Kaung Htet Cho

st124092

Morphological Transformation

We will learn different morphological operations like Erosion, Dilation, Opening, Closing etc.

Reference: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html

Theory

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs **two inputs**, one is our original image, **second one is called structuring element or kernel** which decides the nature of operation. Two basic morphological operators are

- Erosion and
- Dilation.

Then its variant forms like Opening, Closing, Gradient etc also comes into play.

We will see them one-by-one with help of following image:

EROSION

- The kernel slides through the image (as in 2D convolution).
- A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1,
- otherwise it is eroded (made to zero).

In [ ]:
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
```

```python
plt.figure(figsize=(12, 6))
img = cv.imread('assets/j.png', cv.IMREAD_GRAYSCALE)
assert img is not None, "file could not be read, check with os.path.exist
kernel = np.ones((5,5),np.uint8)
erosion = cv.erode(img,kernel,iterations = 1)

plt.subplot(221), plt.imshow(img, cmap='gray'), plt.title('Orignal Image'
plt.subplot(222), plt.imshow(erosion, cmap='gray'), plt.title('Eroded Ima
plt.show()
```

DILATION

- Here, a pixel element is '1' if at least one pixel under the kernel is '1'.
- So it increases the white region in the image or size of foreground object increases.

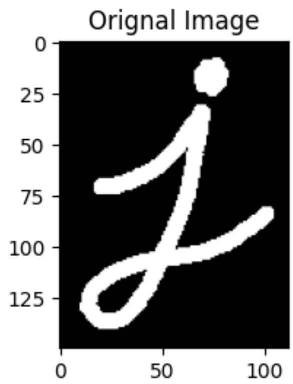
In [ ]:
```python
kernel = np.ones((5,5),np.uint8)
dilation = cv.dilate(img,kernel,iterations = 1)

plt.figure(figsize=(12, 6))
plt.subplot(221), plt.imshow(img, cmap='gray'), plt.title('Orignal Image'
plt.subplot(222), plt.imshow(dilation, cmap='gray'), plt.title('Dilated I
plt.show()
```

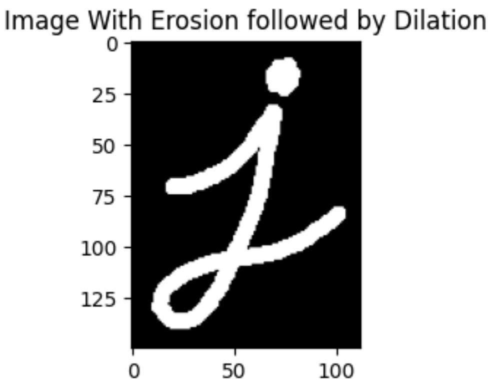
OPENING

- Errosion Followed by Dilation
- Normally, in cases like noise removal, erosion is followed by dilation.
- Erosion removes white noises, but it also shrinks our object.

- So we dilate it. (Meaning we increse the area of foreground image).
- Since noise is gone, they won't come back, but our object area increases.

In [ ]:
```
kernel = np.ones((3,3),np.uint8)
opening = cv.morphologyEx(img, cv.MORPH_OPEN, kernel)

plt.figure(figsize=(12, 6))
plt.subplot(221), plt.imshow(img, cmap='gray'), plt.title('Orignal Image'
plt.subplot(222), plt.imshow(opening, cmap='gray'), plt.title('Image With
plt.show()
```

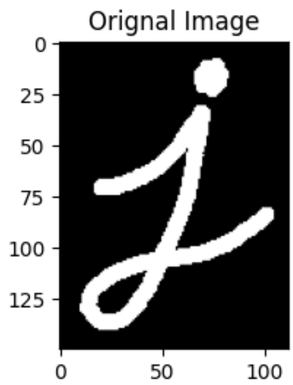


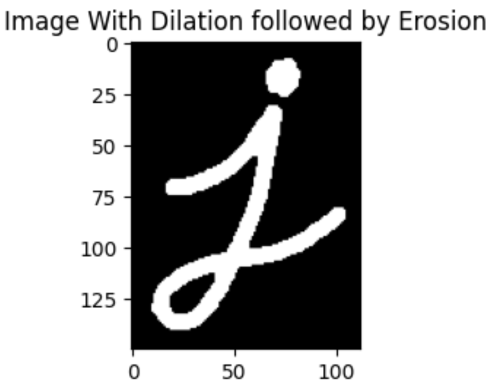


CLOSING

- Closing is reverse of Opening, Dilation followed by Erosion.
- useful in closing small holes inside the foreground objects, or small black points on the object.

In [ ]:
```
kernel = np.ones((3,3),np.uint8)
closing = cv.morphologyEx(img, cv.MORPH_CLOSE, kernel)

plt.figure(figsize=(12, 6))
plt.subplot(221), plt.imshow(img, cmap='gray'), plt.title('Orignal Image'
plt.subplot(222), plt.imshow(closing, cmap='gray'), plt.title('Image With
plt.show()
```





Morphological Gradient

It is the difference between dilation and erosion of an image.

We will get the outline of the object.

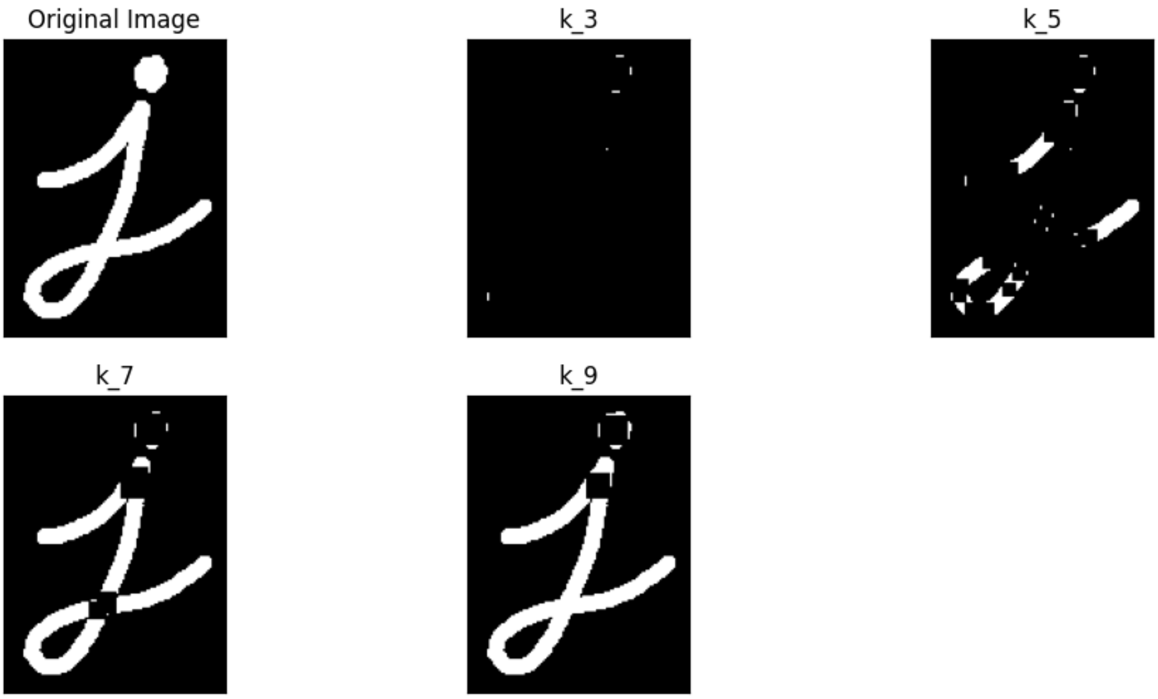
In [ ]:
```
kernel = np.ones((3,3),np.uint8)
gradient = cv.morphologyEx(img, cv.MORPH_GRADIENT, kernel)

plt.figure(figsize=(12, 6))
plt.subplot(221), plt.imshow(img, cmap='gray'), plt.title('Orignal Image'
plt.subplot(222), plt.imshow(gradient, cmap='gray'), plt.title('Image Wit
plt.show()
```

Top Hat

It is the difference between input image and Opening of the image.

Lets try with different kernels.

In [ ]:
```
k = [5,7,9,11]
tophat = []
for i in range(len(k)):
    kernel = np.ones((k[i],k[i]),np.uint8)
    tophat.append(cv.morphologyEx(img, cv.MORPH_TOPHAT, kernel))

titles = ['Original Image','k_3', 'k_5', 'k_7', 'k_9']
images = [img] + tophat

plt.figure(figsize=(12, 6))
for i in range(len(titles)):
    plt.subplot(2,3,i+1),plt.imshow(images[i], cmap='gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```

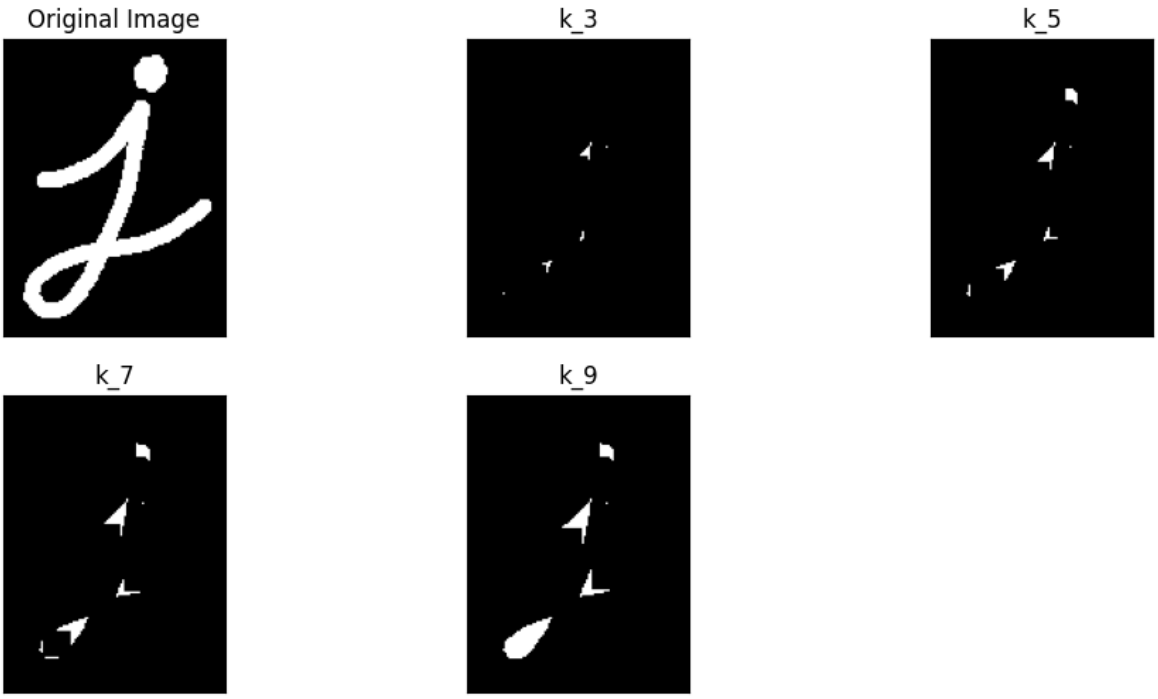


Black Hat

It is the difference between the closing of the input image and input image.

Lets try with different kernels.

In [ ]:
```
k = [5,7,9,11]
blackhat = []
for i in range(len(k)):
    kernel = np.ones((k[i],k[i]),np.uint8)
    blackhat.append(cv.morphologyEx(img, cv.MORPH_BLACKHAT, kernel))

titles = ['Original Image','k_3', 'k_5', 'k_7', 'k_9']
images = [img] + blackhat

plt.figure(figsize=(12, 6))
for i in range(len(titles)):
    plt.subplot(2,3,i+1),plt.imshow(images[i], cmap='gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```



ASSIGNMENT (20 POINTS)

In [ ]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread('brain.jpg')
plt.imshow(img)
plt.title('Original Image')
plt.axis('off')
```

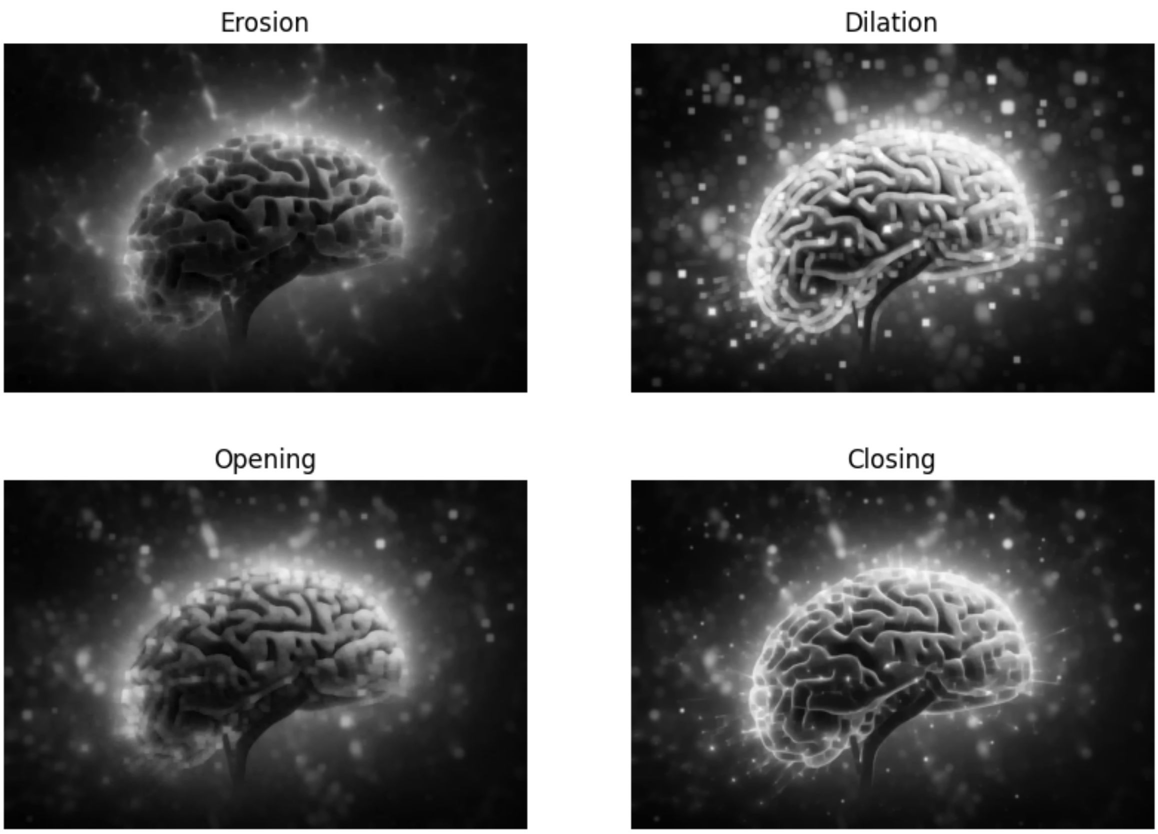
Out[ ]: (-0.5, 539.5, 359.5, -0.5)

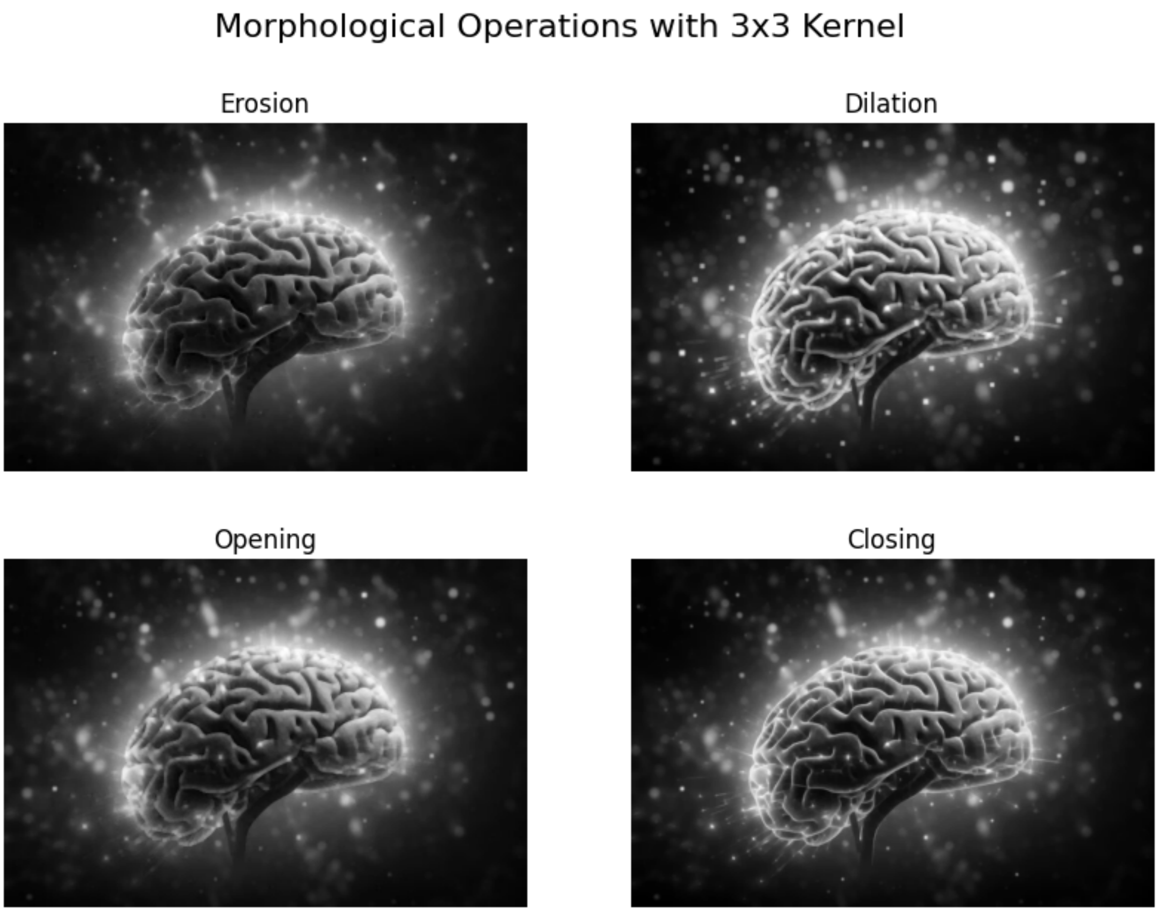
Original Image

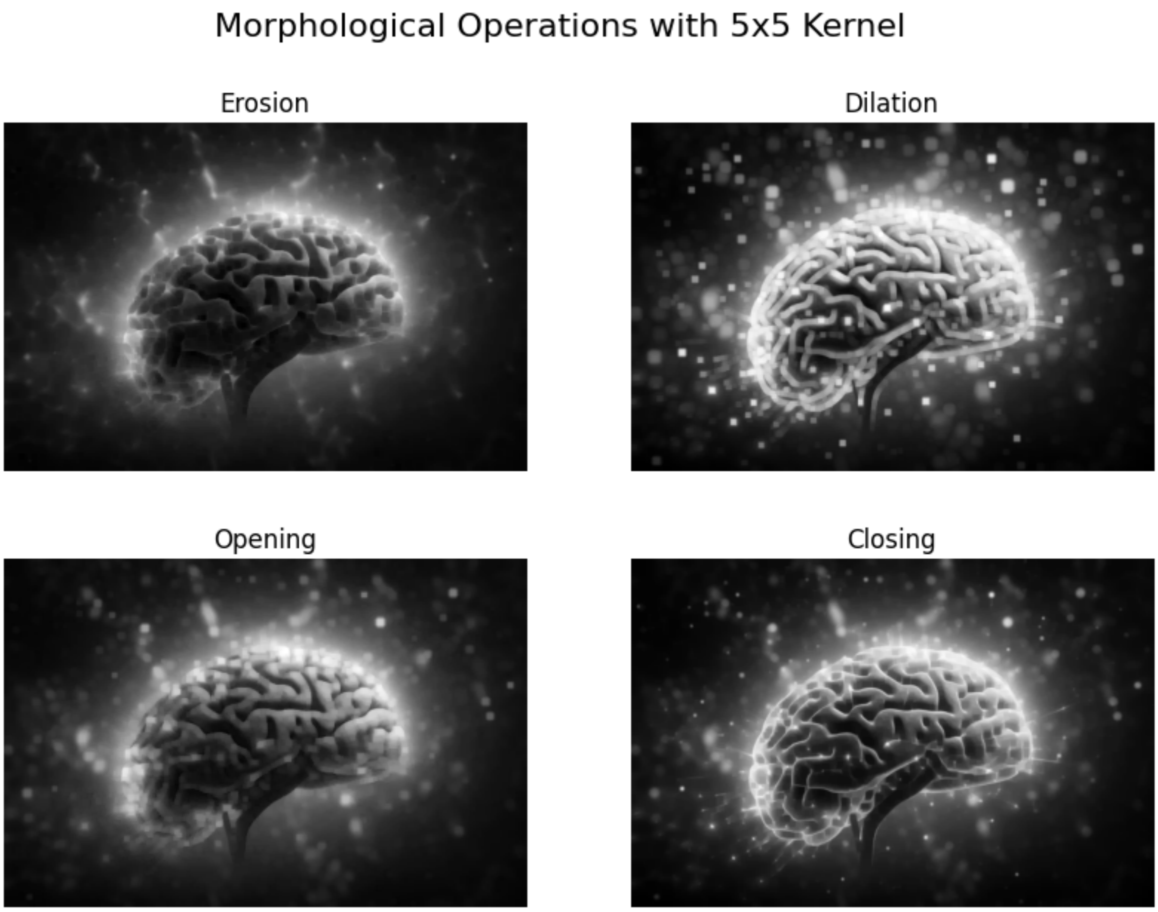
In [ ]:
```
import cv2
```

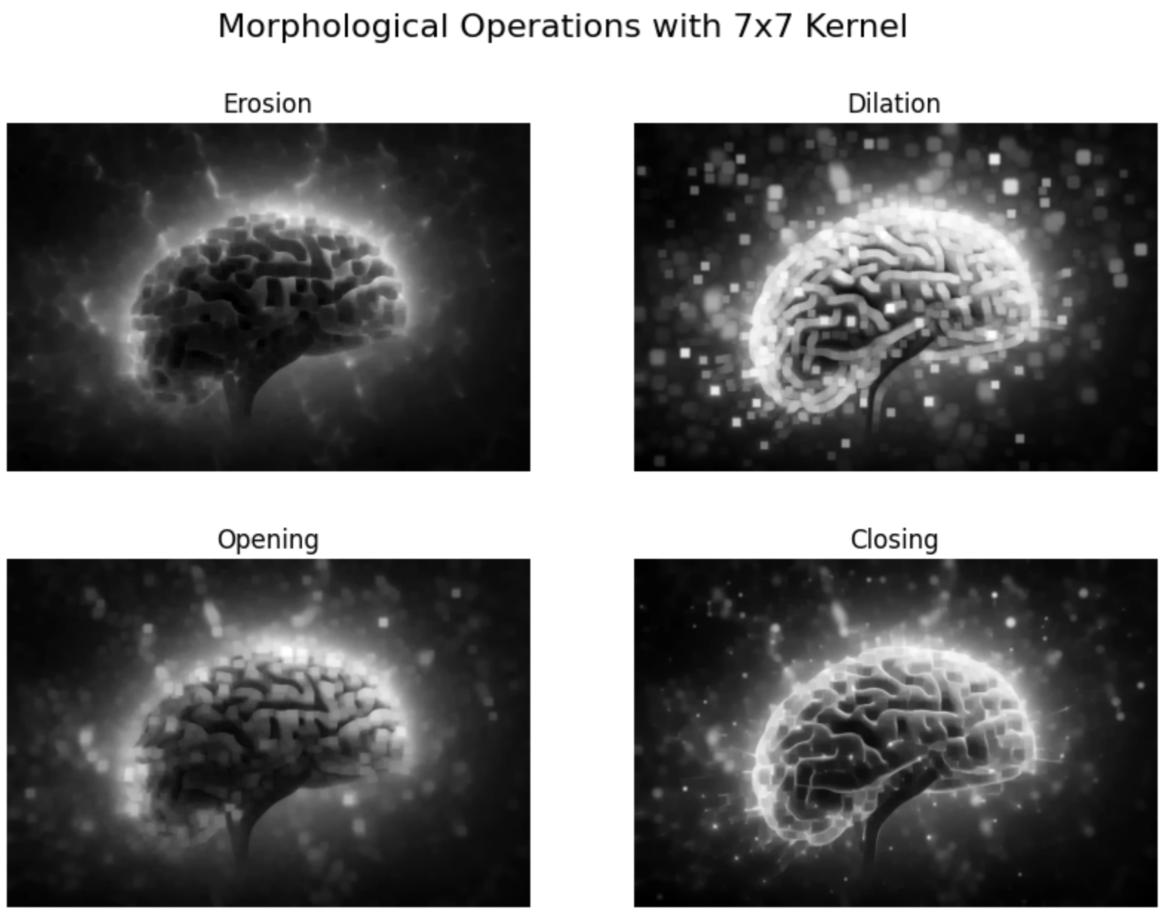
```
import numpy as np
import matplotlib.pyplot as plt

# Step 1: Load the image
img = cv2.imread('brain.jpg', cv2.IMREAD_GRAYSCALE)
assert img is not None, "File could not be read, check with os.path.exist

# Step 2: Define a kernel for morphological operations
kernel = np.ones((5, 5), np.uint8)

# Step 3: Apply morphological operations
erosion_bgr = cv2.erode(img, kernel, iterations=1)
dilation_bgr = cv2.dilate(img, kernel, iterations=1)
opening_bgr = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
closing_bgr = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)

erosion = cv2.cvtColor(erosion_bgr, cv2.COLOR_BGR2RGB)
dilation = cv2.cvtColor(dilation_bgr, cv2.COLOR_BGR2RGB)
opening = cv2.cvtColor(opening_bgr, cv2.COLOR_BGR2RGB)
closing = cv2.cvtColor(closing_bgr, cv2.COLOR_BGR2RGB)




# Step 4: Visualize the results
plt.figure(figsize=(10, 7))
plt.subplot(221), plt.imshow(erosion, cmap='gray'), plt.title('Erosion')
plt.axis('off')
plt.subplot(222), plt.imshow(dilation, cmap='gray'), plt.title('Dilation'
plt.axis('off')
plt.subplot(223), plt.imshow(opening, cmap='gray'), plt.title('Opening')
plt.axis('off')
plt.subplot(224), plt.imshow(closing, cmap='gray'), plt.title('Closing')
plt.axis('off')
plt.show()



# Step 5: Experiment with different kernels to visualize and discuss the

# Define different kernels
kernels = {
    "3x3": np.ones((3, 3), np.uint8),
    "5x5": np.ones((5, 5), np.uint8),
    "7x7": np.ones((7, 7), np.uint8)
}

# Apply and visualize with different kernels
for name, kernel in kernels.items():
    erosion = cv2.erode(img, kernel, iterations=1)
    dilation = cv2.dilate(img, kernel, iterations=1)
    opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
    closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)

    plt.figure(figsize=(10, 7))
    plt.suptitle(f'Morphological Operations with {name} Kernel', fontsize
    plt.subplot(221), plt.imshow(erosion, cmap='gray'), plt.title('Erosio
    plt.axis('off')
    plt.subplot(222), plt.imshow(dilation, cmap='gray'), plt.title('Dilat
    plt.axis('off')
```

```
    plt.subplot(223), plt.imshow(opening, cmap='gray'), plt.title('Openin
    plt.axis('off')
    plt.subplot(224), plt.imshow(closing, cmap='gray'), plt.title('Closin
    plt.axis('off')
    plt.show()

# Discussion:
# - Smaller kernels (e.g., 3x3) result in less aggressive morphological o
#   preserving more details in the image.
# - Larger kernels (e.g., 7x7) lead to more significant changes, such as
#   areas being eroded or dilated, which can be useful for removing noise
#   separating connected objects.
# - Opening is useful for removing small objects from the foreground (bla
#   while closing is useful for closing small holes in the foreground.
```




Morphological Operations with 3x3 Kernel
Erosion
Dilation
Opening
Closing


Morphological Operations with 5x5 Kernel
Erosion
Dilation
Opening
Closing


Morphological Operations with 7x7 Kernel
Erosion
Dilation
Opening
Closing