ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN


# BÁO CÁO ĐỒ ÁN 2

# IMAGE PROCESSING

## MÔN: TOÁN ỨNG DỤNG VÀ THỐNG KÊ – MTH00051

## LỚP: 19CLC2

**Giảng viên giảng dạy:**

**ThS. Vũ Quốc Hoàng**

**CN. Phan Thị Phương Uyên**

**CN. Nguyễn Văn Quang Huy**

**CN. Lê Phúc Lữ**

**Sinh viên thực hiện:**

***Phạm Hồng Quân – 19127250***

*Thành phố Hồ Chí Minh – Ngày 26/07/2021*

-------ଘ ଌ----

1. **Tiến độ yêu cầu:**

| Yêu cầu | Mức độ hoàn thành |
|---|---|
| 1. Thay đổi độ sáng cho ảnh | 100% |
| 2. Thay đổi độ tương phản | 100% |
| 3. Chuyển ảnh RGB thành ảnh xám | 100% |
| 4. Lật ảnh ngang – dọc | 100% |
| 5. Chồng hai ảnh cùng kích thước | 100% |
| 6. Làm mờ ảnh | 100% |

2. **Yêu cầu:**

   a. **Mở ảnh và đọc ảnh:**

   *Ý tưởng:* Sử dụng hàm Open() trong thư viện PIL để thực hiện việc mở ảnh. Sau đó, viết hàm **Read_Image_ToArr(image)** để chuyển ảnh thành mảng để xử lý bằng cách sử dụng **np.array()** trong thư viện numpy.

   b. **Thay đổi độ sáng cho ảnh:**

   *Ý tưởng:* Viết hàm **brightness(image, alpha)** (trong đó alpha là hệ số để chỉnh độ sáng) thực hiện thay đổi độ sáng của ảnh. Đầu tiên, chuyển ảnh thành mảng (**arrayImage)** sử dụng hàm ở câu a, sau đó tiến hành cộng vào mảng **arrayImage** một hệ số alpha được ép kiểu float (*Nếu như không ép kiểu float thì khi alpha là kiểu int, khi cộng vào mảng, lúc này các phần tử trong mảng là kiểu uint8_t, tức là chỉ nằm trong đoạn giá trị từ 0 đến 255, thì khi các phần tử trong mảng đã được cộng vào có giá trị lớn 255 thì sẽ quay về số 0 để thực hiện cộng tiếp. Thực hiện cộng với một float sẽ cho ta một số lớn hơn 255*)**.** Sử dụng hàm np.clip() trong thư viện numpy để giới hạn các phần tử của mảng nằm trong đoạn từ 0 đến 255, vì các giá trị màu trong mảng ảnh chỉ nằm trong đoạn giá trị từ 0 đến 255.

   c. **Thay đổi độ tương phản:**

   *Ý tưởng:* Viết hàm **contrast(image, alpha)** (trong đó alpha là hệ số để chỉnh độ tương phản) thực hiện thay đổi độ tương phản của ảnh. Đầu tiên, chuyển ảnh thành mảng (**arrayImage)** sử dụng hàm ở câu a, sau đó tiến hành tính tích mảng

**arrayImage** với một hệ số alpha được ép kiểu float (tương tự giải thích ở câu b). Sử dụng hàm **np.clip()** trong thư viện numpy để giới hạn các phần tử của mảng nằm trong đoạn từ 0 đến 255, vì các phần tử trong mảng ảnh chỉ nằm trong đoạn giá trị từ 0 đến 255.

d. **Chuyển ảnh RGB thành ảnh xám:**

*Ý tưởng:* Viết hàm **ConvertGray(image)** thực hiện các chức năng sau: đầu tiên là chuyển ảnh thành mảng bằng hàm **Read_Image_ToArr(image)** để xử lý. Tiếp đến, với mỗi pixel, ba màu R, G, B ứng với các hệ số 0.3, 0.59, 0.11. Mỗi pixel sẽ được tính = R*0.3 + G*0.59 + B*0.11:

Vì thế, ta dùng **np.dot** của thư viện numpy để tính tích của từng pixel (mỗi pixel gồm ba màu R, G, B) với mảng [0.3, 0.59, 0.11].

e. **Lật ảnh ngang:**

*Ý tưởng:* Chuyển ảnh thành mảng để xử lý. Sử dụng hàm **np.fliplr** thực hiện việc lật ảnh ngang (axis = 1, chiều từ trái sang phải).

f. **Lật ảnh dọc:**

*Ý tưởng:* Chuyển ảnh thành mảng để xử lý. Sử dụng hàm **np.flipud** thực hiện việc lật ảnh dọc (axis = 0, chiều từ trên xuống).

g. **Chồng 2 ảnh có cùng kích thước (ảnh xám):**

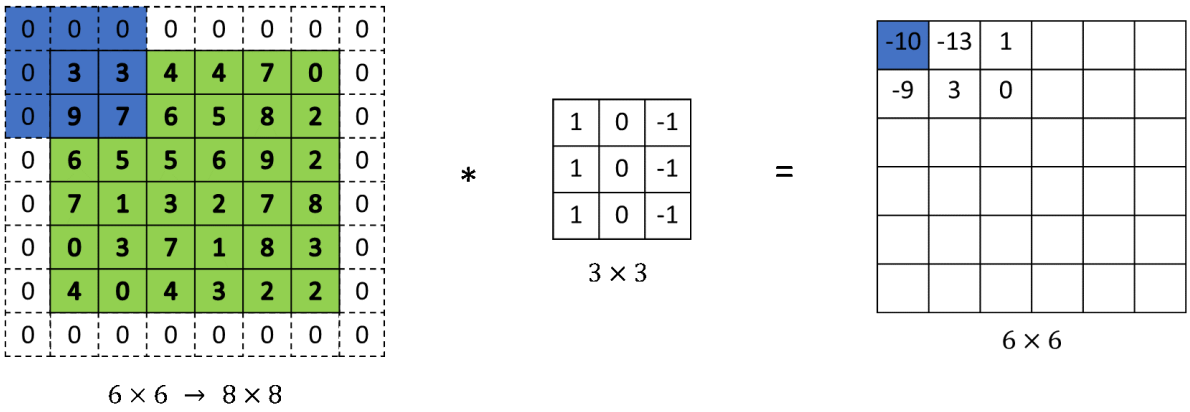
*Ý tưởng:* Đầu tiên, ta tạo một hình **Star.jpg** với kích thước bằng với hình đang xử lý và chuyển 2 hình cần chồng thành 2 mạng. Sau đó, dùng hàm **ConvertGray(image)** chuyển hai hình cần chồng với nhau thành ảnh xám. Thực hiện cộng 2 mảng của hai ảnh lại với nhau với mảng ảnh ban đầu nhân với 0.6, mảng ảnh thứ 2 nhân với 0.4 (lúc này ảnh thứ nhất sẽ hiển thị rõ hơn ảnh thứ hai khi thực hiện chồng 2 ảnh lại với nhau).

h. **Làm mờ ảnh: (Gaussian blur 3x3):**

*Ý tưởng và thực hiện:* (Cần xét 2 ảnh: ảnh màu (mảng 3 chiều) và ảnh xám (2 chiều). Vì hai ảnh này xử lý tương tự nhau nên các bước thực hiện cũng tương tự nhau. Đầu tiên, Chuyển ảnh về mảng để xử lý là mảng **arrayImage**
, sau đó, tạo một mảng mới **tempArr** thực hiện copy mảng **arrayImage** vào mảng **tempArr** bằng cách sử dụng hàm **np.copy** trong thư viện numpy. Vì để thu được một ma trận có kích thước bằng kích thước ma trận ban đầu, thì ta thực hiện thêm các giá trị 0 ở viền ngoài của ma trận **tempArr** bằng hàm **np.pad**, sau đó gán kết quả đã padding vào biến mới là **padArr**.

Thực hiện, duyệt từng điểm ảnh bằng 2 dòng for:

+ Lấy ma trận **padArr** với kích thước 3x3 (ứng với từng điểm ảnh của ma trận ban đầu, bằng với kích thước ma trận kernel) nhân với ma trận kernel (3x3), thu được một ma trận kết quả 3x3. Tiếp theo, thực hiện trên ma trận thu được, đối với ma trận ảnh màu (3 chiều), ta cộng 3 pixel cùng một hàng (axis = 1), sau đó cộng tiếp 3 pixel sau khi cộng theo hàng lại. Còn đối với ma trận ảnh xám (2 chiều), sử dụng **np.sum** thực hiện cộng các giá trị trong ma trận vừa thu được. Ta thu được một điểm ảnh (điểm ảnh làm mờ) và gán điểm ảnh này vào ma trận ban đầu **arrayImage** theo giá trị i, j tương ứng.



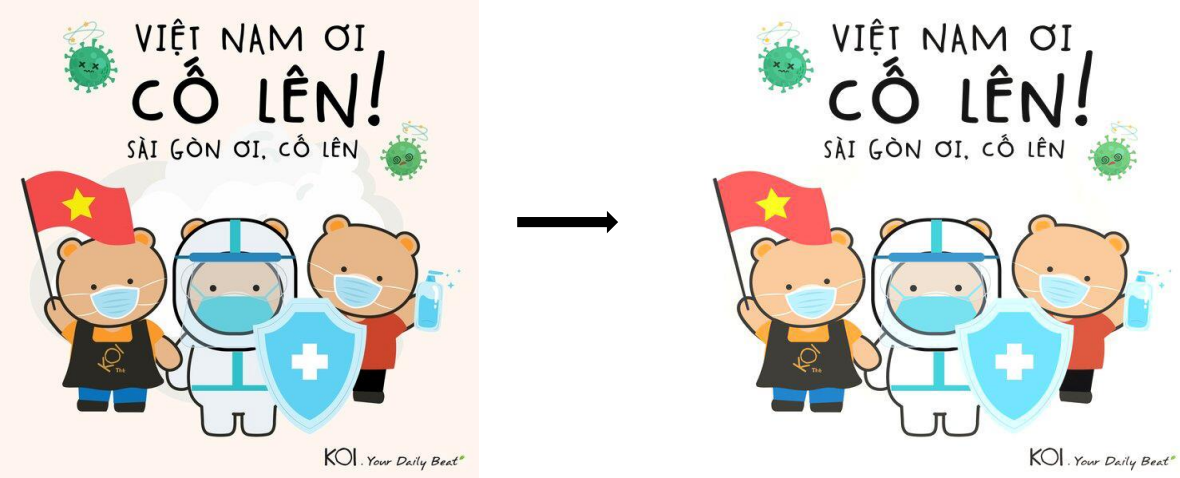

References: https://nttuan8.com/bai-5-gioi-thieu-ve-xu-ly-anh/

http://datahacker.rs/what-is-padding-cnn/

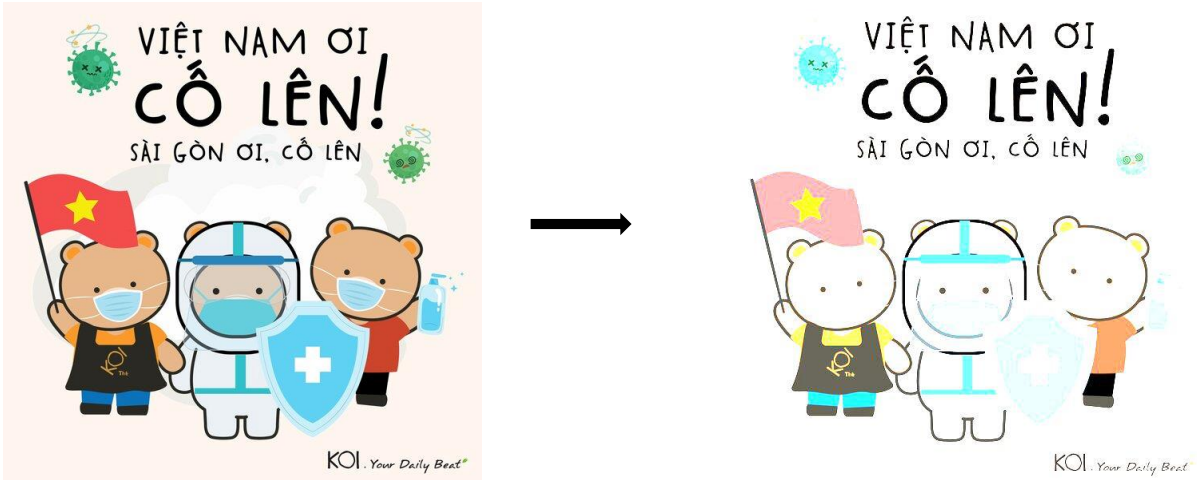
https://note.nkmk.me/en/python-opencv-numpy-alpha-blend-mask/

## 3. Kết quả:

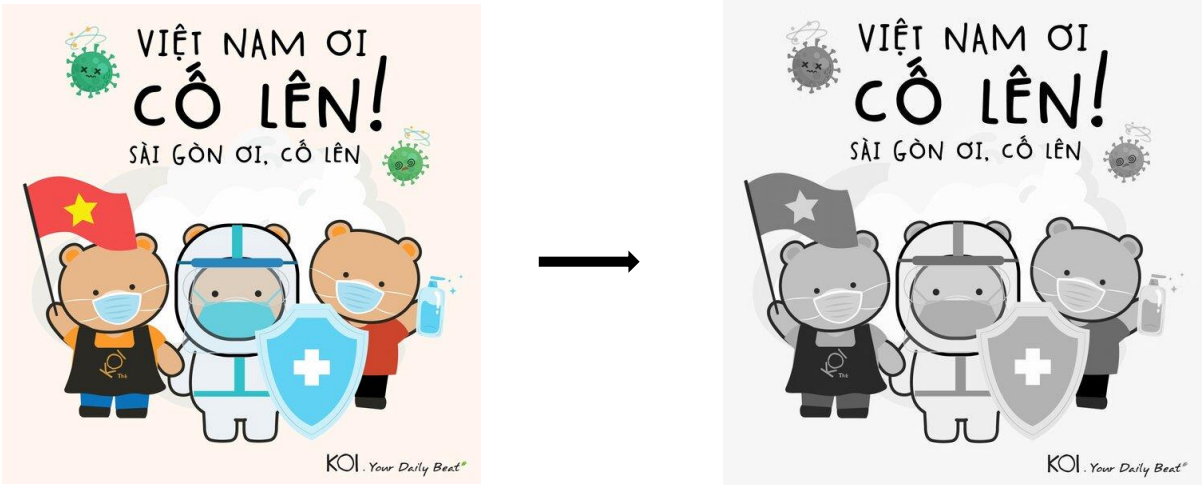
### a. Thay đổi độ sáng cho ảnh:



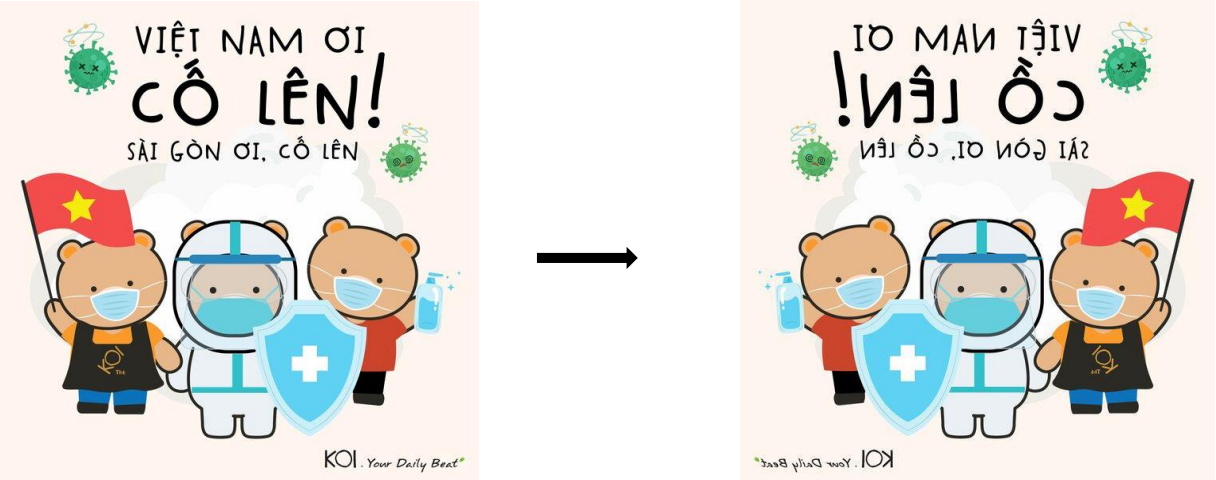

### b. Thay đổi độ tương phản:



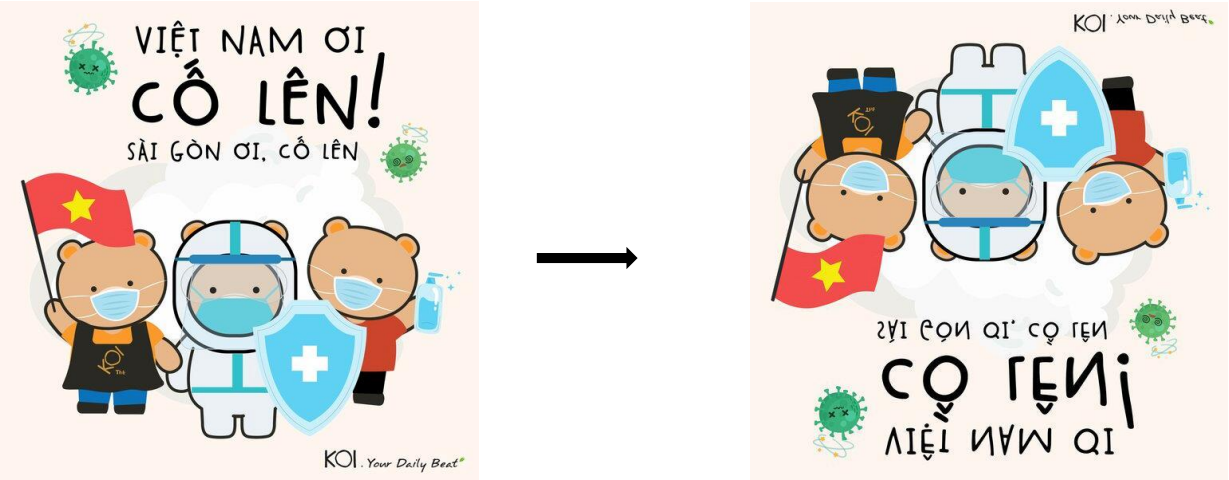

### c. Chuyển ảnh RGB thành ảnh xám:



**d. Lật ảnh ngang:**




**e. Lật ảnh dọc:**

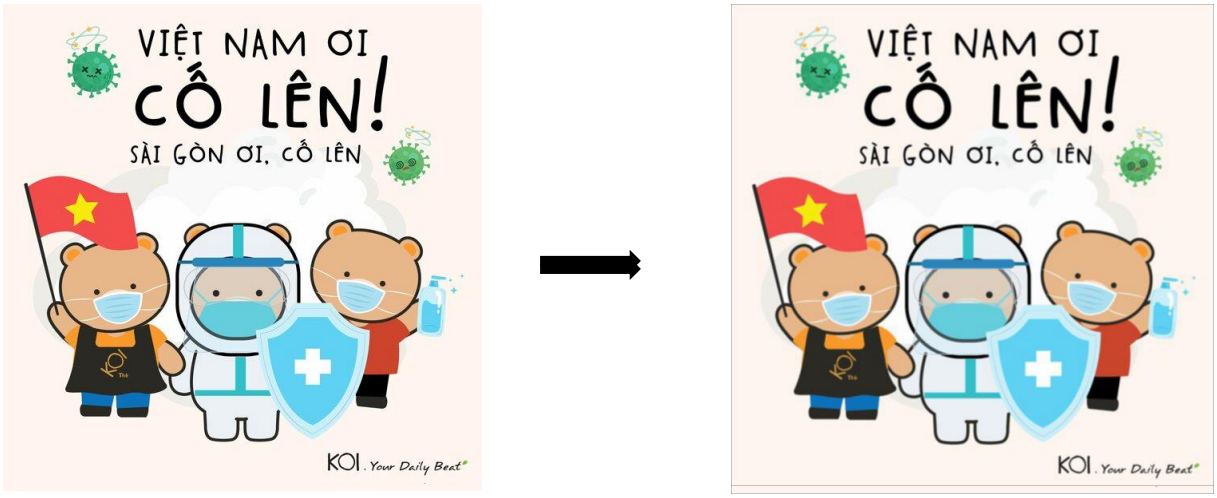



**f. Chồng 2 ảnh có cùng kích thước (ảnh xám):**


*Hình ảnh sau khi chồng nhau*

**g. Làm mờ ảnh: (Gaussian blur 3x3)**

- ***Đối với ảnh màu (mảng 3 chiều):***




- ***Đối với ảnh xám (mảng 2 chiều):***