Computer Vision Home Work 1

Name:許晉偉

Studentid:610415145

program and method

Q1:Read a RGB image and write a function to convert the image to **grayscale image**

```
In [1]: import cv2
        import numpy as np
        img = cv2.imread('liberty.png')
        Height, width, channels = img.shape[:3]
        gray_mat= np.zeros((Height, width, 1), np.uint8)
```

Read RGB image and create a new One-dimensional array(gray_mat) to store gray image

```
In [2]: for i in range(0,img.shape[0]):
            for j in range(0,img.shape[1]):
                r=img[i,j,2]
                g=img[i,j,1]
                b=img[i,j,0]
                gray_mat[i,j,0]=r*0.299+g*0.587+b*0.114
```

Using loop to get each pixel value and transform original Image to grayscale image by using formula

Q2:Write a convolution operation with edge detection kernel and activation function

```
ReLU_img= np.zeros((Height, width, 1), np.uint8)
kernel= np.array([(-1,-1,-1),(-1,8,-1),(-1,-1,-1)])
```

create kernel and three-dimensional array to store the convoluted image

```
In [3]: def convolution(x,y,img):
            result=0
            for i in range(0,3):
                for j in range(0,3):
                    result=result+img[x+i,y+j,0]*kernel[i][j]
            return result

        for i in range(0,img.shape[0]-2):
            for j in range(0,img.shape[1]-2):
                temp=convolution(i,j,gray_mat)
                if(temp)<0 :
                    temp=0
                if(temp)>255 :
                    temp=255
                ReLU_img[i,j,0]=temp
```

Using loop to Transfer coordinates to convolution function and store the return value before do the Rectified Linear Unit operation for convolution feature and store the value

```
In [3]: def convolution(x,y,img):
            result=0
            for i in range(0,3):
                for j in range(0,3):
                    result=result+img[x+i,y+j,0]*kernel[i][j]
            return result
```

In the convolution function the x,y is coordinate that be transferred by the loop,and i,j is seems like offset

so the loop do the addition from img[x+0][y+0]*kernel[0][0] ~ img[x+2][y+2]*kernel[2][2] and return the value

Q3:Write a pooling operation with using **Max pooling, 2x2 filter, and stride 2**

```
maxpooling_img=np.zeros(((Height)//2, (width)//2, 1), np.uint8)
```

Create three-dimensional array to store the polling image

After to do the pooling operation image the size will reduce to half

so the height and width will be divided by 2

```
In [4]: def maxpooling(x,y,img):
            max_value=0
            for i in range(0,2):
                for j in range(0,2):
                    if(img[x+i][y+j]>max_value):
                        max_value=img[x+i][y+j]

            return max_value


        for i in range(0,img.shape[0]-1,2):
            for j in range(0,img.shape[1]-1,2):
                maxpooling_img[i//2,j//2,0]=maxpooling(i,j,ReLU_img)
```

Using loop to Transfer coordinates to the maxpooling function

and store the return value

```
In [4]: def maxpooling(x,y,img):
            max_value=0
            for i in range(0,2):
                for j in range(0,2):
                    if(img[x+i][y+j]>max_value):
                        max_value=img[x+i][y+j]

            return max_value
```

The x,y is coordinate that be transferred by the loop.
After ReLU function has done, the image only have positive value so when the loop run the max_value will become img[x+0][y+0] and compare each other until img[x+1][y+1] to decide which is biggest value then return the value

Q4:Write a **binarization operation** (threshold = 128). (>=128) set 255 (<128) set 0

In [5]:
```
#binarization operation
threshold = 128
for i in range(0,(img.shape[0])//2):
    for j in range(0,(img.shape[1])//2):
        if(maxpooling_img[i][j]>=threshold):
            maxpooling_img[i][j]=255
        else:
            maxpooling_img[i][j]=0
binaryimg=maxpooling_img
cv2.imwrite('binary.jpg',binaryimg)
```

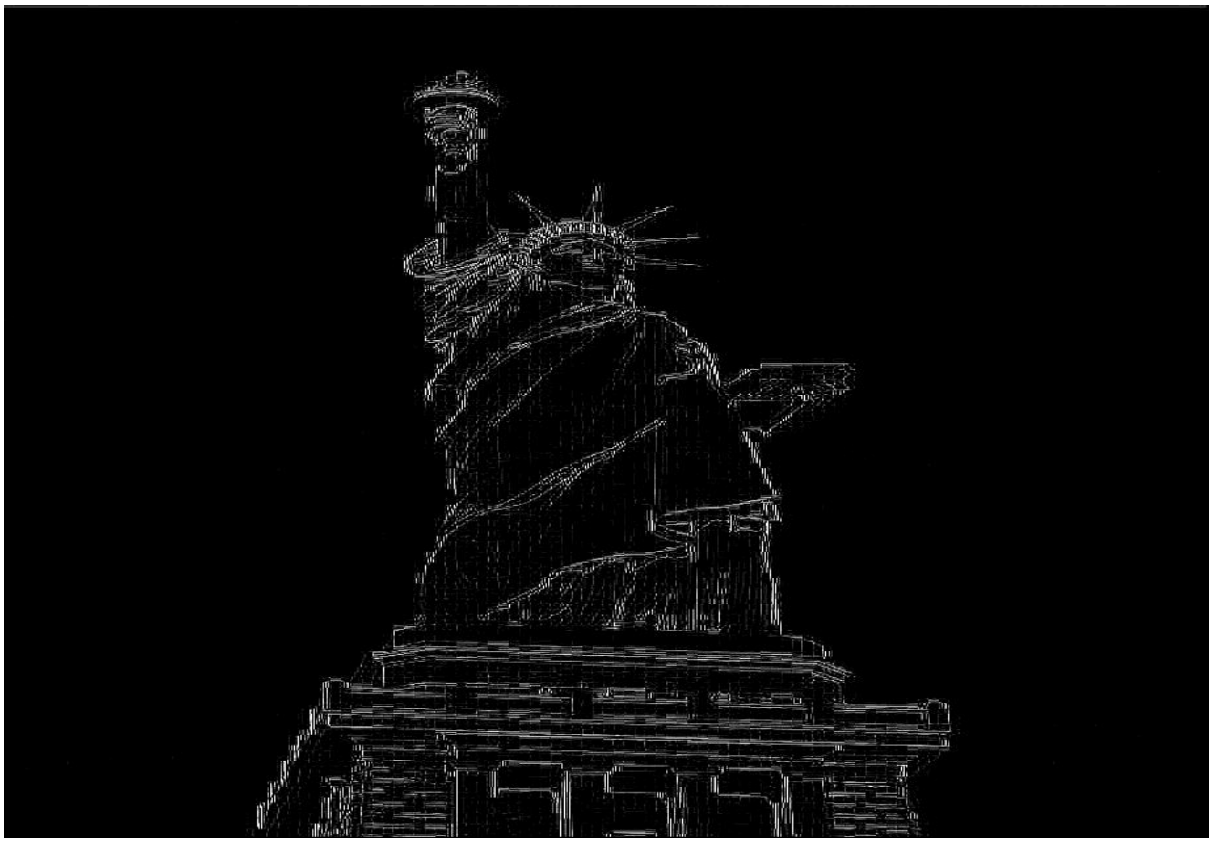
The grayscale image intensity which is decided by its number ,so set the threshold 128(a half of 0~255) and run the loop to get each pixel intensity value and compare the value to determine the intensity value become white(255) or black(0)

Result images:

1. original image(liberty.png)

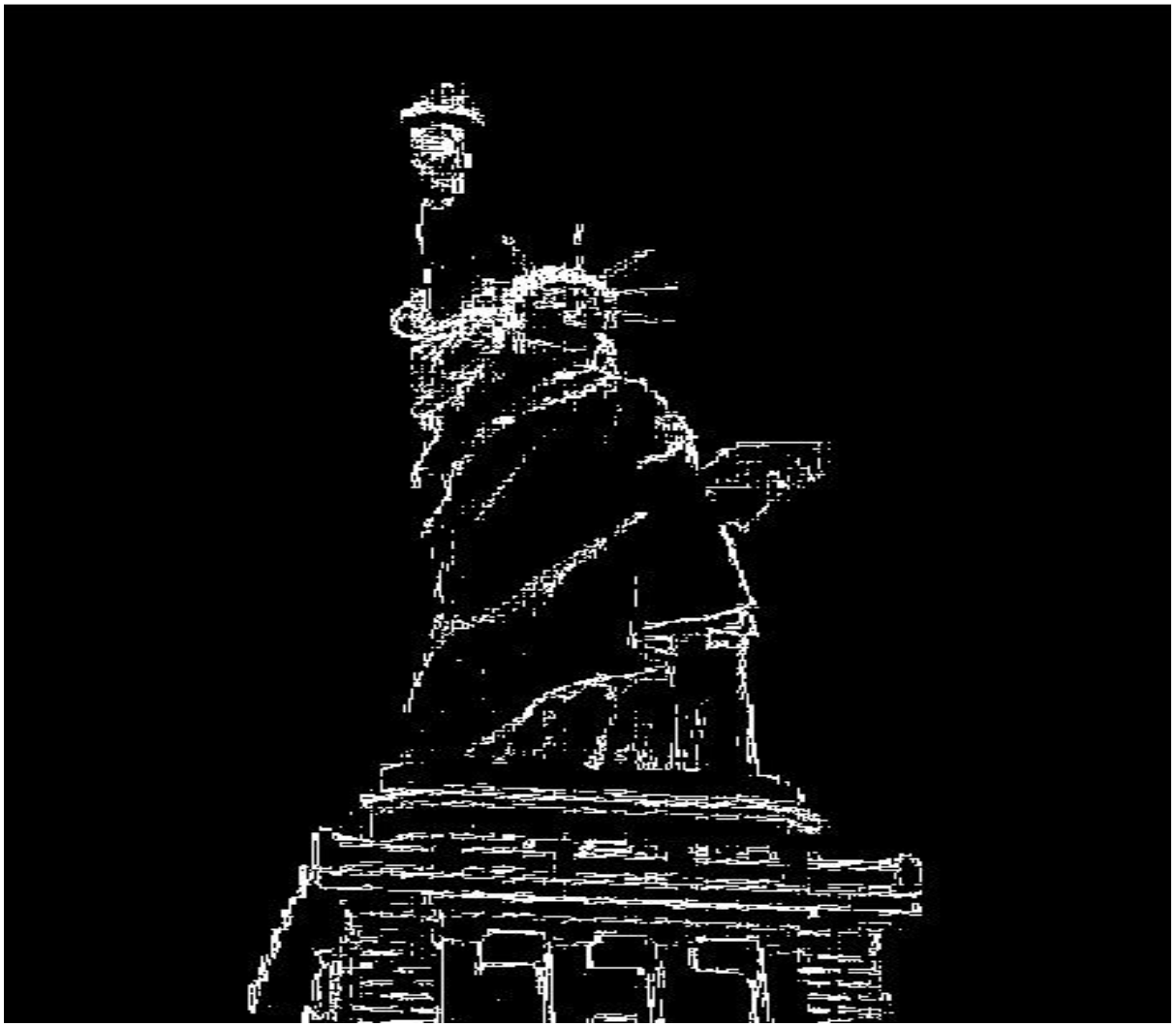
2. grayscale image

3. ReLU image



4.maxpooling image

5. binary image



6. original image(car.png)

7. grayscale image

8. ReLU image

9. maxpooling image

10. binary image