Comparative Analysis of Feature Detection Using 2-D Convolution with Various Kernel Sizes

Kim Minseo

May 16, 2024

Abstract

In this project, we explore the application of 2D convolution in image processing using custom kernels. We focus on detecting primary features such as identity, blur, edge, and sharpen in images. Two different kernel sizes, 3x3 and 5x5, are used to investigate the impact of filter size on the effectiveness of feature detection. The convolution operation is implemented manually without using any pre-built library functions to gain a deeper understanding of the process. The resulting images are normalized to a 0-1 gray-scale for analysis. Our findings provide insights into the role of convolution in image processing and the importance of choosing appropriate filter coefficients and sizes for effective feature detection.

Contents

1 Introduction

1.1 Project Overview

The goal of this project is to perform a two dimensional convolution on a given image and analyze the results. To do this, we need to find the two-dimensional kernels used to detect the four main features: identity, blur, edge , and sharpen. Each kernel will be used in 3x3 and 5x5 dimensions. This project focuses on implementing the convolution yourself using Numpy and cv2, rather than using the convolution functions provided by the library. Also, the resulting image should be normalized to a value between 0 and 1.

In the process of analyzing the results of the project, we will compare and analyze the performance as a function of filter coefficients and filter size. This will allow us to gain a deeper understanding of the characteristics of filters and their impact on images. This understanding will help us to grasp the essence of convolution, which plays an important role in fields such as computer vision, image processing, and machine learning. Through this project, we will experience practical applications of these techniques, and the results will teach us how to apply theoretical knowledge to real-world problems.

1.2 Objectives

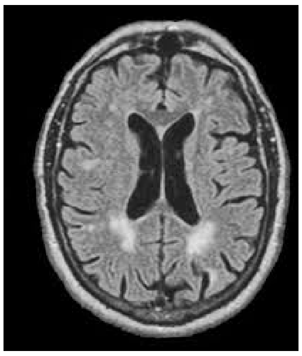


Figure 1: Brain image(gray scale)

The main object of this project is a grayscale brain MRI image with dimensions of 207x243 pixels(Fig. 1). This image is the target of convolution, and we apply different two-dimensional kernels on it to detect and analyze the main features of the image. This task is similar to a real-world application: feature extraction from brain MRI images. This allows us to apply the theoretical concepts of convolution to a real-world problem and learn how to analyze the results.

2 Methods

2.1 Normalization Process

We first loaded a 207x243 pixel brain MRI image, which has been converted to grayscale, where each pixel is represented by an integer with a value between 0 and 255. Before using these images for computer vision tasks, they are typically normalized so that the pixel values are floating point numbers between 0 and 1. This is to prevent overflow when performing convolution operations and to facilitate comparisons between different images. In this project, we used Numpy to normalize the image by converting it to float32 type and dividing by 255.

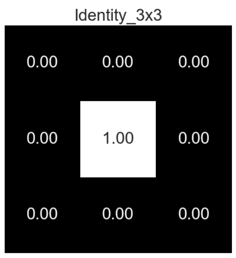
2.2 Image Processing and Convolution Concept

In this project, we will apply convolutional theory to real-world image processing. Convolution is a process of local multiplication and summation between an input image and a filter (or kernel) to create a new 'feature map', which is used to detect and analyze the main features of an image. The kernels used here are typically small, two-dimensional matrices that are set to different values to apply different effects.

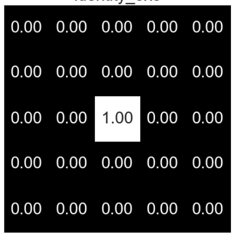
2.3 Definition of Kernels

In this project, we apply the theory of convolution to real-world image processing. Convolution is a process of local multiplication and summation between an input image and a filter (or kernel) to create a new 'feature map', which is used to detect and analyze the main features of an image. The kernels used here are typically small, two-dimensional matrices that are set to different values to apply different effects.

2.3.1 Identity Kernel


Identity_3x3
0.00 0.00 0.00
0.00 1.00 0.00
0.00 0.00 0.00

Identity_5x5



0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	1.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00

The identity kernel is responsible for keeping the original image intact. It consists of a 3x3 or 5x5 matrix with a center value of 1 and all other values being 0.

2.3.2 Blur Kernel

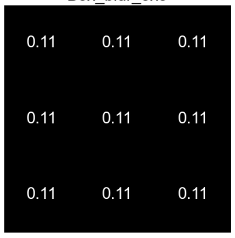
The blur kernel is responsible for blurring the image. This kernel uses techniques such as Box blur or Gaussian blur, where all values are equal.

Box blur Box blur, also known as mean blur, is a simple sliding window spatial filter that replaces the center value in the window with the average (mean) of all other pixels in the window. This results in a uniform blur in the image. The box blur is represented as follows.

$$\text{Box blur 3x3} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix} \tag{1}$$

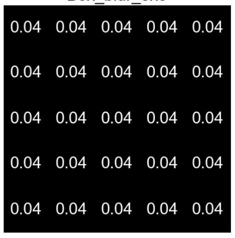
$$\text{Box blur 5x5} = \frac{1}{25}\begin{bmatrix}1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\end{bmatrix} \tag{2}$$

Box_blur_3x3



0.11	0.11	0.11
0.11	0.11	0.11
0.11	0.11	0.11

Box_blur_5x5



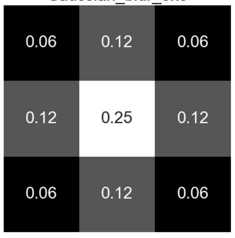
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04

Gaussian blur Gaussian blur is a method that blurs an image by applying a Gaussian function. It is a widely used effect in graphics software, typically to reduce image noise and detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen.

$$\text{Gaussian blur 3x3} = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix} \tag{3}$$

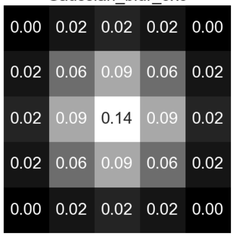
$$\text{Gaussian blur 5x5} = \frac{1}{256}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\4 & 16 & 24 & 16 & 4\\6 & 24 & 36 & 24 & 6\\4 & 16 & 24 & 16 & 4\\1 & 4 & 6 & 4 & 1\end{bmatrix} \tag{4}$$

Gaussian_blur_3x3



0.06	0.12	0.06
0.12	0.25	0.12
0.06	0.12	0.06

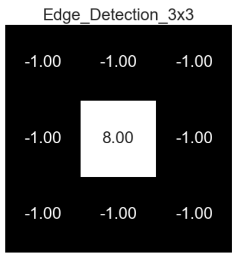
Gaussian_blur_5x5



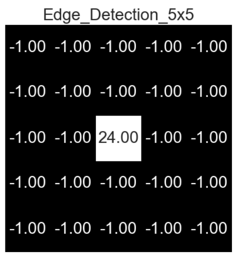
0.00	0.02	0.02	0.02	0.00
0.02	0.06	0.09	0.06	0.02
0.02	0.09	0.14	0.09	0.02
0.02	0.06	0.09	0.06	0.02
0.00	0.02	0.02	0.02	0.00

2.3.3 Edge Detection Kernel

Edge detection is a fundamental operation in image processing used to locate object boundaries in an image. The edge detection kernel, also known as an edge filter, enhances the edge contrast of an image, making it easier to identify individual objects.


Edge_Detection_3x3

-1.00	-1.00	-1.00
-1.00	8.00	-1.00
-1.00	-1.00	-1.00

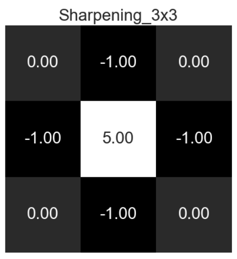

Edge_Detection_5x5

-1.00	-1.00	-1.00	-1.00	-1.00
-1.00	-1.00	-1.00	-1.00	-1.00
-1.00	-1.00	24.00	-1.00	-1.00
-1.00	-1.00	-1.00	-1.00	-1.00
-1.00	-1.00	-1.00	-1.00	-1.00

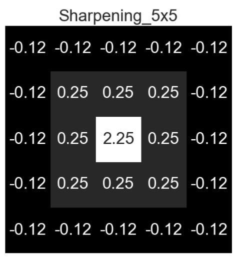
This kernel is usually formed over two or more steps, but since the project aims to use a single kernel, we will create a kernel that allows for edge detection through a single kernel.

2.3.4 Sharpen Kernel

The sharpen kernel is used to make the image sharper. Sharpening is the process of enhancing the edge contrast of an image to make it appear more defined. This is often used in image processing to enhance the details of an image.


Sharpening_3x3

0.00	-1.00	0.00
-1.00	5.00	-1.00
0.00	-1.00	0.00


Sharpening_5x5

-0.12	-0.12	-0.12	-0.12	-0.12
-0.12	0.25	0.25	0.25	-0.12
-0.12	0.25	2.25	0.25	-0.12
-0.12	0.25	0.25	0.25	-0.12
-0.12	-0.12	-0.12	-0.12	-0.12

The sharpen kernel works by emphasizing the high-frequency components of the image, which correspond to rapid changes in color and brightness. This has the effect of making edges and other high-frequency components in the image more distinct.

2.4 Convolution Procedure

The convolution process is as follows: First, padding is added to the input image to allow the kernel to treat all parts of the image equally. Next, the kernel is slid over the image, performing an element-wise multiplication between the kernel and the corresponding part of the image at each location, and summing the results. We store this value at the corresponding pixel in the resulting image. Repeat this process for all image locations. Finally, we normalize the resulting image back to a value between 0 and 1. In this project, we implemented this process using the Numpy library.

3 Results

3.1 Applying Normalization

Through the process of image normalization, we converted the original gray scale image to a value between 0 and 1. This prevents overflow when performing convolution operations and facilitates comparisons between different images. Figure 2 visualizes this process.

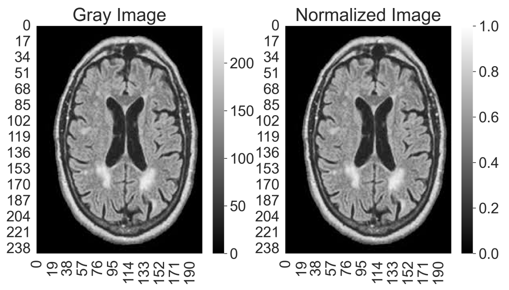


Figure 2: Original gray scale image and normalized image

3.2 Applying Identity Kernel

The Identity Kernel does not make any changes to the image. It is primarily used to understand how convolutional operations affect an image. When we apply the Identity Kernel to an image, it outputs an image that is identical to the original image. This is because each element of the kernel is multiplied by the corresponding pixel in the image, and the results are added together to become the value of that pixel in the output image. This process is visualized in Figure 3.

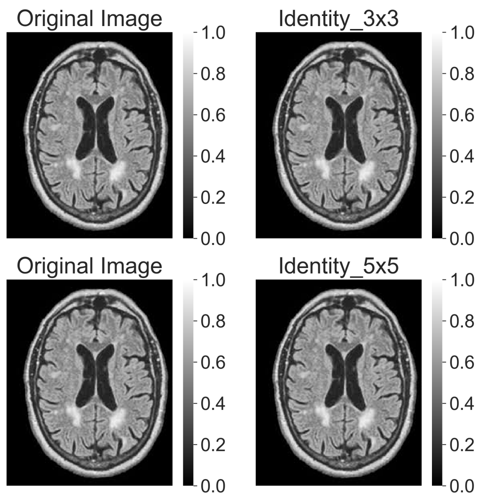


Figure 3: Result of applying Identity Kernel

The advantage of using the Identity Kernel is that it allows us to perform convolutional operations without making any changes to the image. This helps us understand how the output of the convolutional layer affects the input. It also helps us understand how each kernel affects the image compared to the other kernels.

3.3 Applying Blur Kernel

The two types of blur kernels used in this project, the box blur kernel and the Gaussian blur kernel, work in different ways and produce slightly different results. 4

The box blur kernel, also known as a mean filter, simply takes the average of all the pixels in its neighborhood. This has the effect of blurring the image, but it can also result in a noticeable "boxy" appearance to the blur, especially for larger kernel sizes.

On the other hand, the Gaussian blur kernel applies weights to the pixels in its neighborhood according to a Gaussian distribution. This means that pixels closer to the center of the kernel have a greater influence on the output than those further away. This results in a smoother, more natural-looking blur, but it is also more computationally intensive than the box blur kernel.

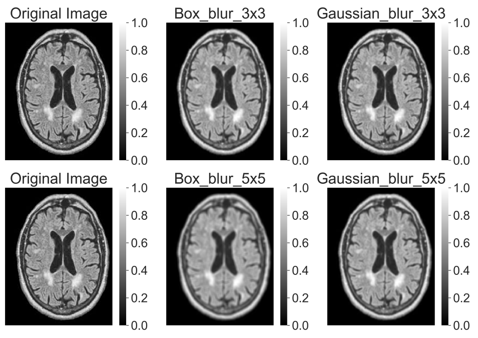


Figure 4: Result of applying blur kernel

As for the difference between the 3x3 and 5x5 kernels, the larger kernel size results in a more pronounced blur. This is because the larger kernel averages over a larger neighborhood of pixels, effectively blending together more of the image's details. However, the trade-off is that a larger kernel requires more computational resources to apply, and it may also remove too much detail from the image, depending on the application.

3.4 Applying Edge Detection Kernel

Edge detection is a key image processing operation that identifies points in an image where the brightness changes sharply, typically marking the boundaries of objects. The edge detection kernels applied in Figure 6 emphasize these boundaries, making them more apparent.

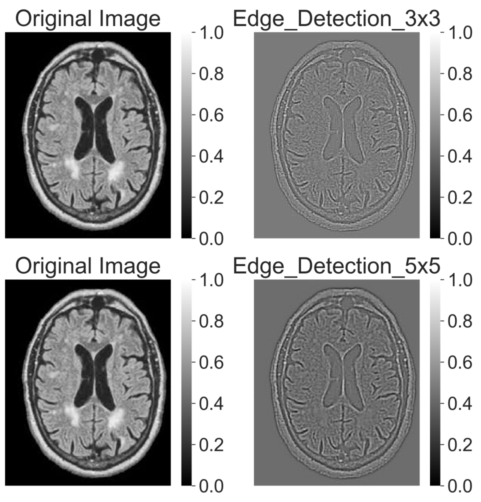


Figure 6: Result of applying edge detection

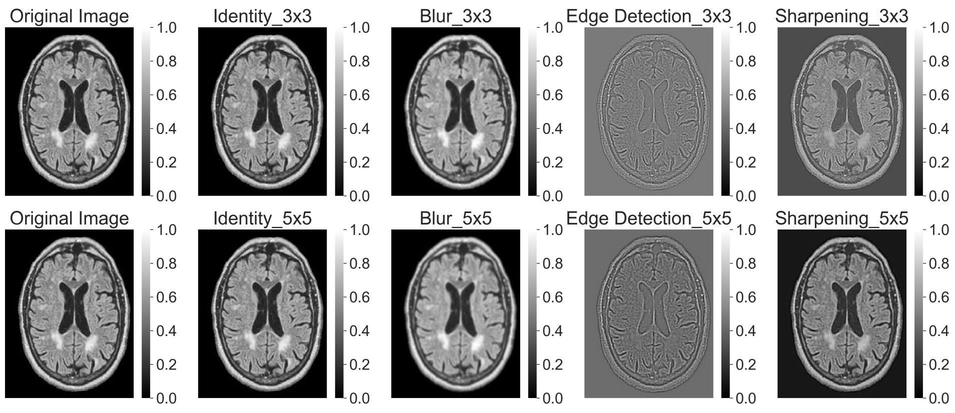


Figure 5: Result of all kernel

The 3x3 kernel considers a neighborhood of 3 pixels in width and height around each pixel, while the 5x5 kernel considers a neighborhood of 5 pixels in width and height. The larger the kernel, the larger the neighborhood of pixels that is considered for each output pixel, and the more global the resulting edge detection will be.

3.5 Applying Sharpen Kernel

Sharpening is a technique used in image processing to enhance the details in an image. This is particularly useful in tasks such as text recognition, where fine details are important. The sharpening operation works by enhancing the contrast of pixels that border light and dark areas, effectively increasing the image's local contrast.

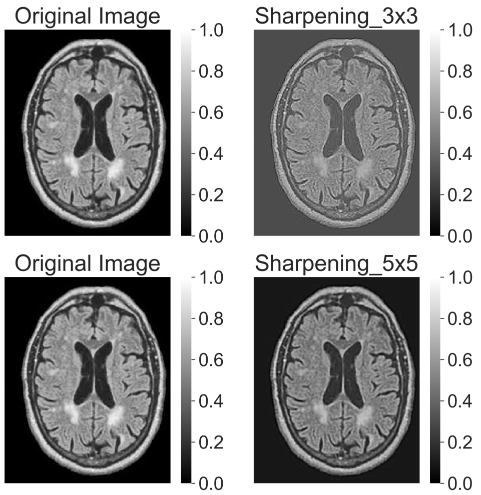


Figure 7: Result of applying Sharpen

The 3x3 kernel has a more localized effect, sharpening only the very fine details. On the other hand, the 5x5 kernel has a more global effect, sharpening larger features as well as fine details. The results of applying these kernels can be seen in Figure 7. The sharpening effect is more pronounced with the 5x5 kernel due to its larger size, which takes into account a larger neighborhood of pixels.

4 Discussion

4.1 Interpretation of Results

The results obtained in this project show the effect of applying 2D convolution to image processing using kernels of different sizes. Each kernel, which can be seen in Figure 5, was used to detect a specific feature of the image, which gives an idea of what information is being extracted from the image.

4.2 Impact of Kernel Size and Coefficients on Results

The coefficients within the kernel play a vital role in the outcome of the convolution process. These coefficients, also known as weights, determine how much each pixel in the image contributes to the final output of the convolution operation.

The relative values of the coefficients are crucial. For instance, a kernel designed to detect edges might have positive coefficients in one part and negative in another. This is because we want to amplify the difference between the pixel values in different regions of the image. A significant change in pixel values indicates an edge, and the kernel is designed to highlight this change.

The absolute values of the coefficients are also important. Larger absolute values will result in a more pronounced effect on the final output, amplifying the features that the kernel is designed to detect. Conversely, smaller absolute values will result in a less pronounced effect, making the detected features more subtle.

Moreover, the sum and the variance of the coefficients can have a significant impact on the output image. If the sum of the coefficients is greater than 1, the output image can be brighter than the original, and if it's less than 1, the output image can be darker.

In addition, the variance of the coefficients is also crucial. The variance indicates the spread of the weights in the kernel. A kernel with a high variance will emphasize or suppress features more strongly, leading to a high-contrast output image. Conversely, a kernel with a low variance will result in a more subtle output, preserving more of the original image's details.

In our project, we normalized the kernels, which means we adjusted the coefficients so that their sum is 1 and their variance is a specific value. This normalization process ensures that the convolution operation does not change the overall brightness of the image while allowing us to control the contrast of the output image by adjusting the variance.

4.3 Comparison of 3x3 and 5x5 Kernels

Comparing the 3x3 and 5x5 kernels, we can see that both kernels are effective at detecting features, but their effectiveness depends on the size of the kernel. In general, the 5x5 kernel considers a larger area of the image, so it can be more effective at detecting the overall structure of the image. On the other hand, a 3x3 kernel may be more effective at detecting detailed features of an image because it considers a smaller region.

5 Conclusions

5.1 Summary of Findings

In this project, we used 2D convolution to perform image processing and explored how different sized kernels can be used to detect features in an image. Our results show that the size and coefficients of the kernel have a significant impact on the results of image processing, which provides important insights for fields such as computer vision, image processing, and machine learning.

5.2 Implications and Future Directions

The results of this project provide a deeper understanding of the importance of convolution and how to use it effectively. Future research directions could include using kernels of different sizes and shapes to detect more complex features, or to perform more complex image processing tasks.