图像压缩与线性代数

图像压缩是数字图像处理中的一个基本任务，旨在减少图像的存储大小，同时保持其视觉质量。实现这一目标的一种强大方法是通过线性代数，特别是奇异值分解（SVD）。这种技术使我们能够以更紧凑的形式表示图像矩阵，有效地丢弃不重要的信息，同时保留基本特征。

以下的 Python 代码演示了如何使用 SVD 压缩图像。该过程涉及将图像分解为其组成部分，通过保留最重要的特征子集来压缩这些组件，然后重建压缩后的图像。这种方法可以应用于灰度图像和彩色图像，提供一种灵活且数学上可靠的方法来减少图像大小。

```
import numpy as np
from PIL import Image
import argparse
import os

def compress_image(image_path, compression_factor=0.1):
    # 打开图像并将其转换为 numpy 数组
    img = Image.open(image_path)
    img_array = np.array(img, dtype=float)

    # 检查图像是灰度图像还是彩色图像
    if len(img_array.shape) == 2:  # 灰度图像
        # 对图像数组执行 SVD
        U, S, Vt = np.linalg.svd(img_array, full_matrices=False)

        # 通过保留最顶部的奇异值来压缩图像
        k = int(compression_factor * min(img_array.shape))
        S_compressed = np.diag(S[:k])
        U_compressed = U[:, :k]
        Vt_compressed = Vt[:k, :]

        # 重建压缩图像
        img_compressed = np.dot(U_compressed, np.dot(S_compressed, Vt_compressed))
    else:  # 彩色图像
        # 对每个通道分别执行 SVD
        img_compressed = np.zeros_like(img_array)
        for i in range(img_array.shape[2]):  # 迭代每个通道
            channel = img_array[:, :, i]
            U, S, Vt = np.linalg.svd(channel, full_matrices=False)

            # 通过保留最顶部的奇异值来压缩通道
```

```
            k = int(compression_factor * min(channel.shape))
            S_compressed = np.diag(S[:k])
            U_compressed = U[:, :k]
            Vt_compressed = Vt[:k, :]

            # 重建压缩通道
            img_compressed[:, :, i] = np.dot(U_compressed, np.dot(S_compressed, Vt_compressed))

    # 将值剪辑在 0 和 255 之间，并转换回 uint8
    img_compressed = np.clip(img_compressed, 0, 255).astype(np.uint8)

    # 通过在原始文件名中添加'_compressed'来生成输出路径
    file_name, file_extension = os.path.splitext(image_path)
    output_path = f"{file_name}_compressed{file_extension}"

    # 保存压缩图像
    compressed_img = Image.fromarray(img_compressed)
    compressed_img.save(output_path)

    return output_path

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description=" 使用 SVD 压缩图像。")
    parser.add_argument("input_file", help=" 输入图像文件的路径")
    parser.add_argument("--compression_factor", type=float, default=0.1, help=" 压缩因子（默认：0.1）")
    args = parser.parse_args()

    output_file = compress_image(args.input_file, args.compression_factor)
    print(f" 压缩图像已保存为：{output_file}")
```