AD18511 – DEEP LEARNING LABORATORY

DATE:

EX.NO: 6(a) **EDGE DETECTION.**

AIM:

To perform edge detection for an image using three algorithms canny, sobel and prewitt using a CNN.

DESCRIPTION:

Edge Detection is a method of segmenting an image into regions of discontinuity. It is a widely used technique in digital image processing

Edge Detection Operators are of two types:

- Gradient – based operator which computes first-order derivations in a digital image like, Sobel operator, Prewitt operator, Robert operator
- Gaussian – based operator which computes second-order derivations in a digital image like, Canny edge detector, Laplacian of Gaussian

Sobel Operator: It is a discrete differentiation operator. It computes the gradient approximation of image intensity function for image edge detection. At the pixels of an image, the Sobel operator produces either the normal to a vector or the corresponding gradient vector. It uses two 3 x 3 kernels or masks which are convoluted with the input image to calculate the vertical and horizontal derivative approximations respectively –

Prewitt Operator: This operator is almost similar to the sobel operator. It also detects vertical and horizontal edges of an image. It is one of the best ways to detect the orientation and magnitude of an image. It uses the kernels or masks –

Canny Operator: It is a Gaussian-based operator in detecting edges. This operator is not susceptible to noise. It extracts image features without affecting or altering the feature. Canny edge detector have advanced algorithm derived from the previous work of Laplacian of Gaussian operator. It is widely used an optimal edge detection technique.

PROGRAM:

```
pip install opencv-python matplotlib numpy

import cv2
import numpy as np
import matplotlib.pyplot as plt
image=cv2.imread("/content/dog.jpg")
plt.imshow(image)

#convert it to grayscale
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
#show the grayscale image
plt.imshow(gray,cmap="gray")
plt.show()

blur=cv2.GaussianBlur(gray,(5,5),0)
```

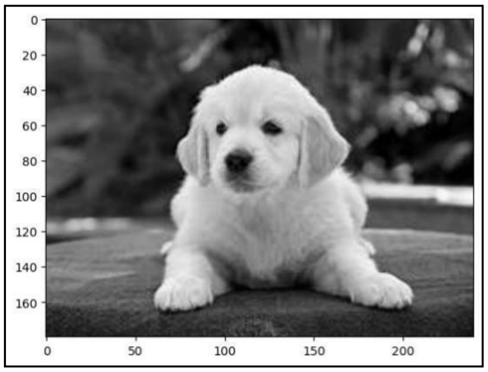
```
plt.imshow(blur,cmap="gray")
plt.show()

#perform the canny edge detector to detect image edges
edges=cv2.Canny(gray,threshold1=30,threshold2=100)#blur or gray or image

plt.imshow(edges,cmap="gray")
plt.show()

#perform the canny edge detector to detect image edges
edges=cv2.Canny(blur,threshold1=30,threshold2=100)#blur or gray or imag

plt.imshow(edges,cmap="gray")
plt.show()

#perform the canny edge detector to detect image edges
edges=cv2.Canny(image,threshold1=30,threshold2=100)#blur or gray or image
plt.imshow(edges,cmap="gray")
plt.show()

#Sobel operator
#Read the original image
img=cv2.imread("/home/user/Desktop/tiger.jpeg")
#display
cv2.imshow('Original',img)
cv2.waitKey(0)
#Convert to gray scale
img_gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
#Blur the image for better edge detection
img_blur=cv2.GaussianBlur(img_gray,(3,3),0)

#Sobel Edge Detection
sobelx=cv2.Sobel(src=img_blur,ddepth=cv2.CV_64F,dx=1,dy=0,ksize=5)
sobely=cv2.Sobel(src=img_blur,ddepth=cv2.CV_64F,dx=0,dy=1,ksize=5)
sobelxy=cv2.Sobel(src=img_blur,ddepth=cv2.CV_64F,dx=1,dy=1,ksize=5)

#Display Sobel Edge Detetcion Images
cv2.imshow('Sobel X',sobelx)
cv2.waitKey(0)
cv2.imshow('Sobel Y',sobely)
cv2.waitKey(0)
cv2.imshow('Sobel XY using Sobel() function',sobelxy)
cv2.waitKey(0)

#cv2.destroyAllWindows()

#Display Sobel Edge Detetcion Images
cv2.imshow('Sobel X',sobelx)
cv2.waitKey(0)
cv2.imshow('Sobel Y',sobely)
cv2.waitKey(0)
cv2.imshow('Sobel XY using Sobel() function',sobelxy)
cv2.waitKey(0)
```

```
# Prewitt
# Defining the kernels
kernelx = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]])
kernely = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])

# Applying convolution
img_prewittx = cv2.filter2D(blur, -1, kernelx)
img_prewitty = cv2.filter2D(blur, -1, kernely)
img_prewitt = img_prewittx + img_prewitty

fig, axs = plt.subplots(2, 2)
titles = ['Original', 'Kernel X', 'Kernel Y', 'Kernel X and Y']
axs[0, 0].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
axs[0, 0].set_title(titles[0])
axs[0, 0].axis('off')
axs[0, 1].imshow(img_prewitt, cmap='gray')
axs[0, 1].set_title(titles[3])
axs[0, 1].axis('off')
axs[1, 0].imshow(img_prewittx, cmap='gray')
axs[1, 0].set_title(titles[1])
axs[1, 0].axis('off')
axs[1, 1].imshow(img_prewitty, cmap='gray')
axs[1, 1].set_title(titles[2])
axs[1, 1].axis('off')
```

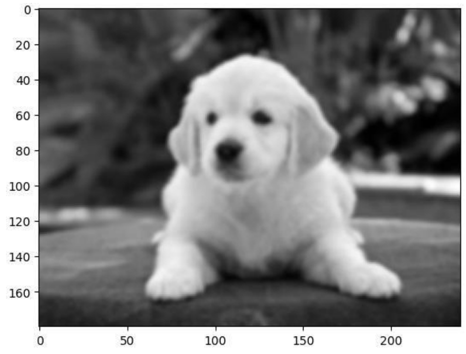
OUTPUT:

Original

Grayscale

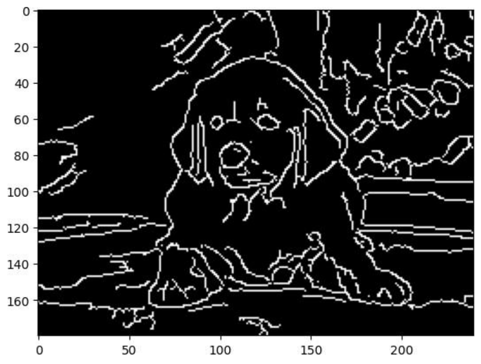


Blur

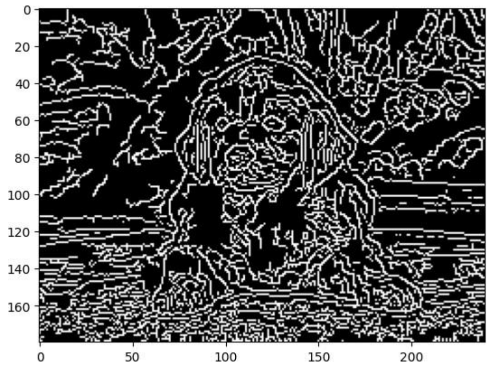


Canny edge detector

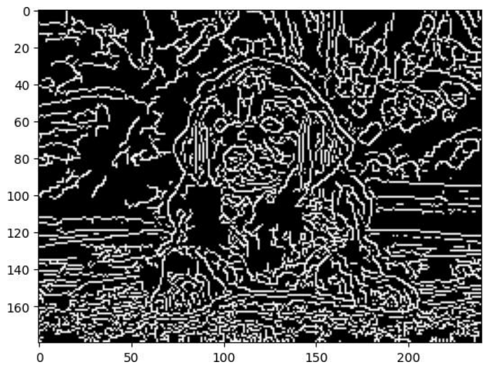
Blur



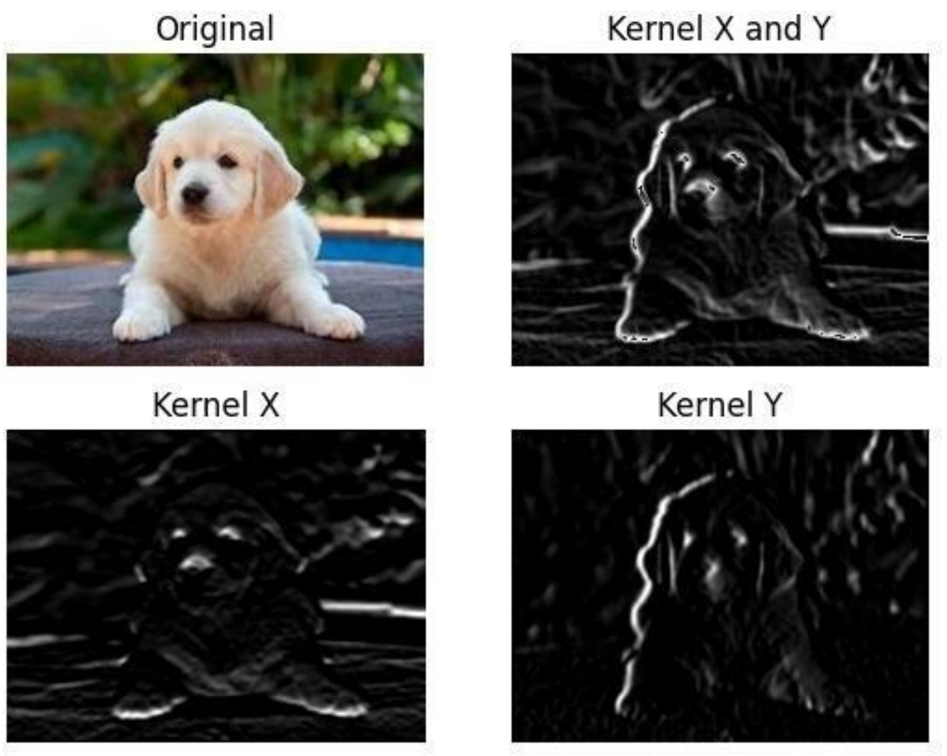
Image



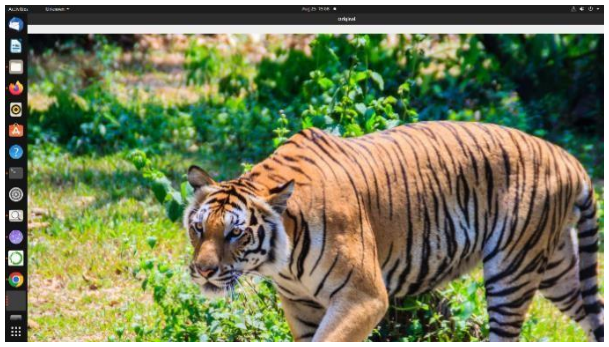
Image

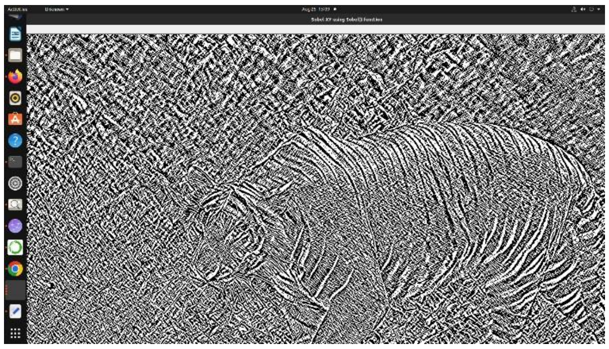


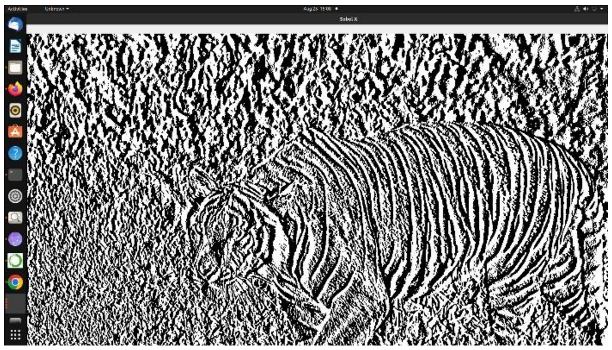
Prewitt:

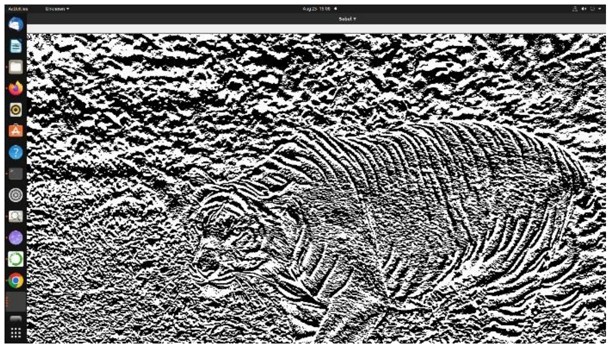


Sobel:









RESULT:

The image is loaded and edge detection is done using the 3 algorithms with a CNN successfully.

DATE:

EX.NO: 6(b) **EDGE DETECTION.**

AIM:

To implement edge detection for an image using Laplacian and Scharr operator.

DESCRIPTION:

Edge Detection is a method of segmenting an image into regions of discontinuity. It is a widely used technique in digital image processing.

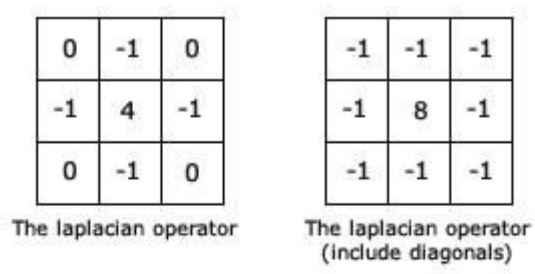
Sch-arr operator: This is a filtering method used to identify and highlight gradient edges/features using the first derivative. Performance is quite similar to the Sobel filter.

Scharr Operator [X-axis] = [-3 0 3; -10 0 10; -3 0 3];

Scharr Operator [Y-axis] = [3 10 3; 0 0 0; -3 -10 -3];

Laplacian operator: Unlike the Sobel edge detector, the Laplacian edge detector uses only one kernel.

It calculates second order derivatives in a single pass. Here's the kernel used for it:



0	-1	0
-1	4	-1
0	-1	0

The laplacian operator

-1	-1	-1
-1	8	-1
-1	-1	-1

The laplacian operator (include diagonals)

PROGRAM:

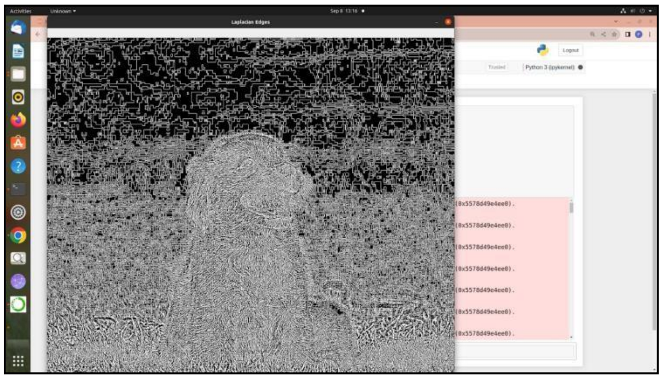
```
#Laplacian

import cv2

#load an image
image=cv2.imread('/home/user/Desktop/pro.jpg',cv2.IMREAD_GRAYSCALE)

#Apply Laplacian edge detection
edges=cv2.Laplacian(image,cv2.CV_64F)

#Display the resulting edges
cv2.imshow('Laplacian Edges',edges)
cv2.waitKey(0)
cv2.DestroyAllWindows()
```

OUTPUT:

```
shape of features matrix: (60000, 28, 28)
shape of target matrix: (60000,)
```

```
fig,ax=plt.subplots(10,10)
for i in range(10):
    for j in range(10):
    k=np.random.randint(0,X_train.shape[0])
    ax[i][j].imshow(X_train[k].reshape(28,28),aspect='auto')
plt.show()
```

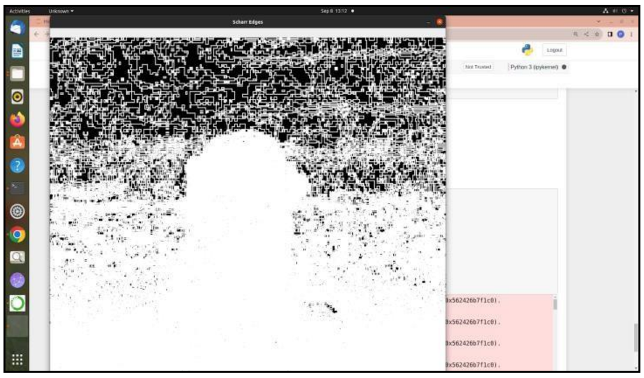
OUTPUT:



```
#Scharr

import cv2

#load an image
image=cv2.imread('/home/user/Desktop/pro.jpg',cv2.IMREAD_GRAYSCALE)

#Apply Scharr edge detection
scharr_x=cv2.Scharr(image,cv2.CV_64F,1,0)
scharr_y=cv2.Scharr(image,cv2.CV_64F,0,1)
#Laplacian

import cv2

#load an image
image=cv2.imread('/home/user/Desktop/pro.jpg',cv2.IMREAD_GRAYSCALE)

#Apply Laplacian edge detection
edges=cv2.Laplacian(image,cv2.CV_64F)
```

```
#Display the resulting edges
cv2.imshow('Laplacian Edges',edges)
cv2.waitKey(0)
cv2.DestroyAllWindows()
#Display the resulting edge map
cv2.imshow('Scharr Edges',edges)
cv2.waitKey(0)
cv2.DestroyAllWindows()
```

OUTPUT:



```
0x562426b7f1c0).
0x562426b7f1c0).
0x562426b7f1c0).
0x562426b7f1c0).
```

RESULT:

The image is loaded and edge detection is done using the Laplacian and Scharr operator algorithms.