삼성 DS^{2}과정 프로젝트

Image Restoration Challenge

Jongho Lee, Ph.D

Laboratory for Imaging Science and Technology
Department of Electrical and Computer Engineering
Seoul National University

SEOUL
NATIONAL
UNIVERSITY

LABORATORY for
IMAGING
SCIENCE &
TECHNOLOGY

~~제5회~~ 제1회

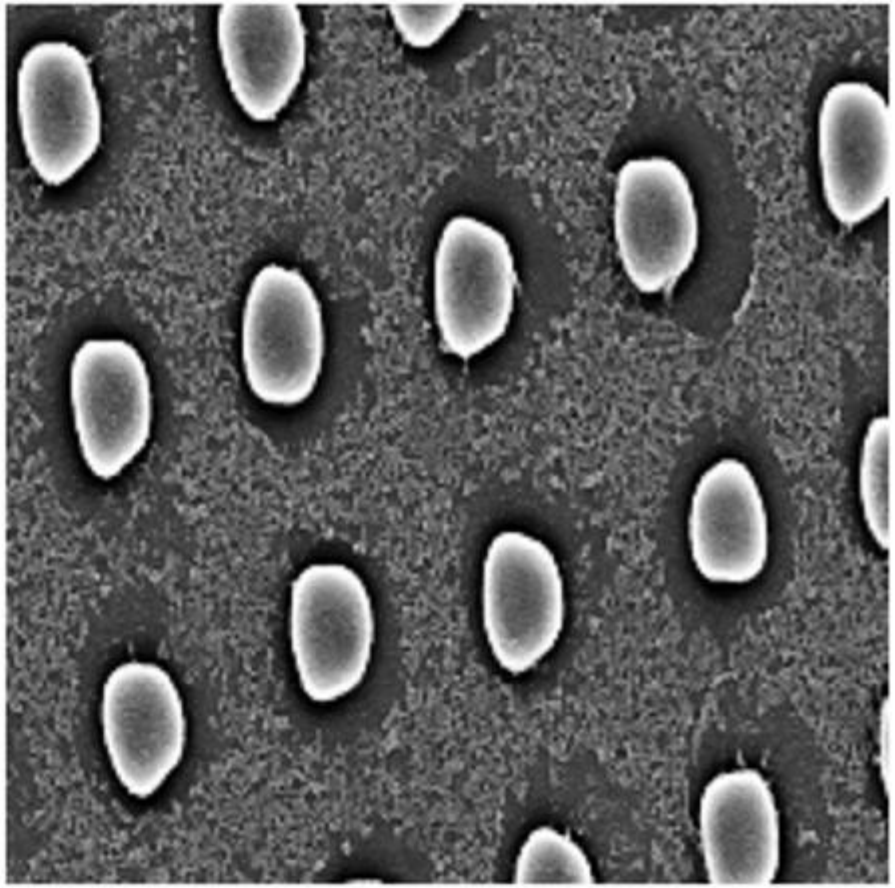
2025 ~~SNU FastMRI Challenge~~ 삼성DS²과정 Digital Image Processing Challenge

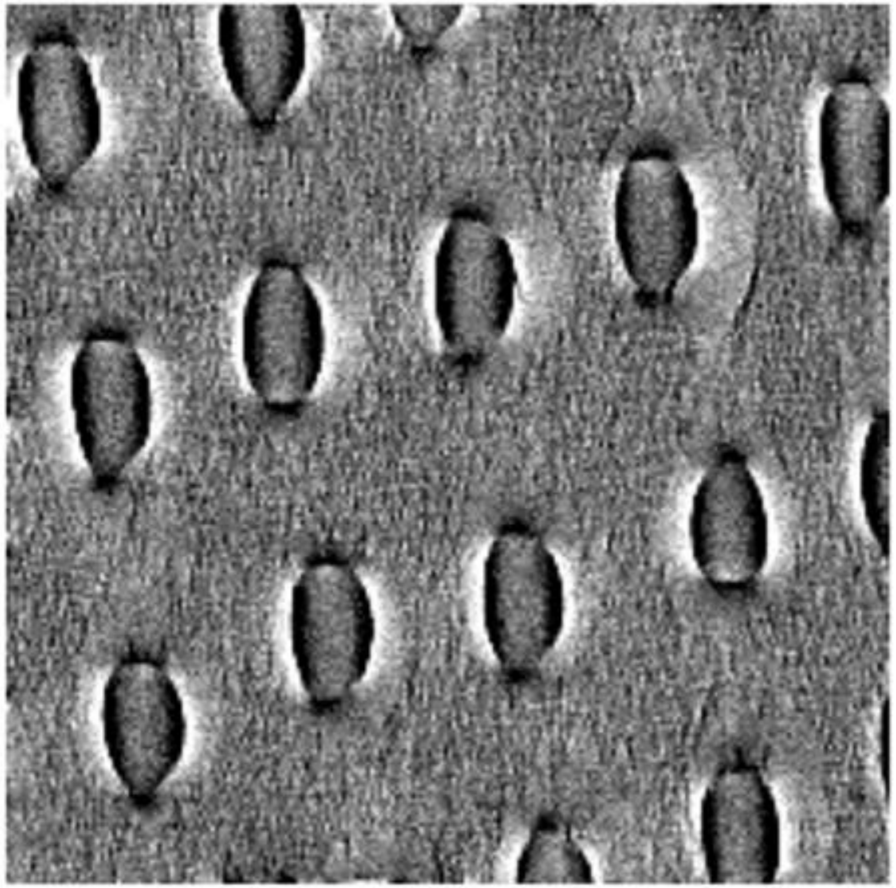
~~FastMRI Challenge~~

Semiconductor Image Processing Challenge

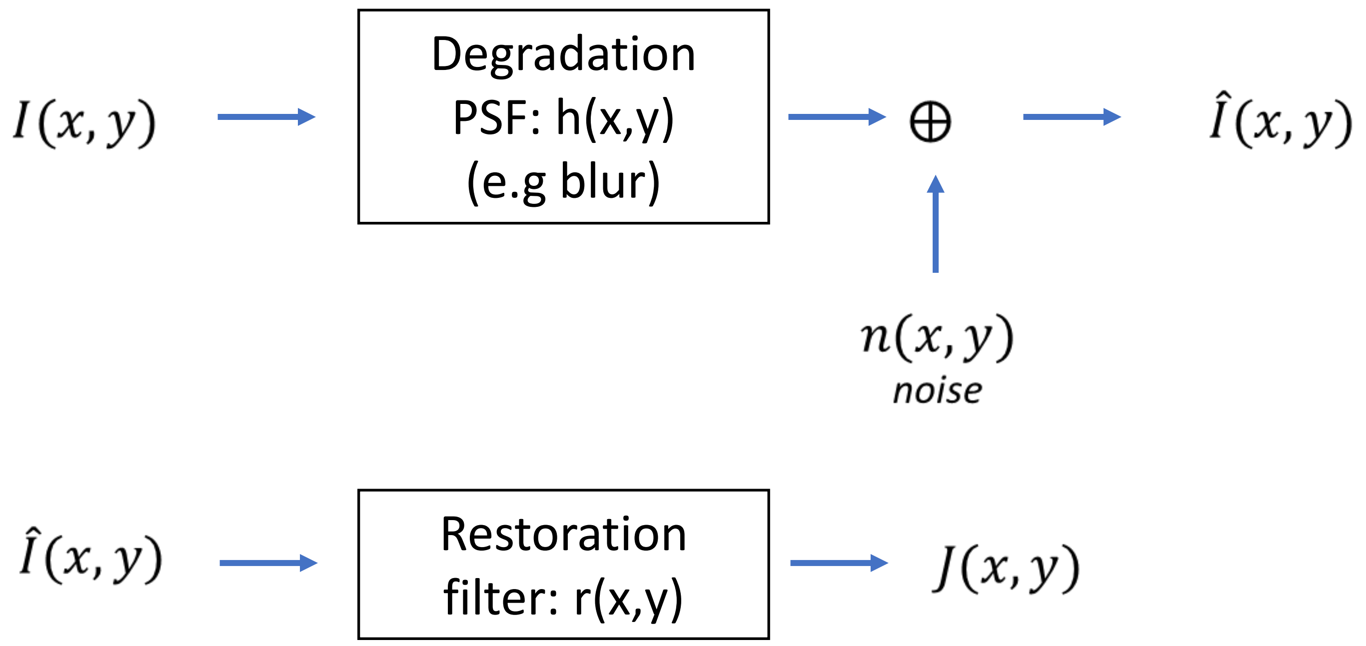
참가대상	~~서울대 학부생(1팀 최대 2명)~~ 삼성DS²과정생	상 금	~~총 2,000만원(1등 1,000만원)~~ To be announced
대회기간	~~2025.07.01(화) - 2025.08.20(수)~~	괜찮아요!	Deep Learning 몰라요? MRI 몰라요? → Tutorial + Q&A 있습니다~
참가방법	온라인접수 (fastmri.snu.ac.kr)		
문의방법	이메일문의 (fastmri.snu@gmail.com)		

Aim of our challenge





Basic model for degradation



$$\hat{I}(x,y) = I(x,y) * h(x,y) + n(x,y) \qquad J(x,y) = \hat{I}(x,y) * r(x,y) + n(x,y) * r(x,y)$$

$$\hat{\boldsymbol{I}}(u,v) = \boldsymbol{I}(u,v)\boldsymbol{H}(u,v) + \boldsymbol{N}(u,v) \qquad \boldsymbol{J}(u,v) = \hat{\boldsymbol{I}}(u,v)\boldsymbol{R}(u,v) + \boldsymbol{N}(u,v)\boldsymbol{R}(u,v)$$

What happens when we also have degradation?

- $\hat{I}(x,y) = I(x,y) * h(x,y) + n(x,y)$
- $\hat{\boldsymbol{I}}(u,v) = \boldsymbol{I}(u,v)\,\underbrace{\boldsymbol{H}(u,v)}_{\text{blur}} + \boldsymbol{N}(u,v)$

- How do we estimate $h(x,y)$?

What happens when we also have degradation?

- Guess: Take a piece of the degraded image and guess what the original image should have looked like

$$\hat{I}_S(u,v) \quad vs. \quad \hat{G}_S(u,v)$$

S = guess area   Manual guess

$$H(u,v) = \frac{\hat{I}_S(u,v)}{\hat{G}_S(u,v)}$$

- Experiment if you have access to the imaging device: Directly acquire the impulse response / point spread function

- Estimate $h(x,y)$ (e.g. Gaussian blur)

Inverse filtering

- We have the degraded image I(x,y)
- We have the estimated blur $h(x, y)$

- Inverse filtering

$$J(u, v) = \frac{\hat{I}(u, v)}{H(u, v)}$$

- Usually is bad. Why?

$$J(u, v) = \frac{I(u, v)H(u, v) + N(u, v)}{H(u, v)} = I(u, v) + \frac{N(u, v)}{H(u, v)}$$

※ If H(u,v) is very small for some (u,v) then $\frac{N(u,v)}{H(u,v)}$ is very large
⇒ poor reconstructions.

Wiener filter

- Wiener filter: minimum mean-square error filtering

$$e^2 = E\left\{\left(I(x,y) - \hat{I}(x,y)\right)^2\right\} \qquad S_F = |I(u,v)|^2 \;, S_n = |N(u,v)|^2$$

$$J(u,v) = \left[\frac{H^*(u,v)\,S_F(u.v)}{S_F(u,v)|H(u,v)|^2 + S_n(u,v)}\right]\hat{I}(u,v)$$

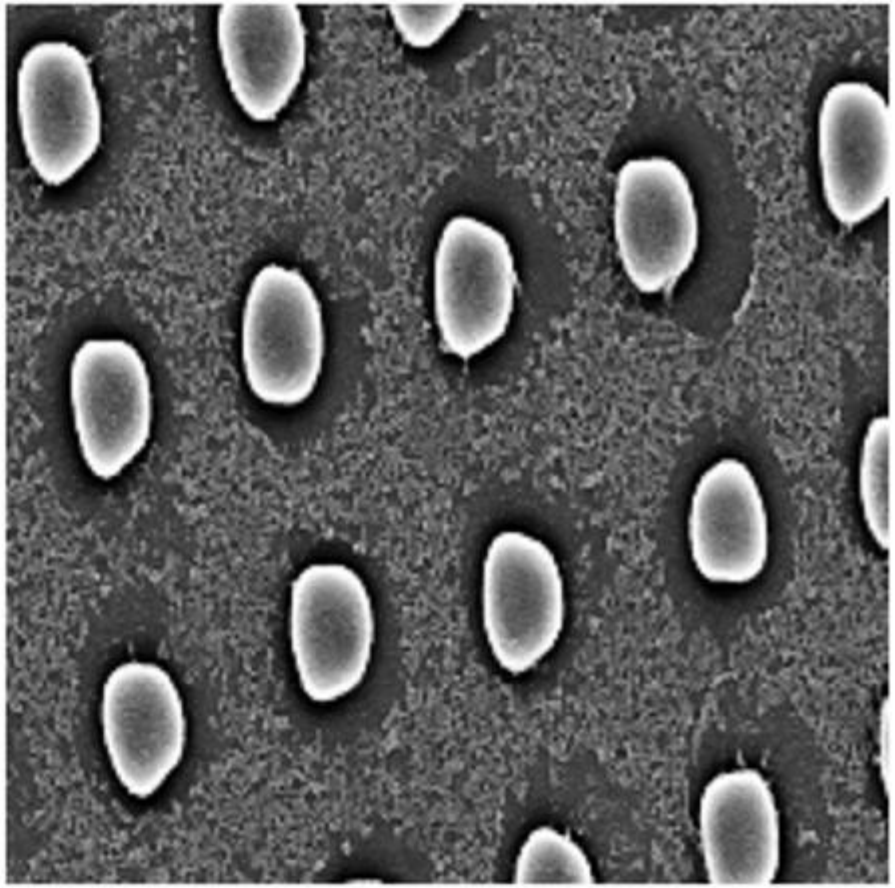
$$= \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + \dfrac{S_n(u,v)}{S_F(u,v)}}\right]\hat{I}(u,v)$$

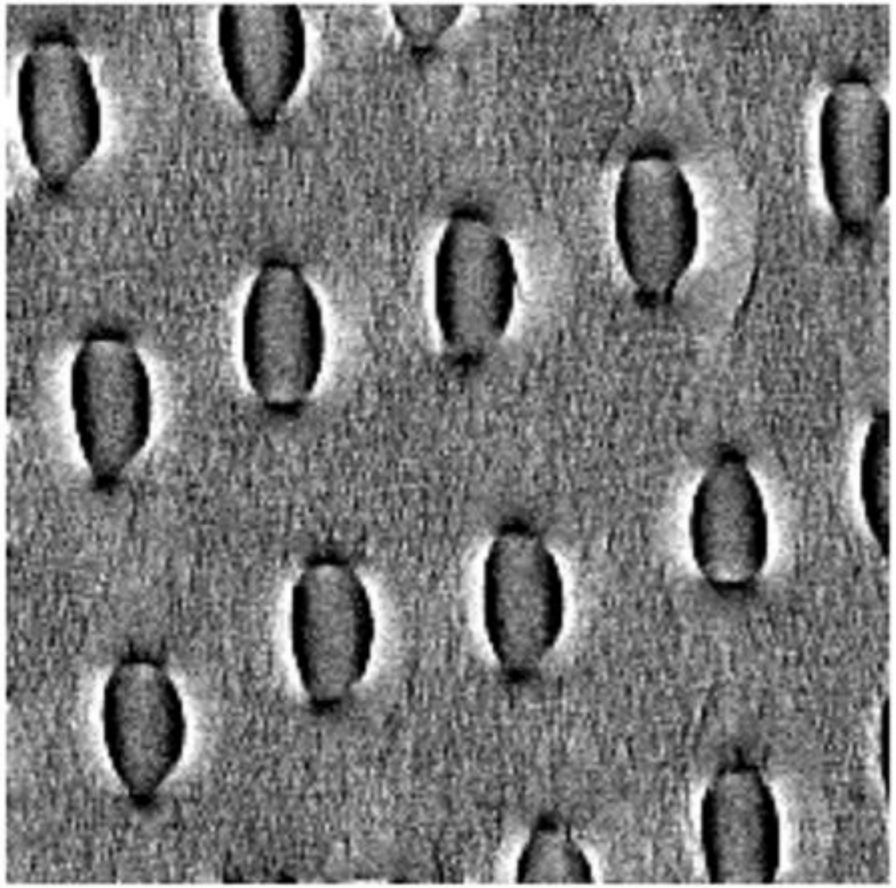
- If we don't know $S_F(u,v)$ (requiring the original image), we use:

$$\bar{\bar{I}}(u.v) = \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right]\hat{I}(u,v)$$

Tunable parameter

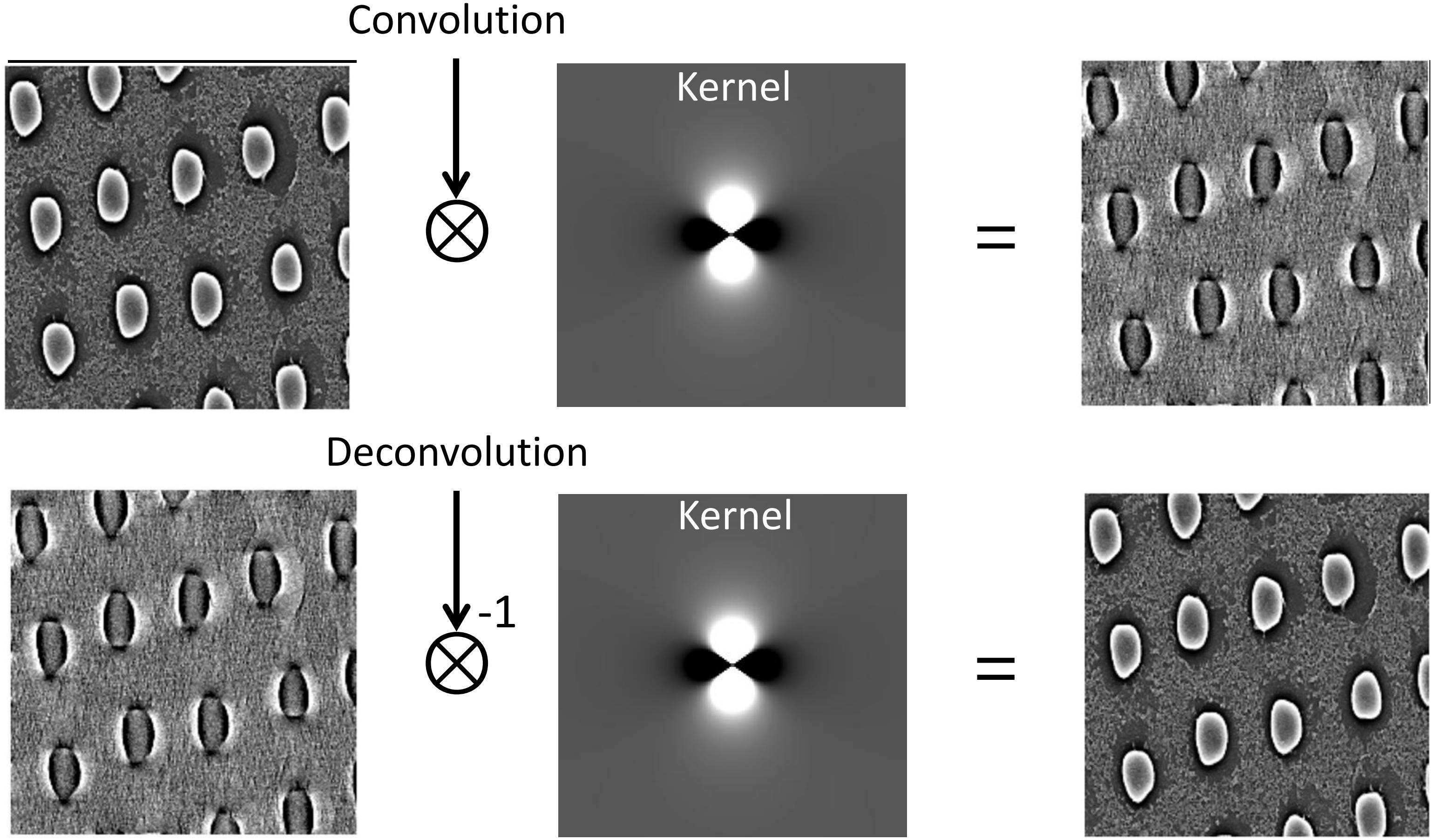
Aim of our challenge





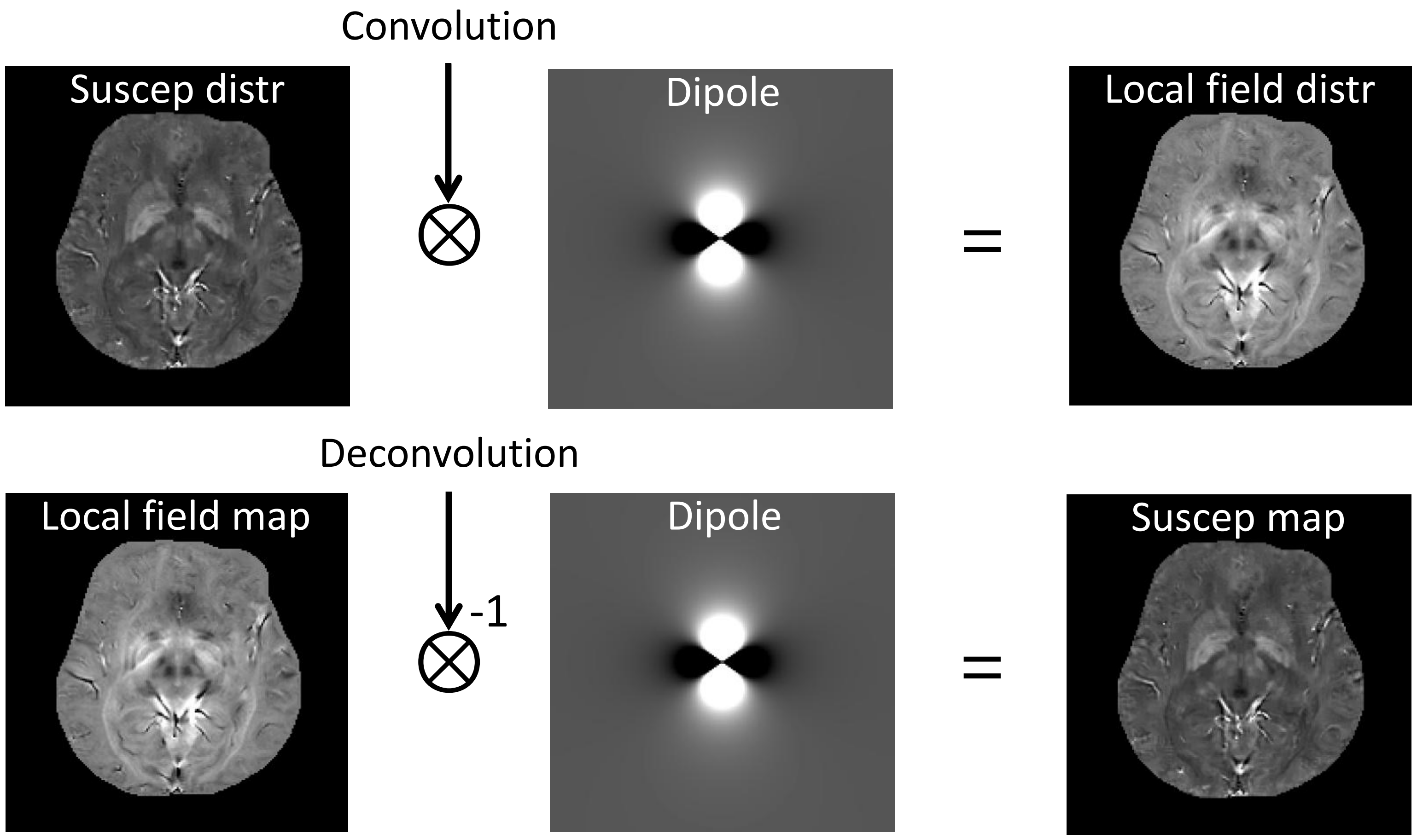
Our convolution model (PSF/IRF/Kernel)

$$\chi * \mathrm{d} = \mathrm{measurement}$$



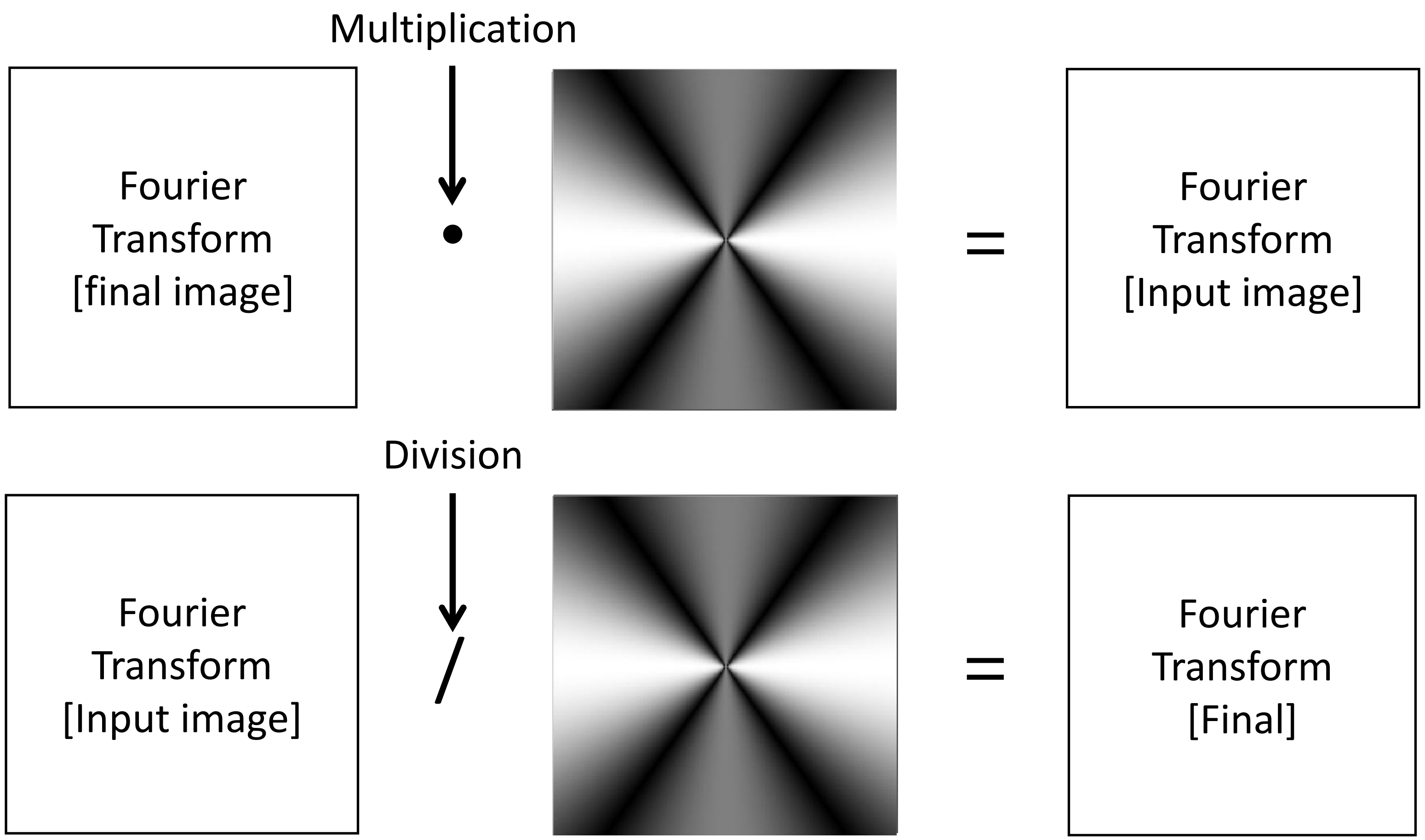
- Seminconductor image is generated from input image by deconvolution

Our convolution model (PSF/IRF/Kernel)



- Fourier transform suggests deconvolution requires division by zeros

Our convolution model (PSF/IRF/Kernel)



- Fourier transform suggests deconvolution requires division by zeros

Estimation or inversion

$$y = Ax$$

- y_i is ith measurement or sensor reading (which we know)
- x_j is jth parameter to be estimated or determined
- a_{ij} is sensitivity of ith sensor to jth parameter

sample problems:

- find x, given y
- find all x's that result in y (*i.e.*, all x's consistent with measurements)
- if there is no x such that $y = Ax$, find x s.t. $y \approx Ax$ (*i.e.*, if the sensor readings are inconsistent, find x which is almost consistent)

Overdetermined linear equations

consider $y = Ax$ where $A \in \mathbf{R}^{m \times n}$ is (strictly) skinny, $i.e.$, $m > n$

- called *overdetermined* set of linear equations (more equations than unknowns)
- for most y, cannot solve for x

one approach to *approximately* solve $y = Ax$:

- define *residual* or error $r = Ax - y$
- find $x = x_{\rm ls}$ that minimizes $\|r\|$

$x_{\rm ls}$ called *least-squares* (approximate) solution of $y = Ax$

Multi-objective least-squares

in many problems we have two (or more) objectives

- we want $J_1 = \|Ax - y\|^2$ small
- and also $J_2 = \|Fx - g\|^2$ small

($x \in \mathbf{R}^n$ is the variable)

- usually the objectives are *competing*
- we can make one smaller, at the expense of making the other larger

common example: $F = I$, $g = 0$; we want $\|Ax - y\|$ small, with small x

Underdetermined linear equations

we consider

$$y = Ax$$

where $A \in \mathbf{R}^{m \times n}$ is fat ($m < n$), *i.e.*,

- there are more variables than equations
- x is *underspecified*, *i.e.*, many choices of x lead to the same y

we'll assume that A is full rank (m), so for each $y \in \mathbf{R}^m$, there is a solution

set of all solutions has form

$$\{ x \mid Ax = y \} = \{ x_p + z \mid z \in \mathcal{N}(A) \}$$

where x_p is any ('particular') solution, *i.e.*, $Ax_p = y$

Least-norm solution

one particular solution is

$$x_{\rm ln} = A^T (AA^T)^{-1} y$$

(AA^T is invertible since A full rank)

in fact, $x_{\rm ln}$ is the solution of $y = Ax$ that minimizes $\|x\|$

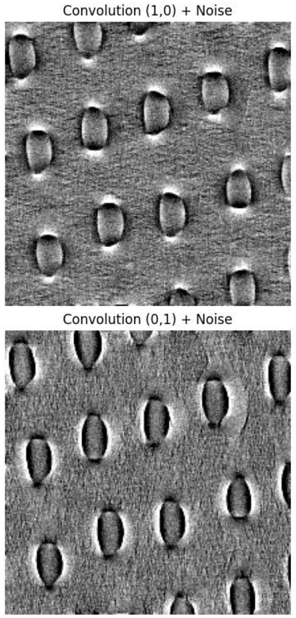
i.e., $x_{\rm ln}$ is solution of optimization problem

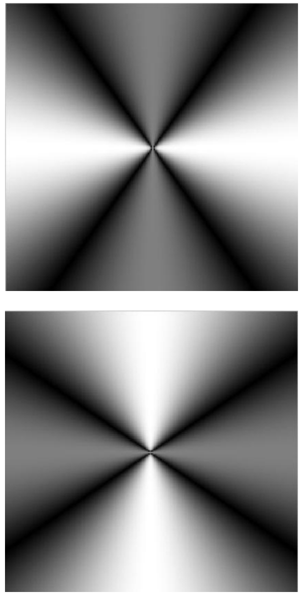
$$\begin{array}{ll} \text{minimize} & \|x\| \\ \text{subject to} & Ax = y \end{array}$$

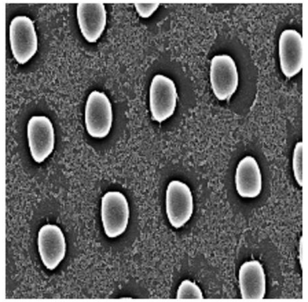
(with variable $x \in \mathbf{R}^n$)

Potential solutions

- Multiple orientation data
  - Remove zeros using three or more orientation data



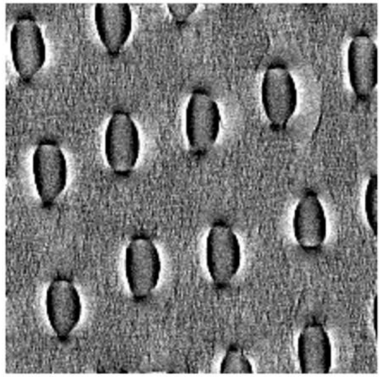




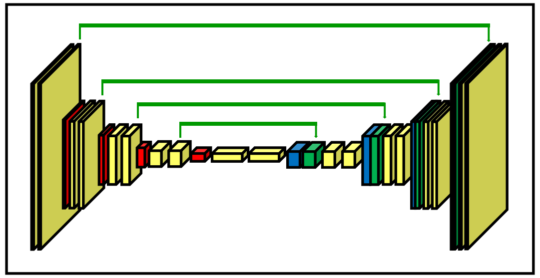
- Single orientation data
  - k-space threshold (TKD), regularization using L2 norm etc…

Potential solutions

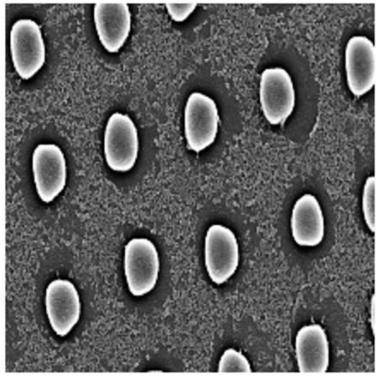
- Neural network



Single input

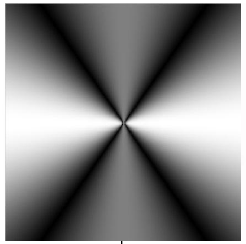


Deep neural network
(Your choice)



Ground truth

How do you implement?

Fourier Transform [Suscep distr] $\cdot$  = Fourier Transform [Local field distr]

$$Y = AX$$

Y : Input in Fourier domain ← we measured it

n^2 x 1 vector for $n \times n$ size image

A: Kernel in Fourier domain ← we know shape but has lots of zeros

n^2 x n^2 matrix

X: Output in Fourier domain ← what we want to know

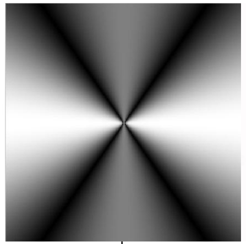
n^2 x 1 vector

$$X = A^{-1}Y$$

Nope! A is not full rank

$$\min_X \|Y - AX\| \; + \text{ regularization term}$$

How do you implement? How do you make skinny matrix?

Fourier Transform [Suscep distr]	•		=	Fourier Transform [Local field distr]

$$Y = AX$$

Y : Input in Fourier domain ← we measured it
n^2 x 1 vector for $n \times n$ size image

A: Kernel in Fourier domain ← we know shape but has lots of zeros
n^2 x n^2 matrix

X: Output in Fourier domain ← what we want to know
n^2 x 1 vector

$$Fy = AF\boldsymbol{\chi}$$

y: Input
n^2 x 1 vector

$$y = F^{-1}AF\boldsymbol{\chi}$$

Perform 2D FT, kernel multiplication, 2D FT^{-1} then columnize matrix

$$\min_{\boldsymbol{\chi}} \|y - F^{-1}AF\boldsymbol{\chi}\|$$ + regularization term