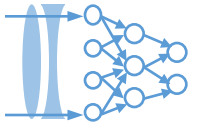


Lecture 24: Review of Machine Learning + Imaging

Machine Learning and Imaging

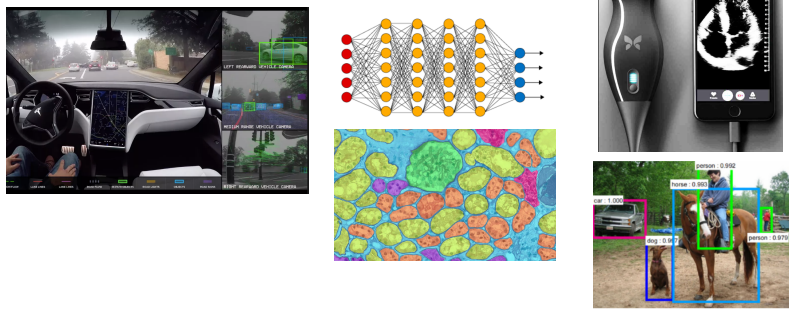
BME 590L
Roarke Horstmeyer

What was this class about?

deep imaging

Where your final project is aimed

*Computer-*centered hardware + software



Human-centered hardware design

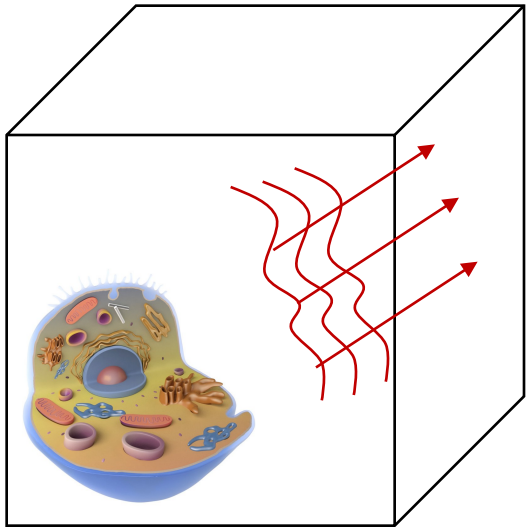
Computer-centered software design

ML+Imaging pipeline introduction

deep imaging

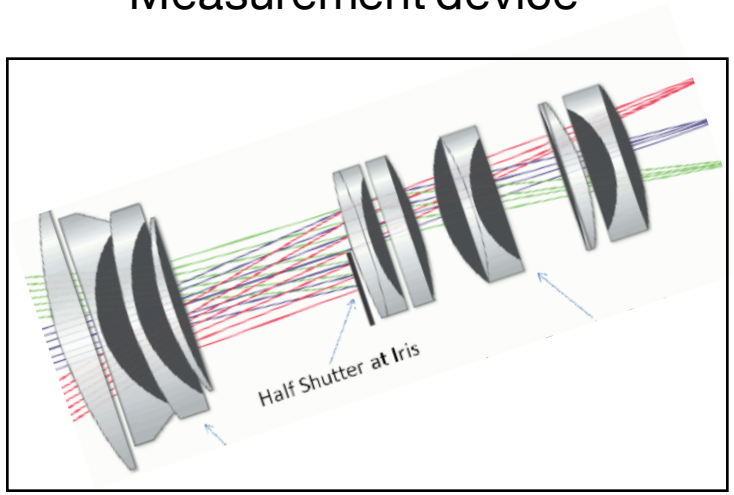
Machine Learning

Real World



Continuous complex fields

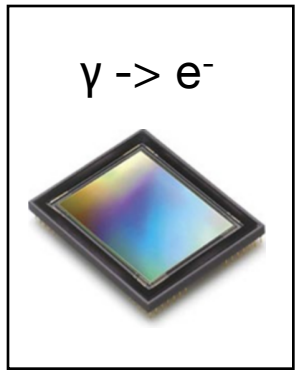
Measurement device



Black box transformations

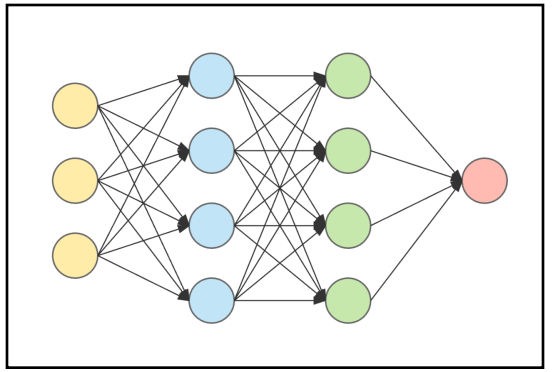
- Convolution
- Fourier Transform

Digitization



Sampling Theorem

Discrete math & Linear algebra



Optimization

Linear classification

Logistic classifier

Neural networks

Convolutional NN's

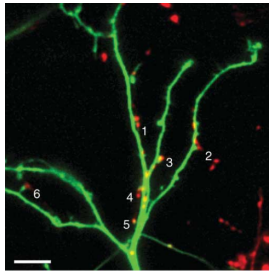
Final project: combine

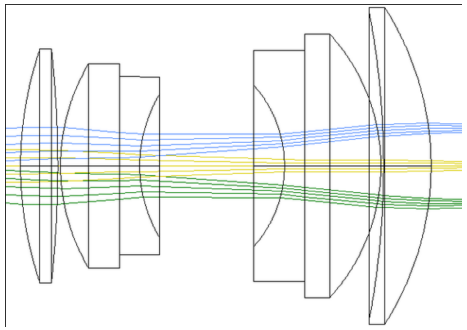
Summarize with simple models

Use Tensorflow

Physical models for light propagation to sensor

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



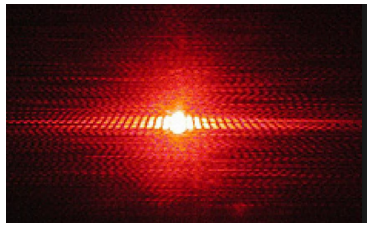


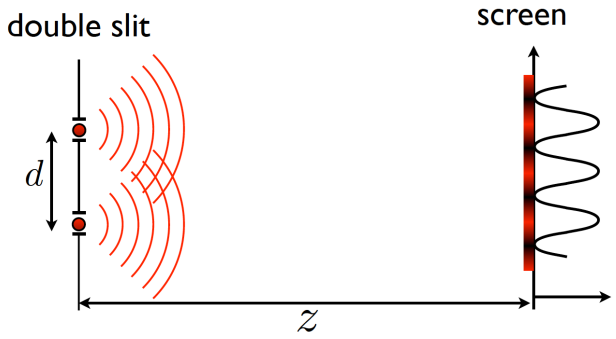
- Real, non-negative

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

- 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

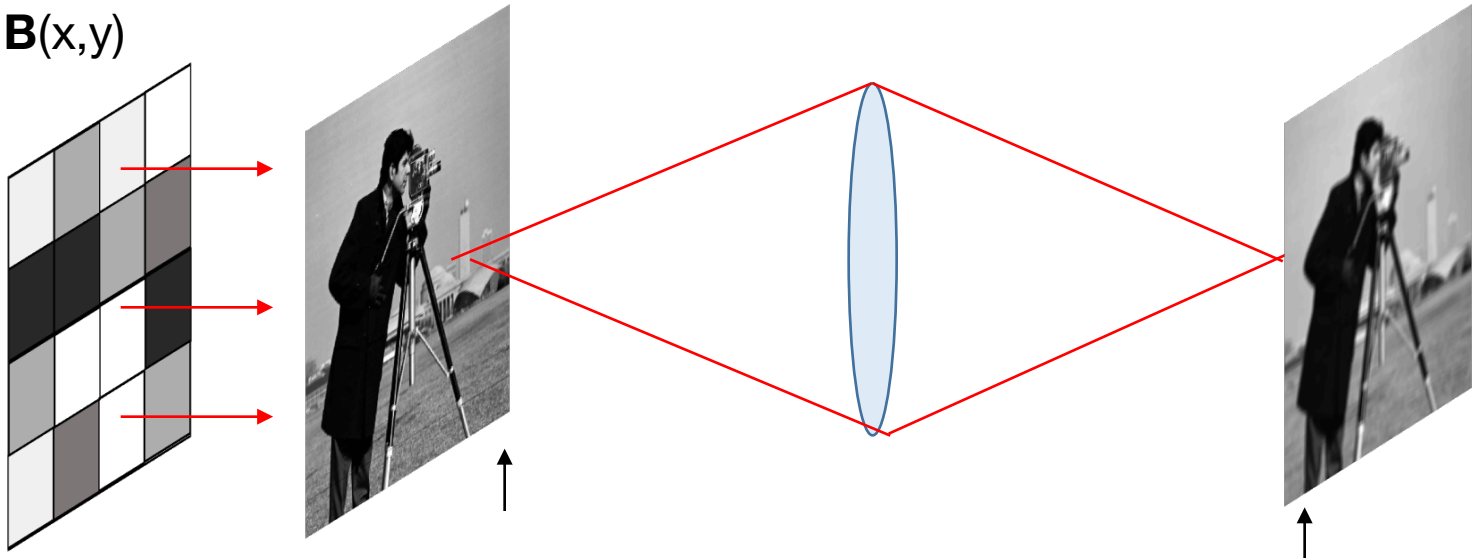
- All quantities are real, and non-negative

Object absorption:

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

Illumination brightness:

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



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

multiplication

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

convolution

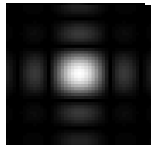
We can also add in some lens blur

Lenses blur and rescale images:

Input intensity

Convolution filter h

*



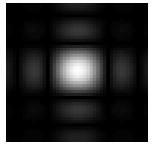
“Incoherent point-spread function”

=

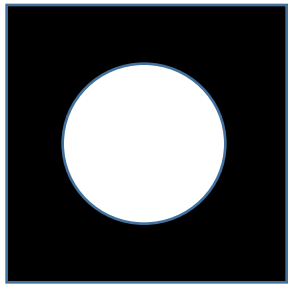
Output intensity

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

h(x,y)



$A(f_x,f_y)$

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

Mathematical model of for incoherent image formation

deep imaging

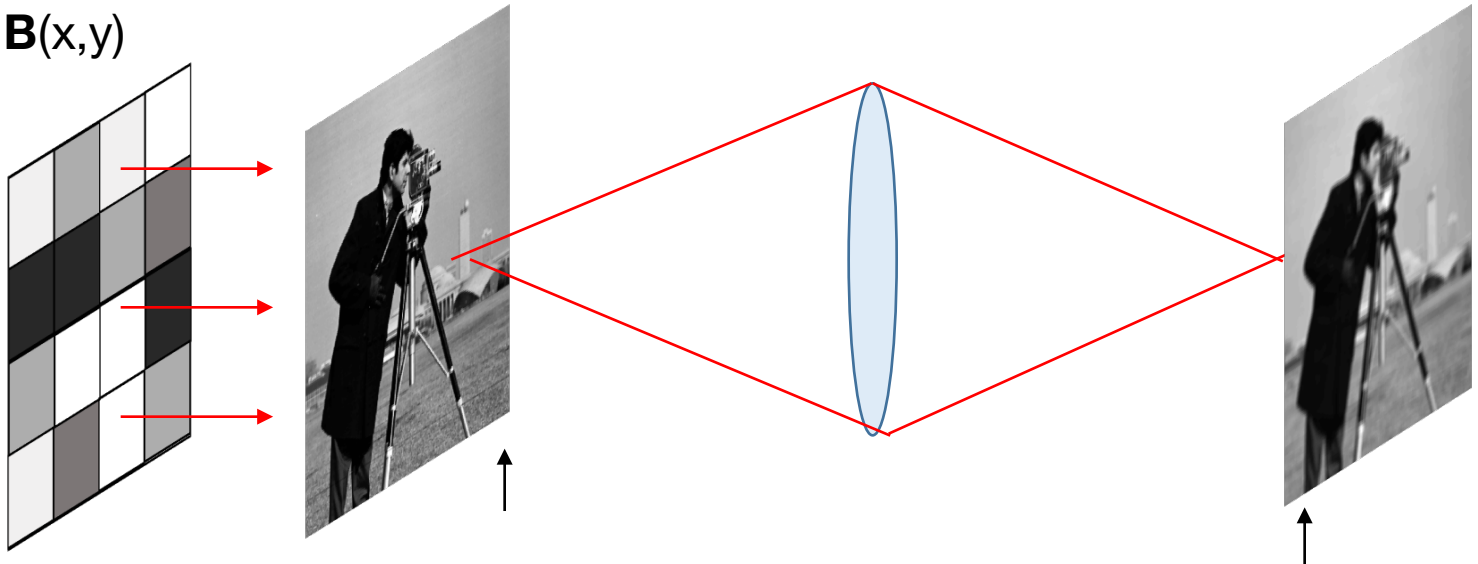
- All quantities are real, and non-negative

Object absorption:

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

Illumination brightness:

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



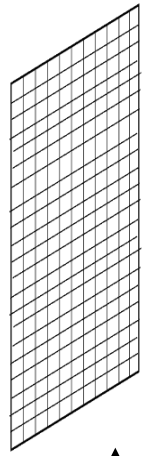
$\mathbf{B}\ \mathbf{S_0}$

multiplication

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

convolution

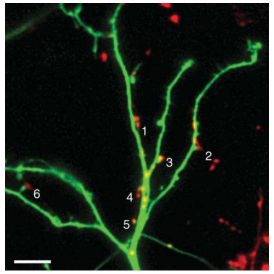
Photons (intensity) hits detector

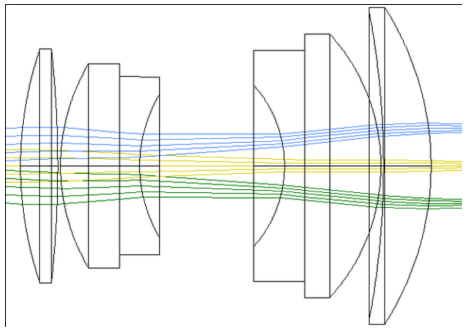


$$\mathbf{I_s} = \mathbf{H}\ \mathbf{B}\ \mathbf{S_0}$$

Physical models for light propagation to sensor

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



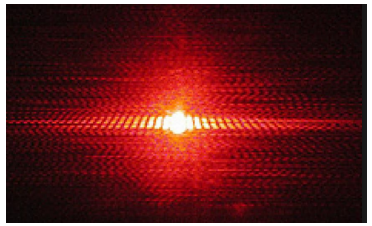


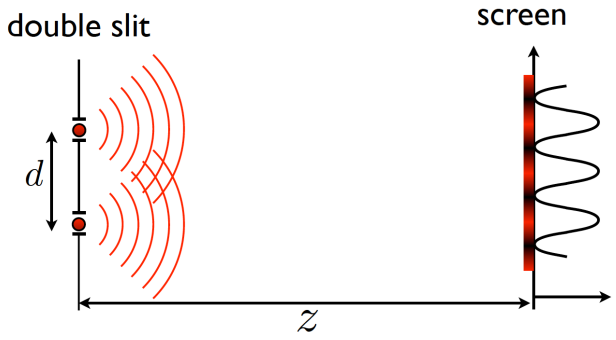
- Real, non-negative

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

- 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:

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

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,

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

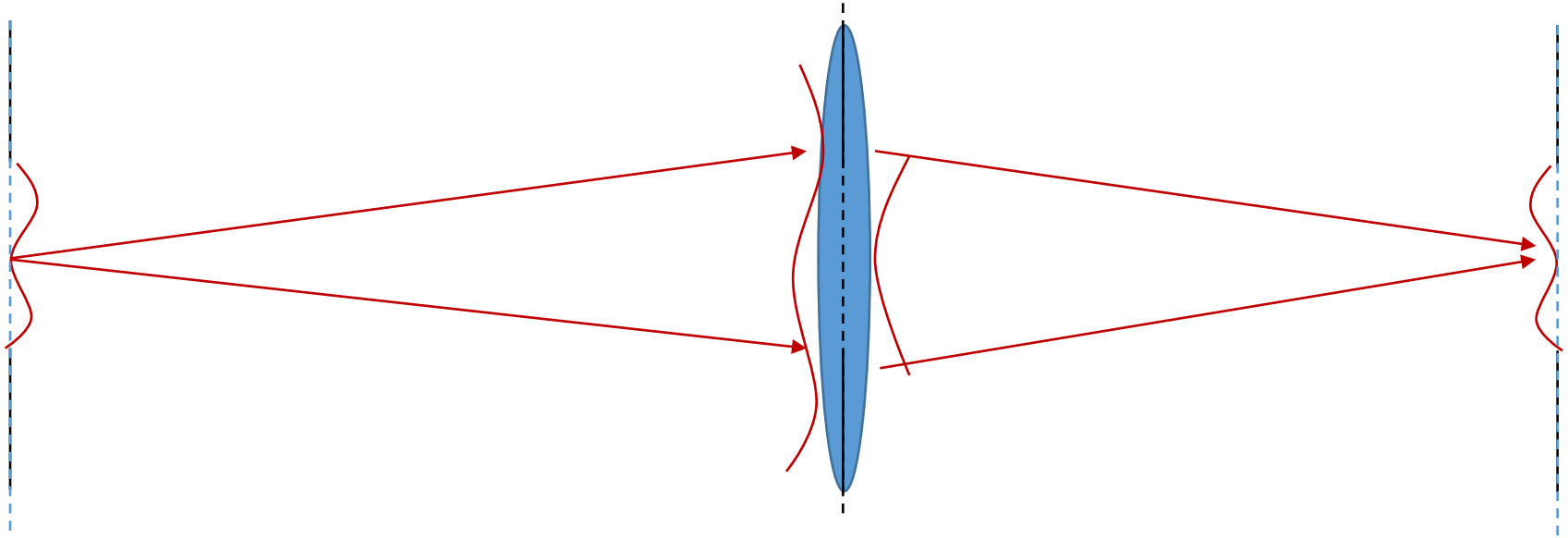
$$C = \frac{e^{ikz}}{i\lambda z}e^{\frac{ik}{2z}(x^2+y^2)}$$

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

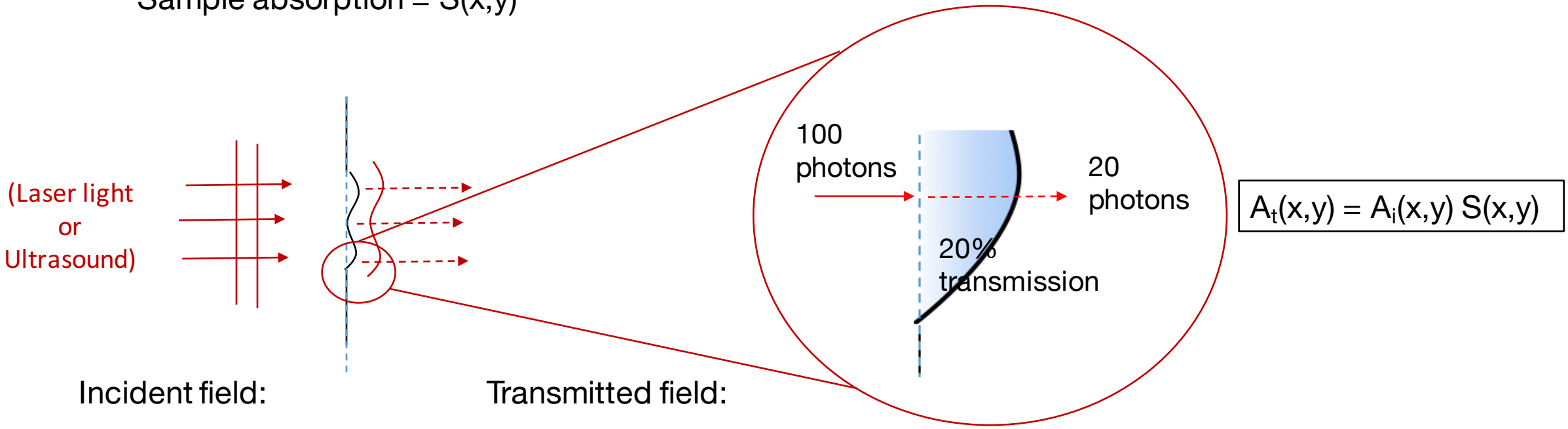
Mathematical model of for coherent image formation

deep imaging

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

Sample absorption = S(x,y)

Point #1: Amplitudes behave just like before



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

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

deep imaging

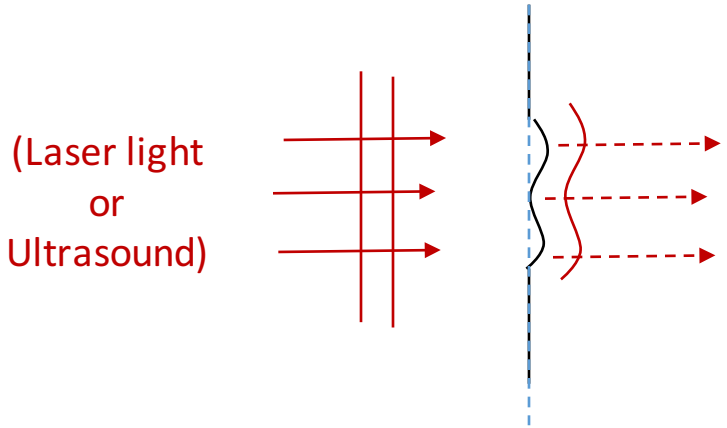
- Pretty much the same thing, but now we have an amplitude and a complex phase

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:

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

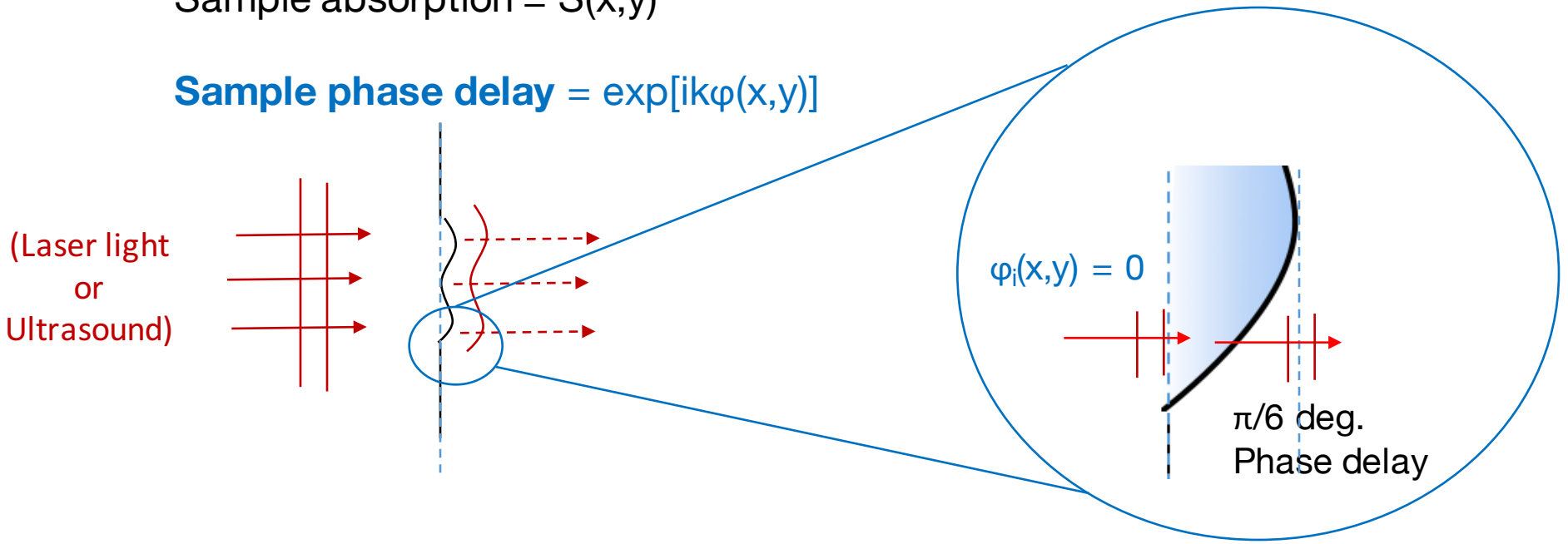
Mathematical model of for coherent image formation

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

Sample absorption = S(x,y)

Sample phase delay = exp[ikφ(x,y)]

New: complex phase delay



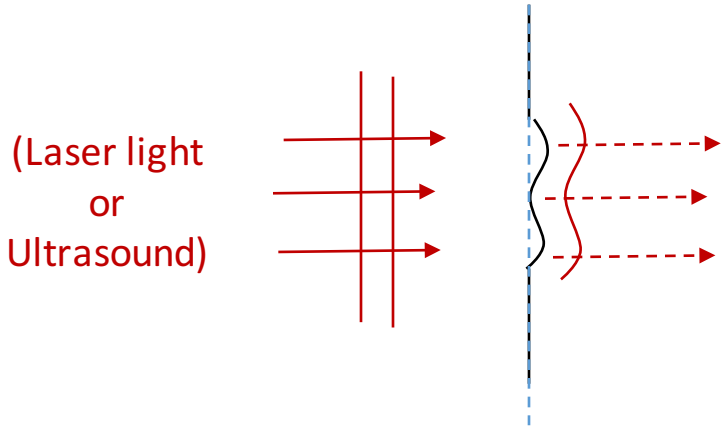
Mathematical model of for coherent image formation

deep imaging

- 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

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

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

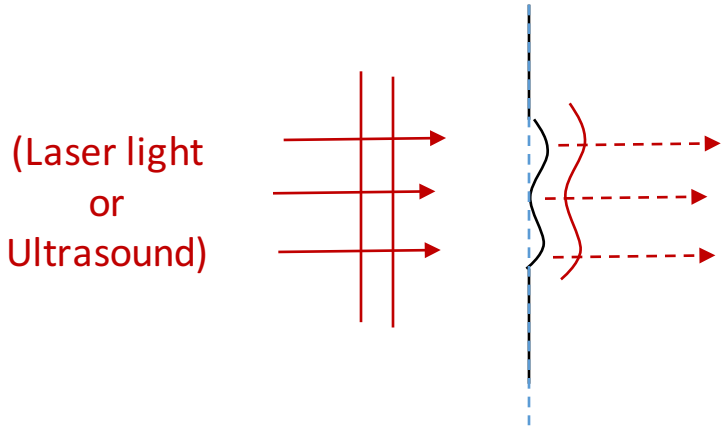
Mathematical model of for coherent image formation

deep imaging

- 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)]

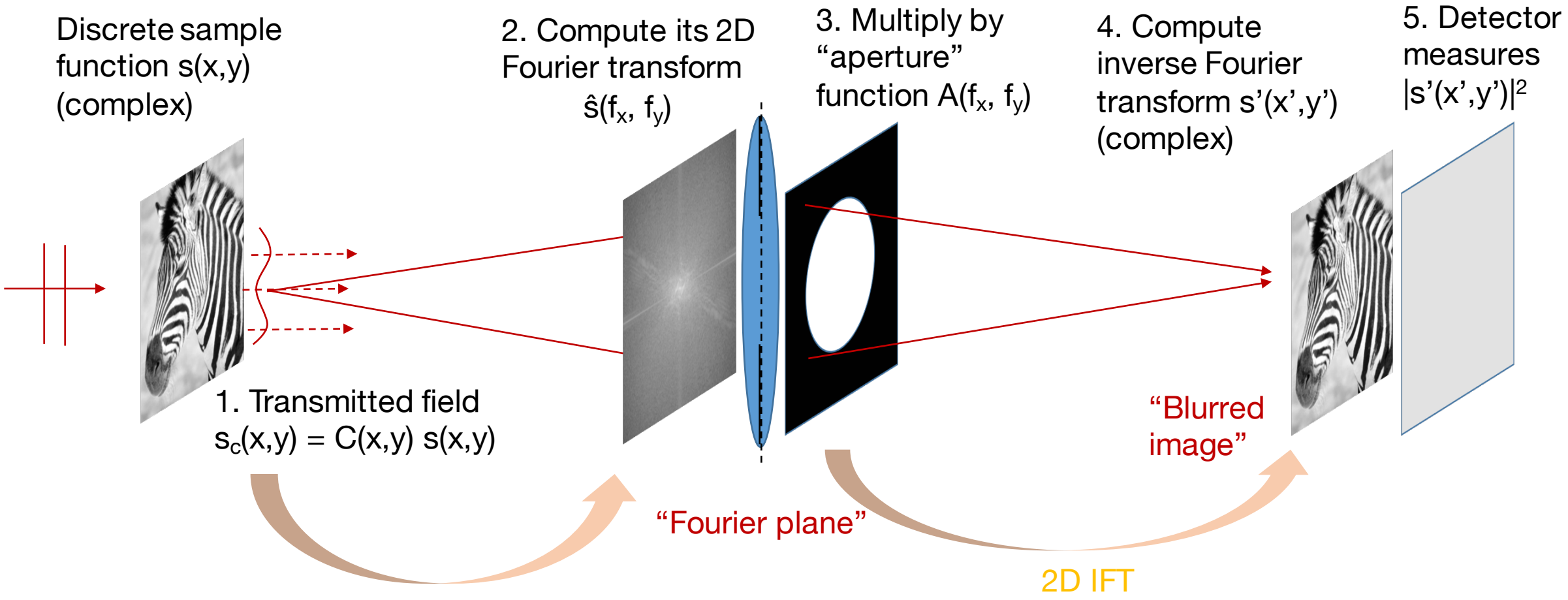


Conclusion:

Transmitted field = incident field x complex sample :

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

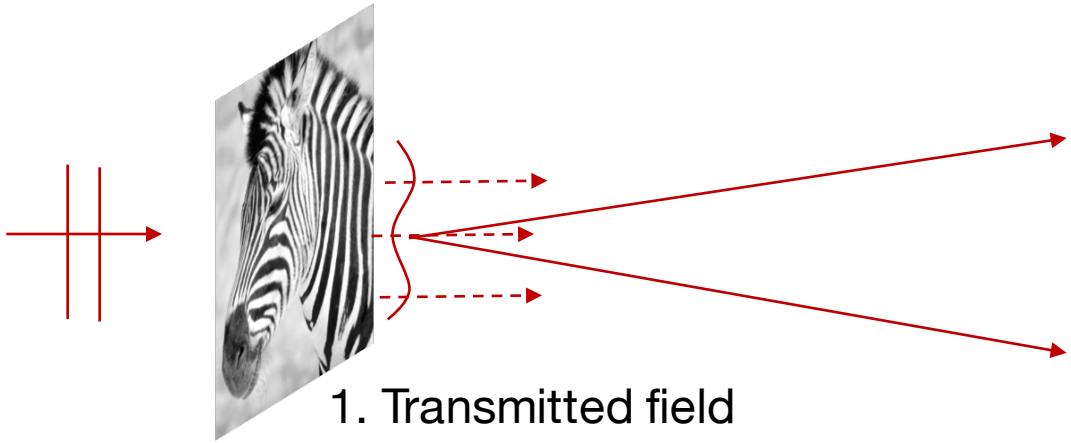
Model of image formation for wave optics (coherent light):



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

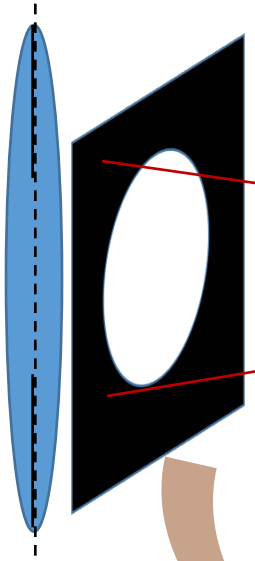
Model of image formation for wave optics (coherent light):

Discrete sample function s(x,y) (complex)



1. Transmitted field
$s_c(x,y) = C(x,y)\, s(x,y)$

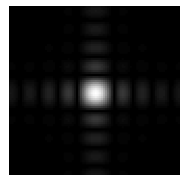
2. "aperture" function $A(f_x, f_y)$



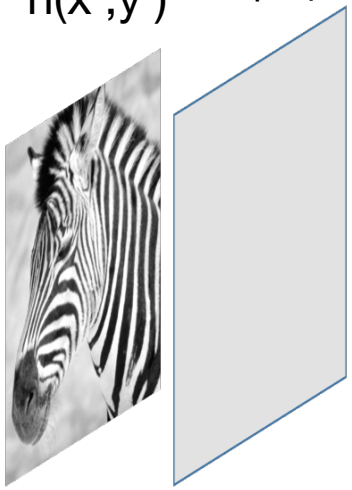
3. Compute blur function
$h(x,y) = F[A(f_x, f_y)]$

2D IFT

4. Blur image:
$s' = s_c(x',y') * h(x',y')$

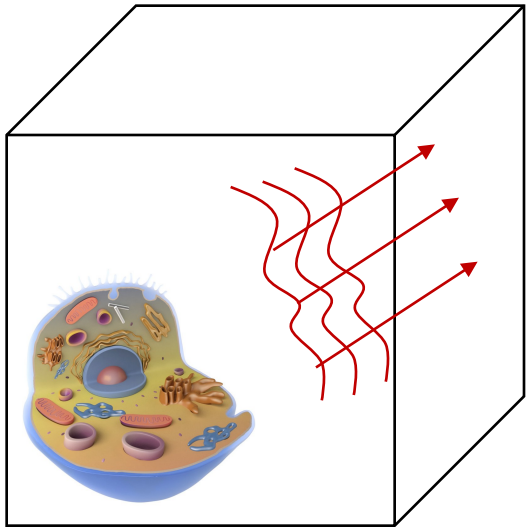


5. Detector measures
$|s'(x',y')|^2$



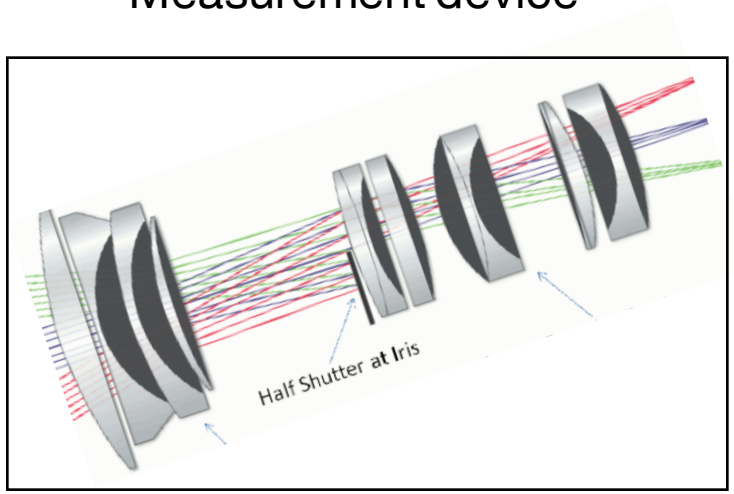
Model #2: $I_c(x,y) = |h * Cs|^2$

Real World



Continuous complex fields

Measurement device



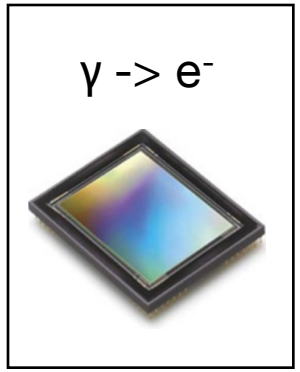
Black box transformations

- Convolution
- Fourier Transform

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

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

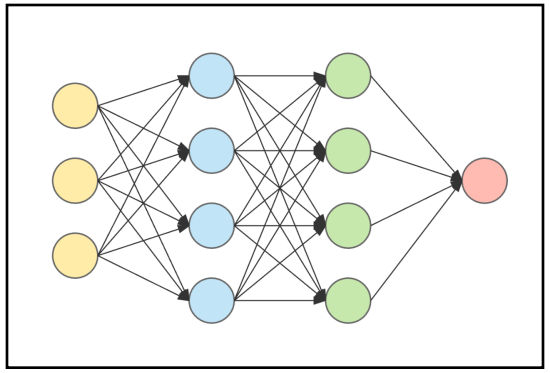
Digitization



Sampling Theorem

Discrete math & Linear algebra

Machine Learning



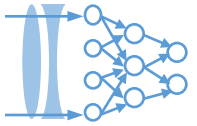
Optimization

Linear classification

Logistic classifier

Neural networks

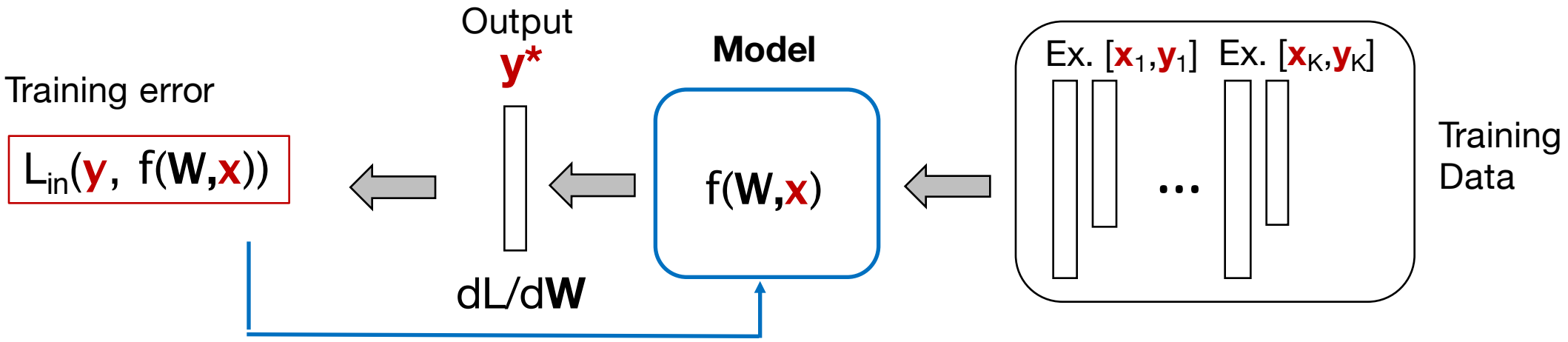
Convolutional NN's



Summary of machine learning pipeline:

deep imaging

1. Network Training

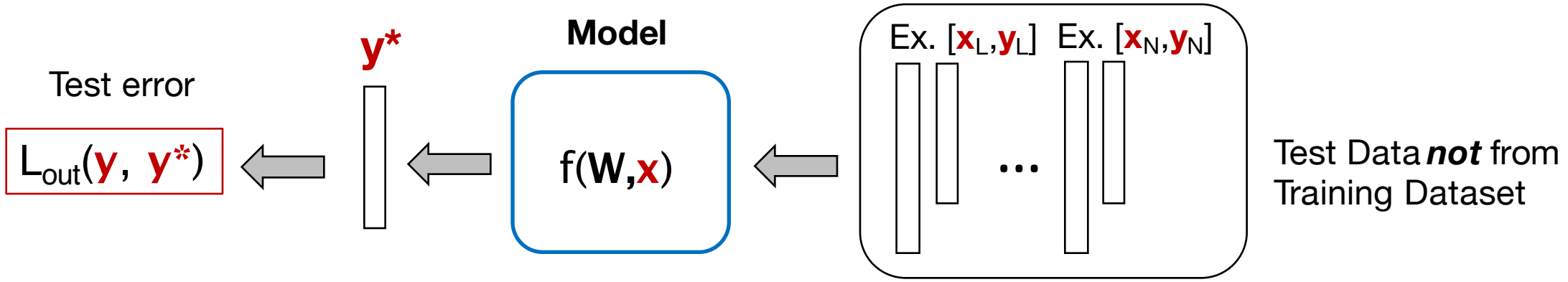


What we need for network training:

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

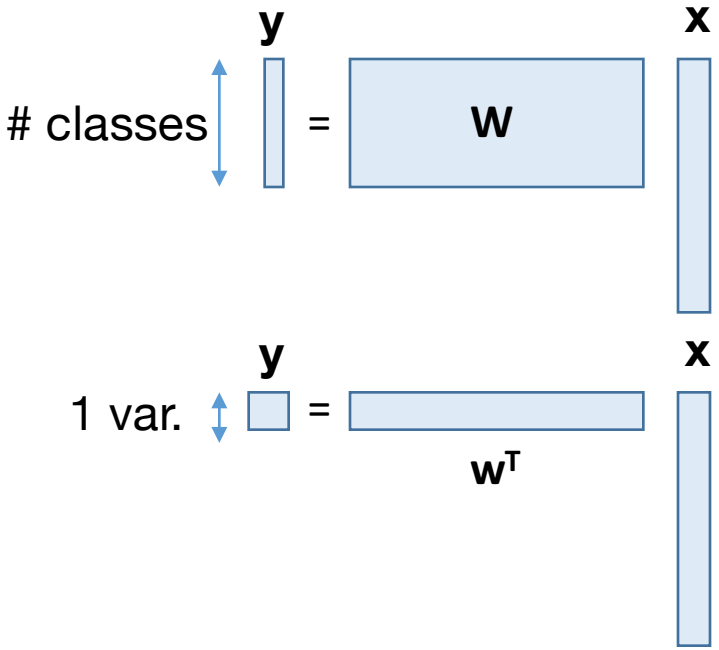
Let's start with a simpler approach: linear regression

deep imaging

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

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

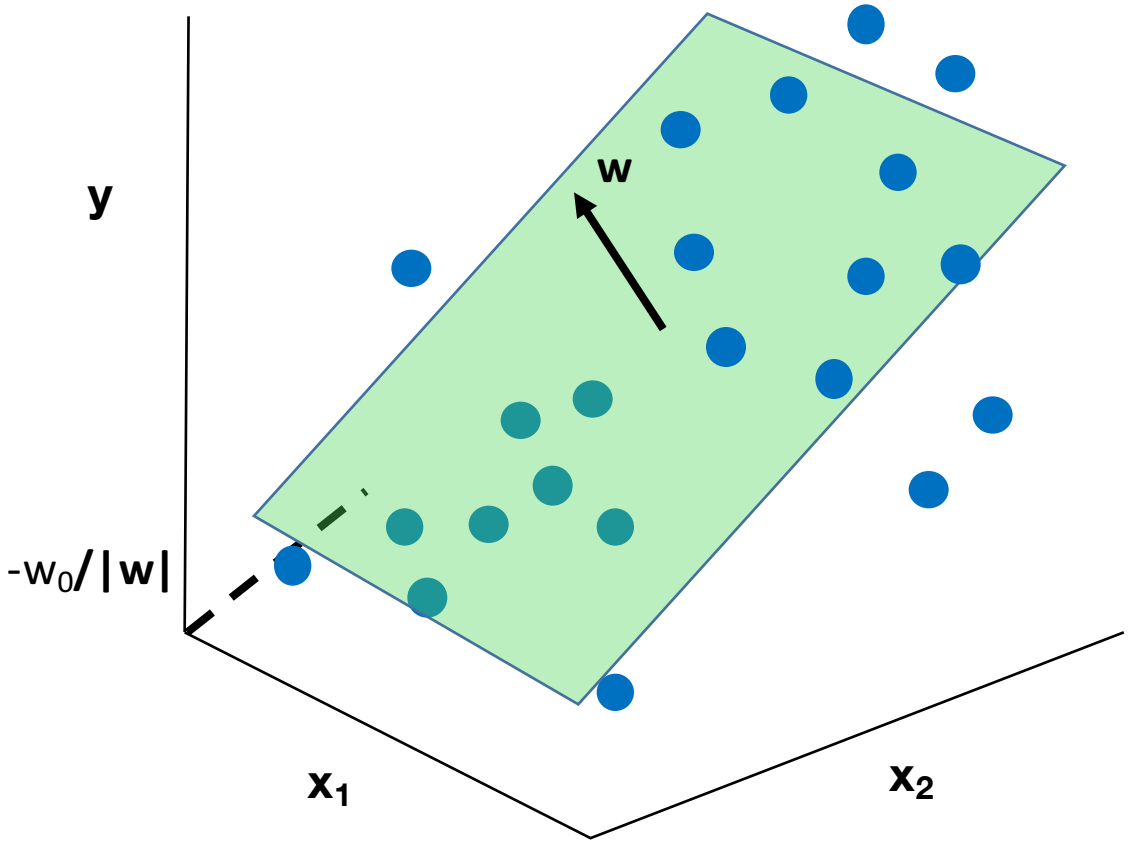
$$L = \frac{1}{N}\sum_{i=1}^{N} (w^T x_i - y_i)^2$$

Where labels determined by thresholding

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

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

Why does linear regression with sgn() achieve classification?



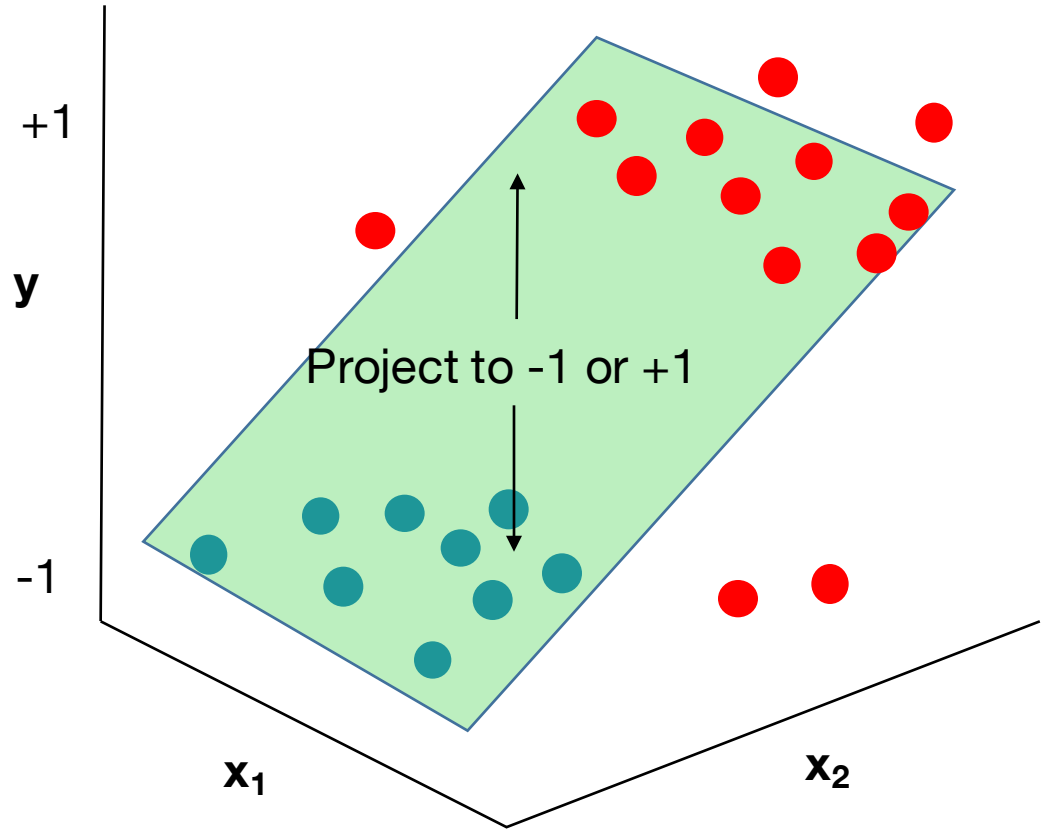
Without sgn(): regression for best fit

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

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

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

Why does linear regression with sgn() achieve classification?



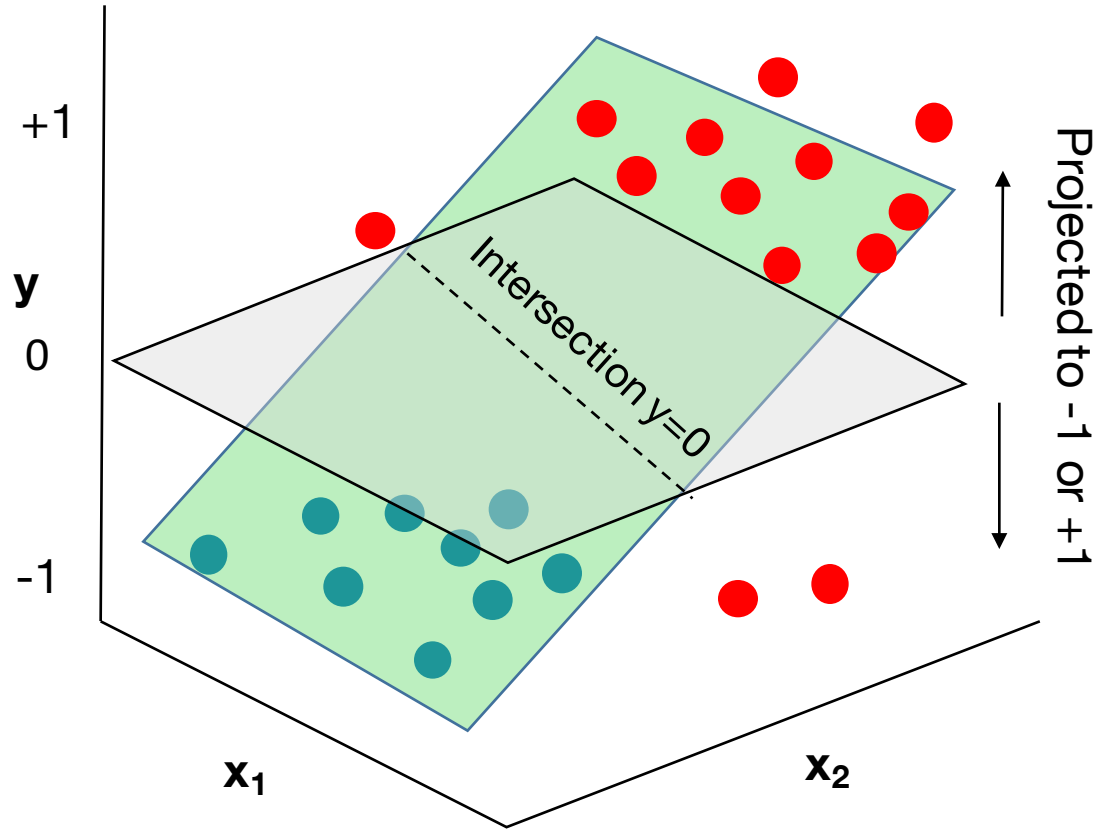
Without sgn(): regression for best fit

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

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

- 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?



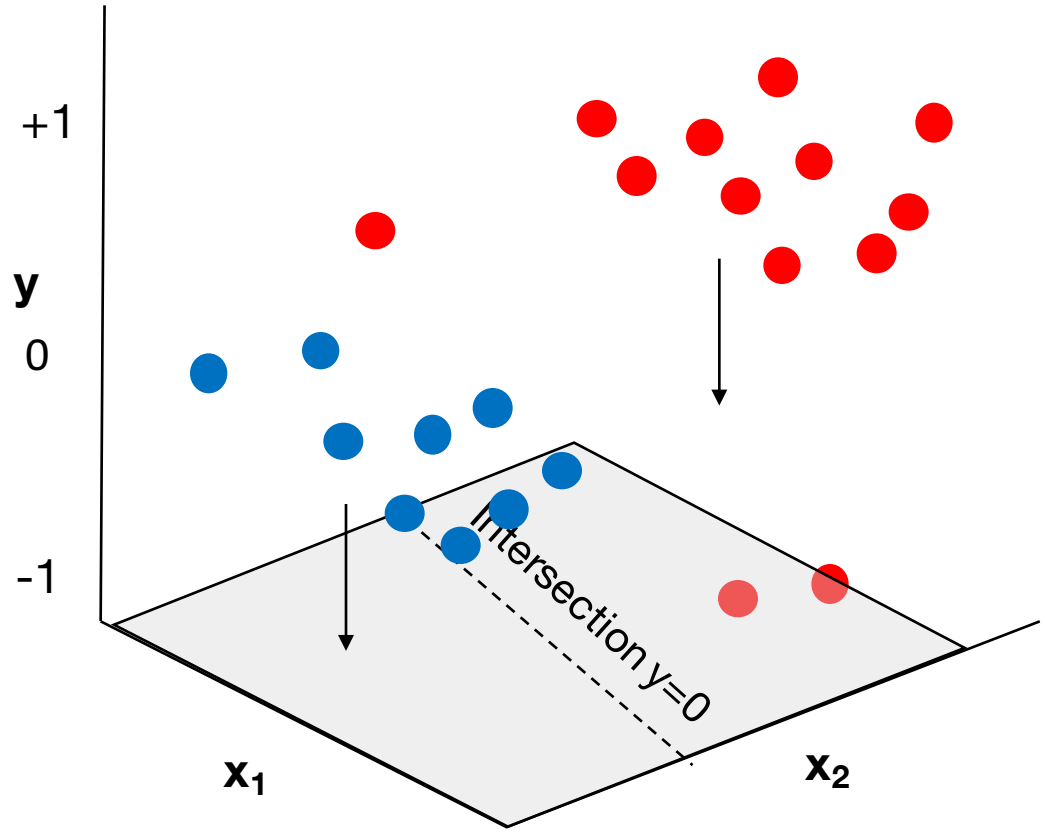
With sgn() operation:

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

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

- 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:

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

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

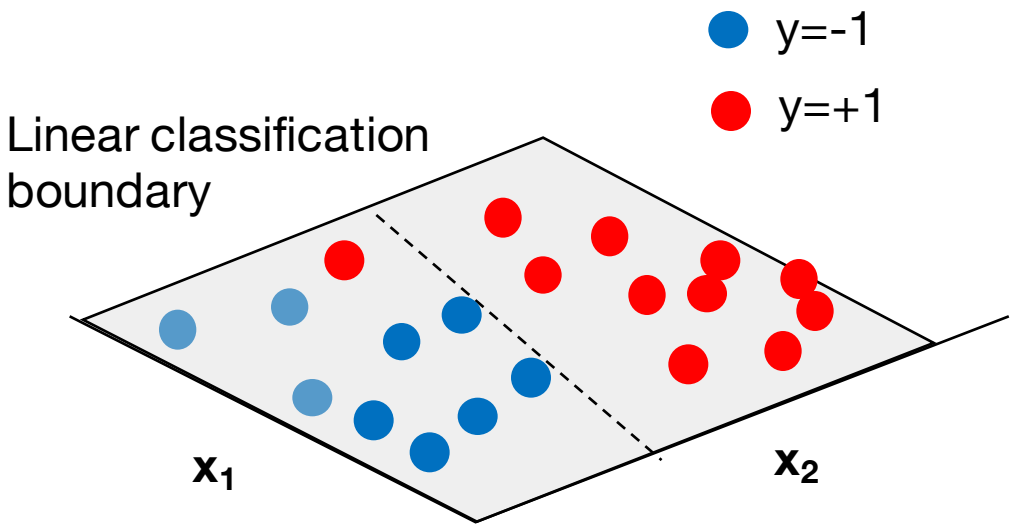
- y axis isn't really needed now & can view this decision boundary in 2D

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

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

d$L(W_1)$/dW_1 = 10

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

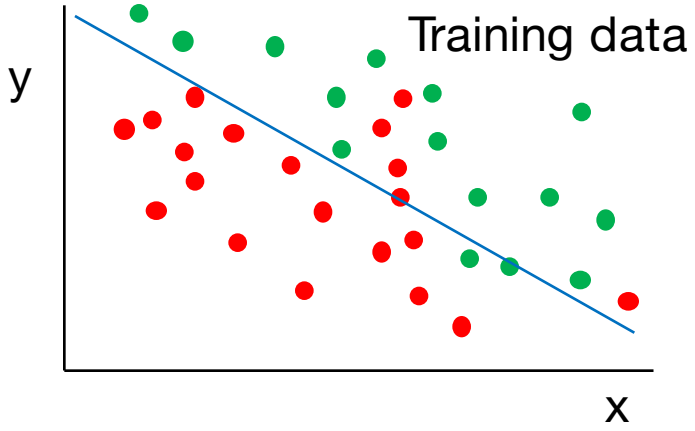
Steepest descent and the best step size ε

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

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

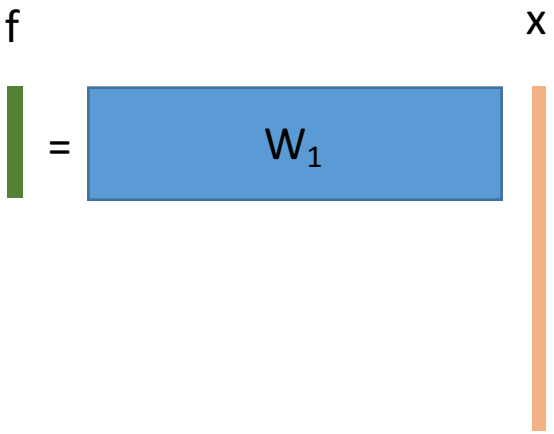
$$\nabla L(w) = \frac{2}{N}X^T(Xw - y) = 0$$

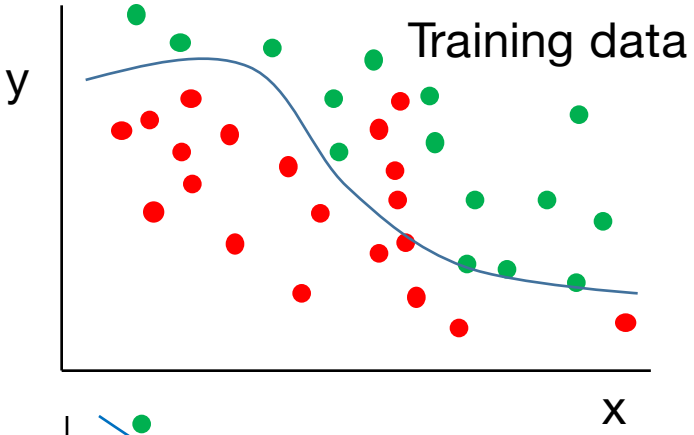
```
while previous_step_size > precision and iters < max_iters:
    prev_x = cur_x
    cur_x -= gamma * df(prev_x)
    previous_step_size = abs(cur_x - prev_x)
    iters+=1
```



$$f = W_1 x$$

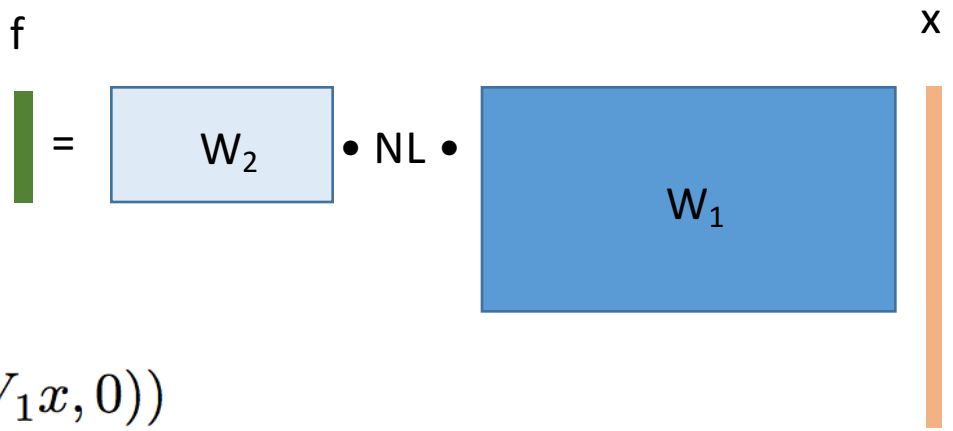
Learned *f:* not flexible

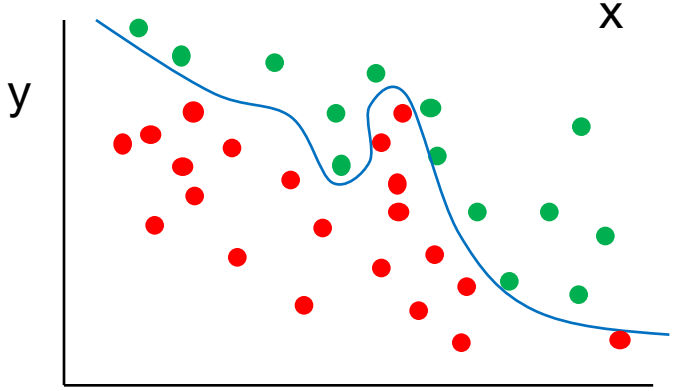




$$f = W_2 \max(W_1 x, 0)$$

Learned *f:* a bit flexible





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

Learned *f:* more flexible

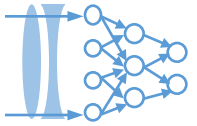
Does it generalize???

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)



Important components of a CNN

CNN Architecture

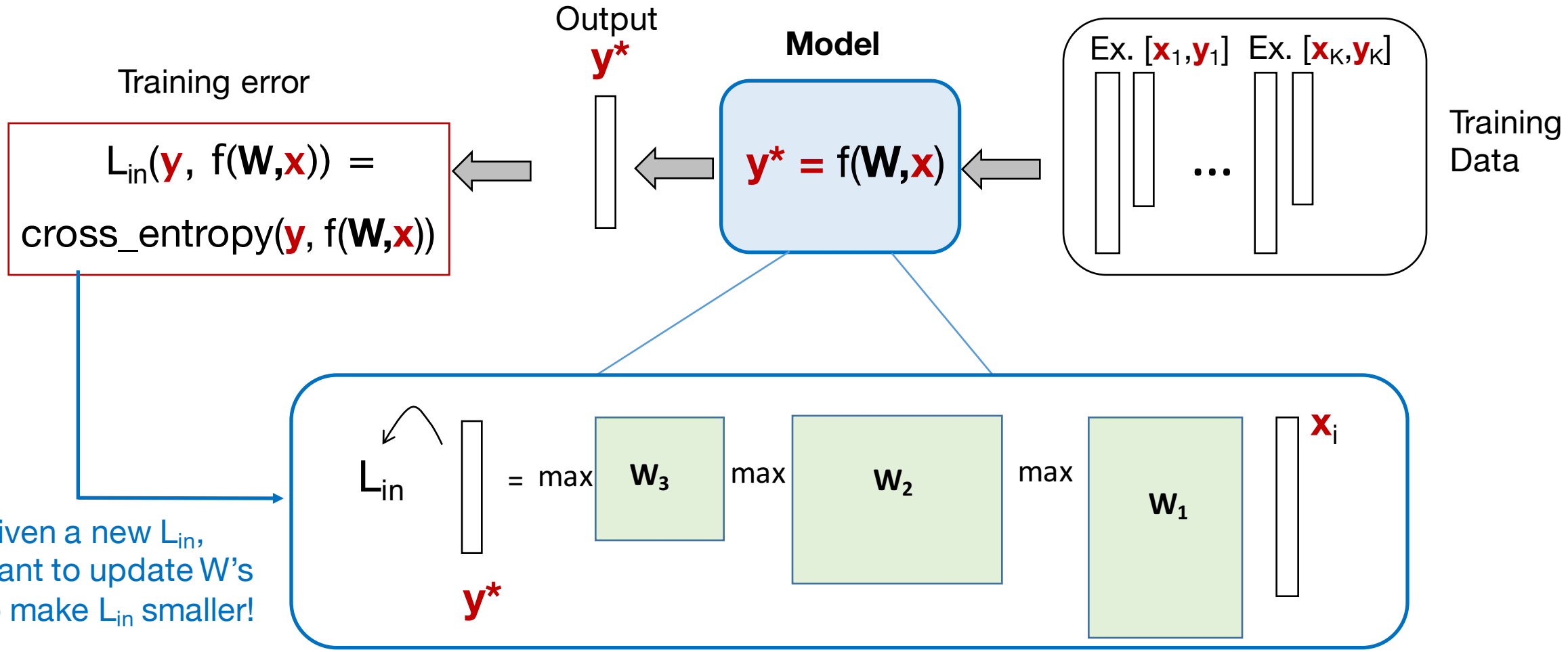
- CONV size, stride, pad, depth
- ReLU & other nonlinearities
- POOL methods
- # of layers, dimensions per layer
- Fully connected layers

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

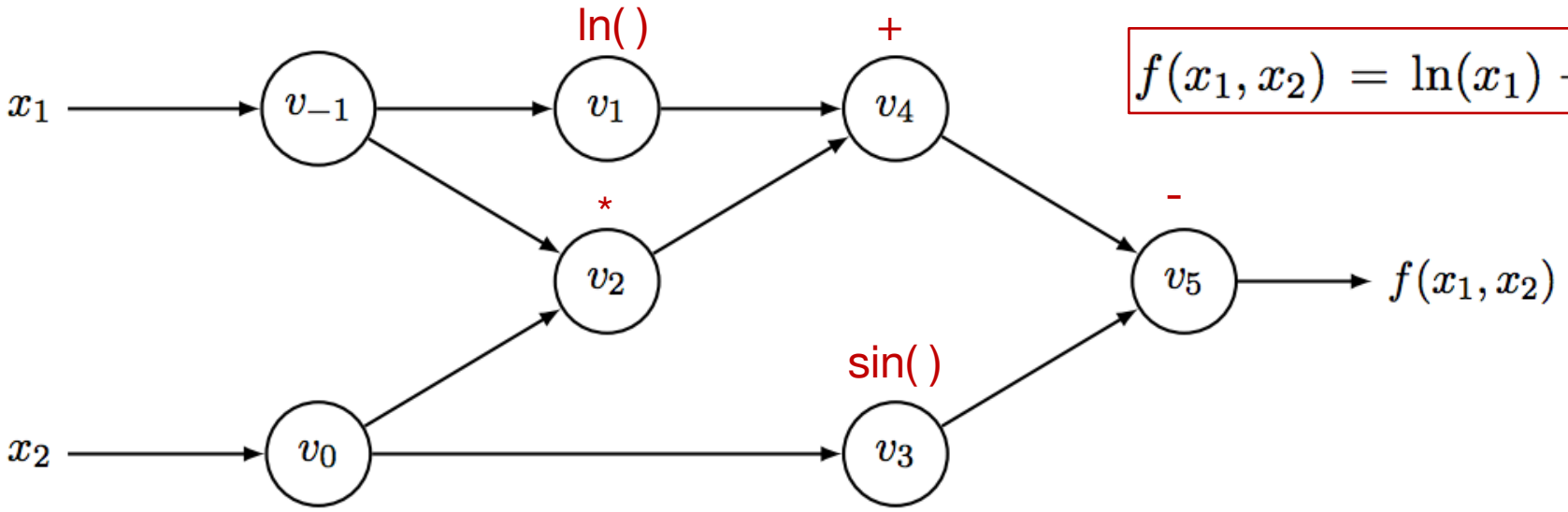


Backwards pass uses new L_{in} to update **W**'s – backpropagation!

Automatic differentiation on computational graphs

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

deep imaging



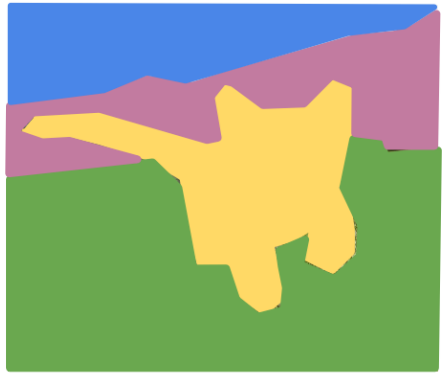
$$f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

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

Other Computer Vision Tasks

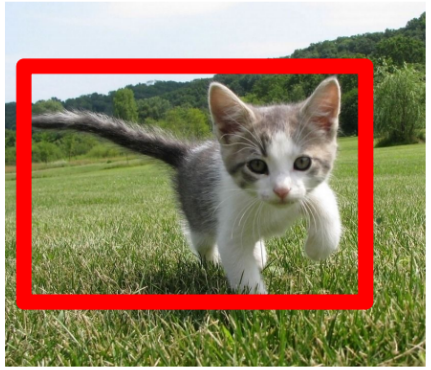
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

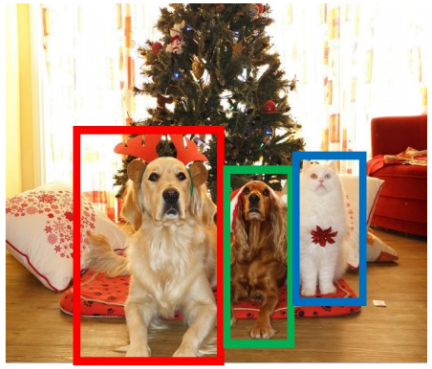
Classification + Localization



CAT

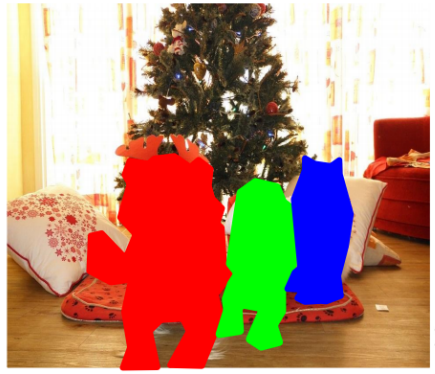
Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

Super-resolution

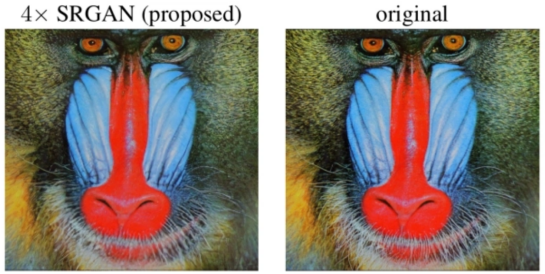


Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

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

Instead, *compress x-y dimensions of* input image

deep imaging

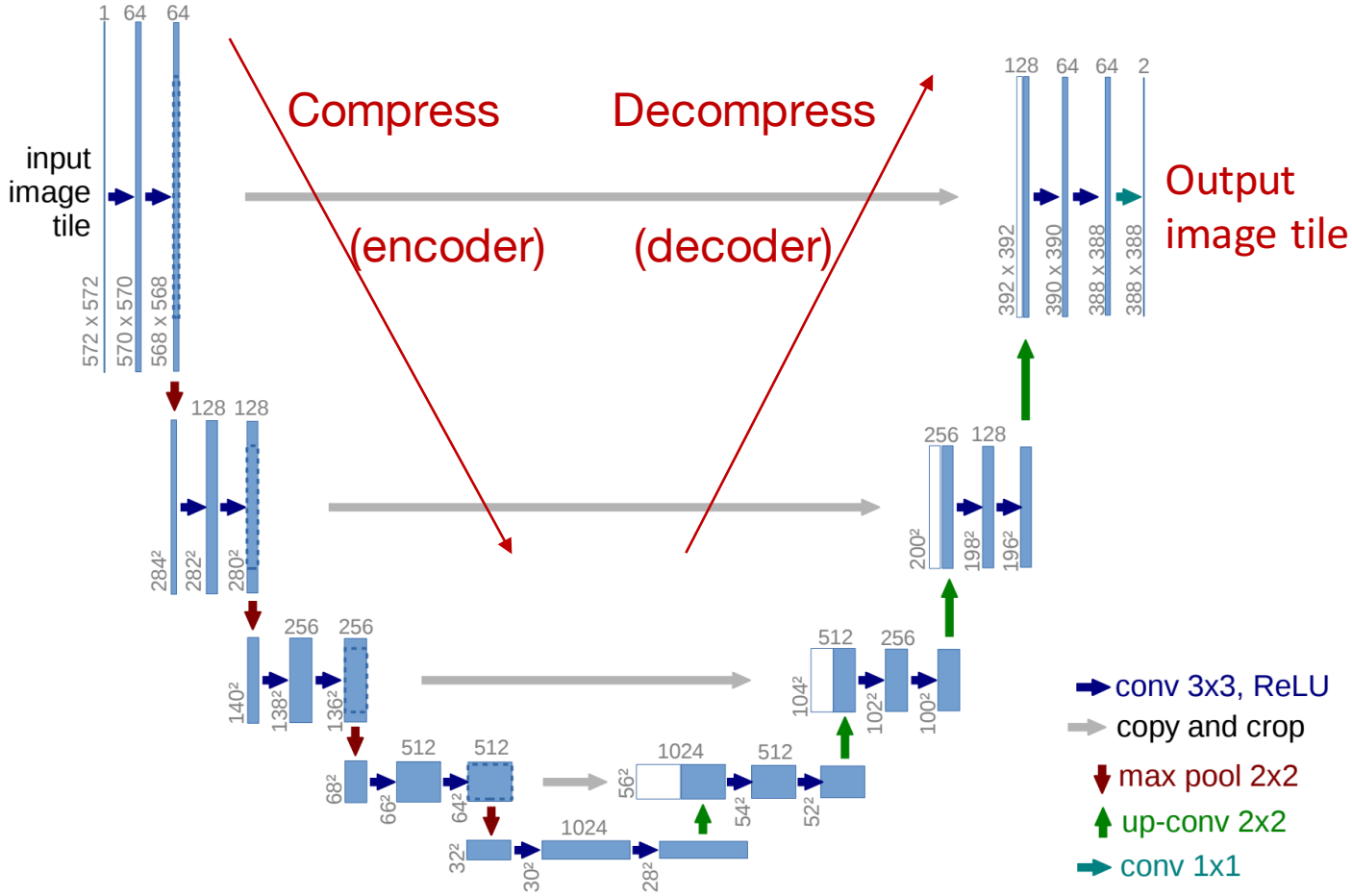
- Compress spatial features into learned filters
- Then, decompress learned filters back into same spatial dimensions
- **Can be an autoencoder**
- Analogous to image compression
- A very powerful idea…

U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/

U-Net Architecture



Bringing together physical and digital image representations

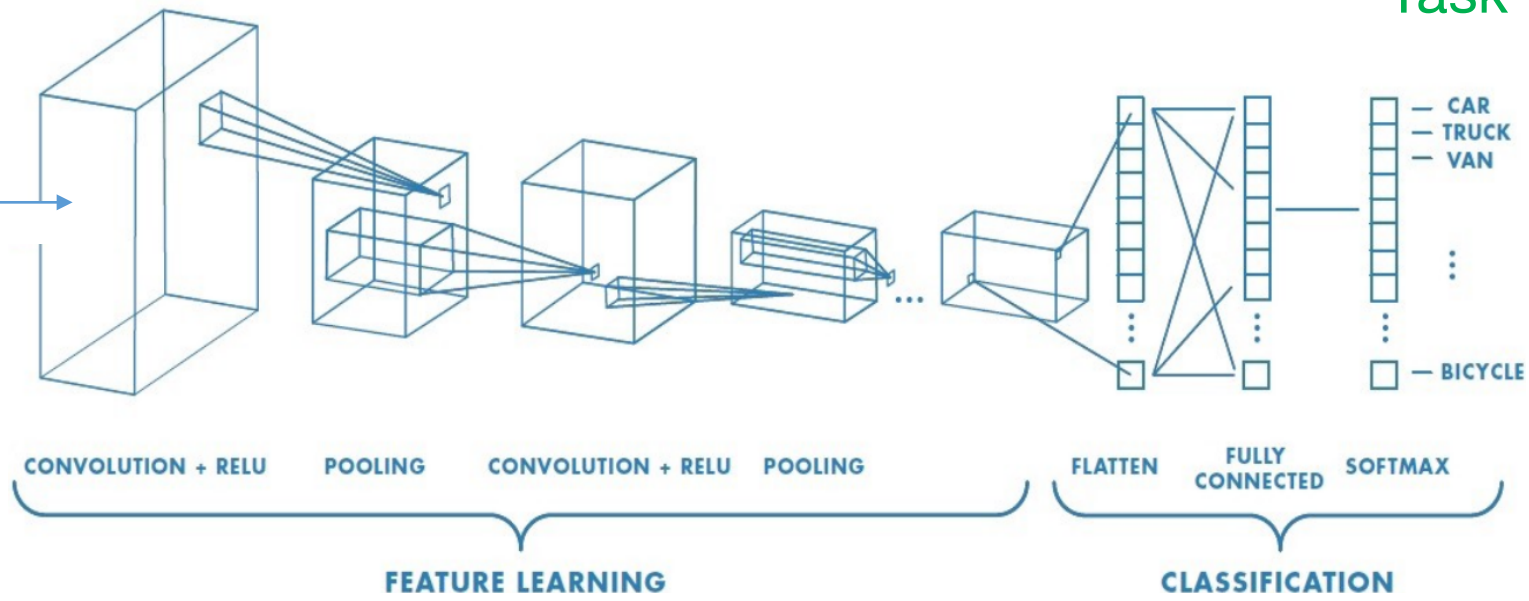
deep imaging

Physical Layers

Digital Layers

Task

Physical Function I_0 → f[ ] → Digitized I_s →



$$I_s = f[I_0]$$

Digital layers

Physical layers

$$\text{Task} = W_n \ldots \text{ReLU}[W_1 \, \text{ReLU}[W_0 \, f[I_0]] \ldots]$$

Physical Layers

Digital Layers

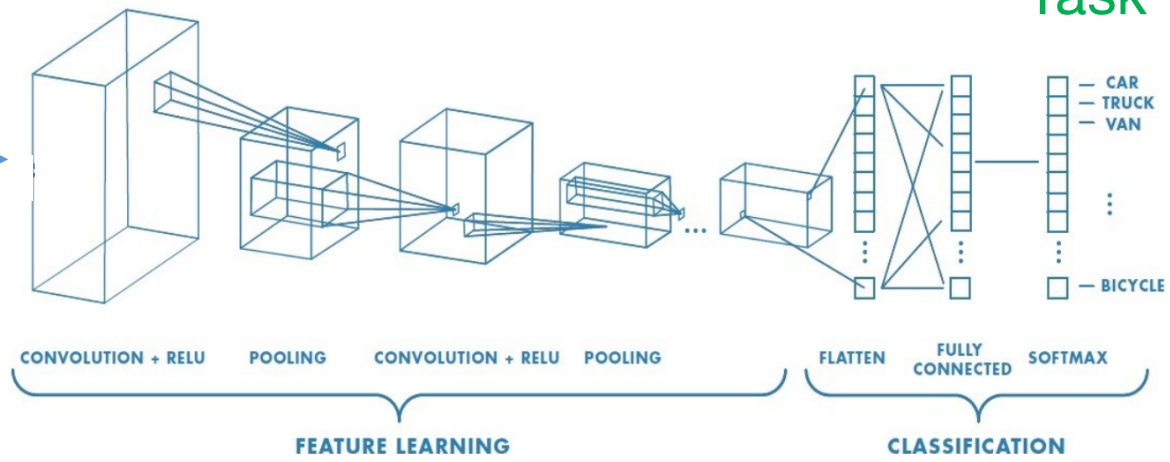
Input object/image data I_0

f[ ]

$I_s = f[I_0]$

Digitized I_s

Task



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

Situation #1: Fully simulated physical layers

Physical Layers

Input object data I_0

simulation

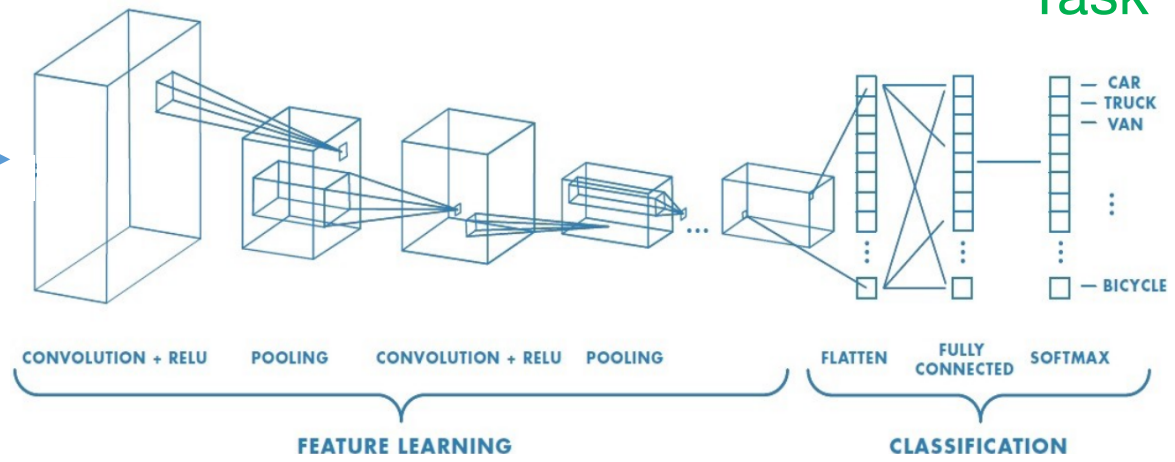
f[ ]

$I_s = f[I_0]$

Digitized I_s

Digital Layers

Task



$I_s = H\ B\ S_0$

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

Examples:

- Simulate ultrasound samples, and learn illumination/detection geometry to best segment data
- Blur high-resolution labeled images, to learn and ideal blur shape
- Undersample your data first to learn ideal sampling strategy

Example code to achieve this is in Homework #5 and linked on course website:

jupyter physical_layers_example (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help Not Trusted Python 3

Code

In [2]:
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
tf.__version__
```

Out[2]: '2.0.0'

In [0]:
```
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_test, x_train = x_test/255.0, x_train/255.0 # normalization
```

In [0]:
```
# add channel dimension, Tensorflow requests NHWC (or BHWC) format. Number/Batch_size, height, width, channels
# for 1D vectors, it should be Number/Batch_size, length
x_train = x_train[...,None]
x_test = x_test[..., None]
```

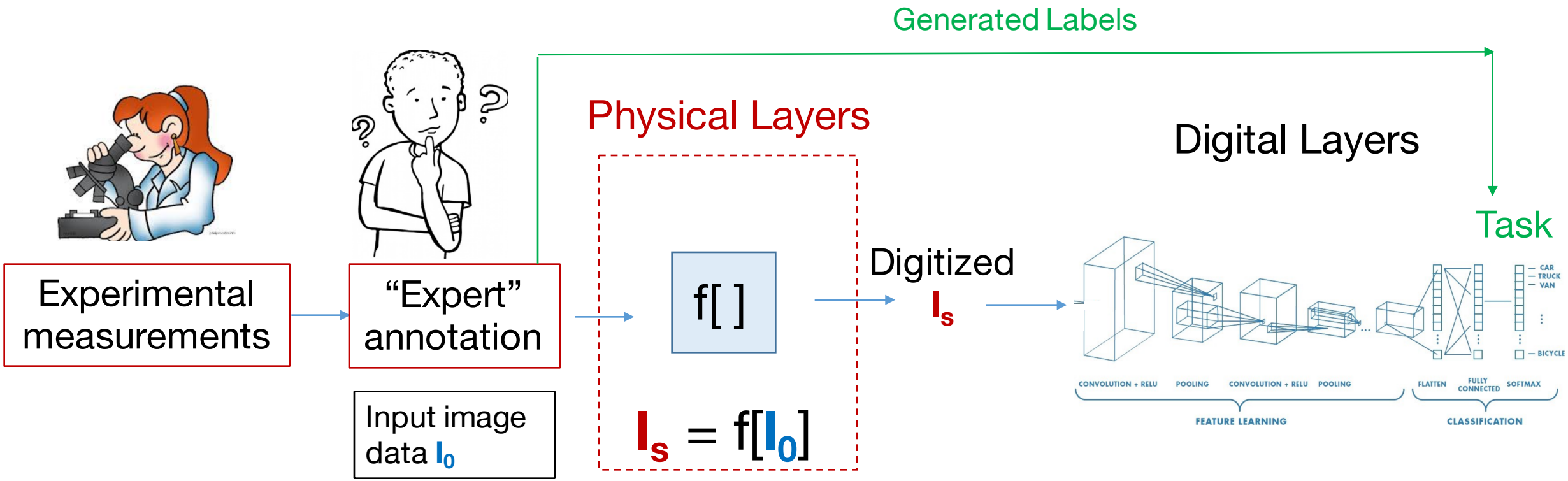
In [0]:
```
# another way to define models
# lets us access intermediate results too

mask_to_multiply = tf.Variable(initial_value = tf.initializers.GlorotNormal()(shape=(28,28,1)), trainable = True)
mask_to_multiply = tf.expand_dims(mask_to_multiply, 0) # add extra axis for batch size

#PHYSICAL LAYERS
input_image = tf.keras.layers.Input(shape = (28,28,1)) # Input layer, shape should be given as HWC
out = tf.math.multiply(input_image, mask_to_multiply) # hadamard product for illumination physical layer
out = tf.keras.layers.Conv2D(filters = 1, kernel_size = 5, strides =1, padding ="same", activation = "relu")(out) # con
##########################################################################################################
```

Situation #2: Experimentally-driven physical layers

deep imaging

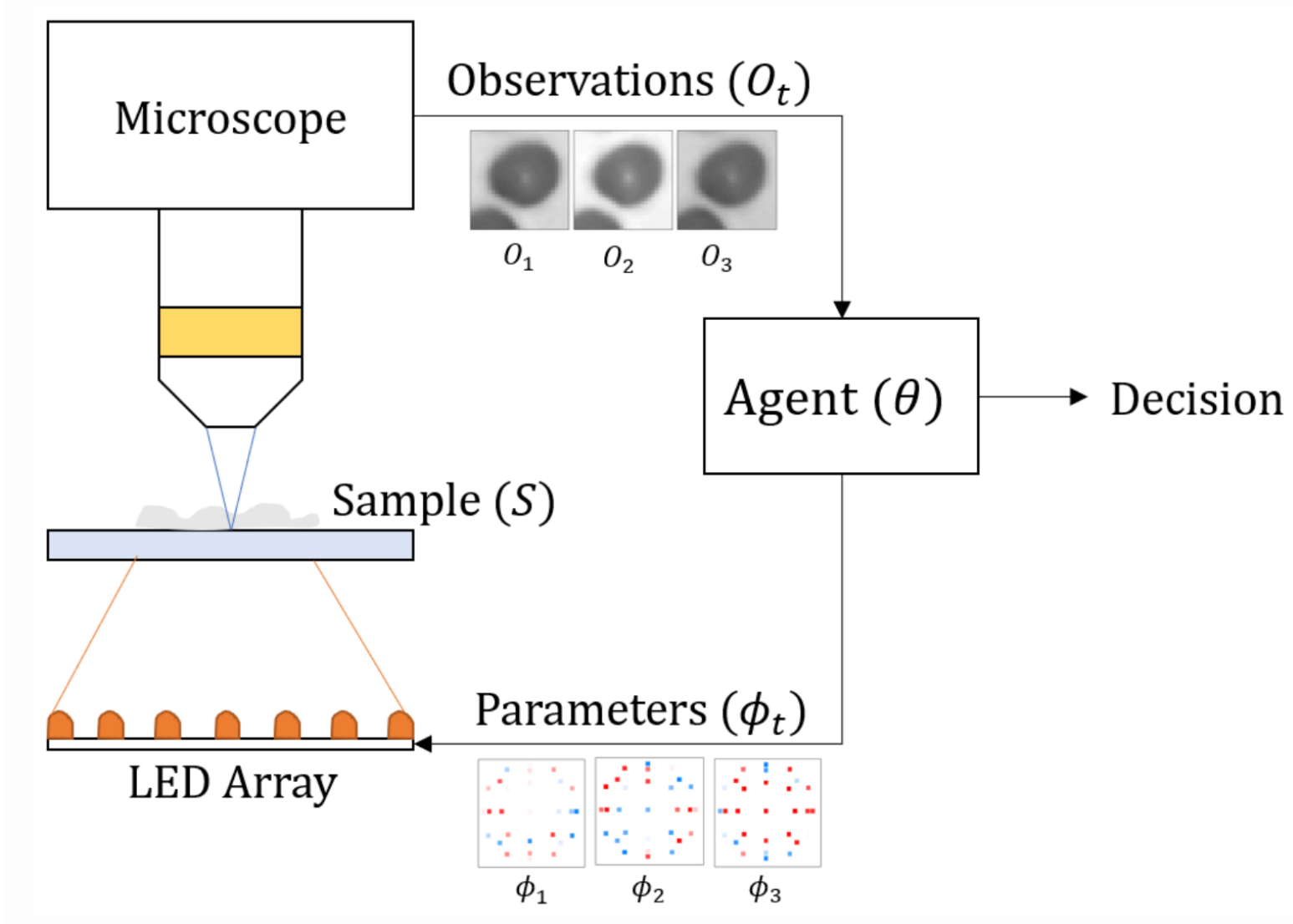


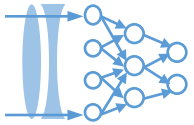
Typically need to use an "insight" to determine function f() for physical layer

Examples:

- Sum of weighted color channels equals a color filter
- weighted sum of uniquely illuminated images equals a pattern-illum. Image

How can this be applied to optimized imaging?





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: