

الخطي الجبر باسخدام الصور ضغط

على تحافظ بينما الصور تخزين حجم تقليل إلى تهدف الرقمية، الصور معالجة في أساسية مهمة هو الصور على التضغط تسمح. $\square\square\square\square$ الغريبة القيمة تفكيك وخاصة الخطي، الجبر باستخدام هو ذلك لتحقيق القوية الطرق أحد البصرية. جودتها تحافظ بينما أهمية الأقل المعلومات من التخلل من تعقيدًا، أكثر شغل في الصورة مصفوفة بتقديم لنا التقنية هذه الأساسيّة. الميزات على

المكونات هذه ضغط مكوناتها، إلى الصورة تفكيك العملية هذا يتضمن $\square\square\square$ باستخدام صورة ضغط كيفية التالى الكودي ووضوح الصور على الطريقة هذه تطبيقي يمكن المضاغطة. الصورة بناء إعادة ثم أهمية، الأكثر الميزات من جزء فقط الاحتفاظ خلل من الصورة. حجم لتخفيض رياضيًا ومثبتة مرنة طريقة يوفر مما والألوان، الرمادية

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

def compress_image(image_path, compression_factor=0.1):
    #
    img = Image.open(image_path)
    img_array = np.array(img, dtype=float)

    #
    if len(img_array.shape) == 2: #
        # SVD
        U, S, Vt = np.linalg.svd(img_array, full_matrices=False)

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

        #
        img_compressed = np.dot(U_compressed, np.dot(S_compressed, Vt_compressed))
    else: #
        # SVD
        img_compressed = np.zeros_like(img_array)
        for i in range(img_array.shape[2]): #
            channel = img_array[:, :, i]
            U, S, Vt = np.linalg.svd(channel, full_matrices=False)
```

```

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

#
img_compressed[:, :, i] = np.dot(U_compressed, np.dot(S_compressed, Vt_compressed))

#           0   255           uint8
img_compressed = np.clip(img_compressed, 0, 255).astype(np.uint8)

#           '_compressed'
file_name, file_extension = os.path.splitext(image_path)
output_path = f"{file_name}_compressed{file_extension}"

#
compressed_img = Image.fromarray(img_compressed)
compressed_img.save(output_path)

return output_path

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description= "           SVD.")
    parser.add_argument("input_file", help= "           " (
    parser.add_argument("--compression_factor", type=float, default=0.1, help= "           :           ) "(0.1)
    args = parser.parse_args()

    output_file = compress_image(args.input_file, args.compression_factor)
    print(f"           :           }output_file}")

```