REPORT

**Name:** Pham Le Nguyen Khang

**ID:** 2250007

**Class:** CC11

**Subject:** Computer Architecture

1. INTRODUCTION:

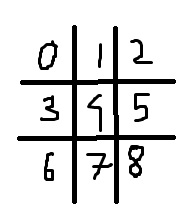
In this modern day and age, artificial intelligence has been developing exponentially fast. Rapid improvements in technique and data lead to a remarkable increase in calculations. Its sub-categories, machine learning and deep learning, have reached a point where billions of calculations must be done to compute the millions of parameters in a model.

Convolutional neural network (CNN) is one of the most widely used type of network used in deep learning. Its usage can be seen in a wide range of applications, particularly in fields that involve image and video analysis. However advanced a CNN model maybe, it always stems from matrix computations and most notably, convolution operations.

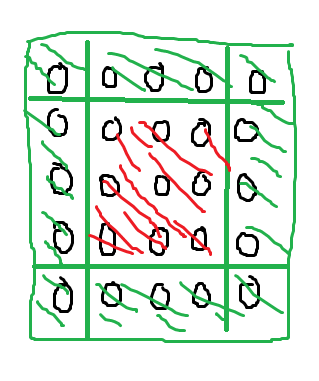
1. IMPLEMENTATION:
2. Apply padding:

One tricky issue when applying convolutional layers is that we tend to lose pixels on the perimeter of our image. One straightforward solution to this problem is to add extra pixels of filler around the boundary of our input image, thus increasing the effective size of the image. Typically, we set the values of the extra pixels to zero.

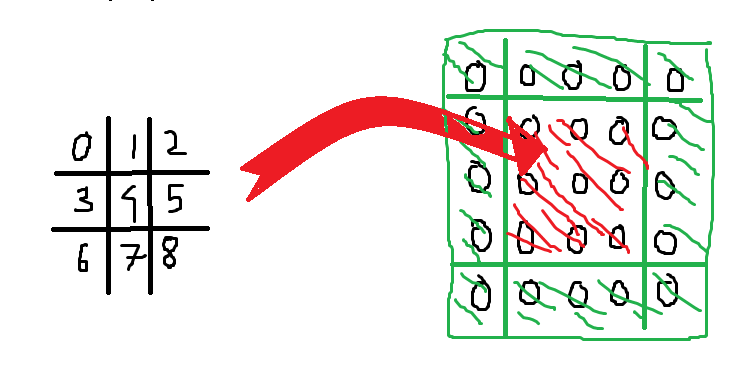
Let the input image matrix as shown:



We initialize the new image matrix and set all values of pixels to zero:



After that, we simply copy the original input image to the middle (the red zone) of the new input image. The **size of new image = size of image + 2 \*padding (New N = N + 2 \* p)**.

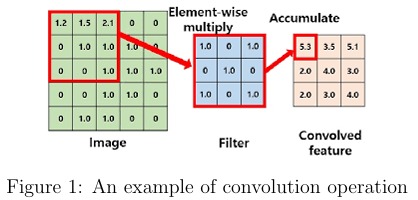


1. Convolution calculation:

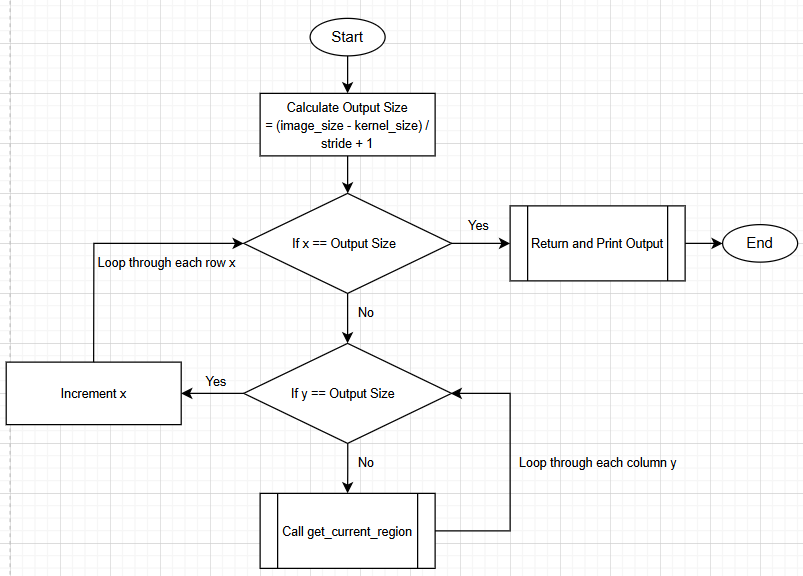
The convolution operation involves sliding a filter, also known as a kernel, across the input image. This filter is a small matrix of weights that is applied to a local region of the input matrix, producing a single value in the output feature map. The process is repeated across the entire matrix, allowing the network to learn various features of the matrix. In the case of image analysis, these features can include edges, textures, and patterns. The mathematical operation performed is essentially a dot product between the filter and the receptive field of the input image.

The process involves taking a small matrix, called a kernel or filter, and sliding it over an input matrix, such as an image. At each position, the dot product of the kernel and the overlapping region of the input matrix is computed, and the result is stored in the output matrix. This process is repeated for all positions of the kernel over the input matrix, effectively blending the kernel with the input to highlight specific features like edges or textures.

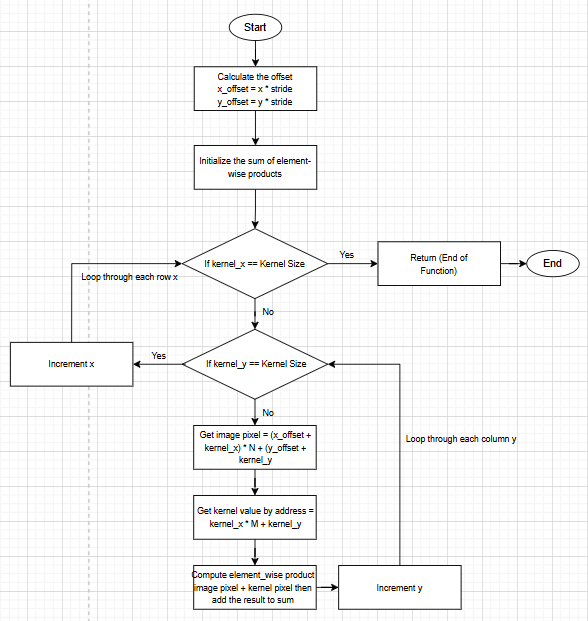
In Figure 1, a 5x5 input matrix and a 3x3 kernel filter have been provided. We first match the kernel onto a corner of the input, in this case the small 3x3 matrix covered in a large red square. We then calculate the sum of element-wise products of both matrices to get a single value. This value will be the first value of the output matrix. Proceed to slide the kernel from left to right, then up to down to cover every square of the input to get the final result. Notice that the stride here is 1 as the kernel move 1 column and 1 row at a time.



Here is the whole implementation of performing convolution calculation:



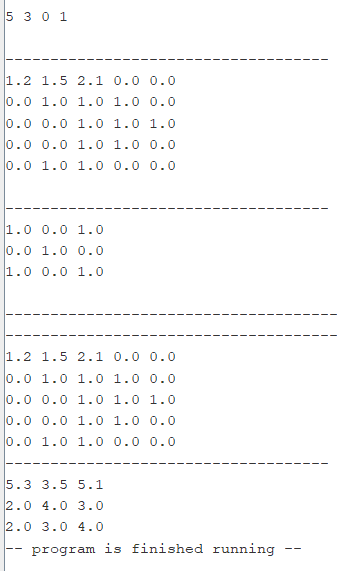
* Function **get\_current\_region()**:



1. DEMO:

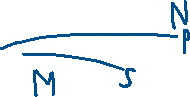
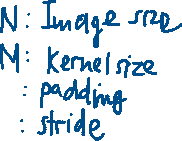
Test 1:



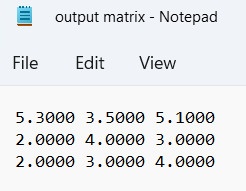


AFTER PADDING

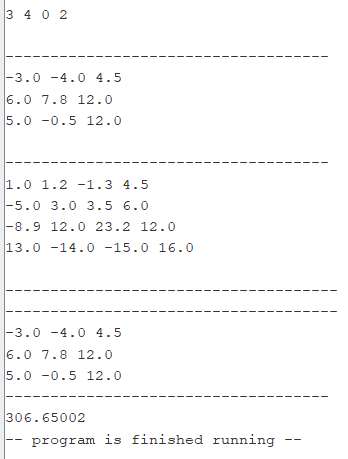
AFTER CONVOLUTION CALCULATION



In output file:



Test 2:

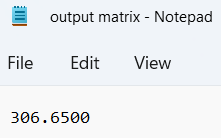


AFTER CONVOLUTION CALCULATION

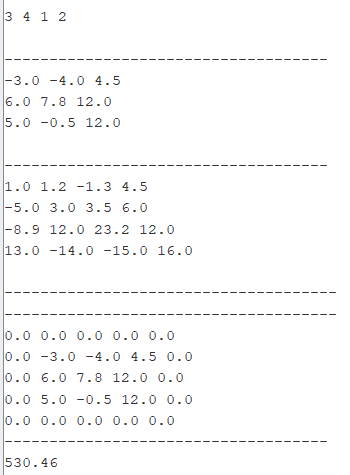
AFTER PADDING



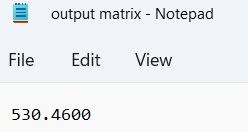
In output file:



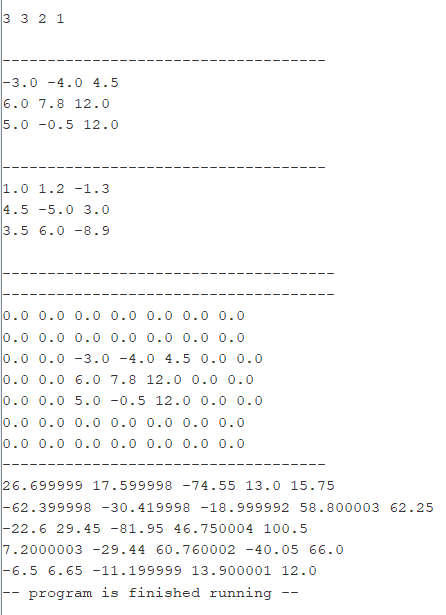
Test 3:



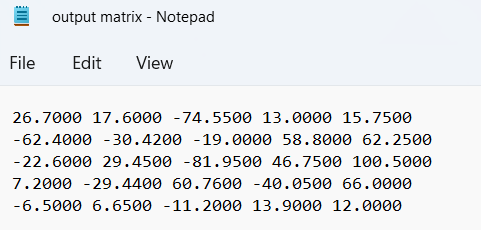
In output file:



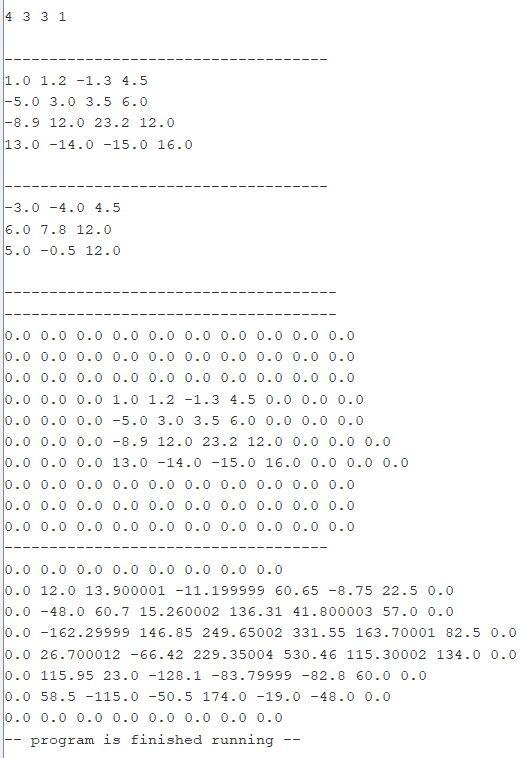
Test 4:



In output file:



Test 5:



In output file:

