# Department of Computer Science and Engineering (Data Science) Image Processing and Computer Vision I

**NAME : AADITYA MALANI** **SAP ID:60009220192** **BATCH:D1-1**

**Aim:** To Perform Histogram Equalisation and stretching.

**Theory:** Histogram equalization is used to enhance contrast. It is not necessary that contrast will always be increase in this. There may be some cases were histogram equalization can be worse. In that cases the contrast is decreased.
Consider for a moment continuous functions, and let the variable r represent the gray levels of the image to be enhanced. We assume that r has been normalized to the interval [0, 1], with r=0 representing black and r=1 representing white. Later, we consider a discrete formulation and allow pixel values to be in the interval [0, L-1]. For any r satisfying the aforementioned conditions, we focus attention on transformations of the form

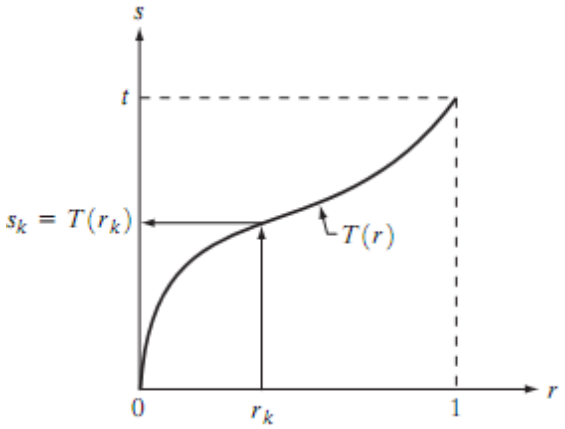
$$s = T(r) \qquad 0 \le r \le 1$$

that produce a level s for every pixel value r in the original image. For reasons that will become obvious shortly, we assume that the transformation function T(r) satisfies the following conditions:

(a) T(r) is single-valued and monotonically increasing in the interval $0 \le r \le 1$; and (b) $0 \le T(r) \le 1$ for $0 \le r \le 1$. The requirement in (a) that T(r) be single valued is needed to guarantee that the inverse transformation will exist, and the monotonicity condition preserves the increasing order from black to white in the output image. A transformation function that is not monotonically increasing could result in at least a section of the intensity range being inverted, thus producing some inverted gray levels in the output image. Finally, condition (b) guarantees that the output gray levels will be in the same range as the input levels. Figure 4.1 gives an example of a transformation function that satisfies these two conditions. The inverse transformation from s back to r is denoted

$$r = T^{-1}(s) \qquad 0 \le s \le 1$$

It can be shown by example that even if T(r) satisfies conditions (a) and (b), it is possible that the corresponding inverse T-1 (s) may fail to be single valued.




A gray-level transformation function that is both single valued and monotonically increasing.

The gray levels in an image may be viewed as random variables in the interval [0, 1]. One of the most fundamental descriptors of a random variable is its probability density function (PDF).Let pr(r) and ps(s) denote the probability density functions of random variables r and s, respectively, where the subscripts on p are used to denote that pr and ps are different functions. A basic result from an elementary probability theory is that, if pr(r) and T(r) are known and T-1 (s) satisfies condition (a),

then the probability density function ps(s) of the transformed variable s can be obtained using a rather simple formula:

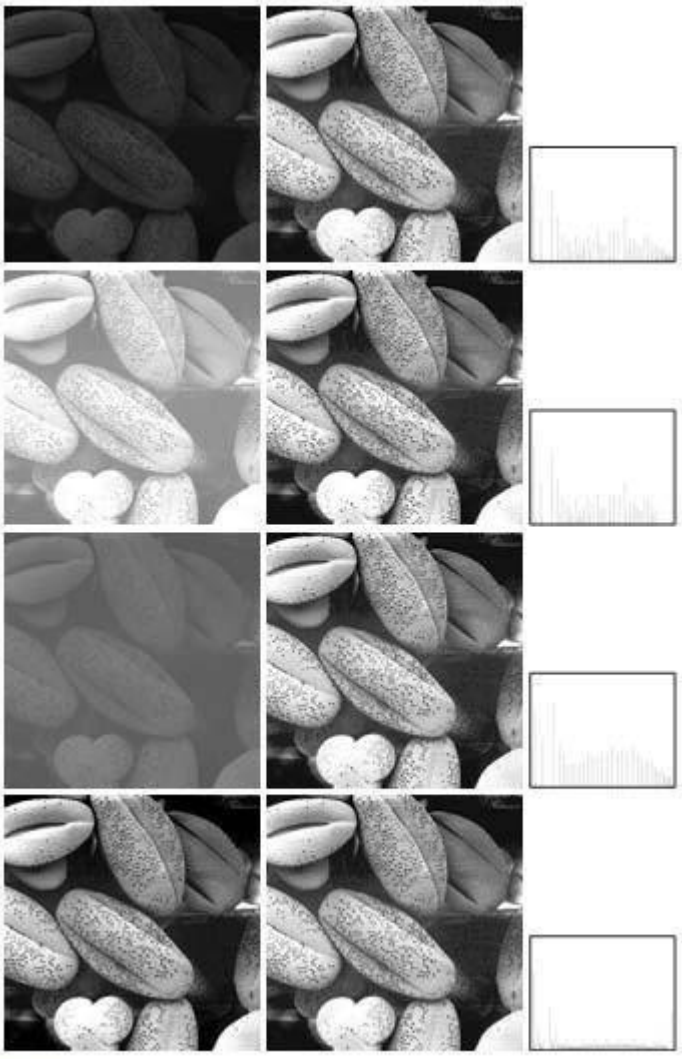
$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

Thus, the probability density function of the transformed variable, s, is determined by the gray- level PDF of the input image and by the chosen transformation function. A transformation function of particular importance in image processing has the form

$$s = T(r) = \int_0^r p_r(w)\,dw$$

where w is a dummy variable of integration. The right side of Eq. above is recognized as the cumulative distribution function (CDF) of random variable r. Since probability density functions are always positive, and recalling that the integral of a function is the area under the function, it follows that this transformation function is single valued and monotonically increasing, and, therefore, satisfies condition (a). Similarly, the integral of a probability density function for variables in the range [0, 1] also is in the range [0, 1], so condition (b) is satisfied as well.

Thus, a processed (output) image is obtained by mapping each pixel with level rk in the input image into a corresponding pixel with level sk in the output image. As indicated earlier, a plot of pr (rk) versus rk is called a histogram. The transformation (mapping) is called histogram equalization or histogram linearization. It is not difficult to show that the transformation in Eq. satisfies conditions (a) and (b) stated previously. Unlike its continues counterpart, it cannot be proved in general that this discrete transformation will produce the discrete equivalent of a uniform probability density function, which would be a uniform histogram.

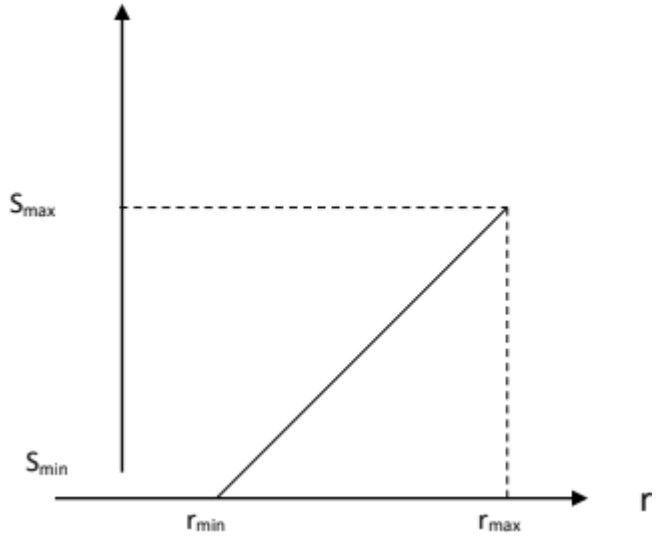


a b c

(a) Images from Fig.3 (b) Results of histogram equalization. (c) Corresponding histograms.

# Histogram Stretching

It is a method to increase the dynamic range of the image. Here we do not alter the basic shape of the histogram, but we spread it so as to cover the entire dynamic range. We do this by using a straight line equation having a slope
(smax – smin)/ (rmax –rmin)




smax = Maximum grey level of output image smin = Minimum grey level of output image. rmax = Maximum grey level of input image rmin = Minimum grey level of input image.

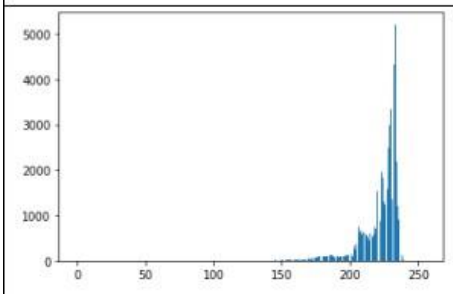
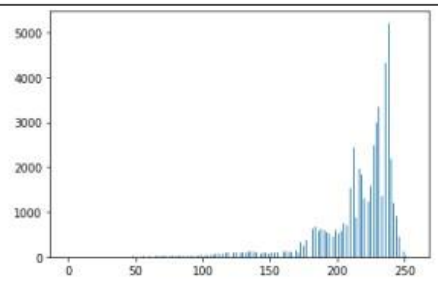
$$S = T(r) = \frac{(smax - smin)}{(rmax - rmin)} (r - r_{min}) + smin$$

This transformation stretches and shifts the grey level range of input image to occupy the entire dynamic range (smax , smin).

**Department of Computer Science and Engineering (Data Science)**

**Image Processing and Computer Vision I (DJ19DSL603)**

Lab 4: Histogram Equalisation

| **Histogram of Original Image** | **Histogram after Histogram Stretching** |
|---|---|
|  |  |
| **Original Image** | **Transformed Image after Histogram Stretching** |
|  |  |

**Department of Computer Science and Engineering (Data Science)**

**Image Processing and Computer Vision I (DJ19DSL603)**

Lab 4: Histogram Equalisation

## Importing packages

```
import numpy as np
import matplotlib.pyplot as plt
from keras.datasets import fashion_mnist
import cv2
import random
```

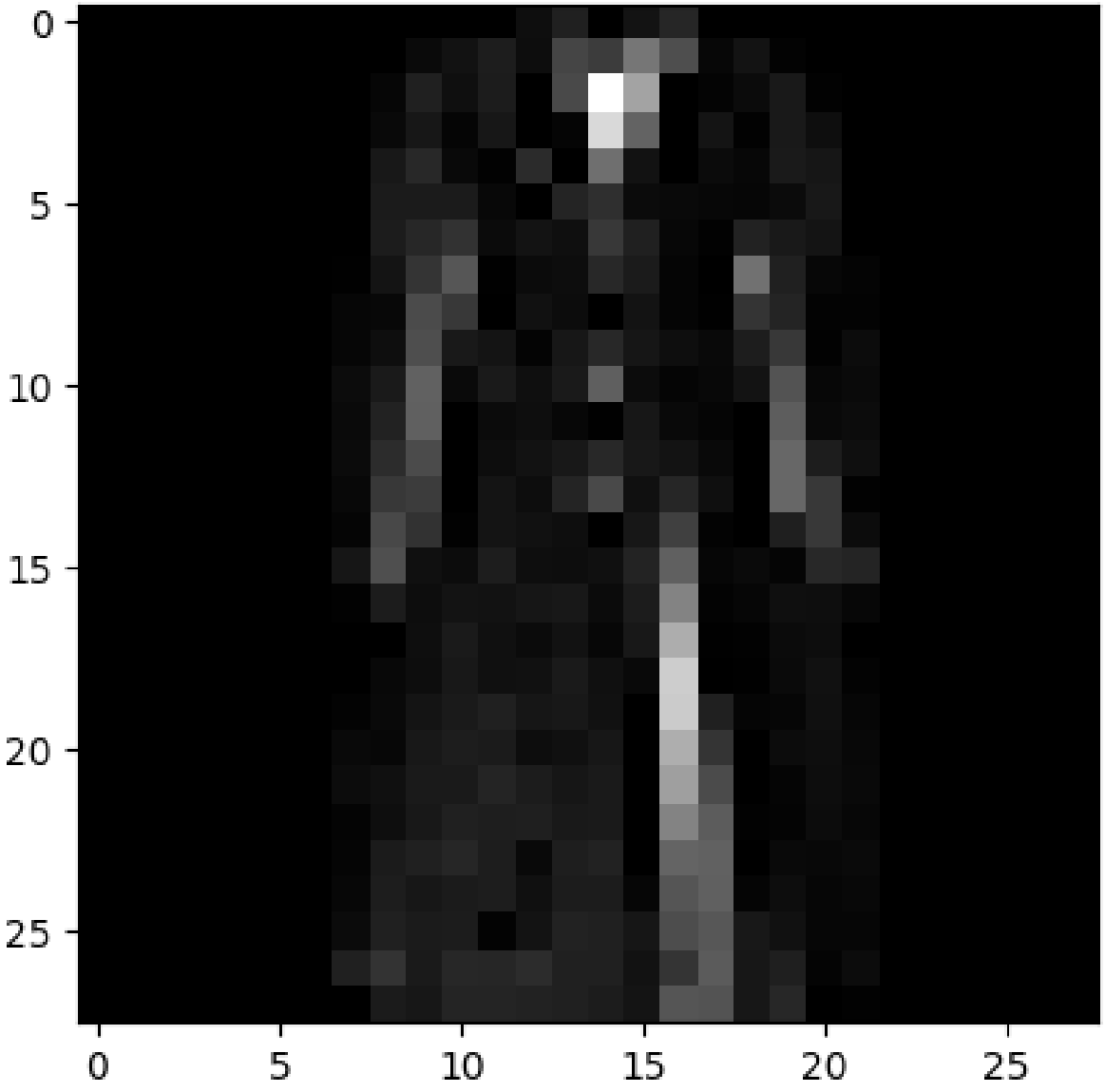
## Loading Fashion MNIST dataset

```
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
def plot(img):
  plt.imshow(img,cmap="gray")
  plt.show()

random_img = random.choice(x_train)
plot(random_img)
```




Histogram Equalization

```
def  histogramEqualization(image):
  unique_intensity_values, pixel_counts = np.unique(image,
return_counts=True)
  normalized_pixel_counts = pixel_counts / np.sum(pixel_counts)
  cumulative_distribution = np.cumsum(normalized_pixel_counts)
  scaled_cumulative_distribution = (max(unique_intensity_values) - 1)
*  cumulative_distribution
  equalized_values = list(map(round, cumulative_distribution * 255))

  equalized_image  =  np.zeros_like(image)
  for intensity_index in range(len(unique_intensity_values)):
    equalized_image[image == unique_intensity_values[intensity_index]]
=      equalized_values[intensity_index]

  return equalized_image
```

## Visualization  Tools

```
def visualizeImages(image):
  figure, axes = plt.subplots(1, 2, figsize=(20, 5))

  # Display the original image
  axes[0].imshow(image, cmap='gray')
  axes[0].set_title("Original  Image")
```

```
  axes[0].axis('off')

  # Display the histogram-equalized image
  axes[1].imshow(histogramEqualization(image), cmap='gray')
  axes[1].set_title("Histogram-Equalized Image")
  axes[1].axis('off')

  plt.show()

def visualizeHistograms(image):
  figure, axes = plt.subplots(1, 2, figsize=(20, 5))

  # Original Image Histogram
  axes[0].hist(image.ravel(), bins=256, color='gray', alpha=0.7)
  axes[0].set_title("Original Image Histogram")
  axes[0].set_xlabel("Pixel Value")
  axes[0].set_ylabel("Frequency")

  # Histogram-Equalized Image Histogram
  axes[1].hist(histogramEqualization(image).ravel(), bins=256,
color='gray', alpha=0.7)
  axes[1].set_title("Histogram-Equalized Image Histogram")
  axes[1].set_xlabel("Pixel Value")
  axes[1].set_ylabel("Frequency")

  plt.show()
```

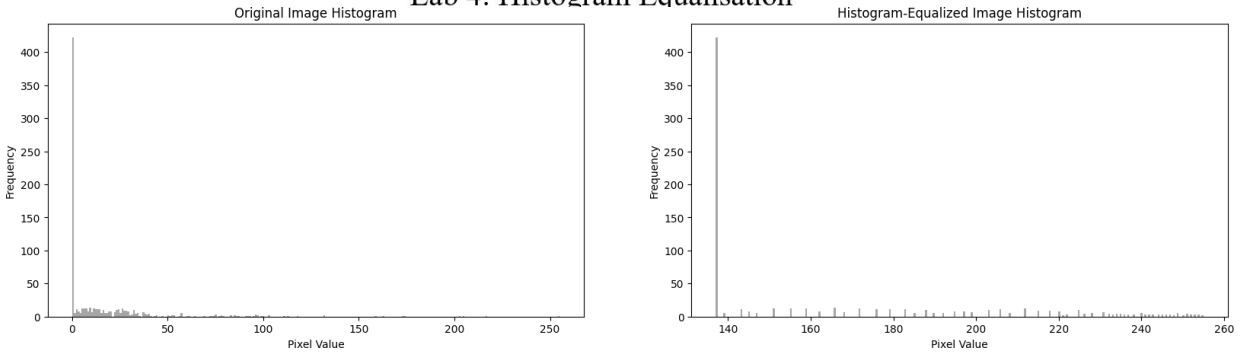
## Performing Histogram Equalization on Fashion Mnist image

```
visualizeImages(random_img)
```


```
visualizeHistograms(random_img)
```

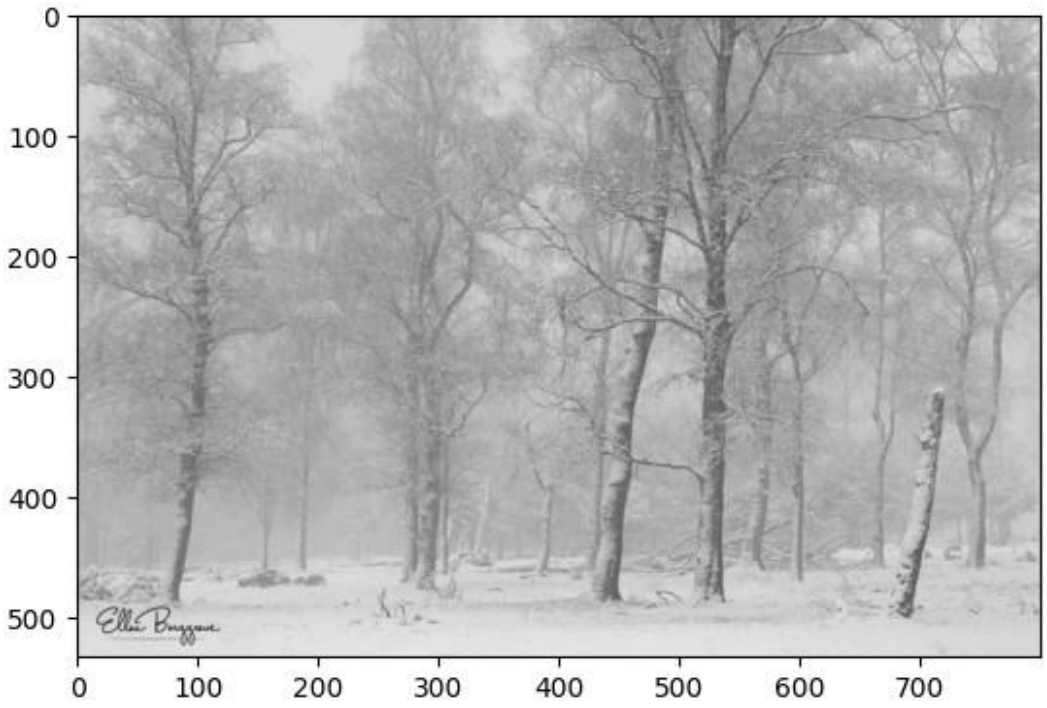
**Department of Computer Science and Engineering (Data Science)**
**Image Processing and Computer Vision I (DJ19DSL603)**

Lab 4: Histogram Equalisation




## Performing Histogram Equalization on low-contrast image

```
img = cv2.imread("/content/tree.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plot(img)
```



```
visualizeImages(img)
```

Shri Vile Parle Kelavani Mandal's
**DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING**
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

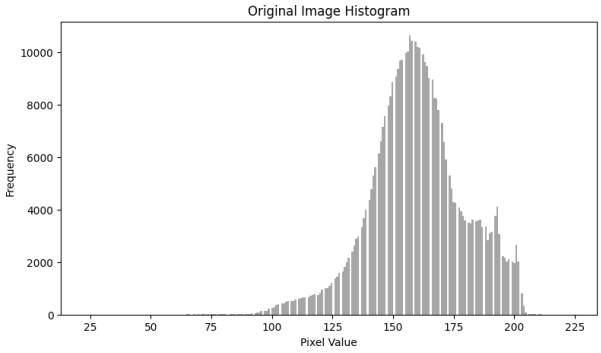
**Department of Computer Science and Engineering (Data Science)**
**Image Processing and Computer Vision I (DJ19DSL603)**

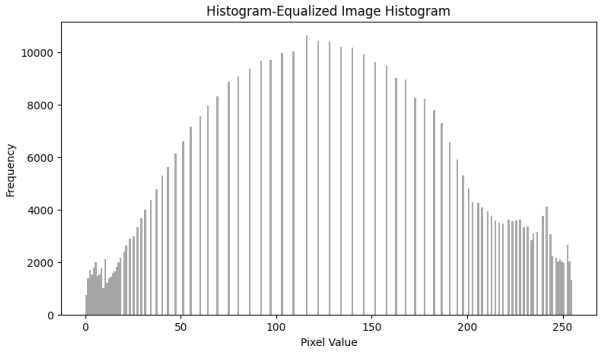
Lab 4: Histogram Equalisation



## Histograms before and after equalization

```
visualizeHistograms(img)
```




## Conclusion:

As observed, the histograms expand after applying histogram equalization. In low-contrast images, pixel values are concentrated within a narrow range. Histogram equalization redistributes these values across a broader range, enhancing contrast.

When applying histogram equalization to the Fashion MNIST dataset, we notice that the value 0 appears more frequently than other pixel values, making it harder to visually detect the histogram stretching. However, it is still evident that the values become more evenly distributed.