河南理工大学

本科毕业设计（论文）

题 目 基于深度学习的盒装药品检测

学院名称 计算机科学与技术学院

专业名称 计算机科学与技术

年级班级 计算机 1604 班

学生姓名 张宇阳

指导教师 芦碧波

2020 年 5 月

摘要

2020 年的伊始，突如其来的疫情让原本喧嚣的城市陷入了看不到尽头的宁静，也改变了生活本来的样子。在这个科技飞速发展的时代，应对疫情也有了很多创新的方式，例如在采购生活用品时，移动支付代替了现金交易，尽可能减少一切不必要接触，很多的超市，购物广场都有无人的自助结账通道。但是，在很多药店，依旧采用较为传统的柜台收银员结账，效率较低。因此设计一种高效药品识别方法有着重要的意义。

随着人工智能技术的不断发展，计算机算力不断提高，卷积神经网络逐渐被应用到图像分类问题中，深度学习算法可以直接将原始图像作为训练数据输入，不需要人们根据经验设计特征，就可以全面高效的提取图像颜色，形状等特征。

本文主要通过对深度学习的图像分类、损失函数的研究设计一种准确率高，低成本的盒装药品识别算法。主要内容如下：

首先概述了神经网络的发展以及传统的图像分类算法，发现传统图像分类算法对操作者图像分类经验要求过高，而构建深度学习模型解决图像分类问题更便于实践操作，然后简单概述了 LeNet，AlexNet，GoogleNet 以及 Vgg 网络模型，并就其结构作了简要分析发现 Vgg 网络模型更适合解决本课题研究内容。

最后利用 PaddlePaddle 实现 Vgg 网络模型：首先构建一个包含 13 类盒装药品的数据集，并利用 PIL 库对数据集进行适当扩充最后得到 13520 张图像，按照 9 ：1 的比例随机划分为训练集和测试集。然后利用飞桨（PaddlePaddle）深度学习框架构建 vgg 网络模型，并选用交叉熵损失函数和 Adam 调优算法。实验结果表明，图像分类算法能准确识别多种盒装药品，准确率达 90%以上，基本满足实际应用中的要求。

关键词：图像分类；深度学习；卷积神经网络；损失函数

ABSTRACT

At the beginning of 2020, the sudden epidemic sent the originally noisy city into tranquility where no end could be seen, and changed the way life was originally. In this era of rapid development of technology, there are many innovative ways to deal with the epidemic. For example, when purchasing daily necessities, mobile payment replaces cash transactions to minimize all unnecessary contact. Many supermarkets and shopping malls have no people. Self-checkout channel. However, in many pharmacies, more traditional counter cashiers are still used for checkout, which is inefficient. Therefore, it is of great significance to design an efficient drug identification method.

With the continuous development of artificial intelligence technology and the continuous improvement of computer computing power, convolutional neural networks are gradually applied to image classification problems. Deep learning algorithms can directly input the original image as training data without requiring people to design features based on experience. It can comprehensively and efficiently extract image color, shape and other features.

This paper mainly designs a high accuracy and low cost boxed medicine recognition algorithm through the research on image classification and loss function of deep learning. The main contents are as follows:

First, construct a data set containing 13 types of boxed medicines, and use the PIL library to appropriately expand the data set to obtain 13,520 images, which are randomly divided into a training set and a test set according to a 9: 1 ratio. Then use the PaddlePaddle deep learning framework to build a VGG network model, and select the appropriate activation function and tuning algorithm. The experimental results show that the image classification algorithm can accurately identify a variety of boxed medicines with an accuracy rate of more than 90%, which basically meets the requirements in practical applications.

Key words: Image classification, deep learning, convolutional neural network, loss function

目录

1.前言

1.1 课题背景

近些年，随着‘互联网+’计划的不断推进，互联网与传统行业不断碰撞，结合，扩展了互联网技术，计算机技术的应用场景，在很大程度上打破了人们对一些传统产业的认知，解放了人们的思维，计算机科学倍受关注，也加快了计算机技术到生产生活的应用进程。在工业 4.0 的大背景下，‘智能’被全社会高度重视，而人工智能越来越多的被提及到，应用到，人工智能技术逐渐渗透到人们的日常生活中。

随着智能扫地机器人，智能家居，无人商店的出现，人工智能在我们的日常生活中扮演着越来越重要的角色，它极大的方便了我们的日常生活，给予人们对未来生活无限的想象，也一点点的改变着这个世界，赋予它以便捷，以美好。

但当时间来到 2020 年的 1 月 23 日，新冠病毒爆发，武汉封城，全国各地纷纷采取措施，人们不得不在家隔离，尽可能减少与外界的接触。但是由于日常生活的需要，人们又不得不出门采购必要的生活物品，我们不得不思考如何降低日常采购时的风险。

而在这次疫情中，口罩，消毒液，预防药物供不求，因此，在传统药店，排着长队付钱成为了很常见的现象，那该如何降低人们购买药品时被感染的风险。无人商店给我们提供了一种解决思路，优化药店结账付款方式。在这次疫情中，商家普遍利用支付宝，微信等移动支付，如果可以实现药品的自动识别，那么结合移动支付可以很好的代替收营员，一方面减少了消费者与服务人员的接触，另一方面也极大的减少了顾客结账付款时间，优化消费者购物体验，对于药店来说，也可以减少日益增加的人工开支，提高药店效益。在这个解决方案中，最核心的就是药品自动识别技术，如何获取药品的相关数据信息（药品名称，价格等）极其关键。如果药品识别不准确，会大大降低消费者的购物体验，直接影响消费者的购物情绪。因此就需要识别准确率高，识别速率快，自动化高的商品自动识别技术。

1.2 研究意义及目的

1.2.1 研究意义

本课题研究意义在于，减少消费者与店员之间的接触，降低疫情中人们在购

买商品时的感染风险，满足消费者高效，便捷的购物体验，同时也为商家提供一种降低人工成本，提供店铺效益的商品自动识别方案。

1.2.2 研究目的

通过深度学习的图像分类技术实现一种分类准确率高，识别速度快的盒装药品自动识别方案。

1.3 盒装药品识别研究现状

目前药品识别技术主要采用条形码，无线射频技术以及基于人工智能的图像处理技术。

20 世纪 50 年代初，伯纳德·塞尔沃和约瑟夫·伍德兰德成功完成了条形码阅读器操作实验，实现了条形码的制造与识读。20 世纪 60 年代，随着激光技术的出现和 1960 年代计算机技术的发展，计算机可以读取，显示和处理条形码信息。主要原理是条形码符号是由“条”和“空”组成的信息符号，根据某些编码规则具有不同的反射率。由于条形码符号“条”和“空”具有不同的光反射率，因此条形码扫描仪会接收强度和强度不同的反射光信号，并因此发送不同电位的电脉冲。产生。条形码符号的“条”和“空”宽度确定具有不同电位水平的电脉冲信号的长度。扫描仪接收到的光信号必须通过光电转换转换为电信号，并由放大器电路放大。由于扫描光的光斑具有一定的大小，因此由电路放大的条形码的电信号是平滑的起伏信号，因为在打印条形码时边缘模糊。它称为“信号”。有必要将“模拟信号”成形为正常的“数字信号”。根据编码系统的编码规则，解码器可以将“数字信号”转换为数字和文本信息。条形码识别技术在现实生活中应用广泛，但是由于他完全依赖与条形码和条形码识别器，需要操作者有一定的使用经验，日常使用成本高，效率低。

无线射频是 1990 年代出现的非接触式自动识别技术，与传统的磁卡和智能卡技术相比，射频技术具有非接触，读取速度快，无磨损的特点。无线射频技术在读取器和射频卡之间执行非接触式双向数据传输，以达到目标识别和数据交换的目的。RFID 技术的基本工作原理并不复杂。标签进入磁场后，它接收从阅读器发送的射频信号，并使用感应电流获得的能量来存储芯片中存储的产品信息已发送。它主动发送特定频率的信号，读取器读取并解码信息后，会将其发送到中央信息系统以进行相关数据处理。完整的 RFID 系统包括三个部分：阅读器（读取器）和电子标签（TAG），所谓的应答器（transponder）和应用软件系统。无线电波的能量被发送到应答器，后者驱动应答器电路并发出内部数据。

人工智能处理图像技术目前应用于谷歌无人商店，主要涉及图像识别，目标

跟踪技术，基本实现即拿即走，但对于硬件设施有着极高的要求，配置成本较高，其算法实现也具有一定的复杂性，并不适合中小规模药店。但随着人工智能技术，尤其是深度学习技术的发展，商品识别技术不断发展。深度学习模型具有强大的学习能力，使我们能够提取图像的整体特征，并结合各个单元之间的联系，以真正找到并表征问题内部的复杂结构特征。在许多情况下，图像信息丰富，图像集具有非常大的数据量，深度学习算法是通用的，并且可以通过学习最大化并解决大数据的优势。用于训练的数据越多，算法的健壮性和泛化能力越高，从而使图像识别的准确率大幅提高，并且能够降低应用成本。

1.4 深度学习图像分类概述

深度学习的概念起源于人工神经网络的研究，具有多个隐藏层的多层感知器是一种深度学习结构。深度学习结合了低级特征以形成更抽象的高级表达属性类别或特征，以发现数据的分布式功能表示。可以利用深度学习模仿人脑对图像的识别以及特征提取，从而进行图像类别的判断，取代了传统的较为繁琐的图像特征提取方式，深度学习方法在图像分类中表现效果很好。

深度学习(DL, Deep Learning)是机器学习(ML, Machine Learning)领域中一个新的研究方向，它被引入机器学习使其更接近于最初的目标——人工智能(AI, Artificial Intelligence)。深度学习是学习样本数据的内在规律和表示层次，这些学习过程中获得的信息对诸如文字，图像和声音等数据的解释有很大的帮助。它的最终目标是让机器能够像人一样具有分析学习能力，能够识别文字、图像和声音等数据。深度学习是一个复杂的机器学习算法，在语音和图像识别方面取得的效果，远远超过先前相关技术。深度学习在搜索技术，数据挖掘，机器学习，机器翻译，自然语言处理，多媒体学习，语音，推荐和个性化技术，以及其他相关领域都取得了很多成果。深度学习使机器模仿视听和思考等人类的活动，解决了很多复杂的模式识别难题，使得人工智能相关技术取得了很大进步。

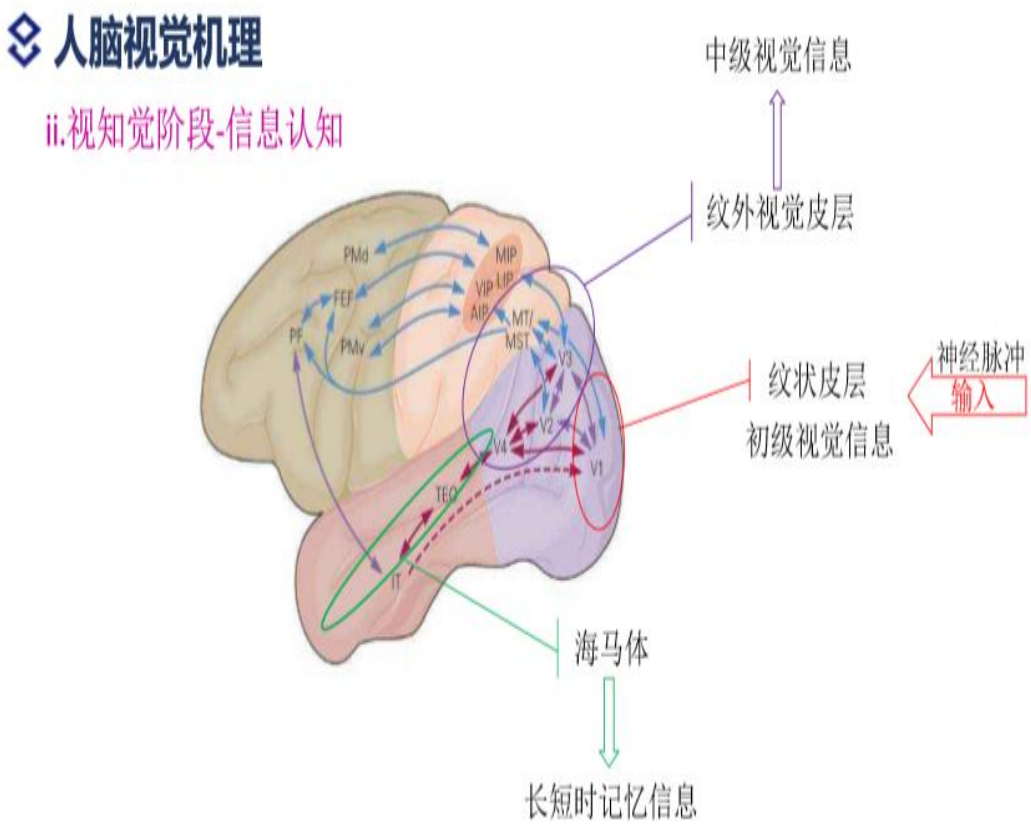


图 1-1 人脑视觉肌理图

通过多层处理，逐渐将初始的“低层”特征表示转化为“高层”特征表示后，用“简单模型”即可完成复杂的分类等学习任务。由此可将深度学习理解为进行“特征学习”（feature learning）或“表示学习”（representation learning）。以往在机器学习用于现实任务时，描述样本的特征通常需由人类专家来设计，这成为“特征工程”（feature engineering）。众所周知，特征的好坏对泛化性能有至关重要的影响，人类专家设计出好特征也并非易事；特征学习（表征学习）则通过机器学习技术自身来产生好特征，这使机器学习向“全自动数据分析”又前进了一步。近年来，研究人员也逐渐将这几类方法结合起来，如对原本是以有监督学习为基础的卷积神经网络结合自编码神经网络进行无监督的预训练，从而利用鉴别信息微调网络参数形成的卷积深度置信网络。

可以利用深度学习模仿人脑对图像的识别以及特征提取,从而进行图像类别的判断,从而取代了传统的较为繁琐的图像特征提取方式,而深度学习方法在图像分类中表现效果很好。

2012 年,Alex 等人提出的 AlexNet 网络在 ImageNet 大赛上以远超第二名的成绩夺冠,。卷积神经网络乃至深度学习引起了广泛的关注。AlexNet 使用 relu 作

为 CNN 的激活函数,解决了 sigmoid 在网络较深时的梯度弥散问题。训练时使用 Dropout 随机丢掉一部分神经元,避免了模型过拟合。网络中使用重叠的最大池化代替了此前 CNN 中普遍使用的平均池化,避免了平均池化的模糊效果,提升了特征的丰富性。从某种意义上说,AlexNet 引爆了神经网络的研究与应用热潮。

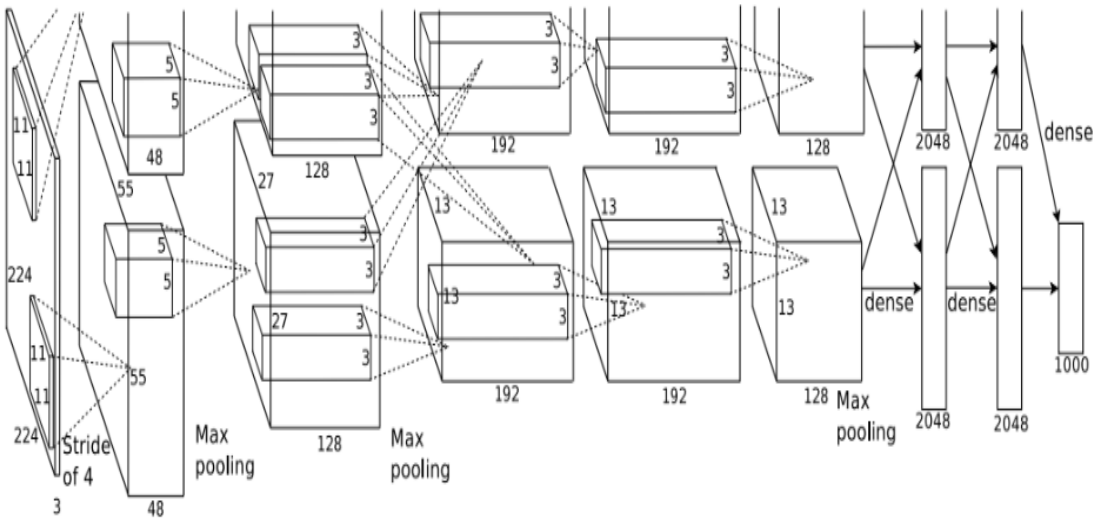


图 1-2 AlexNet 模型结构图

GoogLeNet 是 2014 年由 Google 设计的一种新的神经网络结构,其与 VGG 网络并列成为当年 ImageNet 挑战赛的双雄。GoogLeNet 首次引入 Inception 结构,在网络中堆叠该结构使得网络层数达到了 22 层,这也是卷积网络首次超过 20 层的标志。由于在 Inception 结构中使用了 1x1 的卷积用于通道数降维,并且使用了 Global-pooling 代替传统的多 fc 层加工特征的方式,最终的 GoogLeNet 网络的 FLOPS 和参数量远小于 VGG 网络,成为当时神经网络设计的一道亮丽风景线。

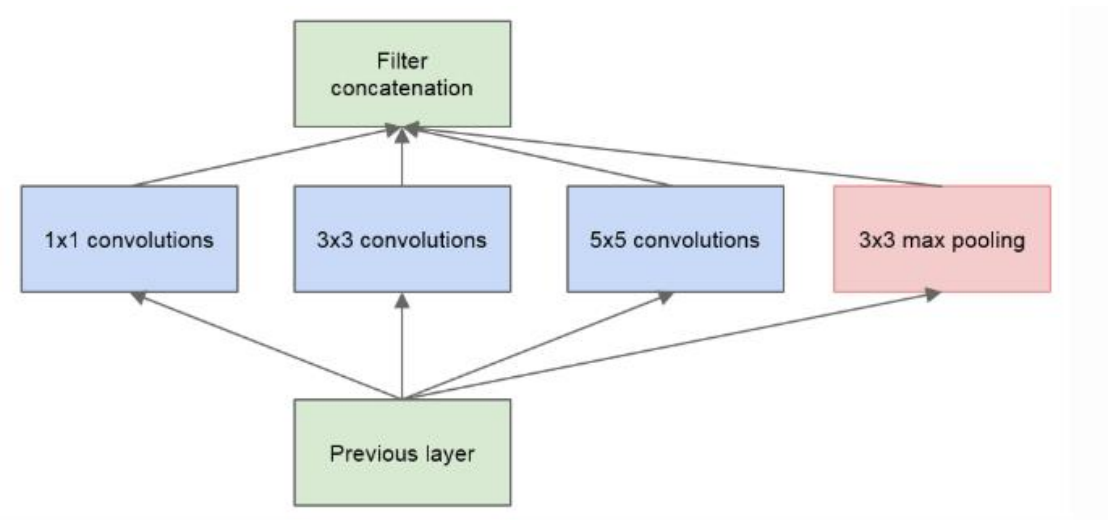


图 1-3 GoogleNet 结构图

Xception 使用了深度可分离卷积代替了传统的卷积操作,该操作大大节省了网络的 FLOPS 和参数量,但是精度反而有所提升。在 DeeplabV3+中,将 Xception 做了进一步的改进,同时增加了 Xception 的层数,设计出了 Xception65 和 Xception71 的网络。

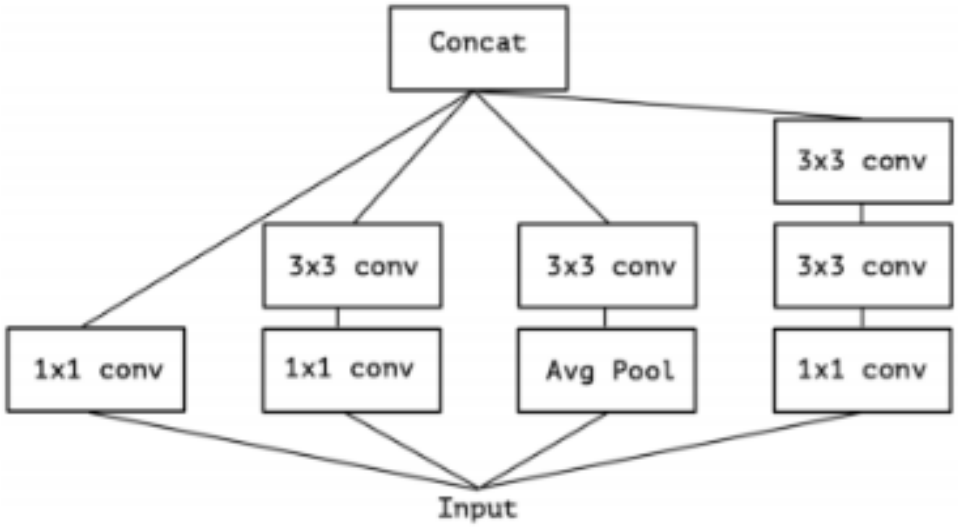


图 1-4 Xecption 网络结构图

InceptionV4 是 2016 年由 Google 设计的新的神经网络,当时残差结构风靡一时,但是设计者认为仅使用 Inception 结构也可以达到很高的性能。InceptionV4 使用了更多的 Inception module,在 ImageNet 上的精度再创新高。

ResNet 该网络创新性的提出了残差结构,通过堆叠多个残差结构从而构建了 ResNet 网络。实验表明使用残差块可以有效地提升收敛速度和精度。由于 ResNet

卓越的性能,越来越多的来自学术界和工业界的学者和工程师对其结构进行了改进,其中 ResNet-vd 的参数量和计算量与 ResNet 几乎一致,但是结合适当的训练策略,最终的精度提升高达 2.5%。

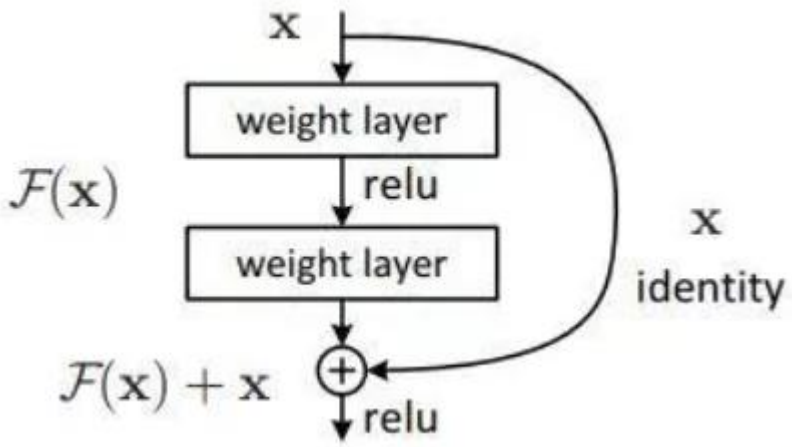


图 1-5 残差学习的基本单元

ResNeXt 是 ResNet 的典型变种网络之一,ResNeXt 发表于 2017 年的 CVPR 会议。在此之前,提升模型精度的方法主要集中在将网络变深或者变宽,这样增加了参数量和计算量,推理速度也会相应变慢。ResNeXt 结构提出了通道分组(cardinality)的概念,作者通过实验发现增加通道的组数比增加深度和宽度更有效。该网络可以在不增加参数复杂度的前提下提高准确率,同时还减少了参数的数量,所以是比较成功的 ResNet 的变种。

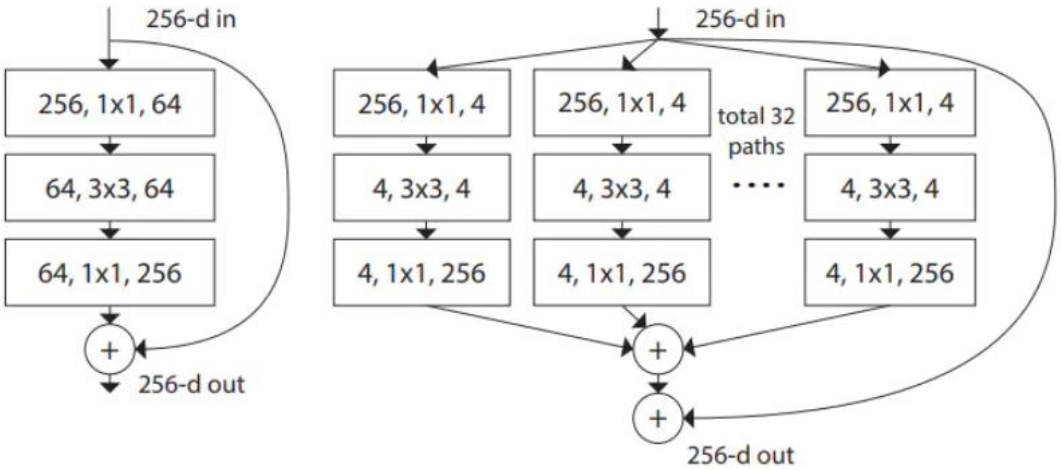


图 1-6 ResNext 的 block 单元

Res2Net 是 2019 年提出的一种全新的对 ResNet 的改进方案,该方案可以和现有其他优秀模块轻松整合,在不增加计算负载量的情况下,在 ImageNet、CIFAR-100 等数据集上的测试性能超过了 ResNet。Res2Net 结构简单,性能优越,进一步探索了 CNN 在更细粒度级别的多尺度表示能力。Res2Net 揭示了一个新的提升模型精度的维度,即 scale,是除了深度、宽度和基数的现有维度之外另外一个必不可少的更有效的因素。该网络在其他视觉任务如目标检测、图像分割等也有相当不错的表现。

ResNeXt 是 Facebook 于 2016 年提出的一种对 ResNet 的改进版网络。在 2019 年,Facebook 通过弱监督学习研究了该系列网络在 ImageNet 上的精度上限,为了区别之前的 ResNeXt 网络,该系列网络的后缀为 wsl,其中 wsl 是弱监督学习(weakly-supervised-learning)的简称。为了能有更强的特征提取能力,设计者将其网络宽度进一步放大,其中最大的 ResNeXt101_32x48d_wsl 拥有 8 亿个参数,将其在 9.4 亿的弱标签图片下训练并在 ImageNet-1k 上做 finetune,最终在 ImageNet-1k 的 top-1 达到了 85.4%,这也是迄今为止在 ImageNet-1k 的数据集上以 224x224 的分辨率下精度最高的网络。Fix-ResNeXt 中,作者使用了更大的图像分辨率,针对训练图片和验证图片数据预处理不一致的情况下做了专门的 Fix 策略,并使得 ResNeXt101_32x48d_wsl 拥有了更高的精度,由于其用到了 Fix 策略,故命名为 Fix-ResNeXt101_32x48d_wsl。

DenseNet 是 2017 年 CVPR best paper 提出的一种新的网络结构,该网络设计了一种新的跨层连接的 block,即 dense-block。相比 ResNet 中的 bottleneck,dense-block 设计了一个更激进的密集连接机制,即互相连接所有的层,每个层都会接受其前面所有层作为其额外的输入。DenseNet 将所有的 dense-block 堆叠,组合成了一个密集连接型网络。密集的连接方式使得 DenseNe 更容易进行梯度的反向传播,使得网络更容易训练。 DPN 的全称是 Dual Path Networks,即双通道网络。该网络是由 DenseNet 和 ResNeXt 结合的一个网络,其证明了 DenseNet 能从靠前的层级中提取到新的特征,而 ResNeXt 本质上是对之前层级中已提取特征的复用。经过进一步分析,发现 ResNeXt 对特征有高复用率,但冗余度低,DenseNet 能创造新特征,但冗余度高。结合二者结构的优势,设计出了 DPN 网络。最终 DPN 网络在同样 FLOPS 和参数量下,取得了比 ResNeXt 与 DenseNet 更好的结果。

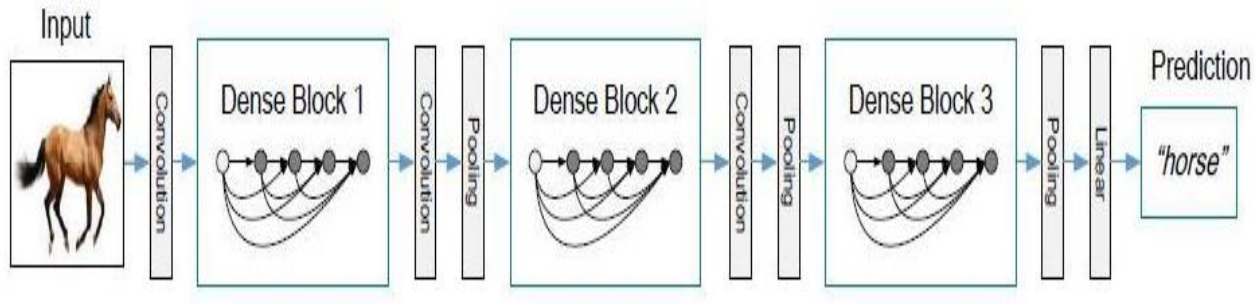


图 1-7 DenseNet 结构图

与传统的学习方法相比，深度学习方法预设了更多的模型参数，因此模型训练难度更大，根据统计学习的一般规律知道，模型参数越多，需要参与训练的数据量也越大。 20 世纪八九十年代由于计算机计算能力有限和相关技术的限制，可用于分析的数据量太小，深度学习在模式分析中并没有表现出优异的识别性能。自从 2006 年， Hinton 等提出快速计算受限玻耳兹曼机(RBM)网络权值及偏差的 CD-K 算法以后，RBM 就成了增加神经网络深度的有力工具，导致后面使用广泛的 DBN(由 Hinton 等开发并已被微软等公司用于语音识别中)等深度网络的出现。与此同时，稀疏编码等由于能自动从数据中提取特征也被应用于深度学习中。基于局部数据区域的卷积神经网络方法今年来也被大量研究。

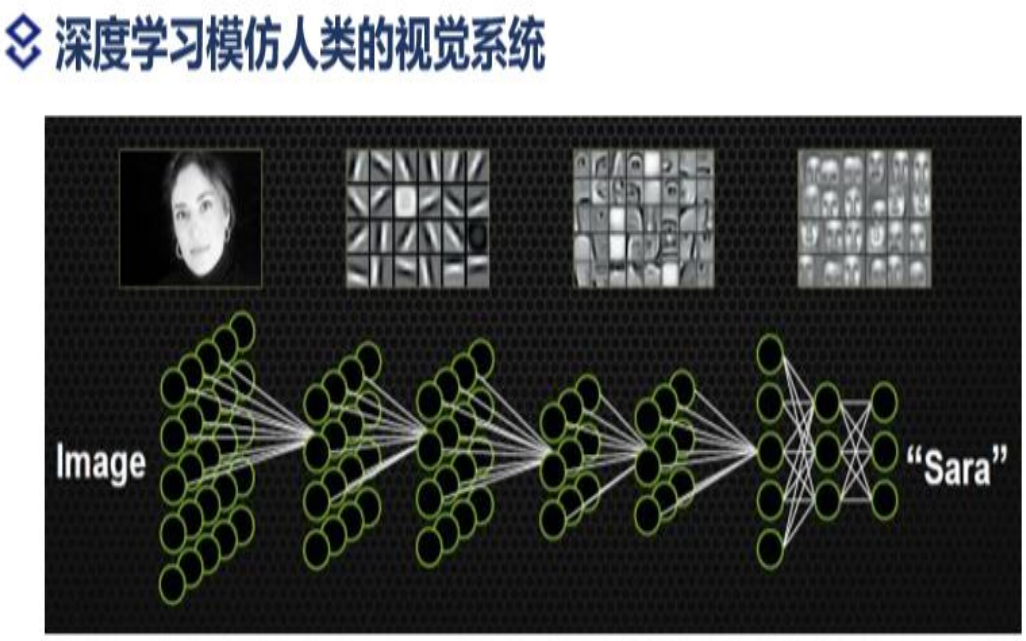


图 1-8 深度学习模仿人类视觉系统

1.5 主要研究内容

本文主要学习了深度学习相关的理论知识，并对基于深度学习的图像分类和损失函数算法进行研究，使用通过自建数据集进行实验分析，实现了基于深度学习的盒装药品识别算法，并分析训练数据集的大小、优化算法对模型分类准确率结果的影响，并通过对损失函数的研究进一步提高网络模型分类的准确率。

2.相关基础知识概述

2.1 神经网络相关知识

2.1.1 人工神经元模型

生物学上的神经元通常由细胞体，细胞核，树突和轴突组成。树突用于接收来自其他神经元的信号。神经元具有多个树突。细胞核核是神经元的核心模块，用于处理所有输入信号。轴突是输出信号的单元，并具有许多轴突终端，可以将信号发送到其他神经元的树突。

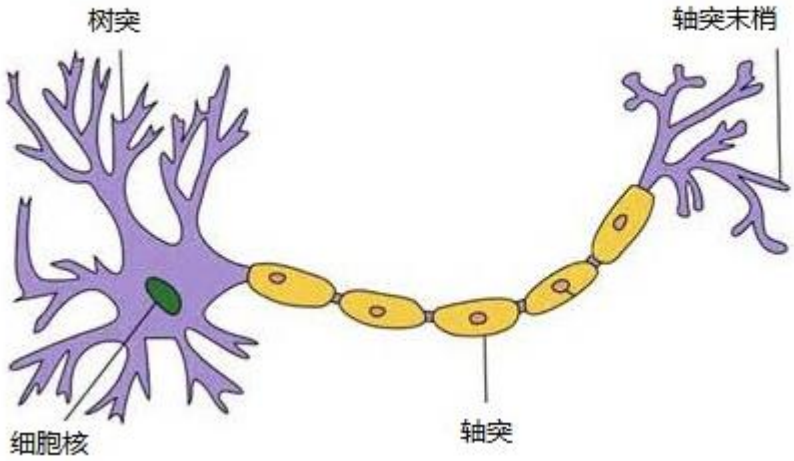


图 2-1 神经元模型图

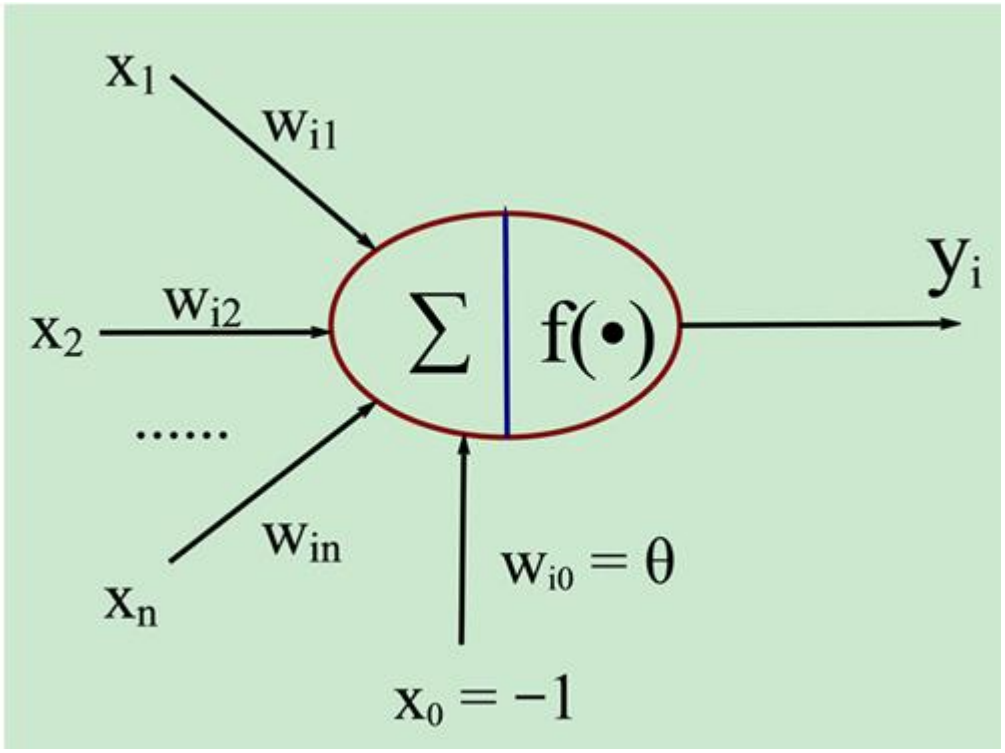


图 2-2 人工神经元模型图

图中 X1 到 Xn 是从其他神经元传入的输入信号，Wi1 到到 Win 分别是传入信号的权重，θ表示一个阈值（偏置），偏置的设置是为了正确分类样本，是模型中一个重要的参数。神经元综合的输入信号和偏置（符号为-1~1）相加之后产生当前神经元最终的处理信号 net，该信号称为净激活或净激励（net activation），激活信号作为上图中圆圈的右半部分 f（*）函数的输入，即 f(net)； f 称为激活函数或激励函数（Activation Function），激活函数的主要作用是加入非线性因素，解决线性模型达、分类能力不足的问题。上图中 y 是当前神经元的输出。

2.1.2 BP 神经网络

在人工神经网络的发展历史上，感知机(Multilayer Perceptron，MLP)网络曾对人工神经网络的发展发挥了极大的作用，也被认为是一种真正能够使用的人工神经网络模型，它的出现曾掀起了人们研究人工神经元网络的热潮。单层感知网络（M-P 模型）做为最初的神经网络，具有模型清晰、结构简单、计算量小等优点。但是，随着研究工作的深入，人们发现它还存在不足，例如无法处理非线性问题，即使计算单元的作用函数不用阀函数而用其他较复杂的非线性函数，仍然只能解决解决线性可分问题．不能实现某些基本功能，从而限制了它的应用。增强网络的分类和识别能力、解决非线性问题的唯一途径是采用多层前馈网络，即在输入层和输出层之间加上隐含层。构成多层前馈感知器网络。20 世纪 80 年代中期，DavidRunelhart。GeoffreyHinton 和 RonaldW-llians、DavidParker 等人分别独立发现了误差反向传播算法(Error Back Propagation Training)，简称 BP，系统

解决了多层神经网络隐含层连接权学习问题，并在数学上给出了完整推导。人们把采用这种算法进行误差校正的多层前馈网络称为 BP 网。BP 神经网络具有任意复杂的模式分类能力和优良的多维函数映射能力，解决了简单感知器不能解决的异或(Exclusive OR，XOR)和一些其他问题。从结构上讲，BP 网络具有输入层、隐藏层和输出层；从本质上讲，BP 算法就是以网络误差平方为目标函数、采用梯度下降法来计算目标函数的最小值。

在 BP 神经网络模型中，有三层结构，输入层、隐藏层、输出层,因此输入层到隐藏层的权值，设为 v_{ih},隐藏层第 h 个神经元的阈值我们设为 γ_h。隐藏层到输出层的权值，设为 w_{hj},输出层第 j 个神经元的阈值我们用 θ_j 表示。在下面这张图里，有 d 输入神经元,q 个隐藏神经元，隐藏有 q 个隐藏神经元阈值，ll 个输出神经元，因此有 ll 个输出神经元阈值。

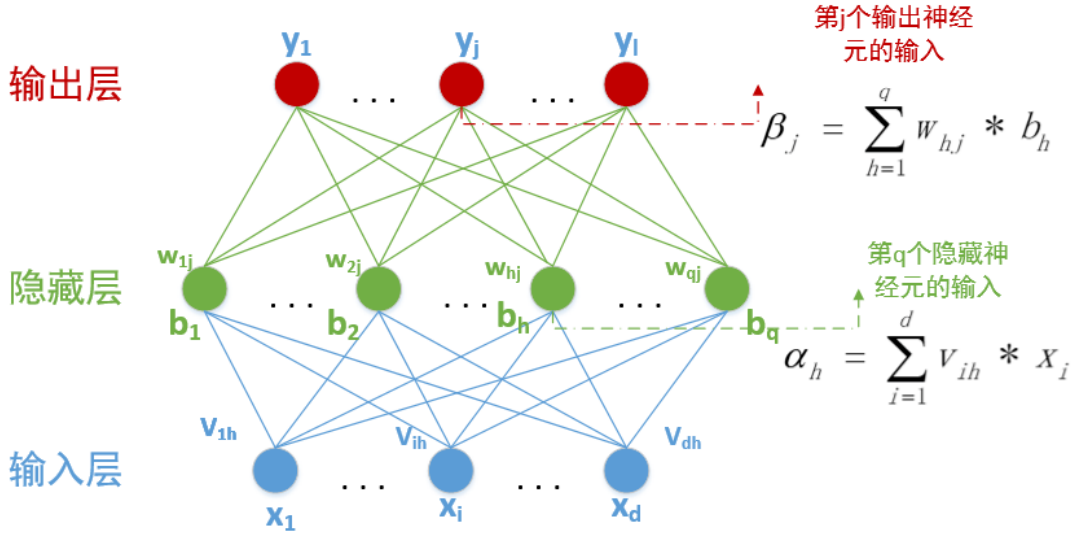


图 2-3 BP 神经网络结构图

其中 β jβj 中的 $b_h = f(\alpha_h - \theta_h)$。隐藏层和输出层的激活函数，在这里我们暂时全部用 Sigmod 函数。在某个训练示例(x_k,y_k)中，假设神经网络的训练输出为 $y_k,=(y^{k,}{}_1, y^{k,}{}_2, y^{k,}{}_l)$，yk 是 l 维向量，其中

$$y_i^k = f(\beta_i - \theta_i) \qquad (2\text{-}1)$$

那么这次预测结果的误差我们可以用最小二乘法表示：

$$E_k = \frac{1}{2}\sum_{j=1}^{l}(y_j^{k'} - y_j^k)^2 \qquad (2\text{-}2)$$

作为目前较为成功的神经网络学习算法，BP 神经网络每次根据训练的结果

与试验前预测的理论结果进行误差分析，不断修改权值和阈值，逐渐达到预期目标，得到一个较为可靠的模型。

2.1.3 卷积神经网络

卷积最早是数学分析中的一种积分变化的方法，卷积神经网络是包含卷积计算的前馈神经网络。

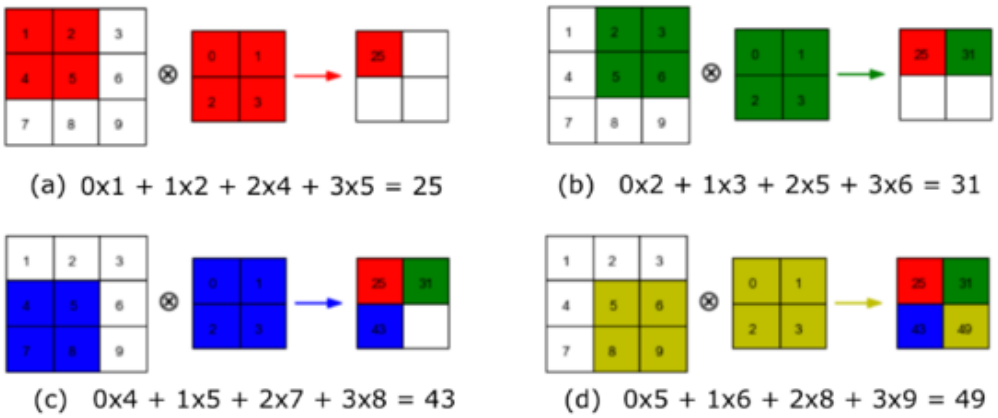


图 2-4 卷积计算过程图

可以对输入的信息进行平移不变分类。在 1988 年，有人提出用于检测医学影像的平移而不变人工神经网络，即 SIANN，次年，同样被用于计算机视觉问题的神经网络模型 LeNet 被提出，它有两个卷积层，全连接层也是两个，共有学习参数 60000 余个，在结构上与现代的卷积神经网络很接近。1989 年，LeCun 在对权重初始化中首次采用了随机梯度下降法，这一方案被之后的深度学习研究者所学习参考，而他在论述其网络模型结构时使用了‘卷积’这个全新概念，卷积神经网络也因此而诞生。

卷积神经网络的输入层能够处理维数多的复杂数据，通常，一维的卷积神经网络其输入层可输入一维数组和二维数组，二维数组可能具有多个通道。二维卷积神经网络它的输入层可以输入二维或三维数组，但是在大部分研究过程中，一般假设三维输入数据，便于采用 RGB 通道。

卷积神经网络最大的特点在与其隐含层，通常卷积神经网络的隐含层由卷积层、池化层和全连接层三部分组成。卷积层和池化层是卷积神经网络最重要的结构。卷积层的卷积核拥有权重系数，但是在池化层中，不包含权重系数。卷积层主要是提取数据特征，具有多个卷积核，而每个卷积核中的元素都有对应的权重系数和偏差量，其内神经元与神经元之间紧密相关，每个神经元在前一层位置与它接近的区域内的多个神经元相连，区域的大小即为感受野。获取数据特征时，在感受野内会将输入的矩阵相乘求和，然后叠加偏差量。

卷积核大小，卷积核步长，这三个超参数决定了卷积层输出结果的尺寸。卷积核大小一般小于输入数据尺寸，卷积核越大，提取的特征越复杂。

卷积步长确定了每次扫描较上次操作的距离，例如，卷积步长为 n 时，下次扫描时会移动 n-1 个元素。

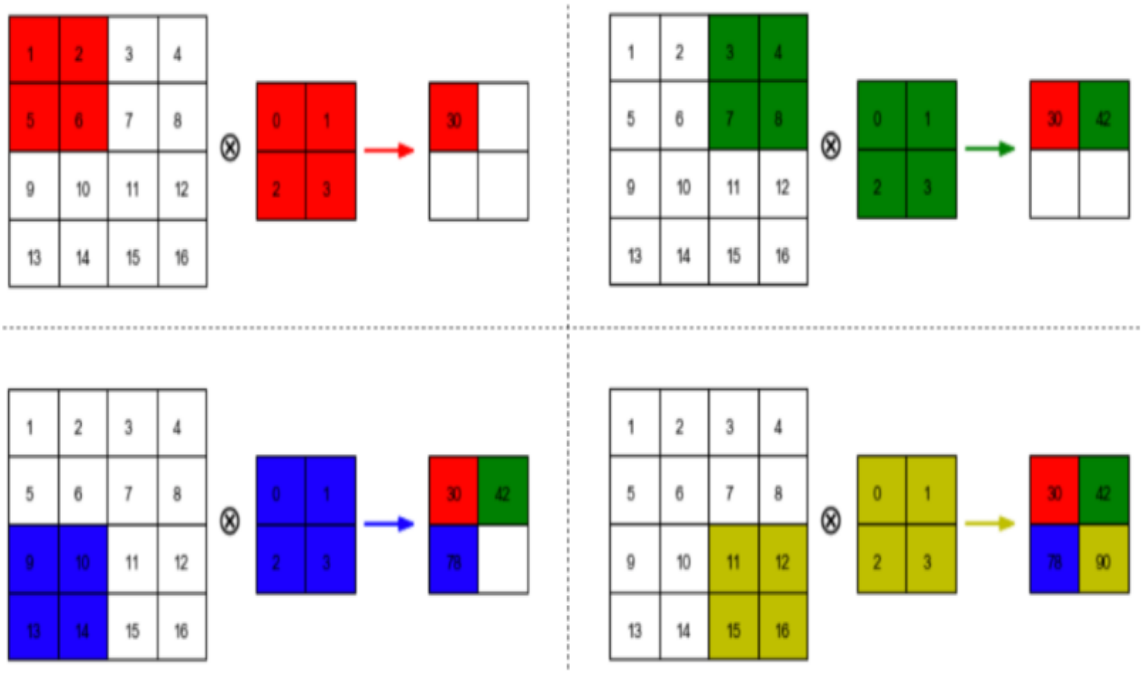


图 2-5 步长为 1 的卷积过程

而填充参数直接影响着输入结果的大小，通常，为确保卷积前后的输入输出数据尺寸保持一致，会对输入数据进行一定数据填充，通常采用 0 填充。

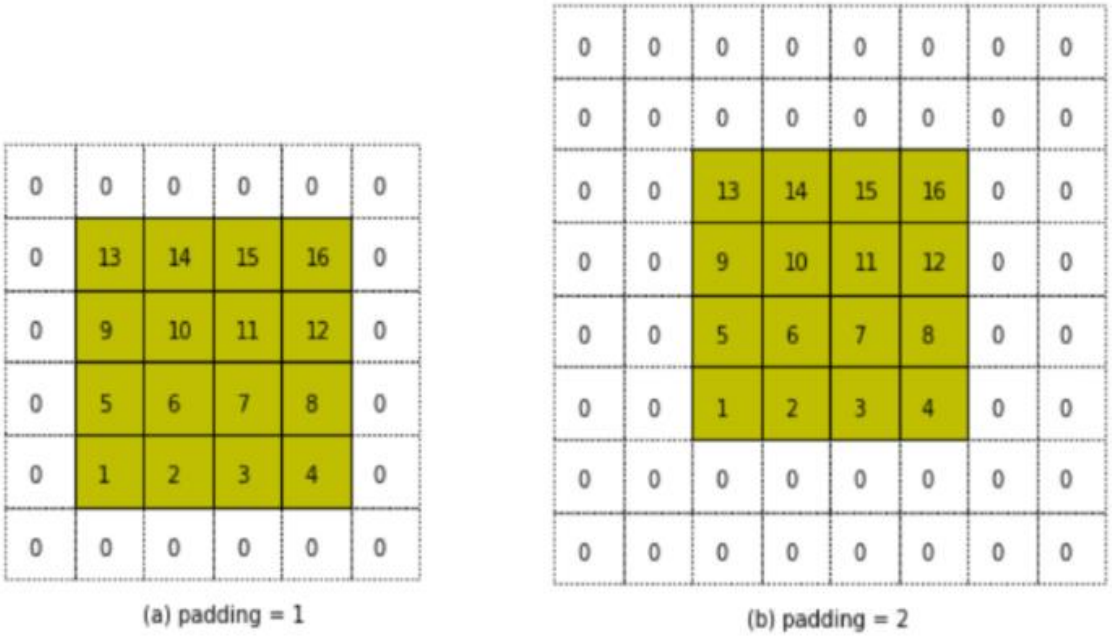


图 2-6 0 填充

一般情况下，为了提取较复杂特征，在卷积中会使用激励函数（激活函数）

$$A_{i,j,k}^{l} = f(Z_{i,j,k}^{l}) \qquad (2\text{-}3)$$

最常用的激活函数一般是 Relu 函数，激活函数操作一般都在卷积核后。

输入数据经过卷积层处理后，会进入池化层，在池化层进一步选择特征核信息过滤，常见的池化方式有 Lp 池化，随机/混合池化以及谱池化。

（1）Lp 池化

Lp 池化一般形式为：

$$A_{k}^{l}(i,j) = \left[\sum_{x=1}^{f}\sum_{y=}^{f} A_{k}^{l}(s_0 i + x, s_0 j + y)^2\right]^{\frac{1}{p}} \qquad (2\text{-}4)$$

其中 s_0 表示步长（i，j）代表像素，p 是预定参数。当 Lp 池化在池化局域内取均值时，此时又可以称为均值池化，取极大值时，即为极大值池化。

（2）随机/混合池化

随机池化，混合池化是 Lp 池化的延申。随机池化是在池化区域内按一定概率随机选取一个值使部分非极大激励信号进入下一个模块中。混合池化是均值池

化核极大池化的一个线性组合。

$$A_k^l = \alpha L_1\left(A_k^l\right) + L_\infty\left(A_k^l\right),\ \alpha\epsilon[0,1] \qquad (2\text{-}5)$$

（3）谱池化

谱池化会对输入数据的每个通道进行 DFT 变化，然后对频谱中心的序列进行 DFT 逆变换得到池化结果。谱池化具有滤波功能，可以调整特征图的大小。

卷积神经网络的全连接层，它将卷积层和池化层提取出的特征数据进行非线性组合，全连接层不能提取数据特征。在某些网络中页可以用全局均值池化代替全连接层。

卷积神经网络的输出层一般在全连接层之后，在图像分类问题中输出层采用逻辑函数或者归一化函数输出分类标签。

2.2 深度学习在图像分类问题中的应用

图像分类主要是根据图像信息中的不同的特征，把不痛类别的区分开来。随着计算机技术的发展，利用计算机的超大规模的计算能力，可以对所要处理的图像进行更好的定量分析，可以精细到各个像素点的特征，从而利用计算机代替人眼进行图像分类。

2.2.1 传统的图像分类技术

传统的图像分类有很多种，包括基于色彩特征索引技术、基于纹理的图像分类技术、基于形状的图像分类技术，和基于空间关系的图像分类技术。以下对其简要概述，

（1）基于色彩特征的索引技术

每个物体都拥有着独一无二的色彩特征，甚至于对于不同的色彩特征人们往往能联想到某一个具体的物体。所以，我们可以利用图像元素的色彩特征来进行图像分类。最早的色彩特征分类方法是色彩直方图法。色彩直方图简单并且当图像被进行一定的处理后，例如对图像尺寸进行一定的缩放来改变图像的尺寸大小。以及对图像进行旋转等操作处理时，色彩直方图不会发生太大变化。现在主要又全局色彩特征索引和局部色彩特征索引。

（2）基于纹理的图像分类技术

纹理特征包括粒度、对比度、方向性、线性、均匀性和粗糙度等，这些纹理属性也是图像的重要特征。纹理度的灰度共生矩阵表示法是较为经典的基于纹理的图像分类技术之一，它对各个像素之间的距离和它们对应的方向建立共生矩阵，从而提取对图像分类有意义的特征向量。

（3）基于形状的图像分类技术

在一个二维空间中，形状是图像可视化的极为重要的特征之一，它描述了图

像的轮廓曲线，对整个图像进行全局描述。根据图像的形状轮廓曲线特征建立对应的图像索引，但是通常会根据区域特征和边界特征形状进行图像分类。

（4）基于空间关系的图像分类技术

在一个具体的图像中，图像中的每个元素都有着特定空间位置，根据这些独特的空间位置关系可以很好的判定图像类别。因此这个问题的关键是存储图像中的各个元素之间的对应关系，为了解决这个问题，很多学者纷纷给出自己的方案。最初的有人采用逐渐细化图像元素，从而精确表示各个元素之间的空间关系，但是很明显这个方案并不适用于那些包含很多元素的复杂图像。随后又有学者从提高符号图精度的角度出发，对所包含元素的大小、距离进行推理，得出较为准确的符号图，但是这个方案需要对各个元素进行不同程度的处理，时间复杂度较高。

传统的图像分类技术计算量大，时间复杂度高，在早期计算机技术尚未成熟的条件下，处理难度高。但是随着后来计算机算力的大幅提升，这些图像分类技术得到一定程度上的发展。不过，后来卷积神经网络被人们首次用到解决图像分类问题中，图像分类问题似乎有了更好的解决方案。

2.2.2 深度学习在图像分类中的应用

最早的深度学习模型就是用来识别图像中非常小的单个对象。随着计算机性能的提升，计算机的算力大幅提高，神经网络可以处理越来越大的图像。目前的深度学习模型能够处理较高分辨率的照片，而且并不需要进行任何尺寸上的缩放处理。并且，最初的神经网络模型只能识别出两种不同类别的对象，但是现有的网络模型已经具备识别出上千种类别对象的能力，而且准确率仍在不断提高。目前有多种深度学习模型，以下对其简要介绍。

（1）Lenet 网络模型

Lenet 于 1986 年被首次提出，但是它的结构比较简单，仅仅只拥有两个卷积层，两个池化层和两个全连接层。在卷积层中，每个卷积核的大小为 5*5。该网络先将输入图像转换为单通道图像数据矩阵，假设输入图像尺寸为 28*28，它对应的矩阵为【1，28， 28】。在第一个步长为 1，卷积核数目为 20 的卷积层中可以得到【20， 24， 24】的输出矩阵。再经过一个步长为 2 的最大池化操作，图像的尺寸会变为原来的一半，即经过第一次池化操作后的输出矩阵为【20，12，12】，然后再经过一次步长为 1，卷积核数目为 50 的卷积操作以及步长为 2 的最大池化操作，最终得到输出矩阵为【50，4，4】，在第二次池化操作之后连接全连接层，再加入 Relu 激励函数，之后接入第二个全连接层，进行分类，输出分类概率。

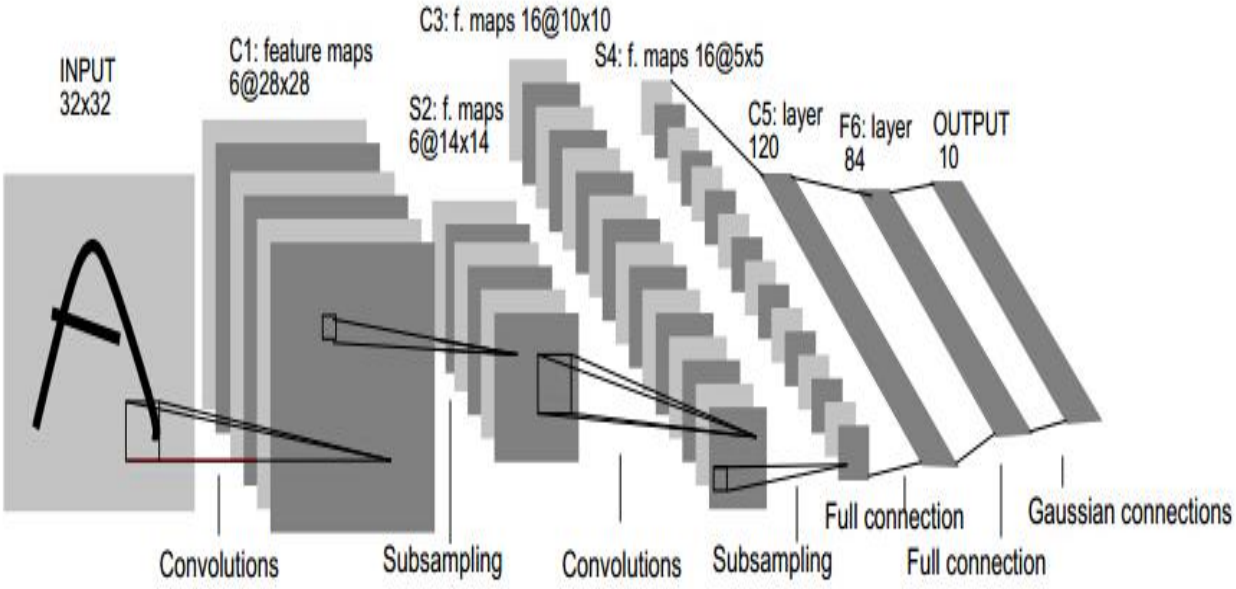


图 2-7 Lenet 模型结构图

（2）Alexnet

2012 年，Alexnet 模型获得 Imagenet 图像分类大赛冠军。Alexnet 采用 ReLu 激活函数，也使得此后 ReLu 被深度学习研究学者广泛使用。并且，在训练过程中为了避免模型出现过拟合现象，训练时采用 Dropout 随机忽略部分神经元。并且为了避免由于平均池化所带来的模糊化效果，在整个模型中全部采用最大池化，从而使提取的模型更丰富。但最重要的是在 Alexnet 模型中采用了 LRN 层，对局部神经元构建竞争机制，提高模型的泛化能力。

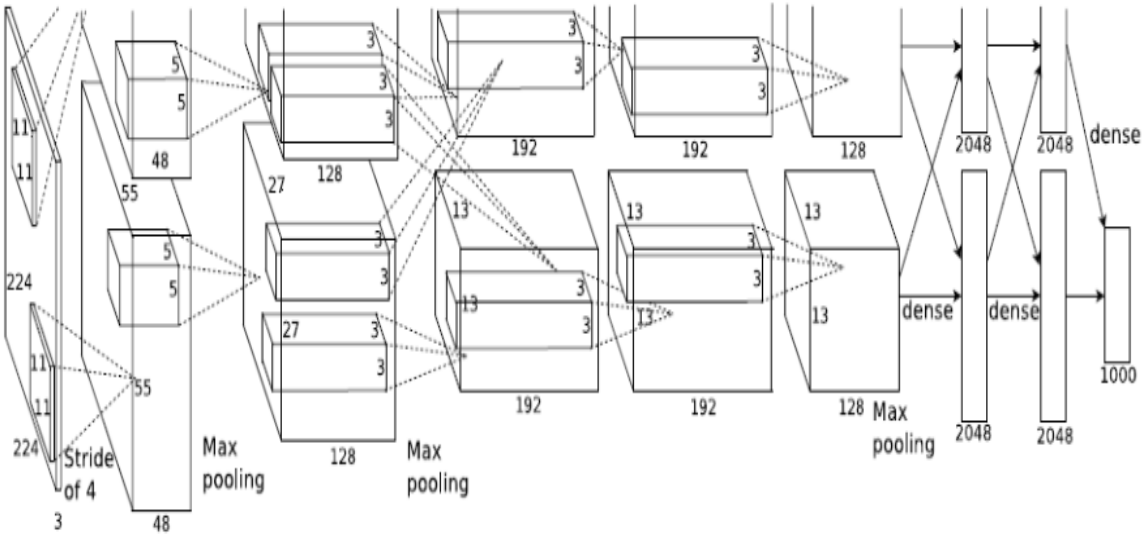


图 2-8 Alexnet 结构图

（3）GoogleNet

GoogleNet 没有全连接层，通过平均池化代替全连接层，并用实验证明该方案 1 可以将模型准确率提高 0.6 个百分点，虽然效果不是很明显，但是为模型优化提供了一种思路。GoogLenet 采用了 Inception 结构，便于后期对代码调试与优化。

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

图 2-9 GoogleNet 层次详解图

从 GoogLeNet 的实验结果来看，效果很明显，差错率比 MSRA、VGG 等模型都要低，对比结果如下表所示：

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

图 2-10 各模型效果对比图

（4） Vgg

Vgg，牛津大学研发的卷积神经网络模型，采用多个较小的卷积核代替大卷积核，通过增加网络模型深度确保提取到更全面的特征，但另一个方面又大幅度减少了参数个数。在相同步长情况下，卷积核大小对卷积参数影响不大，但是卷积核越大，计算量也越大，Vgg 网络主要通过堆积小卷积核形成多层非线性结构层，在拥有相同的感受野的情况下，提升神经网络深度，从而提高模型效果。

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

图 2-11 vgg 网络分析图

Vgg 网络模型具有 5 个卷积部分，每个部分又拥有两个或三个卷积层，并且都连接一个最大池化层，以缩小数据尺寸，每个部分内的卷积层的卷积核数量一样，但是随着卷积部分的递进卷积核数量逐渐增加：64->128->256->512->512，并且这些卷积核都由 3*3 的小卷积核堆积而成。

两个 3*3 卷积核串联的卷积效果与一个 5*5 的卷积核的卷积效果相同，3 个 3*3 的卷积核串联其卷积结果等价于一个 7*7 的卷积核的卷积效果，但是虽然效果相同，需要的卷积参数大幅度递减，也就可以很好的避免过拟合现象，从而使卷积神经网络的学习能力大幅度提高。

2.3 本章小结

本章主要概述了神经网络发展历程以及卷积神经网络在图像分类中的应用，分析了传统图像分类技术，总结了传统分类技术的优缺点，并简要概述了可以用

于解决图像分类问题的深度学习模型。

3.工具介绍和数据处理

‘工欲善其事，必先利其器’，在构建深度学习模型前，需要选择合适的深度学习框架，更需要一个合适的数据集。对本课题进行一定的研究并查阅资料后，决定选取 paddlepaddle 深度学习框架。由于本课题研究内容可移植性强，而目前并没有较为合适的公开数据集，于是我选择用手机拍摄药品盒图片，并利用 python 的 PIL 库对拍摄照片进行一定处理，从而扩充已有数据构建所需数据集。以下对上述内容进行简要阐述。

3.1 paddlepaddle 深度学习框架概述

PaddlePaddle（飞桨）是百度开发的一款开源的深度学习框架。其采用基于编程逻辑的组网范式，便于初学者学习使用。PaddlePaddle 已经实现超大规模深度学习模型训练技术，实现了世界首个支持亿万特征、万亿参数的开源大规模训练平台，攻克了超大规模深度学习模型的在线学习难题，实现了万亿规模参数模型的实时更新。目前飞桨已经同时支持动态图和静态图两种编程方式。

Paddle 和其他主流框架一样，使用 Tensor 数据结构来承载数据，包括模型中的可学习参数（如网络权重、偏置等），网络中每一层的输入输出数据，常量数据等 Tensor 可以简单理解成一个多维数组，一般而言可以有任意多的维度。不同的 Tensor 可以具有自己的数据类型和形状，同一 Tensor 中每个元素的数据类型是一样的， Tensor 的形状就是 Tensor 的维度。

在 Paddle 中可以使用 fluid.data 来创建数据变量， fluid.data 需要指定 Tensor 的形状信息和数据类型， 当遇到无法确定的维度时，可以将相应维度指定为 None。除 fluid.date 之外，还可以使用 fluid.layers.fill_constant 来创建常量。需要注意的是，在静态图编程方式中，上述定义的 Tensor 并不具有值（即使创建常量的时候指定了 value）， 它们仅表示将要执行的操作，在网络执行时（训练或者预测）才会进行真正的赋值操作。在网络执行过程中，获取 Tensor 数值有两种方式：方式一是利用 paddle.fluid.layers.Print 创建一个打印操作， 打印正在访问的 Tensor。方式二是将 Variable 添加在 fetch_list 中。

使用 fluid.data 创建数据变量之后，需要把网络执行所需要的数据读取到对应变量中。在 Paddle 中，数据计算类 API 统一称为 Operator（算子），简称 OP，大多数 OP 在 paddle.fluid.layers 模块中提供。某些场景下，用户需要根据当前网络中的某些状态，来具体决定后续使用哪一种操作，或者重复执行某些操作。在动态图中，可以方便的使用 Python 的控制流语句（如 for，if-else 等）来进行条件判断，但是在静态图中，由于组网阶段并没有实际执行操作，也没有产生中间计算结果，因此无法使用 Python 的控制流语句来进行条件判断，为此静态图

提供了多个控制流 API 来实现条件判断。这里以 fluid.layers.while_loop 为例来说明如何在静态图中实现条件循环的操作。while_loop API 用于实现类似 while/for 的循环控制功能，使用一个 callable 的方法 cond 作为参数来表示循环的条件，只要 cond 的返回值为 True，while_loop 就会循环执行循环体 body（也是一个 callable 的方法），直到 cond 的返回值为 False。

3.2 PIL 简介

Python Image Library（PIL）是 python 的第 3 方图像处理库。PIL 可以处理很多图像相关的问题，PPIL 非常适合于图像归档以及图像的批处理任务。你可以使用 PIL 创建缩略图，转换图像格式，打印图像等等。图像展示(Image Display)。PIL 较新的版本支持包括 Tk PhotoImage，BitmapImage 还有 Windows DIB 等接口。PIL 支持众多的 GUI 框架接口，可以用于图像展示。图像处理(Image Processing)。PIL 包括了基础的图像处理函数，包括对点的处理，使用众多的卷积核(convolution kernels)做过滤(filter),还有颜色空间的转换。PIL 库同样支持图像的大小转换，图像旋转，以及任意的仿射变换。PIL 还有一些直方图的方法，允许你展示图像的一些统计特性。这个可以用来实现图像的自动对比度增强，还有全局的统计分析等。

3.3 数据处理

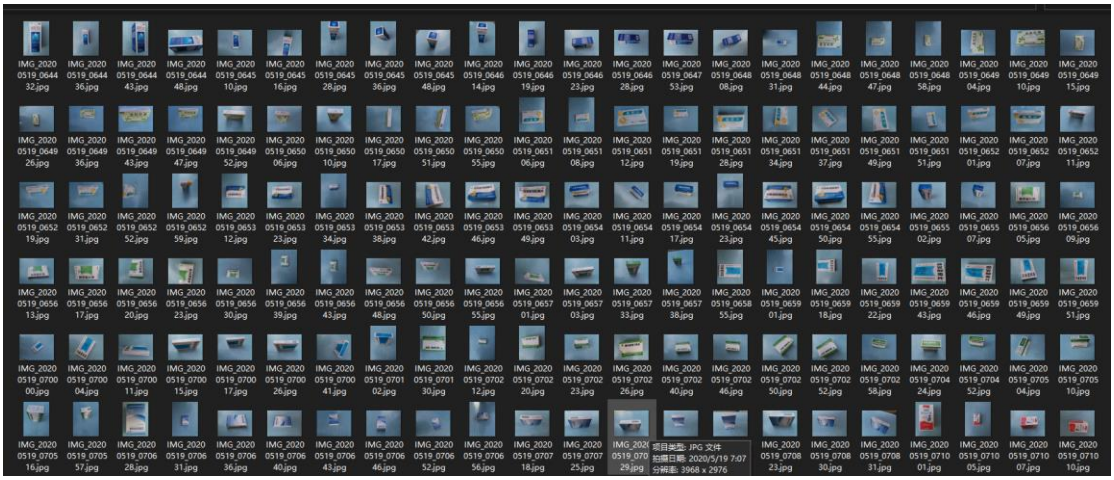
在深度学习模型构建中，选用合适的数据集直接影响着模型的准确度，因此构建一个满足本次课题需求的数据集非常重要。目前并没有已经公开的盒装药品图片数据集，所以，我选择自己利用手机拍摄照片，通过 PIL 库对拍摄照片进行相关处理，以达到数据集扩充的目的。

3.3.1 构建数据集

首先，我收集到自己家里常备的一些药品：清热解毒口服液、桂灵片、三九胃泰等 13 种药品，分别对每个药盒正面、侧面、背面多角度拍摄照片 16 张

图 3-1 单类药品多角度图

共拍摄 13 类药盒初始图片 208 张



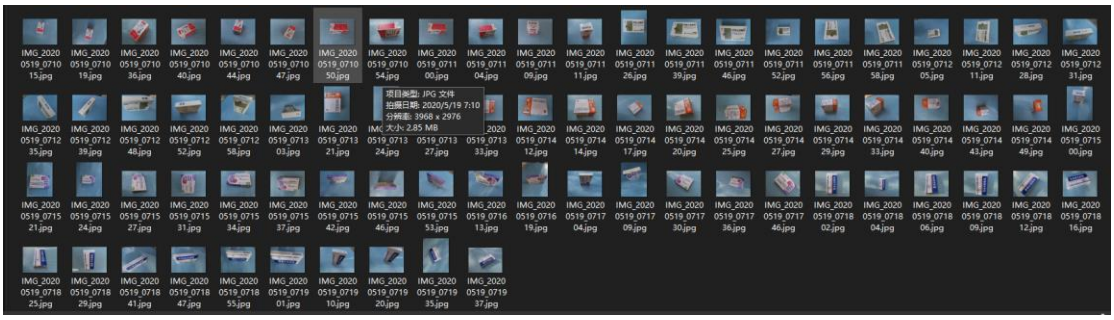


图 3-2 初始数据集图

之后利用 python 的 PIL 库对每一张初始图片进行旋转处理，亮度，色度，对比度，锐度增强处理。部分代码如下：

```
#首先根据所需处理图像所在位置，读取图片
import os
from PIL import Image
from PIL import ImageEnhance
import time
```

```
time_start = time.time()
i = 0
path = 'E:/LunWen/data/picture_16/'
for p in range(5, 13):
    path_p = path + '/' + '%d/' % p
    k = 0
    for root, dirs, files in os.walk(path_p):
        for file in files:
            img_path = root + '/' + file
            print(img_path)
            #  读取图像
            image = Image.open(img_path)
        #将初始图片重命名并保存
            image.save(path_p + '%d_%d.jpg' % (p, k))
```

(1) 旋转处理，变换角度为[15, 30, 45, 60, 75, 90, 105, 120, 115, 150, 165, 180]处理后共得到 12 张图片。代码如下：

```
for angle in [15, 30, 45, 60, 75, 90, 105, 120, 115, 150, 165, 180]:
                image_rotate = image.rotate(angle)
                image_rotate.save(path_p + '%d_%d.jpg' % (p, k))
                k += 1     #便于命名方便，每经过一次变换 k+1
```

图 3-3 旋转处理结果图

(2) 亮度变换，亮度系数为[0.15, 0.3, 0.45, 0.6, 0.75, 0.9, 1.1, 1.25, 1.4, 1.55, 1.7, 1.85, 2]处理后得到 13 张图片。代码如下：

```
#亮度
    enh_bri = ImageEnhance.Brightness(image)
```

```
for brightness in [0.15, 0.3, 0.45, 0.6, 0.75, 0.9, 1.1, 1.25, 1.4, 1.55, 1.7, 1.85,
2]:
        image_brightened = enh_bri.enhance(brightness)
        image_brightened.save(path_p + '%d_%d.jpg' % (p, k))
        k += 1
```

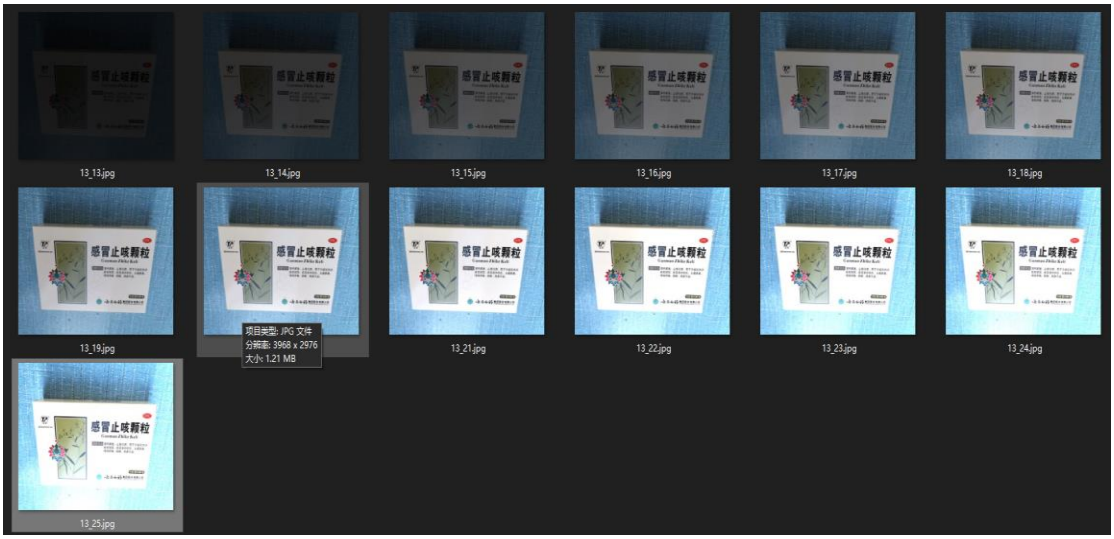


图 3-4 亮度变换结果图

(3) 色度变换，色度改变系数为[0.15, 0.3, 0.45, 0.6, 0.75, 0.9, 1.1, 1.25, 1.4, 1.55, 1.7, 1.85, 2]，代码如下：

```
enh_col = ImageEnhance.Color(image)
for color in [0.15, 0.3, 0.45, 0.6, 0.75, 0.9, 1.1, 1.25, 1.4, 1.55, 1.7, 1.85, 2]:
        image_colored = enh_col.enhance(color)
        image_colored.save(path_p + '%d_%d.jpg' % (p, k))
        k += 1
```

色度变换处理后可得到 13 张图片：

图 3-5 色度处理结果图

(4) 对比度变换，对比度改变系数为[0.15, 0.3, 0.45, 0.6, 0.75, 0.9, 1.1, 1.25, 1.4, 1.55, 1.7, 1.85, 2]，变换代码如下：

```
# 对比度
enh_con = ImageEnhance.Contrast(image)
for contrast in [0.15, 0.3, 0.45, 0.6, 0.75, 0.9, 1.1, 1.25, 1.4, 1.55, 1.7, 1.85, 2]:
        image_contrasted = enh_con.enhance(contrast)
        image_contrasted.save(path_p + '%d_%d.jpg' % (p, k))
        k += 1
```

本次处理后可得到 13 张图片

图 3-6 对比度处理结果图

(5) 锐度变换，锐度变换系数为 0.15, 0.3, 0.45, 0.6, 0.75, 0.9, 1.1, 1.25, 1.4, 1.55, 1.7, 1.85, 2]，变换代码如下：

```
# 锐度
    enh_sha = ImageEnhance.Sharpness(image)
    for sharpness in [0.15, 0.3, 0.45, 0.6, 0.75, 0.9, 1.1, 1.25, 1.4, 1.55, 1.7, 1.85,
```

```
2]:
    image_sharped = enh_sha.enhance(sharpness)
    image_sharped.save(path_p + '%d_%d.jpg' % (p, k))
     k += 1
```

本次处理后可得到 13 张图片:

图 3-7 锐度处理结果图

经过上诉过程中的图像变换处理，每一张初始照片经过变换可得到 13 * 4 + 12 共计 64 张图片，故经过此次数据扩充每类药盒图片数目为 （64 + 1）*16 共 1040 张图片，成功构建数据集，该药品盒数据集共有 13 类药品盒，共计 13520 张图片。

3.3.2 构建训练集和测试集

由于数据集过大，如果依旧对数据集中的每一张照片都按分类标签进行标记，工作量过于庞大，并且目前还没有比较好的自动标注工具，利用标记工具对 13520 张图片进行手动标注，显然不是解决问题的最好办法。

为了解决这个问题，我将每类图片都放在同一个文件夹，之后再构建训练集和验证集时，读取到的文件夹名称即为标签类别。

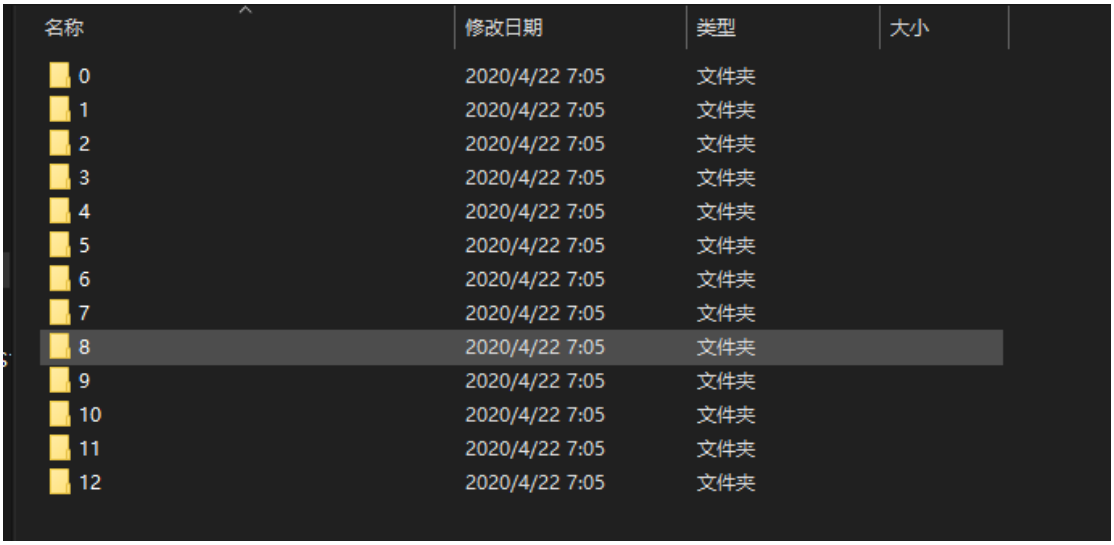


图 3-8 标签处理方法示意图

数据预处理代码如下：

```
for class_dir in class_dirs:
    #每个类别的信息
    class_detail_list = {}
    eval_sum = 0
    train_sum = 0
    #统计每个类别有多少张图片
    class_sum = 0
    #获取类别路径
    path = data_root_path + '/' + class_dir
    #获取所有图片
    img_paths = os.listdir(path)
    for img_path in img_paths:          #遍历文件夹下的每个图片
        name_path = path + '/' + img_path          #每张图片的路径
        if class_sum % 10 == 0:        #每 10 张图片去一个做验证数据
            eval_sum += 1            #eval_sum 为测试集的数目
            eval_list.append(name_path + '\t%d' % class_label + '\n')
        else:
            train_sum += 1
            train_list.append(name_path + '\t%d' % class_label + '\n')
        class_sum += 1
        all_class_images += 1
    class_label += 1
```

```
#说明 json 的 class_detail 数据
class_detail_list['class_name'] = class_dir        #类别名称
class_detail_list['class_label'] = class_label        #类别标签
class_detail_list['class_eval_images'] = eval_sum          #该类数据的测试集数目
class_detail_list['class_train_images'] = train_sum        #该类数据的训练集数目
class_detail.append(class_detail_list)
```

对原有数据集经过上诉处理后，按 9：1 的比例生成了训练集数据列表和测试集数据列表，训练集有数据 12168 条，如下图所示

```
12143 E:/LunWen/data/picture_16/8/8_462.jpg   11
12144 E:/LunWen/data/picture_16/1/1_953.jpg   1
12145 E:/LunWen/data/picture_16/5/5_315.jpg   8
12146 E:/LunWen/data/picture_16/8/8_306.jpg   11
12147 E:/LunWen/data/picture_16/1/1_624.jpg   1
12148 E:/LunWen/data/picture_16/3/3_1005.jpg  6
12149 E:/LunWen/data/picture_16/1/1_123.jpg   1
12150 E:/LunWen/data/picture_16/9/9_111.jpg   12
12151 E:/LunWen/data/picture_16/0/0_241.jpg   0
12152 E:/LunWen/data/picture_16/10/10_830.jpg 2
12153 E:/LunWen/data/picture_16/4/4_208.jpg   7
12154 E:/LunWen/data/picture_16/0/0_191.jpg   0
12155 E:/LunWen/data/picture_16/9/9_172.jpg   12
12156 E:/LunWen/data/picture_16/5/5_773.jpg   8
12157 E:/LunWen/data/picture_16/1/1_926.jpg   1
12158 E:/LunWen/data/picture_16/7/7_614.jpg   10
12159 E:/LunWen/data/picture_16/10/10_473.jpg 2
12160 E:/LunWen/data/picture_16/2/2_834.jpg   5
12161 E:/LunWen/data/picture_16/3/3_782.jpg   6
12162 E:/LunWen/data/picture_16/3/3_428.jpg   6
12163 E:/LunWen/data/picture_16/3/3_552.jpg   6
12164 E:/LunWen/data/picture_16/11/11_164.jpg 3
12165 E:/LunWen/data/picture_16/2/2_533.jpg   5
12166 E:/LunWen/data/picture_16/4/4_977.jpg   7
12167 E:/LunWen/data/picture_16/0/0_622.jpg   0
12168 E:/LunWen/data/picture_16/3/3_17.jpg    6
12169
```

图 3-9 训练集部分数据展示图

测试集拥有数据 1352 条，如下图所示：

```
1330  E:/LunWen/data/picture_16/6/6_639.jpg   9
1331  E:/LunWen/data/picture_16/2/2_495.jpg   5
1332  E:/LunWen/data/picture_16/5/5_477.jpg   8
1333  E:/LunWen/data/picture_16/0/0_332.jpg   0
1334  E:/LunWen/data/picture_16/10/10_521.jpg 2
1335  E:/LunWen/data/picture_16/2/2_378.jpg   5
1336  E:/LunWen/data/picture_16/8/8_107.jpg   11
1337  E:/LunWen/data/picture_16/0/0_927.jpg   0
1338  E:/LunWen/data/picture_16/0/0_837.jpg   0
1339  E:/LunWen/data/picture_16/5/5_576.jpg   8
1340  E:/LunWen/data/picture_16/2/2_396.jpg   5
1341  E:/LunWen/data/picture_16/2/2_161.jpg   5
1342  E:/LunWen/data/picture_16/12/12_81.jpg  4
1343  E:/LunWen/data/picture_16/10/10_611.jpg 2
1344  E:/LunWen/data/picture_16/10/10_81.jpg  2
1345  E:/LunWen/data/picture_16/10/10_567.jpg 2
1346  E:/LunWen/data/picture_16/1/1_1015.jpg  1
1347  E:/LunWen/data/picture_16/11/11_341.jpg 3
1348  E:/LunWen/data/picture_16/4/4_576.jpg   7
1349  E:/LunWen/data/picture_16/6/6_684.jpg   9
1350  E:/LunWen/data/picture_16/4/4_431.jpg   7
1351  E:/LunWen/data/picture_16/7/7_620.jpg   10
1352  E:/LunWen/data/picture_16/12/12_198.jpg 4
1353
```

图 3-10 测试集部分数据展示图

每条数据由各个图像所在位置，以及所属标签构成。并且在构建训练集和测试集时已经将所有数据进行随机排列分布。

4.VGG 网络模型实现

本章将利用 PaddlePaddle 构建 vgg 神经网络模型，实现深度学习图像分类算法，进而实现根据照片识别盒装药品，并输出选购药品总价。

4.1 vgg 网络模型构建

4.1.1 归一化处理

由于所选数据集照片像素维度各不相同，而且，手机拍摄的图片像素较高，处理大量照片时，计算量过高，为确保所有数据都处于相同的数据维度并且适度降低计算量，首先对每个输入数据进行归一化。部分代码如下：

```
def data_mapper(sample):
    img_path, label = sample
#进行图片的读取，由于数据集的像素维度各不相同，需要进一步处理图像进行变换
    img = paddle.dataset.image.load_image(img_path)
#进行简单的图像变换，对图像进行 crop 修剪操作，输出 img 的维度为（3，47，47）
    img = paddle.dataset.image.simple_transform(im=img,
                                                resize_size=47, #
裁剪图片
                                                crop_size=47,
                                                is_color=True,
                                                is_train=True)
#将 img 数组进行归一化处理
    img = img.flatten().astype('float32') / 255.0
    return img, label
```

4.1.2 定义 CNN 模型

利用 padddlepaddle 定义卷积操作并构建 vgg 网络模型，具体代码如下：

```
#定义卷积操作
def conv_block(ipt, num_filter, groups, dropouts):
    return fluid.nets.img_conv_group(
        input=ipt,
        pool_size=2
```

```
            pool_stride=2,
            conv_num_filter=[num_filter] * groups,          #过滤器的个数
            conv_act='relu',
            conv_with_batchnorm=True,           #表示在 Conv2d Layer 之后是否使用 BatchNorm
            conv_batchnorm_drop_rate=dropouts,          #表示 BatchNorm 之后的 Dropout Layer 的丢弃概率
            pool_type='max'         #最大池化
        )

    #VGG 网络
    def vgg_bn_drop(image, type_size):
        conv1 = conv_block(image, 64, 2, [0, 0])
        conv2 = conv_block(conv1, 128, 2, [0, 0])
        conv3 = conv_block(conv2, 256, 3, [0, 0, 0])
        conv4 = conv_block(conv3, 512, 3, [0, 0, 0])
        conv5 = conv_block(conv4, 512, 3, [0, 0, 0])

        drop = fluid.layers.dropout(x=conv5, dropout_prob=0.5)
        fc1 = fluid.layers.fc(input=drop, size=512, act=None)
        bn = fluid.layers.batch_norm(input=fc1, act='relu')
        drop2 = fluid.layers.dropout(x=bn, dropout_prob=0.0)
        fc2 = fluid.layers.fc(input=drop2, size=512, act=None)
        predict = fluid.layers.fc(input=fc2, size=type_size, act='softmax')
        return predict

    def convolutional_neural_network(img):
        #第一个卷积-池化层
        conv1 = fluid.layers.conv2d(
            input=img,          #输入图像
            num_filters=20,       #卷积核数量，与输出的通道相同
            filter_size=5,          #卷积核大小
```

```
        act='relu'
    )

    pool1 = fluid.layers.pool2d(
        input=conv1,  # 输入
        pool_size=2,  # 池化核大小
        pool_type='max',  # 采用最大池化
        pool_stride=2)  # 池化步长
    conv_pool_1 = fluid.layers.batch_norm(pool1)
    # 第二个卷积-池化层
    conv2 = fluid.layers.conv2d(input=conv_pool_1,
                                num_filters=50,
                                filter_size=5,
                                act="relu")
    pool2 = fluid.layers.pool2d(
        input=conv2,
        pool_size=2,
        pool_type='max',
        pool_stride=2,
        global_pooling=False)
    conv_pool_2 = fluid.layers.batch_norm(pool2)
    # 第三个卷积-池化层
    conv3 = fluid.layers.conv2d(input=conv_pool_2,
                                num_filters=50,
                                filter_size=5,
                                act="relu")
    pool3 = fluid.layers.pool2d(
        input=conv3,
        pool_size=2,
        pool_type='max',
        pool_stride=2,
```

```
            global_pooling=False)
        # 以 softmax 为激活函数的全连接输出层，13 类药盒，所以 size=13
        prediction = fluid.layers.fc(input=pool3,
                                     size=3,
                                     act='softmax')
    return prediction
```

4.1.3 定义损失函数和优化算法

因为训练集输入标签已经确定，真实的概率分布也就可以确定，所以信息熵为一个常量。由于相对熵的值表示真实概率分布与预测概率分布之间的差异，值越小表示预测的结果越好，所以需要最小化相对熵，而交叉熵等于相对熵加上一个常量（信息熵），且公式相比相对熵更加容易计算，所以可以使用交叉熵损失函数来计算 loss。代码如下：

```
cost = fluid.layers.cross_entropy(input=predict, label=label)
avg_cost = fluid.layers.mean(cost)   #求平均值
accuracy = fluid.layers.accuracy(input=predict, label=label) #准确率函数
```

Adam 优化器实现简单，计算高效，对内存需求少，并且参数更新不受梯度的伸缩变换的影响，更新步长也可以限制在大致范围内，所以可以选择 Adam 优化算法。以下代码是利用 PaddlePaddle 框架构建 Adam 优化算法：

```
optimizer = fluid.optimizer.Adam(learning_rate=0.0001)
optimizer.minimize(avg_cost)
```

4.2 模型训练

代码如下：

```
  for batch_id, data in enumerate(eval_reader()):   # 遍历 test_reader
        test_cost,test_acc = exe.run(program=fluid.default_main_program(),   #
        #运行测试主程序
        feed=feeder.feed(data),   # 喂入一个 batch 的数据
        fetch_list=[avg_cost, accuracy])   # fetch 均方误差、准确率
        test_accs.append(test_acc[0])   # 记录每个 batch 的准确率
        test_costs.append(test_cost[0])   # 记录每个 batch 的误差
        all_test_iter = all_test_iter + BATCH_SIZE
        all_test_iters.append(all_test_iter)
```

```
            all_test_costs.append(test_cost[0])
            all_test_accs.append(test_acc[0])

            test_cost = (sum(test_costs) / len(test_costs))  # 每轮的平均误差
            test_acc = (sum(test_accs) / len(test_accs))  # 每轮的平均准确率
            print('Test:%d, Cost:%0.5f, ACC:%0.5f' % (pass_id, test_cost,
test_acc))
```

4.3 模型评估

由于受电脑配置限制，所训练最大轮数为 32 轮，经过 32 轮训练后模型的训练结果如下：

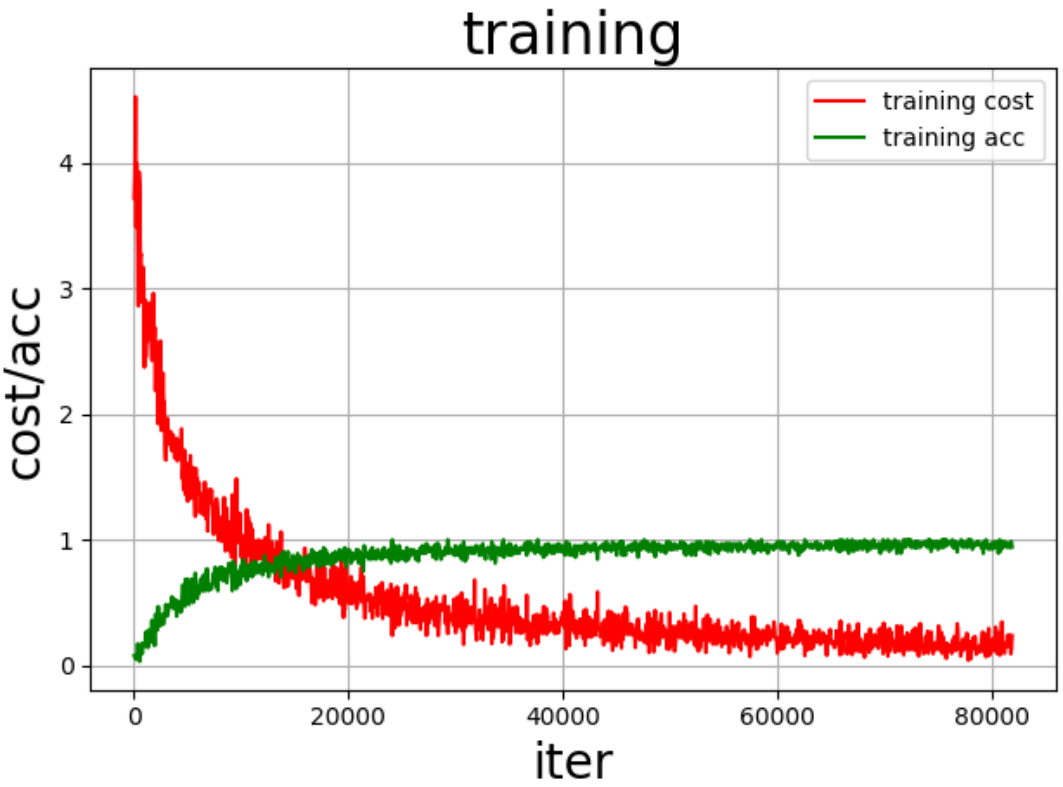


图 4-1 训练集曲线图

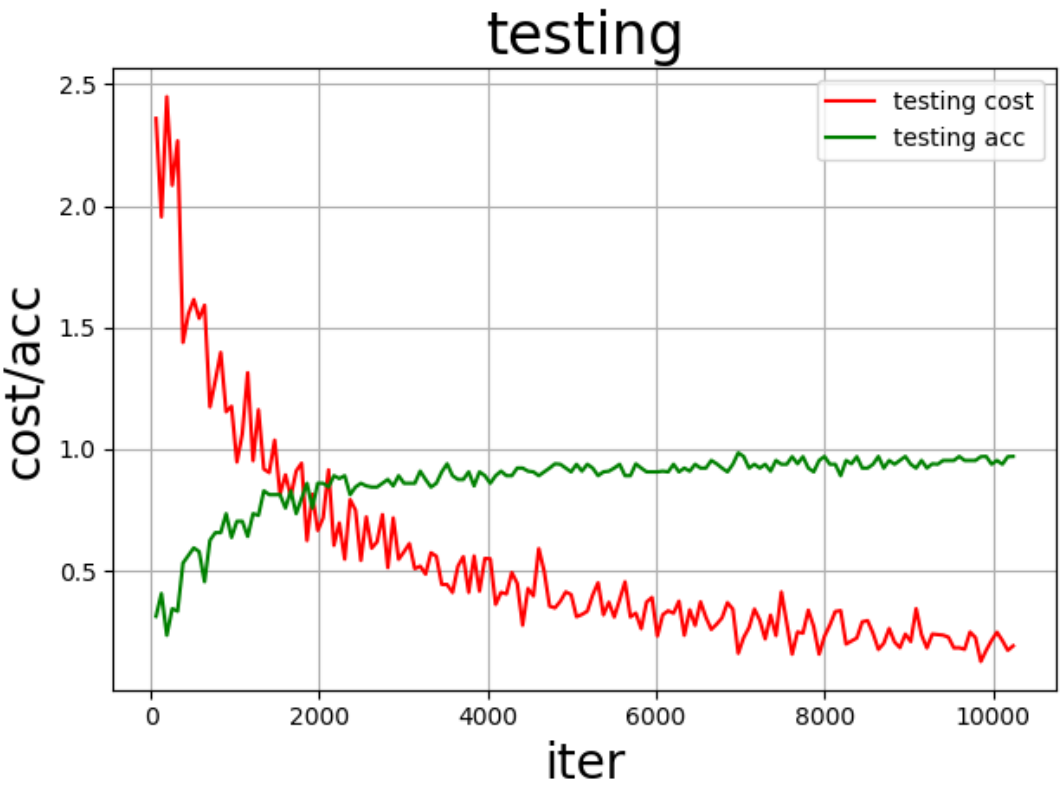


图 4-2 测试集曲线图

经过 32 轮训练后模型的准确率可以达到 90%以上。

模拟药店采购场景，喂入数据进行测试：

(1) 假设消费者选购以下三种药品：

图 4-3 模拟场景一

输出结果为：

```
"E:\Program Files (x86)\code\Anaconda\python.exe" E:/LunWen/data/TEST.py
您选购了：  炎立消胶囊(20元/盒) 1盒   桂林片(30元/盒) 1盒   盐酸氨溴索胶囊(15元/盒) 1盒
共计65元

Process finished with exit code 0
```

图 4-4 实验结果一

假设选购以下药品：

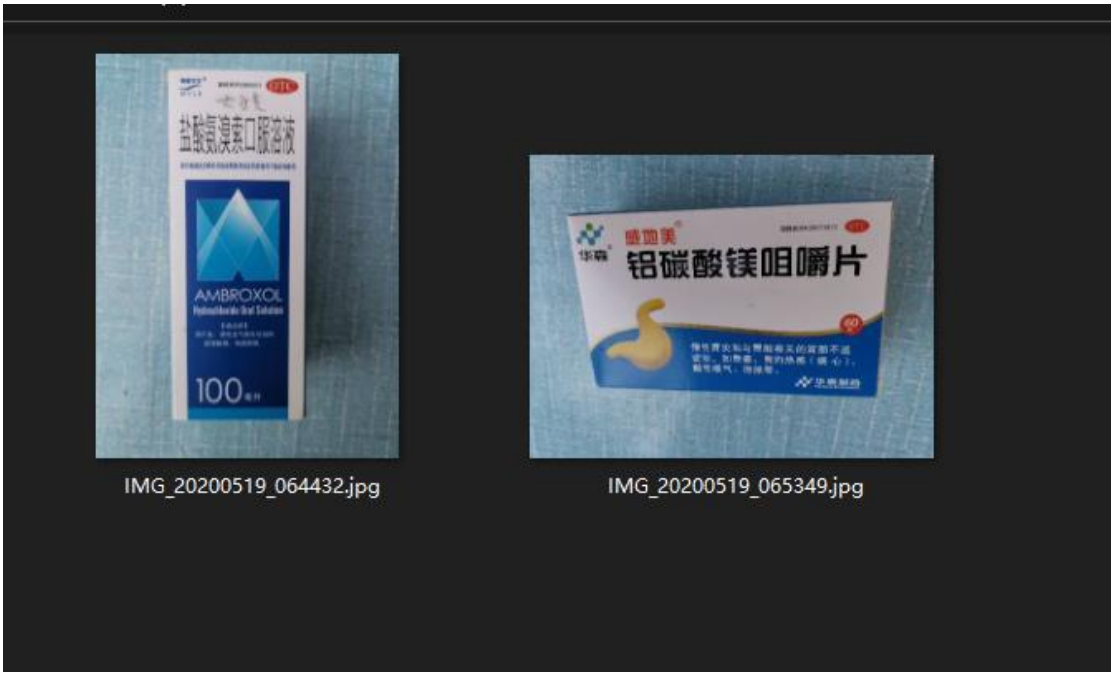


图 4-5 模拟场景二

输出结果为：

```
"E:\Program Files (x86)\code\Anaconda\python.exe" E:/LunWen/data/TEST.py
您选购了：  盐酸氨溴索口服溶液(40元/盒) 1盒   铝碳酸镁咀嚼片(30元/盒) 1盒
共计70元
```

图 4-6 实验结果二

假设选购以下药品：

图 4-7 模拟场景三

输出结果为：

```
"E:\Program Files (x86)\code\Anaconda\python.exe" E:/LunWen/data/TEST.py
您选购了：  益肺胶囊(40元/盒) 3盒
共计120元

Process finished with exit code 0
```

图 4-8 实验结果三

4.4 本章小结

在本章中，就模型的构建，训练，以及评估做了简要论述。通过对模型训练结果的分析发现，虽然由于电脑的性能限制，训练模型需要花费较长时间，但在成功跑完 32 轮后，训练结果基本满足实验需求。

5.总结与展望

本文通过研究学习现有的盒装药品识别方法，总结了多种药品识别方式及其优缺点，选择采用深度学习方法解决盒装药品的识别问题。本文主要内容包括：

(1)通过查阅相关资料并结合当前生活实际，介绍了盒装药品识别研究背景与意义。通过对常用的条形码识别，RFID 识别的分析，并总结它们各自的优缺点，为提高消费者购物体现，节约人工成本，提高模型准确率，决定采用基于深度学习的盒装药品识别方法。

(2)对传统图像分类技术进行概述，对深度学习在图像分类问题的应用进行总结，归纳了目前现有的深度学习模型，并对卷积神经网络的原理和发展进行了详细介绍。

(3)简单介绍了 PaddlePaddle 和 PIL 库，构建了盒装药品数据集并利用 PIL 对数据集进行划分。

(4)利用 paddlepaddle 构建 vgg 网络，通过对损失函数和优化算法的研究，选取了合适的损失函数和优化算法，最终模型准确度达 90%以上

虽然本文的基于深度学习盒装药品识别方法在现有数据集上分类准确，但在很多地方依然可以优化和改进：

(1)盒装药品种类不够多，训练样本和实际生活中的情况存在一定差异。

(2)代码和算法方面还需要进一步优化。尤其是本文仅使用了 vgg 网络模型，并没有探究其他模型在该数据集上的分类效果。

(3)问题研究背景过于理想化，本文只考虑了一张图片中只有一种药品，如果可以实现检测一张图片中有多种药品，将为大大提高识别效率。

致谢

回首大学 4 年生活，有很多精彩片段印在了脑海深处，往昔浮现眼前，心中百般滋味，万语千言却又只想说声感谢。首先感谢我的毕业设计指导老师芦碧波老师，本文是在您悉心指导下完成的。芦老师治学严谨，诲人不倦，每每当我遇到棘手的问题时，芦老师总能一针见血，指出症结之所在，总能给我新的思路和启发。

其次，我想感谢我亲爱室友和朋友们：梅剑书、李岚森、朱柿林、吕凯勋、罗强、管海丹、何鹏程、徐行麒，大学 4 年，我们有很多共同的美好回忆。无论在生活中还是学习中，你们都为我提供了很多帮助，感谢你们。

最后，我想感谢我的家人，感谢你们的相信，理解和支持！

参考文献

[1] 何东梅.细粒度物体分类算法研究与实现[D].北京:北京交通大学,2016.
[2] 赵星.基于深度卷积神经网络的细粒度图像识别与分类算法研究[D].合肥:安徽大学, 2018.
[3] 王雅静,窦震海.条码识别技术的研究[J].包装工程,2008(08):240-241+244.
[4] 王沁.PDF417 条码识别技术研究[D].西安理工大学,2007.
[5] 张佳.条码技术在库存管理中的应用研究[D].昆明理工大学,2012.
[6] 宋红蕊.商品条码的识别[J].中国品牌与防伪,2006(03):76-77.
[7] 徐妮.条码识别技术在快递分拣中的应用[D].西安科技大学,2017.
[8] 陈新河.无线射频识别(RFID)技术发展综述[J].信息技术与标准化, 2005(7):20-24.
[9] 何泉江,夏林.无线射频识别技术应用综述[J]. 现代建筑电气, 2011(8):1-4.
[10] 杨永永.基于 RFID 技术的数字化仓储的研究[D].浙江理工大学,2017.
[11] 石百仟.基于 RFID 的智慧图书馆管理系统[D].吉林大学,2017.
[12] 倪霖.基于 RFID 的汽车生产线信息集成模式及关键技术研究[D].重庆大学,2010.
[13] 王丽科.基于 RFID 的超市购物数据分析算法研究[D].太原理工大学,2017.
[14] 张智.面向物联网的多层次无线感知识和识别系统[D].浙江大学,2012.
[15] S. Feng, C. L. P. Chen. A fuzzy restricted boltzmann machine: Novel learning algorithms based on the crisp possibilistic mean value of fuzzy numbers[J]. IEEE Transactions on Fuzzy Systems, 2018, 26(1):117-130.
[16] C. L. P. Chen, C.-Y. Zhang, L. Chen, et al. Fuzzy restricted boltzmann machine for the enhancement of deep learning[J]. IEEE Transactions on Fuzzy Systems, 2015, 23(6):2163-2173.
[17] Z. Yu, H. Chen, J. Liu, et al. Hybrid K-nearest neighbor classifier[J]. IEEE Transactions on Cybernetics, 2016, 46(6):1263-1275.
[18] 曾志,吴财贵,唐权华,余嘉禾,李雅晴,高健.基于多特征融合和深度学习的商品图像分类[J].计算机工程与设计,2017,38(11):3093-3098.
[19] J.Tang, C. Deng, G.-B. Huang. Extreme learning machine for multilayer perceptron[J]. IEEE Transactions on Neural Networks and Learning Systems, 2016, 27(4):809-821.
[20] M. Gong, J. Liu, H. Li, et al. A multiobjective sparse feature learning model for deep neural networks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2015, 26(12):3263-3277.
[21] M. Leshno, V. Y. Lin, A. Pinkus, et al. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function[J]. Neural Networks, 1993, 6(6):861-867.
[22] B. Igelnik, Y.-H. Pao. Stochastic choice of basis functions in adaptive function approximation and the functional-link net[J]. IEEE Transactions on Neural Networks, 1995, 6(6):1320-1329.
[23] C. L. P. Chen, C.-Y. Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data[J]. Information Sciences, 2014, 275:314-347.