Convolutional Neural Networks

- Specialized NN meant to process data that has 'grid-like topology'.

  For eg: 1) Time series data

  1D grid topology

  2) Image data

  2D grid of pixels

- Yann LeCun (1989)

  "ConvNets" / CNNs

- Aim: Process unstructured data

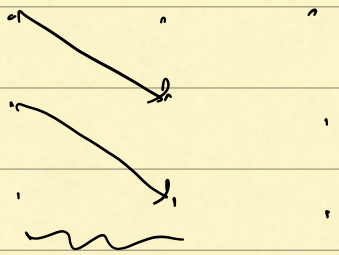
  ↳ doesn't have pre-defined structure

Looking / Finding patterns

e.g.: images, text, audio, speech so on.

* CNN can extract important features out of the data.

* Note: Inspired by MLP.



CNN exploits local spatial correlation.

* Defn: 1) employs mathematical operation "Convolution".

2) Replace matrix - multiplication (that is done in ANN) by

Convolution.

Q. Convolution ?

$A \circledast B = C$

- A → input signal
- $\circledast$ → convolution
- B → another signal

⊛ Convolution of signal A with another signal B gives me a third signal C, which can reveal more about the original signal (A) than the signal itself.

( ) * ( ) = (Modified version)

↓

Better feature representation

For eg:

[Grayscale image] * (Filter / kernel)

2D signal

= Output signal

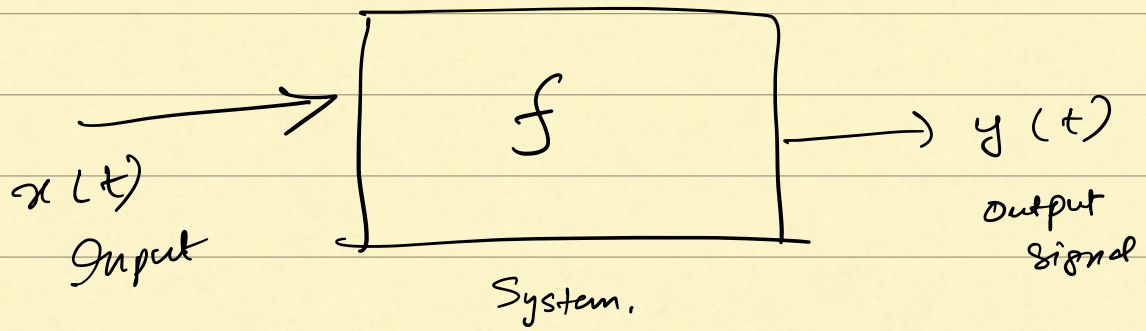
We can choose Kernel in such a way that our output signal contains the edges of the image.

"Edge Detection"

Q. What are edges?

⇒ 1) Object boundaries
2) Illumination changes
3) Depth changes etc.

(*) Linear Time Invariance (LTI) / Linear Shift Invariance (LSI) Systems? -



$$y(t) = f(x(t))$$

(*) When do we call these systems LTI / LSI ?

— Linear ✓

① Scaling : $f(\alpha \cdot x(t)) = \alpha \cdot f(x(t))$

② Superposition : $f(\alpha x_1(t) + \beta x_2(t))$

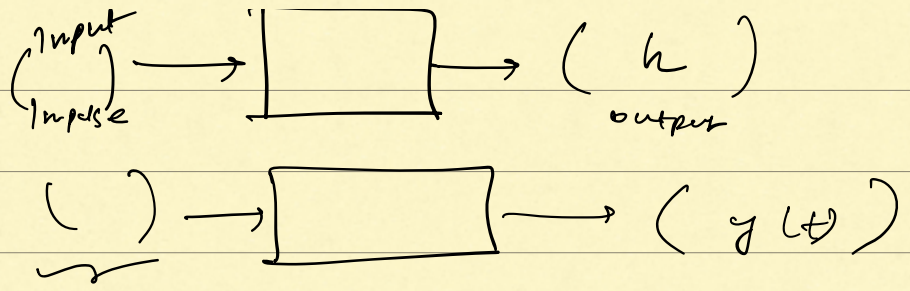
$= \alpha \cdot f(x_1(t)) + \beta \cdot f(x_2(t))$

— Time invariant / Shift invariant ✓

$f(x(t-\tau)) = y(t-\tau)$

"LSI / LTI systems."

Thm: If we know the output of the system to an 'impulse response' then one can compute the output response to any signal.

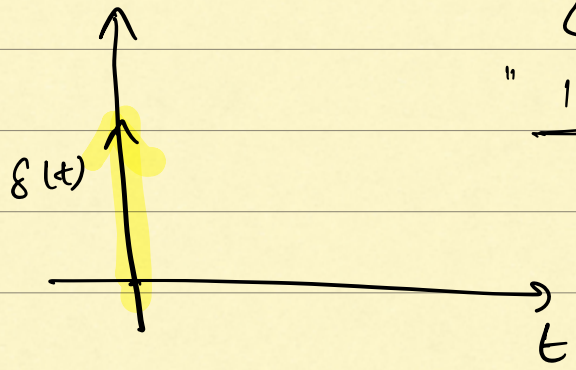


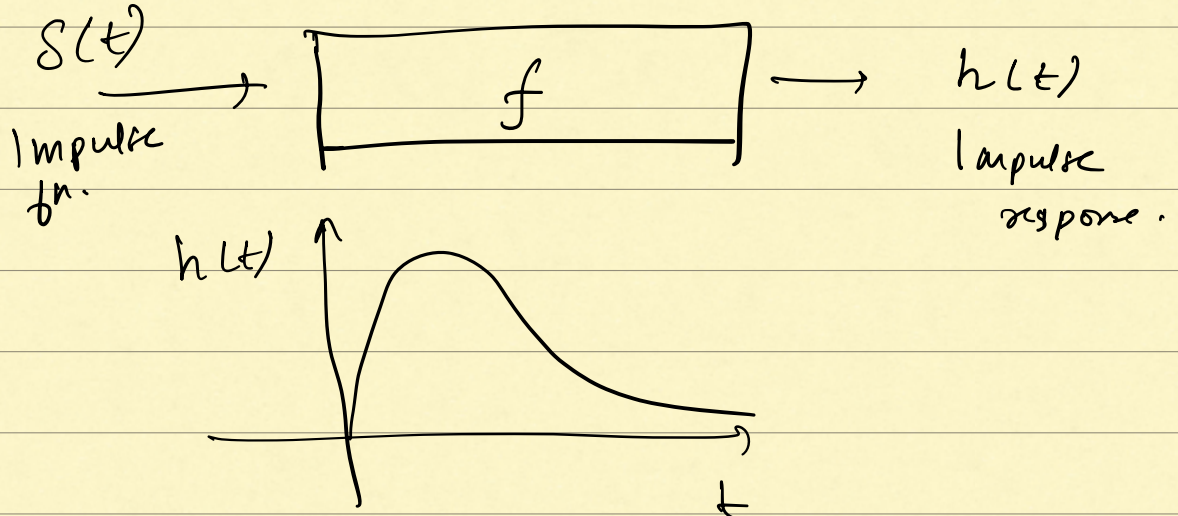
For eg: ① "Direc-Delta" function

Electrodynamics PHY 103

$$\delta(t) = \begin{cases} +\infty & t=0 \\ 0 & t\neq 0 \end{cases}$$

"Impulse"

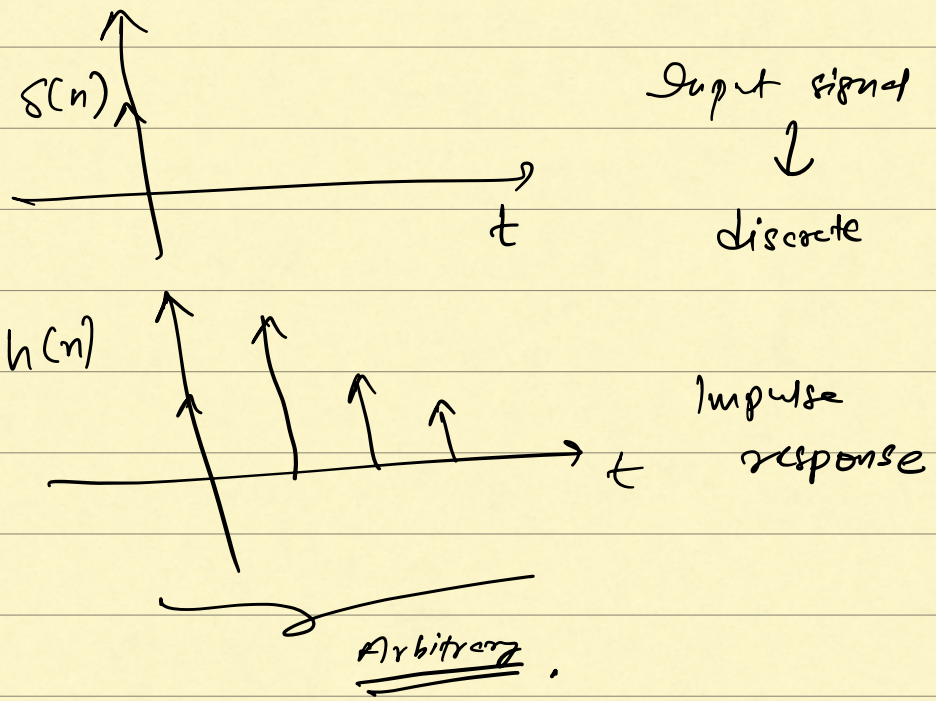




② Unit Impulse Function

$$\delta(n) = \begin{cases} 1 & t=0 \\ 0 & t \neq 0 \end{cases}$$

$\delta(n) \longrightarrow$ [T] $\longrightarrow h(n)$



$n = (0, 1, 2, 3 \ldots)$

Q : ~~Find~~ ?

$\delta \longrightarrow$ [ ] $\longrightarrow h$

$x \longrightarrow$ [ ] $\longrightarrow y$

$$y(t) = x(t) \circledast h(t)$$

↓

??

(*) Convolution for signals 1D :-

— Measure of overlap b/w 2 fns.

— x , h

$h(\tau) \Rightarrow h(-\tau) \Rightarrow h(t-\tau)$ ( ) reversed + translated

Mathematically :-

Contin

$$y(t) = \int_{\tau=-\infty}^{+\infty} x(\tau) \cdot h(t-\tau) \, d\tau$$

Discrete

$$y(t) = \sum_{\tau=-\infty}^{+\infty} x(\tau) \cdot h(t-\tau)$$

(*) Signals :- Any quantity that varies with time and/or space.

For eg: Stock market prices of a Sp. stock over a period of a week

(11) "n is a signal.

Signals

Analog

Digital

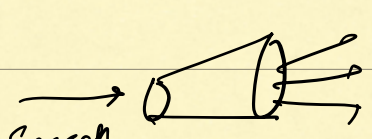
(*) Note: → Digital computers.

(Digital) signals

Analog continuous signal

↓

Digital Discrete signal.



(Analog signal)

↓

(Digital signal)

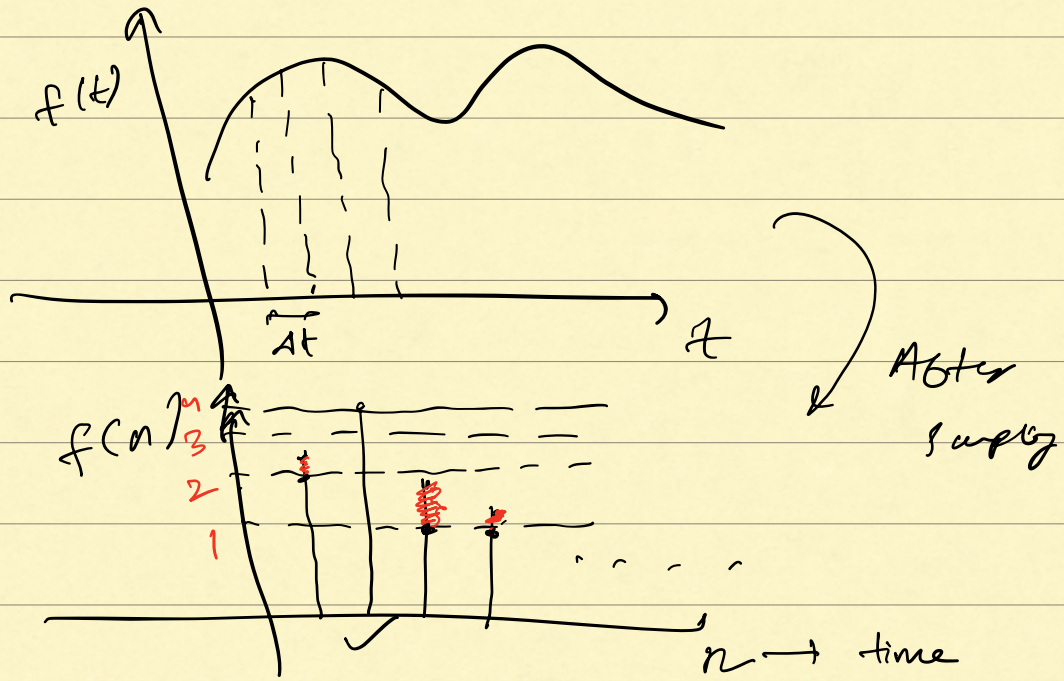
How?

(1) Sampling: (Taking signal at fixed time)

(n = 0, 1, 2 ...)

spatial intervals

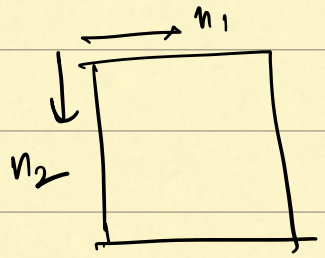
(2) **Quantization** : Quantize / having sp. values for ~~f(t)~~ our response.



Output : "Digital signal"

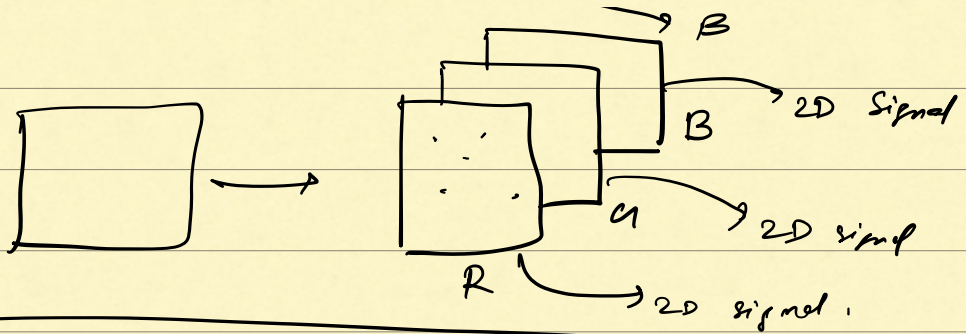
Q. Image → Digital signal

(2D)



Grayscale image (0-255)

RGB image → 3 channels → R

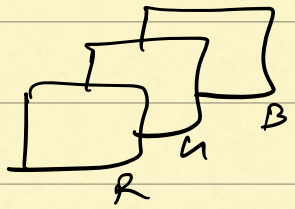


⊛ 2D / 3D Signals :-

Grayscale $N \times M$ $(1 \ldots N)$ $(1 \ldots M)$

$x(n_1, n_2)$ = Signal

↳ scalar $\begin{cases} 0 \leq n_1 \leq M-1 \\ 0 \leq n_2 \leq N-1 \end{cases}$



Colored image

$N \times M \times 3$

$\vec{x}(n_1, n_2) = \vec{x}_R(n_1, n_2) +$

$\vec{x}_G(n_1, n_2) +$

$\vec{x}_B(n_1, n_2)$

$$\begin{bmatrix} x_R(n_1, n_2) & \cdots & \end{bmatrix}^T$$

$N \times M$

$x(n_1, n_2)$

$$0 \le n_1 \le N-1$$
$$0 \le n_2 \le M-1$$

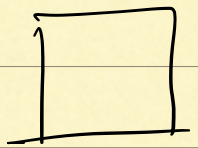
$$0 \le n_1 \le M-1$$
$$0 \le n_2 \le N-1$$

n_1 $M-1$

$N-1$

n_2

(*) 2D Convolution :- $A * B = C$



Input image

$x(n_1, n_2)$

Digital Signal

(*) $\delta(n) = \begin{cases} 1 & n=0 \end{cases}$

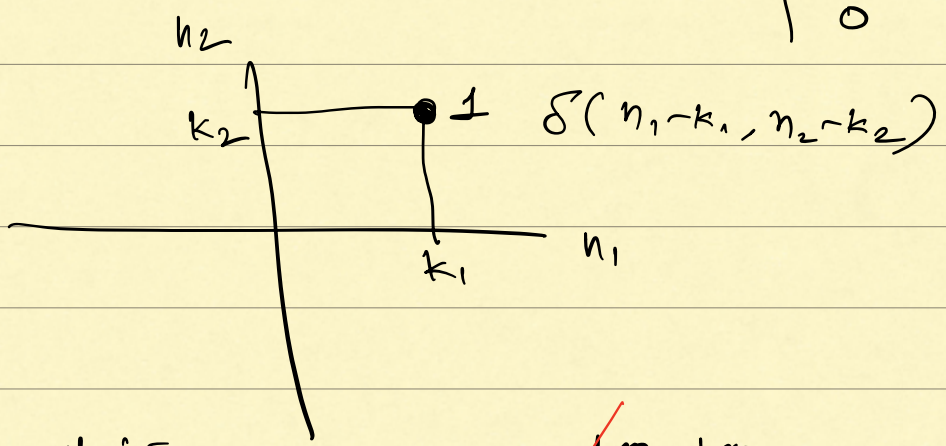
1 0 $n \neq 0$

2D Unit Step Function

$$\delta(n_1, n_2) = \begin{cases} 1 & n_1 = n_2 = 0 \\ 0 & \text{otherwise} \end{cases}$$

on shifting :-

$$\delta(n_1 - k_1, n_2 - k_2) = \begin{cases} 1 & n_1 = k_1 \text{ and } n_2 = k_2 \\ 0 & \text{otherwise} \end{cases}$$



Recall :-

$$f(x(n_1, n_2)) = f\left(\sum_{k_2=-\infty}^{+\infty} \sum_{k_1=-\infty}^{+\infty} x(k_1, k_2) \cdot \delta(n_1 - k_1, n_2 - k_2)\right)$$

(*) LSI System Unit Step Response

$A * B = C$

$x(n_1, n_2) \longrightarrow \boxed{f} \longrightarrow y(n_1, n_2)$

LSI system

sys..

$$y(n_1, n_2) = f(x(n_1, n_2))$$

$$= \sum_{k_2=-\infty}^{+\infty} \sum_{k_1=-\infty}^{+\infty} x(k_1, k_2) \cdot f(\delta(n_1 - k_1, n_2 - k_2))$$

$$\delta(n) \longrightarrow \boxed{f} \longrightarrow h(n)$$

1D

$$h(n) = f(\delta(n))$$

2D

$$h(n_1 - k_1, n_2 - k_2) = f(\delta(n_1 - k_1, n_2 - k_2))$$

Hence,

$$f(x(n_1, n_2)) = \sum \sum x(k_1, k_2) \cdot h(n_1 - k_1, n_2 - k_2)$$

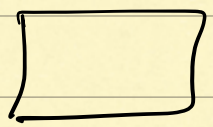
(*) $A * B = C$

B → LSI System Unit Step

Response

Filter / Kernel

(Input image) * Filter/Kernel



→ Output signal

2D discrete signal N×M

(A) $x(n_1, n_2)$

(B) $h(n_1, n_2)$: Image processing filter

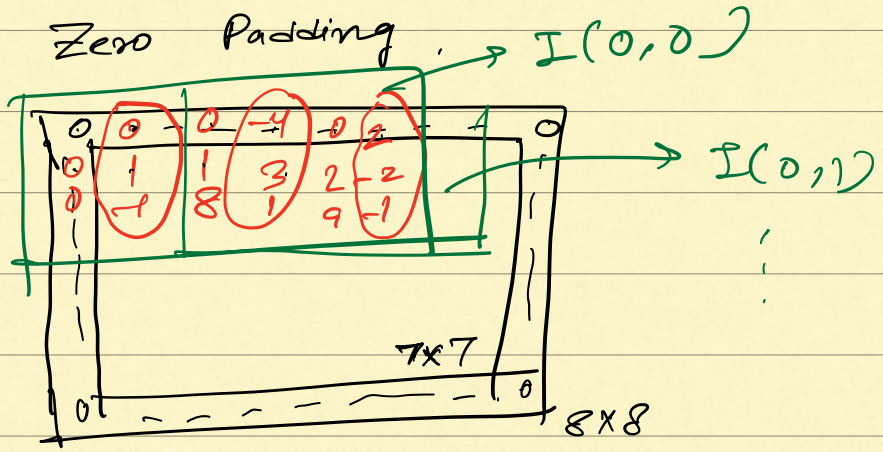
$$y(n_1, n_2) = \sum_{k_2=0}^{N-1} \sum_{k_1=0}^{M-1} x(k_1, k_2) \cdot h(n_1 - k_1, n_2 - k_2)$$

For eg :-

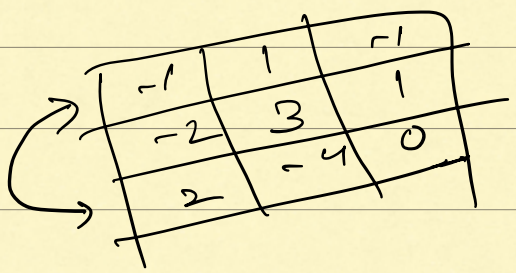
Input Image

	1	2	3 - - -	7
1				
1				
1				

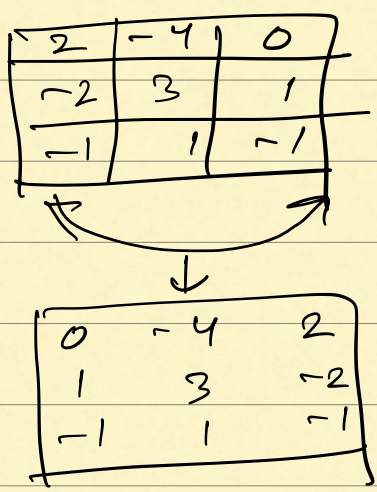
Step 1: Zero Padding



Step 2: Filter / kernel



Step 3: Flip the kernel



"kernel Flipping"

(*) We need to calculate I(i,j)

I(i,j)

$$I(0,0) = 1*3 + (-4) + (-1)$$

$$= -2$$

$I(0,1)$

$I(i,j)$

Common Image Processing Filters :-

$A * B \rightarrow C$

LSI System with Shift Invariant

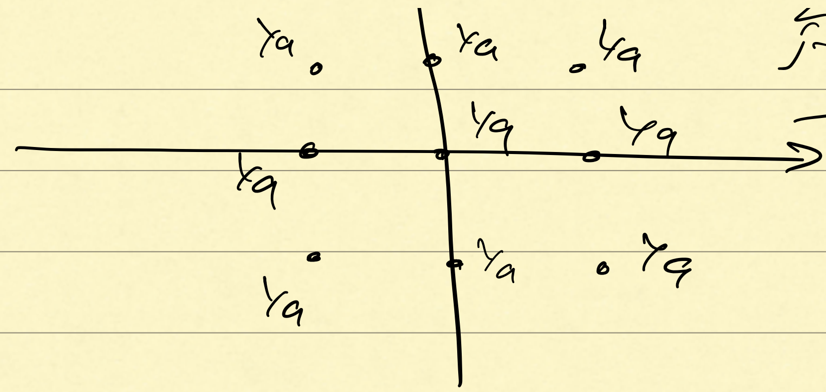
Filter $(1,9,)$

(1) Mean filter :-

Low pass filter

$$\begin{pmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix}$$

$$\frac{3}{5}\sum^{3} I_{i,}$$



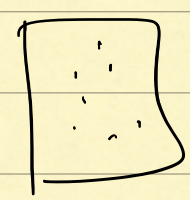
$$\frac{\sum_{j=-1}^{1} \sum_{i=-1}^{1} (\cdot)}{9} = \underline{\text{Average}}$$

Removes "Gaussian noise"

② <u>Median filter</u> :-

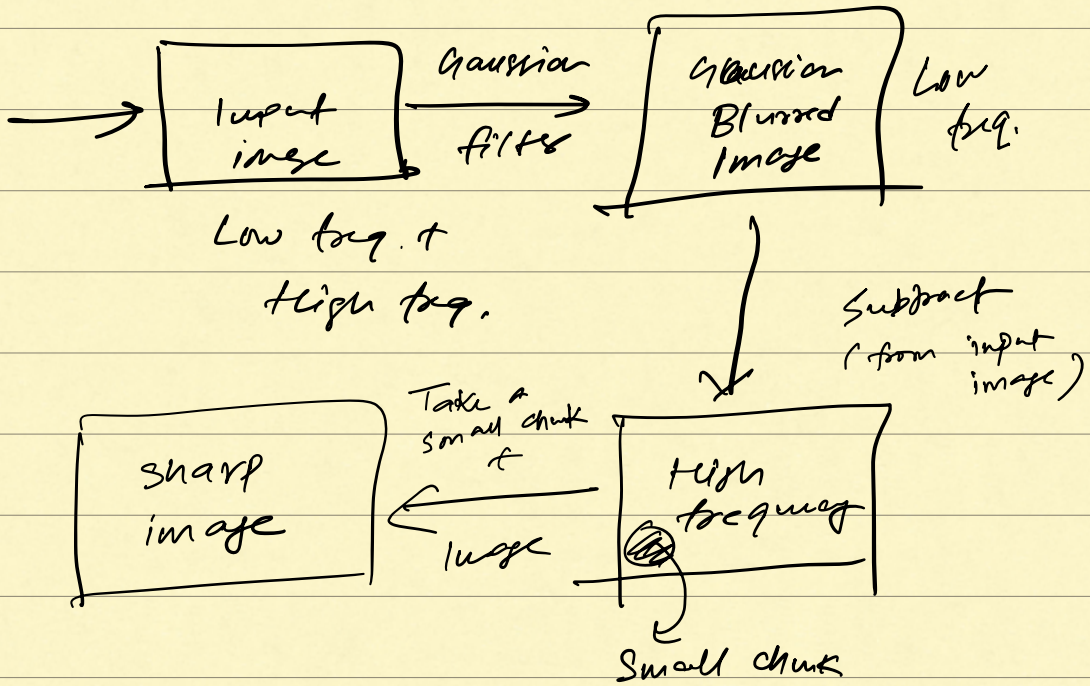
Replace each nbd pixel with median pixel intensity in that nbd.

Removing "Salt (White) and Pepper (Black)" noise



③ <u>Gaussian Filter</u> :-

Modified version of M.F.



- Also Useful in 'sharpening' of images.

⑧ Gradient Based Filter :-

$$\vec{\nabla} I(x,y) = \frac{\partial I}{\partial x}\hat{i} + \frac{\partial I}{\partial y}\hat{j}$$

MTH 111

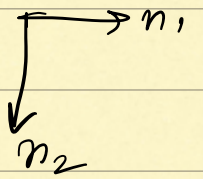
$$\frac{\partial I(x,y)}{\partial x} = \lim_{h \to 0} \frac{I(x+h, y) - I(x,y)}{h}$$

$$= \lim_{h \to 0} \frac{I(x+h, y) - I(x-h, y)}{2h}$$

Continuous

But here, we have discrete signals.

We take $h = 1$



$$\frac{\partial I}{\partial x} = I(x+1, y) - I(x, y)$$

Filter kernel:

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

$$\propto I(x+1, y) - I(x-1, y)$$

Filter:

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

$\frac{\partial I}{\partial y}$ ①:

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

②:

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

(*) Sobel Edge Detection Filter:-

— extensions of Gradient Filters

S.E.D. → H_x, H_y — impulse responses

$$H_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

$$H_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

— Sobel E.D. → high pass filter

$$\vec{I}'(x,y) = \vec{I}_x(x,y) + \vec{I}_y(x,y)$$

$$I'(x,y) = \begin{bmatrix} I_x(x,y) & I_y(x,y) \end{bmatrix}^T$$

$$|I'(x,y)| = \sqrt{(I_x(x,y))^2 + (I_y(x,y))^2}$$

$$= c(x,y)$$

↳ pixel intensity for Sobel

low... filter

Summary:

	Filter	Use
(1)	Mean filter	Reduce Gaussian noise (or Salt & Pepper Noise
(2)	Median filter	
(3)	Gaussian filter	(i) Blur (ii) Sharpen
(4)	Gradient Based Filters.	Detect edges (poor)
(5)	Sobel Filter	Detect edges (good).

EE604 — Prof. Tushar Sandhan.

( )

Deep Learning — Ian Goodfellow.

OpenCV : ( )