



# DEEP LEARNING

---

## Image Processing

Ashish Pujari

# Image Processing

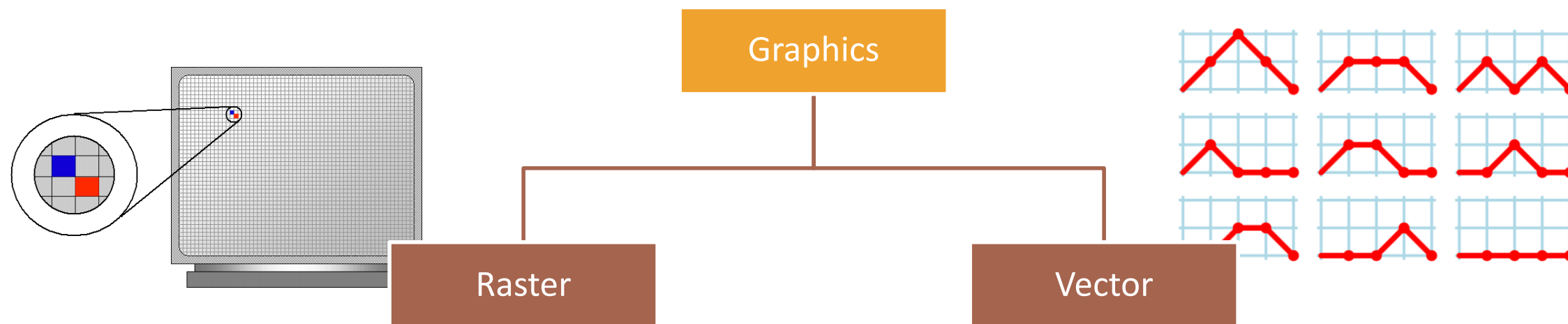
1. Digital Images
2. Image Processing
3. Convolution
4. Hands-on

# DIGITAL IMAGES

---

Applications, Tools, Image Types

# Computer Graphics

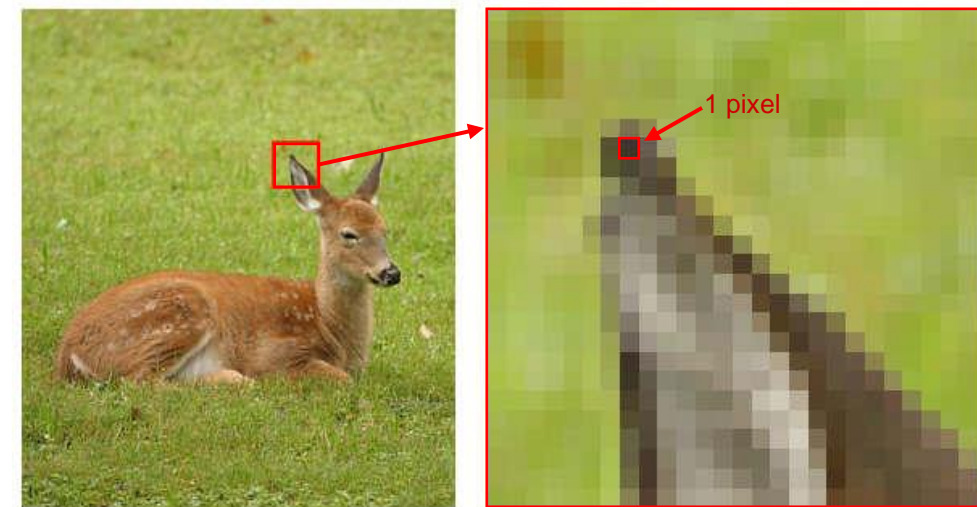


- Finite set of digital values (pixels)
- Fixed set of rows, columns of pixels
- Image and device dependent resolution
- Device captured photos, generated

- Generated using vectors (paths)
- Geometry and equations
- Resolution Independent
- Generated - fonts, icons, animation

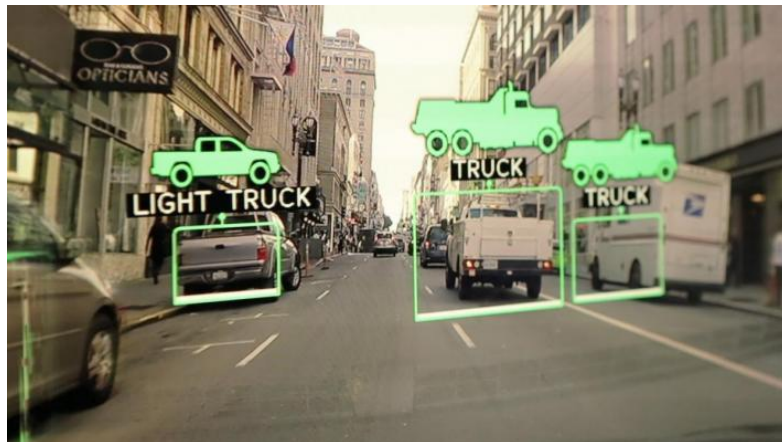
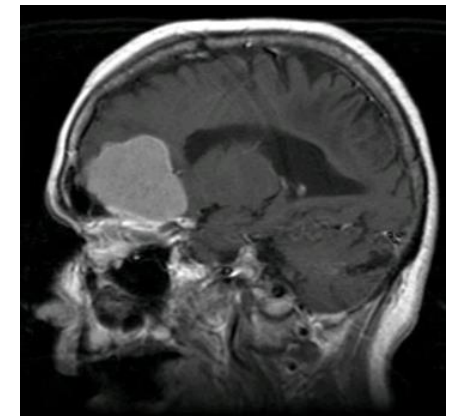
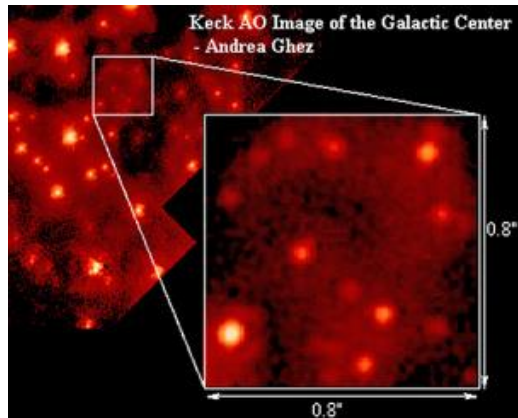
# Digital Images

- A digital image is a representation of a 2d image as a finite set of digital values, called picture elements or pixels
- An image is represented by a rectangular array of integers.
- Pixel integer values typically represent gray levels, colors, heights, opacities etc. of the image at that point
- N: # of rows, M: # of columns, Q: # of gray levels
  - $N = 2^n$  ,  $M = 2^m$  ,  $Q = 2^q$  (q is the # of bits/pixel)
  - Storage :  $N \times M \times Q$  (e.g.,  $N=M=1024$ ,  $q=8$ , 1MB)





# Image Processing Applications



# Image Processing Tools

- Adobe Photoshop
  - Professional and artist use
- Image Magick
  - Industry scale, bulk processing
- OpenCV
  - Advanced library, multi-language support
- Python
  - Numpy – numerical computing and matrices
  - Scipy – ndimage, signal processing, optimization
  - Matplotlib – basic graphing , plotting
  - Pillow – basic image processing
  - Skimage – image processing library
  - Python OpenCV APIs



# PROCESSING

---

Point and Area, Thresholding, Kernels, Pipelines



# Image Processing

- Improvement of pictorial information for human interpretation
- Processing of image data for storage, transmission and representation for autonomous machine perception
- Computer vision goes beyond image processing and involves Machine Learning, Robotics

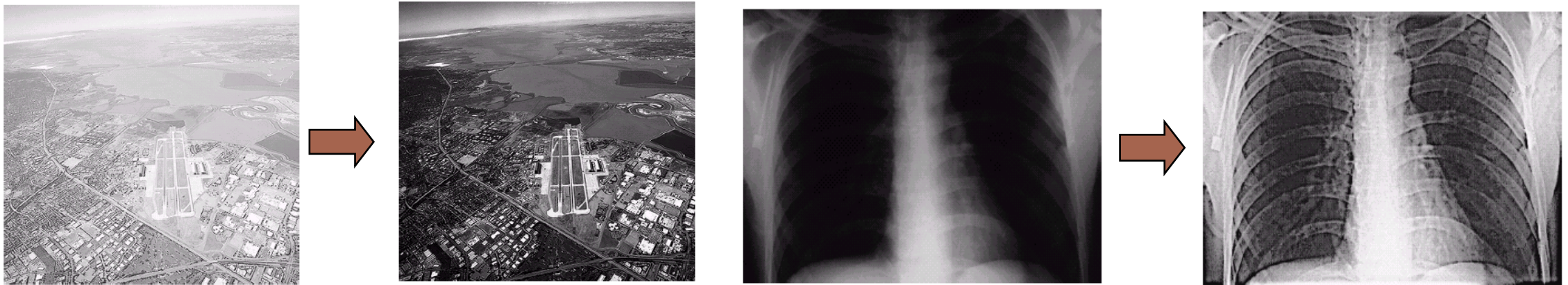


Image Enhancement

# Python - Skimage

- Builds on Scipy and Numpy - images are Numpy arrays containing integer or float with varying degrees of precision

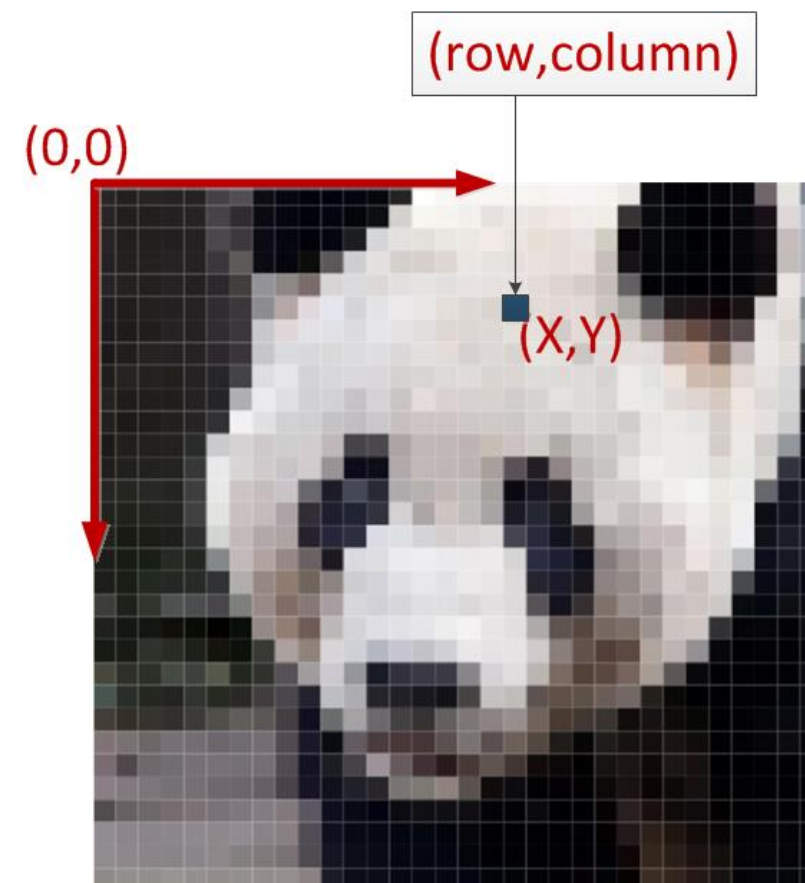
Image type	coordinates
2D grayscale	(row, col)
2D multichannel (eg. RGB)	(row, col, ch)
3D grayscale	(pln, row, col)
3D multichannel	(pln, row, col, ch)

Data type	Range
uint8	0 to 255
uint16	0 to 65535
uint32	0 to $2^{32}$
float	-1 to 1 or 0 to 1
int8	-128 to 127
int16	-32768 to 32767
int32	$-2^{31}$ to $2^{31} - 1$



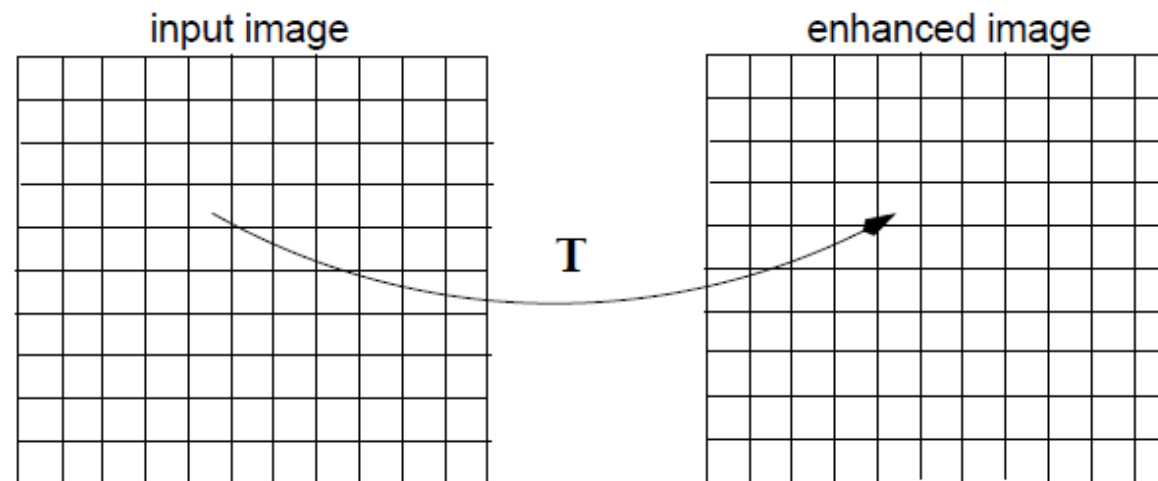
# NumPy Coordinate System

- In NumPy indexing, the first dimension corresponds to rows, while the second corresponds to columns.
- As per matrix notation origin  $(0, 0)$  on the top-left corner
- In contrast to Cartesian  $(x, y)$  coordinates where origin  $(0,0)$  is the bottom left corner.



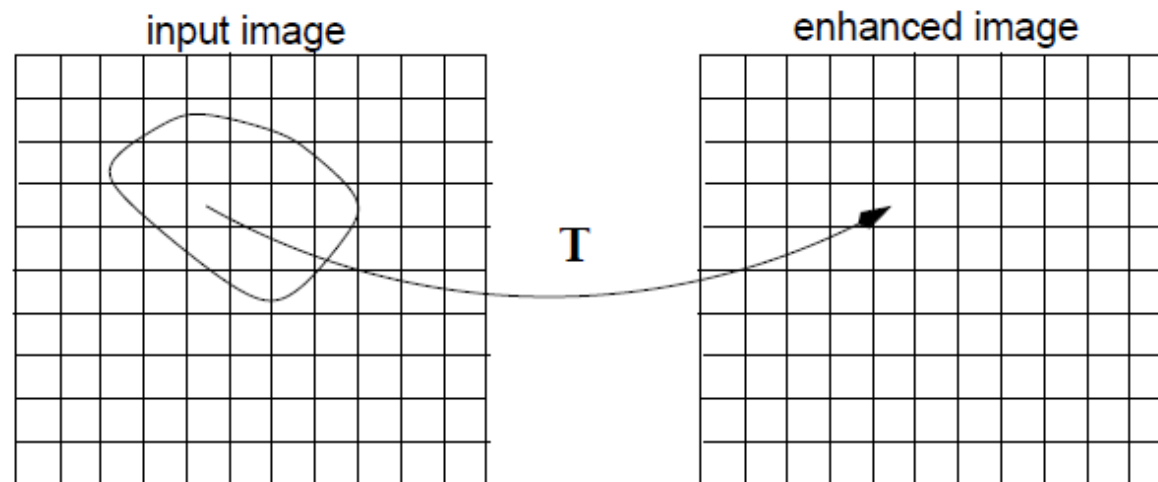
# Point Processing Methods

- Convert a given pixel value to a new pixel value based on some predefined function.
- Thresholding , Negative, Rotation, Histogram equalization, Piecewise linear transform



# Area Processing Methods

- Convert a neighborhood of pixel values to new values based on some predefined function
- Filtering – Mean, Median, Sharpening, Convolution



# Image Quantization

256 gray levels (8bits/pixel)



32 gray levels (5 bits/pixel)



<https://en.wikipedia.org/wiki/Lenna>

8 gray levels (3 bits/pixel)



4 gray levels (2 bits/pixel)



16 gray levels (4 bits/pixel)



2 gray levels (1 bit/pixel)



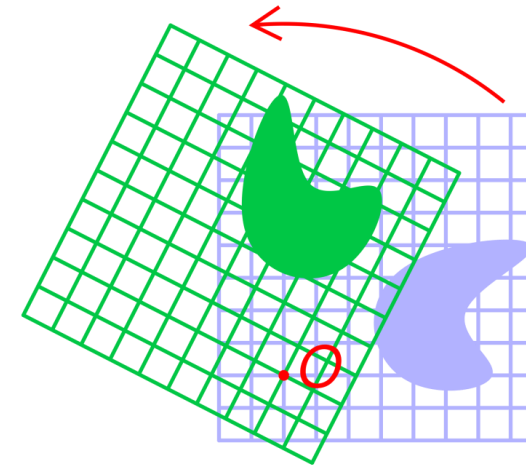
# Transformations



Scaling



Swirl



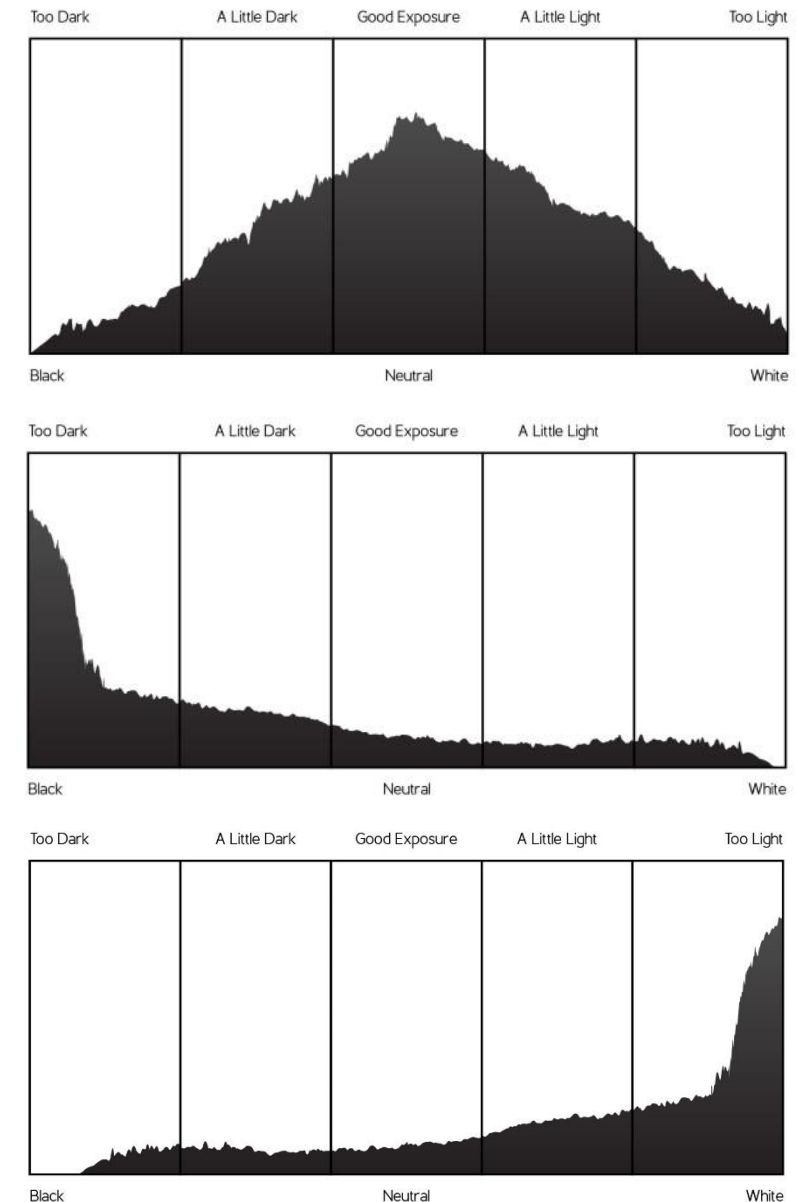
Rotation

<code>skimage.transform.rescale</code> (image, scale[, ...])	Scale image by a certain factor.
<code>skimage.transform.resize</code> (image, output_shape)	Resize image to match a certain size.
<code>skimage.transform.rotate</code> (image, angle[, ...])	Rotate image by a certain angle around its center.
<code>skimage.transform.seam_carve</code> (img, ...[, ...])	Carve vertical or horizontal seams off an image.
<code>skimage.transform.swirl</code> (image[, center, ...])	Perform a swirl transformation.



# Histogram Equalization

- The x-axis of the histogram is the range of the available digital numbers, i.e. 0 to 255. The y-axis is the number of pixels in the image having a given digital number.
- Increases dynamic range of an image
- Enhances contrast of image to cover all possible grey levels
- Ideal histogram = flat
  - same no. of pixels at each grey level
- Ideal no. of pixels at each grey level =  $i = \frac{N*M}{L}$



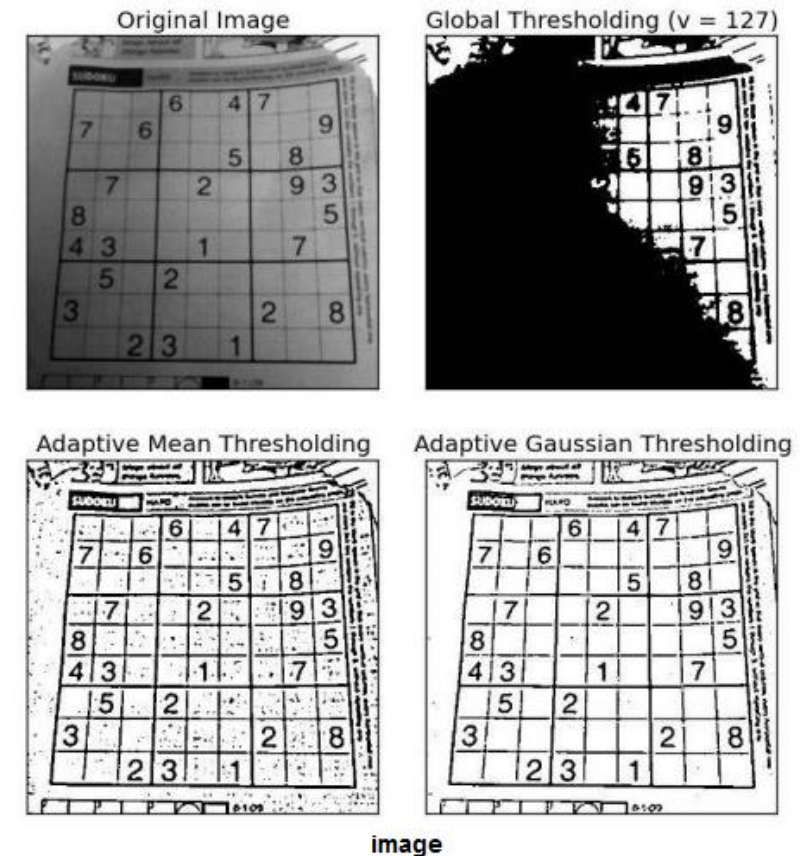
# Thresholding

- Thresholding is used to separate desirable foreground image objects from the background based on the difference in pixel intensities of each region
- Global Thresholding replaces each pixel in an image with a black pixel if the image intensity is less than a threshold



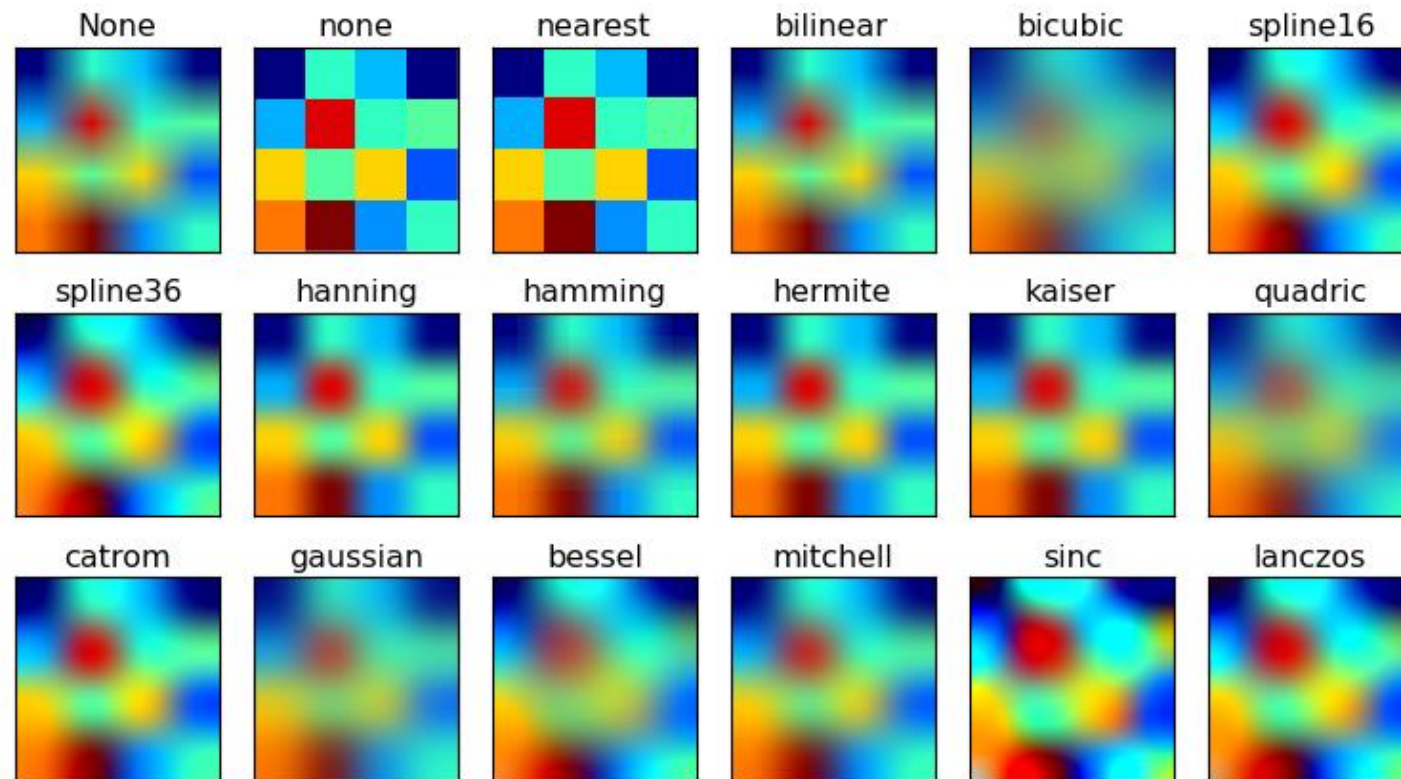
# Adaptive Thresholding

- Statistically examine the intensity values of the local neighborhood of each pixel.
- The statistic which is most appropriate depends largely on the input image – mean, median, min-max average
- Algorithm calculates the threshold for a small regions of the image - different thresholds for different regions gives better results with varying illumination



# Interpolation

Constructing new data points within the range of a discrete set of known data points



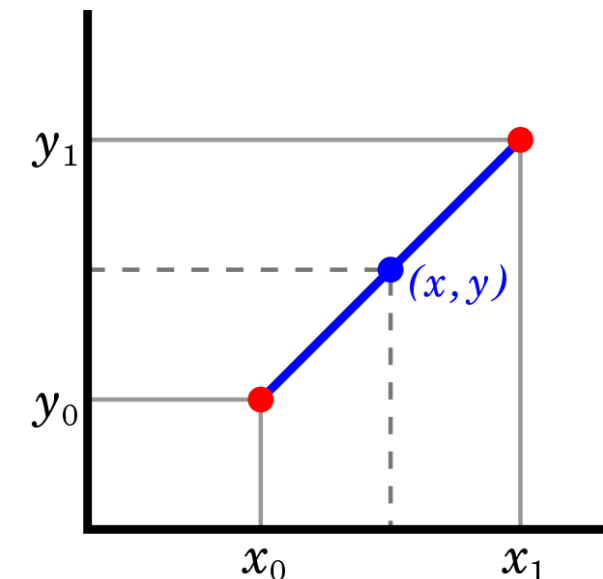
# Linear Interpolation

- Method of curve fitting using linear polynomials to construct new data points within the range of a discrete set of known data points.
- This formula can also be understood as a weighted average. The weights are inversely related to the distance from the end points to the unknown point; the closer point has more influence than the farther point.

If the two known points are given by the coordinates  $(x_0, y_0)$  and  $(x_1, y_1)$  the **linear interpolant** is the straight line between these points.

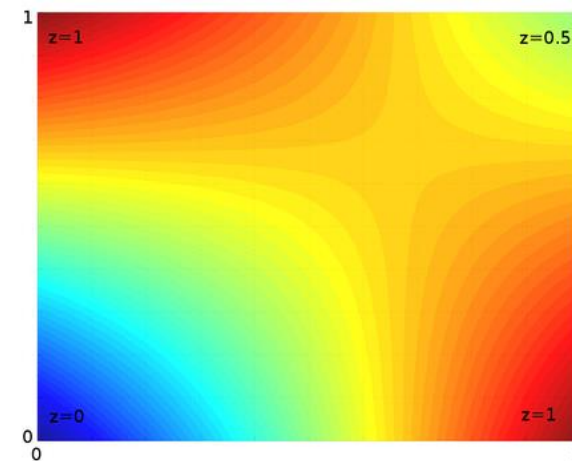
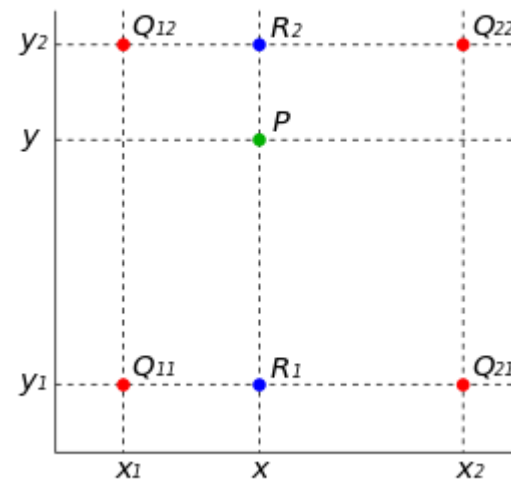
For a value  $x$  in the interval  $(x_0, x_1)$ , the value  $y$  along the straight line is given from the equation of slopes

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0},$$



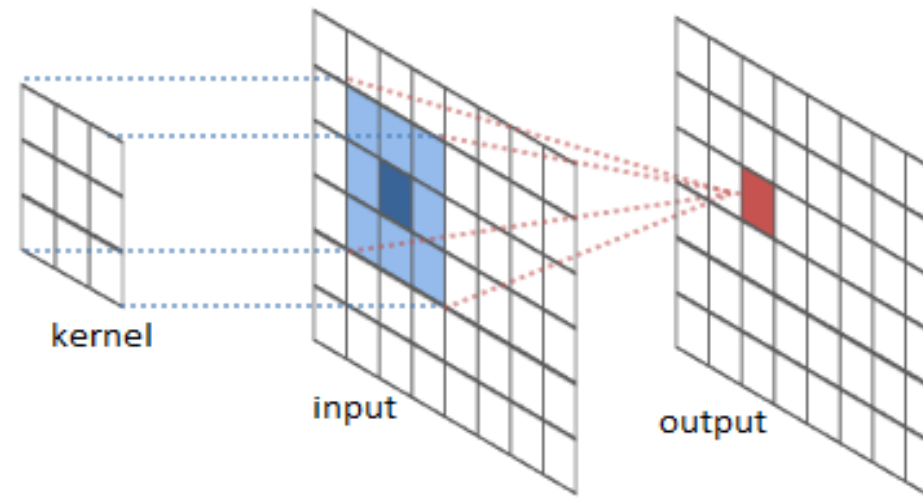
# Bilinear Interpolation

- Extension of linear interpolation for interpolating functions of two variables (e.g.,  $x$  and  $y$ ) on a rectilinear 2D grid
- Perform linear interpolation first in one direction, and then again in the other direction. Although each step is linear in the sampled values and in the position, the interpolation as a whole is not linear but rather quadratic in the sample location.



# Filtering

- Processing an image (applying effects) to produce another image
- Filtering is a *neighborhood operation* - value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel
- Effects - Blur/Smooth, Edge Detection, Sharpen, Emboss





# Kernel

- A small matrix of numbers used to apply effects such as blurring, sharpening, outlining, smoothing, de-noising
- Used as input to a convolution operation over an image
- Also referred to as “mask” or “filter”

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

Identity Kernel

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpen Kernel

# Filtering – Blurring

- Image quality enhancement where sharp points get smoothed out.
- Using of averaging filters – mean, median and gaussian.



*The Original and Blurred Images*

$1/9 \times$

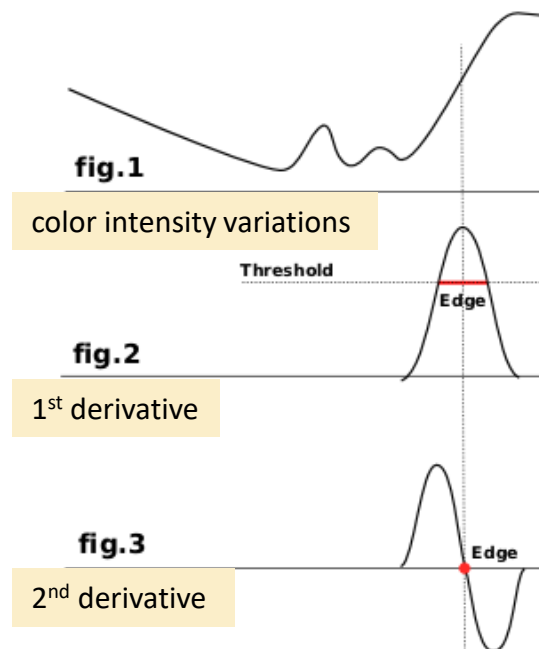
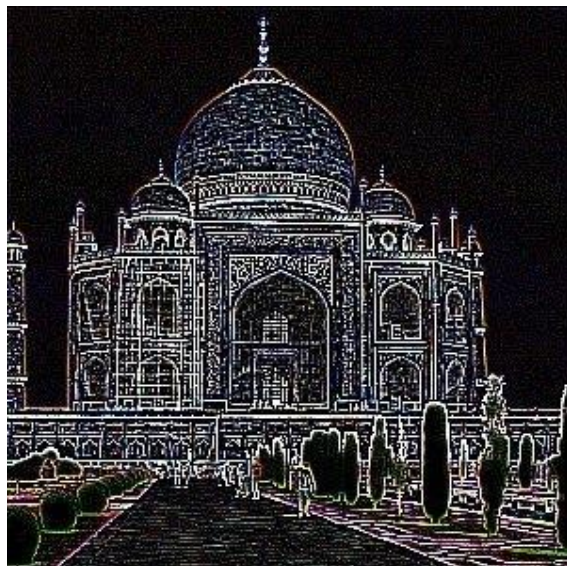
Standard Average		
1	1	1
1	1	1
1	1	1

$1/16 \times$

Weighted Average		
1	2	1
2	4	2
1	2	1

# Filtering – Edge Detection

- Edge detection filters search for borders between different colors and so can detect contours of objects
- The gradient is the derivative (change) of an image; large gradients indicate edges - a border is detected when gradient is more than an assumed threshold value



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

Gx

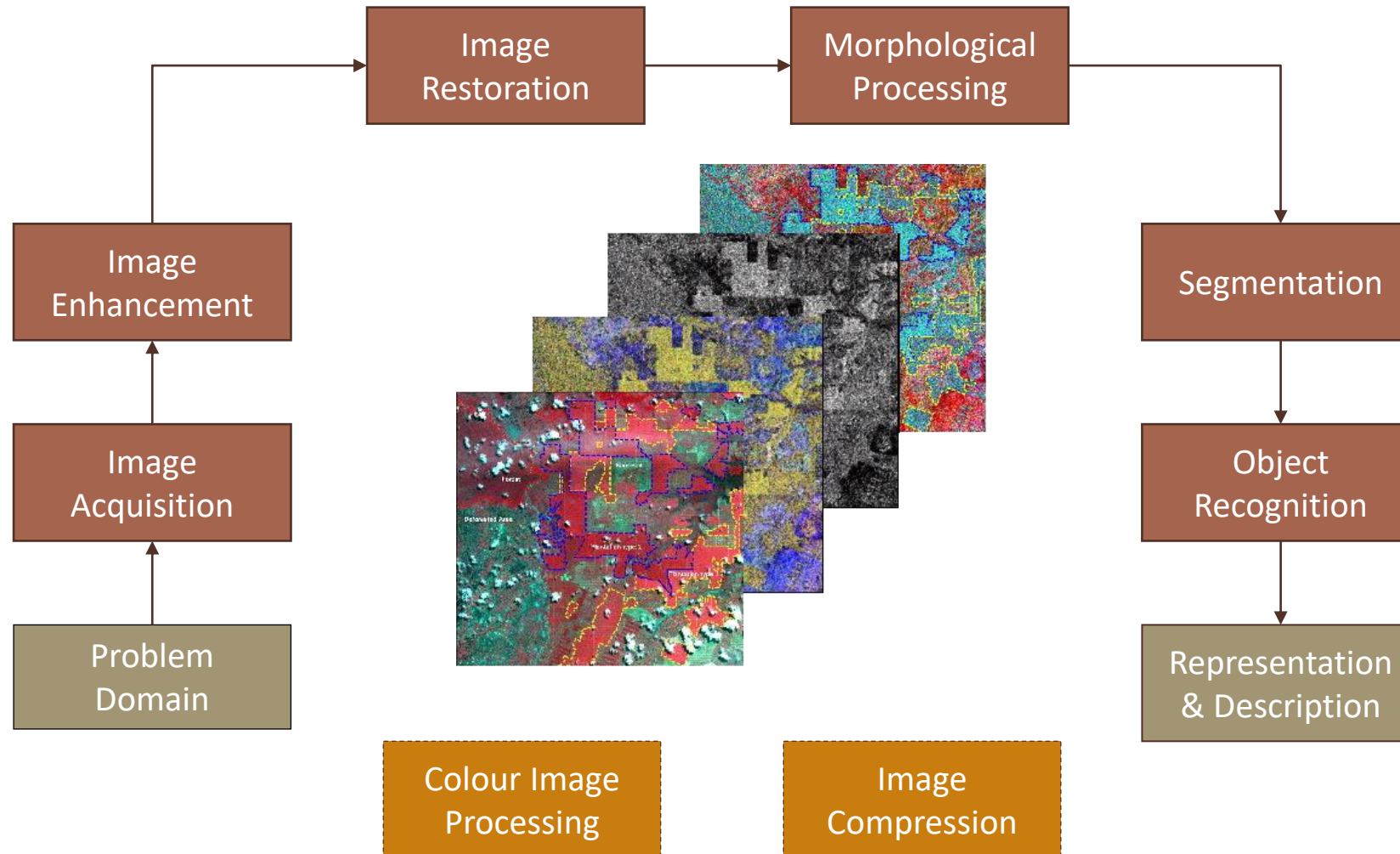
+1	+2	+1
0	0	0
-1	-2	-1

Gy

Sobel kernels show only the differences in adjacent pixel values in a particular direction

<https://docs.gimp.org/en/filters-edge.html>

# Image Processing Pipeline

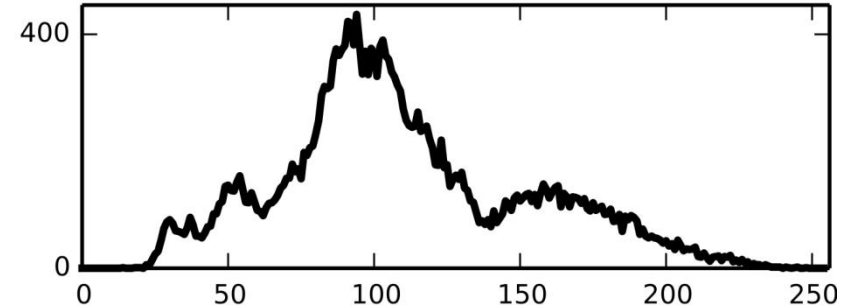


# Pipeline Example

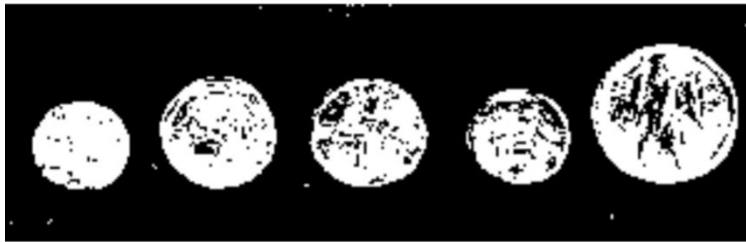
(a) Original



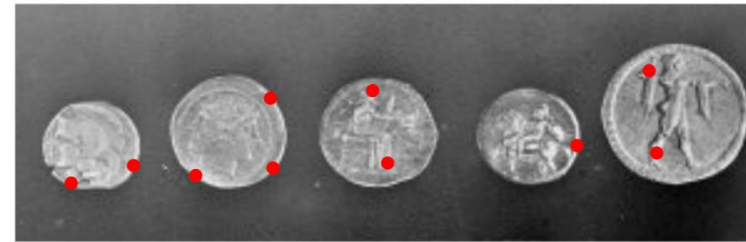
(b) Histogram



(c) Adaptive threshold



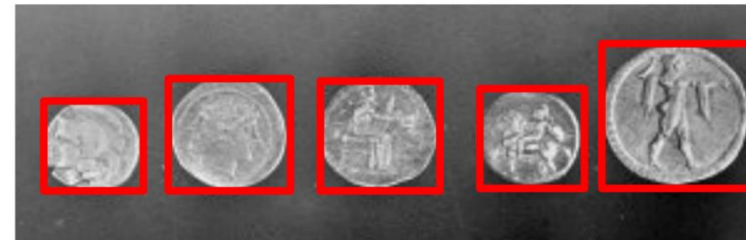
(d) Peak local maxima



(e) Edges



(f) Labeled items



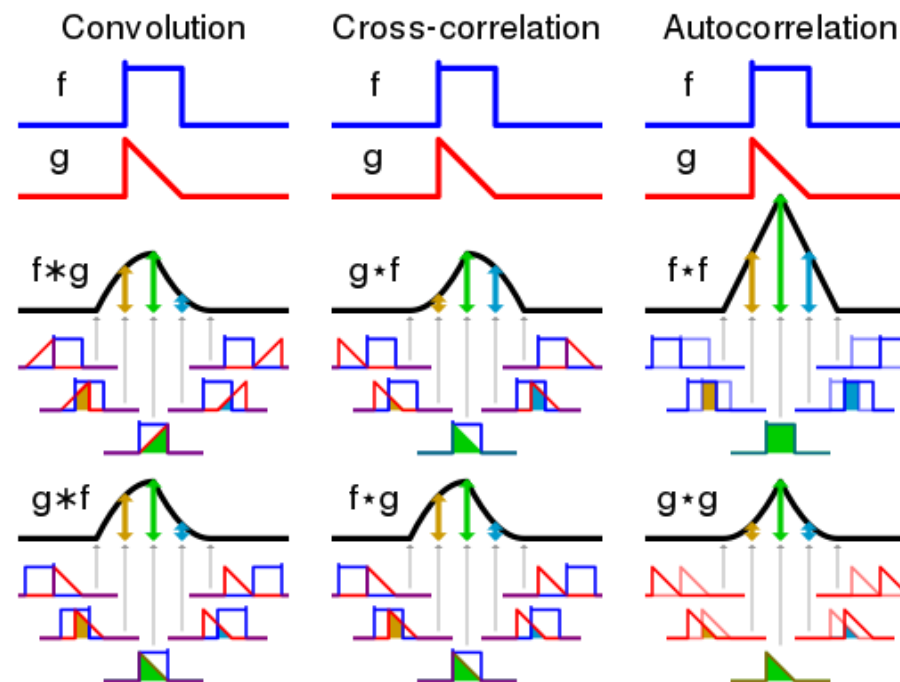
# CONVOLUTION

---

Convolution - 1d, 2d, Gaussian, vs Correlation

# Convolution

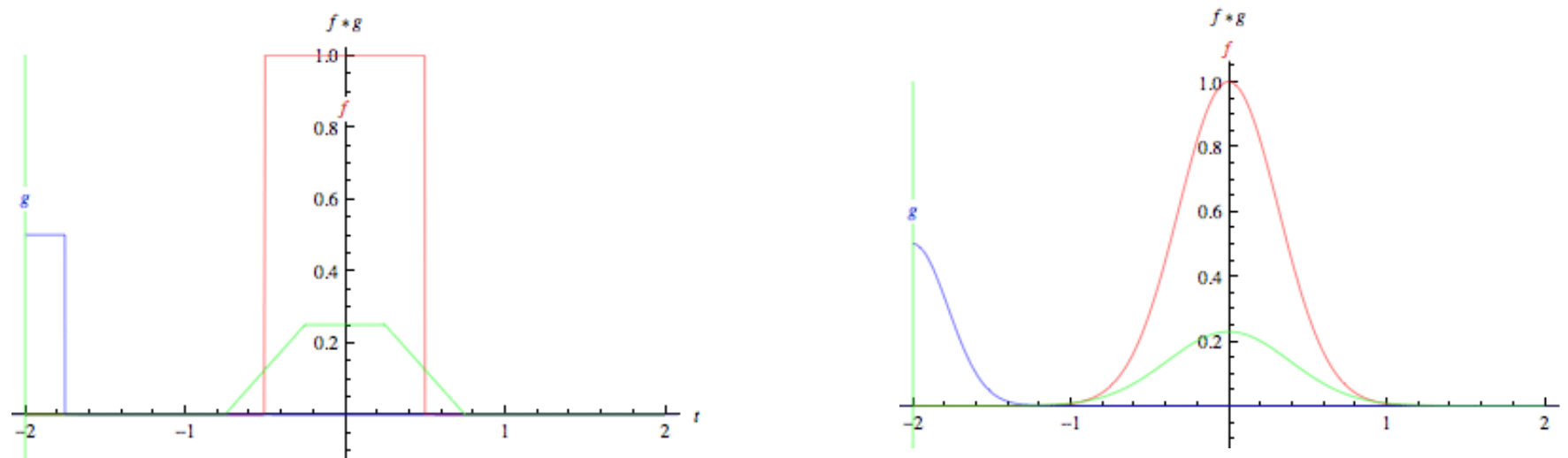
In mathematics convolution is a mathematical operation on two functions that produces a third function expressing how the shape of one is modified by the other. The term convolution refers to both the result function and to the process of computing it.





# Convolution

Convolution “blends” one function with another



The animations above graphically illustrate the convolution of two boxcar functions (left) and two Gaussians (right). In the plots, the green curve shows the convolution of the blue and red curves as a function of  $t$ , the position indicated by the vertical green line. The gray region indicates the product  $g(\tau)f(t - \tau)$  as a function of  $t$ , so its area as a function of  $t$  is precisely the convolution

# Convolution and Correlation

- Widely used operations in Image/Signal Processing and Deep Learning
- Both operations are:
  - Linear - every point can be replaced with a linear combination of its neighbors
  - Shift-invariant - perform same operation at every point in the image
  - Identical when filter is symmetrical

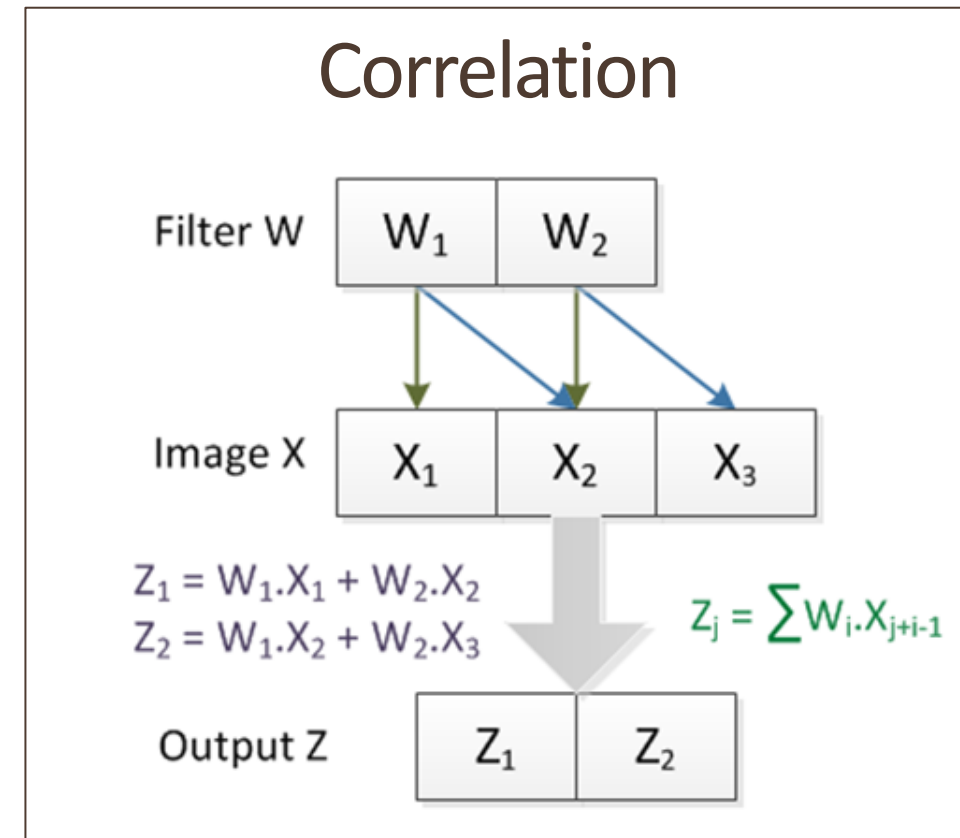
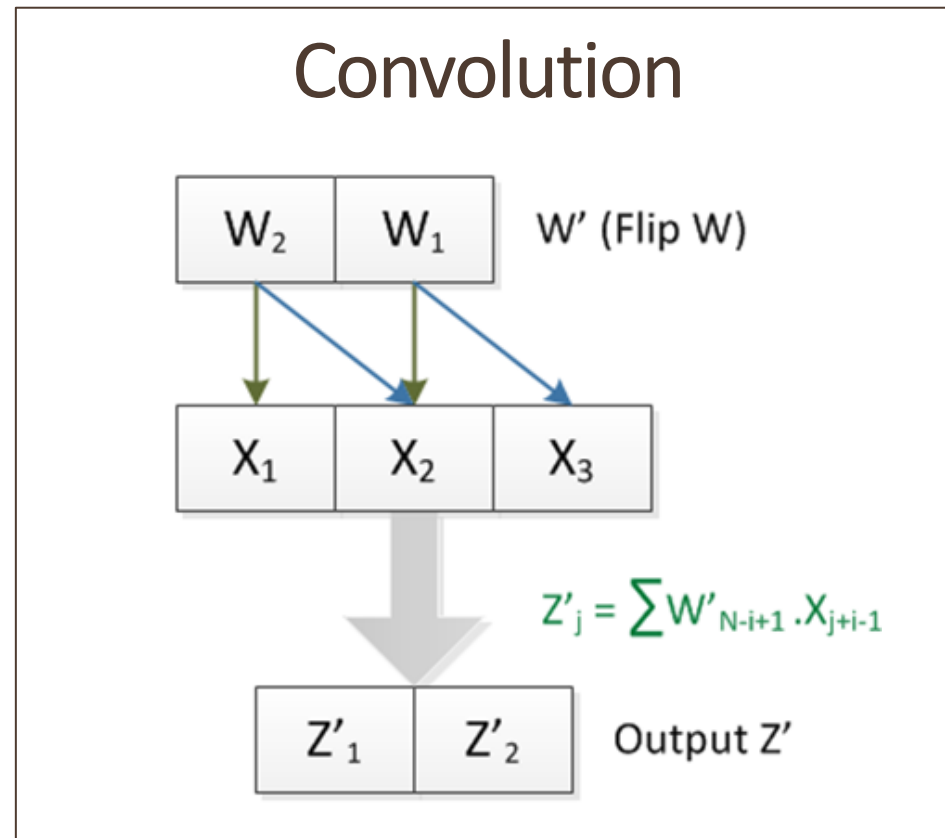
## Convolution

- Measures response to an input signal
- Image processing
- E.g. De-noising, edge detection, etc.

## Correlation

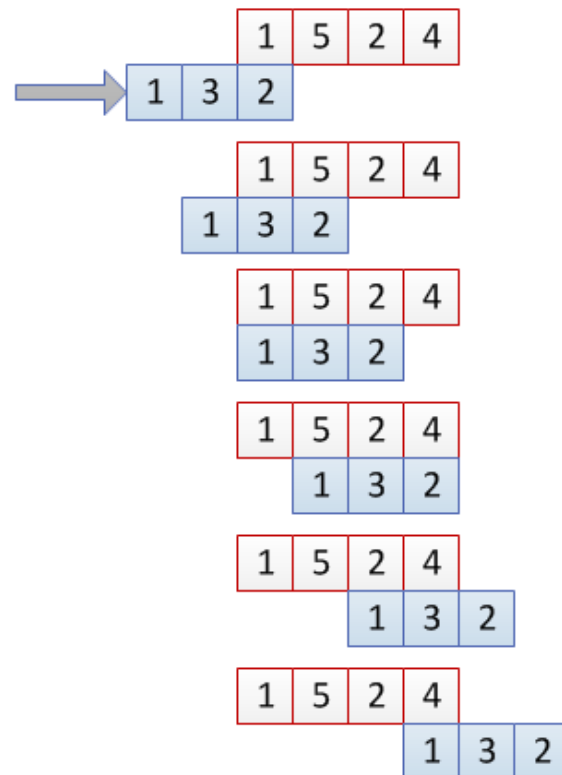
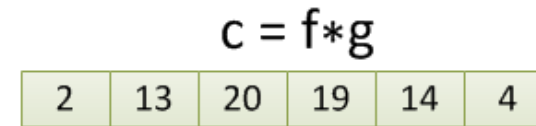
- Measures similarity between signals
- Pattern matching
- E.g. 'Cross-Correlation' looks for a match of a given shape within an image.

# Convolution and Correlation



Both operations are identical when filter is symmetrical

# Convolution (1d)



$$c[0] = 1 * 2 = 2$$

$$c[1] = 1 * 3 + 5 * 2 = 13$$

$$c[2] = 1 * 1 + 5 * 3 + 2 * 2 = 20$$

$$c[3] = 5 * 1 + 2 * 3 + 4 * 2 = 19$$

$$c[4] = 2 * 1 + 4 * 3 = 14$$

$$c[5] = 4 * 1 = 4$$

# Convolution (1d)

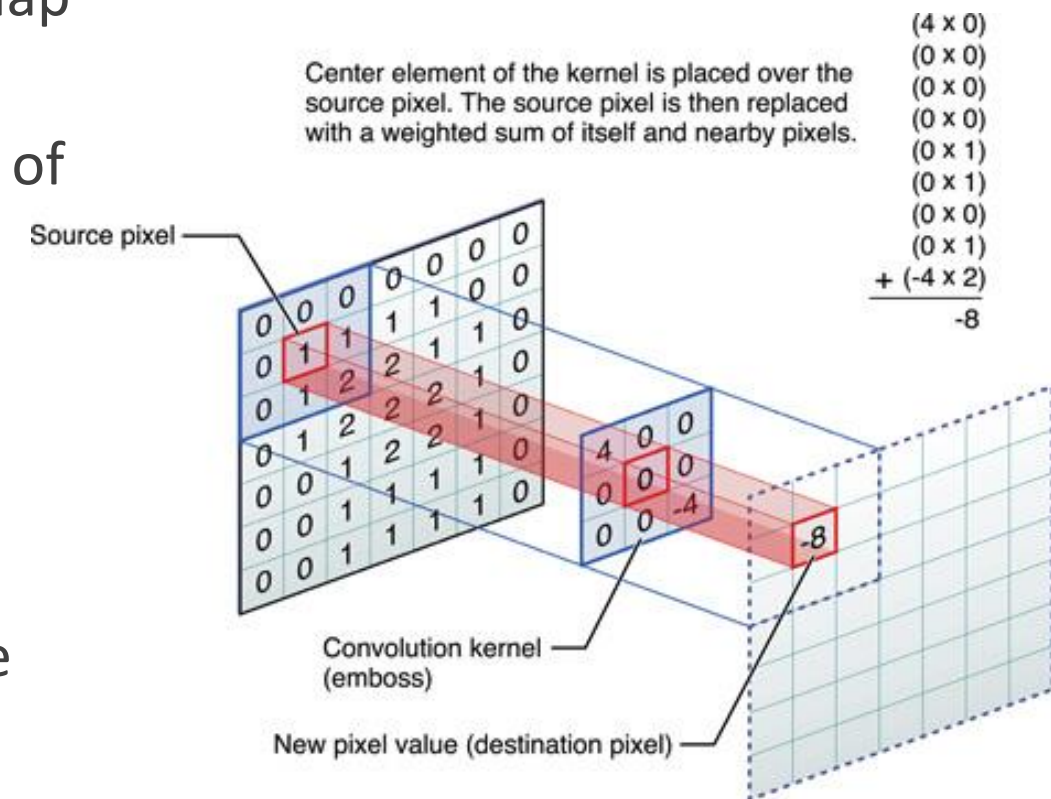
Operation on two functions **f** and **g**, which produces a third function that can be interpreted as a modified ("filtered") version of **f**.

$$f[x] * g[x] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[x - k]$$

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau) \cdot g(x - \tau) d\tau$$

# Convolution (2d)

- Input feature map (or input) and the convolution kernel blend to form a transformed feature map (or output)
- Kernel filters the feature map for information of a certain kind (e.g. edges) and discard other information.
- Slide the filter over the image and create outputs – feature maps for convolution each layer
- Pad images so that filter can slide through the image



# Convolution Steps

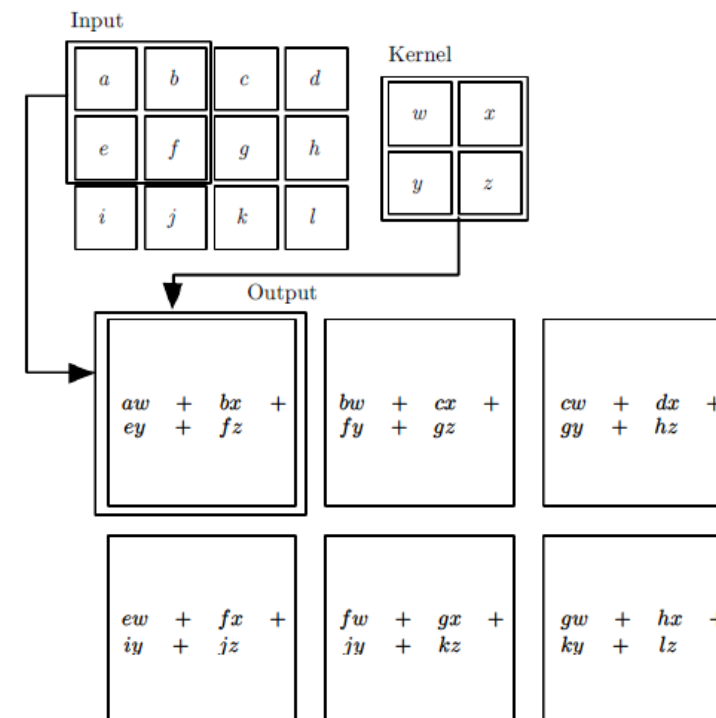
1. Slide kernel over the entire image.
2. Image patch (yellow) of the original image (green) is multiplied by the kernel (red numbers in the yellow patch).
3. Its sum is written to one feature map pixel (red cell in convolved feature).

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature





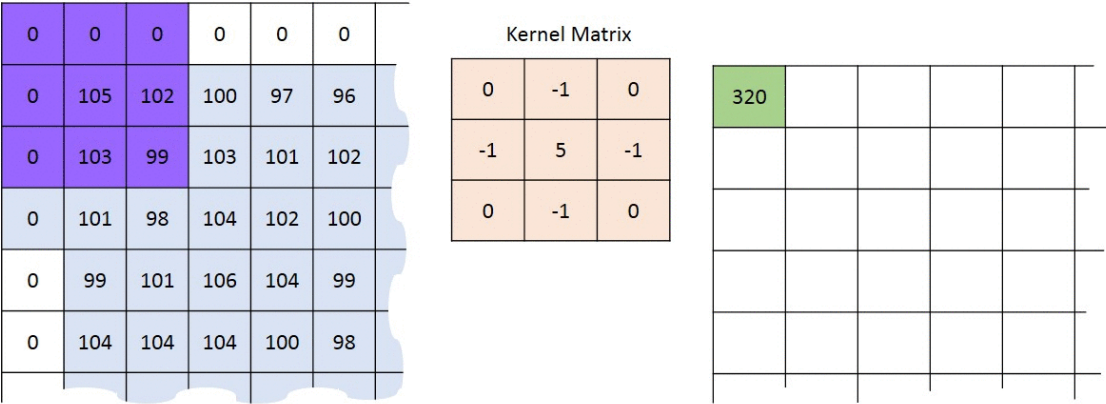
# Convolution (2d)

In digital photography, placing an antialiasing filter in front of the sensor convolves an image  $f(x,y)$  by a smoothing filter  $g(x,y)$

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1,y-n_2]$$

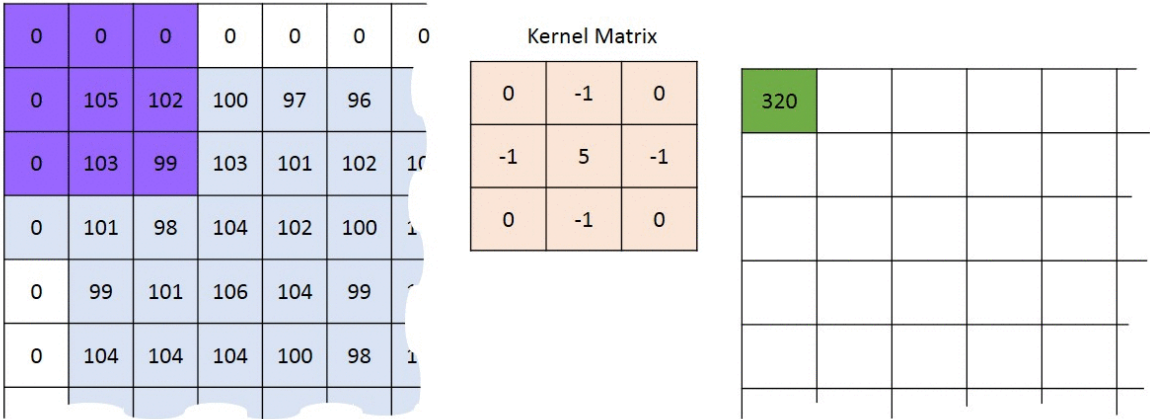
$$f(x,y) * g(x,y) = \int_{\tau_1=-\infty}^{\infty} \int_{\tau_2=-\infty}^{\infty} f(\tau_1,\tau_2) \cdot g(x-\tau_1,y-\tau_2) d\tau_1 d\tau_2$$

# Convolution - Strides



$$0 * 0 + 0 * -1 + 0 * 0 + 0 * -1 + 105 * 5 + 102 * -1 + 0 * 0 + 103 * -1 + 99 * 0 = 320$$

Convolution with horizontal and vertical strides = 1

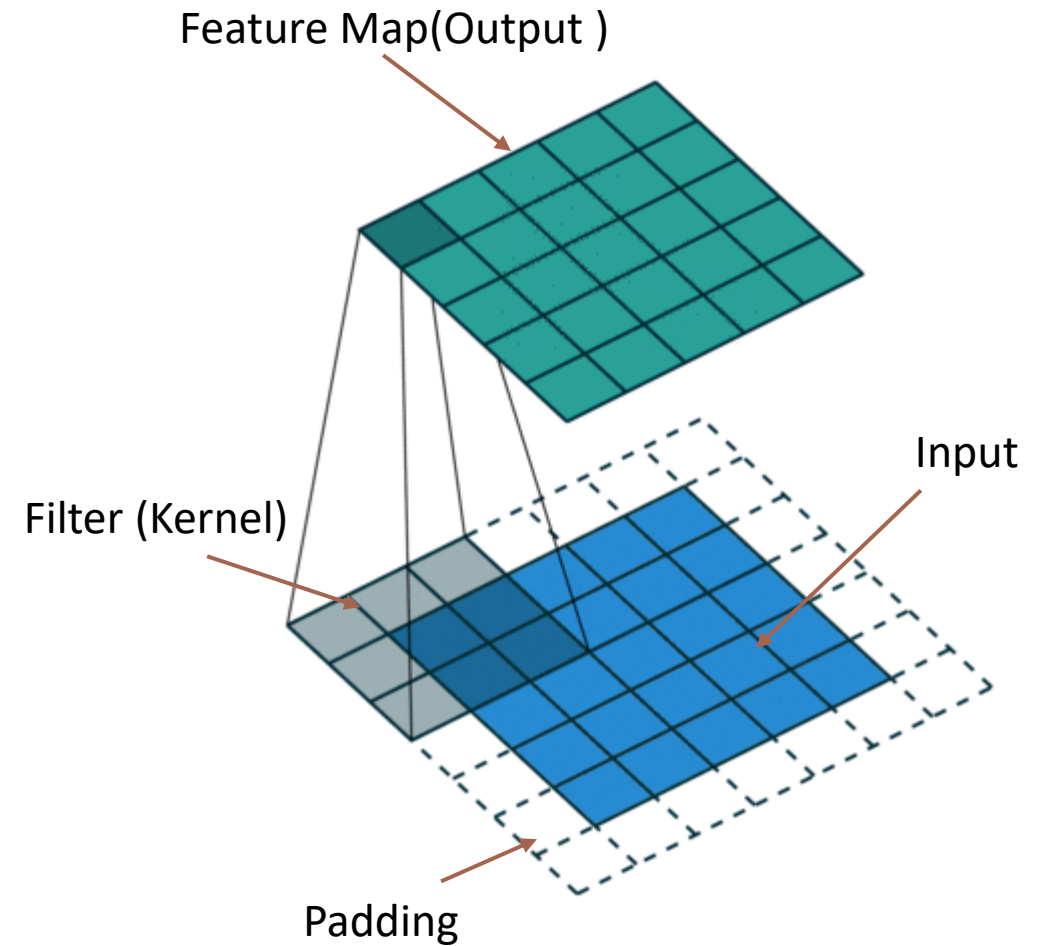


$$0 * 0 + 0 * -1 + 0 * 0 + 0 * -1 + 105 * 5 + 102 * -1 + 0 * 0 + 103 * -1 + 99 * 0 = 320$$

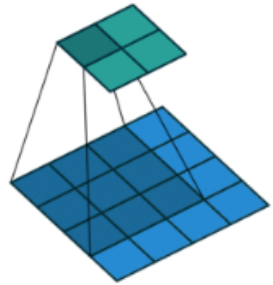
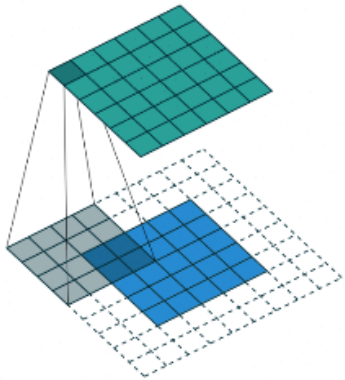
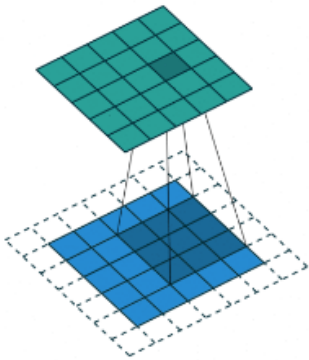
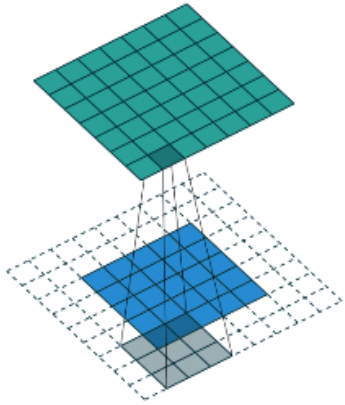
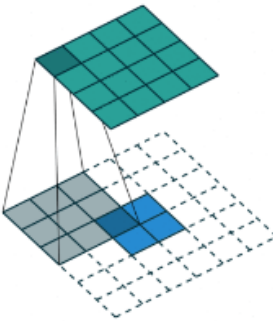
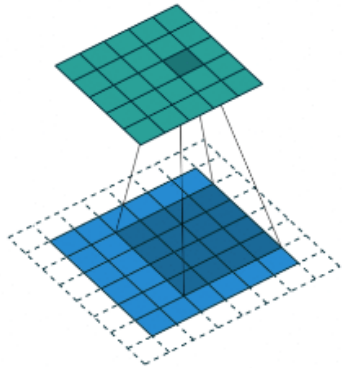
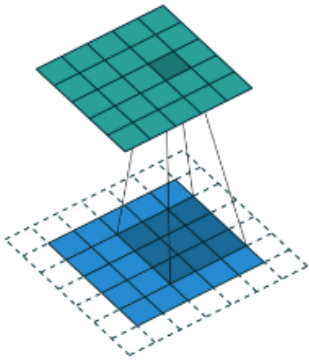
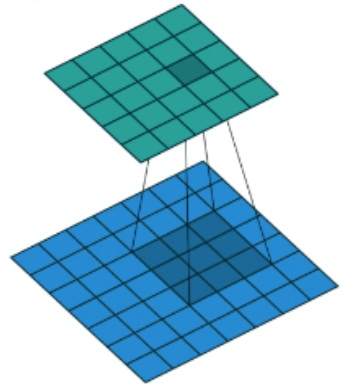
Convolution with horizontal and vertical strides = 2

# Convolution Parameters

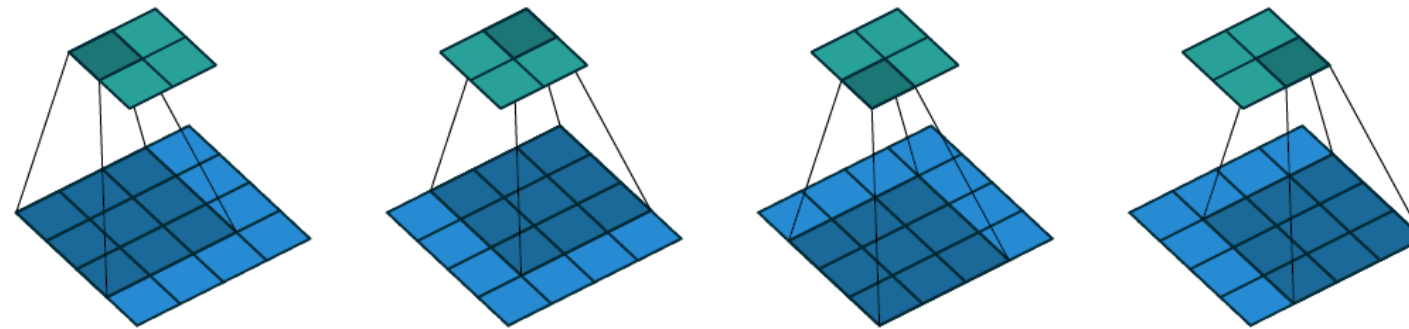
- Filter Size (F)
  - Size of convolution filter
- Padding (P)
  - Allows filter to slide through the image
  - Generates output feature map of desired size.
  - Padding with 0 is most common
- Stride (S)
  - No of steps the filter slides
  - $S = 1$ , 1 pixel at a time
  - $S \geq 2$  produces smaller output volumes



# Convolution - Padding, Strides

			
No padding, no strides	Arbitrary padding, no strides	Half padding, no strides	Full padding, no strides
			
No padding, no strides, transposed	Arbitrary padding, no strides, transposed	Half padding, no strides, transposed	Full padding, no strides, transposed

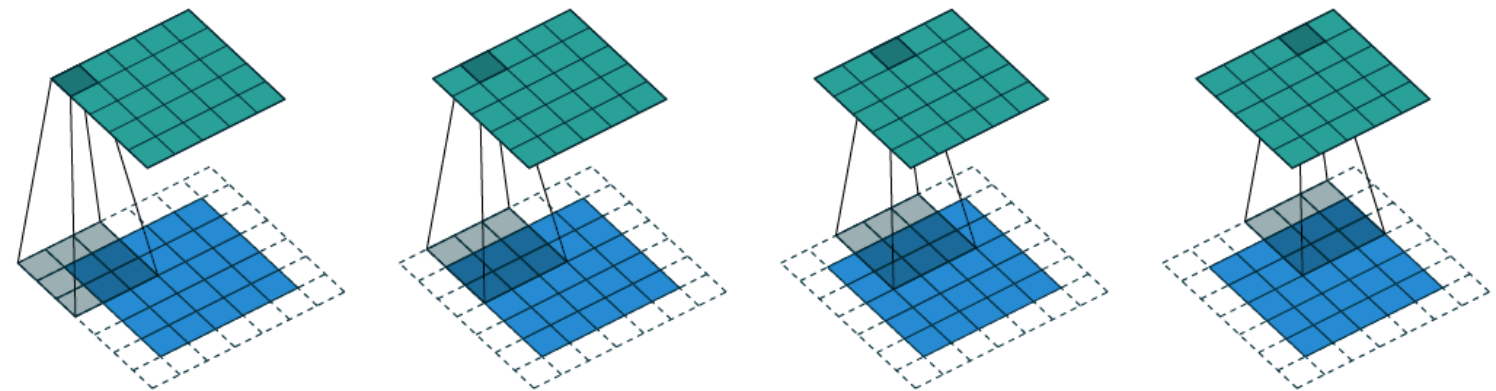
# Convolution Example



2D discrete convolutions ( $N = 2$ ),  
square inputs ( $i_1 = i_2 = i$ ),  
square kernel size ( $k_1 = k_2 = k$ ),  
same strides along both axes ( $s_1 = s_2 = s$ ),  
same zero padding along both axes ( $p_1 = p_2 = p$ )

Convolving a 3x3 kernel over a 4x4 input using unit strides (i.e.,  $i = 4$ ,  $k = 3$ ,  $s = 1$  and  $p = 0$ )

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1$$



Convolving a 3x3 kernel over a 5x5 input using half (same) padding and unit strides (i.e.,  $i = 5$ ,  $k = 3$ ,  $s = 1$  and  $p = 1$ )

# Convolution - Full , Same, Valid

$$\begin{pmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{pmatrix} * \begin{pmatrix} 1 & 3 & 1 \\ 0 & 5 & 0 \\ 2 & 1 & 2 \end{pmatrix}$$

full      same      valid

17	75	90	35	40	53	15
23	159	165	45	105	137	16
38	198	120	165	205	197	52
56	95	160	200	245	184	35
19	117	190	255	235	106	53
20	89	160	210	75	90	6
22	47	90	65	70	13	18

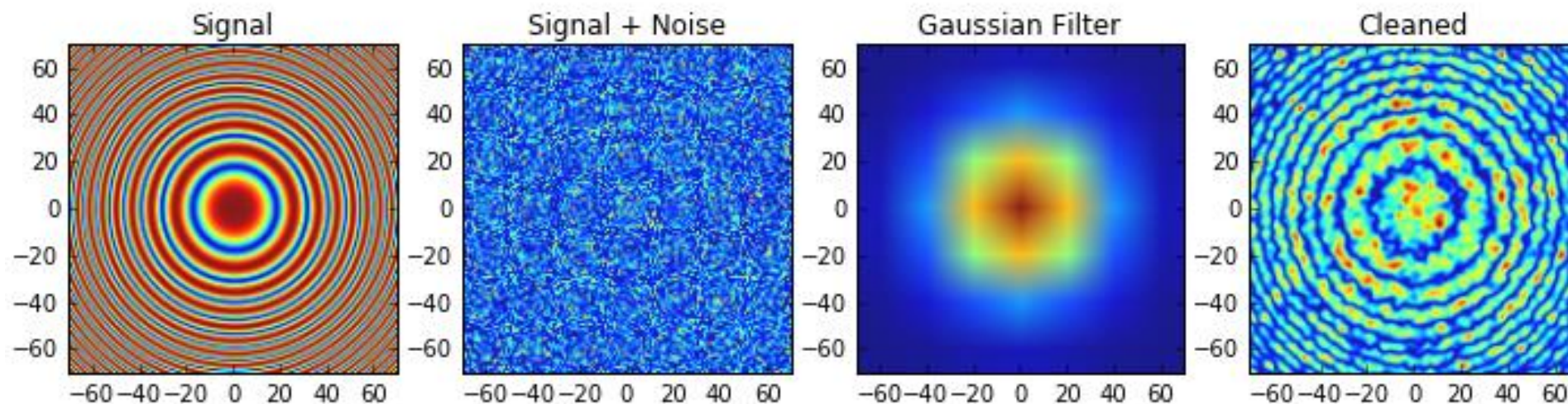
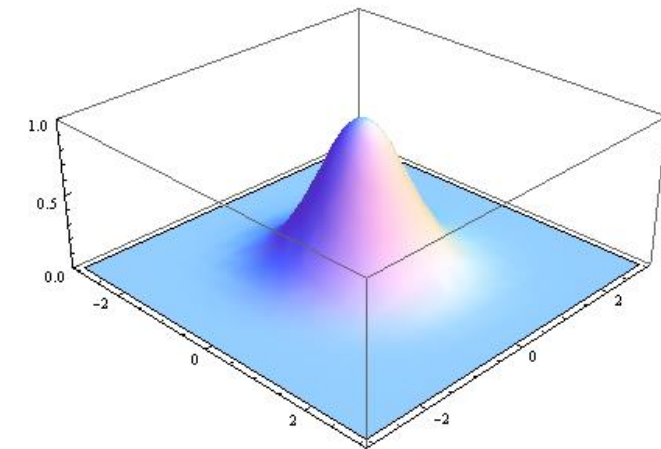
The diagram illustrates the result of a 1D convolution operation. The input is a 5x5 matrix, and the kernel is a 3x3 matrix. The output is a 7x7 matrix. The 'full' output is the entire 7x7 matrix. The 'same' output is the 5x5 sub-region of the full output, centered at (3,3). The 'valid' output is the 3x3 sub-region of the same output, centered at (3,3).



# Convolution - Gaussian Kernel

- The Gaussian smoothing operator is a 2d convolution operator as a point-spread function to remove details or noise.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$





# LAB

---

Image Processing – Convolution, Filtering, Detection

# Lab

1. Read and process images
2. Image filtering – rank, edge detection
3. Convolution
4. Face Detection