گزارش پروژه پایانی

مبانی و کاربردهای جبر خطی

دکتر پیمان ادیبی

آدرینا ابراهیمی

۹۹۳۶۲۳۰۰۲

لینک کد در کولب

خرداد ۱۴۰۲

مقدمه

تصاویر اغلب در طول فرایند دریافت یا انتقال توسط نویز خراب می‌شوند. نویز می‌تواند کیفیت تصویر را کاهش داده و استخراج اطلاعات مفید را دشوار کند. به بیان واضح‌تر، فرایند حذف نویز از تصویر با حفظ ویژگی‌های اصلی مهم است. در این پروژه از SVD (Singular Value Decomposition) برای حذف نویز از تصاویر استفاده می‌کنیم. SVD یک تکنینک فاکتورسازی ماتریس است که می‌تواند برای تجزیه یک ماتریس به مقادیر تکی آن، بردارهای منفرد چپ و بردارهای منفرد راست استفاده شود. با کوتاه کردن مقادیر تکی، می‌توانینم نویز را از تصویر حذف کنیم و ویژگی‌های مهم را حفظ کنیم.

شرح مسئله

در این پروژه ابتدا، روی تصاویری که از دیتاست Google scraped Image Dataset در فولدر Architecture به صورت رندوم انتخاب شده‌اند، یک نویز گاوسی می‌دهیم. سپس، با استفاده از SVD عملیات نویززدایی را انجام می‌دهیم.

گزارش کار الگوریتم

ابتدا به کمک Kaggle و API آن، دیتاست ذکر شده را دانلود کرده و از حالت زیپ آن را خارج کرده و عملیات مورد نظر را روی تصاویر آن اعمال می‌کنیم.

تابع random_gaussian:

در این تابع، یک عدد تصادفی از یک توزیع گاوسی تولید می‌شود به این صورت که ابتدا دو عدد تصادفی u1 و u2 را که بین صفر و یک هستند تولید می‌کنیم. سپس، از تبدیل باکس مولر با فرمول زیر برای تولید عدد تصادفی در یک توزیع نرمال استاندارد استفاده می‌کنیم.

$$z = \sqrt{(-2 * \ln(u1))} * \cos{(2 * \pi * u2)}$$

سپس این عدد تصادفی تولید شده، توسط انحراف معیار مقیاس شده و میانگین به آن اضافه می‌شود تا عدد تصادفی با میانگین و انحراف معبیار مشخص به دست آید.

تابع add_gaussian_noise:

در این تابع، عمل اضافه کردن نویز گاوسی انجام می‌شود. ابتدا طول و عرض تصویر به دست آورده شده و پس از آن، برای هر پیکسل تصویر مقادیر RGB آن را در یک ماتریس از سه‌تایی ذخیره می‌کند. سپس، با پیمایش روی تک‌تک پیکسل‌های تصویر یک عدد تصادفی (روی توزیع گاوسی) مجزا برای هرکدام از RGBهای هر پیکسل به دست آورده و آن را به سه مولفه پیکسل اضافه می‌کند (فرمول Additive Noise Model) و مقادیر را پس از اصلاح به بازه‌ها در صورت نیاز در هر پیکسل تصویر ذخیره کرده و در نهایت تصویر جدید را برمی‌گرداند.

$$w(x,y) = s(x,y) + n(x,y)$$

w مقدار نهایی هر پیکسل، s مقدار اصلی پیکسل، n میزان نویز

در اضافه کردن نویز، هر چه انحراف معیار تابع گاوسی بیشتر باشد، میزان نویز بیشتر خواهد بود و هر چه میانگین بالاتر باشد، شدت روشنایی تصویر بیشتر خواهد بود.

تابع power_iteration:

این تابع، بزرگ‌ترین مقدار ویژه و بردار ویژه متناظر با آن را از روش power iteration محاسبه می‌کند. ابتدا، یک بردار اولیه nتایی با مقادیر تصادفی ساخته سپس، برای تعداد تکرارهای مشخص شده تا همگرایی این بردار، ضرب داخلی ماتریس در بردار را محاسبه کرده و پس از آن نرم را محاسبه می‌کنیم. بعد از آن بردار را تقسیم بر نرمش کرده تا نرمال شود و مقدار بردار را آپدیت می‌کنیم. در نهایت بزگترین مقدار ویژه و بردار ویژه منتاظر با آن را با توجه به فرمول زیر حساب کرده و برمی‌گردانیم.

$$b_k = \lambda b_k \rightarrow A b_k {b_k}^T = \lambda b_k {b_k}^T \rightarrow \lambda = \frac{A b_k {b_k}^T}{b_k {b_k}^T}$$

تابع compute_eigen_values_vectors:

در این تابع، تمامی بردارهای ویژه و مقادیر ویژه یک ماتریس را با استفاده از روش deflation محاسبه می‌کند. ابتدا بعد ماتریس ورودی را دریافت کرده، سپس، یک بردار nتایی برای ذخیره مقادیر ویژه و یک ماتریس n در n برای ذخیره بردارهای ویژه را مقداردهی اولیه می‌کنیم. بعد از آن، در یک حلقه به طول n، با استفاده از تابع قبلی بزرگ‌ترین مقدار ویژه و بردار ویژه را محاسبه کرده و در بردار مقادیر ویژه اصلی و ماتریس مقادیر ویژه اصلی این مقدارهای بازگشتی را ذخیره می‌کنیم. در انتهای حلقه نیز، ماتریس را با استفاده از ماتریس رتبه یک تشکیل شده (حاصل ضرب بیرونی بردار ویژه) و خودش کاهش می‌دهیم تا بتوان سایر مقادیر و بردارهای ویژه را به دست آورد. در نهایت نیز، مقادیر ویژه و بردارهای ویژه را برمی‌گردانیم.

$$A' = A - \lambda(VV^T)$$

تابع SVD:

این تابع با ورودی گرفتن یک ماتریس، ابتدا مقادیر و بردارهای ویژه ماتریس $matrix^T matrix$ را محاسبه کرده، سپس، مقادیر ویژه را به صورت نزولی مرتب کرده و بردارهای ویژه منتاظر با آن‌ها را هم مرتب می‌کنیم. در نهایت، با توجه به فرمول تجزیه SVD مقادیر U، S و V^T را برمی‌گردانیم.

تابع svd_denoising_per_channel:

در این تابع یک ماتریس که یک کانال رنگی از یک عکس است را به همراه تعداد مقادیر منفرد دریافت کرده و پس از محاسبه SVD ماتریس به تعداد مقادیر منفرد آن را نگه می‌داریم. حال با ضرب USV به ماتریس نویززدایی شده از آن کانال رنگی می‌رسیم. و پس از اطمینان حاصل کردن از متعبر بودن مقادیر ماتریس آن را به شکل تصویر آن کانال رنگی برمی‌گردانیم.

$$matrix_{m*n} = U_{m*r} S_{r*r} V^T_{r*n}$$

تابع svd_denoising:

در این تابع، پس از تبدیل عکس به آرایه‌ای سه بعدی، سه کانال R، G و B آن جدا کرده و به صورت جداگانه، آن‌ها را برای نویززدایی به تابع قبلی می‌فرستد و در نهایت حاصل هر سه کانال را با هم تجمیع کرده و تصویر نهایی نویززدایی شده را برمی‌گرداند.

تابع get_concat_h:

در این تابع، یک تصویر که حاصل سه تصویر اصلی، نویزدار و نویززدایی شده است را برمی‌گرداند.

تاثیر پارامترهای نویز گاوسی

تاثیر میانگین:

در اضافه کردن نویز به تصویر، در تابع گاوسی هر چه میانگین بالاتر باشد (چه به سمت مثبت چه به سمت منفی)، مشاهده می‌شود شدت روشنایی تصویر بیشتر خواهد بود. سایر پارامترها ثابت است.

- میانگین ۲۰-

- میانگین ۰

- میانگین ۴۰

- میانگین ۶۰

تاثیر انحراف معیار:

در اضافه کردن نویز به تصویر، در تابع گاوسی هر چه انحراف معیار تابع گاوسی بیشتر باشد، مشاده می‌شود میزان نویز بیشتر خواهد بود. سایر پارامترها ثابت است.

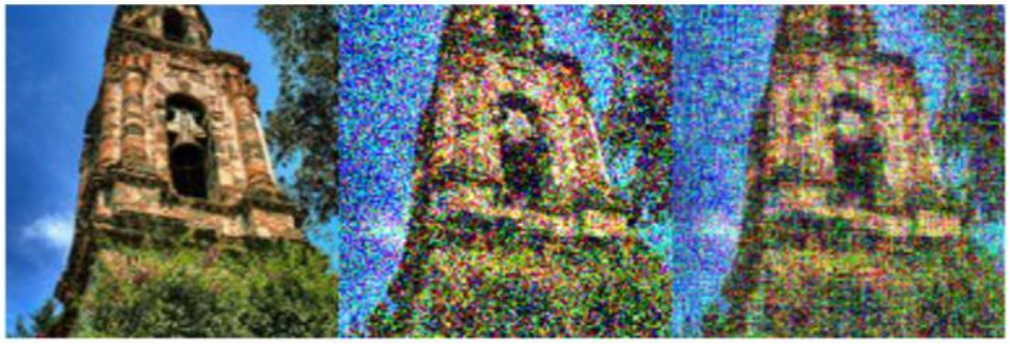
- انحراف معیار ۰

- انحراف معیار ۱۰

- انحراف معیار ۲۰

- انحراف معیار ۴۰

- انحراف معیار ۸۰



تاثیر عدد نگه داشته شده از بردارهای منفرد

در محاسبه SVD تاثیر تعدادهای بردارهای منفرد به این صورت است که هر چه تعدادها آن‌ها کمتر شود نویز بیشتری از تصویر گرفته می‌شود اما از جزئیات تصویر کاسته می‌شود. و هر چه تعداد آن‌ها بیشتر باشد، نویز کمتری از تصویر گرفته می‌شود اما جزئیات بیشتری از تصویر داریم.

- تعداد بردارهای منفرد ۵

- تعداد بردارهای منفرد ۱۰

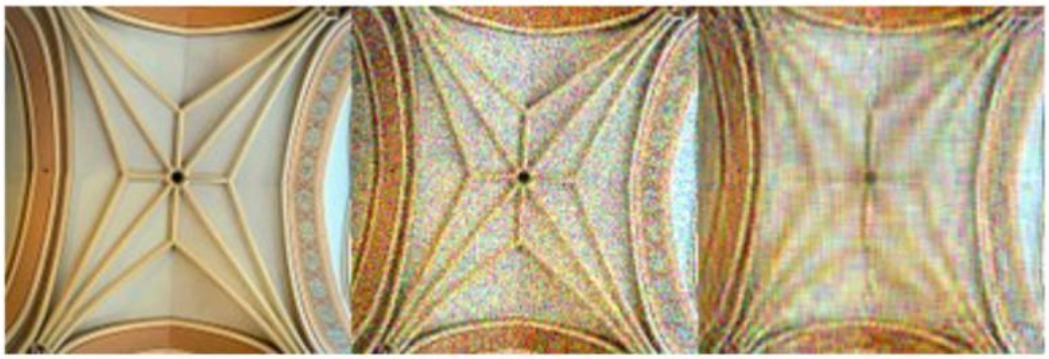
- تعداد بردارهای منفرد ۲۰

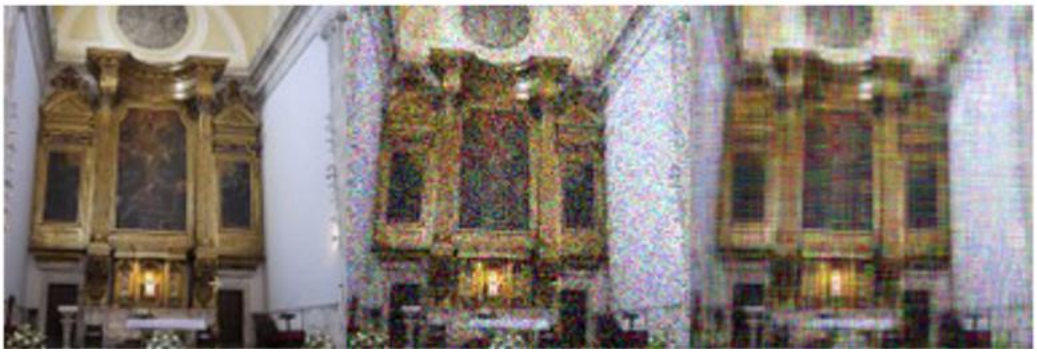
- تعداد بردارهای منفرد ۳۰

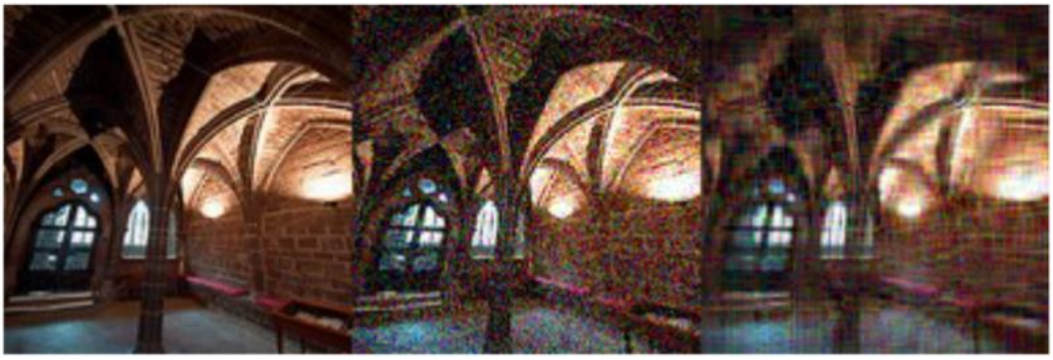
- تعداد بردارهای منفرد ۵۰

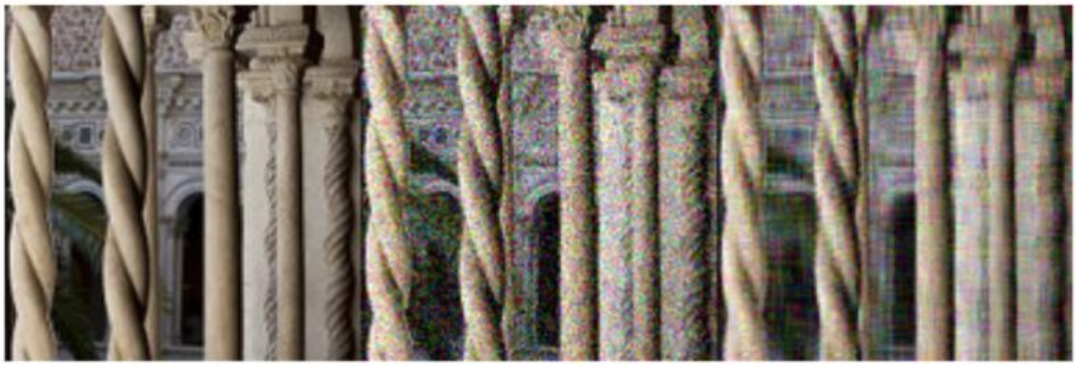
ده نمونه از عملکرد اضافه کردن نویز و نویززدایی

- میانگین نویز گاوسی ۱۰
- واریاس نویز گاوسی ۲۰
- تعداد مقادیر منفرد ۱۵









کتابخانه‌های استفاده شده

- کتاب‌خانه numpy: از توابع این کتاب‌خانه جهت انجام عملیات ریاضی استفاده شده است.
- کتاب‌خانه Pillow: از توابع این کتاب‌خانه جهت خواندن، نوشتن و نمایش دادن عکس‌ها استفاده شده است.
- کتاب‌خانه os: توابع این کتابخانه جهت کاربا فایل‌ها استفاده شده است.

منابع ایده

از منبع زیر برای آشنایی با نحوه پیاده‌سازی و الگو گرفتن اولیه استفاده شده است.

اسلایدهای آموزشی درس

تبدیل باکس-مولر - ویکی‌پدیا، دانشنامهٔ آزاد (wikipedia.org)

Gaussian Noise | Hasty.ai

Andy Jones (andrewcharlesjones.github.io)

Concatenate images with Python, Pillow | note.nkmk.me