Computer Vision

CVI620

Session 8
01/2025

Review

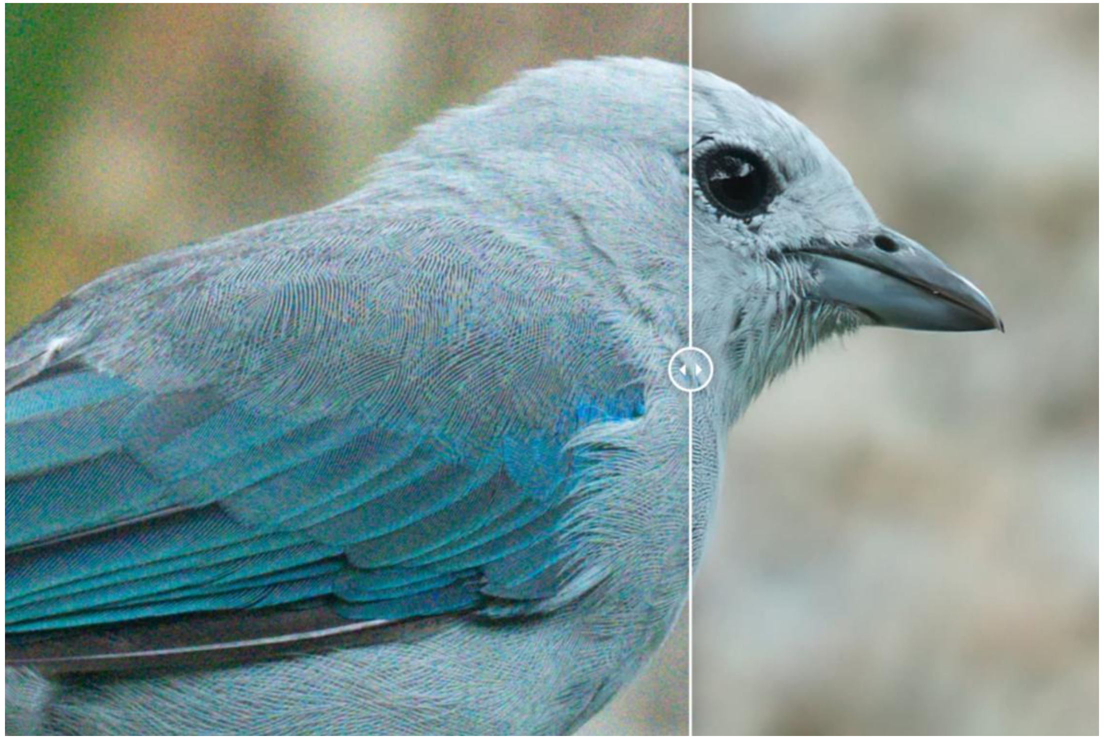
Introduction to Computer Vision and Imaging Systems Cameras
System Configurations Digital Cameras and Images Color Standards Introduction to OpenCV
Image Formats Image Compression OpenCV methods and operations PEP8 standard
Basic Image Arithmetic Pixel Transforms Histograms
Geometric Transformations Image Noise Linear vs. Nonlinear Filtering

Agenda

- Denoising Categories
- Filter
- Convolution
- Mean Denoise

Denoising

- https://www.topazlabs.com/denoise-ai?srsltid=AfmBOopM02_xy6QWHcr_WMSI5iSE0MH4MBimXXkvw7fg80zWAmOHNnUs



Denoising Techniques

Spatial Filtering

Frequency Domain Filtering

Advanced Methods

Another Categorization

1. Filtering Based Denoising
2. Transform Based Denoising
3. Statistical & Probabilistic Methods
4. Machine Learning & Deep Learning Approaches
5. Optimization-Based Methods

Denoising Techniques

Spatial Filtering

Frequency Domain Filtering

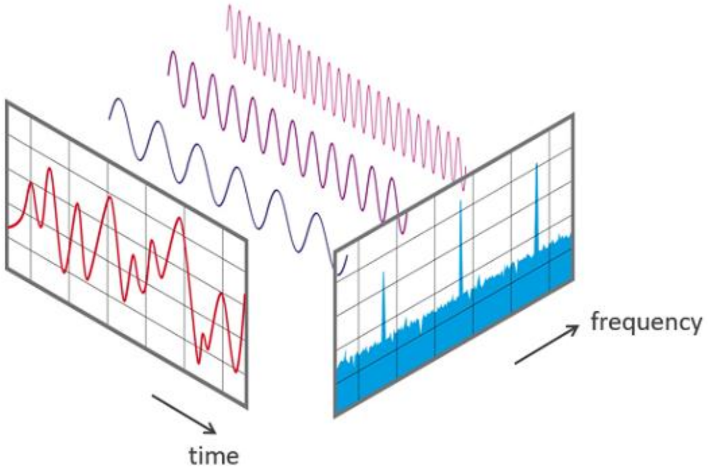
Advanced Methods

Spatial Filtering

- Mean Filter: Averages pixel values in a neighborhood
- Median Filter: Replaces pixel value with the median in a local window
- Gaussian Filter: Weighted averaging with a Gaussian kernel

Frequency Domain Filtering

- Fourier Transform-Based Filtering: Suppresses high-frequency noise



Advanced Methods

01

Wavelet Transform Denoising: Removes noise at different scales

02

Non-Local Means (NLM): Uses similar patches across the image

03

Deep Learning-Based Denoising (Denoising Autoencoders, CNNs)

Another Categorization

1. Filtering Based Denoising
2. Transform Based Denoising
3. Statistical & Probabilistic Methods
4. Machine Learning & Deep Learning Approaches
5. Optimization-Based Methods

Filtering Based Denoising

- Mean Filter
- Gaussian Filter
- Median Filter
- Bilateral Filter
- Wavelet Denoising

Transformer Based Denoising

Fourier Transform Filtering: Removes noise in the frequency domain.

Wavelet Transform: Decomposes the image into different scales and removes noise selectively.

Statistical Denoising

Non-Local Means (NLM): Uses patches from the image to reduce noise.

Bayesian Denoising: Uses probabilistic models to infer the denoised image.

ML and DL Denoising

- Denoising Autoencoders: Neural networks trained to remove noise from images.
- Convolutional Neural Networks (CNNs): Trained models that learn patterns to reconstruct noise-free images.
- Generative Models (GANs, Diffusion Models): Learn distributions of clean images to remove noise effectively.

Optimization Based Denoising

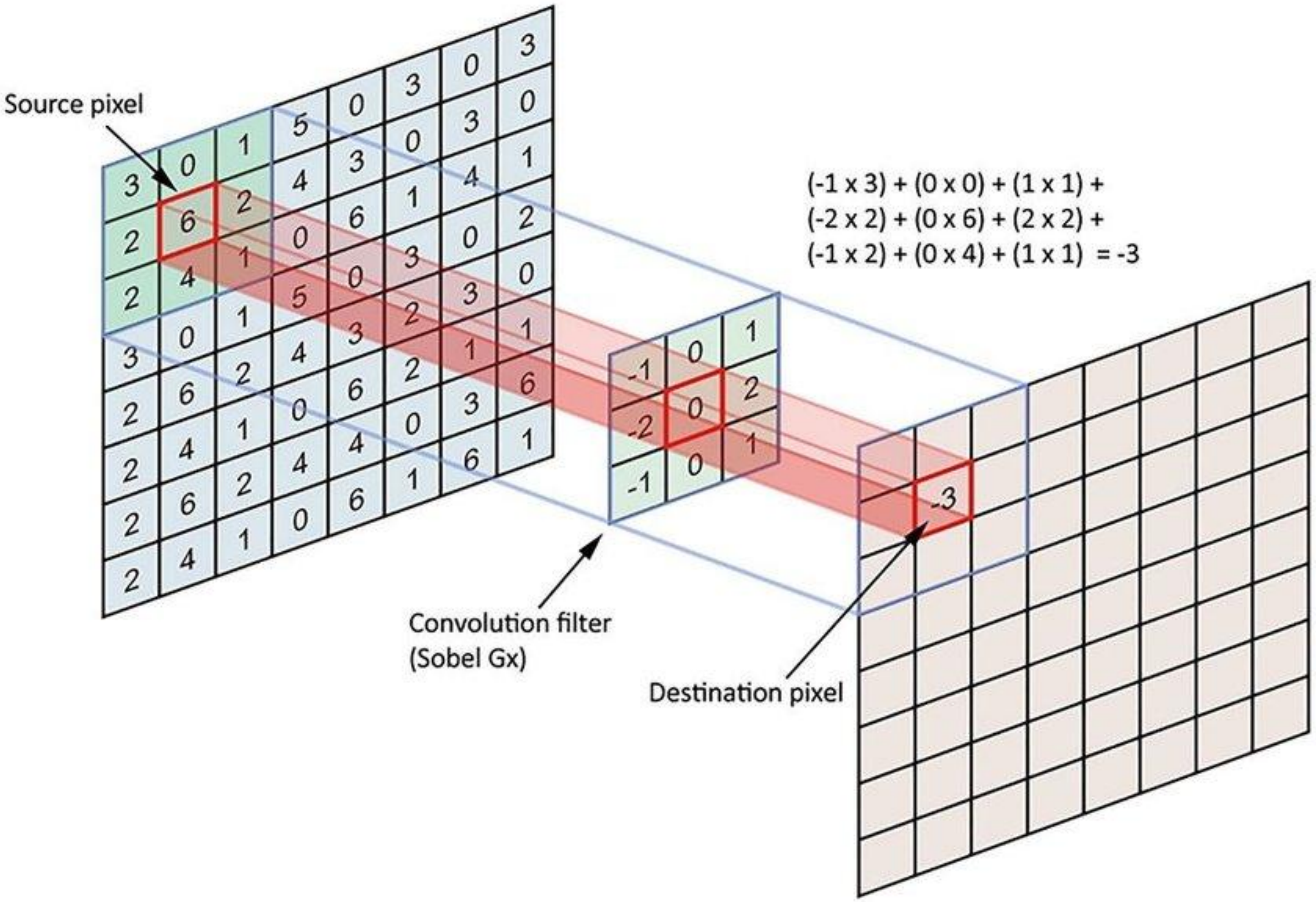
- Total Variation (TV) Denoising: Minimizes variations while preserving edges.
- Sparse Coding: Represents images in a sparse domain and removes noise adaptively.

noise -> filter -> convolution -> denoise

Image Filtering

Filter

- The core idea is to manipulate image features like noise, edges, and textures based on specific objectives.
- Adding, changing, detecting features or filters in a picture is better to be applied step by step and region by region

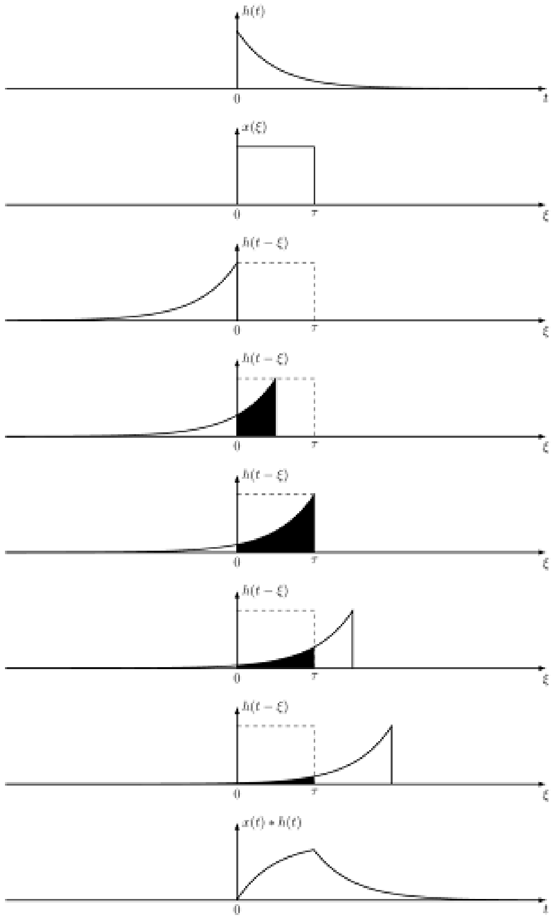


Convolution

Filtering is commonly implemented using **convolution**, especially in spatial domain filtering.

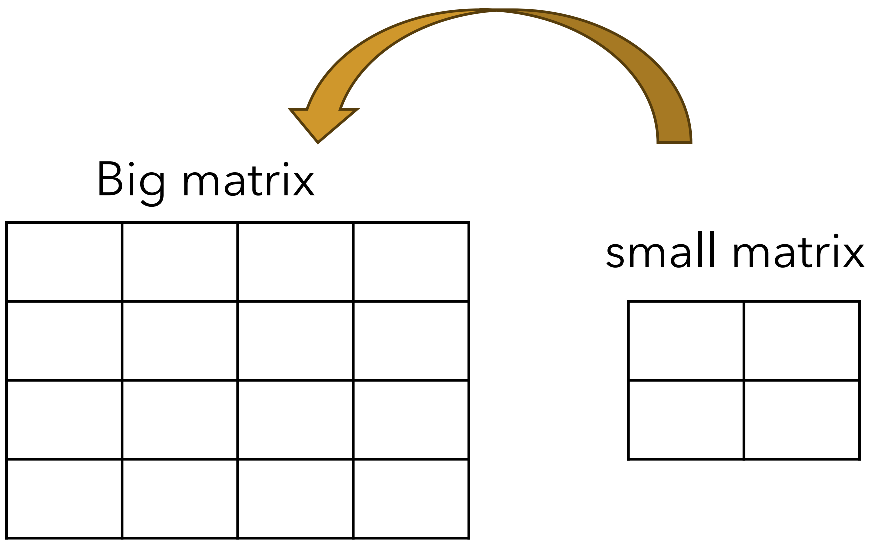
In frequency domain filtering, convolution is performed using Fourier Transform methods.

Convolution in Signal and Systems



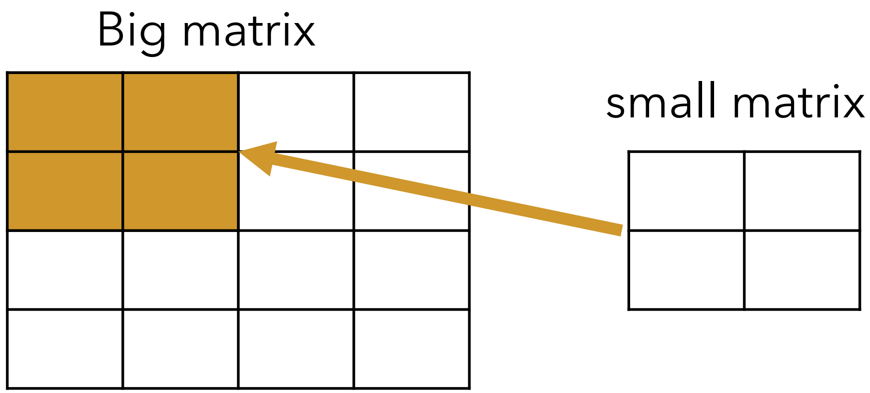
Convolution

- Sum of pixel multiplications in a region



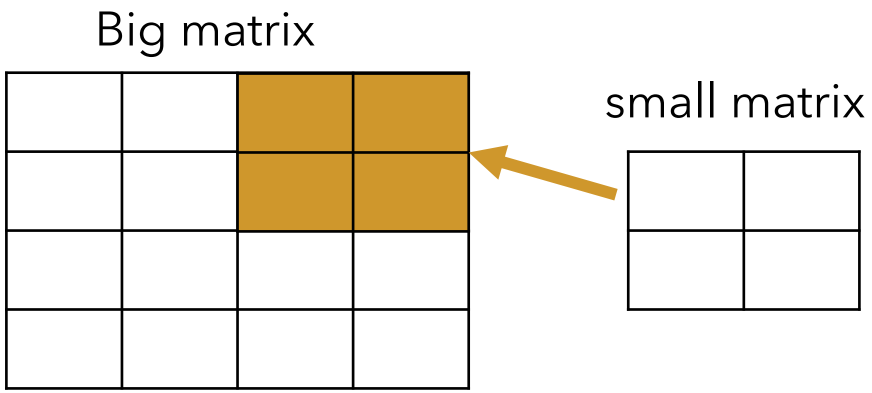
Convolution

- Align from (0,0)



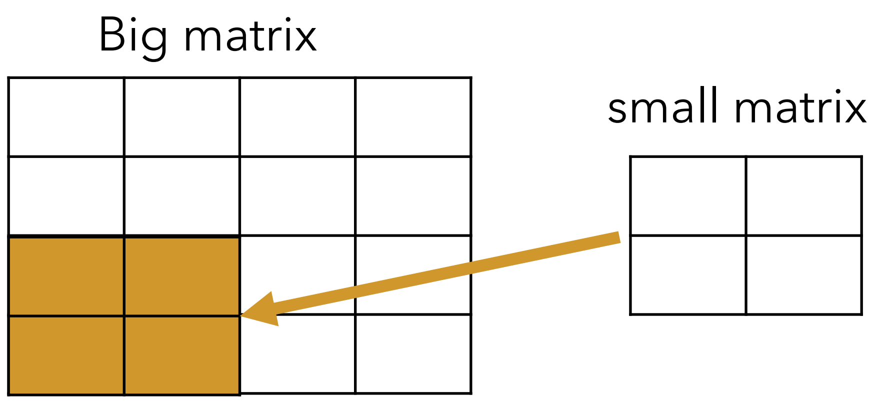
Convolution

- Move left to right with a step size

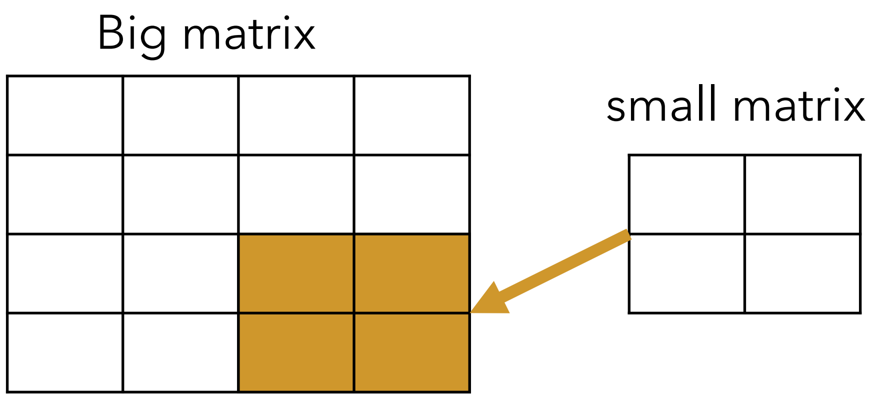


Convolution

- Move top to bottom with a step size

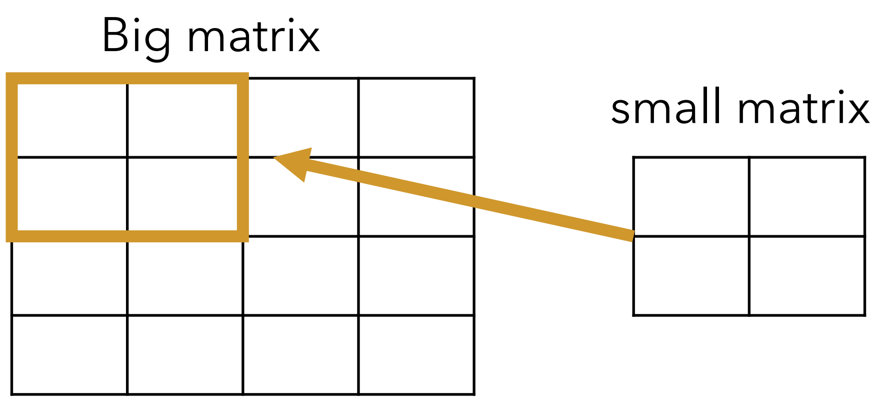


Convolution



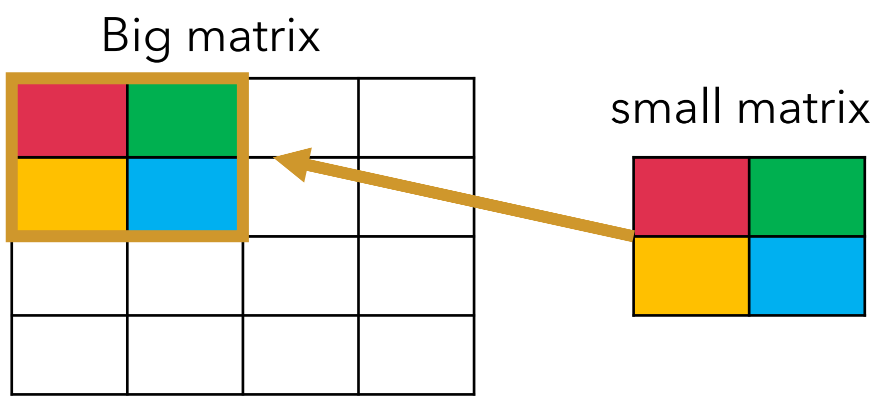
Convolution

- Mathematical operation



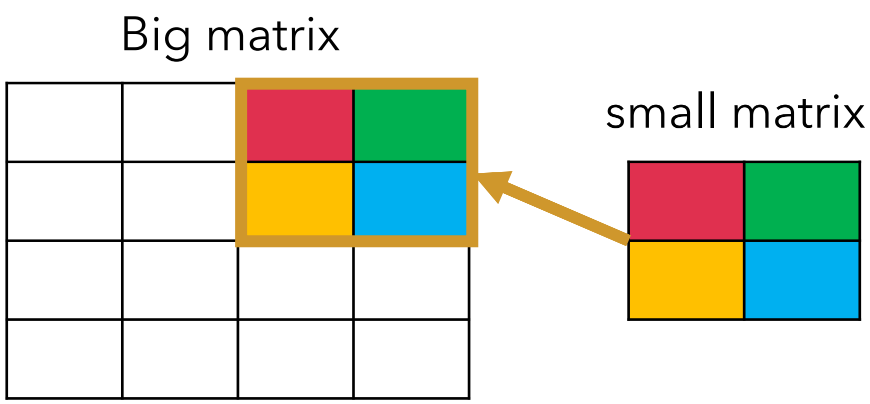
Convolution

- Sum of pixel multiplications in a region
- Multiply aligned pixels and add them together



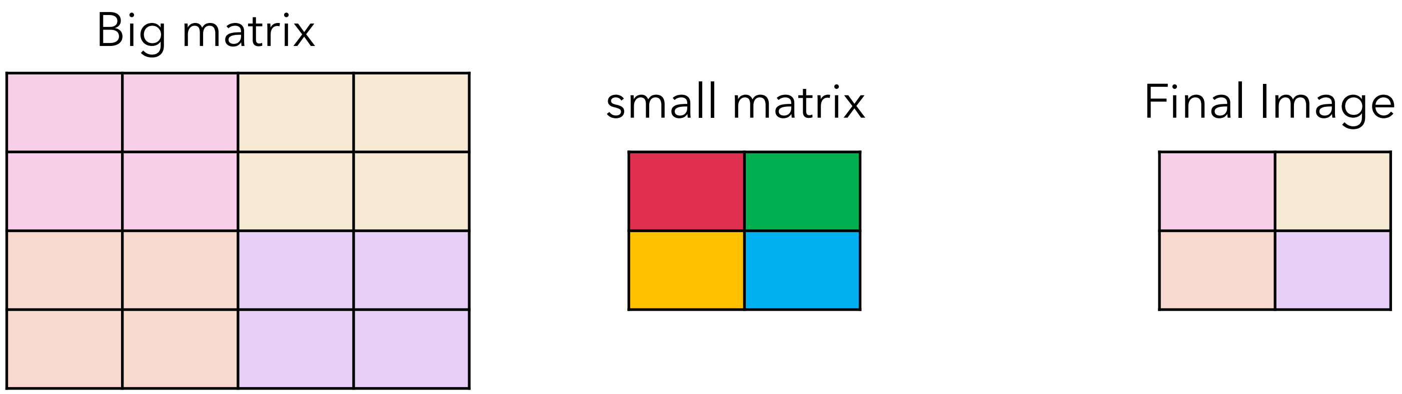
Convolution

- Sum of pixel multiplications in a region
- Multiply aligned pixels and add them together



Convolution

- Sum of pixel multiplications in a region
- Multiply aligned pixels and add them together
- **Make a new image**



Example

Image

0	1	2
3	4	5
6	7	8

Kernel

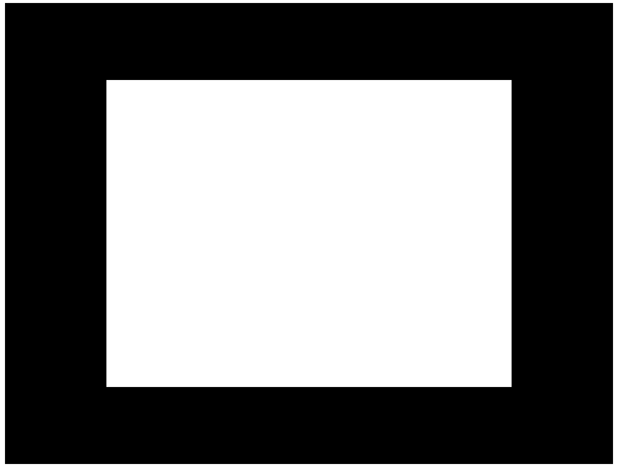
0	1
2	3

19	25
37	43

0*0 + 1*1 + 3*2 + 3*4 = 19
1*0 + 2*1 + 4*2 + 5*3 = 25
...

Real Example

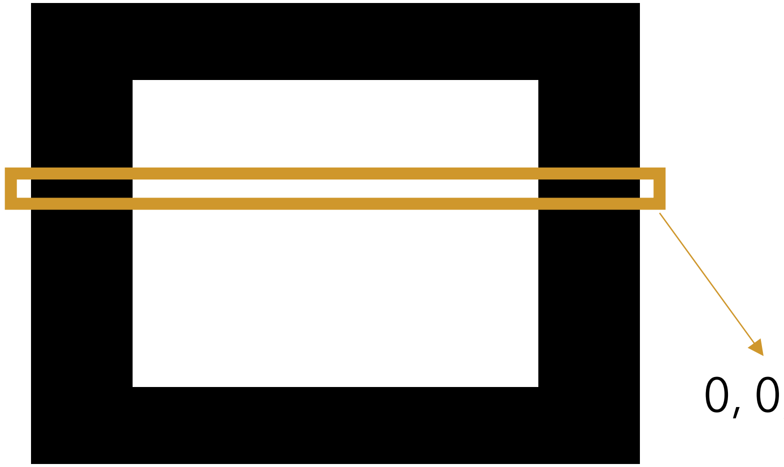
Image



Kernel
[-1,1]

Real Example

Image

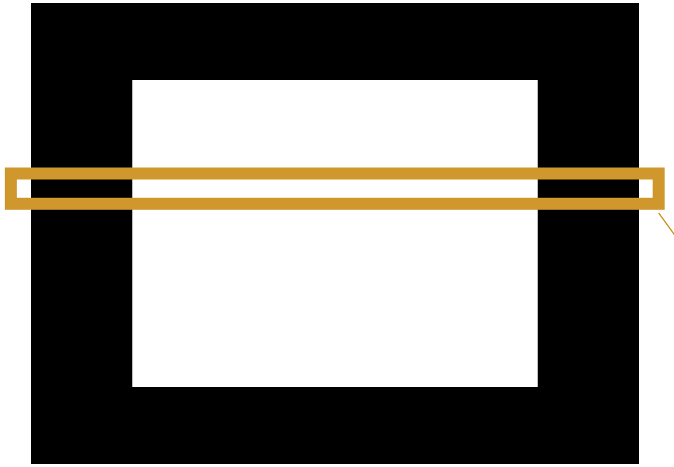


Kernel
[-1,1]

0, 0, 0, …., 255, 255, 255, …., 0, 0, …

Real Example

Image



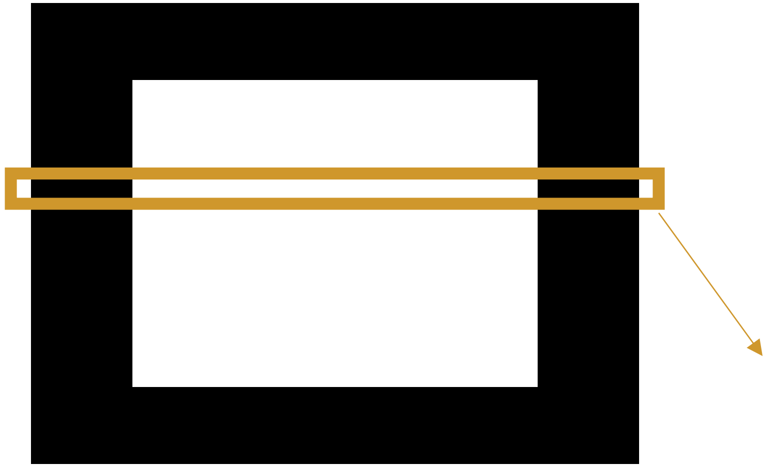
Kernel
[-1,1]

0, 0, 0, ….,0, 255, 255, 255, …., 0, 0, …

0*-1 + 0*1 = 0
0*1 + 255*1 = 255
255*-1 + 255*1 = 0
255*-1 + 255*1 = 0
255*-1 + 255*1 = 0
255*-1 + 0*-1 = -255 → clip(-255) =0

Real Example

Image



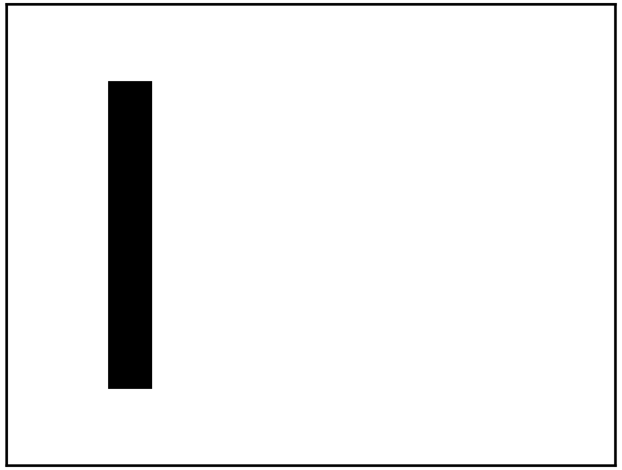
Kernel
[-1,1]

0, 0, 0, ….,0, 255, 255, 255, …., 0, 0, …

0, 0, 0, …, 0, 255, 0, 0, 0, …

Real Example

0, 0, 0, …, 0, 255, 0, 0, 0, …



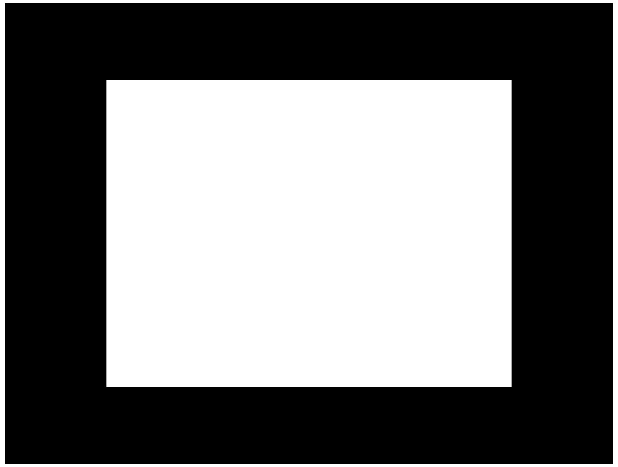
Edge!!

Let’s Code

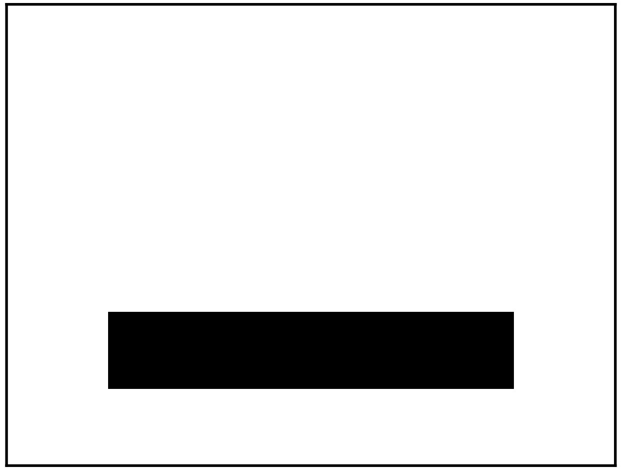
```
import cv2
import numpy as np


img = cv2.imread("square.png")
kernel = np.array([[-1, 1]])

out_image = cv2.filter2D(img, cv2.CV_8U, kernel)

cv2.imshow("left edge", out_image)
cv2.waitKey(0)
```

Let’s Convolve Vertically

Image



Kernel
[[-1],
[1]]



Example

```
import cv2
import numpy as np


img = cv2.imread("Lenna.png")
kernel1 = np.array([[-1, 1]])
kernel2 = np.array([[-1],
                    [1]])

out_image1 = cv2.filter2D(img, cv2.CV_8U, kernel1)
out_image2 = cv2.filter2D(img, cv2.CV_8U, kernel2)

cv2.imshow("left edge", out_image1)
cv2.imshow("bottom edge", out_image2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

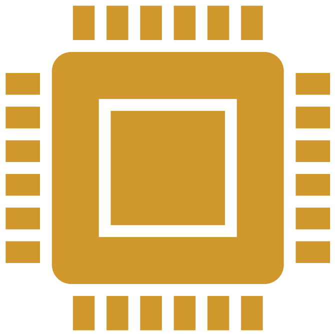
More Complex Convolutions!

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

How to achieve these kernels?

Experiments!



ML to learn kernels!

Now let's get back to denoising!

Mean Denoising

- Replaces each pixel value with the average of its neighboring pixels.
- Smooths the image by averaging pixel intensities.
- Reduces random noise while preserving structure.

Mean Denoising

- The filter slides over the image, replacing each pixel with the mean of its neighbors.
- Uses a convolution operation with a kernel

Mean Denoising

- Applying a 3×3 Mean Filter
- Kernel Example:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} * 1/9$$

- Each pixel value is replaced by the average of surrounding pixels.
- Result: A smoother image with reduced noise.
- Also called blurring!

Example

```
import cv2
import numpy as np

image = cv2.imread("Lucy.jpg")

if image is None:
    print("ERROR! Image not available...!")

# Gaussian Noise
noise = np.random.normal(0, 25, image.shape).astype('float32')

noisy_image = image + noise
noisy_image = np.clip(noisy_image, 0, 255).astype('uint8')

kernel = np.ones((3,3), dtype= np.float32) / 9
denoised_image = cv2.filter2D(noisy_image, -1, kernel)

cv2.imshow("left edge", denoised_image)
cv2.imshow('frame', noisy_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```