Date	03 November 2022
Team ID	PNT2022TMID37911
Project Name	AI Based Localization and Classification of Skin Disease with Erythema

Image Processing

Histogram Manipulation

Import the required libraries.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pdfrom skimage.io import imshow, imread
from skimage.color import rgb2gray
from skimage import img_as_ubyte, img_as_float
from skimage.exposure import histogram,
cumulative_distribution
```

Convert the image to greyscale.

```
plt.figure(num=None, figsize=(8, 6), dpi=80)
dark_image_grey =
img_as_ubyte(rgb2gray(image_dark))
imshow(dark_image_grey);
```

Extract the image's value histogram.

```
freq, bins =
histogram(dark_image_grey)plt.figure(num=None,
figsize=(8, 6), dpi=100, facecolor='white') freq, bins =
histogram(dark_image_grey) plt.step(bins,
freq/freq.sum()) plt.xlabel('intensity value', fontsize =
12)
plt.ylabel('fraction of pixels', fontsize = 12);
```

Intensity Values of Image

It is very clear that this histogram does not resemble a normal distribution. You might be tempted to try and snap this distribution

into a normal distribution. However there is a slightly more intuitive way to handle this issue.

Remember that the theoretical Cumulative Distribution Function (CDF) for a normal distribution is a straight line. This being the case, it is better to snap the CDF of our image into a straight line.

Actual CDF of the Image

To do this, we can make use of the *interpolate* function in NumPy.

```
interpolation = np.interp(freq, target_freq, target_bins)
```

Use the interpolation to help us adjust the actual CDF.

```
dark_image_eq                                                    =
img_as_ubyte(interpolation[dark_image_grey].astype(int))
```

View the actual image.

```
imshow(dark_image_eq);
```

Create a function which will adjust the CDF of any image we feed it.

```
def histogram_adjuster(image):
    dark_image_grey = img_as_ubyte(rgb2gray(image)) freq,
    bins = cumulative_distribution(dark_image_grey)
target_bins    =    np.arange(255)    target_freq    =
    np.linspace(0, 1, len(target_bins))
interpolation  =  np.interp(freq,  target_freq,  target_bins)
    dark_image_eq =
    img_as_ubyte(interpolation[dark_image_grey].astype(int))
    freq_adj, bins_adj =
    cumulative_distribution(dark_image_eq)


    fig,     axes     =     plt.subplots(1,     2,
    figsize=(15,7)); imshow(dark_image_grey, ax =
    axes[0]); imshow(dark_image_eq, ax = axes[1]);

    axes[0].axis('off') axes[1].axis('off')
    axes[0].set_title('Unadjusted Image', fontsize =
    17) axes[1].set_title('Adjusted Image', fontsize =
    17)

    fig,  axes  =  plt.subplots(1,  1,  figsize=(19,7));
plt.step(bins,   freq,   c='blue',   label='Actual   CDF')
plt.step(bins_adj,  freq_adj,  c='purple',  label='Adjusted
CDF') plt.plot(target_bins,
            target_freq,
            c='red',
            label='Target
            CDF',  linestyle  =
            '--')
```

```
plt.legend(prop={'size':    14})
plt.xlim(0, 255) plt.ylim(0, 1)
plt.xlabel('Intensity values', fontsize = 15)
plt.ylabel('Cumulative fraction of pixels', fontsize = 15);
```

Adjust the colored image directly.

```
def histogram_adjuster_color(image):
    freq, bins = cumulative_distribution(image) target_bins =
np.arange(255) target_freq = np.linspace(0, 1,
    len(target_bins))
interpolation = np.interp(freq, target_freq, target_bins)
    image_eq = img_as_ubyte(interpolation[image].astype(int))
    freq_adj, bins_adj = cumulative_distribution(image_eq)


    fig,     axes     =     plt.subplots(1,     2,
    figsize=(15,7)); imshow(image, ax = axes[0]);
    imshow(image_eq, ax = axes[1]);

    axes[0].axis('off') axes[1].axis('off')
    axes[0].set_title('Unadjusted Image', fontsize =
    17) axes[1].set_title('Adjusted Image', fontsize =
    17)

    fig,  axes  =  plt.subplots(1,  1,  figsize=(19,7));
    plt.step(bins,  freq,  c='blue',  label='Actual  CDF')
    plt.step(bins_adj,        freq_adj,        c='purple',
    label='Adjusted
CDF')
    plt.plot(target_bins,
             target_freq,
             c='red',
             label='Target
             CDF',  linestyle  =
             '--')

    plt.legend(prop={'size':
    15})    plt.xlim(0,    255)
    plt.ylim(0, 1)
    plt.xlabel('Intensity values', fontsize = 17)
    plt.ylabel('Cumulative fraction of pixels', fontsize =
    17);
```