Image Convolution Lecture

November 14, 2025

Contents

1 Introduction to Convolution

Key Concept 1.1 (Convolution)**:** *Convolution is a fundamental operation in image processing. It is a* ***linear operator****, defined as a weighted sum of neighbors, which is applied identically to all pixels in an image.*

1.1 Discrete Convolution Formulas

1.1.1 1D Discrete Convolution

For two 1D arrays, f and g, their convolution $h = f * g$ is:

$$h(x) = (f * g)(x) = \sum_a f(a)g(x-a) \tag{1}$$

The length of the resulting convolution of two sequences of length N and M is $N + M - 1$. For example, $[1, 1] * [1, 1] = [1, 2, 1]$.

1.1.2 2D Discrete Convolution

For an image g and a kernel f, the convolved image h is:

$$h(x, y) = \sum_k \sum_l f(k, l)g(x-k, y-l) \tag{2}$$

This is the core formula for image filtering.

1.2 Continuous Convolution

The physical world is continuous. The response of sensors to color or light over an area is a continuous phenomenon. The continuous convolution formula uses an integral instead of a sum:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(a)g(x-a)da \tag{3}$$

For example, the convolution of two 1D "box" functions results in a "triangle" function.

2 The Kernel

The **kernel** (or filter) is a small matrix of weights $f(k, l)$ that determines the effect of the convolution.

- **Identity (Do Nothing) Kernel:** A kernel with a 1 at the center and 0s elsewhere. It returns the original image.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- **Shift Kernel:** Moving the '1' in the kernel shifts the image. For example, a '1' at $f(-1, 0)$ (left of center) results in $h(x, y) = g(x + 1, y)$, which shifts the image to the left.
- **Blur (Mean Filtering) Kernel:** A kernel of uniform values (e.g., all $\frac{1}{9}$) averages a pixel with its 3×3 neighborhood, resulting in a blur.

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

  Other blur kernels include weighted averages, like a Gaussian approximation:

$$f = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

3 Boundary Handling

When the kernel is at the edge of an image, it "hangs off". We must define how to handle these cases.

- **Zero Padding:** Assume all pixels outside the image are 0.
- **Reflection:** Reflect the image content at the boundary.
- **Cyclic (Wrap-around):** The image wraps around from one side to the other. This is used in Fourier analysis.

4 Convolution Properties

4.1 Properties

Convolution is a linear operation and has several key properties:

- **Commutative:** $f * g = g * f$
- **Associative:** $f * (g * h) = (f * g) * h$
- **Distributive:** $f * (g + h) = f * g + f * h$

Note: *The "flip" in the $g(x - a)$ term is what makes convolution commutative.*

4.2 Convolution as Matrix Multiplication

A linear operator can be expressed as a matrix multiplication. A 1D cyclic convolution can be represented by multiplying the signal vector by a **Circulant Matrix**. This matrix is built from the kernel, with the values "wrapping around" the edges. Different matrices can be constructed for zero-padding or reflection boundaries.

5 The Convolution Theorem (Fourier Domain)

The Fourier Transform (denoted Φ) converts a signal from the spatial domain $f(x, y)$ to the frequency domain $F(u, v)$.

Key Concept 5.1 (The Convolution Theorem)**:** *Convolution in the spatial domain is equivalent to pointwise multiplication in the frequency domain.*

$$\Phi(f * g) = F \cdot G \tag{4}$$

Conversely, multiplication in the spatial domain is convolution in the frequency domain:

$$\Phi(f \cdot g) = F * G \tag{5}$$

5.1 Application: Fast Convolution

This theorem provides a massive speedup for convolution.

- Standard convolution complexity: $O(N^2)$
- Using Fast Fourier Transform (FFT): $O(N \log N)$

The fast method is: $f * g = \Phi^{-1}(\Phi(f) \cdot \Phi(g))$

5.2 Fourier Domain Pairs

- **Image:** Convolve f by a **Box** $\Leftrightarrow$ **Fourier:** Multiply F by a **Sinc** function.
- **Image:** Convolve f by a **Triangle** ($[1, 2, 1]$) $\Leftrightarrow$ **Fourier:** Multiply F by a **Sinc**^{2} function.
- **Image:** Convolve f by a **Gaussian** $\Leftrightarrow$ **Fourier:** Multiply F by a **Gaussian**.

6 Applications of Convolution

6.1 Smoothing and Noise Cleaning

Smoothing kernels (where weights sum to 1) are used to remove noise by averaging.

- **Averaging/Blurring:** Effective for additive, zero-mean noise. The process can cause a loss of detail.
- **Median Filtering:** A non-linear operation. It replaces a pixel with the median of its neighborhood. It is highly effective against "salt & pepper" noise (outliers) and is robust, preserving edges better than mean filtering.

6.2 Edge Detection (Derivatives)

Edges are large differences between neighboring pixels. This change is measured with a derivative.

6.2.1 First Derivative (The Gradient)

The derivative is approximated by convolving with a "difference" kernel.

- **Simple difference:** $\frac{\partial f}{\partial x} \approx f(i,j) - f(i-1,j) \rightarrow$ Kernel: $(1 \ -1)$
- **Centered difference:** $\frac{\partial f}{\partial x} \approx \frac{f(i+1,j)-f(i-1,j)}{2} \rightarrow$ Kernel: $\frac{1}{2}(1 \ 0 \ -1)$

The **Gradient**, ∇f, is the vector of partial derivatives:

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \tag{6}$$

The **Gradient Magnitude** measures edge strength:

$$|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2} \tag{7}$$

Sobel Filters are popular derivative kernels that combine a difference in one direction with a blur in the orthogonal direction to reduce noise.

$$\frac{\partial f}{\partial x} = \frac{1}{8} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \frac{\partial f}{\partial y} = \frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

6.2.2 Second Derivative (The Laplacian)

The Laplacian is the sum of the second partial derivatives:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{8}$$

This is approximated by the sum of the 1D second derivative kernels, $(1 \ -2 \ 1)$ and its transpose:

$$\text{Kernel}(\nabla^2 f) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Intuitively, this kernel measures the difference between a pixel and its 4-neighbors. An alternative kernel uses all 8 neighbors:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Note: *Edge Localization (Zero-Crossing): The exact location of an edge is at the maximum of the first derivative (f'), which corresponds to the* ***zero-crossing*** *of the second derivative (f''). Derivatives amplify noise, so the image is typically smoothed (blurred) with a Gaussian first.*

6.3 Image Sharpening

Sharpening is achieved by subtracting the Laplacian (which is non-zero only at edges) from the original image. This exaggerates the edges.

$$f_{\text{sharpened}} = f - \alpha \cdot (\nabla^2 f)$$

This can be combined into a single kernel. For $\alpha = a$:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - a \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -a & 0 \\ -a & 1+4a & -a \\ 0 & -a & 0 \end{pmatrix}$$

7 Convolutional Neural Networks (CNNs)

In traditional image processing, kernels are "hand-crafted" (e.g., Sobel). In CNNs, the weights of the convolution kernels (e.g., 3×3 or 5×5) are **learned** from data to give the best performance on a task. CNNs consist of multiple layers, including convolution, pooling (sampling), and activation functions (e.g., ReLU).