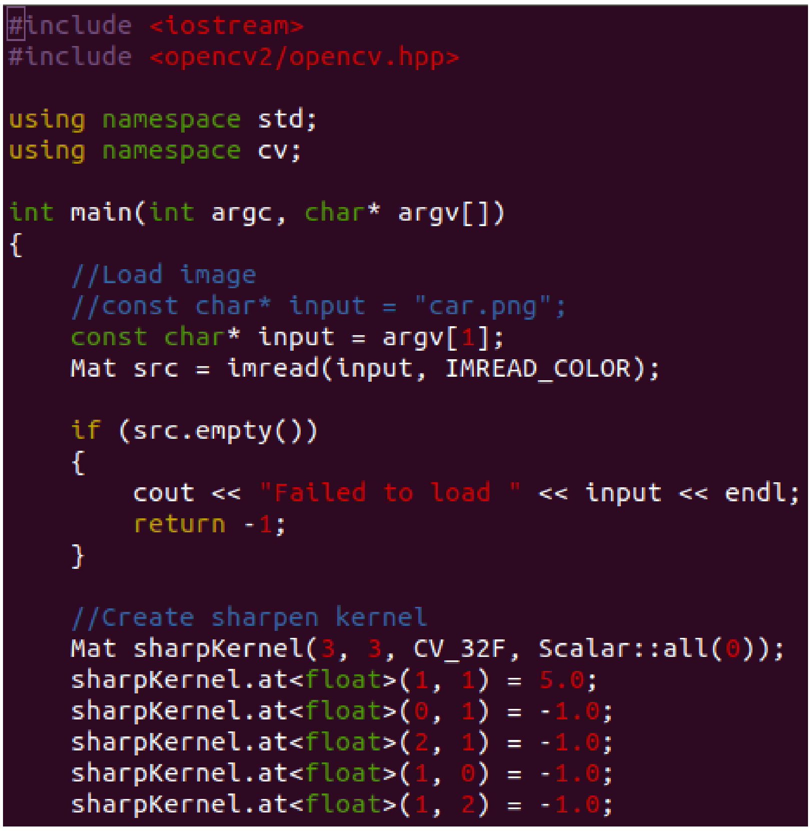
Computer vision HW1

Work flow:

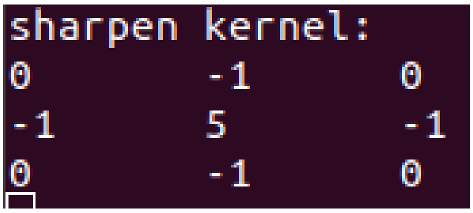
input image -> transform gray -> convolution operation with sharpen kernel
-> activation function (ReLU) -> pooling operation

input image:

use library of opencv “imread”
and set the sharpenkernel



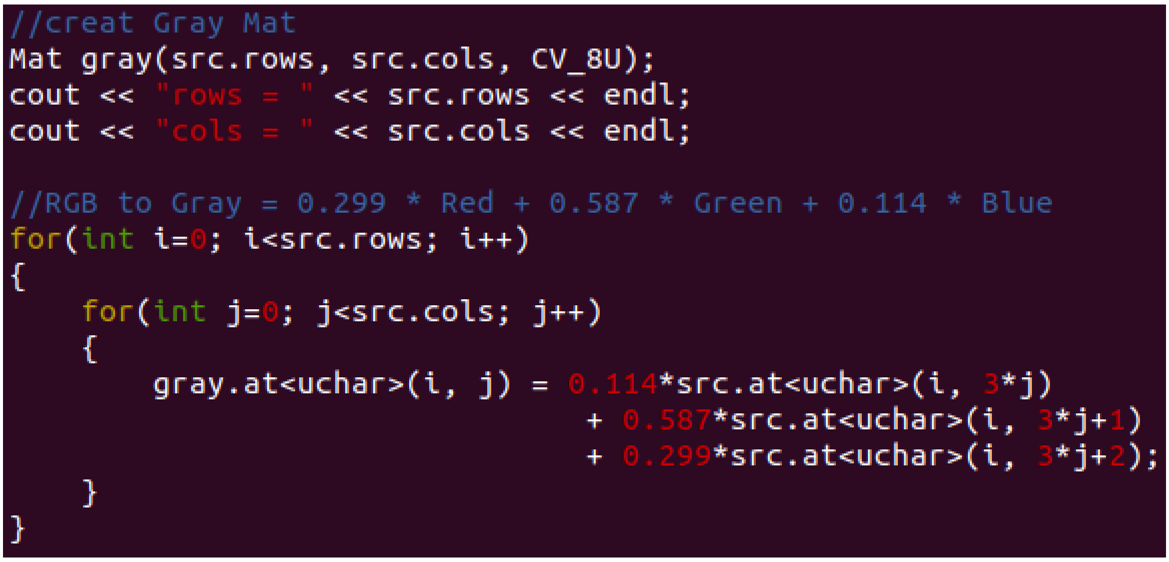
```
#include <iostream>
#include <opencv2/opencv.hpp>

using namespace std;
using namespace cv;

int main(int argc, char* argv[])
{
    //Load image
    //const char* input = "car.png";
    const char* input = argv[1];
    Mat src = imread(input, IMREAD_COLOR);

    if (src.empty())
    {
        cout << "Failed to load " << input << endl;
        return -1;
    }

    //Create sharpen kernel
    Mat sharpKernel(3, 3, CV_32F, Scalar::all(0));
    sharpKernel.at<float>(1, 1) = 5.0;
    sharpKernel.at<float>(0, 1) = -1.0;
    sharpKernel.at<float>(2, 1) = -1.0;
    sharpKernel.at<float>(1, 0) = -1.0;
    sharpKernel.at<float>(1, 2) = -1.0;
```



```
sharpen kernel:
0         -1         0
-1         5        -1
0         -1         0
```

transform gray:

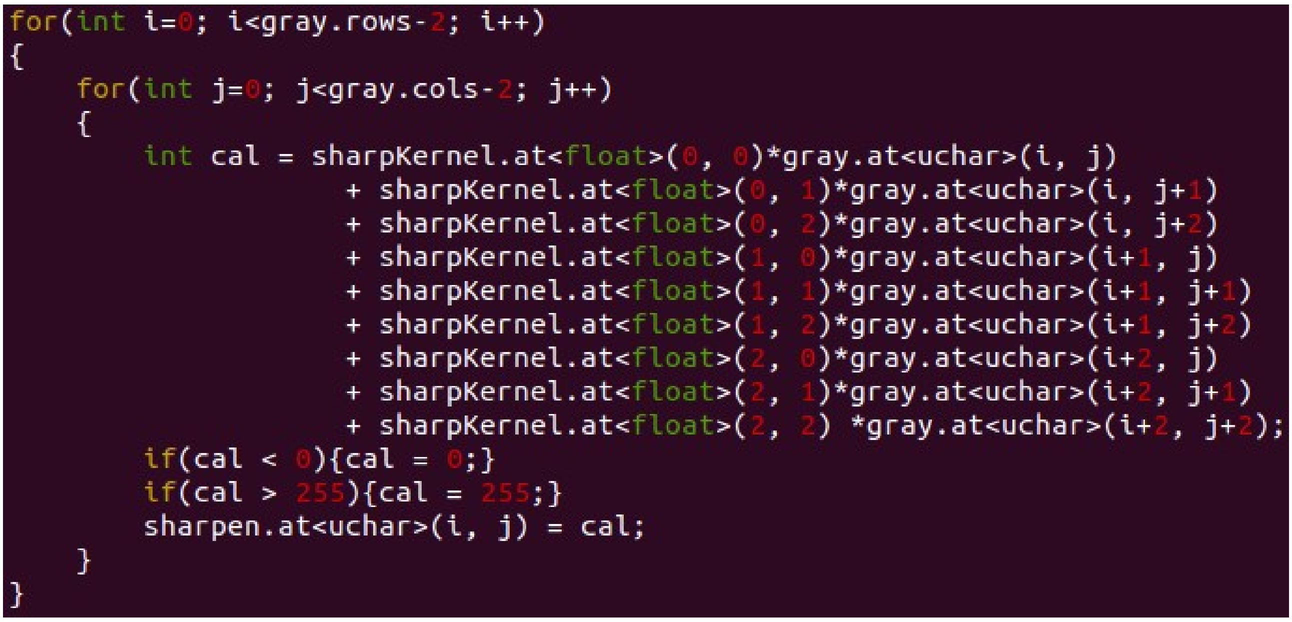
transform formula: Gray = R*0.299 + G*0.587 + B*0.114
Calculation and put output in Mat gray



```
//creat Gray Mat
Mat gray(src.rows, src.cols, CV_8U);
cout << "rows = " << src.rows << endl;
cout << "cols = " << src.cols << endl;

//RGB to Gray = 0.299 * Red + 0.587 * Green + 0.114 * Blue
for(int i=0; i<src.rows; i++)
{
    for(int j=0; j<src.cols; j++)
    {
        gray.at<uchar>(i, j) = 0.114*src.at<uchar>(i, 3*j)
                             + 0.587*src.at<uchar>(i, 3*j+1)
                             + 0.299*src.at<uchar>(i, 3*j+2);
    }
}
```

convolution operation:

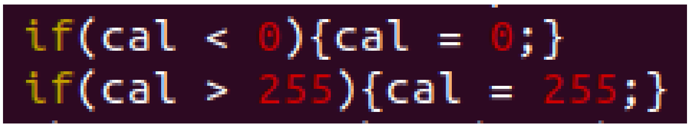
let gray image and sharpen do convolution operation
use basic of matrix multiplication, and put answer in variable name “cal”
then put “cal” in mat of result mat name “sharpen”



```
for(int i=0; i<gray.rows-2; i++)
{
    for(int j=0; j<gray.cols-2; j++)
    {
        int cal = sharpKernel.at<float>(0, 0)*gray.at<uchar>(i, j)
                    + sharpKernel.at<float>(0, 1)*gray.at<uchar>(i, j+1)
                    + sharpKernel.at<float>(0, 2)*gray.at<uchar>(i, j+2)
                    + sharpKernel.at<float>(1, 0)*gray.at<uchar>(i+1, j)
                    + sharpKernel.at<float>(1, 1)*gray.at<uchar>(i+1, j+1)
                    + sharpKernel.at<float>(1, 2)*gray.at<uchar>(i+1, j+2)
                    + sharpKernel.at<float>(2, 0)*gray.at<uchar>(i+2, j)
                    + sharpKernel.at<float>(2, 1)*gray.at<uchar>(i+2, j+1)
                    + sharpKernel.at<float>(2, 2) *gray.at<uchar>(i+2, j+2);
        if(cal < 0){cal = 0;}
        if(cal > 255){cal = 255;}
        sharpen.at<uchar>(i, j) = cal;
    }
}
```

activation function (ReLU):

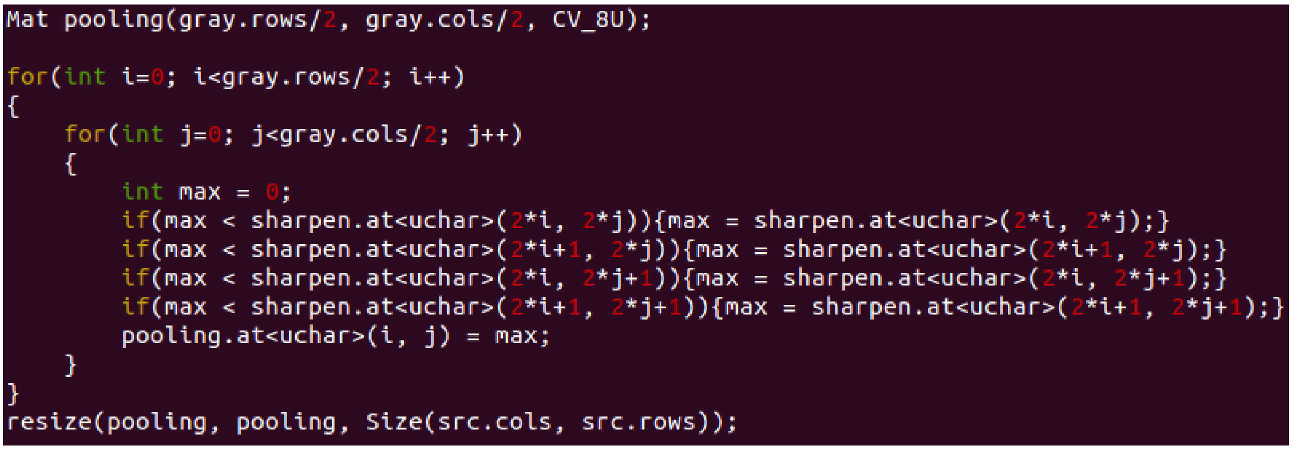
I use sample calculation to achieve ReLU
ReLU : max(0,x)



```
if(cal < 0){cal = 0;}
if(cal > 255){cal = 255;}
```

pooling operation:

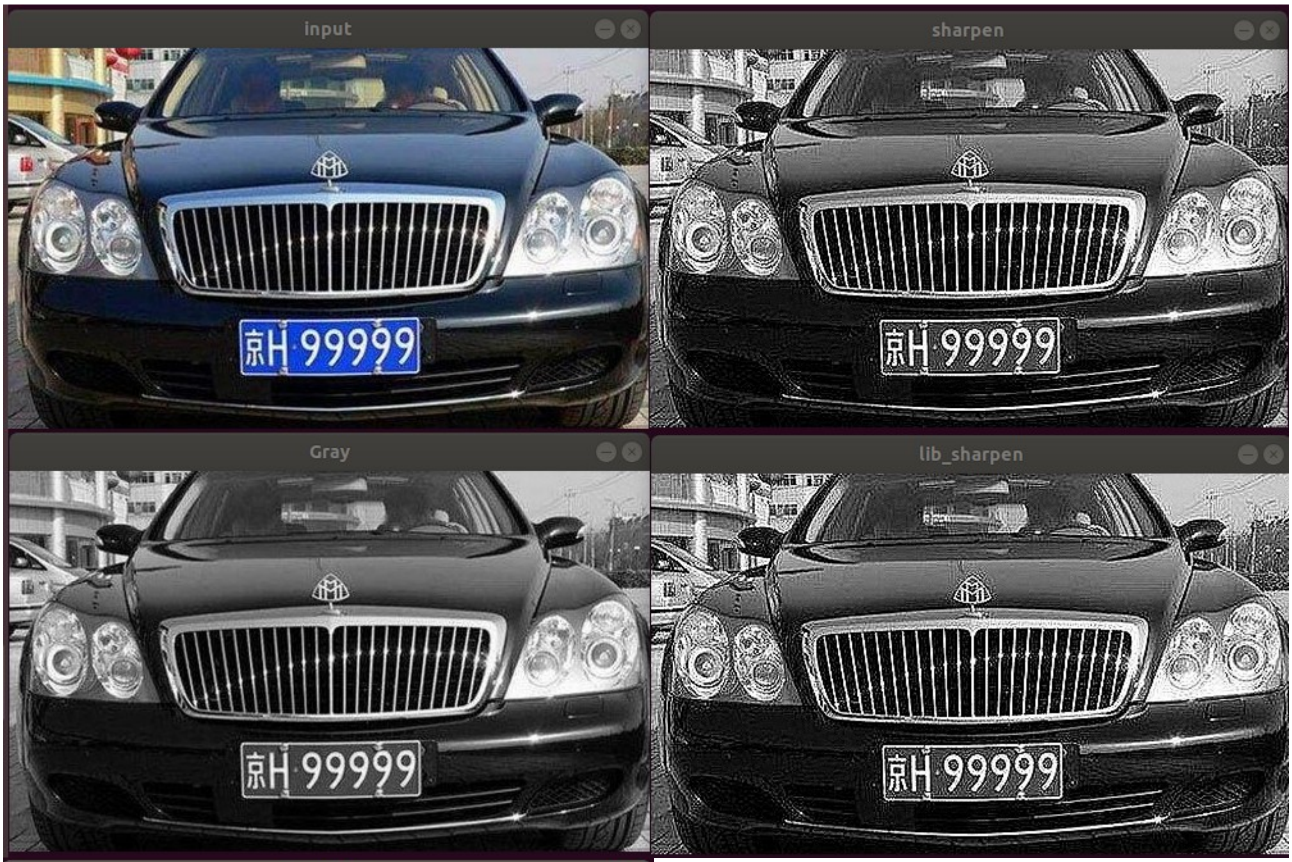
max pooling is use 2x2 matrix. I use for loop to get the largest number, and put in variable name “max”
then put “max” in mat of result mat name “pooling”
final resize the mat and let it’s size is same of other mat

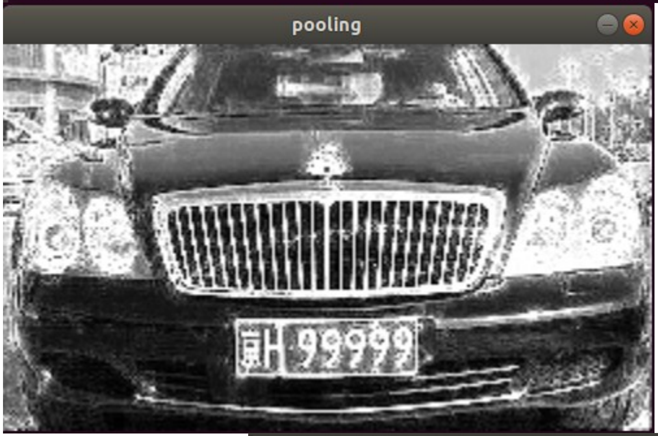


```
Mat pooling(gray.rows/2, gray.cols/2, CV_8U);

for(int i=0; i<gray.rows/2; i++)
{
    for(int j=0; j<gray.cols/2; j++)
    {
        int max = 0;
        if(max < sharpen.at<uchar>(2*i, 2*j)){max = sharpen.at<uchar>(2*i, 2*j);}
        if(max < sharpen.at<uchar>(2*i+1, 2*j)){max = sharpen.at<uchar>(2*i+1, 2*j);}
        if(max < sharpen.at<uchar>(2*i, 2*j+1)){max = sharpen.at<uchar>(2*i, 2*j+1);}
        if(max < sharpen.at<uchar>(2*i+1, 2*j+1)){max = sharpen.at<uchar>(2*i+1, 2*j+1);}
        pooling.at<uchar>(i, j) = max;
    }
}
resize(pooling, pooling, Size(src.cols, src.rows));
```

result mat :

-car.png

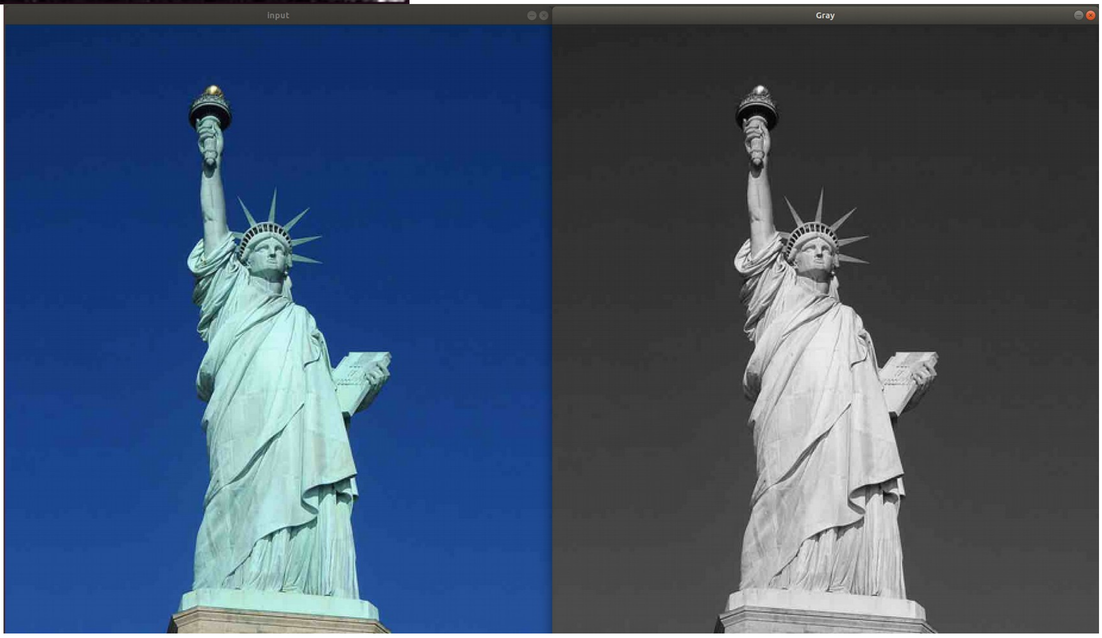


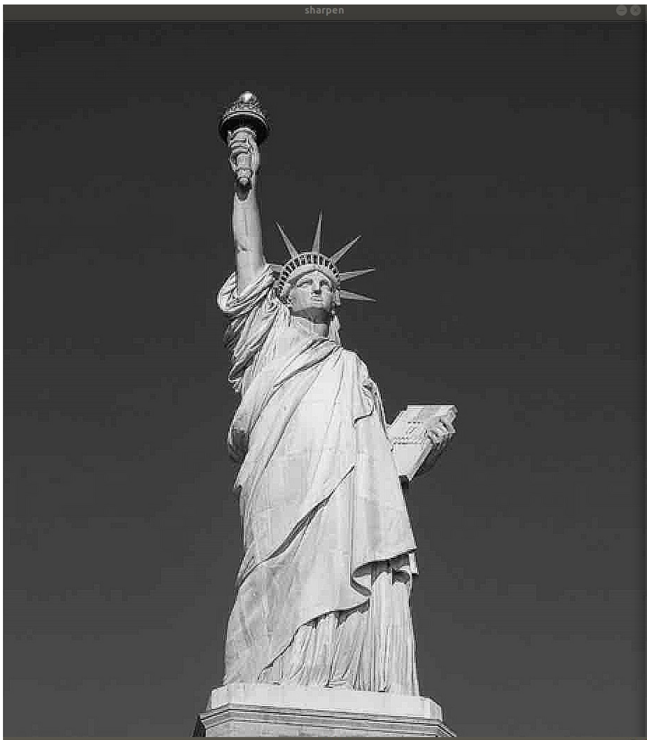


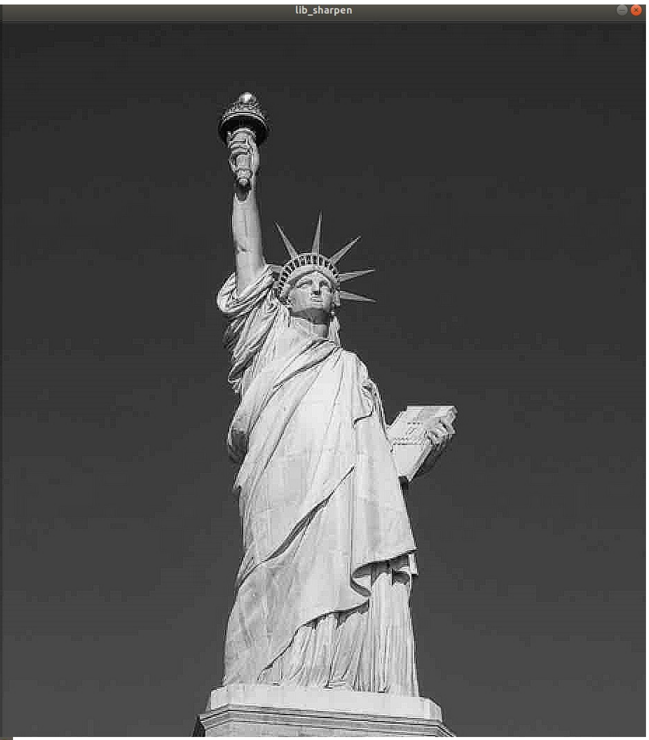
note:

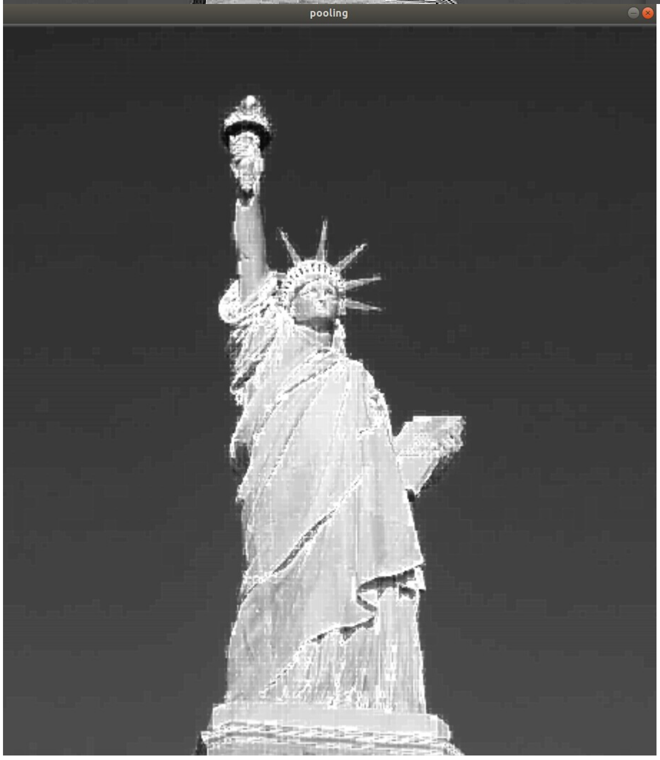
input -> source image
Gray -> source image transform gray
sharpen ->gray do convolution (myself)
lib_sharpen ->gray do convolution
(opencv’s library)
pooling -> sharpen do max pooling

-liberty.png




sharpen


lib_sharpen


pooling