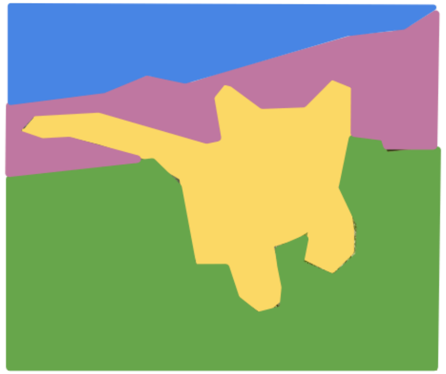
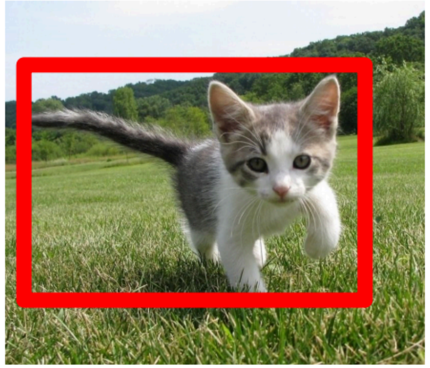
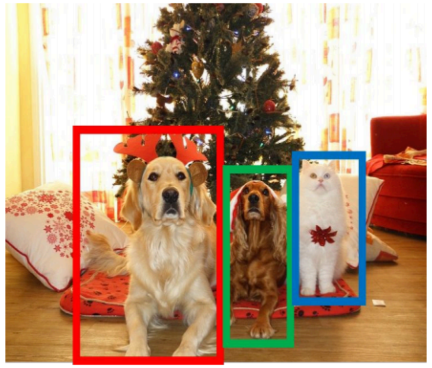
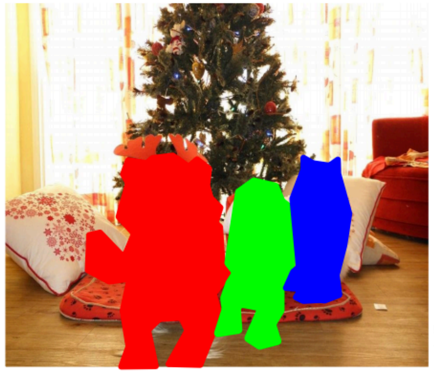
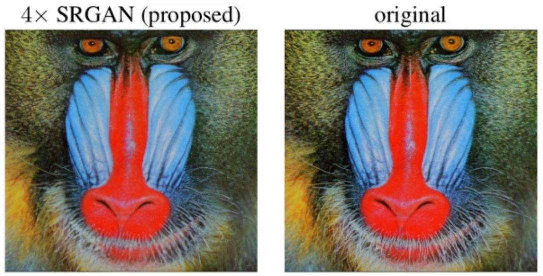
deep imaging

Lecture 15: Introduction to Physical Layers in Machine Learning

Machine Learning and Imaging

BME 548L
Roarke Horstmeyer

Other Computer Vision Tasks - **We'll pick this back up next class**

Semantic Segmentation	Classification + Localization	Object Detection	Instance Segmentation	Super-resolution
				
GRASS, CAT, TREE, SKY	CAT	DOG, DOG, CAT	DOG, DOG, CAT	Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]
No objects, just pixels	Single Object	Multiple Object	Multiple Object	
Balanced equation	**Over-determined**	**Over-determined**	**Over-determined**	**Under-determined**

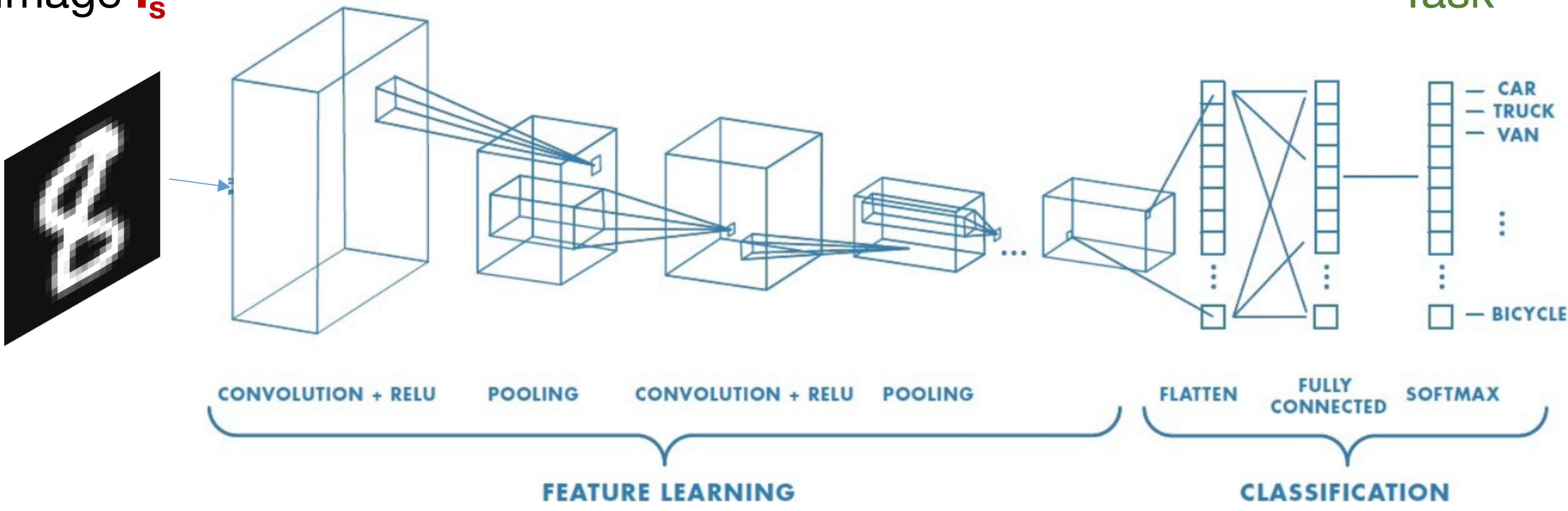
This image is CC0 public domain

Stanford CS231n - http://cs231n.stanford.edu

Bringing together physical and digital image representations

Image $\mathbf{I_s}$

Task

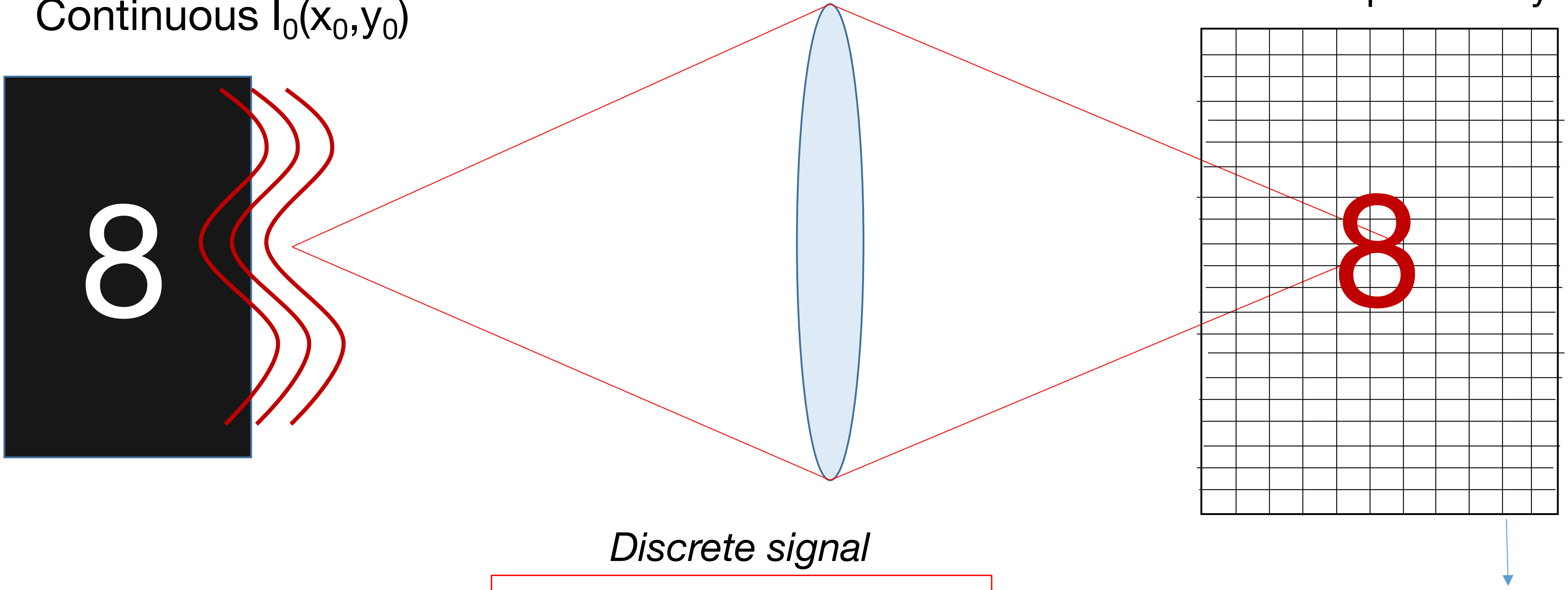


$$\text{Task} = \mathbf{W_n} \ldots \text{ReLU}[\mathbf{W_1}\ \text{ReLU}[\mathbf{W_0}\ \mathbf{I_s}]\ldots]$$

Simple model of image formation

Continuous $I_0(x_0,y_0)$

n x *m* pixel array



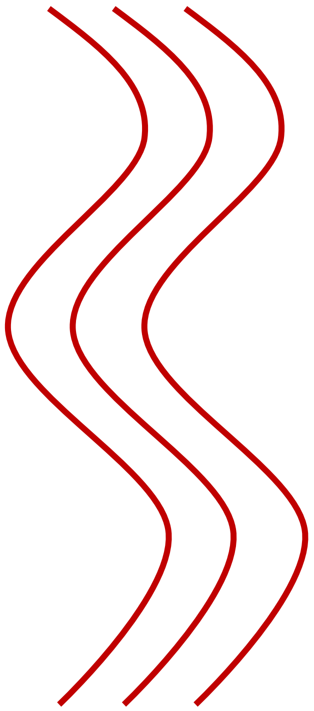
Discrete signal

$$\boxed{I_s(x,y), (x,y) \in Z^{n \times m}}$$

← Digitazation ← Photons to electrons

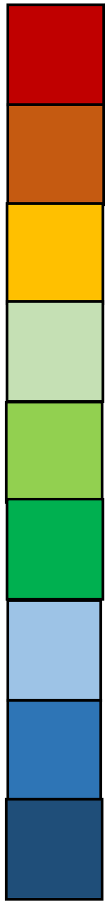
What does the Sampling Theorem mean for us?

Continuous functions



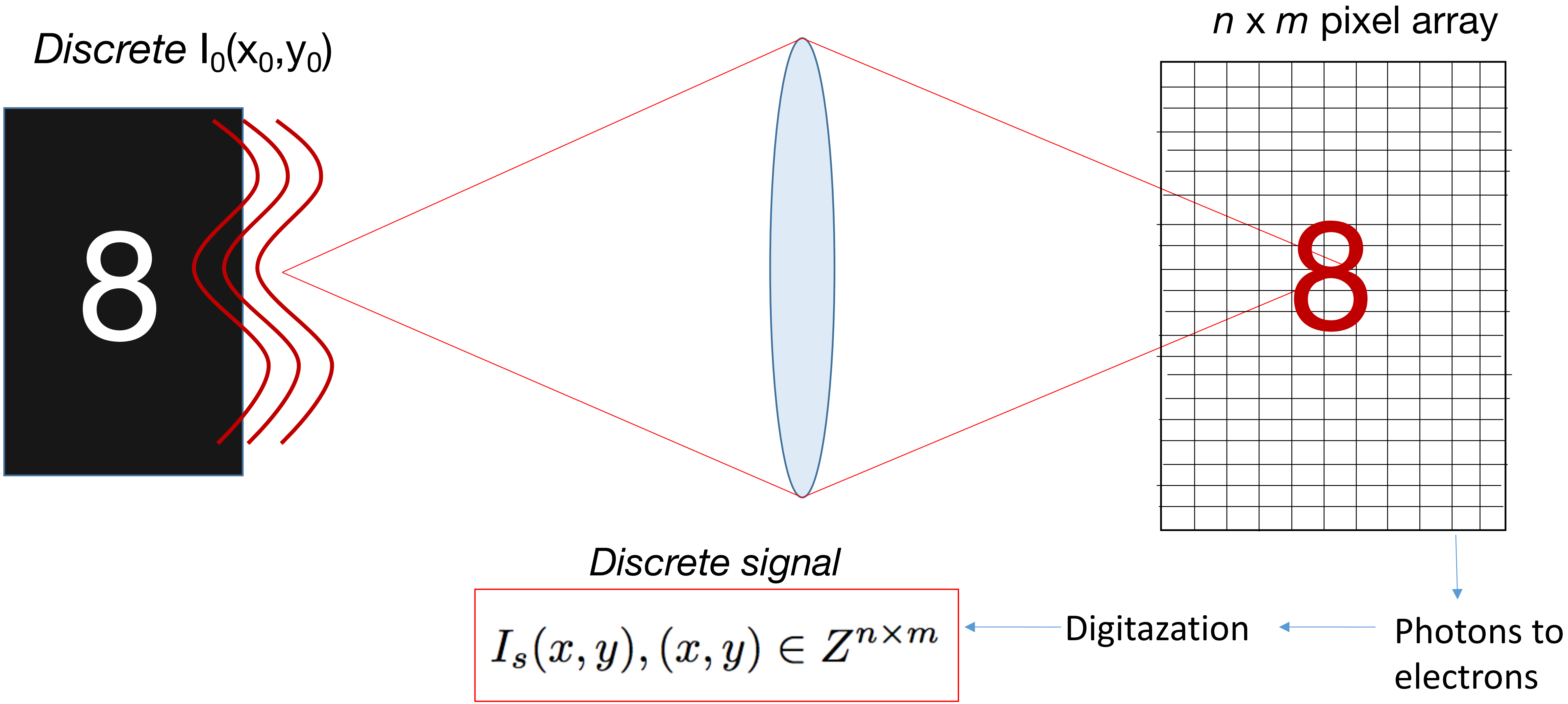
*conditions

Discretize vectors
(and matrices)



17
20
22
21
23
25
24
26
29

Simple model of image formation



Bringing together physical and digital image representations

deep imaging

Physical Layers

Digital Layers

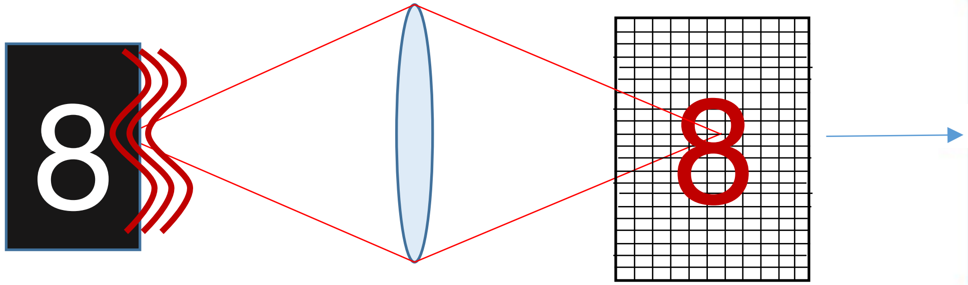
Physical world

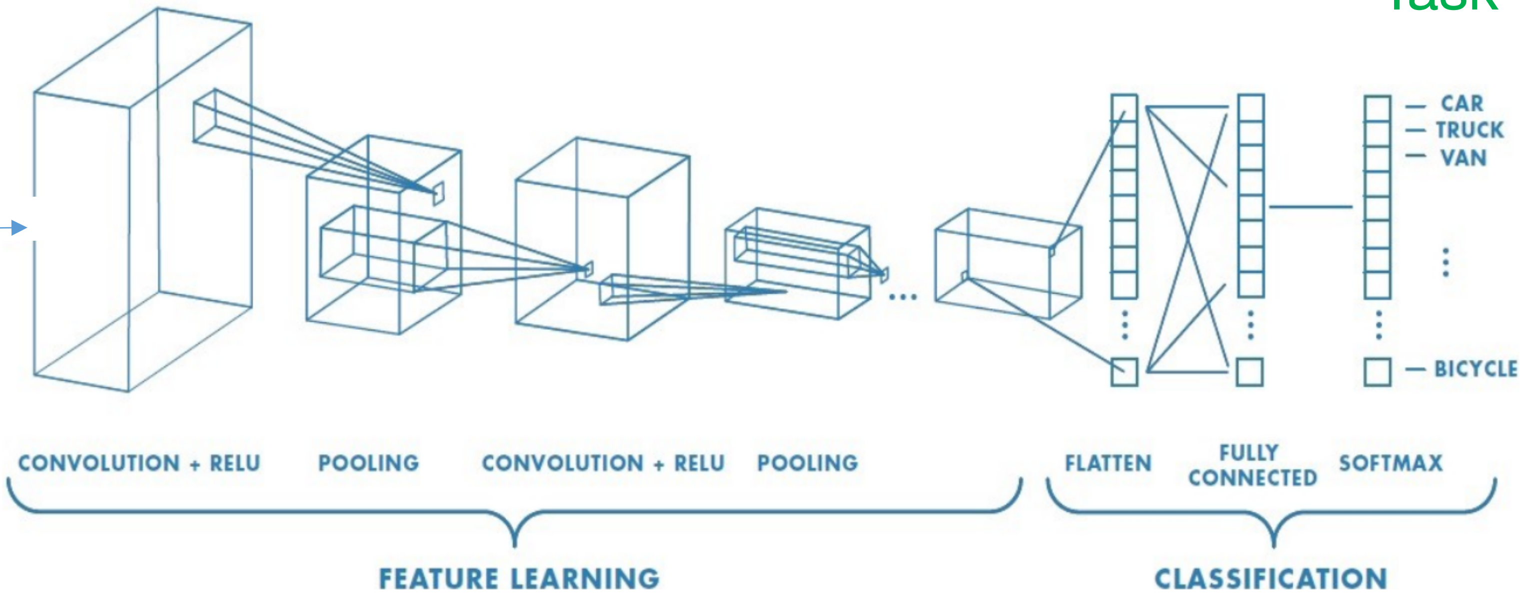
$I_0(x_0,y_0)$

Digital Image

$I_s(x,y)$

Task





$$\mathbf{I_s} = f[\mathbf{I_0}]$$

Digital layers

Physical layers

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

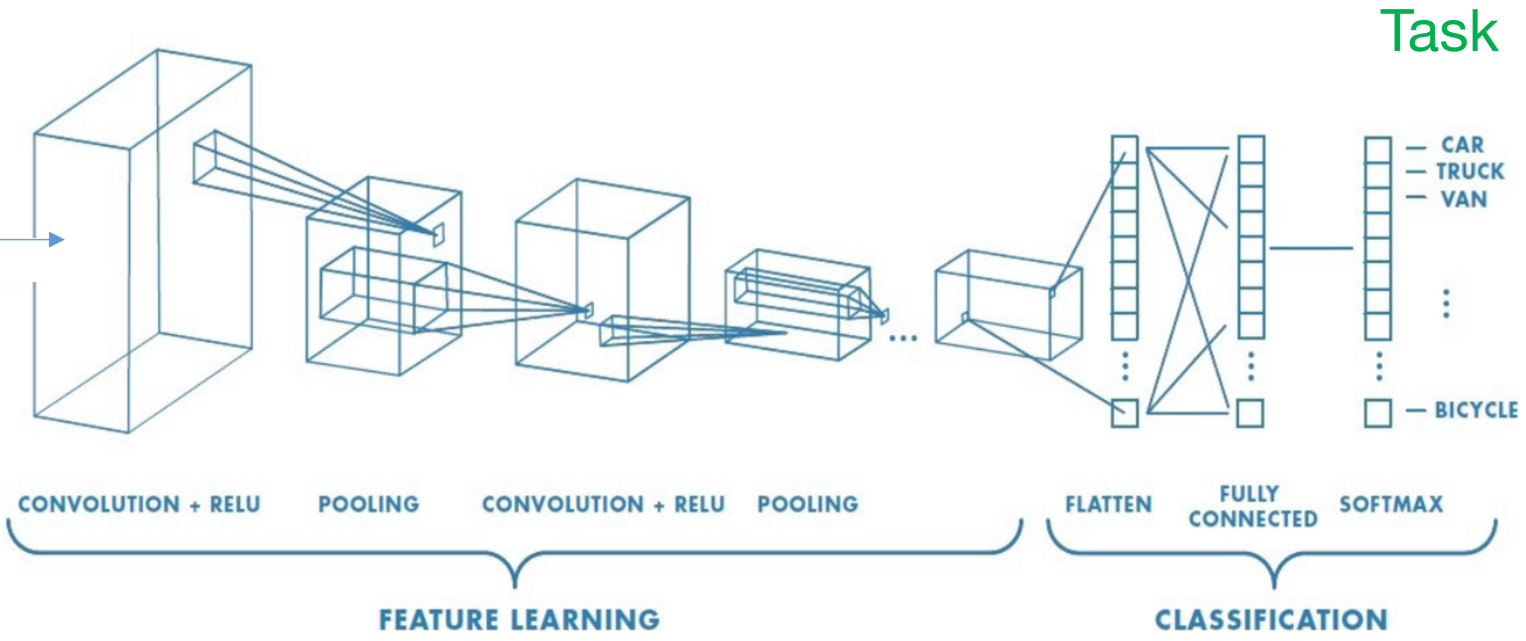
Bringing together physical and digital image representations

deep imaging

Physical Layers

Digital Layers

Physical Function $\mathbf{I}_0$ → f[ ] → Digitized $\mathbf{I}_s$ →



$$\mathbf{I}_s = f[\mathbf{I}_0]$$

Digital layers

Physical layers

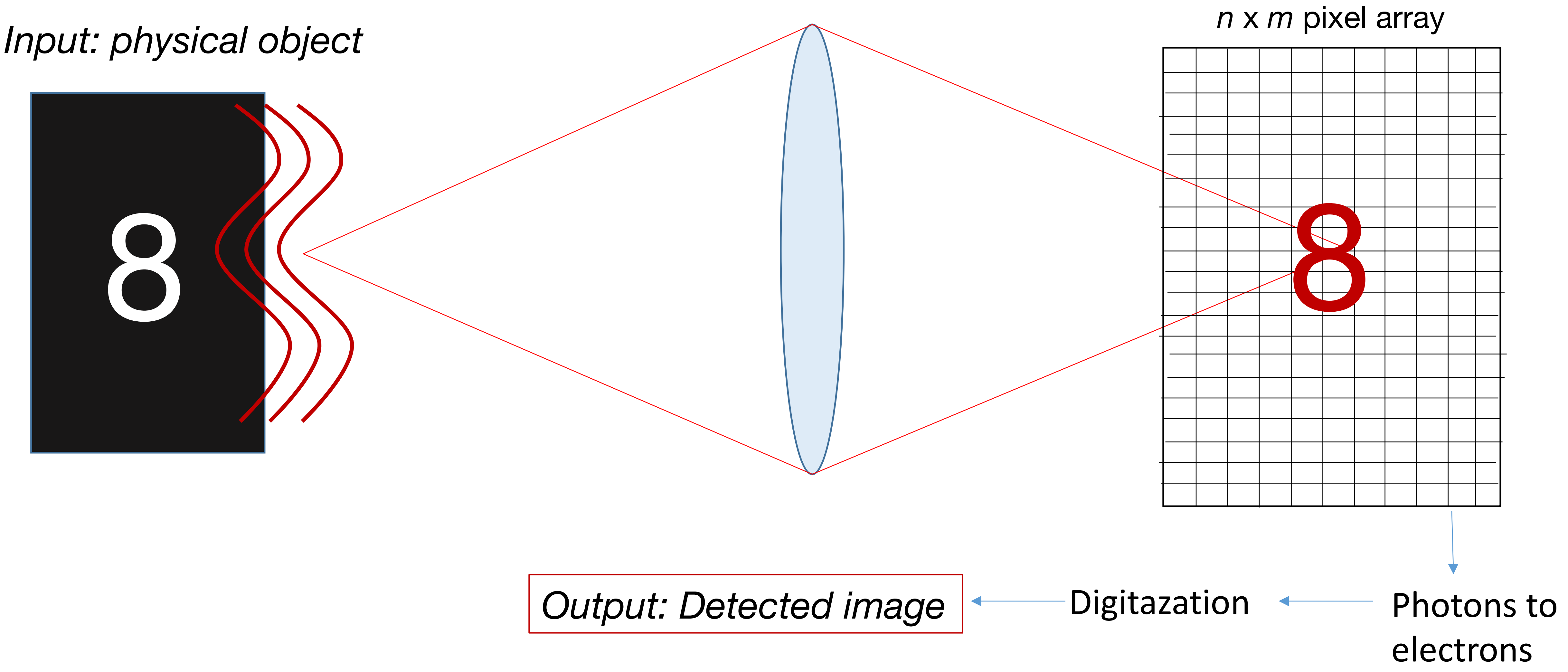
$$\text{Task} = \mathbf{W}_n \ldots \text{ReLU}[\mathbf{W}_1 \,\text{ReLU}[\mathbf{W}_0 \, f[\mathbf{I}_0]]\ldots]$$

Required properties of physical mapping f[ ] for DNN optimization?

deep imaging

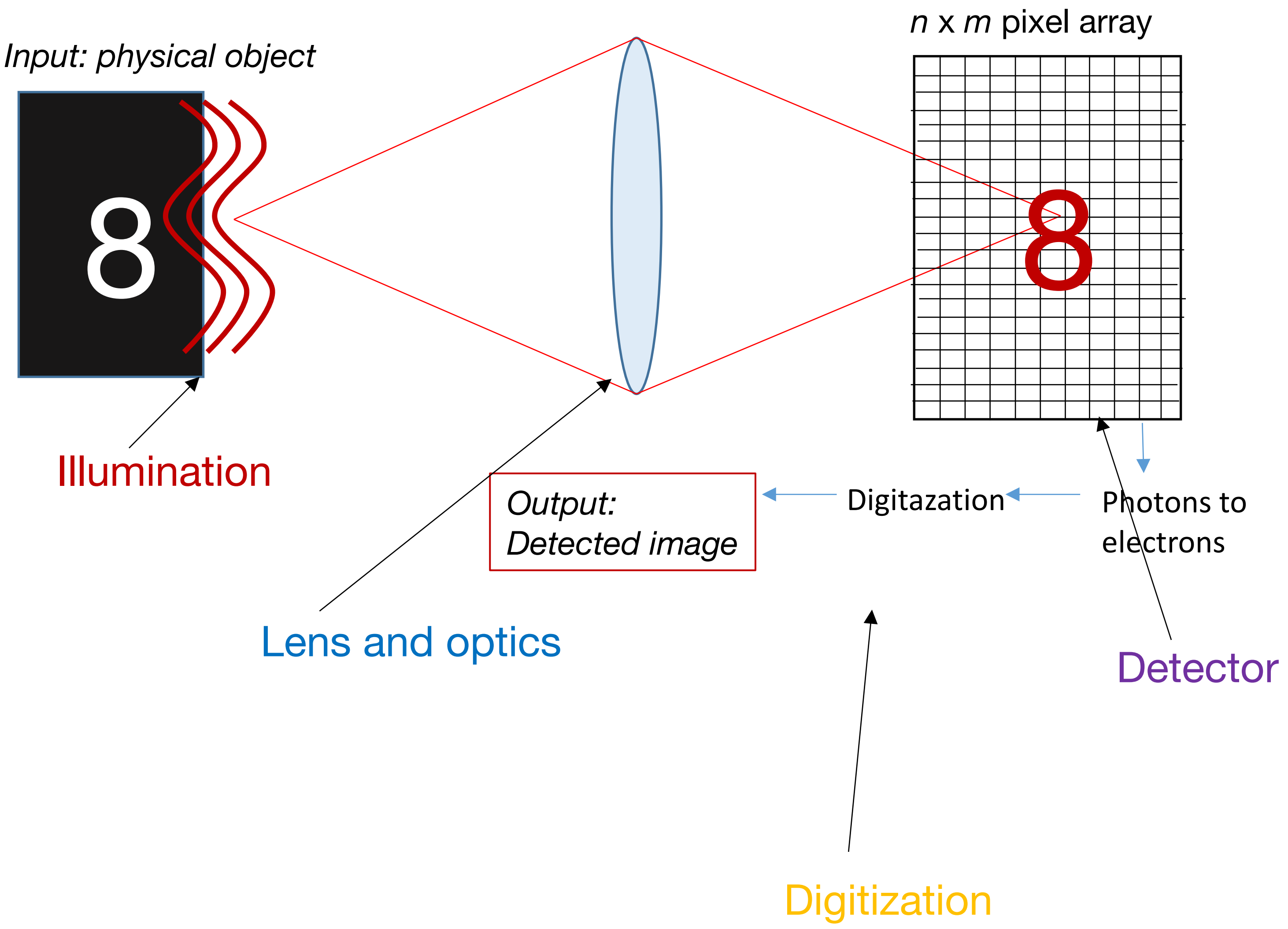
- Finite
- Non-zero gradients
- Differentiable*
- Known structure (for now…)
- Anything else?

What physical parameters effect image formation?

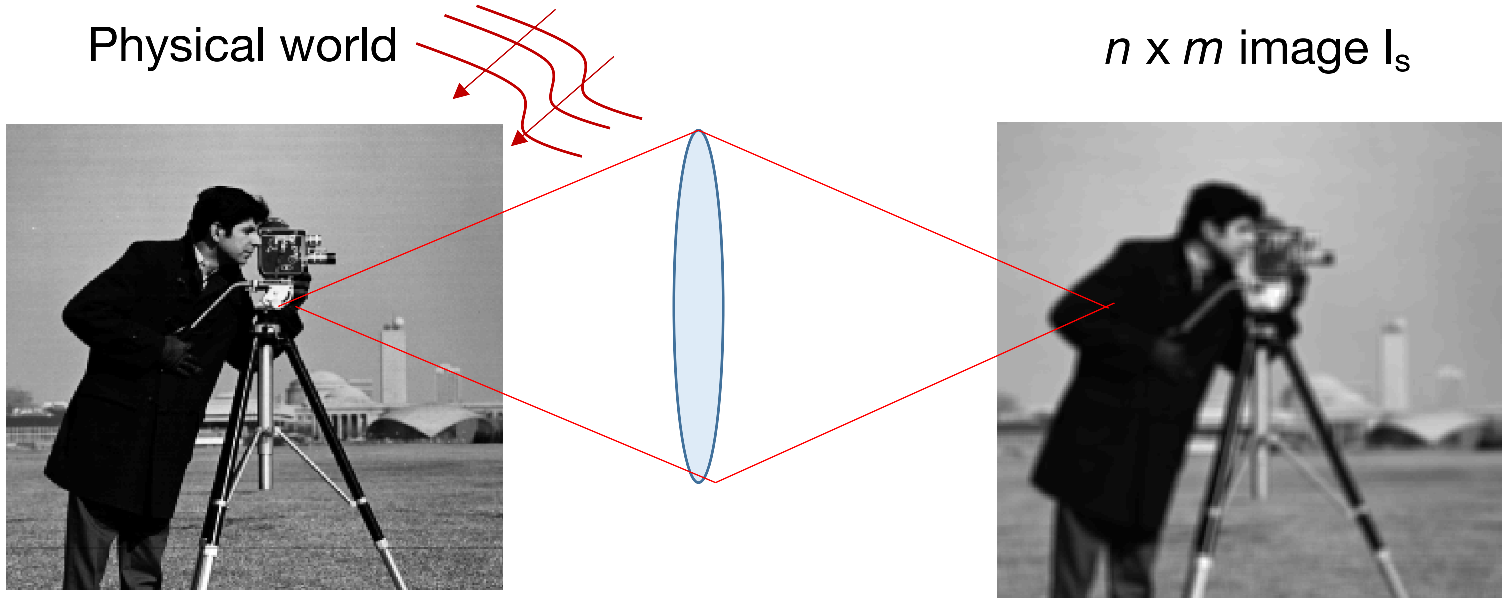


What physical parameters effect image formation?

- 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

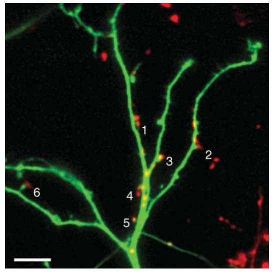


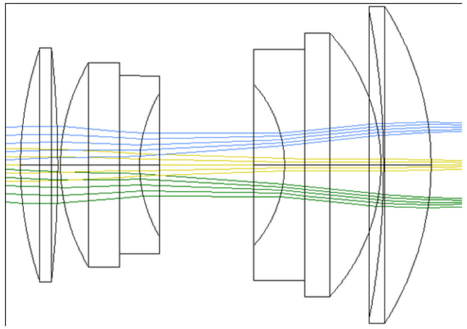
Simple mathematical model of image formation



First - what is light and how can we model it?

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





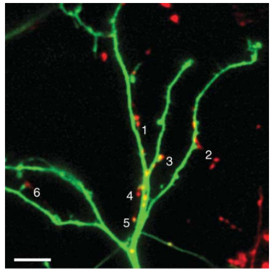
- Real, non-negative
- Models absorption and brightness

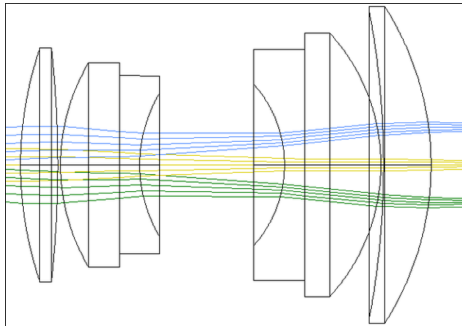
$$I_{tot} = I_1 + I_2$$

deep imaging

First - what is light and how can we model it?

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

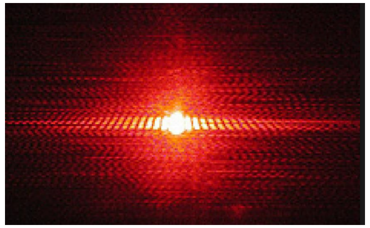


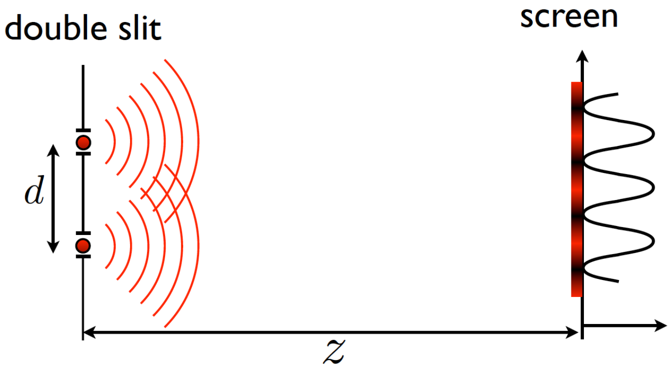


- Real, non-negative
- Models absorption and brightness

$$I_{tot} = I_1 + I_2$$

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



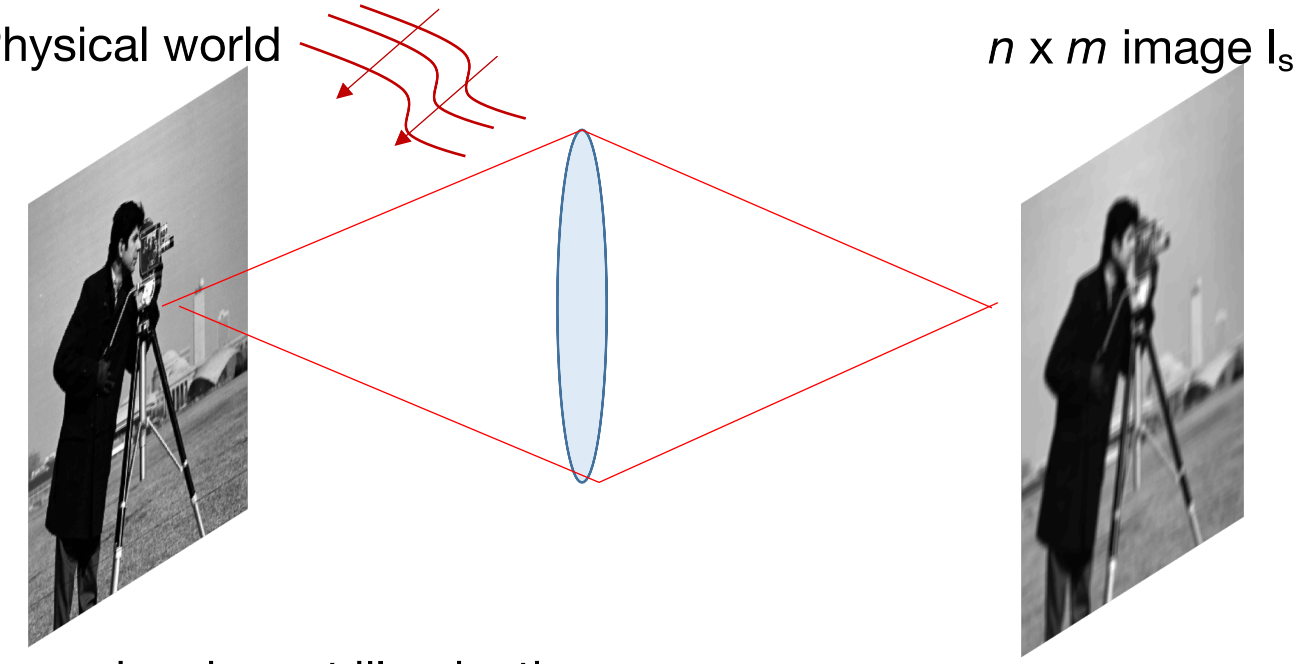


- Complex field
- Models Interference

$$E_{tot} = E_1 + E_2$$

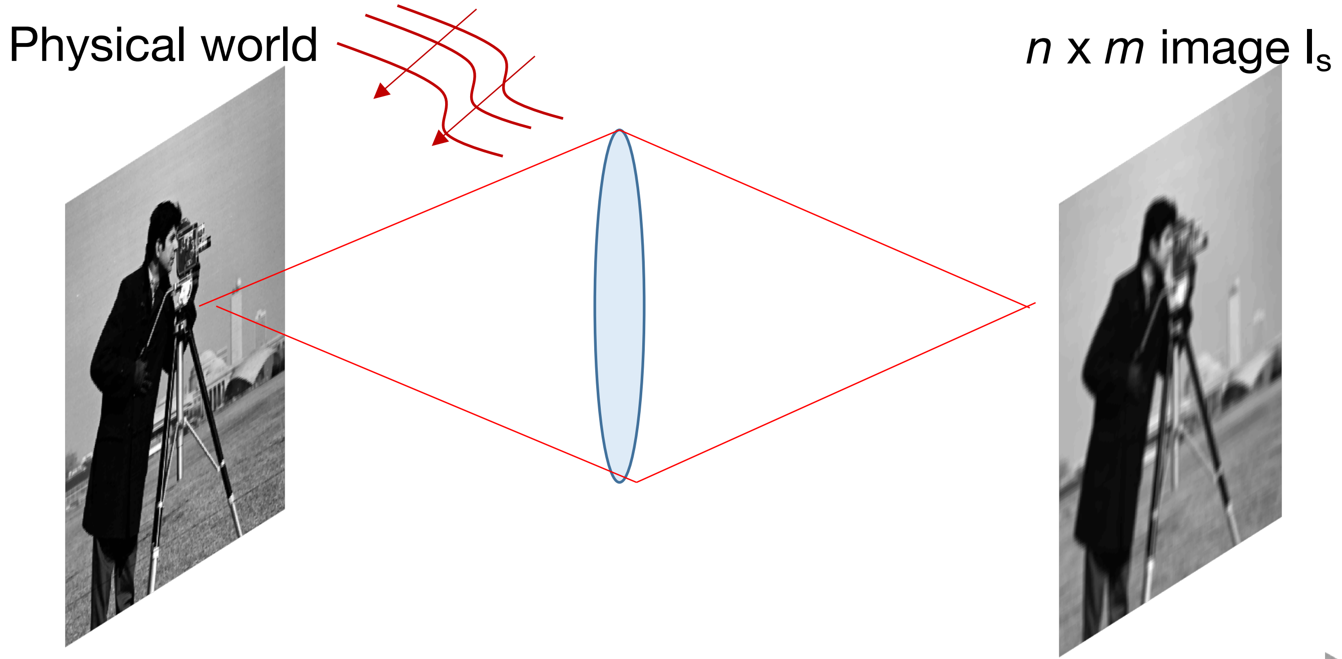
- Interpretation #3: Particle
- Model: Photons

Simple mathematical model of image formation



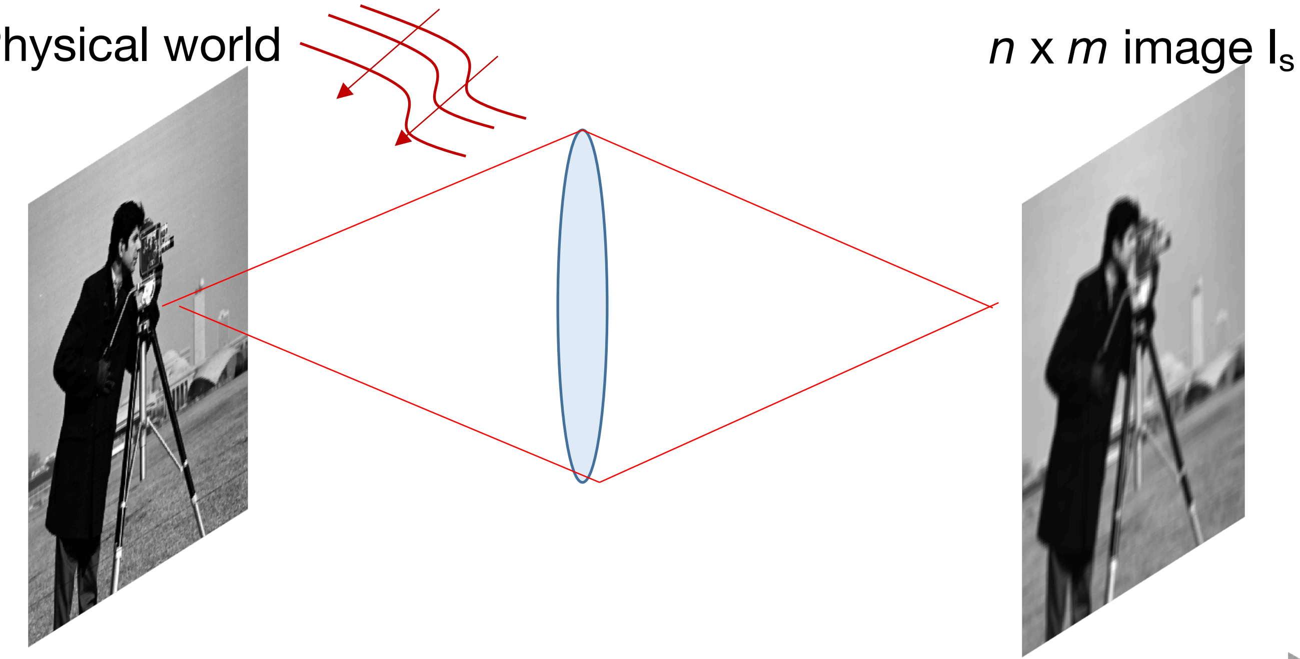
- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity

Simple mathematical model of image formation

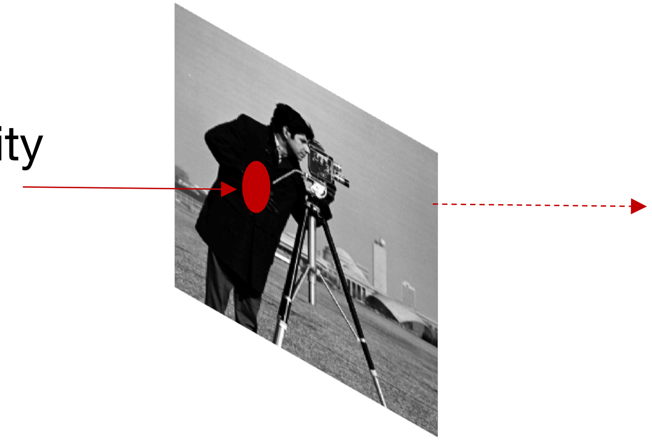


- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity

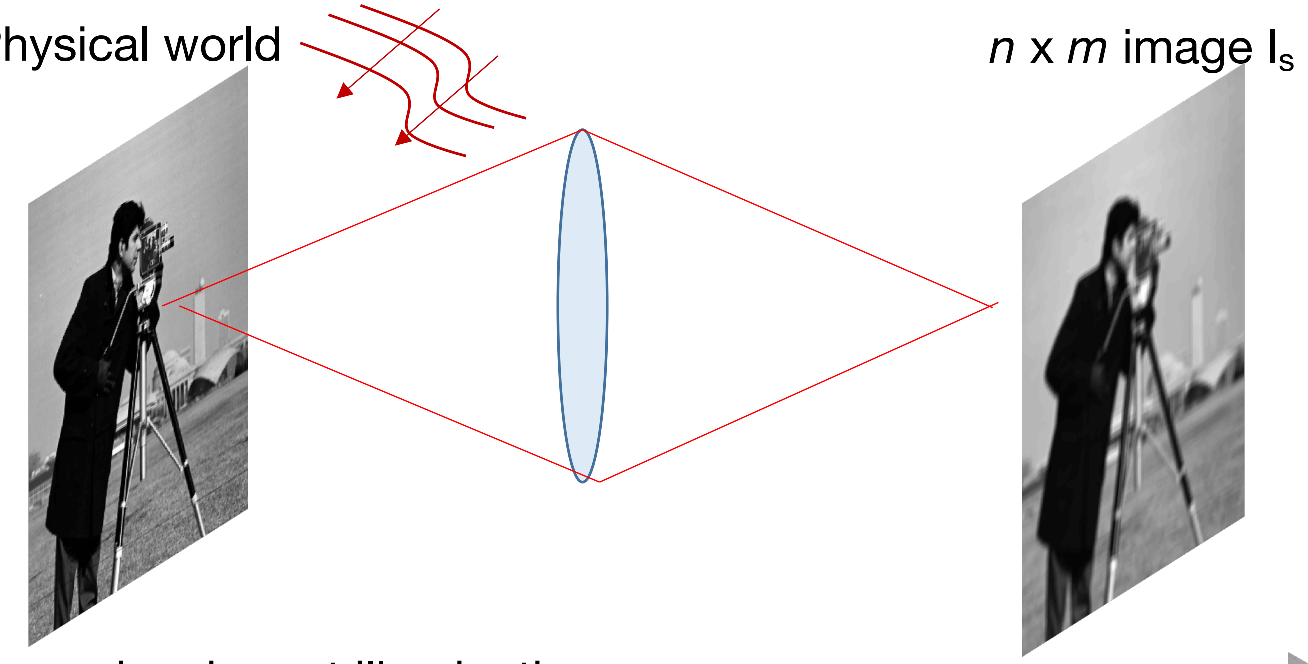
Simple mathematical model of image formation



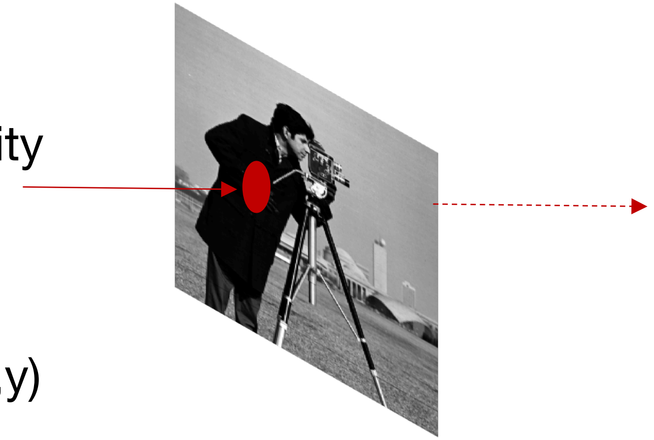
- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity



Simple mathematical model of image formation

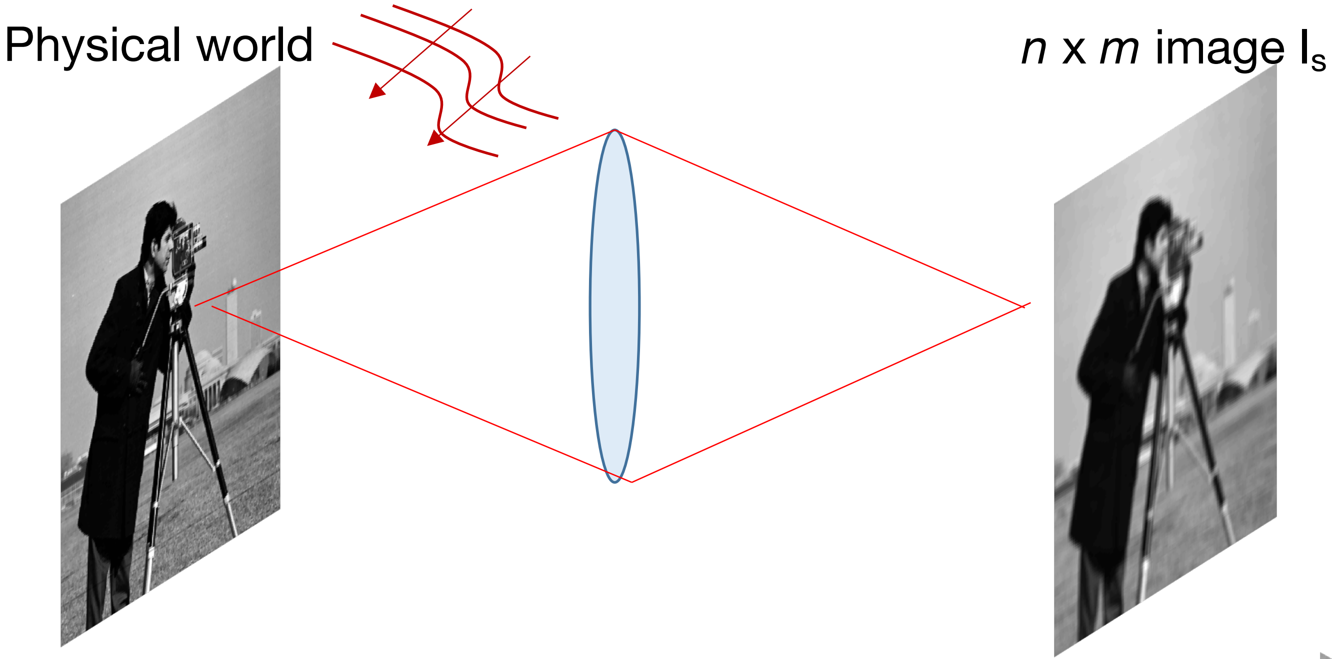


- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity

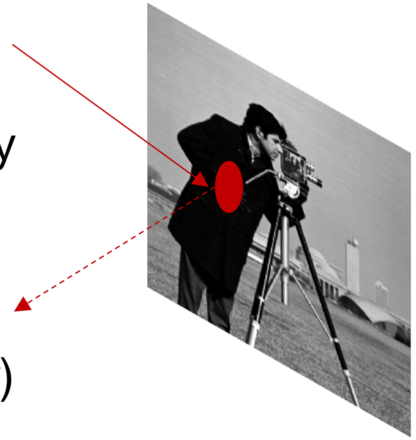


Object transmission: $\mathbf{I_0}(x,y)$
Illumination pattern: $\mathbf{s}(x,y)$
Light exiting object surface: $\mathbf{I_e}(x,y) = \mathbf{I_0}(x,y) \circ \mathbf{s}(x,y)$

Simple mathematical model of image formation

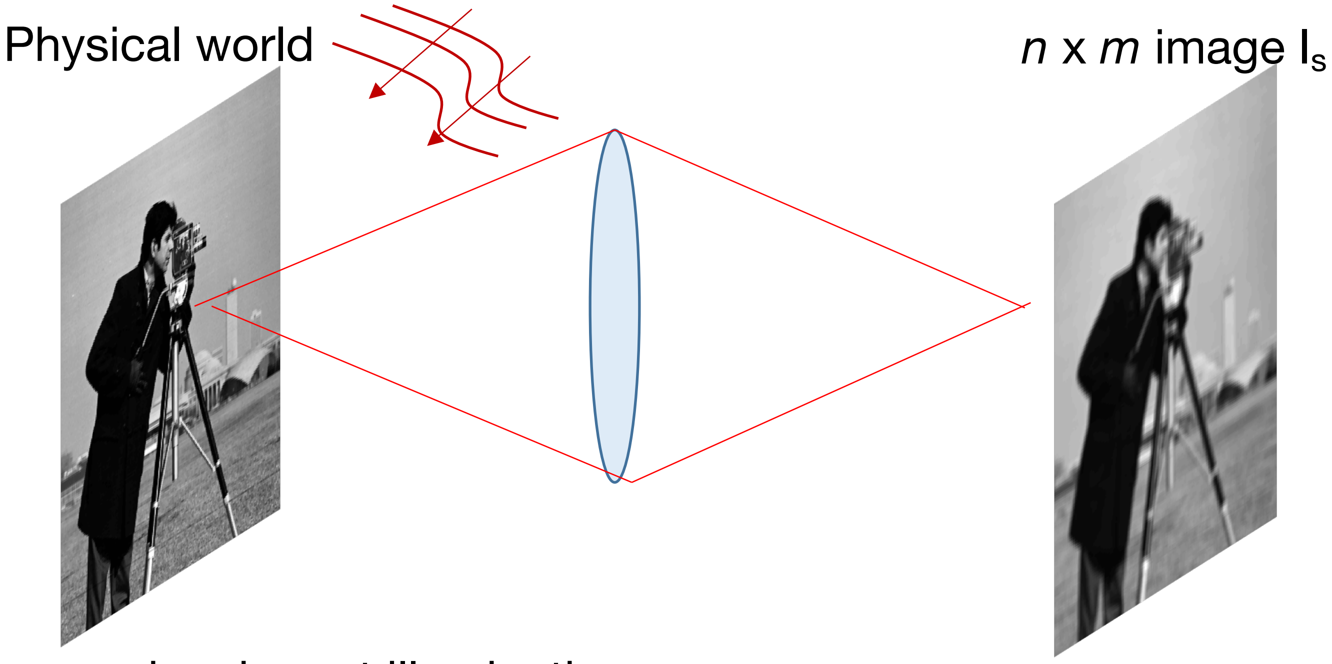


- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity



Object *reflectivity*: $\mathbf{I_0}(x,y)$
Illumination pattern: $\mathbf{s}(x,y)$
Light exiting object surface: $\mathbf{I_e}(x,y) = \mathbf{I_0}(x,y) \circ \mathbf{s}(x,y)$

Simple mathematical model of image formation



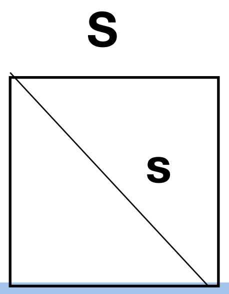
- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity

Modeling incoherent illumination

$$\mathbf{I}_e(x,y) = \mathbf{I}_0(x,y) \circ \mathbf{s}(x,y)$$

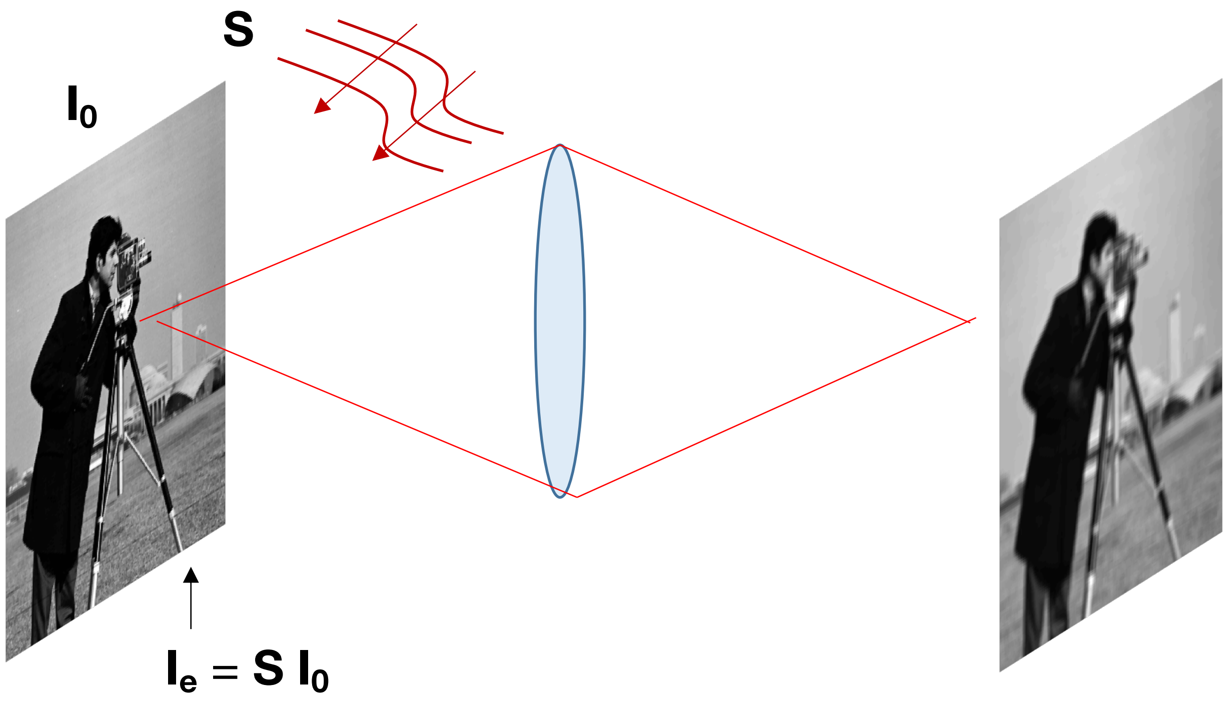
$$\mathbf{I}_e = \mathbf{S}\,\mathbf{I}_0$$

diag(**S**) = **s**



Example #1: Optimized illumination pattern (one color)

deep imaging

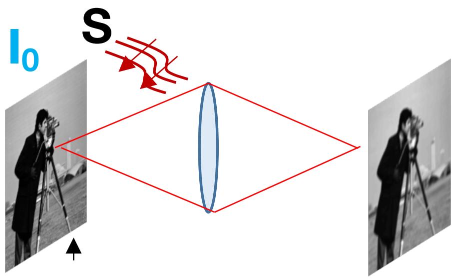


First, assume perfect camera:
intensity at image plane $\mathbf{I_p} = \mathbf{I_e} = \mathbf{S}\,\mathbf{I_0}$

Example #1: Optimized illumination pattern (one color)

Training data:
[$I_0(x, y)$, y]
I_0:100 x 100
Label y: 1x2

Example #1: Optimized illumination pattern (one color)



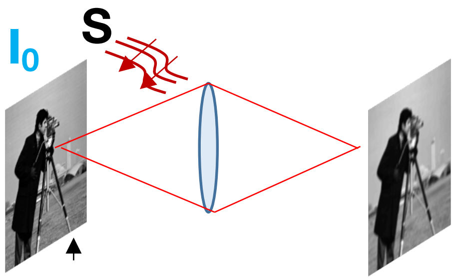
$[I_e(x, y), y]$

Training data:
$[I_0(x, y), y]$
I_0:100 x 100
Label y: 1x2

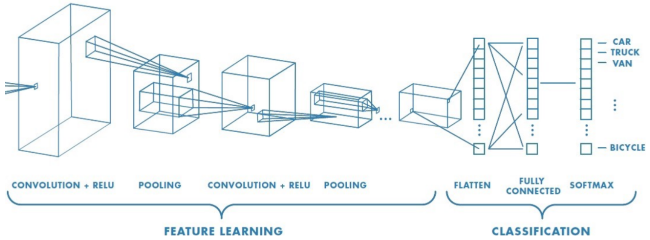
$\boxed{I_e(x,y) = \mathbf{S}\, I_0(x,y)}$

Physical Layer

Example #1: Optimized illumination pattern (one color)



$[I_e(x, y), y]$



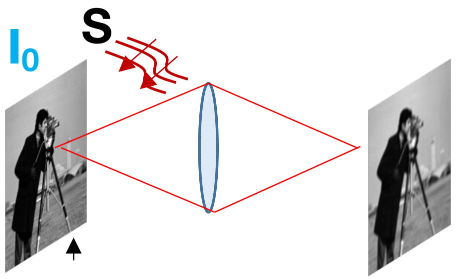
Training data:
$[I_0(x, y), y]$
I_0:100 x 100
Label y: 1x2

$\boxed{I_e(x,y) = \mathbf{S}\, I_0(x,y)}$

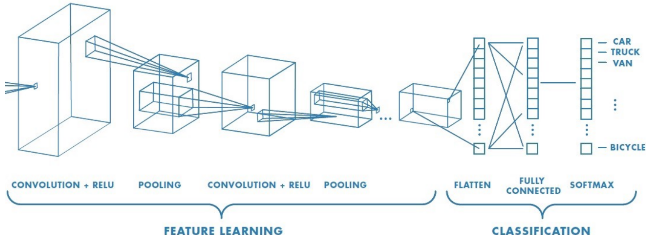
Physical Layer

Task = $\mathbf{W_n} \ldots \mathrm{ReLU}[\mathbf{W_1}\, \mathrm{ReLU}[\mathbf{W_0}\, I_e] \ldots]$

Example #1: Optimized illumination pattern (one color)



$[I_e(x, y), y]$



Training data:
$[I_0(x, y), y]$
I_0:100 x 100
Label y: 1x2

$\boxed{I_e(x,y) = \mathbf{S}\, I_0(x,y)}$

Physical Layer

Task = $\mathbf{W_n} \ldots \mathrm{ReLU}[\mathbf{W_1}\, \mathrm{ReLU}[\mathbf{W_0}\, I_e] \ldots]$

Tensorflow 1.0 code

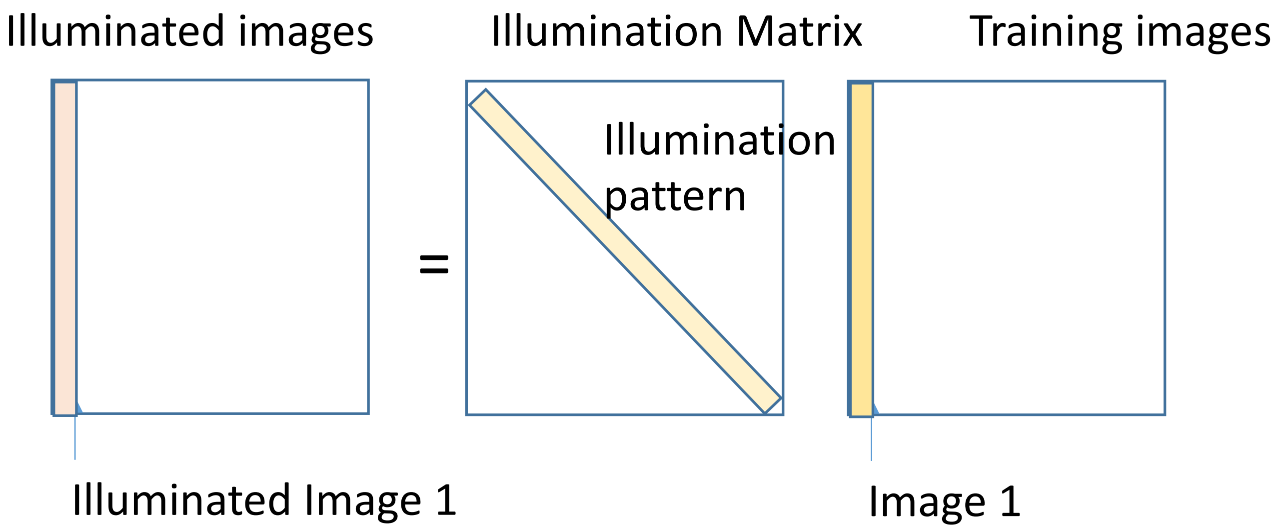
```
training_images = tf.placeholder(tf.float32, [image_size, None])
training_labels = tf.placeholder(tf.float32, [None, 3])
illumination_pattern = tf.truncated_normal([image_size, 1], stddev = 0.1)
illumination_matrix = tf.linalg.diag(illumination_pattern)
illumianted_images = tf.matmul(illumination_matrix, training_images)

#Now, train CNN of your choice using [illumianted_images, training_labels]
#Output of train/test will be CNN weights, classification performance AND illumination_pattern!
```

(Will show CoLab Notebook for implementation in Tensorflow 2.0)

Option 1: tf.linalg.matmul

1.0 code

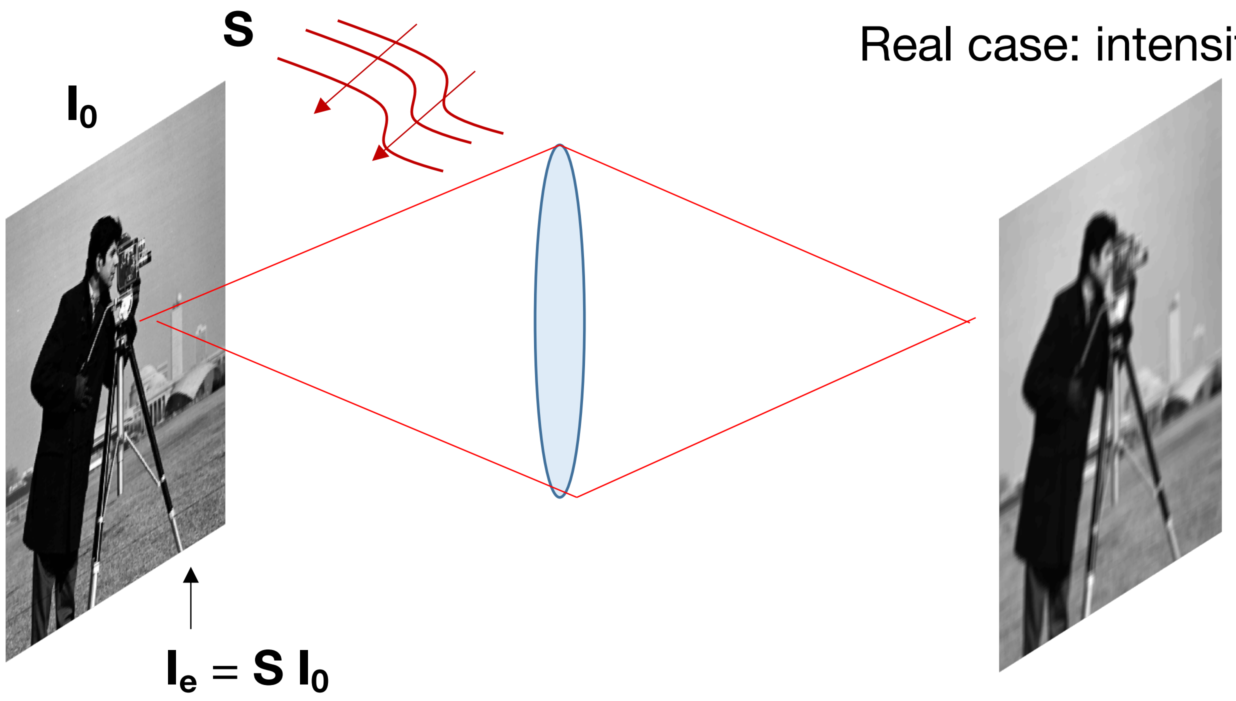
```
training_images = tf.placeholder(tf.float32, [image_size, None])
training_labels = tf.placeholder(tf.float32, [None, 3])
illumination_pattern = tf.truncated_normal([image_size, 1], stddev = 0.1)
illumination_matrix = tf.linalg.diag(illumination_pattern)
illumianted_images = tf.matmul(illumination_matrix, training_images)

#Now, train CNN of your choice using [illumianted_images, training_labels]
#Output of train/test will be CNN weights, classification performance AND illumination_pattern!
```



Option 2: tf.linalg.multiply (will show in CoLab Notebook)

Simple mathematical model of image formation

Real case: intensity at image plane $\mathbf{I_p}$ = **blurred**



Simple mathematical model of image formation

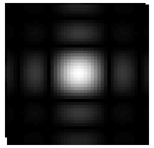
deep imaging

Lenses blur and rescale images:
(We'll learn how exactly next few weeks)

Input intensity

$*$

Convolution filter h



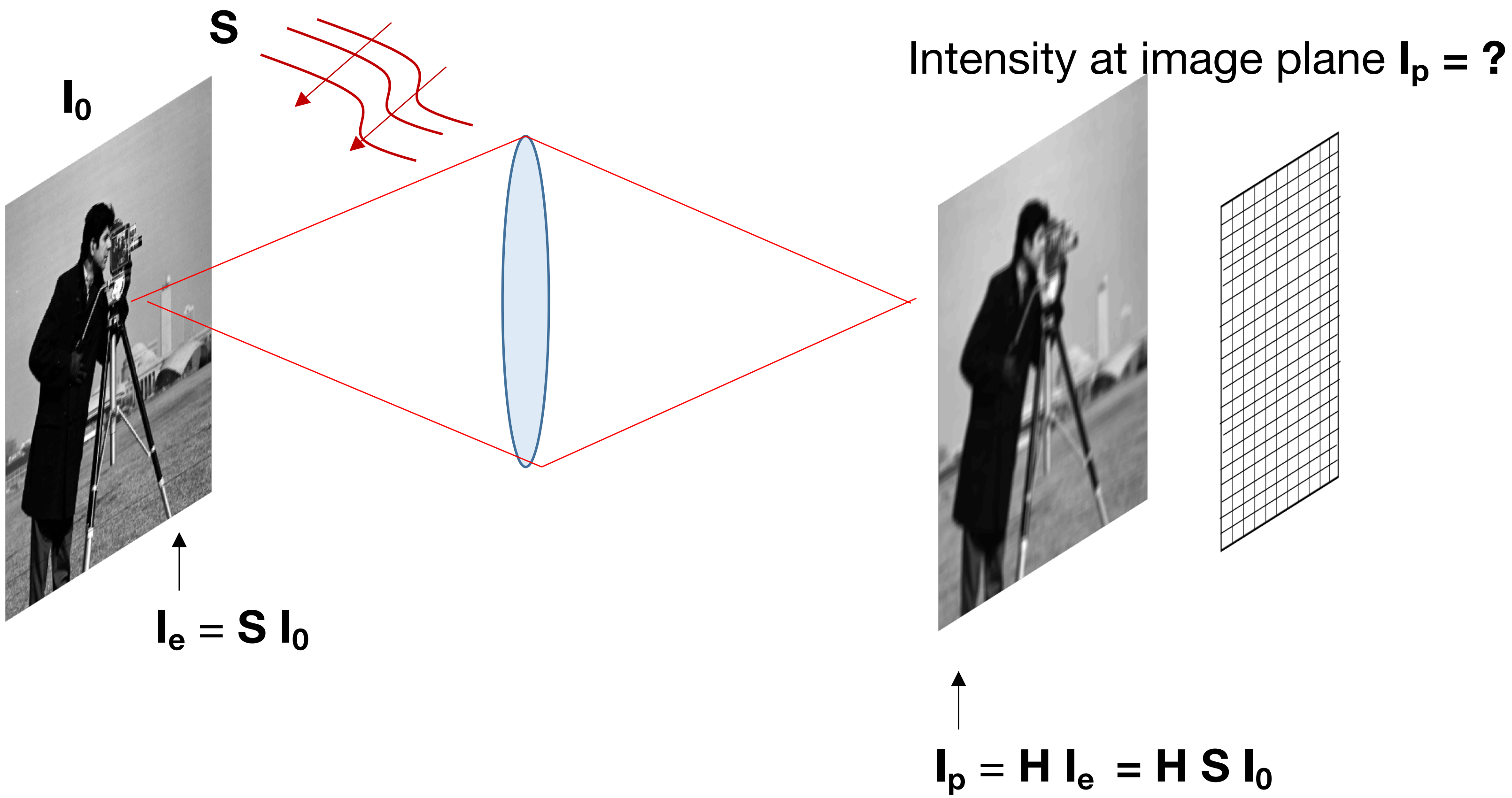
$=$

Output intensity

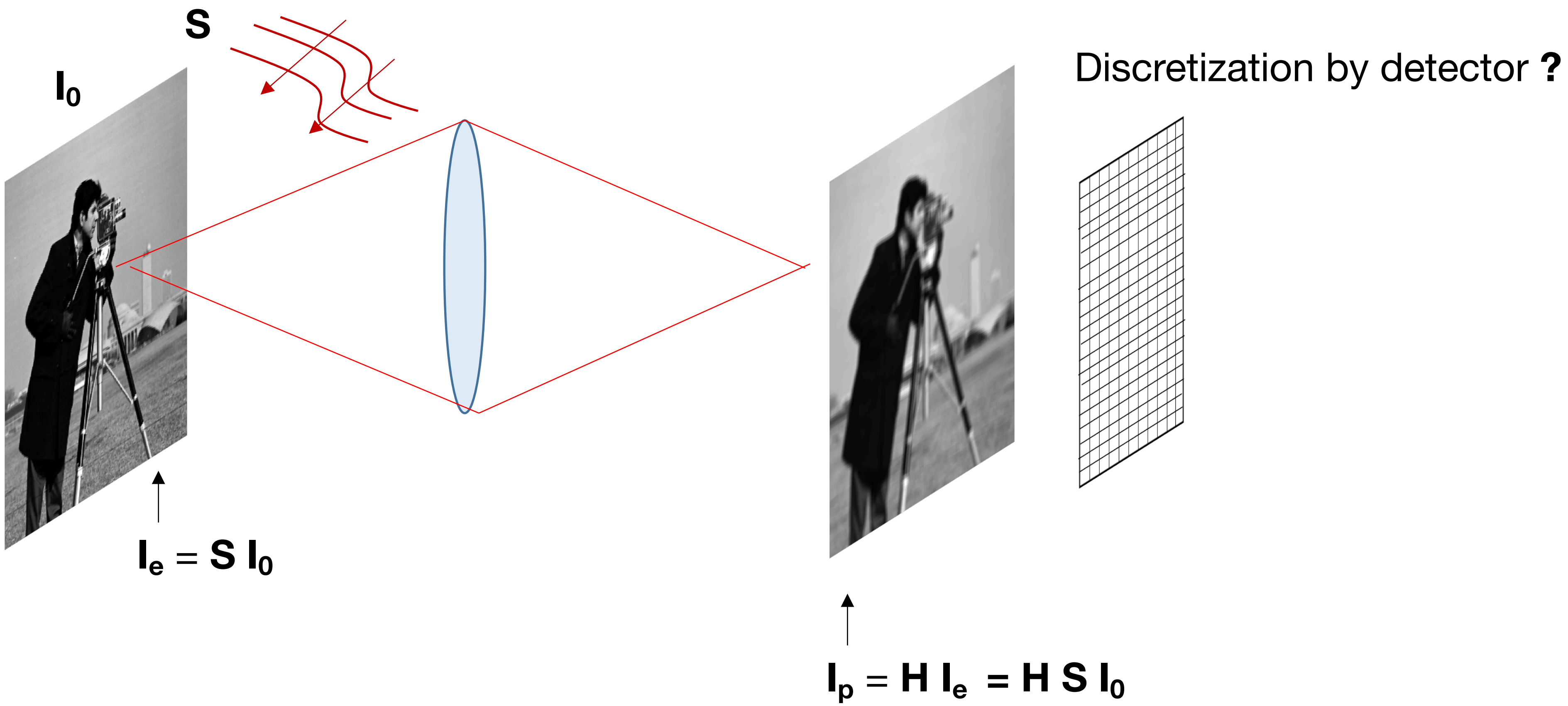
$$\mathbf{I_e}(x/M, y/M) \quad * \quad \mathbf{h}(x,y) \quad = \quad \mathbf{I_p}(x,y)$$

Assuming we've resized by M, $\mathbf{I_p} = \mathbf{I_e} * \mathbf{h} = \mathbf{H}\,\mathbf{I_e}$

Simple mathematical model of image formation



Simple mathematical model of image formation

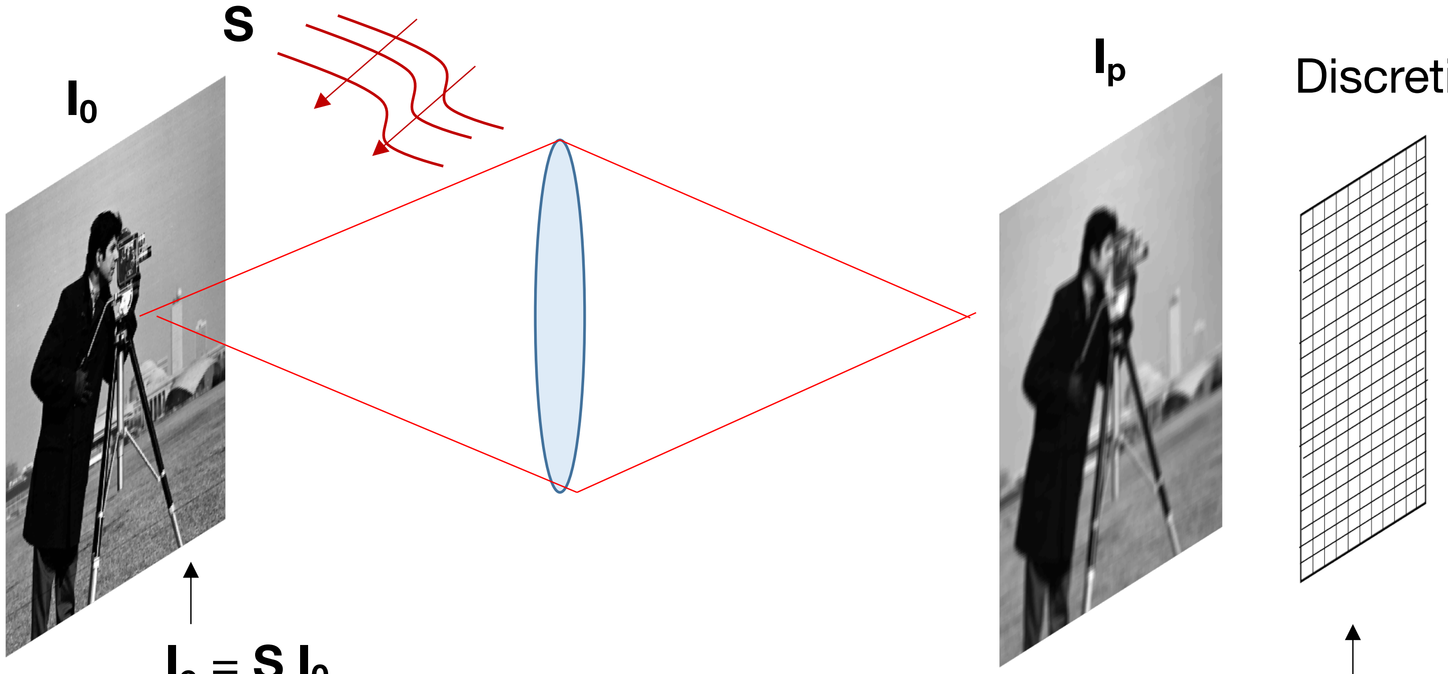


Use downsampling matrix (sum-pooling)

D =

$$\begin{matrix} 0.5 & 0.5 & 0 & 0 & 0 & \cdots & & \\ & & 0.5 & 0.5 & 0 & 0 & 0 & \cdots \\ & & & & 0.5 & 0.5 & 0 & 0 & 0 & \cdots \\ & \vdots & & & & \vdots \end{matrix}$$

Simple mathematical model of image formation



$I_e = S\,I_0$

Discretization by detector ?

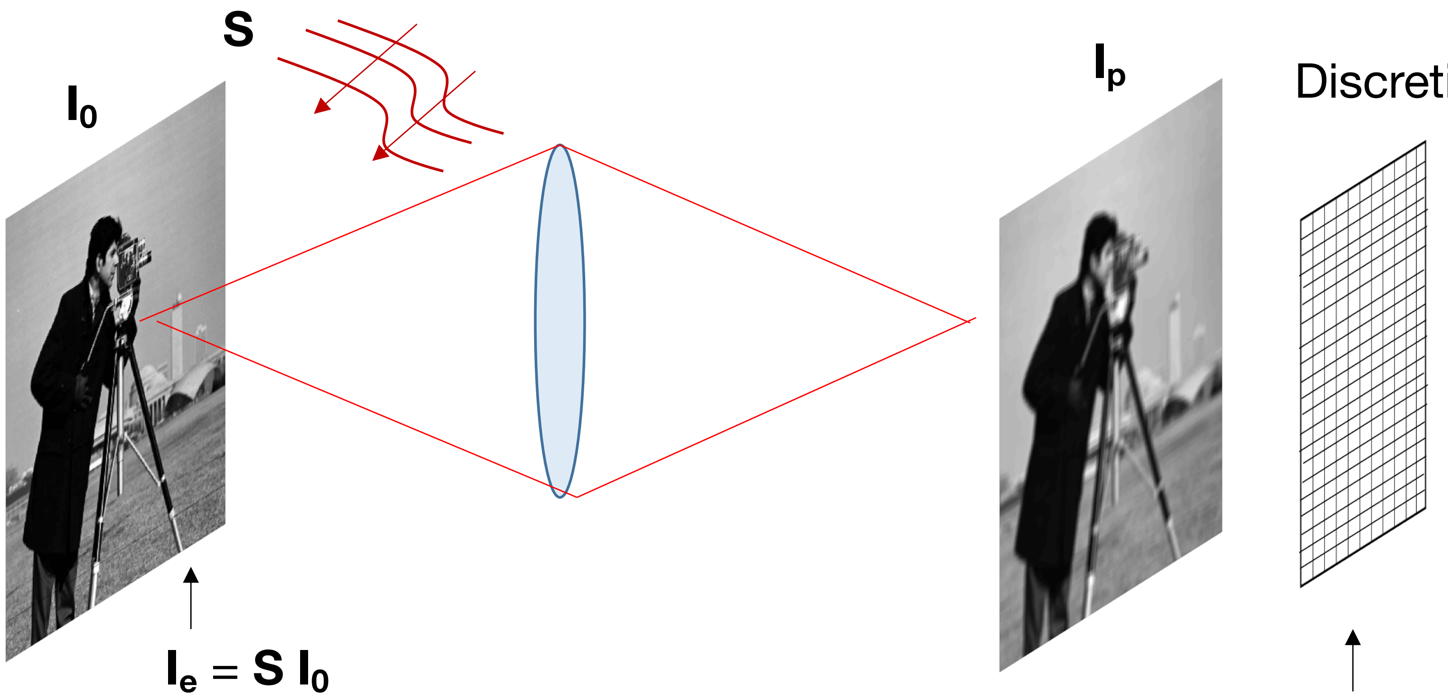
$$\boxed{I_s = D\,I_p = D\,H\,S\,I_0}$$

Use downsampling matrix (sum-pooling)

$$D = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & \dots & & & & \\ & & 0.5 & 0.5 & 0 & 0 & 0 & \dots & & \\ & & & & 0.5 & 0.5 & 0 & 0 & 0 & \dots \\ & & \vdots & & & & \vdots & & & \end{bmatrix}$$

Simple mathematical model of image formation

deep imaging



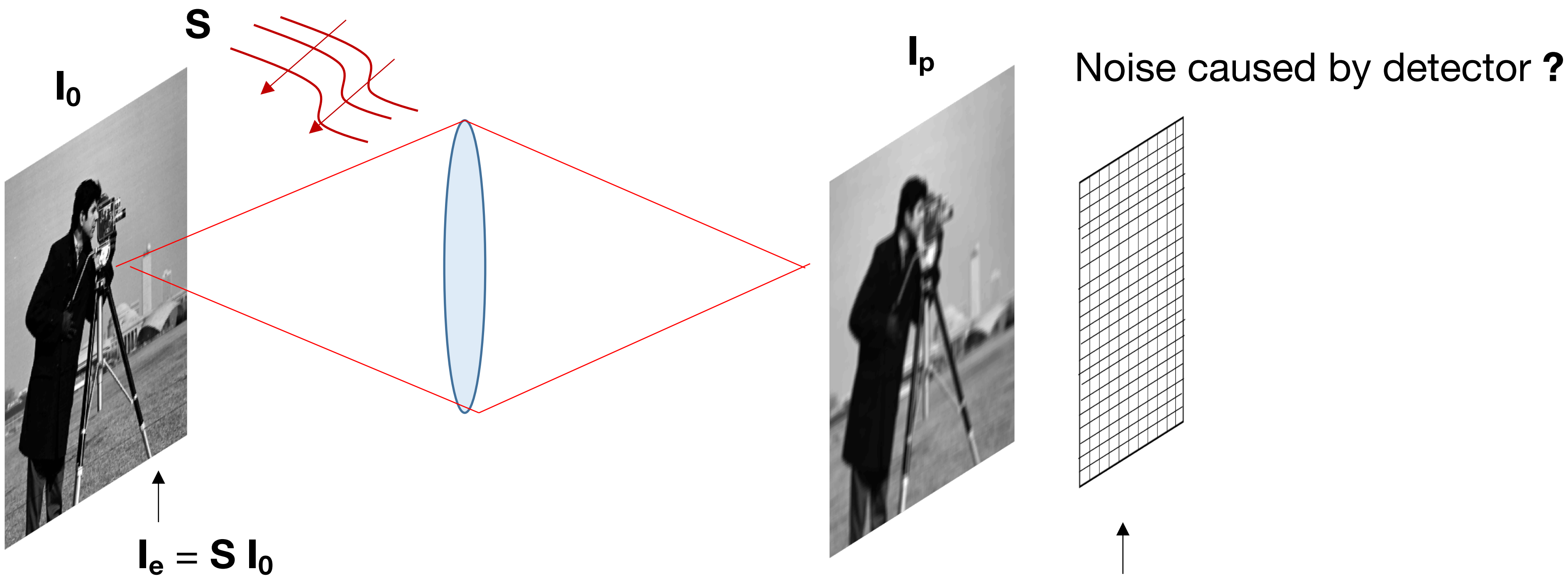
$$\boxed{I_s = D\, I_p = D\, H\, S\, I_0}$$

D =

w1 w2 0 0 0 ….
w3 w4 0 0 0 ….
w5 e6 0 0 0 ….
⋮ ⋮

*Can make these learnable weights

Simple mathematical model of image formation



$$I_s = D\ I_p + N$$

Can also add in detector-dependent noise N = k * np.random.randn(dx, dy)

(zero-mean Gaussian noise, for example)

Pause to take a look at:

physical_layers_example.ipynb

Physical Layers

Digital Layers

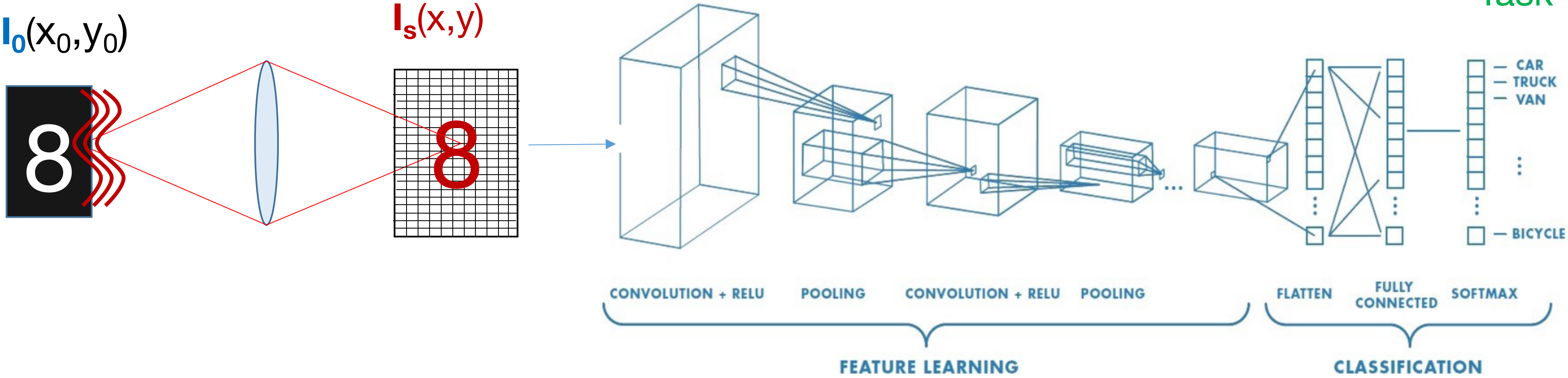
Physical world

$\mathbf{I_0}(x_0,y_0)$

Digital Image

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

Task



Digital layers

Physical layers

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

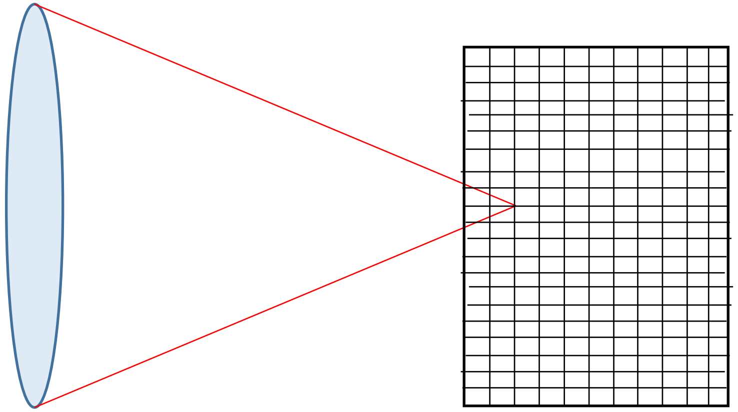
$$\text{Task} = \mathbf{W_n} \ldots \text{ReLU}[\mathbf{W_1}\ \text{ReLU}[\mathbf{W_0}\ \mathbf{D}\ \mathbf{H}\ \mathbf{S}\ \mathbf{I_0}]\ldots]$$

Example #2: Optimized color filter for a grayscale camera

“Ground truth” object:
$I_0(x, y, \lambda)$
100 x 100 pix. x 30 spectral channels

Example #2: Optimized color filter for a grayscale camera

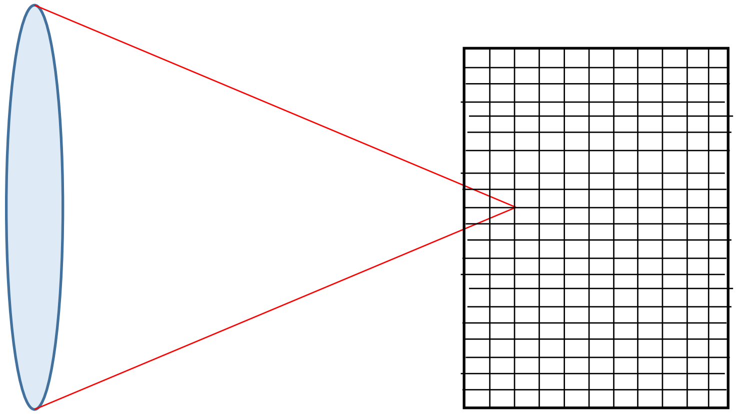
deep imaging



Monochromatic camera sensor

“Ground truth” object:
$I_0(x, y, \lambda)$
100 x 100 pix. x 30 spectral channels

Example #2: Optimized color filter for a grayscale camera



Monochromatic camera sensor

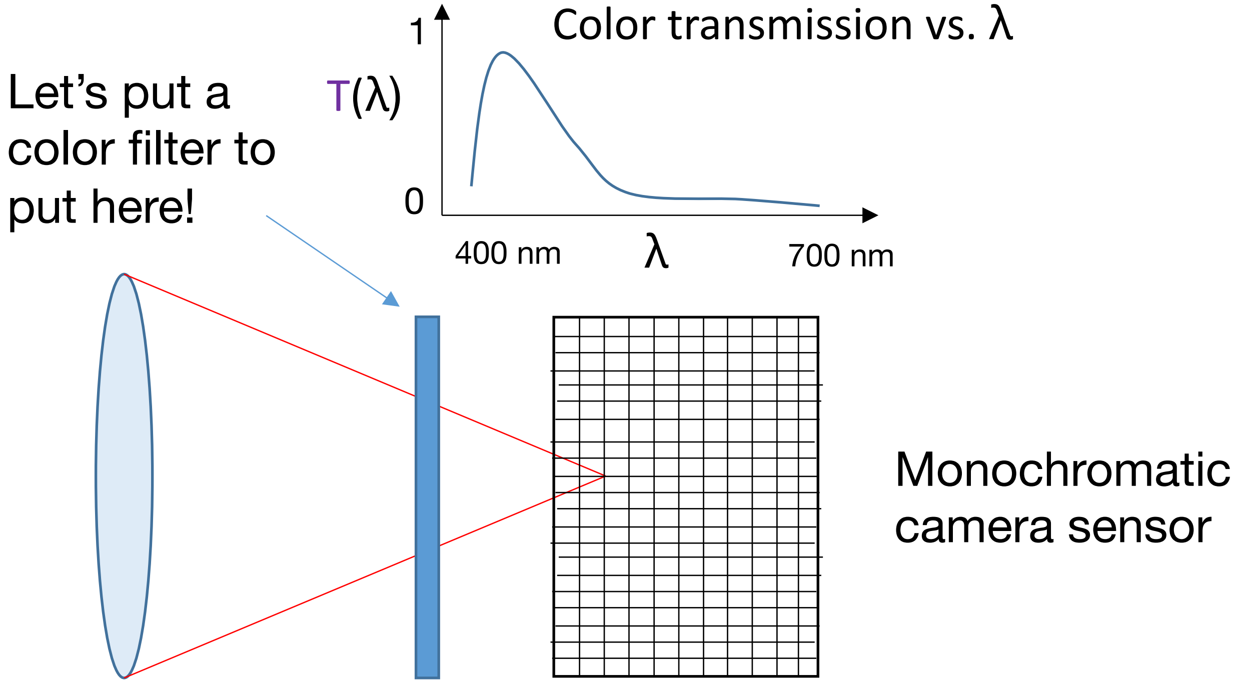
"Ground truth" object:

$I_0(x, y, \lambda)$

100 x 100 pix. x 30 spectral channels

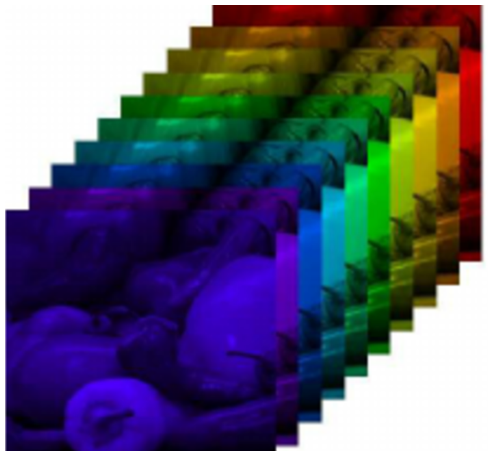
$$I_s(x, y) = \sum_{\lambda} I_0(x, y, \lambda)$$

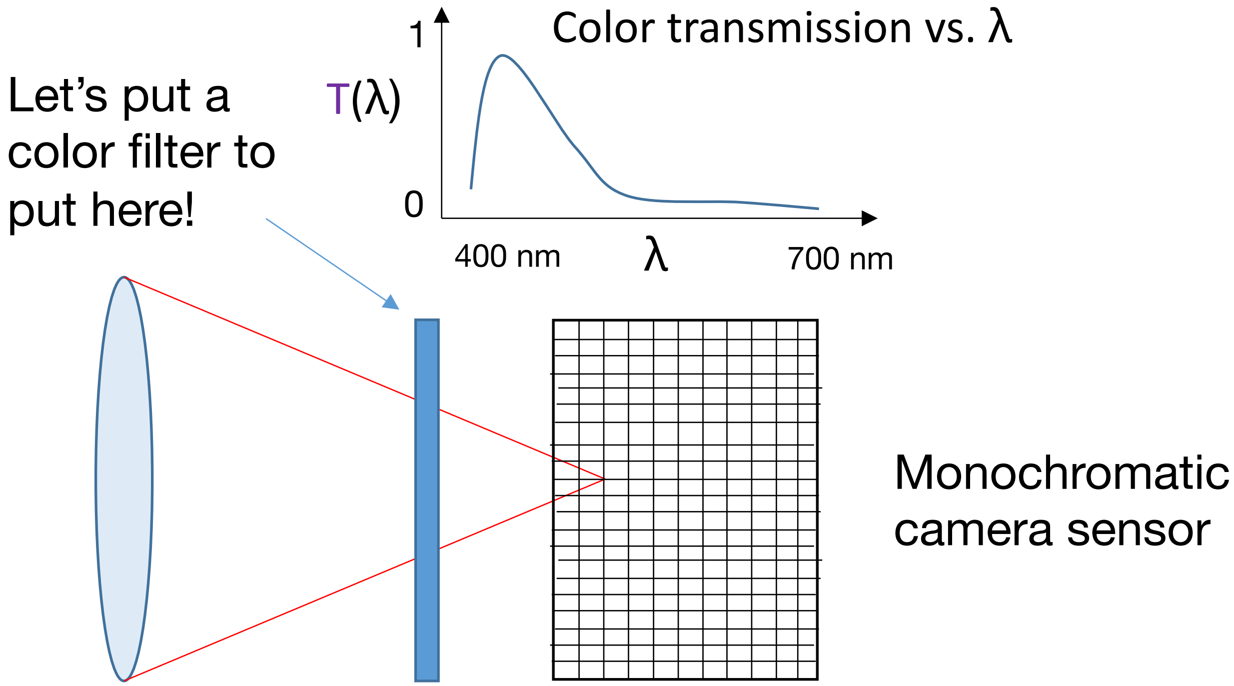
Example #2: Optimized color filter for a grayscale camera



"Ground truth" object:
$I_0(x, y, \lambda)$
100 x 100 pix. x 30 spectral channels

Example #2: Optimized color filter for a grayscale camera



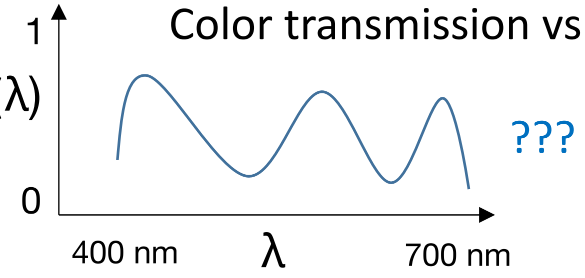


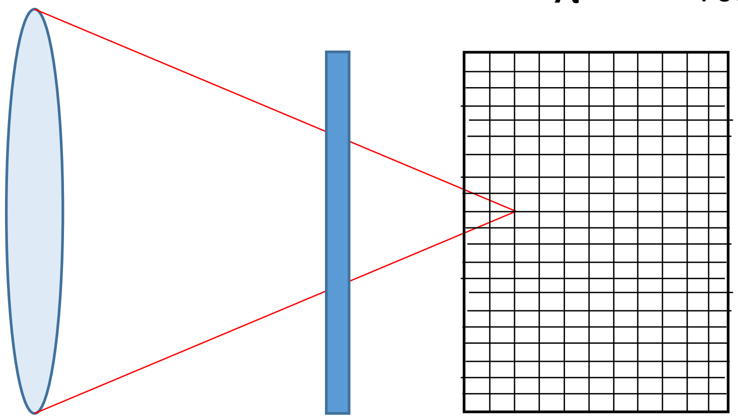
"Ground truth" object:
$I_0(x, y, \lambda)$
100 x 100 pix. x 30 spectral channels

$$\boxed{I_s(x, y) = \sum_{\lambda} T(\lambda)\, I_0(x, y, \lambda)}$$

Example #2: Optimized color filter for a grayscale camera

Design optimal color filter for classification:





Monochromatic camera sensor

Training data:
[$I_0(x, y, \lambda)$, y]
I_0:100 x 100 pix. x 30
Label y: 1x3 - pepper, broccoli, green beans

$$I_s(x, y) = \sum_{\lambda} W_0(\lambda)\, I_0(x, y, \lambda)$$

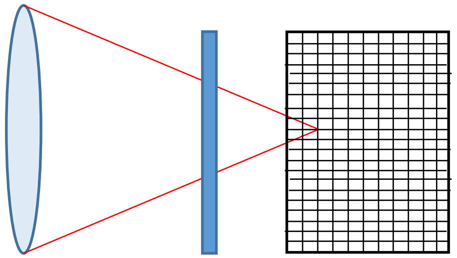
Physical Layer

Example #2: Optimized color filter for a grayscale camera

Training data:
[$\mathbf{I}_0(x, y, \lambda)$, y]
$\mathbf{I}_0$:100 x 100 x 30
Label y: 1x3

Example #2: Optimized color filter for a grayscale camera

deep imaging

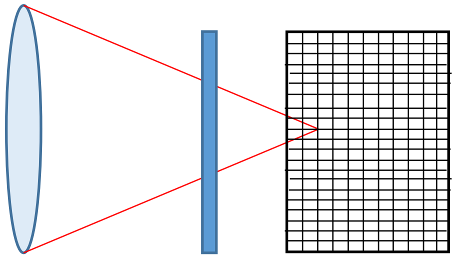


Training data:
[$I_0(x, y, \lambda)$, y]
I_0: 100 x 100 x 30
Label y: 1x3

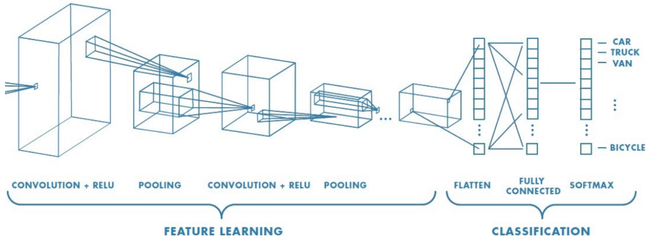
$$I_s(x, y) = \sum_{\lambda} W_0(\lambda)\, I_0(x, y, \lambda)$$

Physical Layer

Example #2: Optimized color filter for a grayscale camera



$[I_s(x, y), y]$



Training data:

$[I_0(x, y, \lambda), y]$

I_0: 100 x 100 x 30

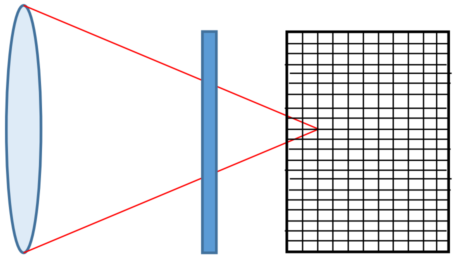
Label y: 1x3

$$\boxed{I_s(x, y) = \sum_{\lambda} W_0(\lambda)\, I_0(x, y, \lambda)}$$

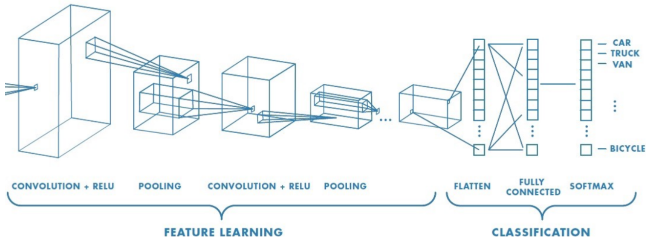
Physical Layer

$$\text{Task} = \mathbf{W_n} \ldots \mathrm{ReLU}[\mathbf{W_1}\, \mathrm{ReLU}[\mathbf{W_0}\, I_s] \ldots]$$

Example #2: Optimized color filter for a grayscale camera



$[I_s(x, y), y]$



Training data:
$[I_0(x, y, \lambda), y]$
I_0: 100 x 100 x 30
Label y: 1x3

$$I_s(x, y) = \sum_{\lambda} W_0(\lambda)\, I_0(x, y, \lambda)$$

Physical Layer

Task = $\mathbf{W_n} \ldots \mathrm{ReLU}[\mathbf{W_1}\, \mathrm{ReLU}[\mathbf{W_0}\, I_s] \ldots]$

```
multispectral_data = tf.placeholder(tf.float32, [None, num_colors, image_size])
veg_labels = tf.placeholder(tf.float32, [None, 3])
filter_weights = tf.truncated_normal([num_colors, 1], stddev = 0.1)
filtered_images = tf.einsum('aij,jk->aik', multispectral_data, filter_weights)

#Now, train CNN of your choice using [filtered_images, veg_labels]
#Output of training/testing will be CNN weights, classification performance AND filter_weights!
```

Example implementation with Tensorflow 1.0 code

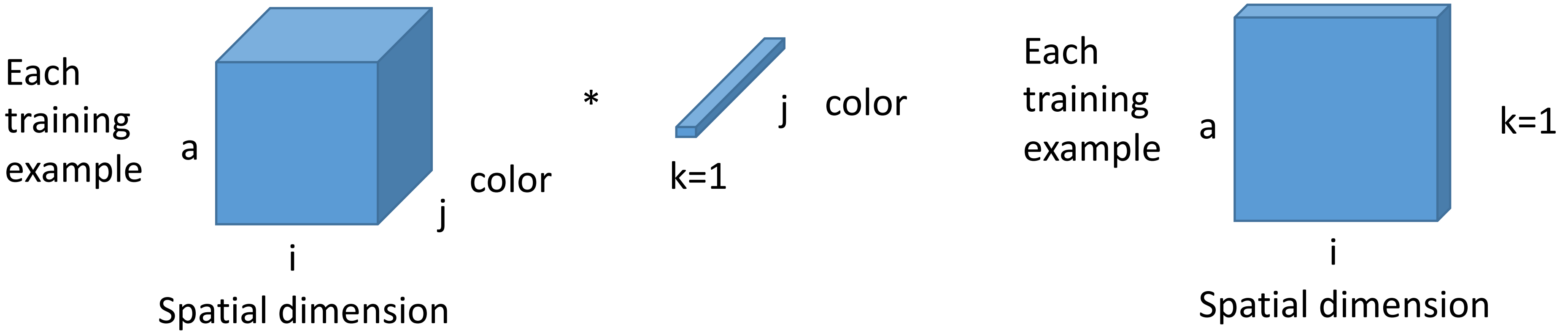
Tensorflow: operations to sum along 3rd (or higher) dimension

Option 1: Einsum (shown as Tensorflow 1.0 code, and also applicable in Tensorflow 2.0)

```
filtered_images = tf.einsum('aij,jk->aik', multispectral_data, filter_weights)
```

Option 2: tf.reduce_sum

```
filtered_images = tf.reduce_sum(multispectral_data * filter_weights, axis=2)
```



Option 3: Locally connected conv2D with a 1x1 filter size

https://github.com/keras-team/keras/blob/master/keras/layers/local.py#L183

Pause to take a look at:

weighted_image_sum_example.ipynb

Summary: simple physical layers for incoherent imaging

deep imaging

- Deal with sample/image intensities I, real and non-negative
- Effect of illumination is element-wise multiplication $I_e(x,y) = \mathbf{S}\, I_0(x,y)$

λ

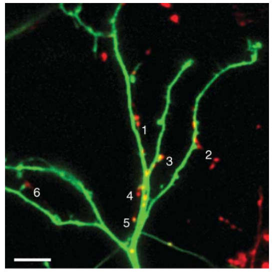
- Imaging systems blur the object via point-spread function matrix **H**

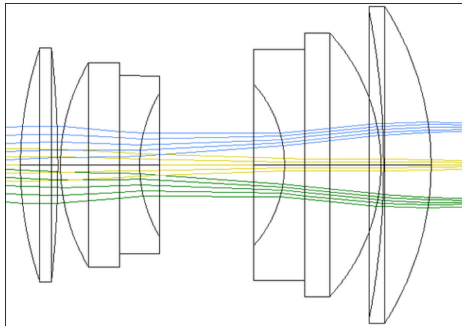
$I_b(x,y) = \mathbf{H}\, I_0(x,y)$

- Discrete pixels down-sample the object via $I_d(x,y) = \mathbf{D}\, I_0(x,y)$
- Add noise into measurement $I_N(x,y) = \mathbf{D}\, I_0(x,y) + \mathbf{N}$
- Different colors add linearly $I_s(x, y) = \Sigma\, I_0(x, y, \lambda)$

What is light and how can we model it?

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

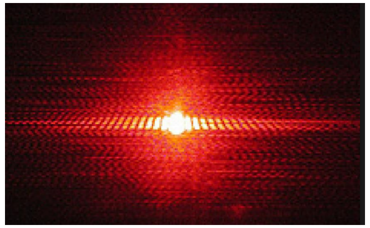


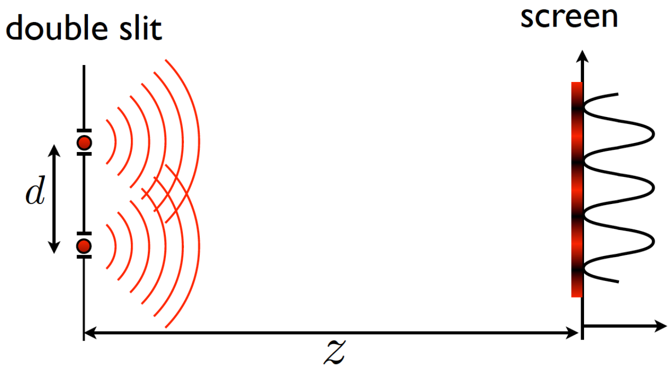


- Real, non-negative
- Models absorption and brightness

$$I_{tot} = I_1 + I_2$$

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





- Complex field
- Models Interference

$$E_{tot} = E_1 + E_2$$

Next class: Modeling coherent radiation as a wave