者，该函数无需进行指数运算，为模型训练节省了时间。这几个优点使得 ReLU 目前大量应用于神经网络。

4）ELU 函数

ELU 函数是由 ReLU 函数改进而来，该函数在输入小于 0 时不再直接使用 0，而是修改为一个比较平滑的指数函数，这样修改的目的是保证在输入为负时模型也有一定的输出能力。

不过这样改动后并没有消除梯度爆炸的问题，还因为增加了指数运算而使求值时耗时更高。函数表达式和几何图像如下(α 通常取 0.1)：

$$f(x)=\begin{cases} x, & x>0 \\ \alpha(e^{x}-1), & x\le 0 \end{cases} \tag{3-4}$$

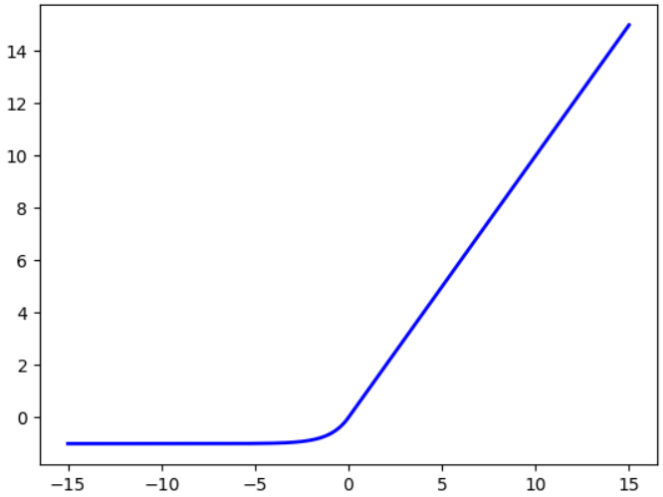


图 3.7　ELU 函数图像

4 AlexNet 卷积神经网络模型

4.1 模型提出背景

ImageNet 项目是一个由人工标记的大量图片组成的数据库，LSVRC 比赛使用 ImageNet 数据集中的一小部分数据，其目的是对这些标记的图片进行分类，从 2012 年到 2017 年，众多基于深度学习而设计的经典模型在该比赛中脱颖而出，推动了深度学习的发展。

2012 年，Alex Krizhevsky、Ilya Sutskever 在多伦多大学 Hinton 的实验室设计出了一个深层的卷积神经网络 AlexNet 模型[15]，一举夺得了 2012 年 ImageNet 大赛的冠军，该模型在比赛数据集上取得的 top-5 准确率远超第二名，在学术界引起了轩然大波。AlexNet 是一个非常有历史意义的模型，原因并不是该模型具有最强的性能和最高的识别率，而是从这以后，卷积神经网络开始席卷图像识别、语音识别等其他多年未有突破的行业，深度学习又一轮出现在大众面前。

4.2 模型特点

AlexNet 之所以能够取得如此高的准确率，跟这个模型设计的特点有关，相较于卷积神经网络的鼻祖模型 LeNet-5 来说，主要有以下几个特点：

1)、使用 ReLU 激活函数

传统的神经网络倾向于使用 Sigmoid 或 Tanh 函数等非线性函数作为激活函数，比如 LeNet-5 就使用 Sigmoid 函数，前面介绍过，这些函数容易在深层网络中出现梯度爆炸或梯度消失的现象。在 AlexNet 模型中，舍弃了非线性函数，将 ReLU 函数作为模型的激活函数，该函数是一个典型的线性函数，并且该函数求导后是一个固定的常量值，这样不仅使计算量大大下降，其模型的收敛速度也会比那些非线性模型更快。

2)、重叠池化

卷积神经网络的池化层通常起到减小特征图像尺寸、减少运算量的作用。一般使用最大值池化，并且池化的窗口不会重复，在实现时就是窗口大小与步长大小一致。在 AlexNet 模型中，池化操作的窗口却是重叠的，所有池化层的窗口尺寸均为 3*3，步长为 2，这样在运算过程中就会重复利用三分之一的参数，虽然增大了池化后输出图像的尺寸，加大了后序操作的运算量，但可以避免过拟合，防止模型因为失去泛化能力而降低识别率。

3)、局部归一化

局部归一化模仿神经生物学中侧抑制的概念，通过对某个局部区域使用归一

化，会使其中较大的值变得相对更大，而较小的值变得相对更小，这样较大值神经元的作用被放大，而抑制了没有实际用处的较小值的神经元。这样做在降低过拟合风险的同时有助于模型快速收敛，达到稳定，提高了泛化能力。

4）、Dropout

Dropout 是 AlexNet 模型中加入的独创性的概念，其主要目的是为了尽量防止模型出现过拟合的现象。Dropout 的本意为辍学者、退学者，在这的意思就是被丢弃的神经元。在神经网络中，依据某种事先约定好的概率模型将模型中某些参数的值设置为 0，这样这个节点就失去了作用，在计算时不会对模型产生任何影响，就好像死亡了一样。具体应该删除哪些神经元，由概率函数决定，通常选择最为简单的 0-1 概率模型，即设置一个 0-1 之间的值作为阈值，随机函数生成一个 0-1 之间的小数，根据这个小数的值在阈值的哪一侧对该节点进行取舍。被删除的节点只在本次运算过程中消失，下一次传播时将继续参与其中。

5）、多 GPU 训练

提出 AlexNet 模型时计算机算力比较弱，GPU 的运算能力也没有现在强，显存最大只有 3GB，因此模型的提出者采用了双 GPU 的训练模式，每个 GPU 只需要运行模型中神经元的一半，并且在某几层互相连接，构成一个整体的网络结构，这样的设计既满足了模型的整体结构，又弥补了硬件上的缺陷。

4.3 网络结构

Alexnet 作为一种卷积神经网络模型，必然与经典模型 LeNet-5 一样，具有典型的卷积层-池化层结构，二者的区别在于卷积层的层数和卷积核尺寸大小的不同。图 4.1 是 AlexNet 的完整网络结构图：

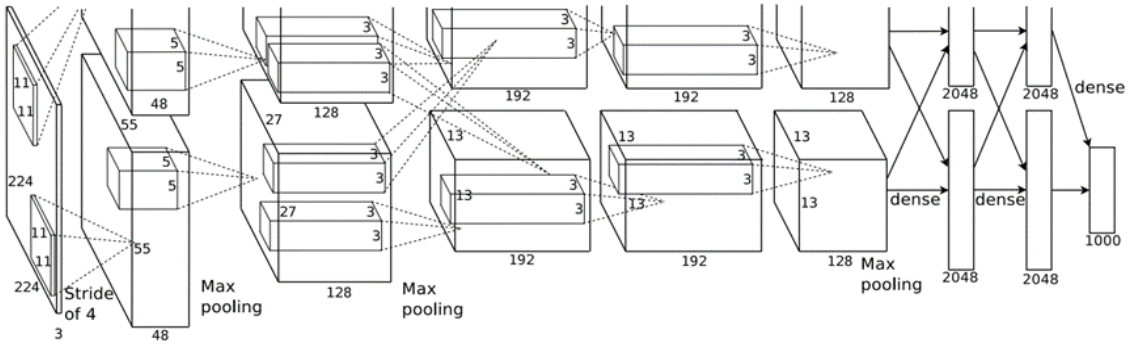


图 4.1　AlexNet 网络结构示意图

AlexNet 网络结构共有 8 层，前面 5 层是卷积层，后面 3 层是全连接层，最后一个全连接层的输出为一个包含 1000 个节点的向量，对应 1000 个类别的分布概率。AlexNet 采用两个 GPU 进行训练，整个模型对称的布置在两个 GPU 上，分别对应上图中的上下两部分，两个 GPU 只在特定的层相互连接。下面对 AlexNet 模型逐层进行分析。

第一层为卷积层，该层的处理流程有四个步骤。它的输入是整个模型结构的起点，输入为 227*227*3 的矩阵，是待识别的图像的三维像素值。卷积核为 96 个 11*11*3 的卷积核，步长 4，经过卷积运算后特征图尺寸为 55*55*96。接着经过激活层运算后到达池化层，池化层采用最大池化算法，窗口大小为 3*3，步长为 2，这样池化后特征图的尺寸变小，为 27*27*96。最后的步骤是进行局部归一化运算，归一化同样需要一个窗口尺寸，归一化操作只在这个窗口内进行，其尺寸为 5*5，经过运算后不会对原有的尺寸有任何改变。

第二层的处理流程与第一层完全一样，只是卷积核大小和卷积核个数不同，并且在卷积运算前多了一个填充的步骤，在特征图的四周填充上一些数据，方便后序步骤的计算。从上图可以直接看出该层卷积核的具体结构，经过第二层的运算后，输出的特征图尺寸为 13*13*128。

第三层与第四层具有相同的结构，虽然与前两层同为卷积层，但这两层少了池化和归一化这两个步骤，经过每层的填充和卷积运算后，第四层输出的特征图尺寸为 13*13*384，可以看出，特征图的尺寸基本维持不变，但个数却不断增加，表明在这几层中不断提取出原始图像中更多的特征。

第五层是卷积层的最后一层，该层又重新出现了池化层，主要目的是为了缩小特征图的尺寸，减少模型参数，并且方便与全连接层进行连接。该层卷积核尺寸依旧为 3*3，但是步长却发生了变化，每次卷积运算后的移动单位缩小为 1，池化层的窗口大小仍然是 3*3，步长为 2，该层运算结束后整个模型的卷积层全部结束，最后输出具有 256 个尺寸为 6*6 的特征图。

第六层到第八层是全连接层。第六层的输入为上层输出的 6*6*128 的特征图，因此首先需要通过卷积运算将其维度降为一维，使用 256 个卷积核，与第五层输出的特征图个数相匹配，同时尺寸也匹配为 6*6，经过这一步以后原来的二维特征图便转变为一维的神经元，第六层的输出为 1 维共 4096 个节点。

第七层和第八层是规范的全连接层，中间经过 Dropout 运算随机删除掉网络中一部分神经元，以减轻运算量，最后第八层的输出为整个模型的输出，该模型输出一个共计 1000 个节点的向量，代表 1000 分类的预测结果。

表 4-1 总结了 AlexNet 模型每层输入、输出尺寸和中间处理过程：

表 4-1　AlexNet 网络每层具体参数

AlexNet 网络模型					
卷积层					
卷积层	输入尺寸	填充后	卷积核尺寸	池化层尺寸	输出尺寸
Conv1	227*227*3	227*227*3	11*11*3, 96 个	3*3	27*27*96
Conv2	27*27*96	31*31*96	5*5*96, 256 个	3*3	13*13*256

Conv3	13*13*256	15*15*256	3*3*256, 384 个		13*13*384
Conv4	13*13*384	15*15*384	3*3*384, 384 个		13*13*384
Conv5	13*13*384	15*15*384	3*3*384, 256 个	3*3	6*6*256
全连接层					
全连接层	输入尺寸			输出节点个数	
Full6	6*6*256		卷积核：6*6*256, 1 个	4096	
Full7	4096			4096	
Full8	4096			1000(代表分类类别)	

5 在 Pytorch 框架上实现车型识别

5.1 Pytorch 简介

2017 年 1 月，由 Facebook 人工智能研究院基于已有的框架 Torch 推出了 Pytorch，它与 Tensorflow 一样是一个深度学习框架，主要帮助开发者迅速搭建和实现各种深度学习算法和模型，它的开发环境主要基于 Python，同时也支持嵌入 C++或者 Qt 平台，目前已经更新到 1.4 版本。与 Tensorflow 所提供的功能一样，该框架内置一套高效的数据存储结构和张量运算模块，同时支持所有机器学习算法和深度学习算法的构建和运行，以便帮助开发人员快速实现需求。

该框架具有两个非常突出的特点，其一是拥有非常高效的张量计算能力，支持 CPU 和 GPU 两种计算模式，并且配置为 GPU 模式时非常方便，其二是该框架内关于神经网络的模块具有自动求导和反向传播的功能，需要使用该功能时只需继承该模块中的类即可，该模块的名称叫 torch.nn，开发人员在使用该模块时，无需自己实现这部分功能。

Pytorch 几乎所有功能和模型都提供了对应对应的 C++、Qt 和 Python 接口，以供开发者选择，近年来广泛应用于图像处理、语音处理和自然语言处理等领域，尤其是那些需要在神经网络的基础上搭建的模型，研究人员编写很少代码便可以构建一个复杂的网络模型，并且加载数据和训练过程均非常容易实现，极大缩短了之前花在程序编写上的时间。

5.2 Pytorch 环境配置

本文所有程序的在同一个环境中运行运行，操作系统为微软的 Windows10 家庭版，python 版本为 3.7.3。此外，Pytorch 框架的版本为 1.4，CUDA 版本为 10.1。本文只介绍安装 Pytorch 及相关依赖。

1)安装 Pytorch

Pytorch 的安装方式有两种，一种是用 Python 内置的第三方库管理工具 pip 安装，另一种就是官网直接下载模块文件，再手动安装，下面介绍第二种安装方法。

首先进入 Pytorch 官网，下载两个核心的模块，一个是框架模块 torch，另一个是提供辅助功能的 torchvision 模块，如果电脑支持 GPU 训练的话，可以选择使用 CUDA，本文所使用的各个模块的版本基本都是最新版本。torch 版本为最新的稳定版本 1.4，torchvision 的版本为 0.5，CUDA 版本是 10.1。

全部下载完成后，对于 torch 和 torchvision 模块，直接将下载的 whl 后缀文件安装在 python 的第三方库文件夹内，如果电脑支持 GPU 训练，安装 CUDA 的驱

动程序即可。

2)测试 Pytorch

全部安装完成后，可以运行一段简单的代码来测试 Pytorch 是否安装成功。新建一个 python 脚本，输入测试代码如下：

```
import torch              # 导入 torch
  x = torch.rand(5,3)     # 建立一个随机数组成的张量
  print(x)                # 打印张量
```

如果能够成功运行，则说明已经成功安装 torch，接下来可以通过以下代码检测是否支持 GPU 计算，输入：

```
if torch.cuda.is_available():                  # 检测 cuda 是否可用
  print(torch.cuda.get_device_name())          # 打印 GPU 型号
```

这两行代码可以检测该计算机是否支持 GPU 计算，如果支持，会打印出具体的 GPU 型号。本文程序均运行于 GPU 上，对应的显卡型号为 MX150。

5.3 制作汽车数据集

汽车数据集是进行网络模型参数学习必不可少的数据，目前被广泛使用的汽车数据集有 BIT-VehicleID 汽车数据集、CompCars 汽车数据集等，这些数据集虽然图片数量庞大，但数据质量却不是太高。并且，这些数据集内包含了大量汽车局部图片和车内布局图片，这些图片对车型的识别会造成极大的干扰，并且在车型识别的实际场景中也不会起到任何帮助。

基于以上分析，本文单独制作了一份数据集，该数据集的汽车图片数量和质量均达到了训练模型所要求的标准，并且图片中汽车轮廓的完整性和质量均高于 CompCars 数据集。汽车分类所需的六种车型均来自可供下载的汽车网站，最终收集到 7530 张图片，每一类车型图片数量不低于 1000 张，且仅包含一辆车的全部轮廓。为了后序方便压缩数据集，需要用 Python 脚本对所有文件进行重命名，命名格式为分类标签+序号，这样汽车数据集便整理完成，方便进行后序处理。

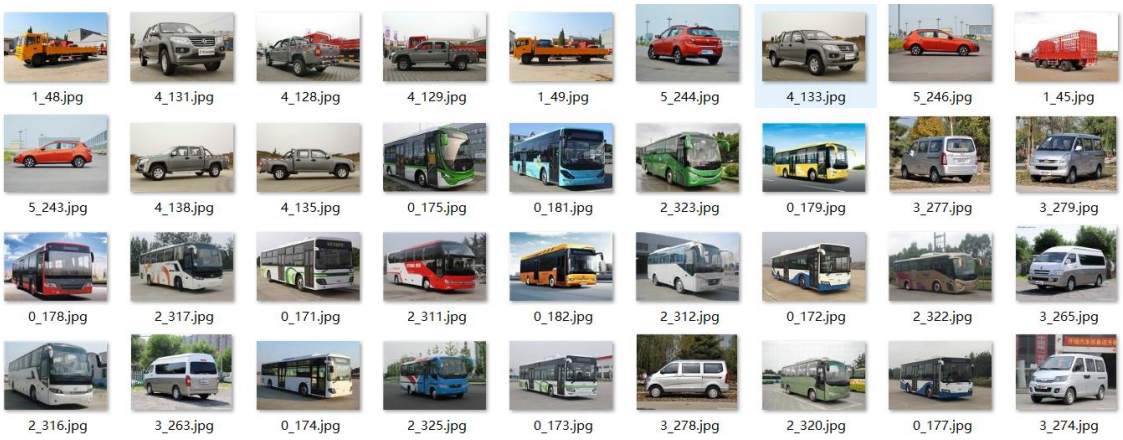


图 5.1　部分车型数据集

车型图片准备好后，需要将其制作成 Pytorch 方便读取和训练的格式。整理好的车型图片均为具有三个颜色通道的格式为 RGB 的图片，但图片尺寸不统一，这样无法统一进行训练，因此需要将其调整为 AlexNet 模型的输入尺寸，这个尺寸作为输入的固定值，宽和高都是 227 个像素，通过 python 脚本对所有图片的尺寸进行调整，方便与模型的输入尺寸相匹配。

同时，为了提高训练时加载数据集的速度和便于对数据进行分批训练，将所有图片的顺序全部打乱，分批保存为二进制文件，每一个二进制文件包含 512 张图片，每张图片在文件中的格式为：标签占第一个字节，接下来是 227*227*3 个字节的 RGB 像素数据，最后将其命名为 car+序号，并且以 car 作为数据集的后缀名。这样就将 7530 张图片重新制作成 15 个二进制文件，其中前 12 个文件共计 6144 张图片作为训练集，剩下的 3 个作为测试集。

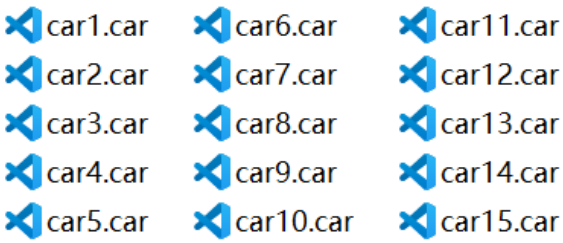


图 5.2　二进制文件示意图

数据集准备好后，需要编写相对应的加载数据集的模块，Pytorch 提供了一个存放数据集的类 Dataset，这个类从数据集文件中提取数据，然后重新组织，将标签与数据分开，然后使用 DataLoader 函数对数据进行一些处理，主要为了对数据进行分批，打乱顺序、开启多线程等准备工作。

Pytorch 内置一些公开数据集的加载类，当加载这些数据集时无需自己编写函数，直接调用即可，但是对于自定义的数据集，需要新建一个数据集类，该类继承自 Pytorch 内置的 Dataset 类，并且重写内部成员函数__len__和__getitem__,以达到能够以下标访问某个数据和遍历数据的目的。最后，将自定义的类作为参数传递给 DataLoader 函数，这样数据集就能成功加载了。

5.4　实现 AlexNet 模型

Pytorch 中的 torchvision 库内有一个模块，该模块实现了一些被广泛使用的卷积神经网络模型，AlexNet 模型的实现就在其中，在实现模型的时候可以参考这个官方给出的实现代码。

实现模型时需要新建自己的模型类，该类必须继承自内置类 torch.nn.Module，这样才会具有自动计算梯度和反向传播的功能，接着在自己定义的模型类内实现模型具体的结构和前向传播函数 forward。

Pytorch 内集成了反向传播的算法，在实现模型时既不用实现反向传播的算法，

也不用考虑求导计算梯度的实现，只需把精力集中在模型的结构上。由于作者的运行环境只有一个 GPU，因此只实现了模型的上一半，并且适当调整了每层卷积核的个数，以此来平衡损失的另一半卷积核，这样做不会对原模型的性能产生太大影响，Pytorch 官方给出的模型例程同样也采用了这种折中的办法。模型定义如下：

```
class AlexNet(torch.nn.Module):
    def __init__(self):
        super(AlexNet, self).__init__()

        self.convnet = nn.Sequential(OrderedDict([
            ('C1   ', nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2)),
            ('ReLU1', nn.ReLU(inplace=True)),
            ('S1   ', nn.MaxPool2d(kernel_size=3, stride=2)),

            ('C2   ', nn.Conv2d(64, 192, kernel_size=5, padding=2)),
            ('ReLU3', nn.ReLU(inplace=True)),
            ('S2   ', nn.MaxPool2d(kernel_size=3, stride=2)),

            ('C3   ', nn.Conv2d(192, 384, kernel_size=3, padding=1)),
            ('ReLU3', nn.ReLU(inplace=True)),
            ('C4   ', nn.Conv2d(384, 256, kernel_size=3, padding=1)),
            ('ReLU4', nn.ReLU(inplace=True)),
            ('C5   ', nn.Conv2d(256, 256, kernel_size=3, padding=1)),
            ('ReLU5', nn.ReLU(inplace=True)),
            ('S5   ', nn.MaxPool2d(kernel_size=3, stride=2)),

            ('Avg6 ', nn.AdaptiveAvgPool2d((6, 6))),
            ('Drop6', nn.Dropout()),
        ]))

        self.fc = nn.Sequential(OrderedDict([
            ('F7   ', nn.Linear(256*6*6, 1024)),
            ('ReLU7', nn.ReLU(inplace=True)),
            ('Drop7', nn.Dropout()),
            ('F8   ', nn.Linear(1024, 6)),
        ]))

    def forward(self, x):
        x = self.convnet(x)
        x = torch.flatten(x, 1)
        x = self.fc(x)
        return x
```

图 5.3　AlexNet 模型

分析这段程序，self.convnet 函数用来实现模型的前五层，即卷积层，self.fc 函数实现了模型的后三层，最后('F8　　', nn.Linear(1024, 6))这段代码为模型的第八层

全连接层，同时也是输出层，最终输出具有 6 个神经元的向量，用来表示最终的预测结果。

5.5 训练数据集

AlexNet 模型的训练过程主要分为五步：载入数据、设置损失函数和梯度优化函数、将数据载入模型进行前向传播计算、通过损失函数得到误差值、反向传播更新模型内的参数。

1)载入数据集

训练数据集由 12 个二进制文件组成，每个文件包含 512 张图片，训练时每次加载一个文件，12 个文件全部加载并训练后代表一轮训练的完成，每个文件的训练分为两个批次，每个批次 256 张图片。这个步骤主要由 Pytorch 内的 DataLoader 函数来完成，该函数将前面提到的自定义数据集加载类作为输入，可以设置每个批次要训练的图片个数、是否打乱图片顺序、是否开启多线程加速等功能，函数返回值是一个可迭代的图像张量和标签组成的元组，每个批次作为一个单独的迭代对象，即一次加载 256*3*227*227 的张量作为模型训练时的输入，256*1 作为对应的标签。

2)设置损失函数和梯度优化函数

上文介绍过，在图片分类领域中通常使用交叉熵来衡量预测值与真实值之间的误差，Pytorch 内包含了用来对比预测值与真实值之间误差的损失函数，torch.nn.CrossEntropyLoss 函数就是其中之一。至于梯度优化函数，Pytorch 同样内置了一系列在反向传播过程中对参数进行优化的函数，比较常用的是 Adam 优化，其对应的函数为 torch.optim.Adam，在训练过程中，还有一个参数值得关注，那就是学习率，通常学习率一般设置为 $1x10^{-3}$，可根据具体情况进行调整。至此，模型训练前的准备工作就全部完成了。

3)前向传播

前向传播的运算过程通过 AlexNet 类的 forward 函数来实现，训练时，将一个批次的数据输入模型，经过模型内的网络结构运算后，输出一个预测值组成的张量，由于每个批次包含 256 张图像，因此会输出 256 个预测值，分别对应输入的 256 张图像。

4)计算误差

经过前向传播运算后，会计算出模型得到的预测值与真实值之间的差异关系，这个差异性用交叉熵来表示，这个步骤直接使用前面已经定义好的交叉熵损失函数，得到的结果是 256 张图像各自的交叉熵。

5)反向传播

反向传播是模型训练的关键之处，模型最终的参数是通过不断对误差进行反

向传播计算梯度得到的。Pytorch 中通过梯度清零、反向传播、更新参数这三个步骤进行参数更新。梯度清零是为了清除上次反向传播时计算出来的梯度，为这次反向传播做准备。反向传播即采用前面设置的梯度优化函数进行梯度计算，反向传播的过程是从最后一层逐层向前一层计算梯度，采用的原理是一个函数往往在梯度最大的地方下降最快，这样可以用最快的速度降低误差。最后一个步骤是更新参数，把网络模型中反向传播所优化的参数全部进行更新。

整个过程全部完成后，一个数据集文件的训练结束，接下来还需要把剩下的 11 个文件依次加载，全部文件训练结束后代表数据集进行了一轮训练，通常情况下，一个模型要经过多轮训练才能达到相对稳定的准确率，轮数太少，模型可能还没有达到最好的效果，轮数太多，学到的特征可能具有针对性，失去了泛化能力，出现过拟合而降低准确率的现象。网络模型训练的关键代码如下：

```
def train(model, epoch):  #model是Alexnet模型,epoch是训练轮数
BATCHSIZE = 256    #每个批次训练的图像个数
model.train()       #模型设置为训练模式
learning_rate = 1e-3  #学习率设为1e-3
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)  #设置梯度优化函数
criterion = torch.nn.CrossEntropyLoss().to(device)  #设置损失函数为交叉熵损失函数
loss_list = []
for i in range(epoch):    #训练轮数
   for j in range(12):   #12个数据集文件为一轮
      # data load
      filename = trainfilename + str(j+1) + '.car'
      print(filename)
      data_train = CarDataset( filename, train=True)  #数据集
      data_train_loader = DataLoader(data_train, batch_size=BATCHSIZE, shuffle=True)
      for k,(img,lab) in enumerate(data_train_loader):
         img = img.to(device)
         lab = lab.to(device)
         output = model(img)
         loss = criterion(output, lab)
         loss_n = loss.detach().cpu().item()
         print('Train - Epoch %d, Batch: %d, Num: %d, Loss: %f' % (i+1, k, k*BATCHSIZE, loss_n))
         loss_list.append(loss_n)
         optimizer.zero_grad()
         loss.backward()
         optimizer.step()

plt.plot(loss_list)
plt.show()
```

图 5.4　模型训练代码

6 车型识别结果

模型训练完成后，Pytorch 有一套用于将模型永久化的函数，保存形式为二进制文件，保存后的文件包含整个模型结构和所有参数，用测试集数据来验证模型的识别准确率时，只需将保存的模型加载并将测试集的数据输入进模型，对比模型的预测值和真实值是否相同即可。以下是模型训练时的部分中间结果：

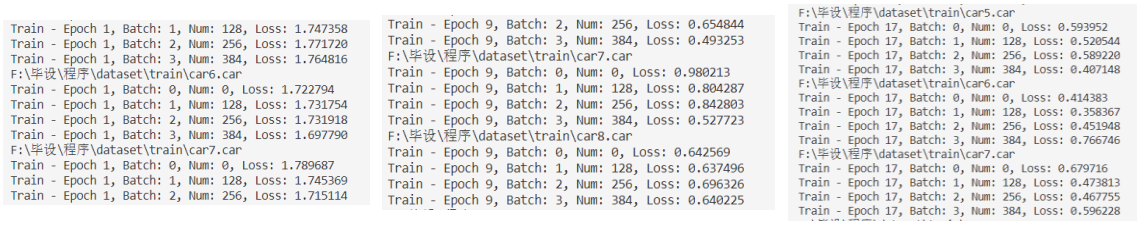


图 6.1　训练过程示意图

从中可以看出，经过一批批数据的训练，损失值在逐步下降，在训练过程中，发现训练的不同轮数对准确率的高低有一定的影响，图 6.2 展示了前 300 轮训练中模型在测试集上的识别准确率：

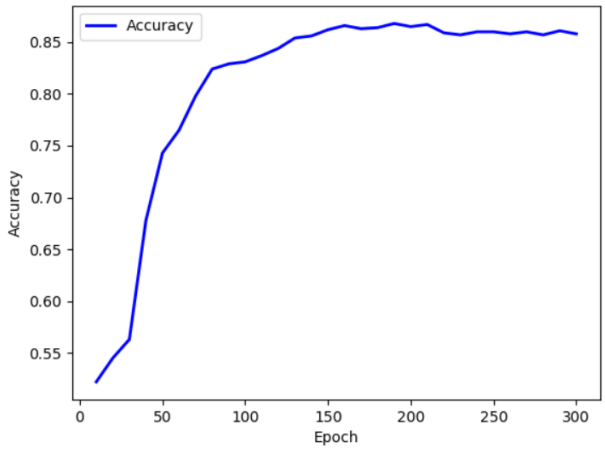


图 6.2　识别准确率

由图中数据可知，在前 150 轮训练过程中，训练好的模型在测试集上的表现越来越好，识别的准确率在前 100 轮快速增长，100 轮到 150 轮的准确率依旧持续增长，说明模型还没有达到稳定，在第 160 轮时达到了最高值 86.9%。在接下来的 150 轮训练中，模型的识别准确率维持在了 86%左右，说明模型趋于稳定，网络对汽车特征的提取能力已经基本达到饱和，再继续增加训练轮数不会对模型有实质性的改进，甚至会出现过拟合的现象，使得模型丧失一定的泛化能力。

为了更方便的对车型图片进行识别，作者开发了一个车型识别的窗口应用程序，该程序可手动选择待识别的车型图片，输入模型识别后输出识别结果，这样能

够更方便更直观的验证模型的识别效果。启动程序后进入一个简易的 GUI 界面，这个界面只有两个按钮，一个是选择图片按钮，另一个则是识别按钮。点击界面中的选择图片按钮，会弹出一个能够选择文件的选项框，从文件夹中选择一张准备识别的车型图片，并且将选择的图片路径显示在控制台上，然后点击界面下方的开始识别按钮，程序会自动加载训练好的模型，对已选择的图片进行识别，最终在控制台输出预测的类别。通过这个程序可以方便的将车型识别应用在实际场景中，图 6.3-a 和 6.3-b 展示了从网络中随意下载几张车型图片并进行识别的效果：

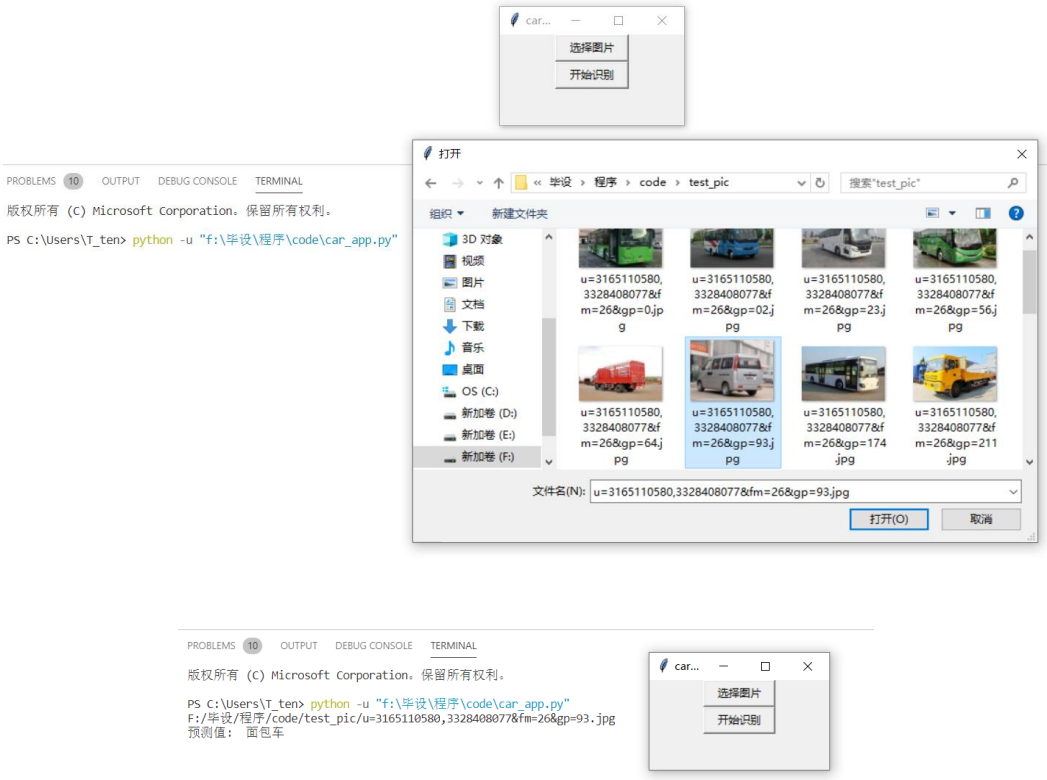
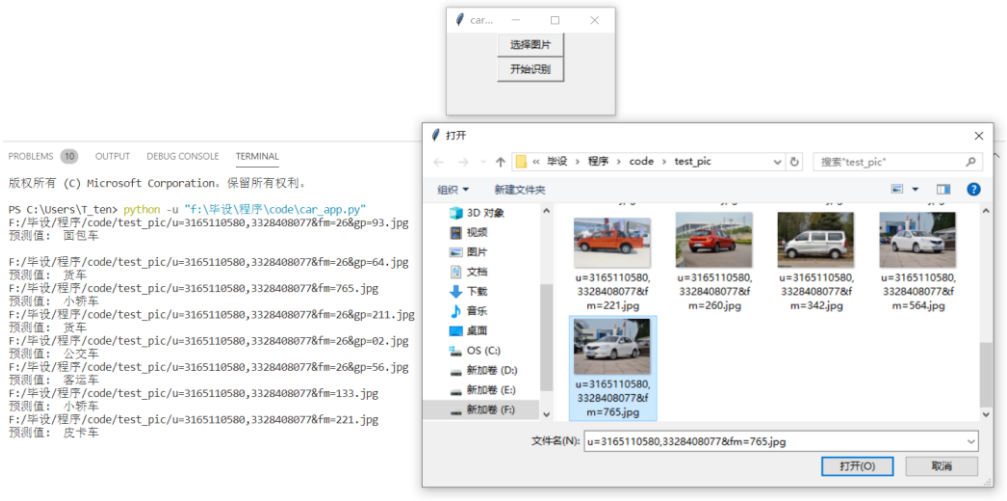


图 6.3-a　车型识别示意图一



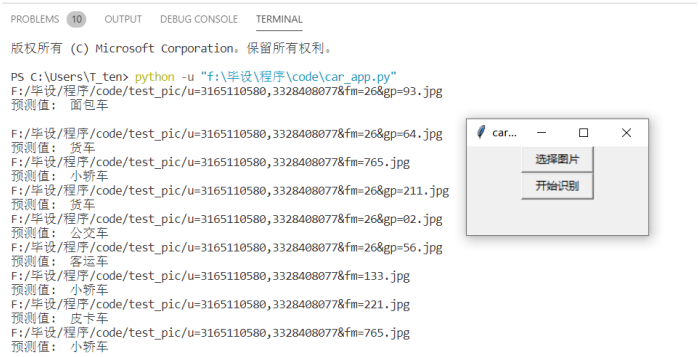


图 6.3-b　车型识别示意图二

识别结果显示，除了其中一辆皮卡车被误识别为小轿车外，其他车型识别均是正确的，说明训练出的模型具有非常高的识别准确率，86%的准确率基本可以满足大部分需要车型识别的场景，并且识别速度极快，识别一张图片的时间在 10 毫秒左右。

7 模型改进

AlexNet 网络原本是为识别 ImageNet 数据集上的 1000 种不同物体而设计的，所以该模型的特征提取能力非常强，比如中间几层卷积核个数达到了 384 个之多，最后的全连接层的节点数更是达到了 4096，这样导致模型的参数千万级别，运算量非常大，对于训练只有 6 个车型的小型数据集太过浪费。而且，车型识别只是一个 6 分类问题，不需要全连接层中上千的神经元，这样有可能会过度提取特征，而导致识别准确率的下降，因此，将模型进行适当改进可能会提高识别的准确率。

为了分别研究卷积层卷积核尺寸和全连接层神经元个数对车型识别率的影响，采用先修改单一变量的方法对这两部分的相应结构进行改变，再将二部分综合起来进行多个位置参数的改变，最终的改进情况和结果如表 7-1 所示：

表 7-1 模型不同改进方案

模型改进				训练轮数	识别准确率
卷积层		全连接层			
位置	改进值	位置	改进值		
Conv3	去掉该层			200	83.5%
Conv3	卷积核个数改为 192			200	84.3%
Conv3	卷积核大小改为 7*7			200	86.3%
Conv5	卷积核个数改为 512			200	84.6%
		Full7	512	200	87.3%
		Full7	256	200	88.3%
		Full7	128	200	84.7%
		Full7	64	200	84.6%
Conv3	去掉该层	Full7	256	200	83.9%
Conv3	卷积核个数改为 192	Full7	256	200	86.0%
Conv3	卷积核大小改为 7*7	Full7	256	200	87.1%

从表中可以看出，前四条改进方案针对卷积层中的卷积核个数和尺寸进行调整，中间四条方案针对全连接层中的第七层节点个数进行调整，最后三组则将两部分的调整综合起来，用这种改进方式可以仔细分析具体的方案对识别准确率的影响。

分析表中数据，将第三层卷积层中卷积核的个数由 384 调整为 192 之后，在测试集上的准确率并没有提高，反而下降到了 84.3%，这说明卷积核个数减少后提取到的特征变少，没有完全覆盖到车型的所有特征，特征数量的降低导致刻画车型

的程度不够，识别率随之下降。第三条改进方案将卷积核大小改为 7*7，比原本的 3*3 升高到了 7*7，但是准确率却没有太大的变化，没有明显的改善效果，还无故增加了大量需要训练的参数，说明模型本身的卷积核大小已经是挑选出的较优大小了。第四条方案修改了第五层卷积层，没有什么效果，不做分析。第五条开始，对第七层全连接层的节点进行了调整，从原来的 1024 直接降到 512，之后每次下降到原来的一半，以此观察节点个数对识别率的影响，发现在节点值降到 256 时模型的准确度得到了最大值，达到了之前所没有的 87.9%，这表明全连接层的节点个数太多可能引起了过拟合的发生，影响到了识别的效果，将节点减少后缓解了这个问题。

最后三个方案是综合起来考虑后的调整方案，虽然最后两个均超过了原始的模型，但是均没有达到单独调整全连接层时的准确率，说明对卷积层的调整并不是关键的改进方向，并不会对原有模型产生实质性的变化，全连接层的节点个数才是起到作用的方面。分析产生这种现象的原因，猜测可能是因为 AlexNet 作为参赛的模型，当初是用来做 1000 分类任务的，自然需要上千个全连接层节点，如今只有 6 个分类，没必要设置这么多节点。

致　　谢

四年的大学生涯即将结束，很快我也将和身边众多的同学一道走出校园，开始新的人生阶段。此刻回首，我发现自己在这过去的几年时间里确实成长了许多，从一个青涩稚嫩的学生变成了一个具有一定专业能力和生存能力的成年人，知识技能日益丰富，视野心胸愈发开阔，世界观、人生观和价值观都变得更加成熟。这些都是我的同学、朋友、老师以及亲爱的母校赐予我的！

在完成毕设的这半年里，从选题到开题，从疫情期间在家完成程序的编写，到在校完成论文的撰写，中间有自己的努力与付出，有同学朋友的帮助和鼓励，更有着导师全心全力的指导和监督。

首先，我要感谢我的导师——项铁铭老师。项老师知识渊博、技能精湛、视野宽广，他对自己方向上的学术研究充满了热忱，与专业同行交流密切，对专业发展的方向把握准确。在完成毕设期间，我在项老师的悉心点拨下，才能够顺利完成毕业设计，做出令自己满意的成果。

其次，我要感谢我身边朝夕相处的同学和朋友，大家团结友爱，互帮互助，一同走过了大学这四年宝贵的学习时光。

最后，我必须要感谢亲爱的母校——杭州电子科技大学。四年前，是您给了我在宽敞明亮的教室里用心学习；在设备先进的实验室中专注工作；在鸟语花香、生机勃勃的校园间惬意徜徉的机会。在杭电的时光已经成为了我人生发展的重要拐点，您让我走上了一条诸多杭电优秀前辈曾经走过的道路，也必将在我今后的人生岁月中不断地激励我前进。

我在此衷心希望母校今后能够越来越棒，为祖国的发展培养越来越多杰出的人才，让我等毕业生能够以一名杭电人的身份骄傲的自居于各自的工作岗位上！

参考文献

[1]国家统计局.中华人民共和国 2019 年国民经济和社会发展统计公报[EB/OL].http://www.stats.gov.cn/tjsj/zxfb/202002/t20200228_1728913.html,2020-02-28.

[2]中国汽车协会行业信息部.2019 年汽车工业经济运行情况概览[EB/OL].http://www.caam.org.cn/chn/3/cate_19/con_5228367.html,2020-01-13.

[3]樊海泉,董德存.基于模式匹配算法的车型识[J].微型电脑应用,2002,04:20-21.

[4]蔡智湘.基于车辆外形几何特征的车型自动分类器的研究[D].北京:中国农业大学,2004.

[5]张伟.用小波变换提取车型特征实现车型分类[J].交通部上海船舶运输科学研究所学报,2002,02:94-99.

[6]孙志军,薛雷,许阳明,等.深度学习研究综述[J].计算机应用研究,2012,29(8):2806-2810.

[7]贾瑞.基于 AlexNet 的车辆型号识别研究[J].现代工业经济和信息化,2018,12(01):1-4.

[8]石维康.基于深度神经网络的车型识别设计与实现[D].西安:西安电子科技大学,2018.

[9]黄强强.基于深度学习的车辆属性识别研究[D].焦作:河南理工大学,2017.

[10]楚翔宇.基于深度学习的交通视频检测及车型分类研究[D].哈尔滨:哈尔滨工业大学,2017.

[11]Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 02:1097-1105.

[12]Gu J, Wang, etc. Recent advances in convolutional neural networks[C]. arXiv preprint. A:arXiv, 2015. 1512-1518.

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

[14]LeCun Y, Bottou L, Bengio Y. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.

[15]Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv, 2014, arVix:1409-1556.

[16]Szegedy C, Liu W, Jia Y. Going deeper with convolutios[J]. Cvpr, 2015, arVix:2105-

2175.
[17]He K, Zhang X, Ren S, etc. Deep residual learning for image recognition[J]. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, 02:770-778.
[18]左雨婷.基于深度学习的车型识别的研究与应用[D].北京:北京邮电大学,2019.
[19]傅云翔.面向多姿态车辆的型号识别方法研究[D].合肥:合肥工业大学,2019.
[20]张志永.监控场景下车型识别与检测算法研究[D].郑州:郑州大学,2019.
[21]代乾龙.基于深度学习的车型识别研究与应用[D].徐州:中国矿业大学,2019.