

Compresión de Imágenes usando Álgebra Lineal

La compresión de imágenes es una tarea fundamental en el procesamiento digital de imágenes, con el objetivo de reducir el tamaño de almacenamiento de las imágenes mientras se mantiene su calidad visual. Uno de los métodos más poderosos para lograr esto es mediante el uso del álgebra lineal, en particular la Descomposición en Valores Singulares (SVD). Esta técnica nos permite representar una matriz de imagen en una forma más compacta, descartando efectivamente la información menos importante mientras se retienen las características esenciales.

El siguiente código en Python demuestra cómo comprimir una imagen utilizando SVD. El proceso implica descomponer la imagen en sus componentes constitutivos, comprimir estos componentes reteniendo solo un subconjunto de las características más significativas y luego reconstruir la imagen comprimida. Este enfoque se puede aplicar tanto a imágenes en escala de grises como a imágenes en color, ofreciendo un método flexible y matemáticamente sólido para reducir el tamaño de la imagen.

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

def compress_image(image_path, compression_factor=0.1):
    # Abrir la imagen y convertirla a un array de numpy
    img = Image.open(image_path)
    img_array = np.array(img, dtype=float)

    # Verificar si la imagen es en escala de grises o en color
    if len(img_array.shape) == 2:  # Imagen en escala de grises
        # Realizar SVD en el array de la imagen
        U, S, Vt = np.linalg.svd(img_array, full_matrices=False)

        # Comprimir la imagen manteniendo solo los valores singulares más importantes
        k = int(compression_factor * min(img_array.shape))
        S_compressed = np.diag(S[:k])
        U_compressed = U[:, :k]
        Vt_compressed = Vt[:k, :]

        # Reconstruir la imagen comprimida
        img_compressed = np.dot(U_compressed, np.dot(S_compressed, Vt_compressed))
    else:  # Imagen en color
        # Realizar SVD en cada canal por separado
        img_compressed = np.zeros_like(img_array)
```

```

for i in range(img_array.shape[2]): # Iterar sobre cada canal
    channel = img_array[:, :, i]
    U, S, Vt = np.linalg.svd(channel, full_matrices=False)

    # Comprimir el canal manteniendo solo los valores singulares más importantes
    k = int(compression_factor * min(channel.shape))
    S_compressed = np.diag(S[:k])
    U_compressed = U[:, :k]
    Vt_compressed = Vt[:k, :]

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

# Limitar los valores para que estén entre 0 y 255, y convertir de nuevo a uint8
img_compressed = np.clip(img_compressed, 0, 255).astype(np.uint8)

# Generar la ruta de salida agregando '_compressed' al nombre del archivo original
file_name, file_extension = os.path.splitext(image_path)
output_path = f"{file_name}_compressed{file_extension}"

# Guardar la imagen comprimida
compressed_img = Image.fromarray(img_compressed)
compressed_img.save(output_path)

return output_path

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Comprimir una imagen utilizando SVD.")
    parser.add_argument("input_file", help="Ruta al archivo de imagen de entrada")
    parser.add_argument("--compression_factor", type=float, default=0.1, help="Factor de compresión (predeterminado: 0.1)")
    args = parser.parse_args()

    output_file = compress_image(args.input_file, args.compression_factor)
    print(f"Imagen comprimida guardada como: {output_file}")

```