

《数据科学与工程算法基础》实践报告

报告题目： 利用主成分分析进行图像有损压缩

姓 名： 郑佳辰

学 号： 10182100359

完成日期： 2020.12.11

摘要:

主成分分析(PCA)能够通过提取数据的主要成分,减少数据的特征,达到数据降维的目的。本文主要研究基于 PCA 的图像压缩改进算法,实现基于 PCA 算法的图像压缩应用,解决原始图像数据占据存储空间大而导致的图像数据计算开销问题。本项目的相关实践证明了 PCA 方法用于图像压缩和图像重构的可行性和有效性。

Abstract:

By extracting the main components in data, the method of principal component analysis (PCA) is applied to reduce the amount of features and achieves the purpose of dimension reduction in data. In this report, the improved image compression algorithms based on PCA are studied, and the applications and practical skills of PCA are exerted in the field of image compression, with the purpose to settle the serious computational overhead of image data resulted from their huge storage space occupancy. Finally, the results of the experiments verify the feasibility and effectiveness of PCA method in Image compression and image reconstruction.

一、项目概述

随着互联网的发展和大数据时代的到来，如何高效传输和存储海量数据成为大数据处理的瓶颈问题。特别是图像数据，由于其占用空间大，包含信息量丰富，如何进行有效的图像压缩成为一个值得研究的热点问题。本项目研究基于主成分分析（PCA）的图像压缩问题，针对当前图像压缩方法依赖于整个数据集，压缩效率低、占据存储空间大等问题，探索改进的 PCA 图像压缩算法及其应用。

本项目研究基于 PCA 的图像压缩改进算法。首先对图像进行分片，然后在每个分片上单独进行压缩。这样在针对一张图像进行压缩的同时，既保持了较低的压缩率又能实现良好的恢复效果。

本项目研究 PCA 方法在图像压缩中的应用。PCA 通过把数据从高维映射到低维来降低特征维度，映射的同时保留尽量多的主要信息。朴素算法可能是利用卷积的思想，用多个像素点的平均值或中位数代替单个像素的值，或是直接取出其中的部分行或列代替整体。这样做即使可以保证识别效果，压缩率仍为线性下降，降低的维数有限。若使用 PCA 的方法，迭代取出分解出来的多维主成分数据，则可实现保持信息几乎没有损失的且降低数据存储空间。这是因为信息量几乎全部集中在主成分中。因此本项目使用 PCA 对图像数据进行降维和去相关，实现基于 PCA 的图像压缩应用，解决原始图像数据占据存储空间大的问题。

二、问题描述

现实中的数据往往存在冗余等现象，在处理时计算量较大，会消耗大量算力资源。所以在图像处理领域，经常需要对图像进行压缩。图

片压缩分为有损压缩和无损压缩。有损压缩指先将一张图片进行分解，使大部分信息存在少量系数中，用较少的数据量就保留一张图片的大部分重要信息。无损压缩指分解后将系数比较小的数据用较少的位数进行编码，在缩小数据量的保证没有信息损失。本实验中的问题属于有损压缩的范畴。

在图像数据压缩技术中，首先对含有多个变量的数据进行观测，收集大量数据后进行分析寻找规律。主成分分析的图像压缩算法可以在减少分析指标的同时，尽量保持原指标的信息，从而实现对原始数据进行全面分析。针对各变量之间的相关关系，可考虑将关系紧密的变量转换成两两不相关的新变量，并减少新变量的数量。这样就可以用较少的综合指标分别代表存在于各个变量中的信息，从而实现高质量的图像压缩。本次实验将利用 PCA 方法，探索其在图像压缩中的应用场景。

三、 方法

主成分分析用于图像数据降维从而实现图像压缩，其核心思路是考察样本或属性之间的相关性并进行空间变换。基本原则是最小化压缩率或重构误差，且同时保证压缩后的样本或属性线性不相关。简而言之，我们需要选择原矩阵特征中方差较大的维度，因为方差较大的维度包含的分布信息更加明显。方差较小则代表所有样本几乎聚集在一块，包含的分布信息很少。

PCA 实现图像压缩的主要思想是将 n 维特征映射到 k 维上，这 k 维是全新的正交特征即主成分，它是在原有 n 维特征的基础上重新构造出来的 k 维特征。PCA 的工作就是从原始空间中顺序找一组相互正交的坐

标轴。由于坐标轴的选择需要使方差尽量大，所以在选择坐标轴时，应该选择与已选择的其他坐标轴正交且在剩余坐标轴中方差最大的方向。正交性用于保证选择出的特征是不相关的。在所有的 n 个坐标轴中，大部分方差集中在前 k 个坐标轴上，后面的坐标轴方差相对之前的较小。此时只保留前 k 个含有大部分方差的坐标轴，即可保留包含绝大部分方差的维度特征，并忽略包含方差较少的特征维度，继而实现对图像数据特征的降维，即实现图片的有损压缩。

选择出方差最大的几个维度看似抽象，实际代码编写中只需计算数据矩阵的协方差矩阵，得到协方差矩阵的特征值和特征向量。选出其最大的 k 个特征值对应特征向量，就是上述方法中方差最大 k 个特征。将它们组成矩阵，就是将原数据矩阵转换到新空间，实现数据特征的降维，进而完成了基于 PCA 的图像压缩过程。

代码实现大致如下。在读入图片后，首先要把 RGB 三个通道分开，然后对每个通道进行 PCA 压缩。这一步在我编写的函数 PCA 中，对每层进行 PCA 压缩的过程包含在函数 comp_2d 中。

```
def PCA(imagepath, newpath, alpha):
    pic = imageio.imread(imagepath)
    pic_np = np.array(pic)
    pic_r = pic_np[:, :, 0]
    pic_g = pic_np[:, :, 1]
    pic_b = pic_np[:, :, 2]

    pic_r_recon = comp_2d(pic_r, alpha)
    pic_g_recon = comp_2d(pic_g, alpha)
    pic_b_recon = comp_2d(pic_b, alpha)

    recon_color_image = np.dstack((pic_r_recon, pic_g_recon, pic_b_recon))
    recon_color_image = Image.fromarray(recon_color_image)
    # recon_color_image.show()
    imageio.imwrite(newpath, recon_color_image)
```

根据 PCA 算法的步骤，先对样本进行去中心化，即每行减去该行的均值。这样相当于做了归一化，可以使生成的图片质量更好。然后使用

`np.linalg.eig` 计算其协方差矩阵的特征值和特征向量，并将这些特征值按照从大到小的顺序排好序。协方差矩阵的计算即是原矩阵乘其转置，即 AA^T 。得到的特征值就是原矩阵的奇异值和左奇异特征向量。

```
def comp_2d(image_2d, alpha):
    # 样本去中心化
    center = np.mean(image_2d, axis=0)
    cov_mat = image_2d - np.mean(image_2d, axis=0)

    # np.cov计算协方差矩阵
    # np.linalg.eig计算特征值和特征向量
    # 求出的特征值就是有序的
    eig_val, eig_vec = np.linalg.eig(np.cov(cov_mat))

    # 特征值从小到大排序
    idx = np.argsort(eig_val)
    idx = idx[::-1]
    eig_vec = eig_vec[:,idx]
    eig_val = eig_val[idx]
```

接下来进行主成分选取，选取一定数量的绝对值最大的特征值，并找出对应的特征向量。选择的特征值的比例用 `alpha` 表示。之后将其投影到新空间里，这样就生成了压缩后的矩阵。

```
# 特征值个数选取
eig_val_sum = np.sum(eig_val)
p = np.linalg.matrix_rank(image_2d)
part_eig_val = 0
for i in range(p):
    part_eig_val = part_eig_val + eig_val[i]
    if part_eig_val > (alpha * eig_val_sum):
        break

# 选取主特征值的个数
numpc = i
eig_vec = eig_vec[:, range(numpc)]

# 投影到新空间中
score = np.dot(eig_vec.T, cov_mat)
```

最后降维输出。计算压缩率并重建图像，将 RGB 值转化为 `uint8` 类型返回。最后将三个通道拼在一起，生成最终的图片。

```
# 计算空间占用
rate = (np.size(eig_vec)+np.size(score)+np.size(center))/np.size(image_2d)
# print(rate)
rates.append(rate)

# 重建图像
recon = np.dot(eig_vec, score) + np.mean(image_2d, axis=0) # 归一化可使图像质量更好
recon_img_mat = np.uint8(np.absolute(recon)) # 绝对值，离散；成为合法rgb值
return recon_img_mat
```

另外，目前获得协方差矩阵的特征值特征向量的方法包括对协方差矩阵进行特征值分解和奇异值分解两种。本项目使用计算了协方差矩阵的特征值和特征向量，基于特征值分解协方差矩阵实现 PCA 算法。使用奇异值对协方差矩阵进行分解有时结果会更好。因为 SVD 分解可以将其分解出左右两个奇异矩阵。左奇异矩阵可以用于对行的压缩，右奇异矩阵可以用于对列的压缩。这样就可以得到行和列两个方向的 PCA 降维。

四、 实验结果

在本次实验中，选取的图片有农田，飞机，海岸三类，每类有 100 张图片，共 300 张图片。所有图片大小均为 256*256 个像素，每张图都有 3 个通道，分别为 RGB（即红绿黄）。经过压缩后得到的图片仍为彩色，256*256 个像素。在计算压缩率时，原图像大小应为原始矩阵的大小。压缩后图像的大小为特征值，特征向量和去中心化时减去的 center 值。

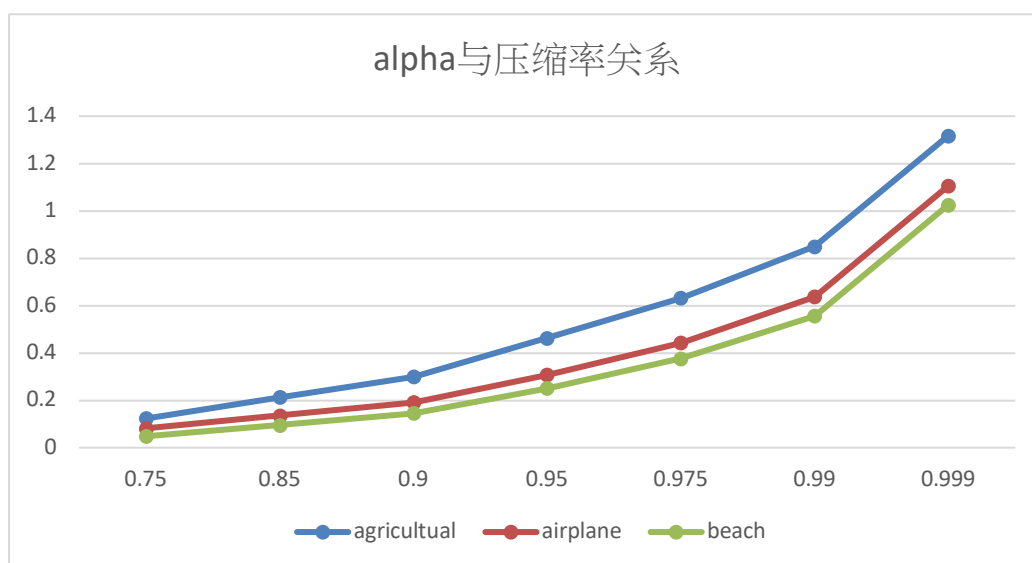
在实验过程中，可以调节的主要参数是选取的特征值的比例，以下继续用 alpha 表示。当 alpha 取值不同的时候，图片压缩的效果也会有很大差异。在实验中，我计算了三组图像的压缩率，当主成分的比例 alpha 从 0.75 增加到 0.999 之后，压缩比的变化如图所示。注意从上到下的三组数值分别为 beach，agricultural，airplane 的压缩比的平均值。

/Library/Frameworks/F	/Library/Frameworks/I	/Library/Frameworks/P
0.04812748015873016	0.09516265707671957	0.14529782572751324
41398.85974124155	41166.71060516785	40964.90419944502
0.12294270833333333	0.21278645833333334	0.29854166666666665
42725.87601605706	42322.91552529278	41771.81674860279
0.08168287837615283	0.13596374752964427	0.19081707015810276
41613.928209840626	42069.822277058025	42248.39911156211
alpha = 0.75	alpha = 0.85	alpha = 0.9

/Library/Frameworks/P	/Library/Frameworks/P	/Library/Frameworks/P
0.2513494130291005	0.37630828373015873	0.5562194113756613
40167.05925681112	39105.52859292717	36820.28387454639
0.46265625	0.632421875	0.8488802083333333
40472.824511492225	38494.425850040825	34664.1031300892
0.3065401124011858	0.44257827939723315	0.6367781414690381
42171.51974341361	41833.51019223021	40843.94092698675
alpha = 0.95	alpha = 0.975	alpha = 0.99

每两个数值为一组，上面的小于1的是压缩率，下面的是重构误差。

重构误差将在下面的分析中用到。根据以上数据作图可得 **alpha** 与压缩比的关系图。注意每两个横坐标之间间隔并不相等。可以看出，**alpha** 和压缩率之间大致满足指数的关系。结合数据并进行分析可知，当 **alpha** 线性下降时，其压缩率呈指数下降趋势。

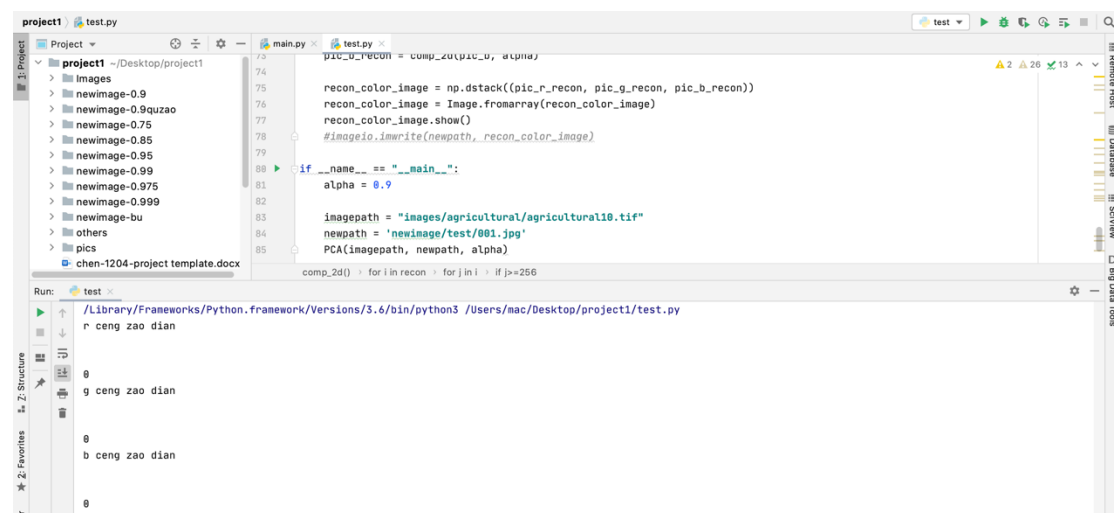


可以看出无论 **alpha** 取值为多少，压缩率的相对大小是相同的。观察输出数据，可以发现压缩率大小从高到低分别为农田，飞机，海滩。即海滩压缩率最低，飞机次之，而农田的压缩率最高。而一般来说，压缩率越高，说明压缩的程度越高，压缩质量一般会相对较差。我的实验结果也大致符合这一规律。即直观上看来，农田的和原来的图片最为相近，压缩效果最好。飞机由于噪点较多，看上去效果并不好，但在去噪

之后整体也比较符合原图。海滩则几乎看不清边缘。

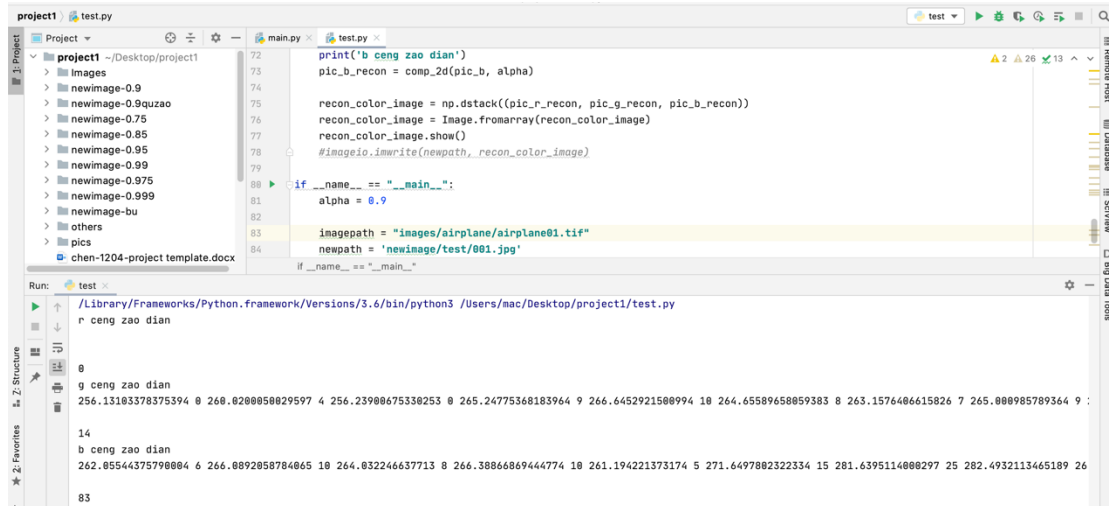
在压缩过程中，经常会出现许多噪点，如所示图片中黑色的部分。噪点产生的原因有很多。根据我的实验，其直接原因是压缩后的图像中存在 RGB 值超过 255 的点。这些点在转化成 uint8 类型时会对 256 进行取模，使得该点的 RGB 值很小。而事实上，这些点的 RGB 值应该很大，所以才会产生超过 255 的值。

取 $\alpha=0.9$ 时，运行测试程序 test.py 查看图片中所有噪点的 RGB 值。其中，第一行指示这些噪点在哪个图层。之后一行为两个数组，前一个为超过 255 的不合法的 RGB 值，第二个数为经过 uint8 转化之后的值。最后为本通道的噪点总数。对于噪点较多且密集的 beach74.tif 图片来说，其输出结果很多。这表示其噪点很多，每层的噪点数量都有几千个。

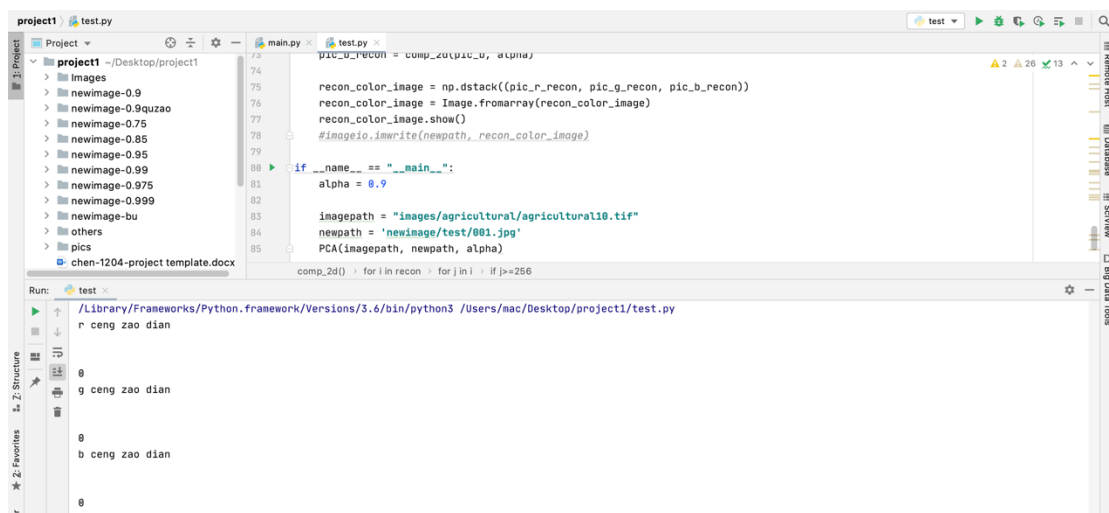


```
project1 test.py
Project ~./Desktop/project1
  Images
  newimage-0.9
  newimage-0.9quzao
  newimage-0.75
  newimage-0.85
  newimage-0.95
  newimage-0.99
  newimage-0.975
  newimage-0.999
  newimage-bu
  others
  pics
  chen-1204-project template.docx
main.py test.py
73 pic_b_recon = comp_2d(pic_b, alpha)
74
75 recon_color_image = np.dstack((pic_r_recon, pic_g_recon, pic_b_recon))
76 recon_color_image = Image.fromarray(recon_color_image)
77 recon_color_image.show()
78 #imageio.imwrite(newpath, recon_color_image)
79
80 if __name__ == "__main__":
81     alpha = 0.9
82     imagepath = "images/agricultural/agricultural10.tif"
83     newpath = 'newimage/test/001.jpg'
84     PCA(imagepath, newpath, alpha)
85
comp_2d() for i in recon for j in i if j>=256
Run: test
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3 /Users/mac/Desktop/project1/test.py
r ceng zao dian
0
g ceng zao dian
0
b ceng zao dian
0
0
```

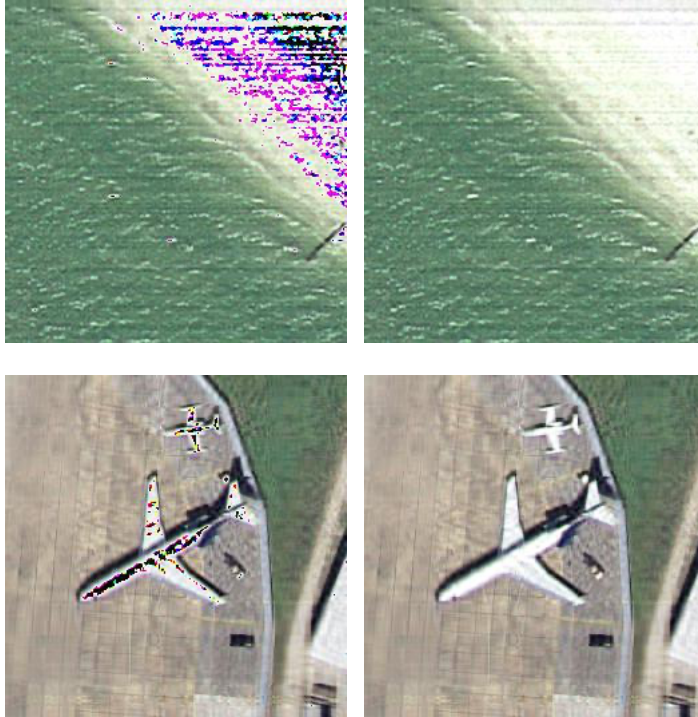
而对于只有少量噪点的 airplane01.tif，其每层的噪点数量较少。如图所示。



对于 agricultural10.tif，其三层的不合法 RGB 值都为 0，所以没有出现噪点。



一个有效的去噪方法是检查所有像素点的 RGB 值，若有超过 255 的值，则将其修正为 255。然后再转化为合法的 uint8 类型作为 RGB 的值。这种去噪方法下得到的压缩后的图片几乎不会有噪点产生。部分图片去噪前后的对比如图所示，以 beach74.tif 和 airplane20.tif 为例。

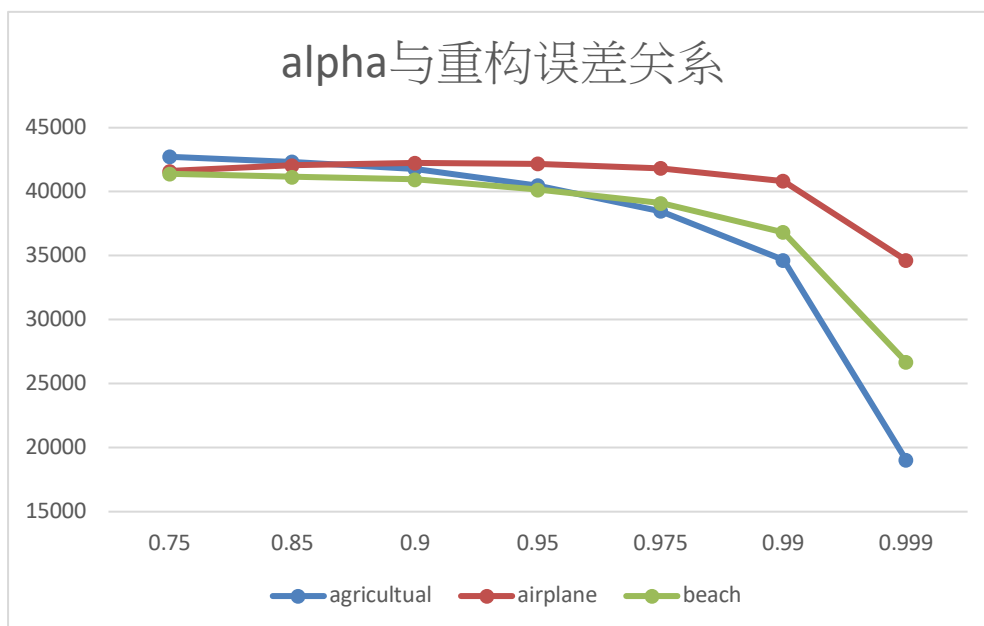


去噪部分的代码如下图所示。

```
for i in range(len(recon)): # 去除噪点
    for j in range(len(recon[i])):
        if recon[i][j] > 256:
            recon[i][j] = 255
```

产生噪点的深层原因可能和图片的同质化程度有关。观察压缩后的图片的噪点多少，从直观上可以看出，飞机图片对应的噪点最多，海滩次之，农田图片对应的噪点最少。这是由于飞机图片的同质化程度较低，即某像素和周围像素的差异较大。对于飞机图片来说，图片中一架飞机形状不规则，并且和周围地面的颜色差异较大，同质化程度低，噪点相对较多。而在农田的图片中，重复部分较多，颜色差异较小，故噪点出现的少。

在去噪之后，我计算了各类图片重构误差的平均值。重构误差用每个通道的压缩后重建出的矩阵与原矩阵的差的矩阵 F 范数表示。部分 α 取值下的重构误差已在前述图中给出。重构误差与 α 之间的关系如下表所示。



重构误差的计算代码如下所示。

```
# 计算重构误差
error = np.linalg.norm(image_2d - recon_img_mat, "fro")
errors.append(error)
```

由上图可以看出，在去噪的情况下，当 **alpha** 取值较大时，农田的重构误差最小，海滩次之，飞机的重构误差最大。从图片同质化的角度来讲，同质化程度越高，重构误差越小。此外，图片的压缩质量不完全依赖于重构误差。在压缩率较小时，农田图像的重构误差相对其他两种较大，而其压缩质量仍然不错。这说明图片的压缩质量不能单纯用重构误差来比较，还应结合其他因素进行综合判断。

五、 结论

通过本项目的实验研究，可以看出将 **PCA** 算法应用于图像压缩有以下优点。它对图像数据实现了降维，使得数据集更易使用。而对降维后的图像数据进行数据处理，可以大幅度降低算法的时空复杂度。同时，**PCA** 算法可以用于数据预处理，利用它可以去除图片的部分噪点。除此之外，**PCA** 算法的应用几乎没有参数的限制。

同时，PCA 算法在应用时也存在一些不足之处。PCA 算法无法很好的利用已知的无法被参数化的特征数据。采用特征值分解时无法对行和列分开处理。并且非高斯分布情况下，PCA 方法得出的主成分可能并不是最优的。

在本项目中，经过对 PCA 方法图片有损压缩的相关工作，得出以下结论。PCA 方法可以很好的将图片有损压缩成相应向量，特征值等值，从而减少图片的存储空间。通过调节取出的特征值的数量，可以调节图片的压缩效果。另一方面，压缩的效果也取决于图片本身的同质化程度。一般来说，取出的特征值越多，压缩效果越好，重构误差越低，但其压缩率也会相应地上升。从图片自身的角度来说，PCA 算法更适用于同质化程度高的图片。

本项目的实验研究主要针对 PCA 算法在图像压缩领域的应用。在本次实验中，噪点产生原因，基于 SVD 的 PCA 算法等方面仍有待进一步研究。PCA 算法的应用领域很广泛，除了去除数据冗余与噪点的数据降维和数据压缩，还包括高维数据集的探索与可视化、数据预处理、图像语音通信的分析处理等等。

通过这次实验，我学习到了使用 PCA 进行图片压缩的具体方法，加深了对特征值，奇异值及其相关算法的了解。最后，感谢高老师和助教的讲解和指导，您们辛苦了！