گزارش پروژه پایانی

درس جبرخطی

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

استاد درس : دکتر پیمان ادیبی

تهیه کننده :

محمد حسین دهقانی اشکذری

تیر ۱۴۰۲

مقدمه :

در این پروژه قصد داشتیم با تعدادی تصویر به شرح زیر کار کنیم:

در قدم اول باید پس از خواندن اطلاعات تصاویر در برنامه، تابعی را پیاده‌سازی می‌کردیم تا یک نویز گاوسی به هرکدام از تصاویر بدهد.

در قدم دوم باید عمل دی‌نویز کردن کد را انجام می‌دادیم که برای این کار از Singular Value Decomposition یا به اختصار SVD استفاده کردیم.

توضیح کد و الگوریتم :

```
import numpy as np
from PIL import Image
import math
import random
from matplotlib import pyplot as plt
```

در اولین بخش از کد کتابخانه‌هایی را که در ادامه کار به آنها نیاز داریم اضافه می‌کنیم:

کتابخانه Numpy برای محاسبات انجام شده

کتابخانه PIL برای کار با تصاویر

کتابخانه‌های Math, Random برای پیاده‌سازی توابع مورد نیاز

کتابخانه Matplotlib برای نمایش نتایج

```
# Hyperparameters
mean = 30
std = 80
```

در قسمت بعدی شاهد مقدار دهی دو عدد ابرپارامتر هستیم که به ترتیب میانگین و انحراف معیار را مقدار دهی می‌کنند. این مقادیر در تابعی که باعث اعمال نویز گاوسی بر روی تصاویر می‌شود کاربرد دارد.

```
def generate_normal(mean, std):
    u1 = random.random()
    u2 = random.random()
    r = math.sqrt(-2 * math.log(u1))
    theta = 2 * math.pi * u2
    z = r * math.cos(theta)
    return mean + z * std
```

این تابع از یک توزیع نرمال گاوسی با درنظر گرفتن میانگین و انحراف معیار داده شده، یک تک مقدار برمی‌گرداند که بیشتر جنبه ریاضیاتی دارد.

```
def g_noiser(input_image, mean, std):
    shape = input_image.shape
    noise = np.zeros(shape)
    for i in range(3):
        for j in range(shape[0]):
            for k in range(shape[1]):
                noise[j, k, i] = generate_normal(mean, std)
    return np.clip(image + noise, 0, 255).astype(np.uint8)
```

این تابع با دریافت یک تصویر ورودی، میانگین و انحراف معیار یک نویز گاوسی را در تصویر داده شده ایجاد می‌کند و تصویر نویزی را خروجی می‌دهد. این تابع در ابتدا با ساخت یک آرایه با درایه‌های صفر و پرکردن درایه‌های آن به وسیله تابع قبلی در پایان، تصویر اولیه را با این مقادیر نویز جمع کرده و حاصل را خروجی می‌دهد.

```python
def eig(A):
    n, m = A.shape
    if n != m:
        raise ValueError("Matrix must be square")

    if not np.allclose(A, A.T.conj()):
        raise ValueError("Matrix must be Hermitian or real symmetric")

    eigvals = np.zeros(n)
    eigvecs = np.eye(n)

    for i in range(n):
        eigval, eigvec = power_method(A, eigvecs[:, i])

        eigvals[i] = eigval
        eigvecs[:, i] = eigvec

        A = A - eigval * np.outer(eigvec, eigvec)

    return eigvals, eigvecs
```

این تابع مقدار ویژه و بردار ویژه را برای یک بردار و با هدف استفاده در تابع SVD محاسبه می‌کند و این دو را خروجی می‌دهد.

```python
def power_method(A, x0, tol=1e-8, maxiter=1000):
    x = x0 / np.linalg.norm(x0)

    for i in range(maxiter):
        Ax = A.dot(x)

        x_new = Ax / np.linalg.norm(Ax)

        if np.linalg.norm(x_new - x) < tol:
            break

        x = x_new

    eigval = x.T.dot(A).dot(x)

    return eigval, x
```

این تابع برای پیدا کردن مقدار ویژه و بردار ویژه را برای بردار داده شده محاسبه می‌کند.

```python
def svd(A):
    ATA = A.T.dot(A)
    eigvals, eigvecs = eig(ATA)

    S = np.sqrt(eigvals)

    sort_indices = np.argsort(S)[::-1]
    S = S[sort_indices]

    Vt = eigvecs[:, sort_indices]

    U = A.dot(Vt) / S[np.newaxis, :]

    return U, S, Vt.T
```

این تابع با استفاده از دو تابعی که در بالا توضیح داده شدند به محاسبه مقادیر تکین می‌پردازد.

```python
def svd_denoiser(noisy_image):
    img = noisy_image.astype(float) / 255.0

    R, G, B = img[:, :, 0], img[:, :, 1], img[:, :, 2]

    Ur, sr, Vr = svd(R)
    Ug, sg, Vg = svd(G)
    Ub, sb, Vb = svd(B)

    threshold = 3

    sr_thresh = np.where(sr < threshold, 0, sr - threshold)
    sg_thresh = np.where(sg < threshold, 0, sg - threshold)
    sb_thresh = np.where(sb < threshold, 0, sb - threshold)

    R_denoised = Ur.dot(np.diag(sr_thresh)).dot(Vr)
    G_denoised = Ug.dot(np.diag(sg_thresh)).dot(Vg)
    B_denoised = Ub.dot(np.diag(sb_thresh)).dot(Vb)

    img_denoised = np.stack([R_denoised, G_denoised, B_denoised], axis=2)

    img_denoised = (img_denoised * 255.0).astype(np.uint8)

    return img_denoised
```

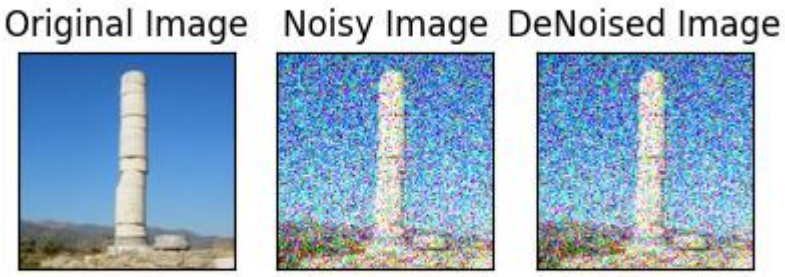
این تابع تصویر داده شده را به سه قسمت رنگی R, G, B تقسیم می‌کند و پس از محاسبه مقادیر تکین هر ماتریس با استفاده از تابع قبلی و با در نظر گرفتن threshold مقادیر تکینی که از این مقدار کمتر هستند را به عنوان نویز در نظر می‌گیرد و مقدار آن‌ها را صفر می‌کند و از سایر مقادیر threshold را کم می‌کند. در نهایت پس از دینویز کردن هرکدام از کانال‌های رنگی آنها را به فرمت تصویر اولیه در می‌آورد و تصویر دینویز شده را خروجی می‌دهد.

```python
if __name__ == '__main__':
    image = Image.open(r'./images/10097995255_6a637f6881_m.jpg')
    image = np.asarray(image)
    noisy_image = g_noiser(image, mean, std)
    denoised_image = svd_denoiser(noisy_image)
    plt.subplot(1, 4, 1)
    plt.imshow(image)
    plt.title('Original Image')
    plt.xticks([])
    plt.yticks([])
    plt.subplot(1, 4, 2)
    plt.imshow(noisy_image)
    plt.title('Noisy Image')
    plt.xticks([])
    plt.yticks([])
    plt.subplot(1, 4, 3)
    plt.imshow(denoised_image)
    plt.title('DeNoised Image')
    plt.xticks([])
    plt.yticks([])
    plt.show()
```

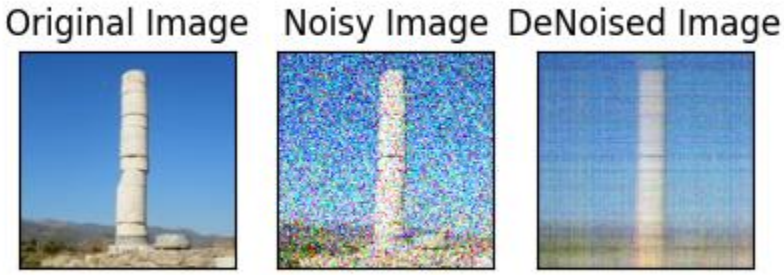
در پایان و در تابع main، اعمالی مانند خواندن تصاویر از فایل، نویزی کردن و دی‌نویز کردن آن و نمایش آن را آنجام می‌دهیم.

درمورد مقدار متغیر threshold می‌توان گفت بالاتر بردن میزان این متغیر باعث بالاتر بردن میزان دینوزینگ و کاهش مقدار جزئیات موجود در تصویر می‌شود در حالی که کاهش مقدار threshold باعث باعث کاهش میزان دینوزینگ و افزایش میزان جزئیات در تصویر می‌باشد برای مثال در تصویر زیر یک پاسخ با سه مقدار متفاوت برای threshold مقایسه شده‌اند.

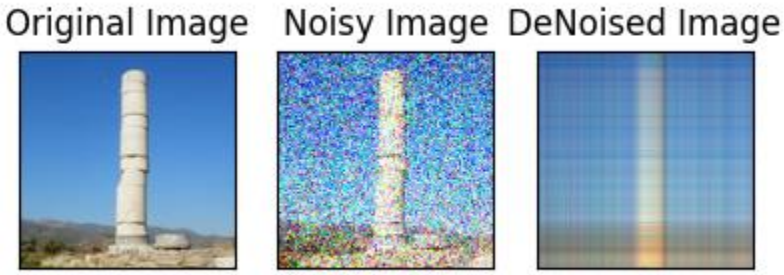
Threshold = 0



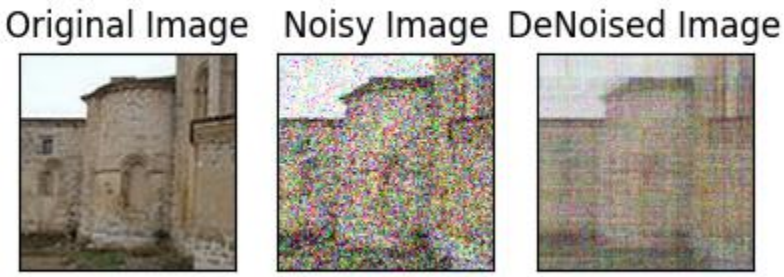
Threshold = 4

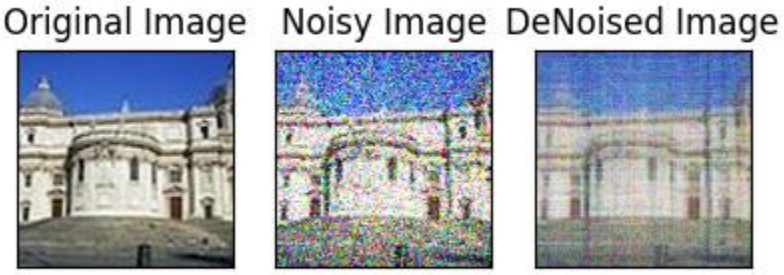


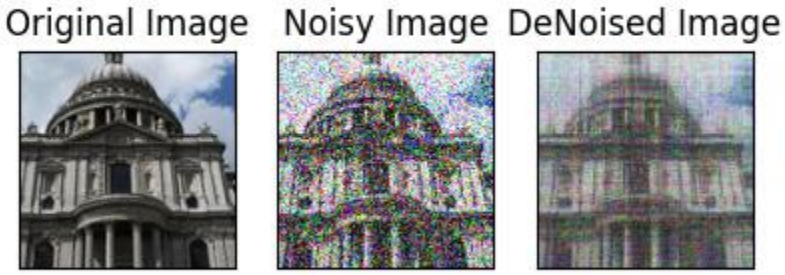
Threshold = 8

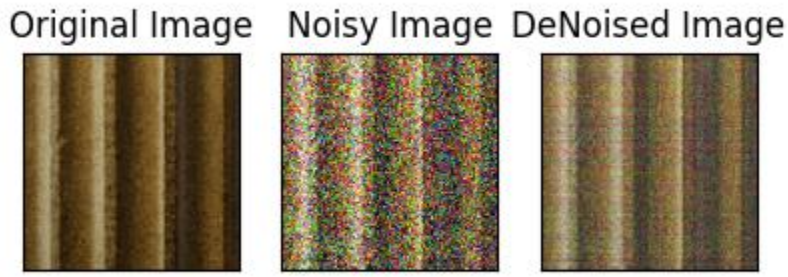


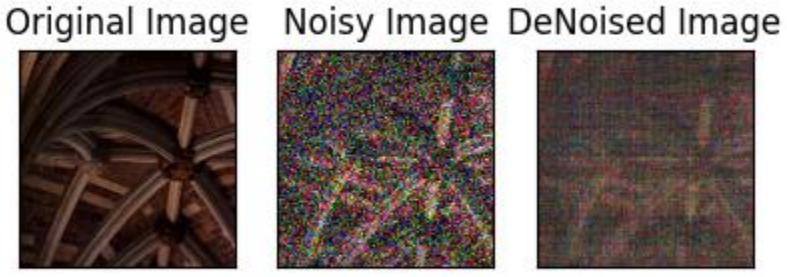
نمونه تصاویر خواسته شده:

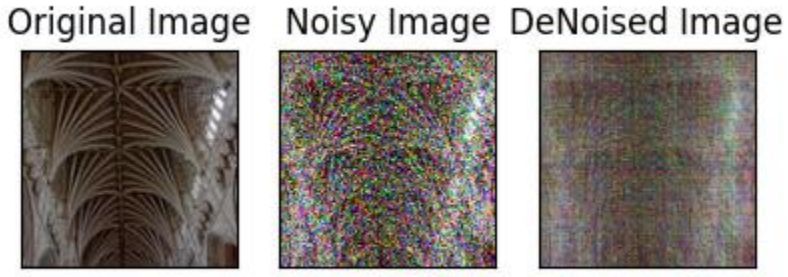


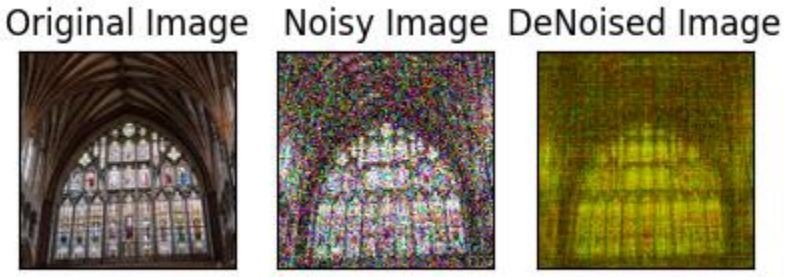


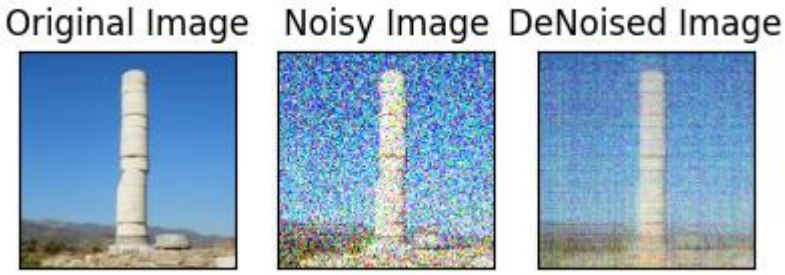


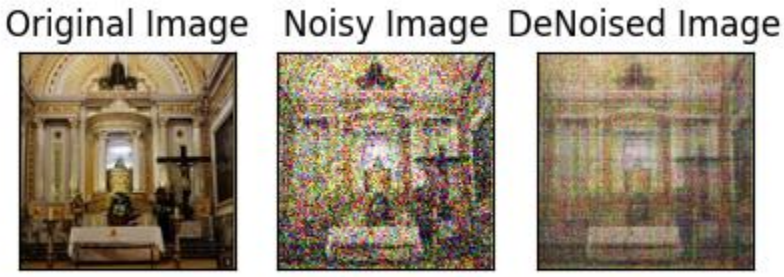

Original Image
Noisy Image
DeNoised Image

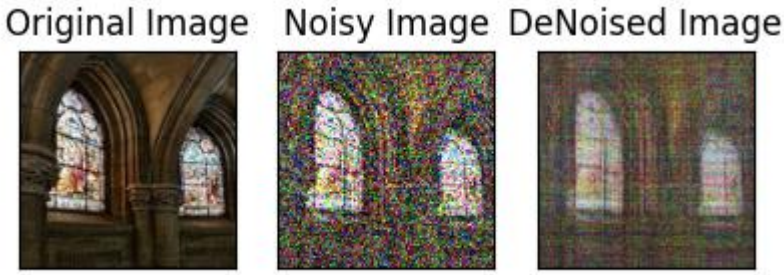

Original Image
Noisy Image
DeNoised Image


Original Image
Noisy Image
DeNoised Image


Original Image
Noisy Image
DeNoised Image







همانطور که قبلا هم مفصل تر توضیح داده شد. به طور خلاصه، این الگوریتم با محاسبه مقادیر تکین برای هر کدام از کانال‌های رنگی قرمز، سبز و آبی برای هر تصویر نویزی و مقایسه آن‌ها با میزان threshold نویزها را شناسایی می‌کند.