

-----马常啸年前工作汇报-----

图片切割部分

图片切割文件夹里有个cut.py是切割一张图片的，给一个图片名，就把这张图片切割成 $rx * ry$ 尺寸的图片。

cut_all.py文件是把一个文件夹里的所有文件全切成尺寸大小为 x_size, y_size 的图片并保存在一个文件夹里。

由于这里的切割是用来跑神经网络的，所以我的切割没有保存原来图片的名字，直接0.jpg, 1.jpg, 2.jpg, ...保存的。也可以用本来的名字保存如果需要我再修改：)

具体代码在图片切割文件夹中。注释也主要写在文件里啦~

对图片数据的二分类

模仿mnist写了一个二分类2part.py。调用了keras的Sequential进行模型的训练。读入的数据是之前用cut_all.py切好的图片直接复制过来的。

数据预处理

预处理部分为16到74行，主要是读入所有的训练集图片到train_image数组并把对应的标签保存到train_label数组。读入所有测试集图片到test_image数组并把对应的标签保存到test_label数组。

于是预处理部分分为了image部分的预处理和label部分的预处理两部分。

image部分预处理

根据keras模型对输入数据的要求，将存储数据本来的List类型用np.asarray函数转化为numpy的数组，并使每一项的类型为float.为了使训练效果好一些进行随机化处理

转化后进行归一化处理（每一个数除以255）最终得到两个shape为（2660， 64， 64， 3）的np.array数组train_image和test_image.

label部分的预处理

label部分取独热码处理。0 转化为（1， 0），1转化为（0， 1）。

我原来写的是十分类的代码，应该转化为（1， 0， 0， 0， 0， 0， 0， 0， 0， 0），
（0， 1， 0， 0， 0， 0， 0， 0， 0， 0），
（0， 0， 1， 0， 0， 0， 0， 0， 0， 0），
0），

...

,

(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1) .

其实二分类应该有自己的模型，我这里就直接沿用了多分类的思路转换独热码了。。。

最终测试结果也为一个数组 (x, y) .若x极其接近1， y极其接近0， 则预测结果为0.

如果x极其接近0， y极其接近1， 则预测结果为1.这里因为只有两个所以我直接把两个数比大小了...如果x > y则结果为0， 如果x < y则结果为1.

模型训练

调用keras的Sequential模型，添加输入层输出层卷积层即可。

跑了30轮最后跑出85%正确率，感觉30轮还没有过拟合，提高轮数可能还能提高一点？

防染色排序部分

数据预处理

对图片内使用k-means算法，将图片内的像素点分类（这部分工作通过KmeansP.py完成），这里我直接将所有图片分了两类（被判定为染色的像素点和判断为未染色的像素点）二分类的效果图片见preventA中所有的图片。preventB中所有图片即为原图片，但排序和preventA中所有处理过的图片排序相同。

排序操作

通过函数

```
def getweight(Proportion, R, G, B):  
  
    return Proportion + 3 * (R + G + B)
```

得到某种权值。（可以更改调参）

经过对权值的排序后得到的图片按顺序输出到result1文件夹中。