手写数字识别

泰迪智能科技（武汉）有限公司

目 录

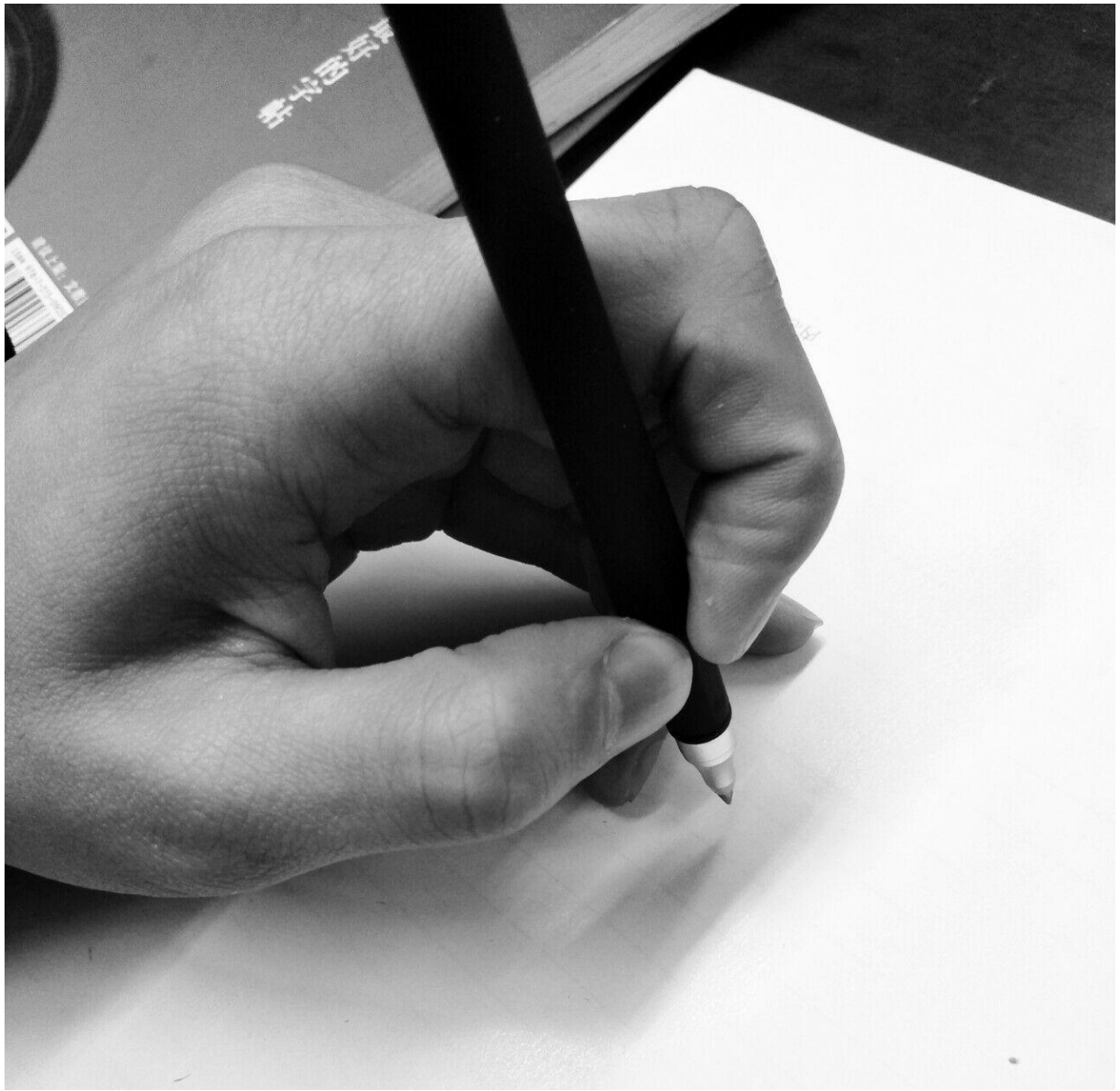
contents

第一部分

研究背景与目标

- 研究背景
- 研究目标
- 数据说明

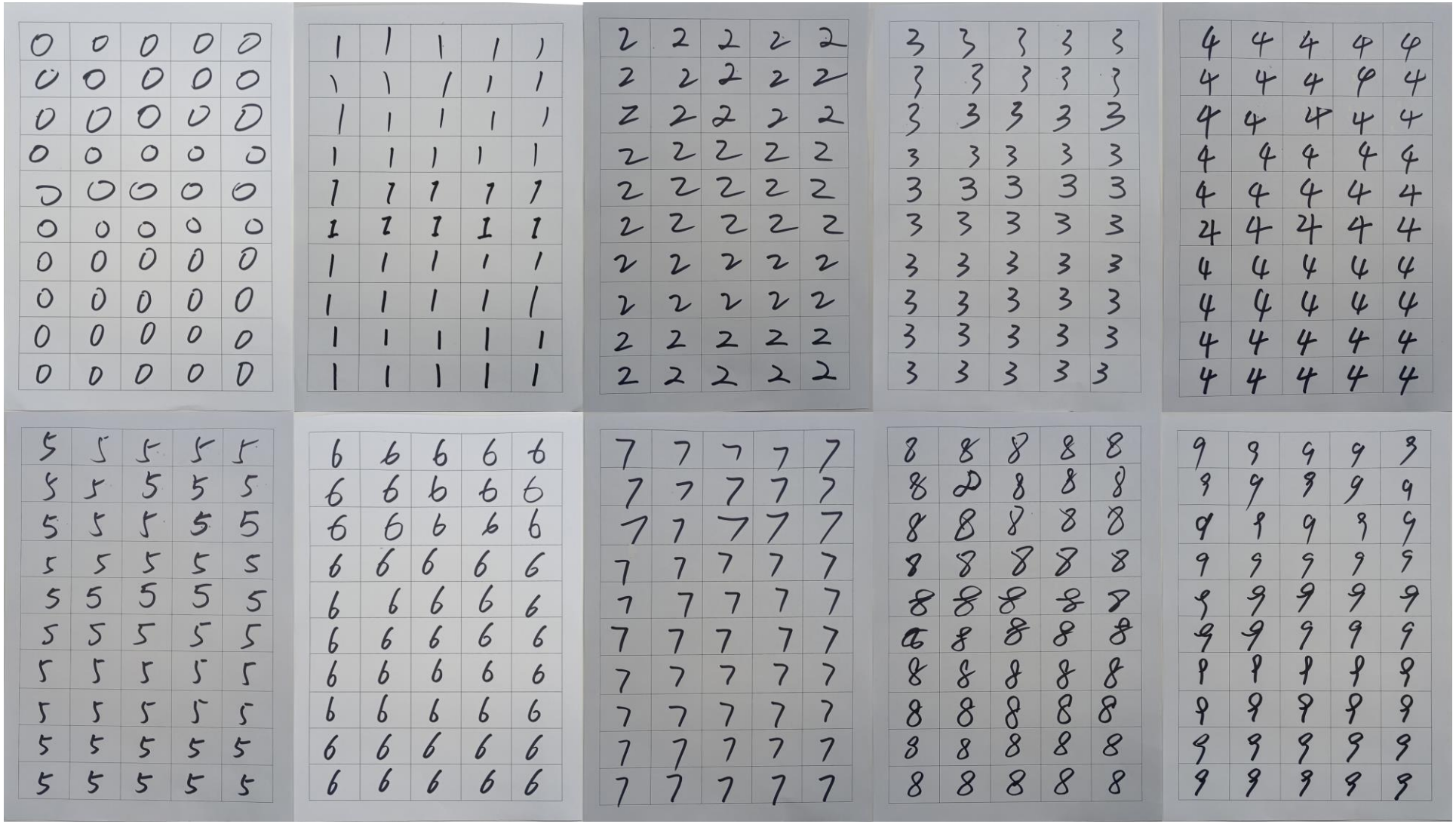
研究背景



手写数字识别最早源于20世纪90年代，当时AT&T实验室的专家开发了第一个手写数字识别系统。这个系统基于统计学习的方法，并使用多层感知器作为其神经网络模型。经过时代不断发展，现在手写数字识别问题已经成了一个经典的机器学习问题，经常被用作入门机器学习任务的示例。

数据说明

本案例数据来源自志愿者手写的数字，每张图片包含一种数字，共十张



研究目标

研究目标主要是构建多个能够自动识别手写数字的模型，使得对于一张新的手写数字图片，系统能够自动识别出对应的数字。此外还要对模型进行对比分析，掌握机器学习、BP神经网络、卷积神经网络的差异。

第二部分

分析方法与过程

- 使用方法
- 项目流程

使用方法

1.图像裁剪：这是一种常用的图像处理技术，主要用于选择图像的特定区域，去除不必要的背景或者突出主题。例如，对于一张包含人脸的图像，通过裁剪可以只保留人脸部分，以便于后续的人脸识别等任务。

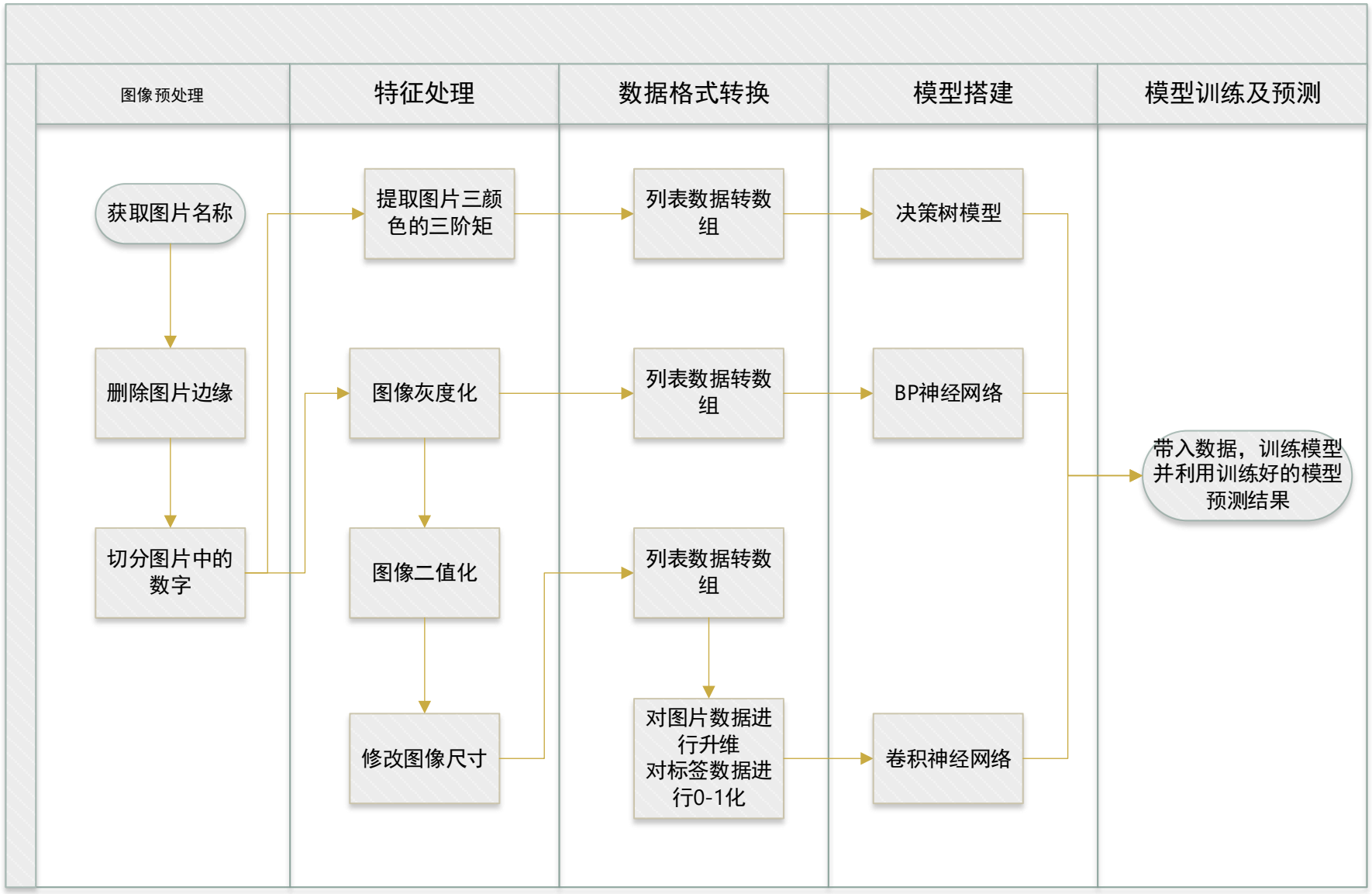
2.图像色彩转换：这种技术主要用于改变图像的颜色空间或颜色属性。例如，将彩色图像转换为黑白图像，或将RGB颜色空间的图像转换为灰度图像。此外，还可以对图像进行色彩平衡、对比度调整等操作，以增强图像的视觉效果。

3.修改图像尺寸：这是一种常用的图像处理技术，主要用于调整图像的大小。例如，在处理不同来源的图像时，可能需要将其大小统一以便于后续的计算机视觉任务。此外，在某些情况下，通过缩小图像可以减少计算量和加速处理速度。

4.决策树：决策树是一种常用的机器学习算法，它通过树形结构来展示决策的逻辑过程。决策树可以用于分类和回归任务，并具有直观易懂的特点。在训练过程中，决策树不断划分数据集，直到达到终止条件（如纯度足够、节点数达到最大等）。

5.神经网络&卷积神经网络：神经网络是一种模拟人脑神经元连接方式的计算模型，由多个神经元组成，各神经元之间通过加权连接进行信息传递。卷积神经网络是一类包含卷积计算且具有深度结构的前馈神经网络，是深度学习的一种代表算法。

分析过程


图像预处理
特征处理
数据格式转换
模型搭建
模型训练及预测
获取图片名称
删除图片边缘
切分图片中的数字
提取图片三颜色的三阶矩
图像灰度化
图像二值化
修改图像尺寸
列表数据转数组
列表数据转数组
列表数据转数组
对图片数据进行升维
对标签数据进行0-1化
决策树模型
BP神经网络
卷积神经网络
带入数据，训练模型并利用训练好的模型预测结果

第三部分

数据预处理

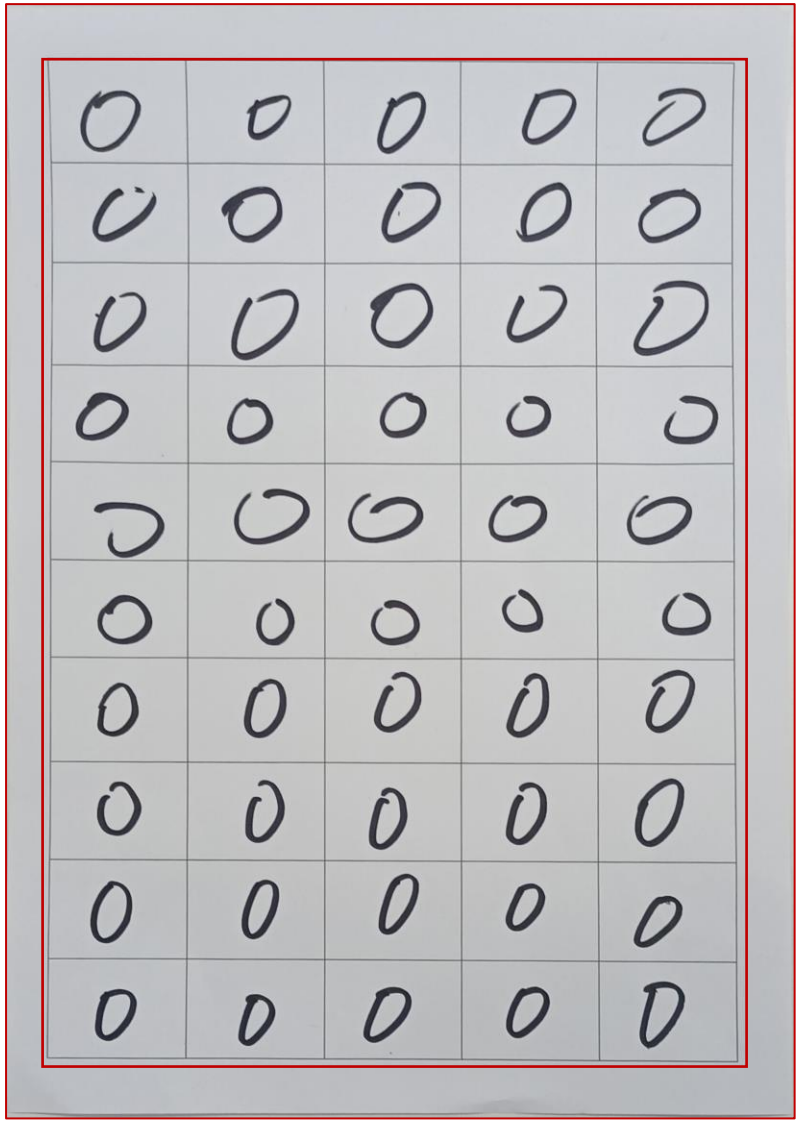
- 图像裁剪边缘
- 图像分割

图像去除边缘

对于原始的图像，存在一个规律的无信息的边缘。为了提升后面预测的精准度，可以将不包含笔记信息的白边进行删除。删除函数如下：

```
def pic_cut(pic_name):
    pic=plt.imread(f'odata/{pic_name}')
    N,M,D=pic.shape
    Npic_ =pic[135: 2835,150:1950]
    return Npic_
```

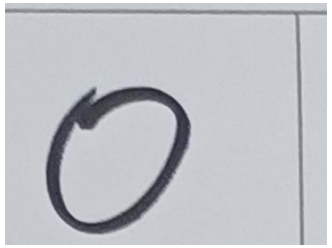
此处主要利用的时图片在计算机内存中是以数组的形式存在的，直接利用数组操作即可完成白边删除工作。



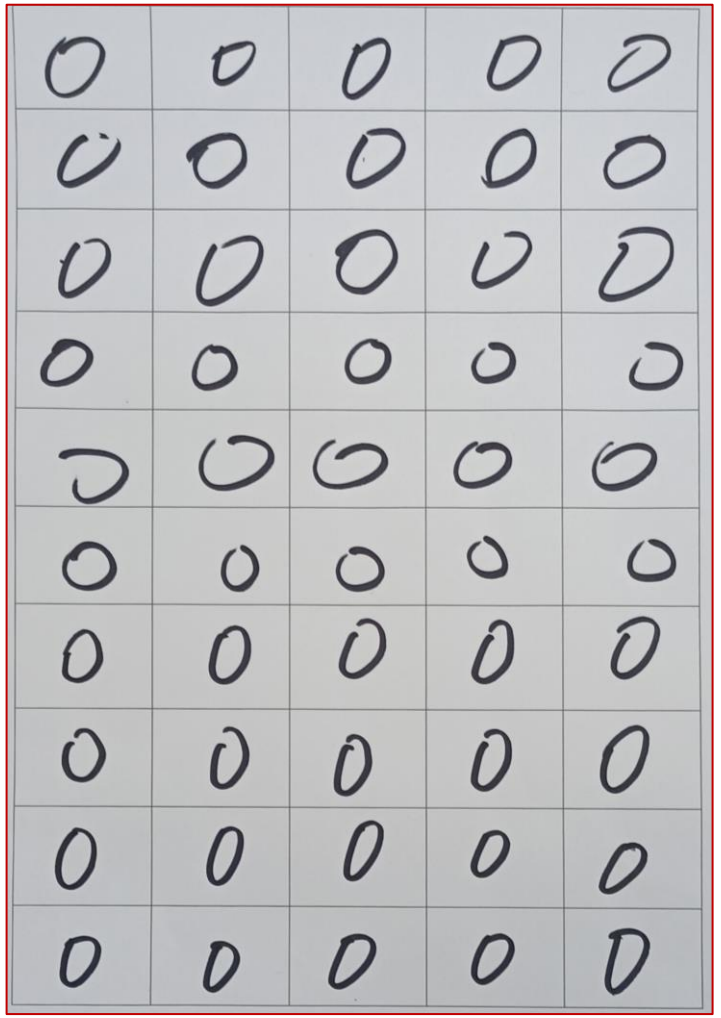
图像分割

裁剪后的图片还需要进行再次裁剪，获取每一个数字对应的小区域。实现原理与前面一样，也是利用数组的特性进行操作。详细函数如下

```
def pic_split(Npic):
    pic_data_=[]
    M, N,D= Npic.shape
    for ii in range(5):
        for jj in range(10):
            pic_new=Npic[270*jj:(270*(jj+1)),360*ii:(360*(ii+1)),:]
            pic_data_.append(pic_new)
    return pic_data_
```



←裁剪后图片示意



第四部分

手写数字建模识别

- 机器学习手写数字识别
- BP神经网络手写数字识别
- 卷积神经网络手写数字识别

特征处理

由于机器学习无法大量接受参数，此时可以利用每个数字所在纸张的颜色有些许差异的特点，利用色彩信息完成手写数字识别。色彩信息获取的时候需要用到三阶颜色矩，三阶颜色矩的含义分别如下：

一阶颜色矩：采用一阶原点矩，反映了图像的整体明暗程度。

$$E = \frac{1}{N}\sum pix$$

二阶颜色矩：采用二阶中心矩的平方根，反映了图像颜色的分布范围。

$$s = \sqrt{\frac{1}{N}\sum (pix - E)^2}$$

三阶颜色矩：采用三阶中心矩的立方根，反映了图像颜色分布的对称性

$$s = \sqrt[3]{\frac{1}{N}\sum (pix - E)^3}$$

机器学习模型：决策树

决策树模型简介

决策树(decision tree)一般都是自上而下的来生成的。每个决策或事件（即自然状态）都可能引出两个或多个事件，导致不同的结果，把这种决策分支画成图形很像一棵树的枝干，故称决策树。

我们以挑选西瓜作为例子，在真实生活中几乎不会在挑瓜的时候进行概率计算。我们通常的选择过程如下：

先看一看，观察一下西瓜的花纹是什么样的

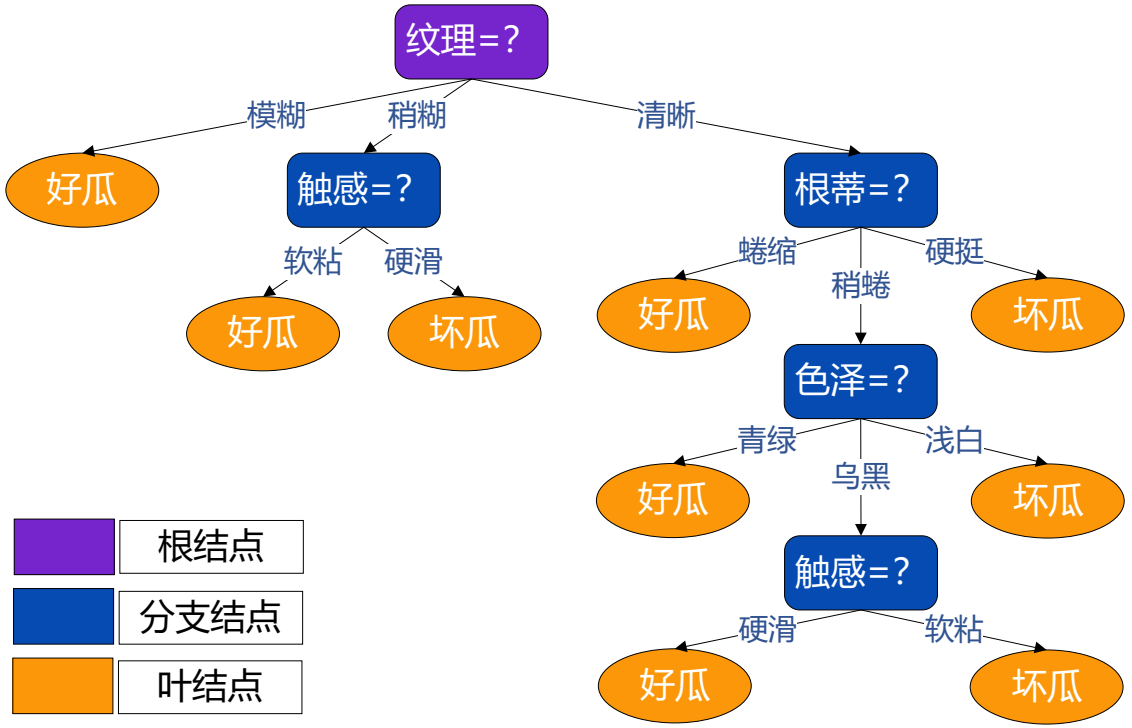
选择一个纹路清晰的西瓜

看看西瓜的根蒂，是不是硬挺的

如果不是我们再掂一掂西瓜，看看触感

最后象征性地敲一敲，听听声音

上述过程抽象化就是决策树算法



模型构建与评价

在使用机器学习算法进行预测时需要遵守如下步骤:

1. 本案例划分训练集测试集比例为8：2
2. 本案例使用模型为决策树分类模型
3. 模型最终预测精准度为0.69

01
数据集划分
train_test_split

02
模型导入
DecisionTreeClassifier

03
模型训练
Model.fit(train_x,train_y)

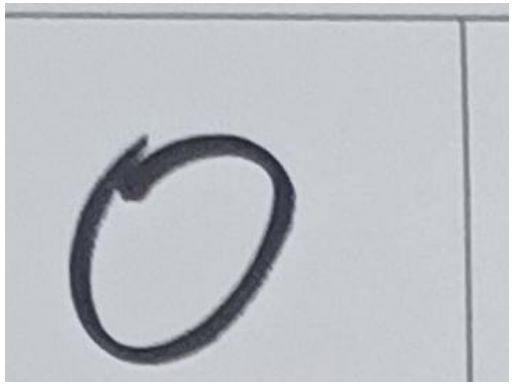
04
模型预测
Model.predict(test_y)

05
模型评价
mean_absolute_error
(test_y,pre_y)

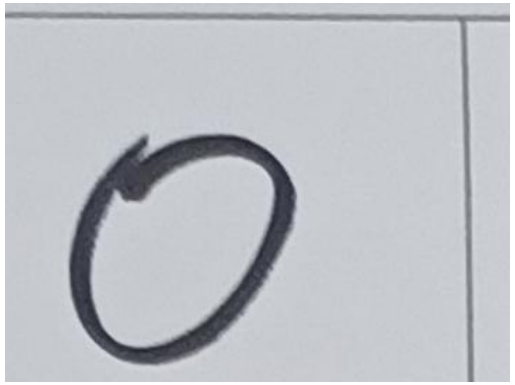
BP神经网络手写数字识别

数据处理

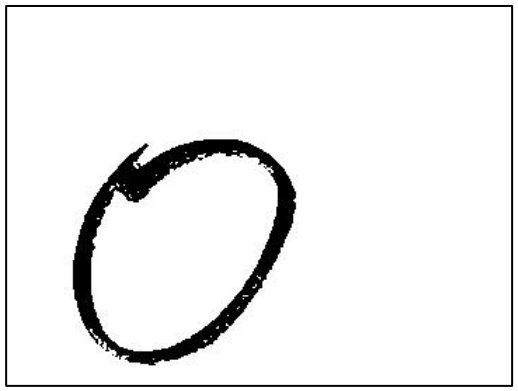
理论上，BP神经网络可以接受无限多的参数，但是为了算法的精准度，在开始预测先需要对数据再次处理。原始图像是彩色图像，而数字表示并不需要色彩，此处可以将图像进行灰度化处理。处理后的图像背景比较明显，为了突出文字本身，这里可以对图像再次进行二值化处理。数字信息本身细节较少，可以尝试将图像缩小尺寸，实际结果标明，即便将数字图片转换成64*64的大小依然不影响图像显示。综上所述，可以对每一个小图片都如此处理。最后将所有的图像数据都存放在一个500*64*64的数组中，标签数据存放在一个500维的数组中。



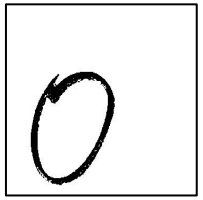
裁剪后的数字



图像灰度化结果



图像二值化结果



图像缩小结果

BP神经网络手写数字识别

模型搭建训练

在PyTorch中，BP（反向传播）神经网络的模型搭建可以通过以下步骤进行：

1. 定义模型：使用torch.nn定义模型的结构。可以选择预定义的层或者自定义层。
2. 编译模型：设置模型的优化器和损失函数，以及评估指标。
3. 训练模型：使用训练数据对模型进行训练。
4. 评估模型：使用测试数据对训练好的模型进行评估。
5. 保存和加载模型：训练完成后，可以保存模型以便以后使用。

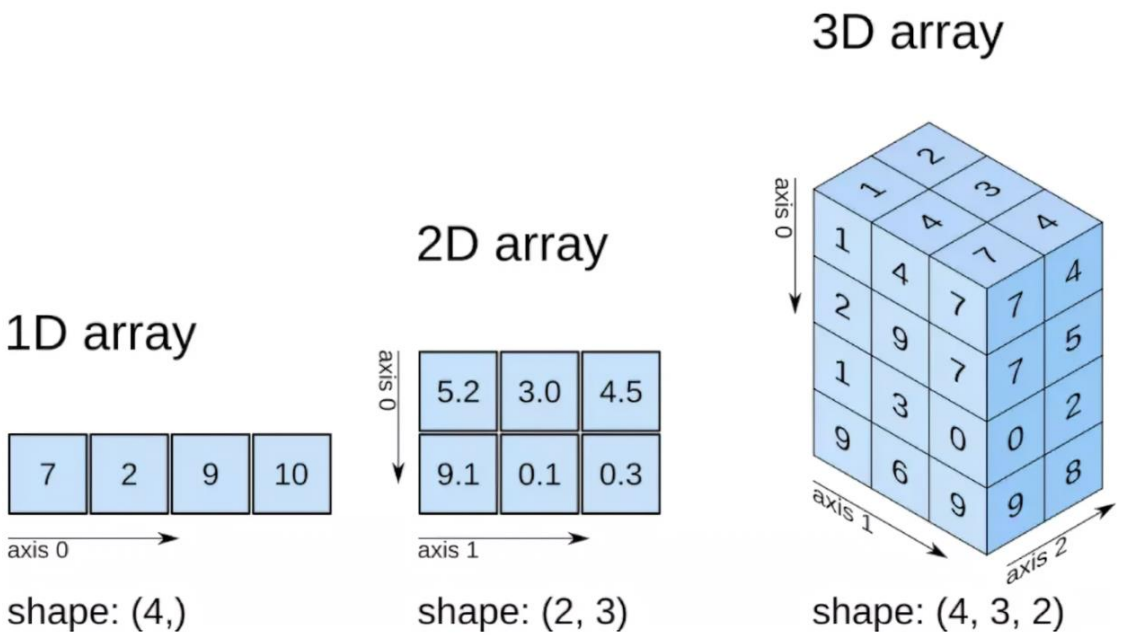
以上是一个简单的PyTorch中BP神经网络的模型搭建过程。根据实际任务和数据集的不同，可能需要进行更复杂的模型设计和调整。本案例使用了一个两层的神经网络，最终预测结果准确度为0.39

```
Sequential(
  (0): Linear(in_features=4096, out_features=500, bias=True)
  (1): ReLU()
  (2): Linear(in_features=500, out_features=128, bias=True)
  (3): ReLU()
  (4): Linear(in_features=128, out_features=64, bias=True)
  (5): ReLU()
  (6): Linear(in_features=64, out_features=10, bias=True)
)
```

卷积神经网络手写数字识别

数据升维& 标签处理

- 卷积神经网络在进行数据带入时要求数据为有深度的图像数据，所以经过BP神经网络处理后的数据还需要进行标准化和升维。标准化用原始数据除以255即可将像素值转到0-1之间。升维则相当于給图片增加一个数值为1的深度。此外，标签数据也需要转变，需要从原来的0，1，2，3……修改成独热编码的标签信息。



不同维度的数据

Human-Readable

Pet
Cat
Dog
Turtle
Fish
Cat

Machine-Readable

Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

独热编码示例

卷积神经网络手写数字识别

模型搭建训练

- 在PyTorch中，卷积神经网络的搭建与BP神经网络并没有太大的区别，这里只需要增加卷积层和池化层即可。本案例使用网络如右图所示，手写数字识别精准度能到0.97。

```
Sequential(
  (0): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU()
  (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (4): ReLU()
  (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (6): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (7): ReLU()
  (8): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (9): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
  (10): ReLU()
  (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (12): Flatten(start_dim=1, end_dim=-1)
  (13): Linear(in_features=256, out_features=64, bias=True)
  (14): ReLU()
  (15): Linear(in_features=64, out_features=10, bias=True)
)
```

第五部分

结果分析及应用

- 结果分析
- 应用

三个模型结果比对

	precision	recall	f1-score	support
0	0.57	0.80	0.67	10
1	1.00	0.83	0.91	12
2	0.89	0.62	0.73	13
3	0.46	0.86	0.60	7
4	0.82	0.69	0.75	13
5	0.60	0.33	0.43	9
6	0.50	0.29	0.36	7
7	0.40	0.80	0.53	5
8	0.60	0.50	0.55	
9	0.79	0.92	0.85	
accuracy			0.67	
macro avg	0.66	0.66	0.64	
weighted avg	0.71	0.67	0.67	

决策树模型结果

	precision	recall	f1-score	supp
0	0.50	0.15	0.24	
1	0.29	0.22	0.25	
2	0.55	0.67	0.60	
3	0.30	0.46	0.36	
4	1.00	0.38	0.56	
5	0.30	0.64	0.41	11
6	0.40	0.20	0.27	10
7	0.30	0.75	0.43	4
8	0.50	0.25	0.33	12
9	0.33	0.50	0.40	6
accuracy			0.39	100
macro avg	0.45	0.42	0.38	100
weighted avg	0.47	0.39	0.38	100

BP神经网络结果

	precision	recall	f1-score	support
0	1.00	0.85	0.92	13
1	1.00	1.00	1.00	9
2	0.90	1.00	0.95	9
3	1.00	1.00	1.00	13
4	0.92	0.85	0.88	13
5	1.00	1.00	1.00	11
6	0.90	0.90	0.90	10
7	0.80	1.00	0.89	4
8	1.00	0.92	0.96	12
9	0.75	1.00	0.86	6
accuracy			0.94	100
macro avg	0.93	0.95	0.93	100
weighted avg	0.95	0.94	0.94	100

卷积神经网络结果

结果分析说明

- 在机器学习和数据分析中，我们经常面临各种复杂的问题和挑战，因此很容易陷入一种误区，认为只有使用更复杂、更高级的方法才能得到更好的任务效果。然而，实际上，方法的复杂性并不总是与任务效果成正比。有时候，简单的方法甚至能够超越复杂方法，实现更优秀的结果。
- 一个关键原因是，复杂的方法通常需要更多的数据来训练模型，以防止过拟合。在没有足够数据的情况下，使用过于复杂的方法可能导致模型在训练集上表现良好，但在测试集上表现糟糕，即出现过拟合现象。此时，简单的方法可能具有更好的泛化能力。
- 此外，选择合适的数据预处理步骤对于任务效果至关重要。原始数据往往包含噪声、异常值或冗余信息，这些问题可能会干扰模型的训练过程。通过适当的数据预处理，如数据清洗、特征选择、特征变换等，我们可以提高数据的质量，使得模型更容易学习到数据的内在规律。
- 综上所述，我们在选择方法和进行数据处理时，应根据具体问题的特点、数据的性质和算法的要求来制定策略。不是方法越难、越复杂就一定能带来更好的任务效果；相反，合适的数据预处理和选择简单有效的方法往往是实现优秀任务效果的关键。

应用

➢ 手写数字识别当前已经有了多方面的应用，例如：

•**邮政编码识别**：在处理大量的邮政信件时，自动识别手写的邮政编码可以大大提高分拣速度。

•**支票识别**：银行和金融机构需要将手写的数字和文字信息转换为可编辑的格式，以实现自动化处理。

•**移动应用**：许多移动应用允许用户通过手写输入数字，如手机解锁、支付密码等，手写数字识别技术可以增强这些输入的准确性。

•**文档分析**：在处理大量的文档时，自动识别其中的数字可以帮助更高效地进行分类、摘要和检索。

手写数字识别技术的广泛应用不仅提高了工作效率，也为用户带来了更便捷的体验。随着技术的不断进步，相信未来手写数字识别将在更多领域发挥其独特的价值。

谢谢