

## پروژه درس جبر خطی

کتابخانه های استفاده شده:

random,PIL.Image, numpy, matplotlib.pyplot, skimage.util

توضیح الگوریتم و نحوه کار آن:

تابع `:add_gaussian_noise(image, mean, std_dev)` نویز گوسی با میانگین(مقدار متوسط نویز) و انحراف استاندارد مشخص شده تولید می کند، نویز را به تصویر اضافه می کند و تصویر نویز دار را بر می گرداند.

تابع سه پارامتر دارد:  
- `shape`: تابلی که شکل آرایه نویز مورد نظر را نشان می دهد. اندازه و ابعاد نویز را تعیین می کند.  
- `mean`: میانگین توزیع گاوی که نویز از آن تولید می شود.  
- `std_dev`: انحراف معیار توزیع گاوی.

این تابع با محاسبه تعداد کل عناصر در آرایه نویز بر اساس "shape" ارائه شده شروع می شود. این اندازه کل آرایه نویز را نشان می دهد.

سپس، دو آرایه تصادفی `u1` و `u2` با استفاده از `np.random.random(size)` تولید می شوند. این آرایه ها حاوی اعداد تصادفی هستند که به طور یکنواخت بین 0 و 1 توزیع شده اند.

سپس تبدیل Box-Muller (تبدیل Box-Muller) روشی است که برای تولید اعداد تصادفی با توزیع نرمال استاندارد ( $\text{میانگین} = 0$ ، انحراف استاندارد = 1) از اعداد تصادفی توزیع شده یکنواخت استفاده می شود. این بر اساس قضیه حد مرکزی است، که بیان می کند که مجموع تعداد زیادی از متغیرهای تصادفی مستقل و یکسان توزیع شده، تمايل به پیروی از توزیع نرمال دارند). به `u1` و `u2` اعمال می شود تا دو عدد تصادفی مستقل تولید کند که از توزیع نرمال استاندارد پیروی می کنند ( $\text{میانگین} = 0$ ، انحراف استاندارد = 1). برای این تبدیل طبق مراحل زیر پیش میریم:

```
z1 = np.sqrt(-2.0 * np.log(u1)) * np.cos(2.0 *  
                           np.pi * u2)  
z2 = np.sqrt(-2.0 * np.log(u1)) * np.sin(2.0 *  
                           np.pi * u2)
```

این مقادیر، `z1` و `z2`، اعداد تصادفی هستند که از توزیع نرمال استاندارد پیروی میکنند. سپس آرایه نویز با استفاده از `noise.reshape(shape)` برای مطابقت با شکل مورد نظر تغییر شکل می دهد. در نهایت، آرایه نویز تولید شده به عنوان خروجی تابع برگردانده می شود.

به طور خلاصه، تابع '`generate_gaussian_noise`' ابتدا با تولید اعداد تصادفی که از یک توزیع نرمال استاندارد با استفاده از تبدیل Box-Muller پیروی می کنند، نویز گوسی تولید می کند. سپس این مقادیر را برای مطابقت با میانگین و انحراف استاندارد مورد نظر مقیاس و تغییر می دهد.

تابع `svd_denoising(image, singular_values_fraction)`: این تابع حذف نویز را با استفاده از تجزیه مقدار منفرد (SVD) انجام می دهد. یک تصویر و کسری از مقادیر مفرد را به عنوان ورودی نگه می دارد. این تابع SVD را برای هر کanal رنگی تصویر اعمال می کند، کسری از مقادیر مفرد را صفر می کند و کanal حذف شده را بازسازی می کند. در نهایت، کanal های حذف شده را ترکیب می کند تا تصویر حذف شده را تشکیل دهد که برگردانده می شود.

تعريف تابع svd\_channel\_denoising(channel, singular\_values\_fraction): این تابع حذف نویز را در یک کانال رنگی با استفاده از SVD انجام می‌دهد. برای نگه داشتن یک کانال (ماتریس) و کسری از مقادیر منفرد به عنوان ورودی نیاز است. این تابع SVD کانال را محاسبه می‌کند، کسری از مقادیر منفرد را صفر می‌کند و کانال حذف شده را با استفاده از مقادیر منفرد اصلاح شده بازسازی می‌کند. کانال حذف شده برگردانده می‌شود.

درتابع اصلی برنامه با بارگذاری یک تصویر درون حلقه و تبدیل آن به آرایه NumPy شروع می‌شود. سپس با استفاده از میانگین و انحراف استاندارد تولید شده به طور تصادفی نویز گوسی را به تصویر اضافه می‌کند. در مرحله بعد، فرآیند حذف نویز با اعمال SVD برای هر کانال رنگی تصویر نویز آغاز می‌شود. کسری از مقادیر منفرد (که با  $K$  تعیین می‌شوند) روی صفر تنظیم می‌شوند و کانال‌های حذف شده بازسازی می‌شوند. در نهایت، تصویر اصلی، تصویر نویزدار و تصویر حذف شده با استفاده از Matplotlib نمایش داده می‌شود.

اثر  $K$  (عدد نگه داشته شده از بردارهای منفرد) بر نتیجه حذف نویز:

تجزیه ارزش منفرد (SVD) یک تکنیک فاکتور سازی ماتریس است که یک ماتریس را به سه جزء تجزیه می‌کند:  $\Sigma$ ,  $U$  و  $V$ . که در آن  $U$  و  $V$  ماتریس‌های معتمد هستند، و  $\Sigma$  یک ماتریس مورب حاوی مقادیر منفرد ماتریس اصلی است. مقادیر منفرد نشان‌دهنده اهمیت یا سهم هر بردار منفرد در ماتریس است.

هنگام انجام نویز زدایی با استفاده از SVD، تنظیم کسری از مقادیر منفرد بر روی صفر به طور موثر سهم آن بردارهای منفرد را در بازسازی تصویر کاهش می‌دهد و منجر به حذف نویز می‌شود. کسری « $K$ » تعداد مقادیر مفرد را که باید حفظ شوند، به عنوان نسبتی از تعداد کل مقادیر مفرد تعیین می‌کند.

با تغییر مقدار  $K$ ، می‌توانیم مبادله بین حذف نویز و حفظ جزئیات تصویر را کنترل کنیم.

1.  $K$  کوچکتر (مثلًا 0.1 یا 0.01): وقتی  $K$  کوچک است، تعداد قابل توجهی از مقادیر منحصر به فرد روی صفر تنظیم می‌شوند که منجر به اثر حذف نویز قوی‌تر می‌شود. با این حال، این می‌تواند منجر به از دست دادن جزئیات مهم تصویر شود و باعث شود که تصویر حذف شده تار یا مخدوش تر به نظر برسد.

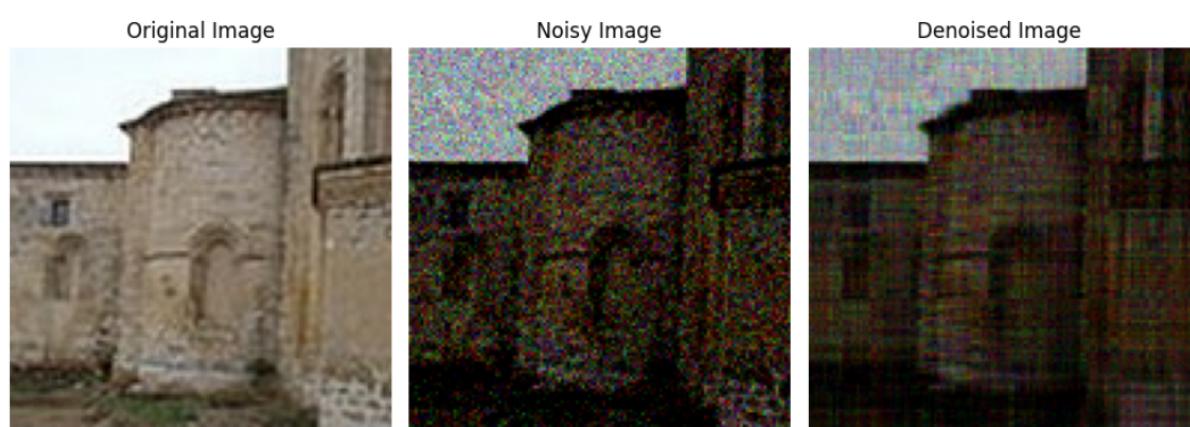
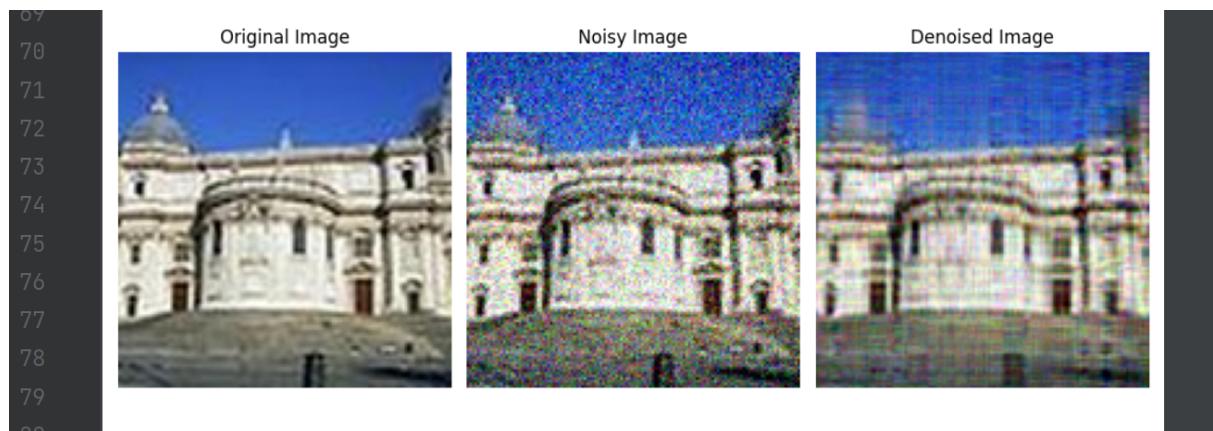
2.  $K$  بزرگتر (مثلًا 0.5 یا 0.9): وقتی  $K$  بزرگ است، مقادیر منحصر به فرد بیشتری حفظ می‌شوند که امکان حفظ بیشتر جزئیات تصویر را فراهم می‌کند. این منجر به یک تصویر حذف شده می‌شود که بسیار شبیه تصویر اصلی است اما مقداری نویز باقی‌مانده هنوز وجود دارد.

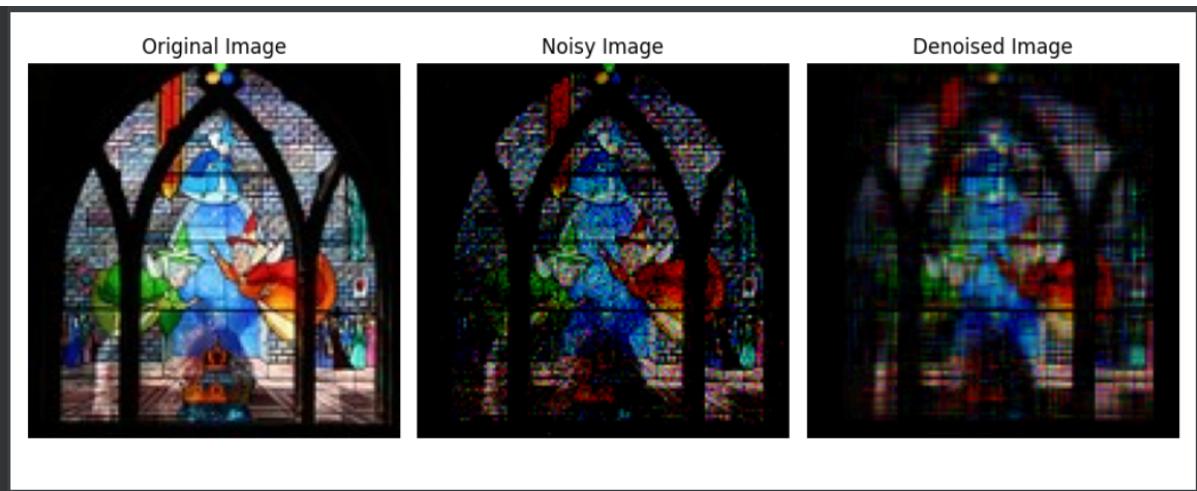
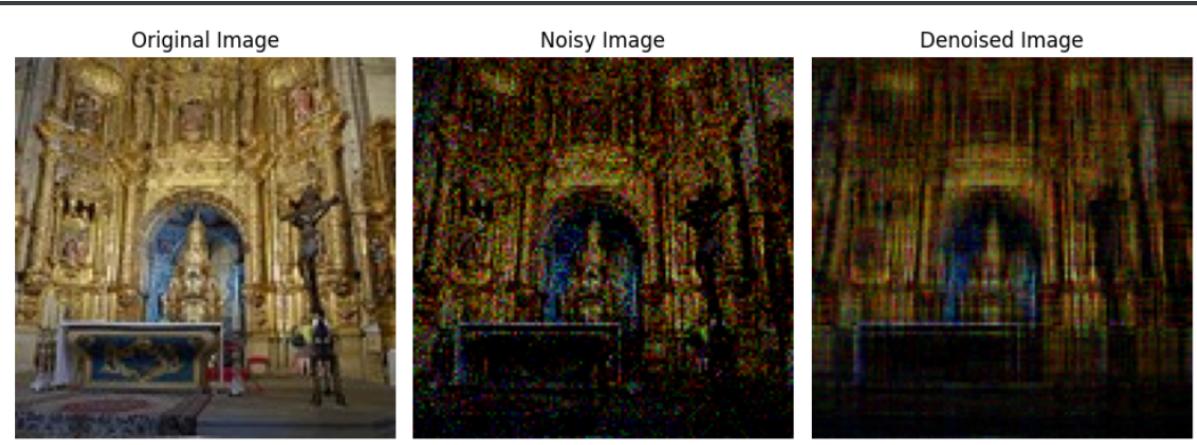
اغلب از طریق آزمایش و بازرسی بصری برای دستیابی به تعادل بین کاهش نویز و حفظ کیفیت تصویر تعیین می‌شود.

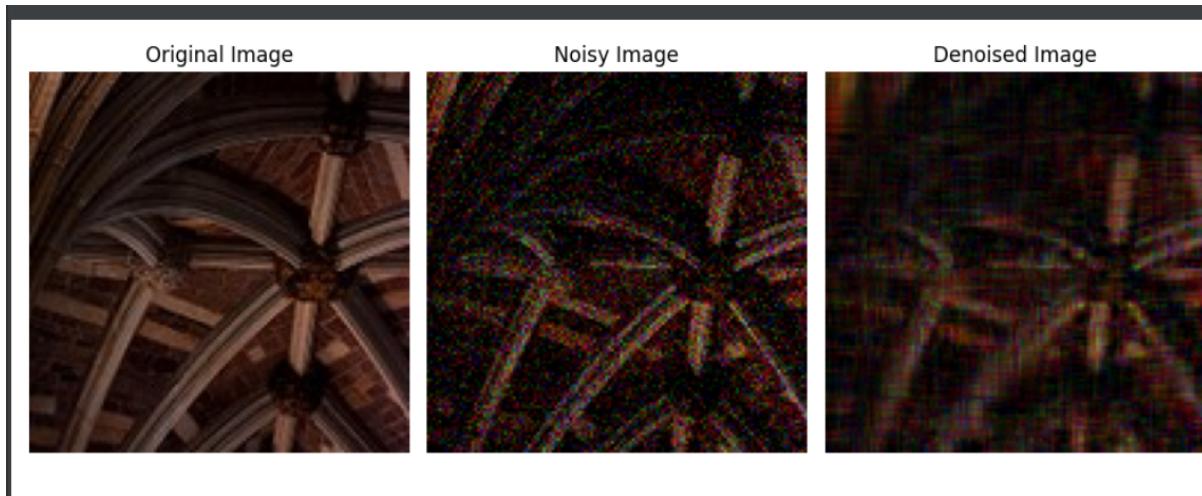
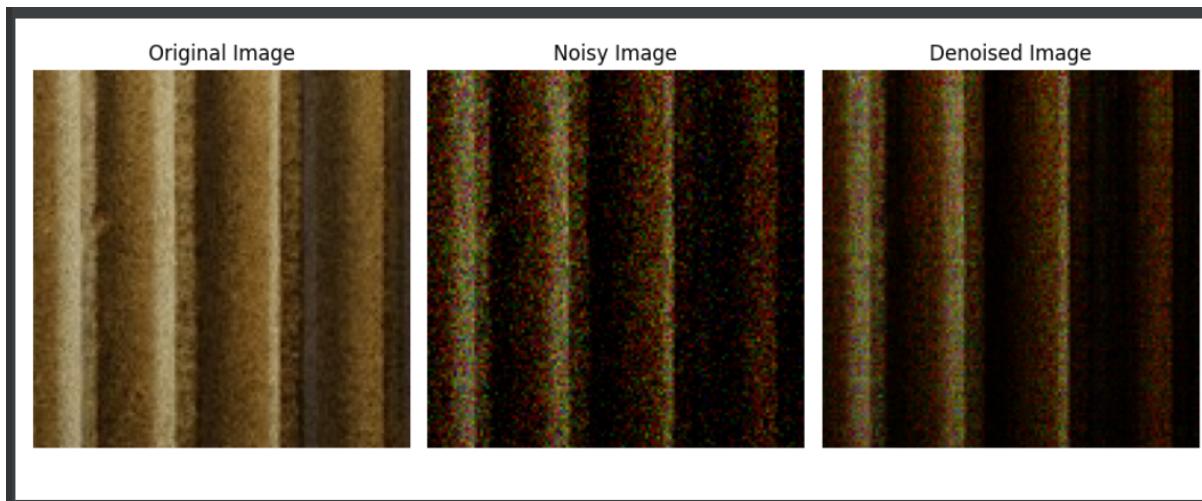
توضیحات حذف نویز تصاویر به کمک این الگوریتم:

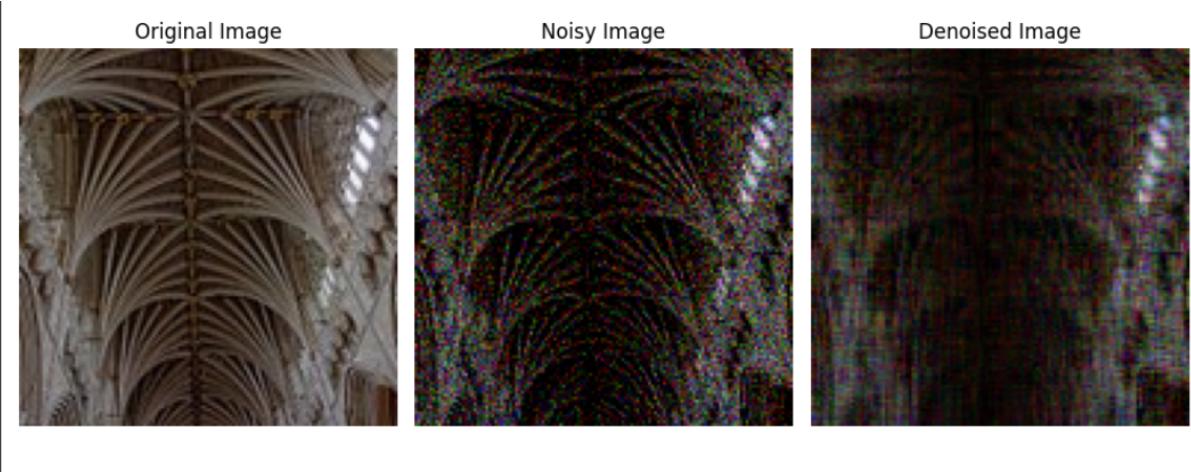
تابع svd\_denoising برای حذف نویز تصویر فرآخوانی می‌شود. این تابع هر کانال رنگی (قرمز، سبز و آبی) تصویر را به طور جداگانه حذف می‌کند. برای هر کانال رنگی، تابع svd\_channel\_denoising فرآخوانی می‌شود. SVD را در کانال انجام می‌دهد و تنها بخشی از مقادیر منحصر به فرد را که توسط پارامتر singular\_values\_fraction تعیین می‌شود، نگه می‌دارد. مقادیر تکی باقیمانده روی صفر تنظیم می‌شوند و به طور موثر نویز در کانال را کاهش میدهند. کانال‌های حذف شده برای اطمینان از اینکه مقادیر آنها در محدوده -1 تا 1 باقی می‌مانند، بریده می‌شوند. کانال‌های حذف شده با هم ترکیب می‌شوند تا تصویر حذف شده نهایی را با استفاده از np.vstack تشکیل دهند. با تنظیم پارامتر Singular\_values\_fraction در تابع svd\_denoising، می‌توانیم میزان حذف نویز اعمال شده روی تصویر را کنترل کنیم.

k=0.1:









$k = 0.6$

