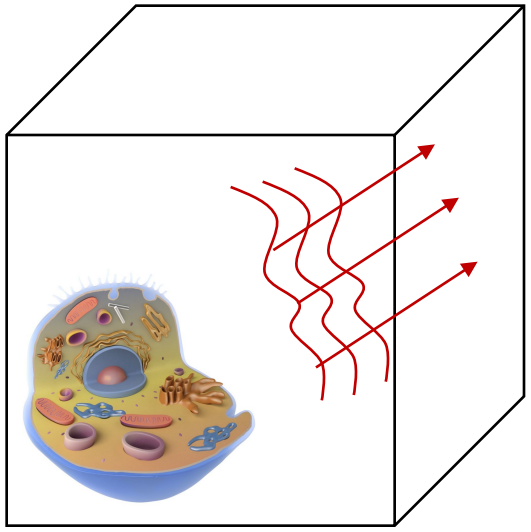
Machine Learning in Imaging

BME 590L
Roarke Horstmeyer

Lecture 25: Review of Machine Learning + Imaging

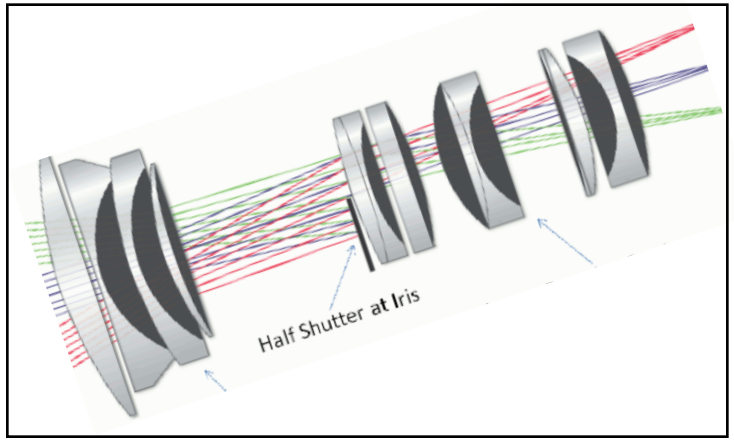
ML+Imaging pipeline introduction

Real World



Continuous complex fields

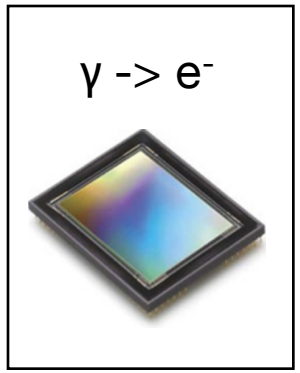
Measurement device



Black box transformations

- Convolution
- Fourier Transform

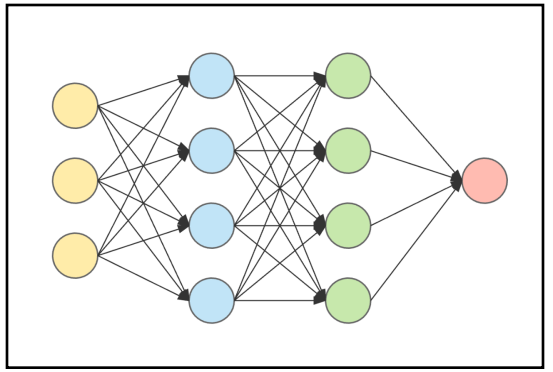
Digitization



Sampling Theorem

Discrete math &
Linear algebra

Machine Learning



Optimization

Linear classification

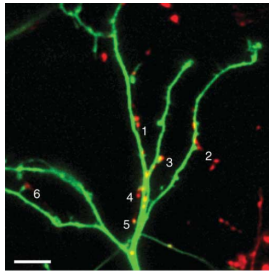
Logistic classifier

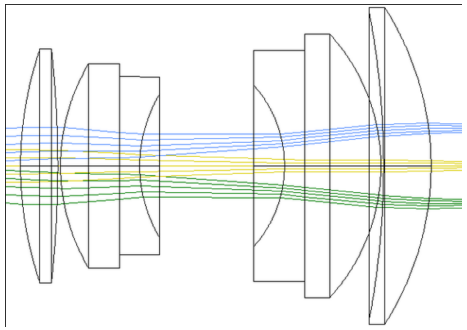
Neural networks

Convolutional NN's

Physical models for light propagation to sensor

- **Interpretation #1: Radiation (*Incoherent*)**
- Model: Rays



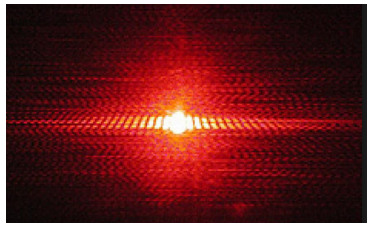


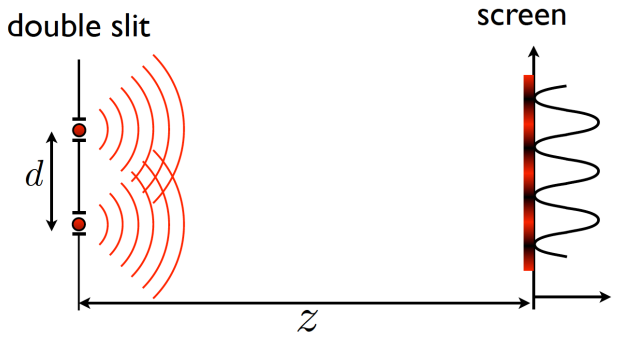
- Real, non-negative

$$I_s = H\,B\,S_0$$

- Sample absorption **S**
- Illumination brightness **B**
- Blur in **H**

- **Interpretation #2: Electromagnetic wave (*Coherent*)**
- Model: Waves





- Complex field

$$I_C = |\,H\,C\,S_C\,|^2$$

- Sample abs./phase **S**
- Illumination wave **B**
- Blur in **H**

Mathematical model of for incoherent image formation

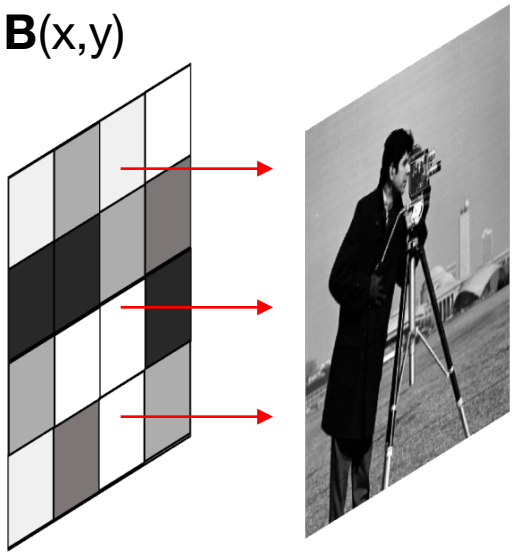
- All quantities are real, and non-negative

Object absorption:

$\mathbf{S_0}(x,y)$

Illumination brightness:

$\mathbf{B}(x,y)$



Mathematical model of for incoherent image formation

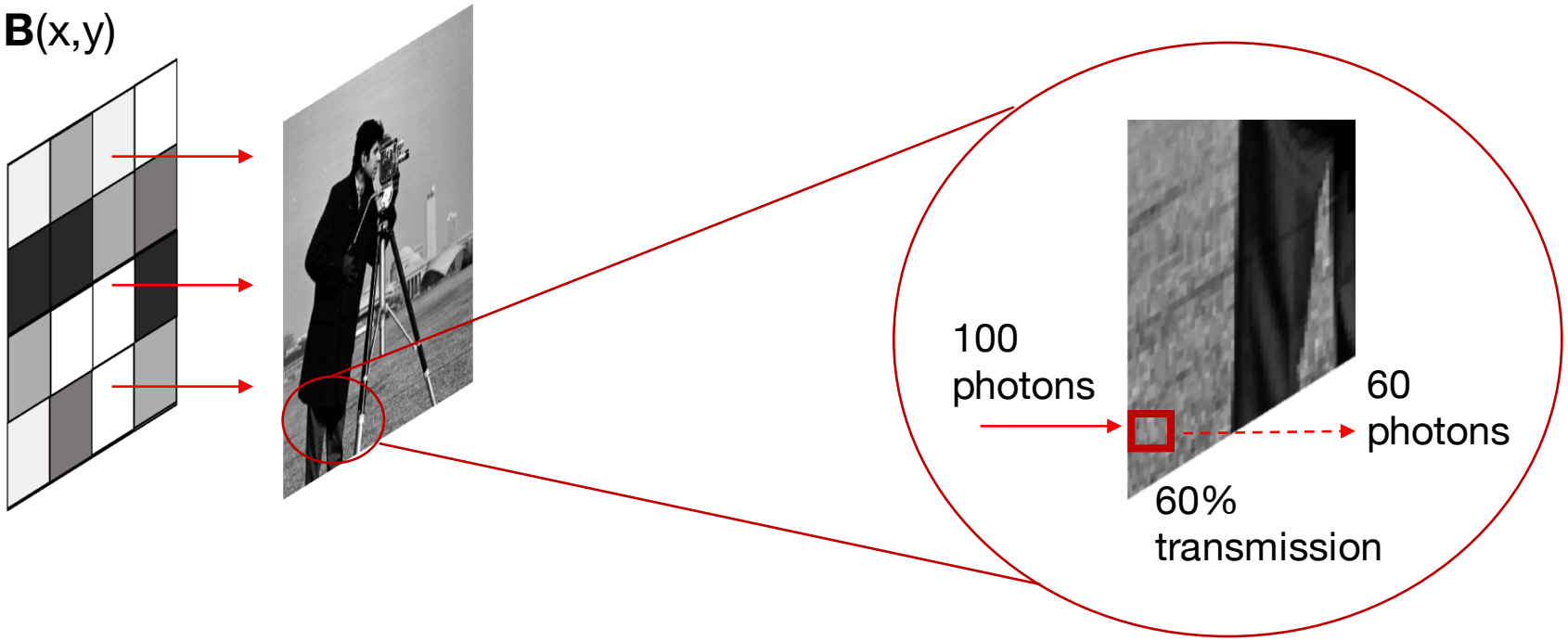
- All quantities are real, and non-negative

Object absorption:

$\mathbf{S_0}(x,y)$

Illumination brightness:

$\mathbf{B}(x,y)$



Mathematical model of for incoherent image formation

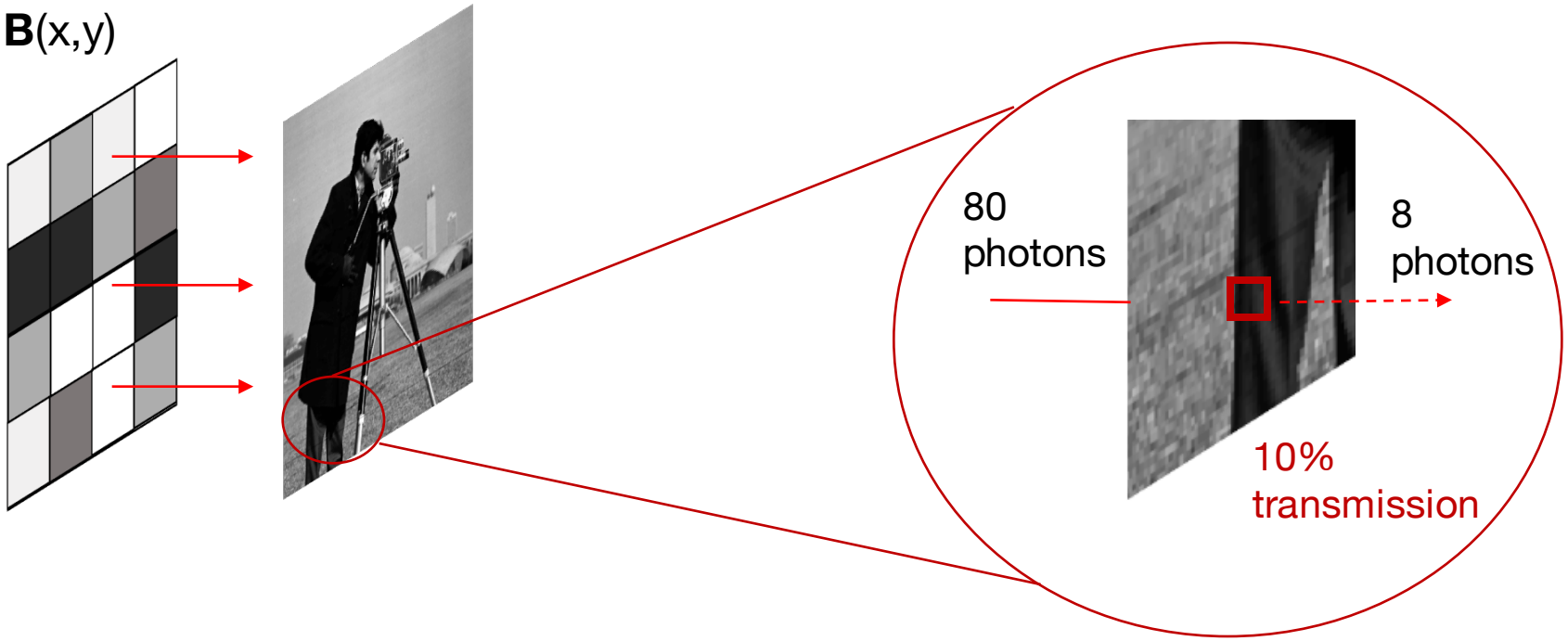
- All quantities are real, and non-negative

Object absorption:

$\mathbf{S_0}(x,y)$

Illumination brightness:

$\mathbf{B}(x,y)$



Mathematical model of for incoherent image formation

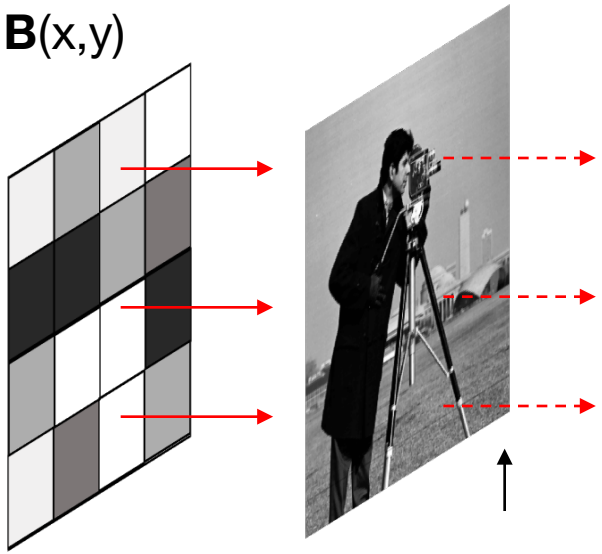
- All quantities are real, and non-negative

Object absorption:

$\mathbf{S_0}(x,y)$

Illumination brightness:

$\mathbf{B}(x,y)$



$\mathbf{B}\ \mathbf{S_0}$

multiplication

Mathematical model of for incoherent image formation

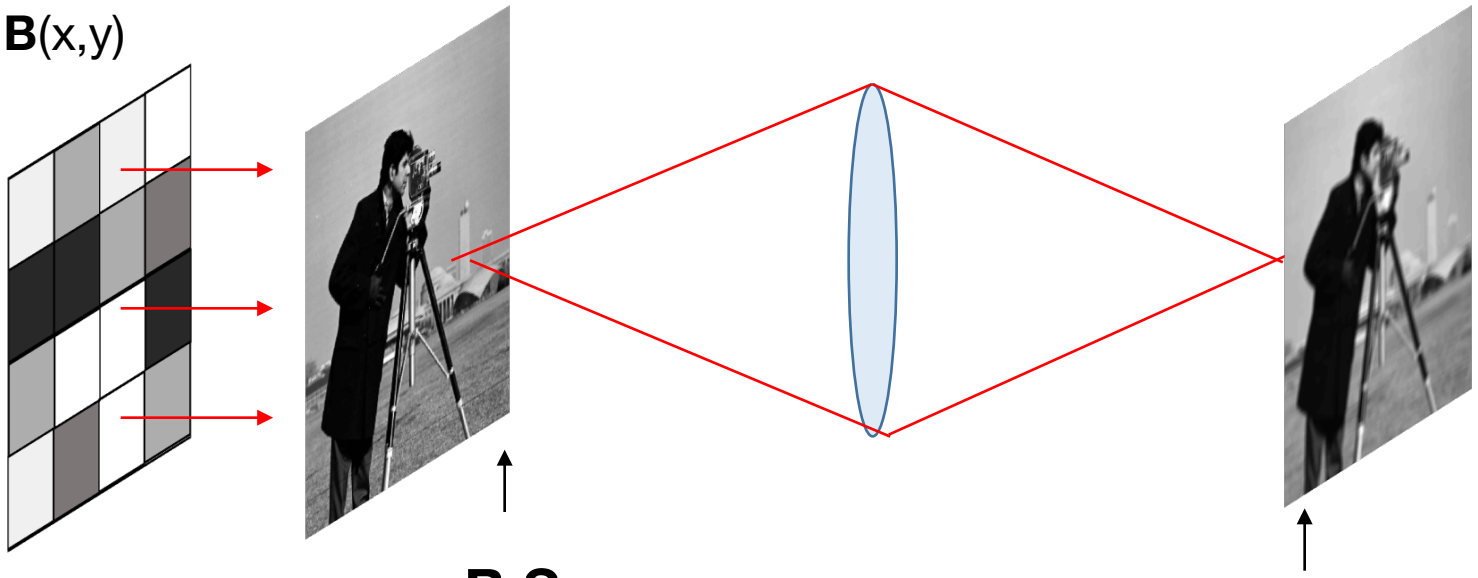
- All quantities are real, and non-negative

Object absorption:

$\mathbf{S_0}(x,y)$

Illumination brightness:

$\mathbf{B}(x,y)$



$\mathbf{B}\ \mathbf{S_0}$

multiplication

$(\mathbf{B}\ \mathbf{S_0}) * \mathbf{h}$

convolution

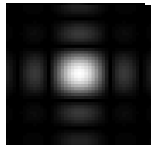
We can also add in some lens blur

Lenses blur and rescale images:

Input intensity

Convolution filter h

$*$

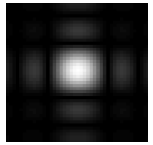


$=$

Output intensity

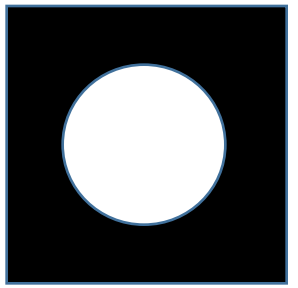
"Incoherent point-spread function"

h(x,y)



$h(x,y) = | F[A(f_x,f_y)] |^2$

$A(f_x,f_y)$

$= | F [$  $] |^2$

Mathematical model of for incoherent image formation

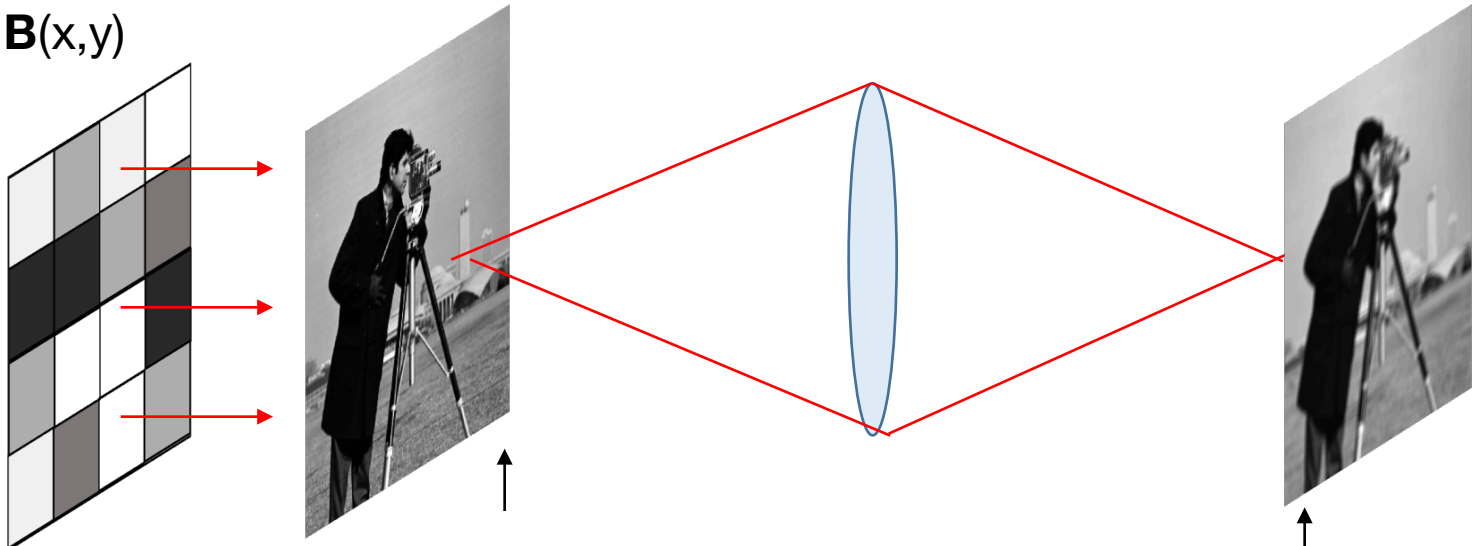
- All quantities are real, and non-negative

Object absorption:

$\mathbf{S_0}(x,y)$

Illumination brightness:

$\mathbf{B}(x,y)$



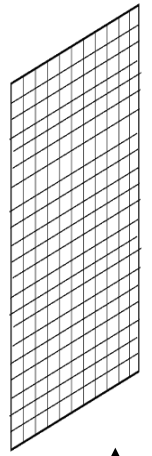
$\mathbf{B\ S_0}$

multiplication

$(\mathbf{B\ S_0}) * \mathbf{h}$

convolution

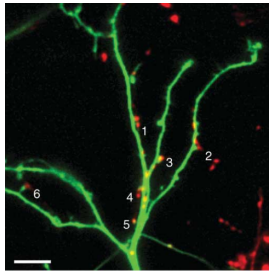
Photons (intensity) hits detector

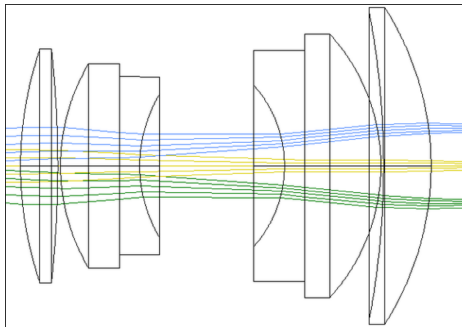


$\mathbf{I_s = H\ B\ S_0}$

Physical models for light propagation to sensor

- **Interpretation #1: Radiation (*Incoherent*)**
- Model: Rays



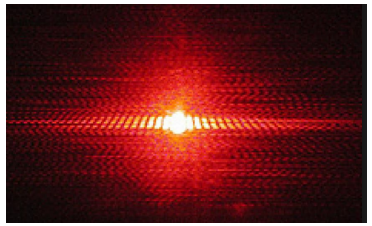


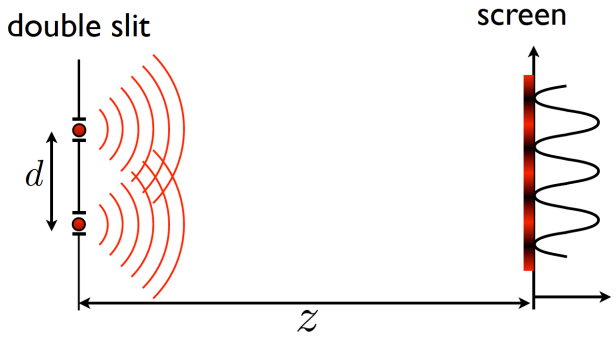
- Real, non-negative

$$I_s = H\,B\,S_0$$

- Sample absorption **S**
- Illumination brightness **B**
- Blur in **H**

- **Interpretation #2: Electromagnetic wave (*Coherent*)**
- Model: Waves





- Complex field

$$I_C = |\,H\,C\,S_C\,|^2$$

- Sample abs./phase **S**
- Illumination wave **B**
- Blur in **H**

Let's take a step back: how does light propagate?

Maxwell's equations without any charge

$$\nabla \times \vec{\mathcal{E}} = -\mu \frac{\partial \vec{\mathcal{H}}}{\partial t}$$

$$\nabla \times \vec{\mathcal{H}} = \epsilon \frac{\partial \vec{\mathcal{E}}}{\partial t}$$

$$\nabla \cdot \epsilon \vec{\mathcal{E}} = 0$$

$$\nabla \cdot \mu \vec{\mathcal{H}} = 0.$$

1. Take the curl of both sides of first equation
2. Substitute 2nd and 3rd equation
3. Arrive at the wave equation:

$$\nabla^2 \vec{\mathcal{E}} - \frac{n^2}{c^2} \frac{\partial^2 \vec{\mathcal{E}}}{\partial t^2} = 0 \qquad n = \left(\frac{\epsilon}{\epsilon_0} \right)^{1/2} \qquad c = \frac{1}{\sqrt{\mu_0 \epsilon_0}}.$$

Let's take a step back: how does light propagate?

Considering light that isn't pulsed over time, we can use the following solution:

$$u(P,t) = A(P)\cos[2\pi\nu t + \phi(P)]$$

$$u(P,t) = Re\{U(P)\exp(-j2\pi\nu t)\},$$

With this particular solution, we get the following important time-independent equation:

Helmholtz Equation

$$(\nabla^2 + k^2)U = 0.$$

$$k = 2\pi n\frac{\nu}{c} = \frac{2\pi}{\lambda},$$

This is an important equation in physics. We won't go into the details, but it leads to the Huygen-Fresnel principle:

$$U(P_2) = \frac{1}{j\lambda}\iint_{\Sigma} U(P_1)\frac{\exp(jkr_{21})}{r_{21}}\cos\theta\, ds$$

Plane-to-plane light propagation via the ”paraxial approximation”

We are usually concerned about propagation between two planes (almost always in an optical system):

Paraxial approximation:

$$\nabla_\perp^2 U + 2ik\frac{dU}{dz} = 0$$

Substitute in $U(P) = E(x,y,z)e^{ikz}$ and crank the wheel,

$$\nabla_\perp^2 E + 2ik\frac{dE}{dz} + 2k^2 E = 0$$

Paraxial Helmholtz Equation. This has an exact integral solution:

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z}\iint_{-\infty}^{+\infty} E(x',y',0)e^{\frac{ik}{2z}\left[(x-x')^2+(y-y')^2\right]}dx'\,dy'$$

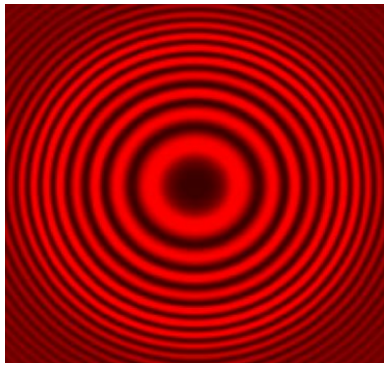
Fresnel diffraction integral

This is how light propagates from one plane to the next. It’s a convolution!

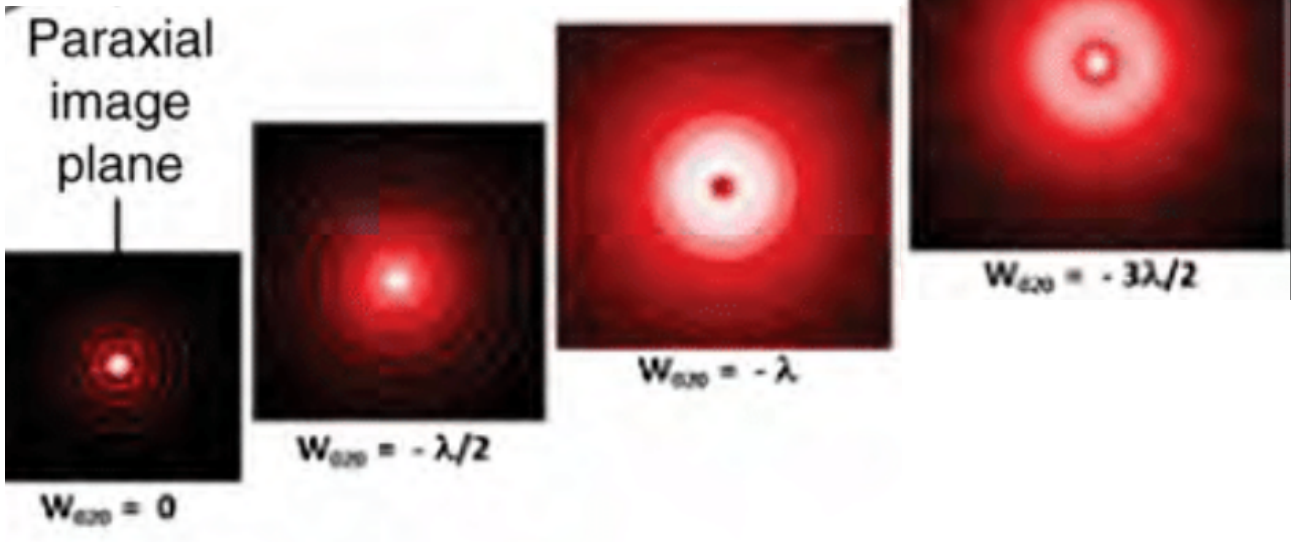
Fresnel light propagation as a convolution

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z} \iint_{-\infty}^{+\infty} E(x',y',0) e^{\frac{ik}{2z}\left[(x-x')^2+(y-y')^2\right]} dx'\,dy'$$

$$h(x,y,z) = \frac{e^{ikz}}{i\lambda z} e^{i\frac{k}{2z}(x^2+y^2)}$$



$$E(x,y,z) = E(x,y,0) * h(x,y,z)$$



From the Fresnel approximation to the Fraunhofer approximation

Fresnel Approximation:

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z}\iint_{-\infty}^{+\infty} E(x',y',0)e^{\frac{ik}{2z}\left[(x-x')^2+(y-y')^2\right]}dx'dy'$$

Lets assume that the second plane is "pretty far away" from the first plane. Then,

$$z > \frac{2D^2}{\lambda}$$

1. Expand the squaring

$$E(x,y,z) = \frac{e^{ikz}}{i\lambda z}\iint E(x',y',0)e^{\frac{ik}{2z}(x^2+y^2)}e^{\frac{ik}{2z}(x'^2+y'^2)}e^{\frac{ik}{2z}(xx'+yy')}dx'dy'$$

2. Front term comes out, assume second term goes away, then,

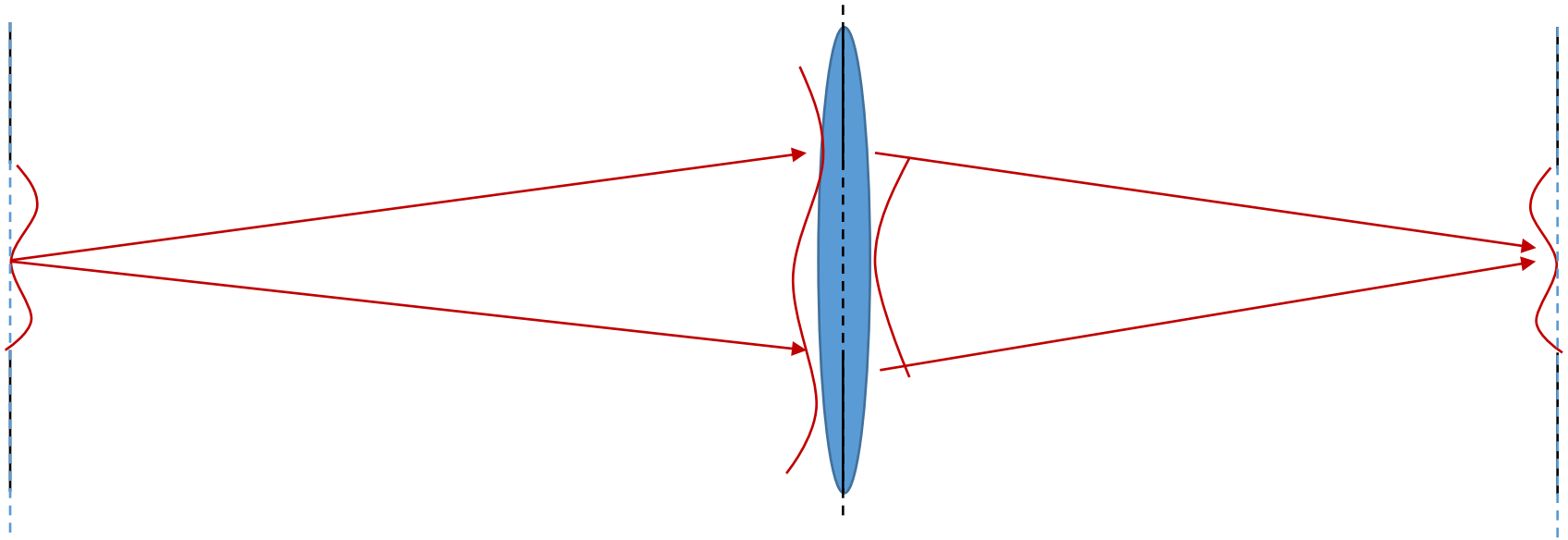
$$E(x,y,z) = C\iint E(x',y',0)e^{\frac{ik}{2z}(xx'+yy')}dx'dy'$$

$$C = \frac{e^{ikz}}{i\lambda z}e^{\frac{ik}{2z}(x^2+y^2)}$$

Fraunhofer diffraction is a Fourier transform!!!!!!!

Model of a microscope (or camera) using Fourier transforms:

from lens to sensor, light undergoes an ***inverse Fourier transform!***



$$E_s(x_s, y_s, 0)$$

2D Fourier Transform

$$E'_a(x_d, y_d) = E_a(x_d, y_d)A(x_d, y_d)$$

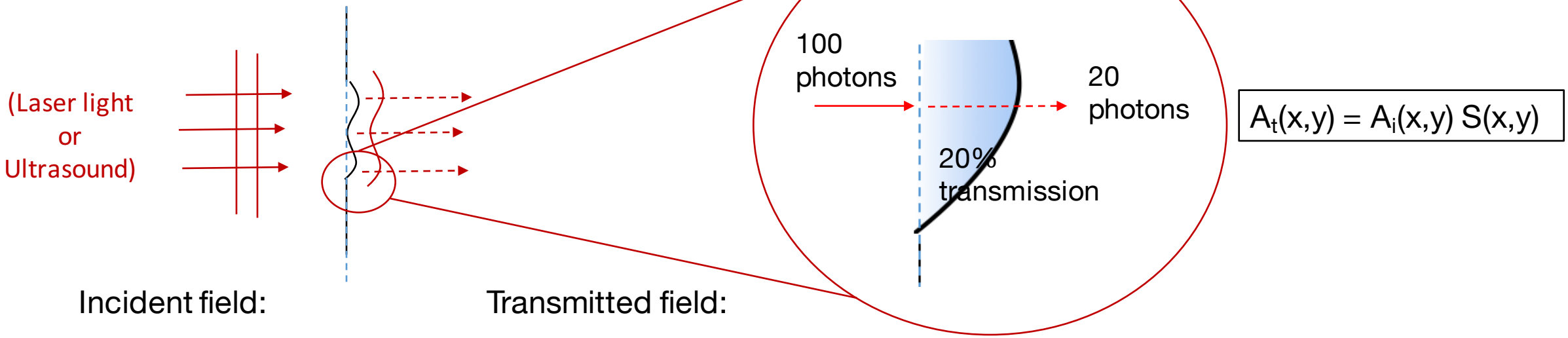
2D inverse Fourier Transform

Mathematical model of for coherent image formation

- Pretty much the same thing, but now we have an amplitude and a complex phase

Point #1: Amplitudes behave just like before

Sample absorption = S(x,y)



$A_t(x,y) = A_i(x,y)\, S(x,y)$

Incident field:

$C(x,y) = A_i(x,y)$

Transmitted field:

$U(x,y) = A_t(x,y) = A_i(x,y)\, S(x,y)$

Mathematical model of for coherent image formation

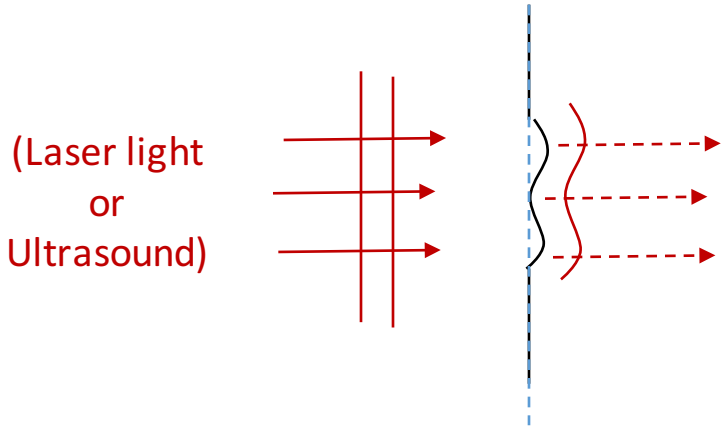
- Pretty much the same thing, but now we have an amplitude and a complex phase

Sample absorption = S(x,y)

Sample phase delay = exp[ikφ(x,y)]

New: complex phase delay

- Needed to represent wave
- Represents wave delay across space



Incident field:

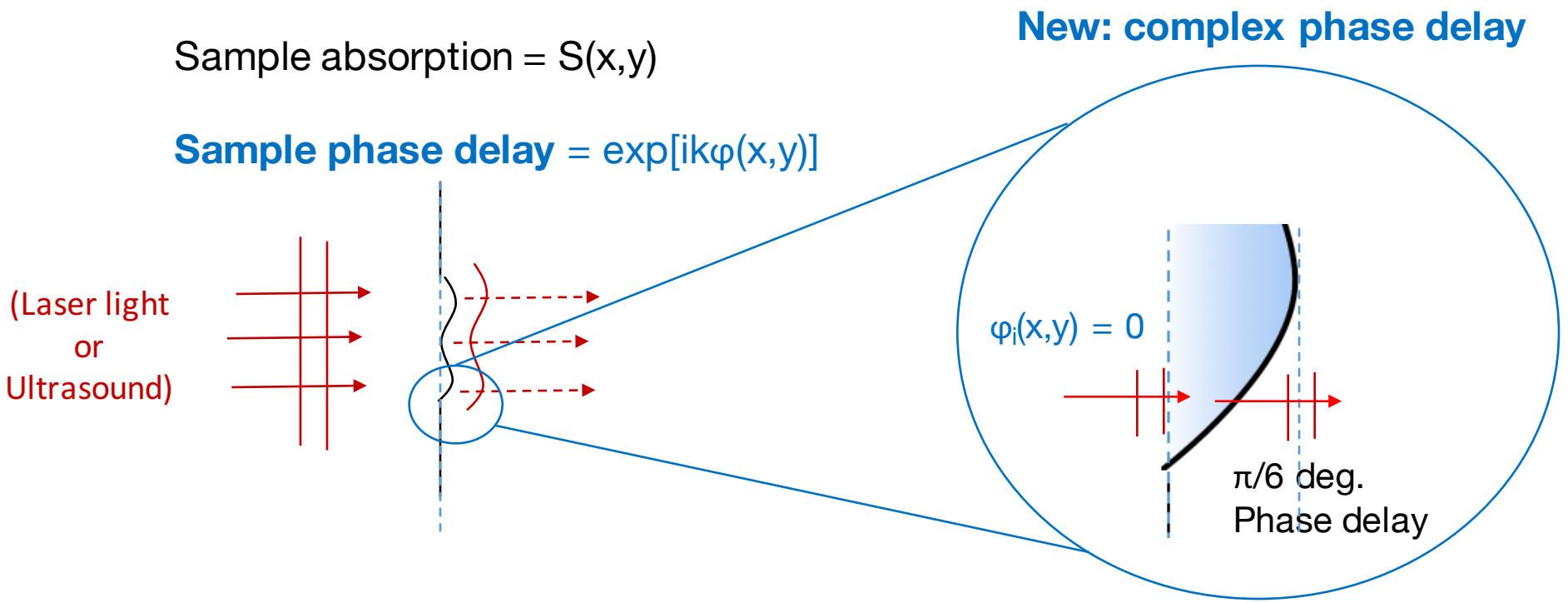
$C(x,y) = A_i(x,y)\ \exp[ik\varphi_i(x,y)]$

Transmitted field:

$U(x,y) = A_i(x,y)\ S(x,y)\ \exp[ik\varphi_t(x,y)]$

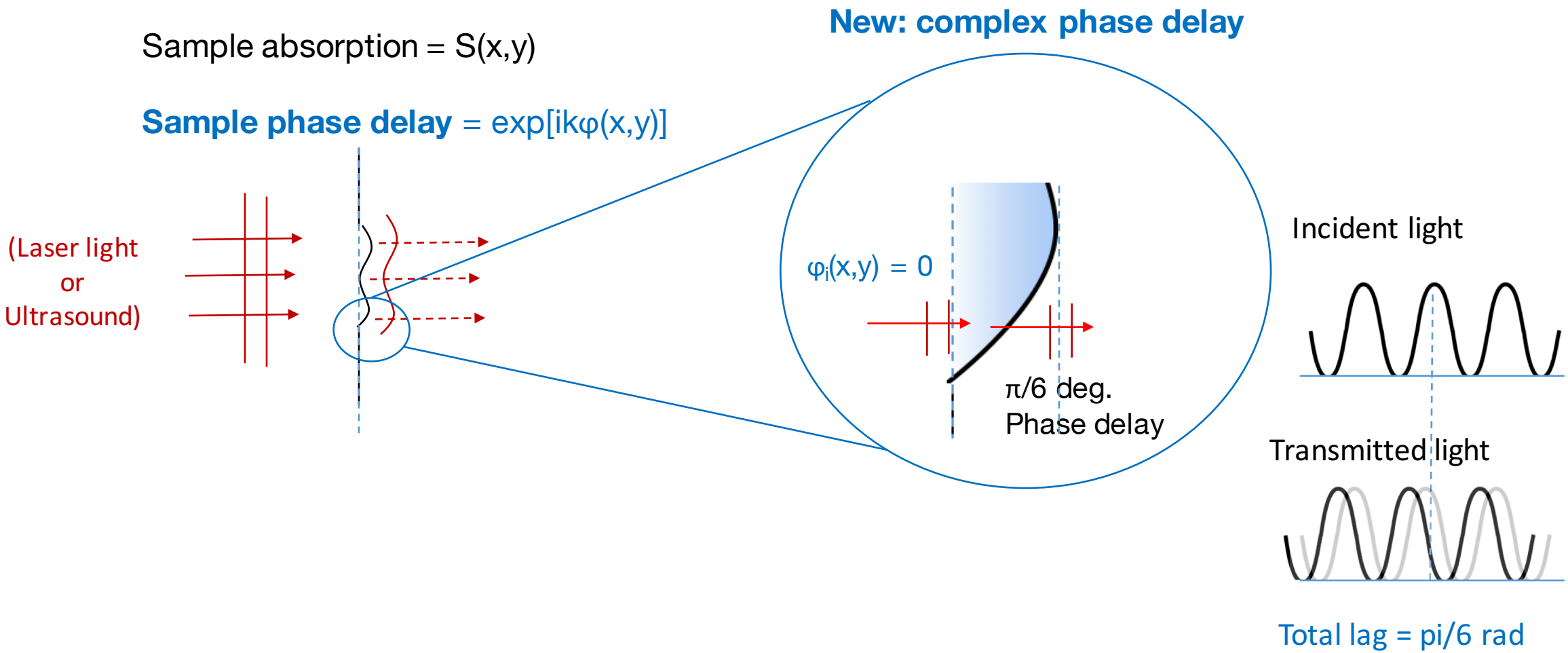
Mathematical model of for coherent image formation

- Pretty much the same thing, but now we have an amplitude and a complex phase



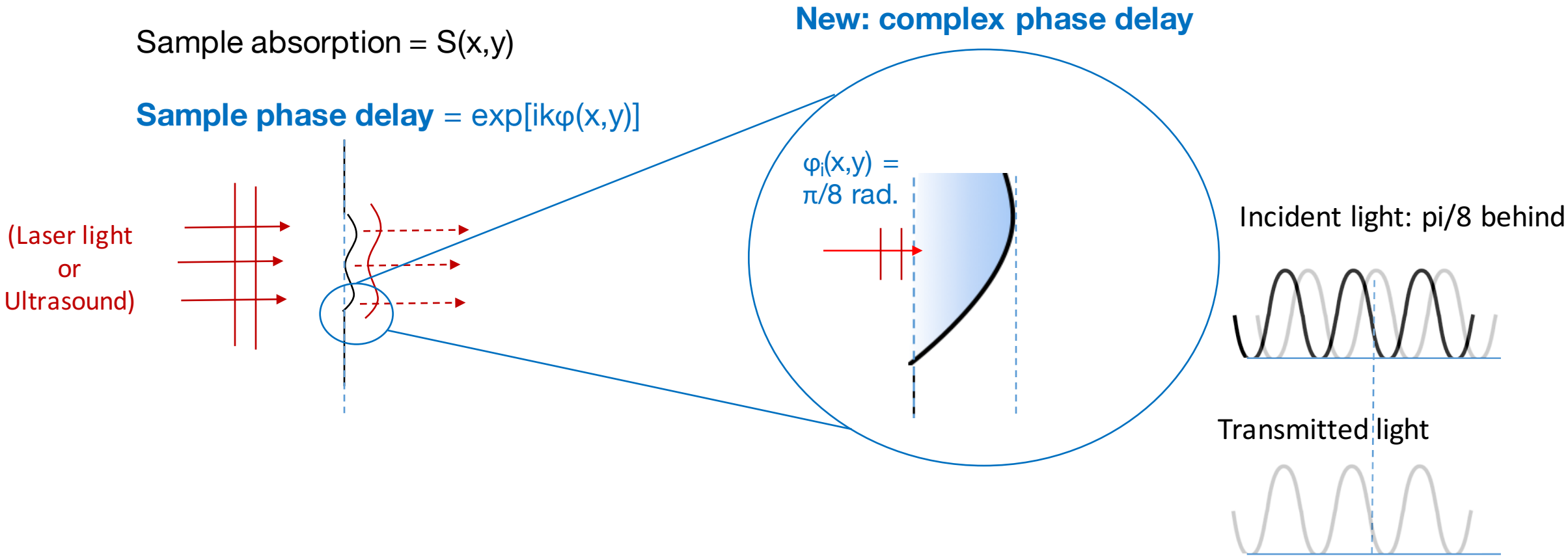
Mathematical model of for coherent image formation

- Pretty much the same thing, but now we have an amplitude and a complex phase



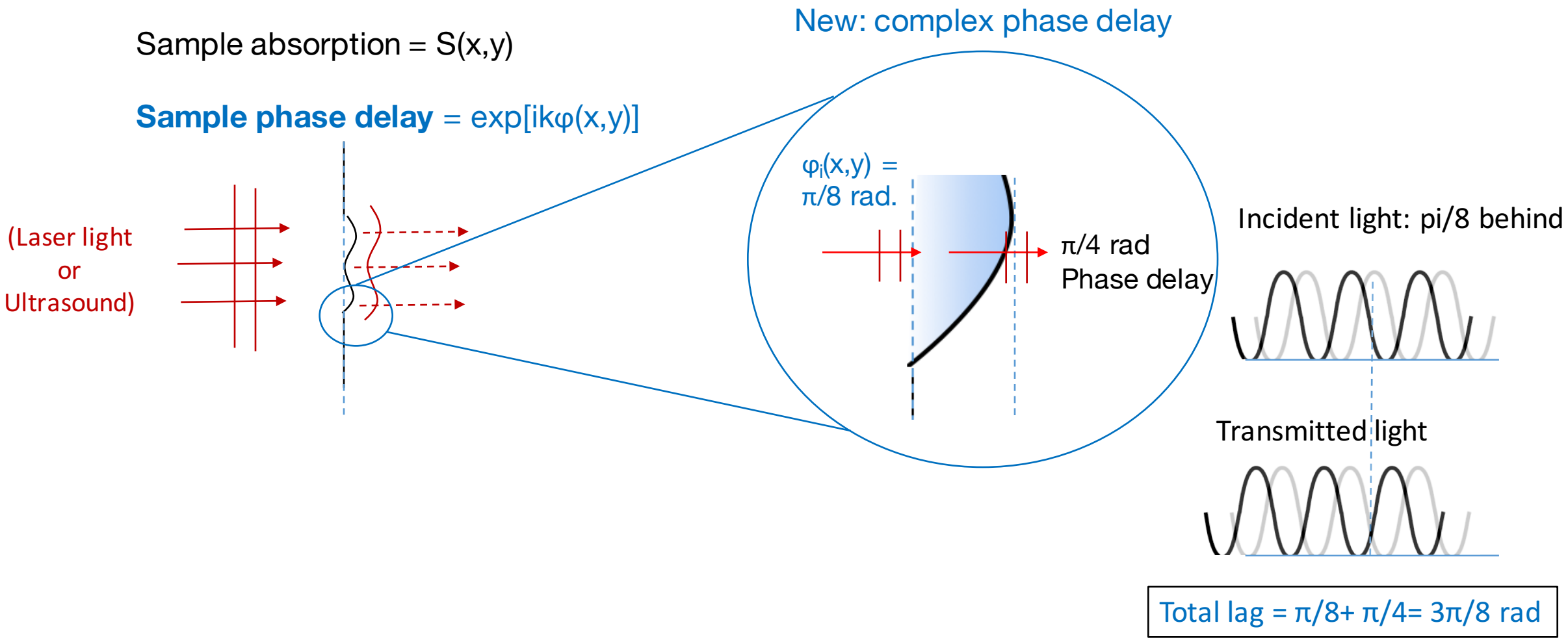
Mathematical model of for coherent image formation

- Pretty much the same thing, but now we have an amplitude and a complex phase



Mathematical model of for coherent image formation

- Pretty much the same thing, but now we have an amplitude and a complex phase



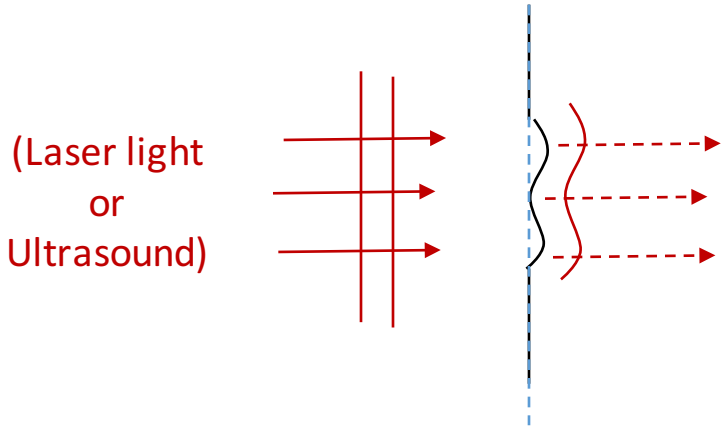
Mathematical model of for coherent image formation

- Pretty much the same thing, but now we have an amplitude and a complex phase

Sample absorption = S(x,y)

Sample phase delay = exp[ikφ(x,y)]

Output phase is sum of phase delays, product of phasors



$$\varphi_t(x,y) = \varphi(x,y) + \varphi_i(x,y)$$

$$\exp[ik\varphi_t(x,y)] = \exp[ik\varphi_i(x,y)]\ \exp[ik\varphi(x,y)]$$

Incident field:

$C(x,y) = A_i(x,y)\ \exp[ik\varphi_i(x,y)]$

Transmitted field:

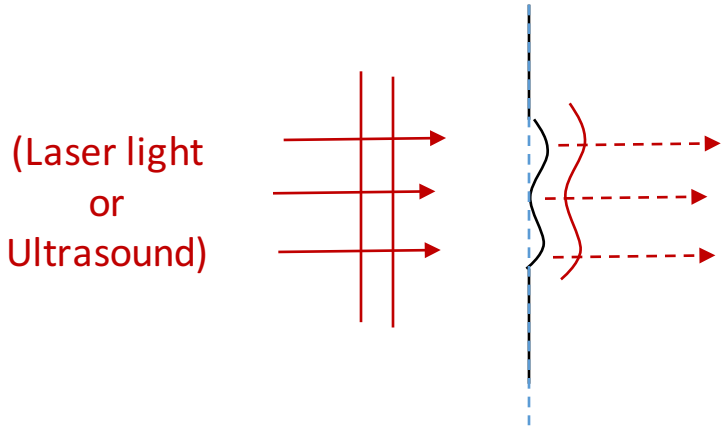
$U(x,y) = A_i(x,y)\ S(x,y)\ \exp[ik\varphi_i(x,y)]\ \exp[ik\varphi(x,y)]$

Mathematical model of for coherent image formation

- Pretty much the same thing, but now we have an amplitude and a complex phase

Sample absorption = S(x,y)

Sample phase delay = exp[ikφ(x,y)]



Conclusion:

Transmitted field = incident field x complex sample :

$U(x,y) = C(x,y)\, S(x,y)\, \exp[ik\varphi(x,y)]$

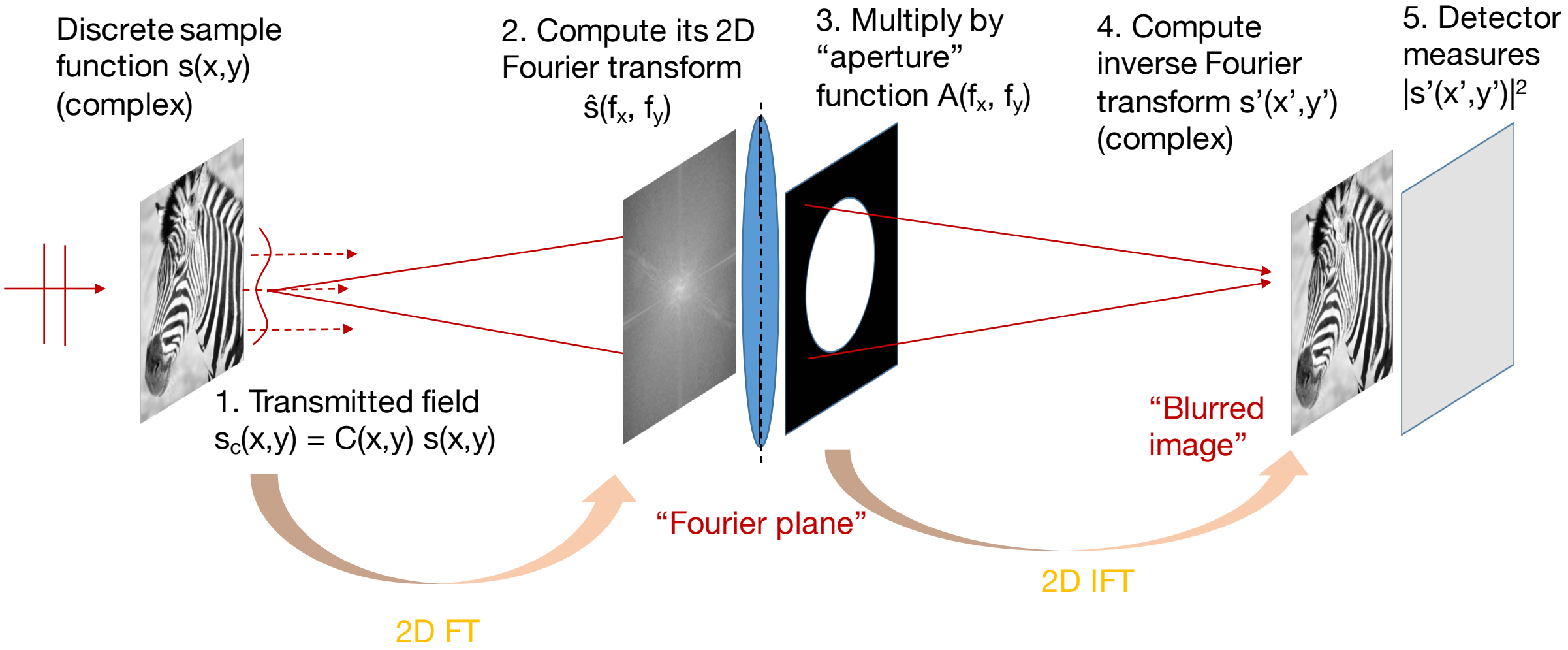
Incident field:

$C(x,y) = A_i(x,y)\, \exp[ik\varphi_i(x,y)]$

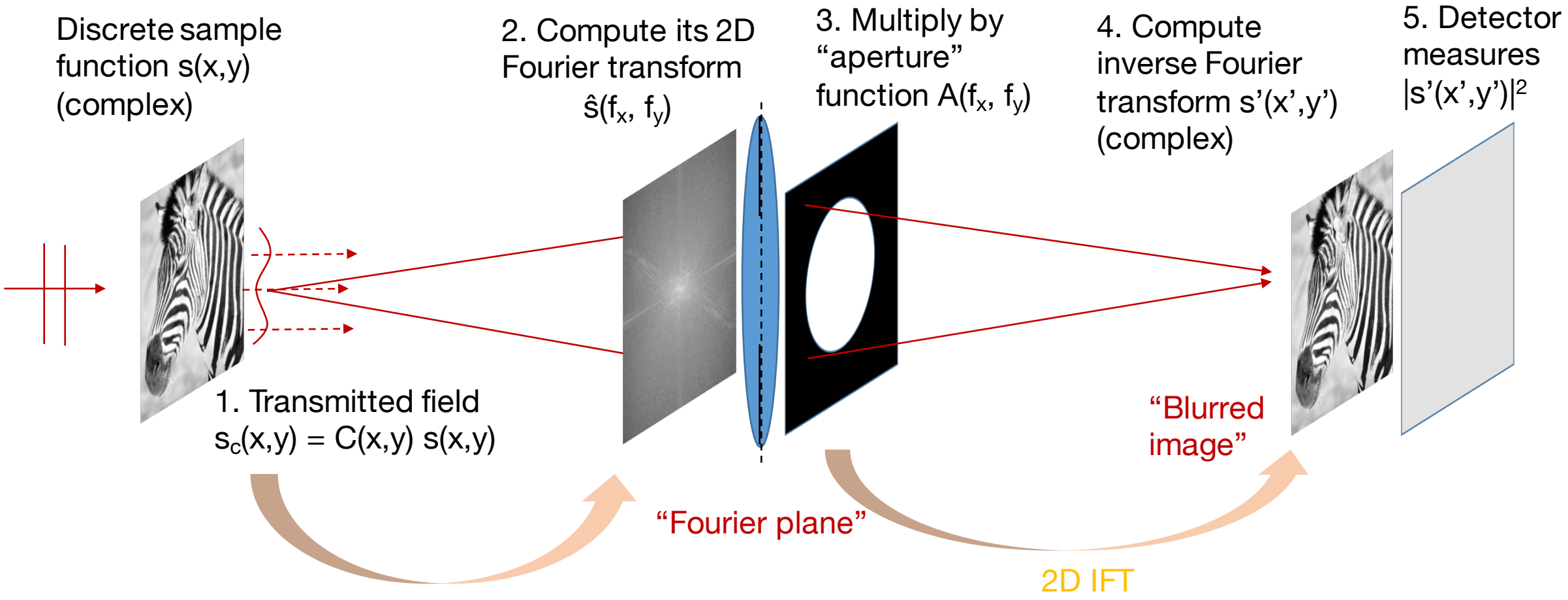
Transmitted field:

$U(x,y) = A_i(x,y)\, S(x,y)\, \exp[ik\varphi_i(x,y)]\, \exp[ik\varphi(x,y)]$

Model of image formation for wave optics (coherent light):



Model of image formation for wave optics (coherent light):



Model #1: $I_c(x,y) = |F^{-1}AFCs|^2$

Coherent image blur – two implementations

Input image

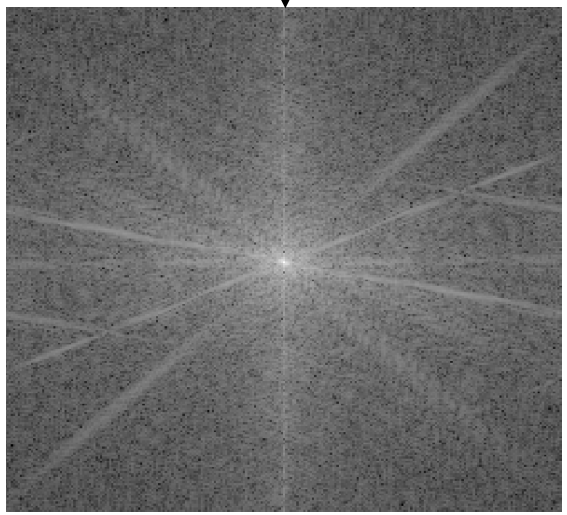
$U_1(x,y)$

$F\,[U_1]$

f_y

Input spectrum

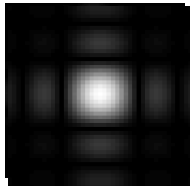
$\hat{U}_1(f_x,f_y)$



f_x

Convolution filter h

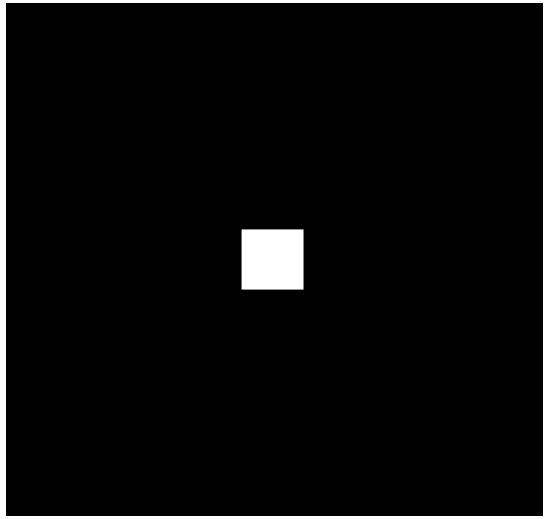
$*$



$F\,[h]$

f_y

$\bullet$



f_x

$=$

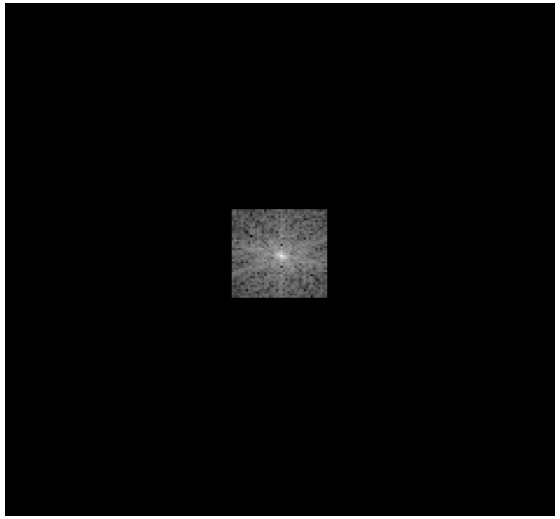
Output image

$U_2(x,y)$

$F^{-1}[H\hat{U}_1]$

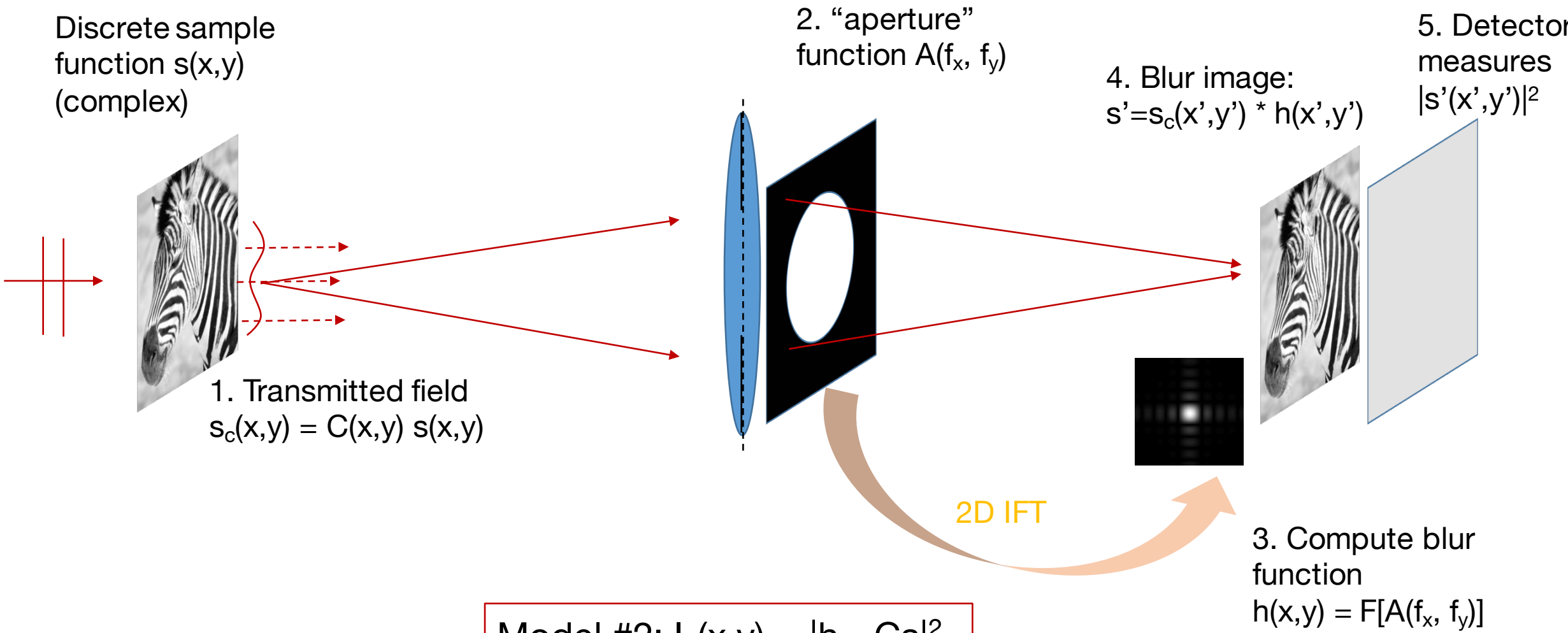
f_y

$=$



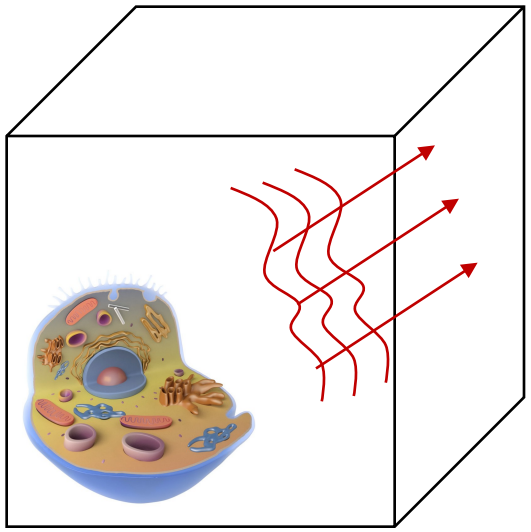
f_x

Model of image formation for wave optics (coherent light):



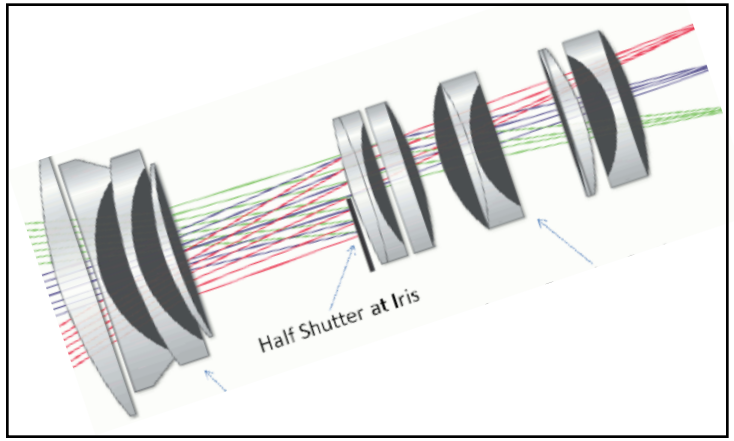
Model #2: $I_c(x,y) = |h * Cs|^2$

Real World



Continuous complex fields

Measurement device



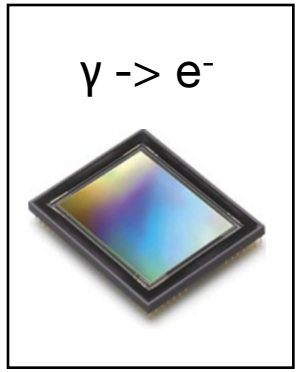
Black box transformations

- Convolution
- Fourier Transform

$$I_{Inc} = H\ B\ S_0$$

$$I_{Coh} = |H\ C\ S_C|^2$$

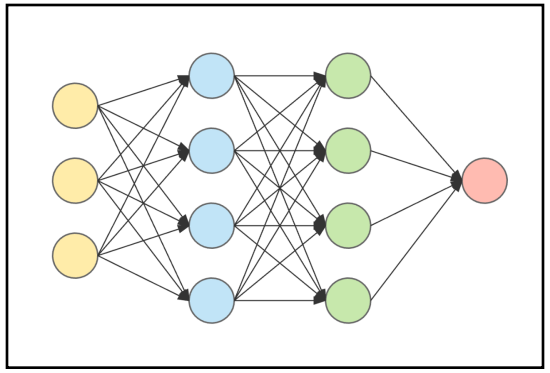
Digitization



Sampling Theorem

Discrete math & Linear algebra

Machine Learning



Optimization

Linear classification

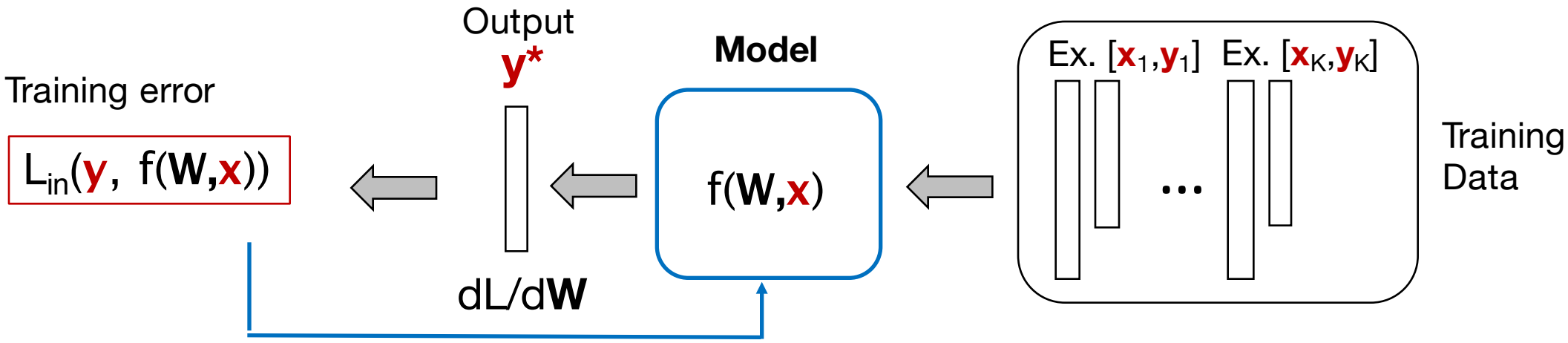
Logistic classifier

Neural networks

Convolutional NN's

Summary of machine learning pipeline:

1. Network Training

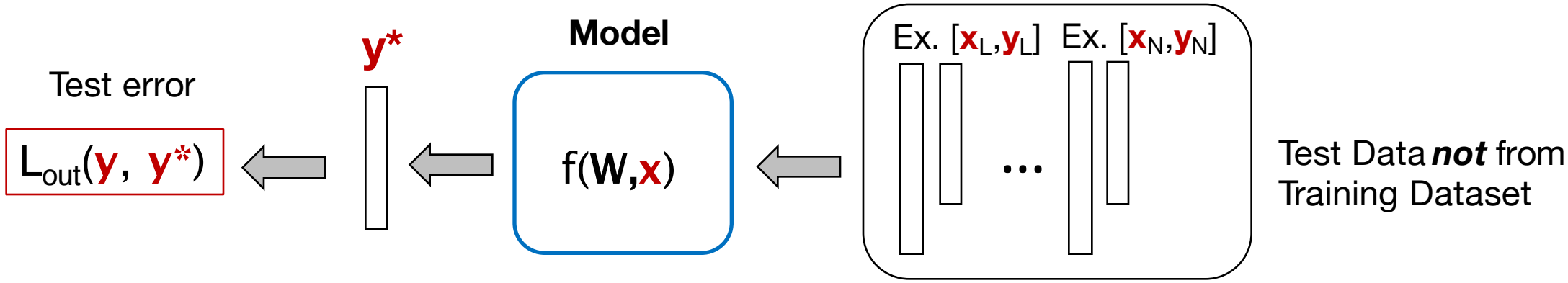


What we need for network training:

1. **Labeled examples**
2. **A model and loss function**
3. **A way to minimize the loss function L**

Summary of machine learning pipeline:

2. Network Testing



What we need for network testing:

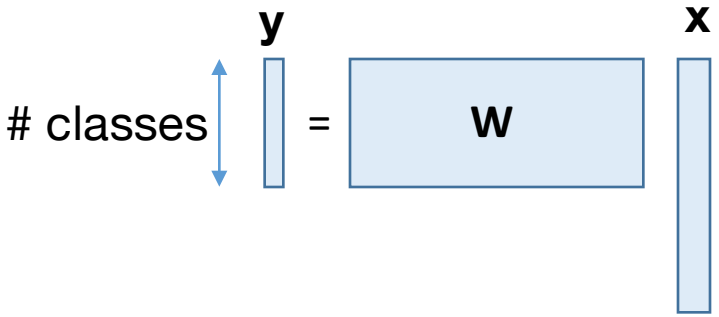
4. *Unique* labeled test data
5. Evaluation of model error

Let's start with a simpler approach: linear regression

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, \mathbf{W}), y_i)$$

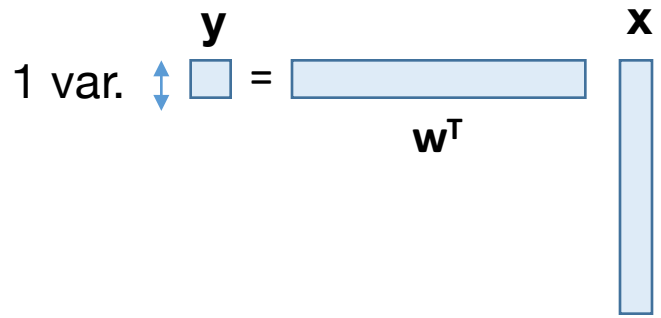
General linear model:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(\mathbf{W}\mathbf{x}_i, y_i)$$



Assume 1 class = 1 linear fit

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i(w^T x_i - y_i)$$



Use MSE error model

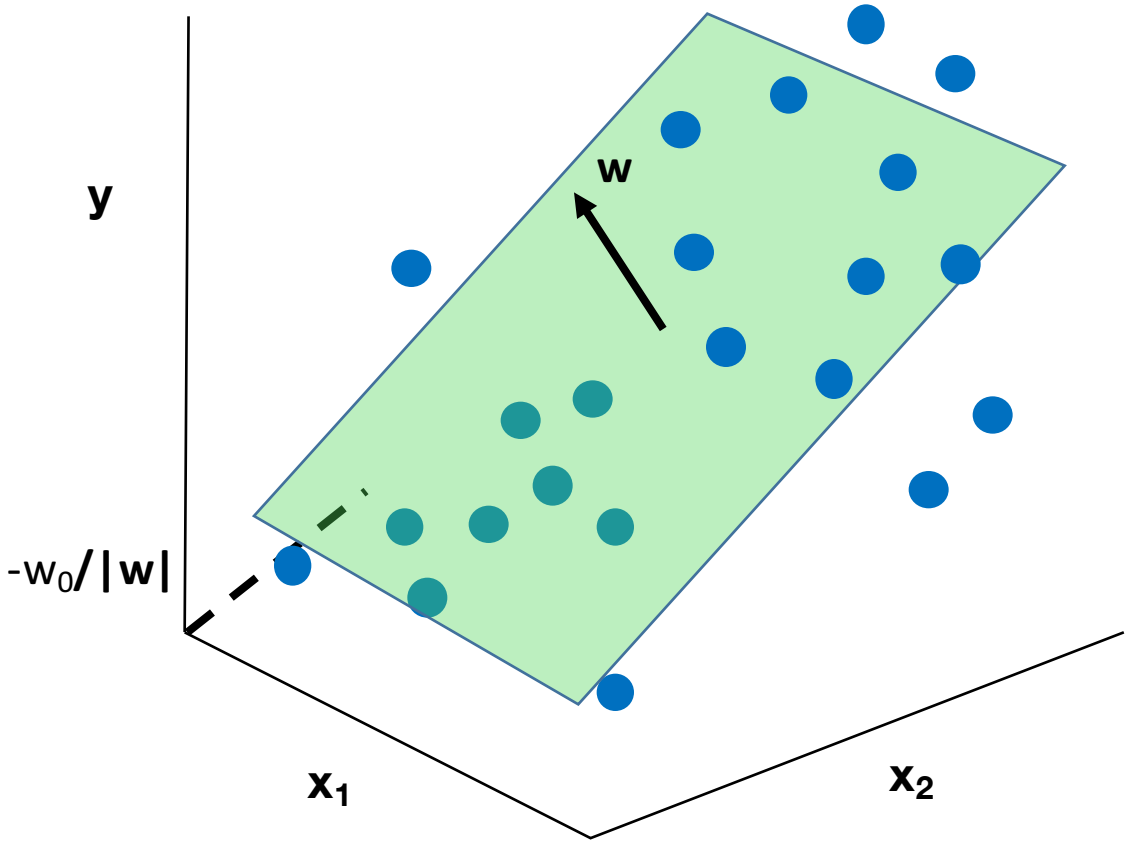
$$L = \frac{1}{N}\sum_{i=1}^{N} (w^T x_i - y_i)^2$$

Where labels determined by thresholding

$$f(\mathbf{x}_i) = y_i^* = \text{sgn}(\mathbf{w}^T\mathbf{x}_i)$$

$$\text{sgn}(x) := \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

Why does linear regression with sgn() achieve classification?



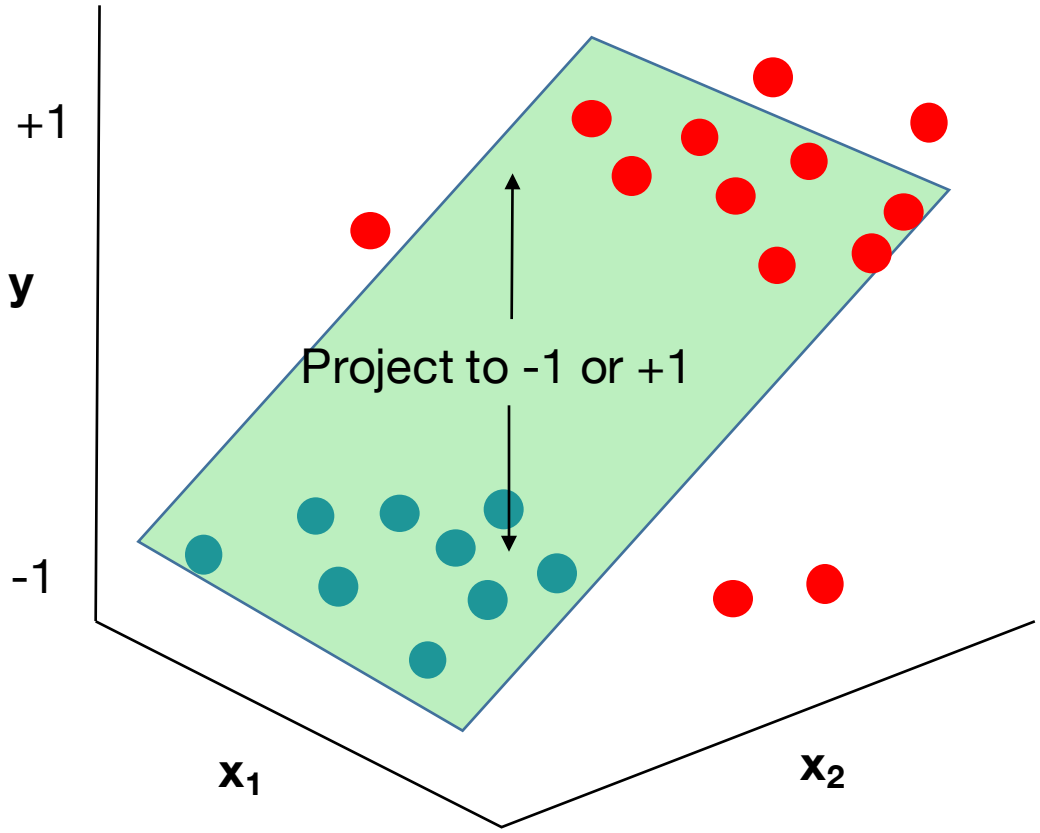
Without sgn(): regression for best fit

$$f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i$$

$$L = \frac{1}{N} \sum_{i=1}^{N} (w^T x_i - y_i)^2$$

- If y_i can be anything, minimizing *L* makes **w** the plane of best fit

Why does linear regression with sgn() achieve classification?



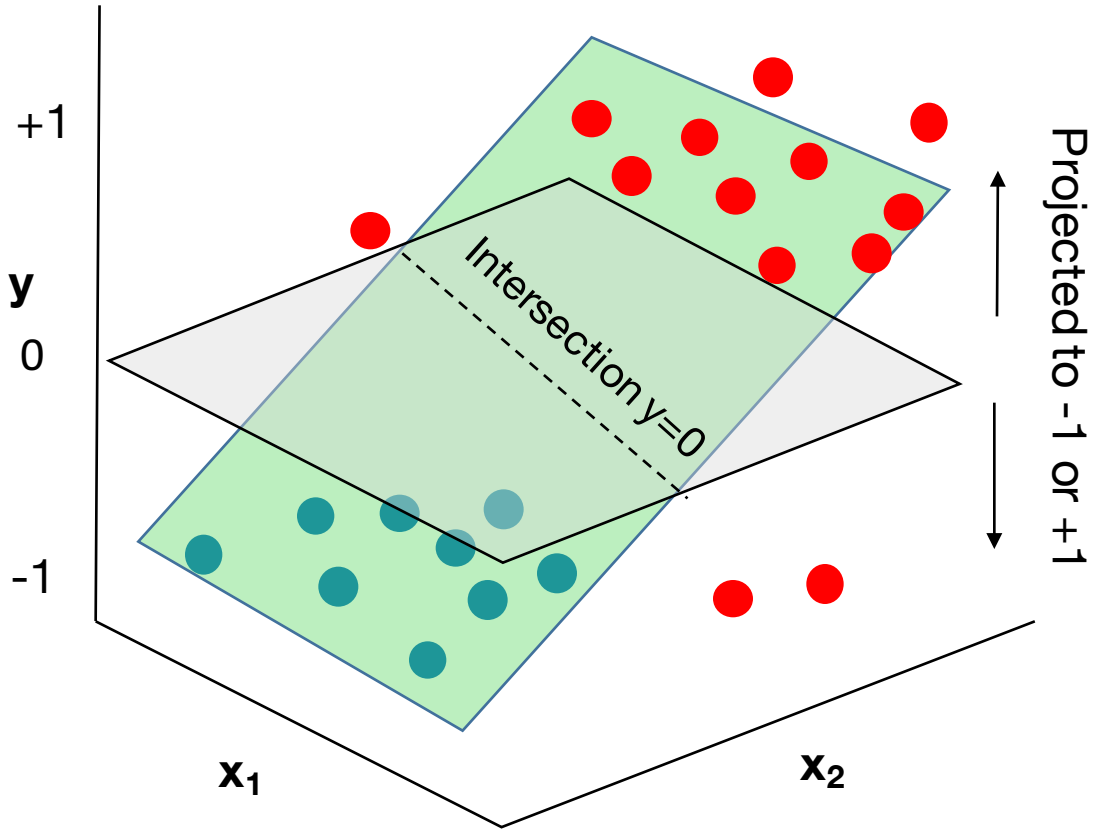
Without sgn(): regression for best fit

$$f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i$$

$$L = \frac{1}{N} \sum_{i=1}^{N} (w^T x_i - y_i)^2$$

- y_i can only be -1 or +1, which defines its class
- Can still find plane of best fit

Why does linear regression with sgn() achieve classification?



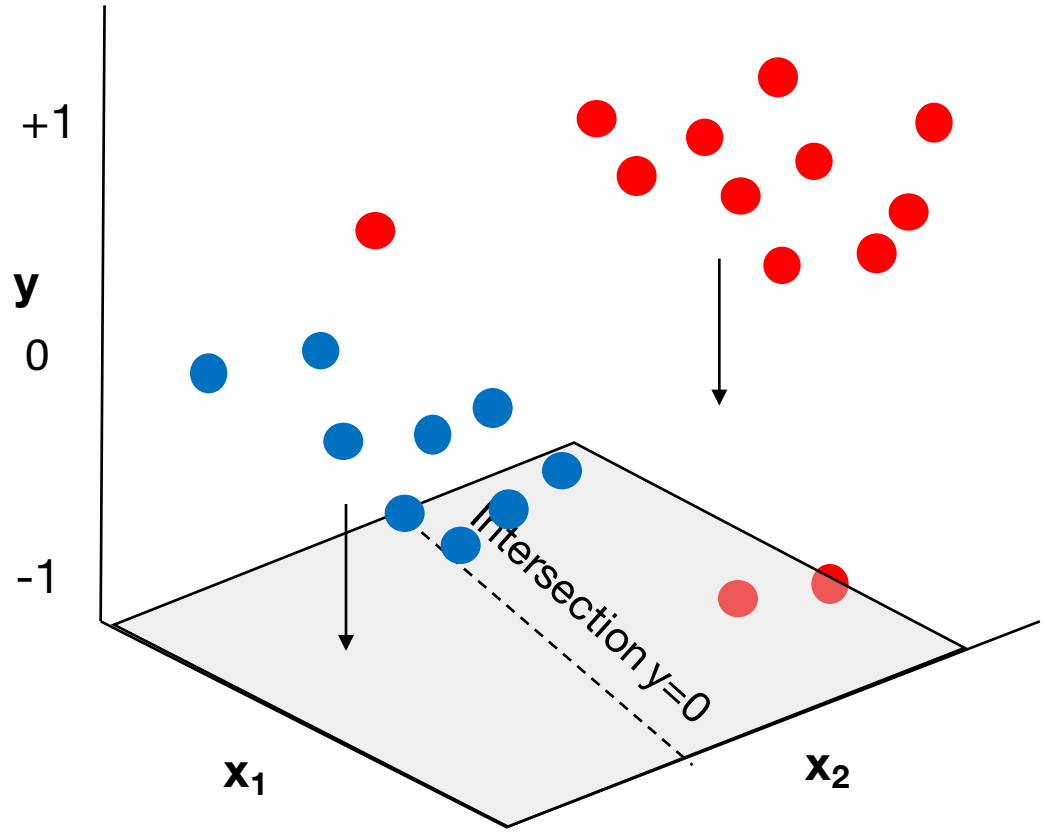
With sgn() operation:

$$f(\mathbf{x}_i) = y_i^* = \text{sgn}(\mathbf{w}^T\mathbf{x}_i)$$

$$L = \frac{1}{N}\sum_{i=1}^{N}(w^T x_i - y_i)^2$$

- Anything point to one side of y=0 intersection is class +1, anything on the other side of intersection is class -1

Why does linear regression with sgn() achieve classification?



With sgn() operation:

$$f(\mathbf{x}_i) = y_i^* = \text{sgn}(\mathbf{w}^T \mathbf{x}_i)$$

$$L = \frac{1}{N} \sum_{i=1}^{N} (w^T x_i - y_i)^2$$

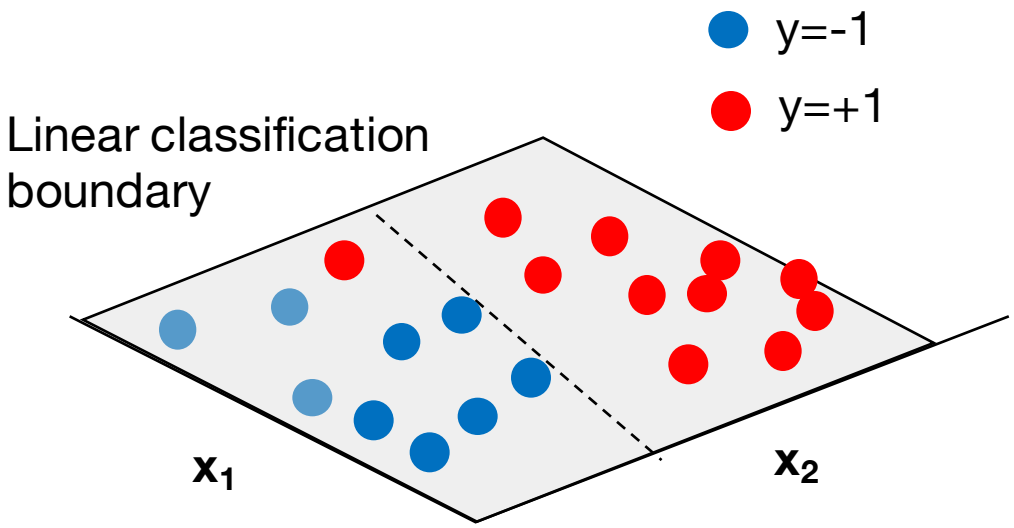
- y axis isn't really needed now & can view this decision boundary in 2D

Why does linear regression with sgn() achieve classification?

With sgn() operation:

$$f(\mathbf{x}_i) = y_i^* = \text{sgn}(\mathbf{w}^T \mathbf{x}_i)$$

$$L = \frac{1}{N} \sum_{i=1}^{N} (w^T x_i - y_i)^2$$



Sign operation takes linear regression and makes it a classification operation!

In the rest of this class: solve via gradient descent

With a matrix, compute this for each entry:

$$\frac{dL(W_i)}{dW_i} = \lim_{h \to 0} \frac{L(W_i + h) - L(W_i)}{h}$$

Example:

W = [1,2;3,4]
L(W, x, y) = 12.79

W_1+h = [1.001,2;3,4]
L(W_1+h, x, y) = 12.8

$dL(W_1)/dW_1$ = 12.8-12.79/.001

$dL(W_1)/dW_1$ = 10

- Repeat for all entries of **W**, dL/d**W** will have NxM entries for NxM matrix

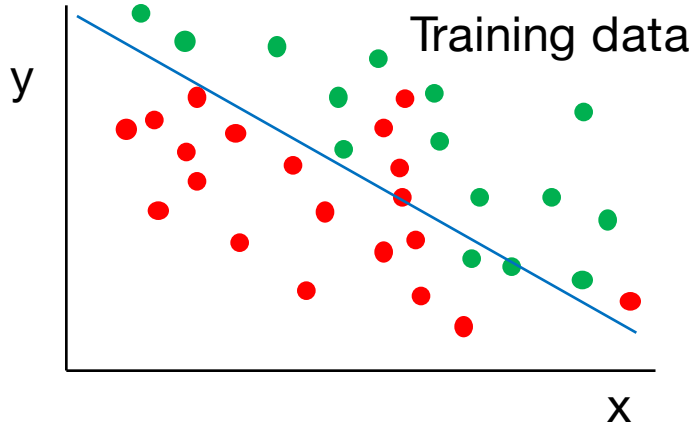
Steepest descent and the best step size ε

1. Evaluate function f($\mathbf{x}^{(0)}$) at an initial guess point, $\mathbf{x}^{(0)}$
2. Compute gradient $\mathbf{g}^{(0)} = \nabla_x f(\mathbf{x}^{(0)})$
3. Next point $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \boldsymbol{\varepsilon}^{(0)}\mathbf{g}^{(0)}$
4. Repeat – $\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \boldsymbol{\varepsilon}^{(n)}\mathbf{g}^{(n)}$, until $|\mathbf{x}^{(n+1)}-\mathbf{x}^{(n)}| <$ threshold t

$$L = \frac{1}{N}\sum_{i=1}^{N}(w^T x_i - y_i)^2$$

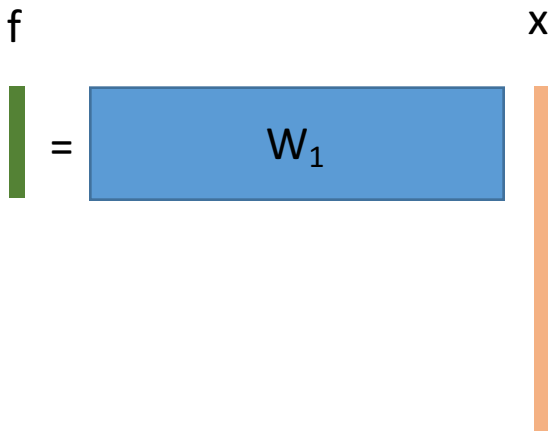
$$\nabla L(w) = \frac{2}{N}X^T(Xw - y) = 0$$

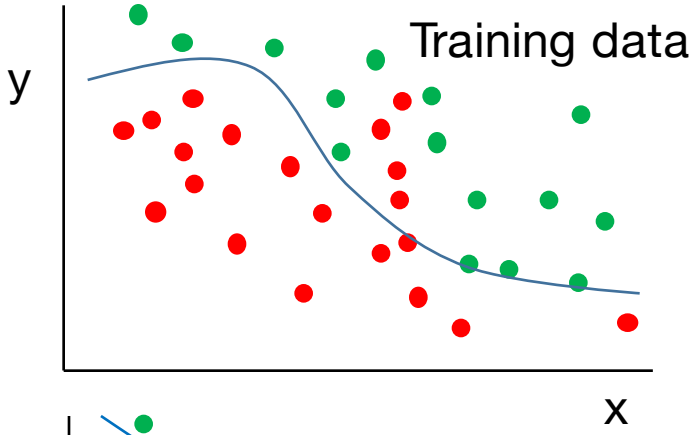
```
while previous_step_size > precision and iters < max_iters:
    prev_x = cur_x
    cur_x -= gamma * df(prev_x)
    previous_step_size = abs(cur_x - prev_x)
    iters+=1
```



$$f = W_1 x$$

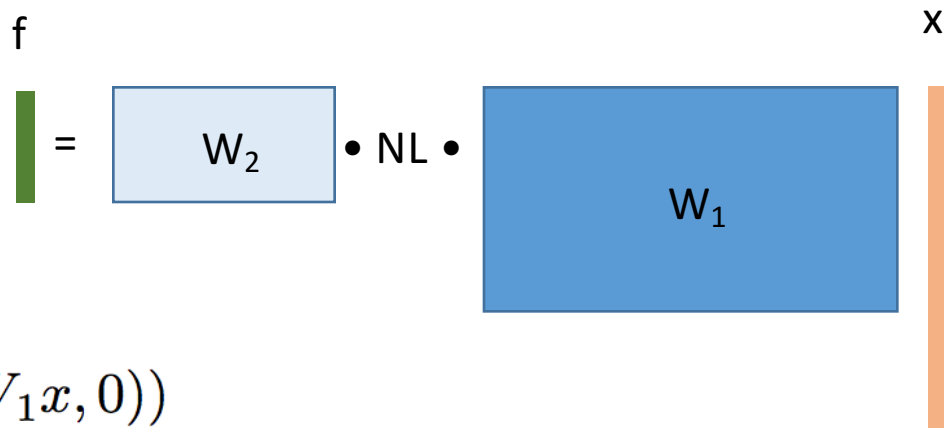
Learned *f:* not flexible

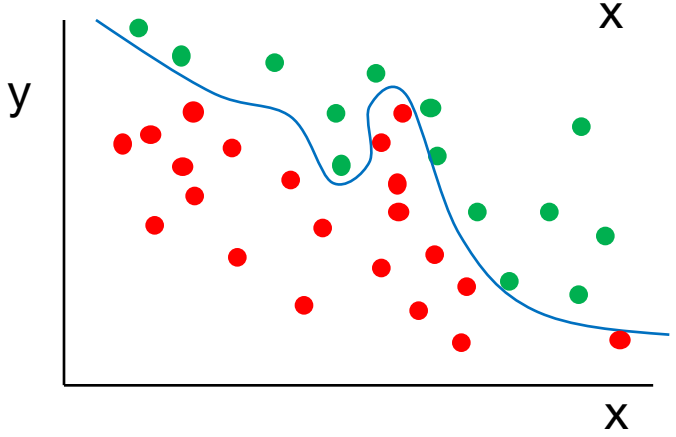




$$f = W_2 \max(W_1 x, 0)$$

Learned *f:* a bit flexible





$$f = W_3 \max(0, W_2 \max(W_1 x, 0))$$

Learned *f:* more flexible

Does it generalize???

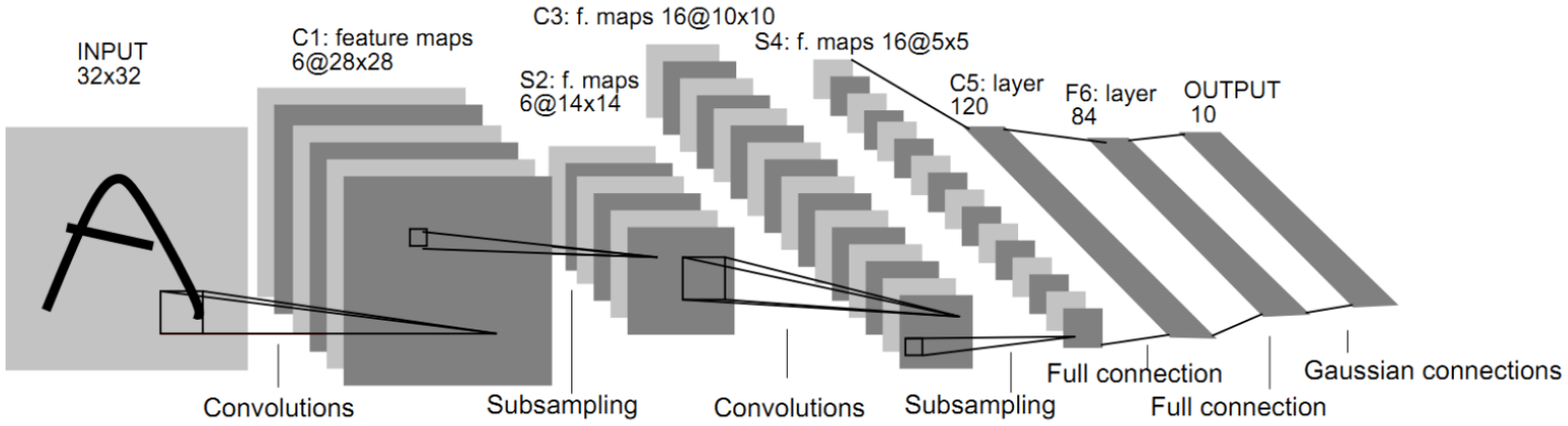
We can keep adding these “layers”…

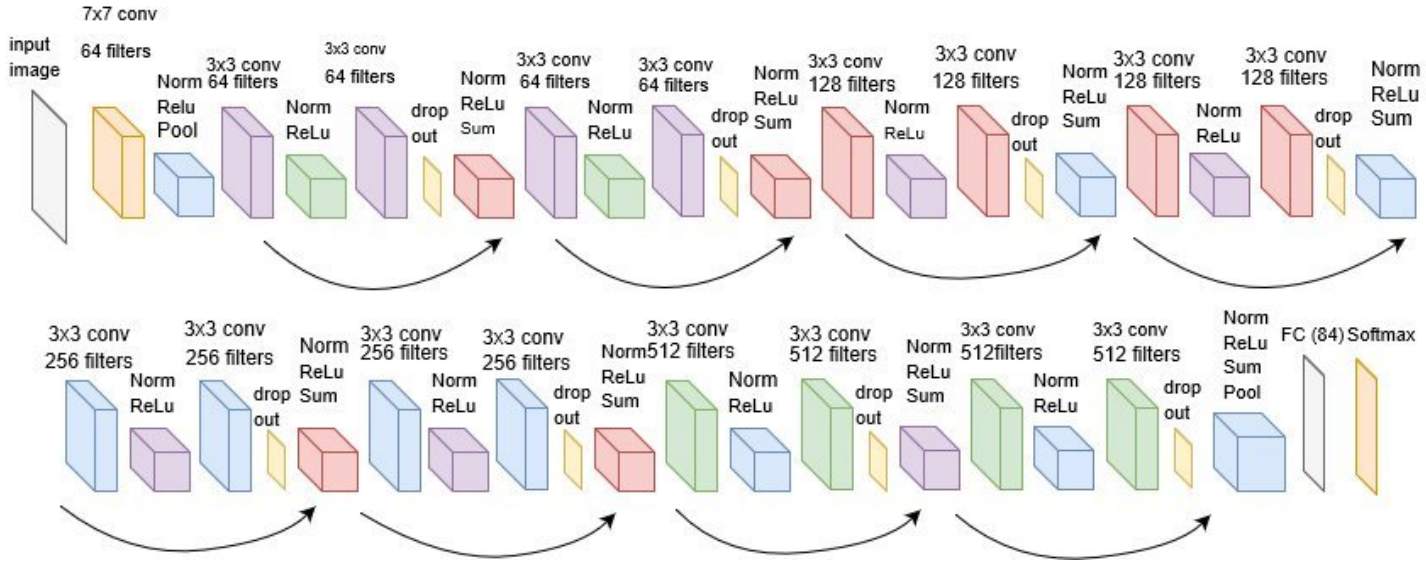
Before CNN's – understand two competing goals in machine learning

1. Can we make sure the in-sample error $L_{in}(y, f(x,W))$ is small enough?
   - Appropriate cost function
   - "complex enough" model

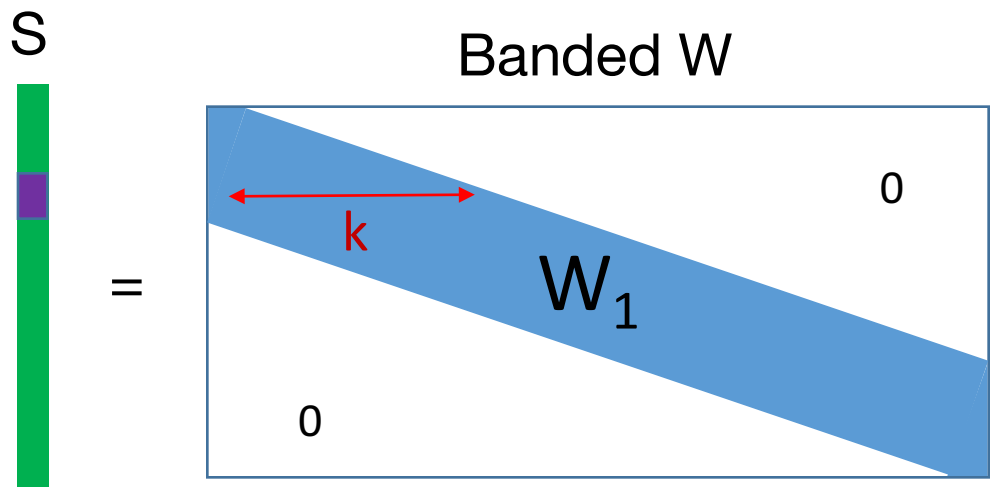
2. Can we make sure that $L_{out}(y, f(x,W))$ is close enough to $L_{in}(y, f(x,W))$?
   - Probabilistic analysis says yes!
   - $|L_{in} - L_{out}|$ bounded from above
   - Bound grows with model capacity (bad)
   - Bound shrinks with # of training examples (good)

Gets us to Convolutional Neural Networks





Why should we use these convolutions?



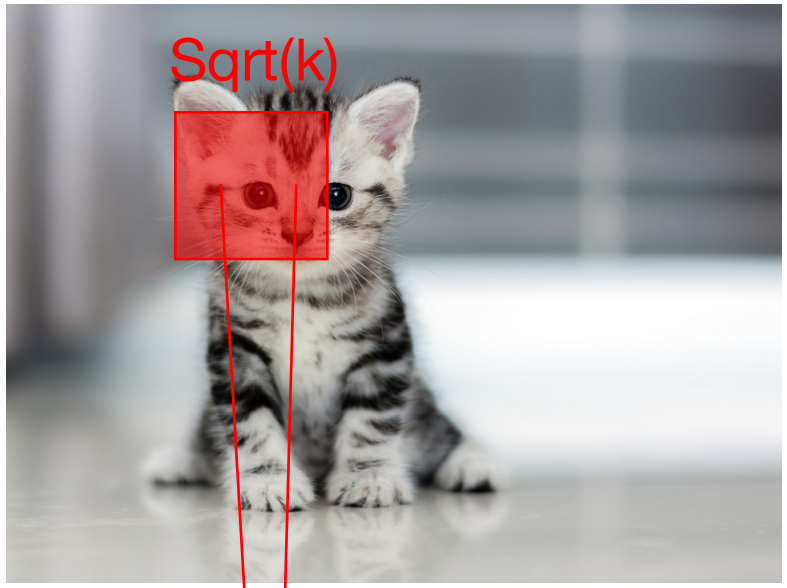
x = cat image

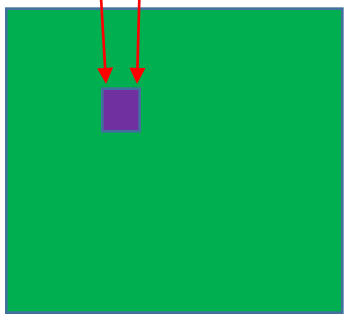
This type of matrix can dramatically reduce the number of weights that are used while still allowing *local* regions to mix:

Full matrix: $O(n^2)$

Banded matrix: k·O(n)

Image interpretation





Mix all the pixels in the red box, with associated weights, to form this entry of S

Important components of a CNN

CNN Architecture

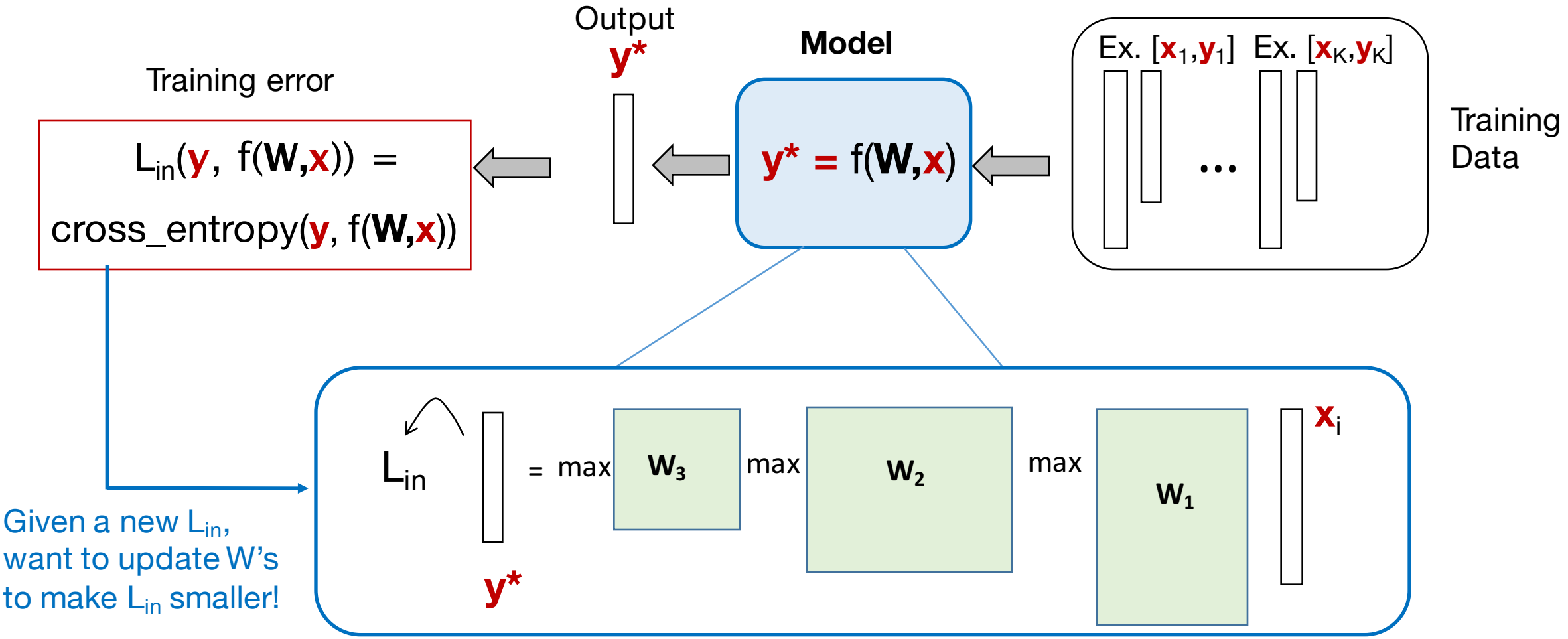
- CONV size, stride, pad, depth
- ReLU & other nonlinearities
- POOL methods
- # of layers, dimensions per layer
- Fully connected layers

Loss function & optimization

- Type of loss function
- Regularization
- Gradient descent method
- SGD batch and step size

Other specifics: Pre-processing**,** initialization**,** dropout, batch normalization, augmentation

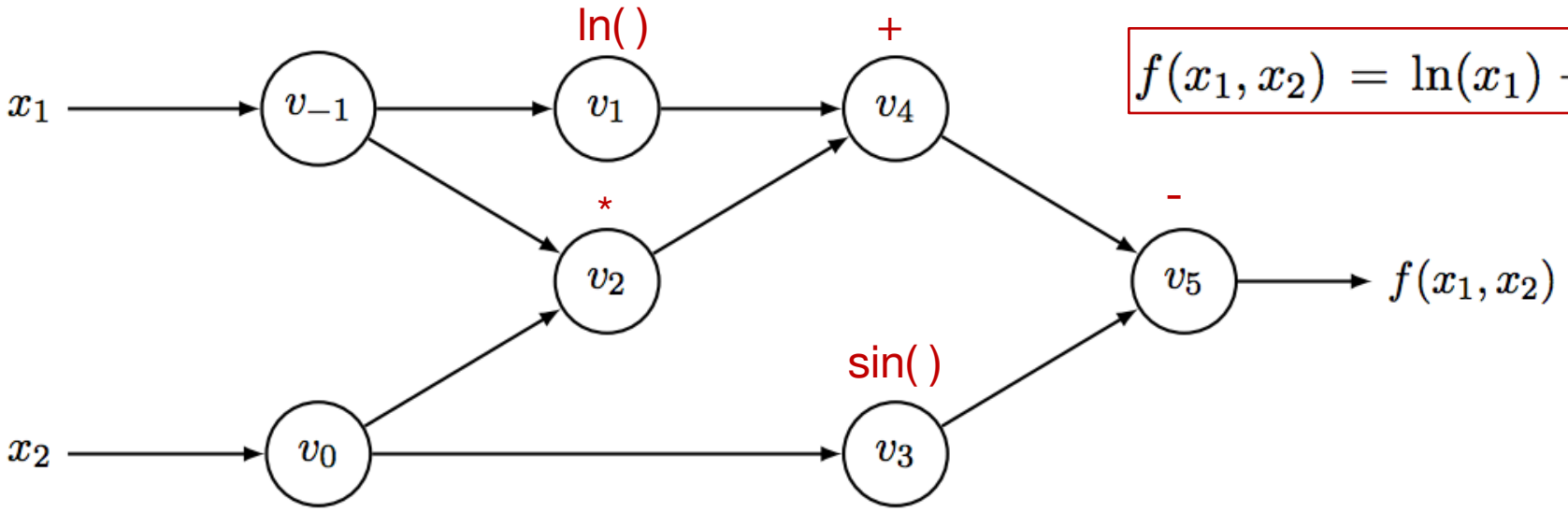
Our very basic convolutional neural network



Backwards pass uses new L_{in} to update **W**'s – backpropagation!

Automatic differentiation on computational graphs

A. Baydin et al., Automatic Differentiation in Machine Learning: a Survey



$$f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

To both determine f and find df/dx_i :

- Create graph of local operations
- Compute analytic (symbolic) gradient at each node (unit) in graph
- Use inter-relationships to establish final desired gradient, df/dx_1
  - Forward differentiation
  - Backwards differentiation = Backpropagation

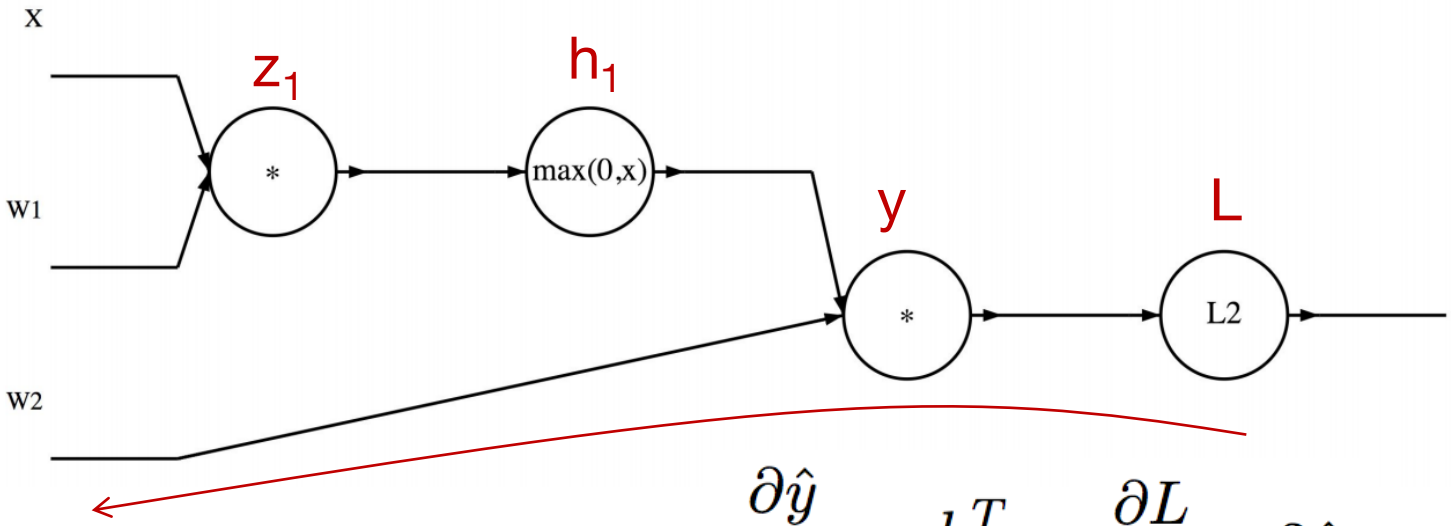
Let's go through an example:

$$L = \| \mathbf{W_2}\, \mathrm{ReLU}(\mathbf{W_1}\, \mathbf{X}) \|_2^2$$

(2-layer network with MSE where we neglect labels **y** for now)

$dL/d\mathbf{W}_1 = ?$ $\boxed{dL/d\mathbf{W}_2 = ?}$

$$z_1 = XW_1$$
$$h_1 = \mathrm{ReLU}(z_1)$$
$$\hat{y} = h_1 W_2$$
$$L = ||\hat{y}||_2^2$$

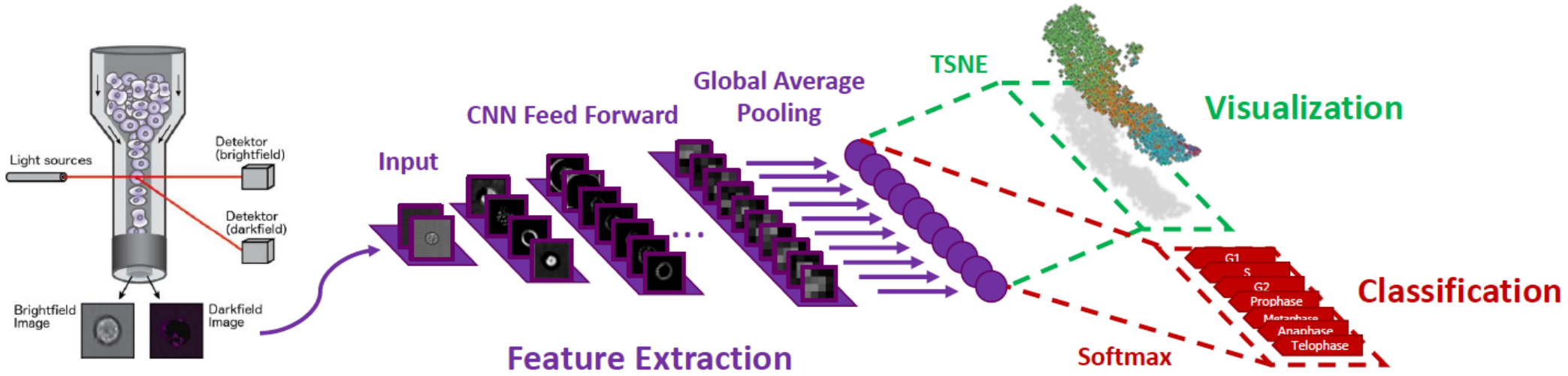


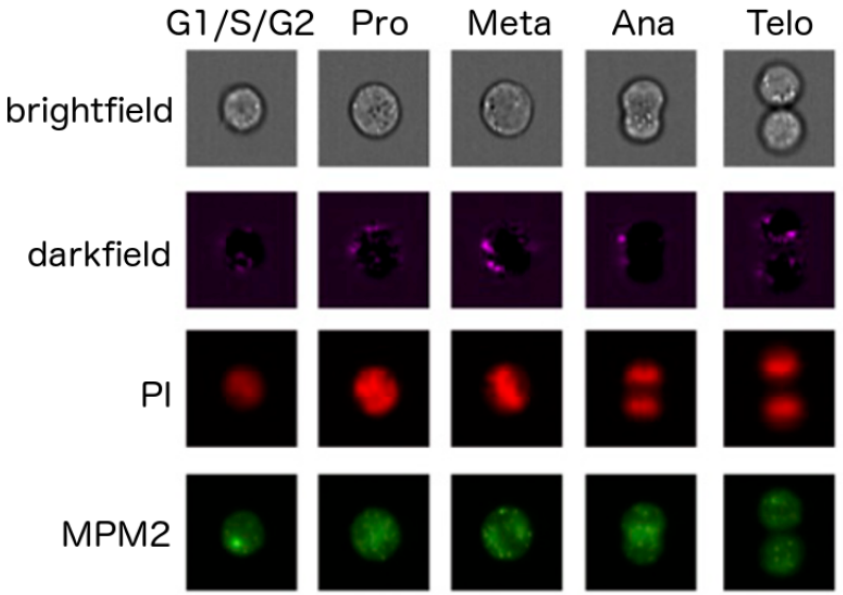
$$\frac{\partial \hat{y}}{\partial W_2} = h_1^T \qquad \frac{\partial L}{\partial \hat{y}} = 2\hat{y}$$

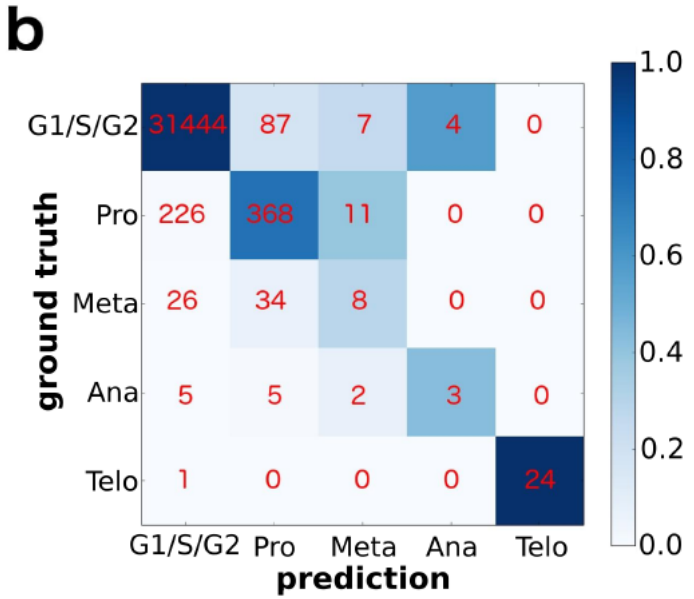
$$\boxed{\frac{\partial L}{\partial W_2} = \frac{\partial \hat{y}}{\partial W_2}\frac{\partial L}{\partial \hat{y}} = 2h_1^T \hat{y}}$$

CNNs for classification

P. Eulenberg et al., “Reconstructing cell cycle and disease progression using deep learning”

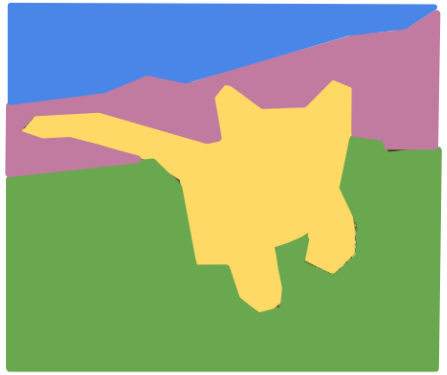






Other Computer Vision Tasks

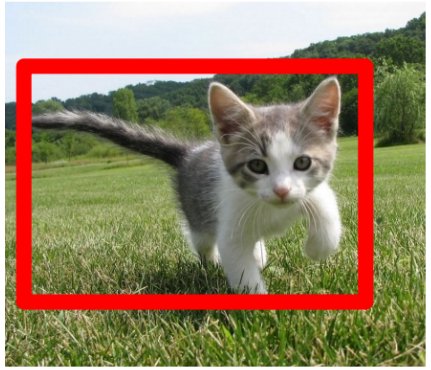
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

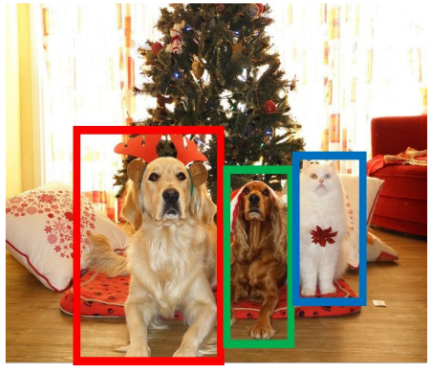
Classification + Localization



CAT

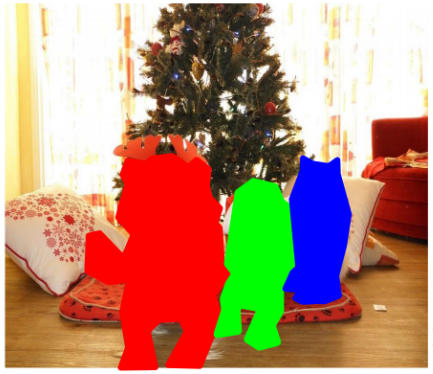
Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

Super-resolution

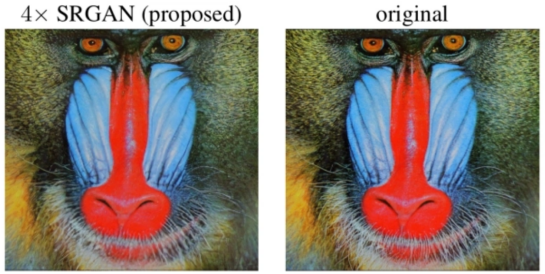


Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

Post-processing of your results: a few options at different stages

Options to examine your test data after processing:

- ROC curve, Precision-Recall
- Confusion matrix
- Sliding window visualization
- Layer visualizations
- Saliency maps etc.
- tSNE visualization

Bringing together physical and digital image representations

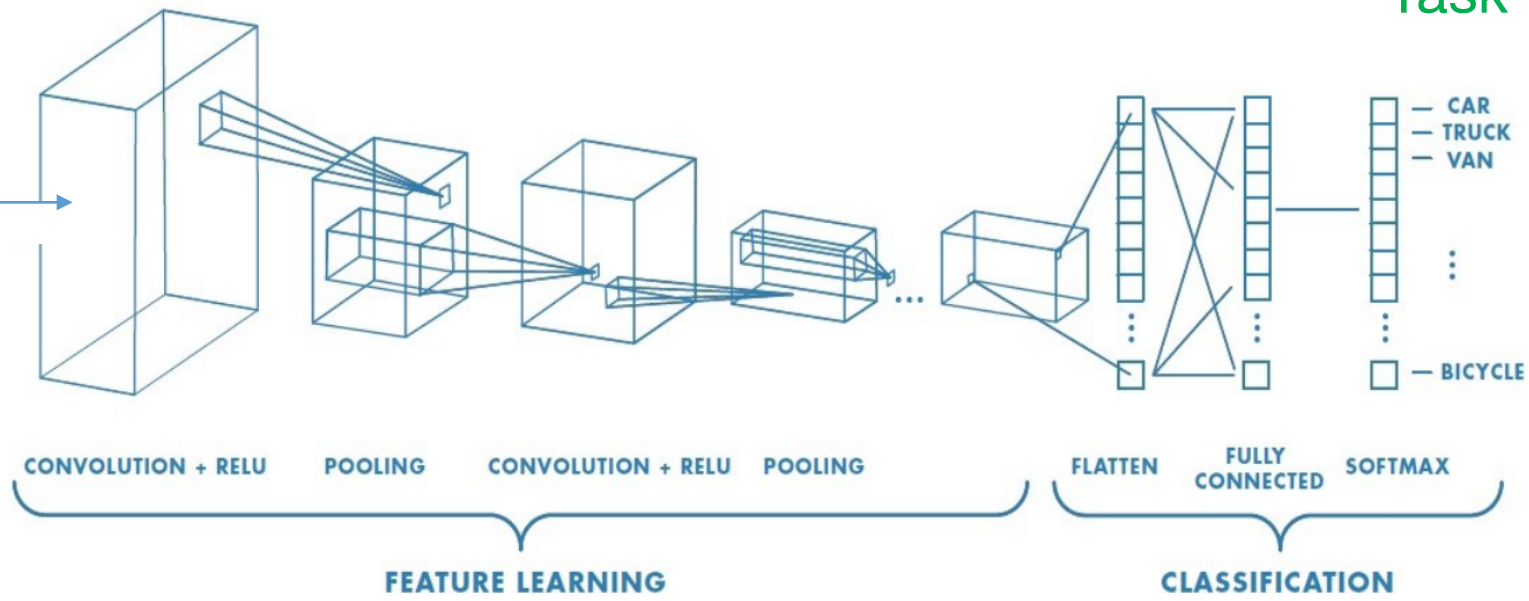
Physical Layers

Digital Layers

Task

Physical Function I_0 → f[ ] → Digitized I_s

$$I_s = f[I_0]$$



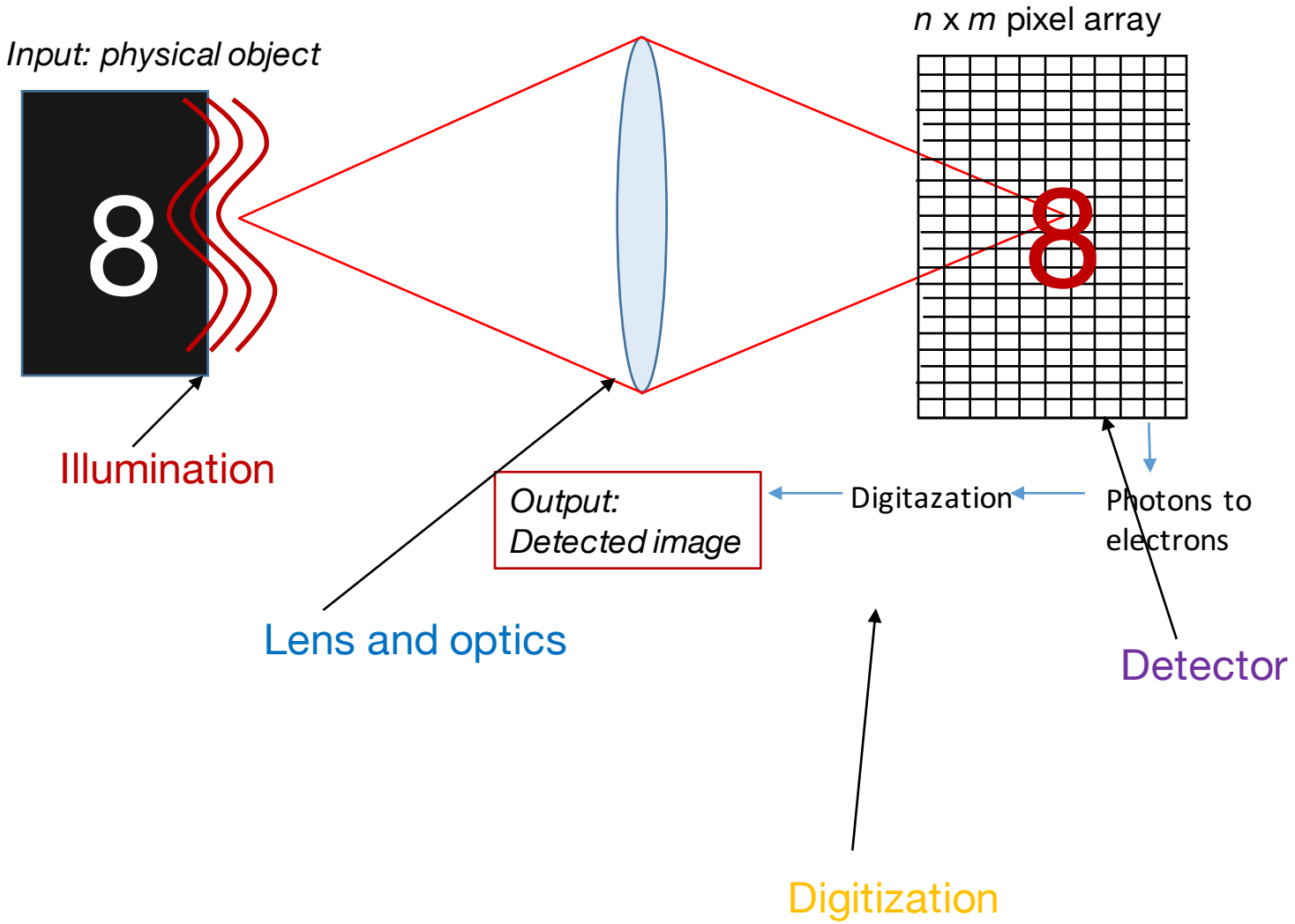
Digital layers

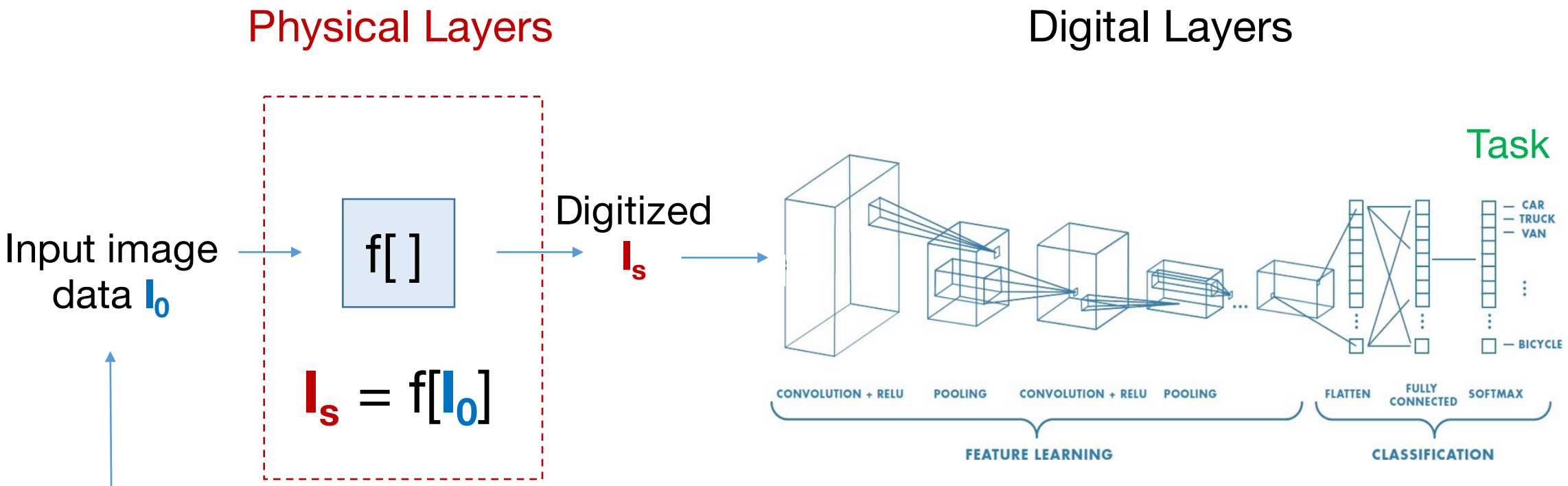
Physical layers

$$\text{Task} = \mathbf{W}_n \ldots \text{ReLU}[\mathbf{W}_1 \, \text{ReLU}[\mathbf{W}_0 \, f[I_0]]\ldots]$$

What physical parameters effect image formation?

- Illumination
  - Spatial pattern
  - Angle of incidence
  - Color, polarization
- Lens and optics
  - Position/orientation
  - Shape
  - Focus
  - Transparency
- Detector
  - Pixel size
  - Pixel shape & fill factor
  - Color filters
  - Other filters
- Digitization
  - E to P curves
  - Digitization schemes/thresholds
  - Data transmission, multiplexing
- Physical object





Q: Where and how should I implement my physical layer?

A: It depends on your data and implementation

- Situation #1: Fully simulated physical layers
- Situation #2: Experimentally-driven physical layers

Aside on simulated data: Combining forward and inverse solvers

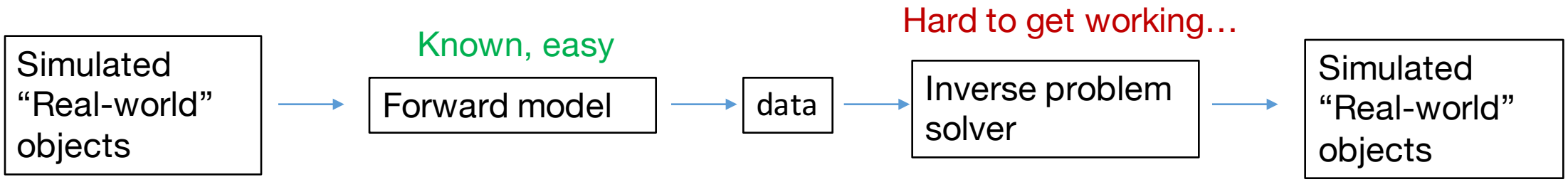
Forward problem: Start with the causes (objects in the real world) and compute the results (captured data)

(Typically easy)

Inverse problem: Start with the results (captured data) and infer about the causes (objects in the real world)

(Typically hard)

What I did in grad school to get ready for an experiment:



Classic examples: Inverse Radon Transform, US image reconstruction, image deblurring/denoising, diffraction tomography, phase retrieval, super-resolution (structured illumination, STORM/PALM), etc.

Aside on simulated data: Combining forward and inverse solvers

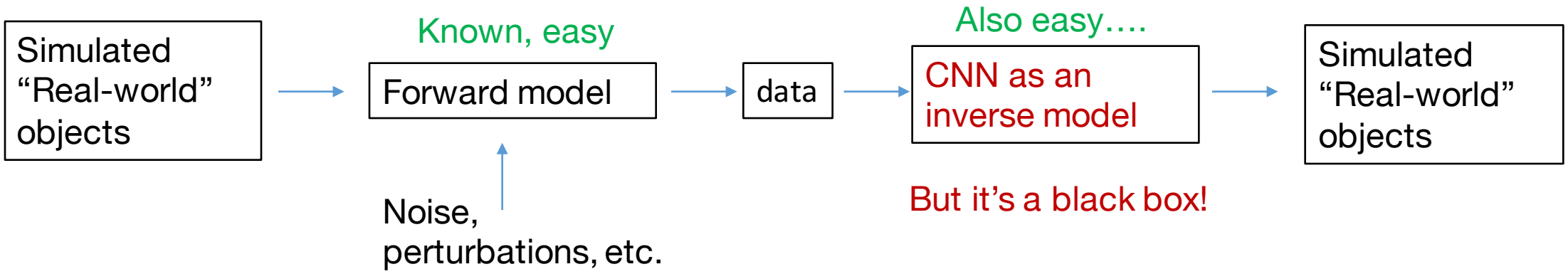
Forward problem: Start with the causes (objects in the real world) and compute the results (captured data)

(Typically easy)

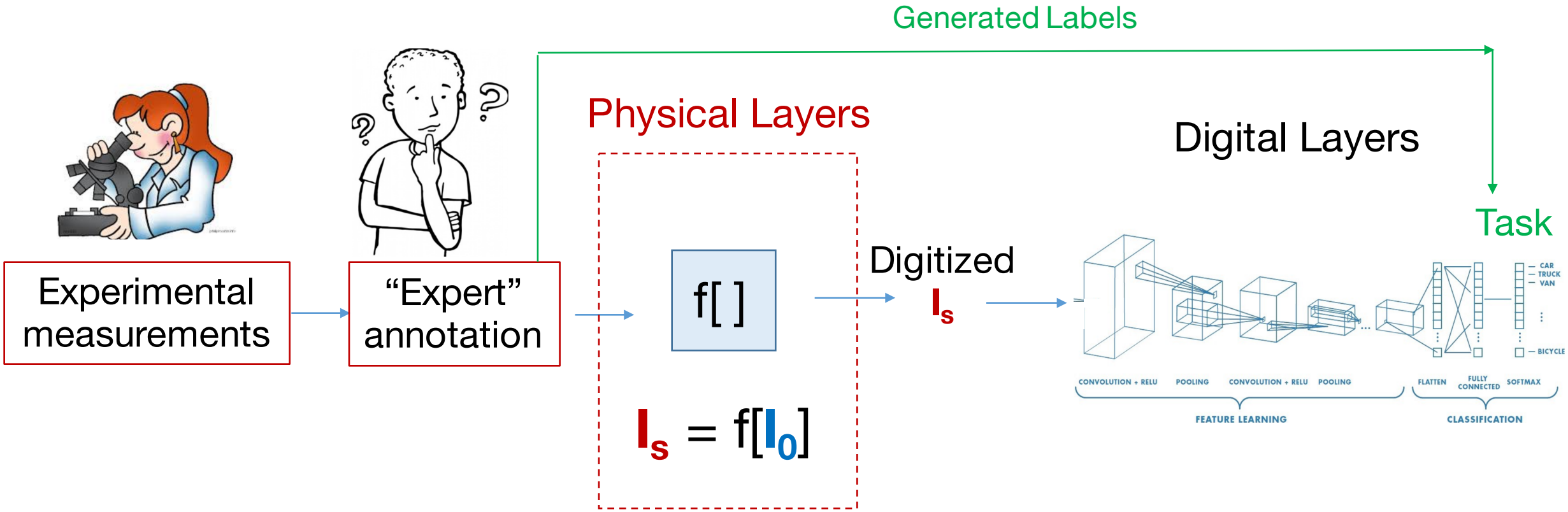
Inverse problem: Start with the results (captured data) and infer about the causes (objects in the real world)

(Typically hard)

What you can do now with CNN's:



Situation #2: Fully simulated physical layers



Pro's of experimental measurements: Don't need to worry about making your simulations match the setup! (HUGE)

Con's of experimental measurements: You'll need to label them, limited access to desired sample information, often need to exploit some fundamental physical property

Taxonomy of Generative Models

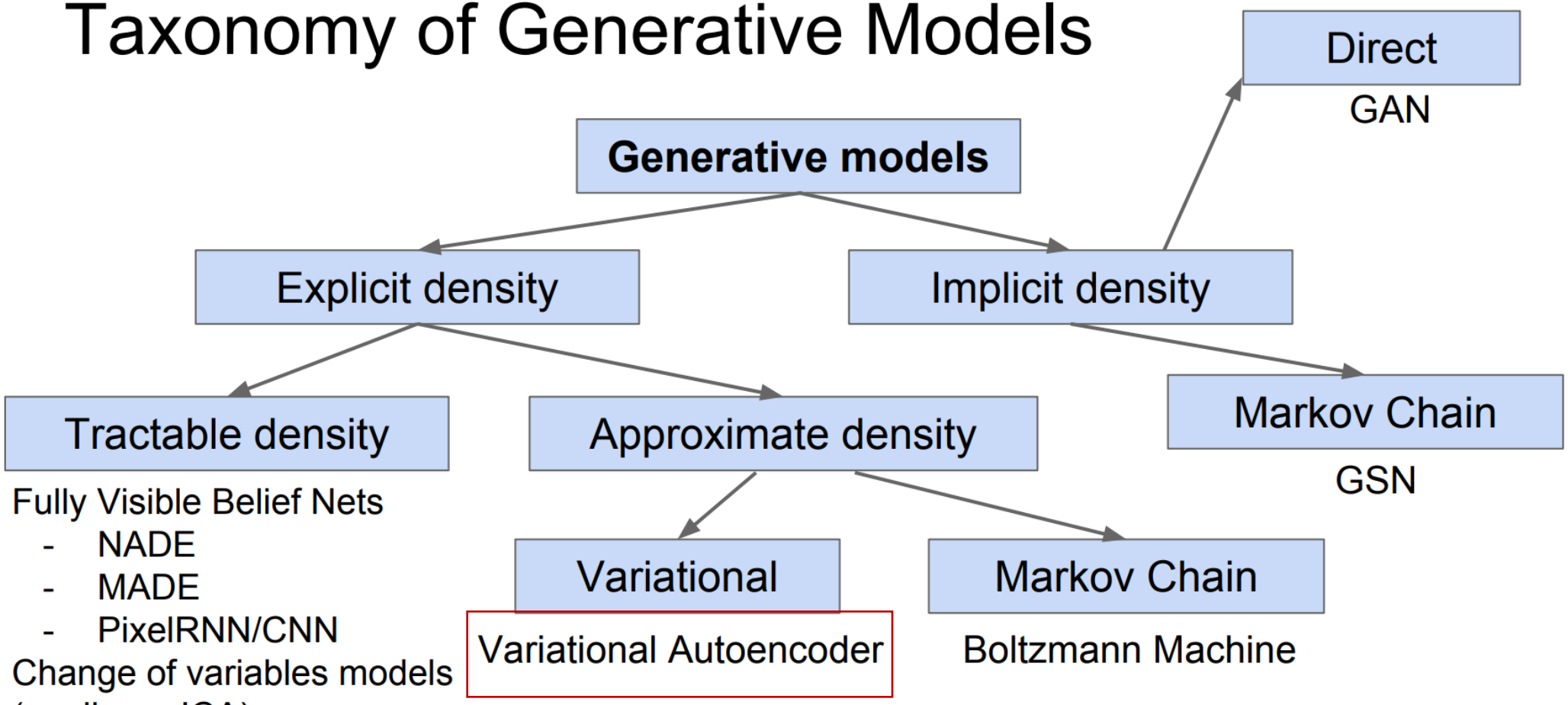
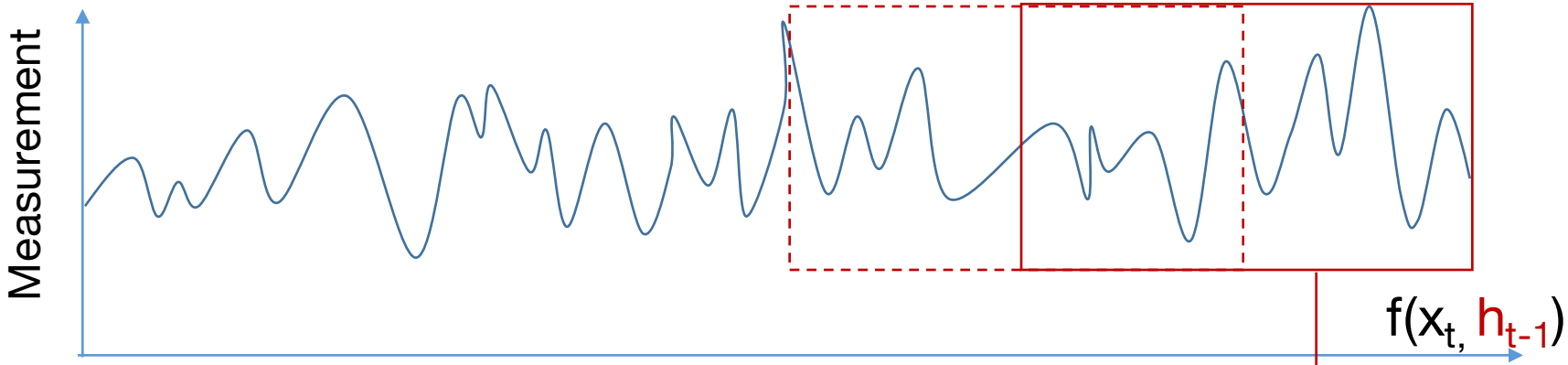
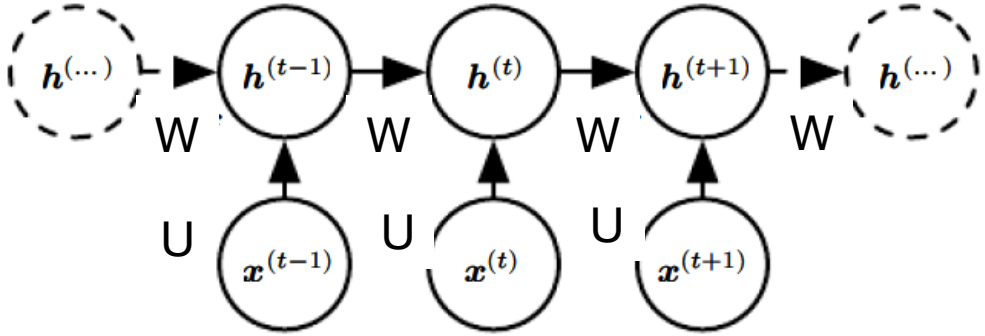


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Many-to-one recurrent neural network



State h_t



V

Output o_t

Backpropagate to minimize dL/dW, dL/DU

Simple network structure:

$$h_t = \text{ReLU}[\mathbf{W}h_{t-1} + \mathbf{U}x_t + b]$$

Label y → Loss function $L(o_t, y)$

The long short-term memory network

Forget gate:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right)$$

Internal state:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right)$$

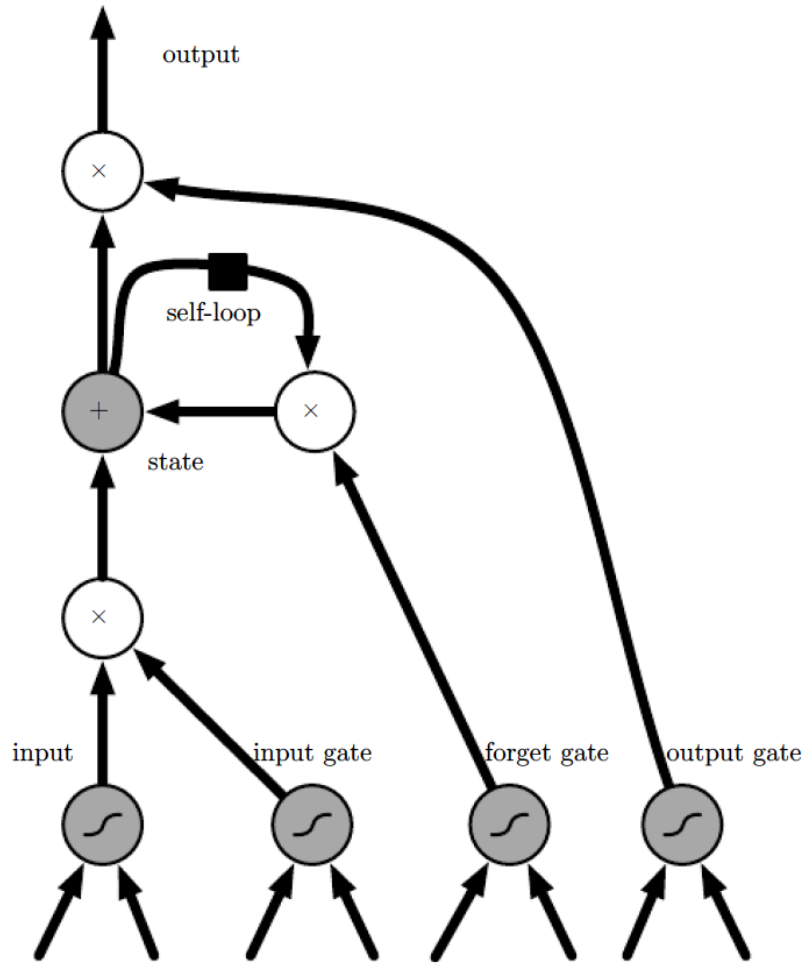
External input gate:

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right)$$

LSTM output:

$$h_i^{(t)} = \tanh\left(s_i^{(t)}\right) q_i^{(t)}$$

S. Hochreiter and J. Schmidhuber (1997)



Reinforcement learning - in a nutshell

- So far, we've looked at:
  1) Decisions from fixed images (classification, detection, segmentation)

CNN's

2) Decisions from time-sequence data (captioning as classification, etc.)

Decisions from images and time-sequence data (video classification, etc.)

RNN's

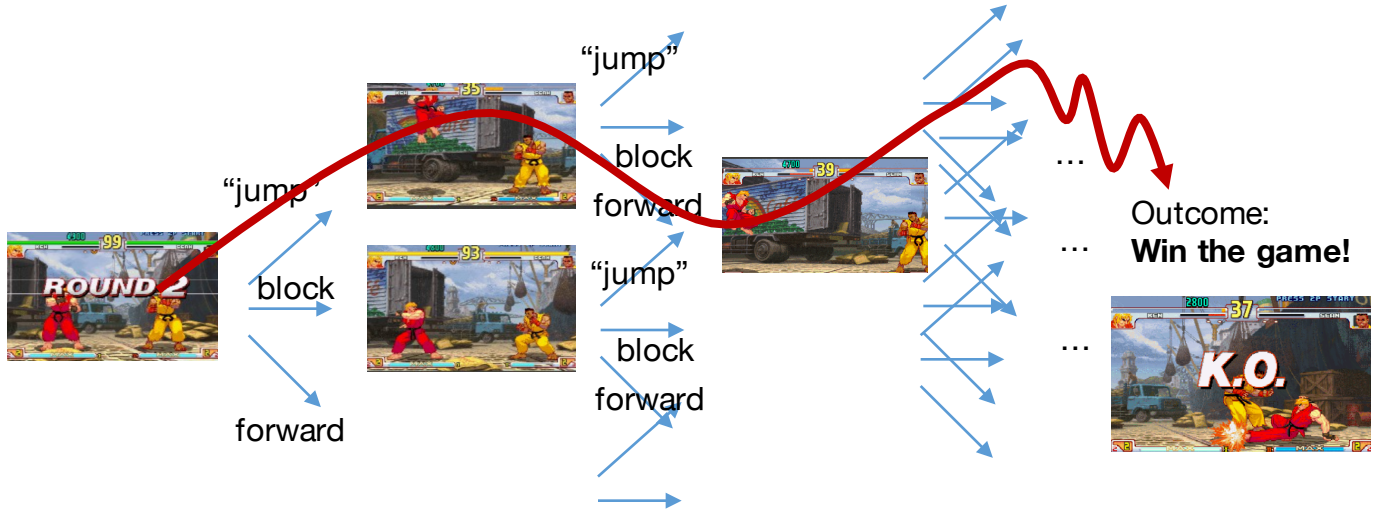
- Now, we're going to consider decisions for *dynamic data*
  - Most successful application: dynamic image data
    e.g.: video games, images of a Go game, car turning through obstacles

Reinforcement Learning

Supervised ML

Outcome:
Cell type B

Reinforcement learning



- Fixed image sequence
- Goal: match to known label
  (large labeled dataset needed)
- Output: label
- Examines all data

- Dynamic image sequence
- Goal: get to known desired outcome
  (no labels needed, really…)
- Output: sequence of actions
- Not possible to examine *all* data

The Machine Learning in Imaging Ethics Questionnaire

Situation 4: In 10 years, you go up to a modified microscope, “the Tissue Scanner 3000”, that has a number of fancy lenses and lights. As a machine learning expert by now, you’re aware that this microscope is optimized for looking at skin lesions. It performs a scan with a particular lighting configuration and reports a score of 98% confident that the lesion is benign, allowing you to look through other examples. It asks If you’d like another scan for additional confidence or a different outcome, at which point the illumination changes and it does some more scanning and reports a 99% confidence level. You can continue with another scan, but…

Are you now comfortable with leaving the office?

Yes:

No: