HW1 報告

D1149576 曹世杰

程式碼：

```
import cv2 as cv;
import numpy as np;

# _____庫_____
#|           |
#|           |庫
#|___________|


def con(img,m,n,kernel,bias,padding):
    if(padding==1): #補白邊
        pad = np.zeros((m+(len(kernel[0])-1)*2,n+(len(kernel)-1)),np.uint8);        #創建一個空圖片(黑)

        for j in range(n+(len(kernel)-1)):                                          #填滿白色
            for i in range(m+(len(kernel[0])-1)*2):
                pad[i][j] = 255;

        for j in range(n+(len(kernel)-1)):                                          #將原本的圖片放到中間
            for i in range(m+(len(kernel[0])-1)):
                if((i<len(kernel[0])-1+m and i>=len(kernel[0])-1) and (j<len(kernel)-1+n and
j>=len(kernel)-1)):
                    pad[i][j] = img[i-(len(kernel[0])-1)][j-(len(kernel)-1)];

    elif(padding==0):   #純黑邊
        pad = np.zeros((m+(len(kernel[0])-1)*2,n+(len(kernel)-1)),np.uint8);

        for j in range(n+(len(kernel)-1)):                                          #將原本的圖片放到中間
            for i in range(m+(len(kernel[0])-1)):
                if((i<len(kernel[0])-1+m and i>=len(kernel[0])-1) and (j<len(kernel)-1+n and
j>=len(kernel)-1)):
                    pad[i][j] = img[i-(len(kernel[0])-1)][j-(len(kernel)-1)];
    else:
        pad = img;  #不補邊

    (x,y) = pad.shape;
    if(padding==1 or padding==0):   #如果有補邊則使用原圖片的長寬作為計算次數
        leng=m;
        wid=n;
    else:                           #若無則計算輸出圖片的大小作為計算次數
        leng=m-len(kernel[0])+1;
        wid=n-len(kernel)+1;

    Out = np.zeros((leng,wid),np.uint8);                                    #創建輸出用的圖片
    for w_num in range(wid):                                                #寬的當行次數
        for l_num in range(leng):                                           #長的當行次數
            temp=0;                                                         #記錄點的數值
            for w in range(len(kernel)):                                    #Kernel的寬
                for l in range(len(kernel[w])):                             #Kernel的長
                    temp += int(pad[l+l_num][w+w_num])*kernel[l][w];        #計算
            temp += bias;                                                   #最後加上偏差值
            Out[l_num][w_num] = max(0, min(255, temp));                     #將範圍限制於0~255
    return Out;


def pool(img,size,stride,type):
    (x,y) = img.shape;                      #原圖片的長寬
    length = (x-size)//stride+1;            #計算縮小後的圖片長度
    width = (y-size)//stride+1;             #計算縮小後的圖片寬度
    Out = np.zeros((length, width),np.uint8);   #建立輸出的圖片

    if(type==0):    #average pool
        for w_num in range(width):                                          #寬數量
            for l_num in range(length):                                     #長數量
                temp=0;total=0;                                             #temp平均後給Out的值；total為加總
                for w in range(w_num*stride,w_num*stride+size):             #size要用長寬
                    for l in range(l_num*stride,l_num*stride+size):         #size要的長度
                        if(w<y and l<x):                                    #避免超出範圍
                            total += int(img[l][w]);                        #加總
                temp = total//(size*size);                                  #平均
                Out[l_num][w_num] = max(0, min(255, temp));                 #將範圍限制到0~255
    else:   #max pool
        for w_num in range(width):
            for l_num in range(length):
                Fmax=0;
                for w in range(w_num*stride,w_num*stride+size):
                    for l in range(l_num*stride,l_num*stride+size):
                        if(Fmax<img[l][w] and w<y and l<x):
                            Fmax = img[l][w];
                Out[l_num][w_num] = max(0, min(255, Fmax));
    return Out;

def signName(img,name,m,n):
    (x1,y1)=name.shape;                 #簽名檔的長寬
    if(m>n): adjust = max(1,x1//(m//10));   #長邊若大於縮成原圖長邊的1/10
    else: adjust = max(1,y1//(n//10));      #長邊若大於縮成原圖長邊的1/10
    ad_name = pool(name,adjust,adjust,1);   #利用pool縮小
    (x2,y2)=ad_name.shape;                  #縮小後的長寬

    if (m - x2 >= 0) and (n - y2 >= 0):     #確保範圍正確
        for j in range(y2):
            for i in range(x2):
                if(ad_name[i][j]>=245 or ad_name[i][j]<=200):   #將簽名檔的部分覆蓋到圖片上
                    img[m-x2+i-10][n-y2+j-10]=ad_name[i][j];
    return img;



img = cv.imread("test1.jpg",0);  #讀入原圖
name = cv.imread("name.jpg",0);
cv.imshow("img",img);           #展示原圖
cv.waitKey();                   #等待任意鍵


average = [[1/9,1/9,1/9],[1/9,1/9,1/9],[1/9,1/9,1/9]];  #average filter
sobel = [[-1,0,1],[-2,0,2],[-1,0,1]];                    #sobel filter
gaussian = [[1/16,2/16,1/16],[2/16,4/16,2/16],[1/16,2/16,1/16]]; #gaussian filter

(x,y) = img.shape;                  #原圖長寬
Out_con = con(img,x,y,average,0,1);    #計算Con
(x1,y1) = Out_con.shape;
temp = signName(Out_con,name,x1,y1);
cv.imshow("Con_ave",Out_con);           #輸出Con後的圖片
cv.imwrite("Con_ave.jpg",Out_con);


Out_sob = con(img,x,y,sobel,0,-1);    #計算Con
(x1,y1) = Out_sob.shape;
temp = signName(Out_sob,name,x1,y1);
cv.imshow("Con_sob",Out_sob);           #輸出Con後的圖片
cv.imwrite("Con_sob.jpg",Out_sob);


Out_gu = con(img,x,y,gaussian,0,-1);    #計算Con
(x1,y1) = Out_gu.shape;
temp = signName(Out_gu,name,x1,y1);
cv.imshow("Con_gu",Out_gu);             #輸出Con後的圖片
cv.imwrite("Con_gu.jpg",Out_gu);


Out_pool_max = pool(img,2,2,1);             #計算Pool
(x1,y1) = Out_pool_max.shape;
temp = signName(Out_pool_max,name,x1,y1);
cv.imshow("Pool_max",Out_pool_max);         #輸出Pool後的圖片
cv.imwrite("Pool_max.jpg",Out_pool_max);


Out_pool_ave = pool(img,2,2,0);             #計算Pool
(x1,y1) = Out_pool_ave.shape;
temp = signName(Out_pool_ave,name,x1,y1);
cv.imshow("Pool_ave",Out_pool_ave);         #輸出Pool後的圖片
cv.imwrite("Pool_ave.jpg",Out_pool_ave);)
}
```

註解：

Con：

```
def con(img,m,n,kernel,bias,padding):
    if(padding==1): #補白邊
        pad = np.zeros((m+(len(kernel[0])-1)*2,n+(len(kernel)-1)),np.uint8);        #創建一個空圖片(黑)

        for j in range(n+(len(kernel)-1)):                                          #填滿白色
            for i in range(m+(len(kernel[0])-1)*2):
                pad[i][j] = 255;

        for j in range(n+(len(kernel)-1)):                                          #將原本的圖片放到中間
            for i in range(m+(len(kernel[0])-1)):
                if((i<len(kernel[0])-1+m and i>=len(kernel[0])-1) and (j<len(kernel)-1+n and
j>=len(kernel)-1)):
                    pad[i][j] = img[i-(len(kernel[0])-1)][j-(len(kernel)-1)];

    elif(padding==0):   #補黑邊
        pad = np.zeros((m+(len(kernel[0])-1)*2,n+(len(kernel)-1)),np.uint8);

        for j in range(n+(len(kernel)-1)):                                          #將原本的圖片放到中間
            for i in range(m+(len(kernel[0])-1)):
                if((i<len(kernel[0])-1+m and i>=len(kernel[0])-1) and (j<len(kernel)-1+n and
j>=len(kernel)-1)):
                    pad[i][j] = img[i-(len(kernel[0])-1)][j-(len(kernel)-1)];
    else:
        pad = img;  #不補邊

    (x,y) = pad.shape;
    if(padding==1 or padding==0):   #如果有補邊則使用原圖片的長寬作為計算次數
        leng=m;
        wid=n;
    else:                           #若無則計算輸出圖片的大小作為計算次數
        leng=m-len(kernel[0])+1;
        wid=n-len(kernel)+1;

    Out = np.zeros((leng,wid),np.uint8);                                #創建輸出用的圖片
    for w_num in range(wid):                                            #寬的運行次數
        for l_num in range(leng):                                       #長的運行次數
            temp=0;                                                     #記錄點的數值
            for w in range(len(kernel)):                                #kernel的寬
                for l in range(len(kernel[w])):                         #kernel的長
                    temp += int(pad[l+l_num][w+w_num])*kernel[l][w];    #計算
            temp += bias;                                               #最後加上偏差值
            Out[l_num][w_num] = max(0, min(255, temp));                 #將範圍限制到0~255
    return Out;
```

Pool：

```
def pool(img,size,stride,type):
    (x,y) = img.shape;                          #原圖片的長寬
    length = (x-size)//stride+1;                #計算縮小後的圖片長度
    width = (y-size)//stride+1;                 #計算縮小後的圖片寬度
    Out = np.zeros((length, width),np.uint8);   #建立輸出的圖片

    if(type==0):    #average pool
        for w_num in range(width):                                      #寬數量
            for l_num in range(length):                                 #長數量
                temp=0;total=0;                                         #temp平均後給Out的值，total為加總
                for w in range(w_num*stride,w_num*stride+size):          #size要得寬度
                    for l in range(l_num*stride,l_num*stride+size):     #size要的長度
                        if(w<y and l<x):                                #避免超出範圍
                            total += int(img[l][w]);                    #加總
                temp = total//(size*size);                              #平均
                Out[l_num][w_num] = max(0, min(255, temp));             #將範圍限制到0~255
    else:   #max pool
        for w_num in range(width):
            for l_num in range(length):
                Fmax=0;
                for w in range(w_num*stride,w_num*stride+size):
                    for l in range(l_num*stride,l_num*stride+size):
                        if(Fmax<img[l][w] and w<y and l<x):
                            Fmax = img[l][w];
                Out[l_num][w_num] = max(0, min(255, Fmax));
    return Out;
```

signName：

```
def signName(img,name,m,n):
    (x1,y1)=name.shape;                         #簽名檔的長寬
    if(m>n): adjust = max(1,x1//(m//10));       #長邊較大則縮成原圖長邊的1/10
    else: adjust = max(1,y1//(n//10));          #寬邊較大則縮成原圖寬邊的1/10
    ad_name = pool(name,adjust,adjust,1);       #利用pool縮小
    (x2,y2)=ad_name.shape;                      #縮小後的長寬

    if (m - x2 >= 0) and (n - y2 >= 0):         #確保範圍正確
        for j in range(y2):
            for i in range(x2):
                if(ad_name[i][j]>=245 or ad_name[i][j]<=200):    #將要顯示的部分覆蓋到圖片上
                    img[m-x2+i-10][n-y2+j-10]=ad_name[i][j];
    return img;
```

Main：

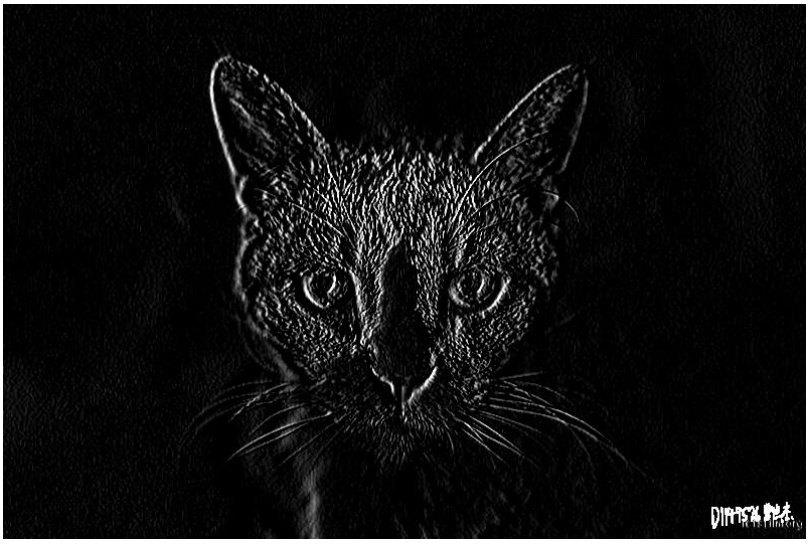
```
img = cv.imread("self.jpg",0);  #讀入原圖
name = cv.imread("name.jpg",0);
cv.imshow("img",img);           #顯示原圖
cv.waitKey();                   #等待按鍵


average = [[1/9,1/9,1/9],[1/9,1/9,1/9],[1/9,1/9,1/9]];  #average filter
sobel = [[-1,0,1],[-2,0,2],[-1,0,1]];                    #sobel filter
gaussian = [[1/16,2/16,1/16],[2/16,4/16,2/16],[1/16,2/16,1/16]]; #gaussian filter

(x,y) = img.shape;                  #原圖長寬
Out_con = con(img,x,y,average,0,-1);    #計算Con
(x1,y1) = Out_con.shape;
temp = signName(Out_con,name,x1,y1);
cv.imshow("Con_ave",Out_con);           #顯示Con後的圖片
cv.imwrite("Con_ave.jpg",Out_con);


Out_sob = con(img,x,y,sobel,0,-1);    #計算Con
(x1,y1) = Out_sob.shape;
temp = signName(Out_sob,name,x1,y1);
cv.imshow("Con_sob",Out_sob);           #顯示Con後的圖片
cv.imwrite("Con_sob.jpg",Out_sob);


Out_gu = con(img,x,y,gaussian,0,-1);    #計算Con
(x1,y1) = Out_gu.shape;
temp = signName(Out_gu,name,x1,y1);
cv.imshow("Con_gu",Out_gu);           #顯示Con後的圖片
cv.imwrite("Con_gu.jpg",Out_gu);


Out_pool_max = pool(img,2,2,1);           #計算Pool
(x1,y1) = Out_pool_max.shape;
temp = signName(Out_pool_max,name,x1,y1);
cv.imshow("Pool_max",Out_pool_max);       #顯示Pool後的圖片
cv.imwrite("Pool_max.jpg",Out_pool_max);


Out_pool_ave = pool(img,2,2,0);           #計算Pool
(x1,y1) = Out_pool_ave.shape;
temp = signName(Out_pool_ave,name,x1,y1);
cv.imshow("Pool_ave",Out_pool_ave);       #顯示Pool後的圖片
cv.imwrite("Pool_ave.jpg",Out_pool_ave);
```

執行結果：(皆縮小 50 %)

1-1. 原圖

1-2. Average filter

1-3. Sobel filter



1-4. Gaussian

1-5. Average pool

1-6. Max pool(大小為 100%)

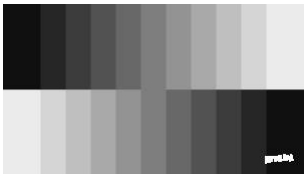
2-1.原圖

2-2.Average filter

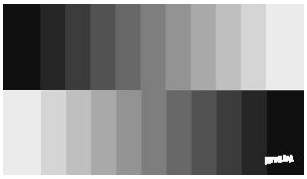
2-3.Sobel filter

2-4. Gaussian

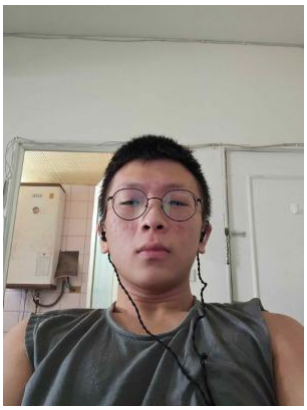
2-5.Average pool



2-6.Max pool



3-1.原圖(讀入為灰階)



3-2.Average filter

3-3.Sobel filter

3-4. Gaussian

3-5.Average pool

3-6.Max pool

心得：

原本用透明的簽名檔來覆蓋，後才知道需要多判斷一個 alpha 層，後來改用黑字白底的簽名檔，但覆蓋上去的樣子依舊很怪。但至少看的出來是簽名檔。