模式识别与机器学习

课程报告

姓　　名：刘瑾瑾

班　　级：231202

学　　号：20201000128

指导老师：陆承达

二 0 二三年十一月

摘　要

随着 BP 神经网络和卷积神经网络的快速发展，基于人工神经网络的图像识别技术赋予了人工智能设备“看”的能力。本文综合探讨了 BP 神经网络和卷积神经网络在图像识别领域的发展现状和应用。通过深入研究这两种神经网络的原理和特点，深入分析了它们在图像分类任务中的优势和限制。特别是，本研究使用了 VGG-11 网络作为实验模型，并在经典的 cifar-10 数据集上进行了图像分类任务。通过实验结果的分析，训练集的准确率为 95%，测试集的准确率为 85%，展示了卷积神经网络在图像识别方面的卓越性能，特别是在处理复杂图像和大规模数据集时的优势。

关键词：BP 神经网络；卷积神经网络；图像分类

ABSTRACT

With the rapid development of BP neural networks and convolutional neural networks, artificial neural network-based image recognition technology has endowed artificial intelligence devices with the capability to "see." This article comprehensively explores the current state and applications of BP neural networks and convolutional neural networks in the field of image recognition. By delving into the principles and characteristics of these two neural networks, it provides an in-depth analysis of their advantages and limitations in image classification tasks. In particular, this study utilized the VGG-11 network as an experimental model and conducted image classification tasks on the classic cifar-10 dataset. Through the analysis of experimental results, the training set achieved an accuracy of 95%, while the test set reached an accuracy of 85%, showcasing the outstanding performance of convolutional neural networks in image recognition, especially when dealing with complex images and large-scale datasets.

Keywords： Backpropagation Neural Network； Convolutional Neural Network； Image Classification

目录

一、简介 1
二、基于 BP 神经网络的图像识别技术 1
2.1 算法原理 1
2.2 基于 BP 神经网络的图像识别技术发展现状 2
2.3 BP 神经网络在图像识别中应用 3
三、基于卷积神经网络的图像识别技术 3
3.1 卷积神经网络的图像识别原理 3
3.2 基于卷积神经网络的图像识别技术发展现状 4
3.3 卷积神经网络在图像识别中的应用 5
四.基于 VGG-11 卷积神经网络的图像识别应用 6
4.1 内容简介 6
4.2 算法原理与分析 6
4.3 所使用数据集简介 8
4.4 神经网络的模型参数设计 10
4.5 神经网络模型的训练与测试 10
4.6 结果分析与讨论 12
4.7 总结 14
参考文献 15
附：问题回答 16

一、简介

图像识别，是指利用计算机对图像进行处理、分析和理解，以识别各种不同模式的目标和对象的技术，是应用深度学习算法的一种实践应用。它利用计算机对图像进行定量分析，把图像或图像中的每个像元或区域划归为若干个类别中的某一种，以代替人的视觉判读。

图像识别是一种特定的模式识别任务，其目标是将输入的图像分为预定义的类别或标签。这意味着将图像与一个或多个类别相关联，使计算机能够识别图像中的对象、场景或特征。图像分类强调了对图像内容的语义理解，如识别猫、狗、车辆等物体。

目前，基于神经网络的图像识别是一种比较新型的技术，是以传统图像识别方式为基础，有效融合神经网络算法。针对基于神经网络的图像识别技术，主要有两类：一类是基于 BP 神经网络（Backpropagation Neural Network）的图像识别技术，另一类是基于卷积神经网络（Convolutional Neural Network，CNN）的图像识别技术。

二、基于 BP 神经网络的图像识别技术

2.1 算法原理

BP 神经网络是一种按照误差反向传播算法训练的多层前馈神经网络，是应用最广泛的神经网络模型之一。

人工神经网络无需事先确定输入输出之间映射关系的数学方程，仅通过自身的训练，学习某种规则，在给定输入值时得到最接近期望输出值的结果。作为一种智能信息处理系统，人工神经网络实现其功能的核心是误差反向传播算法，它的基本思想是梯度下降法，利用梯度搜索技术，以期使网络的实际输出值和期望输出值的误差均方差为最小。

基本 BP 算法包括信号的前向传播和误差的反向传播两个过程。即计算误差输出时按从输入到输出的方向进行，而调整权值和阈值则从输出到输入的方向进行。正向传播时，输入信号通过隐含层作用于输出节点，经过非线性变换，产生输出信号，若实际输出与期望输出不相符，则转入误差的反向传播过程。误差反

传是将输出误差通过隐含层向输入层逐层反传，并将误差分摊给各层所有单元，以从各层获得的误差信号作为调整各单元权值的依据。通过调整输入节点与隐层节点的联接强度和隐层节点与输出节点的联接强度以及阈值，使误差沿梯度方向下降，经过反复学习训练，确定与最小误差相对应的网络参数（权值和阈值），训练即告停止。此时经过训练的神经网络即能对类似样本的输入信息，自行处理输出误差最小的经过非线形转换的信息。

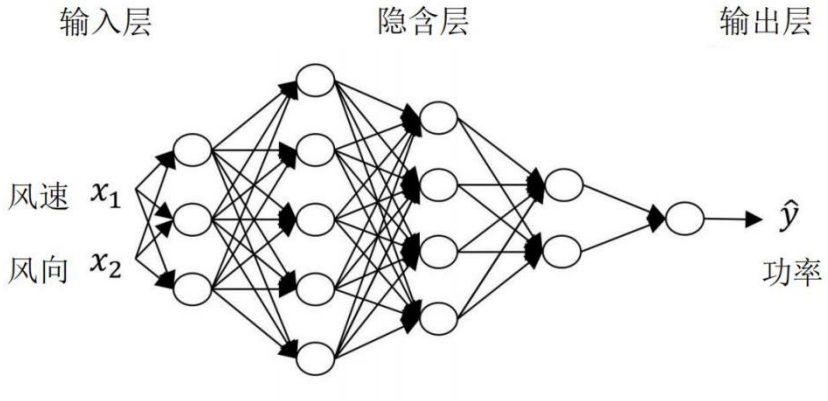


图 1　BP 神经网络结构图

2.2 基于 BP 神经网络的图像识别技术发展现状

将 BP 算法用于具有非线性转移函数的三层前馈网络,可以以任意精度逼近任何非线性函数一,这一非凡优势使三层前馈网络得到越来越广泛的应用。然而标准的 BP 算法在应用中暴露出不少内在的缺陷。

1)收敛速度慢;为保证算法的收敛性，学习率 q 必须小于一个上限。这就决定了 BP 算法的收敛速度不可能较快，并且越是接近极小值处，由于梯度变化值逐渐趋于零，算法的收敛速度就越慢;

2)所得的网络容错能力差;

3)算法不完备，易陷入局部极小。不能保证收敛到全局最小点。实际问题的求解空间往往是极其复杂的多维曲面，存在着很多局部极小点，使得陷于局部极小的可能性大大增加，这使权值的初始值选择对网络学习结果有较大的影响，通过随机设置的初始权值经训练而达到全局最优点较难;

4)网络隐含层层数及隐含层单元数的选取尚无理论上的指导.而是根据经验确定。因此，网络往往有很大的冗余性，无形中增加了网络学习的时间。

如果增加 BP 神经网络隐含层的数量,可以降低误差,提高精度,但同时也使 BP 网络学习收敛时间变慢，使学习效率下降。BP 算法改进的主要目标是既能加快训练速度和网络收敛速率，又能使程序避免陷入局部极小值,常见的对 BP 网络改进方法有带动量因子算法、自适应学习速率、弹性 BP 算法、LM 算法、正

则化方法以及作用函数后缩法等，而对 BP 极小值问题的改进方法有合理选择初始权值和阈值、模拟退火算法、遗传算法和改进激活函数等。

我们通常把网络中的节点比拟成人脑神经元，如同人记实际事物一样，BP 神经网络在样本数据学习中，节点系数权值的改变就像是人脑神经元轴突与树突连接阈值的改变，神经网络会把权系数调整到最佳；从而使分类做到最好。作为一个整体结构，BP 神经网络按整个对象的特征向量来记忆图像的，只要大多数特征值符合曾学习过的训练样本数据特征，就可识别为同一类别，所以当样本存在较大噪声时神经网络分类器仍可正确识别。在图像识别领域，只要将图像的点矩阵向量作为神经网络的输入，经过网络的计算，BP 神经网络就能输出识别结果。

2.3 BP 神经网络在图像识别中应用

目前，基于 BP 神经网络的图像处理技术已经在各个行业普遍应用，在人脸识别、车牌识别等领域被广泛应用。诸如智能汽车监控中采用的拍照识别技术，若有汽车从该位置经过时，检测设备将产生相应的反应，检测设备启动图像采集装置，获取汽车正反面的特征图像，在对车牌字符进行识别的过程中，就采用了基于神经网络和模糊匹配的两类算法。

三、基于卷积神经网络的图像识别技术

3.1 卷积神经网络的图像识别原理

卷积神经网络 CNN 是一种前馈神经网络，人工神经元可以响应周围单元，可以进行大型图像处理。卷积神经网络包括卷积层和池化层。卷积神经网络是受到生物思考方式启发的多层感知器，它有着不同的类别层次，并且各层的工作方式和作用也不同。

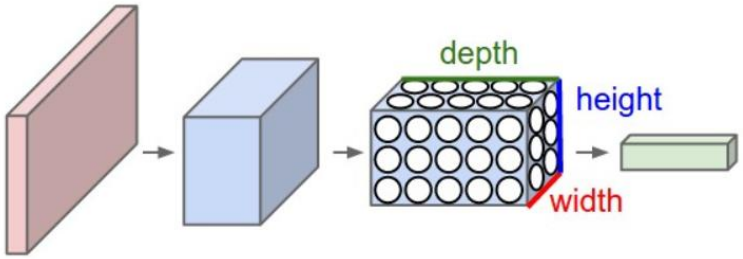


图 2 CNN 网络结构

CNN 网络一共有 5 个层级结构：输入层、卷积层、激活层、池化层、全连接层。

（1）输入层

与传统神经网络/机器学习一样，模型需要输入的进行预处理操作，常见的3种预处理方式有：去均值、归一化、PCA/SVD降维等。

（2）卷积层

局部感知：人的大脑识别图片的过程中，并不是一下子整张图同时识别，而是对于图片中的每一个特征首先局部感知，然后更高层次对局部进行综合操作，从而得到全局信息。通过局部感知特性，大大减少了模型的计算参数。但是仅仅这样还是依然会有很多参数。权值共享机制：同一层下的神经元的连接参数只与特征提取的有关，而与具体的位置无关，因此可以保证同一层中所有位置的连接是权值共享的。

（3）激活层

所谓激励，实际上是对卷积层的输出结果做一次非线性映射。常用的激励函数有:Sigmoid函数、Tanh函数、ReLU函数、Leaky ReLU函数、Maxout函数。

（4）池化层

池化：也称为欠采样或下采样。主要用于特征降维，压缩数据和参数的数量，减小过拟合，同时提高模型的容错性。主要有：最大池化Max Pooling和平均池化Average Pooling。

（5）全连接层

经过若干次卷积+激励+池化后，终于来到了输出层，模型会将学到的一个高质量的特征图片全连接层。其实在全连接层之前，如果神经元数目过大，学习能力强，有可能出现过拟合。因此，可以引入dropout操作，来随机删除神经网络中的部分神经元，来解决此问题。当来到了全连接层之后，可以理解为一个简单的多分类神经网络（如：BP神经网络），通过softmax函数得到最终的输出。

3.2 基于卷积神经网络的图像识别技术发展现状

目前卷积神经网络的设计思路基本上朝着更深，更宽（如Xception网络），更多支路，更轻量以及更有效的卷积方式等多个方向同时发展。目前。由于深度学习在图像识别的任务已经超过人类，卷积神经网络能够很好地从输入映射到隐层地特征表达，并且能够层级式地提取特征并通过最后通过内嵌的分类网络完成分类任务。并且卷积神经网络通过轻量话网络或者模型压缩能够在嵌入式或者移动端运行，已慢慢从实验室走向更多的商业化应用。

卷积神经网络在计算机视觉方面的应用，包括图像的分类识别、视频识别等都有着明显的优势，但是图像识别在实际运用中仍然存在一些挑战。

1）图像是识别的基础数据，图像识别中首要的挑战就是模糊图像、受环境影响的图像(如受光线、噪声影响)、遮挡的图像等情况。

2）卷积神经网络在检测中需要对数据进行标注，一个模型的训练过程中往往需要大量手工标注的数据，随着大规模数据量的涌现，无标签的未知数据占绝大多数，为这些数据做标签已经变得不够现实了。我们必须学会从无标注的数据里面进行学习，现有的研究方法包括生成对抗网络、主动学习等.

3）非欧空间数据不存在平移不变性，采用图卷积神经网络可以处理图数据。在该领域中，目前存在的主要方法为谱方法和空间方法，研究表明，虽然图卷积神经网络取得了一定的成果，但仍然有很多问题需要解决。

4）目前在深度学习中训练网络模型需要大量的数据样本，如何在少量样本情况下即保证识别精度又能大大提高网络的训练速度是很多研究者的一个重要目标。数据扩充是一项非常重要的技术，其可以从现有的数据中产生更多的有用数据。神经网络中Dropout的引入使得样本不足的条件下也能够比较高的识别率，文献提出采用滑动窗口技术增加训练数据的方法. Chen 等人提出了一种 Grid Mask 的数据扩增策略，该策略删除均匀分布的区域，最后形成网络形状，使用此形状删除信息比设置完全随机位置更加有效.

总之，CNN 目前还存在很多待解决的问题，这些问题不影响在各领域中图像识别的运用和发展．仍然是研究的一大热点。

3.3 卷积神经网络在图像识别中的应用

在车牌识别中，特征提取和分类识别是图像识别的重点和关键，识别算法主要集中于特征提取任务，一般涉及大量的人工标记工作量和高昂的人工费用，且人工标记效率低。针对人工图像识别存在的弊端，可借助卷积神经网络实现端对端的图像分类识别。近年来，随着车辆识别技术不断发展，新的车牌识别技术不断涌现，如模板匹配法、特征匹配法、机器学习识别法等。卷积神经网络(CNN)是深度学习神经网络的代表算法之一，能够有效将大数据量图片降维为小数据量，在有效保留图片特征的前提下将大数据量图片降维为小数据量，目前已广泛应用于人脸识别、自动驾驶、安防等领域。

如今，由于最新的卷积神经网络在某种程度上解决了计算机视觉领域特征表达的问题，卷积神经网开始在诸多研究方向如目标检测、图像分割、实例分割、图像生成、人脸识别、车辆识别和人体姿态估计等大方光彩，取得的研究成果也是远超传统算法。

四.基于 VGG-11 卷积神经网络的图像识别应用

4.1 内容简介

（1）目的：

实现在 ciffar-10 数据集上的图像分类,同时该模型可以用于其他图像的分类识别，只需传入相应的训练集进行训练，保存为另一个模型即可，进行调用使用。

（2）配置环境：

pycharm(python3.9)，torch，torchvision

（3）知识预备：

了解 anaconda 环境配置，卷积神经网络的基本结构，torch、torchvision 包的内置函数的使用

4.2 算法原理与分析

4.2.1 模型训练原理的分析

BP 算法(即误差反向传播算法)适合于多层神经元网络的一种学习算法，它建立在梯度下降法的基础上。BP 网络的输入输出关系实质上是一种映射关系：一个 n 输入 m 输出的 BP 神经网络所完成的功能是从 n 维欧氏空间向 m 维欧氏空间中一有限域的连续映射，这一映射具有高度非线性。它的信息处理能力来源于简单非线性函数的多次复合，因此具有很强的函数复现能力。这是 BP 算法得以应用的基础。

反向传播算法主要由两个环节(激励传播、权重更新)反复循环迭代，直到网络的对输入的响应达到预定的目标范围为止。

BP 算法的学习过程由正向传播过程和反向传播过程组成。在正向传播过程中，输入信息通过输入层经隐含层，逐层处理并传向输出层。如果在输出层得不到期望的输出值,则取输出与期望的误差的平方和作为目标函数,转入反向传播,逐层求出目标函数对各神经元权值的偏导数，构成目标函数对权值向量的梯量,作为修改权值的依据,网络的学习在权值修改过程中完成。误差达到所期望值时,网络学习结束。

（1）激励传播

每次迭代中的传播环节包含两步：

① (前向传播阶段)将训练输入送入网络以获得激励响应；

② (反向传播阶段)将激励响应同训练输入对应的目标输出求差,从而获得隐层和输出层的响应误差。

（2）权重更新

对于每个突触上的权重，按照以下步骤进行更新：

① 将输入激励和响应误差相乘，从而获得权重的梯度；

② 将这个梯度乘上一个比例并取反后加到权重上；

③ 这个比例将会影响到训练过程的速度和效果，因此称为“训练因子”。梯度的方向指明了误差扩大的方向，因此在更新权重的时候需要对其取反，从而减小权重引起的误差。

4.2.2 网络架构

本网络模型使用的是经典的 VGG-11 架构。VGG 网络可以分为两部分：第一部分主要由卷积层和汇聚层组成，第二部分由全连接层组成，VGG 网络结构如图：

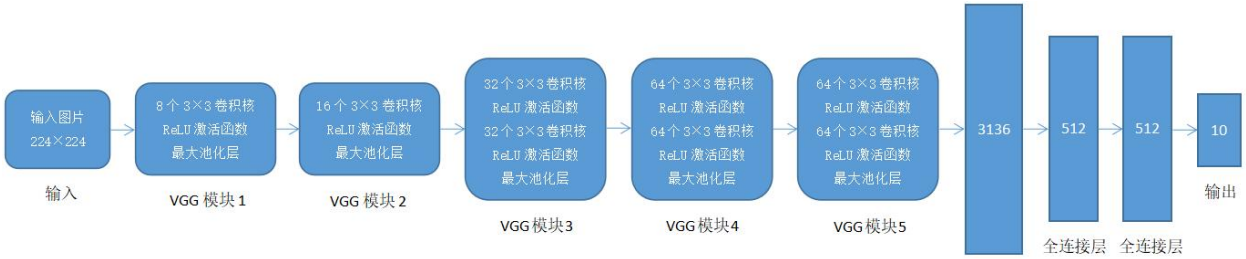


图 3 VGG-11 模型结构图

原始 VGG 网络有 5 个卷积块，其中前两个块各有一个卷积层，后三个块各包含两个卷积层。 第一个模块有 64 个输出通道，每个后续模块将输出通道数量翻倍，直到该数字达到 512。由于该网络使用 8 个卷积层和 3 个全连接层，因此它通常被称为 VGG-11，VGG 网络块的代码如下：

```
def vgg_block(num_convs, in_channels, out_channels):
    layers = []
    for _ in range(num_convs):
        layers.append(nn.Conv2d(in_channels, out_channels,
                                kernel_size=3, padding=1))
        layers.append(nn.ReLU())
        in_channels = out_channels
    layers.append(nn.MaxPool2d(kernel_size=2,stride=2))
    return nn.Sequential(*layers)
```

该函数有三个参数，分别对应于卷积层的数量 num_convs、输入通道的数量 in_channels 和输出通道的数量 out_channels. 这三个参数由 conv_arch 指定

conv_arch = ((1, 64), (1, 128), (2, 256), (2, 512), (2, 512))

下面的代码实现了 VGG-11，通过 for 循环实现对 conv_arch 的提取

```
def vgg(conv_arch):
    conv_blks = []
    in_channels = 3
```

```
    # 卷积层部分
    for (num_convs, out_channels) in conv_arch:
        conv_blks.append(vgg_block(num_convs, in_channels, out_channels))
        in_channels = out_channels

    return nn.Sequential(
        *conv_blks, nn.Flatten(),
        # 全连接层部分
        nn.Linear(out_channels * 3 * 3, 4096), nn.ReLU(), nn.Dropout(0.4),
        nn.Linear(4096, 4096), nn.ReLU(), nn.Dropout(0.4),
        nn.Linear(4096, 10))
```

4.3 所使用数据集简介

（1）数据集简介

CIFAR-10 是由 Hinton 的学生 Alex Krizhevsky 和 Ilya Sutskever 整理的一个用于识别普适物体的小型数据集。CIFAR-10 的图片样例如图所示：

图 4　CIFAR-10 数据集示意图

数据集由 6 万张 32×32 的彩色图片组成，一共有 10 个类别。每个类别 6000 张图片。其中有 5 万张训练图片及 1 万张测试图片。数据集被划分为 5 个训练块和 1 个测试块，每个块 1 万张图片。测试块包含了 1000 张从每个类别中随机选择的图片。训练块包含随机的剩余图像，但某些训练块可能对于一个类别的包含多于其他类别，训练块包含来自各个类别的 5000 张图片。这些类是完全互斥的，及在一个类别中出现的图片不会出现在其它类中。

一共包含 10 个类别的 RGB 彩色图 片：飞机（airplane）、汽车（automobile）、鸟类（bird）、猫（cat）、鹿（deer）、狗（dog）、蛙类（frog）、马（horse）、

船（ship）和卡车（truck）。图片的尺寸为 32×32 ，数据集中一共有 50000 张训练图片和 10000 张测试图片。

（2）数据集下载以及读取

```
def load_data_cifar_10(batch_size, resize=None):
    """Download the cifar_10 dataset and then load it into memory."""
    trans = [transforms.ToTensor()]
    if resize:
        trans.insert(0, transforms.Resize(resize))
    trans = transforms.Compose(trans)
    mnist_train = torchvision.datasets.CIFAR10(
        root="../data", train=True, transform=trans, download=True)
    mnist_test = torchvision.datasets.CIFAR10(
        root="../data", train=False, transform=trans, download=True)
    return (data.DataLoader(mnist_train, batch_size, shuffle=True,
                            num_workers=get_dataloader_workers()),
            data.DataLoader(mnist_test, batch_size, shuffle=False,
                            num_workers=get_dataloader_workers()))

def show_images(imgs, num_rows, num_cols, titles=None, scale=1.5):    #@save
    """绘制图像列表"""
    figsize = (num_cols * scale, num_rows * scale)
    _, axes = d2l.plt.subplots(num_rows, num_cols, figsize=figsize)
    axes = axes.flatten()
    # print("axes 的大小为{},axes 的 type 为{}".format(axes.shape,axes.astype))
    # imgs = imgs.reshape(6,3,32,32)
    for i, (ax, img) in enumerate(zip(axes, imgs)):
        # print('img1 shape = {}'.format(img.shape))
        if torch.is_tensor(img):
            #  图片张量
            if img.ndim == 3:
                img = img.numpy().transpose(1,2,0)
            if img.ndim == 2:
                img = img.numpy()
            ax.imshow(img)
            # print('img2 shape = {}'.format(img.shape))
        else:
            # PIL 图片
            ax.imshow(img)
            # print('is pil')
        ax.axes.get_xaxis().set_visible(False)
        ax.axes.get_yaxis().set_visible(False)
        if titles:
            ax.set_title(titles[i])
    plt.show()
```

```
    return axes
def get_ciffar_labels(labels):    #@save
    """返回 ciffar 数据集的文本标签"""
    text_labels = ['airplane', 'automobile', 'bird', 'cat', 'deer',
                   'dog', 'frog', 'horse', 'ship', 'truck']
    return [text_labels[int(i)] for i in labels]

X, y = next(iter(data.DataLoader(mnist_train, batch_size=20)))
show_images(X.reshape(20, 3, 32, 32), 4, 5, titles=get_ciffar_labels(y))
```

4.4 神经网络的模型参数设计

(1) 输入图片大小 resize=112×112
(2) 通道压缩因子 ratio=0.25
(3) 迭代次数 epochs=20
(4) 批量大小 batch_szie=128
(5) 权重衰退 weight_decay=0.001，
(6) 动量因子 momentum=0.9
(7) 初始学习率 base_lr=0.1。
(8) Droupout=0.4

4.5 神经网络模型的训练与测试

（1）丢弃法

丢弃法是 ImageNet 中提出的一种方法，通俗一点讲就是在训练的时候让神经元以一定的概率不工作，防止过拟合。

（2）学习率指数衰减

指数衰减学习率是先使用较大的学习率来快速得到一个较优的解，然后随着迭代的继续,逐步减小学习率，使得模型在训练后期更加稳定。

（3）SGD 随机梯度下降法

在深度学习中，目标函数通常是训练数据集中每个样本的损失函数的平均值。给定个样本的训练数据集，我们假设$f_i(\mathbf{x})$是关于索引 i 的训练样本的损失函数，其中 x 是参数向量。然后我们得到目标函数:

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x})$$

x 的目标函数的梯度计算为:

$$\nabla f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x})$$

如果使用梯度下降法，则每个自变量迭代的计算代价会随线性增长。因此，当训练数据集较大时，每次迭代的梯度下降计算代价将较高。

随机梯度下降（SGD）可降低每次迭代时的计算代价。在随机梯度下降的每

次迭代中，我们对数据样本随机均匀采样一个索引 i，并计算梯度$\nabla f_i(\mathbf{x})$，以更新 x，其中η为学习率。

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f_i(\mathbf{x})$$

（4）损失函数

交叉熵的公式如下：

$$H(p,q) = -\sum_{i=1}^{n} p(x_i)\log\ (q(x_i))$$

在深度学习训练网络时，输入数据与标签常常已经确定，那么真实概率分布 P (x)也就确定下来了，所以信息熵在这里就是一个常量。由于 KL 散度的值表示真实概率分布 P (x)与预测概率分布 Q(x)之间的差异，值越小表示预测的结果越好，所以需要最小化 KL 散度，而交叉熵等于 KL 散度加上一个常量（信息熵），且公式相比 KL 散度更加容易计算，所以在深度学习中常常使用交叉熵损失函数来计算 loss。

交叉熵能够衡量同一个随机变量中的两个不同概率分布的差异程度，在机器学习中就表示为真实概率分布与预测概率分布之间的差异。交叉熵的值越小，模型预测效果就越好。

交叉熵在分类问题中常常与 softmax 是标配，softmax 将输出的结果进行处理，使其多个分类的预测值和为 1，再通过交叉熵来计算损失。

（5）训练代码

```
def train(net, train_iter, test_iter, num_epochs, device, loss, optimizer, scheduler=None):
    def init_weights(m):
        if isinstance(m, nn.Conv2d):
            nn.init.kaiming_normal_(m.weight, mode="fan_out")
            if m.bias is not None:
                nn.init.zeros_(m.bias)
        elif isinstance(m, (nn.BatchNorm2d, nn.GroupNorm)):
            nn.init.ones_(m.weight)
            nn.init.zeros_(m.bias)
        elif isinstance(m, nn.Linear):
            nn.init.normal_(m.weight, 0, 0.01)
            nn.init.zeros_(m.bias)
    net.apply(init_weights)
    print('training on', device)
    net.to(device)
    animator = d2l.Animator(xlabel='epoch', xlim=[1, num_epochs],
                            legend=['train loss', 'train acc', 'test acc'])
    timer, num_batches = d2l.Timer(), len(train_iter)
    for epoch in range(num_epochs):
        # 训练损失之和，训练准确率之和，样本数
```

```
            print("epoch = {}, scheduler = {}".format(epoch, scheduler.get_last_lr()[0]))
            metric = d2l.Accumulator(3)
            net.train()
            for i, (X, y) in enumerate(train_iter):
                timer.start()
                optimizer.zero_grad()
                X, y = X.to(device), y.to(device)
                y_hat = net(X)
                l = loss(y_hat, y)
                l.backward()
                optimizer.step()
                with torch.no_grad():
                    metric.add(l * X.shape[0], d2l.accuracy(y_hat, y), X.shape[0])
                timer.stop()
                train_l = metric[0] / metric[2]
                train_acc = metric[1] / metric[2]
                if (i + 1) % (num_batches // 5) == 0 or i == num_batches - 1:
                    animator.add(epoch + (i + 1) / num_batches,
                                 (train_l, train_acc, None))
            test_acc = evaluate_accuracy_gpu(net, test_iter)
            animator.add(epoch + 1, (None, None, test_acc))
            if scheduler:
                if scheduler.__module__ == lr_scheduler.__name__:
                    # UsingPyTorchIn-Builtscheduler
                    scheduler.step()
                else:
                    # Usingcustomdefinedscheduler
                    for param_group in optimizer.param_groups:
                        param_group['lr'] = scheduler(epoch)
        plt.show()
        print(f'loss {train_l:.3f}, train acc {train_acc:.3f}, '
              f'test acc {test_acc:.3f}')
        print(f'{metric[2] * num_epochs / timer.sum():.1f} examples/sec '
              f'on {str(device)}')
if __name__ == '__main__':
    lr, num_epochs, batch_size = 0.1, 20, 128
    optimizer = torch.optim.SGD(net.parameters(), momentum=0.9, lr=lr,
weight_decay=1e-3)
    loss = nn.CrossEntropyLoss()
    scheduler = ExponentialLR(optimizer, gamma=0.9)
    train_iter, test_iter = d2l.load_data_ciffar_10(batch_size, resize=112)
    train(net, train_iter, test_iter, num_epochs, d2l.try_gpu(), loss, optimizer, scheduler)
```

4.6 结果分析与讨论

（1）在经过 20 次迭代后，训练集的准确率为 95%，测试集的准确率为 85%，虽然存在过拟合，但是整体效果还算良好。

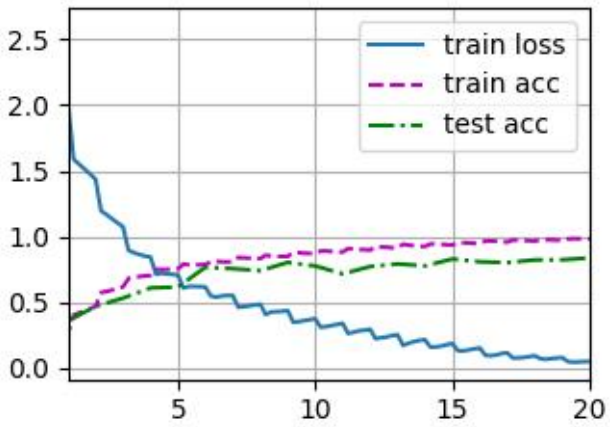
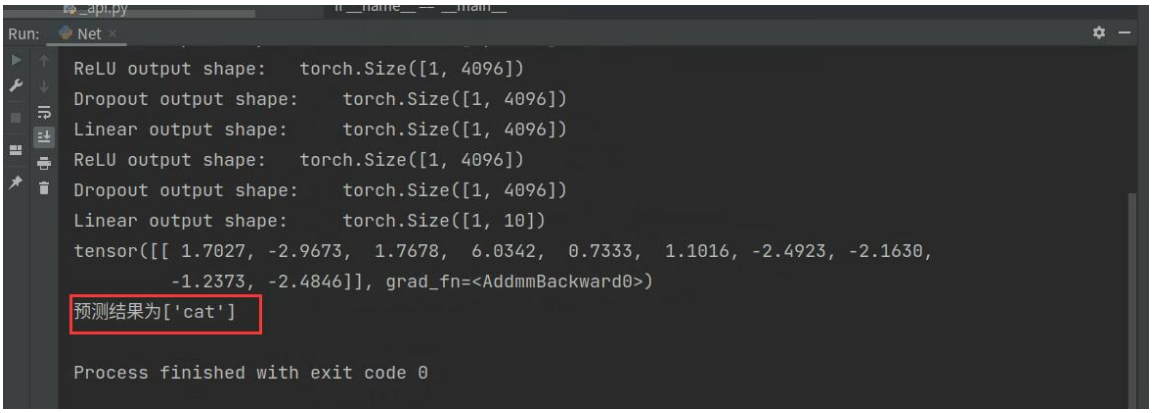


图 4. 4 准确率变换曲线图示

（2）模型测试

① 输入猫的图片：

预测结果为：



② 输入鸟的图片：

预测结果为：

```
ReLU output shape:   torch.Size([1, 4096])
Dropout output shape:    torch.Size([1, 4096])
Linear output shape:     torch.Size([1, 4096])
ReLU output shape:   torch.Size([1, 4096])
Dropout output shape:    torch.Size([1, 4096])
Linear output shape:     torch.Size([1, 10])
tensor([[ 3.3921, -3.6686,  5.1286, -1.1113,  0.8845,  1.6921,  0.2763, -2.4323,
         -1.5978, -2.5632]], grad_fn=<AddmmBackward0>)
预测结果为['bird']

Process finished with exit code 0
```

4.7 总结

本次通过实际构架 VGG-11 卷积神经网络模型，使用学习率指数衰减、梯度下降和交叉熵函数等进行神经网络训练，我对卷积网络进行图像识别的原理和有了进一步的了解，深入地学习了输入层、输入层、卷积层、激活层、池化层、全连接层的作用，对于神经网络模型的训练过程中的 BP 学习算法理解的更加清楚，对于深度学习的相关概念也更加理解。在本次的实际操作中，我将专业知识用于实践，无论是网络的架构，或者是训练算法的设计，还是相关参数的调试，都提升了我的实践能力。

当然，本次模型训练也存在不足，模型存在过拟合，虽然使用了如丢弃法等方法进行优化，但还是没能很好的解决这一问题，经过上网查资料发现该问题可以通过正则化方法、数据增强以及提前终止法优化，这些将会成为我未来重点学习的方向。

参考文献

[1]王彬,张正平,贾明俊等.基于卷积神经网络 VGG 的猫狗识别[J].智能计算机与应用,2021,11(07):162-165.

[2]冯国徽. 基于卷积神经网络 VGG 模型的小规模图像分类[D].兰州大学,2018.

[3]朱骏宇.基于卷积神经网络的图像识别的技术分析[J].长江信息通信,2023,36(08):66-68.

[4]李文静,白静,彭斌等.图卷积神经网络及其在图像识别领域的应用综述[J/OL].计算机工程与应用:1-25[2023-11-05].http://kns.cnki.net/kcms/detail/11.2127.tp.20230512.1640.016.html.

[5]毛少华,王文东.卷积神经网络的深度与宽度对猫狗图像识别模型性能的影响[J].河南科学,2023,41(07):956-963.

附：问题回答

一、简述模式识别系统的主要组成部分和设计步骤。

模式识别系统是一种人工智能系统，用于自动识别、分类和分析数据中的模式。模式识别系统由数据获取、预处理、特征提取、分类决策和分类器设计 5 部分组成。模式识别系统各组成单元的功能如下：

1) **数据获取：**利用计算机可以运算的符号来表示所研究的对象，对应于外界物理空间向模式空间的转换。一般，获取的信息类型有以下几种：
一维波形：心电图、脑电波、声波、震动波形等。
二维图像：文字、地图、照片等。
物理参量：体温、化验数据、温度、压力、电流、电压等。

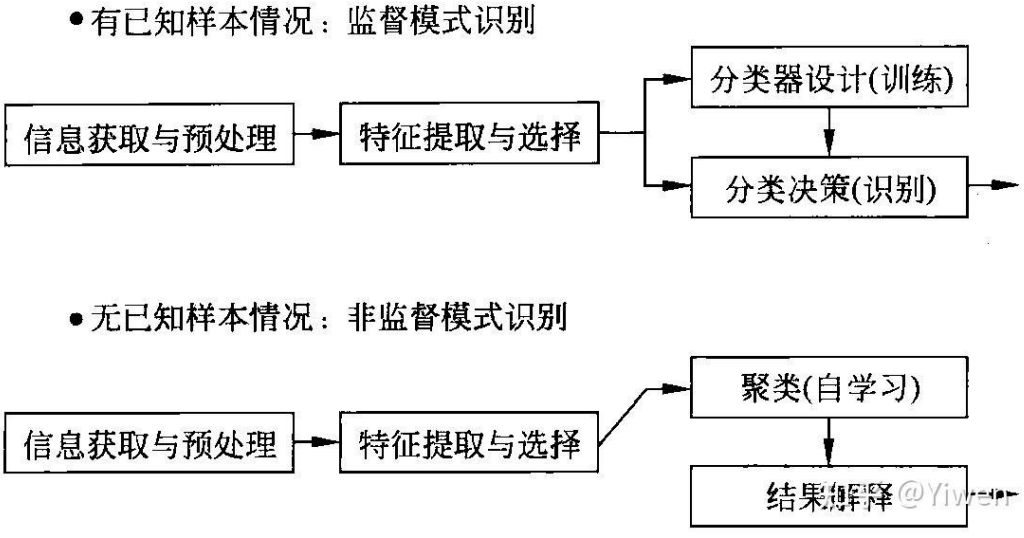
2) **预处理：**对由于信息获取装置或其他因素所造成的信息退化现象进行复原、去噪，加强有用信息。

3) **特征提取：**由信息获取部分获得的原始信息，其数据量一般相当大。为了有效地实现分类识别，应对经过预处理的信息进行选择或变换，得到最能反映分类本质的特征，构成特征向量。其目的是将维数较高的模式空间转换为维数较低的特征空间。

4) **分类决策：**在特征空间中用模式识别方法（由分类器设计确定的分类判别规则）对待识模式进行分类判别，将其归为某一类别，输出分类结果。这一过程对应于特征空间向类别空间的转换。

5) **分类器设计：**为了把待识模式分配到各自的模式类中，必须设计出一套分类判别规则。基本做法是收集一定数量的样本作为训练集，在此基础上确定判别函数，改进判别函数和误差检验。

下图给出了监督模式识别系统和非监督模式识别系统的典型构成框图：



处理监督模式识别问题的设计步骤：

1) **分析问题：**深入研究应用领域的问题，分析是否属于模式识别问题，把所研究的目标表示为一定的类别，分析给定数据或者可以观测的数据中哪些因素可能与分类有关。
2) **原始特征获取：**设计实验，得到已知样本，对样本实施观测和预处理，获取可能与样本分类有关的观测向量（原始特征）。
3) **特征提取与选择：**为了更好地进行分类，可能需要采用一定的算法对特征进行再次提取和选择。
4) **分类器设计：**选择一定的分类器方法，用已知样本进行分类器训练。
5) **分类决策：**利用一定的算法对分类器性能进行评价；对未知样本实施同样的观测预处理和特征提取与选择，用所设计的分类器进行分类，必要时根据领域知识进行进一步的后处理。

处理非监督模式识别问题的设计步骤：

1) **分析问题：**深入研究应用领域的问题，分析研究目标能否通过寻找适当的聚类来达到；如果可能，猜测可能的或希望的类别数目；分析给定数据或者可以观测的数据中哪些因素可能与聚类有关。
2) **原始特征获取：**设计实验，得到待分析的样本，对样本实施观测和预处理，获取可能与样本聚类有关的观测向量（原始特征）。
3) **特征提取与选择：**为了更好地进行聚类，可能需要采用一定的算法对特征进行再次提取和选择。
4) **聚类分析：**选择一定的非监督模式识别方法，用样本进行聚类分析。
5) **结果解释：**考查聚类结果的性能，分析所得聚类与研究目标之间的关系，根据领域知识分析结果的合理性，对聚类的含义给出解释；如果有新样本，把聚类结果用于新样本分类。

二、查资料，找文献，结合自己研究内容/兴趣，写出对“模式识别与机器学习”的应用的认识。

"模式识别与机器学习"的应用是多种多样的，它们基于算法和数据分析技术，能够自动识别、分类和预测数据中的模式，从而产生有价值的信息和决策。以下是模式识别与机器学习的一些应用：

1) **计算机视觉：**在图像和视频处理中，模式识别和机器学习被用于对象检测、人脸识别、图像分类、图像分割、动作识别等任务。例如，人脸解锁、自动驾驶汽车的障碍物检测和跟踪、医学图像分析等领域都使用了这些技术。
2) **自然语言处理：**在文本和语音处理中，机器学习用于情感分析、文本分类、自动翻译、语音识别和生成。智能助手（如 Siri、Alexa、Google Assistant）和自然语言处理应用程序都利用了这些技术。
3) **医学诊断：**模式识别和机器学习可用于医学图像分析（如 X 射线、MRI、CT 扫描）、癌症诊断、疾病预测和基因序列分析。它们有助于提高医生的决策支持和病情诊断。
4) **金融领域：**在金融领域，模式识别和机器学习用于欺诈检测、股票市场预测、

信用评分、风险管理和高频交易。这些技术可以帮助金融机构更好地管理风险和优化决策。

5) **制造业：**机器学习和模式识别可用于质量控制、产品缺陷检测、设备维护预测和生产过程优化。这有助于提高制造效率和降低成本。
6) **市场营销：**在市场营销中，模式识别和机器学习被用于个性化推荐、广告定位、客户细分和消费者行为分析。社交媒体和电子商务平台广泛使用这些技术，用于分析社交媒体数据，以了解用户情感、趋势和舆论，有助于决策制定和市场营销。
7) **环境监测：**用于分析气象数据、空气质量监测、地震预测等，有助于提前警告和应对自然灾害。
8) **安全领域：**用于入侵检测、威胁分析、反欺诈系统和生物识别。
9) **交通和物流：**交通规划、路况预测、包裹分拣和无人机配送等领域。
10) **教育：**用于个性化教育、自动化评估、学习分析和课程推荐。