生活·技术·探索

巧用Python创建个性化数据集

郑祥　浙江省温州市第四中学

摘要: 本文介绍了中小学学校环境下，人工智能模型训练所需的个性化图像数据集的设计、收集、制作等过程，其中的关键技术是运用Python语言编程，对大批量的图像进行分割、重命名、分类保存等批处理操作，快速高效地生成个性化的数据集。

关键词: 个性化数据集；Python；图像分割；批处理；人工智能

中图分类号: G434　**文献标识码:** A　**论文编号:** 1674—2117（2022）13—0070—03

人工智能的智能识别算法往往取决于人工智能模型的训练，而数据集则是人工智能模型训练的基础。纵观网络资源，虽有着海量的资源网站，却难以找到符合项目的数据集，更别提能满足个性化的人工智能模型训练的需求了。因此，本文将以电路、天气、表情等图标为例，对中小学校园中个性化数据集创建展开讨论。

● 问题提出

由于校园中人工智能项目，需要创建一系列的人工智能图像识别模型来实现简单的电路、天气、表情等图标的识别功能，而在强大的网络中并没有现成的电路、天气、表情等图标的数据集，因此如何动手创建数据集成了项目开展的关键。

● 思考、分析与设计

用于人工智能模型训练的数据集需要满足一定的数量，因为有学生的助力，中小学中数据集的制作非常容易实现，其关键的三个环节分别是：①获得纸质数据集原始图像，转为数字原始图像；②从数字原始图像中分割得到电路、天气等图标；③按图标分类保存图标。其中，环节②和环节③可通过编写程序批量处理实现。

1.数据集收集表的分析与设计

本文中涉及的数据集包含电路、天气、表情和植物4种图像数据，其中每种图像数据中又有4个小分类，具体如表1所示。

为了保证学生绘制图标和后期图像批量处理的质量，需要进行范例图标的设计和手绘区域的设计。在Word空白文档中，插入一张表格，参数如表2所示。在数据集收集表文档中增加提示性文字和范例图标（如下页图1）。

2.图像处理的算法分析

如下页图2所示，手绘图标分别位于区域1至区域16的位置，批量分

表1　数据集图像分类

图标种类	分类 1	分类 2	分类 3	分类 4
电路图标	断开	闭合	1 电阻	2 电阻
天气图标	晴天	雨天	阴天	多云
表情图标	笑脸	难过	无语	开心
植物图标	花	草	树	林

表2　数据集收集表文档设置

属性名称	具体参数
Word 空白文档纸张大小	A4 大小
Word 空白文档页边距	上边距：1.27 厘米；下边距：1.27 厘米 左边距：1.27 厘米；右边距：1.27 厘米
表格	8 行，5 列；每个单元格宽度、 高度大小一致

割区域1至16并分类存放，是图像处理算法的核心。鉴于Python语言语法简单、图像处理库丰富的特点，本案例采用Python中的pillow库编程实现图像批量分割、分类处理。

图像的分割裁剪，需知待分割区域的左上角和右下角点的坐标，如分割区域1需知点A1和A7的坐标。

以A0为原点(0,0)建立平面直角坐标系，如图3所示。

由于每个单元格的宽度、高度大小统一，因此A1至A40点的坐标数值间的关系如下：

$$x_1 = 0$$

$$y_1 = 0$$

$$x_{n+1} = x_n + w \div 5$$

$$y_{n+1} = y_n + h \div 8$$

(w为表格宽度；h为表格高度)

裁剪数字原始图像，仅留表格部分，w、h分别为新图像的宽度、高度。因此，图像分割分类前需进行预处理：裁剪图像，仅留表格区域。

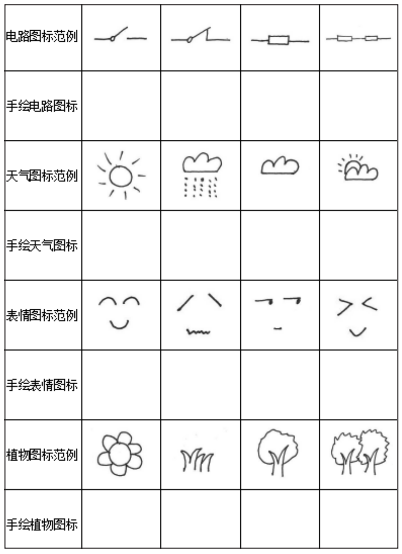

图1

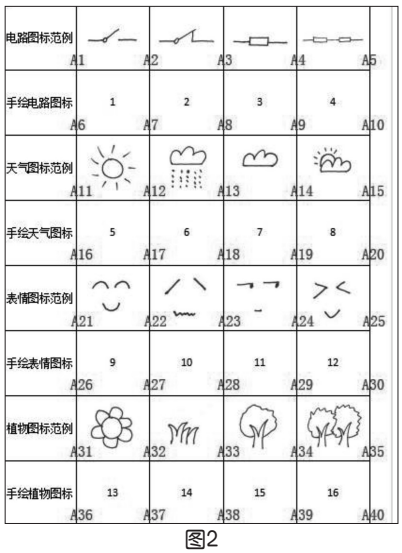

图2

核心代码

根据以上分析，图像处理一共分为以下几步：①图像预处理，得到表格宽度w、高度h数据；②计算区域坐标；③分割图像，分类存储。

1.图像预处理核心代码（如下页图4）

2.计算区域坐标核心代码

定义imageCutXY函数，获取待分割区域坐标，代码如下页图5所示。

3.图像分割、分类核心代码

在imageCut函数代码基础上，增加图像分类代码，图像分类的命名如表3所示。

因篇幅有限，仅展示1个大类的图片分类保存的核心代码，如下页图6所示。

图像分割批处理的主程序代码如下页图7所示。

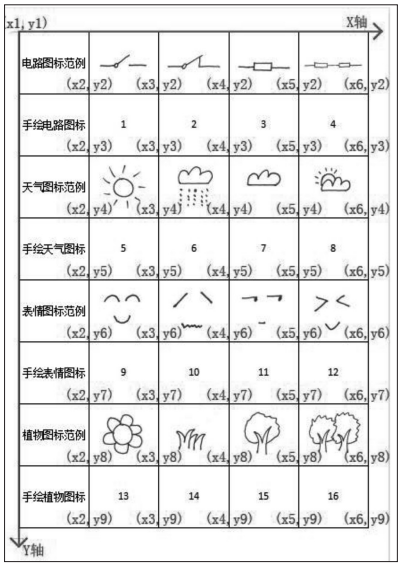

图3

数据集制作

有了前期的准备工作，后期只需要进行以下几个简单步骤就能得到想要的数据集了：

①打印数据集收集表，并请学生完成手动图标的绘制；②收集纸质数据集收集表，并扫描得到JPG格式的数字图像；③运行图像预处理程序，得到预处理的图像文件；④运行图像处理程序，得到电路、天气、表情等图标的数据集(如下页图8)。

表3　　图标分类命名

图标种类	分类 1	分类 2	分类 3	分类 4
电路图标	断开	闭合	1 电阻	2 电阻
电路图标命名	duankai	bihe	1dianzu	2dainzu
天气图标	晴天	雨天	阴天	多云
天气图标命名	qingtian	yutian	yintian	duoyun
表情图标	笑脸	难过	无语	开心
表情图标命名	xiaolian	nanguo	wuyu	kaixin
植物图标	花	草	树	林
植物图标命名	hua	cao	shu	lin

```
#自定义函数 imageSizeGet，返回图像宽度 w，高度 h，对象 pic
def imageSizeGet(filename):
    pic=Image.open(filename)
    w,h=pic.size #获取图像宽度 w，高度 h
    print("宽度：",w,"高度：",h) #打印图像宽度和高度
return w,h,pic  #返回图像宽度 w，高度 h，对象 pic

#自定义函数 imageCut，分割图像并保存图像文件
def imageCut(x1,y1,x2,y2,flag,picKey):
    bbox=(x1,y1,x2,y2) #图像分割区域坐标(x1,y1),(x2,y2)
    pic=picKey
    im=pic.crop(bbox)
    im.save(str(flag) +'.jpg') #保存图像
```

图4

```
def imageCutXY(weight,height):
    flag=0#定义计数变量 flag，用于待分割区域编号
    stepj=int(height/8)#设置 y 轴循环遍历步长
    stepi=int(weight/5)#设置 x 轴循环遍历步长
    xy_dict={}#定义空字典，用于存放区域 1 至区域 16 的分割坐标
    for j in range(stepj,height,2*stepj):
        flag2=0#定义计数变量 flag2，用于判断是否为最后一个待分割区域
        for i in range(stepi,weight,stepi):
            if flag2>3:#若条件成立，表示最后一个待分割区域坐标已完成获取
                break #跳出当前循环
            flag=flag+1
            flag2=flag2+1
            #字典存放区域坐标，常数为后期分割时的坐标微调
            xy_dict[flag]=[i+10, j+10, i+stepi-24, j+stepj-24]
    return xy_dict #返回区域 1 至 16 的坐标
```

图5

```
#imageCut 函数基础上，增加分类代码 ，每大类中又分 4 类保存
    if flag<=4:
        n="dianlu"
        if flag==1:
            flagnum[flag]=flagnum[flag]+1
            n=n+"//duankai//duankai"#分类路径
            n=n+str(flagnum[flag])
        elif flag==2:
            flagnum[flag]=flagnum[flag]+1
            n=n+"//bihe//bihe"#分类路径
            n=n+str(flagnum[flag])
        elif flag==3:
            flagnum[flag]=flagnum[flag]+1
            n=n+"//1dianzu//1dianzu"#分类路径
            n=n+str(flagnum[flag])
        elif flag==4:
            flagnum[flag]=flagnum[flag]+1
            n=n+"//2dianzu//2dianzu"#分类路径
            n=n+str(flagnum[flag])
#因篇幅有限，后面 3 大类图片分类保存的代码与以上代码类似，此处省略
    im.save(n +'.jpg')#修改裁剪图片名称，按同类型图的数量命名
```

图6

```
for i in range(1,97):
    #打开图片文件,获取图片分辨率数据，裁剪图片
    w,h,pic=imageSizeGet(str(i)+".jpg")
    xy_dict=imageCutXY(w,h)#获取当前原始图像的待分割区域坐标
    #遍历字典，获取 16 个待分割区域坐标，进行图像分割、分类保存处理
    for key in xy_dict:
        value=xy_dict[key]
        imageCut(value[0],value[1],value[2],value[3],key,pic)
```

图7

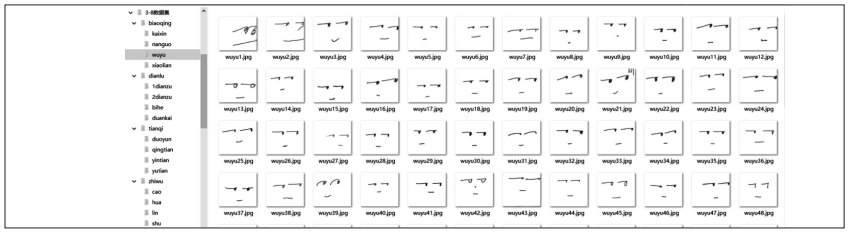

图8

结语

在数据集制作的过程中，Python简洁的语言和强大的图像处理库，大大减少了大量图像分割、分类、命名等烦琐重复的工作，让枯燥的制作过程变得简单而快速。除了pillow库，Python还有很多其他功能强大的图像处理库，如OpenCV等。

在制作过程中也遇到了一些问题，在此做两点交流分享。

问题1：数据集收集表的印刷。虽然数量需求较多，但仍然建议采用“打印”而非“速印”的方式印刷。究其原因是，“速印”会造成印刷内容位置偏移、大小变化，影响图像后期的批量处理。

问题2：纸质数据集收集表的扫描。在采用扫描仪批量扫描时，确保：①纸质收集表的整齐；②扫描仪托盘夹的位置调整，保证稿件扫描过程中位置偏移最小化。

参考文献：

[1]王媛媛，卜凡亮.基于Python的数字图像处理简单应用[J].电子技术与软件工程，2022(03)：129—132.

[2]雷霖，范烊，赵永鑫.一种基于图的图像分割法及Python实现[J].内江科技，2020(07)：34.

[3]李志，陈入云.Python在数字图像处理课程教学中的应用探索[J].创新创业理论研究与实践，2022(10)：11—13.