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)

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 1: Arithematic and Logical Operations on Image

SHIVAM NAGORI 60009210083 D1

Aim: To Perform Basic Image Processing Operations in Python.

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

1. Read random image(s) from the MNIST digits dataset given the digit value.
2. Display the before & after image(s) used in every task below.
3. Convert the image to grayscale.
4. Implement image cropping functionality.
5. Perform arithmetic operations on the images - Addition & Subtraction. 6. Implement logical operations on the image(s) - AND, OR, NOT & XOR.

Theory:

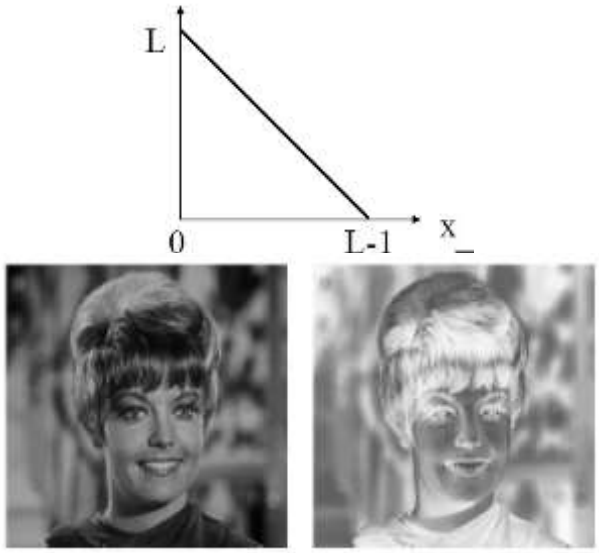
Grayscale Conversion:

Image Negatives: The negative of an image with gray levels in the range [0, L-1] is obtained by using the negative transformation shown in Fig.1.1, which is given by the expression $s = L - 1 - r$.

Reversing the intensity levels of an image in this manner produces the equivalent of a photographic negative. This type of processing is particularly suited for enhancing white or gray detail embedded in dark regions of an image, especially when the black areas are dominant in size

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)

$$s = T(r) = L - 1 - r$$



Arithmetic operations: pointwise addition: image + image (or constant)

Department of Computer Science and Engineering (Data Science)

Image Processing and Computer Vision I (DJ19DSL603)

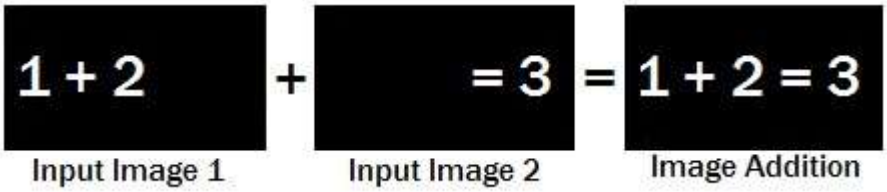
Lab 1: Arithematic and Logical Operations on Image

Arithmetic operations, specifically pointwise addition, involve adding corresponding pixel values of two images or adding a constant to every pixel value in an image. This operation is commonly used in image processing to adjust brightness, combine images, or perform other pixel-wise manipulations.

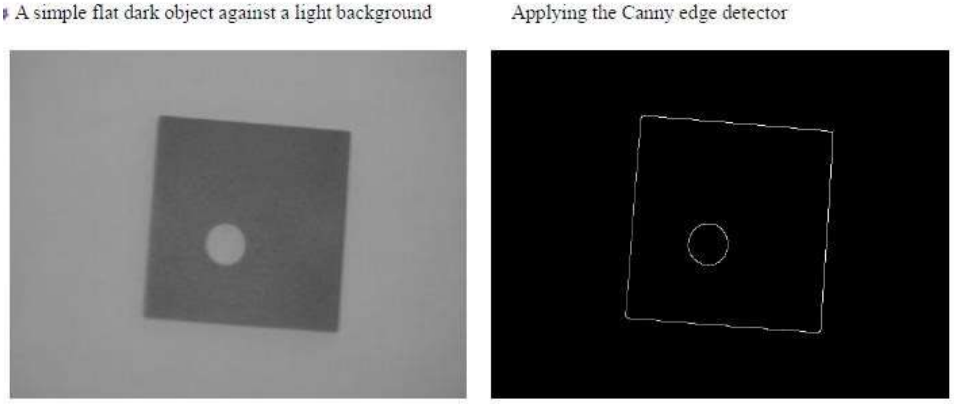
- **Pointwise Addition of Two Images:** For two images A and B of the same size, the pointwise addition $C=A+B$ is performed as follows: $Ci,j = Ai,j + Bi,j$ where Ci,j is the pixel value at position (i,j) in the resulting image C and Ai,j and Bi,j are the pixel values at the same position in images A and B respectively.
- **Pointwise Addition of a Constant to an Image:** For an image A and a constant k, the pointwise addition $B=A+k$ is performed as follows: $Bi,j = Ai,j + k$ where Bi,j is the pixel value at position (i,j) in the resulting image B and Ai,j is the pixel value at the same position in the original image A.

Pointwise addition is a fundamental operation in image processing and is used in various applications such as contrast adjustment, intensity scaling, and blending. It allows for the modification of pixel values based on corresponding elements, providing a versatile way to enhance or combine images.

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)



Addition of two images is performed straightforwardly in a single pass.
The output pixel values Or if it is simply desired to add a constant value C to a single image



Subtraction - pointwise subtraction: image - image (or constant)

Department of Computer Science and Engineering (Data Science)

Image Processing and Computer Vision I (DJ19DSL603)

Lab 1: Arithematic and Logical Operations on Image

Pointwise subtraction, also known as element-wise subtraction, is a mathematical operation applied to corresponding elements of two matrices or vectors. In the context of images, pointwise subtraction involves subtracting each pixel value of one image from the corresponding pixel value of another image.

For images A and B of the same size, the pointwise subtraction $C=A-B$ is performed as follows:

$$Ci,j = Ai,j - Bi,j$$

where Ci,j is the pixel value at position (i,j) in the resulting image C, Ai,j is the pixel value at the same position in image A, and Bi,j is the pixel value at the same position in image B.

Pointwise subtraction is commonly used in image processing for tasks such as computing the difference between two images, creating contrast-enhanced images, or isolating specific features. It is a

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)

straightforward operation that is applied independently to each pair of corresponding elements in the matrices involved.

Multiplication - pointwise multiplication: images * image (or constant)
Pointwise multiplication, often referred to as element-wise or Hadamard multiplication, is a mathematical operation applied to corresponding elements of two matrices or vectors. In the context of images, pointwise multiplication involves multiplying each pixel value of one image by the corresponding pixel value of another image or a constant.

For images A and B of the same size, the pointwise multiplication $C=A \odot B$ is performed as follows:

$$C_{i,j} = A_{i,j} \times B_{i,j}$$

where $C_{i,j}$ is the pixel value at position (i,j) in the resulting image C and $A_{i,j}$ and $B_{i,j}$ are the pixel values at the same position in images A and B respectively.

This operation is useful in various image processing tasks, such as blending, masking, and filtering. When a constant is involved, it is applied to every pixel in the image. Pointwise multiplication is computationally efficient and widely used in tasks where the interaction between corresponding elements is crucial for the desired outcome.

Division - pointwise division: images / image (or constant)
Pointwise division, also known as element-wise division, is a mathematical operation applied to corresponding elements of two matrices or vectors. In the context of images, pointwise division involves dividing each pixel value of one image by the corresponding pixel value of another image or by a constant.

For images A and B of the same size, the pointwise division $C=B/A$ is performed as follows:

$$C_{i,j} = B_{i,j} \ / \ A_{i,j}$$

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)

where $C_{i,j}$ is the pixel value at position (i,j) in the resulting image C, $A_{i,j}$ is the pixel value at the same position in image $A$and $B_{i,j}$ is the pixel value at the same position in image B.

When a constant is involved, it is divided by each pixel value in the image.

Pointwise division is used in various image processing applications, such as normalization, contrast adjustment, and scaling. It allows for the modification of pixel values based on corresponding elements, providing a flexible way to control the intensity or relationship between pixels in images.

Blending - pointwise linear combination of two images
Blending, in the context of image processing, involves combining two images to create a new image. A common method for blending is the pointwise linear combination, where each pixel value in the resulting image is obtained through a linear combination of the corresponding pixel values in the source images.

For images A and B of the same size, the pointwise linear combination C is calculated as follows:

$$C_{i,j} = \alpha A_{i,j} + (1-\alpha) B_{i,j}$$

where $C_{i,j}$ is the pixel value at position (i,j) in the resulting blended image C, $A_{i,j}$ and $B_{i,j}$ are the pixel values at the same position in images A and B respectively, and α is a blending parameter between 0 and 1. The parameter α determines the weight given to each image in the blend: a higher α gives more weight to image A, while a lower α gives more weight to image B.

Pointwise linear combination is commonly used for tasks such as image fading, cross-dissolving, and creating smooth transitions between images. It provides a simple yet effective way to visually merge two images by controlling the influence of each pixel in the final result.

Department of Computer Science and Engineering (Data Science)

Image Processing and Computer Vision I (DJ19DSL603)

Lab 1: Arithematic and Logical Operations on Image

Logical AND/NAND - pointwise logical ANDing/NANDing of 2 binary images
Logical AND and NAND operations are binary operations commonly used in image processing to combine information from two binary images pixel-wise. In these operations, each pixel value in the resulting image depends on the corresponding pixel values in the input images.

- **Logical AND Operation:** For two binary images A and B, the logical AND operation $C=A \wedge B$ is performed as follows: $C_{i,j} = A_{i,j} \wedge B_{i,j}$ where $C_{i,j}$ is the pixel value at position (i,j) in the

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)

resulting image C, and Ai,j and Bi,j are the pixel values at the same position in images A and B respectively.

The resulting image C will have pixel values of 1 only where both A and B have pixel values of 1, otherwise, it will be 0.

- **Logical NAND Operation:** For two binary images A and B, the logical NAND operation $C=A\uparrow B$ is performed as follows: $Ci,j =\neg(Ai,j \wedge Bi,j)$ where Ci,j is the pixel value at position (i,j) in the resulting image C, and Ai,j and Bi,j are the pixel values at the same position in images A and B respectively.

The resulting image C will have pixel values of 0 only where both A and B have pixel values of 1, otherwise, it will be 1.

These logical operations are useful in image segmentation, where you might want to extract regions that are common (AND) or not common (NAND) between two binary masks or images.

Logical OR/NOR - pointwise logical ORing/NORing of two binary images
Logical OR and NOR operations are binary operations commonly used in image processing to combine information from two binary images pixel-wise. In these operations, each pixel value in the resulting image depends on the corresponding pixel values in the input images.

- **Logical OR Operation:** For two binary images A and B, the logical OR operation $C=A\vee B$ is performed as follows: $Ci,j =Ai,j \vee Bi,j$ where Ci,j is the pixel value at position (i,j) in the resulting image C and Ai,j and Bi,j are the pixel values at the same position in images A and B respectively.

The resulting image C will have pixel values of 1 where either A or B (or both) have pixel values of 1.

- **Logical NOR Operation:** For two binary images A and B, the logical NOR operation $C=A\downarrow B$ is performed as follows: $Ci,j =\neg(Ai,j \vee Bi,j)$ where Ci,j is the pixel value at position (i,j) in the resulting image C and Ai,j and Bi,j are the pixel values at the same position in images A and B respectively.

The resulting image C will have pixel values of 0 only where neither A nor B have a pixel value of 1.

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)

These logical operations are useful in image processing for tasks such as image fusion, where you might want to combine information from multiple sources based on the presence or absence of certain features.

Logical XOR/XNOR - pointwise logical XORing/XNORing of two binary images
Logical XOR (exclusive OR) and XNOR (exclusive NOR) operations are binary operations used in image processing to combine information from two binary images pixel-wise. In these operations, each pixel value in the resulting image depends on the corresponding pixel values in the input images.

- **Logical XOR Operation:** For two binary images A and B, the logical XOR operation $C=A \oplus B$ is performed as follows: $Ci,j = Ai,j \oplus Bi,j$ where Ci,j is the pixel value at position (i,j) in the resulting image C, and Ai,j and Bi,j are the pixel values at the same position in images A and B respectively.

The resulting image C will have pixel values of 1 where either A or B (but not both) have a pixel value of 1.

- **Logical XNOR Operation:** For two binary images A and B, the logical XNOR operation $C=A \odot B$ is performed as follows: $Ci,j = \neg(Ai,j \oplus Bi,j)$ where Ci,j is the pixel value at position (i,j) in the resulting image C and Ai,j and Bi,j are the pixel values at the same position in images A and B respectively.

The resulting image C will have pixel values of 1 where either both A and B or neither have a pixel value of 1.

These logical operations are useful in image processing for tasks such as feature extraction, where you might want to isolate regions where the binary images differ or are similar based on pixel values.

Invert/Logical NOT - pointwise inversion of a binary image, Inverting a binary image, also known as applying the logical NOT operation, involves changing each pixel value in the image to its complement. If a pixel in the original image has a value of 0, it becomes 1 in the inverted image, and if it has a value of 1, it becomes 0.

For a binary image A, the inverted image B is obtained using the logical NOT operation as follows:

$$Bi,j = \neg Ai,j$$

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)

Lab 1: Arithematic and Logical Operations on Image

where $B_{i,j}$ is the pixel value at position (i,j) in the resulting inverted image B and $A_{i,j}$ is the pixel value at the same position in the original image A.

This operation is useful in various image processing tasks, such as creating a negative of an image, isolating the background, or preparing images for certain computer vision algorithms. It simply flips the values of pixels in a binary image, turning 0s to 1s and vice versa.

Bitshift Operators - pointwise scaling of an image, Bitshift operators, commonly denoted as << (left shift) and >> (right shift), can be used for pointwise scaling of pixel values in an image. These operators perform bitwise shifts on the binary representation of numbers. In the context of image processing, you can use them to scale the pixel values by shifting their binary representation to the left or right.

For a grayscale image A, the pointwise scaling can be performed using bitshift operators as follows:

- **Left Shift (<<):** $B_{i,j} = A_{i,j} << n$ where $B_{i,j}$ is the pixel value at position (i,j) in the resulting image B $A_{i,j}$ is the pixel value at the same position in the original image A, and n is the number of bits to shift to the left. This operation effectively multiplies the pixel values by n.
- **Right Shift (>>):** $B_{i,j} = A_{i,j} >> n$ where $B_{i,j}$ is the pixel value at position (i,j) in the resulting image B, $A_{i,j}$ is the pixel value at the same position in the original image A, and n is the number of bits to shift to the right. This operation effectively divides the pixel values by n.

It's important to note that bitshift operations may result in loss of information, especially when shifting to the right, as the fractional part of pixel values might be truncated.

This type of scaling is a simple and computationally efficient way to adjust the intensity values of an image. However, caution should be exercised to ensure that the bitshift operations do not cause pixel values to overflow or underflow, which may lead to unintended artifacts in the image.

Lab Assignments to complete in this session

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

1. Read random image(s) from the MNIST digits dataset given the digit value. **Dataset Link:** MNIST Digits Kaggle

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)

2. Display the before & after image(s) used in every task below.
3. Convert the image to grayscale.

Department of Computer Science and Engineering (Data Science)

Image Processing and Computer Vision I (DJ19DSL603)

Lab 1: Arithematic and Logical Operations on Image

4. Implement image cropping functionality.
5. Perform arithmetic Addition operations on the images
6. Perform arithmetic Subtraction operations on the images
7. Perform arithmetic Multiplication operations on the images
8. Perform arithmetic Division operations on the images
9. Perform arithmetic blending operations on the images
10. Implement AND logical operations on the image(s) 11. Implement OR logical operations on the image(s)
12. Implement NOT logical operations on the image(s)
13. Implement XORlogical operations on the image(s)
14. Implement bitshift leftshift logical operations on the image(s) 15. Implement bitshift Rightshift logical operations on the image(s)

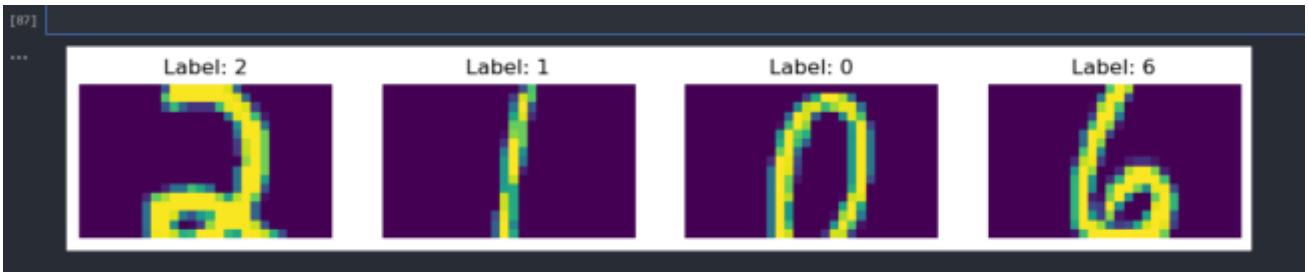
The solution to the operations performed must be produced by scratch coding without the use of built in OpenCV methods.

SOLUTION :

CROPPING THE IMAGE :

```
plt.figure(figsize=(15, 3))  # Adjust the figure size as needed

for i, index in enumerate(random_indices, 1):
    plt.subplot(1, 5, i)
    resized_image = train_images[index][4:21]
    plt.imshow(resized_image,)
    plt.title(f"Label: {train_labels[index]}")
    plt.axis('off')

plt.show()
```

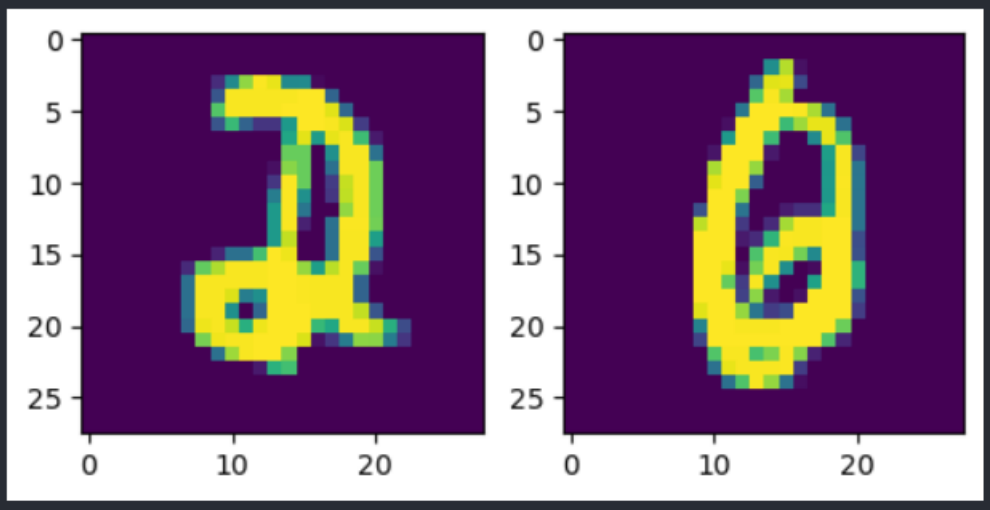
Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)



ARITHMATIC ADDITION OF IMAGE :

```
plt.figure(figsize= (15,3))

plt.subplot(1,5,1)
added_image1 = cv2.add(train_images[random_indices[0]],
train_images[random_indices[1]])
plt.imshow(added_image1,)

plt.subplot(1,5,2)
added_image2 = cv2.add(train_images[random_indices[2]],
train_images[random_indices[3]])
plt.imshow(added_image2,)
```

```
<matplotlib.image.AxesImage at 0x22d62554e50>
```



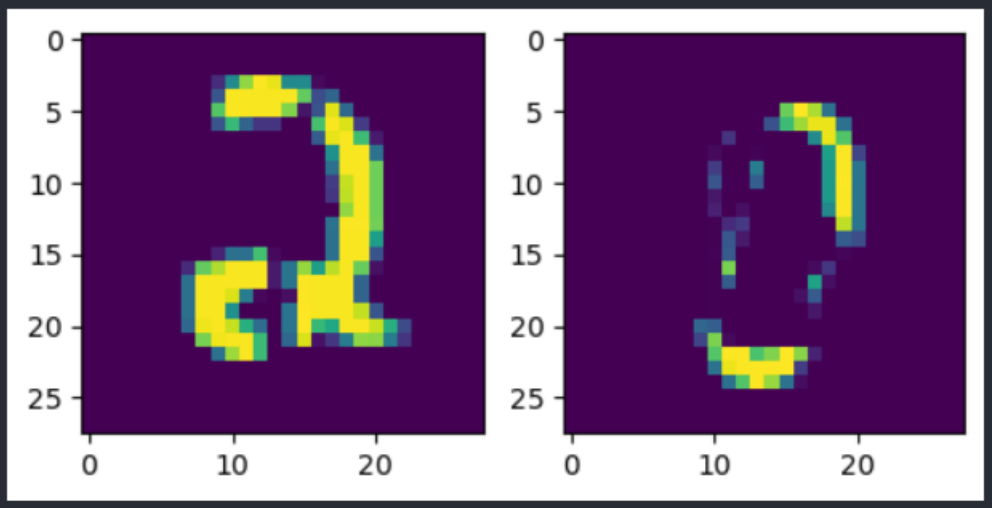
ARITHMATIC SUBTRACTION OF IMAGE :

```
plt.figure(figsize= (15,3))

plt.subplot(1,5,1)
```

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)

```
subtract_image1 = cv2.subtract(train_images[random_indices[0]],
train_images[random_indices[1]])
plt.imshow(subtract_image1,)

plt.subplot(1,5,2)
subtract_image2 = cv2.subtract(train_images[random_indices[2]],
train_images[random_indices[3]])
plt.imshow(subtract_image2,)
```

```
<matplotlib.image.AxesImage at 0x22d6220e050>
```

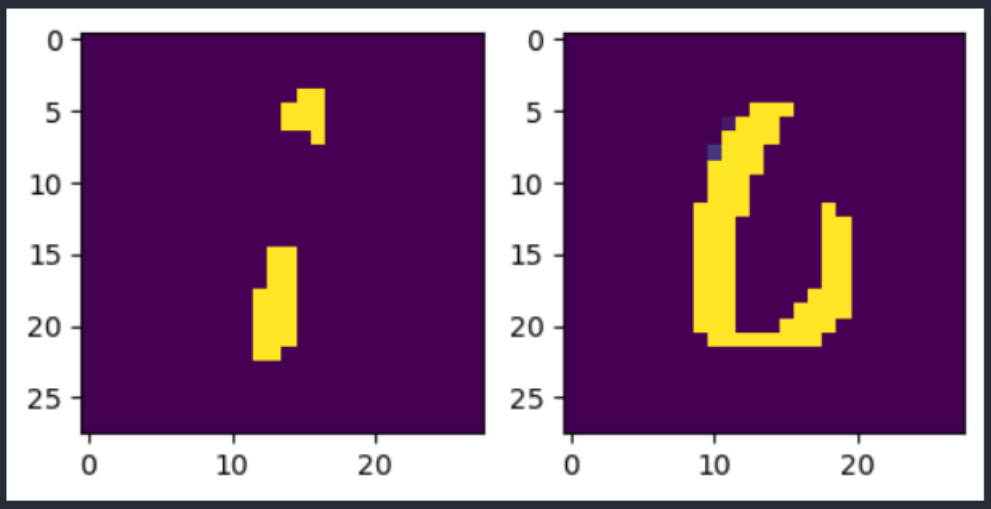


ARITHMATIC MULTIPLICATION OF IMAGE :

```
plt.figure(figsize= (15,3))

plt.subplot(1,5,1)
mul_image1 = cv2.multiply(train_images[random_indices[0]],
train_images[random_indices[1]])
plt.imshow(mul_image1,)

plt.subplot(1,5,2)
mul_image2 = cv2.multiply(train_images[random_indices[2]],
train_images[random_indices[3]])
plt.imshow(mul_image2,)
```

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)

```
<matplotlib.image.AxesImage at 0x22d62198950>
```

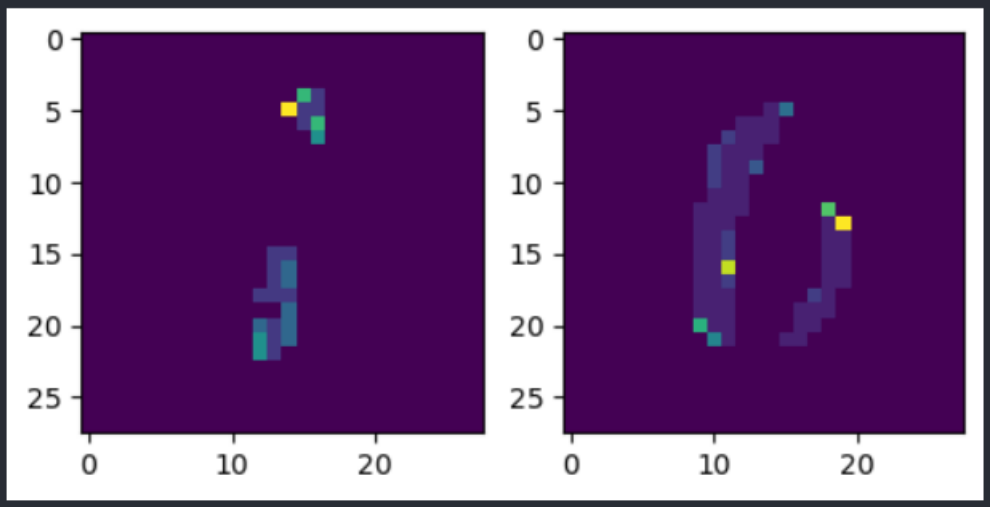


ARITHMATIC DIVISION OF IMAGE :

```
plt.figure(figsize= (15,3))

plt.subplot(1,5,1)
div_image1 = cv2.divide(train_images[random_indices[0]],
train_images[random_indices[1]])
plt.imshow(div_image1,)

plt.subplot(1,5,2)
div_image2 = cv2.divide(train_images[random_indices[2]],
train_images[random_indices[3]])
plt.imshow(div_image2,)
```

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)

```
<matplotlib.image.AxesImage at 0x22d6237d890>
```

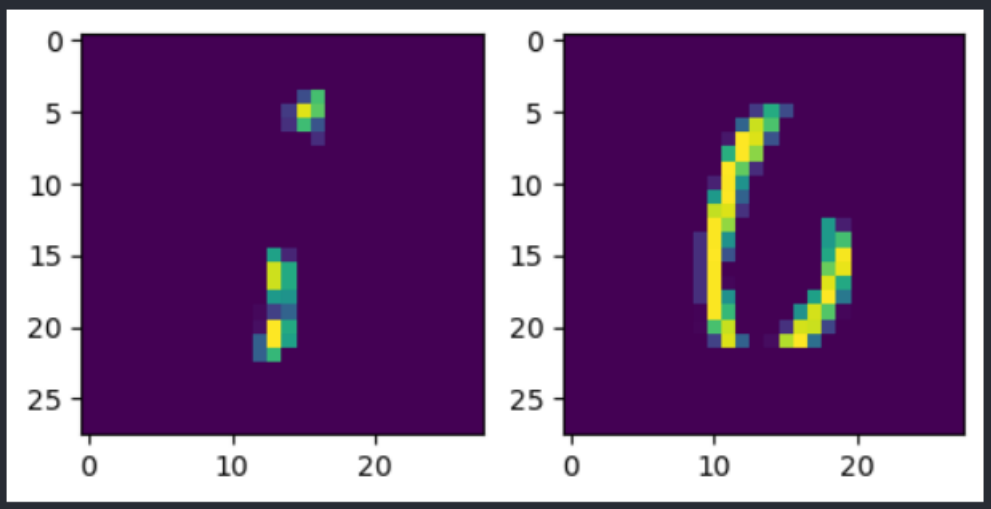


LOGICAL AND :

```
plt.figure(figsize= (15,3))

plt.subplot(1,5,1)
and_image1 = cv2.bitwise_and(train_images[random_indices[0]],
train_images[random_indices[1]])
plt.imshow(and_image1,)

plt.subplot(1,5,2)
and_image2 = cv2.bitwise_and(train_images[random_indices[2]],
train_images[random_indices[3]])
plt.imshow(and_image2,)
```

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)

```
<matplotlib.image.AxesImage at 0x22d62024e50>
```

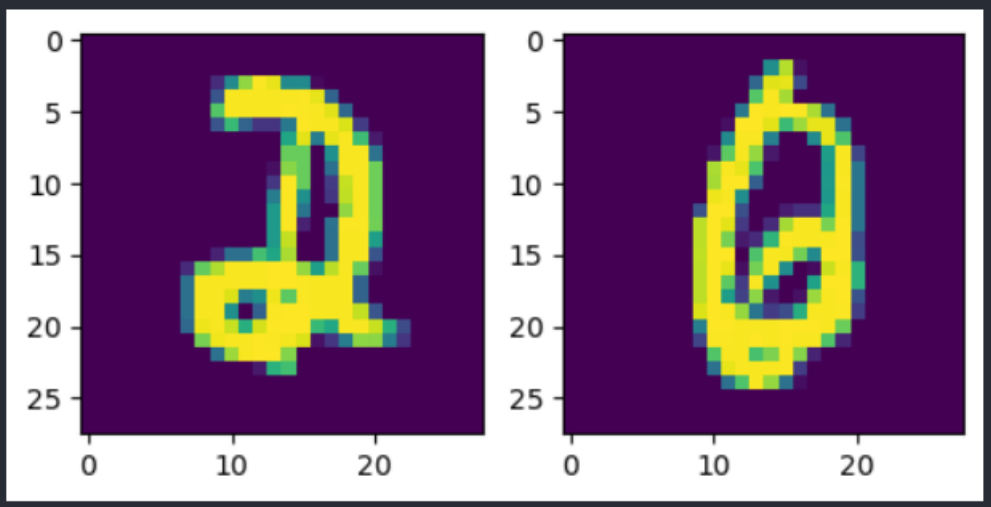


LOGICAL OR :

```
plt.figure(figsize= (15,3))

plt.subplot(1,5,1)
OR_image1 = cv2.bitwise_or(train_images[random_indices[0]],
train_images[random_indices[1]])
plt.imshow(OR_image1)

plt.subplot(1,5,2)
OR_image2 = cv2.bitwise_or(train_images[random_indices[2]],
train_images[random_indices[3]])
plt.imshow(OR_image2,)
```

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)

```
<matplotlib.image.AxesImage at 0x22d61de4790>
```

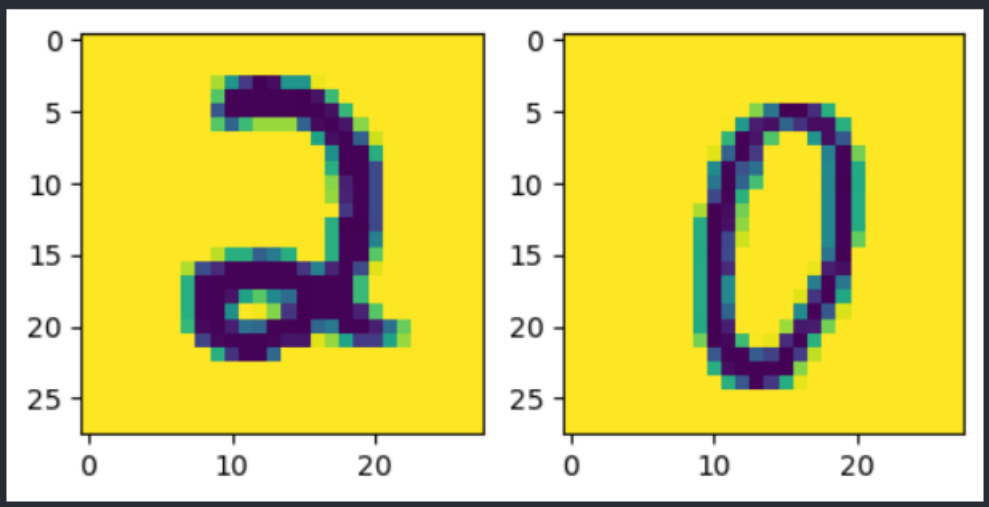


LOGICAL NOR OF IMAGE ;

```
plt.figure(figsize= (15,3))

plt.subplot(1,5,1)
NOT_image1 = cv2.bitwise_not(train_images[random_indices[0]],
train_images[random_indices[1]])
plt.imshow(NOT_image1,)

plt.subplot(1,5,2)
NOT_image2 = cv2.bitwise_not(train_images[random_indices[2]],
train_images[random_indices[3]])
plt.imshow(NOT_image2,)
```

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)

```
<matplotlib.image.AxesImage at 0x22d662cabd0>
```

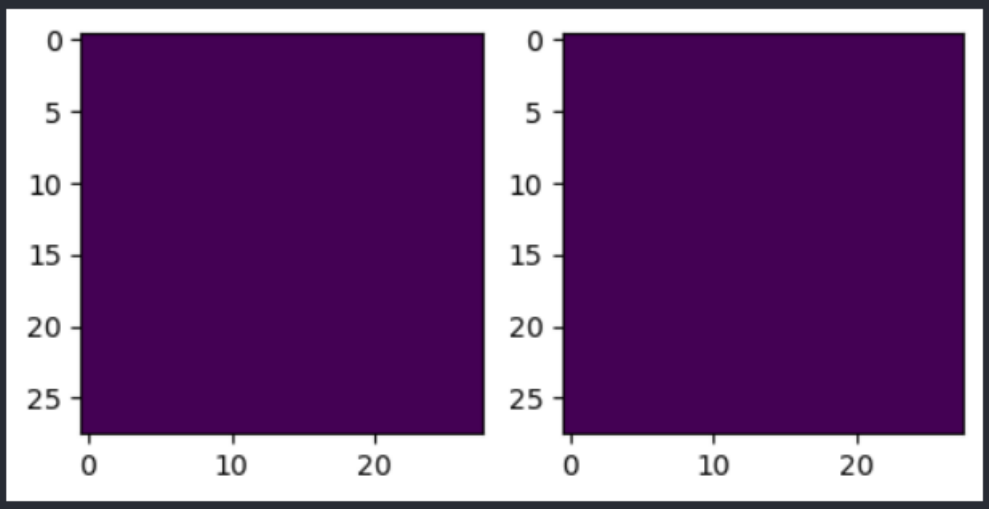


LOGICAL XOR :

```
plt.figure(figsize= (15,3))

plt.subplot(1,5,1)
xor_image1 = cv2.bitwise_xor(train_images[random_indices[0]],
train_images[random_indices[1]])
plt.imshow(xor_image1,)

plt.subplot(1,5,2)
xor_image2 = cv2.bitwise_xor(train_images[random_indices[2]],
train_images[random_indices[3]])
plt.imshow(xor_image2,)
```

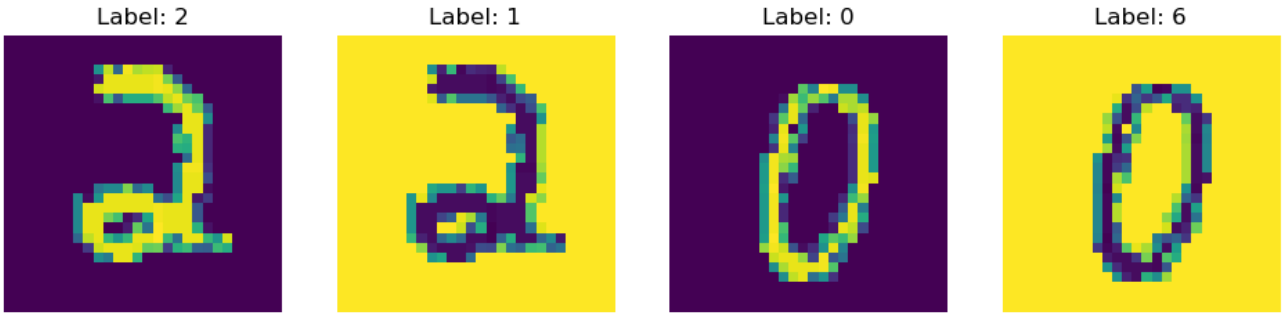
Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

```
<matplotlib.image.AxesImage at 0x22d66653650>
```



BITSHIFT LEFT :

```
plt.figure(figsize=(15, 3))  # Adjust the figure size as needed

for i, index in enumerate(random_indices, 1):
    plt.subplot(1, 5, i)
    plt.imshow(np.left_shift(train_images[index] ,2), )
    plt.title(f"Label: {train_labels[index]}")
    plt.axis('off')

plt.show()
```



BITSHIFT RIGHT :

```
plt.figure(figsize=(15, 3))  # Adjust the figure size as needed
```

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)

```
for i, index in enumerate(random_indices, 1):
    plt.subplot(1, 5, i)
    plt.imshow(np.right_shift(train_images[index] ,2),)
    plt.title(f"Label: {train_labels[index]}")
    plt.axis('off')

plt.show()
```