Из предыдущего скрипта мы забираем цикл, который ищет jpg изображения в папке

while True:

## get the directory listing

jpegfiles = [name for name in os.listdir(path)

if name.endswith('.jpg')]

а также способ задания пути к папке с помощью argv.

files\_path = argv[1]

os.chdir(files\_path)

Шаг 1:

В папке с фото нужно добавить на каждую фотографию 15% шума вида Salt And Pepper (соль и перец)

Это можно сделать с помощью библиотеки Skimage и функции util.random\_noise().

Для установки skimage через pip необходимо выполнить(scikit-image это skimage):

*# Update pip*

python -m pip install -U pip

*# Install scikit-image*

python -m pip install -U scikit-image

Для проверки установленной версии можно воспользоваться:  
python -c "import skimage; print(skimage.\_\_version\_\_)"

Если возникнет проблема при установке попробуйте python3 вместо python.

---------------------------------------------------------------------------------------------------------------------------

Для добавления шума используется команда

util.random\_noise (image, mode=, seed=None, clip=True, \*\*kwargs)

image-входные данные

mode-позволяет выбрать тип шума:

gaussian-гауссовский шум

localvar

poisson

salt-соль

pepper-перец

s&p-соль и перец-он то нам и нужен

speckle

seed-зернистость, параметр можно оставить на None

clip-можно оставить на True

\*\*kwargs-имеет несколько параметров:

mean-используется в gaussian и speckle

var--используется в gaussian и speckle

local\_vars-используется в localvar

amount-позволяет задать коэффициент шума, используется в salt, pepper и s&p

salt\_vs\_pepper-позволяет задать коэффициент шума соли, чем больше значение, тем больше соли, используется в salt, pepper и s&p

Далее нам дать этой функции входные данные, а точнее изображение.

После того как мы нашли все jpg файлы в папке нам нужно их открыть, так сказать загрузить изображение из файла.

Это можно сделать с помощью io.imread, она тоже находится в skimage

Также необходимо сделать из нашего изображения массив с помощью np.array

from skimage import io

import numpy as np

После этого уже можно приступать к наложению шумов

for art in jpegfiles:

img = io.imread (art)

img = np.array (img)

img = skimage.util.random\_noise(img,mode="s&p",seed=None, clip=True, amount = 0.15)

io.imsave('test.jpeg')

break

Наш цикл работает и функционирует, но постоянно пересоздаёт test.jpeg и в нём всегда будет находится последнее изображение из списка.

Шаг 2: Сделать так, чтобы сохранялось и исходное изображение и копия с шумом

Можно воспользоваться enumerate - это встроенная функция в python, используемая для перебора с автоматической индексацией.

Пример:

for count, item in enumerate(\_\_name\_\_, 100):

print(count, item)

Таким образом наш код будет выглядеть так:

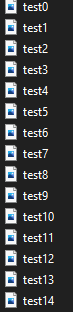
for count, art in enumerate(jpegfiles):

img = io.imread (art)

img = np.array (img)

img = skimage.util.random\_noise(img,mode="s&p",seed=None, clip=True, amount = 0.15)

io.imsave('test' + str(count) + '.jpeg', img)

break

Результат:

Шаг 3: Сделать так, чтобы можно было делать несколько копий(например, с 10 и 15 % шума)

Для создания нескольких копий с разным количеством шума, можно создать свою функцию и переместить туда util.random\_noise и при вызове этой функции указать % шума списком

add\_noise(img, [0.2, 0.15, 0.25], name)

def add\_noise(img, coeff, name):

for s in coeff:

temp = skimage.util.random\_noise(img,mode="s&p",seed=None, clip=True, amount = s)

io.imsave(name + '\_noise' + str(s).replace('.', '') + '.jpg', temp)

Для того, чтобы util.random\_noise в функции работал нормально, необходимо передать в нее параметры

img-само изображение

coeff-список коэффициентов шума

name-оригинальное имя фото, чтобы копия находилась рядом с ним

str(s).replace('.', '')- мы добавляем коэффициент в имя файла, чтобы понять какое изображение имеет какой % шума. Изначально коэф, который мы даём как параметр идёт int значением, чтобы он мог попасть в имя файла мы его преобразуем в string с помощью str(s).replace(). Чтобы избавиться от точки в имени файла мы записываем в replace('.', '')

Итоговый код будет выглядеть примерно так:

