

Compression d'images par algèbre linéaire

La compression d'image est une tâche fondamentale dans le traitement numérique des images, visant à réduire la taille de stockage des images tout en maintenant leur qualité visuelle. Une méthode puissante pour y parvenir est l'utilisation de l'algèbre linéaire, en particulier la décomposition en valeurs singulières (SVD). Cette technique nous permet de représenter une matrice d'image sous une forme plus compacte, en éliminant efficacement les informations moins importantes tout en conservant les caractéristiques essentielles.

Le code Python suivant montre comment compresser une image en utilisant la SVD. Le processus implique la décomposition de l'image en ses composants constitutifs, la compression de ces composants en ne conservant qu'un sous-ensemble des caractéristiques les plus significatives, puis la reconstruction de l'image compressée. Cette approche peut être appliquée à la fois aux images en niveaux de gris et aux images en couleur, offrant une méthode flexible et mathématiquement solide pour réduire la taille des images.

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

def compress_image(image_path, compression_factor=0.1):
    # Ouvrir l'image et la convertir en un tableau numpy
    img = Image.open(image_path)
    img_array = np.array(img, dtype=float)

    # Vérifier si l'image est en niveaux de gris ou en couleur
    if len(img_array.shape) == 2: # Image en niveaux de gris
        # Effectuer la SVD sur le tableau d'image
        U, S, Vt = np.linalg.svd(img_array, full_matrices=False)

        # Compresser l'image en ne conservant que les valeurs singulières les plus élevées
        k = int(compression_factor * min(img_array.shape))
        S_compressed = np.diag(S[:k])
        U_compressed = U[:, :k]
        Vt_compressed = Vt[:k, :]

        # Reconstruire l'image compressée
        img_compressed = np.dot(U_compressed, np.dot(S_compressed, Vt_compressed))
    else: # Image en couleur
        # Effectuer la SVD sur chaque canal séparément
```

```

img_compressed = np.zeros_like(img_array)
for i in range(img_array.shape[2]): # Itérer sur chaque canal
    channel = img_array[:, :, i]
    U, S, Vt = np.linalg.svd(channel, full_matrices=False)

    # Compresser le canal en ne conservant que les valeurs singulières les plus élevées
    k = int(compression_factor * min(channel.shape))
    S_compressed = np.diag(S[:k])
    U_compressed = U[:, :k]
    Vt_compressed = Vt[:k, :]

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

# Limiter les valeurs entre 0 et 255, et convertir en uint8
img_compressed = np.clip(img_compressed, 0, 255).astype(np.uint8)

# Générer le chemin de sortie en ajoutant '_compressed' au nom de fichier d'origine
file_name, file_extension = os.path.splitext(image_path)
output_path = f"{file_name}_compressed{file_extension}"

# Enregistrer l'image compressée
compressed_img = Image.fromarray(img_compressed)
compressed_img.save(output_path)

return output_path

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Compresser une image en utilisant la SVD.")
    parser.add_argument("input_file", help="Chemin vers le fichier image d'entrée")
    parser.add_argument("--compression_factor", type=float, default=0.1, help="Facteur de compression (par défaut 0.1)")
    args = parser.parse_args()

    output_file = compress_image(args.input_file, args.compression_factor)
    print(f"Image compressée enregistrée sous: {output_file}")

```