Department of Computer Science and Engineering (Data Science)

Image Processing and Computer Vision I (DJ19DSL603)

Lab 2: Image Negative Transformation, Thresholding, Gray Level Slicing with without background

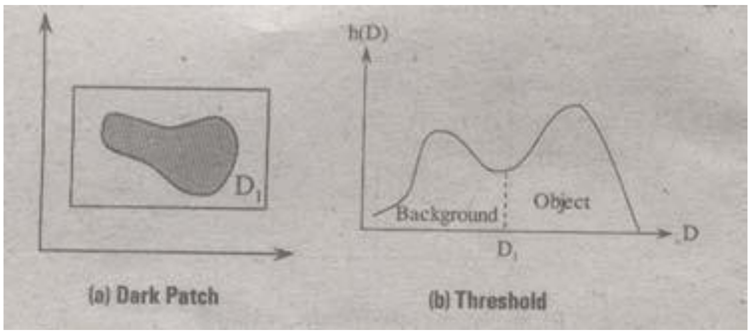
Aim: To Perform Gray Level slicing with and without background

Theory: Thresholding is a simple but effective image processing technique that is widely used in computervision and image analysis applications. The basic idea behind thresholding is to segment an image into foreground and background regions based on the intensity values of the pixels in the image

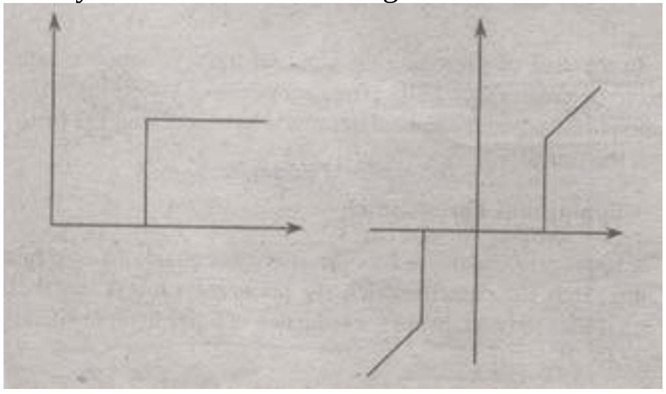
Clipping and Thresholding:

Clipping is considered as the special scenario of contrast stretching. It is the case in which the parameters are $\alpha = \gamma = 0$. Clipping is more advantageous for reduction of noise in input signals of range [a, b].

Threshold of an image is selected by means of its histogram. Let us take the image shown in the following figure.



(b) consists of two peaks i.e., background and object. At the abscissa of histogram minimum (D1) the threshold is selected. This selected threshold (D1) can separate background and object to convert the image into its respective binary form. The thresholding transformations are shown in figure



Intensity Level Slicing:

The images which consist of grey levels in between intensity at background and other objects require to reduce the intensity of the object. This process of changing intensity level is done with the help of intensity level slicing. They are expressed as

$$V = \begin{cases} L, & a \le u \le b \\ 0, & \text{elsewhere} \end{cases} \quad \text{without background}$$

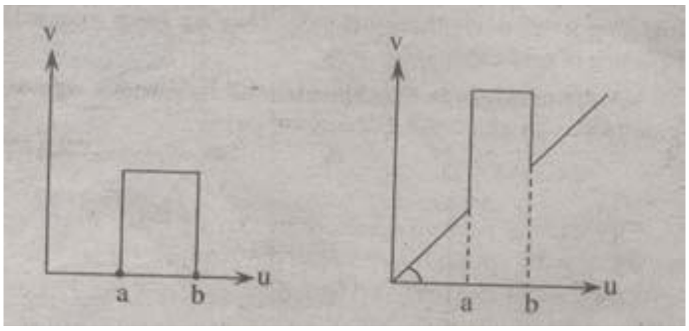
$$\text{And} \quad V = \begin{cases} L, & a \le u \le b \\ u', & \text{elsewhere} \end{cases} \quad \text{with background}$$

The histogram of input image and its respective intensity level slicing is shown in the figure

Department of Computer Science and Engineering (Data Science)

Image Processing and Computer Vision I (DJ19DSL603)

Lab 2: Image Negative Transformation, Thresholding, Gray Level Slicing with without background



When an image is uniformly quantized then, the nth most significant bit can be extracted and displayed. Let, u = k1 2^B-1 + k2 2^B-2 + ...+ k^B-1 2 + kB Then, the output is expressed as

$$V = \begin{cases} L, & \text{for } k_n = 1 \\ 0, & \text{elsewhere} \end{cases}$$

Grey Level Slicing without background:

Grey level slicing is an image processing technique used to enhance the contrast of an image by selectively enhancing a range of pixel intensities. The basic idea behind grey level slicing is to set all pixel values within a specific intensity range to a maximum value, while leaving all other pixelvalues unchanged

.

Lab Assignments to complete in this session

Problem Statement: Develop a Python program utilizing the OpenCV library to manipulate images from the Fashion MNIST digits dataset. The program should address the following tasks:

1. Read random image(s) from the MNIST fashion dataset.
2. **Dataset Link:** Fashion MNIST Github
3. Display the before & after image(s) used in the task below.
4. Perform image negative transformation.
5. Perform image thresholding operation at various threshold level and write your observation.
6. Perform gray level slicing with and without background intensity slicing and write your observation.

The solution to the operations performed must be produced by scratch coding without the use of built in OpenCV methods.

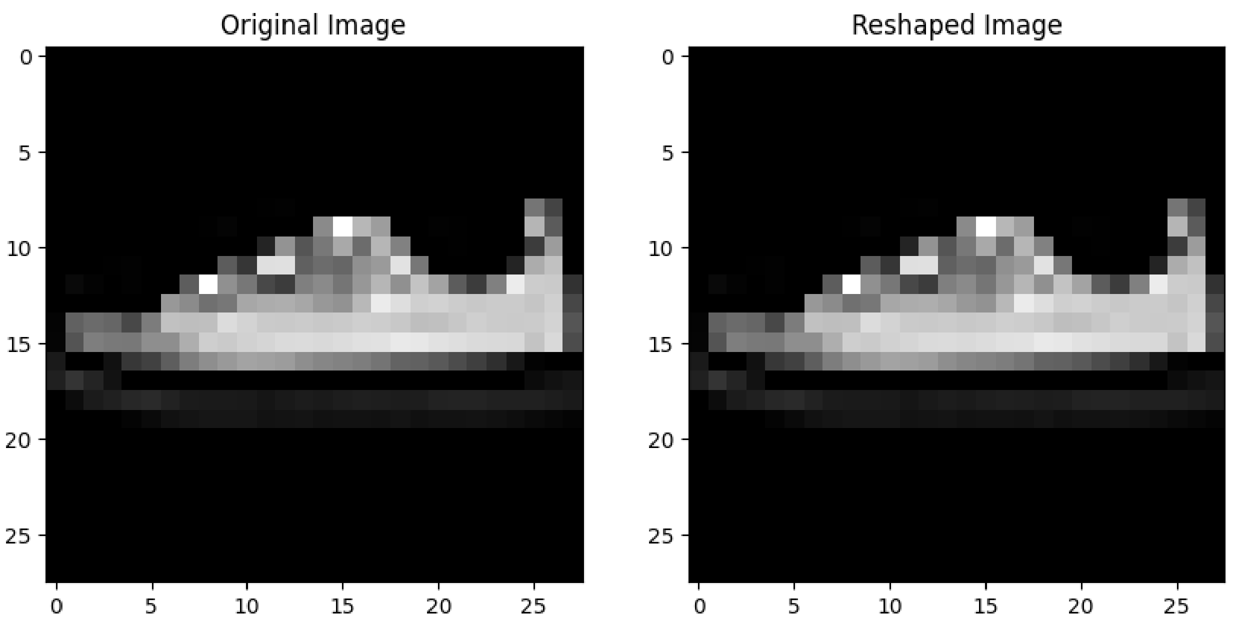
ipcv-lab2-60009220192

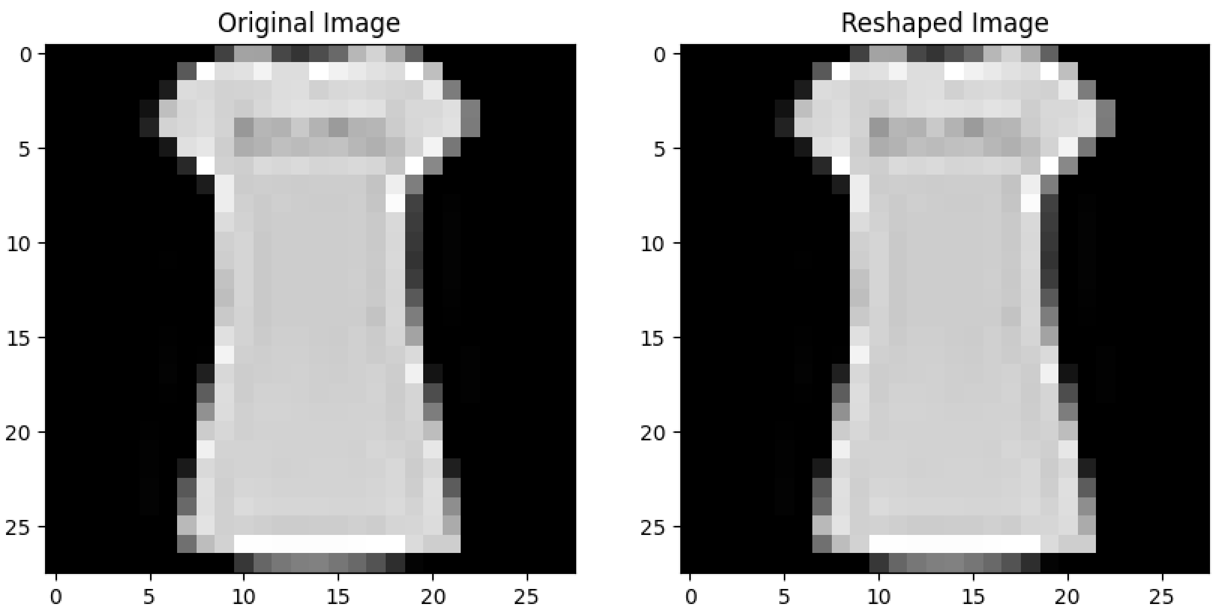
August 19, 2024

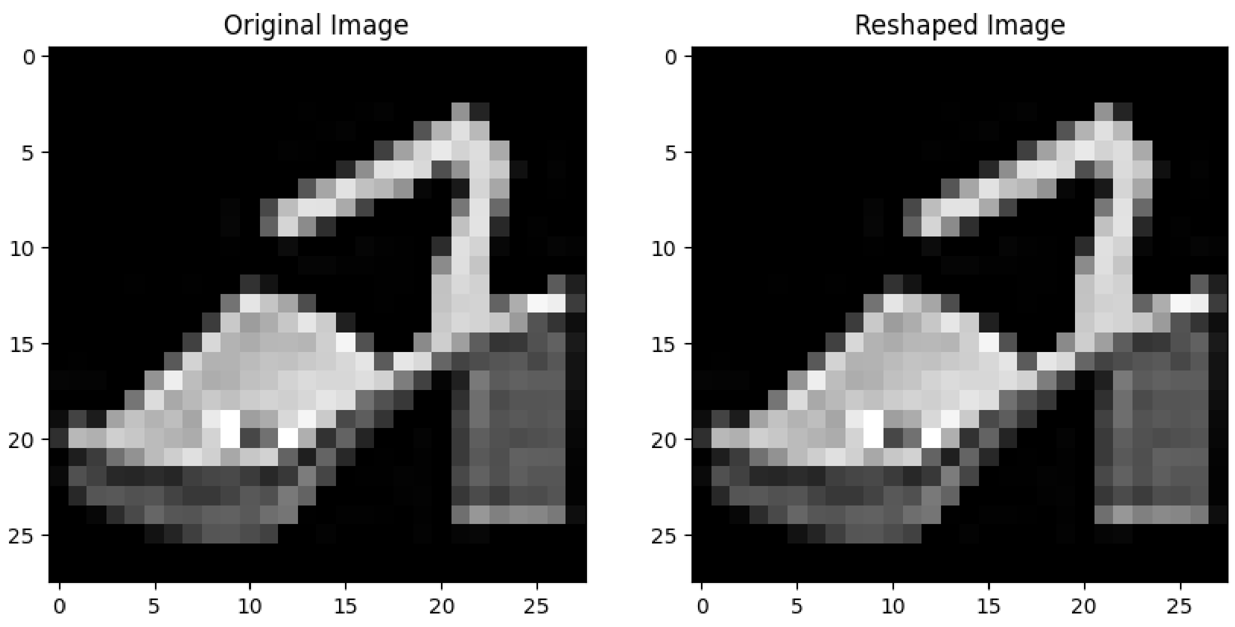
```
[ ]: import tensorflow as tf
import random
import matplotlib.pyplot as plt

(train_images,train_labels)=tf.keras.datasets.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 ━━━━━━━━━━━━━━━━━━━━ 1s 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 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

```
train_images = train_images[0].reshape(60000, 28, 28, 1)

import matplotlib.pyplot as plt
import random

for i in range(4):
    random_index = random.randint(0, len(train_images))
    fig, axs = plt.subplots(1, 2, figsize=(10, 5))
    axs[0].imshow(train_images[random_index], cmap="gray")
    axs[0].set_title('Original Image')
    axs[1].imshow(train_images[random_index].reshape(28, 28),
cmap="gray")
    axs[1].set_title('Reshaped Image')
    plt.show()
```


Original Image
Reshaped Image
0
5
10
15
20
25


Original Image
Reshaped Image
0
5
10
15
20
25



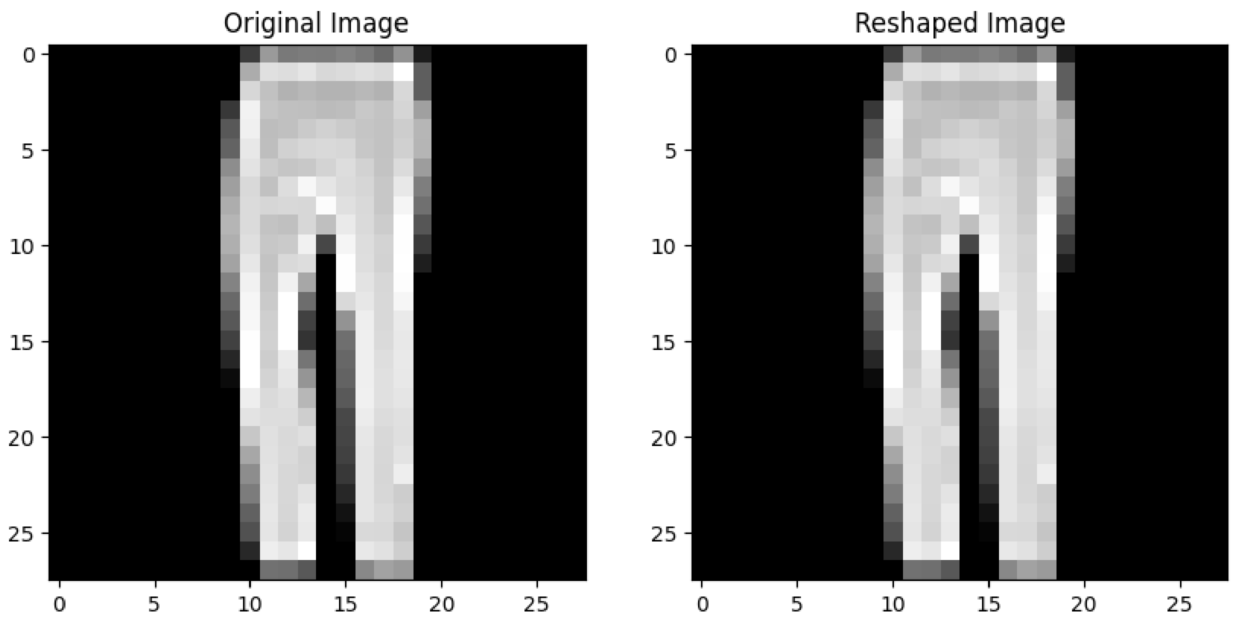
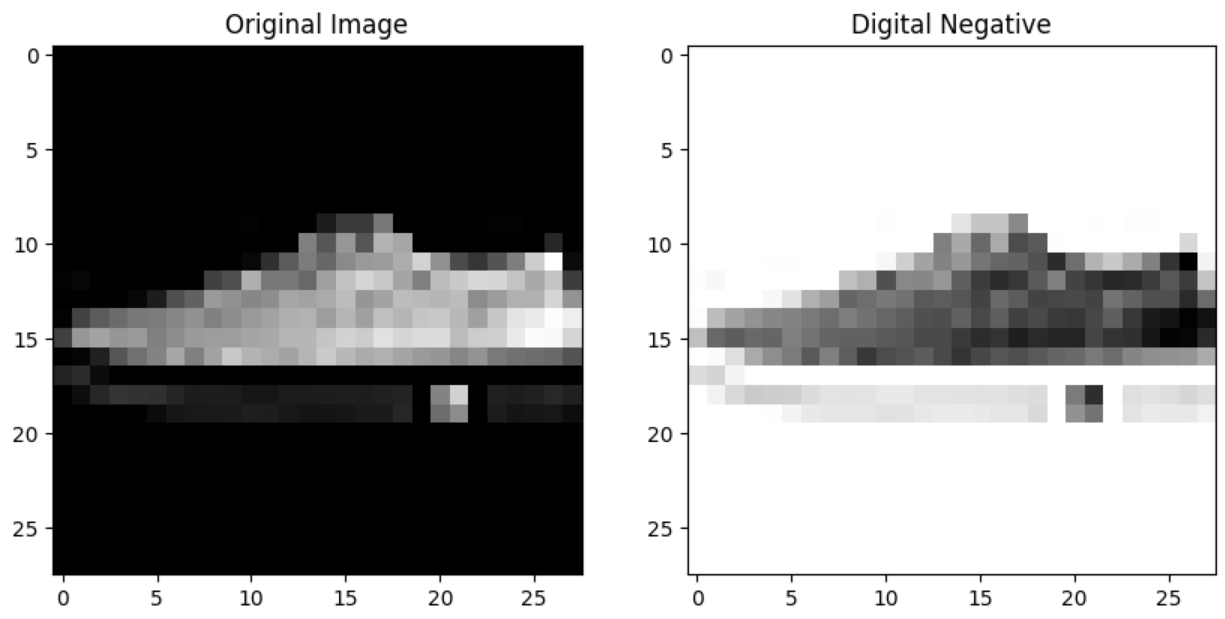
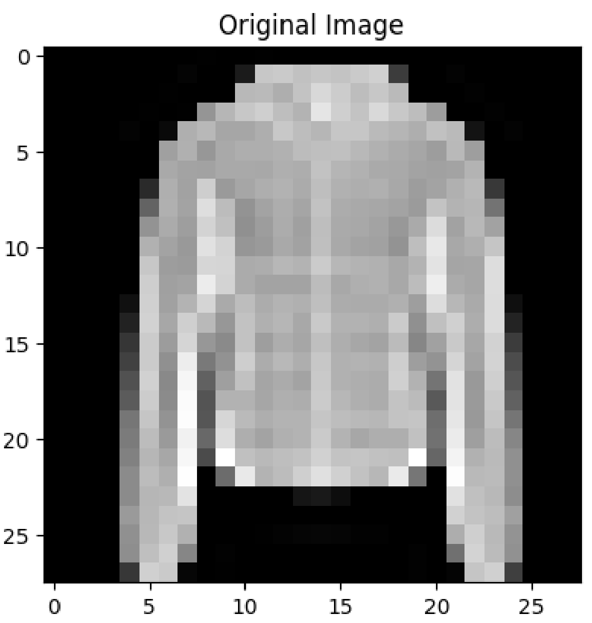


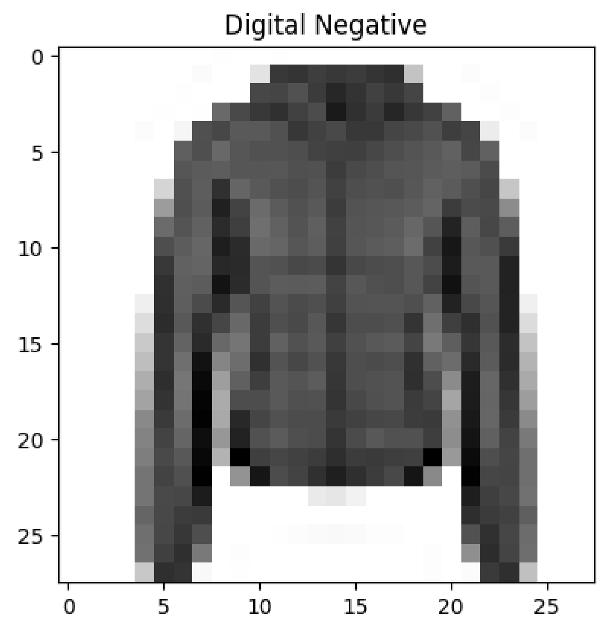
IMAGE NEGATIVES

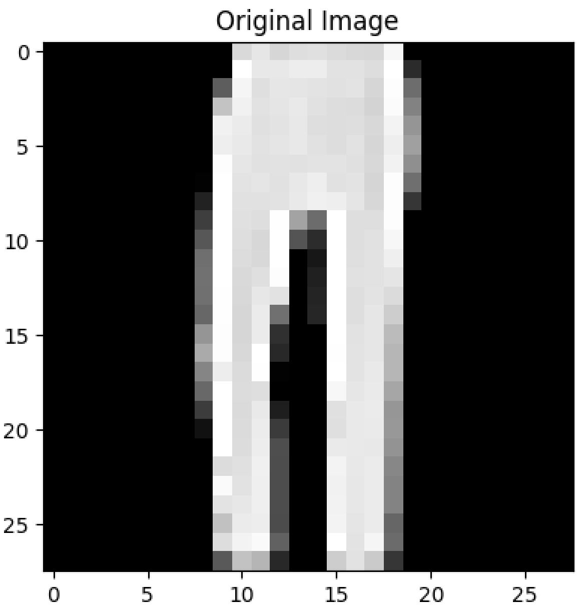
```
import matplotlib.pyplot as plt
import random

def digital_negative(img):
   neg=[]
   for i in range(len(img)):
      temp=[]
      for j in range(len(img[i])):
         temp.append(255-img[i][j])
```

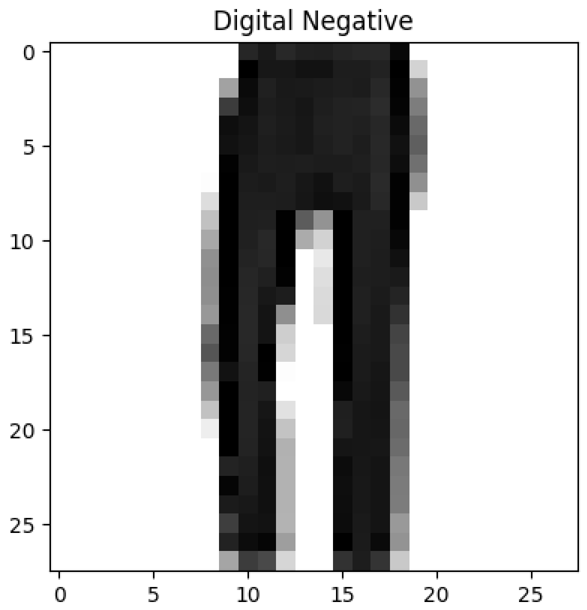
```
    neg.append(temp)
  return neg

for i in range(4):
    random_index = random.randint(0, len(train_images))
    fig, axs = plt.subplots(1, 2, figsize=(10, 5))
    axs[0].imshow(train_images[random_index], cmap="gray")
    axs[0].set_title('Original Image')
    axs[1].imshow(digital_negative(train_images[random_index]),
cmap="gray")
    axs[1].set_title('Digital Negative')
    plt.show()
```

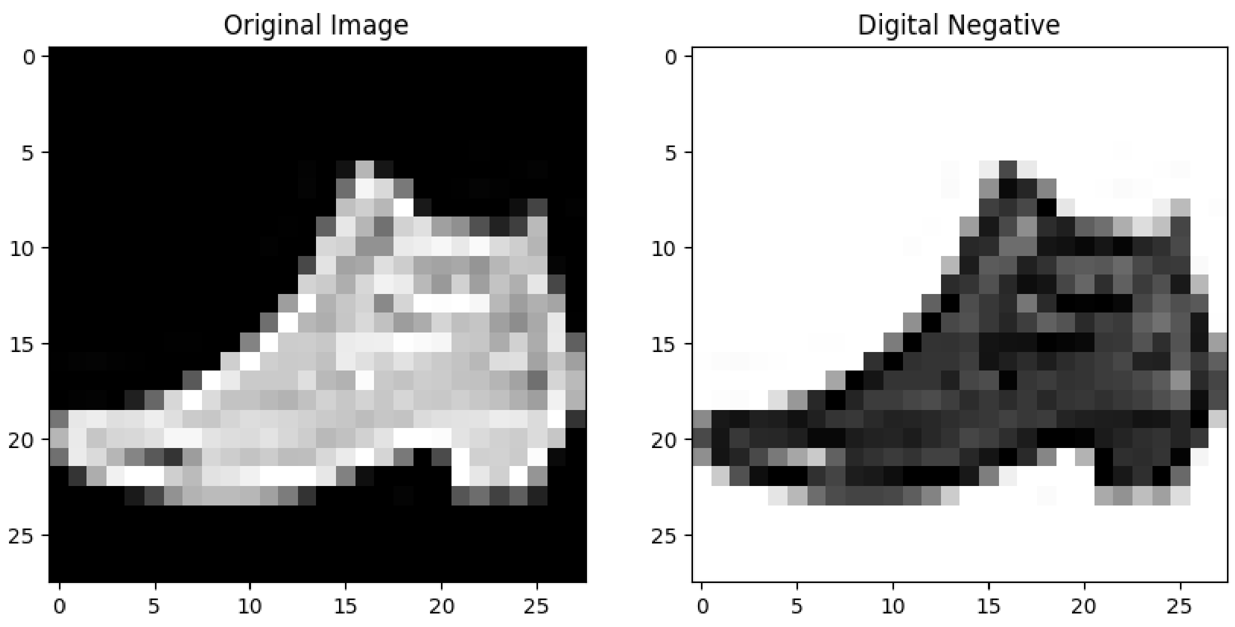



Original Image
0
5
10
15
20
25
0
5
10
15
20
25


Digital Negative
0
5
10
15
20
25
0
5
10
15
20
25

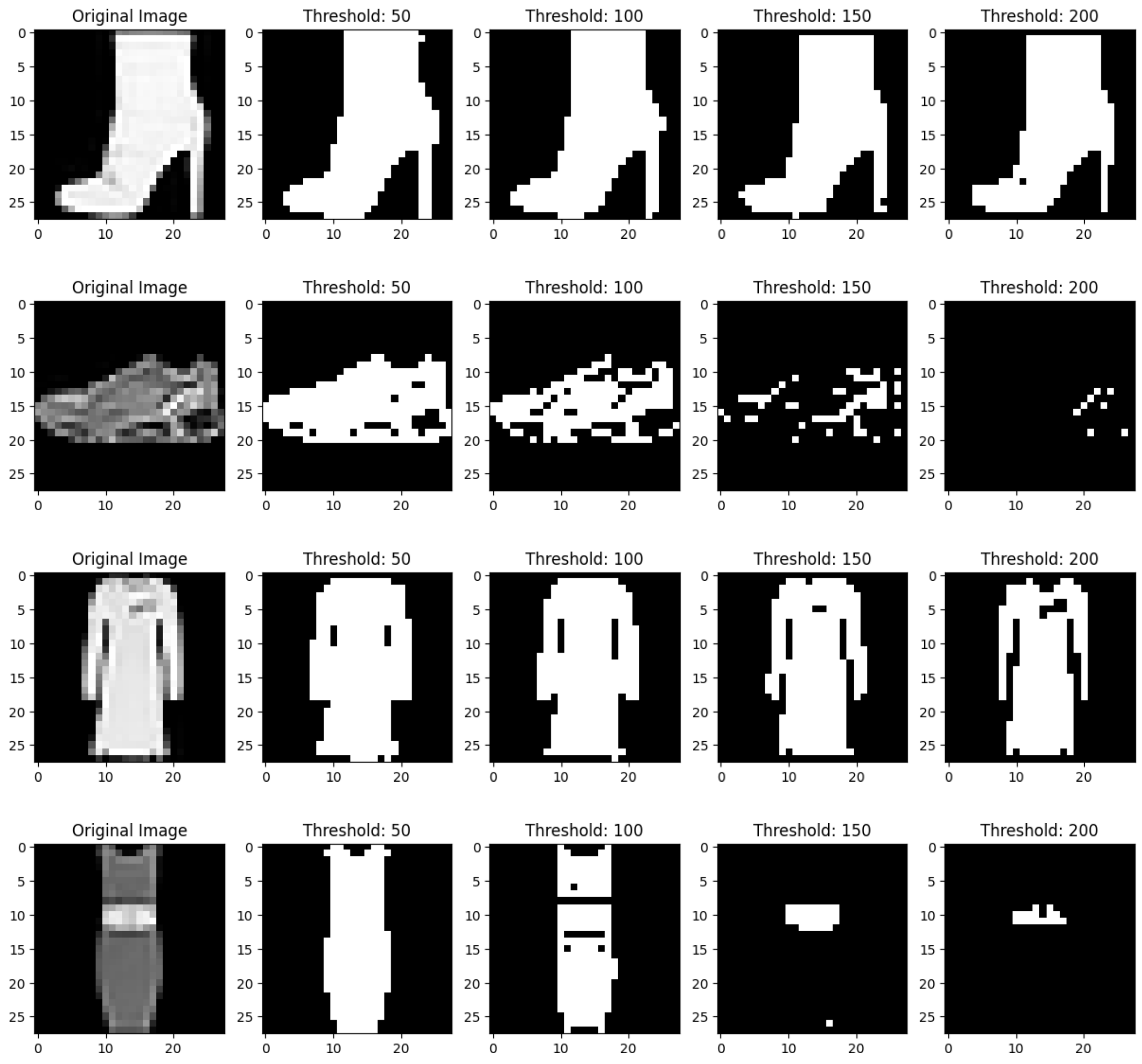

Original Image
0
5
10
15
20
25
0
5
10
15
20
25


Digital Negative
0
5
10
15
20
25
0
5
10
15
20
25

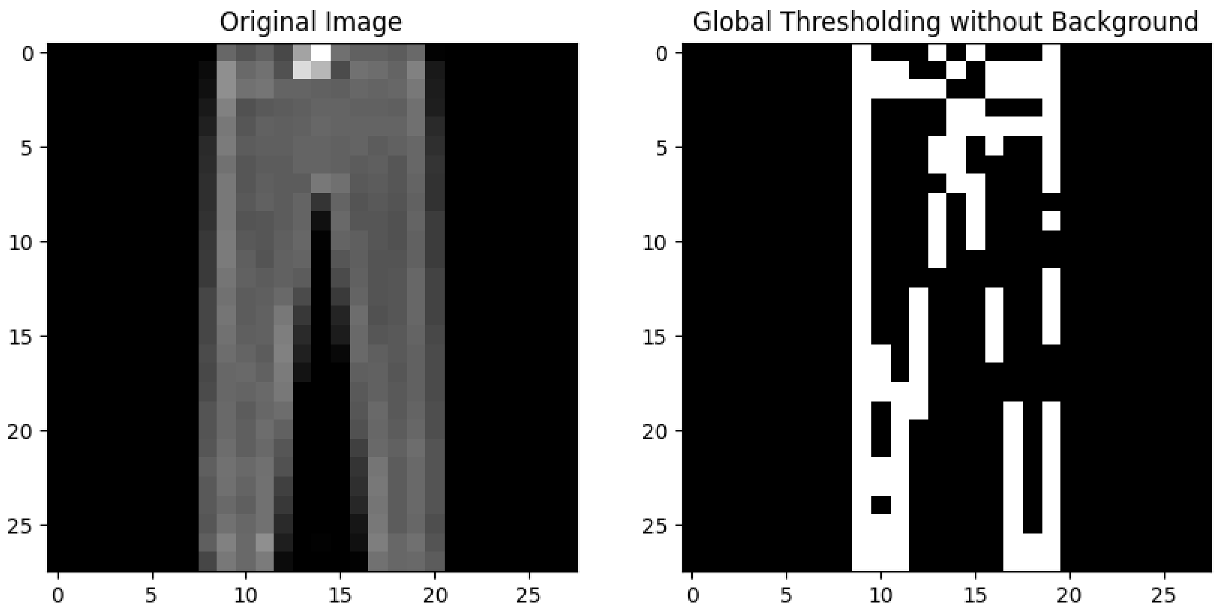


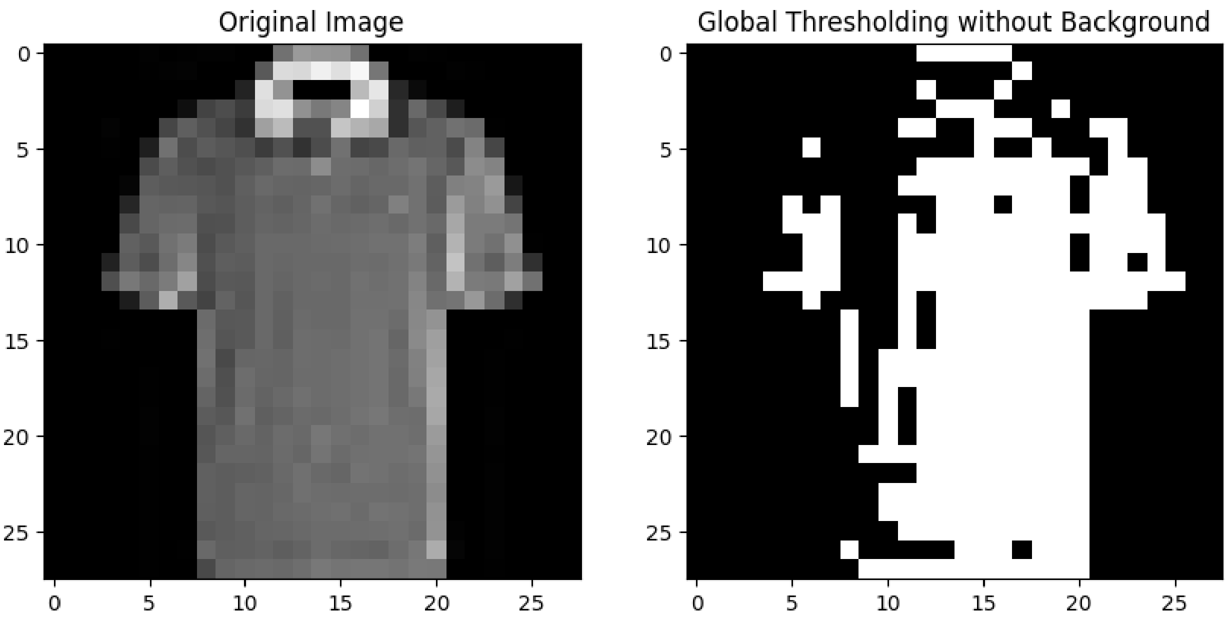
THRESHOLDING

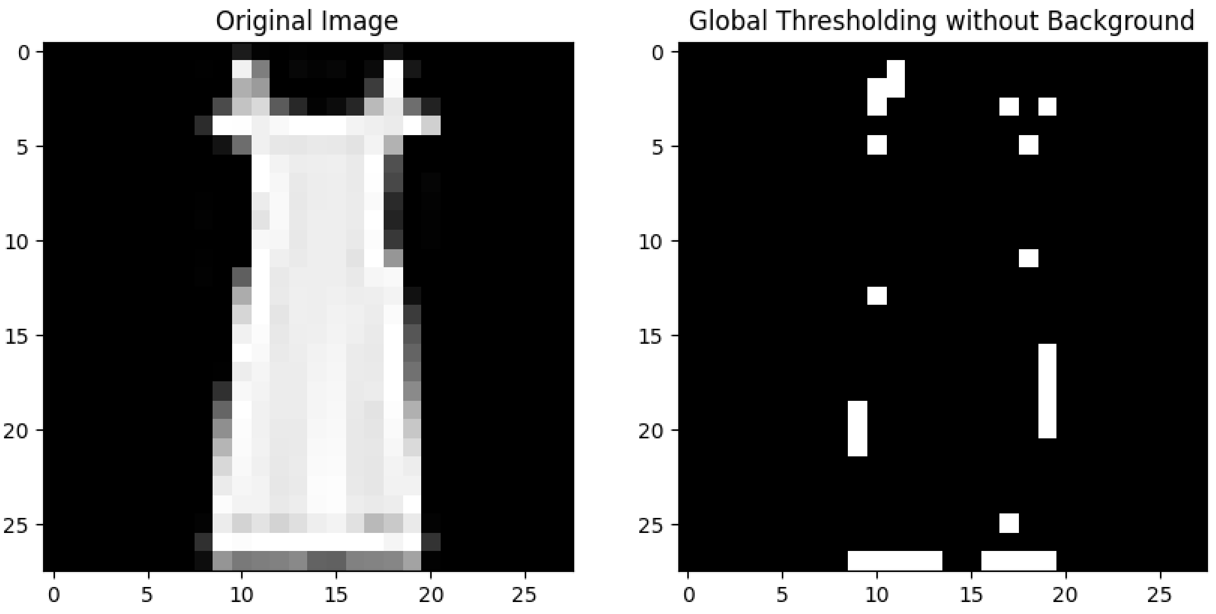
```
import matplotlib.pyplot as plt
import random

def thresholding(img, threshold=127):
    new_img = img.copy()
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i, j] > threshold:
                new_img[i, j] = 255
            else:
                new_img[i, j] = 0
    return new_img

threshold_values = [50, 100, 150, 200]  # List of thresholds to apply

for i in range(4):
    random_index = random.randint(0, len(train_images))

    fig, axs = plt.subplots(1, len(threshold_values) + 1, figsize=(15,
5))

    axs[0].imshow(train_images[random_index], cmap="gray")
    axs[0].set_title('Original Image')

    for j, threshold in enumerate(threshold_values):
        threshold_img = thresholding(train_images[random_index],
threshold)
        axs[j+1].imshow(threshold_img, cmap="gray")
        axs[j+1].set_title(f'Threshold: {threshold}')
    plt.show()
```

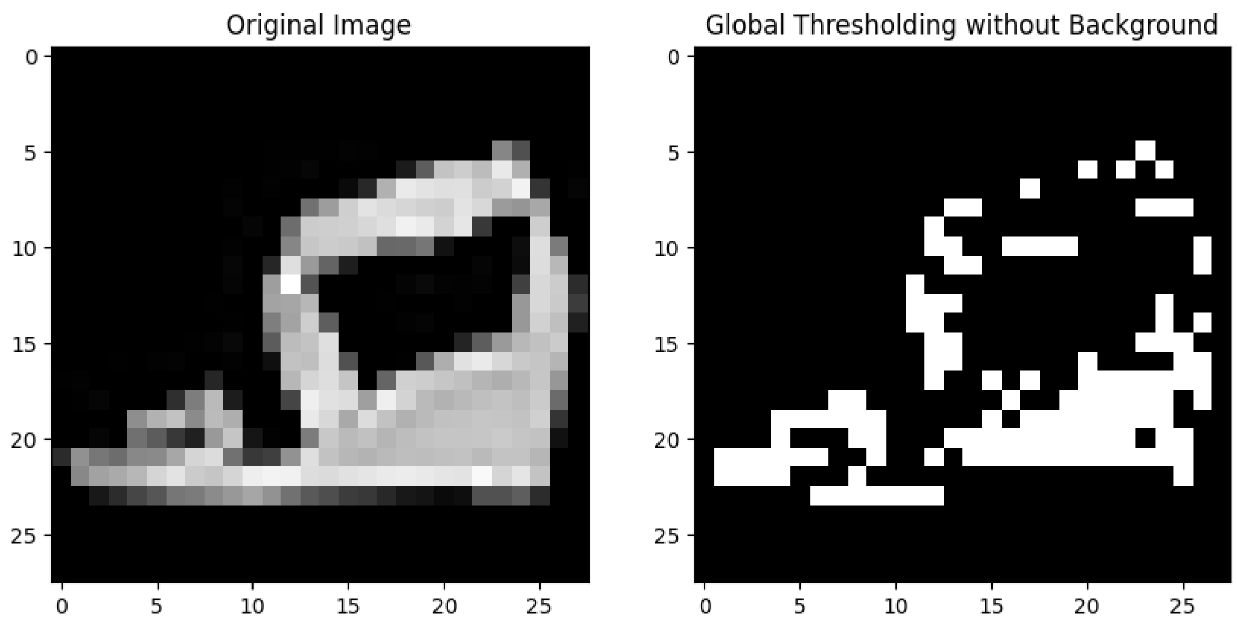


```
import matplotlib.pyplot as plt
import random

def gls_without_bg(img, a, b):
    neg = []
    for i  in range(len(img)):
        temp = []
        for j in range(len(img[i])):
            if(img[i][j] >= a and img[i][j] <= b):
                temp.append(255)
            else:
                temp.append(0)
        neg.append(temp)
    return  neg

for i in range(4):
    random_index = random.randint(0, len(train_images))
    gls_img = gls_without_bg(train_images[random_index], 100, 200)
    fig, axs = plt.subplots(1, 2, figsize=(10, 5))
    axs[0].imshow(train_images[random_index], cmap="gray")
    axs[0].set_title('Original  Image')
    axs[1].imshow(gls_img, cmap="gray") axs[1].set_title('Global
    Thresholding without Background')plt.show()
```



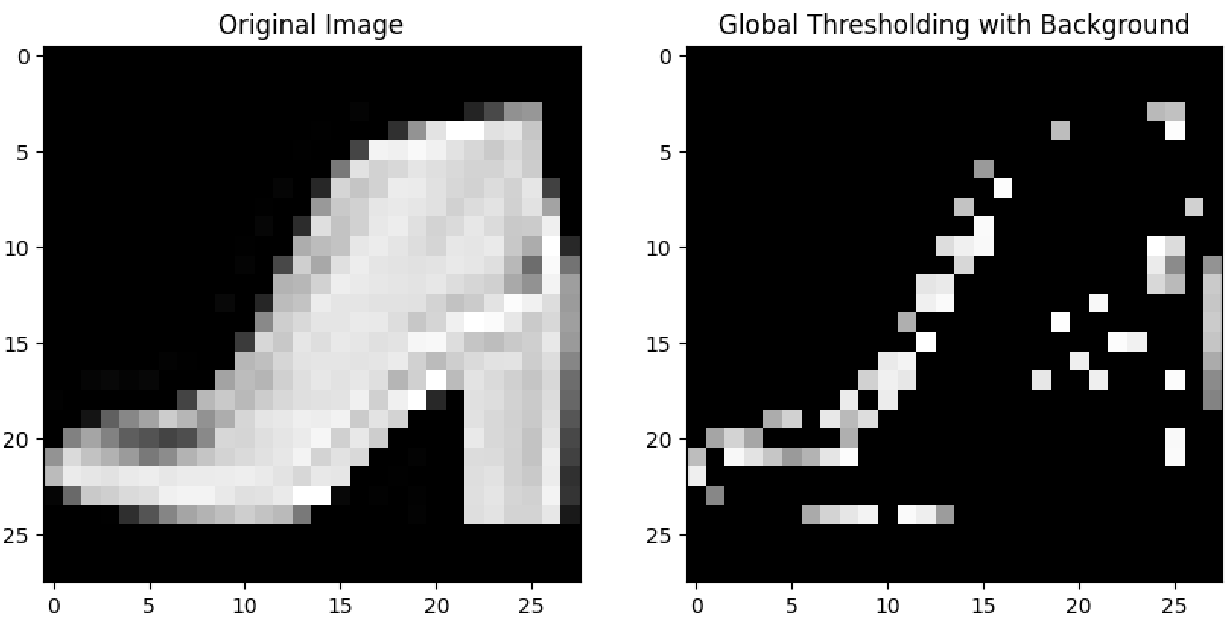

Original Image
Global Thresholding without Background
0
5
10
15
20
25

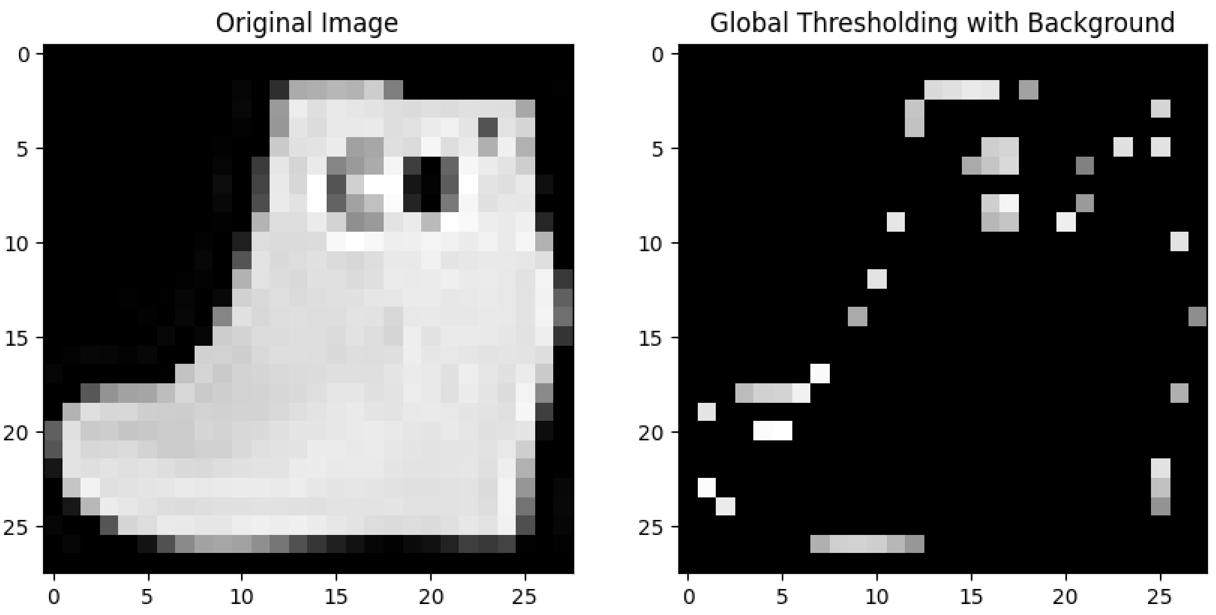

Original Image
Global Thresholding without Background
0
5
10
15
20
25

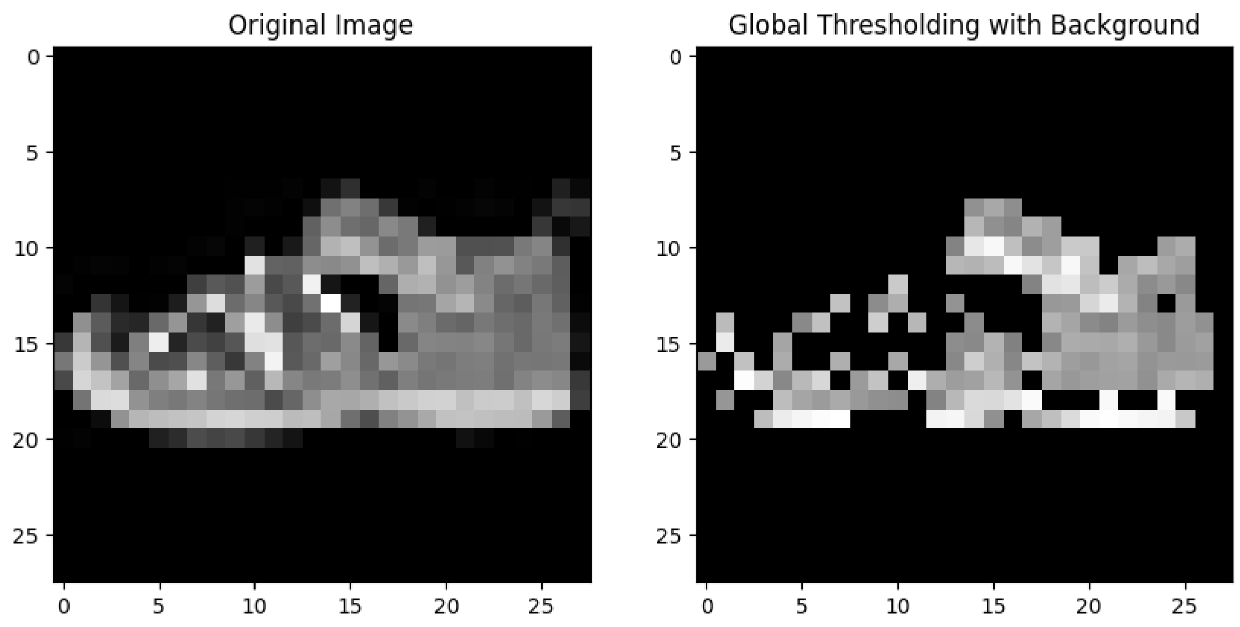


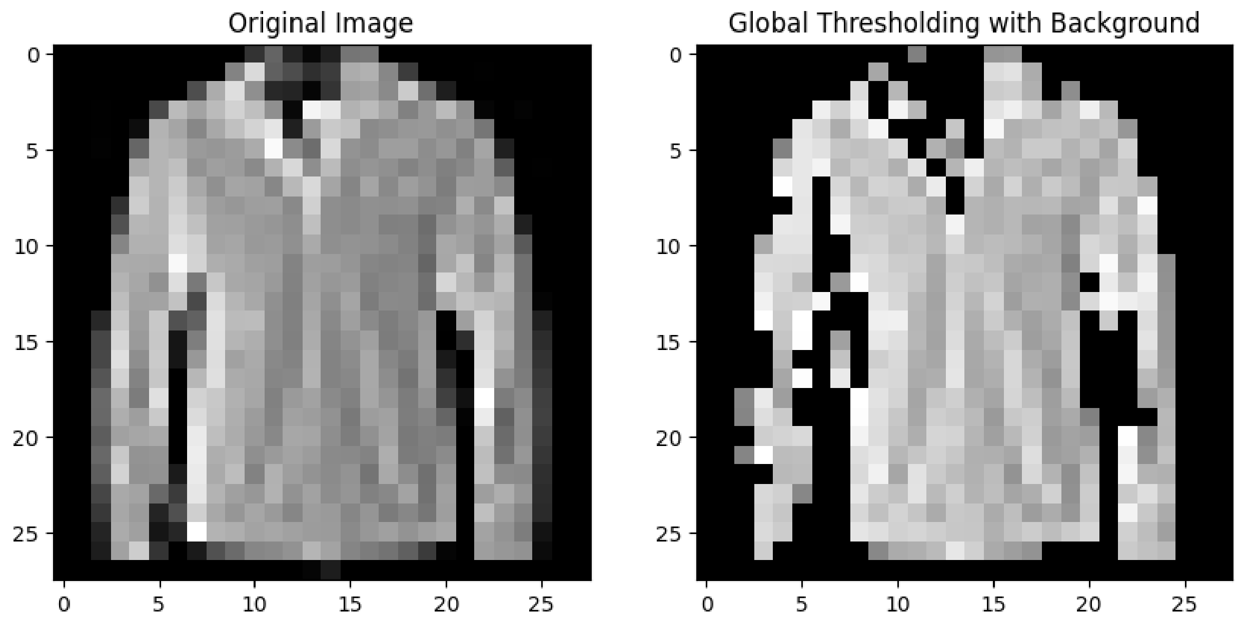
```
import matplotlib.pyplot as plt
import random

def gls_with_bg(img, a, b):
  neg = []
  for i in range(len(img)):
    temp = []
    for j in range(len(img[i])):
      if(img[i][j] >= a and img[i][j] <= b):
        temp.append(int(img[i][j]))
      else:
        temp.append(0)
    neg.append(temp)
  return neg

for i in range(4):
  random_index = random.randint(0, len(train_images))
  gls_img = gls_with_bg(train_images[random_index], 100, 200)
  fig, axs = plt.subplots(1, 2, figsize=(10, 5))
  axs[0].imshow(train_images[random_index], cmap="gray")
  axs[0].set_title('Original Image')
  axs[1].imshow(gls_img, cmap="gray")
  axs[1].set_title('Global Thresholding with Background')
  plt.show()
```

```
<ipython-input-9-cf3e184d64dc>:10: DeprecationWarning: Conversion of
an array with ndim > 0 to a scalar is deprecated, and will error in
future. Ensure you extract a single element from your array before
performing this operation. (Deprecated NumPy 1.25.)
  temp.append(int(img[i][j]))
```


Original Image
Global Thresholding with Background


Original Image
Global Thresholding with Background

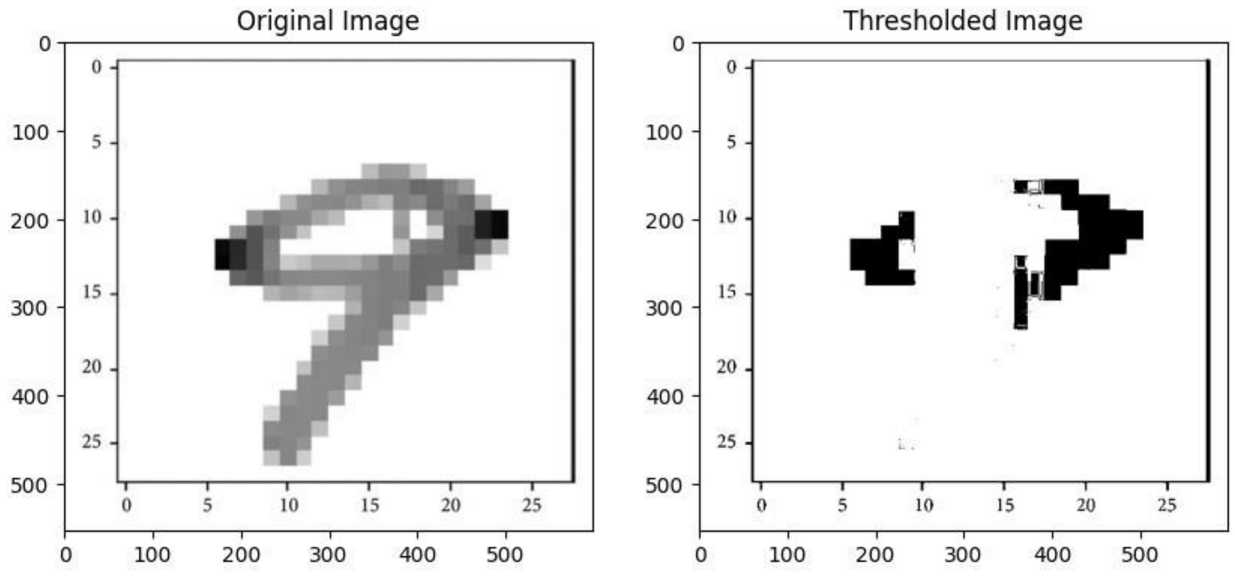




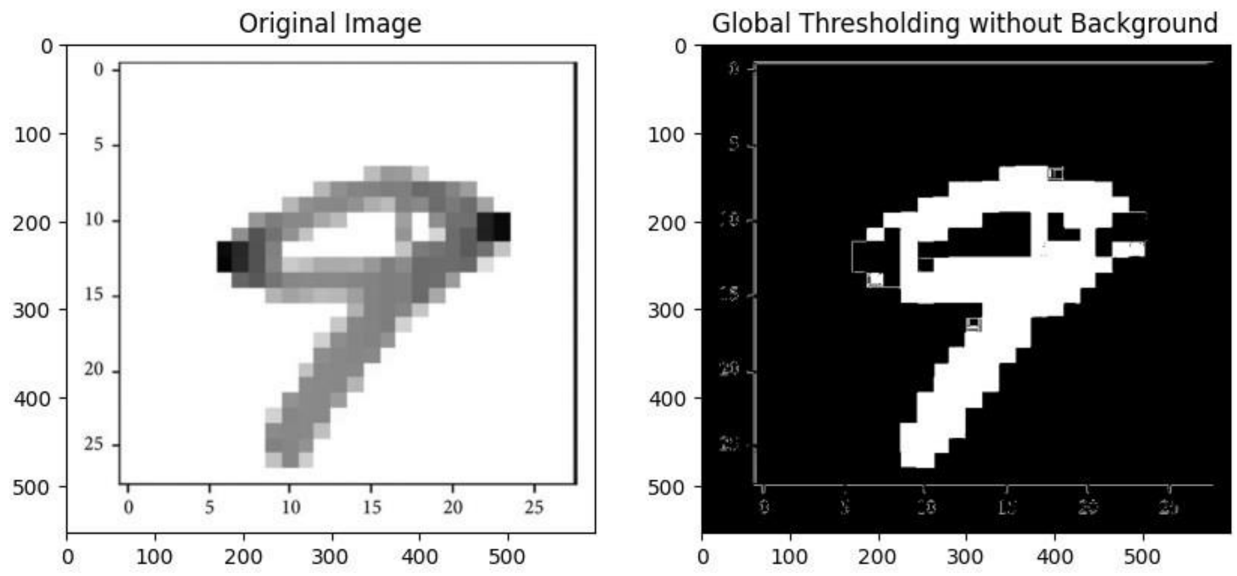
```
import cv2
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow


img = cv2.imread('/content/test_image.png')


cv2_imshow(img)
```

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


resized_gray = cv2.resize(gray, (28, 28))


fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axs[0].set_title('Original Image')
axs[1].imshow(resized_gray, cmap="gray")
axs[1].set_title('Resized Image')
plt.show()

---------------------------------------------------------------------------------------------------------------
---------
AttributeError                                    Traceback (most recent call
last)
<ipython-input-10-515321d4b0de> in <cell line: 9>()
      7
      8
----> 9 cv2_imshow(img)
     10
     11

/usr/local/lib/python3.10/dist-packages/google/colab/patches/__init__.
py in cv2_imshow(a)
     16       (N, M, 4) is an NxM BGRA color image.
     17   """
---> 18   a = a.clip(0, 255).astype('uint8')
     19   # cv2 stores colors as BGR; convert to RGB
     20   if a.ndim == 3:

AttributeError: 'NoneType' object has no attribute 'clip'

def digital_negative(img):
  neg=[]
  for i in range(len(img)):
    temp=[]
    for j in range(len(img[i])):
      temp.append(255-img[i][j])
    neg.append(temp)
  return neg


digital_neg = digital_negative(gray)


fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axs[0].set_title('Original Image')
axs[1].imshow(digital_neg, cmap="gray")
```

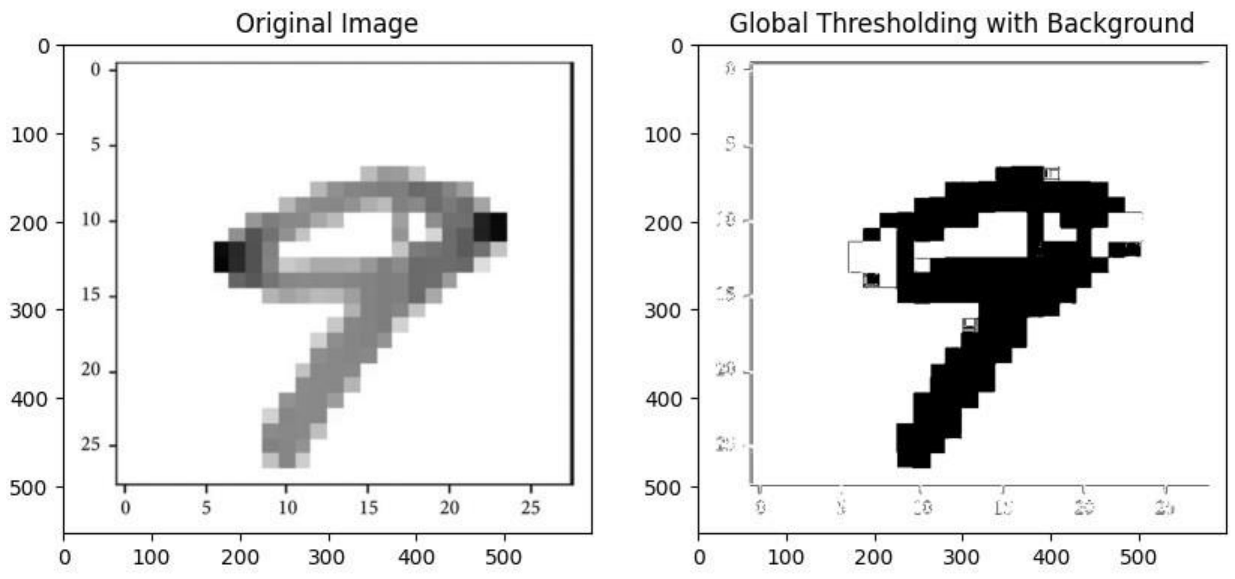
```
axs[1].set_title('Digital Negative')
plt.show()

def thresholding(img, threshold=127):
    new_img = img.copy()
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i, j] > threshold:
                new_img[i, j] = 255
            else:
                new_img[i, j] = 0
    return new_img

threshold_img = thresholding(gray)

fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axs[0].set_title('Original Image')
axs[1].imshow(threshold_img, cmap="gray")
axs[1].set_title('Thresholded Image')
plt.show()
```



```
def gls_without_bg(img, a, b):
    neg = []
    for i in range(len(img)):
        temp = []
        for j in range(len(img[i])):
            if(img[i][j] >= a and img[i][j] <= b):
                temp.append(255)
            else:
                temp.append(0)
        neg.append(temp)
    return neg

gls_img = gls_without_bg(gray, 100, 200)


fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
axs[0].set_title('Original Image')
axs[1].imshow(gls_img, cmap="gray")
axs[1].set_title('Global Thresholding without Background')
plt.show()
```



```
def gls_with_bg(img, a, b):
    neg = []
    for i in range(len(img)):
        temp = []
        for j in range(len(img[i])):
            if(img[i][j] >= a and img[i][j] <= b):
                temp.append(0)
            else:
                temp.append(255)
        neg.append(temp)
    return neg


gls_img = gls_with_bg(gray, 100, 200)


fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```



```
axs[0].set_title('Original   Image')
axs[1].imshow(gls_img, cmap="gray")
axs[1].set_title('Global Thresholding with Background')
plt.show()
```

0.4 Conclusion

0.4.1 In this lab, we explored various image processing techniques using the Fashion MNIST dataset, focusing on gray level slicing, thresholding, and image transformation. We developed Python programs to perform image negative transformation, thresholding at different levels, and gray level slicing with and without background. These techniques allowed us to enhance and manipulate image contrast, segment images into foreground and background, and selec tively highlight specific intensity ranges. The lab provided hands-on experience in understanding and implementing fundamental concepts in image processing, showcasing how these operations can be applied to real-world datasets without relying on built-in functions.