Convolution

Prof. Seungchul Lee
Industrial AI Lab.

POSTECH

Table of Contents

Systems

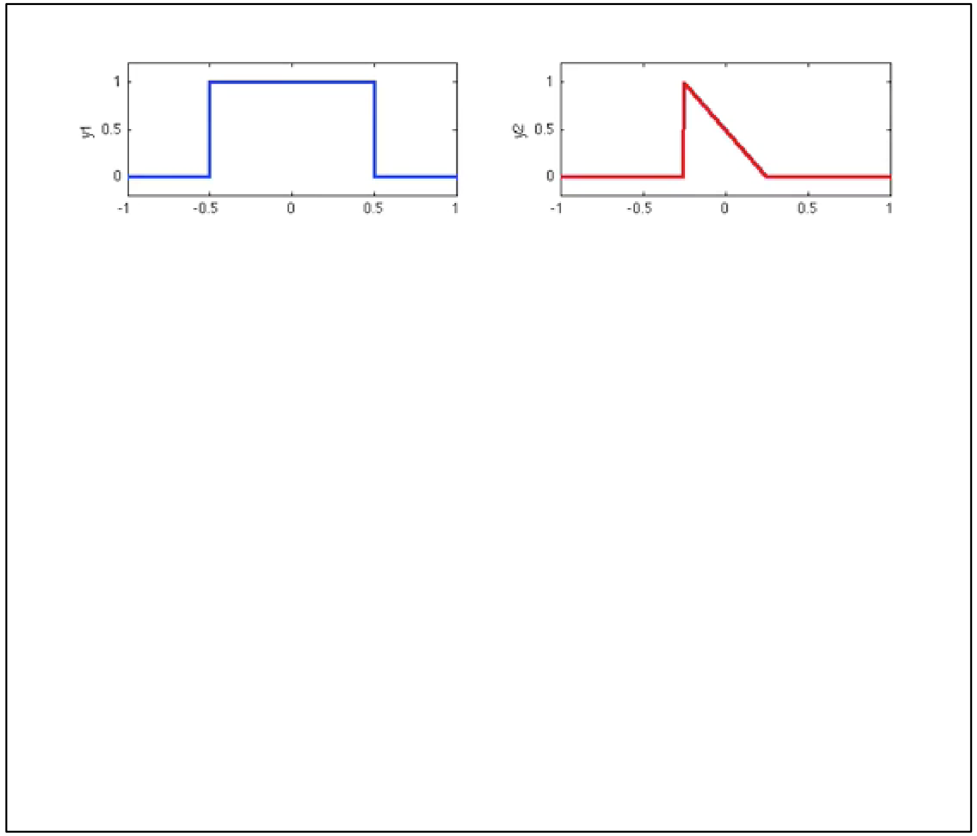
- A discrete-time system $\mathcal{H}$ is a transformation (a rule or formula) that maps a discrete-time input signal x into a discrete-time output signal y

$$x \longrightarrow \boxed{\mathcal{H}} \longrightarrow y$$

Convolution

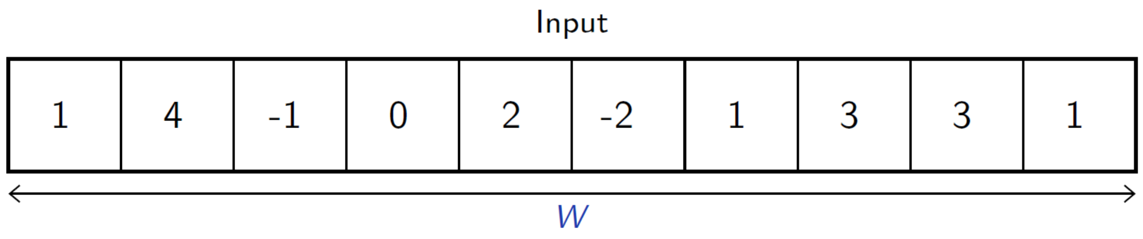
- Integral of the product of the two signals after one is reversed and shifted
- Cross correlation and convolution

$$y[n] = \sum_{m=-\infty}^{\infty} h[n-m]\, x[m] = x[n] * h[n]$$

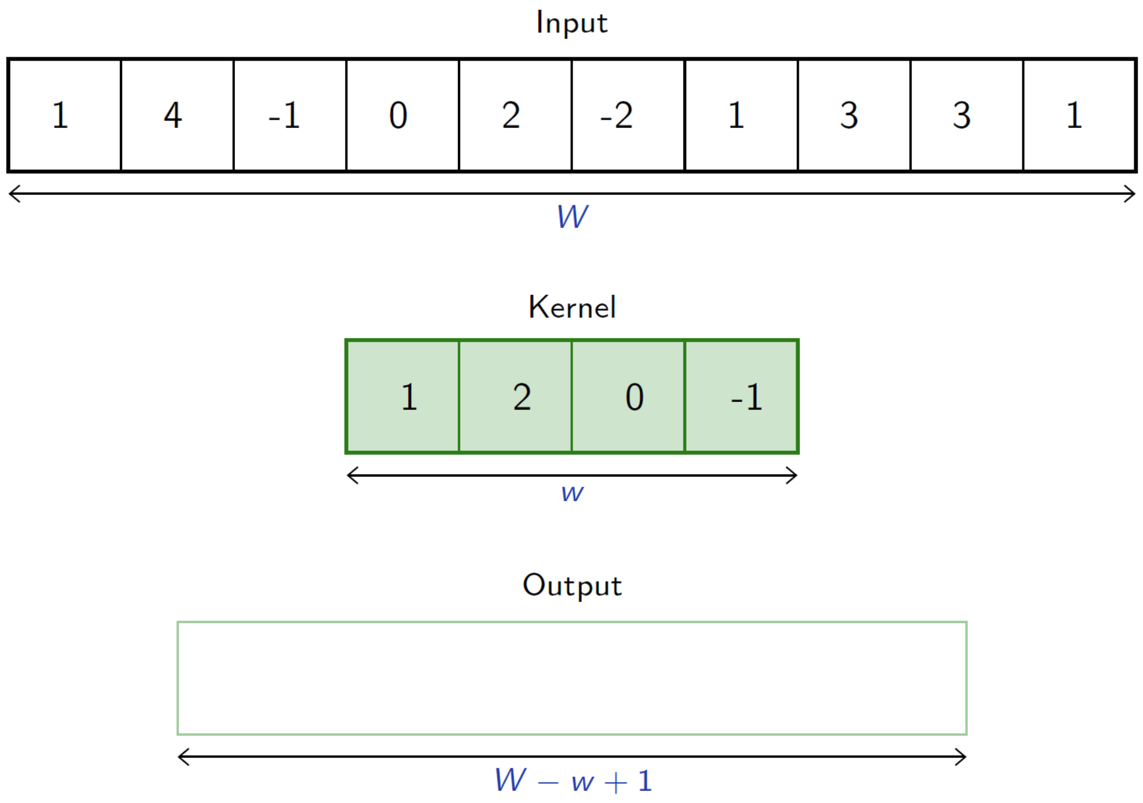


1D Convolution

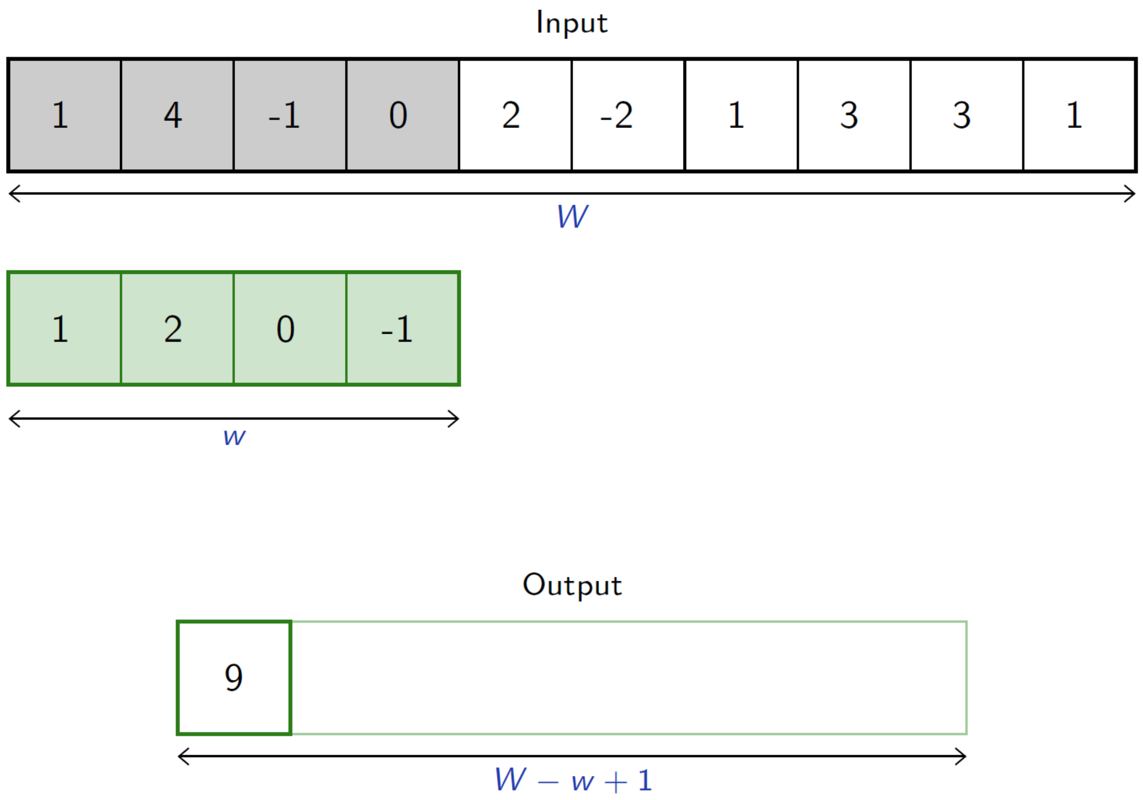
- (actually cross-correlation)



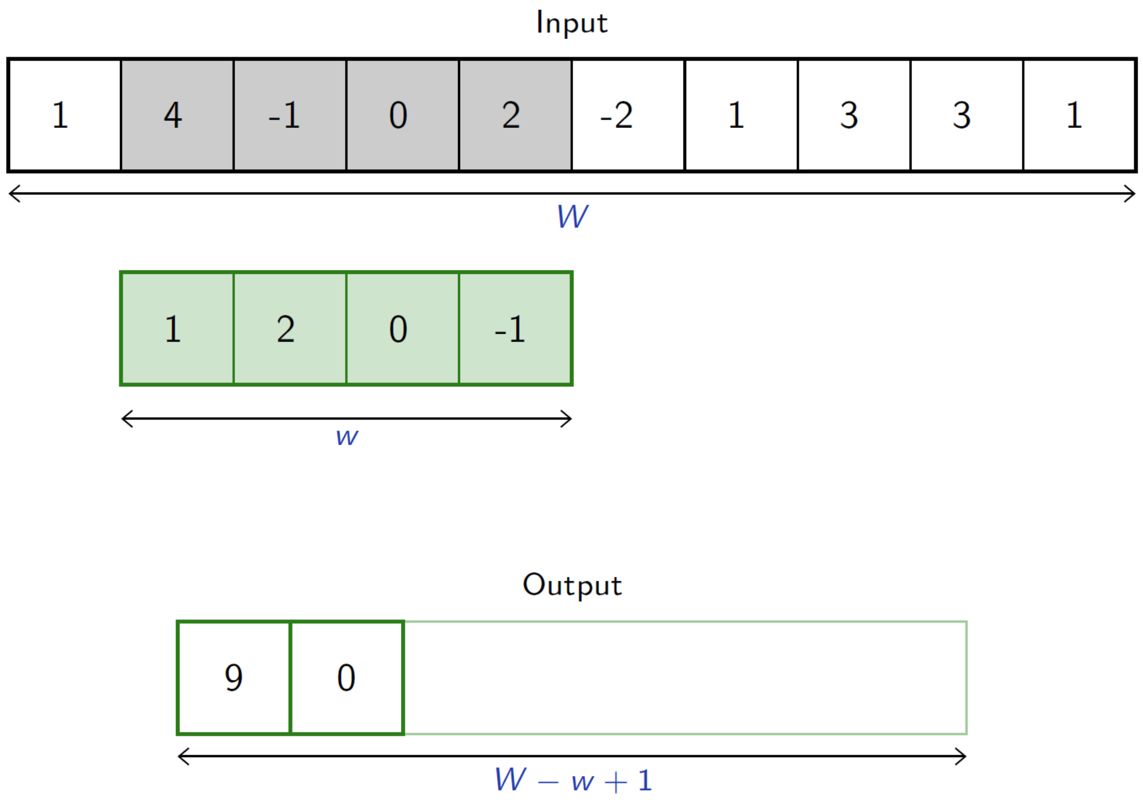
1D Convolution



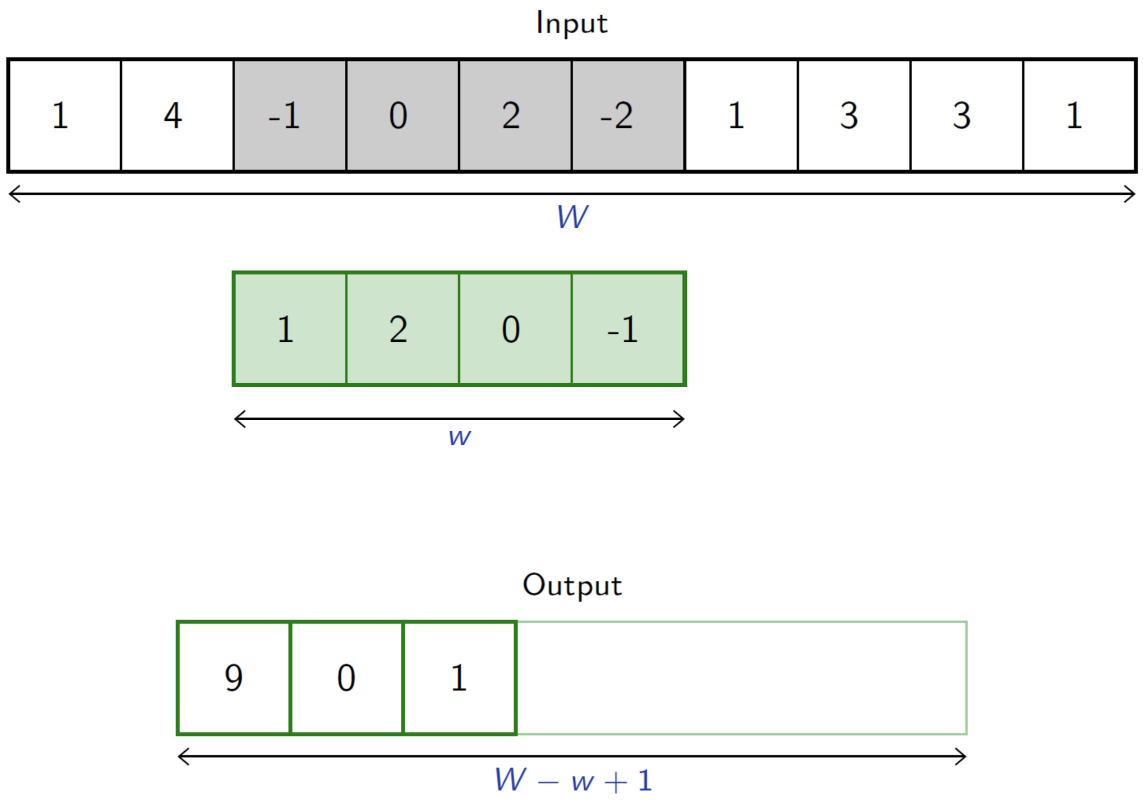
1D Convolution



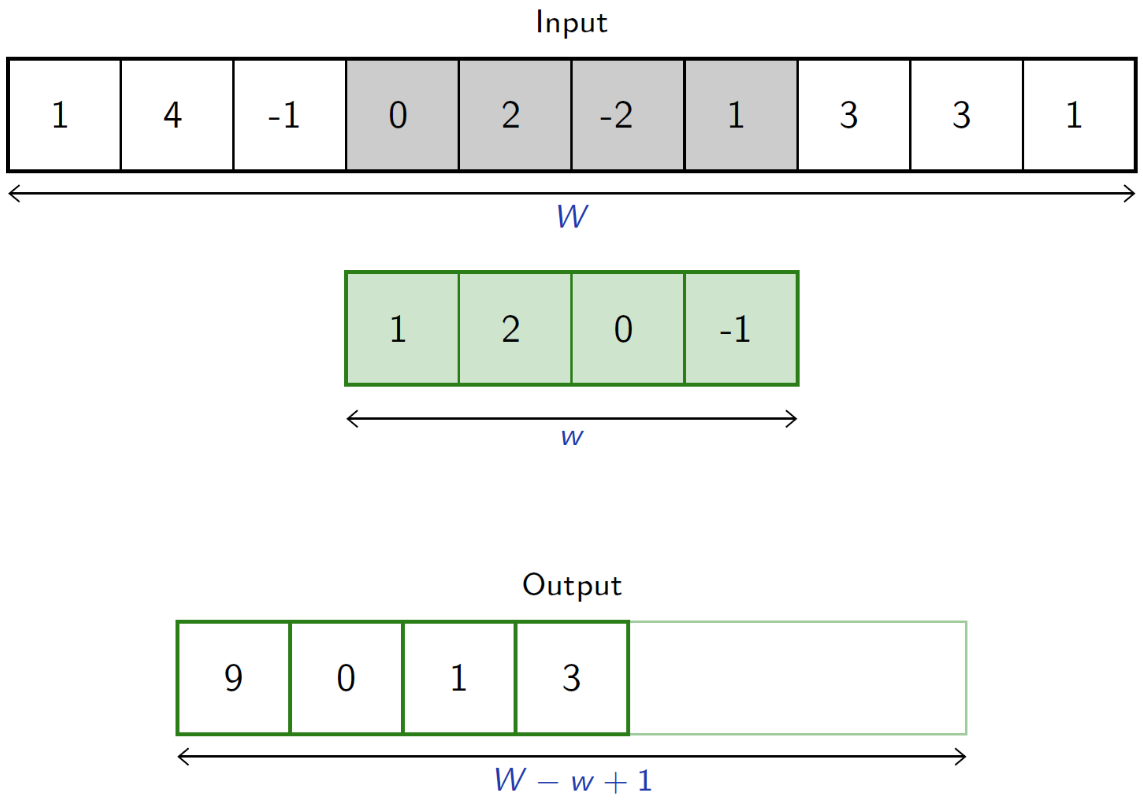
1D Convolution



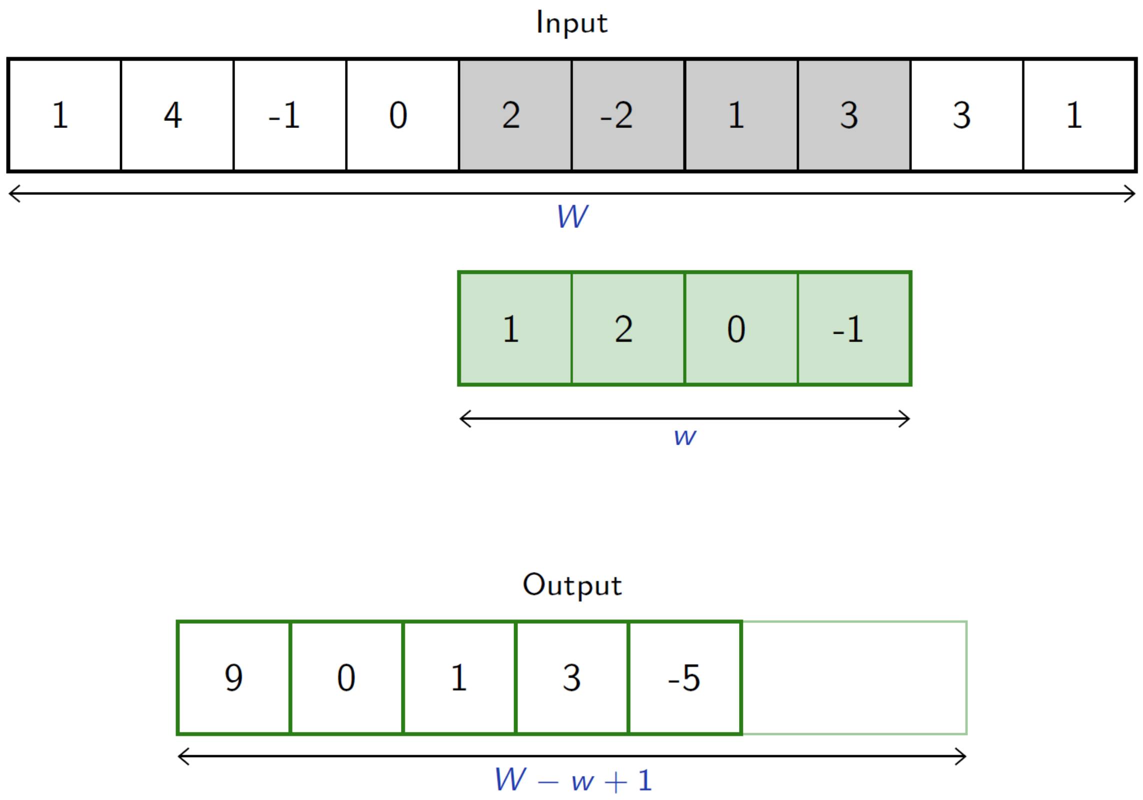
1D Convolution



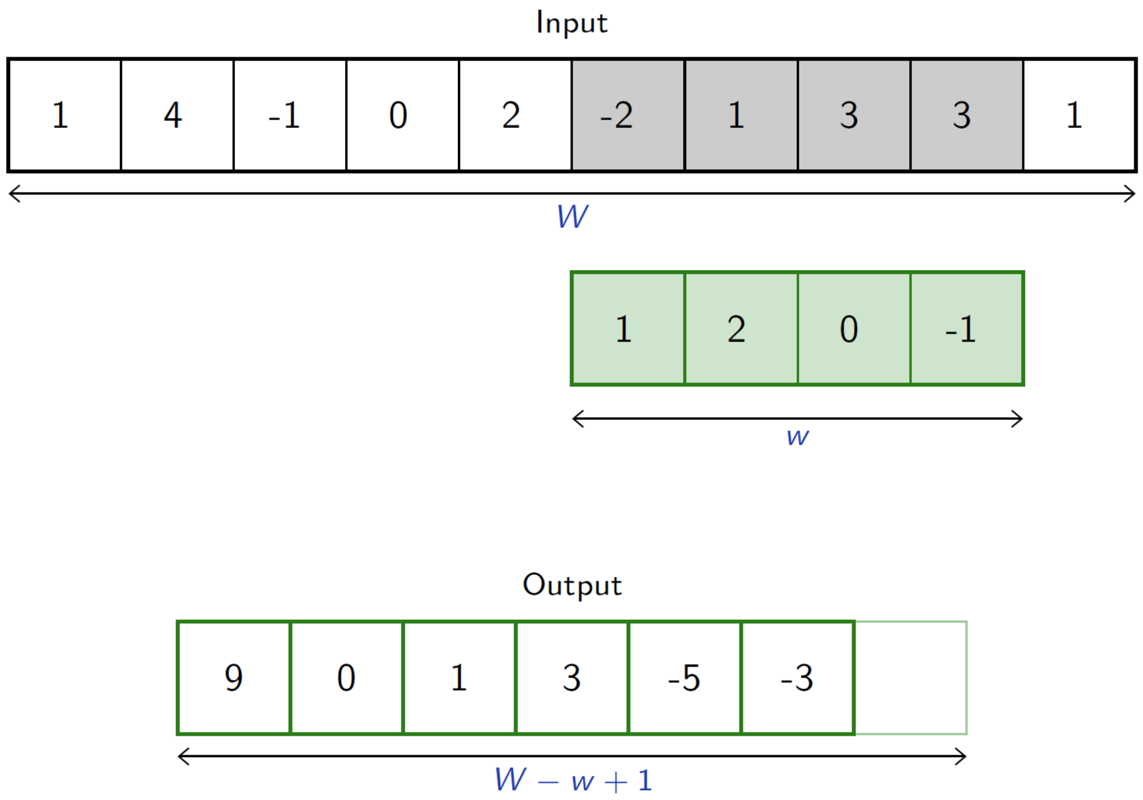
1D Convolution



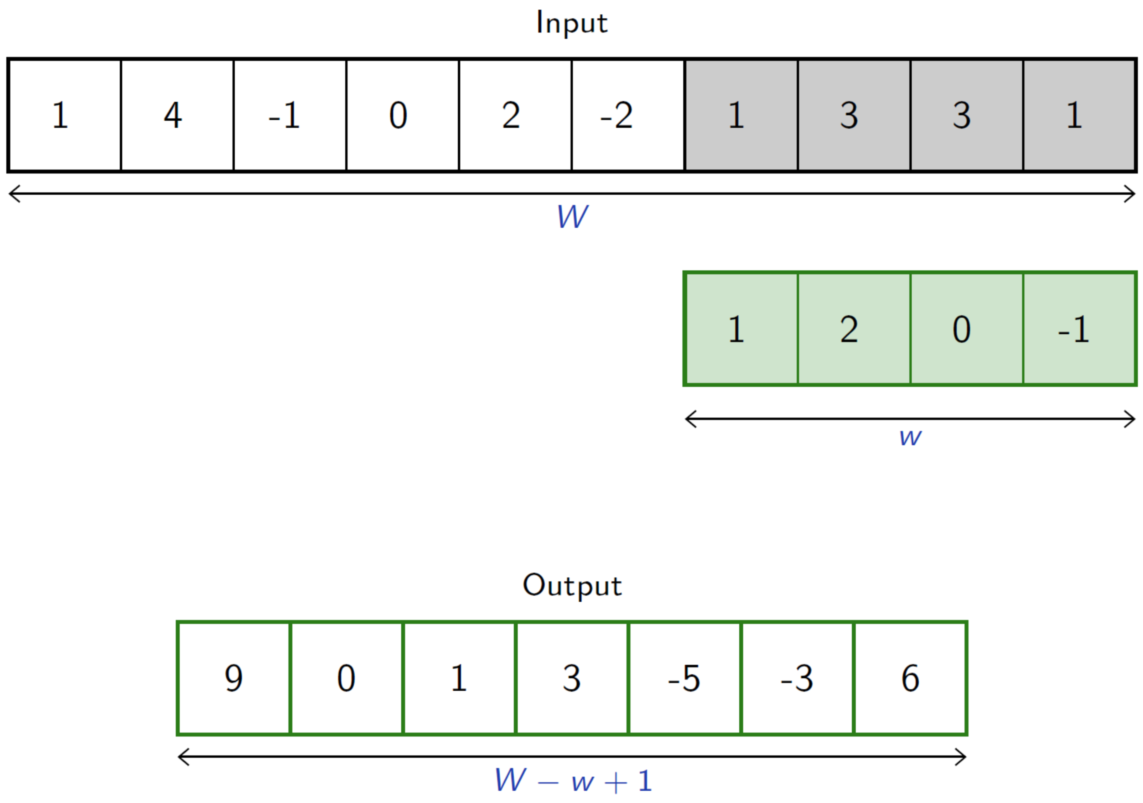
1D Convolution



1D Convolution



1D Convolution



1D Convolution

Input

1	4	-1	0	2	-2	1	3	3	1

W

Kernel

1	2	0	-1

w

Output

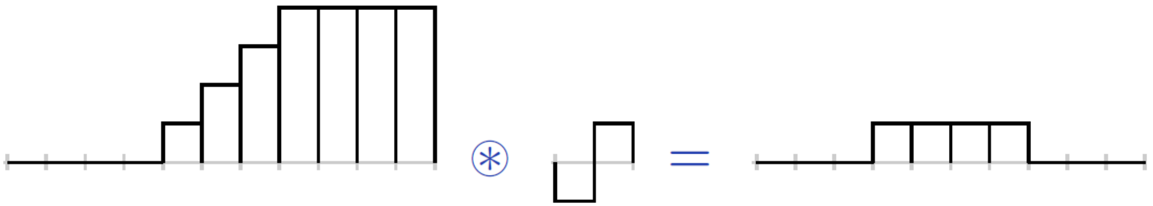
9	0	1	3	-5	-3	6

$W - w + 1$

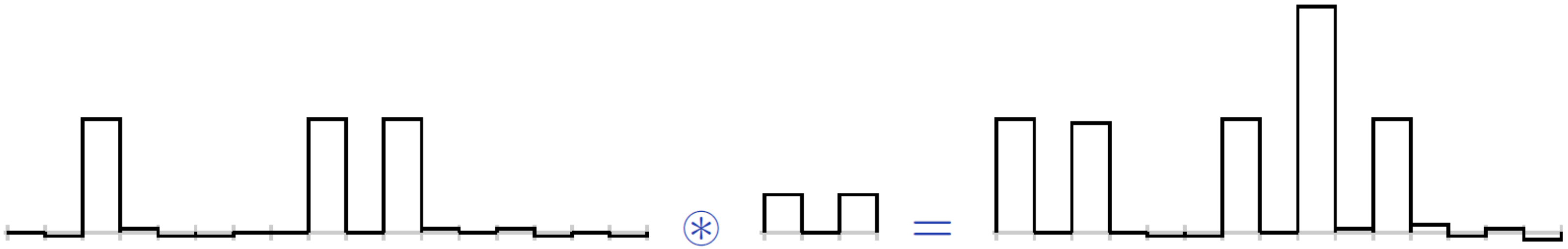
More on 1D Convolutions

- Convolution can implement in particular differential operators,

$$(0,0,0,0,1,2,3,4,4,4,4) \circledast (-1,1) = (0,0,0,1,1,1,1,0,0,0).$$



- or crude "template matcher"



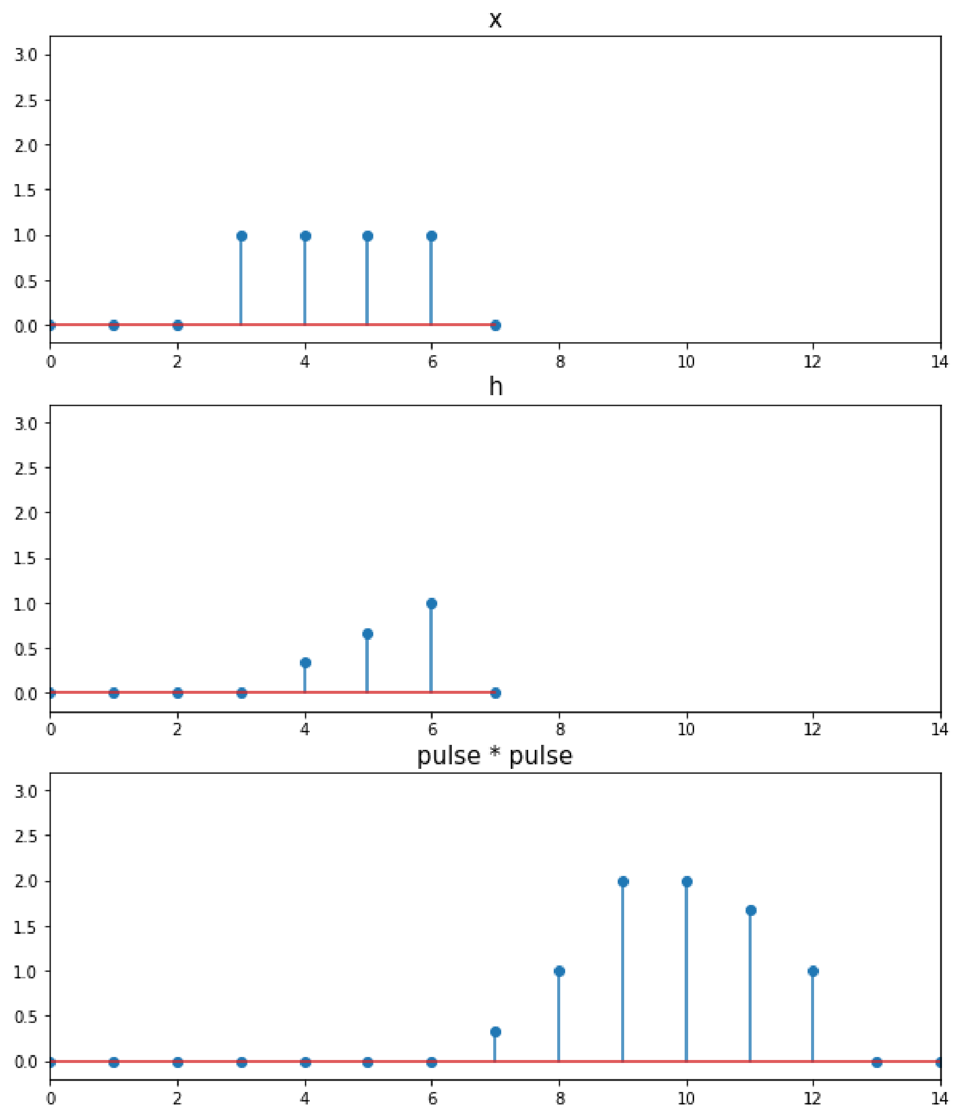
- Both of these computation examples are indeed "invariant by translation".

Table of Contents

POSTECH

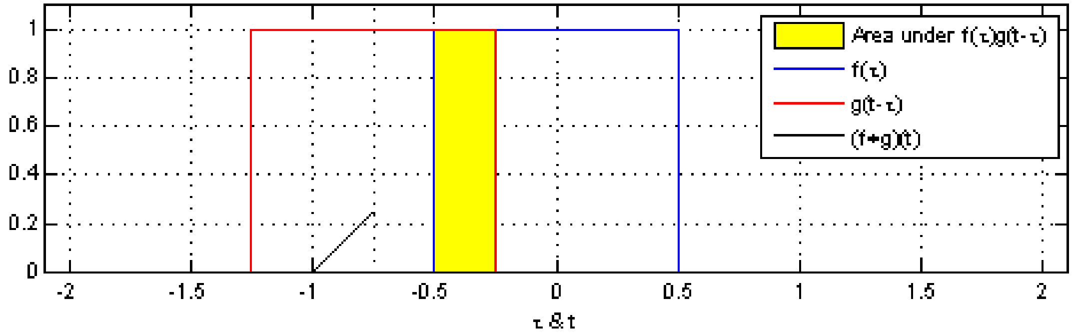
1D Convolution in Python

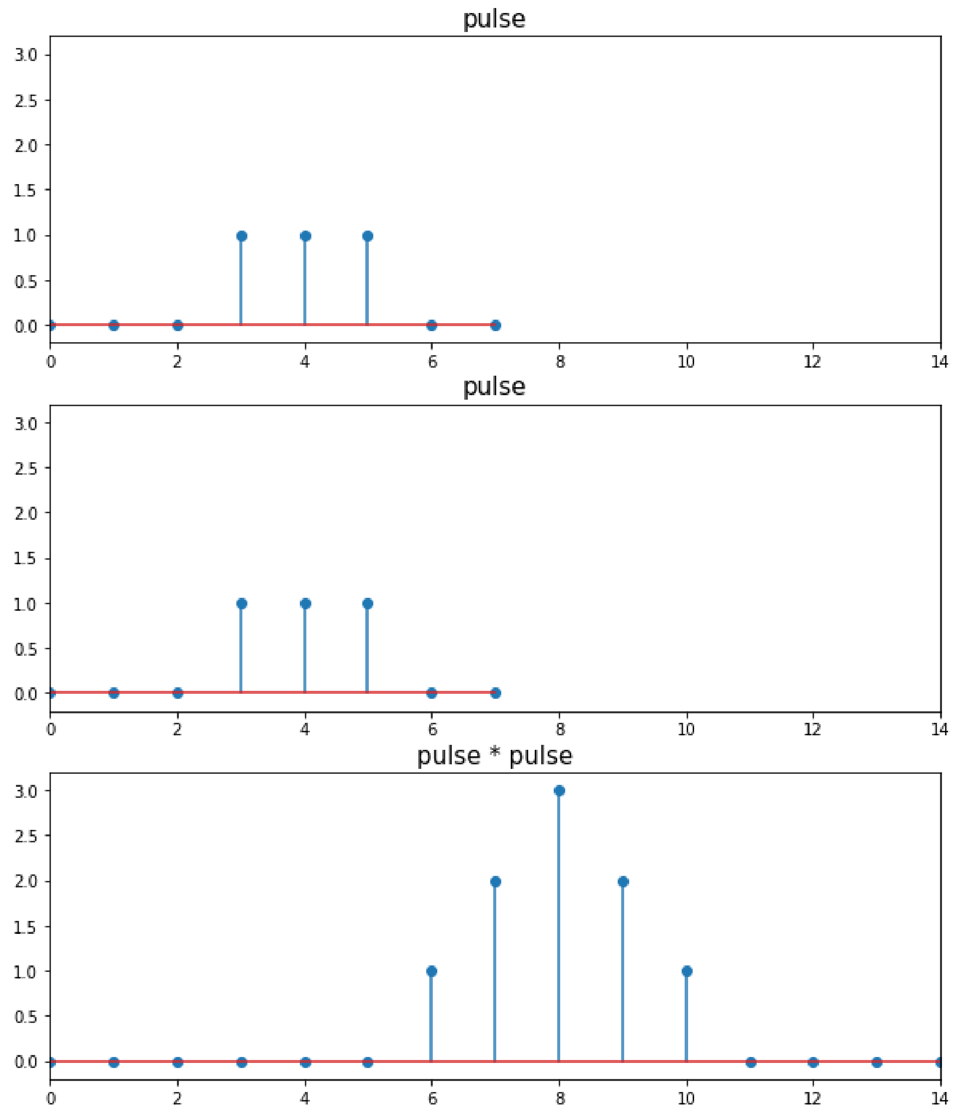
```
N = 8
n = np.arange(N)
x = [0, 0, 0, 1, 1, 1, 1, 0]
h = [0, 0, 0, 0, 1/3, 2/3 ,1, 0]

# Convolve
y = np.convolve(x, h)
```



1D Convolution in Python

```python
# pulse
N = 8
n = np.arange(N)
x = [0, 0, 0, 1, 1, 1, 0, 0]

# Convolve pulse with itself
y = np.convolve(x, x)
```





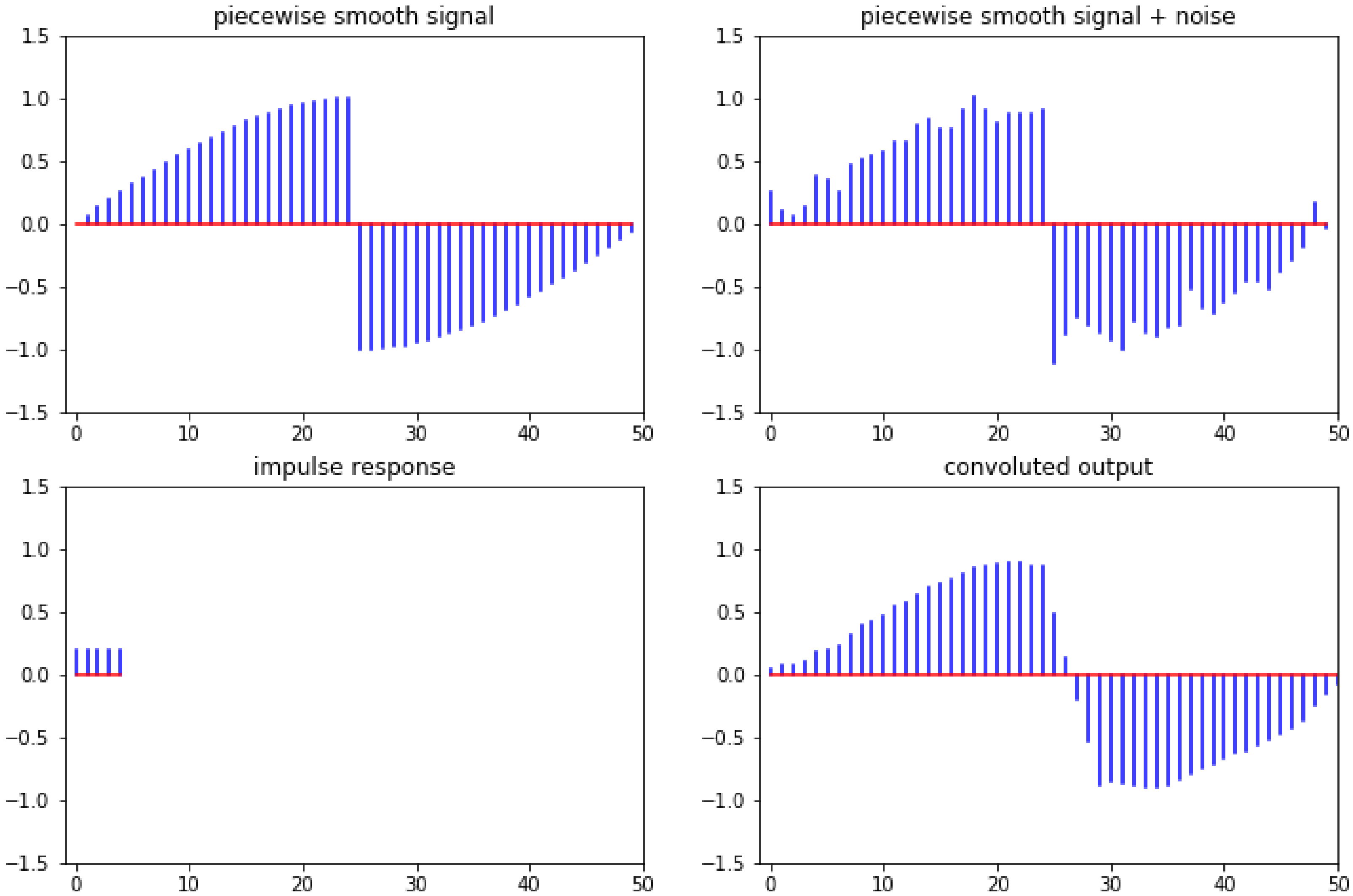
De-noising a Piecewise Smooth Signal

- Moving average (MA) filter
  - A moving average is the unweighted mean of the previous m data

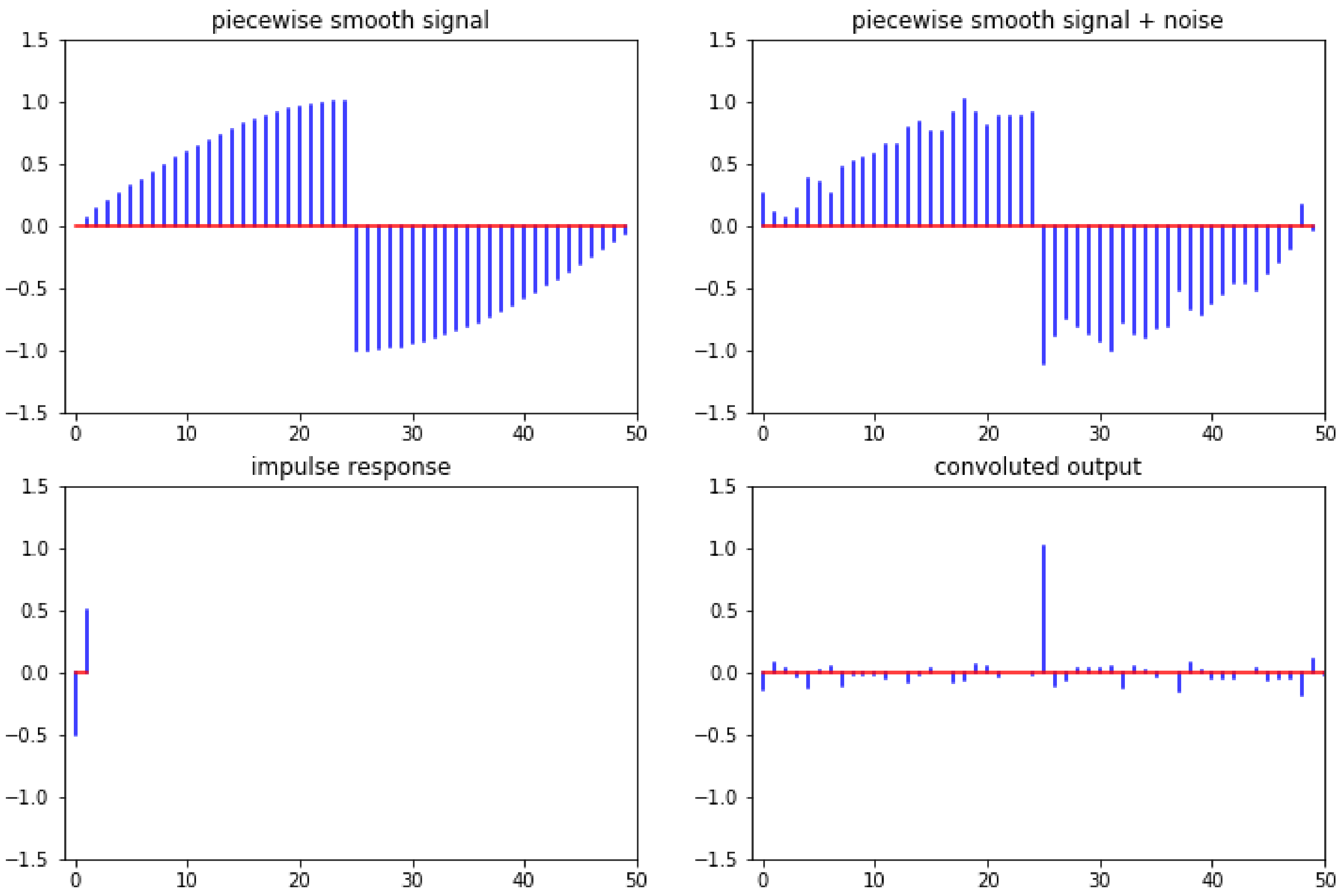
$$\begin{aligned}\bar{x}[n] &= \frac{x[n] + x[n-1] + \cdots + x[n-m+1]}{m} \\ &= (x[n], x[n-1], \cdots, x[n-m+1]) * \left(\frac{1}{m}, \frac{1}{m}, \cdots, \frac{1}{m}\right)\end{aligned}$$

- Convolution with $\left(\frac{1}{m}, \frac{1}{m}, \cdots, \frac{1}{m}\right)$
- low-pass filter in time domain

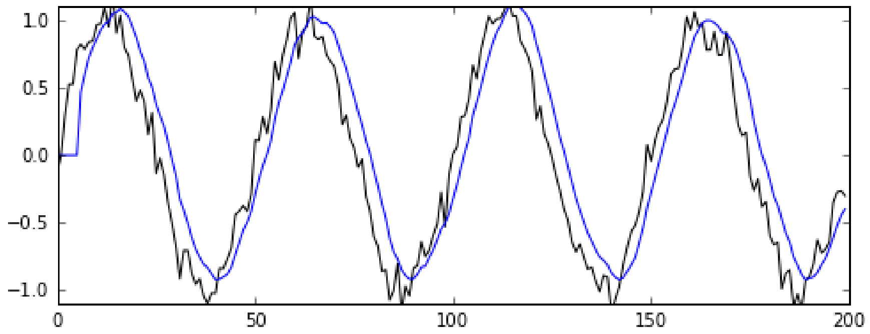
De-noising a Piecewise Smooth Signal

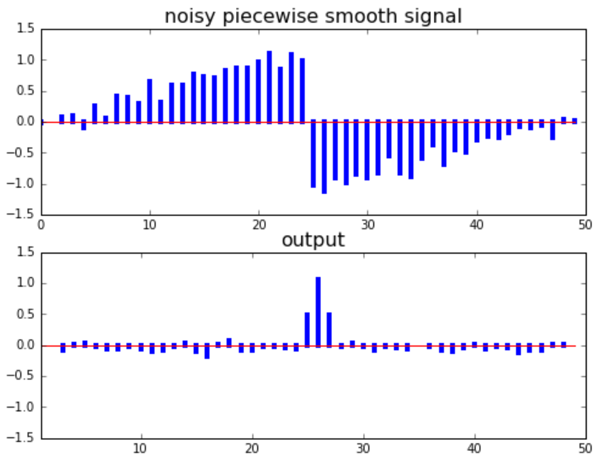


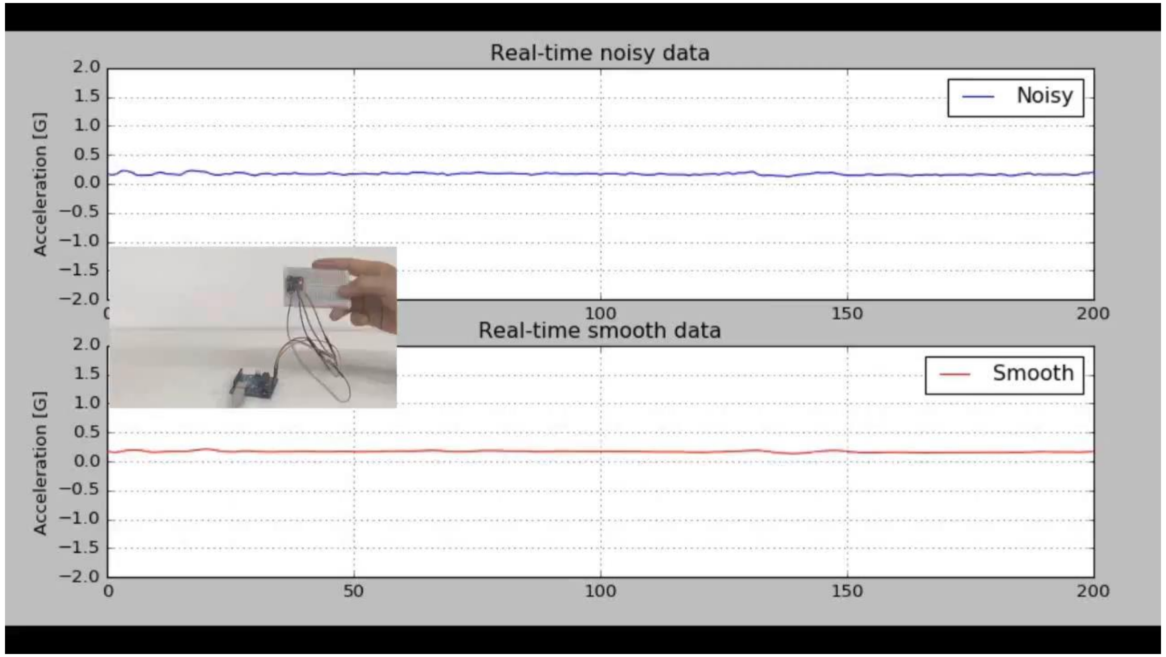
Edge Detection

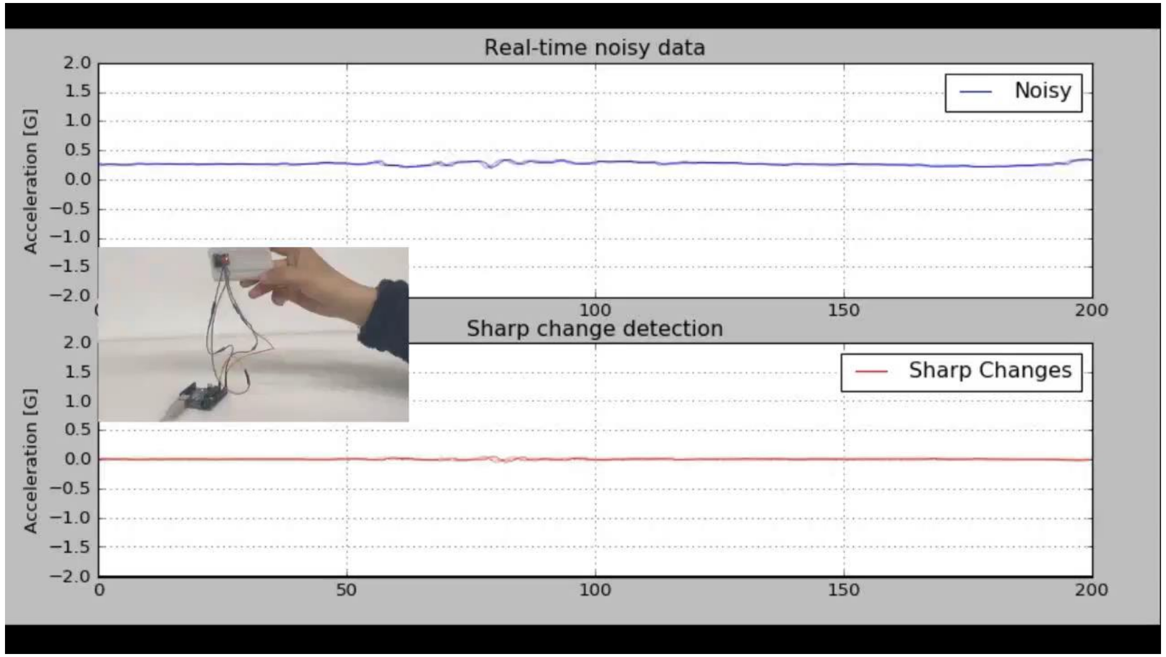


Smoothing and Detection of Abrupt Changes





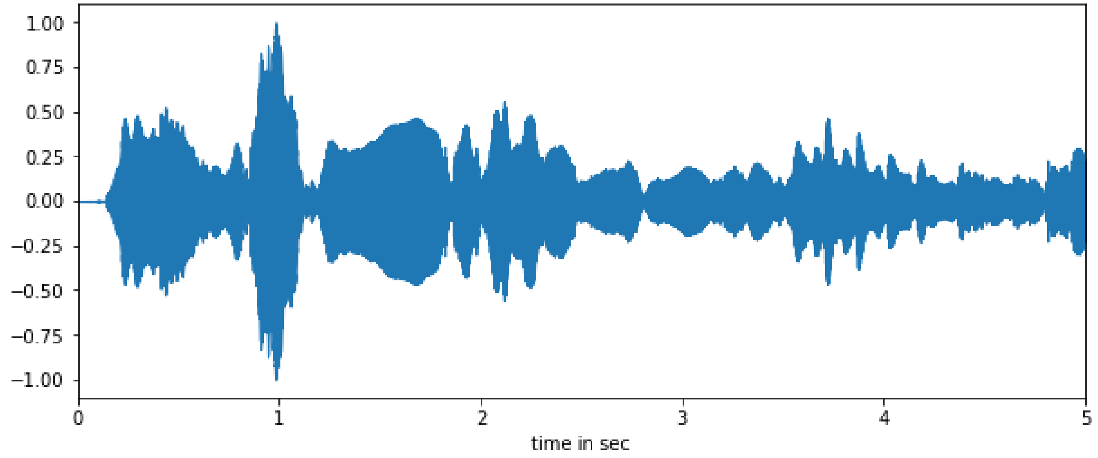




POSTECH

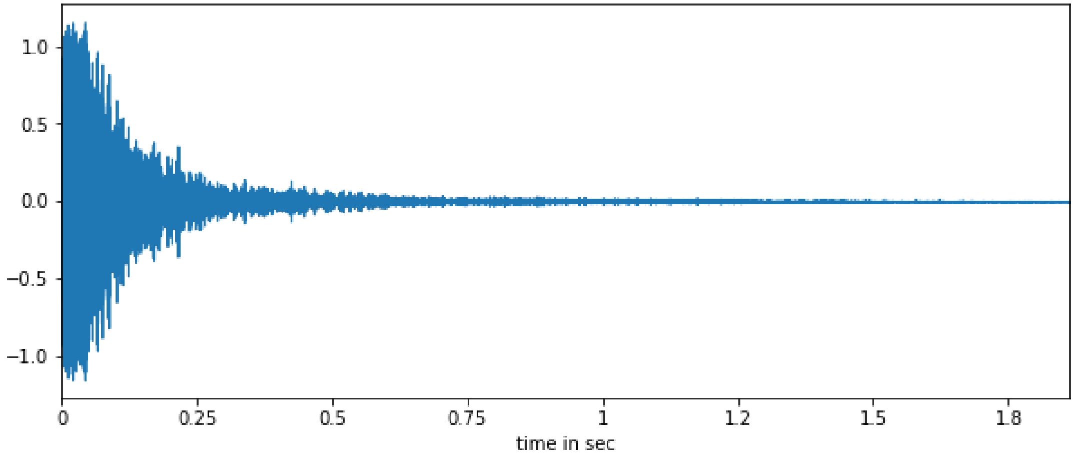
Example: Convolution on Audio

```
x, sr = librosa.load('./data_files/violin_origional.wav')

x = x/max(x)     # normalized

ipd.Audio('./data_files/violin_origional.wav', rate=sr) # play a wave file with sampling rate sr
```



Example: Convolution on Audio

```
# impulse response in a closed room (by gunshot)

h, sr = librosa.load('./data_files/gunshot.wav')

ipd.Audio('./data_files/gunshot.wav', rate=sr) # play a wave file with sampling rate sr
```



Example: Convolution on Audio

```python
y = signal.convolve(x, h)
y = y/max(y)

# plot
plt.figure(figsize=(10,8))
plt.subplot(2,1,1)
librosa.display.waveplot(x, sr=sr)
plt.xlim([0, 6])
plt.title('original')

plt.subplot(2,1,2)
librosa.display.waveplot(y, sr=sr)
plt.xlim([0, 6])
plt.title('convoluted')
plt.show()
```

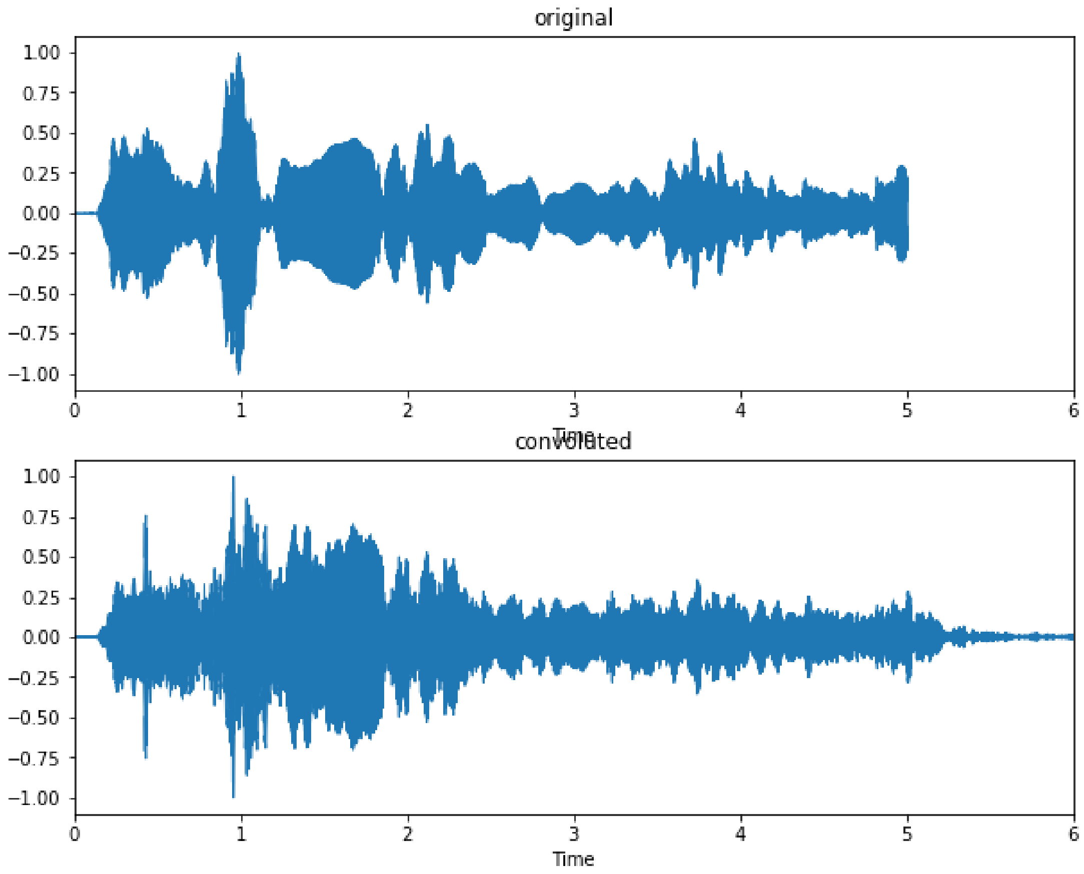
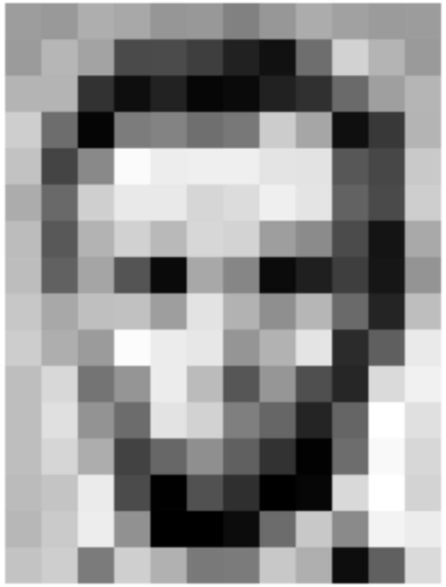


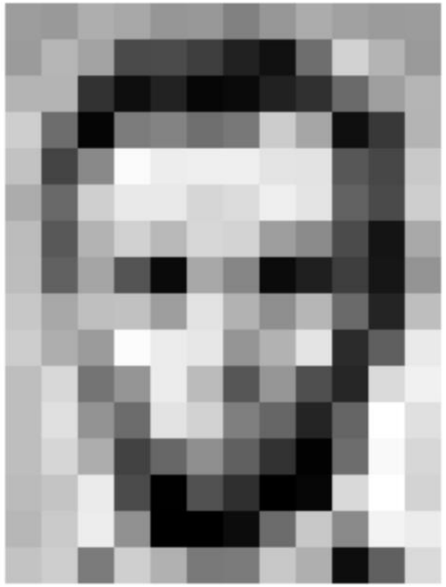
Table of Contents

What Computers “See”

Images Are Numbers

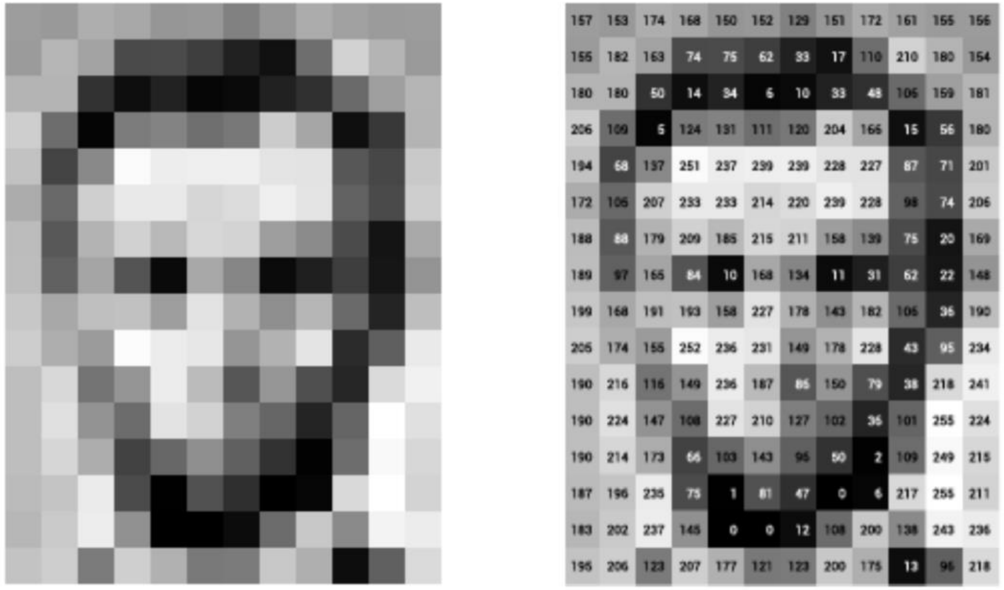


Images Are Numbers



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Images Are Numbers



What the computer sees

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

Images

Original image

R

G

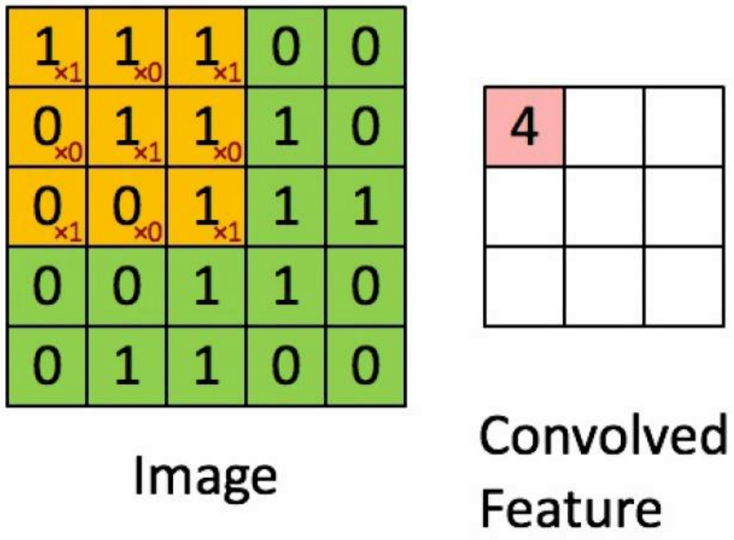
B

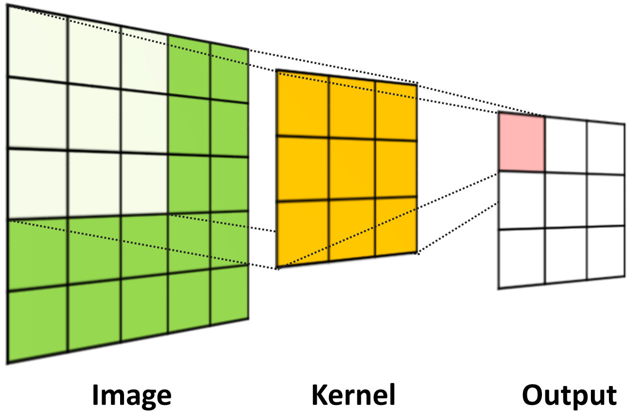
Gray image

Table of Contents

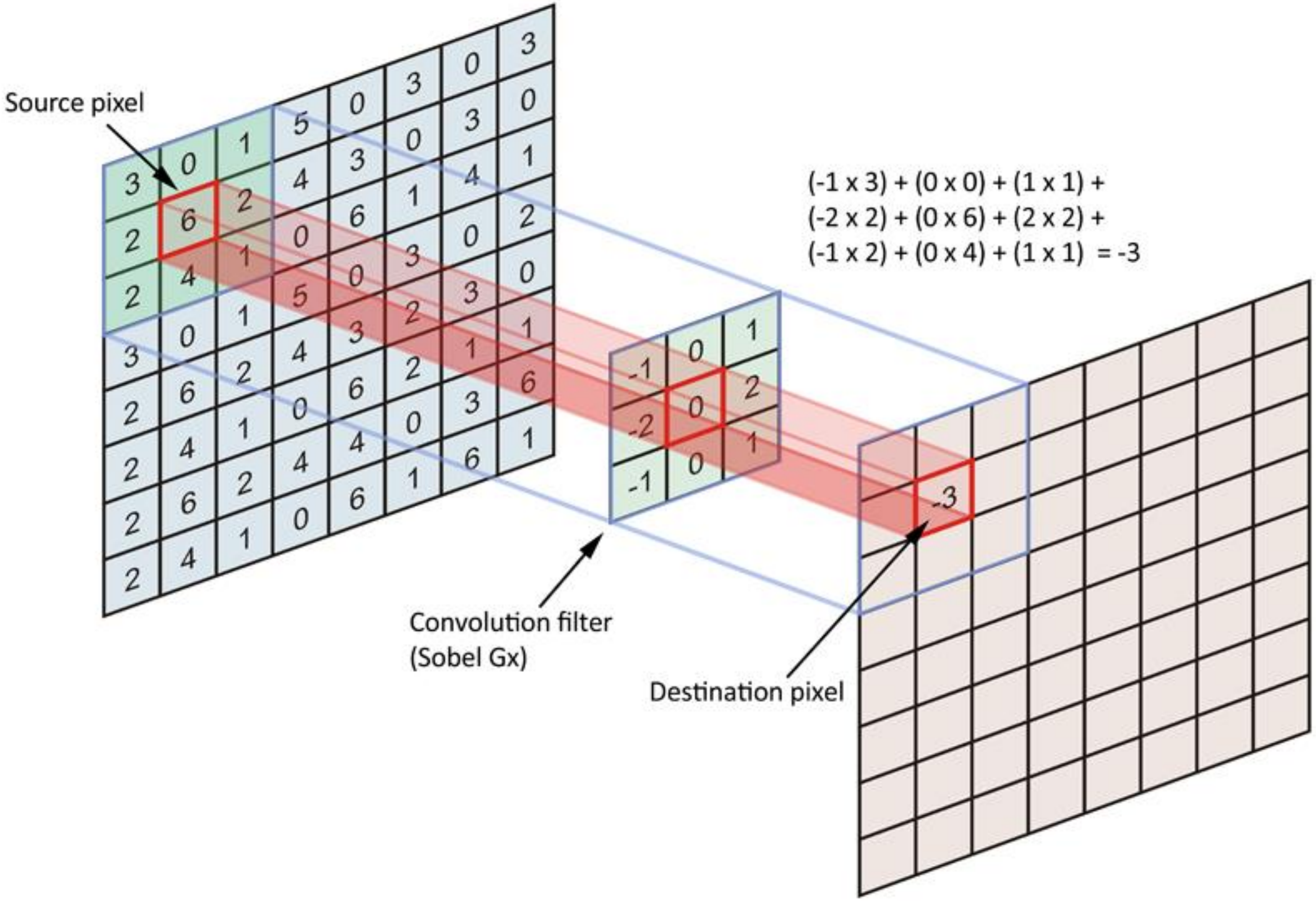
Convolution on Image (= Convolution in 2D)

- Filter (or Kernel)
  - Modify or enhance an image by filtering
  - Filter images to emphasize certain features or remove other features
  - Filtering includes smoothing, sharpening and edge enhancement
  - Discrete convolution can be viewed as <u>**element-wise multiplication**</u> by a matrix





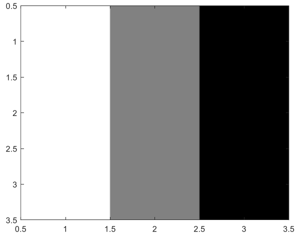
Convolution on Image (= Convolution in 2D)



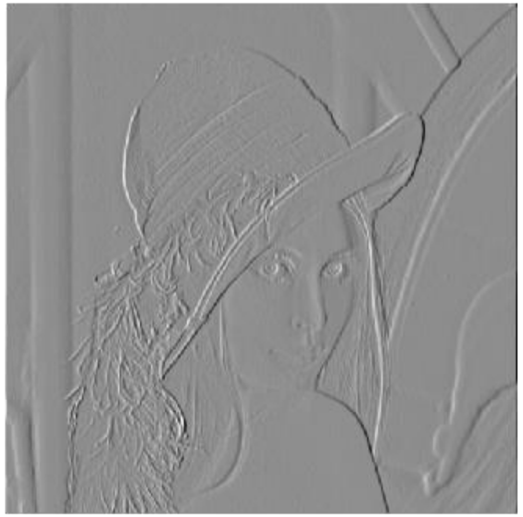
Convolution on Image

Image

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



Kernel

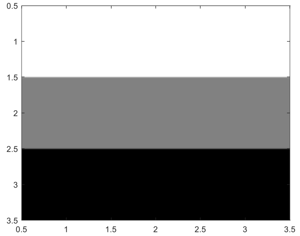


Output

Convolution on Image

Image

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



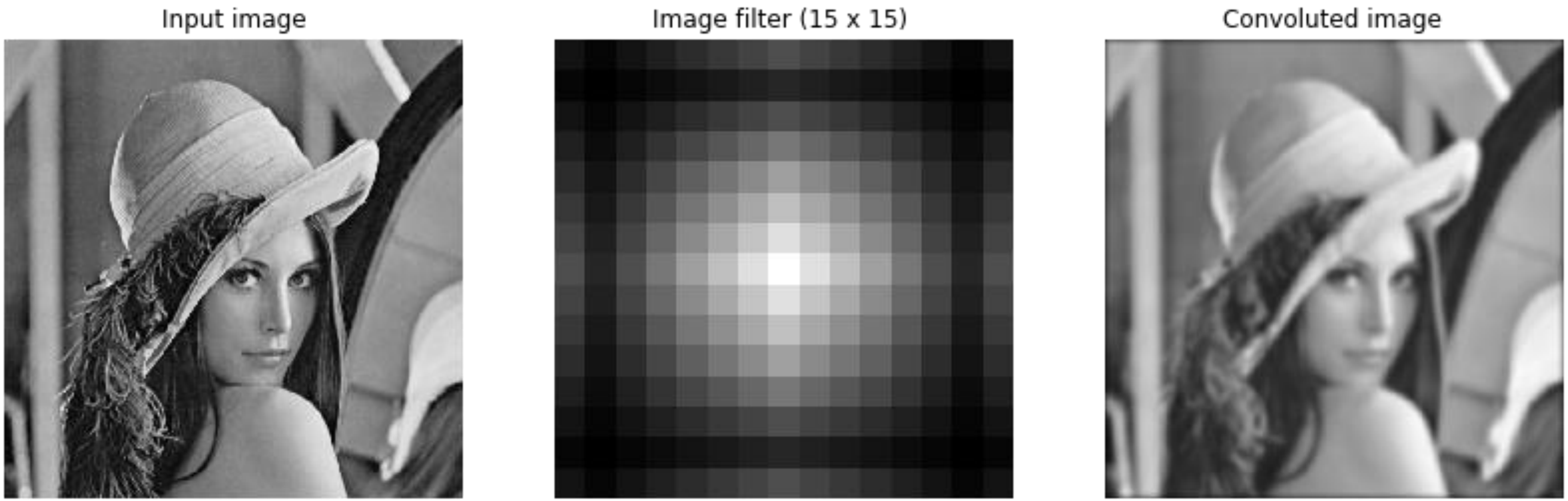
Kernel

Output

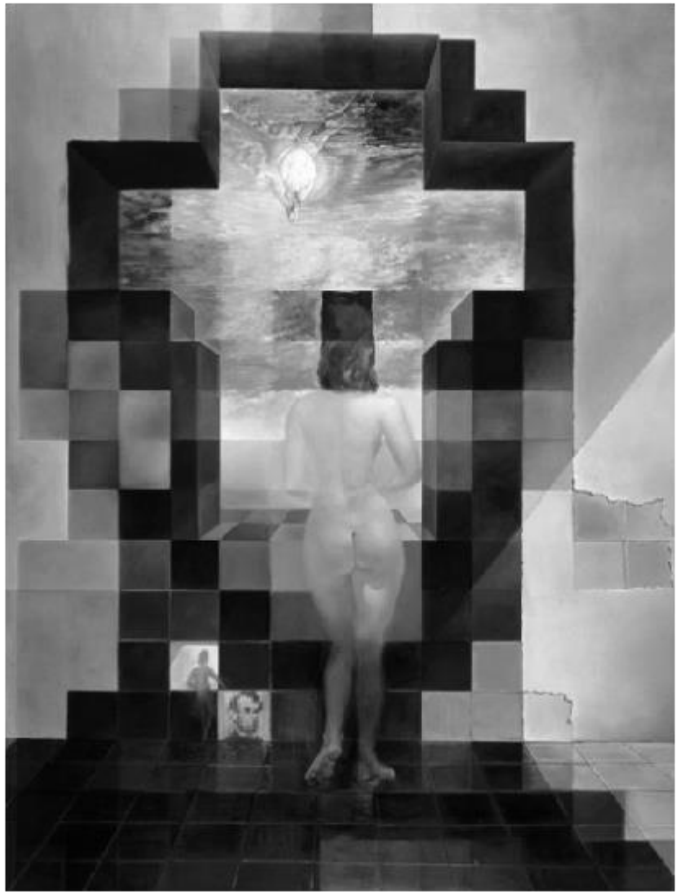
Convolution on Image

```
M = np.ones([3,3])/9

img_conv = signal.convolve2d(img, M, 'same')
```

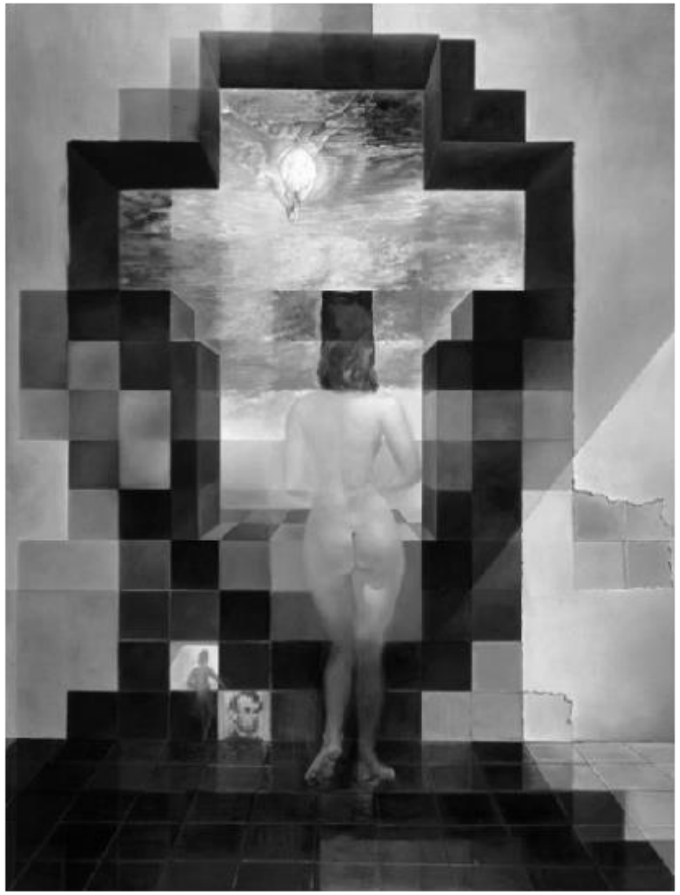
Gaussian Filter: Blurring



Gala Contemplating the Mediterranean Sea



Gala Contemplating the Mediterranean Sea

- Gala Contemplating the Mediterranean Sea Which at Twenty Meters Becomes the Portrait of Abraham Lincoln
- http://thedali.org/exhibit/gala-contemplating-mediterranean-sea/