Filters / Kernels / Masks in Image Processing

Bindeshwar Singh Kushwaha
PostNetwork Academy

What is a Filter / Kernel / Mask?

- A **filter** (also called a kernel or mask) is a small matrix of numbers used to transform an image.

What is a Filter / Kernel / Mask?

- A **filter** (also called a kernel or mask) is a small matrix of numbers used to transform an image.
- Purpose: Emphasize certain features like edges, textures, or smoothing details.

What is a Filter / Kernel / Mask?

- A **filter** (also called a kernel or mask) is a small matrix of numbers used to transform an image.
- Purpose: Emphasize certain features like edges, textures, or smoothing details.
- Filters are applied to an image using **convolution** or correlation.

What is a Filter / Kernel / Mask?

- A **filter** (also called a kernel or mask) is a small matrix of numbers used to transform an image.
- Purpose: Emphasize certain features like edges, textures, or smoothing details.
- Filters are applied to an image using **convolution** or correlation.
- During convolution, the filter "slides" over the image, computing a weighted sum of neighboring pixels to produce the output.

Types of Filters and Their Effects

- Filters are used to achieve different image processing effects:

Types of Filters and Their Effects

- Filters are used to achieve different image processing effects:
  - **Smoothing / Blurring:** Reduces noise, softens edges.

Types of Filters and Their Effects

- Filters are used to achieve different image processing effects:
  - **Smoothing / Blurring:** Reduces noise, softens edges.
  - **Sharpening:** Enhances edges and fine details.

Types of Filters and Their Effects

- Filters are used to achieve different image processing effects:
    - **Smoothing / Blurring:** Reduces noise, softens edges.
    - **Sharpening:** Enhances edges and fine details.
    - **Edge Detection:** Highlights boundaries in intensity changes.

Types of Filters and Their Effects

- Filters are used to achieve different image processing effects:
    - **Smoothing / Blurring:** Reduces noise, softens edges.
    - **Sharpening:** Enhances edges and fine details.
    - **Edge Detection:** Highlights boundaries in intensity changes.
- Typical filter sizes: $\mathbf{3} \times \mathbf{3}$, $\mathbf{5} \times \mathbf{5}$, $\mathbf{7} \times \mathbf{7}$.

Types of Filters and Their Effects

- Filters are used to achieve different image processing effects:
  - **Smoothing / Blurring:** Reduces noise, softens edges.
  - **Sharpening:** Enhances edges and fine details.
  - **Edge Detection:** Highlights boundaries in intensity changes.
- Typical filter sizes: $\mathbf{3} \times \mathbf{3}$, $\mathbf{5} \times \mathbf{5}$, $\mathbf{7} \times \mathbf{7}$.
- Each element in the filter is called a **weight** and determines the influence of surrounding pixels.

Mathematics of Convolution

- The output image is computed as the convolution of input image $\boldsymbol{I}$ with filter $\boldsymbol{K}$:

$$(I * K)(x, y) = \sum_{i} \sum_{j} I(x + i, y + j)\, K(i, j)$$

Mathematics of Convolution

- The output image is computed as the convolution of input image $\boldsymbol{I}$ with filter $\boldsymbol{K}$:

$$(I * K)(x, y) = \sum_{i} \sum_{j} I(x + i, y + j)\, K(i, j)$$

- Explanation of terms:

Mathematics of Convolution

- The output image is computed as the convolution of input image $\boldsymbol{I}$ with filter $\boldsymbol{K}$:

$$\boldsymbol{(I * K)(x, y) = \sum_{i} \sum_{j} I(x + i, y + j)\, K(i, j)}$$

- Explanation of terms:
  - $\boldsymbol{I}$ = input image

Mathematics of Convolution

- The output image is computed as the convolution of input image $\boldsymbol{I}$ with filter $\boldsymbol{K}$:

$$(I * K)(x, y) = \sum_{i} \sum_{j} I(x + i, y + j)\, K(i, j)$$

- Explanation of terms:
  - $\boldsymbol{I}$ = input image
  - $\boldsymbol{K}$ = filter/kernel/mask

Mathematics of Convolution

- The output image is computed as the convolution of input image $\boldsymbol{I}$ with filter $\boldsymbol{K}$:

$$\boldsymbol{(I * K)(x, y) = \sum_{i} \sum_{j} I(x + i, y + j)\, K(i, j)}$$

- Explanation of terms:
  - $\boldsymbol{I}$ = input image
  - $\boldsymbol{K}$ = filter/kernel/mask
  - $\boldsymbol{(x, y)}$ = coordinates of the current output pixel

Mathematics of Convolution

- The output image is computed as the convolution of input image $\boldsymbol{I}$ with filter $\boldsymbol{K}$:

$$(\boldsymbol{I} * \boldsymbol{K})(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{i}} \sum_{\boldsymbol{j}} \boldsymbol{I}(\boldsymbol{x} + \boldsymbol{i}, \boldsymbol{y} + \boldsymbol{j})\, \boldsymbol{K}(\boldsymbol{i}, \boldsymbol{j})$$

- Explanation of terms:
  - $\boldsymbol{I}$ = input image
  - $\boldsymbol{K}$ = filter/kernel/mask
  - $(\boldsymbol{x}, \boldsymbol{y})$ = coordinates of the current output pixel
  - $(\boldsymbol{i}, \boldsymbol{j})$ = coordinates inside the filter

Mathematics of Convolution

- The output image is computed as the convolution of input image $\boldsymbol{I}$ with filter $\boldsymbol{K}$:

$$(\boldsymbol{I} * \boldsymbol{K})(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{i}} \sum_{\boldsymbol{j}} \boldsymbol{I}(\boldsymbol{x} + \boldsymbol{i}, \boldsymbol{y} + \boldsymbol{j})\, \boldsymbol{K}(\boldsymbol{i}, \boldsymbol{j})$$

- Explanation of terms:
  - $\boldsymbol{I}$ = input image
  - $\boldsymbol{K}$ = filter/kernel/mask
  - $(\boldsymbol{x}, \boldsymbol{y})$ = coordinates of the current output pixel
  - $(\boldsymbol{i}, \boldsymbol{j})$ = coordinates inside the filter
- Step-by-step convolution:

Mathematics of Convolution

- The output image is computed as the convolution of input image $\boldsymbol{I}$ with filter $\boldsymbol{K}$:

$$\boldsymbol{(I * K)(x, y) = \sum_{i} \sum_{j} I(x + i, y + j)\, K(i, j)}$$

- Explanation of terms:
  - $\boldsymbol{I}$ = input image
  - $\boldsymbol{K}$ = filter/kernel/mask
  - $\boldsymbol{(x, y)}$ = coordinates of the current output pixel
  - $\boldsymbol{(i, j)}$ = coordinates inside the filter
- Step-by-step convolution:
  1. Place the filter at the top-left corner of the image.

Mathematics of Convolution

- The output image is computed as the convolution of input image $\boldsymbol{I}$ with filter $\boldsymbol{K}$:

$$(\boldsymbol{I} * \boldsymbol{K})(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{i}} \sum_{\boldsymbol{j}} \boldsymbol{I}(\boldsymbol{x} + \boldsymbol{i}, \boldsymbol{y} + \boldsymbol{j})\, \boldsymbol{K}(\boldsymbol{i}, \boldsymbol{j})$$

- Explanation of terms:
  - $\boldsymbol{I}$ = input image
  - $\boldsymbol{K}$ = filter/kernel/mask
  - $(\boldsymbol{x}, \boldsymbol{y})$ = coordinates of the current output pixel
  - $(\boldsymbol{i}, \boldsymbol{j})$ = coordinates inside the filter
- Step-by-step convolution:
  1. Place the filter at the top-left corner of the image.
  2. Multiply each filter value by the corresponding image pixel.

Mathematics of Convolution

- The output image is computed as the convolution of input image $\boldsymbol{I}$ with filter $\boldsymbol{K}$:

$$\boldsymbol{(I * K)(x, y) = \sum_{i} \sum_{j} I(x + i, y + j)\, K(i, j)}$$

- Explanation of terms:
  - $\boldsymbol{I}$ = input image
  - $\boldsymbol{K}$ = filter/kernel/mask
  - $\boldsymbol{(x, y)}$ = coordinates of the current output pixel
  - $\boldsymbol{(i, j)}$ = coordinates inside the filter
- Step-by-step convolution:
  1. Place the filter at the top-left corner of the image.
  2. Multiply each filter value by the corresponding image pixel.
  3. Sum the results to get a single output pixel.

Mathematics of Convolution

- The output image is computed as the convolution of input image $\boldsymbol{I}$ with filter $\boldsymbol{K}$:

$$(\boldsymbol{I} * \boldsymbol{K})(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i} \sum_{j} \boldsymbol{I}(\boldsymbol{x} + \boldsymbol{i}, \boldsymbol{y} + \boldsymbol{j})\, \boldsymbol{K}(\boldsymbol{i}, \boldsymbol{j})$$

- Explanation of terms:
  - $\boldsymbol{I}$ = input image
  - $\boldsymbol{K}$ = filter/kernel/mask
  - $(\boldsymbol{x}, \boldsymbol{y})$ = coordinates of the current output pixel
  - $(\boldsymbol{i}, \boldsymbol{j})$ = coordinates inside the filter
- Step-by-step convolution:
  1. Place the filter at the top-left corner of the image.
  2. Multiply each filter value by the corresponding image pixel.
  3. Sum the results to get a single output pixel.
  4. Move the filter and repeat for all pixels.

Example: 3×3 Mean Filter on Image Patch

- Image patch:

$$I = \begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{bmatrix}$$

Example: 3×3 Mean Filter on Image Patch

- Image patch:

$$I = \begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{bmatrix}$$

- Mean filter kernel:

$$K = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Example: 3×3 Mean Filter on Image Patch

- Image patch:

$$I = \begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{bmatrix}$$

- Mean filter kernel:

$$K = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Convolution for center pixel:

$$(I * K)(2,2) = \frac{1}{9}(10 + 20 + 30 + 40 + 50 + 60 + 70 + 80 + 90) = 50$$

10	20	30
40	50	60
70	80	90

Common Example Kernels and Summary

- Some widely used 3×3 filters:

Common Example Kernels and Summary

- Some widely used 3×3 filters:
  - **Edge Detection (Sobel X):** $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

Common Example Kernels and Summary

- Some widely used 3×3 filters:
  - **Edge Detection (Sobel X):** $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
  - **Edge Detection (Sobel Y):** $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

Common Example Kernels and Summary

- Some widely used 3×3 filters:
  - **Edge Detection (Sobel X):** $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
  - **Edge Detection (Sobel Y):** $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
  - **Sharpening:** $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

Common Example Kernels and Summary

- Some widely used 3×3 filters:
  - **Edge Detection (Sobel X):** $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
  - **Edge Detection (Sobel Y):** $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
  - **Sharpening:** $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
- Key Points:

Common Example Kernels and Summary

- Some widely used 3×3 filters:
  - **Edge Detection (Sobel X):** $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
  - **Edge Detection (Sobel Y):** $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
  - **Sharpening:** $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
- Key Points:
  - Filter = small matrix applied to image.

Common Example Kernels and Summary

- Some widely used 3×3 filters:
  - **Edge Detection (Sobel X):** $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
  - **Edge Detection (Sobel Y):** $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
  - **Sharpening:** $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
- Key Points:
  - Filter = small matrix applied to image.
  - Convolution = sliding filter and computing weighted sum.

Common Example Kernels and Summary

- Some widely used 3×3 filters:
  - **Edge Detection (Sobel X)**: $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
  - **Edge Detection (Sobel Y)**: $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
  - **Sharpening**: $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
- Key Points:
  - Filter = small matrix applied to image.
  - Convolution = sliding filter and computing weighted sum.
  - Filter effects depend on weights: smooth, sharpen, detect edges.

Common Example Kernels and Summary

- Some widely used 3×3 filters:
  - **Edge Detection (Sobel X):** $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
  - **Edge Detection (Sobel Y):** $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
  - **Sharpening:** $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
- Key Points:
  - Filter = small matrix applied to image.
  - Convolution = sliding filter and computing weighted sum.
  - Filter effects depend on weights: smooth, sharpen, detect edges.
  - Visual examples help understand the transformation.

Python Hands-on: Applying a 3×3 Kernel

- Let's apply a 3×3 mean filter to an image patch in Python.

Manual Convolution Example

```
import numpy as np

# Image patch
I = np.array([[10, 20, 30],
[40, 50, 60],
[70, 80, 90]])

# 3x3 mean filter
K = np.ones((3,3)) / 9

# Manual convolution (center pixel)
center_pixel = np.sum(I * K)
print("Center pixel value:", center_pixel)
```

Python Hands-on: Applying a 3×3 Kernel to a Real Image

- Now, let’s apply a 3×3 mean filter to a real image.

Using OpenCV

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load the image in grayscale
img = cv2.imread(’example.jpg’, cv2.IMREAD_GRAYSCALE)

# Define 3x3 mean filter kernel
K = np.ones((3,3), np.float32) / 9

# Apply filter using convolution
filtered = cv2.filter2D(img, -1, K)

# Display original and filtered images
plt.subplot(1,2,1), plt.imshow(img, cmap=’gray’), plt.title(’Original’)
plt.subplot(1,2,2), plt.imshow(filtered, cmap=’gray’), plt.title(’Filtered’)
plt.show()
```

Reach PostNetwork Academy

Website

www.postnetwork.co

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Facebook Page

www.facebook.com/postnetworkacademy

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Facebook Page

www.facebook.com/postnetworkacademy

LinkedIn Page

www.linkedin.com/company/postnetworkacademy

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Facebook Page

www.facebook.com/postnetworkacademy

LinkedIn Page

www.linkedin.com/company/postnetworkacademy

GitHub Repositories

www.github.com/postnetworkacademy

Thank You!