به نام خدا

Table of Contents

[**سوال سوم** 3](#_Toc196479289)

[**بخش الف** 3](#_Toc196479290)

[**انواع نویز** 3](#_Toc196479291)

[**گزارش کار** 4](#_Toc196479292)

# **سوال سوم**

برای این سوال 3 فایل notebook پیاده‌سازی شده است که نتایج خروجی آنها هم به صورت فایل تصویر در همین پوشه قرار دارد (تحت عنوان PART?\_output) هم در خود فایل ها قرار دارد و هم در قسمت گزارش کار وجود دارد. توضیحات مربوط به کد در قسمت گزارش کار بخش مرتبط با آن قرار داده شده است. در هر بخش توضیحات مورد نیاز نیز داده شده است.

## **بخش الف**

در این قسمت ما از یک تصویر grayscale به عنوان ورودی برای اعمال نویزهای مختلف ورود کردیم. برای محاسبه یک تصویر grayscale از روش Luminosity استفاده کردیم که یک جمع وزن دار از کانال‌های RGB است و بازتاب دهنده حساسیت چشم انسان است.

در این پروژه ما 3 نوع نویز را پیاده‌سازی کردیم که در ادامه هر کدام را توضیح مختصر میدهم.

### **انواع نویز**

#### **نویز Gaussian**

نویز گوسی یک نویز آماری است که در آن تغییراتی که به هر پیکسل اضافه میشود از یک توزیع گوسی پیروی میکند که مانند bell curve است. به این نویز معمولا نویز گوسی سفید افزایشی نیز گفته میشود. افزایشی به این معنا است که مقدار نویز به مقدار اولیه پیکسل‌ها اضافه میشود. سفید یعنی اینکه نویز مستقل هست بین پیکسل‌های مختلف و توان آن در تمام فرکانس‌های مختلف برابر است. مقدار نویز در این روش از یک توزیع گوسی به صورت تصادفی نمونه برداری میشود. میانگین معمولا 0 است. و با پارامتر سیگما شدت نویز را کنترل میکنیم هر چه پارامتر سیگما بالاتر باشد نویز قوی تر است که همه این موارد در پیاده‌سازی نیز قابل مشاهده است.

#### **نویز فلفل و نمک**

این نویز یک نویز ضربه‌ای است که به جای اینکه یک مقدار تصادفی را به مقادیر پیکسل اضافه کنیم، مقدار پیکسل اولیه را یا با کمینه شدت ممکن ( 0 که سیاه است و معنی فلفل را میدهد) و یا بیشینه شدت ممکن (255 که همان سفید است و معنی نمک را میدهد) جایگزین میکنیم. در این روش فقط درصد مشخصی از پیکسل‌ها تحت تاثیر قرار میگیرند. همانطور که خروجی مشاهده خواهد شد شبیه این است که یک سری نقاط سفید و سیاه به صورت تصادفی در سراسر عکس پخش شده اند شبیه دانه‌های نمک و فلفل.

برای انتخاب تعدادی از پیکسل‌ها ما یک احتمالی را در نظر میگیریم به این صورت که اگر اون پیکسل انتخاب شده احتمال سیاه شدن برابر با تعداد آنها ضربدر 1 منهای تعداد نمک و فلفل ها است و همین حالت برای احتمال سفید شدن و اگر هیچکدوم از این 2 نشد مقدار آن پیکسل دست نمیخورد. به صورت دقیق تر:

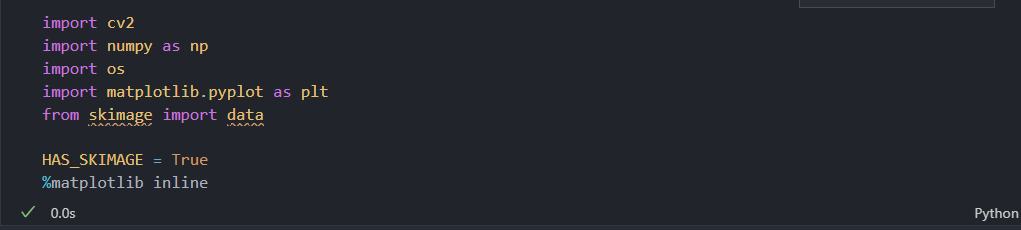
* Noisy\_Pixel = 0 (Pepper) with probability amount \* (1 - salt\_vs\_pepper)
* Noisy\_Pixel = 255 (Salt) with probability amount \* salt\_vs\_pepper
* Noisy\_Pixel = Original\_Pixel otherwise.

#### **نویز speckle**

این نویز ضربی است به جای افزایشی بودن به این معنی که مقدار پیکسل‌ها به نوعی مقیاس میشود (scale میشود). اغلب شبیه یک بافت دانه‌دانه است که روی تصویر قرار گرفته که نواحی روشن‌تر در تصویر اصلی، معمولا تغییرات نویز شدیدتری نسبت به نواحی تاریک تر دارند. نکته‌ای که مطرح است اینکه مقدار نویز در اینجا نیز یک مقدار متغیر تصادفی با میانگین صفر است مانند توزیع گوسی و تنها فرق آن ضرب شدن نویز در مقدار پیکسل اولیه است.

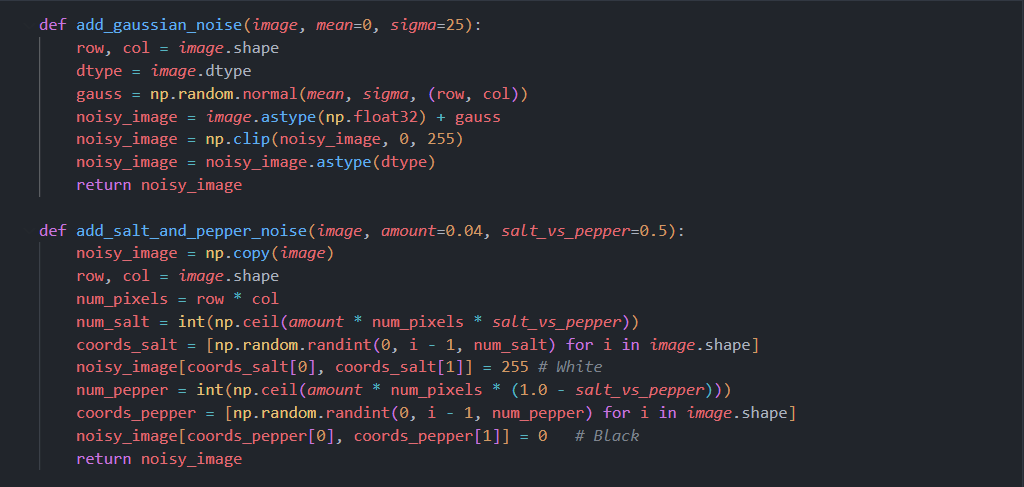
### **گزارش کار**

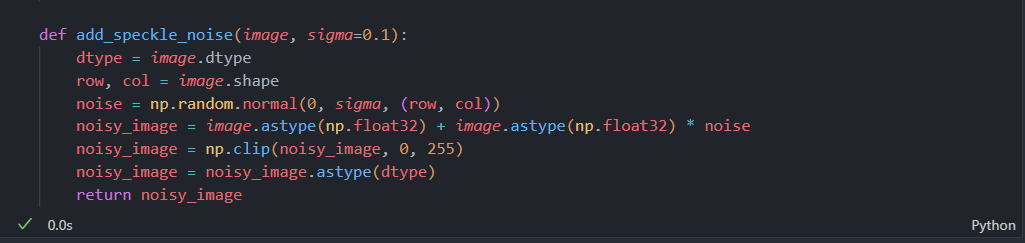
در این قسمت کد پیاده‌سازی را بخش به بخش توضیح میدهیم:



از کتابخانه cv2 برای اعمال عملیات ها مختلف مانند خواندن عکس‌ها استفاده میکنیم.

برای نشان دادن خروجی از matplotlib استفاده کرده‌ایم. همچنین تصاویر مورد نیاز را از scikit-image تحت عنوان skimage بارگذاری کرده‌ایم. که در این پروژه از تصویر معیار cameraman استفاده کردیم. خط آخر هم صرفا برای این است که تمام خروجی به صورت inline داده شود.

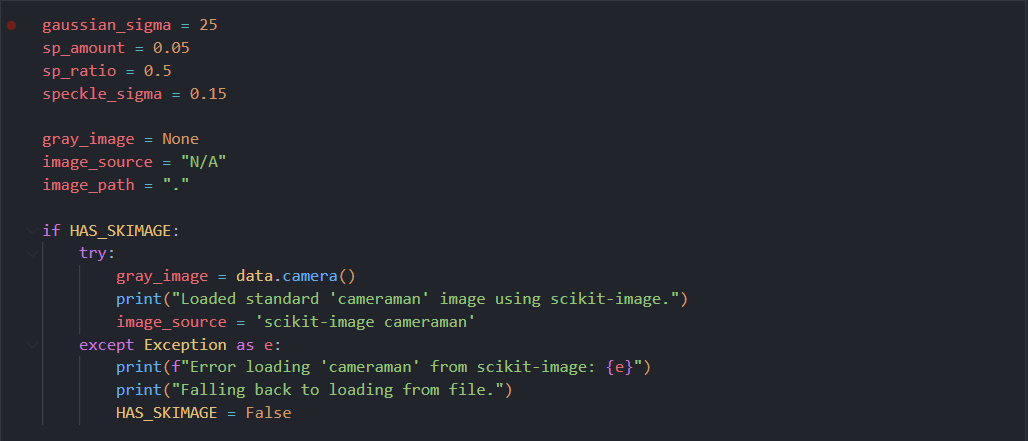


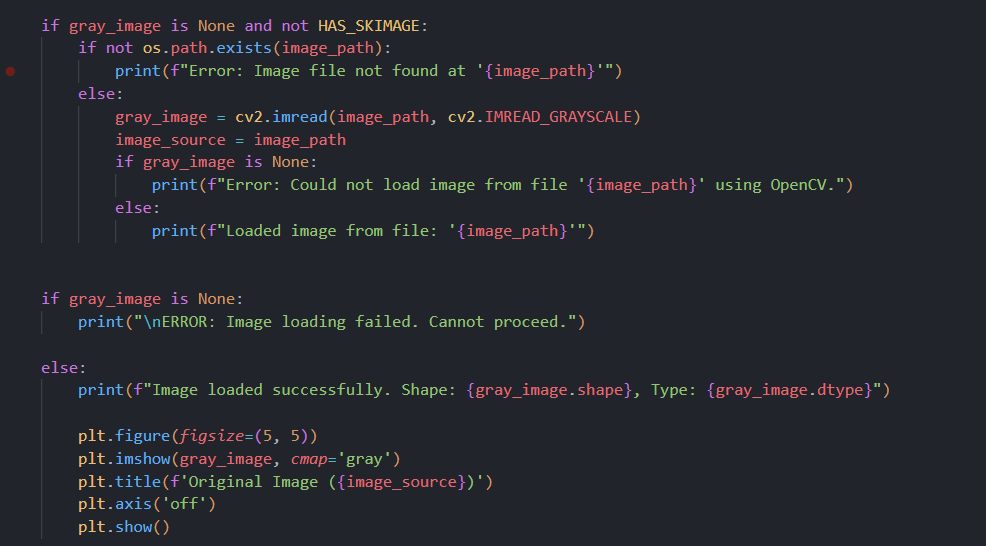


3 تابع مختلف را برای 3 نویزی که قبل توضیح دادیم پیاده‌سازی کردیم، در تابع gaussian ابتدا مطمئن شدیم اعداد float هستند همراه با dtype اولیه بعد از آن با کمک noisy\_image به تصویر نویز اضافه کردیم. در مرحله بعدی مقادیر را بین 0 تا 255 محدود کردیم با کمک clip و در نهایت تصویر را به dtype اصلی برگرداندیم.

در تابع salt\_and\_pepper 3 تا آرگومان داریم. آرگومان اول که تصویر grayscale است. آرگومان دوم درصدی از پیکسل ها هستند که قرار است با نویز جایگزین شوند(amount). آرگومان سوم درصدی از نویزها هستند که قرار است یا سفید شوند و یا سیاه شوند همانطور که در بالا توضیح دادیم(salt\_vs\_pepper). و در نهایت خروجی نویزی شده را برگرداندیم. در ادامه همین تابع یک سری مختصات تصادفی برای نویزهای سفید و سیاه تولید کردیم و نویزهای سفید و سیاه را به تصویر اضافه کردیم.

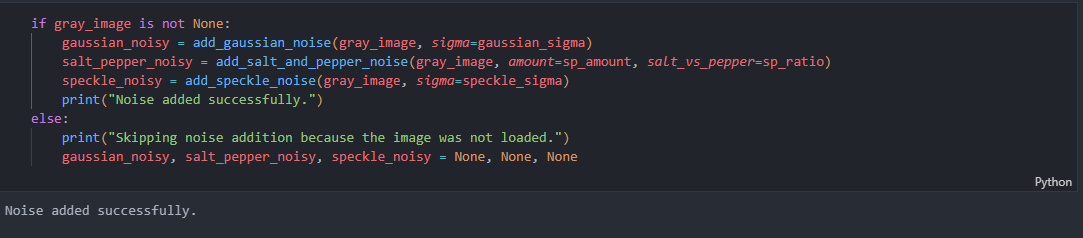
در تابع speckle\_noise نیز همانطور که توضیح دادیم قرار است مقادیر پیکسل را در مقدار نویز ضرب کنیم. چون قرار است به یک توزیع نیاز داشته باشیم پس از پارامتر سیگما در آرگومان استفاده کرده‌ایم در این قسمت از توزیع گوسی استفاده کردیم.

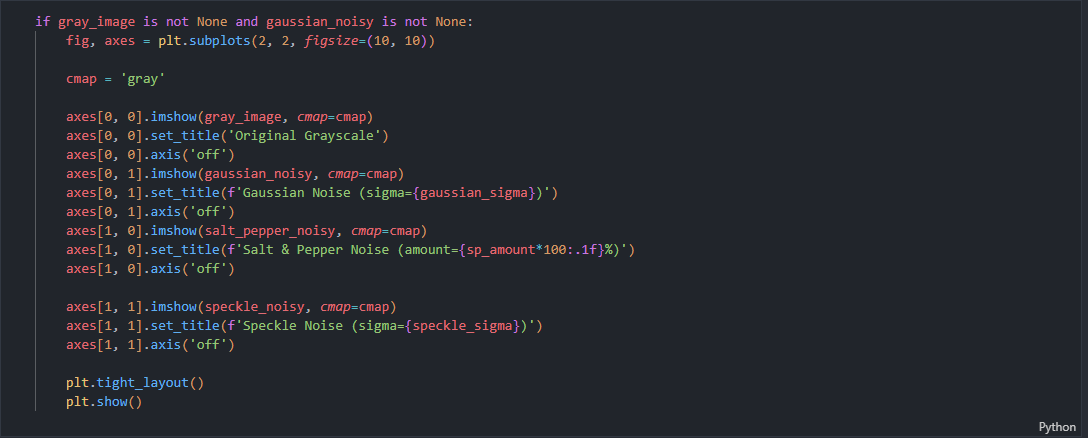




در این قسمت ابتدا پارامترهای نویز را تنظیم کردیم برای هر کدام از 3 تابعی داریم. Gaussian\_sigma برای میزان شدت نویز در تابع گوسی، sp\_amount و sp\_ration برای تابع salt and pepper است که میزان پیکسل‌های که قرار است توسط این تابع تحت تاثیر قرار بگیرند را مشخص میکنیم و نسبت سیاه به سفید را مشخص میکنیم. متغیر speckle\_sigma نیز برای شدت نویز تابع speckle است. بعد در ادامه از کتابخانه scikit-image کمک میگیریم و تصویر cameraman را بارگذاری میکنیم و بررسی میکنیم عکس به درستی در دسترس قرار گرفته باشد و در نهایت خروجی اولیه را که عکس اصلی cameraman هست به درستی دریافت کرده و نمایش میدهیم:

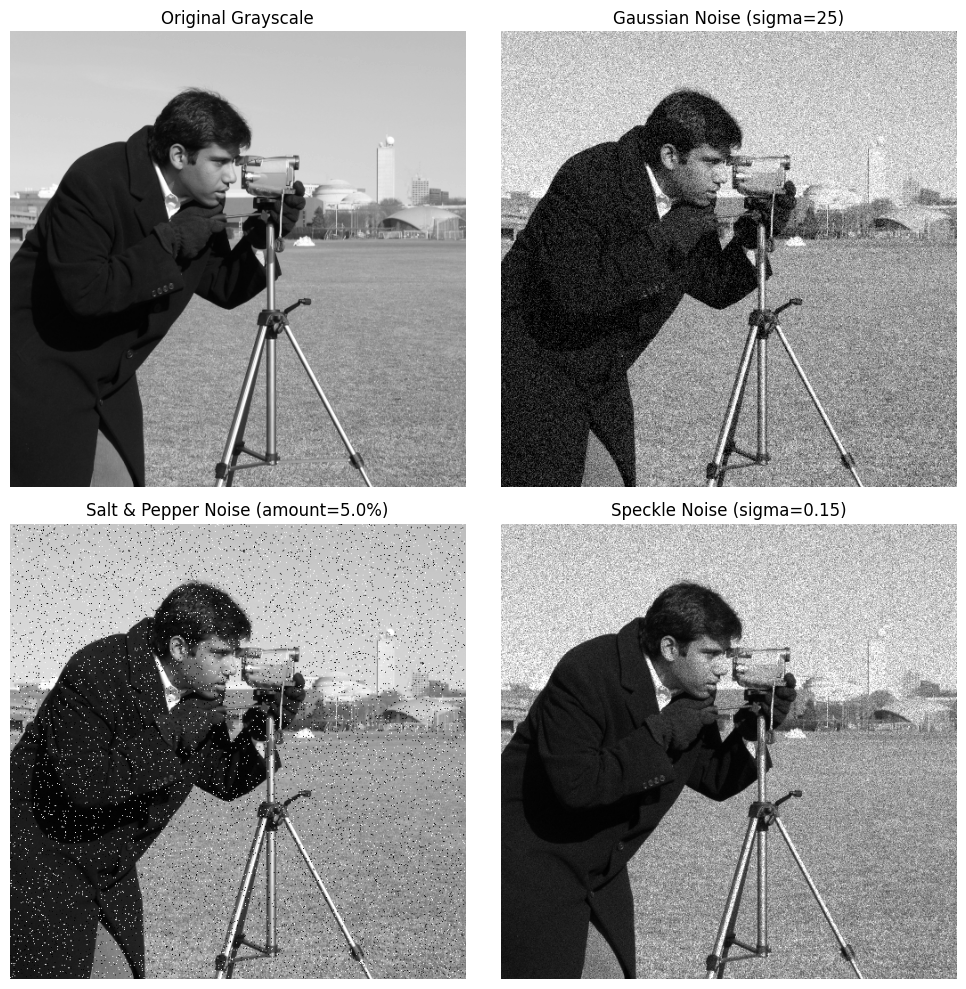


همانطور که در تصویر بالا مشخص است نویز را بر روی تصویر دریافت شده اعمال کردیم.



در نهایت خروجی نویزی شده را به نمایش میگذاریم.

خروجی:



بخش ب

س