Navneet Krishna

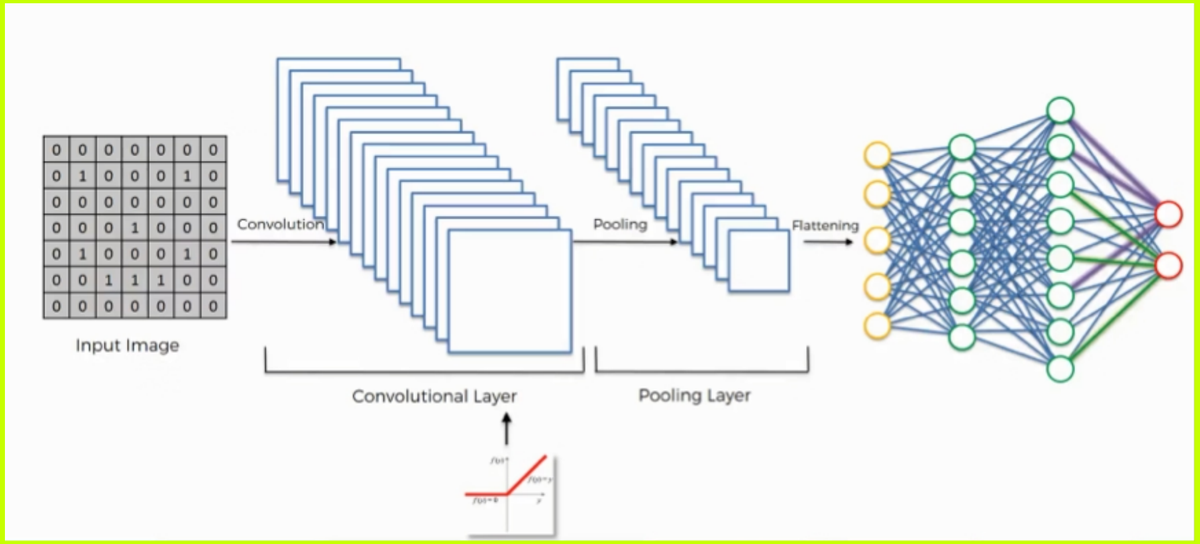
CB.EN.U4AIE22011

MFC - 4

ASSIGNMENT - 1

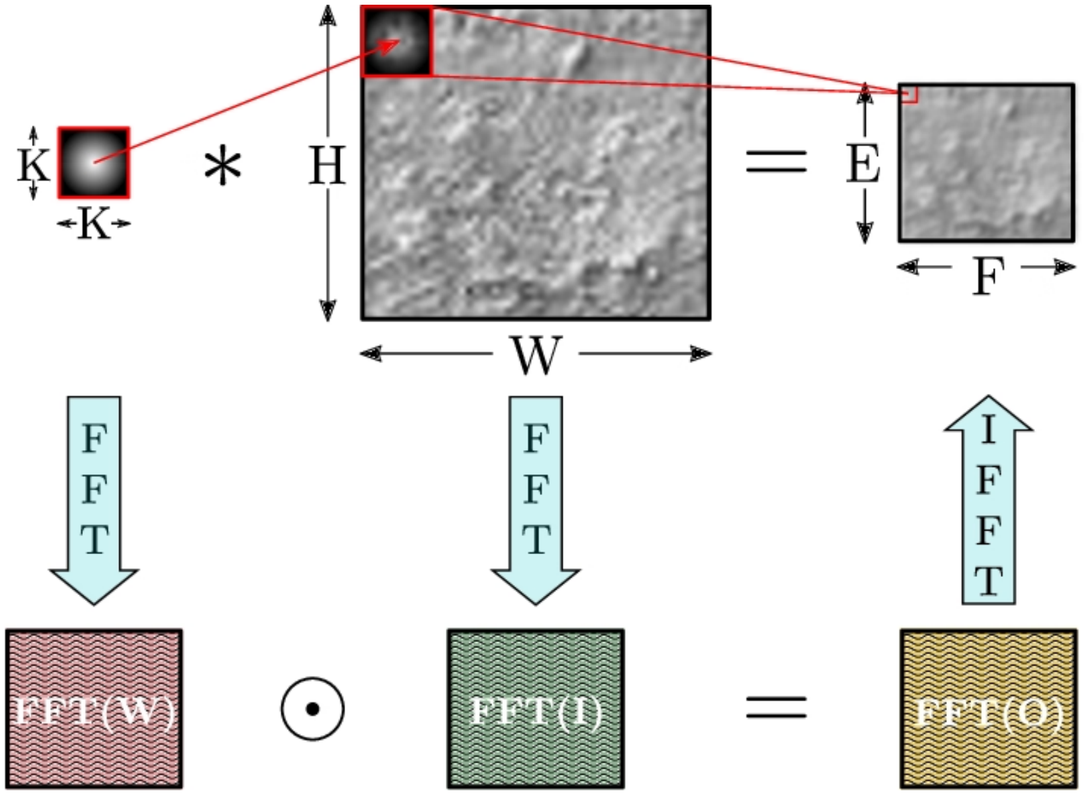
Topic

SUBSTITUTE FOR CONVOLUTIONAL LAYER



Convolution layer
Substituted with -

FFT LAYER



But for this assignment, DFT was used

How?

Using 2-Dimensional DFT Algorithm

$$\hat{h}(k,l) = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} e^{-i(\omega_k n + \omega_l m)}\, h(n,m)$$

(or)

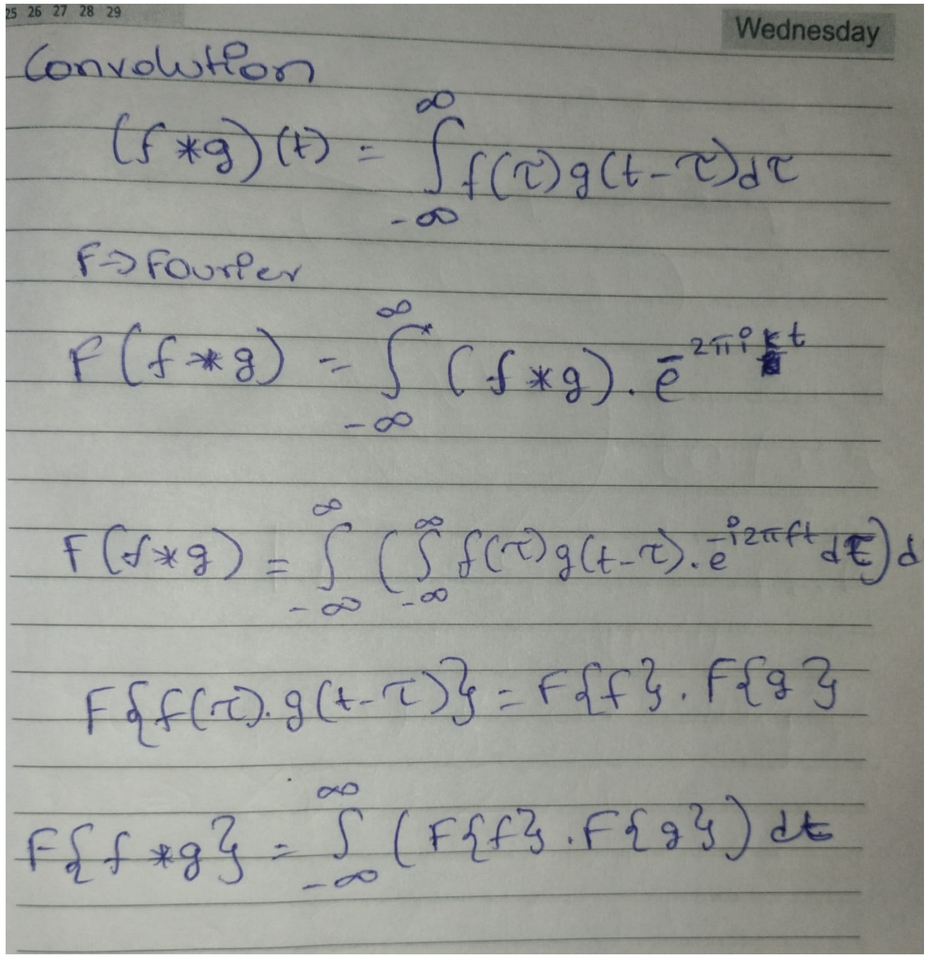
$$h(n,m) = \frac{1}{NM}\sum_{k=0}^{N-1}\sum_{l=0}^{M-1} e^{i(\omega_k n + \omega_l m)}\, \hat{h}(k,l)$$

Can also use 1D DFT Algorithm

$$X(k) = \sum_{n=0}^{N-1} x(t_n) e^{-2i*pi*k*t_{n/N}}$$

How did we know???

PROOF



Steps

1. Take an image and a filter/kernel to perform convolution.
2. Pad the filter with zeros to match its shape with the image.
3. Apply FFT or DFT on the image and the filter.
4. Multiply both, FFT{image} and DFT{filter} (element wise).
5. Apply IFFT or IDFT on the the output.
6. Zero bit would’ve shifted to the left. So pull it back and place it in the center.
7. ”Convolved” output is ready to pass into the next layer.

Overview

DFT → f

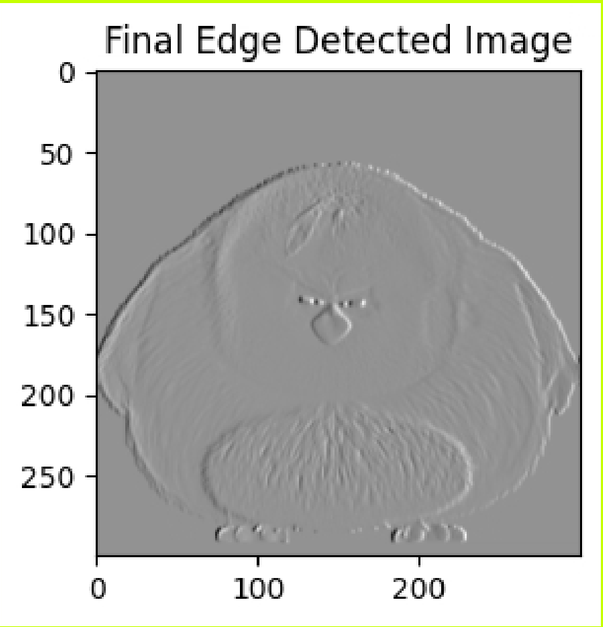
-1	0	+1
-2	0	+2
-1	0	+1

DFT → g

f * g

IDFT

(after shifting zero)



Code

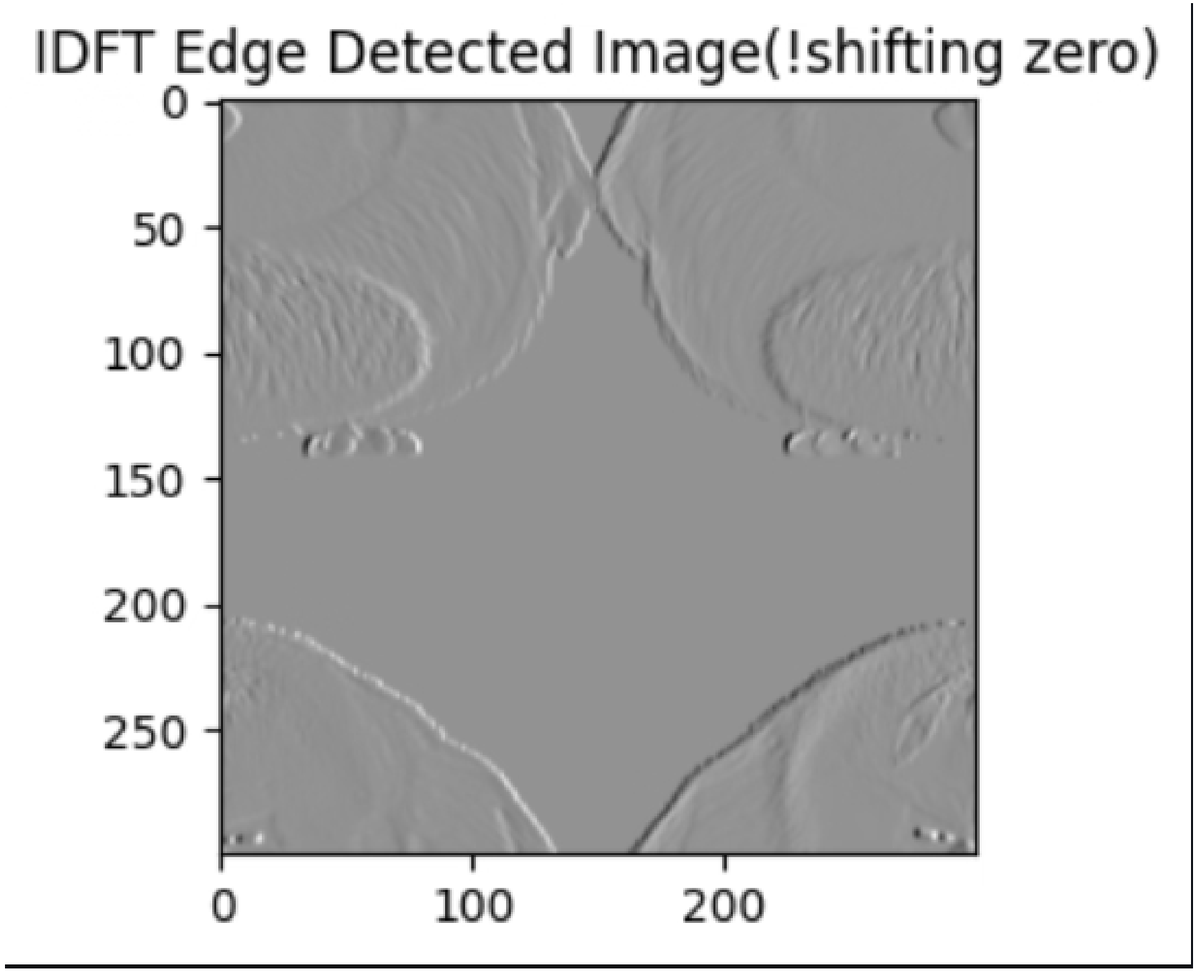
DFT2D

```
def Dft2(ip):
    row, col = ip.shape
    op = np.zeros_like(ip, dtype=complex)
    for k in range(row):
        if (k+1)%100==0:print(k)
        y = np.arange(col)
        bas1 = np.exp(-2j * np.pi * k * np.arange(row) / row)
        for l in range(col):
            bas2 = np.exp(-2j * np.pi * l * y / col)
            bas = np.outer(bas1, bas2)
            print(bas.shape,bas1.shape,bas2.shape,ip.shape)
            op[k,l] = np.sum(ip*bas)
    return op
```

IDFT2D

```
def IDft2(ip):
    op=np.zeros_like(ip,dtype=complex)
    row, col = ip.shape
    for k in range(row):
        bas1=np.exp(2j*np.pi*np.arange(row)*k/row)
        for l in range(col):
            bas2=np.exp(2j*np.pi*np.arange(col)*l/col)
            bas=np.outer(bas1,bas2)
            op[k,l]=(np.sum(ip*bas))/(row*col)
    return op
```

Output

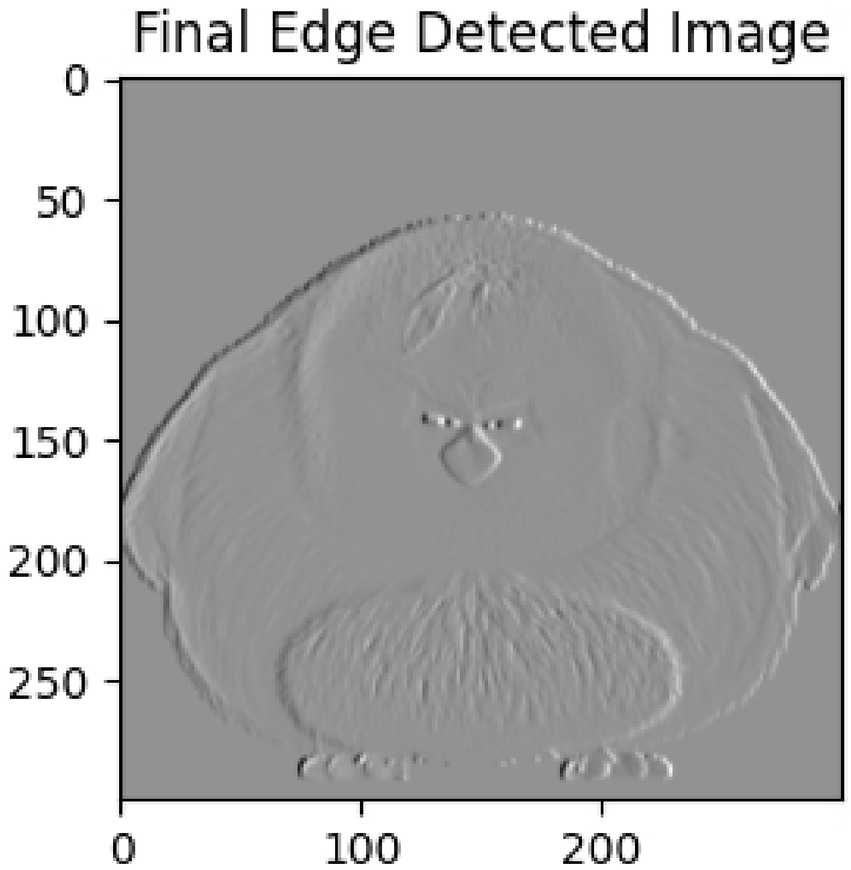


Code

Shifts zero

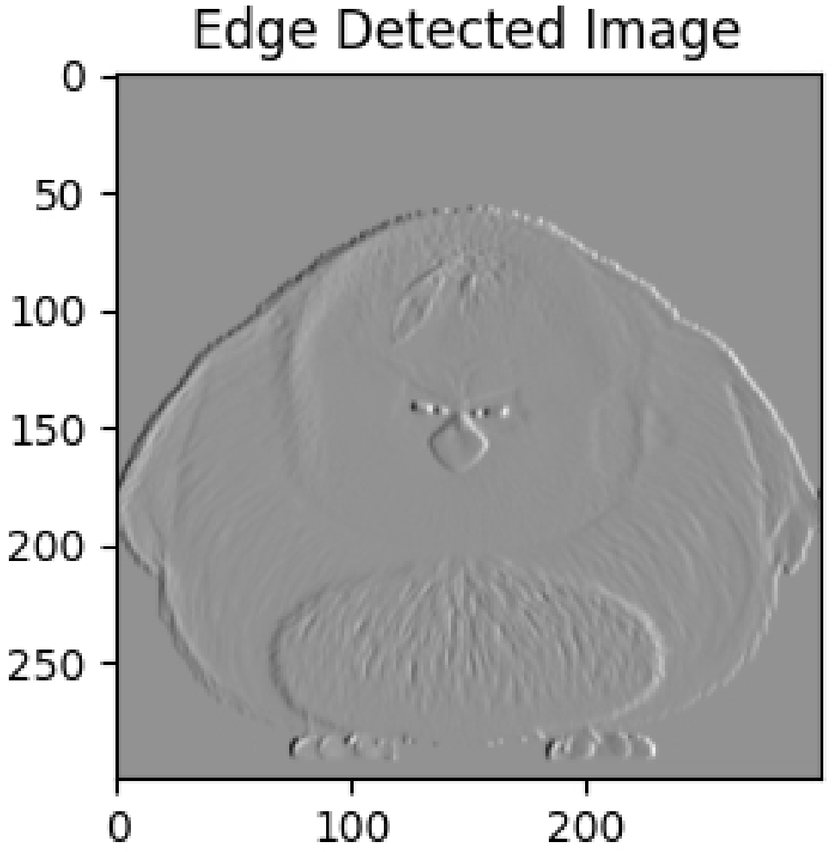
```
from math import ceil
r=ceil(op.shape[0]/2)
c=ceil(op.shape[1]/2)
zer=np.zeros(op.shape)

zer[0:r,0:c]=op[r:,c:]
zer[r:,c:]=op[0:r,0:c]
zer[0:r,c:]=op[r:,0:c]
zer[r:,0:c]=op[0:r,c:]
plt.figure(figsize=(3,3))
plt.imshow(zer,cmap='gray')
plt.title('Final Edge Detected Image')
plt.show()
```

Output



Output and Code using inbuilt

```
f1=np.fft.fftn(img)
k1=np.fft.fftn(k1)
op=f1*k1
op=np.fft.ifftn(op)
plt.figure(figsize=(3,3))
plt.title('Original Image')
plt.imshow(np.real(op),cmap='gray')
```



Done

THE END