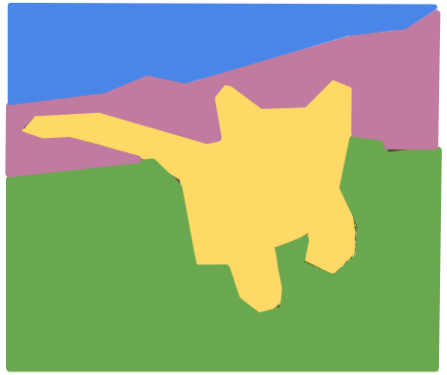
Machine Learning in Imaging

BME 590L
Roarke Horstmeyer

Lecture 15: Introducing the physical layers of a CNN

Other Computer Vision Tasks

Semantic Segmentation

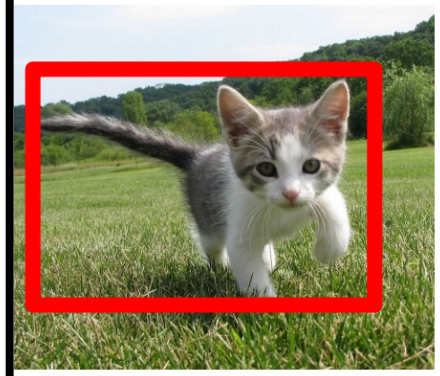


GRASS, CAT, TREE, SKY

No objects, just pixels

Balanced equation

Classification + Localization

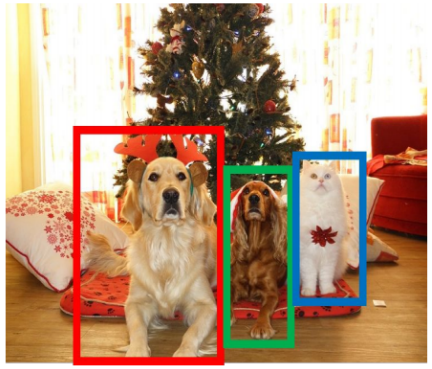


CAT

Single Object

Over-determined

Object Detection

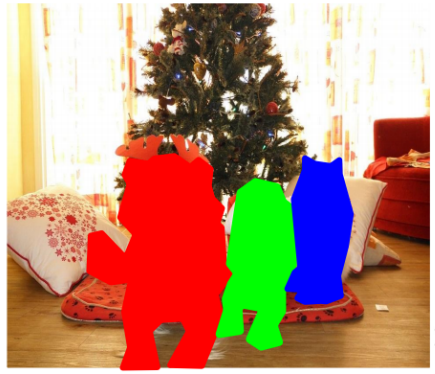


DOG, DOG, CAT

Multiple Object

Over-determined

Instance Segmentation



DOG, DOG, CAT

Over-determined

Super-resolution

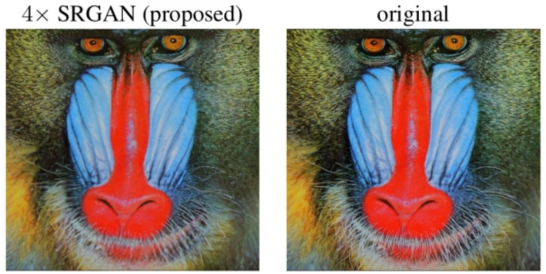


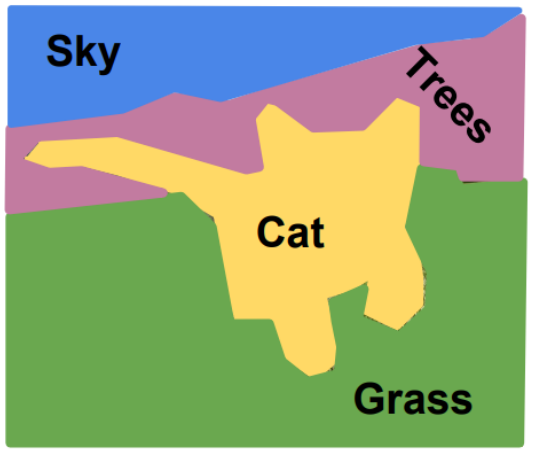
Figure 1: Super-resolved image (left) is almost indistinguishable from original (right). [4× upscaling]

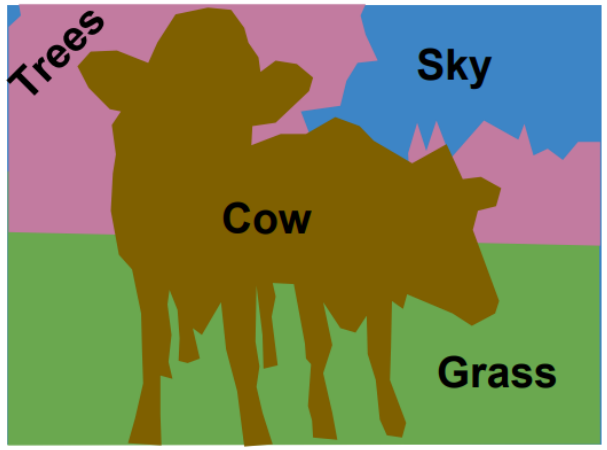
Under-determined

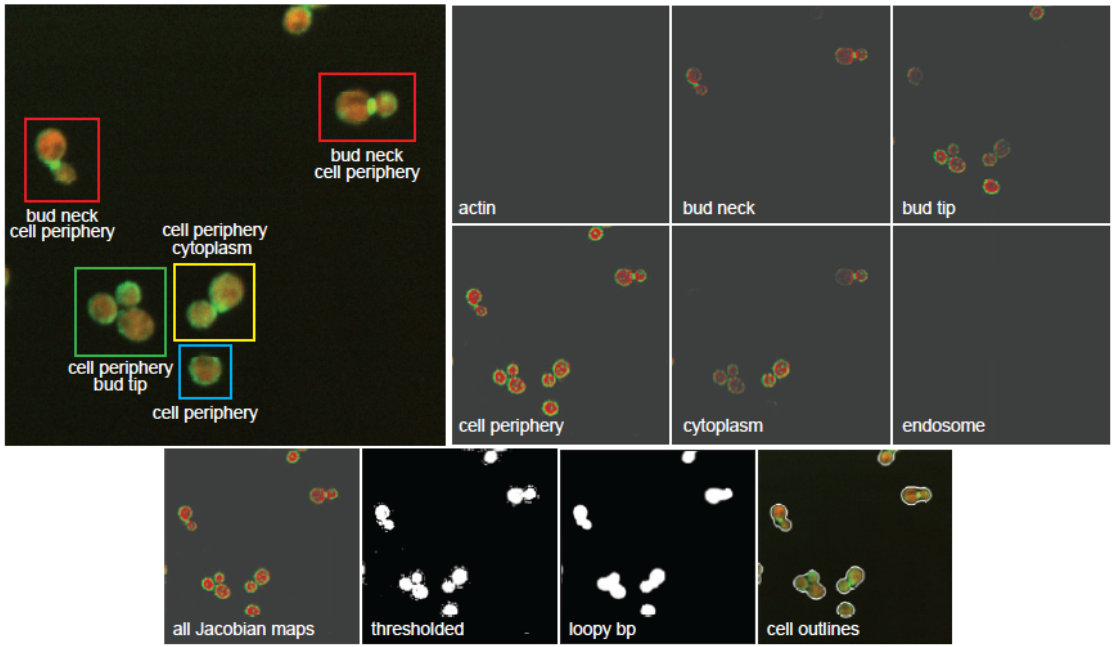
Semantic Segmentation

Label each pixel in the image with a category label

Don’t differentiate instances, only care about pixels

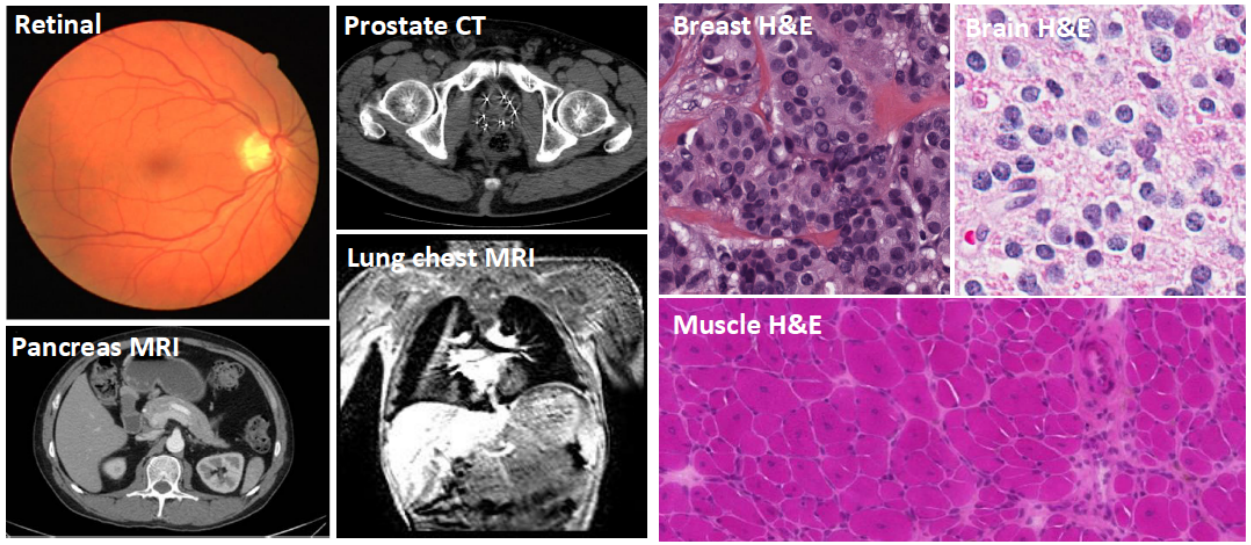


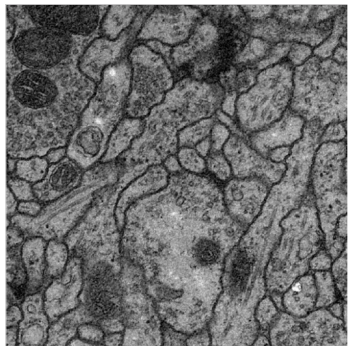




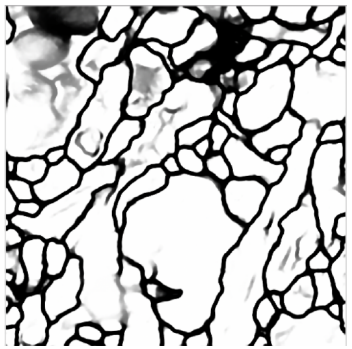
Oren Z. Kraus et al., "Classifying and Segmenting Microscopy Images Using Convolutional Multiple Instance Learning," arXiv 2015

Other possible examples:

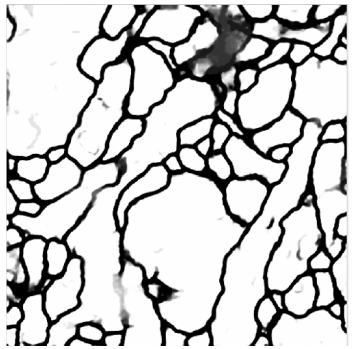




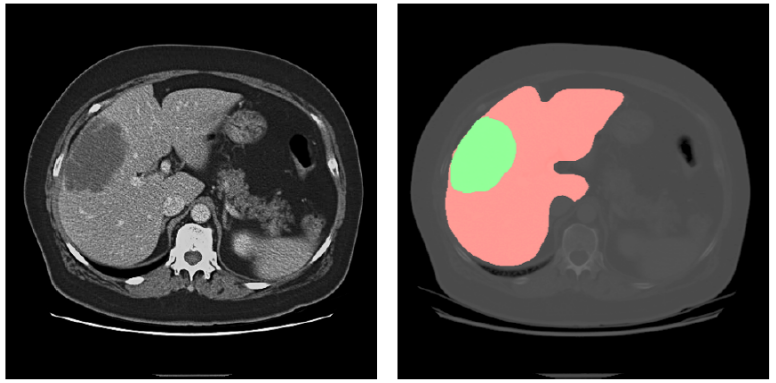
(a) Input image



(b) FC-ResNet with dropout at test time [17]



(c) Segmentation result of our pipeline



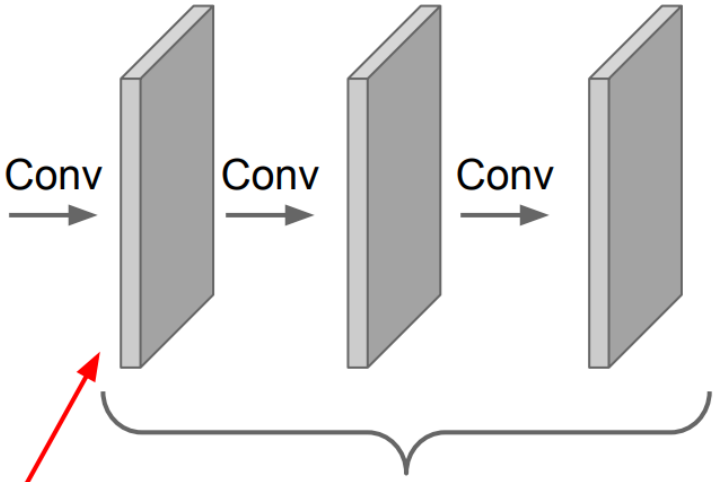
M. Drozdzal et al., Learning Normalized Inputs for Iterative Estimation in Medical Image Segmentation (2017)

Z. Zhang et al., Recent Advances in the Applications of Convolutional Neural Networks to Medical Image Contour Detection (2017)

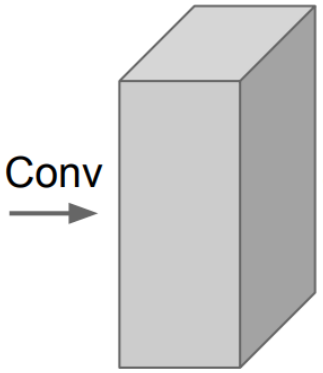
Semantic Segmentation Idea: Fully Convolutional ?

Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!

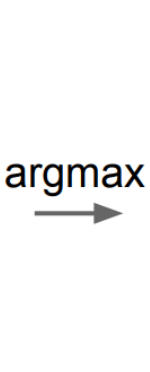
Input:
3 x H x W



Convolutions:
D x H x W



Scores:
C x H x W

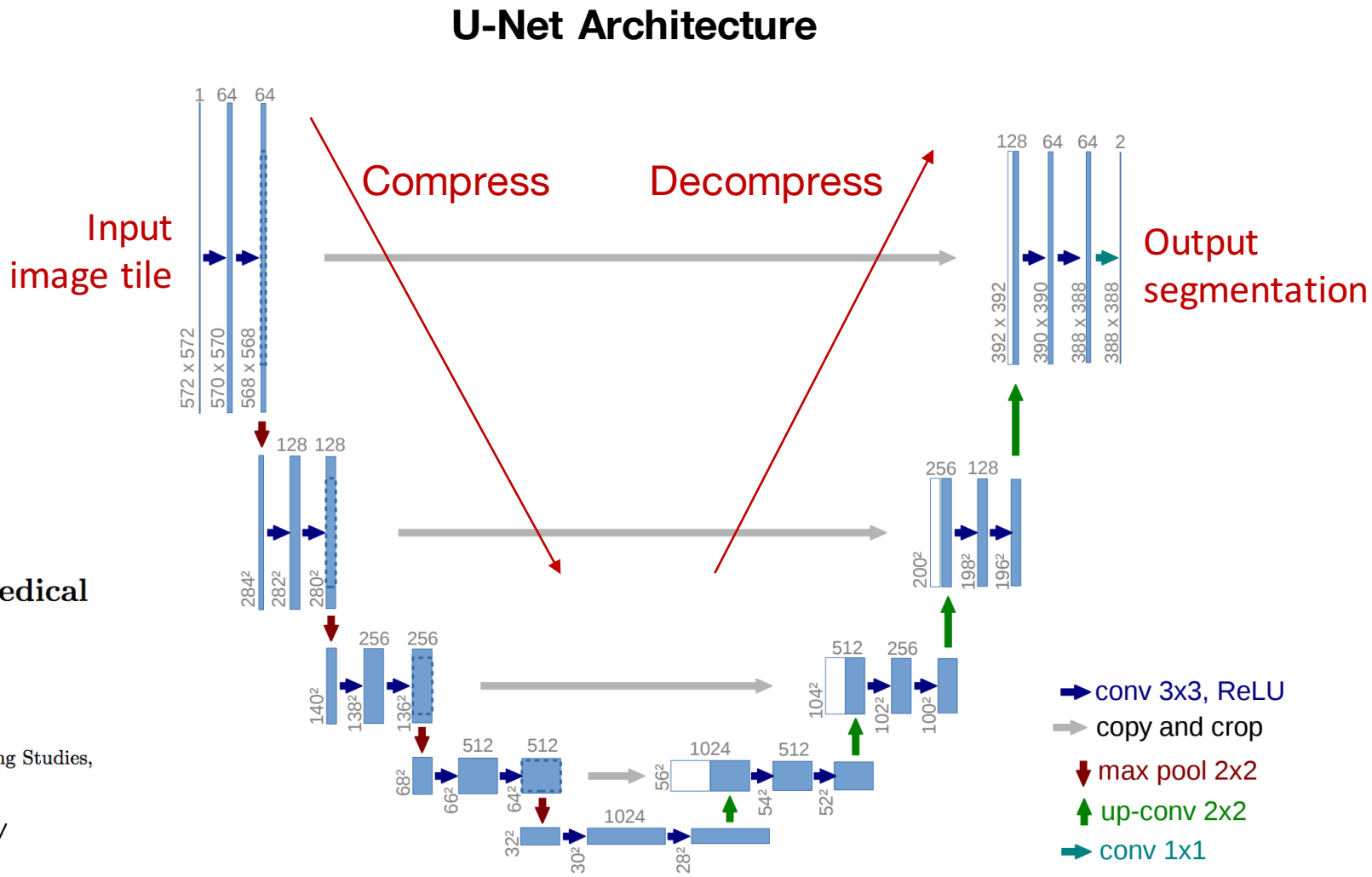


Predictions:
H x W

Problem: convolutions at original image resolution will be very expensive ...

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

- Compress spatial features into learned filters
- Then, decompress learned filters back into same spatial dimensions



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/

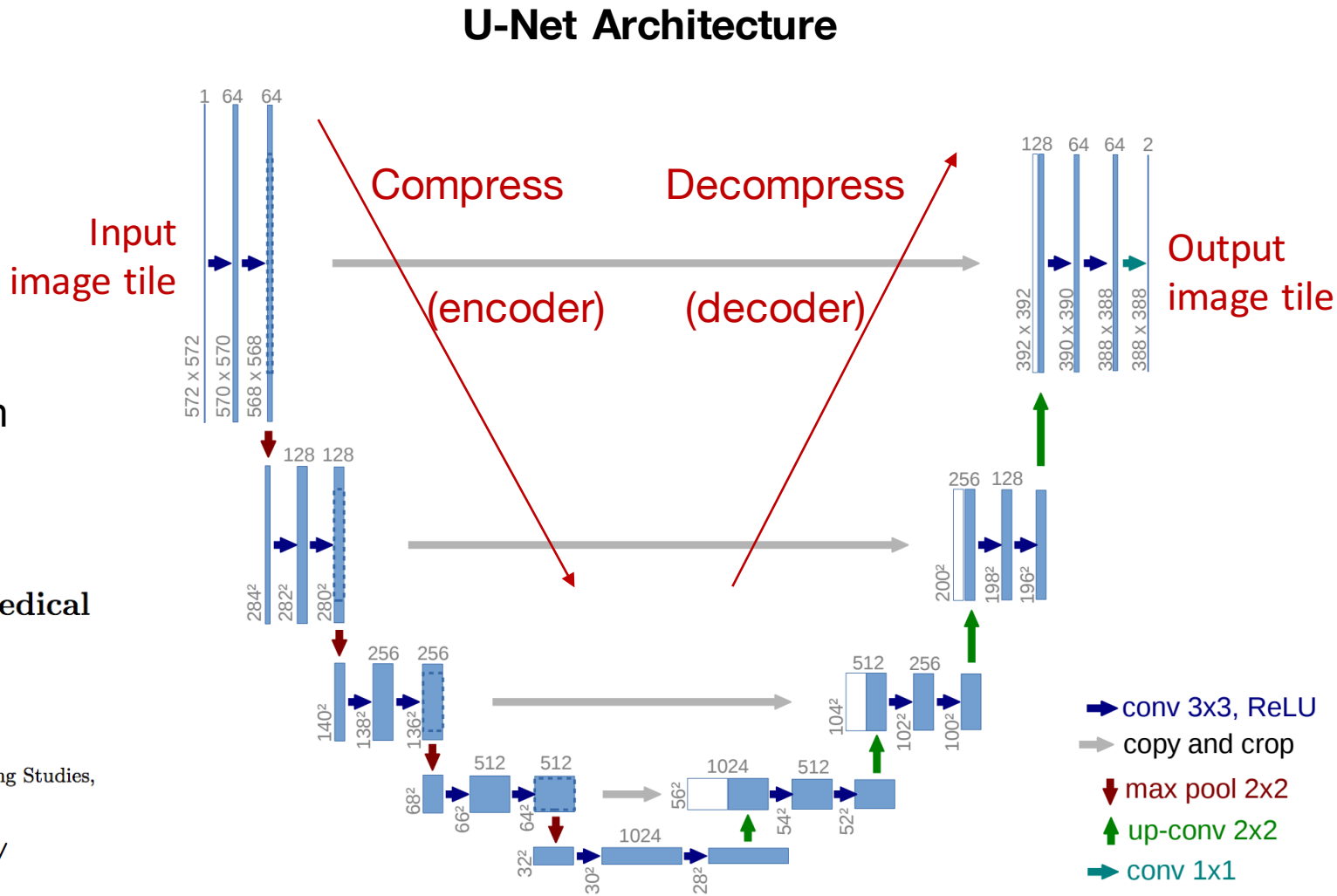
Instead, *compress x-y dimensions of* input image

- 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…

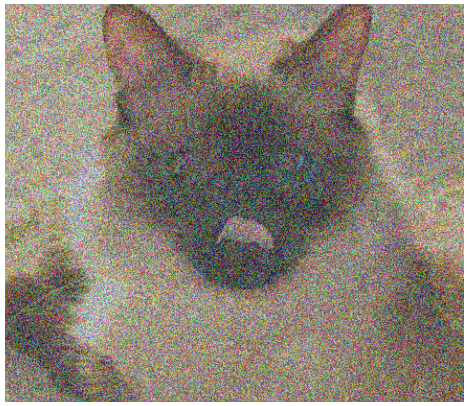
U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

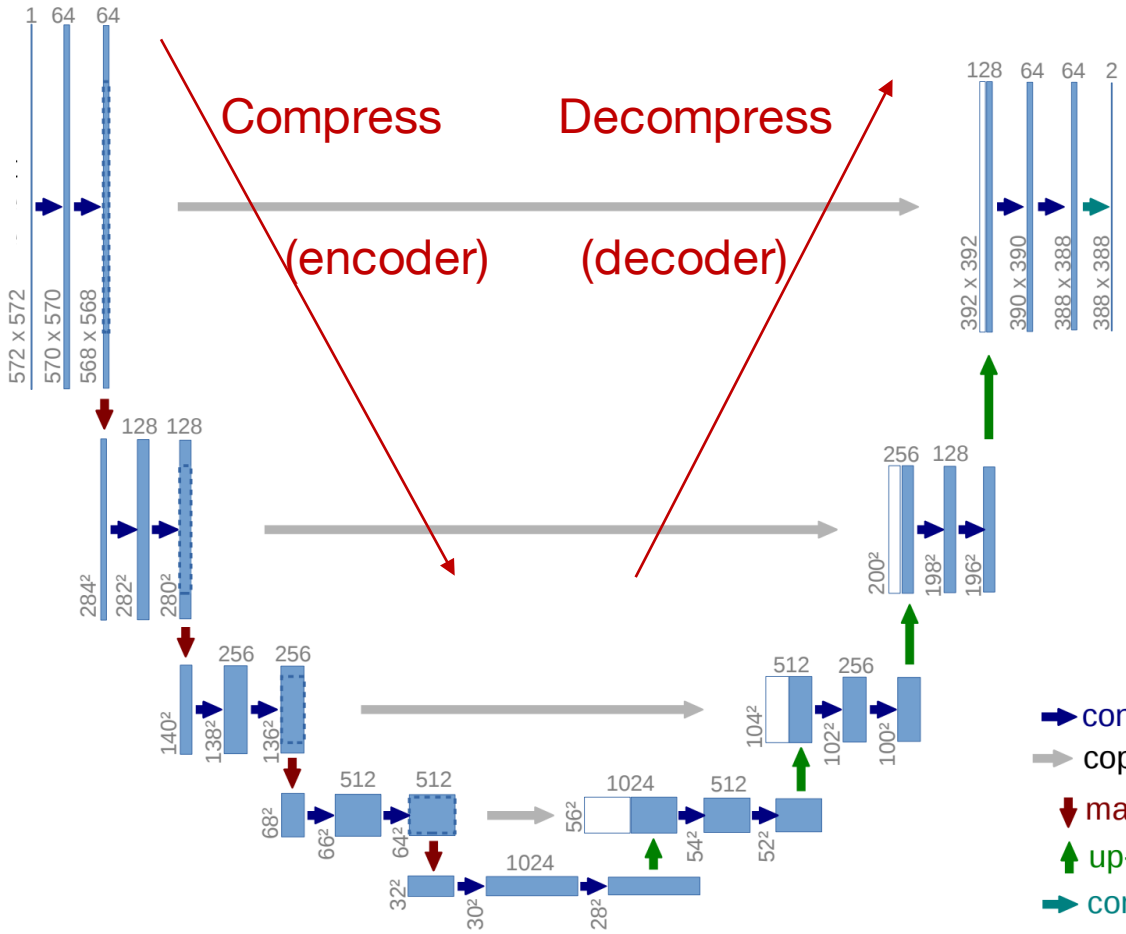
Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/



Example: Denoising Autoencoder



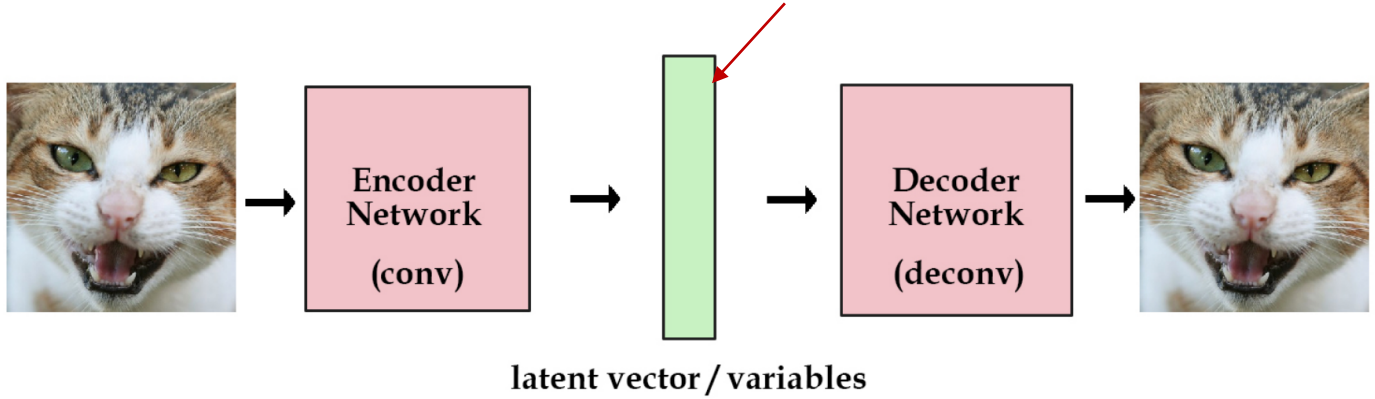
Input noisy image



"Label" is noiseless image!

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF



- Good *generative model*
- Have a clean probability distribution to select from to generate new examples

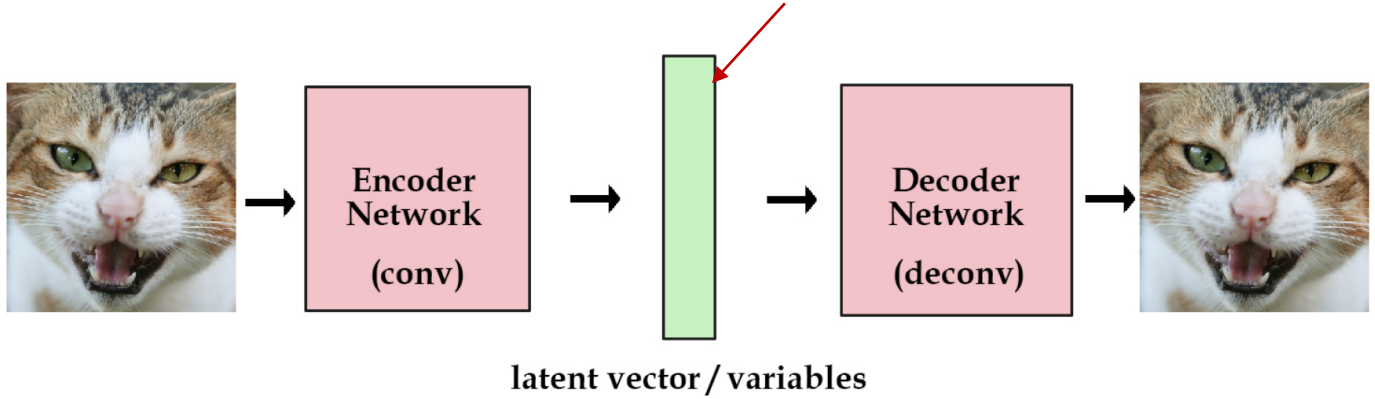
Minimize (KL) distance between latent vector and Gaussian normal

Input

VAE reconstruction

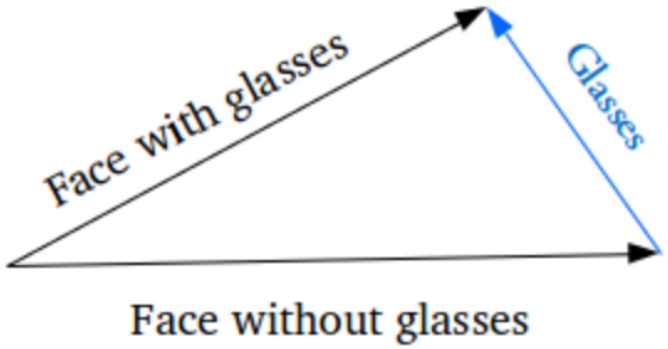
Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF



Minimize (KL) distance between latent vector and Gaussian normal

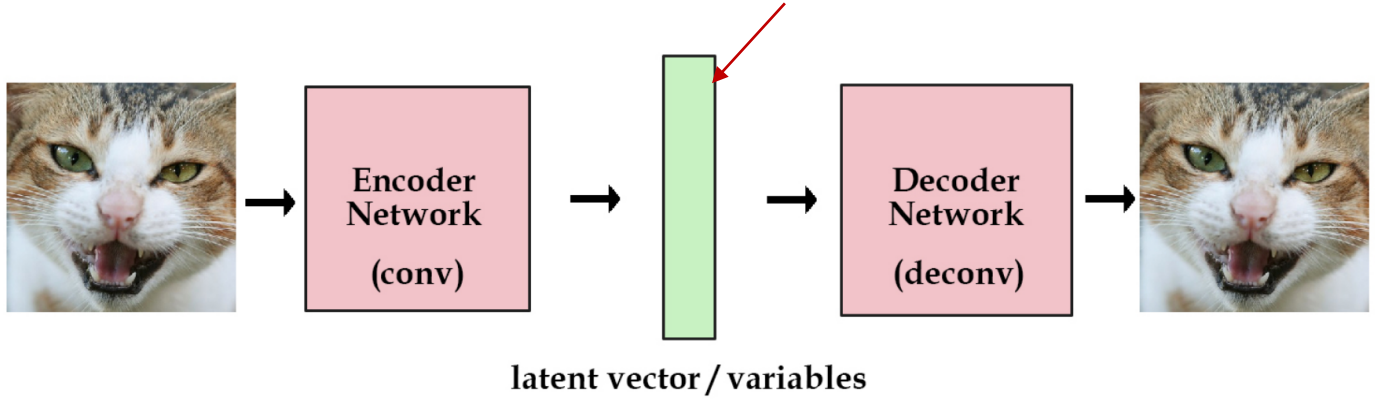
- With Gaussian PDF, can start to add/subtract latent vector in a normalized vector space



Adding new features to samples

Example: Variational Autoencoder (VAE)

Force this vector to follow a Gaussian PDF

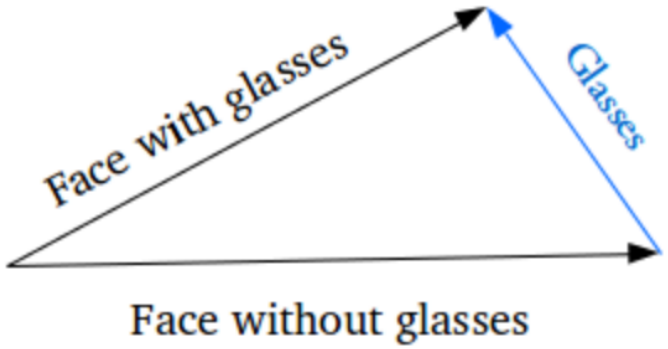


Minimize (KL) distance between latent vector and Gaussian normal

Generative Example (once trained):

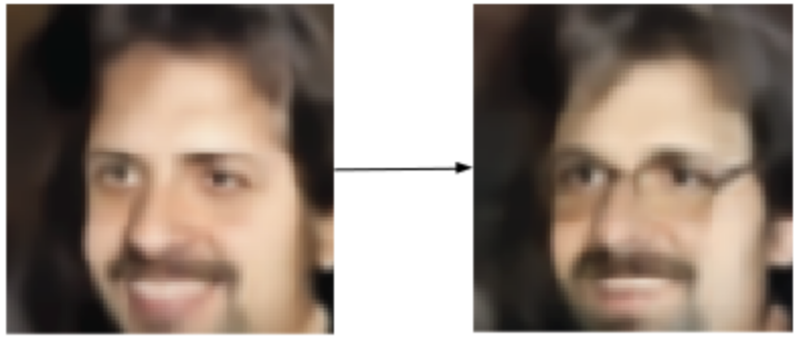
- Encode image with glasses, obtain latent vector PDF $\mathbf{P_g}$
- Encode image without glasses, obtain PDF $\mathbf{P_{ng}}$
- Compute **diff** = $\mathbf{P_g - P_{ng}}$
- Encode new image to obtain $\mathbf{P_{new}}$, add in **diff**
- Decode $\mathbf{P_{new}}$ + **diff** to get guy with glasses!

- With Gaussian PDF, can start to add/subtract latent vector in a normalized vector space



Adding new features to samples

Glasses



Exploring a specific variation of input data[1]

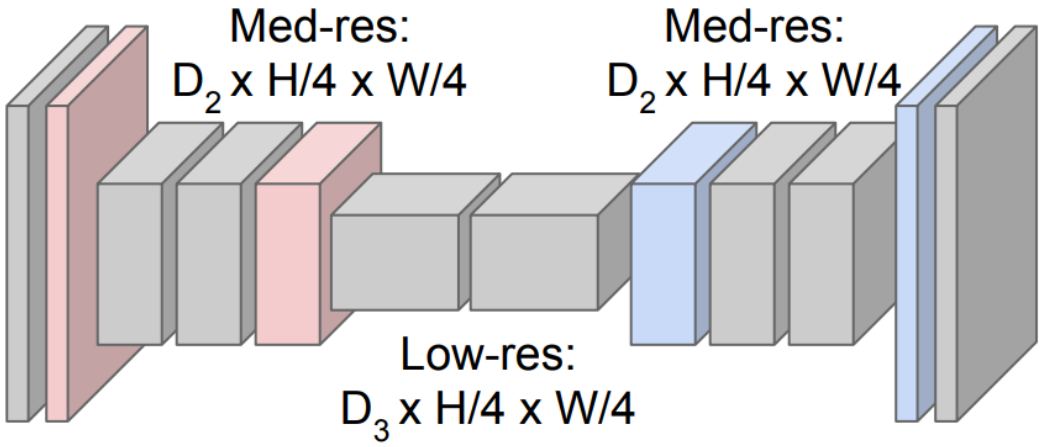
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
???

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

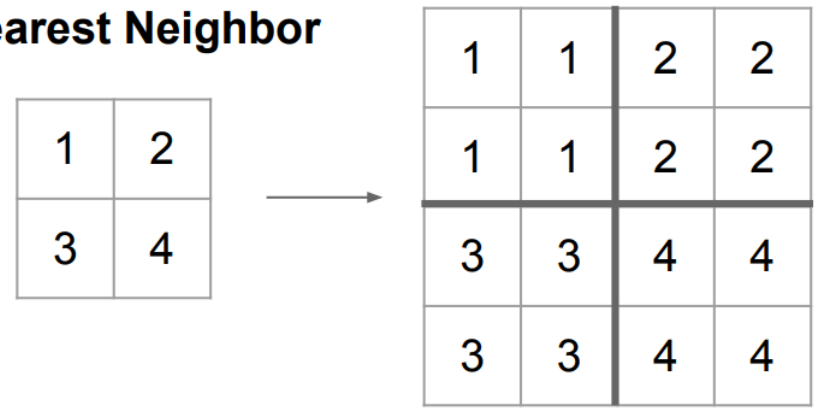
High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015
Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

In-Network upsampling: “Unpooling”

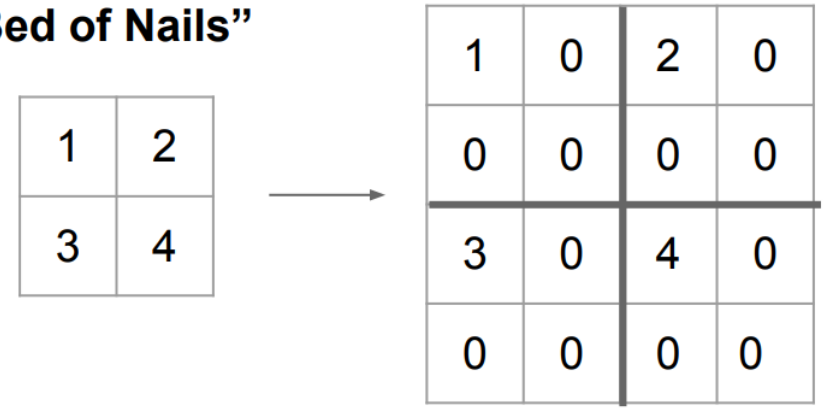
Nearest Neighbor



Input: 2 x 2

Output: 4 x 4

“Bed of Nails”



Input: 2 x 2

Output: 4 x 4

In-Network upsampling: “Max Unpooling”

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

...

Rest of the network

Max Unpooling
Use positions from pooling layer

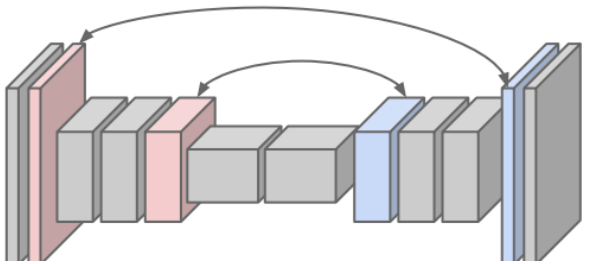
1	2
3	4

Input: 2 x 2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

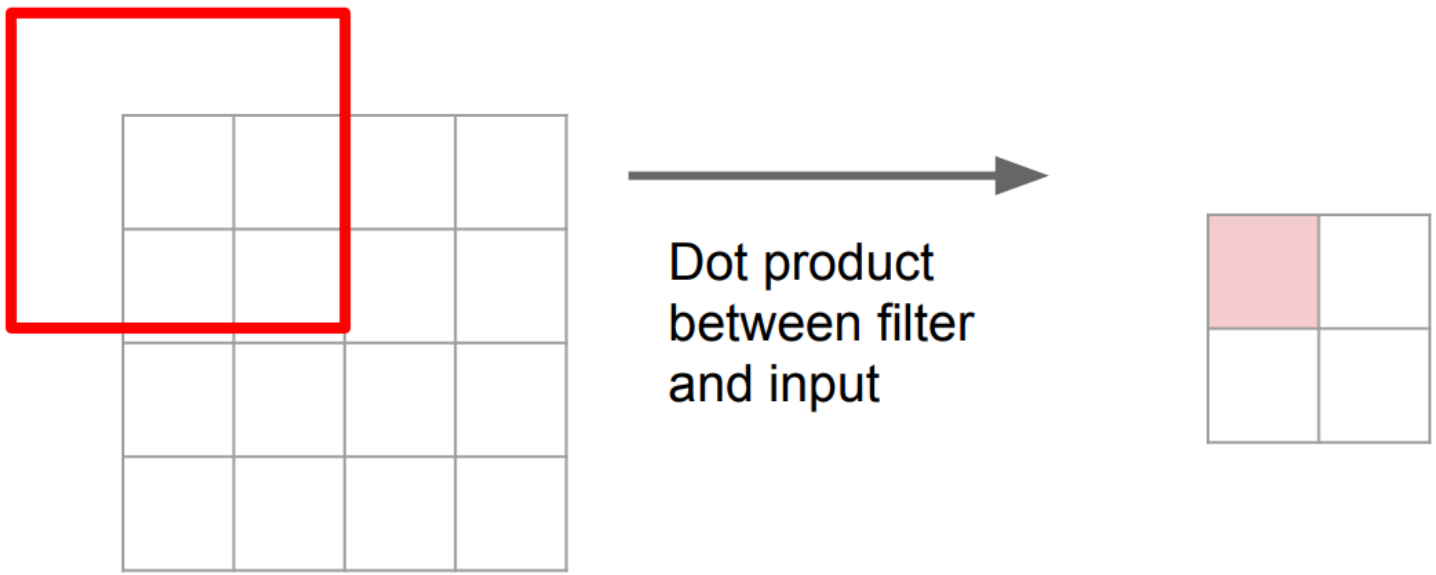
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1

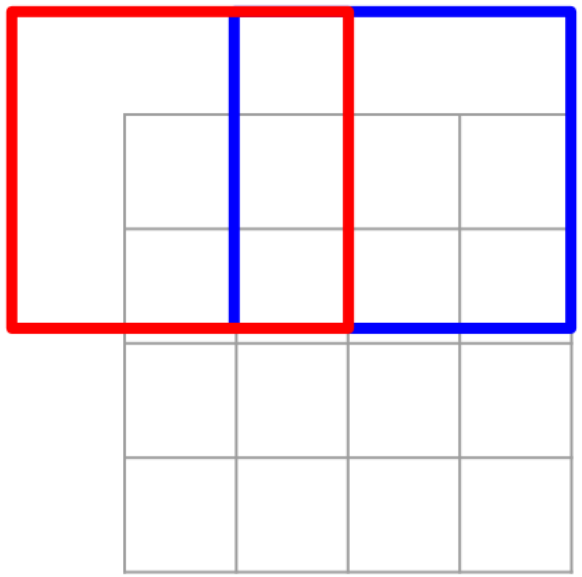


Input: 4 x 4

Output: 2 x 2

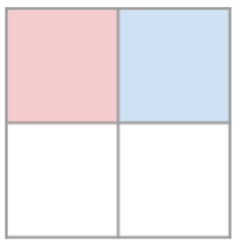
Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

Dot product between filter and input



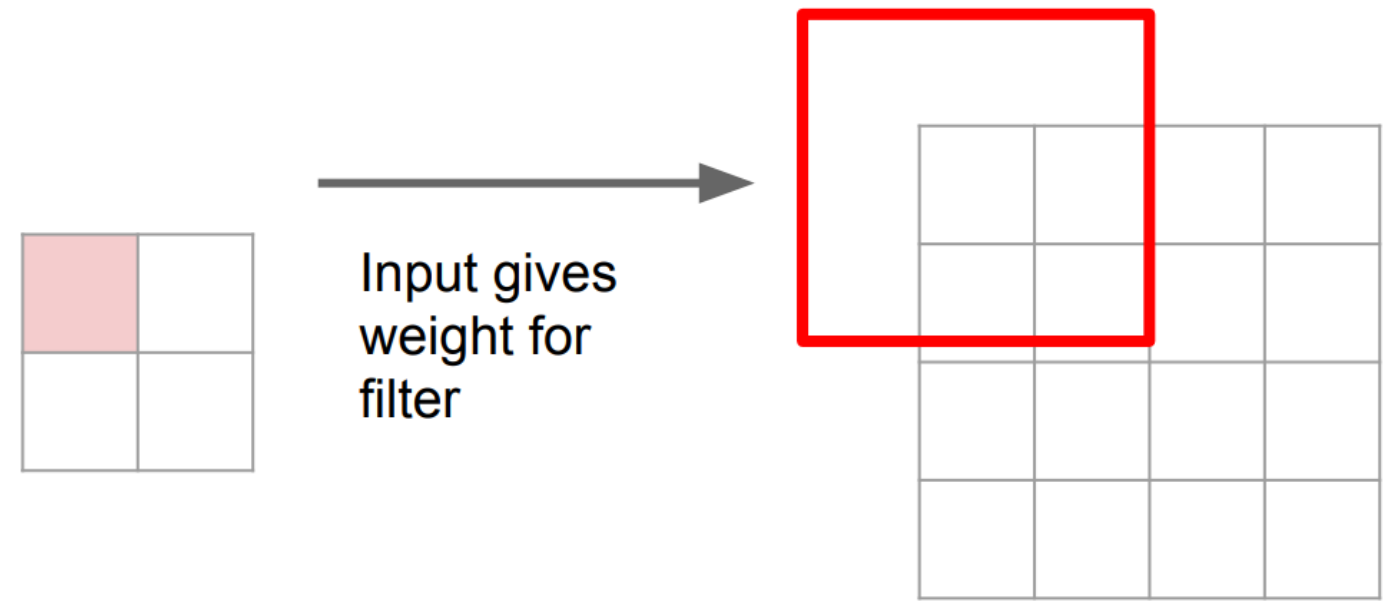
Output: 2 x 2

Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

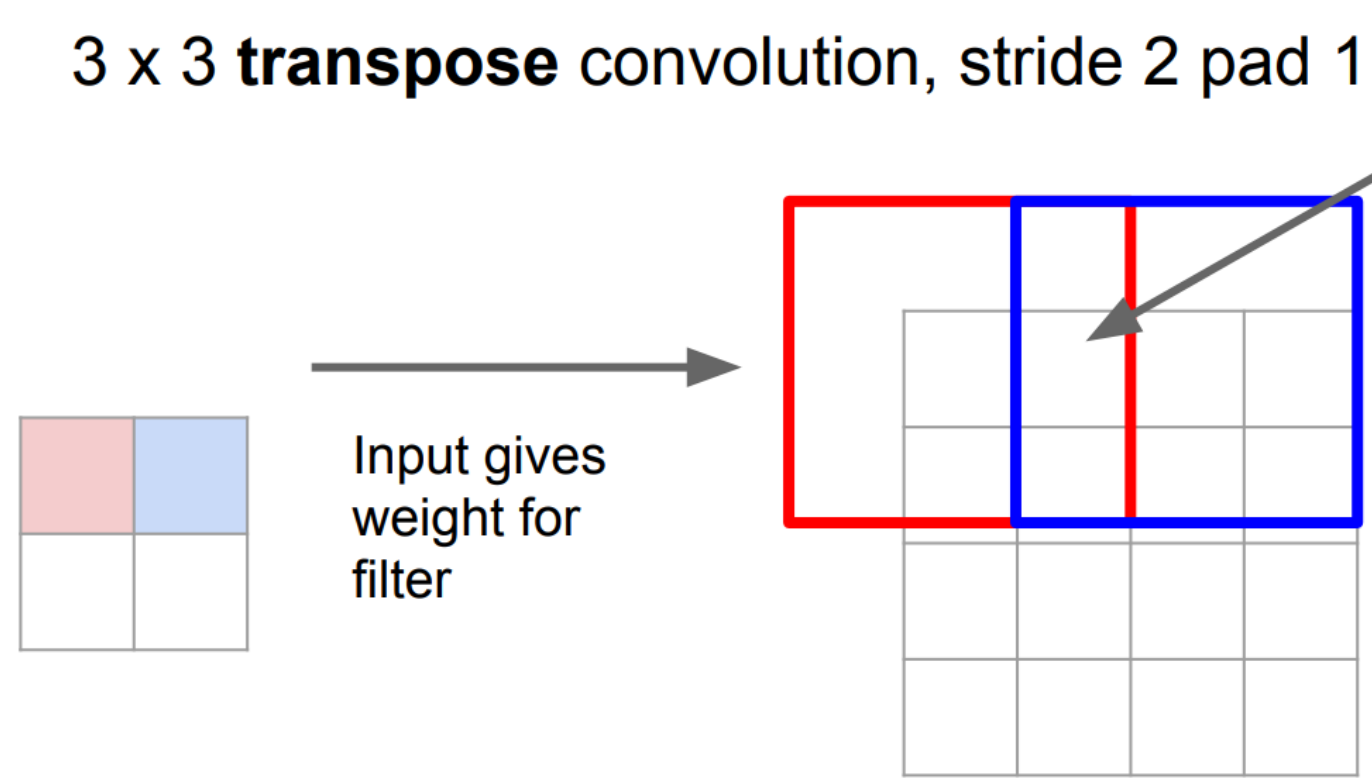
Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1

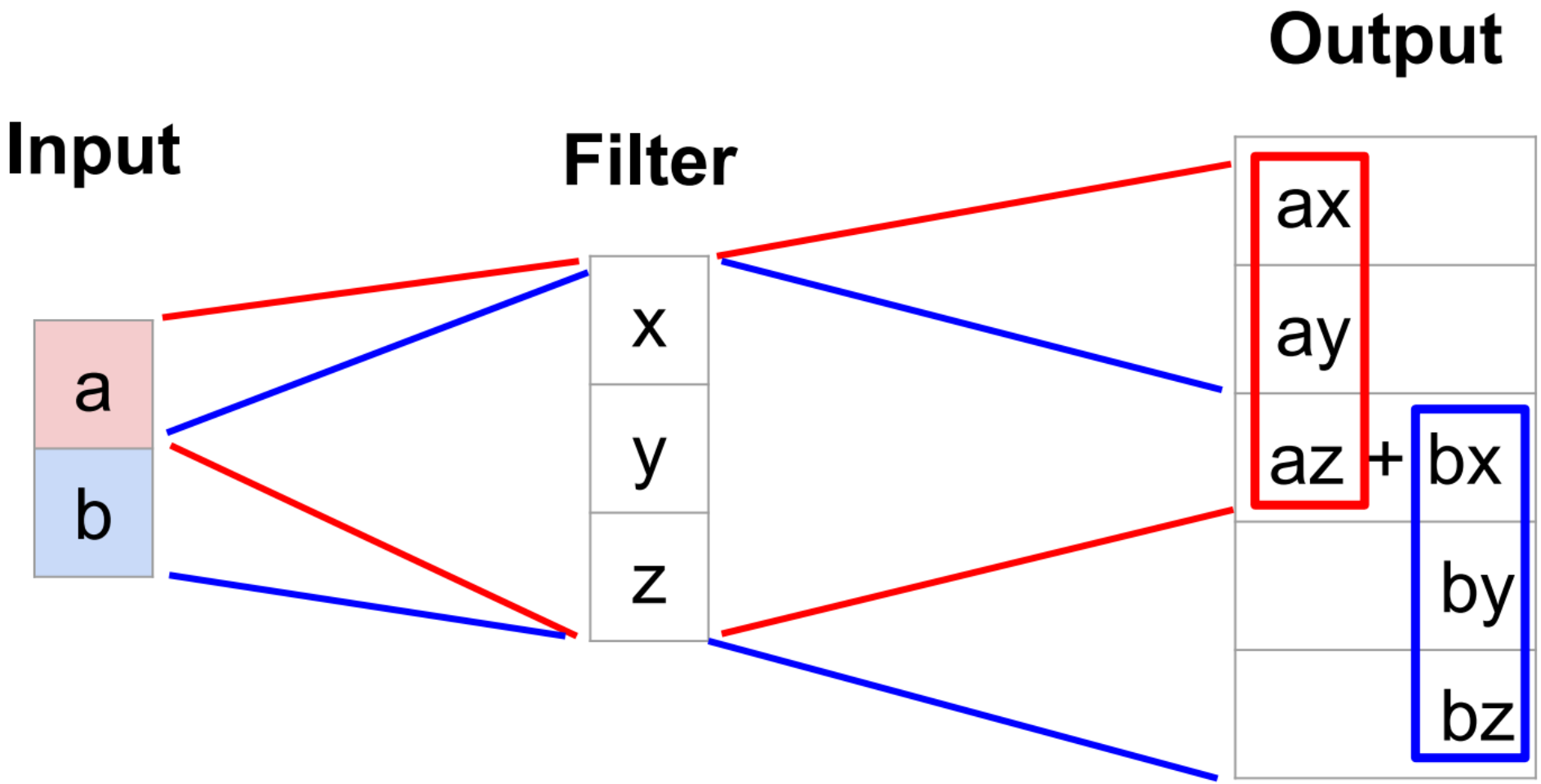


Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1



Learnable Upsampling: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Need to crop one pixel from output to make output exactly 2x input

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & 0 & x & y & x & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

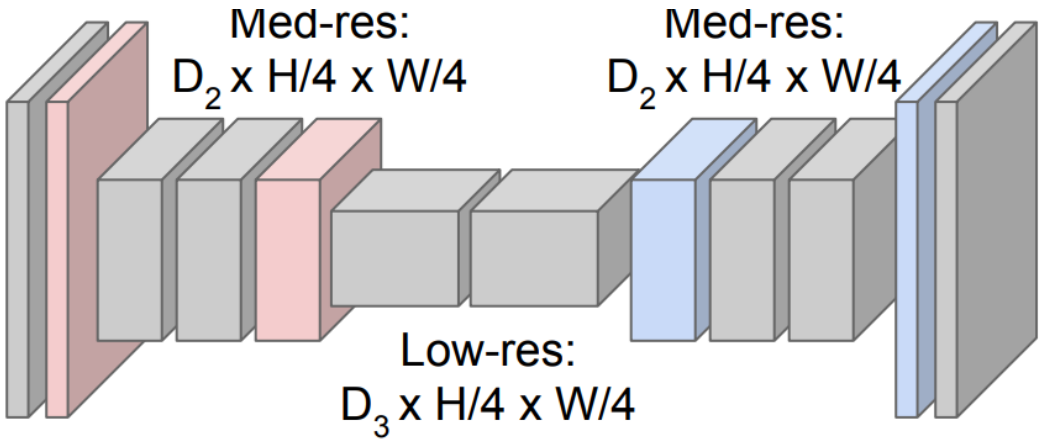
$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

When stride>1, convolution transpose is no longer a normal convolution!

Semantic Segmentation Idea: Fully Convolutional

- Train and test the same way as for classification, now just with labels that are H x W maps!

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

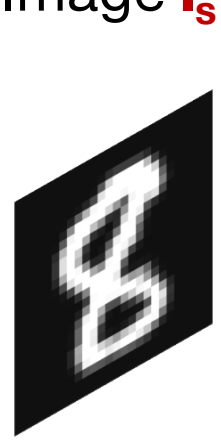
Predictions:
H x W

Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015
Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

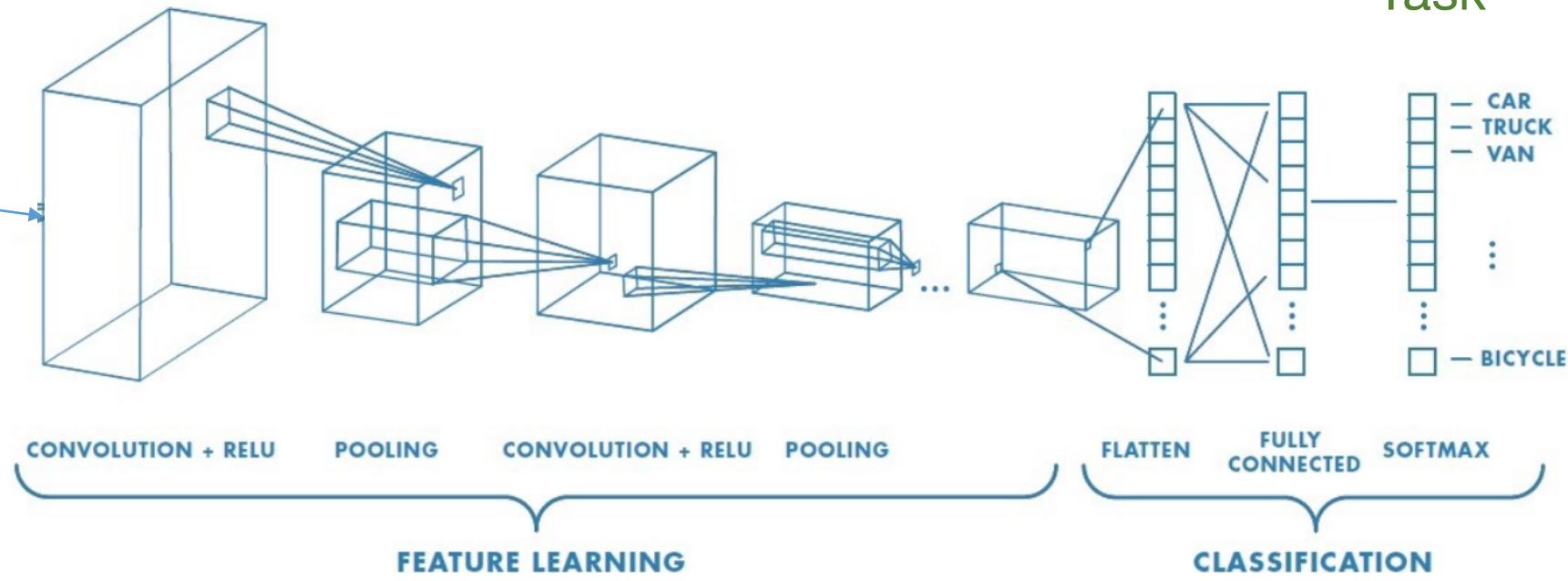
Adding Physical Layers to a deep network

Bringing together physical and digital image representations

Image $\mathbf{I_s}$

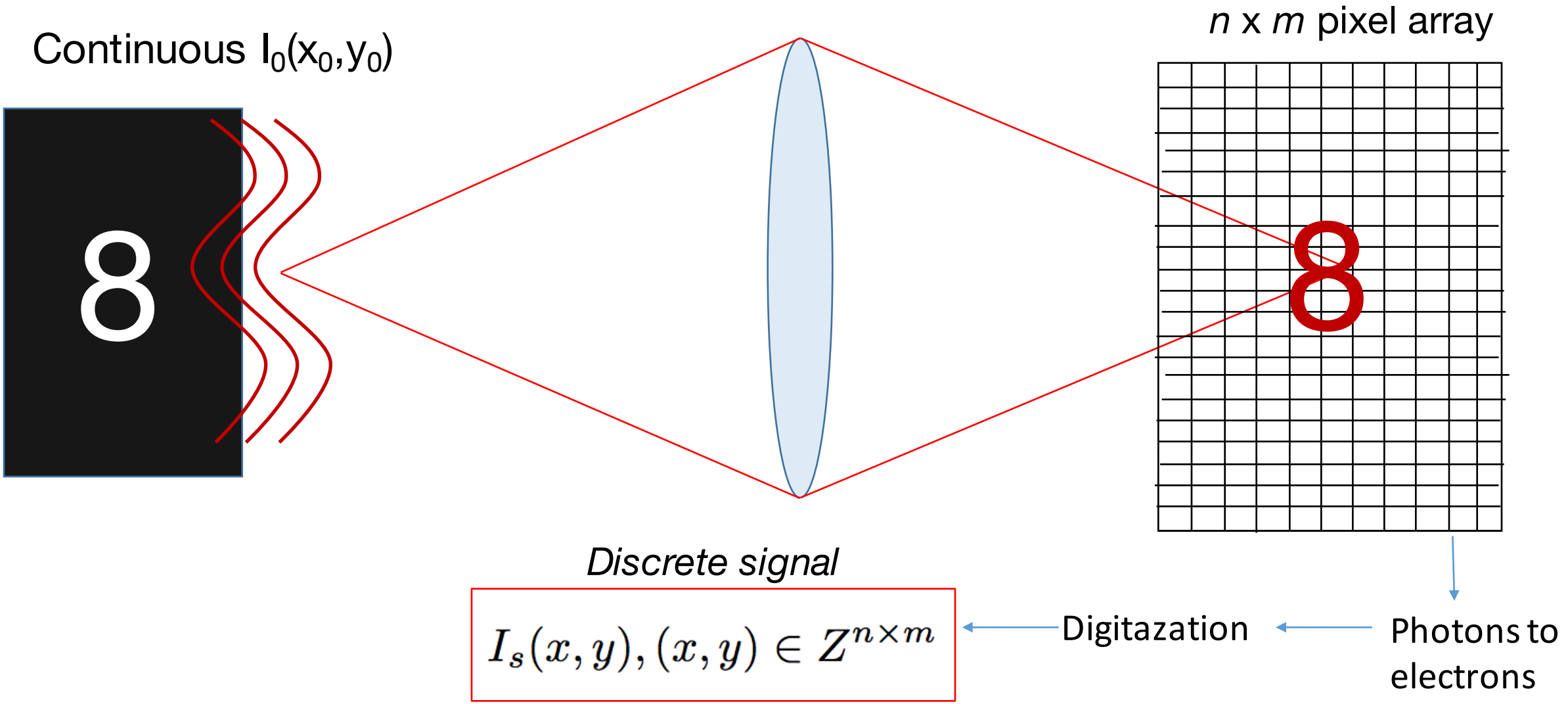


Task



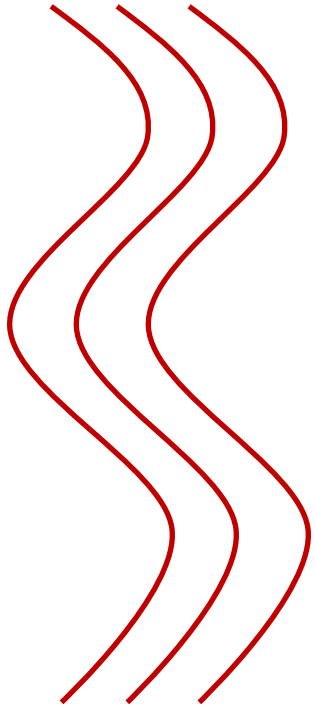
Task = $\mathbf{W_n}$ …ReLU[$\mathbf{W_1}$ ReLU[$\mathbf{W_0}$ $\mathbf{I_s}$]…]

Simple model of image formation



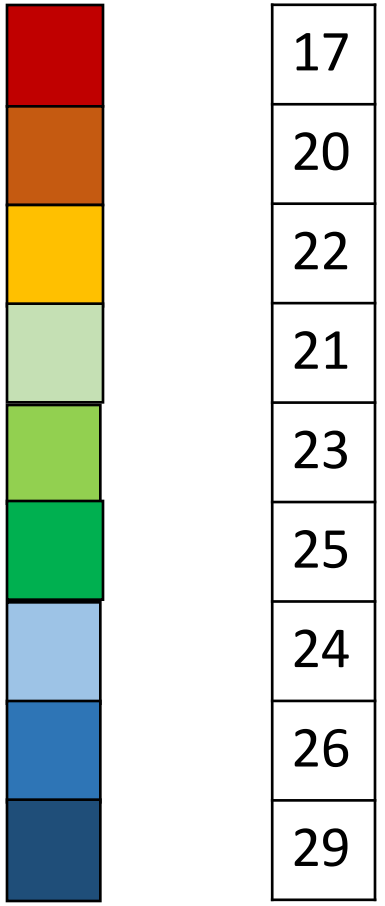
What does the Sampling Theorem mean for us?

Continuous functions



*conditions

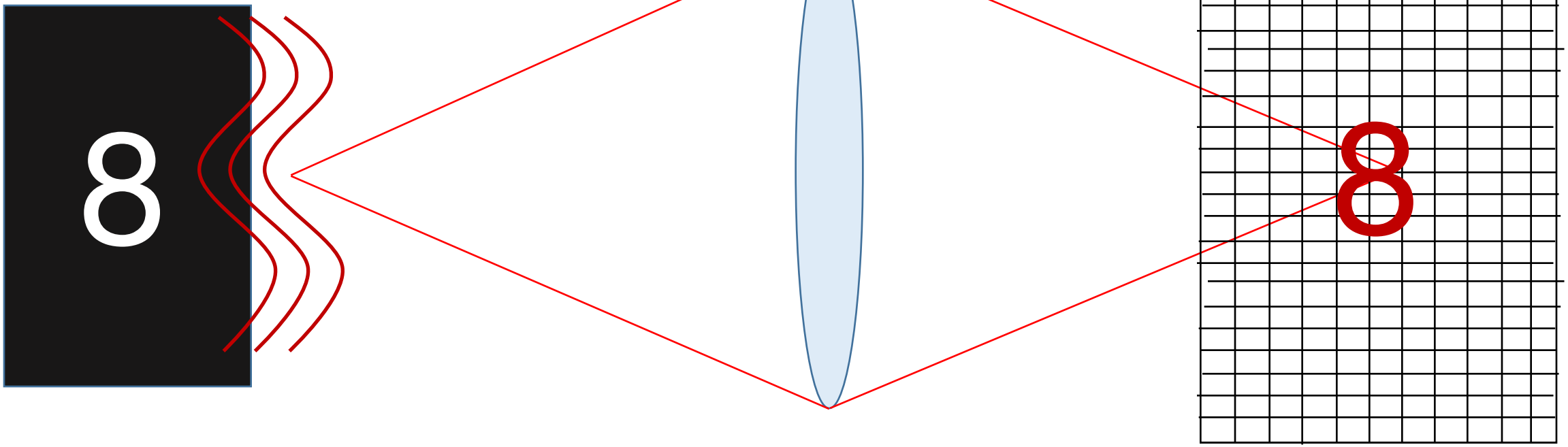
Discretize vectors
(and matrices)



17
20
22
21
23
25
24
26
29

Simple model of image formation

Discrete $I_0(x_0,y_0)$



Discrete signal

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

← Digitazation ← Photons to electrons

Bringing together physical and digital image representations

Physical Layers

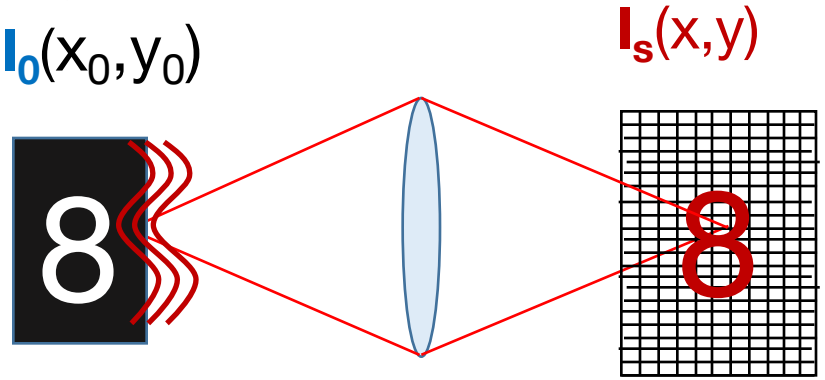
Digital Layers

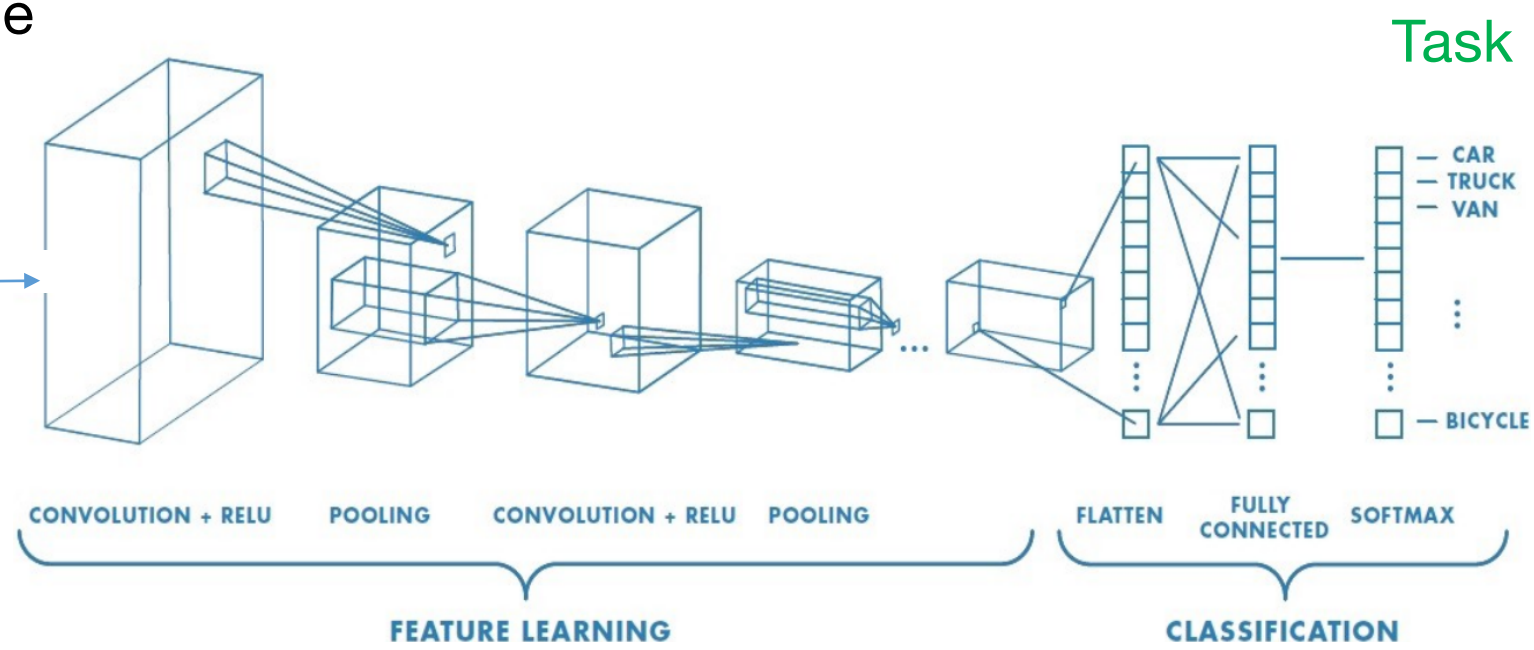
Physical world

$I_0(x_0,y_0)$

Digital Image

$I_s(x,y)$





$$I_s = f[I_0]$$

Digital layers

Physical layers

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

Bringing together physical and digital image representations

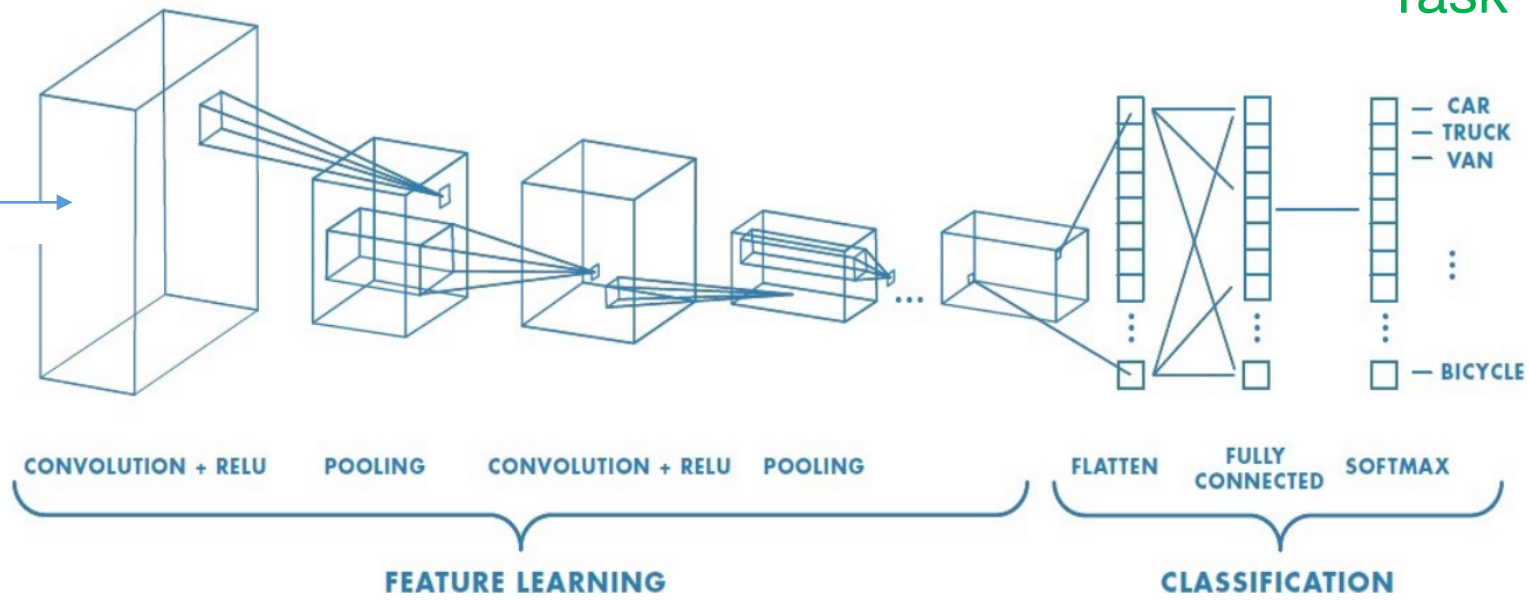
Physical Layers

Digital Layers

Task

Physical Function I_0 → f[ ] → Digitized I_s

$$I_s = f[I_0]$$



Digital layers

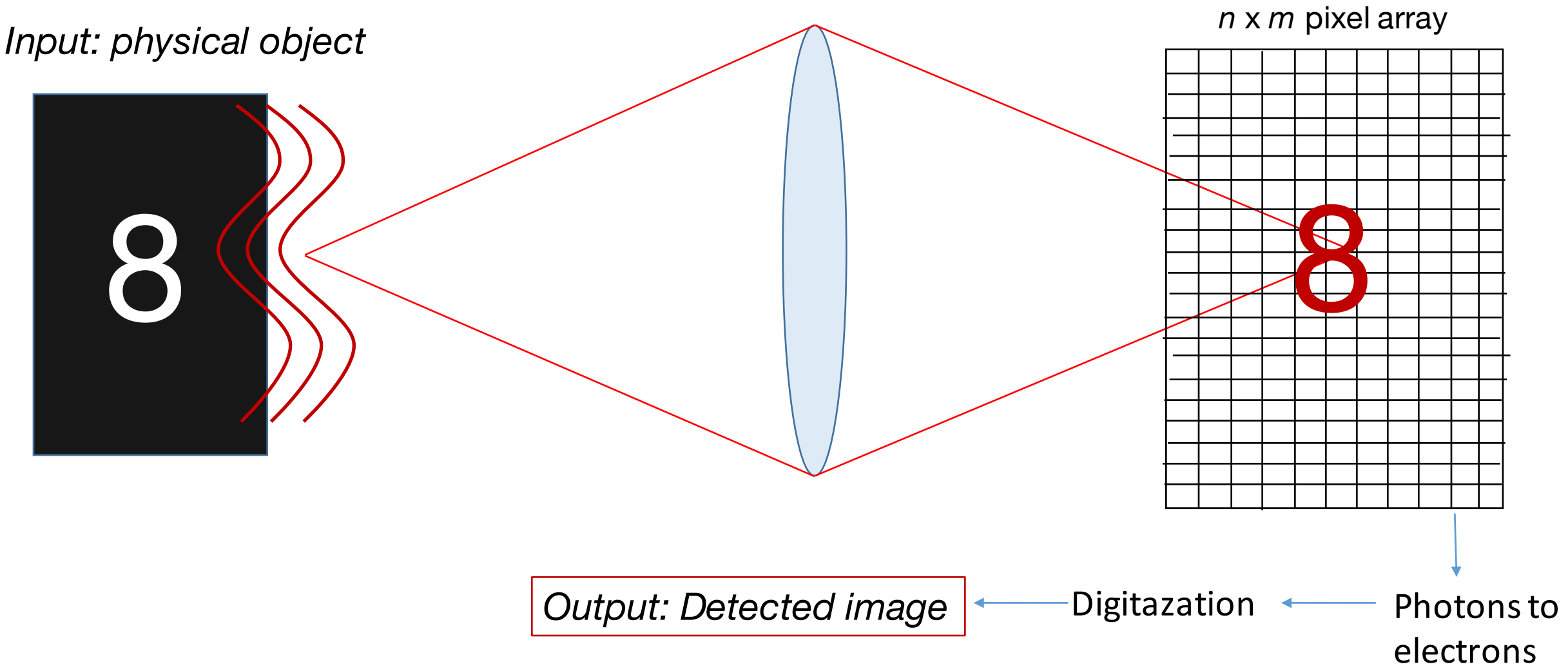
Physical layers

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

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

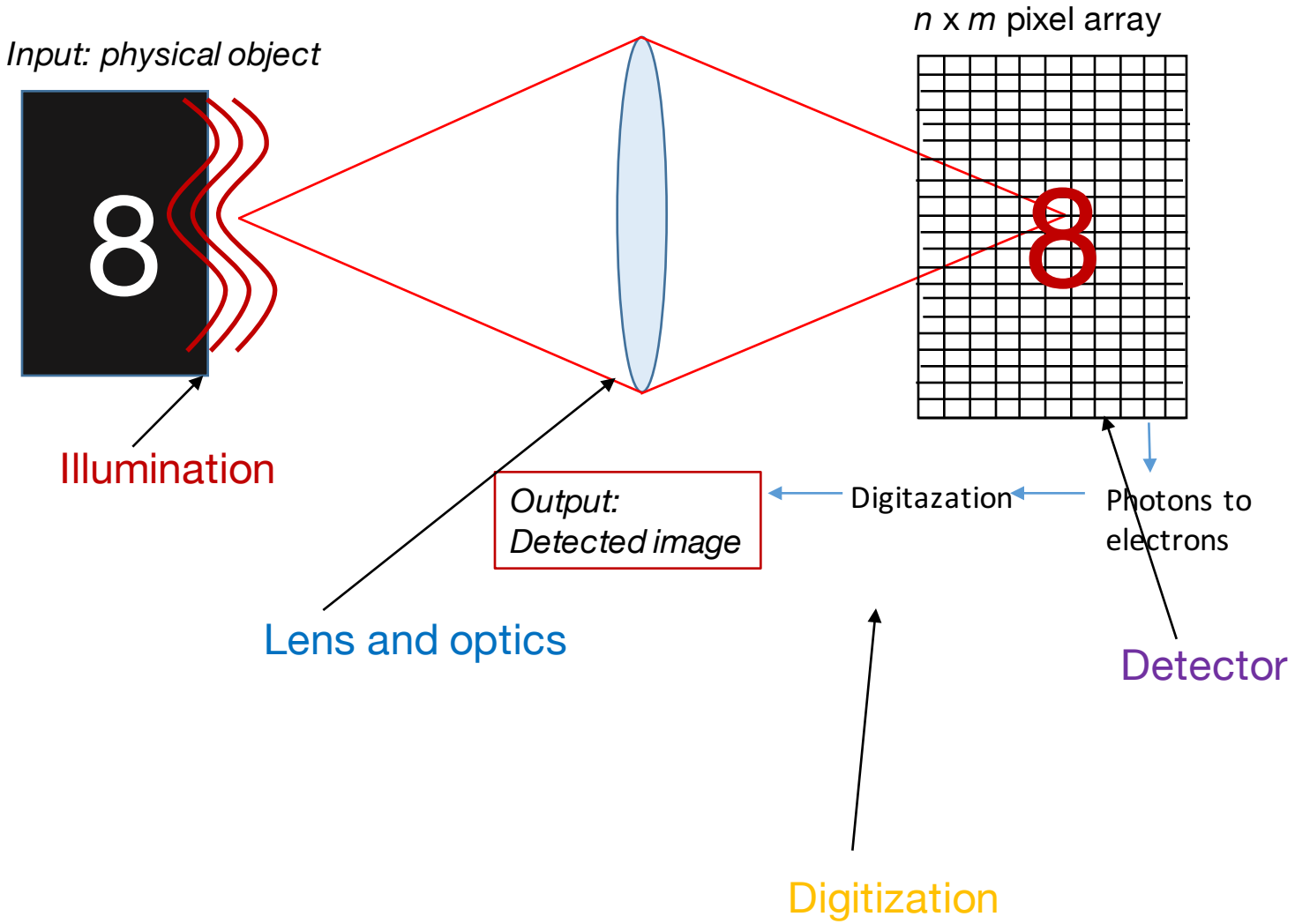
- Finite
- Non-zero gradients
- Differentiable*
- Known structure (for now…)
- Anything else?

What physical parameters effect image formation?

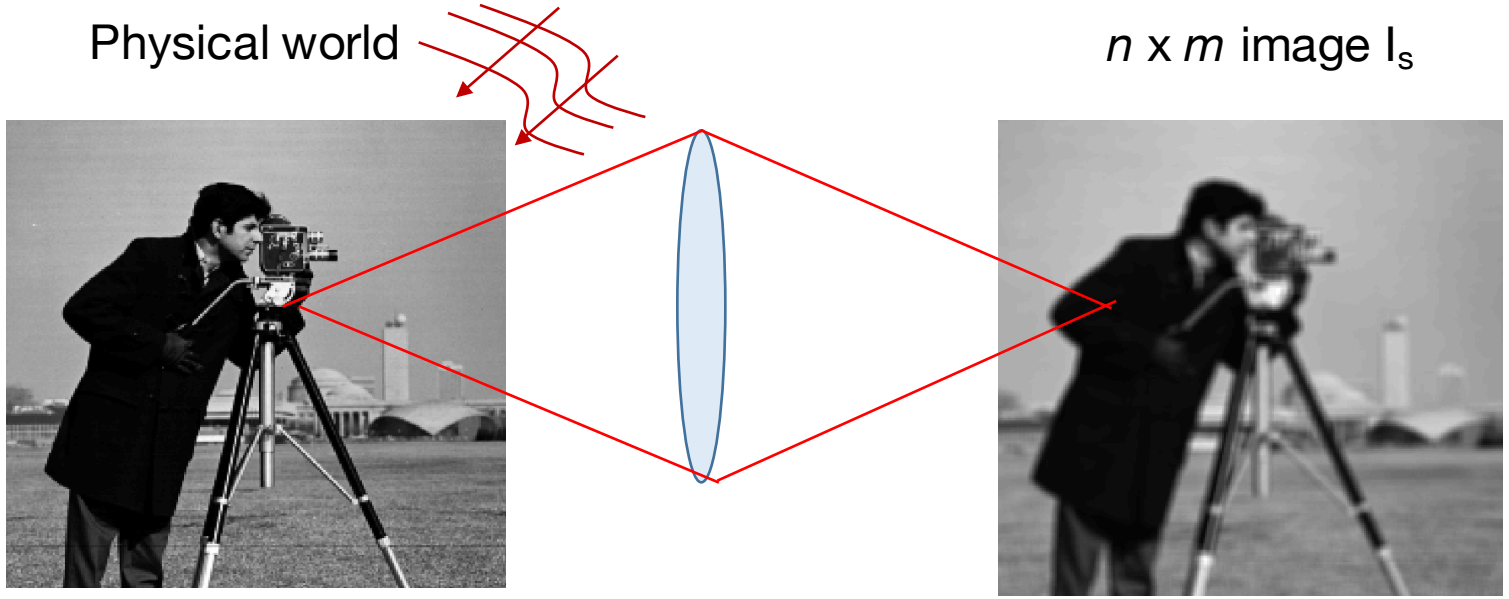


What physical parameters effect image formation?

- 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

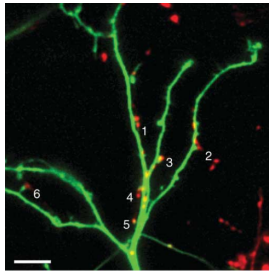


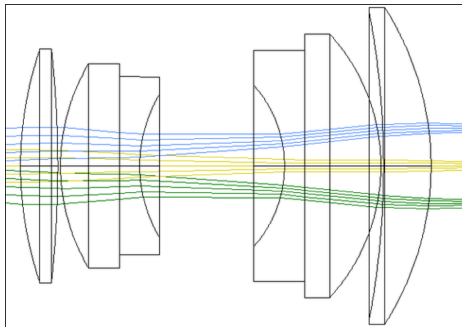
Simple mathematical model of image formation



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

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



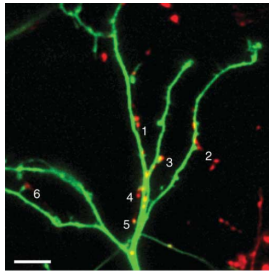


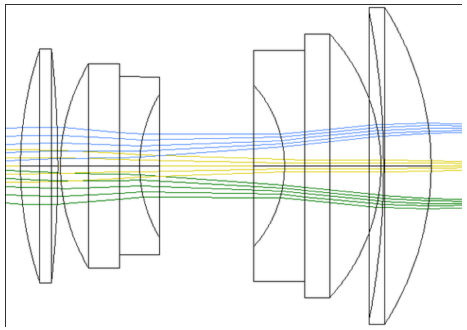
- Real, non-negative
- Models absorption and brightness

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

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

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

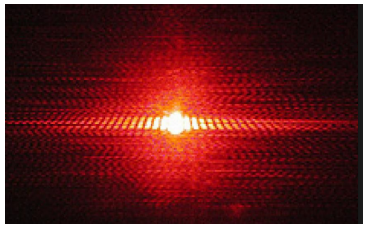


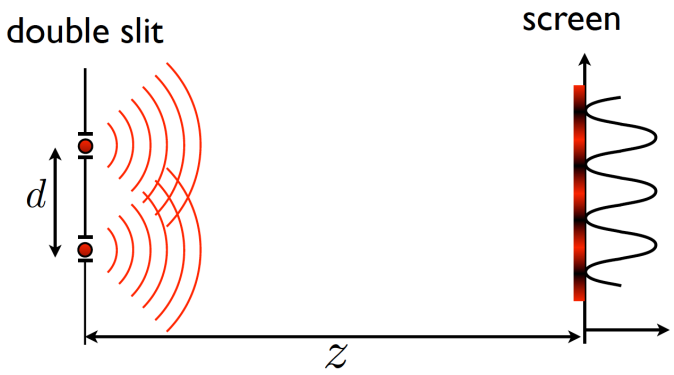


- Real, non-negative
- Models absorption and brightness

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

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



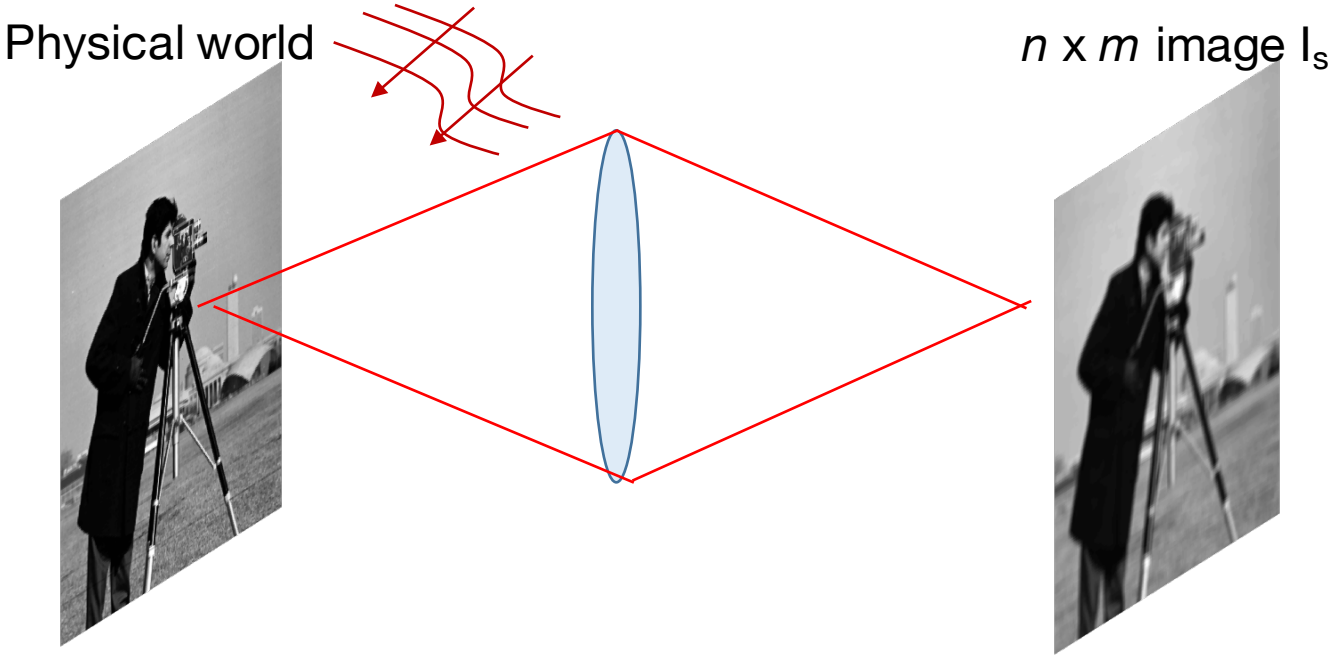


- Complex field
- Models Interference

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

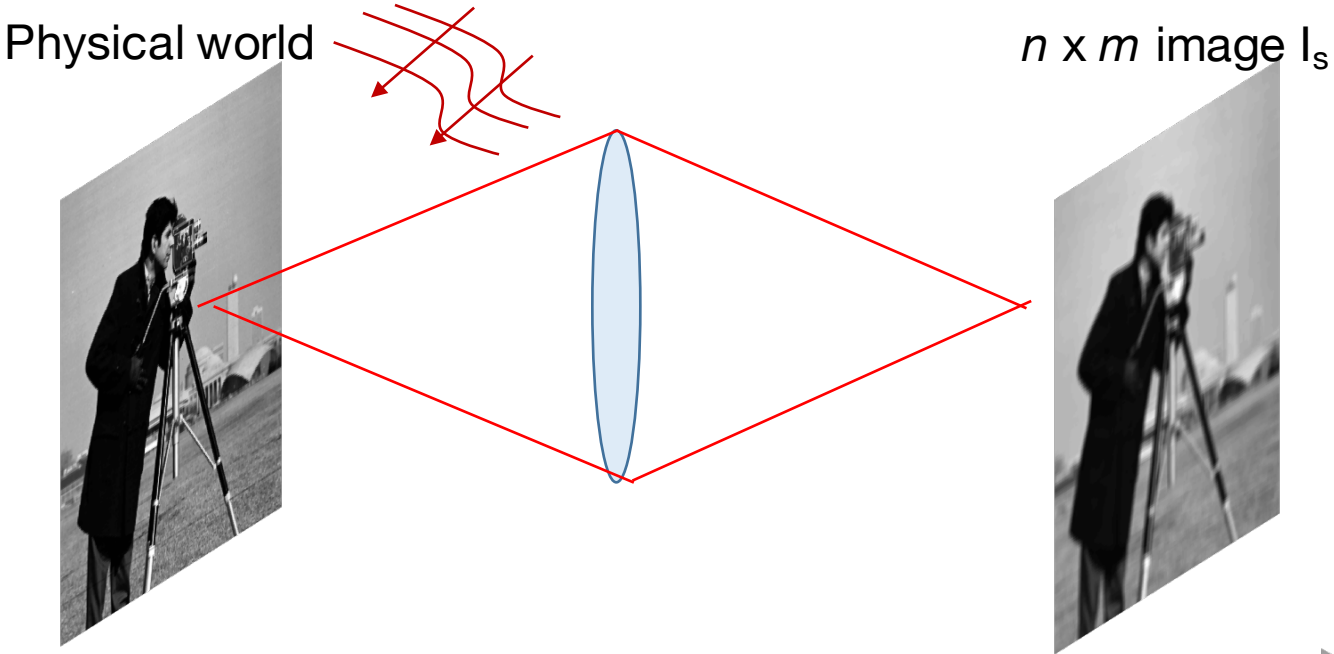
- Interpretation #3: Particle
- Model: Photons

Simple mathematical model of image formation



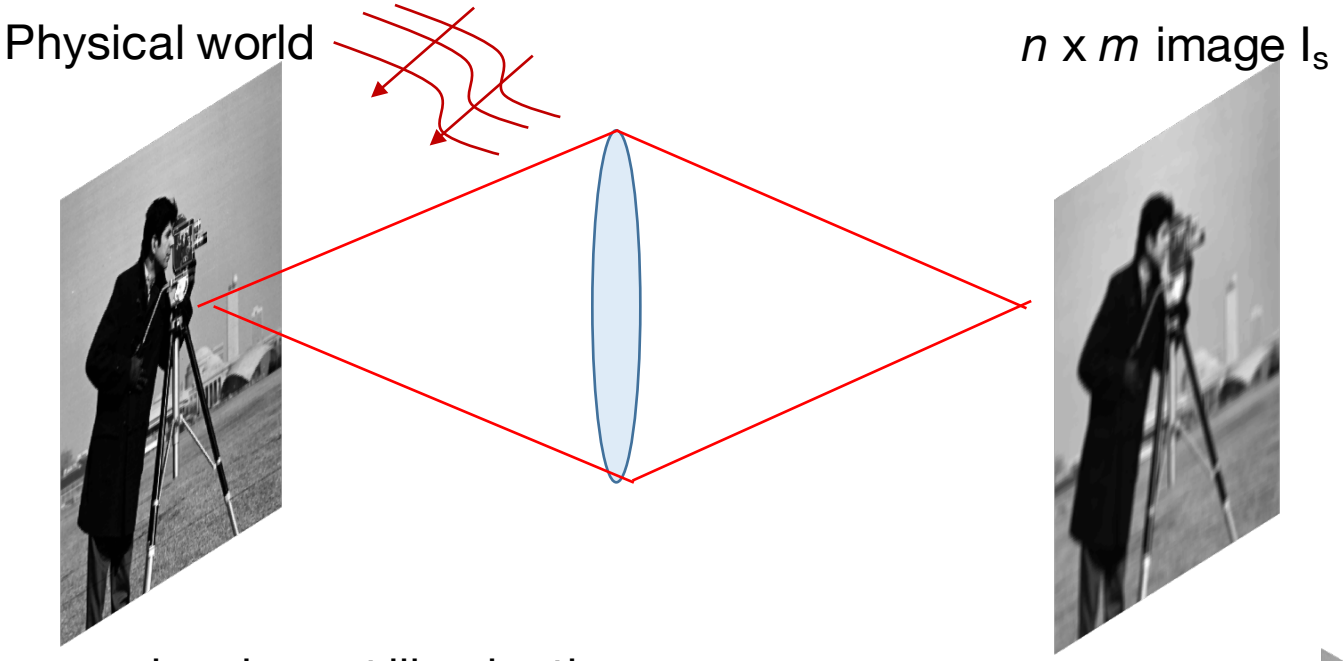
- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity

Simple mathematical model of image formation

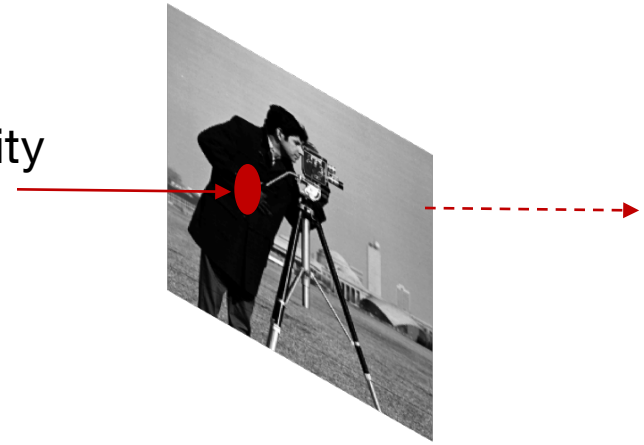


- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity

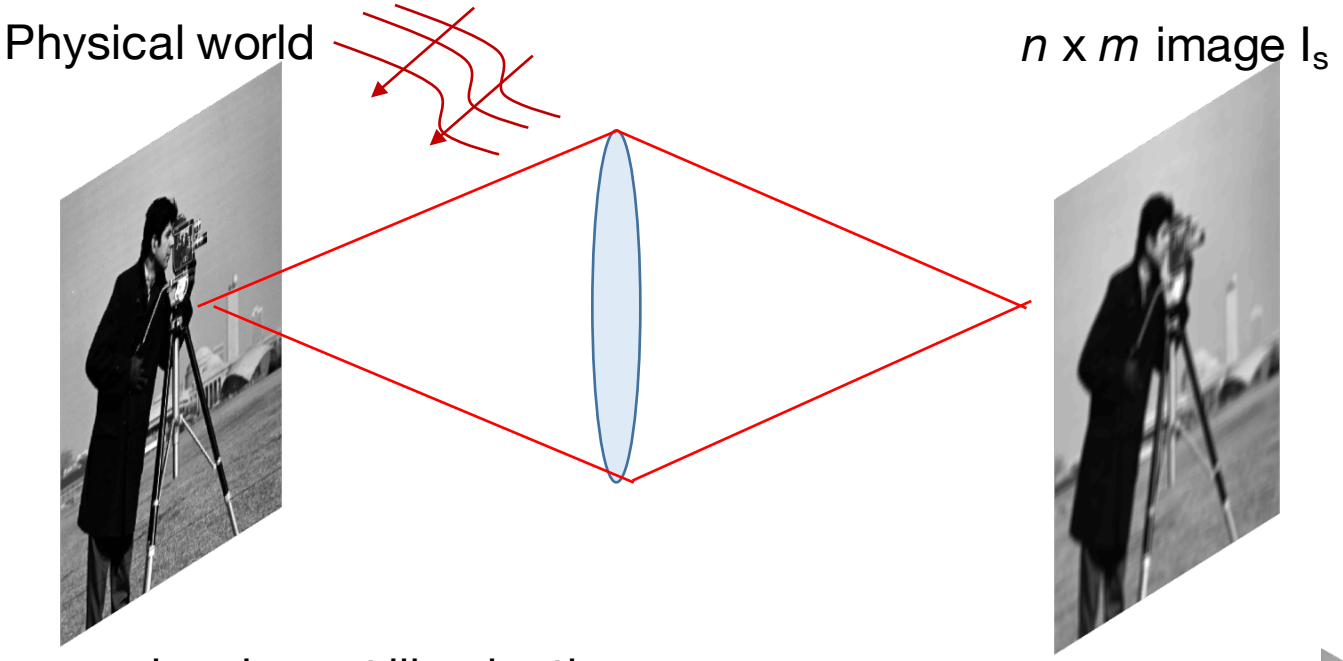
Simple mathematical model of image formation



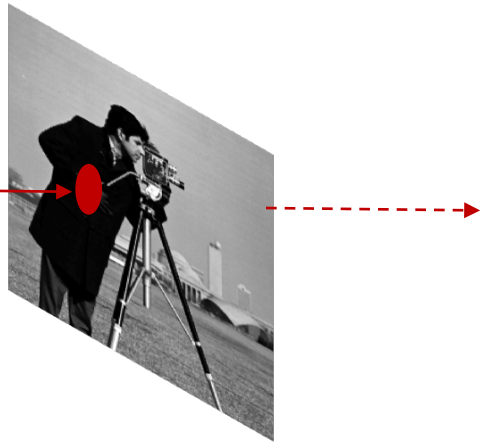
- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity



Simple mathematical model of image formation

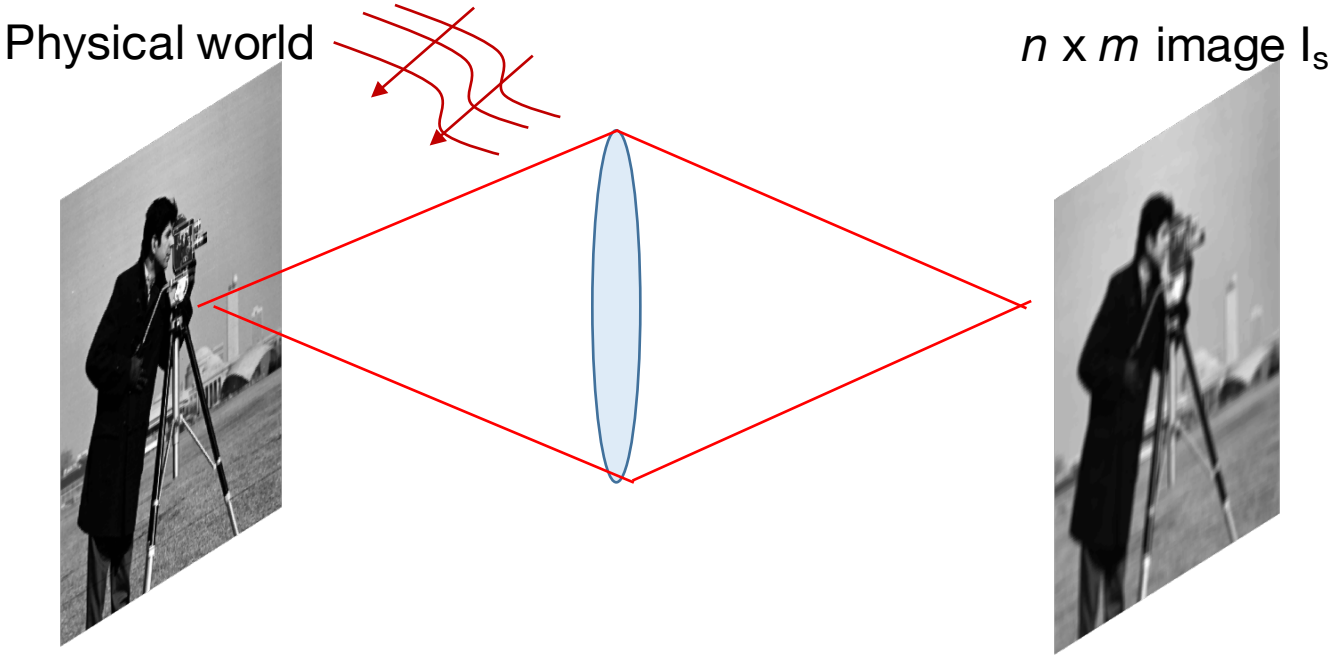


- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity



Object absorption: $\mathbf{I_0}(x,y)$
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



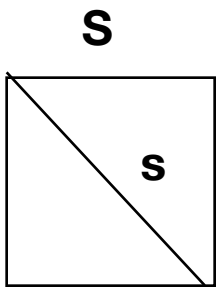
- Assume incoherent illumination
- Assume thin 2D object
- Object is real, non-negative map of absorption/reflectivity

Modeling incoherent illumination

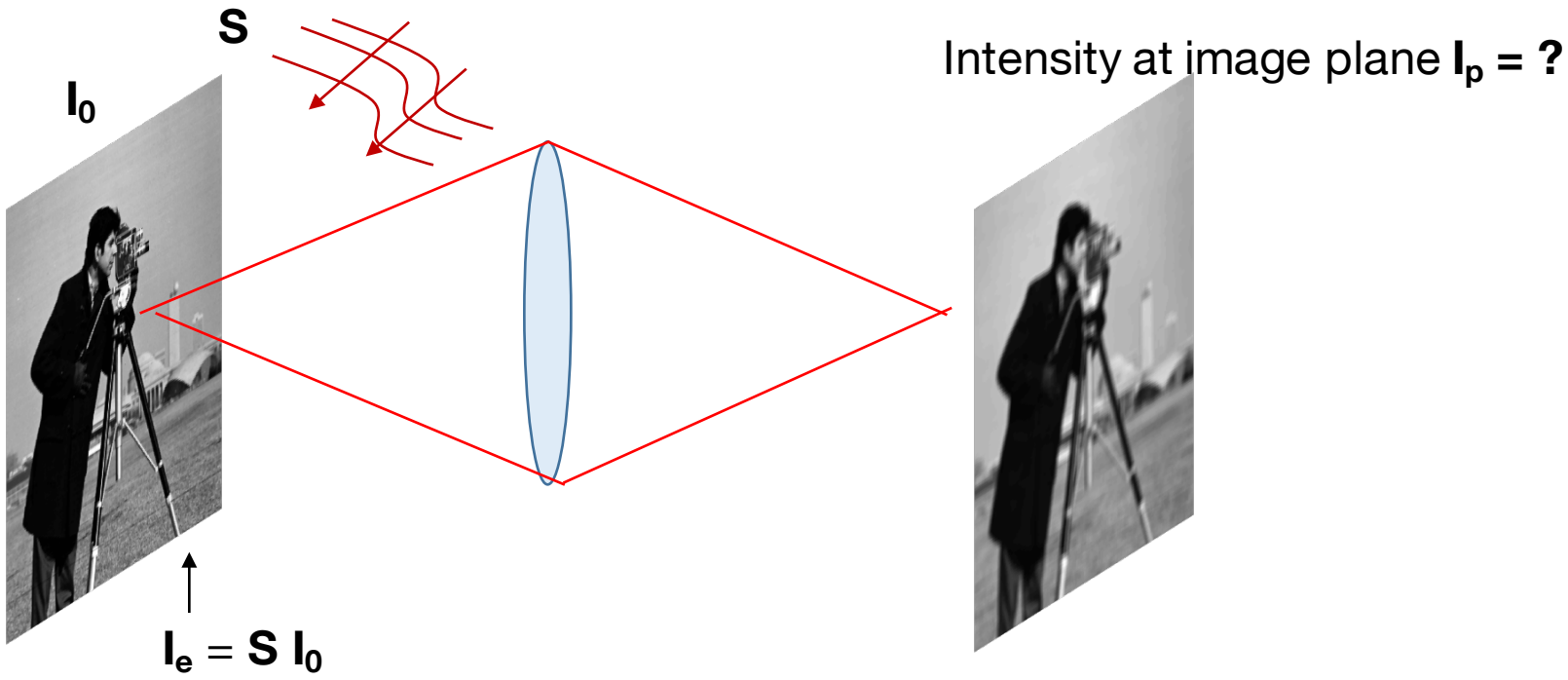
$$\mathbf{I_e}(x,y) = \mathbf{I_0}(x,y) \circ \mathbf{s}(x,y)$$

$$\mathbf{I_e} = \mathbf{S}\,\mathbf{I_0}$$

$\mathrm{diag}(\mathbf{S}) = \mathbf{s}$



Simple mathematical model of image formation

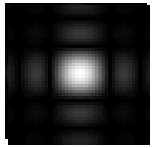


Simple mathematical model of image formation

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

Input intensity

Convolution filter h

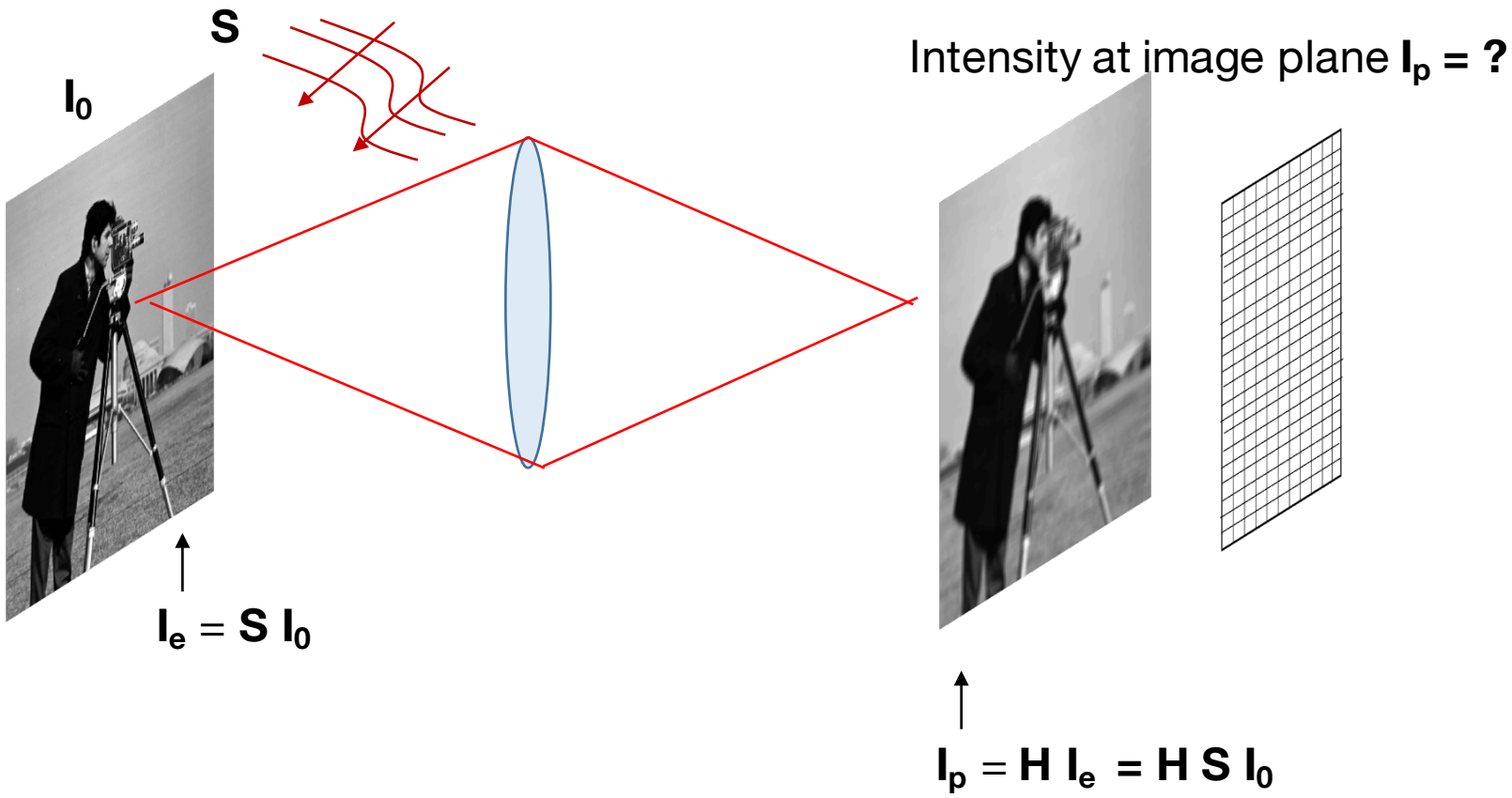


Output intensity

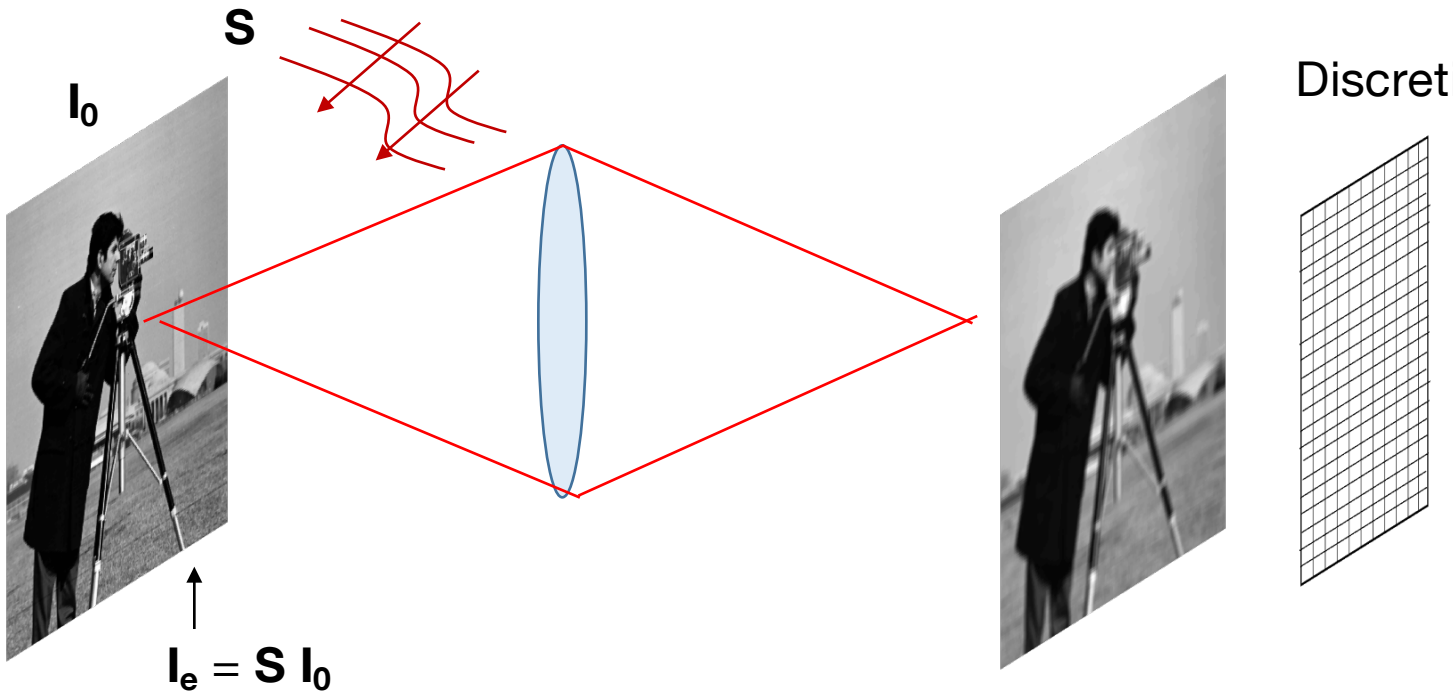
$\mathbf{I_e}(x/M, y/M) * \mathbf{h}(x,y) = \mathbf{I_p}(x,y)$

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



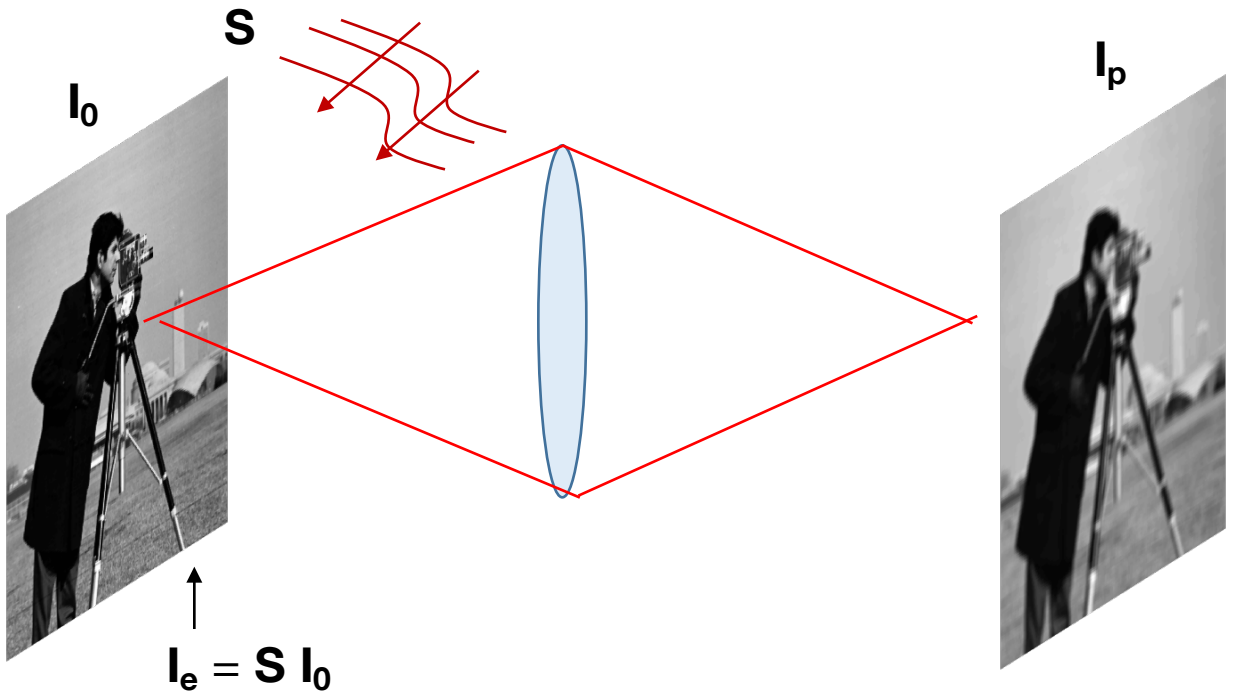
Discretization by detector ?

Use downsampling matrix (sum-pooling)

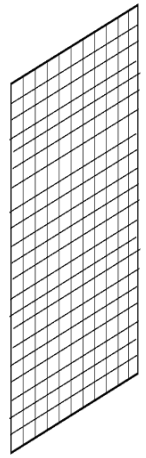
$D =$

$$\begin{bmatrix} 0.5\ 0.5\ 0\ 0\ 0\ \ldots. & & \\ & 0.5\ 0.5\ \ 0\ 0\ 0\ \ldots. & \\ & & 0.5\ 0.5\ \ 0\ 0\ 0\ \ldots. \\ \vdots & & \vdots \end{bmatrix}$$

Simple mathematical model of image formation



Discretization by detector ?



$$I_s = D\ I_p = D\ H\ S\ I_0$$

Use downsampling matrix (sum-pooling)

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

Physical Layers

Digital Layers

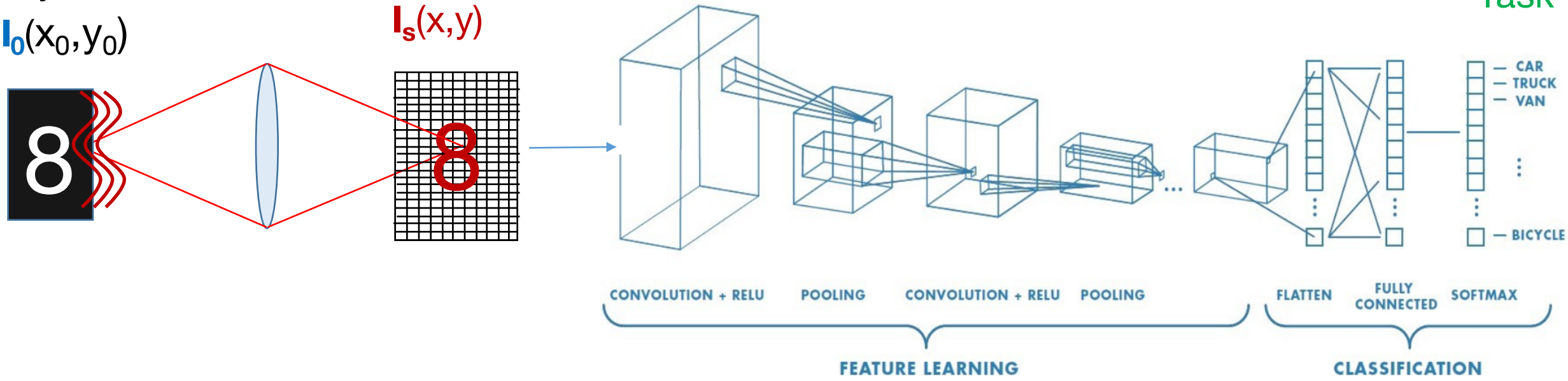
Physical world

$I_0(x_0,y_0)$

Digital Image

$I_s(x,y)$

Task



Digital layers

Physical layers

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

$$\text{Task} = \mathbf{W}_n \ldots \text{ReLU}[\mathbf{W}_1\ \text{ReLU}[\mathbf{W}_0\ \mathbf{D}\ \mathbf{H}\ \mathbf{S}\ \mathbf{I}_0]\ldots]$$